ISSN 1349-1113
JAXA-RR-10-005

宇宙航空研究開発機構研究開発報告
JAXA Research and Development Report

数値シミュレータIII - 導入と運用, 性能評価, 次世代への課題

松尾　裕一, 坂下　雅秀, 末松　和代, 染谷　和広, 高木　亮治,
土屋　雅子, 藤岡　晃, 藤田　直行

2010年10月

宇宙航空研究開発機構
Japan Aerospace Exploration Agency

目　次

略語

HPC	High Performance Computing
CFD	Computational Fluid Dynamics
NS	Numerical Simulator
NWT	Numerical Wind Tunnel
FLOPS	Floating Point Operations Per Second
MIPS	Mega Instructions Per Second
MB	Mega Byte (=10^6 Byte)
GB	Giga Byte (=10^9 Byte)
TB	Tera Byte (=10^{12} Byte)
PB	Peta Byte (=10^{15} Byte)
CPU	Central Processing Unit
MPU	Micro Processing Unit
SMP	Symmetric Multi Processing
NUMA	Non-Uniform Memory Access
SAN	Strage Area Network
NAS	Network Attached Strage
HSM	Hierarchical Storage Management
MPI	Message Passing Interface
XPF	Extended Parallel Fortran
CeNSS	Central Numerical Simulator System
CeMSS	Central Mass Storage System
CeViS	Central Visualization System
DTU	Data Transfer Unit
SRFS	Shared Rapid File System
LTO	Linear Tape-Open
GSN	Gigabyte System Network
SSH	Secure Shell
WANS	Web Access to Numerical Simulator
CMS	CeNSS Monitoring System
RANS	Reynolds-Averaged Navier-Stokes
DNS	Direct Numerical Simulation
LES	Large Eddy Simulation
DES	Detached Eddy Simulation
CAA	Computational Aeroacoustics
FFT	Fast Fourier Transform

数値シミュレータ III - 導入と運用，性能評価，次世代への課題*

松尾　裕一*1，坂下　雅秀*1，末松　和代*1，染谷　和広*1，高木　亮治*1，
土屋　雅子*1，藤岡　晃*1，藤田　直行*1

The Numerical Simulator III – Acquisition and Installation , its Operation, Performance Evaluation, and the Critical Issues to the Next Generation Supercomputing*

Yuichi MATSUO*1, Masahide SAKASHITA*1, Kazuyo SUEMATSU*1, Kazuhiro SOMEYA*1, Ryoji TAKAKI*1, Masako TSUCHIYA*1, Akira FUJIOKA*1 and Naoyuki FUJITA *1

Abstract

In this report, we first describe the acquisition and installation of the Numerical Simulator III, which has started operation at October 2002 in the former National Aerospace Laboratory, and has been operated until October 2008 as part of the JAXA Supercomputer System even after the consolidation of three space organizations into JAXA. Next, by clarifying what we were able to achieve or fail by the acquisition, what we learned from the operation experience with using the performance evaluation data and the operational statistics, we draw and visualize the important technical aspects in aerospace supercomputing, and the know-how and implicit knowledge for the large equipment operation. In particular, we address the performance characteristics of the JAXA applications in terms of hybrid parallelism which come from the architectural features of the central computing engine, i.e. a SMP cluster. Finally, with those materials, we discuss what the JAXA's supercomputing system should be, and the critical issues to the future supercomputing in order to earn useful views for the next-generation practitioners.

Key Word: Supercomputing, Computer System, Numerical Simulator, Computational Fluid Dynamics, CeNSS, CeViS, Performance Evaluation

概　要

本報告は，旧航空宇宙技術研究所において 2002 年 10 月に導入され，宇宙航空研究開発機構（JAXA）に統合された以降も JAXA スーパーコンピュータシステムの一部として 2008 年 10 月まで稼動したスーパーコンピュータシステム「数値シミュレータ III」に関して述べる．まず，調達から設置・運用までの経緯を俯瞰し，システム概要・特徴を明確化することにより，今回の導入において成功した点，あるいは注意点・課題を洗い出す．次に，性能評価データや運用統計データを用いて，技術的に実際にできたこと・できなかったことや，運用によって得られたものを明らかにするとともに技術課題や運用上の課題を分析する．特に，SMP クラスタという中核計算機の構成上の特徴から来る JAXA アプリケーションのハイブリッド並列における特性や性能推定法について言及する．これらの材料をもとに，航空宇宙分野におけるスーパーコンピューティングの重点技術やスーパーコンピュータシステムのあり方を考察するともに，設備運用のノウハウや勘所（＝暗黙知）を抽出・可視化し，次世代実務者の礎とする．

* 平成 22 年 7 月 26 日受付(Received 26 July 2010)
*1 情報・計算工学センター　計算機運用・利用技術チーム
(Computing Resource Management Team, JAXA's Engineering Digital Innovation Center)

序章 まえがき

0.1 本報告の目的

本報告は，旧航空宇宙技術研究所（以下，「航技研」と略）において2002年10月に導入され，宇宙航空研究開発機構（以下，「JAXA」と略）に統合された以降も JAXA スーパーコンピュータシステムの一部として2008年10月まで稼動したスーパーコンピュータシステム「数値シミュレータ III」に関して，調達から導入までの経緯を俯瞰するとともに，性能評価データや運用統計データを分析し，技術的に実際にできたこと・できなかったことや運用によって得られたもの，さらには次世代への技術課題を論ずることにより，航空宇宙分野におけるスーパーコンピューティング技術及びスーパーコンピュータシステムのあり方並びに設備運用のノウハウや勘所（＝暗黙知）を抽出・可視化し，次世代実務者の礎とすることを目的とする．

0.2 本報告の構成

本報告は，以下の 5 部 15 章構成より成る．

第I部 導入とシステム概要
- 序章 まえがき
- 第1章 NS-III 導入の背景
- 第2章 NS-III の基本構想と調達
- 第3章 NS-III のシステム概要

第II部 性能評価・性能推定とチューニング
- 第4章 NS-III のシステム基本特性
- 第5章 CeNSS における JAXA アプリケーションの性能評価
- 第6章 CeNSS におけるスカラー性能チューニングの指針に関する一考察
- 第7章 性能評価と性能チューニングの実例（その 1）
- 第8章 性能評価と性能チューニングの実例（その 2）
- 第9章 性能評価と性能チューニングの実例（その 3）
- 第10章 CeNSS における並列アプリケーションの実効性能推定法とその有効性検証

第III部 運用と利用
- 第11章 CeNSS の運用分析と課題
- 第12章 CeViS の運用分析と課題
- 第13章 NS-III の利用，航空宇宙への初期応用成果と将来展望

第IV部 次世代への課題と展望
- 第14章 JAXA 統合スーパーコンピュータの導入に向けての準備的考察
- 第15章 次世代スーパーコンピューティングへの課題と展望，あとがき

第V部 付録
- 付録 A NS-III の導入根拠資料
- 付録 B スーパーコンピューティングをめぐる内外の情勢
- 付録 C スーパーコンピュータの技術動向
- 付録 D 並列処理の技術動向
- 付録 E スーパーコンピュータ政府調達手続き
- 付録 F 新中央可視化システム（CeViS）の導入とその概要
- 付録 G NS-III システム運用設計書 第 2.1 版
- 付録 H 中央 NS システム（CeNSS）性能チューニングガイド Ver. 1.0

本報告において，各章の執筆は以下の者が分担した．

第 1 章～第 2 章	松尾裕一
第 3 章	藤田直行，松尾裕一
第 4 章～第 7 章	松尾裕一
第 8 章	高木亮治
第 9 章	坂下雅秀
第 10 章	松尾裕一
第 11 章	土屋雅子，藤岡晃，染谷和広，松尾裕一
第 12 章	末松和代，松尾裕一
第 13 章～第 15 章	松尾裕一
付録 A～E	松尾裕一
付録 F	末松和代，松尾裕一
付録 G～H	松尾裕一
編集，校正	松尾裕一

見やすさの観点から，参考文献は各章の最後にまとめている．また，年度表記はできるだけ西暦に統一するとともに，コンピュータ用語や省略語は必要に応じて脚注等で説明するようにした．最後に，本報告を執筆するにあたり，中村孝氏には特別のご協力を賜った．また，主システム提供ベンダーである富士通株式会社の関係者の方々，とりわけ矢澤克己，稲荷智英の両氏には，多大なるご支援ご協力を賜った．ここに記して謝意を表したい．

表 0.1 年号対応表

平成 10 年	11 年	12 年	13 年	14 年	15 年
1998 年	1999	2000	2001	2002	2003

平成 16 年	17 年	18 年	19 年	20 年	21 年
2004 年	2005	2006	2007	2008	2009

第 1 章　NS-III 導入の背景

1.1　はじめに

本章では，本報告の主題えだる数値シミュレータ III（NS-III）の導入の基礎となった数値シミュレータ計画について，数値シミュレータ I と II のそれぞれ中核計算機であった VP400，数値風洞（当時の世界最高速を達成）とそれらによって目指したもの・達成したものについて述べる．また，NS-III の導入の概要及びその後の状況について簡単に触れ，NS-III 導入の背景や全体像を俯瞰することとする．

1.2　数値シミュレータ計画と数値風洞

数値シミュレータ III の導入は，旧航技研時代に始まった「**数値シミュレータ（Numerical Simulator; NS）計画**」にその由来を求めることができる．航技研の数値シミュレータ計画は，スーパーコンピュータの計算処理能力を利用して，**計算流体力学（Computational Fluid Dynamics; CFD）**に代表される数値シミュレーション技術の発展と普及ならびに航空宇宙機の国際共同開発における我が国の地位の向上と確立を目指して 1980 年代に立ち上げられたものである．**図 1.1** は，数値シミュレータ計画において導入されたスパコンシステムの変遷を示したものである．

第 1 世代数値シミュレータ NS-I は，1987 年，富士通のスパコン VP400 の導入によりスタートした．VP400（**図 1.2**）は，1GFLOPS[1]の演算性能を有し，三次元翼のナビエ・ストークス解析や全機形態の非粘性解析をはじめて可能とした．

第 2 世代の数値シミュレータ NS-II は，1993 年，「**数値風洞（Numerical Wind Tunnel; NWT）**」の導入とともに始動した[1]．NWT（**図 1.3**）は，故三好甫（みよしはじめ）氏（写真 1.4）が富士通とともに開発した世界屈指のスパコンである．ナビエ・ストークス方程式をベースとする CFD 技術の実機開発への展開を目指し，クリーン全機のパラメータ解析を行うのに 1M（=100 万）格子点の計算を 10 分で行うことを念頭に VP400 の 100 倍以上の性能をターゲットに開発された[2]. NWT は，1 台当たり 1.6GFLOPS のベクトル要素計算機（PE）166 台から構成されるピーク性能 280GFLOPS，主記憶容量 44.5GB の分散メモリ並列計算機である．NWT の構成を**図 1.5** に，主なスペックを**表 1.6** に示す．NWT の要素計算機（PE）は，ベクトルユニット（VU）を含む計算ユニットとメモリユニットから成る．NWT の特徴として，当時としては高速の GsAs の LSI の使用と冷却に水冷を用いたということが挙げられる．水冷を用いたのは，半導体のジャンクション温度を下げて電力消費を下げたいという現代的な理由からではなく，常に動いている VU が大量の熱を発生するために，空冷では熱を取りきれなったためであり，より洗練された後代のシステム（富士通製 VPP5000 等）は空冷となっている[3].

NWT の導入により，航技研は 100GFLOPS 級の計算パワーを武器に本格的に並列シミュレーションを行う時代へと突入した．NWT の処理性能は当時としては破格であり，90 年代中盤から後半にかけての我が国における CFD の研究開発活動をリードした．**航空宇宙分野の設計開発におけるナビエ・ストークス解析定着への足がかりを作る**とともに，**乱流シミュレーションなどを通じて流体の基礎分野の発展にも寄与**した．この間，94 年からは 3 年連続して米国電気電子学会（IEEE）のゴードンベル賞（Gordon Bell Prize，**図 1.7**）を受賞した[2]．そのときの性能値を**表 1.8** に掲げる．ベクトル機といえばその実効効率の高さに象徴されるが，並列計算に関する技術蓄積がほとんどなかった時代に，この数字をたたき出しているのは驚異的ともいえる．また，計算機としての処理性能・メモリ性能・通信性能のバランスが如何に良かったかをあらわしている．今でこそ，地球シミュレータの実績により，ゴードンベル賞といえば我が国の十八番のような印象を受けるが，競争の激しい計算機の分野で同一の計算機が 3 年続けて受賞したというのは，この計算機が如何に時代を先取りしたものであったかを示している．

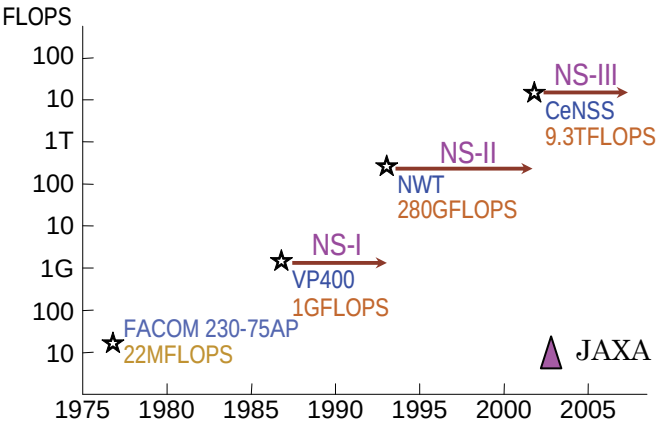


図 1.1　数値シミュレータの変遷

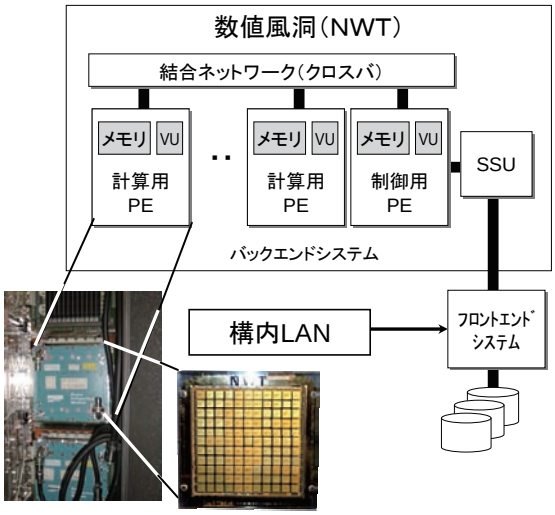

図 1.2　VP400 の外観

[1] Giga Floating Point Operations/sec，FLOPS は，1 秒間に 1 回の浮動小数点演算を行う計算処理性能のこと．

[2] 正確には，94 年には「乱流の直接シミュレーション」により「性能部門特別賞（Honorable Mention）」を，95 年には「量子色力学（QCD）シミュレーション」により「性能部門賞（Winner）」を，96 年には「圧縮機の全周シミュレーション」により同賞を受賞した．

図 1.3 数値風洞 NWT の外観

写真 1.4 故三好甫氏



VU: Vector Unit
PE: Processing Element
SSU: System Storage Unit

図 1.5 数値風洞の構成概要

表 1.6 NWT の主なスペック

ハードウェア	
全体性能	280GFLOPS 45GB メモリ
要素計算機数	166
要素計算機性能	1.6GFLOPS/10ns 256MB メモリ
クロック周波数	9.5ns(=105MHz, 当初), 10ns(=100MHz, 1999.6〜)
メモリバンド幅	6.4GB/s/PE
結合ネットワーク	クロスバ(421MB/s×2)
制御ノード数	2
SSU	24GB
FEP	NWT-FEP(63.8MIPS×2, 256MB)
ログインサーバ	SUN S4×5
ディスク量	チャネル接続ディスク=300GB, SCSI ディスク =2TB
テープ量	500GB
ソフトウェア	
OS	UNIX SVR4(UXP/M)
コンパイラ	F77, F90, C, NWT-F/VPP-F, MPI/PVM/PARMACS
ツール	Eventlog, Vamir, Work Bench
設備	
設置面積	410m²
消費電力	1,000KVA/166PE, 6KW/1PE
冷却設備	284,494kcal/h 空冷 696,250kcal/h 水冷

図 1.7 ゴードンベル賞(上 1995 年, 下 1996 年)

表 1.8　NWT における主なプログラムの処理性能

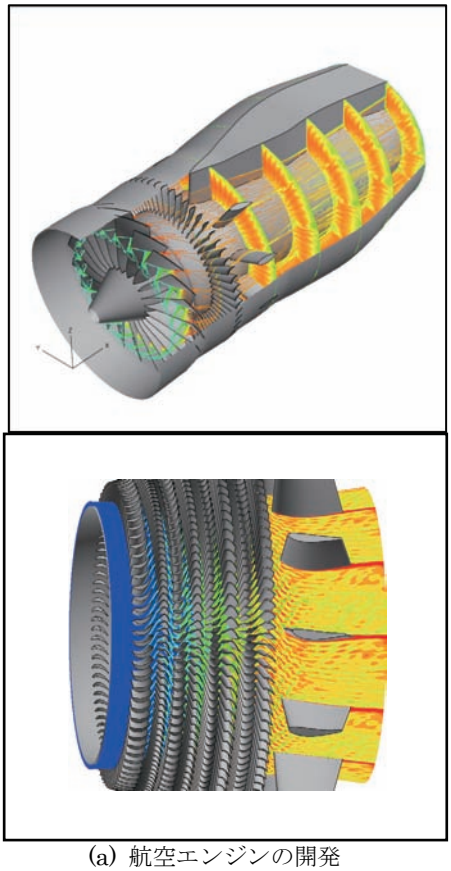
ﾌﾟﾛｸﾞﾗﾑ名	ｱﾌﾟﾘｹｰｼｮﾝ	性能実測値 GFLOPS	PE 数	実効効率	測定時期
NS3D	航空宇宙 RANS	150.4	166	53.7%	1994 年 2 月
BIGCUBE	一様等方性乱流 DNS	90.3	128	41.5%	1994 年 12 月
LINPACK	連立一次方程式	229.7	167	82.1%	1995 年 8 月
QCD	量子色力学	215.8	160	79.3%	1995 年 2 月
CMPRSSR	エンジン圧縮機 URANS	111.0	160	40.8%	1995 年 9 月

本格的な並列計算機などというものが，NWT 以外にはたいしてなかった時代であるから，並列計算技術という点では相当に未熟であった．ちなみに，PVM[3]が出てきたのは NWT の時代の中盤以降と記憶している．従って，NWT で作成された CFD コードは，実用からはほど遠いものであり，どちらかといえば可能性提示のレベルであったといえる．しかし，NWT のおかげで，我が国の CFD 技術は諸外国と肩を並べられるようになったのであり，航空宇宙分野において CFD がいち早く実用に供されるに至った一役を担っているのではなかろうか．しかし，可能性提示は可能性提示であって，どちらかといえば計算機パワー頼みのところがある．数値シミュレーションにおいてほんとうに難しく大変なのは，その先の学術的発見や実用的用途に結びつけるところであり，数値シミュレータ計画としては，可能性提示の先に進む必要があった．（無論，可能性提示のような話は今日でもあり，それは依然として，JAXA のような研究機関の重要な役割でもあるが，当時は何をやっても新しいという雰囲気があった．）

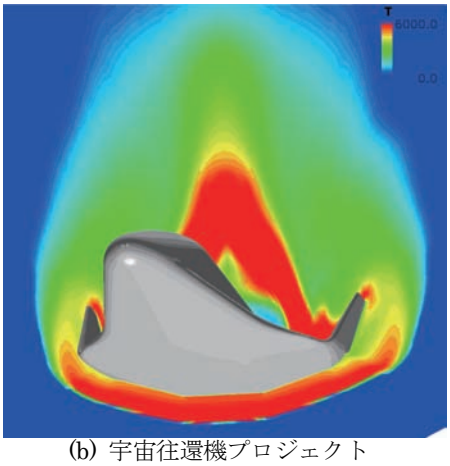
そういったこともあって NWT 時代の後半には，富士通が VPP5000 という一段と成熟したベクトル並列計算機を世に送り出したせいもあり，また，LINPACK トップの座も明け渡していたため，NWT に対しては，使いにくい，移植性がない，というような厳しいコメントが寄せられ，産業界も含めてポスト NWT への期待が高まっていた．

そういう中で，数値風洞は当初更新予定の 7 年目の運用を終えたものの，航技研の独立法人化（2001 年 4 月）などの諸事情も重なり，結局 2002 年 6 年までの 9 年 2 ヶ月の長期にわたって運用されることとなった．この間，（後の**図 2.2** に示すように）高稼働率を維持しつつも大きな問題はまったく発生していない．このことは，とりもなおさず，NWT が，信頼性の高い優れた計算機であったことを物語っている．NWT の主な成果は文献[4]を参照されたいが，NWT による数値シミュレーションが実際に貢献した航空宇宙関連の代表的プログラムとしては，次のものが挙げられる．各プロジェクトの代表的な計算結果を**図 1.8** に示す．

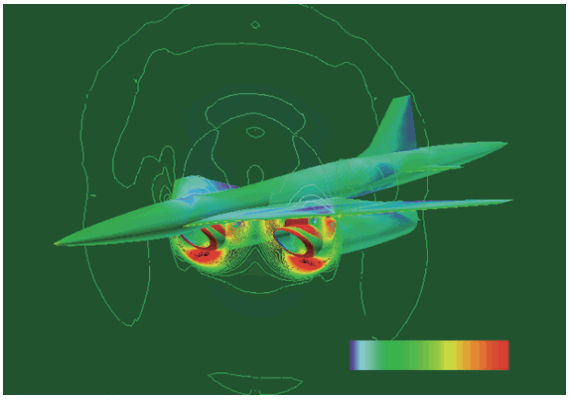
- V2500，CF34 等航空エンジンの国際共同開発（1987～）
- 宇宙往還機プロジェクト（1990～）
- 次世代超音速機プロジェクト（1997～）



(a) 航空エンジンの開発



(b) 宇宙往還機プロジェクト



(c) 次世代超音速機プロジェクト

図 1.8　代表的な計算結果

[3] Parallel Virtual Machine. メッセージパッシングの環境とライブラリを提供.

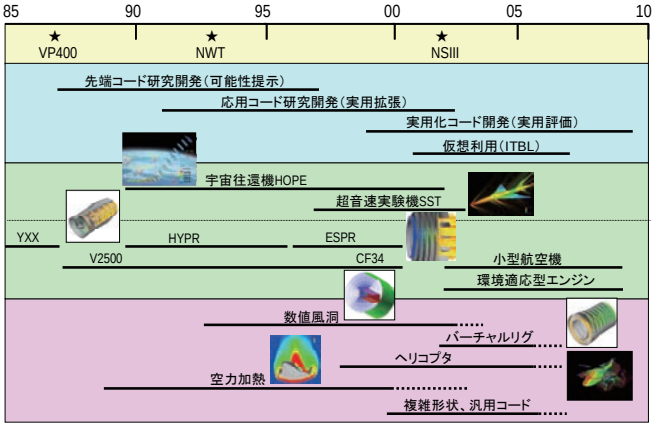


図 1.9　数値シミュレータ計画と応用

図 1.9 には，数値シミュレータ計画とそれが係わった代表的なプロジェクトや課題を示す[4].

1.3　数値シミュレータ III(NS-III)と JAXA スーパーコンピュータ

NWT は，運用の終期においても計算エンジンとしてはそこそこの性能を有してはいたが，後述するように周辺がついていかなかった．また，利用アプリケーション的にも高性能ベクトル機だけがポツンとあれば良いという時代は過ぎつつあった．

こうした状況の中で第 3 世代の数値シミュレータ NS-III は，2002 年 10 月，中核となる計算機 CeNSS（第 3 章で詳述）の本格稼動とともにスタートした．CeNSS は，富士通製のスカラー型 SMP の PRIMEPOWER HPC2500　18 筐体から成り（**図 1.10**），計算部分の性能としては，ピーク性能 9.3TFLOPS および総メモリ 3.6TB を有する．NS-III はまた，総容量で 500TB を超える大規模ストレージや，高性能の可視化システムを有する．NS-III の導入経緯，システム要件, 構成概要, 性能特性などについては, 第 2 章で述べる．

航技研は，2001 年 4 月の時点で独立行政法人化（独法化）されたが，数値シミュレータ計画（すなわち数値シミュレータの予算）そのものは，独法化後も存続した．独法化される（2001 年 4 月）以前は，スーパーコンピュータの調達に係る重要案件は，「数値シミュレーション等技術検討委員会」なる全所的な委員会において審議決定されていた．独法化後は，管理業務の簡素化という路線の中で委員会は廃止され，「CFD 技術開発センター」が企画と実行の両方の役割を担った．その後，2003 年 10 月，旧航技研，旧宇宙科学研究所，旧宇宙開発事業団が統合し，宇宙航空研究開発機構（JAXA）が誕生したのは未だ記憶に新しいところであるが，NS-III は，そのまま JAXA に移管され，「情報技術開発共同センター」の所管とされた．その後，2005 年 10 月からは，「計算・情報工学センター」により，角田事業所，相模原キャンパスにあるスーパーコンピュータと合わせて，JAXA スーパーコンピュータの一部として JAXA 情報化事業[5]の中で管理運営されている．

上記では，数値シミュレータ計画において導入された主なスーパーコンピュータについてのみ言及したが，1989 年には，富士通製の VP2600（2GFLOPS）が，1993 年にはインテル製の Paragon 366 システム（25.2GFLOPS），クレイ製の Y-MP，M92（700MIPS，8GB メモリ）が導入されている．また，数値シミュレータと言ったときは，調布事業所のスーパーコンピュータのみを指しているが，角田事業所においては，2002 年に NEC 製の SX-6（512GFLOPS,「数値宇宙エンジン」）が導入されている．2004 年 10 月に JAXA となってからは，相模原キャンパスの NEC 製の SX-6（1.1TFLOPS,「宇宙科学シミュレータ」）も加えて，JAXA スーパーコンピュータとして 3 箇所のスーパーコンピュータは統合的に運用され，今日に至っている．**図 1.11** に，各事業所におけるスーパーコンピュータの設置経緯を, **表 1.12** に，2007 年 10 月現在の JAXA スーパーコンピュータの諸元を示す.

図 1.10　CeNSS の外観

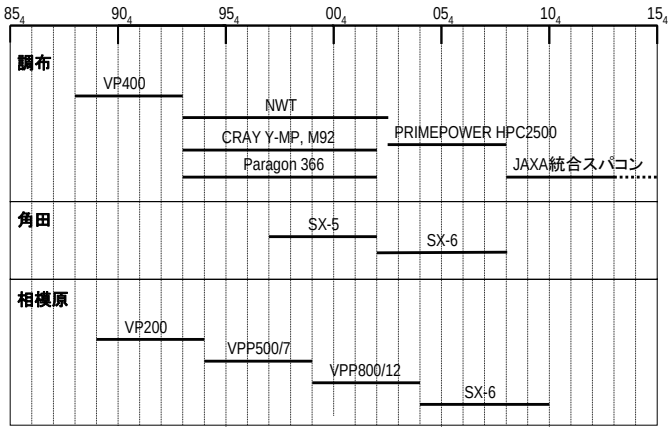


図 1.11　3 事業所のスーパーコンピュータの導入経緯

表 1.12　JAXA スーパーコンピュータ

事業所	調布	角田	相模原
ｼｽﾃﾑ名	数値ｼﾐｭﾚｰﾀ	数値宇宙ｴﾝｼﾞﾝ	宇宙科学ｼﾐｭﾚｰﾀ
呼称	NS	NSE	SSS
ﾋﾟｰｸ性能	9.3TFLOPS	0.5TFLOPS	1.15TFLOPS
総ﾒﾓﾘ量	3.6TB	0.5TB	1TB
ﾉｰﾄﾞ構成	SMP	SMP	SMP
ﾉｰﾄﾞ性能	166GFLOPS	64GFLOPS	72GFLOPS
ﾉｰﾄﾞﾒﾓﾘ	32GB	64GB	64GB
ﾉｰﾄﾞ数	56	8	16
ｲﾝﾀｰｺﾈｸﾄ	単段クロスバ	単段クロスバ	単段クロスバ

NS:　Numerical Simulator
NSE:　Numerical Space Engine
SSS:　Space Science Simulator

1.4　おわりに

本章では，数値シミュレータ III（NS-III）の導入の基礎となった数値シミュレータ計画について，数値シミュレータ I の中核計算機であった VP400，II の中核計算機であった数値風洞（当時の世界最高速を達成）と，それらによって目指したものとその成果を中心に述べた．また，全体の流れを俯瞰するために，NS-III の導入概要とその後の近況についても簡単に触れた．

数値シミュレータ計画における NS-II（NWT）の存在は，言うまでもなくきわめて大きなものであったし，それが世の中に与えた影響も相当なものがあり，その技術の集大成が「地球シミュレータ」につながり，それがまた大きな成功を収めたことは，プロジェクトとしての数値シミュレータ計画の成功局面の一つとして位置づけることができよう．しかし，我々の航空宇宙の分野に目を転じてみれば，NWT といえども（航空宇宙でも大きな成果を上げたとはいえ），当初の計算要求さえ十分満たしたとはいえず，それどころかもっと多様なニーズを生み出し，NS-III への要求要件・導入へとつながった．これには，要求→計算機→要求→計算機という正のスパイラル・メカニズムが働いているからという分析が従来からある．**図 1.13** は，数値シミュレータ計画によって導入されたシステム性能と，行われた解析の事例を年代順に列挙したものであり，**単純形状→複雑形状**，**単一分野→多分野統合**，**研究開発→実利用**，**単純解析→最適化・設計適用**という形でシミュレーションの方も計算機と並行して発展して来ており，その発展は未だもって経路の途上にある．設備としてみたとき，**利用要求と設備仕様がこのようなスパイラル的な関係**を保ち続けている例は他には少ないのではないかと思われる．我々はこの良好な関係を崩すべきではなく，そのあたりの事情，メカニズムを記録し将来の糧とすべきであり，本報告を執筆した理由の一つもそこにある．

参考文献

[1] 三好甫：航技研超高速数値風洞（UHSNWT）の構想 - 第二期数値シミュレータ計画，航技研報告 TR-1108，1991.
[2] 三好甫：CFD の推進に必要な計算機性能，航技研特別資料 SP-13，1990，pp.1-26.
[3] http://museum.ipsj.or.jp/computer/super/0020.html
[4] 数値風洞報告集：航空宇宙技術研究所 CFD 技術開発センター，2002 年.
[5] JAXA 情報化計画：宇宙航空研究開発機構，2006 年 11 月.

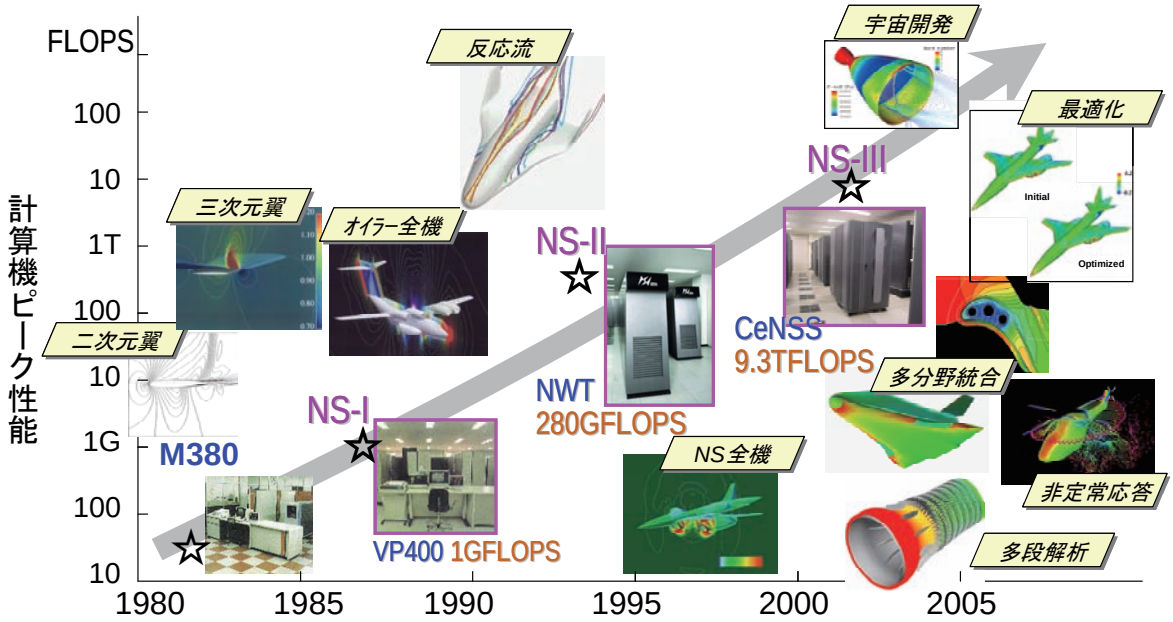


図 1.13　数値シミュレータ計画によって導入されたシステムと行われた解析例

第 2 章　NS-III への基本構想と調達

2.1　はじめに

本章では，JAXA の共用計算機システム「数値シミュレータ III」の基本構想とその調達の経緯について述べる．NWT の特性と課題について振り返った後，数値シミュレータ III に付随する具体的なアプリケーションや利用に関する基本構想について言及し，そこから提起されるシステム要件，構成イメージを述べる．最後に，システム調達及び導入までの経緯について述べる．

2.2　数値風洞の特性と NS-III 導入への機運

NWT は，**表 1.8** に示したように，航技研が当時所有していた各種流体解析プログラムに対して，かなり高い実効効率を達成している．その理由は，i) CFD プログラムの多くが，多重ループ構造を持つため NWT のようなベクトル計算機に向いていたから，ii) NWT のノード間通信性能が比較的高かったからと推察される．しかしながら，オブジェクト指向の記述・言語を使った最近の開発コードや，ファイルを多数書き出す非定常解析においては，実効性能が出ない，I/O 性能がボトルネックになる，といったケースも出現していた．**表 2.1** は，NWT にかけられた全てのジョブ[4]の平均ベクトル利用率[5]の年度別推移を示したものであるが，平成 10 年を境に下降しているのはまさにそのような傾向の現れといえる．航技研では，多くの研究者が自身でプログラムを書き並列化も行っている．並列プログラミングには，データ並列系の言語として **NWT-Fortran** を，メッセージパッシング系の言語として **MPI** を用いていた（**付録 D** 参照）．NWT-Fortran は，**仮想グローバル空間の採用**，**通信の明示**など良い面もあったが，NWT でしか走らないので移植性に難があり，また，MPI は，まだ世に出たばかりで，逐次プログラムからの移行が簡単でないという問題があった．

航技研の当時の CFD においては，エンジニアリング系では，付属物付きの全機のナビエ・ストークス解析や多段のエンジン内部流解析が可能となっていた．これは，ある意味では，NWT で求めた方向が実現されたといっても過言ではない．格子点規模は 5M 点～50M 点，計算時間は 10～20 時間程度を要した．一方，サイエンス系では，直接数値シミュレーション（Direct Numerical Simulation; DNS）が主要な解析手段となり始めており，10M～100M点規模で乱流や燃焼の DNS が行われていた．処理性能的には何とか耐えられるものではあったが，扱う問題が複雑になるにつれ，メモリやディスクの少なさが，問題点として顕在化し始めていた．

表 2.1　平均ベクトル利用率の推移

1996 年度	1997 年度	1998 年度	1999 年度	2000 年度
50.6%	62.4%	64.0%	61.2%	59.1%

[4] 計算機に依頼する仕事の単位を「ジョブ」という．
[5] ベクトル利用率とは，全 CPU 時間に対してベクトル演算器が利用された時間の割合を指す．

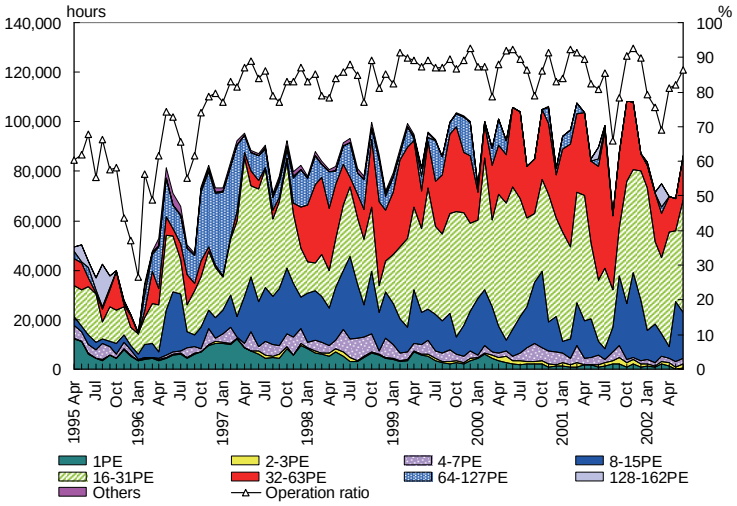


図 2.2　数値風洞の稼働率の推移

図 2.2 は，NWT の導入当初からの稼働実績をプロットしたものである[1]．導入 3 年目以降は，定常的に 90%という高稼/働率を達成している．利用時間からいうと，ノード（PE）を多く使う（16 台以上の）並列ジョブの割合が増えており，利用者が並列計算に徐々に馴染んできているのがわかる．しかし，稼働率 90%という状態が長く続き，システムとしてはもはや次の世代に移行するタイミングであった．NWT の性能や成果に関する記述は枚挙に暇がないが，システムとしての特性や問題については，福田正大氏のコメント[2]が次の NS-III を考える上で大変参考になるので，ここに掲載しておきたい．

「このように書いた以上，数値風洞のシステムとしての問題点も書いておかなければならないだろう．その最大の問題点はシステムとしてのバランスの悪さにある．つまり計算機"だけ"は速いが，磁気ディスク容量や I/O 性能，前後処理システム（少なくとも数値風洞導入当初には可視化システムはなかった），ネットワークの有りよう，等々である．車で言えば，エンジンだけは立派だがシャーシーや足回りなどがエンジン性能に見合っていない代物であった．このことは計算機を研究手段として利用する研究者にとってはある意味では致命的欠陥である．（中略）このシステムバランスの悪さは三好さんが手掛ける計算機プロジェクトについて回り，地球シミュレータにおいても"然り"である．敢えていえば"計算機それ自身のスピードを追い求める"三好さんのプロジェクトの限界ともいえる．一方それはまた，我が国の多くのセンターがこれまで"目に見える数値"として，導入する計算機の CPU（処理）性能によって競争せざるを得ない，という状況に置かれていたことにも原因があるように思う．処理性能というのは分かりやすい数値であるが，システムバランスというのは説明の受け手に対する印象が弱く，限られた予算であればできるだけ多くを処理性能に投資しよう，というのがこれまでの流れであった．三好さんといえどもその制約の中にいた，という方が正鵠を射ているのかもしれない．」

2.3 NS-III の基本構想とシステム要件

こうした状況を踏まえ，航技研では，数値シミュレーション等技術検討委員会（以下，「NS 委員会」と略.）を中心に，当時の国の答申等（**付録 A** 参照）の流れを参考に，次世代の CFD として何を目指すべきか，次のスパコンとしてどの程度の性能・機能のものが必要か等について調査検討し，「**NAL 計算科学ビジョン 21 報告書**」として纏めた（1999 年 10 月，**付録 A**）．その中で，航技研が果たすべき役割を，1)先駆的 CFD 技術研究開発への挑戦，2)実用に耐えうる CFD 技術の確立，3)CFD 技術の研究拠点たること，4)利用方法，応用分野の開拓と実用性の実証，であると再整理した上で，今後取り組むべき重点課題として，**① ボトルネック技術課題への挑戦と克服，② 信頼性の高い標準設計解析ツールの整備，③ 次世代統合シミュレーション技術の構築，④ 高速高機能計算機の実現，**の 4 項目を掲げた[2][3]．以下に述べる第 3 世代数値シミュレータ NS-III の主なシステム要求項目は，このような調査検討の中から生まれてきたものである．

CFD におけるボトルネック的技術課題と言えば，複雑形状まわりの格子生成と物理現象のモデリングは今日的課題の代表的なものであろう．特に，実用形状に関する大規模格子の作成能力は，CFD の信頼性・生産性を左右する重要因子であり，品質を確保しつつ短時間で作業を行うことが求められる．航空宇宙の CFD が難しいのは，物体表面上に極めて薄い境界層が発達し，その境界層を如何に正確に捉えるかによって全体の計算精度に大きな影響を与えるからである．この境界層の計算精度が高いのが「構造格子法」である．しかし，複雑形状まわりを単一の構造格子で覆うのは不可能なので，複数の領域（ブロック）に領域分割して各領域で独立に格子を作成し，それらを連結させ全体格子とする．領域単位で並列化すれば並列計算にもなじみやすい．この方法は「マルチブロック構造格子法」と呼ばれ，対象形状が複雑化しても原理的には対応可能である．ただし，実用形状では，ブロック数が簡単に 100 以上となり，多数のブロック分割を如何に効率良く行うか，並列計算の負荷分散を如何に促進するか等が課題となる．システム要件としては，**並列計算を効率良く行う機能や，計算負荷を把握する機能，プロセスのスケジューリング機能**などが必要となる．

CFD の技術課題としてもう一つ重要なのは物理モデリングである．レイノルズ平均ナビエ・ストークス（Reynolds-averaged Navier-Stokes; RANS）解析が CFD の主流となっている今日，CFD 技術の定量性を高めるには高精度な乱流モデルの確立が必須である．そのモデルを開発，改良するのに，今後は直接数値シミュレーション（Direct Numerical Simulation; DNS）から得られたデータベースの利用が中心となるであろう．モデリング技術を向上させるには，より現実的な形状及び多様なパラメータ条件（レイノルズ数，プラントル数など）に対する DNS データを取得する必要がある．また，流体現象と化学反応との連成問題である燃焼などのマルチフィジクス問題に対する DNS にも取り組む必要がある．NWT では各空間方向に 250 分割程度の DNS が行われていたが，次のステップとして各空間方向について数倍以上の空間解像度，すなわち 10 億格子点規模の DNS が求められる．それを実現するには，**テラバイト規模の主記憶容量を有する高性能計算機**が必要となる．また，得られた DNS の結果を**チャンピオン・データベースとして発信するに足る十分な容量のデータ蓄積能力，データ管理能力**が求められる．

航技研は NS-II において，「数値風洞」と題して CFD を風洞試験の代わりに使うというコンセプトによって，風洞試験を指標に一種の標準的な CFD 技術の確立を目指した．今後，そのコンセプトをさらに押し進め，信頼性の高い標準 CFD 解析ツールを整備開発して行くには，そのような開発を可能とする利用環境を構築して行く必要がある．また，これからはネットワークを通じての利用を主体的に考慮して行かなければならない．こうしたことから，**標準化やオープン化への対応，運用性・操作性の統一，信頼性・レスポンスの保証，セキュリティの確保といった主としてソフトウェアの機能的側面**がより重要となる．これは，従前のスパコンシステムにはなかった要件であり，次のシステムは単なる計算エンジンとしてだけではなく，**広範なサーバ機能**を併せ持つ必要があることを意味する．

NS-III の導入当時，我が国で進められていた小型超音速実験機計画や再使用宇宙往還機開発などのプロジェクトにおいては，CFD 技術を様々な形で設計開発に利用し，試験・試作回数を極力減らすといった種々の試みがなされていた．メーカの開発現場からの開発期間の短縮，コスト削減，環境への配慮といった要求を満足させるためにも，この方向をますます加速させる必要があった．CFD 技術としては，より現実・実際に近い状態や条件に対する適応能力と同時に，1 日程度の現実的なターンアラウンド時間で答えを出すことが求められる．そのために，従来の要素ごとの解析技術を融合して，より精度の高い性能評価や設計を行うために，**図 2.3** に例示したような CFD と他の分野の連成問題を扱う多分野統合解析技術の確立を図る必要がある．特に，機体同士の分離やフラッタなどの物体移動や動的応答などを伴う非定常挙動を厳密に追跡する必要が出てくる．

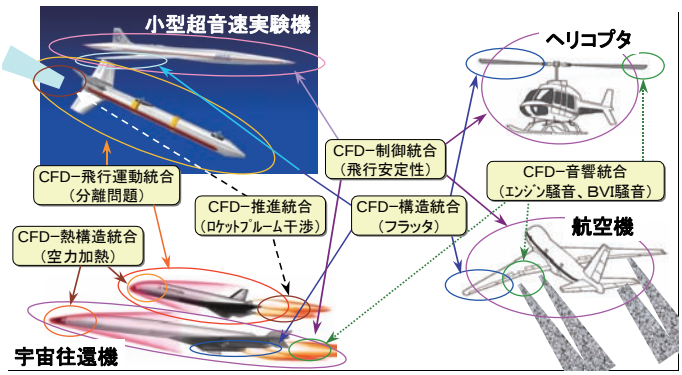


図 2.3 多分野統合解析の事例

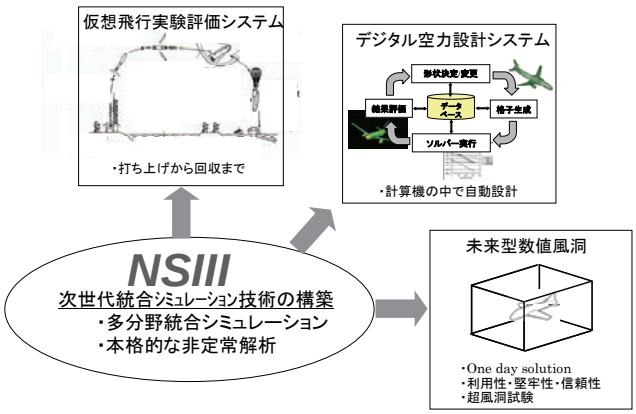


図 2.4 次世代統合シミュレーションの目指すもの

これらの要求に応えるには，**その日のうちに性能曲線を書くなり，最適化ループを回すといった 10 ケース前後の解析（パラメータスタディ）をこなす程度の処理能力と，本格的非定常計算を行うための膨大な時系列データを処理管理する能力や高速に結果を可視化する能力，すなわちデータハンドリング能力**が求められる．このような方向の将来ターゲットとして，例えば，打ち上げから回収までを計算機の中で行う仮想飛行実験評価システム，航空機等の空力設計を計算機の中で自動的に行うデジタル設計システム，現実の風洞試験では不可能な条件での試験までも可能にする未来型数値風洞などが考えられ（**図 2.4**），**計算パワーは無論のこと計算機システムとしての総合処理能力**が問われるようになる．

以上の基本構想下に，スーパーコンピューティングの内外の情勢（**付録 B** 参照）を視野に入れつつ，NS-III として具備すべき主要システム要件を次のように試算・整理した．

処理速度として，「30M 点（1M 点＝100 万点）の多分野統合解析を翌日までに 10 ケース程度処理する」という目標下に，現行（前述）と照らして，点数で数倍程度，解析の複雑さで 2 倍，計算時間は同程度，ケース数では 10 倍，積算で現行の 30-50 倍程度が必要として，きりの良い数字として **10TFLOPS** 程度と設定した．

メモリ量については，「1G 点程度の燃焼の DNS を行えるように」という目標を念頭に，現行（前述）と照らして，点数で 4^3=64 倍，解析の複雑さで 2 倍，積算で 100 倍程度が必要として **5TB** 程度と設定した．この物量は，前述の NWT での反省を踏まえるとともに，現行の大規模システムの調査結果や OS などのオーバーヘッド分を加味して性能の 5 割程度（RAM 比=0.5）とした．また，ストレージについては，「メモリ量の 100 倍」として **500TB** 程度とした．100 倍あれば 5 年程度の運用に耐えられると判断したためである．また，これだけのデータ量を扱うのに転送が障壁とならないように，システムまわりのデータ転送速度として，「数 10GB の規模データを 1 分以内に転送する」を目安に，**1GB/s** 程度と設定した．

機能的には前述のソフトウェア的側面の他に「NWT の**資産の継承性**」に配慮する必要がある．これらの性能，機能に係わるシステム要件を**表 2.5** に示す

表 2.5 NS-III に対する主な要求要件

性能	① 10TFLOPS の処理性能 ② 5TB のユーザメモリ ③ 500TB のストレージ ④ 1GB/s のデータ転送性能
機能等	⑤ NWT からのソフトウェア資産の継承性 ⑥ 標準性，汎用性，使いやすさ ⑦ データハンドリング能力 ⑧ 将来への拡張性

2.4 NS-III の構成検討

表 2.5 の性能要件は，単一プロセッサでは到達不可能であり，このような高性能の計算機は「**並列計算機**」構成とならざるを得ない．問題は，どのように並列させるかである．そのときに問題となるのが，第一に要素計算機（ノード）どうしを連結する**結合ネットワーク**である．我々の場合は，NWT での経験と並列数やプログラミング形態が極端に変わらない等のシステムとしての継続性を考慮し，（そういう選択が可能であるならば）結合ネットワークには**単段のクロスバ**が好ましいと判断した．ただし，その場合にスイッチに接続される計算ノードの数は高々100 以下でないとコスト的に現実的ではなく，ノード性能との兼ね合いとなる．単段クロスバが調達不可能な場合には，ファットツリーやクロス網が次善の策となる．また，結合線として，PC クラスタなどに採用されているギガビットイーサ系は，バンド幅の進歩は著しいものの，ミリネットなどの独自なものに比べレイテンシ（立ち上がり）が悪く，同期等のプロセス間での密な連携処理が必要な CFD 向けの結合ネットワークには適さない．

いま，単段クロスバが可能なノード数を 50～100 とすれば，10TFLOPS の処理性能を実現するには**ノード当たりの性能として 100～200GFLOPS** が必要となる．同様の試算から 5TB 程度の総メモリ容量を実現するには，**ノードあたりのメモリは 50～100GB** が必要となる．100～200GFLOPS の性能，50～100GB のメモリを有するノード計算機は，やはり単一プロセッサでは実現不可能であり，**多数の CPU の結合体**にならざるを得ない．しかし，ノード計算機を分散メモリにすると，ユーザから見た計算機ビューはますます煩雑，プログラミングも複雑になるので，ノード計算機は「**共有メモリ型**」が望ましい．**NWT からの構成上のマッピング**という観点からも好都合である．ただし，共有メモリでも，CPU から見たメモリ配置で，**SMP(Symmetric Multi Processor)** か **NUMA(Non Uniform Memory Access)**かという選択肢がある．CPU そのものも，**ベクトル**型か**スカラー**型かという選択肢がある．ベクトル型は，広範囲の大規模科学技術計算に適応することは認識されているものの，性能に見合う高速メモリの開発が技術的にもコスト的にも苦しくなりつつある．また，すべてが特殊なので移植性に乏しいという問題もある．一方，スカラー型は，ピーク性能の向上，コスト，電力などの点で有利ではあるが，キャッシュ技術，並列の場合はコヒーレンシなどの問題点があり，このあたりの問題を如何に解決し如何に実効効率を上げるかに課題がある．

次に，ストレージについては，実現方式の選択肢として，各計算ノードからファイバチャネル等の入出力経路を出し，スイッチを経由してストレージに接続する **SAN**（Storage Area Network）方式と，結合ネットワークに入出力ノード（ファイルサーバ）を計算ノードと一緒に接続し，入出力ノードにストレージを接続する **NAS**（Network Attached Storage）方式とがある．詳細は参考書等に譲るが，それぞれの方式に一長一短があり，コスト性能比等で現実的な選択が決まってくる．

一方，500TB 規模のストレージを実現するには，磁気ディスクのみでは高価になりすぎるので，現実的には**ディスクとテープ装置の混在**という構成にならざるを得ない．ただし，ユーザからみたとき，ファイルがディスクにあるかテープにあるかを区別しなければならないのは面倒であり，使わないファイルは自動的にテープにマイグレートされる**階層的ストレージ管理（Hierarchical Storage Management; HSM）**を実現するのが望ましい．ディスク量はメモリ量の 10 倍程度とする．

1GB/s のデータ転送性能を実現するには，現行の単一の転送技術では不可能であり，**ストライピング**などの線を束ねる工夫も必要となる．また，多くのストレージ装置を高速に長距離で連結できる（例えば**ファイバチャネル**）インターフェースの採用も重要な要素である．利用性を勘案すると，可視化システムなども含めた何らかの形の**高速バックボーンネットワーク**として構築するのが望ましく，システムの構成イメージとしては**図 2.6** のようになる．

ソフトウェアとしては，基本ソフトウェアとしてのOSは，ノードメモリ量や標準性・汎用性の点から，**業界標準の 64 ビット UNIX** を採用するのが現実的であろう．並列プログラミングに関しては，**NWT-Fortran は引き続き利用可能とする**必要がある．また，プロセス並列にはメッセージパッシング系の **MPI** や共有メモリ内での**スレッド並列**にも対応している必要がある．さらに，使いやすさの点から，**ファイルシステム**を透過的かつ高速にするとともに，ユーザからみたとき**シングルシステムイメージ**を実現することが望ましい．また，使いやすい開発環境などへの留意も必要となる．

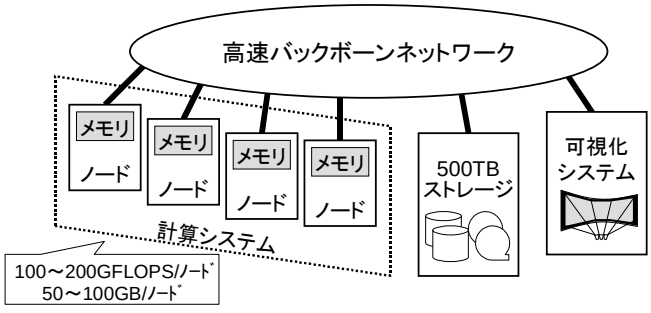


図 2.6　NS-III の構成イメージ

可視化については，最大 1G 点の計算結果を処理するためには，数 10GB のメモリを有するシステムが求められる．また，その場合に，市販のソフトが動くような形態（例えば，共有メモリ）である必要がある．計算系と別システムになるのであれば，高速データ転送についての配慮（実現方法，使い勝手など）が必要である．また，表示系についても，解像度や表示方法などに対して留意する必要がある．

NS-III の構成検討をするに際して参考にした当該分野の技術動向を**付録 B，C，D** にまとめた．

2.5　NS-III の調達

スーパーコンピュータについては，**付録 E** に示したように，「政府調達」手続きによって調達手続きが細部に渡って決められており，この調達手続きに則って行われる必要がある．**図 2.7** に，政府調達手続きによるスパコン調達のマイルストーンと必要な日数を示した．

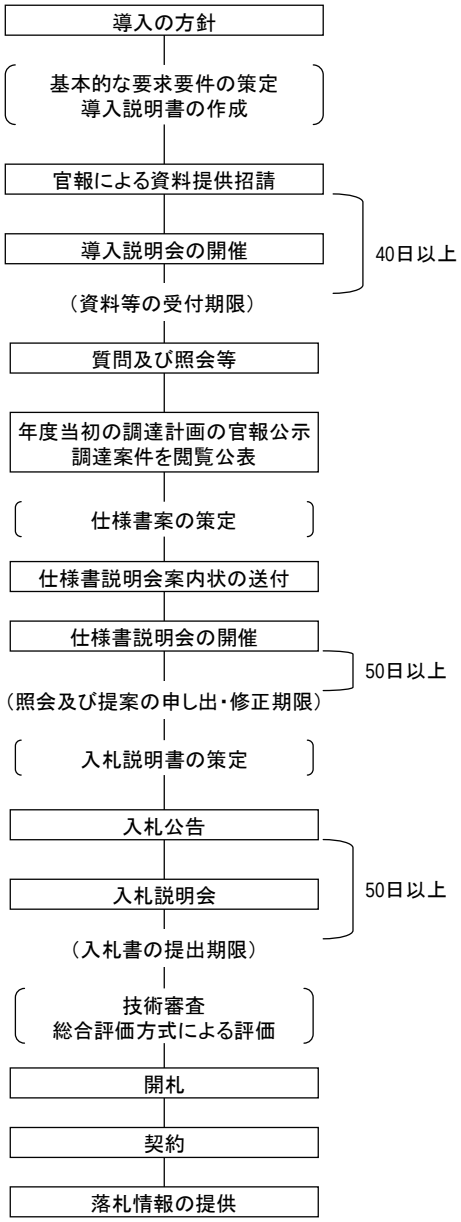


図 2.7　スーパーコンピュータ政府調達手続きフロー

まず行わなければならないのが「資料招請」または「資料提供招請」と呼ばれる手続きである．資料招請においては，技術情報，計画，コストなどの情報を各ベンダーに呼びかけ取得するものである．数値風洞 NWT は，2001 年 2 月に当初設定したレンタル期間（7 年間）を終了する予定であったので，余裕を見て 1999 年 8 月に資料招請を実施した．結果として，部分提案を含め 10 社からの提案があった．しかし，**表 2.5** の要求要件（性能）をクリアしたのは 1 社しかなく，また，コスト的には相当に厳しい提案が多かったことを踏まえ，さらに 1 年程度待てば各社から新機種が提案され，要求要件やコスト制限を満たす可能性が高いことから，スパコン調達の所内審議機関である NS 委員会にて審議した結果，調達を 1 年延期することとした．

図 2.8 は，更新する前の NS-II のシステム全体構成を示した．NS-II においては，計算機として NWT の他に，ファイルサーバ（富士通製 VP2100，**図 2.9(a)**），可視化サーバ（クレイ製 Y-MP M92，**図 2.9(b)**），超並列計算サーバ（インテル製パラゴン XP/S25，**図 2.9(c)**）が存在した．このうち，超並列計算サーバ及び可視化サーバについては，1993 年度の補正予算で導入されたものであり，レンタル品ではないので，性能等はすでに陳腐化しており，今後の取り扱いが懸念されていた．超並列計算サーバについては，並列化や使い方が特殊であり，性能的には新システムに吸収可能であることから，この機会に廃棄することとした．

一方，可視化サーバについては，1999 年前後の可視化は，グラフィックスワークステーションと呼ばれる主にデスクサイドの計算機により部課ベースで行われており，特にメモリが大きなものは非常に高価につくために，部課では整備できないメモリサイズの大きな中央可視化サーバに対するニーズがあった．また，科学技術会議 25 号答申「未来を拓く情報科学技術の戦略的な推進方策の在り方」（1999 年 6 月，**付録 A** 参照）に基づいて，最先端フロンティアの開拓を目指した重点領域が設定され，統合シミュレーション技術，可視化技術，並列分散ソフトウェア技術，アーキテクチャ技術が重点項目として選定される（**付録 A** の第 2 文書参照）ということもあり，可視化に対する世の中の注目度も高まりつつあった．このような背景を踏まえ，可視化サーバについてはこれを更新することとし，費用については，NWT の調達を 1 年遅らせることでレンタル費が減額されたことによる資金を充当することとした．

このような経緯により，**NS-III の調達は，第 1 段階が可視化サーバの代替，第 2 段階が NWT の代替**という形で行われることとなった．NS-III の調達スケジュールを，**図 2.10** に示す．

2.5.1 可視化システムの調達

第 1 段階の可視化システム（可視化サーバを含むシステム全体を指す．）の調達は，スパコンではないのでスパコン調達に従う必要はないが，やはり政府調達に「コンピュータ調達」（**付録 E** 参照）という分類があり，金額規模的にはそれに含まれるものであるため，安全を期すためにこの手続きに則って調達することとした．可視化システムの導入の背景，要求要件，導入，構成等の詳細は**付録 F** を参照のこと．なお，新可視化システムは，2001 年 2 月より稼動し，可視化システムとしてもさることながら，NWT が新機種に換装されるまでの間，CPU 資源提供エンジンとしても重要な役割を果たしたことをここに記しておきたい．

2.5.2 ポスト NWT システムの調達

新システム（NS-III）の調達に当たっては，所内委員会（NS 委員会）において計算科学ビジョンを策定し，今後の CFD 研究開発の方向性を明らかにするとともに，それに基づいた新計算機システムに必要な性能と機能を提示してきた（前述）．並行して，新たなスペースを確保することでシステム停止期間を最少化することを主眼に新たな計算機建屋の建設の提案（2000 年度補正予算，2001 年度日本新生枠）をしてきたが，これらについては予算措置が認められなかった．また，2001 年度配算からはレンタル予算が一部削減されることとなった．

従来の導入（更新）では，旧システムの撤去，新システムの搬入・設置・調整を，一定期間の停止期間を設け，あるいは，年末年始やゴールデンウイークの比較的長期の休館期間を利用して「突貫工事的」に行ってきた．しかしながら，今回の更新においては，NWT システムの冷却方式が水冷式でかなり特殊であり，今後のことを考えると空冷式に設備変更が必要であること，また，NS-III のシステム規模だと，設置・調整には相応の時間と労力が必要と予想されることから，従来の更新方法では最悪数ヶ月の停止期間が発生してしまう懸念があった．このような状況の中で，次期システムに必要な性能と機能を確保しつつ業務への実影響を極力抑えて円滑な導入を行うための適切な導入スケジュールの設定が必要であると判断された．

そこで，計算機の停止を極力短くしたスムーズな移行を実現するために，数ヶ月程度の明示的な「**移行期間**」を予め設定した調達を行うこととした．移行には現行機の解体撤去費用と新システムの導入経費，空調等設備更新経費が別途必要となるが，現状の厳しい資金状況の中で，特別の経費負担を所に要求するのは無理があることを踏まえ，現行のレンタル料の中でやりくり可能な調達計画とすることとした．また，**移行用の小規模システムを導入**し，移行期間中はシステムの停止を避けつつ漸次移行を達成する導入スケジュールを策定した（**図 2.11**）．

このような一種の年度を跨ぐような調達計画が可能となった背景には，航技研が独法化されたことが挙げられる．中期計画の中では，その組織の責任において柔軟な調達や契約が認められているという背景がある．とはいうものの，前例主義の中で，このような実利的な計画を認めていただいた当時の所の幹部の皆様および事務サイドの方々には改めて感謝申し上げたい．

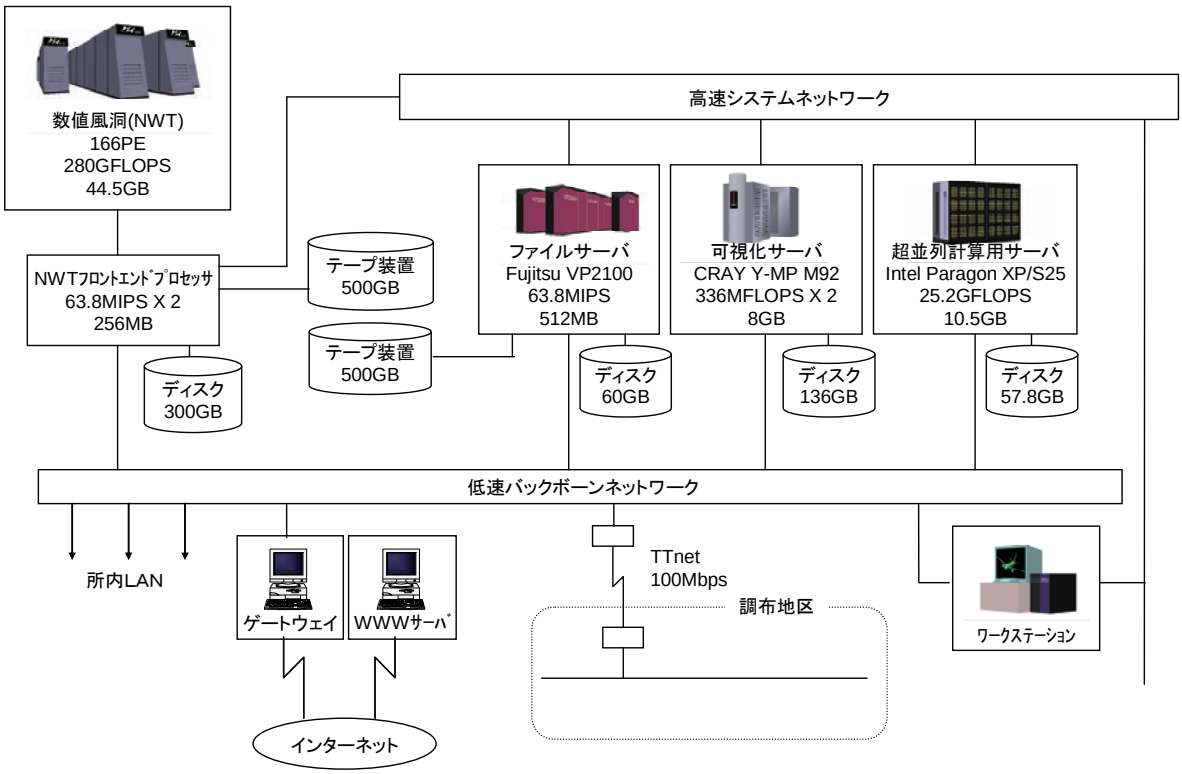


図 2.8　NS-II の構成概要

(a) ファイルサーバ　(b) 可視化サーバ　(c) 超並列計算サーバ

図 2.9　NS-II における各サーバ

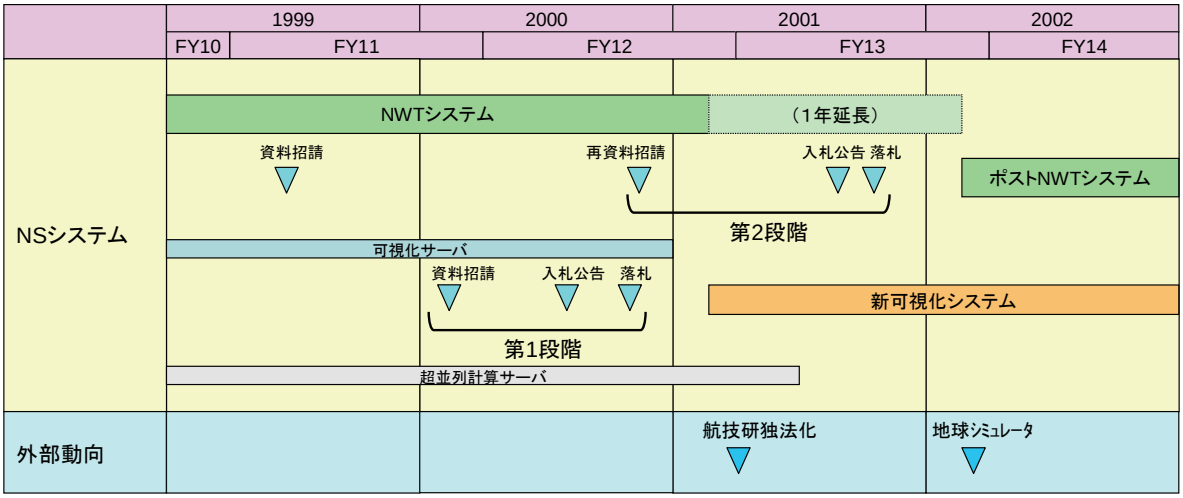


図 2.10　NS-III の調達スケジュール

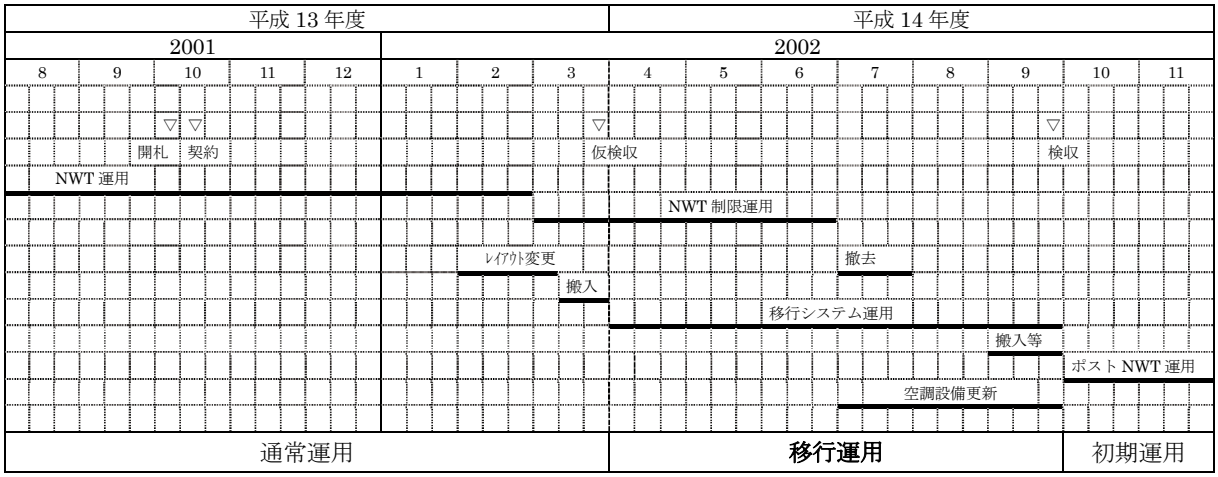


図 2.11 ポスト NWT の導入スケジュール

この**移行期間**を設けた導入スケジュールの論点を整理すると以下のようになる.

1) **移行期間中に冷却設備等の設備を全面的に更新：** 現状の水冷式の冷却設備及び老朽化した設備を移行期間中に一新し，空冷式のものに全面更新し，今後 10 年程度の安定した運用を実現する.

理由： NWT 用は水冷式であるが，現在の計算機はほとんどが空冷式である．空調設備の一部に老朽化が進んでいるもの（15 年経過）があり，維持管理コストが増大するだけでなく，故障してからの修理では運用を一旦止めるためユーザに迷惑をかけるので，そうした事態は避ける必要があるため.

2) **移行用システムを運用することで業務の停滞を最小化**：移行時の混乱や負担を軽減するために，旧システムが停止してから次期システムの本体稼働までの移行期間を予め設定し，その間は移行用の専用小システム（本システムの一部）を運用し，完全停止期間をなくす.

理由： 中期計画期間中は．特にプロジェクトなどにおいてタイムリーに成果を出して行かなければならないので，システムが全面的に停止して計算機が使えなくなる時間を極力短くして所全体の業務の停滞を避けるため.

3) **ユーザの負担を軽減し,スムーズに資産移行**： 旧システムにはユーザが作成したソフトウェア資産(プログラムやデータ) が蓄積しており，これを新システムでも使えるようにスムーズ移行するには,それ相応の時間と労力を要する. そこで, 新システムと旧システムを一定期間並行運用し,ユーザに移行のための大きな負担をかけることなくこれを行う.

理由： NWT システムは，基本的には UNIX ベースの一般的な OS を使っているが，性能を出すために一部特殊な仕様となっており，新システムとの間にプログラムスタイルやデータ形式は単純な互換性はなく，プログラムスタイルの変更やデータ形式の移管が必要である．また，ファイルサーバ VP2100 は，UNIX と異なる旧型の OS を使用しているために，これの移行にはかなりの時間がかかることが予想され，短期間に一斉に移行を行うと予期せぬ問題やデータ消失が発生するなどしてユーザに負担をかけることが懸念されるため.

4) **夏期縮退運転期間を利用し効率良く更新作業を実施**：夏期縮退運転期間中は電力制限により全システム稼働ができないので，検収や調整等の導入作業は行わず，この間に空調設備の更新を行う.

理由： 夏期縮退運転期間運転中は電力供給が制限されるため，その期間中に全システムを設置しても全システム稼働による検収や試験・調整運転はいずれにせよできない．然からば，その間は移行用の専用小システムで何とかやりくりしつつ，必要な空調機の更新などを行ってしまい，通常の電力供給が始まった段階で全システム稼働として検収や調整を行うのが合理的であると判断されるから.

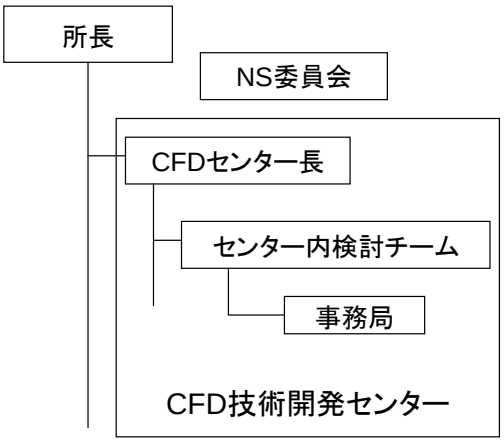


図 2.12 NS-III の調達体制

NS-III の調達における第 2 段階であるポスト NWT の調達は，可視化の部分を除く形でスパコンの政府調達手続き（**付録 E** 参照）に則って行われた．まず，2000 年 11 月に再度の資料招請を実施した．部分提案を含め 9 社からの提案があった．その後の調達経過については**表 2.13** に示した．ここで，参考のため調達体制の概略を**図 2.12** に示す．たたき台案はセンター内検討チームの事務局が作り，検討チームで議論しオーソライズする，全所的なオーソライズは NS 委員会が行うという体制を取った．ただし，NS 委員会は，航技研の独法化（2001 年 4 月）とともに廃止されたので，それ以降は担当部（CFD 技術開発センター）が中心となって調達作業を進めた．

表 2.13　ポスト NWT の調達経過

スパコン調達手続き		時期	担当	作業内容
2001 年				
3/7	意見招請公示手続き開始	3 月上旬	事務局	意見招請起案
		3 月上旬	計算科学内検討ﾁｰﾑ	仕様提案･検討
		3 月中旬	計算科学内検討ﾁｰﾑ	仕様修正，記述チェック
		3 月中旬	NS 委員会	仕様策定委員会の設置
		3 月下旬	計算科学内検討ﾁｰﾑ	仕様書原案（計算科学案）
		4 月上旬	仕様策定委員会	（第 1 回）仕様審査･検討
		4 月中旬	仕様策定委員会	（第 2 回）仕様書原案
4/16	意見招請公示（官報）	4 月中旬	会計職員	
4/20	仕様書原案説明会開催	4 月下旬	事務局，会計職員	
		5 月上旬	CFD ｾﾝﾀｰ内検討ﾁｰﾑ	ベンチマークプログラム準備開始
		5 月中旬	CFD ｾﾝﾀｰ内検討ﾁｰﾑ	性能評価基準書案の作成開始
		5 月下旬	仕様策定委員会	（第 3 回）性能評価基準書案の検討･審議
6/8	意見の提出期限	6 月上旬	事務局	意見のとりまとめ
		6 月上旬	CFD ｾﾝﾀｰ内検討ﾁｰﾑ	意見の検討
	業者への照会，追加資料の要請	6 月中旬	事務局	
		6 月中旬	CFD ｾﾝﾀｰ内検討ﾁｰﾑ	仕様書修正，修正仕様書 CFD ｾﾝﾀｰ案 性能評価基準書案審議，総合評価基準書作成開始
		6 月下旬	仕様策定委員会	（第 4 回）修正仕様書案，性能評価基準書案審議
6/29	仕様書修正内容通知	6 月下旬	事務局	
7/9	入札公告手続き開始	7 月上旬	事務局	入札公告起案
		7 月中旬	CFD ｾﾝﾀｰ内検討ﾁｰﾑ	性能評価基準書 CFD ｾﾝﾀｰ案，総合評価基準書ｾﾝﾀｰ案
		7 月下旬	仕様策定委員会	（第 5 回）仕様書，総合評価基準書，性能評価基準書決定
8/1	技術審査職員任命手続き開始	8 月上旬	事務局	技術審査委員の候補者選出
8/6	入札公告（官報)）	8 月上旬	会計職員	
8/13	入札説明会開催	8 月中旬	事務局，会計職員	
	技術審査委員の委嘱	9 月中旬	会計職員	技術審査委員の委嘱
9/28	入札書類締切	9 月中旬	事務局	
	技術審査 業者への照会，追加資料の要請	9 月下旬	事務局 技術審査委員会	仕様書，総合評価基準，性能評価基準に基づく評価 技術審査報告書の作成
		9 月下旬	CFD ｾﾝﾀｰ内検討ﾁｰﾑ	空調機更新工事仕様検討
10/9	開札	10 月上旬	事務局，会計職員	落札者決定，入札結果一覧表作成
	契約	10 月上旬	事務局，会計職員	
	落札者の公示手続き開始	10 月上旬	事務局	落札者の公示起案
11/1	落札者等の公示（官報）	11 月上旬	会計職員	

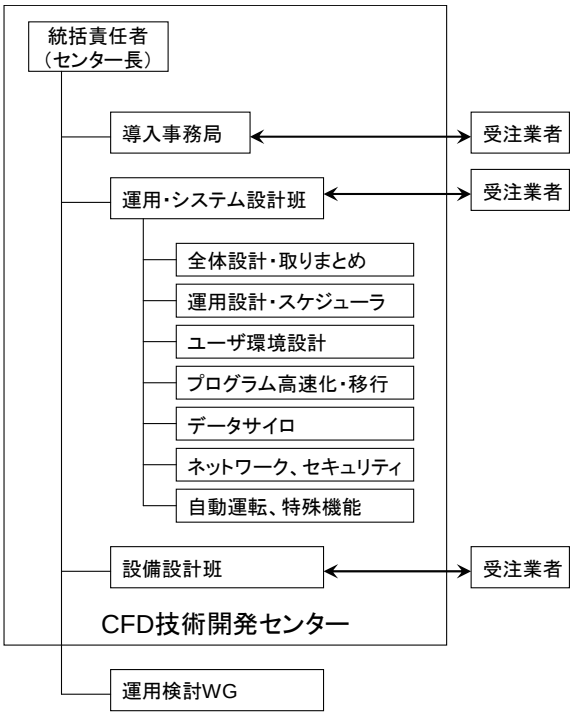


図 2.14　NS-III の導入体制

2.6　NS-III の導入とシステム設計

技術審査委員会（10/3）の後，2001 年 10 月 9 日に落札者（富士通）が決まり，導入フェーズに入った．**図 2.14** に導入体制を示した．このような細かい体制をとった理由は，新システムは設定パラメータが多く，新たな利用形態も想定されているので，管理者，ユーザの意見を十分取り入れる形で設計して行く必要があると考えたためである．

この時点で可視化システムは運用を開始していたことに注意する．従って，この後の導入の記述は主にポスト NWT システムについてのものである．本運用開始（2002.10.1）までのマイルストーンとしては，**図 2.15** のように進んだ．線表の形で示すと**図 2.17** のようになる．当初の予定通りの移行期間を設定した．

2001 年	10 月中	設計班の立ち上げ
	12 月まで	概念設計
2002 年	3 月まで	詳細設計
	1 月	旧システム一部撤去
	2 月	設備工事
	2−3 月	移行システムの搬入・設置
	3 月	移行システムの構築
	3 月末	移行システムの検収
	4 月 1 日	移行システムの仮運用
	5 月 1 日	移行システムの本運用
	6 月末	旧システム運用停止
	7 月	旧システムの残部撤去
	5−8 月	移行作業
	7−9 月	空調設備工事
	9 月	本システムの搬入・設置
	9 月	本システムの構築
	9 月末	本システムの検収
	10 月 1 日	本システム仮運用

図 2.15　本運用までの作業マイルストーン

移行運用までの作業経緯を**図 2.16** に示した．紙面の都合で，ここには書かないが，移行システム運用後も，本システム運用に向けて**図 2.17** に示した項目程度の作業はあった．システムパラメータの設定に時間がかかり，何回も打ち合わせを行わなければならなかった．

1）運用・システム設計会議

2001/10/18	運用・システム設計班の立ち上げ，システム構成概要
10/22	パーティション分割について，データサイロ構築について
10/29	セキュリティについて，システム構成概要（その 2）
11/8	ネットワーク運用概念について
11/12	導入スケジュール，ユーザ利用環境について
11/19	データサイロストレージシステム，セキュアノードについて
11/22	ジョブ運用概念について
12/3	ジョブ運用について
12/5	NSJS 仕様について
12/12	ジョブ運用概念について
12/18	概念設計書報告会

2）システム導入 WG

10/16	第 1 回：体制について，プログラム移行，概念設計，構成検討
10/25	次期 NWT ﾈｯﾄﾜｰｸ構成概要確認，次期 NWT ｾｷｭﾘﾃｨ方針検討
11/9	第 2 回：ユーザ利用環境，スケジュール，概念設計素案
11/20	第 3 回：セキュアノード利用，データサイロ
11/26	第 4 回：周辺装置自動化，計算機資源の分割及び使用用途
12/3	第 5 回：ジョブ運用，外部接続
12/7	概念設計内容ヒアリング

3）詳細設計ヒアリング

2002/1/17	第 1 回：詳細設計全般
1/24	詳細設計個別ｱｲﾃﾑﾋｱﾘﾝｸﾞ（ジョブ関連の制限に関する内容）
1/24	詳細設計個別アイテムヒアリング（サイロについて）
1/30	第 2 回：詳細設計全般
2/6	第 3 回：詳細設計全般
2/8	詳細設計個別ｱｲﾃﾑﾋｱﾘﾝｸﾞ（ﾈｯﾄﾜｰｸ，ｾｷｭﾘﾃｨについて）
2/26	詳細設計ヒアリング・レビュー
2/26	詳細設計個別アイテムヒアリング（言語環境）

4）個別案件打ち合わせ

1/24	HSM 運用

3）移行 WG

12/26	第 1 回（プログラム移行方法）
1/10	第 2 回（ドキュメント，スケジュール）
1/16	第 3 回（SMP 実行結果報告）
1/30	第 4 回（不具合調査，XPF，手引き書）
2/8	センタマシン・データ移行打ち合せ
2/13	第 5 回（スケジュール確認，XPF）
2/13	XPF 説明会
2/27	第 6 回（移行計画書案，利用環境）
3/7	第 7 回（移行計画書）

図 2.16　移行運用までの作業経緯

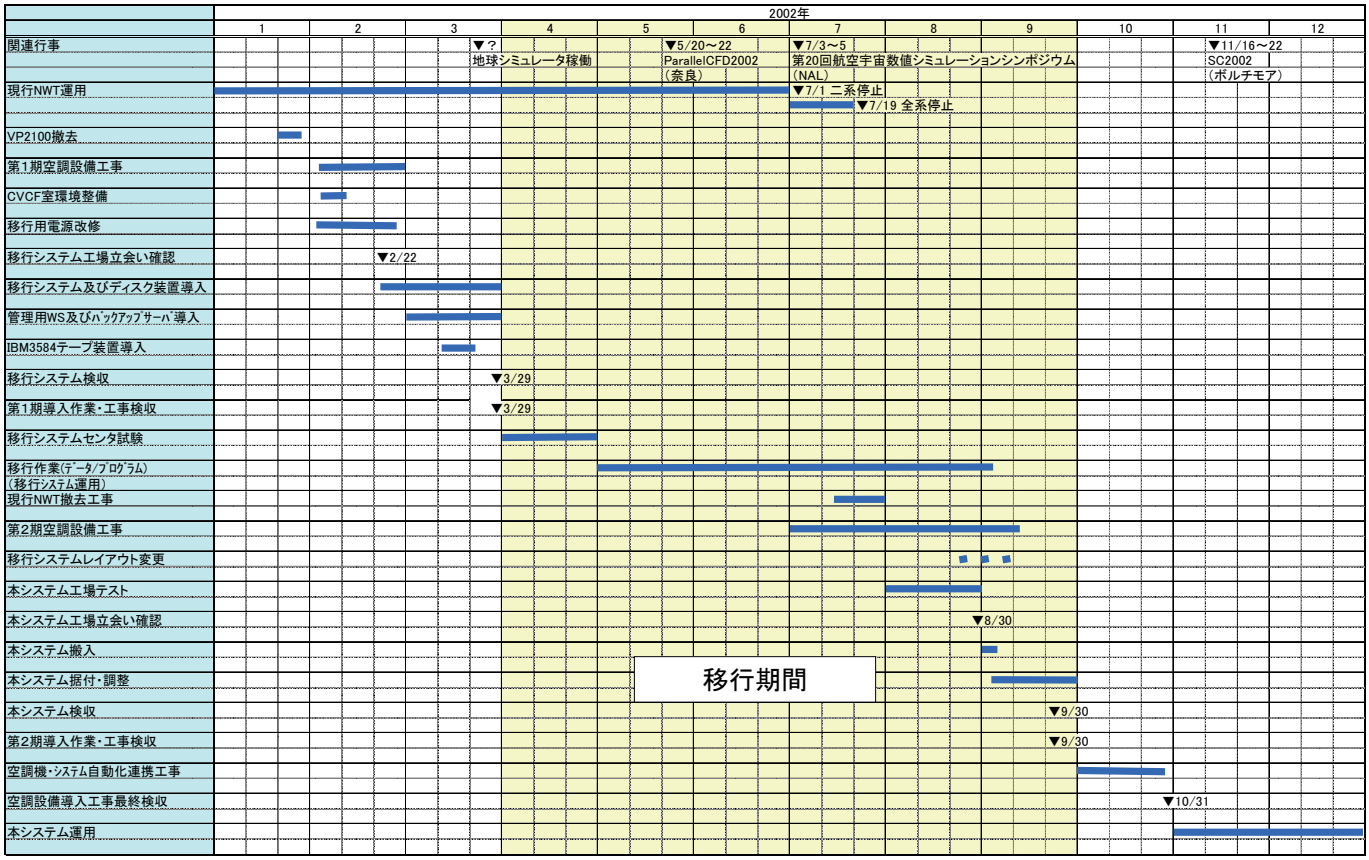


図 2.17　本運用までのスケジュール

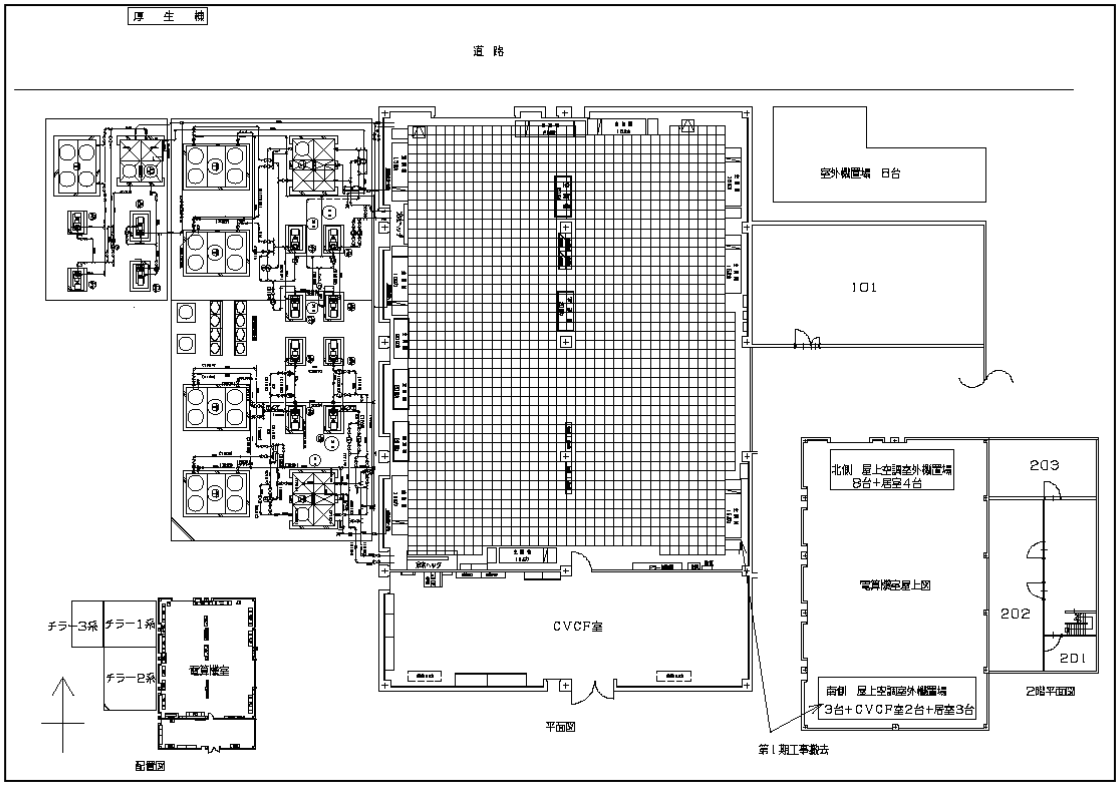
2.7　設備更新工事

NS-II の中核マシン NWT は水冷式(**図 2.18**)であったが，水冷式は，外部チラーが必要なこと，維持保守に手間とコストがかかるなど設備に関する運用サイドの負担が大きいのが問題であった．NWT 以外は空冷式であり，計算機室自体は空冷で冷却していたが，一部の空調器は購入後 15 年以上経過し老朽化が進んでいた．一方，NS-III の機器は全て空冷なため，冷却設備を空冷式に変更する必要がある．また，既存建屋への収容するためには，機器やケーブル類はコンパクトに収納する必要があり，さらには，冷却効率が下がらないように気流を考えた機材配置に配慮する必要がある．このような要請から，空調工事（**図 2.19**）及び電源工事（**図 2.21**）などを計画的に実施し，設備類を更新することとした．

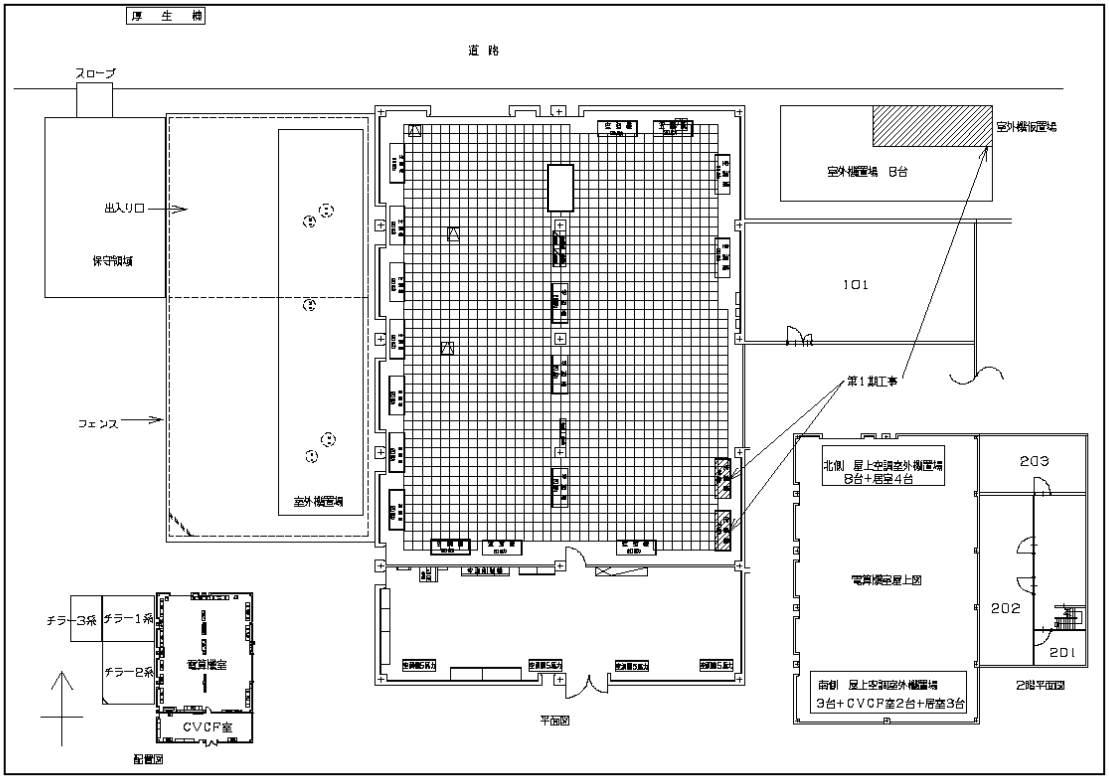
図 2.17 に示したように，空調器の更新工事は移行期間中に行うこととした．**図 2.20** に，設備工事前後の計算機建屋（**計算科学 3 号館**）の平面図を示す．図(a)の左側（西側）の部分が全てチラーと呼ばれる水冷のための室外機器である．これが，図(b)にあるように，工事後は面積では半分以下になっているのがわかる．また，空調器が計算機室の西面に集中し，室の東側の冷却効率に問題があったので，東側にも十分な冷気が行くように空調器を再配置した．**図 2.22** に，設備工事後の室外機の様子を示した．**図 2.18** に比べると非常にすっきりしたのがわかる．騒音レベル的にもそれとわかるほど低下した．

図 2.18　NWT 時代の屋外水冷設備

図 2.19　冷却設備工事の様子



(a) 設備工事前



(b) 設備工事後

図 2.20 計算機建屋の平面図

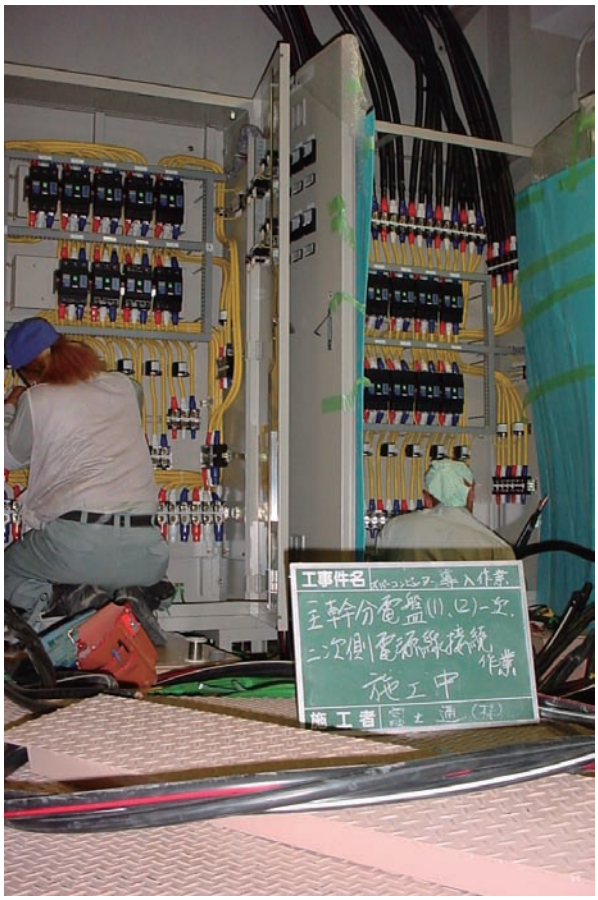


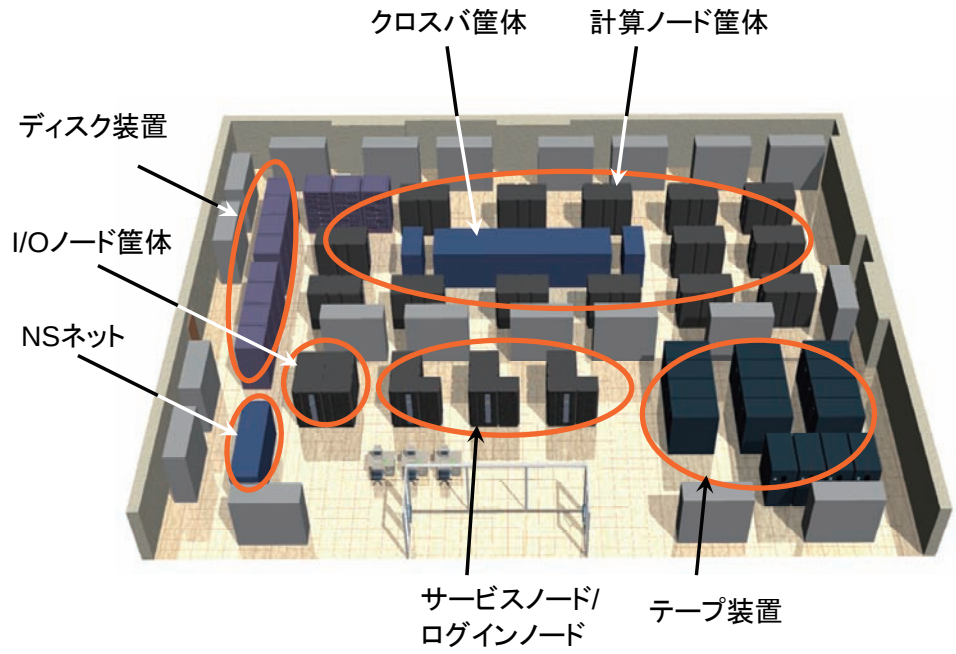
図 2.21　電気設備工事の様子

図 2.22　空調設備の導入の様子

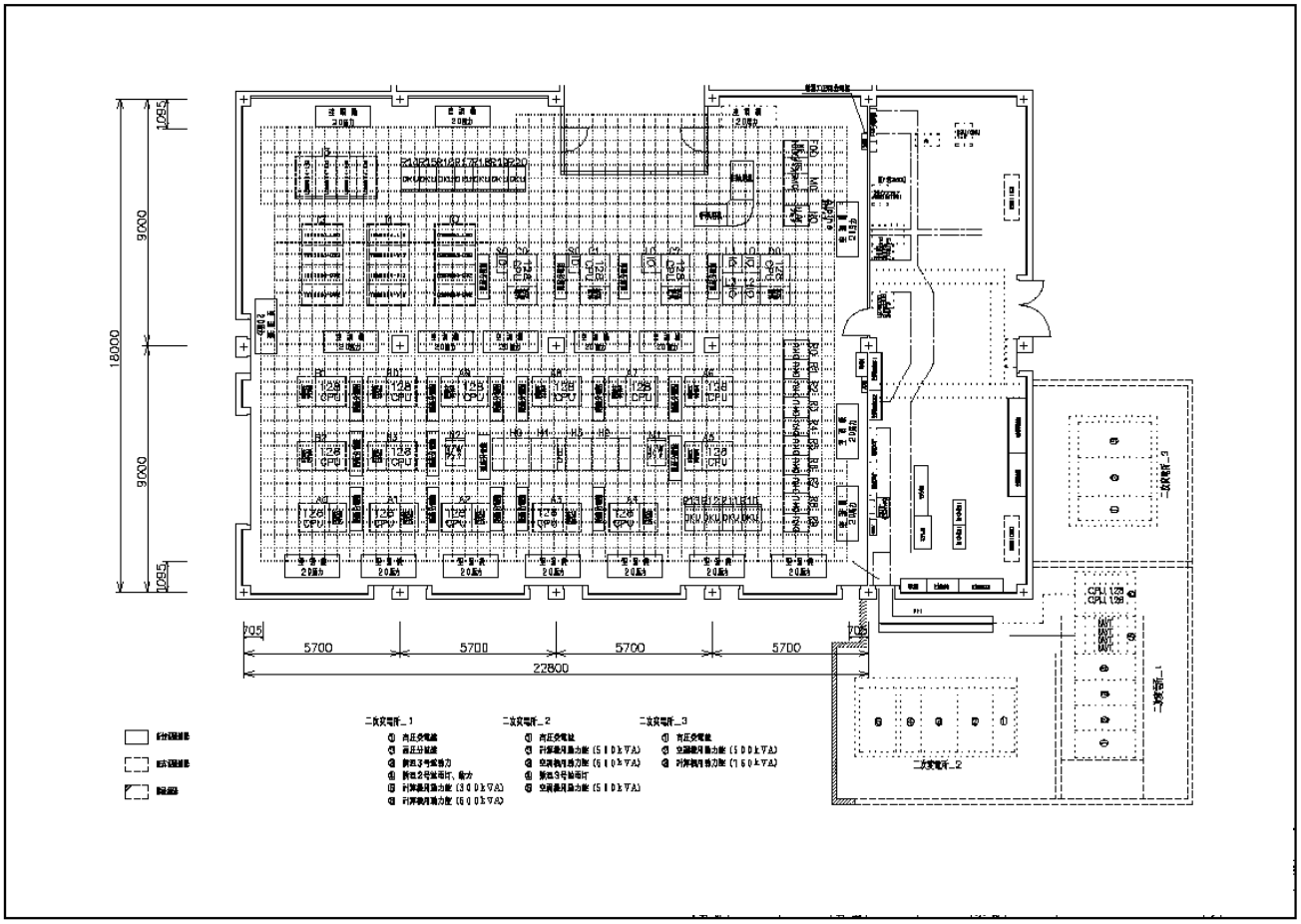
2.7　ポスト NWT の搬入と設置

ポスト NWT 本システム及び周辺装置（2002 年 9 月）の搬入・設置の様子を**図 2.23** に示す．計算機室における最終的な NS-III の機器の設置状況（可視化システムを除く）を**図 2.24, 図 2.25** に示す．設置面積は，400m² (約 20m 四方) となっている．入口から手前部分に熱の発生が少ない機器（ログイン/サービスノード，テープ装置，ネットワーク機器など）を置き，奥の部分に，熱の発生と配線を考慮してクロスバ筐体と計算ノード筐体を置く，という配置としている．冷却方式は，床下吹き出し式とし，パッケージ型の空調器 19 台を，四方の壁と中央の柱に沿って並べ，気流を中央部分に均等に吹き出すようにして，暖気の停留をなくしている．

図 2.23　ポスト NWT 搬入時の様子



(a) 鳥瞰図



(b) 設置図面

図 2.24 ポスト NWT の設置状況

図 2.25　調布航空宇宙センターにおける NS-III の設置場所

2.8　おわりに

本章では，前システムの中核計算機 NWT の特性と課題について振り返った後，新システム NS-III に付随する構想に言及し，そこから提起されるシステム要件，構成イメージを述べた．また，システム調達及び導入までの経緯について概観した．

空調設備の更新工事があったのでそうせざるを得なったとはいえ，「移行期間」を確実に設定するという方法は，この NS-III の更新において初めて採用したものである．やはり，ある程度以上の規模のシステムの場合，「移行」という考え方は必要であるし有効であると思われる．ただし，「移行期間」中は，計算機資源が減少する，運用サイドに負担がかかるなどのデメリットもあることに注意したい．さらに，いったん移行期間を設定したからには，移行期間後の本運用はきちっと立ち上げないと意味がないことにも留意すべきである．その点で，ユーザから大きな不満が出ない，運用側にも過度の負担がかからない「移行」として，どのようなサービス・作業内容が適当か，必要かつ十分な期間を如何に合理的に設定するか等は議論のあるところである．

参考文献

[1] 中村孝：「数値風洞」8 年の運用の軌跡，航技研特別資料 SP-57，2003，pp.9-13.

[2] 数値風洞報告集，航空宇宙技術研究所 CFD 技術開発センター，2002 年.

[3] 永安正彦：航技研の数値シミュレーション技術に関する今後の取り組み，航技研特別資料 SP-46，2000，pp.7-10.

[4] 松尾裕一：航技研次期数値風洞システムの構想，航技研特別資料 SP-46，2000，pp69-72.

第 3 章 NS-III のシステム概要

3.1 はじめに

本章では，JAXA の共用計算機システム「数値シミュレータ III（NS-III）」のシステム概要について述べる．ハードウェアとソフトウェアの観点からシステム構成・機能の概要を述べ，周辺のネットワーク環境や基本的な利用法について論ずる．さらに，設備管理システムについても言及する．

3.2 NS-III のシステム概要

3.2.1 ハードウェア

NS-III は，政府調達にかけた後，導入期間・移行期間を経て 2002 年 10 月より全面稼動を開始した[6]．その全体構成を**図 3.1** に示した．ハードウェアとしては，計算システム（Computing system），ストレージシステム（Mass Storage System），可視化システム（Visualization System），ネットワークシステム（Network System）の 4 つのサブシステムから成る[1][2].

NS-III の中核となる計算システムは，**中央 NS システム（Central Numerical Simulation System; CeNSS）**と呼ばれ，128 個のスカラーCPU が搭載された筐体（**富士通製 PRIMEPOWER HPC2500** [3]）18 台が単段クロスバでネットワーク結合されたものである（**図 3.2**）. 筐体は, 最大 256GB のメモリを共有する 128 ウエイの SMP を構成することができる．近年の大規模システムは，ベクトル，スカラーを問わず多数のプロセッサから成るが，メモリ配置形態により，集中共有メモリ型，分散共有メモリ型，分散メモリ型に分類される．集中共有メモリ型で，特にプロセッサからメモリのアクセス遅延がプロセッサによらず一定である構成を **SMP**（Symmetric MultiProcessor）と呼ぶ[7]．そのような構成の計算機自体を SMP と呼ぶこともある. SMP が高速結合ネットワークで結合されたものが **SMP クラスタ**であり，CeNSS はその形態を有する（**図 3.3**）．CeNSS では，メモリ競合等を低減する意味で，筐体を(64 ウエイ SMP)×2 または(32 ウエイ SMP)×4 に分割して運用している．それぞれの SMP は独立の OS を有し，**ノード計算機**または単に**ノード**と呼ばれる．18 筐体のうちの 14 筐体は，32CPU ずつ 4 つの SMP ノードに分割し，全部で 56 ノードを計算用（**計算ノード**または**演算ノード**）に割り当てている．残りの 4 筐体のうち 3 筐体は，それぞれ 64CPU の SMP に分割し，全部で 6 ノードあるうち，4 ノードは，**サービスノード**として，コンパイル，小規模処理，アプリケーション実行などの様々な処理を担当させ，残りの 2 ノードは，**ログインノード**として，ユーザがシステムを利用するための入り口処理を担当させている．最後の 1 筐体は，128way SMP として，I/O 処理やファイル処理専用ノード（**I/O ノード**）に割り当てている．

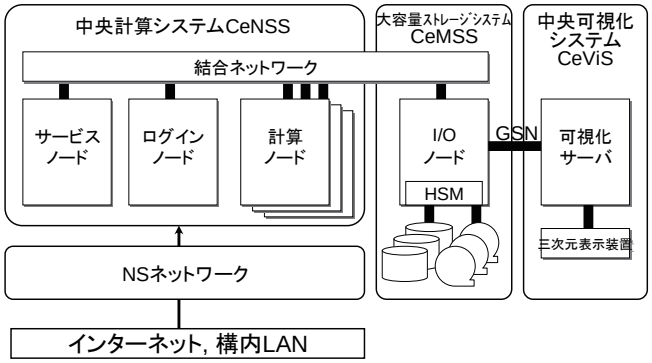


図 3.1 数値シミュレータ III のハードウェア構成

図 3.2 CeNSS の計算ノード外観

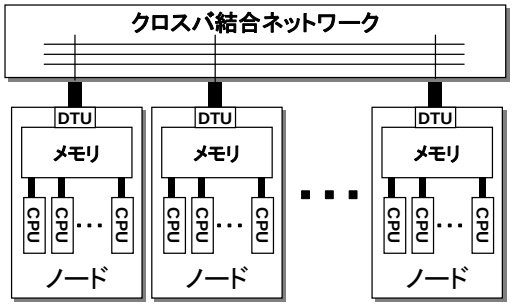


図 3.3 中央 NS システム（CeNSS）の構成

表 3.4 CeNSS の構成諸元

総演算処理性能	9.3TFLOPS
全メモリ量	3.6TB
計算ノード数	56
計算ノード内 CPU 数	32
計算ノード内メモリ量	64GB
計算ノード構成	SMP
メモリバンド幅	1.04GB/s（CPU あたり）
B/F 比	0.2
計算用 CPU 総数	1,792
CPU タイプ	SPARC64 V
CPU ピーク性能	5.2GFLOPS
L2 キャッシュ	2MB オンチップ
結合ネットワークトポロジ	クロスバ
結合ネットワーク性能	4GB/s×2
サービスノード数（CPU 数）	4（64CPU）
ログインノード数（CPU 数）	2（64CPU）
IO ノード数	1（128CPU）

[6] 可視化システムは，2001 年 2 月より稼動.
[7] 詳細は付録 D 参照.

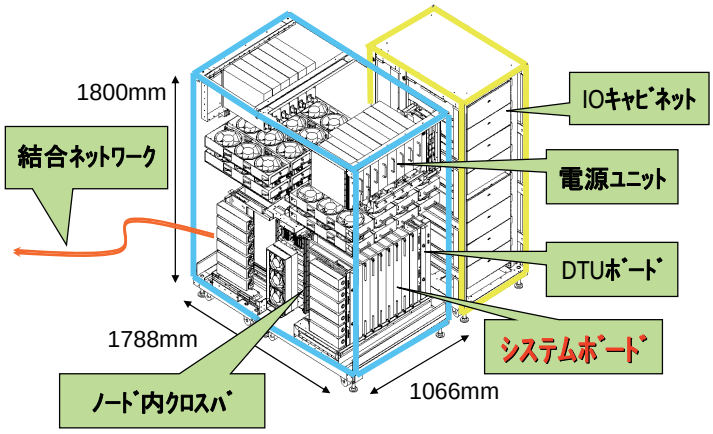


図 3.5　計算ノード筐体内の構造

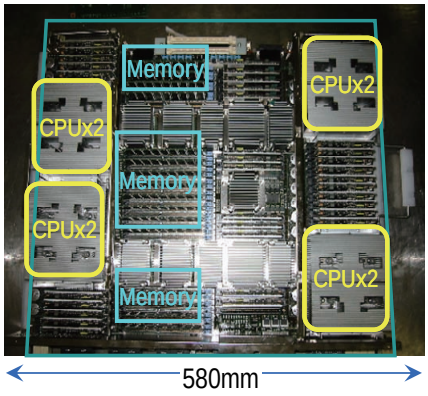


図 3.6　システムボード

CPU（**SPARC64 V** [4]）は，クロック 1.3GHz，オンチップ 2MB の L2 キャッシュ，プリフェッチ，out-of-order 実行，浮動小数点演算の 4 令同時実行等の最新スカラー高速化技術が取り入れられており，5.2GFLOPS の単体性能（ピーク性能）を有する．また，メモリバンド幅は，筐体あたりトータルで 133GB/s（1CPU あたり 1.04GB/ｓ）であり，B/F 比[8]に直すと 0.2 となる．SMP なので，すべての CPU が使われた場合の最小値であることに注意する．計算ノード（32CPU）の処理性能は 166.4Gflop/s，メモリ容量は共有 64GB，大規模 SMP クラスターシステムの全体としては，9.3Tflop/s の計算処理性能，3.6TB のメモリを有する．**表 3.4** に CeNSS の諸元を示した．

[8] Byte per FLOPS，すなわち演算性能に対するメモリ性能の比のこと．

計算ノード筐体は，**図 3.5** のような内部構造となっており，筐体中には**図 3.6** に示したシステムボード（SB）と呼ばれる 60cm 角のボードが 16 枚格納され，筐体内クロスバで結合されている．システムボード上には 8 個の CPU が搭載されている．

各ノードは，**DTU（Data Transfer Unit**；数は各ノード 1 個，I/O ノードのみ 2 個）と呼ばれるインタフェース機構を通じて単段クロスバ・結合ネットワークに結合されている．クロスバは往復それぞれ 4GB/s の転送性能（レイテンシは〇マイクロ秒）を有する．CPU からみた結合ネットワークの B/F 比は 0.77，計算ノード単体からみた B/F 比は 0.02 である．SMP なので，実際の値はこの中間的なものであると思われる．**図 3.7** は，ノードに対するクロスバの状態を示したものである．ここに，◆はスイッチをあらわす．**図 3.8** にクロスバ筐体の外観，**図 3.9** に床下内の配線状況を示す．光ケーブルを利用しているので非常にコンパクトな収納となっている．また，後述するようにノード間ハードウェアバリア機構を有する．**図 3.10** に SB や DTU の配置を反映させた構成の詳細を示す．

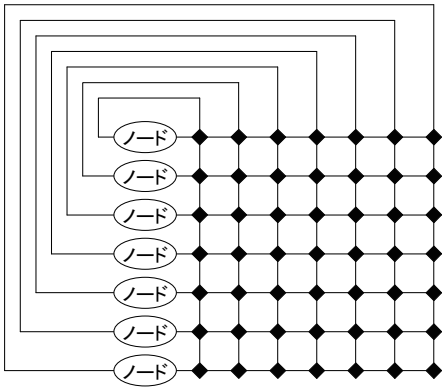


図 3.7　クロスバ・結合ネットワーク

図 3.8　CeNSS のクロスバ筐体

図 3.9　CeNSS の床下内の結合ネットワークケーブル

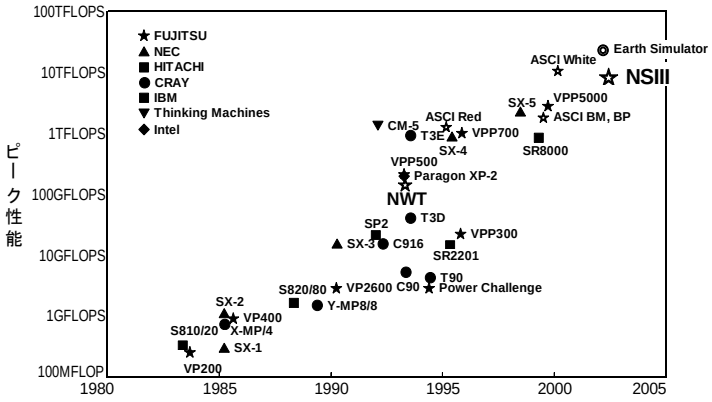


図 3.11　高性能計算機のピーク性能の発展

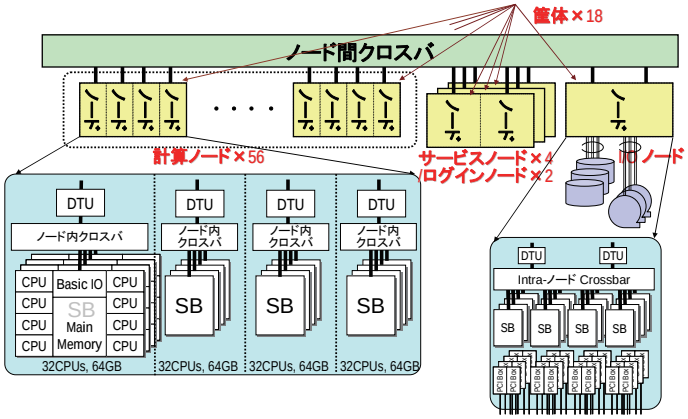


図 3.10　CeNSS の論理構成

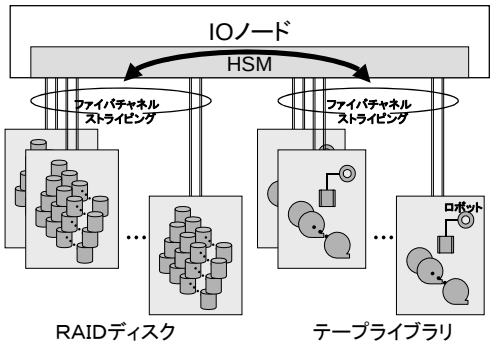


図 3.12　中央マスストレージシステム（CeMSS）の構成

図 3.11 は，高速計算機のピーク性能の発展経緯を示したものである．導入時点での世界最高速スパコンは，地球シミュレータであったが，NS-III は世界第 7 位の LINPACK 性能を有した（2003 年 6 月，第 4 章参照）．

数値シミュレータ III により，航技研は 10TFLOPS 近い処理性能を獲得し，本格的なテラフロップスの計算パワーを駆使する時代へと突入した．ここで，第 1 世代の数値シミュレータ NS-I の性能を思い出していただきたい．1987 年のスタート時の性能はちょうど 1GFLOPS であった．それから 15 年経過した 2002 年，数値シミュレータの性能は約 10TFLOPS となり，この間ピーク性能で約 10,000 倍の性能向上があった．いわゆるムーアの法則による性能向上は 5 年で 10 倍，15 年で $10^3 = 1000$ 倍なので，数値シミュレータはムーアの法則の 10 倍の性能向上を達成したことになる．そのような性能向上を達成できた理由は，並列計算機とする（CPU をたくさん並べる）ことで，物理的な性能向上以上の性能を取得可能となったこと，性能単価が低下したことなどに因る．それでは，小さな計算機をたくさん並べればよいかというと，そう単純にはいかないところに計算機構築の難しいところがある．（**付録 D** 参照．）

次に，ストレージシステム[5]は，**中央マスストレージシステム（Central Mass Storage System; CeMSS）**と呼ばれている．CeMSS は，物理容量 57TB の RAID5 ディスクアレイ装置（**富士通製 PW-D500B1**, ディスク玉は 72GB, 9D+1P 構成）が 80 本のファイバチャネル（1 本あたり 100MB/s の転送性能）で 128 ウエイ SMP 構成の I/O ノードに接続されている．一方，テープライブラリ装置（**IBM 製 IBM3584**）は，40 台のドライブを有し，40 本のファイバチャネルで I/O ノードに接続され，総容量 620TB を有する．テープ媒体として第一世代 LTO（**LTO Ultrium 1**, データ非圧縮時 100GB/媒体）を採用し，ドライブあたり 15MB/s のピーク転送速度を有する．I/O ノードとディスク間の 1GB/s の実効バンド幅を実現するために，ベンチマークを行った結果（後述）として，16 本のファイバチャネルをストライピングしている．

ディスクとテープの連携については，**階層型ストレージ管理（Hierarchical Storage Management; HSM**，ソフトウェアはサン・マイクロシステムズ製の **SAM-QFS**，**図 3.12**）を採用し，ユーザからはディスクとテープの区別なく利用できるようにしている．テープは単なるバックアップ装置ではなく，ストレージの一部と考え，使う頻度が少ないデータはテープに退避させている．テープの転送速度はディスクに比べ

ると比較にならないので，信頼性を考えて 4 ストライプ＝ピーク転送速度 60MB/s（非圧縮）で運用することとした．また，データの保全性を考えてダブルコピーを取っている．大容量ストレージを各ノードに接続する（並列 I/O）のではなく，単一の I/O ノードに接続するようにしたのは，並列 I/O をしなくても転送速度を十分にとれる見込みがあったこと，システム外からファイルをみたとき単一ファイルに見えるようにしたかったこと等に因る．ただし，このような形態にすると I/O ノードに負担がかかるので，プロセスの配置やシステムの信頼性向上への配慮をしている．**図 3.13** は，ディスク装置及びテープ装置の外観を示した．

図 3.13　CeMSS におけるディスクとテープ装置

可視化システムは，**中央可視化システム（Central Visualization System; CeViS）**と呼ばれている（**図 3.14**）[9]．このシステムの導入は 2001 年に行われた[6]．可視化サーバ（**SGI 製 Onyx 3400**，**図 3.15**）は，32 個の CPU，64GB の共有メモリ，1.5TB のディスク容量を有し，4.6m×1.5m の画面を持つ大型 3 次元表示装置 Aerovision（**図 3.16**）やグラフィックス端末から成る．Aerovision は，通常の CRT の 3 倍の解像度とステレオ表示などの機能を有するとともに，AVS や EnSight といった市販の可視化ソフトを利用できる．

CeNSS の大規模計算データを中央マスストレージ装置から直接高速に読み込めるように，GSN（Gigabyte System Network）リンクを用いて I/O ノードと実効 500MB/s の転送速度より接続（GSNLink）している．GSN は，HiPPI6400 規格[10]の高速チャネルであり，1 本あたりピーク転送性能 800MB/s を有する．ギガイーサ系に比べ CPU 負荷が低いということから採用されたものであった．中央可視化システムの詳細は，**付録 F** を参照されたい．

図 3.14　中央可視化システムの構成

図 3.15　CeViS における可視化サーバ

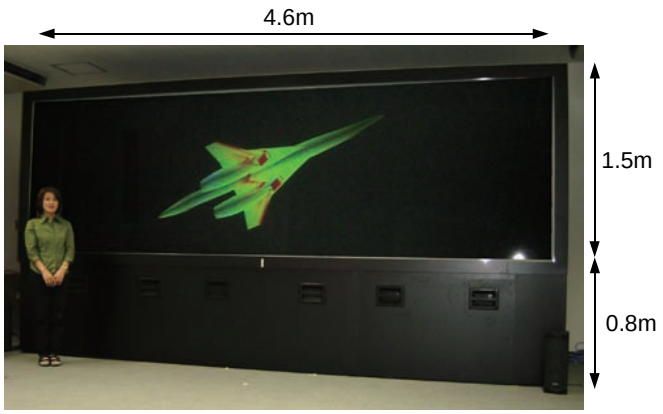


図 3.16　パワーウオール型大画面表示装置 Aerovision

[9] 2007 年 10 月の時点で，可視化サーバは，8CPU，16GB メモリを有する SGI 製 Prisum に，GSN リンクは，ギガイーサ接続に変更された．

[10] HiPPI は，スパコンのネットワークに主に用いられている 800Mbps の IO インターフェースの ANSI 標準規格．

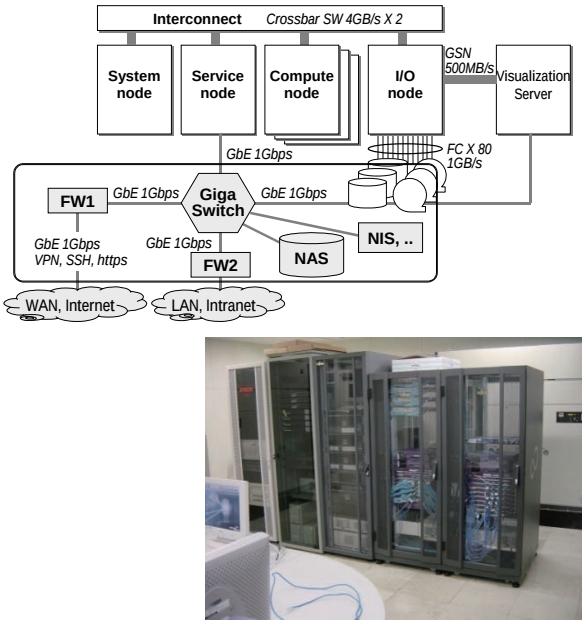


図 3.17 NS ネット（連携ネット）の構成

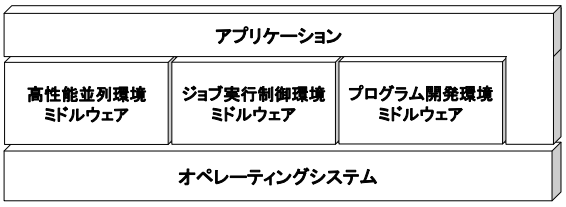


図 3.18 NS- III のソフトウェア構成

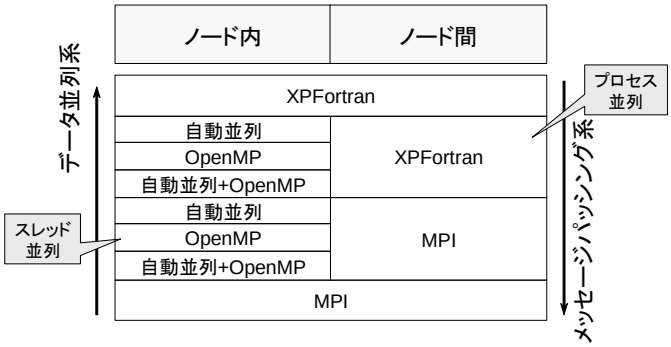


図 3.19 NS-III の並列プログラミング体系

ネットワークシステムは，NS ネット（NS System Network; NSnet) あるいは**連携ネット**と呼ばれる（**図 3.17**）．ギガビットスイッチを中心に基本的にギガビット・イーサネット回線で接続されている．運用性と保全性を担保するためにユーザホーム領域はネットワークディスク NAS とし，NIS サーバ[11]等のサーバ類も別立てとしている．図には示していないが，運用ネットワーク（冗長構成）を別に設けて不測の事態への対応を行うことにした．

3.2.2 ソフトウェア[7]

NS-III 全体としての OS は，主要な部分は，信頼できる 64 ビット UNIX に統一した．ここでは特に，CeNSS のソフトウェアについて述べる．**図 3.18** に，CeNSS のソフトウェア全体構成を示す．OS の上に 3 つの機能別のミドルウェアが存在し，その上にアプリケーションが載るという階層構造をなしている．

オペレーティングシステム（OS）には，**業界標準の 64 ビット UNIX OS** の一つである **Solaris 8** を採用した．64 ビット UNIX なので，大容量のメモリ及び 2GB 以上のファイルを扱える．また，OS としての高セキュリティ機能及び各種ネットワーク機能を有する．また，各ノードが OS を持っているため，どれかのノードがダウンしても全システム停止に至ることはなく，障害に強いシステムを構築している．

高性能並列環境ミドルウェアは，大規模並列計算を実行するためのラージページと呼ばれるメモリを効率的に利用する環境や多数のプロセス資源をスケジューリングする環境を提供する．また，**SRFS（Shared Rapid File System）**と呼ばれる高速ネットワークファイルシステムを提供する．これにより，どのノードからも同一のファイルシステムを参照可能である．また，可視化システムとの高速インターフェースを提供する．ジョブ実行制御環境ミドルウェアは，各種のジョブを効率的に実行する環境を提供する．NS-III では，通常のバッチジョブクラスの他に，インタラクティブジョブとして，パラメータを対話的に入力可能なジョブクラスや，可視化ジョブとして，メモリ to メモリで可視化システムにデータを送りリアルタイム可視化を実現するジョブクラスを順次提供して行く予定である．ジョブ実行のキューイングシステムは NQS を基本とするが，航技研独自開発のスケジューラを介して小さなジョブから大規模ジョブまで空いている CPU を極力少なくし計算リソースを有効活用する環境を構築している．

プログラム開発環境ミドルウェアは，コンパイラやツールを提供する．コンパイラとしては，Fortran，C，C++を提供する．CeNSS では，Fortran 並列プログラミングの標準形として，ノード内ではスレッド並列，ノード間ではプロセス並列というハイブリッド並列プログラミングモデルを採用している．**図 3.19** に，CeNSS の並列プログラミング体系を示す．スレッド並列は，共有メモリ計算機特有の並列化手法であり，特長は並列化のためのオーバーヘッドが小さいことが挙げられる．実際には，図にあるようにスレッド並列には，**自動並列**または **OpenMP**（両者混在可能）を，プロセス並列には **MPI** または **XPFortran**（混在は不可）を使用する．このうち，XPFortran は NWT-Fortran と互換の並列化言語であり，NWT-Fortran で書かれた並列プログラムは，NS-III では再コンパイルのみで使用できる（**付録 D** 参照）．

図 3.20 は，流体解析コードでよく現れる 3 重 DO ループを例に，NS-II から NS-III へのプログラミング・スタイルの典型的なマッピングを示す．NS-II（NWT）では，最外側ループが並列化され，内側ループがベクトル化されるというパターンであったのが，NS-III（CeNSS）では，最外側は同様

[11] 2004 年 5 月に LDAP サーバ方式に移行した．

に並列化されるのは変わらないまま，内側がスレッド並列化される点だけが異なる．自動並列は，実行時に環境変数で指定するので，ソースの変更は基本的には生じず，ベクトルコードを自然にスカラーシステムにかけることが可能となる．

CeNSS では，Parallelnavi Workbench と呼ばれる統合的なソフトウェア開発環境を提供した．グラフィカルエディタ，デバッガ，プロファイラ，ソース解析，などのツールを利用できる．また，数学ライブラリとして，SSL II，C-SSL II，BLAS， LAPACK，ScaLAPACK を利用できる．

3.2.3 利用法とアプリケーション[8]

NS-III の利用方法として，今後はインターネット等のネットワークを通じてのアクセスが当たり前になって来る．そうした状況に対応し，かつ，運用や構成の自由度を増すために，**図 3.21** に示したように NS-III のセグメント NSnet をイントラネットから独立させている．

インターネットからの利用で重要になるのはセキュリティである．NS-III では，SSH サーバを導入し通信を暗号化している．**SSH（Secure Shell）**を利用するには，クライアントソフトが必要である．最初のアクセスの際にクライアントソフトのインストールと暗号通信路開設のための設定が必要であるが，以後はそのクライアント端末からはログイン認証のみで NS-III にアクセスできる．従来は，ログインの度にワンタイムパスワードによる認証が必要だったが，そのような手続きは不要である．**図 3.22** に，SSH を利用する手続きを示した．

```
:
!XOCL PARALLEL REGION                    NWT-F, MPIによる並列化
        :
!XOCL SPREAD DO /IPN
      do 1000 n = 1, nblock
      do 1 l = 1, lmax
      do 1 k = 1, kmax
v     do 1 j = 1, jmax                   自動ベクトル化
v       di       = 1./q(j,k,l,1,n)
v       u(j,k,l) = q(j,k,l,2,n)*di
v        :
v       rmu(j,k,l,n) = (cc**1.5)*c2bp/(cc+c2b)
v       turmu(j,k,l,n) = 0.
v    1 continue
 1000 continue
!XOCL END SPREAD DO
        :
!XOCL END PARALLEL REGION
        :
```

NSII

```
    :
!XOCL PARALLEL REGION                    XPFort, MPIによる並列化
        :
!XOCL SPREAD DO /IPN
      do 1000 n = 1, nblock
p     do 1 l = 1, lmax                   自動並列、OpenMP
p     do 1 k = 1, kmax                   による並列化
p     do 1 j = 1, jmax
p       di       = 1./q(j,k,l,1,n)
p       u(j,k,l) = q(j,k,l,2,n)*di
p        :
p       rmu(j,k,l,n) = (cc**1.5)*c2bp/(cc+c2b)
p       turmu(j,k,l,n) = 0.
p    1 continue
 1000 continue
!XOCL END SPREAD DO
        :
!XOCL END PARALLEL REGION
        :
```

NSIII

図 3.20　並列プログラムの移行

また，NS-III では，ウェブブラウザを利用してジョブ操作やファイル操作，簡易可視化を行うアクセスシステム **WANS（Web Access to NS-III）**を開発した（**図 3.23，図 3.24**）．WANS では，サーバサイド Java 技術を用いてセッション管理やレスポンス保証を行っている．また，暗号化プロトコルの https を利用してセキュリティを強化している．WANS は，NS-III のポータルサイト（http://wans.jaxa.go.jp）からの利用が可能である．機能は限られるが，ウェブブラウザがあれば，どの端末からでも NS-III にアクセスできる．NS-III では，アクセスポイントの一元化を目指し，ポータルサイトから情報提供，利用申請，マニュアル参照，セッション開設などの利用に必要な全ての操作が行えるようにしている．

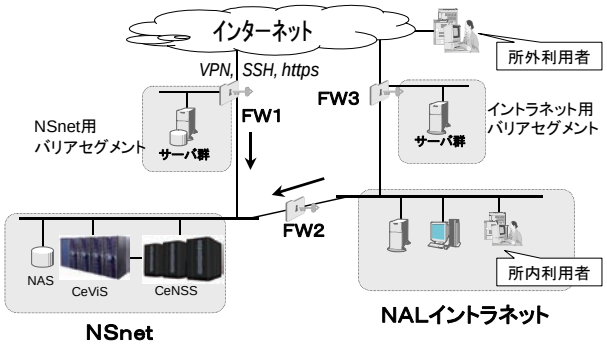


図 3.21　スパコンセグメント

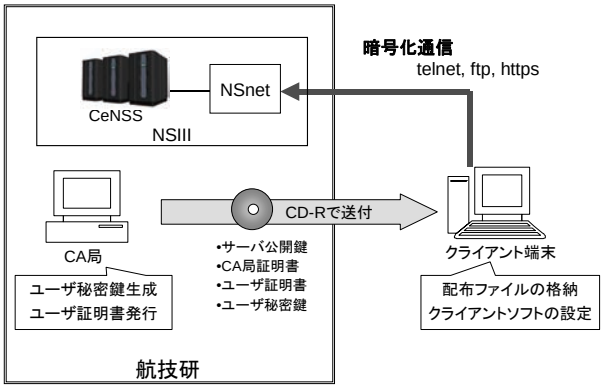


図 3.22　SSH による NS-III へのリモートアクセス

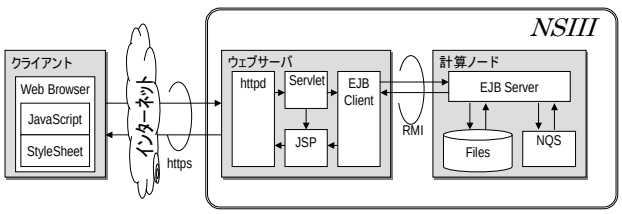


図 3.23　WANS の構成

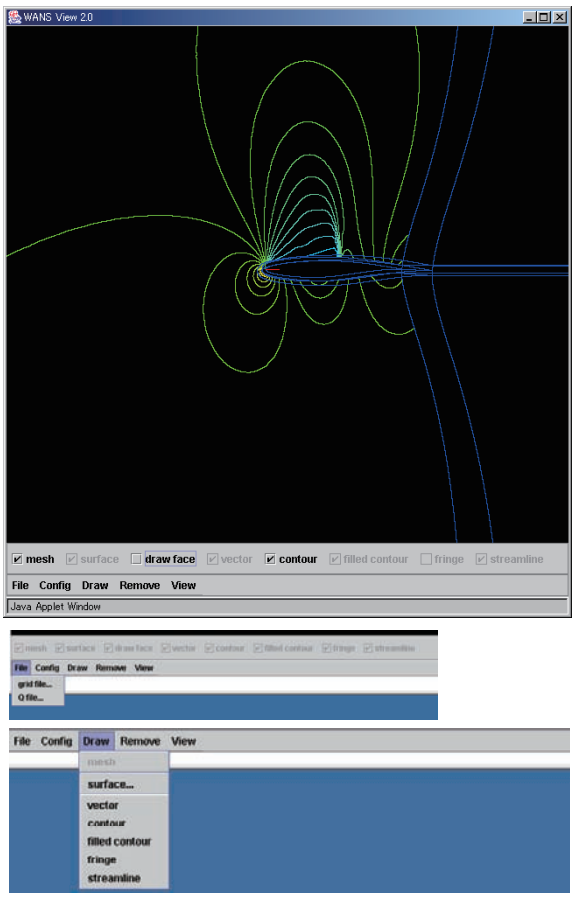


図3.24　WANS-Viewによる遠隔可視化の表示例

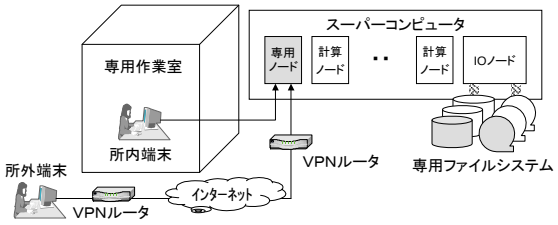


図 3.25　セキュア利用の概要

一方，もっとセキュアな利用が必要なメーカ等からのアクセスを前提として，セキュア利用と呼ばれる環境を構築した（**図 3.25**）．セキュア利用は，ジョブ，ファイルシステム等，一般利用とは切り離された状態で NS-III を利用するものである．セキュア利用のためのノード＝セキュアノードを設けている．業務の内容はシステム管理者以外に見られることはない．航技研内部から利用する際には，外部から遮断されたセキュアルームを利用することでセキュアな利用を確保できる．NS-III のアプリケーションとしては，業界標準の構造解析ソフトウェアである NASTRAN を導入した．また，OS に Solaris を採用しているので，GNU 等のオープンソースのツール類や各種画像ツールを利用できる．

3.3　NS-III の設備管理

NS-III を設備的にみれば，システムが従来より複雑かつ大規模になることに鑑み，イーサネットによる設備監視制御系ネットワークを構築し，このネットワークを用いて大規模システムの自動運転システムを構築した．この自動運転システムは，ジョブスケジューラと連動し，ジョブの有無や運転計画に応じたきめ細かな設備制御・運転制御を可能とする．たとえば，ノード毎に動いている数 10 の OS や空調器を含む機器類の動作状態の監視とブート/シャットダウンなどの自動制御や，障害が発生した場合の管理者への通知やフェイルオーバー，運用データの収集等を担当する．**図 3.26** に自動運転システムの制御画面のスナップショットを，**図 3.27** にシステムの構成概要を示した．設備側は設備管理 PC が管理し，計算機側は SMC と呼ばれるサーバが管理し，両者をイーサネット（TCP/IP プロトコル）で接続している．従来，スーパーコンピュータと空調設備のこのような連携はほとんど例がなく，今後の大規模クラスタシステム全盛の時代，あるいは省エネの時代に向けての先駆的な試みであった．導入後の問題や不便は発生せず，有効に機能した．

このようなシステムの構築により，初期コストは必要となるものの，設備面での運営の自動化と省力化（例えば空調器も含めた完全自動停止や自動立ち上げ，温度等の常時状態監視）を実現するとともに，計算機と空調器より成るシステム全体の電力消費量を前システムの約 2/3（800KW）に削減することができた．その実質的な効果は予想以上に大きかったということができる．

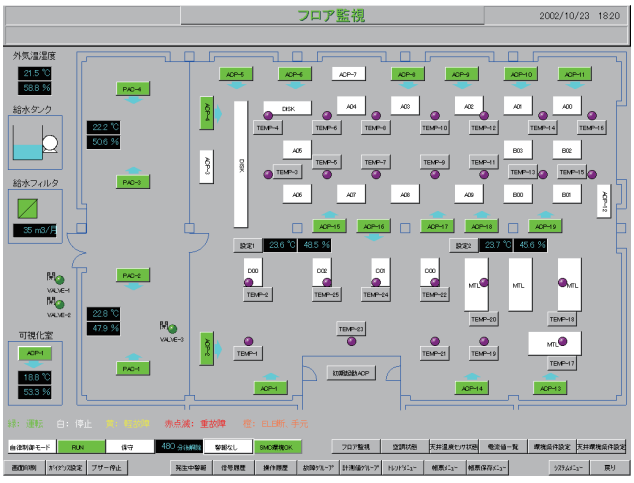


図 3.26　自動運転システムの制御画面

3.4 おわりに

本章では，NS-III のハード・ソフトに係るシステムの概要やネットワーク環境，利用法等について述べた．

NS-III の構想をキーワードとして今一度整理すれば，①CFD における技術課題克服，②標準設計解析ツールの整備，③次世代統合シミュレーションの展開等になり，NS-III は，この構想（アプリケーション）を実現可能な性能・機能を持つシステムを導入し構築したということがいえる．その帰結が SMP クラスタであったり，大規模ストレージや可視化システム，ソフトウェア開発環境や利用環境であったりする．再度強調すべきは，NS-III は，アプリケーションにドライブされた結果として，性能だけでなく機能や使い勝手を重視している点である．それがスパコン処理性能のデモンストレーターであった NWT とは明確に区別されるところであろうし，第 3 期数値シミュレータの大きな特徴でもあると思われる．

3.5 導入フェーズのまとめと課題

以上までで，NS-III の導入の背景や調達開始までの経緯，調達手続き，設置のプロセス，システム概要について述べた．ここで，NS-III の導入フェーズにおける良かった点，悪かった点，課題などを整理しておきたい．

調達手続きに入るまでの過程においては，**利用ニーズを明確化**する活動をしっかりと行い，「計算科学ビジョン 21」という形にまとめ，それをもとに構想，仕様と徐々に固めて行った．その結果として，**アプリケーション主導によるシステム設計・構築**が比較的スムーズかつ成功裏に実施できたと分析する．我々の場合，大学等と違い航空宇宙に特化しているという点で，中核アプリケーションを設定しやすいという事情はあるにせよ，**導入根拠のしっかりしたバランスの良いシステム**を構築することが将来にわたる継続的な発展という意味では重要であろうと思われる．一時期だけ性能の高いシステム，目立つシステムがあったところで，それですべての航空宇宙における技術課題が解決するわけではないので，あまりメリットがないのは明白だからである．

システムの性能・機能を決定する際，**前システムの課題克服**を重点の一つに置いた．例えば，メモリ量，ストレージ量，使い勝手などである．また，移行においては，**プログラム資産やデータの円滑な移行に配慮**した．例えば，プログラミングスタイルや並列化方針を大きく変えないこと，新旧システム間のデータ移動の運用側からのサポートなどである．これらにより，**利用側に過大な負担をかけることなく業務の連続性と発展性がうまく確保**されたのではないかと考えている．

一般にアプリケーションの寿命はシステムの寿命より長い．システムは 5 年おきに更新されるが，大半のアプリケーションは 10 年以上は使われるだろう．研究用コードは，それ自体が研究の対象なので比較的寿命は短いが，プロダクション系のコード（例えば NASTRAN など）で 20 年以上使われているものもある．研究開発機関という性格上，誰もやったことのない解析やシミュレーションに挑戦できる程度の性能向上は必要であろうが，**長期スパンで考えたときの生産性・効率のような視点**を調達においては考慮しておくことが重要と思われる．

始めてのスカラー機の導入ということもあり，実際のシステム設計やパラメータの設定等に大変な時間と労力を要した．これには，細かいところまで気配りの効いたシステム構築ができるメリットがある反面，そこまで本当に必要なのか，という疑問が常に残った．限られたコストの中で**何をどこまでやるかの線引き**と上手な線引きを可能とする経験・実績の積み上げが重要であろう．

中央可視化システムを含む周辺機器の導入の是非については常に議論の分かれるところである．このあたりの考え方に事前検討の十分さや導入根拠がしっかりしているかどうかが表れる．中央可視化システムの効果・課題は評価や運用の場面で述べることとするが，NS-III の調達では，移行期間中における CPU 資源の提供元としてうまく機能したことはここに記しておきたい．

参考文献

[1] 松尾裕一:航技研次期数値シミュレータシステム（NSIII）の概要，航技研特別資料 SP-57，2003，pp.15-21.
[2] 矢澤克己，稲荷智英：宇宙航空研究開発機構様数値シミュレータ III を支える PRIMEPOWER HPC, FUJITSU 54，6，pp.550-555，2003.
[3] 新庄直樹：新世代 HPC サーバ PRIMEPOWER HPC2500，宇宙航空研究開発機構特別資料 JAXA-SP-03-002, 2004, pp.48-52.
[4] 井上愛一郎：UNIX サーバ用プロセッサ：SPARC64 V, FUJITSU 63，6，pp.450-455，2002.
[5] 藤田直行：NSIII における大規模ストレージシステムの設計と性能，航技研特別資料 SP-57，2003，pp.22-27.
[6] 松尾裕一：航技研新中央可視化システムの概要とその戦略，航技研特別資料 SP-53，2001，pp.149-154.
[7] 高木亮治：NSIII におけるソフトウェア開発環境とユーザ利用環境，航技研特別資料 SP-57，2003，pp.28-32.
[8] 大川博文：NSIII におけるネットワークの設計と実装，航技研特別資料 SP-57，2003，pp.33-36.

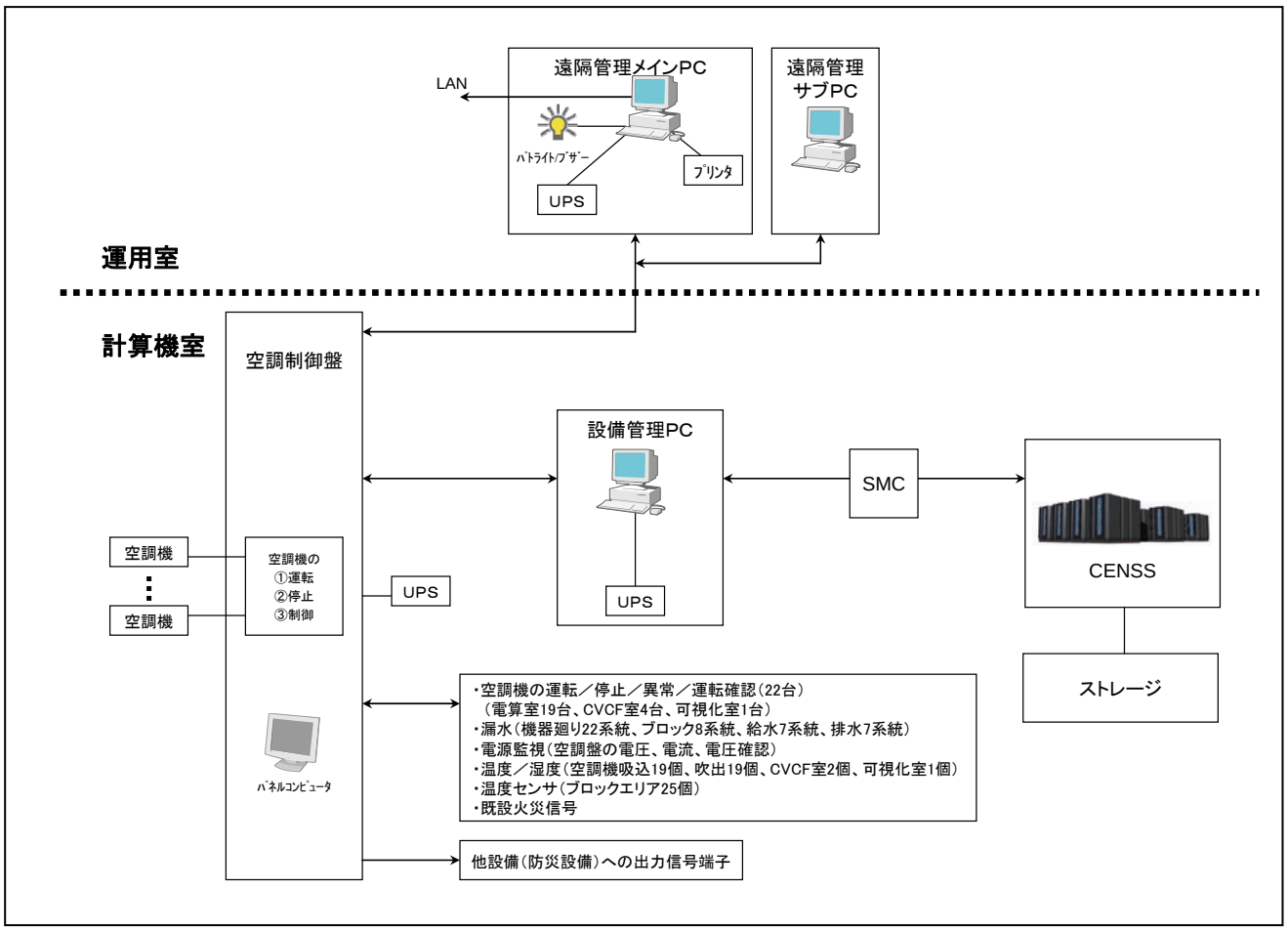


図 3.27　自動運転システムの構成概要

第 4 章　NS-III のシステム基本特性

4.1　はじめに

本章では，汎用ベンチマークやテストプログラムの性能評価結果に基づき NS-III（特に CeNSS）の性能上の基本特性を探る．また，システムの特徴や運用方針について触れる．

NS-III の中核計算エンジンたる中央計算システム CeNSS は，市場からの調達とはいえ相当な規模ゆえに，その実効性能（実測値）やシステム特性は，カタログスペック[12]だけでは推し量れない部分がある．また，当初予定していた性能が実際に達成できているかという点も，経験がないだけに実際に確かめる必要がある．次の調達に向けての客観的な基礎データにもなる．そこで，よく用いられるベンチマークプログラムや簡単な試験プログラムを用いて性能測定をできるだけ多角的に実施し，主要な特性を調べた[1][2].

4.2　CeNSS の基本演算処理性能

CPU の基礎演算性能評価として，**Euroben ベンチマーク**[13]を用いた．**図 4.1** は，DotProd(s = s + x1(i)*x2(i), i=1,n)と呼ばれるカーネル 7 の測定結果を，著名な CPU と CeNSS の CPU（SPARC64 V）とで比較（データは富士通より）したものである．横軸に問題規模，縦軸に性能値の対数を取ってある．他のカーネルの測定結果も含めて，SPARC64 V は，同時代の他の CPU と比べて，単体性能としては遜色ないものであることがわかる．（著名な CPU の値は，ホームページからの抽出（2003 年 5 月）であり実際に測定したものではないことに注意する．）なお，ここで，ベクトル計算機 VPP5000 の値も同時にプロットしている．ベクトル計算機は，問題規模が小さいうちは，ベクトル演算器の立ち上がりのオーバーヘッドのためにスカラーCPU より性能は良くないが，一定規模に達するとスカラーCPU を抜いている．これは，VPP5000 の単体性能が 9.6GFLOPS と高いことによる．ベクトルの場合は，問題規模が大きくなっても一定の性能を続けるが，スカラーCPU の場合には，データがキヤッシュから溢れると性能が低下する．縦軸は対数なので，見た目の差は小さいが半分以下になってしまう．逆にキャッシュに載っていれば性能が落ちないのがスカラーCPU の特徴でもある．

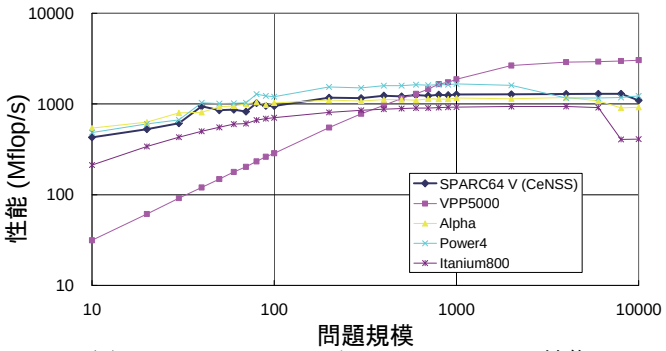


図 4.1　Euroben ベンチマーク Kernel7 の性能

[12] http://primeserver.fujitsu.com/primepower/catalog/pdf/primepowerhpc2500.pdf
[13] http://www.euroben.com

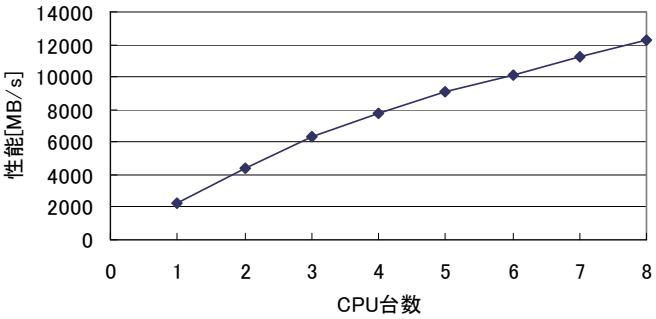


図 4.2　STREAM ベンチマーク Triad の性能

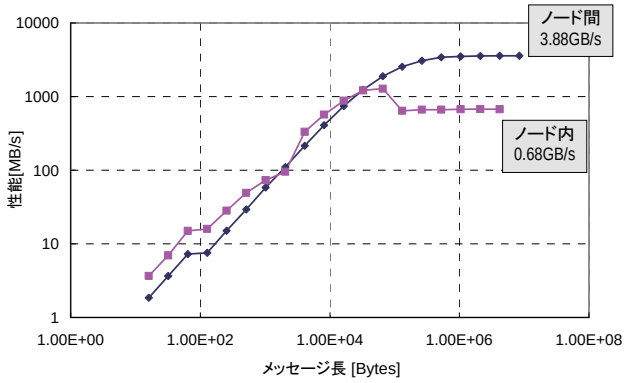
表 4.3　STREAM Triad の結果

条件	測定結果[MB/s]	性能比
単体 CPU	2,254	1
8CPU	12,290	5.42

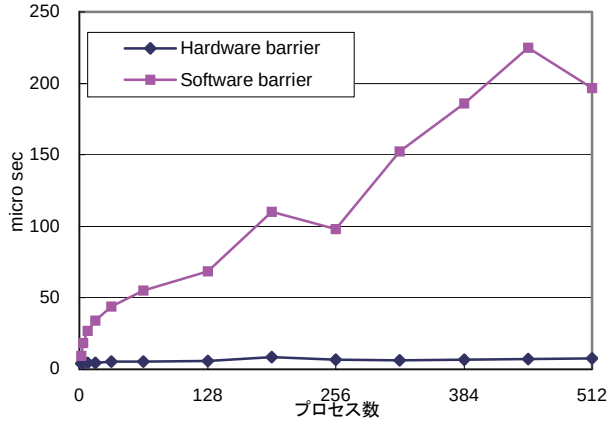
メモリ性能の測定には，良く知られた **STREAM ベンチマーク**[14]を用いた．そのうちから Triad[15]の測定結果を，**図 4.2** には CPU 台数による性能値の変化を，**表 4.3** には単体 CPU 及び 8CPU 使用時における性能値及び性能比（=単体 CPU に対する比）を示した．単体 CPU の性能が，ノード内の全メモリバンド幅を使えるのにこの値に留まってしまうのは，単体 CPU から連続して発行できるメモリリクエスト数に制約があるので，そこで律速されるためである．8CPU では，SMP 内のメモリ競合（メモリ・コンテンションと呼ばれる）により，単体 CPU の 8 倍の値に対し 67.8%に低下している．実運用上は 1 ノード 1 プロセスということはないので，メモリバンド幅が低くなる，あるいはノードに配置するプロセスの数によって使えるメモリバンド幅が変わってしまうのは実運用上問題となる可能性がある．

次に，結合ネットワークの基礎性能を示す．バンド幅（転送性能）の計測には，**MPI PingPong**通信プログラム[16]を使用し，通信プロセスを異なるノード間に配置した場合と同一ノード内に配置した場合とでそれぞれ実行し，メッセージ長に対する通信時間から平均転送性能を算出した（**図4.4(a)**）．ノード間転送性能に関しては，ピーク性能4GB/sに対し，最大実効性能で3.88GB/sを記録した．実効効率は，ピークに対して97%であり，極めて高い（効率の良い）ものであることがわかる．一方，ノード内（メモリコピー）性能は0.68GB/sであり，ノード間とノード内でのデータ転送性能のアンバランス（数倍違う）が懸念材料として指摘される．一方，レイテンシ（遅延）に関しては，**MPI Barrier**の時間を測定（**図4.4(b)**）した．ソフトバリアでは，プロセス数の増大とともにレイテンシが増大してしまっているが，ハードバリアを使えばプロセス数によるレイテンシはプロセス数によらずほぼ一定で

[14] http://www.stream.com
[15] STREAM ベンチマークには，Copy[a(i)=b(i)]，Scale[a(i)=q*b(i)]，Sum[a(i)=b(i)+c(i)]，Triad[a(i)=b(i)+q*c(i)]の 4 種類がある．
[16] 例えば http://software.intel.com/en-us/articles/intel-mpi-benchmarks/



(a) MPI PingPong



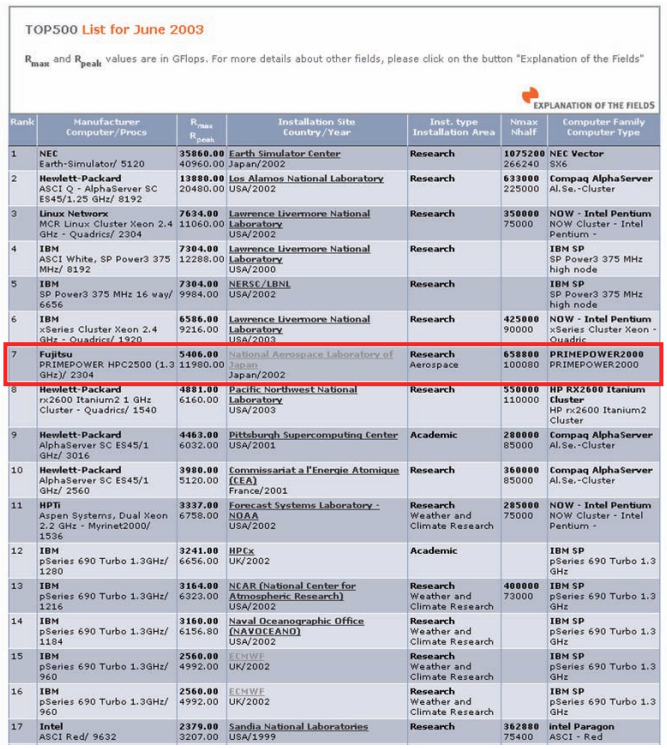
(b) MPI Barrier

図 4.4 結合ネットワークの性能

あり，最小値7μ秒という値が得られている．ハードバリアにより，多プロセス並列時の同期処理の高速化が期待できる．

有名な線形連立一次方程式を解く **Linpack ベンチマーク**[17]については，実効性能で R_{max}= 5.406TFLOPS という数値を記録（2003 年 6 月）した．このとき次元数は N_{max}=658,800 であり，すべての筐体を（64 ウエイ SMP）×2 に構成し直し，63 スレッド×36 プロセスという形態で計測した．ここに，R_{peak}=11.98TFLOPS である．計測には，10 時間ほどを要し，Linpack プログラムレベルではあるが，システムとしてきちんと動くことが確かめられた．**表 4.7** に，2003 年 6 月における Linpack トップ 10 のリスト，**図 4.5** にその時点での Top500 サイト[18]の表示を示した．ここで，比 = R_{max} / R_{peak} をあらわす．地球シミュレータや ASCI マシンといったカスタムメイドの大規模システムが多い中で，ほぼ市販品で構築したシステムとしては世界トップランクの性能に位置づけられる．ただ，表からわかるように比 = R_{max} / R_{peak}=0.451 は他のシステムに比べて実効効率が低く，性能向上に向けての関係者のより一層の努力が望まれるところである．また，一般のアプリケーションの実効性能が Linpack を超えることはまずないことを考えると，実効性能の向上は次期への課題の一つでもある．

[17] http://www.netlib.org/benchmark/hpl/
[18] http://www.top500.org



TOP500 List for June 2003

R_{max} and R_{peak} values are in GFlops. For more details about other fields, please click on the button "Explanation of the Fields"

EXPLANATION OF THE FIELDS

Rank	Manufacturer Computer/Procs	R_{max} R_{peak}	Installation Site Country/Year	Inst. type Installation Area	Nmax Nhalf	Computer Family Computer Type
1	NEC Earth-Simulator/ 5120	35860.00 40960.00	Earth Simulator Center Japan/2002	Research	1075200 266240	NEC Vector SX6
2	Hewlett-Packard ASCI Q - AlphaServer SC ES45/1.25 GHz/ 8192	13880.00 20480.00	Los Alamos National Laboratory USA/2002	Research	633000 225000	Compaq AlphaServer Al.Se.-Cluster
3	Linux Networx MCR Linux Cluster Xeon 2.4 GHz - Quadrics/ 2304	7634.00 11060.00	Lawrence Livermore National Laboratory USA/2002	Research	350000 75000	NOW - Intel Pentium NOW Cluster - Intel Pentium -
4	IBM ASCI White, SP Power3 375 MHz/ 8192	7304.00 12288.00	Lawrence Livermore National Laboratory USA/2000	Research		IBM SP SP Power3 375 MHz high node
5	IBM SP Power3 375 MHz 16 way/ 6656	7304.00 9984.00	NERSC/LBNL USA/2002	Research		IBM SP SP Power3 375 MHz high node
6	IBM xSeries Cluster Xeon 2.4 GHz - Quadrics/ 1920	6586.00 9216.00	Lawrence Livermore National Laboratory USA/2003	Research	425000 90000	NOW - Intel Pentium xSeries Cluster Xeon - Quadric
7	Fujitsu PRIMEPOWER HPC2500 (1.3 GHz)/ 2304	5406.00 11980.00	National Aerospace Laboratory of Japan Japan/2002	Research Aerospace	658800 100080	PRIMEPOWER2000 PRIMEPOWER2000
8	Hewlett-Packard rx2600 Itanium2 1 GHz Cluster - Quadrics/ 1540	4881.00 6160.00	Pacific Northwest National Laboratory USA/2003	Research	550000 110000	HP RX2600 Itanium Cluster HP rx2600 Itanium2 Cluster
9	Hewlett-Packard AlphaServer SC ES45/1 GHz/ 3016	4463.00 6032.00	Pittsburgh Supercomputing Center USA/2001	Academic	280000 85000	Compaq AlphaServer Al.Se.-Cluster
10	Hewlett-Packard AlphaServer SC ES45/1 GHz/ 2560	3980.00 5120.00	Commissariat a l'Energie Atomique (CEA) France/2001	Research	360000 85000	Compaq AlphaServer Al.Se.-Cluster
11	HPTi Aspen Systems, Dual Xeon 2.2 GHz - Myrinet2000/ 1536	3337.00 6758.00	Forecast Systems Laboratory - NOAA USA/2002	Research Weather and Climate Research	285000 75000	NOW - Intel Pentium NOW Cluster - Intel Pentium -
12	IBM pSeries 690 Turbo 1.3GHz/ 1280	3241.00 6656.00	HPCx UK/2002	Academic		IBM SP pSeries 690 Turbo 1.3 GHz
13	IBM pSeries 690 Turbo 1.3GHz/ 1216	3164.00 6323.00	NCAR (National Center for Atmospheric Research) USA/2002	Research Weather and Climate Research	400000 73000	IBM SP pSeries 690 Turbo 1.3 GHz
14	IBM pSeries 690 Turbo 1.3GHz/ 1184	3160.00 6156.80	Naval Oceanographic Office (NAVOCEANO) USA/2002	Research Weather and Climate Research		IBM SP pSeries 690 Turbo 1.3 GHz
15	IBM pSeries 690 Turbo 1.3GHz/ 960	2560.00 4992.00	ECMWF UK/2002	Research Weather and Climate Research		IBM SP pSeries 690 Turbo 1.3 GHz
16	IBM pSeries 690 Turbo 1.3GHz/ 960	2560.00 4992.00	ECMWF UK/2002	Research Weather and Climate Research		IBM SP pSeries 690 Turbo 1.3 GHz
17	Intel ASCI Red/ 9632	2379.00 3207.00	Sandia National Laboratories USA/1999	Research	362880 75400	intel Paragon ASCI - Red

図 4.5 Top500 サイト（2003 年 6 月）

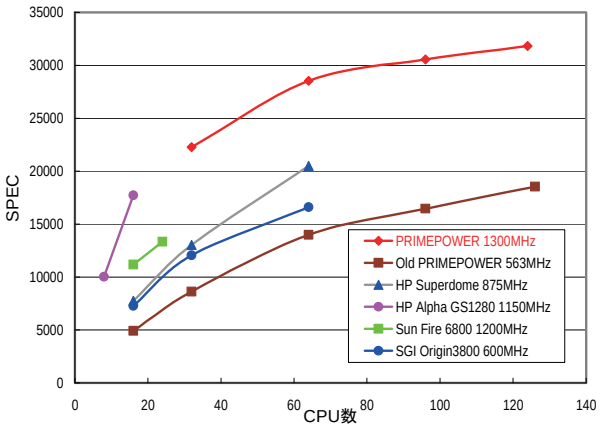


図 4.6 SpecOMP 2001M の性能

筐体内のスレッド並列性能の評価には，**SpecOMP 2001M ベンチマーク**[19]を用いた．**図 4.6** に 2003 年 6 月時点での，PRIMEPOWER HPC2500 の性能及び他システムの性能の比較を示す．横軸にスレッド数，縦軸に SPEC 値を示している．他 CPU のクロックがその時点では HPC2500 の SPARC64 V より低かったため，また大規模 SMP が HPC2500 以外にないため，HPC2500 は数値的には良い性能を示している．しかし，多スレッド時の直線性は良いとはいえない．

[19] http://www.spec.org/omp/

表 4.7　Linpack の結果（2003 年 6 月）

順位	システム	/CPU 数	機関（国，年）	提供者	Rmax	Rpeak	比
1	地球シミュレータ	/5,120	ES（日本，2002）	NEC	35,860	40,960	0.875
2	ASCI Q - AlphaServer SC ES 45 1.25GHz	/8,192	LANL（米，2002）	HP	13,880	20,480	0.678
3	MCR Linux Cluster Xeon 2.4GHz	/2,304	LLNL（米，2002）	Linux NW	7,634	11,060	0.690
4	ASCI White SP Power3 375MHz	/8,192	LLNL（米，2000）	IBM	7,304	12,288	0.594
5	SP Power3 375MHz16way	/6,656	NERSC（米，2002）	IBM	7,304	99,84	0.732
6	xSeries Cluster Xeon 2.4GHz Quadrics	/1,920	LLNL（米，2003）	IBM	6,586	9,216	0.715
7	**PRIMEPOWER HPC2500 1.3GHz**	**/2,304**	**NAL（日本，2002）**	**富士通**	**5,406**	**11,980**	**0.451**
8	Rx2600 Itanium2 1GHz Cluster Quadrics	/1,540	Pacific NW Lab.（米，2003）	HP	4,881	6,160	0.792
9	AlphaServer SC ES45 1GHz	/3,016	Pittsburgh SC（米，2001）	HP	4,463	6,032	0.740
10	AlphaServer SC ES45 1GHz	/2,560	CEA（仏，2001）	HP	3,980	5,120	0.777

ES: 地球シミュレータセンター，　LANL: Los Alamos National Lab.，　LLNL: Lawrence Livermore National Lab.，
NERSC: National Energy Research Scientific Computing Center，　CEA: Commissariat à l'énergie atomique

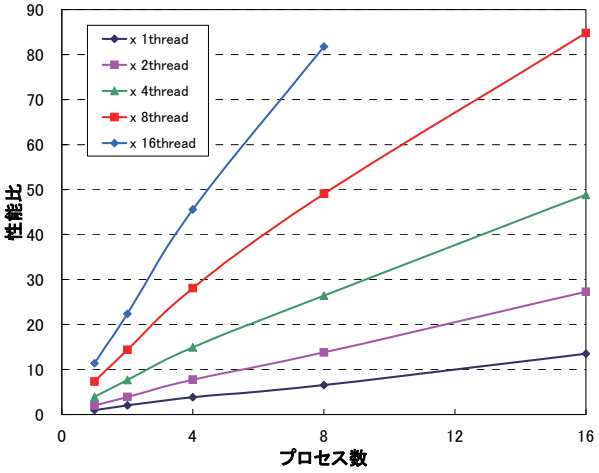


図 4.8　姫野ベンチマークの性能

図 4.8 に，良く知られた **Himeno ベンチマーク** M モデルの測定結果を示す．Himeno ベンチマーク[20]は，非圧縮流体を MAC 法で解く際に現れるポアソン方程式のプログラムのカーネル部分を取り出したものであり，メモリへのストライドアクセスがあり，裸のメモリ性能が現れる．問題規模は一定で，プロセス数と並列数を変えている．プロセスに対してもスレッドに対しても比較的良い直線性を示しているのがわかる．ただし，良く見ると，同一 CPU 数の場合，スレッドをたくさん使った方が性能が高い場合があり，後の章でも示すが，このあたりのスレッド選択の是非が SMP システムの難しいところでもある．

図 4.9 に，アプリケーションのベンチマークプログラムとして良く知られている NAS Parallel Benchmark の測定結果を示した．**NAS Parallel Benchmark**[21]は，NASA Ames 研究所で開発された CFD コードのいくつかをベンチマークプログラムとして公開しているものであり，採用している解法によって BT,CG,MG など幾つかの種類がある．流体のコードという意味では JAXA のアプリケーションと特性上は多くの共通点がある．図 4.9 は，CG と MG について，クラス B（中規模）とクラス C（大規模）のスケールアップ性能を示したものである．クラス B では，プロセス数の増加とともに通信量が増加するので性能のスケール性が低下している．クラス C では，規模が大きいので相対的な通信量の増加が小さいのでスケール性の低下割合は小さい．

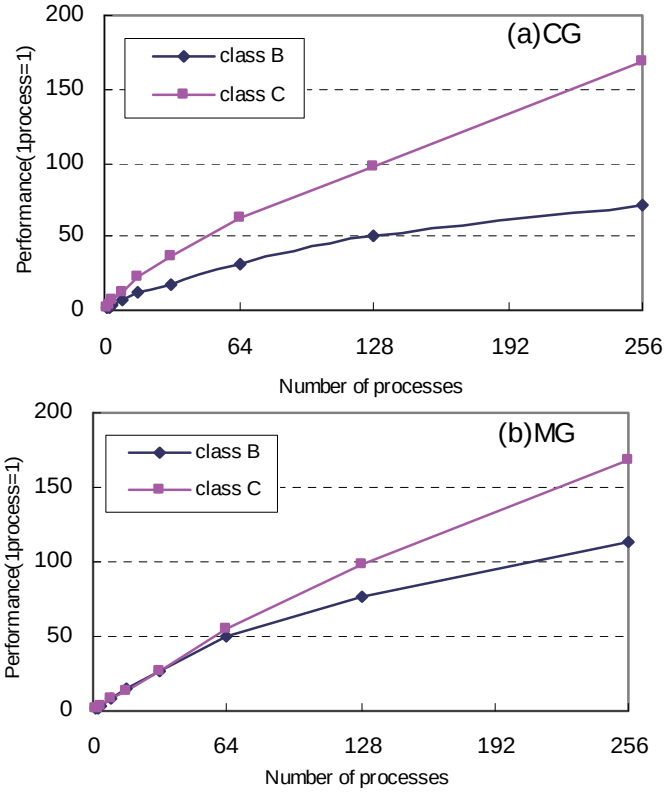


図 4.9　NAS Parallel Benchmark の性能

[20] http://accc.riken.jp/HPC/HimenoBMT.html
[21] http://www.nas.nasa.gov/Resources/Software/npb.html

4.3　入出力性能とストレージ特性

大容量ストレージへの基本（最大）入出力性能を調べるために，FC パス 80 本を用いた逐次読み書きのテストを，SAM-QFS，SRFS，STF，Fortran のレベルで実施した．ここに，SAM-QFS，SRFS は，それぞれ階層型ストレージ管理，ノード間高速ファイルシステムのことであり（第 3 章参照），STF とは，中央可視化システムから CeMSS のデータを読み込むときに使うライブラリ（**付録 F** 参照）を指す．それぞれのレベルからの，I/O 長に対する入出力性能の実測結果を**図 4.10** に示す．ローカルファイルへのバンド幅の最大は 6.6GB/s，ノード間では最大は 3.2GB/s であった．ノード間では，結合ネットワーク（ピーク 4GB/s）を介するために，入出力性能は低下する．また，Fortran レベルからは，（処理系にもよるが）Fortran バッファ（メモリコピー）が介在するため，さらに入出力性能は低下することに注意する．また，I/O 長によって数倍の性能差がある．これらのベンチマークにより，入出力データのブロックサイズを 8MB（標準はは 64KB），Fortran バッファの標準値を 64MB（標準）として運用することとした．

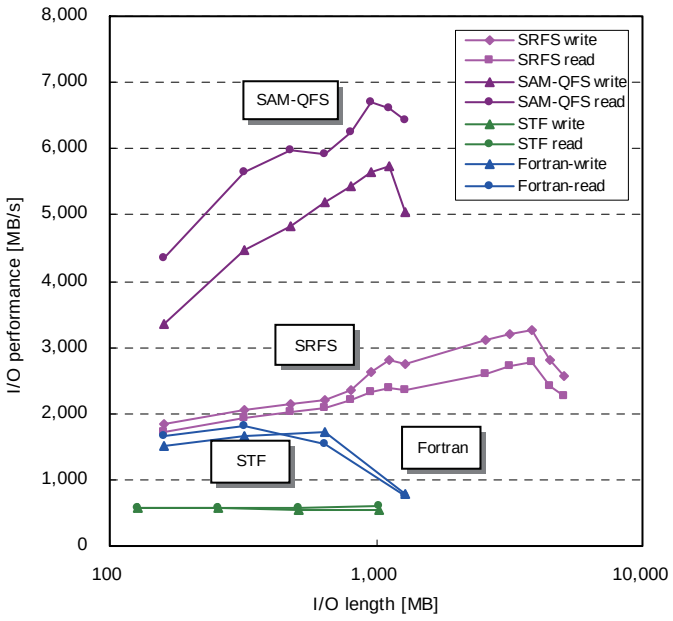


図 4.10　入出力性能ベンチマークの結果

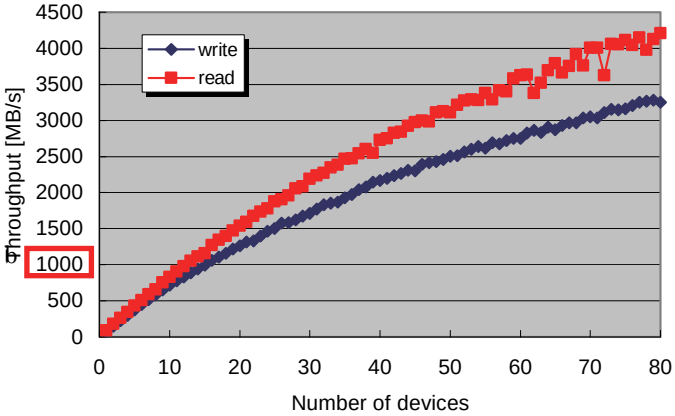


図 4.11　FC チャネル数による入出力性能値

当初予定していた 1GB/s という入出力性能を実現するために，**図 4.11** に示したように FC チャネル数による入出力性能のベンチマークを実施し，この結果から 16 ストライプという数字を決定した．ちなみに，32 ドライブを用いた SAM-QFS からの測定値は，アーカイブ 358MB/s，ステージ 387MB/s であった．

一方，STF については，GSN リンクを 4 本ストライプすることにより 500MB/s という入出力性能を実現している．

4.4　中央可視化システム CeViS の表示特性[3]

中央可視化システム CeViS の技術的側面については，**付録 F** を参照されたいが，ここでは，NS-III による CFD 解析結果の幾つかの可視化事例を紹介することにより，CeViS の大画面表示装置による可視化表示の特性（得失）を示す．

まず，三次元表示（ステレオ表示）や大画面表示をするからといって可視化表示の方法論が従来と様変わりするということは基本的にはないことに注意したい．変わるのはむしろ受け取り方，印象の方である．表示法としては，線画によるコンター，ポリゴンによる等値面，新しいところではボリューム表示等が使われる．線画によるコンター表示では，解析対象が複雑になって，線を多数表示するような場合に三次元表示が有効である．**図 4.12** は，航空機機体形状のまわりの CFD 解析結果から，物体表面の圧力分布と，主翼のある一定断面におけるマッハ数の分布を同時に示したものである．コンターの線が多くなると線の位置関係の把握が困難になるが，三次元表示により，線の前後関係や空間構造を容易に把握することができる．多数の線を表示した場合と同様に，多数の粒子を表示した場合も三次元表示は有効に機能する．**図 4.13** は，重合格子を用いて遷移飛行するヘリコプタの回転ブレード及び胴体まわりの流れを非定常的に解いたものである．物体表面の色は圧力分布を示し，パーティクル粒子は，ブレードと後流渦，胴体の干渉状況を表している．粒子数が多いと空間位置把握が困難だが，三次元表示により流れの構造をクリアにつかむことができる．

三次元大画面表示は，直感的理解の増進というメリットもある．同じコンテンツでも，例えばノートパソコン上で見るのと，大画面上や三次元表示で見るのとでは感覚的にかなり違った印象がある（**図 4.14**）．アピール度，写実感，鮮明度などこれほど違うものかと驚かされるものである．単に慣れの問題なのかもしれないし，個人差もあるだろうが，スケール効果とか体験感とでも分析することができるのではなかろうか．

また，このシステムは，Infinite Reality 3 というグラフィックスエンジンを持っていて，26 億 8800 万ピクセル/秒，680 億色の描画性能を有する．この性能と，ボリューム表示や半透明表示と組み合わせることにより，非常に高精細な画像を実現することができる（**図 4.15**）．こうした機能は，新たな現象の発見や知見の蓄積に有効な場合もある．

一方，このような三次元表示のデメリットとしては，1）そこに行かないと使えない，体感できない，2）発表で使えない，

論文に載せられない，3) 装置が大掛かり，コストもかかる，4) ソフトウェアが特殊で高価，使い勝手も特殊，5) データが大きいと表示系への負荷がかかる，などが挙げられ，システムの有効利用のためには運用上の工夫が必要と思われる．また，最近の可視化技術（特にハードウェア）の進展速度は著しく，一つの有力な技術があっという間に陳腐化してしまう，という点にも考慮する必要がある．

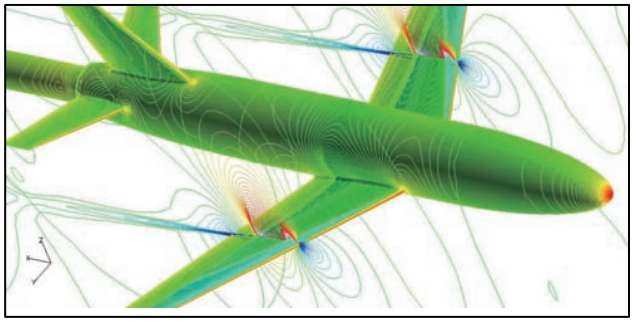


図 4.12　航空機全機まわりの CFD 解析結果

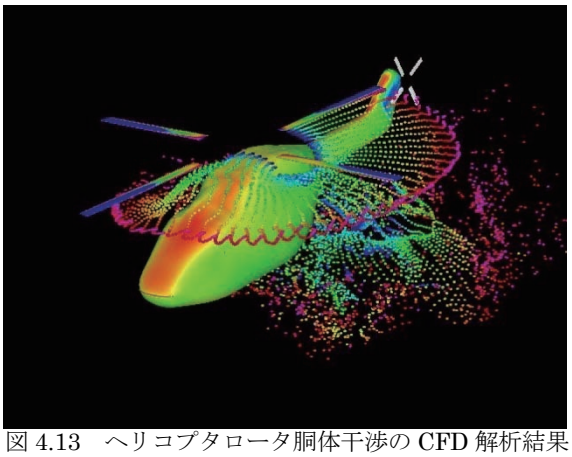


図 4.13　ヘリコプタロータ胴体干渉の CFD 解析結果

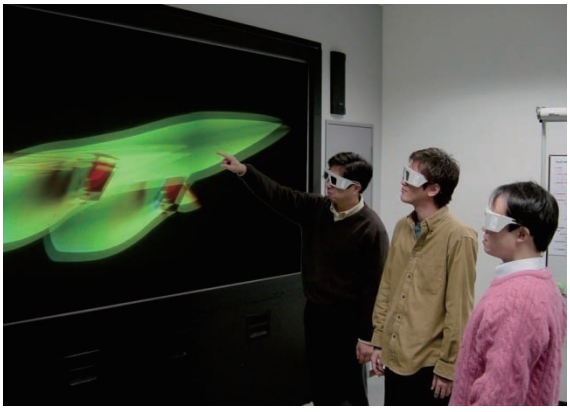


図 4.14　CeViS を利用している研究者

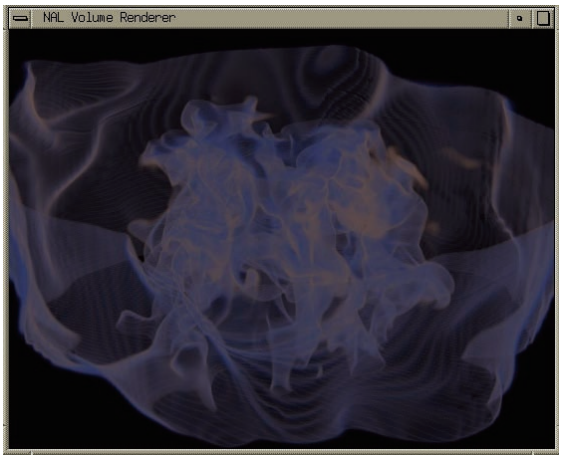


図 4.15　CeViS による高精細な画像の作成

4.5　NS-III の構成上の特徴とシステム運営

NS-III のシステム的な特徴の一つに，メモリを含む大規模ストレージ容量，さらに高入出力/転送性能がある．スーパーコンピュータというと，とかくその演算処理性能だけが注目されがちであるが，システムの使い勝手あるいは準備から結果の処理にいたる一連の処理速度(スループット)という観点からは，データの処理性能は極めて重要である．**表 4.16** は，NS-III におけるメモリ容量，ディスク容量などを前システムに対して比較したものであるが，処理性能に比べ大幅に増加しているのがわかる．各計算ノードのメモリ容量は 64GB あり，これは NS-II 全体のメモリ量より多く，計算規模でいえば NS-II 全体を使った計算が NS-III では 1 ノードでできてしまうことになる．しかも共有メモリなので，この範囲ではユーザは明示的な並列化をしないでもメモリ空間をそのまま利用できる．高速化したければ，自動並列等を利用すれば良い．一方，ストレージの大きさは，実際，システム移行時における新システムへの対応処理（例えば，再コンパイルやデータ/ファイルシステムの再構築）の迅速化，データを多く使う計算（例えば非定常計算）やデータを介在させる計算（連成解析）の効率化に繋がった．また，大量のデータをやりとりする入出力/転送性能（バンド幅）にも留意し，数 10GB のデータを 1 分程度で読み込むことを目標に，ファイバチャネルをストライピング技術によって複数本束ねることで，ディスクからの読み出し実効速度 1GB/s を実現した．これらの上に，SRFS と呼ばれるノード間高速ネットワークファイルシステムを実装することで，ユーザからはシングルシステムイメージでファイルを見ることができる．さらに，SAM-QFS

表 4.16　現行ストレージ容量の前システムとの比較

	NS-III	NS-II	倍率
ピーク(GFLOPS)	9,318	282	33.3
メモリ容量(TB)	3.58	0.0425	84.2
ディスク容量(TB)	57	0.3	190
テープ容量(TB)	620	10	62

表 4.17 主な NS コマンド

コマンド	機能	コマンド	機能
f90ns	Fortran のコンパイル	*ncan*	ジョブをキャンセル
nsub	ジョブを投入	*nlog*	ジョブの結果を表示
waitjob	実行待ちジョブを表示	*nquota*	ファイル割当使用量表示
actjob	実行中のジョブを表示	*ncp*	ファイルを高速にコピー

と呼ばれる階層ストレージ管理（HSM）ソフトウェアを導入した．これにより，ディスクが満杯になった場合には古いデータから自動的にテープに書き込み（アーカイブ）され，古いデータを再利用する場合には自動的にディスクに呼び出し（ステージング）され，ユーザはファイルの物理的配置を意識しなくて済むようになった．この高速大容量ストレージをユーザから効率良く使えるように，ブロックサイズの最適化, Fortran I/O バッファの最適化と指定, 1 レコード 2GB 以上の Fortran ファイルの作成, 高速ファイルコピー *ncp* ，高速ファイル転送 *gsncp* などの機能を支援した．

標準性や汎用性を強く意識したことも NS-III の特徴である．従来のスパコンといえば，特殊 OS，特殊環境というイメージが強かったと思われるが，NS-III では，OS に 64 ビット UNIX の Solaris8 を採用し，ほとんどのフリーウェアを実装した．昨今の研究開発環境で，GNU ツールや TeX 等はほぼ標準になっていることを勘案すると，CeNSS 上でもこうしたフリーウェアが使えるメリットは大きい．また，市販アプリなどの移植も比較的容易であり，Fluent や NASTRAN が利用できるようにした．UNIX や MPI は，LINUX ベースの並列クラスタマシンと共通の環境であり，ゆえにプログラムの移植も容易である．この辺りの事情は，運用管理サイドに対しても，システムツール開発の動機を与えるとともに，ミドルウェア層の充実を促した．例えば，従来から NS コマンドや NS ツールと呼ばれる一種のユーティリティを開発してきたが，これらを一旦見直した（スクラップ＆ビルド）上で系統的に再整備し（**表 4.17**），利用性を向上させた．このとき，プログラムのコンパイルやデバッグ，インタラクティブ処理などの小規模処理が大規模処理と同一環境でできるのは単に時間の節約以上に単純ミスを防ぐとか使い勝手を良くするという意味で意義がある．CeNSS では，サービスノードをその任に充当している．

システム運営という面では，効率性・透明性を高めるために **ISO9000 認証**[22]を取得し（2002 年 11 月），NS-III の運用を ISO9000 に基づく**品質マネジメントシステム（Quality Management System; QMS）**[4]と連動させた．体制やステアリング等の詳細は**付録 G** を参照されたい. いろいろな判断はあろうが, ISO9000 に基づく役割分担の明確な運営体制を敷くことにより，結果的に定常安定運用への迅速な移行，機動性のある障害に強いシステムの構築に繋がったと考えている．マニュアル作成やホームページの立ち上げ，特別利用や設備貸付などのミッション指向の運営に対しても，実務の削減やセキュリティ向上などに効果をもたらすとともに，障害対応やユーザ相談などの効率的ワークフローの確立，統計分析と情報公開によるアカウンタビリティの向上などに繋がった．

4.6 運用の基本方針

NS-III の運用については，その詳細は**付録 G** において詳しく述べているが，基本は，「**CeNSS は独自のスケジューラに基づくバッチシステム**として，**CeViS はインタラクティブモードに基づくオンラインシステム**として」運用した．CeViS については，特に導入初期は，LSF[23]と呼ばれるスケジューラによりバッチシステムとしても運用し，移行期間中の CPU リソース不足の補填に貢献した．運用時間については，24 時間 365 日を原則としたが，年末年始等の休日が続くときは，ジョブが枯渇する可能性があること，パッチ当て等のシステム作業が入る可能性があることを踏まえ，当初は以下の通りとした．また，次年度の運用方針は，上部会議体での承認・報告を経るようにし，公平性とアカウンタビリティを担保した. **表 4.18** に NS-III の運用の基本方針を示した．

4.7 おわりに

本章では，ベンチマーク測定や使用の初期段階から得られた NS-III の基本特性やシステムの特徴について主要な部分を示した．これらの結果は概ね以下に要約される．

(1) 単体 CPU 性能は，基本的な演算ベンチマークに関して, 同時代の CPU に対し遜色ない処理性能を示した．
(2) STREAM ベンチマークでは，複数 CPU 使用時には性能低下がみられた．
(3) スレッド単独でのベンチマークテストの結果は比較的良い．
(4) データの 1：1 の通信性能は，結合ネットワークの実効性能は高い．一方，**ノード内メモリコピー転送の性能はノード間に比べて相対的に低く，差が大きくアンバランス**である．STREAM の結果と合わせると，メモリ性能に不安がある．
(5) ハイブリッド処理性能については，同一並列数でもプロセス数×スレッド数の組み合わせによって性能が変わる可能性がある．
(6) Linpack 性能は，2003 年 6 月の時点では世界第 7 位の性能を記録した. ただし, **実効性能が 45.1%と低い**.
(7) 入出力性能は，I/O ノードからのディスクへの読み書き性能は高いが，計算ノードからの性能は劣化し，Fortran からとなるとさらに低下する．
(8) 可視化システムにおける，大画面表示や三次元表示の効果は確かにある．ただし，デメリットもあるので使い方や運用には工夫が必要．

[22] ISO9000 とは，国際標準化機構（ISO）による品質マネジメントシステム関係の国際規格．http://www.iso.ch/iso/en/iso9000-14000/index.html

[23] http://www.platform.co.jp/products/LSF_Family/

また，NS-III システムの特徴や運営，ジョブ運用の基本方針について述べた．上述した通り，NS-III の運営・運用は，2002 年 11 月より ISO9000 に基づいた QMS 活動と連動させた．ISO9000 認証はその後も継続して取得している．新しい試みではあったが，「明確な方針・責任・権限の下，業務プロセスをマニュアル化（手順化）して，それを仕組みとして継続的に実行，検証を行う」という QMS の考え方が，JAXA 以降も極めて有効に機能した（ている）ことを，ここに改めて付記しておきたい．

参考文献

[1] 松尾裕一 : 航技研数値シミュレータIIIの性能と特性，宇宙航空研究開発機構特別資料 JAXA-SP-03-002, 2004, pp.41-47.

[2] Matsuo, Y., Tsuchiya, M., Aoki, M., Sueyasu, N., Inari, T. and Yazawa, K.: Early Experience with Aerospace CFD at JAXA on the Fujitsu PRIMEPOWER HPC2500, *Proc. SC'04*, Pittsburgh, USA (Nov. 2004).

[3] 松尾裕一 : 航技研新中央可視化システムの概要とその戦略，航空宇宙数値シミュレーション技術シンポジウム2001論文集，航空宇宙技術研究所特別資料 NAL SP 53, 2002, pp.149-154.

[4] 品質マネジメントシステム規格国内委員会監修：対訳 ISO9001:2008，品質マネジメントの国際規格[ポケット版]，日本規格協会編，2009.

表 4.18　NS-III の運用基本方針

(1) NS 計算機設備

① 窓口業務（受付，クレーム対応，プログラム相談）
　　平日　　9:30～12:00，13:00～17:00

② アーカイブ室
　　平日　　9:00～20:00
　　保守点検日　13:30～20:00

③ NSシステムの運用時間帯
　　平日　　9:00～翌9:00
　　閉庁日の前日　　9:00～翌7:00
　　保守点検日　　13:30～翌9:00

（注）平日18:00以降において，
- 中央NSシステム（CeNSS）のジョブが途絶えるとCeNSSは停止する．
- 中央可視化システム（CeViS）のジョブが途絶えるとCeViSは停止する．

④ システム作業が必要であるとCFD技術開発センターが判断した場合は，運用を一時停止することがある．この場合は，原則として停止予定時刻の6時間以上前に運用停止予定時刻と運用再開時刻をユーザにメールする．ただし，障害発生時は除く．

⑤ 年末年始休館
（例）2003年12月26日～2004.年1月5日

⑥ 年度末休館　　3月31日

⑦ 運用不可能な状況(機器の導入，撤去，システム関連工事等)の場合は休館とする．その他，CFD技術開発センターが運用不可能と判断した場合は休館とする．

⑧ 定期保守　　毎月第2月曜日　8:00～13:30

（注）保守点検日が祝・祭日と重なった場合は，その次の日に行う．

(2) NSネットワーク

⑨ 窓口業務（申請受付，クレーム対応など）
　　平日　9:30～12:00，13:00～17:00

⑩ 定期保守　　毎月第2木曜日　8:00～10:00

（注）保守点検日が祝・祭日と重なった場合は，第3木曜日に定期保守を行う．

(3)NS 電気・空調設備

⑪ 運用方針は，NS 計算機設備と同一とする．

⑫ 定期保守は，NS 計算機設備に合わせて行う．

第 5 章　CeNSS における JAXA アプリケーションの性能評価

5.1　はじめに

本章では，実際に使われている JAXA 実アプリケーションの CeNSS における性能評価結果とその解釈上の問題点について論ずる．まず，単体 CPU 性能とその解釈上の課題について述べる．次に，並列性能とアプリケーション特性と性能との関連付け等について考察し，課題についても述べる．これらの課題に対する考察や対策は，後章で述べることとする．

5.2　CeNSS の構成とプログラミングスタイル

I/O などの部分を除いた CeNSS の構成イメージはすでに**図 3.3** に示したところである．計算ノードは，32 個の CPU から成る SMP（Symmetric Multi Processors）を構成し，1 ノードは 64GB の共有メモリ空間を有する．**図 3.3** において，DTU とは，Data Transfer Unit の略で，結合ネットワークにデータを送り出す/受け取る論理上の装置を表すが，DTU あたり，16 プロセスを同時に処理することができる．計算筐体は全部で 14 筐体，ノード数では 56 ノードあり，筐体は，富士通製 PRIMEPOWER HPC2500 である．CPU は，SPARC64 V を採用し，ピークで 5.2GFLOPS の性能と 2MB のオンチップ L2 キャッシュを有する．こうした構成のシステムは SMP クラスタと呼ばれることがあり，メモリはどの CPU からみても厳密に対称な配置を有している．

CeNSS における Fortran の並列プログラミング体系は，すでに**図 3.18** に示した．ノード内では，自動並列または OpenMP あるいはそれらの混在によるスレッド並列を用い，ノード間では，MPI またはデータ並列言語の一種である XPFortran によるプロセス並列を組み合わせることにより，いわゆるハイブリッド・プログラミングのスタイルを標準として採用している．**図 5.1** に並列ジョブ実行のユーザビューを示す．例として，4 スレッド×8 プロセスの場合を示した．これにより，1,000 を超えるような多数 CPU 使用時のスケーラビリティの問題や，大規模並列プログラミングの困難に対して一定の現実的な解決を与えている．詳細は，文献[1][2]を参照されたい．ここで，ノード内のスレッド並列は必須ではなく，MPI や XPFortran によるプロセス並列だけのプログラミングも可能であることに注意する．

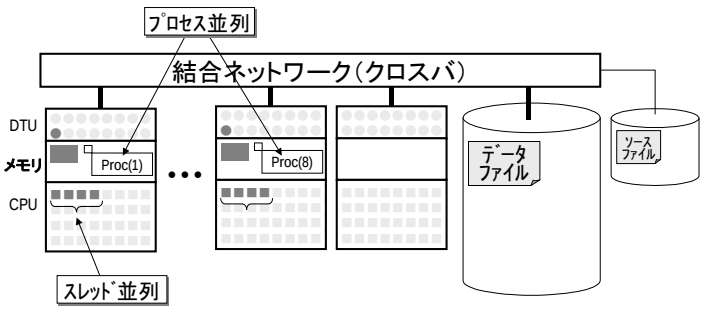


図 5.1　並列ジョブ実行イメージ

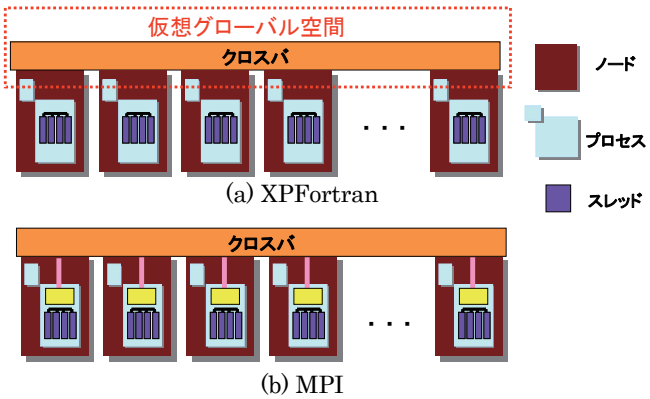


図 5.2　メモリ確保イメージ

XPFortran と MPI とでは，メモリ管理の考え方に違いがあることにも留意が必要である．**図 5.2** に示すように，XPFortran では，仮想グローバルメモリ空間を取るのに対し，MPI ではプロセス単位でメモリ空間を取る．従って，XPFortran では，メモリ管理が簡便で並列プログラミングが容易なのに対し，MPI では，プログラミングは大変だが複雑な通信が可能であり，それぞれ一長一短がある．JAXA における利用割合は，MPI : XPF ≒ 6 : 4 であり，近年は MPI の割合が漸増しつつある．

5.3　JAXA 実アプリケーション

表 5.3 に，本報告において性能評価やチューニング事例に取り上げた JAXA 解析コード（JAXA 実アプリケーションまたは JAXA 実アプリ[3]-[8]，ただし，JAXA を代表するという意味ではないことに注意.）の一覧を示す．これらは，JAXA で実際に用いられている並列 CFD コードであり，適用される対象により数値解法/格子形状や並列化手法が異なっている．コード P2，P3 が学術系の課題（P2：燃焼流，P3：平行平板間非圧縮乱流）を解析するためのものであり，あとの 4 本は工学系のコード（設計開発に使われるもの）である．基礎式は，いずれのコードも Navier-Stokes 方程式である．

表 5.3　性能評価した JAXA 並列 CFD コード一覧

コード（名前）	ｱﾌﾟﾘｹｰｼｮﾝ	ｼﾐｭﾚｰｼｮﾝﾓﾃﾞﾙ	数値解法	並列化	言語（行数）
P1 (LESFOIL)	翼	LES	FDM（構造格子）	OpenMP + MPI	F77 (7,000)
P2 (HJET)	燃焼	DNS	FDM ＋化学反応	OpenMP + MPI	F77 (10,000)
P3 (CHANL)	非圧縮乱流	DNS	FDM ＋FFT	OpenMP + XPF	F77 (17,000)
P4 (HELI)	ヘリコプタ	URANS	FDM（重合格子）	自動並列 + XPF	F77 (13,000)
P5 (UPACS)	航空汎用	RANS	FVM（ﾏﾙﾁﾌﾞﾛｯｸ）	MPI	F90 (20,000)
P6 (JTAS)	航空汎用	RANS	FVM（非構造格子）	MPI	F77 (20,000)

5.4 JAXA 実アプリの単体 CPU 性能とその意味

まず，**表 5.3** に掲げた 6 本のコードの単体 CPU 性能を調べた結果をコードの他の特性とともに**表 5.4** に示す．コード P1～P4 の 2 次キャッシュミス率が 1%以下なのを見てもわかる通り，これらのコードの性能評価には性能チューニングされた版を用いている．ここで，キャッシュと[24]は，CPU と主メモリの間にあって，容量は小さい（CeNSS の場合，CPU あたり 2MB）が主メモリよりアクセス時間が高速なメモリのことを言い，CPU のクロックと主メモリへのアクセス時間のギャップを埋めるために使われる．また，キャッシュミス率とは，CPU からキャッシュにアクセスしたときに，必要なデータがキャッシュ上になかった割合を指し，ミスが発生すると主メモリにデータを取りに行くので時間がかかる（性能が落ちる）ため，これが小さいほうが望ましい．我々は，富士通株式会社の指導により，一応 1%以下にすることを目安にチューニング等を行っているが，コード P6 のように，プログラムによっては，ミス率を減らすのが原理的に困難なものもある．なお，このあたりの性能の測定法やチューニングの実例については，以下の 7-9 章を参照されたい．

コード P1 から P6 に向かって，メモリコストが約 10%から 70%に増加しており，それと反比例して実効性能は，700MFLOPS 弱から 100MFLOPS 強まで低下しているのがわかる．ここで，メモリコストとは，CPU 時間中に占めるメモリアクセス時間の割合を示す．また，実効効率としては，12.8%から 2.2%まで低下している．ここで，

実効効率 = 実効性能値（MFLOPS）/理論ピーク性能値 (5.1)

で算出した．この場合，理論ピーク性能は，CeNSS では 5.2GFLOPS（= 5,200MFLOPS）である．計算機でよく言われる「ピーク性能より実効性能・実効効率が重要」というのは，このようにピーク性能と実効性能の間に大きな乖離があることに由来する．利用する側からすれば，実際にそのコードが速く走れば良いのであって，ピーク性能や実効効率は単なる一つの目安にすぎない．

富士通製 PRIMEPOWER HPC2500 に基づく CeNSS システムの CPU あたりのピーク性能が 5.2GFLOPS であるというのは，1 マシンサイクルに M&A 命令が 2 命令動作（FMADD オプション）することから，

1.3GHz（クロック）×2FLOPS×2 = 5.2GFLOPS

という意味である．一方，1 マシンサイクルに普通の浮動少数点演算命令が 2 命令動作（NOFMADD オプション）の場合には，ピーク性能は

1.3GHz×2FLOPS = 2.6GFLOPS

となる．**表 5.4** の M&A 命令の出現割合に眼をやると，実際にはせいぜい 13%以下であることがわかる．この場合，ピーク性能は，

5.2×0.13 + 2.6×0.87 = 2.938GFLOPS

ということになり，単体ピーク性能 = 約 3GFLOPS の CPU を使っていることと同等になる．とすれば，**表 5.4** でいうところの（5.2GFLOPS に対する）実効効率というのは，単に一つの比を表すに過ぎず，M&A 命令の出現割合を考慮してピーク性能及び実効効率を換算し直す必要がある．そこで，M&A 命令比 = α として

実ピーク性能 = 5,200×α + 2,600×（1－α）

として

実効効率 = 実効性能値（MFLOPS）/実ピーク性能値 (5.2)

として換算し直すと**表 5.5** のようになる．このように，バルクの値としての実効効率の意味はそれなりにあるとしても，細かくみてみると実効効率の意味は曖昧なものである．一般的な傾向としては，メモリコストと実効性能は相反する傾向にある．また，M&A 命令の効果は残念ながらあまり出ていない．

それでは，実効性能（絶対値）だけを見ればよいのではないか，と言われるかもしれないが，ここで問題になるのは，あるいは，ユーザとして関心が高いのは，その実効性能の持つ意味である．つまり，自分のコードに関する現状の実効性能は高いのか低いのか，あるいは，もっと高いところを狙えるのかどうか，ということである．例えば，コード P1 は，**表 5.4** では実行性能 666MFLOPS，実効効率は 12.8%という数字を出しているが，もっと高いところを狙いたいといったときにそれは可能なのかどうか？ 一方，コードＰ6 は，実効性能はかなり低いのでもっと上を狙いたいと思うのは当然であろうが，ここでやみくもに性能チューニングに入っても，果たして目標をいくらにおいたら良いのか，どの程度の改善が望めるのか，という情報がなければ，徒労に終わる可能性もある．このような個別のケースに適応できる何か指標のようなもの，あるいは有効な方針が強く望まれるところである．本課題・疑問に対する一つの考え方を，後の 6 章において論ずる．

表 5.4 JAXA 並列 CFD コードの特性と単体 CPU 性能

コード	P1	P2	P3	P4	P5	P6
2 次ｷｬｯｼｭ ﾐｽ率	0.43%	0.34%	0.52%	0.60%	1.04%	2.51%
M&A 命令比	9.7%	12.4%	4.4%	6.6%	5.7%	6.7%
計算コスト	87%	91%	78%	76%	66%	29%
メモリコスト	13%	9%	22%	24%	34%	71%
実効性能 MFLOPS	666	648	241	422	160	114
実効効率 (5.1)	12.8%	12.5%	4.6%	8.1%	3.1%	2.2%

[24] キャッシュは通常，階層構造をなし，CPU に近い方から 1 次キャッシュ，2 次キャッシュと呼ぶ．

表 5.5 M&A 命令の出現割合を考慮した性能及び実効効率

コード	P1	P2	P3	P4	P5	P6
ピーク性能 GFLOPS	2.852	2.922	2.714	2.772	2.748	2.774
実効効率 (5.2)	22.7%	22.3%	8.9%	15.2%	5.8%	4.1%

5.5 JAXA 実アプリの並列性能とその解釈

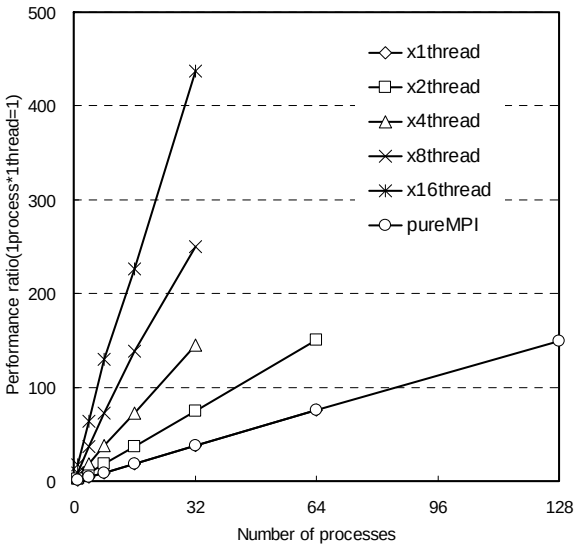
ここまでは，CeNSS における JAXA アプリの単体 CPU 上での性能評価結果とその解釈及び問題点について論じた．一方，実際の解析においては，当然のことながら**並列実行**が主となる．そこで次に，並列実行時の処理性能を分析する．

並列実行においてまず重要となるのは，プロセス数を変化させたときの処理性能がどうなるかである．一般的な並列化では，逐次（非並列）コード→スレッド並列→プロセス並列，と進むのが通例と思われるが，JAXA の場合，前システムからの経緯でプロセス並列化は何らかの形（NWT-Fortran または MPI）で既に済んでいるからである．並列処理の主目的は高速化にある（**付録 D** 参照）ので，プロセス数を増やして高速化したいと思うのが普通であろう．しかし，アムダールの法則（**第 10 章，付録 D**）により，並列度を 10 倍にしても処理性能は 10 倍になるわけではないので，性能測定や並列特性の把握が必要になる．

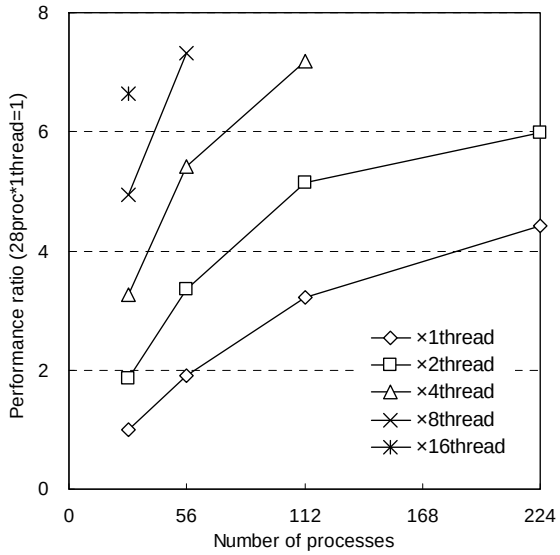
CeNSS は SMP クラスタゆえ，ノード内では自動並列または OpenMP によりスレッド並列が可能である．よって，スレッド並列を使ったときの性能が次に問題になる．スレッド並列をプロセス並列と組み合わせて使う状況として幾つかのパターンが考えられる．プロセス数を固定して，スレッド数を増やして性能向上を図るというパターンが最も多いと思われるが，並列度（CPU 数）を固定してプロセス数とスレッド数の組み合わせを変えてより高い性能を狙うという方法もある．以上は，問題規模は一定と考えたが，問題規模そのものを変えるという，という並列実行もある。

実際の測定は，まずは問題サイズ（空間の格子分割数）を一定にして，スレッド数を固定し，プロセス数を変化させることにより，経過時間を実測した．（これを「スピードアップ性能」という。）格子分割数は，できるだけ実問題に近いサイズとし，また，他のジョブからの影響を避けるために，他のジョブが一切ない状態で測定した．コードには，性能チューニングされた版を用いた．測定法やチューニングの実際については，以下の第 7-9 章を参照されたい．

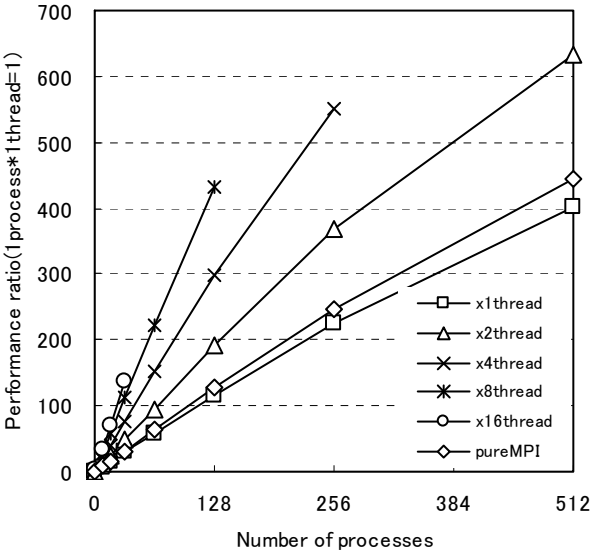
図 5.6(a)は，コード P1 の性能測定結果を示したものである．横軸にプロセス数，縦軸には，1 プロセス×1 スレッドのときの性能を基準(=1)としたときの性能比を取ってあり，何本かの線は，スレッドを 1,2,4,8,16 に固定してプロセス数を変化させた場合に相当する．格子サイズは，200×300×360 である．格子サイズがあまり大きくないこともあり並列数は多くないが，このコードの場合，最大 400 並列程度までは，プロセスに対してもスレッドに対しても直線性（スケーラビ



(a) コード P1



(b) コード P3



(c) コード P5

図 5.6 JAXA 並列 CFD コードの性能

リティ，**付録 D** 参照）は良好であることがわかる．

図 5.6(b)に，コード P3 の測定結果を示す．縦軸は，28 プロセス×1 スレッドの値を基準としたときの性能比である．格子サイズは，実際に合わせて 2,048×448×1,536 とした．このコードでは，プロセス性能の直線性は良くなく，プロセス数が多くなると性能曲線の傾きは小さくなる．これは，FFT おいて配列の持ち替えによる全対全通信が発生し，プロセス間の通信量が多い（通信のオーバーヘッドが大きい）ためと思われる．一方で，スレッド並列については，プロセス数一定のラインでみると，スレッドが多くなるにつれ一定の割合で性能が向上しているのがわかる．CPU 数一定の観点でみると，プロセス数を倍にするよりスレッド数を倍にした方が性能が上がっている．

一方，**図 5.6(c)**は，コード P5 の測定結果である．格子サイズは，40×20×80×512 である．通信の少ないアプリなのでプロセス性能の直線性は最大 1,000 プロセスまでかなり良い．しかし，スレッドの性能向上はあまり大きくないので，このコードの場合は，スレッド数を倍にするよりプロセス数を倍にした方が高い性能が得られている．また，純 MPI と 1 スレッドで，スレッドによるオーバーヘッドの性能差が見えている（スレッドのオーバーヘッドもばかにならない）ことに注意する．

以上のように，各コードのプロセスやスレッドに対する性能特性は，コードによって明確に違いがあるのがわかる．上記 6 本の JAXA アプリの場合，主に用いる格子形状（構造か非構造か）により，メモリアクセス量・パターン（ローカル/グローバル/連続/リスト）が異なる．また，補間や FFT の使用の有無で隣接データ以外のデータを参照するかにより，通信量･パターンが異なる．

表 5.7 は，JAXA アプリについて，メモリアクセス量，通信量などを測定した結果である．これから，横軸にメモリコスト，縦軸に通信コストを取って，各コードの特性をプロットしてみたのが**図 5.8** である．ここで，**メモリコスト**=メモリアクセス時間/CPU 時間，**通信コスト**=通信時間/経過時間として，性能測定時にプロファイラ等で採取したデータから持って来たものである．ただし，同じ CPU 数でも使用したスレッド数，プロセス数の組み合わせによって，あるいは問題サイズなどによってプロットされる位置は多少異なることに注意する．図にプロットした値は，プロセス数はすべて 4 で統一して測定したものである．ここで，表 5.4 とメモリコストが異なるのは，ノード内のメモリアクセスの競合によるものと考えられる．

これより，**表 5.3** の 6 本の JAXA コードは，**図 5.8** に示したように，メモリアクセスも通信も少ない計算処理中心の**グループ I**（コード P1, P2），メモリアクセスは中程度で通信コストが多い（10%以上）**グループ II**（コード P3, P4），通信は少なくメモリアクセスが多い**グループ III**（コード P5, P6）の 3 グループに概ね分類できることがわかった．ちなみに，ベンチマークとして有名な Linpack 及び NAS Parallel Benchmarks の MG と CG (内容は第 4 章参照のこと.) を，参考までにチャート上にプロットした．このことから性能分析あるいは性能チューニングの指針として，プロセス間（ノード間通信）の並列チューニングが重要なのはコード P3, P4，メモリチューニングが重要なのはコード P5，P6 であることがわかる．このように，コードの大まかな特性を知ることは，性能向上やチューニングの指針を得る上で重要である．

表 5.9 は，各分類グループの特性を整理したものである．代表的な計算結果を合わせて示した. グループ II のコードに通信が多い理由は，計算領域の全ての点の情報を必要とする FFT を使っていたり，領域間の補間による多量の通信があるからである．しかし，このような重い通信が航空宇宙のアプリケーションとして典型というわけではない（むしろ異例）．今日の CFD では，差分法や有限体積法が主で，その場合，隣接情報しか必要としないので，通信量は，グループ I，III のように比較的少ないのが普通のように思われる．グループ III のコードは，**表 5.4** を見てもわかる通り，確かにメモリコストは高いが，アドレス操作やバリア同期などの処理も含まれていることを処理性能を考える上で頭に入れておく必要がある．

表 5.7　JAXA 並列 CFD コードのメモリアクセス量と通信量の測定結果

コード	ジョブ実行条件				特性情報	
	格子サイズ	並列化方式	プロセス数	スレッド数	メモリコスト	通信コスト
P1	40x150x90x4	MPI+OMP	4	30	32.75%	0.63%
P2	120x15x120x4	MPI+OMP	4	30	16.87%	0.01%
P3	2,048x32x1,536	XPF+OMP	4	10	31.12%	24.84%
P4	82x20x70	XPF+OMP	4	4	56.50%	23.44%
P5	80x80x80x4	MPI+自動	4	1	46.69%	0.12%
P6	80,925 接点	MPI	4	1	72.31%	0.01%
NPB_CG	150,000	MPI	4	1	64.60%	1.14%
NPB_MG	512x512x512	MPI	4	1	55.43%	0.51%
LINPACK	1000 元	MPI+OMP	4	30	13.86%	2.22%

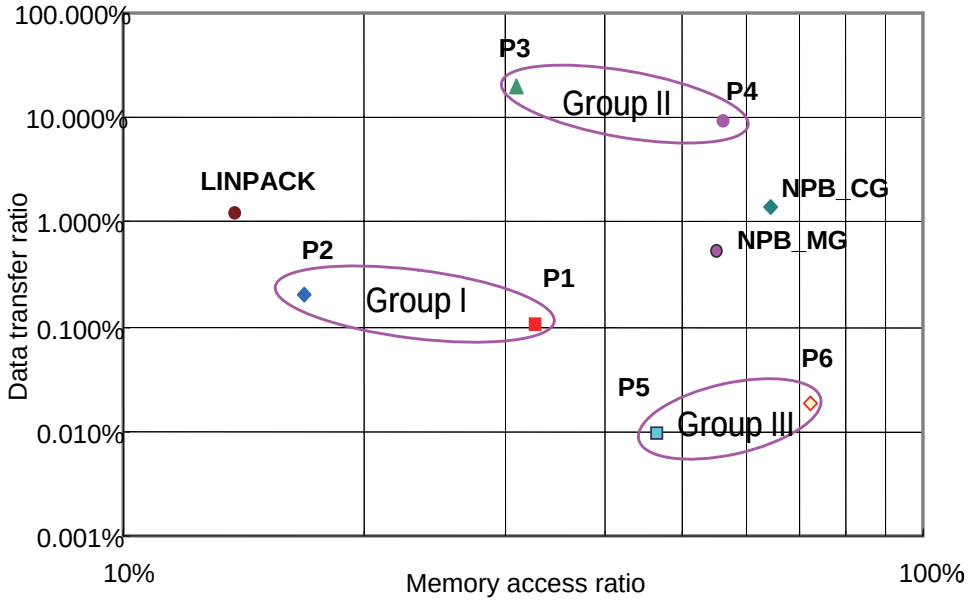
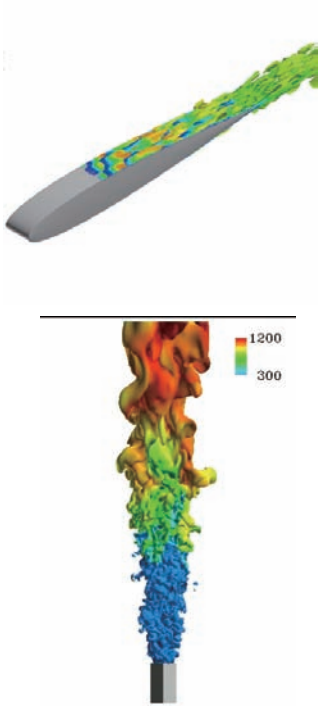
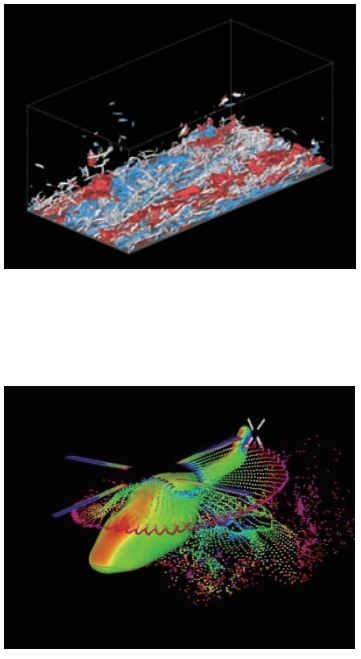
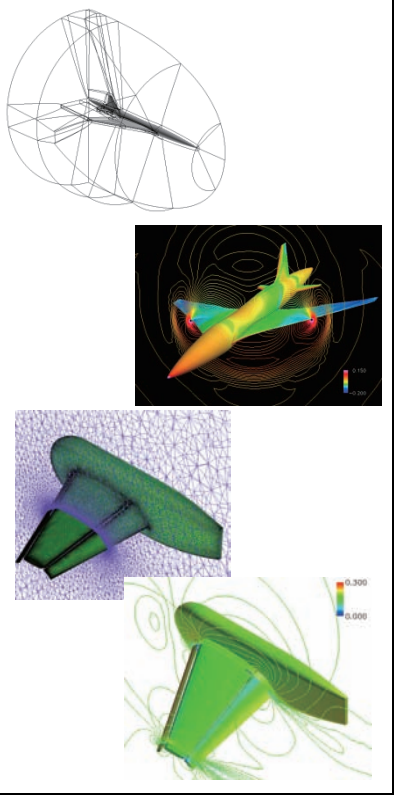


図 5.8 JAXA 並列 CFD コード（P1～P6）の特性

表 5.9 JAXA 並列 CFD コードの各グループの特性の整理

グループ	I	II	III
コード	P1,P2	P3,P4	P5,P6
コードの特徴	・軽メモリアクセス，軽通信 ・高演算密度 ・古典的なコード ・600MFLOPS 以上/CPU ・学術系用途	・中メモリアクセス，重通信 ・中演算密度 ・領域間の補間，FFT の使用 ・300 - 400MFLOPS/CPU ・特定の課題用途	・重メモリアクセス，軽通信 ・低演算密度 ・リストアクセス，アドレス操作，バリア同期 ・150MFLOPS/CPU ・工学系用途
計算結果の例			

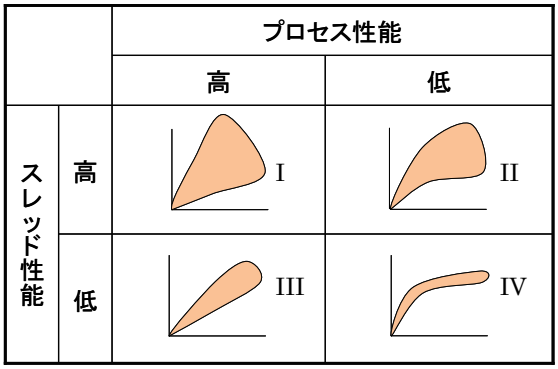


図 5.10　コードによる並列特性の整理

コード P2，P4 の性能特性は，**図 5.12** に示した．コード P2 は，グループ I に属するので，コード P1 と同様の傾向を示している．この状況をポンチ絵的に整理したのが**図 5.10** である．それぞれの図で横軸はプロセス数，縦軸は性能向上比を示す．それぞれのグループとプロセス，スレッドに対するスケール特性の対応付けを示している．このことから，ある並列コードが**図 5.7** のどの位置にあるかがわかれば，プロセスやスレッドに対するスケール特性を予め知ることができるとともに，ハイブリッドプログラミングによる並列化の戦略指針をある程度得ることができる．たとえば，**グループ I** に属する場合は，**プロセス，スレッドの組み合わせを如何様に選んでも 1CPU の場合の性能のほぼ並列数倍の性能が出る**．**グループ II** に属するコードの場合には，上記**図 5.6(b)** に示したように，プロセスとスレッドをうまく組み合わせることにより，**同一 CPU 数でもより高い性能を得ることができる場合がある**．一方，**グループ III** に属するコードの場合は，スレッド特性があまり良くないので，使うとしてもせいぜい 2 スレッドまでで，**スレッド並列を大々的に使うメリットは大きくない**ということになる．ここで，IV と書いたグループに属するコードは，6 本の JAXA コードでいうと P4 がそれに近い（**図 5.12(b)**）．このようなコードは**図 5.10** でいうと右下に属し，実際には数は少ないと思われるが，システムの運用効率を考えるとチューニングやアルゴリズムの見直し等で II または III の特性に持って行く必要がある．コードの移植性を考えると，まずは通信量を減らして III のタイプに持って行くのが妥当であろうと思われる．また全般的に，スレッド利用は，性能が上がるのはぜいぜい 2−4 スレッドまでで，それ以上は性能向上率が低下しているように見える．

以上の分析から，CeNSS においては，**図 5.6** に示したように，コードによってその並列性能は大きな差異を示すが，その性能パターンは，**図 5.8** のようなコードの特性パラメータ（メモリコスト，通信コスト）によってうまく整理できることがわかった．なお，このような特性は，**定性的には一般性はある**と考えられる一方で，その計算機システムのメモリ特性や通信特性によって具体的な数値は当然のことながら変わってくるものである（普遍性はない）ことに注意する．また，その特性が，アルゴリズム的に本質的なものかにも留意する必要がある．例えば，非構造格子を使ったときのメモリコストは非連続アクセスにより避けられないが，通信などはちょっとした工夫（順番の入れ替え等）で劇的に減ることもあるからである．このことは，やみくもに性能チューニングに入らず，アルゴリズムの見直しも含めた全体の見渡しが重要であることを意味しているし，（当然のことながら）**性能チューニングが全てを解決するわけではない**ことも頭に入れておかねばならない．

また，ここでの測定結果は，問題サイズ一定での並列性能（**スピードアップ性能，付録 D** 参照）を見たものであることに注意されたい．これは，並列数が大きくなると，CPU あたりの処理の規模は小さくなるので，並列処理のオーバーヘッドやレイテンシが効いてくるという試験である．一方，CPU 負荷を一定にして並列数を増やして（問題サイズを大きくして）いったときの性能（**スケールアップ性能，付録 D** 参照）をみるという性能評価試験もある．この場合，全対全通信のような通信パターンによっては非常に厳しい試験となるが，CFD コードの場合は，境界面のデータのみの通信となることが多いことから，並列数が増えても通信量が大幅に増えることは少ない（計算量は問題規模の 3 乗に比例するが通信量は 2 乗に比例する）ので，それほどシビアなことにはならないと推察される．実際の応用の場では，こうした使い方の方が多いかもしれない．例えば，コード P1 のスケールアップ性能を**図 5.11** に示す．ここで，ノードあたり 64CPU を 32 ノードという構成にして，OpenMP によるスレッド数を 30 一定とし，MPI にプロセス数を 1,8,14,28,56,72 と変更して性能を調べたものである．ここで，プロセスあたりの格子サイズは，120×15×120 の一定とした．図より，1,000 並列以上まで極めて良い（直線的な）スケールアップ性能を示しており，72 プロセス時には，実効性能で 1TFLOPS を超えている．

ここまで来た時点で，次の疑問・課題として，

① コードによってどうして並列特性に差が出るのか，

② そのコードがもしプロセス数とスレッド数の組み合わせが自由に選べるものであれば，組み合わせをどうすると一番性能が上がるか，

③ 並列数を変えた時にどのような性能になるのか，

などが挙げられる．このあたりの観点に対する分析・回答について，次の第 6 章において引き続き論ずることとする．

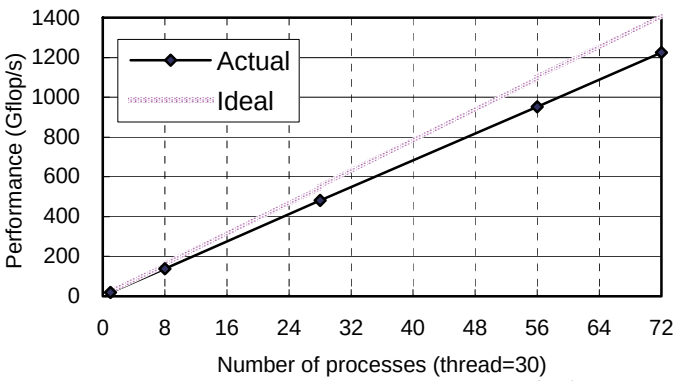
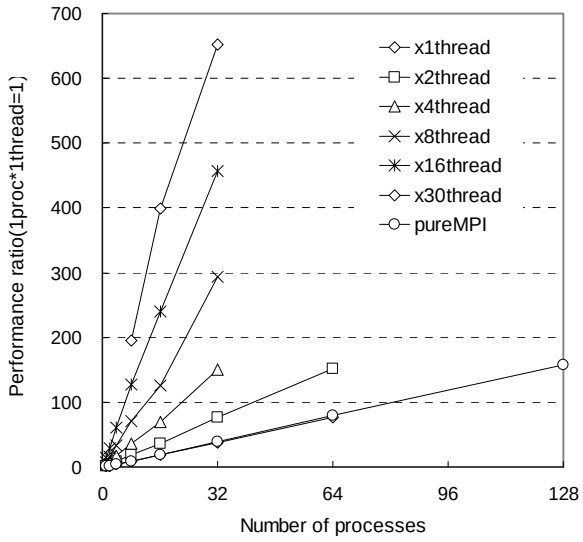


図 5.11　コード P1 のスケールアップ性能

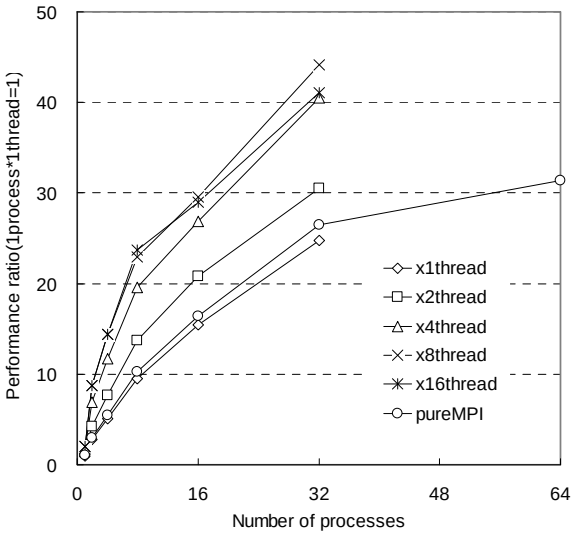
5.6 おわりに

本章では，実際に使われている JAXA 実アプリケーションの CeNSS における性能評価結果とその解釈上の問題点について論じた．まず，単体 CPU 性能とその解釈上の課題について述べ，次に，並列性能とアプリケーション特性と性能との関連付け等について考察し，課題について触れた．

単体 CPU の性能に関しては，ピーク性能値の設定によっては，実効効率の値・解釈に変化が生じてしまうこと，また，JAXA 実アプリケーションの性能は，メモリコストと通信コストによって特性が分類され，JAXA アプリは 3 つのグループに分類されること，ハイブリッドプログラミング（プロセス×スレッド）による並列特性（スピードアップ性能）もそのグループ分けに対応づけられることがわかった．



(a) コード P2



(b) コード P4

図 5.12 JAXA 並列 CFD コードの性能（その 2）

参考文献

[1] 松尾裕一：航技研数値シミュレータIIIの性能と特性，宇宙航空研究開発機構特別資料 JAXA-SP-03-002, 2004, pp.41-47.

[2] Y. Matsuo, et.al: Early Experience with Aerospace CFD at JAXA on the Fujitsu PRIMEPOWER HPC2500, Proceedings of the 2004 ACM/IEEE conference on Supercomputing, Pap. , 2004.

[3] 松尾裕一：差分法による翼まわり流れのLES，第12回数値流体力学シンポジウム講演論文集，pp.153-154 (Dec. 1998).

[4] 溝渕泰寛，新城淳史，小川哲：CeNSSを用いた水素噴流浮き上がり火炎詳細シミュレーション，航空宇宙数値シミュレーション技術シンポジウム2004論文集，JAXA特別資料SP-04-012, pp.202-207 (Mar. 2005).

[5] 阿部浩幸，松尾裕一：平行平板間乱流の大規模直接数値シミュレーション，航空宇宙数値シミュレーション技術シンポジウム2004論文集，JAXA特別資料SP-04-012, pp.21-26 (Mar. 2005).

[6] 近藤夏樹，青山剛史，齊藤茂：重合格子法を用いたロータ/胴体干渉の計算，航空宇宙数値シミュレーション技術シンポジウム2003論文集，JAXA特別資料SP-03-002, pp.232-237 (Mar. 2004).

[7] Takaki, R., Yamamoto, K., Yamane, T., Enomoto, S. and Mukai, J.: The Development of the UPACS CFD Environment, *High Performance Computing, Proc. ISHPC2003*, LNCS 2858, pp.307-319 (Oct. 2003).

[8] 村山光宏，山本一臣：非構造格子法を用いた航空機高揚力装置周りの流れ場解析の精度検証，航空宇宙数値シミュレーション技術シンポジウム2004論文集，JAXA特別資料SP-04-012, pp.82-86 (Mar. 2005).

第 6 章　CeNSS における性能チューニングの指針に関する一考察

6.1　はじめに

近年，NS-III における CeNSS のような大規模 SMP スカラー・システムや PC クラスタに代表されるいわゆる超並列のクラスタ・システムが出現し，旧来型のベクトル型も含めて計算機システムの構成の多様化が進んでいる．こうした中で，あるプログラム/コードが，このシステムでは性能が出るが，別のシステムでは出ない，といったシステムによって性能がばらつくケースが増えて来ている．グリッドコンピューティングという遠隔の異種計算機資源をむしろ積極的に使い合おうというような話もあり，今後一つのプログラム/コードをいろいろなシステムで走らせる時代になれば，こうした実行速度のばらつき，あるいはそれにどう対処する，という問題は，より切実さを増すであろう．こうした状況において，プログラム/コードの性能評価や性能チューニングによる性能向上の重要性は年々高まっているといえる．特に，クラスタ・システムの台頭とともに，並列化や並列実行が卑近なものとなっている今日，並列チューニングは非常に重要である．しかしながら，並列化についてはもとより，プログラム/コードの性能評価や性能向上の問題は，意外と議論されていないのが実情ではなかろうか．プログラム/コードのチューニングが必要・重要であるとわかっていても，では現状の性能はどうで，今後の努力でどの程度の性能向上が可能なのか，といった具体的な項目については，かつてのベクトル化率のような漠然とした指標はあっても，そのプログラム/コードに相応しい指標や方針といった考え方は今のところないように思われる．

本章では，そのような事情を踏まえ，CeNSS における JAXA アプリケーションの処理性能の解釈，性能向上の重要性やチューニングの方法論について考察する．また，実効ピーク性能という概念を提示し，その有効性やそれを用いたチューニング法について論ずる．

6.2　性能評価・性能向上・チューニングの重要性と課題

図 6.1 は，並列性能チューニングを含む性能チューニングの一般的な内容を整理したものである．CeNSS では，スレッド並列とプロセス並列を組み合わせたハイブリッド・プログラミングのスタイルを採用しているので，並列化において，スレッドを使っている場合には，プロセス並列に加えてスレッド並列チューニングも必要である．こうしたチューニング内容の切り分けも頭に入れておかないと，いま何をしているのかわからなくなってしまう危険があるので注意を要する．また，切り分けだけでなく，作業を効率良く行うためには，チューニング作業の順番も大切である．このあたりは経験がものを言う世界かもしれないが，他方で管理側からの支援も重要である．その意味もあり，JAXA コードに対する性能評価やチューニングの事例を，後の第 7 章～第 9 章に示すこととする．また，JAXA と富士通で整備したチューニングガイド[1]を**付録 H** に示した．

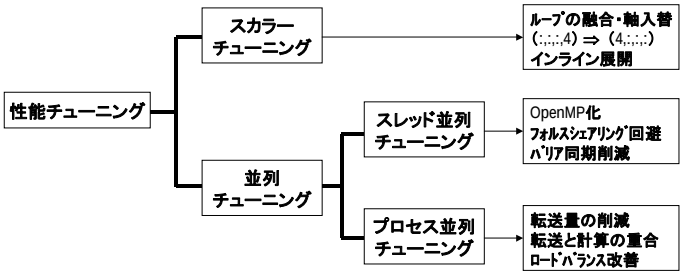


図 6.1　性能チューニングの類型

ここ 2−3 年で作られたコードを除き，我々の航空宇宙分野の CFD コードは，そのほとんどがベクトル機の時代に作られたものであり，処理性能は「**ベクトル化率**」というほとんど単一の指標により判断されてきた．ベクトル化率が高ければ実効性能も高い，というわけである．一方，CFD のコードは，（ぎりぎりにベクトルチューンされたもの以外は，）比較的単純な DO ループの多数の組み合わせから成るので，スカラー・システムでもそこそこの実効性能が出る場合は思いのほか多い．少なくともサブルーチン単位やループ単位でみると，実効効率で 20%を越えるような例もあり，かつて言われたような，「CFD コードは，スカラー機では数%の実効効率しか出ない（出せない）」ということでは必ずしもない．

しかしながら，スカラー機の場合には，メモリからデータを持って来る際に，ベクトル機のように連続的に持ってくるのではなく，一度キャッシュを介在するために，**キャッシュミスが**発生すると，ベクトル機に比べて性能がかなり落ちてしまうのも事実である．また，以下の例で示すように，場合よっては，チューニングにより性能が劇的に向上することもあるので，ベクトル機に比べると話としては厄介である．

図 6.2 は，JAXA のいくつかの実コードについて，単体性能 1.7GFLOPS のベクトル機 NWT と，チューニングなしで CeNSS にかけた場合，チューニング後に CeNSS で実行させた場合の性能を比較した事例である．CeNSS では，どの

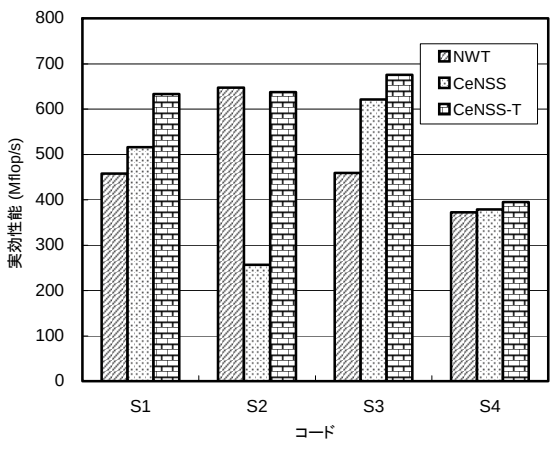


図 6.2　チューニングによる性能向上の例

コードもチューニング後には性能は向上している．絶対値としては，400-700MFLOPS の性能であり，ピークに対して10-15%の実効値が得られている．ただ，コード S2 については，チューニングなしでは，ベクトル機に比べ性能は低下するものの，チューニング後にはベクトルと同様かやや凌ぐ実効性能が得られている．このことは，性能評価やチューニングが重要であることを示すとともに，その困難さも示唆している．

現状，性能評価やチューニングに対する指標や方針は，キュッシュミス率をある一定値（例えば 1%）以下にするとか，ループ操作（融合，分割，軸入替）やメモリアクセスの（時間的・空間的）局所化，ホットスポットの検出のような一般的な例・法則を示すことはできる[1]．しかし，個々のコードではどうなのか，これをやれば定量的にどういう効果があるのか，については，ユーザの個人的な判断や勘・経験にまかせるしかない状況にある．今後，並列機のシステム構成がますます多様化し，プログラム/コード自体のプログラミング/コーディング・スタイルも多様化して行くようなことになれば，混迷の度合いも一層深まる可能性があり，普通のユーザが個別に性能向上を図るための支援策を如何に講ずるかについては，システムを有効に利用するためにもシステム運用側の大きな課題でもある．

6.3　実効ピーク性能とチューニング指針

そこで我々は，このような困難に対する解決策を模索するために，富士通株式会社の協力を得て，科学技術計算系の実際のコード 250 本の CeNSS 上での単体 CPU 実効性能を調査した．**図 6.3** は，メモリアクセス状況が性能に与える影響が大きいとして，メモリアクセスコストを横軸に取り整理したものである．これにより，実コード 250 本の平均メモリアクセスコスト=29%，平均実効性能=421MFLOPS であることがわかった．直線は，平均線を示している．JAXA コード 6 本についてもプロットしてみたが，コード P1，P2 は，平均に対しては十分性能は出ているが，コード P3，P5，P6 については，改善の余地があることがわかった．コード P1，P2 は，平均よりは高い性能が出てはいるが，もっと性能の高いコードもあるので，がんばりようによってはもっと高い性能が狙えるかもしれない．

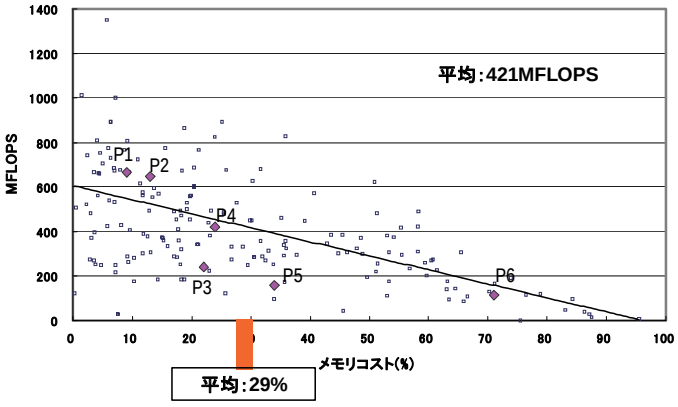


図 6.3　実効性能の調査結果

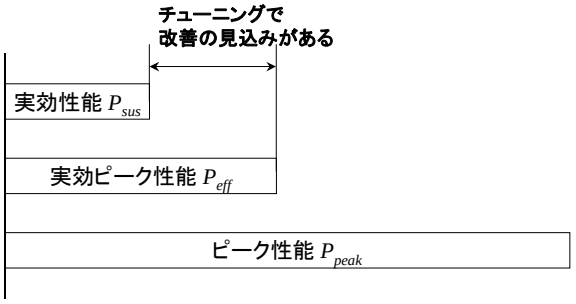


図 6.4　実効ピーク性能の考え方

以上のような実コード性能の評価分析活動から，**図 6.4** のような構図を考えるのが妥当であろうという結論に達した．すなわち，「**実効ピーク性能**」というものを考え，これをチューニングの指標としたらどうかというものである．実効ピーク性能は，理想的にチューニングして到達できる最高の性能値のようなものを意味する，問題は，実効ピーク性能をどう決めるかである．例えば，メモリコピーのみのプログラムは 0FLOPS である．A=B+C というプログラムは，シーケンシャルに実行すると

```
load B
load C
add
store A
```

のように 1 演算に 4 サイクルかかるので，1 演算/4 サイクル×1.3GHz=375MFLOPS の性能であるが，CeNSS では，1 サイクルに浮動小数点 2 命令同時実行可能なので，

```
load B & load C & add
store A & load B
load C & add & store A
```

のように 3 サイクルで 2 演算の実行が可能であるから，最高で，2 演算/3 サイクル×1.3GHz=867MFLOPS の性能を出すことができる．これの類推から，実効ピーク性能 P_{eff} を，

M&A 命令は 2 演算，他は 1 演算でカウント（ただし，DIV, SQRT も 1 演算）

$$P_{eff}\ (\mathrm{MFLOPS}) = \frac{\text{浮動小数点演算数}}{\text{理想実行時間}} \times 10^6 \quad (6.1)$$

MAX（浮動小数点命令数，それ以外の命令数）/（1.3GHz×2）

と定義し，**図 6.3** で調べたのと同じ 250 本の実コードに対して P_{eff} を調べた．ただし，式(6.1)は，

- 無駄な演算，データ移動はない
- メモリアクセス，整数演算は，2 命令/サイクル
- 浮動小数点演算命令は，2 命令/サイクル
- メモリアクセス，整数命令と浮動小数点演算は同時実行可能

として考えている．以下の**図 6.5** は，実効ピーク性能をメモリコストに対してプロットしている．ここで，都合により，FMADD オプションは付けていないので，実効ピーク性能の最大値は 2.6GFLOPS になっており，最大になっているコードも幾つかある．◇は，JAXA コードの値，直線は平均値を示している．全体での平均実効ピークは 1435MFLOPS であることがわかった．JAXA コードに関していえば，コード P1，P2 は平均より高く，コード P3，P5，P6 は平均より低い．調査の結果，P1，P2 が平均より高い理由は，浮動小数点演算が多いからであることがわかっている．逆に，コード P3 以下が平均より低い理由は，浮動小数点演算以外の命令の影響の可能性が高い．式(6.1)により，実効ピーク性能は浮動小数点演算数が多いほど高くなるので，これらのコードについては，浮動小数点演算数を増やす等のチューニングを行えば，より高い実効性能に到達可能ということである．

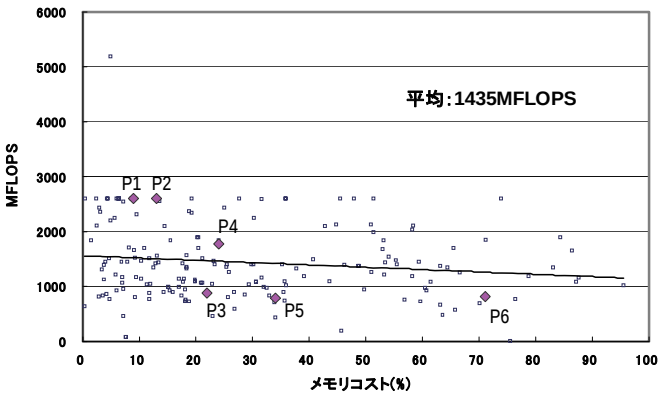


図 6.5　実効ピーク性能の調査結果

表 6.6 は，今までの測定結果から，6 本の JAXA コードの実効性能 P_{sus}，実効ピーク性能 P_{eff}，及びその比（「**実効ピーク性能比**」と呼ぶことにする．）P_{sus}/P_{eff} を示したものである．コード P1, P2 の括弧内の値は FMADD オプションを付けて測定したときの値であり，コード P1 については，FMADD により 20%以上の性能改善の可能性がある．P_{sus}/P_{eff} は，コード P6 を除き 20～30%という値が得られた．

図 6.7 は，250 本のコードに対して実効ピーク性能比 P_{sus}/P_{eff} をプロットしたものである．◇は，JAXA コードの値，直線は平均値を示している．全体平均は，29%であった．一般的にいえるのは，メモリアクセスコストが低い（図の左側）ほど，ピーク性能比は高くなることであり，コードの持っている本来の性能をより高く出しているといえる．JAXA コードについていえば，P3 を除きどのコードも平均線を下回っている．これは，本来の性能を出し切っていないことを意味しており，性能改善の余地があることを示している．無論，その改善の中身はコードによって異なるので，コード毎に検討してみる必要がある．例えば，詳細に調べてみると，コード P1 は割り算が，コード P2 は SQRT が多いことがわかった．これは，化学反応項や，Roe スキームの影響であることが想定される．よって，この辺りを集中的に工夫すれば，もっと性能を上げられる可能性はある．コード P3 は，チューニングという意味では，平均的な実力を出しているといえる．ただし，絶対性能は高くないので，（アルゴリズム的に工夫の方法がなければ仕方ないが，）基本的な性能を上げる努力をするかどうかである．コード P5，P6 は，平均値からすると悪い値ではない．逆に，メモリアクセスコストを変えないまま工夫しても，性能はあまり上がらない，あるいは，上げようがないことを意味している．図でいうと，もっと左側の位置に来るようにメモリアクセスのチューニングをすれば性能改善の余地は大きくなる．

表 6.6　JAXA コードの実効ピーク性能比

コード	P1	P2	P3	P4	P5	P6
実効性能	666	648	241	422	160	114
実効ﾋﾟｰｸ性能	2600 (3218)	2600 (3077)	873	1764	777	810
P_{sus}/P_{eff}	25.61%	24.91%	27.55%	23.90%	20.65%	14.07%

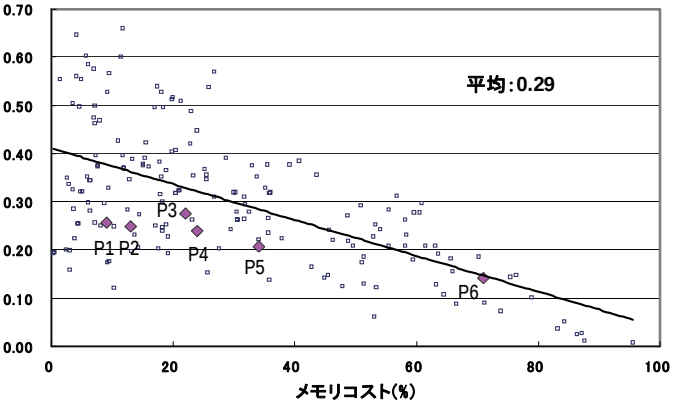


図 6.7　実効ピーク性能比の調査結果

以上の分析，考察から，実効ピーク性能を用いたスカラー性能チューニングについて以下のことがいえる．

実効ピーク性能

- コードの特性を知る指標として有効．
- それが高いということは，がんばり次第では，かなり高い実効性能が期待できることを意味する．
- それが低いということは，がんばっても性能がでない，コードそのものに何か問題を抱えていることを意味する．例えば，Load/Store をもっと減らして，浮動小数点の演算密度を増やす等の対策が必要．

実効ピーク性能比

- チューニングの指標として有効．
- それが高いということは，コードの持っている本来の性能に近いことを意味する．上記の例からは，30%程度が大括りな目標となろう．
- それが低いということは，本来の性能が出ていない，何か問題があり，チューニングでがんばれば性能向上も可能であることを意味する．

簡単にいえば，**「実効ピーク性能が高くて実効ピーク性能比も高ければ，そのコードのチューニングはうまく行っていて，コードの特性に応じた性能は出ている」**ことになる．ここで考えた実効ピーク性能や実効ピーク性能比という考え方は，まさにチューニングにおける一つの指標になると思われる．

6.4 おわりに

本章では，NS-III の中核並列計算機 CeNSS における大規模シミュレーションの性能向上の重要性や方法論について論じた．CeNSS における実コードの性能評価と実効性能に関する考察から，実効ピーク性能及び実効ピーク性能比という指標を提案し，それらを用いたコードの性能評価とチューニングの方法論と指標・方針について論じた．

本章と前章で述べた性能評価やチューニングというのは，本章のはじめに述べたように，単にシステム構成がベクトル機からスカラー機に代わって性能をきっちり出したい（出さなければならない）からというだけでなく，今後並列システムの有り様がますます多様化し，グリッドコンピューティングのようにあちこちでコードを使いまわす時代になった場合には，その必要性・重要性は一層高まって行くと予想される．そのような状況において，本章で考察・分析したような手法は有効な一手段となる可能性があることをここでは言及しておきたい．

参考文献

[1] 中央 NS システム（CeNSS）性能チューニングガイド Ver1.0，宇宙航空研究開発機構情報技術開発共同センター，富士通株式会社，2003.

[2] 松尾裕一：航技研数値シミュレータ III の性能と特性，宇宙航空研究開発機構特別資料 JAXA-SP-03-002, 2004, pp.41-47.

[3] 松尾裕一，土屋雅子：CeNSS における大規模シミュレーションの性能向上，宇宙航空研究開発機構特別資料 JAXA-SP-04-012, 2005, pp.45-50.

第 7 章　性能評価と性能チューニングの実例（その１） - 平行平板間乱流解析コード CHANEL

7.1　はじめに

乱流や燃焼のような自然現象の本質を解明することは，スーパーコンピューティング利用の典型的な分野である．この分野においては，現象の最小単位まで解明する必要があるため，基礎方程式をモデル化なしに直接数値的に解く DNS（Direct Numerical Simulation）という手法が採られる．本章で扱う平行平板間乱流解析コード[1]は，2 枚の平行な平板間を流れる乱流を DNS により解析するものである．この種の問題では，解析の規模が重要になるので，システムの数分の一を使うような大規模シミュレーションとなるため，処理性能を少しでも上げることが重要となる．本章では，学術系の CFD コードの代表である平行平板間乱流解析コードのスレッド並列チューニングの概要とハイブリッド並列性能について述べる．

7.2　コードの概要

平行平板間乱流解析コード（以下，コード P3）は，非圧縮 Navier-Stokes 方程式を時間進行で解き，並列化に XPFortran を用いたループベースの並列化方法に基づく約 20,000 行の Fortran77 プログラムである．時間進行には，粘性項に対しては２次精度クランクニコルソン法を，その他の項には３次精度ルンゲクッタ法を用いている．空間の離散化には４次精度差分を用いている．プログラムは，多くの 3 重 DO ループの計算から成り，一部に 3 重対角行列の行列解法，圧力ポアソン方程式の求解のために FFT を含む[1]．JAXA アプリケーションの中では，図 2.2 にあるようにタイプ 2 に属し，メモリアクセス的には中程度であるが，並列軸の持ち替えを行っているために転送負荷は非常に高い．

7.3　ASIS コードの性能分析

性能評価区間は，コード P3 の主計算ループ部分とし，時間計測には gettod を用いた．問題サイズは，実際は 2,048×448×1,536 であるが，チューニングの機動性を確保するため 2,048×32×1,536 とした．スレッド並列特性を調べるために，プロセス数は 4 に固定し，スレッド並列数を変化させてプロファイラによりコスト分布，性能情報等を採取した．測定条件の概要を**表 7.1** に，測定結果を**表 7.2** 及び**図 7.3** に示す．また，プロファイラによる出力を**リスト 7.4** に示す．

これより，ASIS コードは，スレッド並列の性能向上特性は悪く，以下の問題点が指摘される．

① 統計関連の処理ルーチン(budget, budgek1, budgek2)のコストが大きい(8 スレッド時で合わせて約 46%).
② バリア同期待ち(_libc_poll)のコストが大きい.
③ 実数のべき乗(g_adxi)のコストが大きい.
④ スレッド非並列のサブルーチン(ave)が残っている.

表 7.1　計算条件

実行ノード	32cpu×2 ノード
並列規模	プロセス並列…　XPFortran 4 プロセス スレッド並列…　自動並列 1,2,4,8 スレッド
格子サイズ	2,048×32×1,536（評価用データサイズ）
Iteration 回数	2 回（計算処理 2 回＋統計処理 1 回を実行）
翻訳ｵﾌﾟｼｮﾝ	-Kfast_GP2=3,ocl,hardbarrier,mfunc=2,parallel, reduction,noeval -O5 -x40 –Nrnotrap

表 7.2　性能測定結果

スレッド数	経過時間	性能比
1	747.47sec	1.00
2	745.18sec	1.00
4	614.95sec	1.22
8	553.67sec	1.35

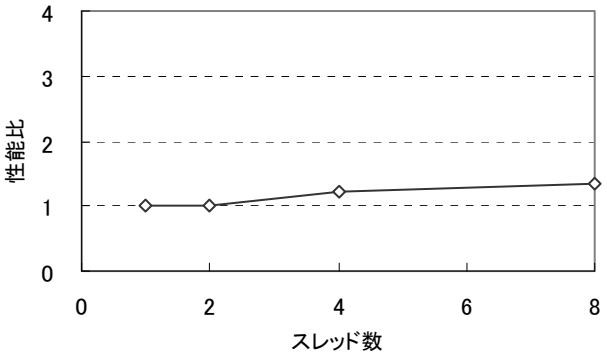


図 7.3　スレッド数に対する性能向上比

7.4　スレッド並列性能チューニング

上記の測定結果を基に原因を分析し，次の性能チューニングを段階的に試みた．

【Tune　1】実数のべき乗計算の乗算化

リスト 7.5 のようにべき乗計算を行う箇所が含まれているが，ソースは-Knoeval で翻訳されているため，べき乗を乗算化する最適化が抑止されていた．これに対して，翻訳時に-Keval を指定することで，べき乗を乗算に置き換えた．

【Tune　2】 False Sharing が発生する配列の次元追加

多重ループのワーク変数が１次元配列になっている箇所で，複数スレッドによるキャッシュライン競合(False Sharing)が発生し，スレッド数が増加するほど性能が劣化している可能性があったので，**リスト 7.6** のように配列に１次元追加してすき間を空けることで，False Sharing を回避した．

【Tune 3】 ロードバランス不均等なループの融合

バリア同期待ち(_libc_poll)の呼び出しコストが大きいサブルーチンを調査したところ, 特定のプロセスだけ動作するような spread do ループが複数連続して使用されていた. これに対し, **リスト 7.7** のように複数の spread do ループを融合し, 各プロセスが同時に動作するようにした.

【Tune 4】 並列化率の拡大

自動並列化では構造が複雑なため並列化されていないループを**リスト 7.8** のように OpenMP で並列化した.

リスト 7.4

コスト分布（プログラム全体）

Samples	% Run	Barrier	Start	End	
--<1thread>----------					
9.627300e+04	29.33	0.000000e+00	-	-	_libc_poll
3.438300e+04	10.47	0.000000e+00	-	-	g_adxi
1.767100e+04	5.38	0.000000e+00	-	-	_ioctl
1.030100e+04	3.14	0.000000e+00	3621	5454	ave_
8.121000e+03	2.47	0.000000e+00	774	785	cntdmat._PRL_10_
5.516000e+03	1.68	0.000000e+00	1232	1241	cntdma._PRL_9_
5.293000e+03	1.61	0.000000e+00	12	625	budgek1._PRL_1_
5.235000e+03	1.59	0.000000e+00	-	-	prbar_probe
5.025000e+03	1.53	0.000000e+00	12	625	budgek2._PRL_1_
4.956000e+03	1.51	0.000000e+00	533	673	rk2._PRL_2_
--<2thread>----------					
3.877300e+04	11.23	0.000000e+00	5631	5885	budget._PRL_4_
3.773200e+04	10.93	0.000000e+00	12	625	budgek2._PRL_1_
3.729500e+04	10.80	0.000000e+00	-	-	g_adxi
3.697400e+04	10.71	0.000000e+00	12	625	budgek1._PRL_1_
1.033200e+04	2.99	4.000000e+00	3621	5454	ave_
8.108000e+03	2.35	0.000000e+00	3931	3980	ave._PRL_7_
6.839000e+03	1.98	0.000000e+00	774	785	cntdmat._PRL_10_
6.447000e+03	1.87	0.000000e+00	-	-	prbar_probe
5.476000e+03	1.59	0.000000e+00	1232	1241	cntdma._PRL_9_
4.959000e+03	1.44	0.000000e+00	533	673	rk2._PRL_2_
--<4thread>----------					
5.842200e+04	13.34	0.000000e+00	12	625	budgek2._PRL_1_
5.838600e+04	13.33	0.000000e+00	12	625	budgek1._PRL_1_
5.770600e+04	13.17	0.000000e+00	5631	5885	budget._PRL_4_
4.045400e+04	9.23	0.000000e+00	-	-	g_adxi
1.954000e+04	4.46	0.000000e+00	3931	3980	ave._PRL_7_
1.188300e+04	2.71	1.549000e+03	3621	5454	ave_
6.711000e+03	1.53	0.000000e+00	774	785	cntdmat._PRL_10_
6.653000e+03	1.52	0.000000e+00	-	-	prbar_probe
6.288000e+03	1.44	0.000000e+00	389	454	stcs._PRL_23_
5.860000e+03	1.34	0.000000e+00	3880	3914	ave._PRL_9_
--<8thread>----------					
9.751900e+04	16.34	0.000000e+00	12	625	budgek2._PRL_1_
9.302900e+04	15.58	0.000000e+00	12	625	budgek1._PRL_1_
8.546400e+04	14.32	0.000000e+00	5631	5885	budget._PRL_4_
4.150100e+04	6.95	0.000000e+00	-	-	g_adxi
4.024600e+04	6.74	0.000000e+00	3931	3980	ave._PRL_7_
2.317900e+04	3.88	0.000000e+00	389	454	stcs._PRL_23_
2.265300e+04	3.79	0.000000e+00	543	608	stcs._PRL_21_
1.545800e+04	2.59	0.000000e+00	3880	3914	ave._PRL_9_
1.425400e+04	2.39	3.892000e+03	3621	5454	ave_
8.737000e+03	1.46	0.000000e+00	-	-	_libc_poll

関数性能測定状況（プログラム全体）

CPU	Commit	MIPS	MFLOPS	L2-miss	mTLB-ir	mTLB-or	Cover	
--<8thread>----------								
9.679459e+02	1.082560e+11	1.118410e+02	5.863436e+01	2.9545	0.0000	0.0000	80.0	budgek2._PRL_1_
9.232460e+02	1.080596e+11	1.170432e+02	6.137338e+01	2.8635	0.0000	0.0000	98.7	budgek1._PRL_1_
8.482010e+02	1.038536e+11	1.224398e+02	6.239195e+01	3.6745	0.0000	0.0000	99.4	budget._PRL_4_
4.111113e+02	2.529987e+11	6.154019e+02	9.890186e+01	0.3131	0.0000	0.0000	91.9	g_adxi
3.993641e+02	5.699815e+10	1.427223e+02	2.717652e+01	1.1222	0.0000	0.0000	97.5	ave._PRL_7_
2.302111e+02	3.134895e+10	1.361748e+02	2.204465e+01	3.1923	0.0000	0.0000	98.7	stcs._PRL_23_
2.249198e+02	2.945408e+10	1.309537e+02	2.009890e+01	3.3601	0.0000	0.0000	99.4	stcs._PRL_21_
1.533622e+02	1.140311e+10	7.435412e+01	1.451418e+01	4.0057	0.0000	0.0000	99.4	ave._PRL_9_
1.416606e+02	1.372047e+11	9.685453e+02	7.923003e+01	0.0135	0.0000	0.0000	86.6	ave_
2.235310e+00	2.945278e+09	1.317615e+03	4.221320e-01	0.0076	0.0000	0.0000	0.9	_libc_poll
6.119807e+02	2.270770e+12	3.710525e+03	1.244961e+03	1.1135	0.0000	0.0000	62.0	Process Total

リスト 7.5

実数のべき乗使用例

```
!XOCL SPREAD DO /ISE
      DO 61 J=1,JG
      DO 61 K=1,KG
      DO 61 I=1,IG
       U_RMST(J)=U_RMST(J)+( U(I,J,K,1)-EA_U(J) )**2
       V_RMST(J)=V_RMST(J)+( U(I,J,K,2)-EA_V(J) )**2
       W_RMST(J)=W_RMST(J)+( U(I,J,K,3)-EA_W(J) )**2
      ・・・
       UVT(2,J)=UVT(2,J)+
     &     (((U(I,J,K,1)-EA_U(J))+(U(I-1,J,K,1)-EA_U(J)))*0.5D0)
     &    *(((U(I,J,K,2)-EA_V(J))+(U(I,J-1,K,2)-EA_V(J-1)))*0.5D0)
     &          *(TMPT2)
  61  CONTINUE
!XOCL END SPREAD
```

リスト 7.6

変更前

```
      DIMENSION U11(0:JG),U33(0:JG)
          ・・・
!XOCL LOCAL U11(/ISF),U33(/ISF)
          ・・・
!XOCL SPREAD DO /ISE
p       DO 110 J=1,JG
p       DO 110 K=1,KG
p       DO 110 I=1,IG
C-------------------------------
p         U11(J)=U11(J)+UT(I,J,K,1)**2
p         U33(J)=U33(J)+UT(I,J,K,3)**2
C-------------------------------
          ・・・
p     110 CONTINUE
!XOCL END SPREAD
          ・・・
!XOCL SPREAD DO /ISE
P       DO 210 J=1,JG
P         AK(J)=0.5*(U11(J)+・・・+U33(J))*DDDP
          ・・・
P     210 CONTINUE
!XOCL END SPREAD
```

XPF の分割ループで並列化したため回転数が少ない

変更後

```
      PARAMETER (NPad=8)
      DIMENSION U11(NPad,0:JG),U33(NPad,0:JG)
          ・・・
!XOCL LOCAL U11(:,/ISF),U33(:,/ISF)
          ・・・
!XOCL SPREAD DO /ISE
p       DO 110 J=1,JG
p       DO 110 K=1,KG
p       DO 110 I=1,IG
C-------------------------------K
p         U11(1,J)=U11(1,J)+UT(I,J,K,1)**2
p         U33(1,J)=U33(1,J)+UT(I,J,K,3)**2
C-------------------------------
          ・・・
p     110 CONTINUE
!XOCL END SPREAD
          ・・・
!XOCL SPREAD DO /ISE
P       DO 210 J=1,JG
P         AK(J)=0.5*(U11(1,J)+・・・+U33(1,J))*DDDP
          ・・・
P     210 CONTINUE
!XOCL END SPREAD
```

リスト 7.7

変更前

```
C=================================FOR JA
!XOCL SPREAD DO /ISE
  DO 601 J=JA,JA
   IF (J.EQ.JA) THEN
   DO 201 K=1,KG
      ・・・
 201 CONTINUE
   ENDIF
 601 CONTINUE
!XOCL END SPREAD
C=================================FOR JB
!XOCL SPREAD DO /ISE
  DO 602 J=JB,JB
   IF (J.EQ.JB) THEN
   DO 202 K=1,KG
      ・・・
 202 CONTINUE
   ENDIF
 602 CONTINUE
!XOCL END SPREAD
      ・・・
C=================================FOR JV
!XOCL SPREAD DO /ISE
  DO 611 J=JV,JV
   IF (J.EQ.JV) THEN
   DO 211 K=1,KG
      ・・・
 211 CONTINUE
   ENDIF
 611 CONTINUE
!XOCL END SPREAD
```

J=JA に対応するプロセスのみ動作(残りは同期待ち)

変更後

```
!XOCL SPREAD DO /ISC
  DO 601 J=JA,JV
C=================================FOR J
   IF (J.EQ.JA) THEN
   DO 201 K=1,KG
      ・・・
 201 CONTINUE
C=================================FOR JB
   ELSE IF (J.EQ.JB) THEN
   DO 202 K=1,KG
      ・・・
 202 CONTINUE
      ・・・
C=================================FOR JV
   ELSE IF (J.EQ.JV) THEN
   DO 211 K=1,KG
      ・・・
 211 CONTINUE
C
   ENDIF
 601 CONTINUE
!XOCL END SPREAD
```

JA～JV の各プロセスが同時に動作

リスト 7.8

変更前	変更後

DO 201 K=1,KG
DO 201 I=1,IG
IF(UT(I,JA,K,1).GE.(RPDFU(1,0)))
THEN
PDFN(0,JA,1)=PDFN(0,JA,1)+1.0D0
GO TO 2001
ELSE
L=1
ENDIF
1001 CONTINUE
IF(UT(I,JA,K,1).GE.(RPDFU(1,L)))
THEN
PDFN(L,JA,1)=PDFN(L,JA,1)+1.0D0
GO TO 2001
ELSE
L=L+1
ENDIF
IF(L.LE.NBIN) THEN
GO TO 1001
ELSE
 	
PDFN(NBIN+1,JA,1)=PDFN(NBIN+1,JA,1)+1.0
D0
ENDIF
2001 CONTINUE
201 CONTINUE
————————————————————
jwd5133i-i:この DO ループは構造が複雑なため並列化されません.
	!$omp parallel do reduction(+:pdfn)
	private(i,k,l)
	DO 201 K=1,KG
	DO 201 I=1,IG
	IF(UT(I,JA,K,1).GE.(RPDFU(1,0)))
	THEN
	PDFN(0,JA,1)=PDFN(0,JA,1)+1.0D0
	GO TO 2001
	ELSE
	L=1
	ENDIF
	1001 CONTINUE
	IF(UT(I,JA,K,1).GE.(RPDFU(1,L)))
	THEN
	PDFN(L,JA,1)=PDFN(L,JA,1)+1.0D0
	GO TO 2001
	ELSE
	L=L+1
	ENDIF
	IF(L.LE.NBIN) THEN
	GO TO 1001
	ELSE
	
	PDFN(NBIN+1,JA,1)=PDFN(NBIN+1,JA,1)+1.0
	D0
	ENDIF
	2001 CONTINUE
	201 CONTINUE
	!$omp end parallel do

7.5 スレッド並列チューニング後の性能測定結果

以下の**表 7.9～7.13** に，プロセス数を 4 に固定しスレッド数を変化させた場合について，Tune1 から Tune4 まで段階的に適用した場合の性能測定結果を示す．ここで，「性能向上比」は，1 スレッド実行=1 としたときの性能向上比，「ASIS に対する性能比」は，ASIS の 1 スレッド実行=1 としたときの性能向上比を示す．

表 7.9 ASIS(チューニング無し)

スレッド数	経過時間[sec]	性能向上比	ASIS に対する性能比
1	747.47	1.00	1.00
2	745.18	1.00	1.00
4	614.95	1.22	1.22
8	553.67	1.35	1.35

表 7.10 Tune1 を適用した場合

スレッド数	経過時間[sec]	性能向上比	ASIS に対する性能比
1	595.46	1.00	1.26
2	664.20	0.90	1.13
4	559.61	1.06	1.34
8	506.03	1.18	1.48

表 7.11 Tune1+Tune2 を適用した場合

スレッド数	経過時間[sec]	性能向上比	ASIS に対する性能比
1	600.46	1.00	1.24
2	607.08	0.99	1.23
4	495.52	1.21	1.51
8	453.81	1.32	1.65

表 7.12 Tune1+Tune2+Tune3 を適用した場合

スレッド数	経過時間[sec]	性能向上比	ASIS に対する性能比
1	497.53	1.00	1.50
2	493.78	1.01	1.51
4	356.12	1.40	2.10
8	321.98	1.55	2.32

表 7.13 Tune1+Tune2+Tune3+Tune4 を適用した場合

スレッド数	経過時間[sec]	性能向上比	ASIS に対する性能比
1	425.36	1.00	1.76
2	299.08	1.42	2.50
4	203.61	2.09	3.67
8	164.08	2.59	4.56

図 7.14 は，チューニングによるスレッド並列性能（**表 7.9～7.13** の ASIS に対する性能向上比）を，ASIS の 1 スレッド実行を基準にプロットしたものである．Tune1～Tune3 の効果は，スレッド並列性能に対する貢献としては大きくはないが，表からもわかるように ASIS に比べると素性能は 5 割近く向上しており，チューニングの効果は出ているといえる．Tune2 の False Sharing は，素性能の向上には繋がらないものの，スレッド並列時の不自然な性能低下は回避されている．Tune4 は，スレッド並列の効果を上げるのに大きく役立っているのがわかる．これにより，スレッド並列性能が上がらなかった主たる要因は，スレッド並列化されていないサブルーチンのせいであった．

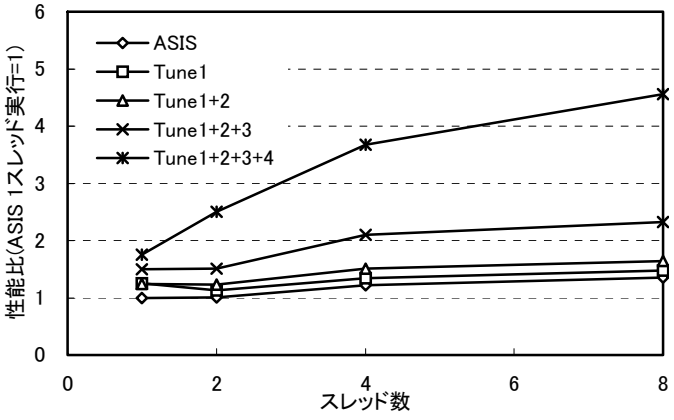


図 7.14 チューニングによるスレッド並列性能

7.6 プロセス並列チューニング

このコード P3 は，XPFortran でプロセス並列化されている．従って，領域分割ではなくループ単位の並列となるので，負荷バランスの不均衡は発生しない．通信によるオーバーヘッドを隠蔽するため，通信と計算をオーバーラップさせるチューニングを施した．

7.7 ハイブリッド並列の性能測定結果

以上のチューニング後に，コード P3 の実際の問題サイズ（=2,048×448×1,536）での並列性能を調べた．スレッド数を固定し，プロセス数を変化させることにより，経過時間を測定した．他のジョブからの影響を避けるために，他のジョブが走っていない状態で測定した．

図 7.15 は，横軸にプロセス数，縦軸には 28 プロセス×1 スレッドのときの性能を基準(=1)としたときの性能比を取ったものを示す．何本かの線は，スレッドを 1,2,4,8,16 と変化させたものに相当する．通信が多いために，プロセス性能の直線性は良くなく，プロセス数が多くなると性能曲線の傾きは寝てくる．一方で，スレッド並列については，プロセス数一定のラインでみると，スレッドが多くなっても一定の割合で性能が向上しているのがわかる．

一方，CPU 数一定で整理した場合を**図 7.16** に示す．これを見ると，特定のプロセス数×スレッド数の組み合わせ（448CPU の場合 56 プロセス×8 スレッド）のときに最も良い性能を示し，**純 MPI の場合に比べ，2 割ほど高い性能**を得ているのがわかる．これは，P3 のような通信が多いコードの場合，プロセス数×スレッド数の組み合わせを適切に選ぶことにより，プログラムに手を加えることなしに性能向上を図ることができることを意味している．このことは，何らかの性能推定モデルがあれば，（プロセス数，スレッド数を任意に変えることができるという条件下で）適切なプロセス数×スレッド数を予め選択できることを示唆しており，ハイブリッド並列の一つのメリットともいえる．

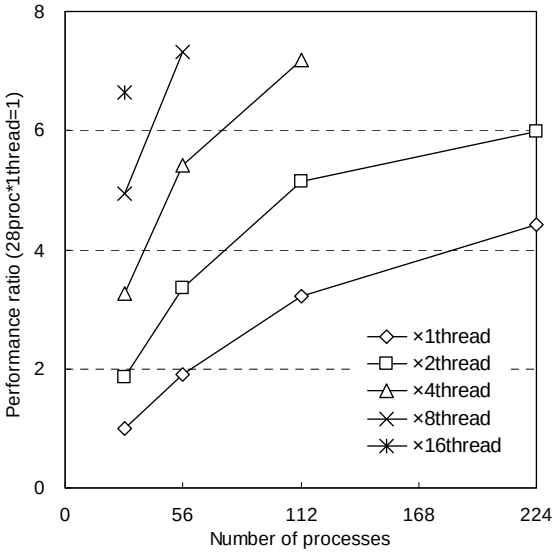


図 7.15 コード P3 のハイブリッド並列性能

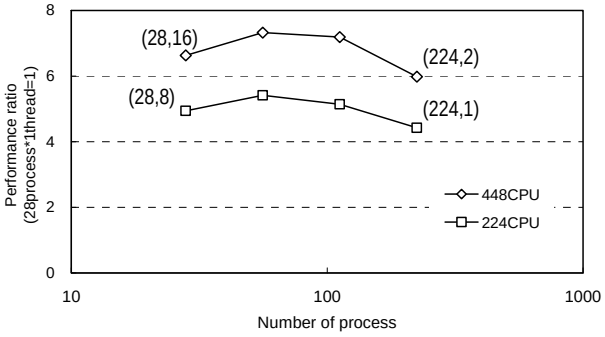


図 7.16 CPU 数一定時のハイブリッド並列性能の比較

7.8 おわりに

本章では，学術系のコードの代表である平行平板間乱流解析コードの並列性能チューニングの概要とハイブリッド並列性能について述べた．スレッド並列性能は，実数のべき乗計算の乗算化，False Sharing が発生する配列の次元追加，ロードバランス不均等なループの融合，並列化率の拡大等により，ASIS に比べ 5 割近く向上させることができた．プロセス並列性能は，通信と計算をオーバーラップさせることで，通信によるオーバーヘッドを隠蔽させた．性能測定の結果，同一 CPU 数であっても，プロセス数とスレッド数の組み合わせにより，性能向上が図れることがわかった．

参考文献

[1] 阿部浩幸，松尾裕一：平行平板間乱流の大規模直接数値シミュレーション，航空宇宙数値シミュレーション技術シンポジウム2004論文集，JAXA特別資料SP-04-012, pp.21-26 (Mar. 2005).

第 8 章　性能評価と性能チューニングの実例（その 2）- CFD 共通基盤コード UPACS

8.1　はじめに

スパコンと CFD の進展とともに，解析を実用形状や複雑形状に適用する必要性が生じてきている．この場合，次の 9 章に示すような非構造格子を使うのも手だが，構造格子の精度の良さ，処理性能（計算速度）の高さというのも捨てがたいものがある．このようなところから，構造格子の一種であるマルチブロック格子を用いて並列化に対応するために開発されたのが，UPACS（Unified Platform for Aerodynamic Conputational SImulation）[1]と呼ばれる汎用 CFD コードである．ここでは，UPACS について，並列性能評価と性能チューニングの実例について示す．

8.2　プログラム概要

UPACS は，中核となる解析ソルバである UPACS ソルバと，解析の前後処理を行う各種ツール，ユーティリティ群からなる CFD 共通基盤環境である．UPACS コードの特徴として，A)拡張性と共有性，B)並列化，等が挙げられる．**表 8.2** に従来の CFD コードと UPACS の相違をまとめた．

A) 拡張性と共有性

(a) オブジェクト指向の考え方を取り入れることで，データ，手続きのカプセル化とプログラム構造の階層化を行った．特に，プログラムを三階層として，本来別の処理であるシングルブロックの解析ソルバ部と，マルチブロック/オーバーセット処理部および並列処理部を分離した（**図 8.1**）．下記に UPACS の階層構造を示す．最下部が単一ブロックの解析ソルバ，中間にマルチブロック/オーバーセット処理を実施する部分，最上部にプログラムの流れを制御する部分となっている．この結果解析ソルバの開発者は並列処理やマルチブロック/オーバーセット処理を考慮する必要がなく，それぞれの専門家による分散した開発が可能となった．

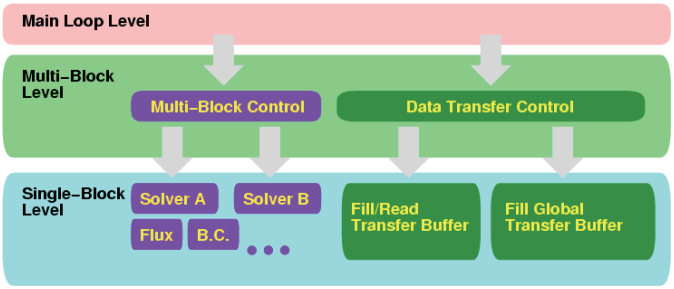


図 8.1　UPACS の階層構造

(b) CFD 研究者による共有化とカプセル化，コードの階層化を実現するため，UPACS は，Fortran90 を用いて開発された．C++など計算科学の新しい道具であるオブジェクト指向型の言語は非常に便利ではあるが，これまでの資産の継承，CFD 研究者の習熟度，更には大型計算機での実効性能と開発環境の実績を考慮すると，C++を用いて開発するのは時期尚早と判断した．一方，伝統的な科学技術用開発言語である Fortran にも Fortran90 になって構造体，ポインタ等，我々の目的を実現するための機能が導入されており，開発言語として Fortran90 を選定した．

B) 並列化

(a) 複雑形状への適用性と解析精度の維持のバランスを保つため，マルチブロック/オーバセット構造格子法を採用している．そのため，マルチブロック/オーバセット構造格子の複数ブロックを並列化の際の領域分割にマッピングすることで並列化を行っている．複数個のブロックが並列処理単位に自由にマッピングできるため，任意の並列度数での解析が簡単に実行できる．

(b) 並列化は MPI を用いたプロセス並列を採用した．並列化には他にも XPFortran を用いたプロセス並列，OpenMP によるスレッド並列などがあるが，並列化されたプログラムの汎用性（移植性）を重視して MPI による並列化を行った．MPI は PC クラスタから大型計算機まで並列計算機なら一般的に利用可能な並列環境であり，移植性を考えると非常に有望である．

表 8.2　従来の CFD コードと UPACS の相違

	従来の CFD コード	UPACS
開発言語	Fortran77	Fortran90 構造体，ポインタ配列…
並列化	データ，手続き並列 VPP-Fortran(XPFortran)	明示的な領域分割 MPI
格子	単一構造格子	複合格子（非構造格子）
行列反転	１行列の反転を並列化（ADI 等を用いた巨大なシングルブロックでの行列反転）	１行列の反転は非並列（マルチブロックでの並列化）
時間積分	定常解析が主	今後は非定常解析が主
ﾃﾞｰﾀ転送	行列の転置などで AlltoAll が必要	ブロック間の陽的なデータ転送で良い

8.3　基本性能

性能評価には UPACS ver.1.3 を使用した．基本的な性能として MPI 並列と OpenMP 並列の組合わせによる SpeedUp（計算量一定で並列数可変）性能計測を行った．**表 8.3** に測定条件，**表 8.4** 及び**図 8.5** に測定結果を示す．その結果，プロセスに比べてスレッドの並列効果が低く，同じ CPU 数ならプロセス/スレッドのハイブリッドより純粋 MPI の効率が良いことがわかった．

表 8.3 基本性能の測定条件

実行環境	機種： 富士通 PRIMEPOWER HPC2500 （SPARC64V 1.3GHz） 使用規模： 32cpu×32node 開発環境： Parallelnavi2.4 （Fujitsu Fortran Compiler V5.6）
並列数	[PureMPI] MPI 並列のみで 1～512 プロセスを使用 [Hybrid] MPI 並列 1～512 プロセスに OpenMP 並列 1～16 スレッドを併用
計算格子	1 ブロックあたり 40×20×80：計 512 ブロック 実行時に各プロセス均等に分散
計算反復回数	2 回
翻訳時ｵﾌﾟｼｮﾝ	[PureMPI] mpifrt -Kfast_GP2=3, V9, largepage=2, hardbarrier -x- [Hybrid] mpifrt -Kfast_GP2=3, V9, largepage=2, OMP, hardbarrier -x-

表 8.4 基本性能の測定結果

経過時間[秒]	Process 数									
	1	2	4	8	16	32	64	128	256	512
PureMPI	1238.4	682.9	-	156.0	77.6	38.9	19.5	9.8	5.0	2.8
×1thread	1360.9	749.4	-	170.5	85.3	42.6	21.3	10.8	5.5	3.1
×2thread	820.7	-	-	102.9	51.3	25.7	13.0	6.5	3.4	-
×4thread	517.3	-	-	65.0	32.3	16.3	8.2	4.2	-	-
×8thread	349.8	-	87.0	44.0	21.9	11.1	5.6	-	-	-
×16thread	288.4	-	71.7	36.3	18.0	9.1	-	-	-	-

性能比 (MPI 1proc.=1)	Process 数									
	1	2	4	8	16	32	64	128	256	512
PureMPI	1.00	1.81	-	7.94	15.95	31.85	63.53	126.38	247.47	444.99
×1thread	0.91	1.65	-	7.26	14.52	29.05	58.03	114.99	223.77	401.19
×2thread	1.51	-	-	12.03	24.14	48.20	95.63	190.23	367.85	-
×4thread	2.39	-	-	19.05	38.31	76.20	151.23	297.83	-	-
×8thread	3.54	-	14.24	28.13	56.65	111.31	222.31	-	-	-
×16thread	4.29	-	17.28	34.13	68.77	136.32	-	-	-	-

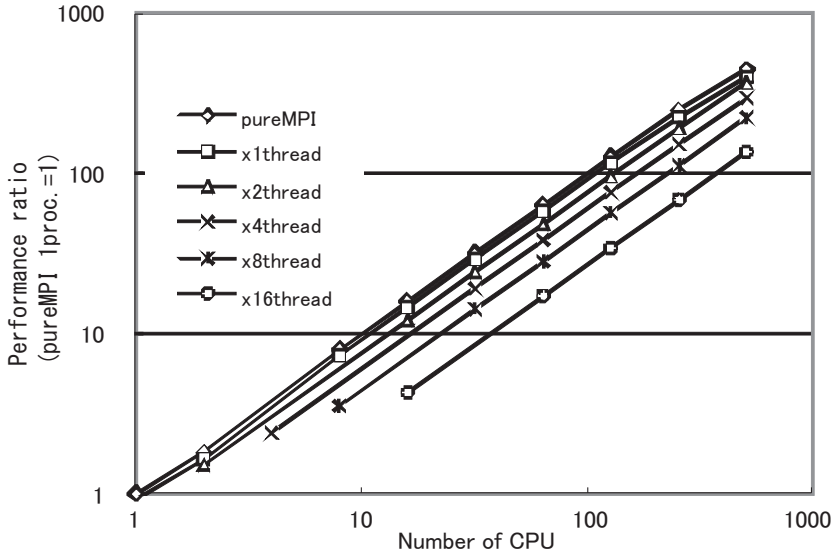


図 8.5 基本性能の測定結果のグラフ

次に，プロファイラを用いて，8 プロセス実行の性能情報を採取した．**表 8.6** に測定条件，**表 8.7** に測定結果を示す．その結果，2 キャッシュミス率や TLB ミス率が高いルーチンが多く，演算性能(MFLOPS)の阻害要因となっていることが判明した．

表 8.6　プロファイラによる性能情報取得の測定条件

実行環境	機種： 富士通 PRIMEPOWER HPC2500 （SPARC64V 1.3GHz） 使用規模： 64cpu×3node 開発環境： Parallelnavi2.4 （Fujitsu Fortran Compiler V5.6）
並列数	MPI 並列 8 プロセス
計算格子	1 ブロックあたり 100×100×100：計 8 ブロック 実行時に各プロセス均等に分散
計算反復回数	5 回
翻訳時ｵﾌﾟｼｮﾝ	mpifrt -Kfast_GP2=3, V9, largepage=2, hardbarrier -x-

表 8.7　プロファイラによる性能情報の取得結果

①プロセス単位

	CPU(Sec)	MIPS	MFLOPS	L2miss(%)	TLBmiss(%)	Cover(%)
Process 0	181.3	667.8	147.9	1.0	0.6	68.5
Process 1	179.1	676.8	149.7	1.0	0.6	69.4
Process 2	177.7	682.0	150.9	1.0	0.6	68.8
Process 3	180.2	671.1	148.7	1.0	0.6	69.8
Process 4	178.6	682.1	150.6	1.0	0.6	69.1
Process 5	178.3	682.0	150.9	1.0	0.6	69.2
Process 6	180.3	672.9	149.2	1.0	0.6	69.8
Process 7	179.0	680.2	150.3	1.0	0.6	69.3
Total	272.7	3560.3	787.8	1.0	0.6	69.2

②ルーチン単位

ルーチン名	Cost(%)	MIPS	MFLOPS	L2miss(%)	TLBmiss(%)	Cover(%)
blk_mfgs.implhs_mfgs_	30.6	478.8	152.3	0.4	3.0	42.0
blk_rhsviscous.cellfacevariables_	15.3	773.9	217.4	0.6	0.9	59.3
blk_rhsconvect.rhs_convect_	6.8	127.3	13.4	13.0	0.0	99.0
blk_flux.flux_roe_	5.1	925.0	295.6	0.3	0.0	99.1
blk_muscl.muscl_co_	4.4	946.4	216.5	0.5	0.0	99.1
blk_metrics.calcmetrics_	4.0	809.6	44.5	0.2	1.0	56.4
blk_rhsviscous.rhs_viscous_	3.7	161.7	24.4	14.3	0.0	99.1
blk_rhsviscous.flux_vis_	3.4	393.4	103.3	1.4	0.0	98.8
blk_dt.calcdt_original_	2.9	769.6	191.1	0.6	2.2	37.9
blk_tm_spalartallmaras.muscl_2ndorder_	2.9	898.4	178.5	0.5	0.0	99.1
blk_tm_spalartallmaras.diffusion_	2.4	1639.7	333.6	0.8	0.0	99.0
jwe_gdgemm	1.8	1743.1	145.7	0.2	0.0	98.4
blk_muscl.muscl_	1.6	147.9	4.8	7.9	0.0	99.0
blk_tm_spalartallmaras.convection_ausm_	1.5	607.1	159.1	2.7	0.0	99.0
blk_metrics.calccellvrtx_	1.4	926.6	0.0	0.3	1.8	38.2

8.4 スカラチューニング

単体 CPU 性能向上のため，データアクセスの効率化を促進する以下のスカラチューニングを実施し，性能を調べた．

8.4.1 チューニング概要

オリジナルソースに対して，**表 8.8** の性能チューニングを段階的に適用した．

表 8.8 実施したスカラーチューニングの概要

項目名	内容
Tune1	配列の軸入替え
Tune2	サブルーチンにまたがるループ融合 ＋ ワーク配列の次元削減

【Tune1 - 配列の軸入替え】

TLB ミス率の高いルーチンでは，配列の最外次元が変化するデータアクセスが多用されており，ストライド幅が大きくなっていた．そこで，最内次元に入れ替えることにより，データアクセスを連続化した．→ **リスト 8.9**

実際の入替えは，ソース変更の代わりに C プリプロセッサマクロを使用して行った．→ **リスト 8.10**

【Tune2 - サブルーチンにまたがるループ融合＋ワーク配列の次元削減】

L2 キャッシュミス率の高いルーチンでは，ソースが複雑なため自動的にループ融合できない箇所があった．そこで，ループ融合した状態にソースを書換えた．→ **リスト 8.11**

また，このループ融合により大きな領域を取る必要がなくなった作業配列については，配列次元数を削減した状態にソースを書換えた．→ **リスト 8.12**

リスト 8.9

ソース変更前

```
subroutine implhs_mfgs(blk,sweepID,cdt,cdiag)
  !
  type(blockDataType),intent(inout) :: blk
 ・・・・・・・・
  real(8), pointer, dimension(:,:,:,:) :: dq_star
  allocate(dq_star(0:blk%in+1,0:blk%jn+1, &
             0:blk%kn+1,bdtv_nFlowVar))
  dq_star(:,:,:,:) = 0.0
 ・・・・・・・・
    do k=is(3),ie(3),istep(3)
      do j=is(2),ie(2),istep(2)
        do i=is(1),ie(1),istep(1)
          rho = blk%q(i,j,k,1)
          rhoi = 1.d0/(rho+epsilon(rho))
          u(:) = blk%q(i,j,k,2:4)*rhoi
 ・・・・・・・・
          nv(:)= blk%fNormal    (i-1,j,k,1,:)
          nt     = blk%fNormal_t(i-1,j,k,1)
          q(:) = q0(:)+dq_star(i-1,j,k,:)
 ・・・・・・・・
          nv(:)= blk%fNormal    (i,j-1,k,2,:)
          nt     = blk%fNormal_t(i,j-1,k,2)
          q(:) = q0(:)+dq_star(i,j-1,k,:)
 ・・・・・・・・
        enddo
      enddo
    enddo
```

ソース変更後

```
subroutine implhs_mfgs(blk,sweepID,cdt,cdiag)
  !
  type(blockDataType),intent(inout) :: blk
 ・・・・・・・・
  real(8), pointer, dimension(:,:,:,:) :: dq_star
  allocate(dq_star(bdtv_nFlowVar,0:blk%in+1, &
             0:blk%jn+1,0:blk%kn+1))
  dq_star(:,:,:,:) = 0.0
 ・・・・・・・・
    do k=is(3),ie(3),istep(3)
      do j=is(2),ie(2),istep(2)
        do i=is(1),ie(1),istep(1)
          rho = blk%q(1,i,j,k)
          rhoi = 1.d0/(rho+epsilon(rho))
          u(:) = blk%q(2:4,i,j,k)*rhoi
 ・・・・・・・・
          nv(:)= blk%fNormal    (:,i-1,j,k,1)
          nt     = blk%fNormal_t(1,i-1,j,k)
          q(:) = q0(:)+dq_star(:,i-1,j,k)
 ・・・・・・・・
          nv(:)= blk%fNormal    (:,i,j-1,k,2)
          nt     = blk%fNormal_t(2,i,j-1,k)
          q(:) = q0(:)+dq_star(:,i,j-1,k)
 ・・・・・・・・
        enddo
      enddo
    enddo
```

リスト 8.10

common.f90inc

```
#if !defined(common_f90inc)
#define common_f90inc

#define q(i,j,k,n)    Q(n,i,j,k)
#define fNormal_t(i,j,k,n)    FNORMAL_T(n,i,j,k)
#define fNormal(i,j,k,n,A)    FNORMAL(A,i,j,k,n)
#define dq_star(i,j,k,n)    DQ_STAR(n,i,j,k)
 ・・・・

#endif /* !defined(common_f90inc) */
```

リスト 8.11

ソース変更前

```
subroutine rhs_viscous(blk)
  type(visCellFaceType), pointer, dimension(:,:,:) :: cface
    ・・・
    if(bv_viscous%fullns) then
      call cellFaceVariables(blk,cface,dir)
    else if(bv_viscous%thinlayer) then
      call cellFaceVars_thinlayer(blk,cface,dir)
    else
      write(6,*) ' error: Unknown viscous term model '
      write(6,*) ' rhs_viscous '
    end if

    call flux_vis(blk,cface,dir)
    call blk_saveBoundaryFlux_viscous(blk,cface,dir)
    ・・・
end subroutine rhs_viscous
```

```
subroutine cellFaceVariables(blk,f,dir)
  ・・・
  do k = isrt(3),iend(3)
    do j = isrt(2),iend(2)
      do i = isrt(1),iend(1)
        ・・・
        f(i,j,k)%nv      = blk%fNormal(i,j,k,ixi,:)
        f(i,j,k)%area    = blk%fArea   (i,j,k,   ixi)

      end do
    end do
  end do

end subroutine cellFaceVariables
```

```
subroutine flux_vis(blk,f,dir)
  ・・・
  do k = isrt(3),iend(3)
    do j = isrt(2),iend(2)
      do i = isrt(1),iend(1)
        ・・・
        f(i,j,k)%flux(1) = 0.
        f(i,j,k)%flux(2:4) =  ・・・
        f(i,j,k)%flux(5) =    ・・・

        f(i,j,k)%flux = -f(i,j,k)%area * f(i,j,k)%flux

      end do
    end do
  end do

end subroutine flux_vis
```

リスト 8.12

ソース変更後

```
  subroutine rhs_viscous(blk)
    type(visCellFaceType), pointer, dimension(:,:,:) :: cface
      ・・・
      if(bv_viscous%fullns) then
        call cellFaceVariables(blk,cface,dir)
      else if(bv_viscous%thinlayer) then
        call cellFaceVars_thinlayer(blk,cface,dir)
      else
        write(6,*) ' error: Unknown viscous term model '
        write(6,*) ' rhs_viscous '
      end if

!      call flux_vis(blk,cface,dir)
      call blk_saveBoundaryFlux_viscous(blk,cface,dir)
      ・・・
  end subroutine rhs_viscous
```

flux_vis 全体を融合

```
  subroutine cellFaceVariables(blk,f,dir)
    real(8), dimension(3)     :: f_dTdx,f_u,f_nv
    real(8) :: f_mu,f_mu_t,f_area
    ・・・
    do k = isrt(3),iend(3)
      do j = isrt(2),iend(2)
        do i = isrt(1),iend(1)
          ・・・
          f_nv      = blk%fNormal(i,j,k,ixi,:)
          f_area    = blk%fArea   (i,j,k,   ixi)

          ・・・
          f(i,j,k)%flux(1) = 0.
          f(i,j,k)%flux(2:4) =  ・・・
          f(i,j,k)%flux(5) =    ・・・

          f(i,j,k)%flux = -f_area * f(i,j,k)%flux

        end do
      end do
    end do

  end subroutine cellFaceVariables
```

別ループに渡すため，(i, j, k)座標の情報を全て保存していたのが，ループ融合で不要になりスカラ変数化した

元のループ

flux_vis から移したループ

8.4.2 性能測定

「8.3 基本性能」と同じ測定条件で，以下（**表 8.13**）の 3 パターンの性能を測定した（**表 8.14，図 8.15**）．

表 8.13 測定パターン

パターン名	内容
Original	オリジナルソース
Tune1	Original に Tune1 を適用したソース
Tune1+2	Tune1 に Tune2 を適用したソース

表 8.14 性能測定結果

MPI プロセス数	実行時間[秒]			性能比(Original 1proc.=1)		
	Original	Tune1	Tune1+2	Original	Tune1	Tune1+2
1	2048.4	1211.8	1037.7	1.00	1.69	1.97
2	1059.6	635.1	532.6	1.93	3.23	3.85
4	538.8	325.4	278.1	3.80	6.30	7.37
8	280.5	177.1	148.5	7.30	11.57	13.80

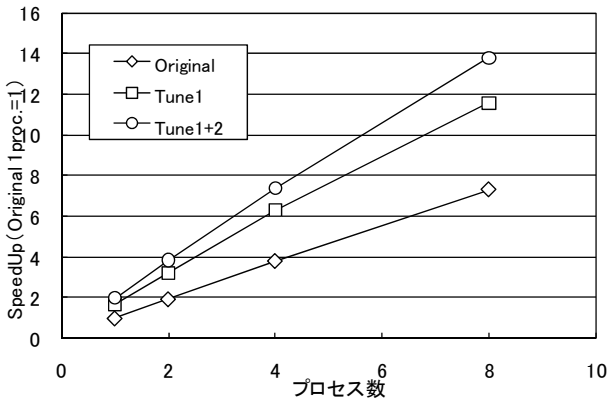


図 8.15 測定した性能のグラフ

2 段階のスカラチューニングにより，オリジナルソースから約 2 倍の性能向上が得られた．測定パターンごとに，8 プロセス実行のプロファイラ情報を採取した結果，L2 キャッシュミス率や TLB ミス率の改善に応じて，全体の演算性能も改善されていることがわかった．チューニング後も L2 キャッシュミス率の高い箇所がいくつか残っているが，これらの中にはプログラム構造が複雑なため有効なループ融合が出来なかったルーチンも含まれている．強制的に融合するにはアルゴリズムの変更が必要なため，今回は対象外とした．さらに，プロファイラを用いて以下の詳細情報を計測した．

【コスト比率】実行時間ベースのコスト分布とその中を占めるメモリアクセス時間(MEM)およびそれ以外の命令処理時間(CPU)の比率

【命令数比率】発行命令数における以下の命令の割合
Load/Store 命令(Ld/St)，浮動少数点演算命令(Float)，プリフェッチ命令(Pref)，分岐命令(Branch)，その他命令(Other)

【実効性能】命令数情報とコスト情報から算出した，MIPS 値および MFLOPS 値の情報

計測結果を**表 8.16** に示す．コスト比率では CPU 時間（61%）がメモリアクセス時間（39%）に比べて高いのに対し，命令数比率では Float の割合（22.4%）が少なかった．ポインタや構造体のアドレス計算など，その他の命令数の割合が多いため MFLOPS 値が向上しないと考えられる．

表 8.16 詳細情報のプロファイラによる測定結果

コスト比率	MEM	CPU			
	39%	61 %			
命令数比率	Lt/St	Float	Pref	Branch	Other
	42.9%	22.4%	1.8%	5.0%	27.9%
実効性能	MIPS	MFLOPS			
	937.8	210.9			

8.5 自動並列化

自動並列化オプションを追加して翻訳した場合の並列化状況を調査した．また，自動並列化のソース解析能力を比較するため，富士通株式会社の協力を得て，ベクトル機での自動ベクトル化状況も併せて調査した（**表 8.17**）．

表 8.17 測定環境一覧

	自動並列化	自動ベクトル化
機種	富士通 PRIMEPOWER HPC2500 (SPARC64V 1.3GHz)	富士通 VPP5000
言語環境	Parallelnavi2.4 (Fujitsu Fortran Compiler V5.6)	UXP/V Fortran V20L20
翻訳時ｵﾌﾟｼｮﾝ	mpifrt -Kfast_GP2=3,V9,largepage=2,hardbarrier -x--Kparallel,reduction	-Pa -Wv,-m3
使用ﾌﾟﾛｸﾞﾗﾑ	UPACS （前回報告の Tune1+2 版） ※自動並列化や自動ベクトル化を促進するための追加変更は行っていない．	

8.5.1 調査方法

並列化/ベクトル化状況を調査するにあたり，コンパイラが出力するメッセージ数を単純にカウントする方法だけでは，以下の問題が考えられる．

- 並列化の規模が判りにくい（外側の大きなループでも内側の小さなループでもカウント数は同じ）．
- 並列化とベクトル化の比較が難しい（並列化は外側から，ベクトル化は内側からの解析で軸が異なる場合がある）．

そこで，軸になった DO ループそのものではなく，階層構造の末端にある最内ループ（以下の**リスト 8.18** の点線範囲）が並列化あるいはベクトル化されたかどうかをカウント対象とした．

リスト 8.18

```
DO K=KID(1),KID(2)
  DO J=1,JM
    DO I=1,IM
      A(I,J,K)= …
      …
    END DO
    DO I=1,IM
      B(I,J,K)= …
      …
    END DO
  END DO
  DO J=1,JM
    C(J,K)= …
    …
  END DO
END DO
DO I=1,IM
  D(I,KM)= …
END DO
```

自動並列化/自動ベクトル化のコンパイルリストをそれぞれ出力し，並列化やベクトル化の軸の内部に含まれる最内ループ数をカウントする．例えば下記**リスト 8.19** の場合，自動並列化と自動ベクトル化で軸になる DO ループは異なるが，最内ループのカウント数はどちらも 2 となる．

リスト 8.19

【自動並列化リスト】

```
      DO K=KID(1),KID(2)
p       DO J=1,JM
p         DO I=1,IM
p           A(I,J,K)= …
p           …
p         END DO
p
p         DO I=1,IM
p           B(I,J,K)= …
p           …
p         END DO
p       END DO
        DO J=1,JM
          C(J,K)= …
          …
        END DO
      END DO
```

【自動ベクトル化リスト】

```
      DO K=KID(1),KID(2)
        DO J=1,JM
v         DO I=1,IM
v           A(I,J,K)= …
v           …
v         END DO

v         DO I=1,IM
v           B(I,J,K)= …
v           …
v         END DO
        END DO
        DO J=1,JM
          C(J,K)= …
          …
        END DO
      END DO
```

8.5.2 調査結果

(a) 全体情報

上記で測定した 8 プロセス並列（スレッド並列無し）実行コストの上位ルーチンを対象に集計したところ，**表 8.20** のように，自動並列化/自動ベクトル化ともに最内ループ数はゼロとなった．

以下，UPACS のコスト上位 5 ルーチンについて，ループ構造と並列化阻害要因を調べた．代表例としてサブルーチン blk_mfgs.implhs_mfgs_のコンパイルリストから抜粋したループ構造とメッセージ情報を**リスト 8.21** に示す．下線部の DO ループは，いずれもループ内部のポインタ引用が自動並列化の制約となっている．

表 8.20　実行コスト上位ルーチンの自動並列化/自動ベクトル化最内ループ数

サブルーチン名	HPC2500 8 プロセス 実行コスト	サブルーチン内 最内ループ数	HPC2500 自動並列化 最内ループ数	VPP5000 自動ベクトル化 最内ループ数
blk_mfgs.implhs_mfgs_	14.0%	2	0	0
blk_rhsviscous.cellfacevariables_	10.8%	1	0	0
blk_muscl.muscl_co_	10.2%	1	0	0
blk_flux.flux_roe_	10.1%	1	0	0
blk_tm_spalartallmaras.muscl_2ndorder_	7.8%	2	0	0
blk_tm_spalartallmaras.diffusion_	5.4%	2	0	0
blk_rhsconvect.rhs_convect_	5.1%	2	0	0
blk_muscl.minmod_co_	2.4%	0	0	0
blk_rhsviscous.rhs_viscous_	2.3%	2	0	0
blk_metrics.calcmetrics_	1.6%	2	0	0
top_timeint.implicit_onestep_	1.5%	0	0	0
blk_tm_spalartallmaras.production_destruction_	1.4%	1	0	0
blk_tm_spalartallmaras.lhs_gaussseidel_	1.4%	3	0	0
blk_tm_spalartallmaras.vanalbada_	1.3%	0	0	0
blk_tm_scalar_measure.vorticity_	1.3%	1	0	0
blk_dt.calcdt_original_	0.9%	1	0	0
上位ルーチン合計	77.6%	21	0	0

リスト 8.21

blk_mfgs.implhs_mfgs_：コンパイルリスト（抜粋）

```
    20                          subroutine implhs_mfgs(blk,sweepID,cdt,cdiag)
    21                            !       Matrix Free Gauss-Seidel (MFGS) method by E. Shima (KHI)          *
    22                            !
    23                            type(blockDataType),intent(inout) :: blk
    24                            integer,intent(in) :: sweepID
    25                            real(8),intent(in) :: cdt,cdiag
    26
    27                            real(8), pointer, dimension(:,:,:,:) :: dq_star
     •
     •
     •
    54                            allocate(dq_star(0:blk%in+1,0:blk%jn+1,0:blk%kn+1,bdtv_nFlowVar))
    55              p u           dq_star(:,:,:,:) = 0.0
    56                            imax(1)=blk% in ; imax(2)=blk% jn ; imax(3)=blk% kn
    57
    58                            n = sweepID
    59          1                 do ifb = 1,2
     •
     •
     •
    72          2                   do k=is(3),ie(3),istep(3)
    73          3                     do j=is(2),ie(2),istep(2)
    74          4                       do i=is(1),ie(1),istep(1)
    75          4                         rho = blk%q(i,j,k,1)
    76          4                         rhoi = 1.d0/(rho+epsilon(rho))
    77          4       u                 u(:) = blk%q(i,j,k,2:4)*rhoi
    78          4                         p       = blk%p(i,j,k)
    79          4                         c       = sqrt(abs(GAMMA*p*rhoi))
    80          4
    81          4       u                 uu_ui = abs(dot_product(u(:),blk%fNormal(i   ,j   ,k   ,1,:)) + blk%fNormal_t(i   ,j   ,k   ,1))
     •
     •
     •
   238          4       u                 dq0(:) = dq_star(i,j,k,:)
   239          4     p u                 dq_star(i,j,k,:) = (dh*df(:)*blk%inv_vol(i,j,k) + blk%dq(i,j,k,:))*inv_diagonal
   240          4
   241          4       u                 ddq(:) = dq_star(i,j,k,:) - dq0(:)
   242          4                         if(abs(ddq(1)) > 1.D5) dq_star(i,j,k,1) = dq0(1)
   243          4                         if(abs(ddq(2)) > 1.D5) dq_star(i,j,k,2) = dq0(2)
   244          4                         if(abs(ddq(3)) > 1.D5) dq_star(i,j,k,3) = dq0(3)
   245          4                         if(abs(ddq(4)) > 1.D5) dq_star(i,j,k,4) = dq0(4)
   246          4                         if(abs(ddq(5)) > 1.D5) dq_star(i,j,k,5) = dq0(5)
   247          4
   248          4                       enddo
   249          3                     enddo
   250          2                   enddo
   251          1
   252          1                 end do
     •
     •
     •
   267                            deallocate(dq_star)
   268
   269                          end subroutine implhs_mfgs
     •
     •
     •
Module subprogram name(implhs_mfgs)
 jwd5101i-i   "blk_mfgs.f90", line 59: DO ループ内に,自動並列化の制約となる文が存在します.
 jwd5101i-i   "blk_mfgs.f90", line 72: DO ループ内に,自動並列化の制約となる文が存在します.
 jwd5101i-i   "blk_mfgs.f90", line 73: DO ループ内に,自動並列化の制約となる文が存在します.
 jwd5101i-i   "blk_mfgs.f90", line 74: DO ループ内に,自動並列化の制約となる文が存在します.
```

また，ユーザ定義の関数呼び出しを含んでいる場合，DO 変数がモジュール内のデータ実体である場合も性能阻害要因となっている．これらコスト上位 5 ルーチンに共通する，ループ内ポインタ引用の自動並列化について，現在のコンパイラの対応状況および回避方法は以下の通りである．

【機能改善について】

コンパイラでポインタの振る舞いを完全に解析することは不可能であり，汎用的な自動並列化は対応困難．もしポインタを使わなくても書ける処理内容であれば，以下の回避方法による改善の可能性がある．

【回避方法】

次の 2 種類の方法がある．

1) ループ内のポインタ変数同士に領域の重なりが無い場合，ディレクティブ(!ocl noalias)あるいは翻訳時オプション(-Knoalias)で指示することにより，自動並列化が促進される場合がある（効果があるかどうかはプログラム依存）．
2) ソース修正により，配列ポインタを割付配列(allocatable)または形状明示配列（F77 の整合配列）などに置き換える．

そこで実際に，今回のソースについて，1)の翻訳時オプション（自動並列：-Knoalias，自動ベクトル：-Wv,-noalias）を追加して翻訳したところ，**表 8.22** のように，自動並列化/自動ベクトル化ともに最内ループ数は増加した．ただし，新たに並列化/ベクトル化されたのは，比較的小規模のループであり，コスト比率の高い大規模ループには変化が無かった．

表 8.22　実行コスト上位ルーチンの自動並列化/自動ベクトル化最内ループ数の変化

サブルーチン名	HPC2500 8 プロセス 実行コスト	サブルーチン内 最内ループ数	HPC2500 自動並列化 最内ループ数	VPP5000 自動ベクトル化 最内ループ数
blk_mfgs.implhs_mfgs_	14.0%	2	0 → 1	0 → 1
blk_rhsviscous.cellfacevariables_	10.8%	1	0	0
blk_muscl.muscl_co_	10.2%	1	0	0
blk_flux.flux_roe_	10.1%	1	0	0
blk_tm_spalartallmaras.muscl_2ndorder_	7.8%	2	0	0 → 1
blk_tm_spalartallmaras.diffusion_	5.4%	2	0	0
blk_rhsconvect.rhs_convect_	5.1%	2	0 → 1	0 → 1
blk_muscl.minmod_co_	2.4%	0	0	0
blk_rhsviscous.rhs_viscous_	2.3%	2	0 → 1	0 → 1
blk_metrics.calcmetrics_	1.6%	2	0	0
top_timeint.implicit_onestep_	1.5%	0	0	0
blk_tm_spalartallmaras.production_destruction_	1.4%	1	0	0 → 1
blk_tm_spalartallmaras.lhs_gaussseidel_	1.4%	3	0	0 → 3
blk_tm_spalartallmaras.vanalbada_	1.3%	0	0	0
blk_tm_scalar_measure.vorticity_	1.3%	1	0	0
blk_dt.calcdt_original_	0.9%	1	0	0
上位ルーチン合計	77.6%	21	0 → 3	0 → 8

(b) ポインタ引用の変更

そこで，自動並列化の阻害要因と考えられるポインタ引用の書き換えを行なった．実行コスト上位ルーチンを対象に，ソース中で配列のポインタ引用が使われている箇所を，同じ動的割当て方式で最適化への制約が少ないと想定されるアロケータブル配列に書き換えた．単独で宣言されている配列の場合，**リスト 8.23** 下線部のように，宣言文の pointer 属性を，target + allocatable 属性に変更した．ま，た構造型の成分として宣言されている配列の場合，Fortran の仕様により，構造型の成分には target 属性を指定できないため，**リスト 8.24** のように allocatable 属性に変更した．なお，今回の変更に関して大半の実行文は変更不要であるが，別のポインタに代入される箇所については，target 属性あるいは pointer 属性が無いと翻訳時エラーになるが，今回の調査ではコスト上位に含まれないため対象外とした．

変更後に実行コスト上位ルーチンを対象に集計すると，**表 8.25** に示したように自動並列化ループ数が前回に比べて 3 箇所増加したが，コストの大部分を占めるサブルーチンには変化がなかった．ほかにも自動並列化の阻害要因が含まれている可能性が考えられるが，コンパイラの出力メッセージ上では変化が見られないため，オブジェクト内部レベルの調査が必要と考えられる．

リスト 8.23

書き換え前（配列ポインタ）

```
type(cellFaceType),dimension(:,:,:),pointer:: cface
integer :: ii,jj,kk
allocate(cface(-1:blk%in+1, -1:blk%jn+1, -1:blk%kn+1))

do kk=-1,blk%kn+1
   do jj=-1,blk%jn+1
      do ii=-1,blk%in+1
         cface(ii,jj,kk)%area = 0.0
         cface(ii,jj,kk)%nt       = 0.0
               ・ ・ ・
      enddo
   enddo
enddo
```

書き換え後（アロケータブル配列）

```
    type(cellFaceType),dimension(:,:,:),target,allocatable::
cface
    integer :: ii,jj,kk
    allocate(cface(-1:blk%in+1, -1:blk%jn+1, -1:blk%kn+1))

    do kk=-1,blk%kn+1
       do jj=-1,blk%jn+1
          do ii=-1,blk%in+1
             cface(ii,jj,kk)%area = 0.0
             cface(ii,jj,kk)%nt       = 0.0
                   ・ ・ ・
          enddo
       enddo
    enddo
```

リスト 8.24

書き換え前（配列ポインタ）	書き換え後（アロケータブル配列）
type blocckDataType ・・・ real(8),pointer,dimension(:,:,:) :: inv_vol real(8),pointer,dimension(:,:,:,:,:):: fNormal,xix ・・・ end type blockDataType	type blockDataType ・・・ real(8),allocatable,dimension(:,:,:) :: inv_vol real(8),allocatable,dimension(:,:,:,:,:):: fNormal,xix ・・・ end type blockDataType

表 8.25 変更後の自動並列化最内ループ数

サブルーチン名	HPC2500 8プロセス 実行コスト	サブルーチン内 最内ループ数	自動並列化最内ループ数	
			書き換え前（前回の結果）	書き換え後（今回の結果）
blk_mfgs.implhs_mfgs_	14.0%	2	1	1
blk_rhsviscous.cellfacevariables_	10.8%	1	0	0
blk_muscl.muscl_co_	10.2%	1	0	0
blk_flux.flux_roe_	10.1%	1	0	0
blk_tm_spalartallmaras.muscl_2ndorder_	7.8%	2	0	0→1
blk_tm_spalartallmaras.diffusion_	5.4%	2	0	0
blk_rhsconvect.rhs_convect_	5.1%	2	1	1→2
blk_muscl.minmod_co_	2.4%	0	0	0
blk_rhsviscous.rhs_viscous_	2.3%	2	1	1→2
blk_metrics.calcmetrics_	1.6%	2	0	0
top_timeint.implicit_onestep_	1.5%	0	0	0
blk_tm_spalartallmaras.production_destruction_	1.4%	1	0	0
blk_tm_spalartallmaras.lhs_gaussseidel_	1.4%	3	0	0
blk_tm_spalartallmaras.vanalbada_	1.3%	0	0	0
blk_tm_scalar_measure.vorticity_	1.3%	1	0	0
blk_dt.calcdt_original_	0.9%	1	0	0
上位ルーチン合計	77.6%	21	3	6

8.6 おわりに

三次元圧縮性流体解析プログラム UPACS について，スカラチューニングを実施した．その結果オリジナルに比べて約 2 倍程度の速度向上が得られた．本プログラムは，コスト比率では CPU 時間の割合(61%)がメモリアクセス時間(39%)に比べて比較的高いのに対し，命令数比率では Float の割合が少なく 22.4%程度であった．ポインタや構造体のアドレス計算など他の命令数の割合が多いため FLOPS 値が向上しないことが判明した．

自動並列化の阻害要因に関して調査を行なった．ループ内部のポインタ引用が阻害要因となっているが，アロケータブル配列に変更することで自動並列化が適用されたループが 3 から 6 に増加したが，コストの大部分を占めるルーチンに関しては改善は見られなかった．オブジェクト内部レベルでの調査が必要と考えられる．

参考文献

[1] Takaki, R., Yamamoto, K., Yamane, T., Enomoto, S. and Mukai, J.: The Development of the UPACS CFD Environment, *High Performance Computing, Proc. ISHPC2003*, LNCS 2858, pp.307-319 (Oct. 2003).

第 9 章　性能評価と性能チューニングの実例（その 3）- 3 次元ハイブリッド非構造格子 Euler／Navier-Stokes コード JTAS

9.1 はじめに

JAXAでは現在，小型超音速実験機や国産航空機に関するプロジェクトが進められているが，これらのプロジェクトにおいては，複雑形状の回りにおける複雑な流れ場に対するCFD解析技術が求められている．このような解析には非構造格子法が良く用いられる．JAXAにおいて現在用いられる非構造格子ソルバの一つにJTASがある[1]．JTASは，東北大学で開発された TAS(Tohoku university Aerodynamic Simulation code)[2]をもとに，CeNSSに適合するように若干の変更が加えられたコードであり，オリジナルのTASと区別する意味でJTASと呼ばれている．しかし，その変更は配列の次元入れ替え等限定的なものであり，CeNSSの性能を十分有効に活用できていない．そこで，コンパイラによる自動並列化のより効率的な活用を促進することでCeNSSに対する適合性を向上させることを目的として，より内容に踏み込んだ変更を行った．また，テストデータを用いた性能測定を実施し，変更による性能向上を確認した．

9.2 JTAS 概要

性能評価用には，支配方程式として 3 次元Euler方程式を用いた．JTASではこれを，空間方向にはセル節点有限体積法により，時間方向にはEuler陰解法により離散化し，時間積分にはLU-SGS陰解法を用いて計算する．流束の評価は，HLLEW法により行われる．高次精度差分における制限関数としては，Venkatakrishnanの制限関数が用いられている．詳細は文献[3]を参照されたい．

JTASは，元来ベクトル計算機用に開発されているため，LU－SGS法の適用にあたっては非構造格子に適用可能な超平面（Hyper plane）を構成し再帰参照を回避している．ところで，有限体積法において，流束ベクトルは，各検査体積を構成する面ごとに求める必要がある．一方で，ある面は異なる二つの検査体積の境界を構成するのであるから，その面を通る流束はその二つの検査体積に対して同じ量でかつ符号が反対であるように寄与することになる．従って，流束ベクトルを検査体積ごとにそれを構成する全ての面に対して計算することは，流束ベクトルを二重に計算することとなり効率的ではない．面ごとに計算すれば流束ベクトルの計算を最小限に押さえることが出来る．ここで，セル節点体積法の場合，検査体積がその唯一含む節点の番号で指定されるのと同様に，面はそれが唯一含む辺の番号で指定されるので，実際のプログラムにおける流束の計算は，辺IEが含む両端の節点の番号N1 及びN2 を保持する配列の名前を"NEDG2ND"とした場合，例えば**リスト 9.1** のようになる．

リスト 9.1　流束計算の例

```
    DO 100 IE = 1, Nedges
       N1=NEDG2ND(1,IE)
       N2=NEDG2ND(2,IE)
       HLLEW=FLUX_FUNC(Same arguments required)
       FLUX(N1)=FLUX(N1)+HLLEW
       FLUX(N2)=FLUX(N2)-HELLW
100 CONTINUE
```

ここで，このDOループは配列"FLUX"に対して再帰参照になっていることに注意されたい．なぜならば，検査体積は複数の面から構成されているため，異なる面（辺）IEにおいて同じ検査体積番号（節点番号）がN1 またはN2 に表れるからである．

JTASでは，この再帰参照を回避しベクトル計算を行うため，色分け（Coloring）によるグループ化の手法が用いられている．これは，ある検査体積（節点）に含まれる全ての面（辺）は必ず別の色を持つように前もって色分けしておき，DOループ内では同じ色を持った面（辺）のみを計算することにより，再帰参照を避ける方法である．この場合，実際のプログラムは，例えば**リスト 9.2** に示すような二重ループになる．外側のDOループが全ての色に対する処理のループであり，内側のDOループがその内のある一つの色に対する処理を行うDOループである．色分けを適切に行うことにより，内側のループで一度に処理される面（辺）は必ず異なる検査体積（節点）を構成するものとなり，再帰参照が回避されベクトル化される．

リスト 9.2　流束計算のベクトル化例

```
      DO 100 IC = 1, MAX_Colors
*VOCL LOOP,NOVREC
      DO 110 IE = 1, Num_edges(IC)
         N1=NEDG2ND(1,IE,IC)
         N2=NEDG2ND(2,IE,IC)
         HLLEW=FLUX_FUNC(Some arguments required)
         FLUX(N1)=FLUX(N1)+HLLEW
         FLUX(N2)=FLUX(N2)-HELLW
  110 CONTINUE
  100 CONTINUE
```

同様のベクトル化手法は，速度，圧力，密度及び温度の勾配（Gradient）の計算（以下，単に勾配の計算という），制限関数の計算，LU－SGS法における計算の一部においても使用されている．ただし，勾配の計算においては，計算が面（辺）ごとではなく要素毎に行われることが異なる．

JTASでは，MPIを利用したプロセス並列化もなされている．プロセス並列化を行うにあたっては，まず計算に先立って各プロセスに割り当てるために格子空間を領域分割し，この分割された領域をそれぞれのプロセスが分担して計算する．

9.3 計算性能最適化のための変更

全体の性能及びスレッド並列における性能向上を目的にJTASに加えた変更内容は，以下の二点である．

(1) LU－SGS法における節点番号の付け替え

(2) 色分けの削除及びDOループの分割

以下，この変更を加えたJTASをスレッド版JTAS，または単にスレッド版という．より具体的な変更内容を以下に示す．

9.3.1 LU-SGS法における節点番号の付け替え

JTASでは非構造格子にLU－SGS法を適用するために超平面が構成されている．ところが，節点における各物理量等を配列に保存する際には，格子生成時に付されたオリジナルの節点の番号でインデックスされる場所に保存されている．このような方法は，或る一つの超平面内に存在する節点の番号が不連続であるため，効率的なメモリアクセスを疎外する要因となることが予想された．そこで，LU－SGS法の計算を行う部分では，ハイパー面を考慮した節点番号の付け替えを行うこととし，新たな節点番号として超平面内でオリジナルの節点番号の昇順に連続な番号を与えた．この際，LU－SGS 法に関係する部分以外では従来通りの番号付けとし，LU－SGS法で必要となる保存量ベクトル等のデータは，従来の番号付けで保存されている配列から新たな番号付けで保存される配列に一旦コピーした後LU－SGS法の計算を行い，新しい時間ステップでの値を計算する際に従来の番号付けの配列に戻す方法をとった．

9.3.2 色分けの削除及びDOループの分割

オリジナルのJTASは，既にベクトル化されているため一切の変更を加えることなしに，FORTRAN の持つ自動並列化機能によりスレッド並列化することが可能である．ところで，ベクトル化は基本的に最内側DOループに対してなされる．一方で，スレッド並列化では，スレッド生成のオーバーヘッドをなるべく小さくするために，スレッド生成回数の少ない，より外側の DO ループで並列化することが望ましい．色分けによるベクトル化では，例えばリスト 9.2 において，内側のDOループであるDO 110がベクトル化，即ちスレッド並列化されることとなり，スレッド生成のオーバーヘッドによる性能低下が予想される．加えて，スレッド並列化されるDOループの回転数は，生成されたスレッドの中でなるべく多くの演算が行われるように，ベクトル化における場合同様なるべく多い方が望ましいが，色分けによるベクトル化では，一度に計算されるのは一つの色に属する辺（勾配の計算の場合要素）のみであり，全ての辺を一度に処理するのに比べて性能が低下することが予想される．一方で，全ての辺について同時に計算することにすれば，リスト9.1に示すようにDOループは一重ループとなり，外側かつ回転数の多い理想的なループの構成となるが，再帰参照を含むため，ベクトル化もスレッド並列化も行うことが出来ない．

そこで，**リスト 9.3** に示すように色分けの削除をすると共にDOループの分割を行うことで，色分けによる方法で問題になると予想される点の改善を図った．**リスト 9.3** は，流束ベクトルの計算を簡略化した例であり，DO 100 において辺ごとに計算された流束ベクトルは，一旦，作業用配列"EDG_WK"に辺ごとの値として保存された後，DO 110 において，節点（検査体積）ごとの配列"FLUX"に足し込まれている．ここで，DO 100 における配列"EDG_WK"及びDO 110 における配列"FLUX"は，インデックス参照ではなく直接参照となっていることに注意されたい．これにより，メモリアクセスの効率化も期待される．

尚，この変更は，LU－SGS法の計算に対しては適用しなかった．これは，色分けを行った方が良い性能が得られたためである（「9.6.5 LU－SGS法の計算における測定結果及び評価」参照）．

リスト 9.3 流束計算の変更例

```
    DO 100 IE = 1, Nedges
       HLLEW=FLUX_FUNC(Some arguments required)
       EDG_WK(IE)=HLLEW
100 CONTINUE
    DO 110 N = 1, Nnode
       NEDGE = IEDGE_in_NODE(0,N)
       DO 111 IE=1,NEDGE
          IEDGE=IEDGE_in_NODE(IE,N)
          XSIGN=SIGN(1.0D0,IEDGE)
          IEDGE=ABS(IEDGE)
          FLUX(N)=FLUX(N) + XSIGN*EDG_WK(IEDGE)
111    CONTINUE
110 CONTINUE
```

9.4 計算性能測定条件

表 9.4 に使用したコンパイラ，リンケージエディタ及びMPIライブラリのバージョン情報を示す．また，**表 9.5** に翻訳時に指定したコンパイルオプションを示す．

表 9.4 ソフトウエアバージョン情報

ソフトウエア	バージョン情報
コンパイラ	Fujitsu Fortran Version 5.6
リンケージエディタ	Software Generation Utilities Solaris Link Editors: 5.8-1.296
MPIライブラリ	MPI Patchlevel 2.21.0 MPLib version MPLIB-sr2.3.1

表 9.5 指定したコンパイルオプション一覧

コンパイルオプション
-Umpi -Qi,p -NRtrap -Kparallel -Kvppocl -Et

9.5 JTAS 実行条件

9.5.1 初期条件

性能測定のために設定した初期条件を**表 9.6** に示す.

表 9.6 性能測定に使用した主な初期条件

設定データ	設定値	
時間積分ステップ数	500 ステップ	
クーラン数	1.0×10^{5}	
迎角	0 度	
レイノルズ数	1.0×10^{6}	
比熱比	1.4	
自由流温度	273.15 K	
壁面温度	0.5	
自由流マッハ数	0.8	
流れの種別	層流	
空力係数計算	あり	
境界流入条件	初期密度	1.0
	流入速度(x 方向)	9.5×10^{-1}
	初期圧力	1.0
	初期渦動粘性係数	20.0
	非スリップ境界	あり

9.5.2 格子点数

性能測定のために設定した計算格子点の数を**表 9.7** に示す. MPI による並列実行時には, 分割された領域の間で情報の授受を行うために余分の格子点が付与される．ここでは，この情報授受のためにオーバーラップして設けられた格子点数が示されている．実際に解析が行われる格子点数は「正味」として示した.

表 9.7 性能測定に使用した計算格子点の数

要素名	要素			辺	格子点
	三角錐	三角柱	四角錐		
2 process					
proc.0	716,199	0	0	861,171	128,802
proc.1	481,662	141,636	879	870,779	160,656
小計	1,197,861	141,636	879	1,731,950	289,458
合計			1,340,376	1,731,950	289,458
正味					
小計	1,173,840	141,636	879	1,690,955	280,969
合計			1,316,355	1,690,955	280,969

9.6 計算性能測定結果及び評価

以下に, JTAS の性能測定結果及びその評価について示す. 測定は, MPI による並列化におけるプロセス数を 2 で固定とし，スレッド並列については 1 プロセス当たりのスレッド数 1，2，4 及び 8 の 4 ケースについて行った．なお，測定は他のジョブも存在する通常の運用状態の元で行った．このため, 特に経過時間の測定結果には変動要因が含まれていることに注意されたい．以下，先ず全体の測定結果について概観した後，主な処理ごとの測定結果について示し，簡単な評価を行う.

9.6.1 全体の測定結果及び評価

表 9.8 及び**図 9.9** に JTAS 全体及び時間積分計算に要した経過時間を測定した結果及びスレッド並列化により得られた加速率を示す．**表 9.8** において，各欄の上段が秒を単位とした経過時間であり，下段が加速率である．また，経過時間はバリア同期を取ってルートプロセス（ランク 0）における測定結果のみを示している（以下同様）.

表 9.8 より，いずれのスレッド数による実行においても，オリジナルに比べてスレッド版における経過時間が短縮されており，最適化の効果が確認できた．また，8 スレッド実行において，加速率が全体では 6 倍を下回っているものの，時間積分の計算においては 6 倍を上回っており，当初の目標を達成する十分な加速率が得られた.

表 9.10 及び**図 9.11** にスレッド版のプロファイラによる実行状況の解析結果についてその抜粋を示す．表中，各解析項目の"Rank0"及び"Rank1"は，MPI プロセスのランク 0 及びランク 1 における解析結果である．また，"Average"は，ランク 0 とランク 1 の測定値を単純平均した値であり, "Total" はプロファイラが"Process Total"として出力した結果である．L2 キャッシュミス率，アドレス変換バッファミス率共に十分に小さいとは言えず，この結果，浮動小数点演算性能も 150MFLOPS を若干上回る程度に留まった.

表 9.12 及び**図 9.13** に経過時間を元に算出した JTAS オリジナルに対してスレッド版でどの程度性能が向上したかを表す性能向上率を示す．性能向上率は，オリジナルの実行に要した経過時間をスレッド版の実行に要した経過時間で除したものとして計算した．いずれのスレッド数による実行でも性能が向上することが確認出来た.

表 9.8 JTAS 全体の経過時間測定結果

上段：経過時間［秒］
下段：加速率［−］

測定区間	1Thread	2Thread	4Thread	8Thread
オリジナル	8803.71	4978.60	2770.26	1814.81
全体	1.00	1.77	3.18	4.85
スレッド版	6012.07	3229.03	1810.94	1075.91
全体	1.00	1.86	3.32	5.59
オリジナル	8733.57	4908.60	2703.89	1748.62
時間積分	1.00	1.78	3.23	4.99
スレッド版	5889.41	3106.27	1688.14	954.76
時間積分	1.00	1.90	3.49	6.17

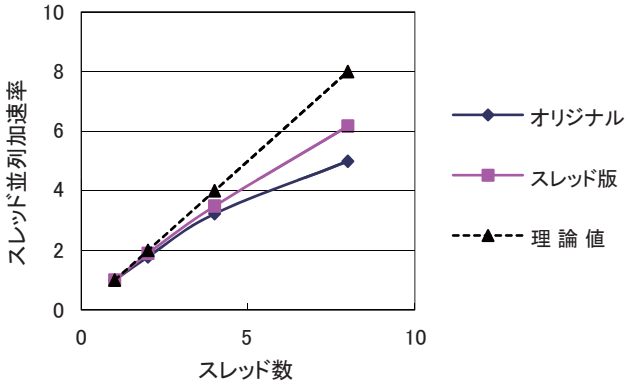


図 9.9 時間積分計算におけるスレッド並列実行加速率

表 9.10 スレッド版プロファイラ情報抜粋

Rank	1Thread	2Thread	4Thread	8Thread
浮動小数点演算性能 FLOPS [MFLOPS]				
Rank0	154.8	154.0	151.4	137.7
Rank1	164.4	171.1	173.1	158.1
Average	159.6	162.6	162.3	147.9
L2 キャッシュ・ミス率 L2-miss [%]				
Rank0	3.08	2.78	2.91	2.75
Rank1	1.96	2.61	2.53	2.39
Total	2.42	2.69	2.71	2.57
アドレス変換バッファ・ミス率 mTLB-op [%]				
Rank0	0.0066	0.0040	0.0019	0.0012
Rank1	0.0035	0.0028	0.0009	0.0006
Total	0.0048	0.0034	0.0014	0.0009

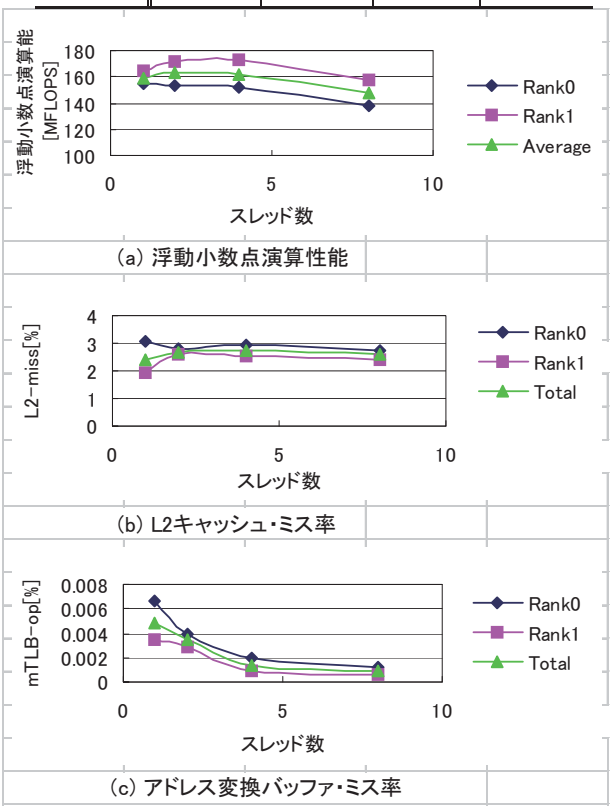


図 9.11 スレッド版プロファイラ情報抜粋

表 9.12 最適化による性能向上率

1Thread	2Thread	4Thread	8Thread
1.46	1.54	1.53	1.69

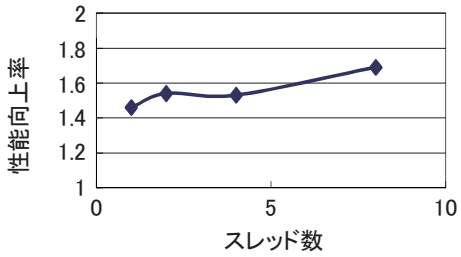


図 9.13 最適化による性能向上率

9.6.2 流束の計算における測定結果及び評価

表 9.14 及び**図 9.15** から **9.16** に，流束の計算について経過時間の測定を行なった結果を示す．

流束の計算においては，オリジナルに比べて処理に要する経過時間が短縮されると同時に，**表 9.14** 及び**図 9.15** に示すように 8 スレッド（2 プロセス，16CPU）実行においてオリジナルで 5.55 倍だったスレッド並列による加速率がスレッド版において 7.42 倍に向上しており，最適化による効果が確認できた．

表 9.14 流束の計算における測定結果

上段：経過時間［秒］
下段：加 速 率［−］

version	1Thread	2Thread	4Thread	8Thread
オリジナル	1141.31	773.14	374.34	205.75
	1.00	1.48	3.04	5.55
スレッド版	1036.89	531.88	273.57	139.73
	1.00	1.95	3.79	7.42

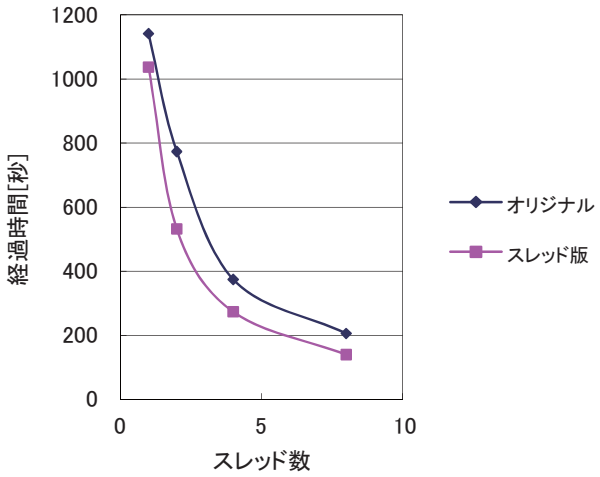


図 9.15 流束の計算における測定結果

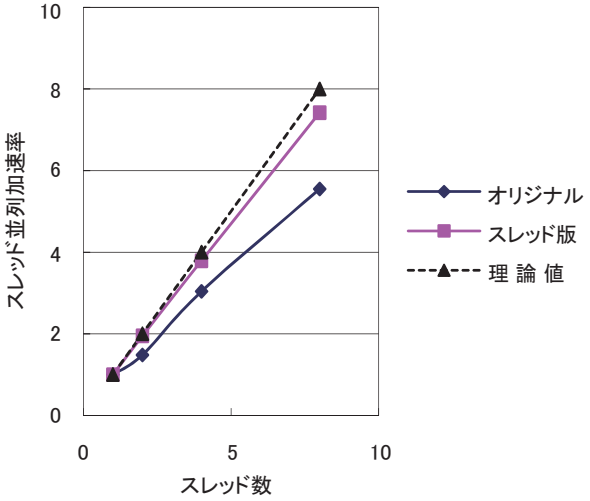


図 9.16　流束の計算におけるスレッド並列実行加速率

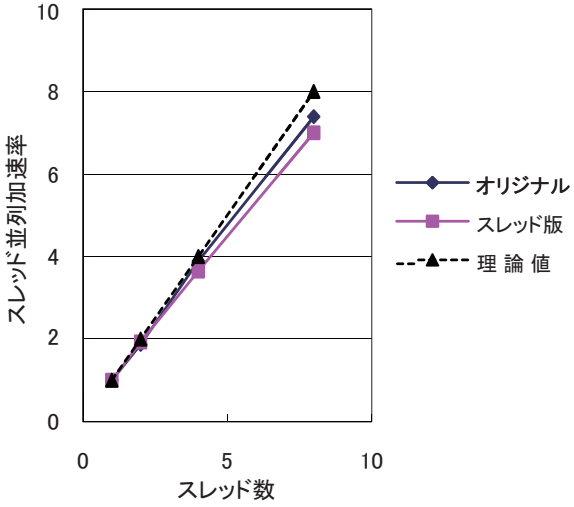


図 9.19　勾配の計算におけるスレッド並列実行加速率

9.6.3　勾配の計算における測定結果及び評価

表 9.17 及び**図 9.18** から **9.19** に，勾配の計算について経過時間の測定を行なった結果を示す．

勾配の計算においては，8 スレッド実行の場合にオリジナルで 7.39 倍あったスレッド並列化による加速率がスレッド版では 7.00 倍に低下している（**表 9.17** 参照）．しかし，処理に要した経過時間は 362.70 秒から 237.36 秒に短縮されており最適化による計算性能向上の効果が確認された．

表 9.17　勾配の計算における測定結果

上段：経過時間［秒］
下段：加 速 率［−］

Version	1Thread	2Thread	4Thread	8Thread
オリジナル	2679.61	1432.21	689.44	362.70
	1.00	1.87	3.89	7.39
スレッド版	1660.64	862.60	454.79	237.36
	1.00	1.93	3.65	7.00

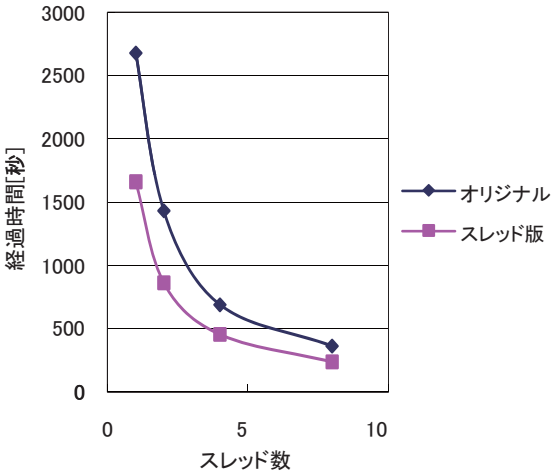


図 9.18　勾配の計算における測定結果

9.6.4　制限関数の計算における測定結果及び評価

表 9.20 及び**図 9.21** から **9.22**，制限関数の計算について経過時間の測定を行なった結果を示す．

制限関数の計算においては，8 スレッド実行の場合にオリジナルで 7.54 倍あったスレッド並列化による加速率がスレッド版では 6.71 倍に低下している．しかし，処理に要した経過時間は，211.96 秒から 161.96 秒に短縮されており最適化の計算性能向上の効果が確認された．

表 9.20　制限関数の計算における測定結果

上段：経過時間［秒］
下段：加 速 率［−］

Version	1Thread	2Thread	4Thread	8Thread
オリジナル	1599.02	861.88	410.54	211.96
	1.00	1.86	3.89	7.54
スレッド版	1087.35	581.07	310.65	161.96
	1.00	1.87	3.50	6.71

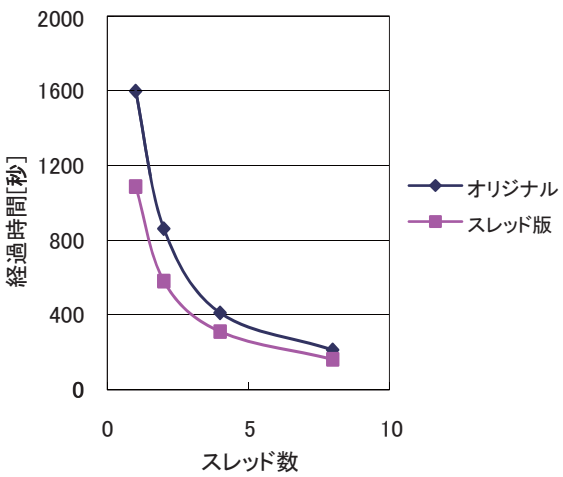


図 9.21　制限関数の計算における測定結果

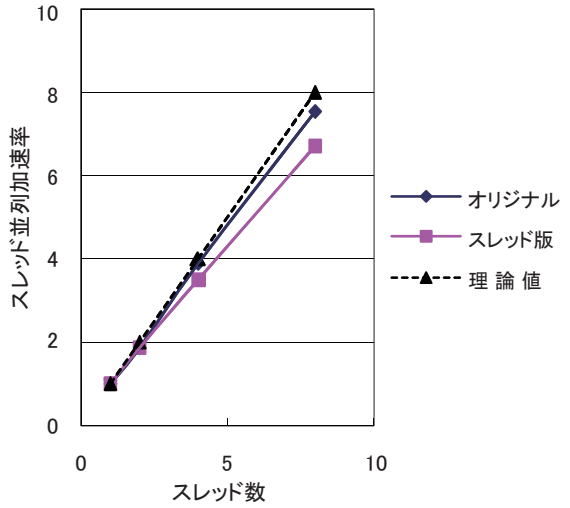


図 9.22　制限関数の計算におけるスレッド並列実行加速率

9.6.5　LU−SGS 法の計算における測定結果及び評価

表 9.23 及び**図 9.24** から **9.25** に，LU−SGS 法の計算について経過時間の測定を行なった結果を示す．

LU−SGS 法の計算においては，8 スレッド実行の場合にオリジナルで 2.97 倍であったスレッド並列化による加速率がスレッド版では 5.76 倍に向上した．同時に，処理に要した経過時間も 773.07 秒から 238.55 秒に短縮されており最適化の効果が確認された．

表 9.26 及び**図 9.27** にコーディング方法を変更した場合に，LU−SGS 法の計算に要する経過時間がどのように変化するかを示す．コーディング方法としては，(1) 色分けにより再帰参照を回避した場合（オリジナル，リスト 9.1 参照，「色分け」と表記），(2) 色分けを削除し DO ループを分割した場合（リスト 9.3 参照，「色分け削除」と表記），(3) 色分けによる再帰参照の回避を行うとともに，超平面を考慮した節点番号の付け替え（Reordering）を行った場合（スレッド版，「色分け+Reo.」と表記），(4) 色分けを削除するとともに DO ループの分割を行い，さらに，節点番号の付け替えを行った場合（「色除+Reo.」と表記）の四種類を検討した．これより明らかなように，色分けの削除を行うとオリジナル（色分けによる再帰参照の回避）に比べて性能が低下すること，節点番号の付け替えを行った場合に最も良い性能が得られることが確認できた．色分けの削除を行った場合について LU−SGS 法の計算に含まれるあるサブルーチンについて調べてみると，MPI 並列のみによる実行の場合にリスト 9.1 DO 100 に相当する部分の計算時間に対して，単に節点毎の配列に足し込むのみである DO 110 に相当する部分の計算に 60%以上の時間を要しており，このことが色分けを削除し DO ループを分割した場合に性能が低下する原因であると考えられる．以上のことから，スレッド版では節点番号の付け替えのみを行い色分けの削除は行わなかった．

表 9.23　LU−SGS 法の計算における測定結果

上段：経過時間［秒］
下段：加 速 率［−］

Version	1Thread	2Thread	4Thread	8Thread
オリジナル	2297.07	1414.62	964.67	773.07
	1.00	1.62	2.38	2.97
スレッド版	1374.72	699.87	389.80	238.55
	1.00	1.96	3.53	5.76

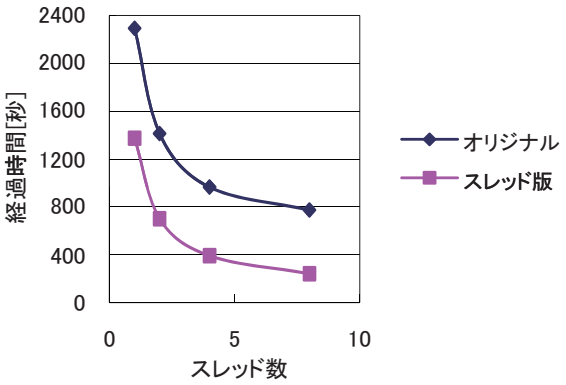


図 9.24　LU−SGS 法の計算における測定結果

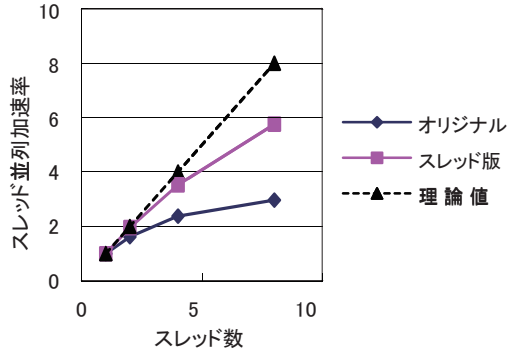


図 9.25　LU−SGS 法の計算におけるスレッド並列実行加速率

表 9.26　LU－SGSの計算におけるコーディング方法と性能の関係

経過時間［秒］

Version	1Thread	2Thread	4Thread	8Thread
(1) 色　分　け　（オリジナル）	2297.07	1414.62	964.67	773.07
(2) 色分け削除	2671.32	1950.94	1638.87	1269.93
(3) 色分け＋Reo.（スレッド版）	1374.72	699.87	389.80	238.55
(4) 色　除＋Reo.	1789.36	896.81	485.75	278.96

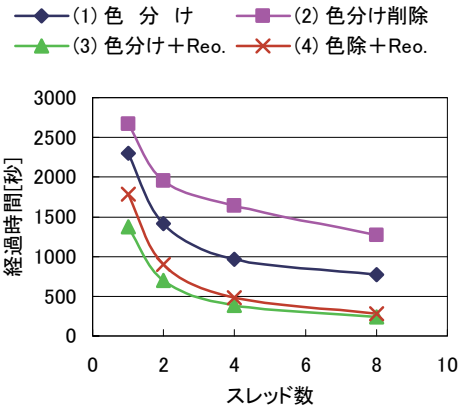


図 9.27　LU－SGS法の計算における
コーディング方法と性能の関係

9.6.6　スレッド並列化におけるオーバーヘッド

勾配の計算及び制限関数の計算において，オリジナルに比べてスレッド版の計算性能は向上したものの，スレッド並列化による加速率は低下した．

そこで，加速率 S ，並列化率 α 及びスレッド数 n としてアムダールの法則，

$$S = \frac{1}{(1-\alpha)+\alpha/n} \qquad (6.1)$$

を適用し，スレッド並列化によるオーバーヘッド O を

$$O = (1-\alpha)T \qquad (6.2)$$

によって評価してみる．但し， T は当該測定区間の計算に要した経過時間であり，従って，オーバーヘッドは，当該区間の計算に要する時間の内，並列化されていない部分の処理に要した時間である．この時，勾配の計算において，オリジナルで 98.8%であった並列化率 α がスレッド版では 98.0%に低下し，オーバーヘッドは，オリジナル版の 4.2 秒に対してスレッド版では 4.9 秒に増加している．この性能低下の原因は，リスト 9.3 の DO 110 に相当する計算のオーバーヘッドが大きいためである．とはいえ，計算時間はオリジナル版に比べて短縮されておりチューニングとしての効果はあったと言える．

9.7　おわりに

3 次元ハイブリッド非構造格子有限体積法 Euler／Navier-Stokes ソルバ JTAS について，全体性能の向上とスレッド並列化の最適化を目的とした変更を加え，JTAS スレッド並列版を開発して，全体の性能が約 1.5 倍向上し，時間積分計算部分で，8 スレッド実行によるスレッド並列化加速率が約 6.2 倍と，理論値の 7 割を越える性能が得られることを確認した．

参考文献

[1] 村山光宏，山本一臣：非構造格子法を用いた航空機高揚力装置周りの流れ場解析の精度検証，航空宇宙数値シミュレーション技術シンポジウム2004論文集，JAXA特別資料 SP-04-012, pp.82-86 (Mar. 2005).

[2] Nakahashi, K., Ito, Y., and Togashi, F: Some challenges of realistic flow simulations by unstructured grid CFD, Int. J. for Numerical Methods in Fluids, Vol.43, pp.769-783, 2003.

[3] 坂下雅秀，松尾裕一，村山光宏：非構造格子 Euler/Navier-StokesソルバJTASの計算性能最適化，宇宙航空研究開発機構研究開発報告 JAXA-RR-06-004, 2006.

第 10 章 CeNSS における並列アプリケーションの実効性能推定法とその有効性検証

10.1 はじめに

CeNSSでは，ノード内は共有メモリ並列（スレッド並列），ノード間は分散並列（プロセス並列）を組み合わせたハイブリッド並列を標準の並列化スタイルとして採用しているのは前に述べた．ハイブリッド並列では，ノード内のスレッド並列に関してコンパイラによる自動並列化機能を用いることができるなど，特にプログラミングの点で有利と言われている．確かに，ベクトルの前システムからスカラーの今システムへ移行する際，ベクトルループをスレッド並列で置き換えることにより混乱なく移行を実現できた．しかし，性能に関しては，ハイブリッド並列は，純MPIによる並列化を上回る性能は得られないという報告[3]もある一方で，JAXA実アプリでは，コードP3のようにハイブリッド並列の方が純MPI並列より性能が良いケースもあり，計算資源の有効利用という観点からは，コードの特性パラメータと並列性能の相関を把握すると同時に，ハイブリッド並列に有効な性能モデルや簡便な性能推定法を見出すことが重要と思われる．

本章では，JAXA実アプリの性能測定結果を元にアムダールの法則を拡張したハイブリッド並列における簡易な実効性能推定法を提示し，その推定精度を検証し有効性を示す[1][2].

10.2 JAXA アプリの並列性能の分析

第 5 章で論じたように，プロセス数とスレッド数の組み合わせによって，CPU 数が一定でも性能値が変わってくるものがある．そこで，CPU 数一定で整理した場合の性能を，コード P1，P3，P5 でプロットしてみたものを**図 10.1** に示す．ここで，(P,T) とあるのは，P がプロセス数，T がスレッド数を表す．コード P1，P5 の場合は，純 MPI の場合が最も良い性能を示しているのがわかる．これは，ハイブリッド並列についての一般に報告されている結果[3]，「純 MPI 並列の方がハイブリッド並列より性能が良い」と矛盾しない．しかし，通信量の多いコード P3 の場合には，特定のプロセス×スレッドの組み合わせ（448CPU の場合 56 プロセス×8 スレッド）のときに最も良い性能を示し，純 MPI の場合に比べ，2 割ほど高い性能を示している．これは，P3 のようなコードの場合，プロセス×スレッドの組み合わせを適切に選ぶことにより，プログラムに手を加えることなく性能向上を図ることができることを意味している．従って，何らかの性能推定モデルがあれば，（プロセス数，スレッド数を任意に変えることができるという条件下で）適切なプロセス数×スレッド数を予め選択できることを示唆している．

10.3 拡張アムダールの法則による性能推定

10.3.1 アムダールの法則

並列システムの性能を表す法則の一つにアムダールの法則がある．いま，あるプログラムにおいて，並列処理できる

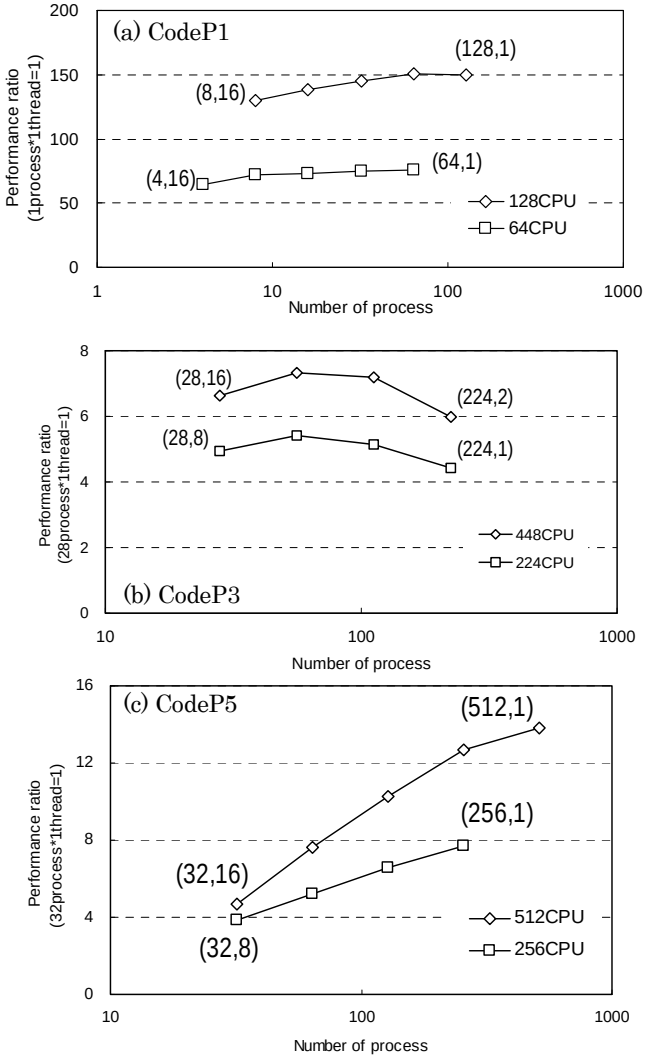


図 10.1 CPU 数一定時の並列性能の比較

部分の割合（並列化率）を a とし，n 台の CPU を用いて得られる性能向上率を $S(n)$とすると，

$$S(n) = T_{serial}/T_{parallel} \tag{10.1}$$

ただし，T_{serial}，T_{parallel} は，それぞれ逐次実行，並列実行時の CPU 時間をあらわし，

$$T_{parallel} = T_{serial} \times \{(1-a) + a/n\} \tag{10.2}$$

の関係がある．もし，$a = 1$ なら $T_{\mathrm{parallel}} = T_{\mathrm{serial}}/n$，ゆえに $S(n) = n$ となり，並列性能は CPU 数の増加とともに完全にリニアに上昇する．また，$n \to \infty$ とすると $S(\infty) \to 1/(1-a)$であり，これは並列化率に応じて性能向上に上限があることを意味している．たとえば，$a = 0.95$ の場合 $S(\infty) = 20$，すなわち性能向上率はたかだか 20，よって，20CPU 以上使うのは無駄ということになる．ただし，アムダールの法則は，問題規模一定の場合の性能（スピードアップ性能）を予測するものであることに注意．問題規模を大きくする場合の性能（スケールアップ性能）はグスタフソンの法則に因る（**付録 D** 参照）．また a は一定ではなく，$a = a(n)$の場合が多いことにも注意．

10.3.2 ハイブリッド並列におけるアムダールの法則の拡張とその検証（その 1）

ハイブリッド並列における並列形態には，プロセス並列とスレッド並列が存在するので，上記の一般的なアムダールの法則は直接的には適用できない．いま，プロセス並列の並列数を n_p，並列化率を a_p，スレッド並列の並列数を n_t，並列化率を a_t とする．**表 10.2(a)**は，コード P1 の並列性能の実測値の一部を表にしたものである．4 プロセス×1 スレッドの場合の性能を基準(=1)としたときの性能比を示してある．プロセス並列化率 a_p は，スレッド 1 の場合の性能向上比（第 1 行）から，スレッド並列化率 a_t は，プロセス 1 の場合の性能向上比（第 1 列）から，式(10.1)(10.2)により導かれる関係 $a = (1\text{-}1/S)(1\text{-}1/n)$ を用いて求めることができ，これより平均の $a_p = 1.018$，$a_t = 0.994$ と求まる．いま，ハイブリッド並列におけるアムダールの法則として，通常のアムダールの法則(10.1)(10.2)の自然な拡張として，

$$S(n) = T_{serial}/T_{hybrid} \tag{10.3}$$

ただし，

$$T_{hybrid} = T_{serial} \times \{(1 - a_p) + a_p/n_p\} \times \{(1 - a_t) + a_t/n_t\} \tag{10.4}$$

を考える．ここに，$(1 - a_p) + a_p/n_p$ はプロセス並列による加速分，$(1 - a_t) + a_t/n_t$ はスレッド並列による加速分をあらわす．上記のa_p及びa_tを式(10.3)(10.4)に代入して，性能向上比を推定したのが**表10.2(b)**である．**表10.2(c)**は，**表10.2(a)**の実測値と**表10.2(b)**の推定値をもとに，推定値/実測値を示したものである．塗りつぶした欄の値を除きほとんどの値が1に近いことから，このコードP1の場合は，拡張アムダールの法則(10.3)(10.4)により，性能推定が可能であることがわかる．なお，高並列で誤差が大きくなるのは，このコードP1の場合，データアクセスがオンキャッシュになり，並列数以上に性能が出てしまっているからである．

表 10.3 は，コード P5 の場合の実測値と拡張アムダールの法則による推定値を比較したものである．**表 10.3(a)**は，実測値であるが，これより平均の $a_p = 0.995$，$a_t = 0.826$ と求まる．この値から式(10.3)(10.4)を用いて性能向上比を推定したのが**表 10.3(b)**，実測値と推定値を比較したものが**表 10.3(c)**である．このコード P5 の場合も，**図 10.4** に示したように，推定値と実測値はすべてのレンジで比較的良く一致している．

表 10.2 拡張アムダールの法則の検証（コード P1）

(a) 性能向上比の実測値

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	2.11	4.23	8.83	17.40
	2	2.00	4.21	8.14	17.42	34.06
	4	3.93	8.30	15.97	34.37	-
	8	7.60	16.06	28.61	67.15	-
	16	14.04	29.14	54.75	104.56	-

(b) 性能向上比の推定値

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	2.04	4.28	9.44	23.75
	2	1.99	4.06	8.50	18.75	47.20
	4	3.93	8.03	16.80	37.04	93.22
	8	7.66	15.66	32.78	72.29	181.92
	16	14.61	29.83	62.53	137.90	347.01

（数字は，4 プロセス×1 スレッドの値を基準）

(c) 推定値と実測値の比較（推定値/実測値）

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	0.97	1.01	1.08	1.36
	2	1.00	0.97	1.04	1.08	1.36
	4	1.00	0.97	1.05	1.08	-
	8	1.01	0.98	1.15	1.08	-
	16	1.04	1.03	1.14	1.32	-

表 10.3 拡張アムダールの法則の検証（コード P5）

(a) 性能向上比の実測値

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	2.00	3.96	7.70	13.81
	2	1.66	3.29	6.55	12.66	21.78
	4	2.62	5.21	10.25	18.92	-
	8	3.83	7.65	14.83	-	-
	16	4.69	-	-	-	-

（数字は，32 プロセス×1 スレッドの値を基準）

(b) 性能向上比の推定値

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	1.99	3.94	7.72	14.83
	2	1.70	3.39	6.71	13.14	25.27
	4	2.63	5.23	10.35	20.27	38.97
	8	3.60	7.17	14.20	27.82	53.47
	16	4.43	8.81	17.44	34.18	65.70

(c) 推定値と実測値の比較（推定値/実測値）

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	1.00	1.00	1.00	1.07
	2	1.03	1.03	1.02	1.01	1.16
	4	1.00	1.00	1.01	1.07	-
	8	0.94	0.94	0.96	-	-
	16	0.94	-	-	-	-

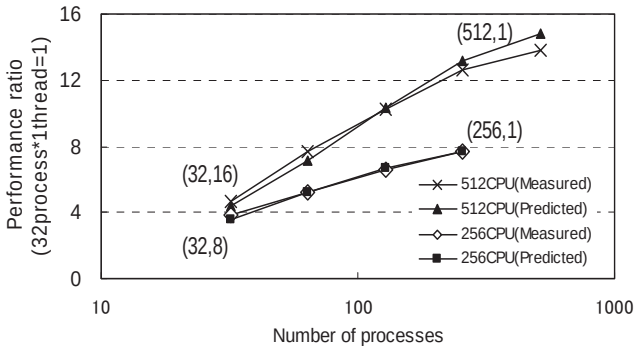


図 10.4 コード P5 における推定値と実測値の比較

このことから，通信コストの少ないグループ I やグループ III のコードについては，拡張アムダールの法則(10.3)(10.4)により性能推定が可能であることがわかる．その理由としては，基本的に通信コストが少ないことの他に，式(10.4)が示しているように，プロセスとスレッドの並列効果に依存性がないことが考えられる．実際，多くの CFD コードでは，多数の多重 DO ループ構造を持っており，外側の DO ループに対してプロセス並列を，内側のループに対してはスレッド並列を適用するハイブリッド戦略を採っていることが強く関係していると考えられる．データは省略するが，コード P4 の場合，多重 DO ループに依存性があるためにこの拡張アムダールの法則による性能推定はうまく行かない．

一方, コード P3 の場合の結果を**表 10.5** に示す. この場合，平均の $a_p = 0.916$，平均の $a_t = 0.915$ である．表 10.5(c)によれば，塗りつぶした欄の値のように，プロセス数×スレッド数の大きいところで推定の誤差が大きくなっている．

表 10.5 拡張アムダールの法則の検証（コード P3）

(a) 性能向上比の実測値

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	1.90	3.21	4.43	-
	2	1.85	3.35	5.14	5.98	-
	4	3.25	5.41	7.19	-	-
	8	4.94	7.32	-	-	-
	16	6.63	-	-	-	-

（数字は，28 プロセス×1 スレッドの値を基準）

(b) 性能向上比の推定値

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	1.85	3.20	5.04	7.09
	2	1.84	3.40	5.90	9.30	13.08
	4	3.19	5.89	10.20	16.09	22.63
	8	5.02	9.27	16.06	25.34	35.63
	16	7.05	13.01	22.53	45.55	50.00

(c) 推定値と実測値の比較（推定値/実測値）

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	0.97	1.00	1.14	-
	2	1.00	1.02	1.15	1.55	-
	4	0.98	1.09	1.42	-	-
	8	1.02	1.27	-	-	-
	16	1.06	-	-	-	-

10.3.3 ハイブリッド並列におけるアムダールの法則の拡張とその検証（その 2）

そこで次に，コード P3 における性能推定誤差の原因とアムダールの法則の改良（高精度化）を考える．いま，プロセス数が変化したときの並列化率を調べてみると, **表 10.6** 上段のようになり，並列化率は一定ではなく，プロセス数の増加とともにかなり低下しているのがわかる．これは，プロセス数とともに増加する通信コスト等が存在するためである．プロファイラを使いコストの内訳を測定した結果, **表 10.6** 下段に示すようにプロセス数の増加に伴う通信コストの増大を実際に確認した．

表 10.6 コード P3 における通信コストの影響

	プロセス数			
	28	56	112	224
実測性能向上比	1.00	1.90	3.21	4.43
並列化率	-	0.946	0.916	0.885
経過時間（秒）	464.87	247.66	149.69	110.29
通信時間（秒）	31.65	28.78	34.76	46.14

（並列化率は 28 プロセスを基準に算出）

そこで，アムダールの法則(10.1)(10.2)に対して，プロセス並列における通信コストの影響を考慮する高精度化を考える．いま，プロセスの並列化率 a_p，プロセス数 n_p に加えて，通信の影響として，プロセス数によらずコスト一定の通信（たとえば通信立ち上がり時間）の割合を c_t，油転送のようなプロセス数にコストが比例する通信の割合を c_n とすると，アムダールの法則(10.1)(10.2)の自然な拡張として，

$$S(n) = T_{serial}/T_{parallel} \tag{10.5}$$

ただし，

$$T_{parallel} = T_{serial} \times \{(1 - a_p - c_t - c_n) + a_p/n_p + (c_t + c_n \times n_p)\} \tag{10.6}$$

という関係を考えることができる．コード P3 におけるそれぞれのコストをプロファイラによって調べてみると, 28 プロセスの場合，$a_p = 0.925$，$c_t = 0.057$，$c_n = 0.005$，$1\text{-}a_p\text{-}c_t\text{-}c_n = 0.013$ となり，6%強の実際に無視できない通信コストが存在する．

（ちなみに，これはプロセス別の演算コストを合計した，いわば非並列状態での比率であり，プロセス数に応じた通信コストの比率は式(10.6)をもとに算出することができる．例えば,4 プロセス並列時の実行時間比は，$T_{parallel}/T_{serial}$ = 0.013 + 0.925/4 + 0.057 + 0.005 ×4 = 0.321 であり，それに含まれる通信の影響は，$c_t + c_n \times n_p$ = 0.057 + 0.005 × 4 = 0.077 であるから，通信コストの比率は，0.077/0.321 ≈ 0.24 となり，これは**図 5.8** で示した特性図の実測値とも概ね一致する．）式(10.5)(10.6)により求めた推定性能向上比と実測値とを比較したのが**表 10.7** であり，式(10.1)(10.2)によるものと比べると，多プロセスにおける性能推定の精度は向上している．さらに，実測範囲以上の多プロセスの場合の性能向上比を推定してみると，**表 10.7** や**図 10.8** にあるように，896 プロセス以上では，通信コストの影響で性能向上比が低下する傾向が現れている．

表 10.7　高精度アムダールの法則によるプロセス性能推定（コード P3）

	プロセス数						
	28	56	112	224	448	896	1,792
実測性能向上比	1.00	1.90	3.21	4.43	-	-	-
(10.1)(10.2)による推定性能向上比	1.00	1.85	3.20	5.04	7.08	8.88	10.17
(10.5)(10.6)による推定性能向上比	1.00	1.84	3.11	4.43	4.81	3.86	2.47

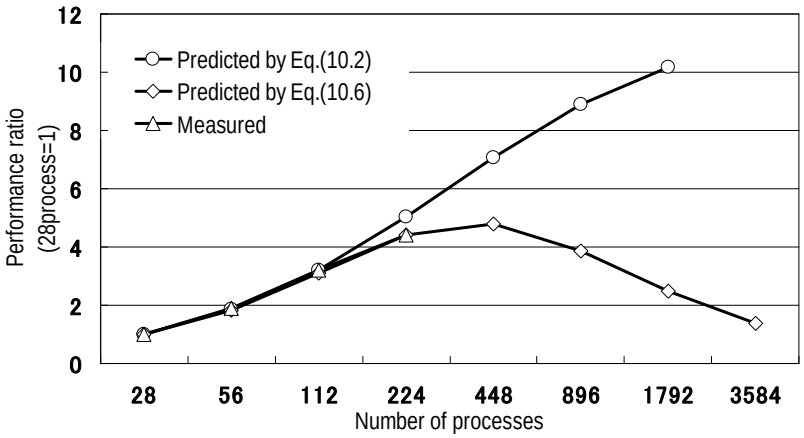


図 10.8　コード P3 における推定値と実測値の比較

この結果を利用して，ハイブリッド並列における拡張アムダールの法則を高精度化してみると，通信の部分はスレッド並列による加速の影響は受けないことから，

$$S(n) = T_{serial}/T_{hybrid} \tag{10.7}$$

ただし，

$$T_{hybrid} = T_{serial} \times \left[\{(1 - a_p - c_t - c_n) + a_p/n_p\} \times \{(1 - a_t) + a_t/n_t\} + (c_t + c_n \times n_p)\right] \tag{10.8}$$

とすることができる．**表 10.9** は，コード P3 に対して式(10.7)(10.8)による推定値と実測値を比較したものであるが，両者はすべてのレンジでよく合致しており，表 10.5(c)と比べても高プロセス×高スレッドの場合の推定精度は改善されているのがわかる．**表 10.10** は，**表 10.8(a)**を CPU 数一定で整理したものである．**図 10.6(b)**で示した特定のプロセス×スレッドで性能が高くなる挙動がよく再現されており，高精度拡張アムダールの法則(10.7)(10.8)が通信量の多いコード P3 の場合の性能推定に有効であることを示している．

表 10.9　高精度拡張アムダールの法則の検証（コード P3）

(a)　性能向上比の推定値

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	1.84	3.11	4.43	4.81
	2	1.84	3.22	4.94	6.14	5.77
	4	3.18	5.12	7.00	7.60	6.41
	8	4.99	7.29	8.84	8.62	6.78
	16	6.97	9.23	10.18	9.24	6.99

(b)　推定値と実測値の比較（推定値/実測値）

		プロセス比				
		1	2	4	8	16
スレッド比	1	1.00	0.97	0.97	1.00	-
	2	1.00	0.96	0.96	1.03	-
	4	0.98	0.95	0.97	-	-
	8	1.01	1.00	-	-	-
	16	1.05	-	-	-	-

表 10.10 高精度拡張アムダールの法則による性能推定（コード P3）

		CPU 数（プロセス数×スレッド数）			
		112	224	448	896
スレッド数	1	3.11	4.43	4.81	3.86
	2	3.22	4.94	6.14	5.77
	4	3.18	5.12	7.00	7.60
	8	2.99	4.99	7.29	8.84
	16	2.63	4.52	6.97	9.23

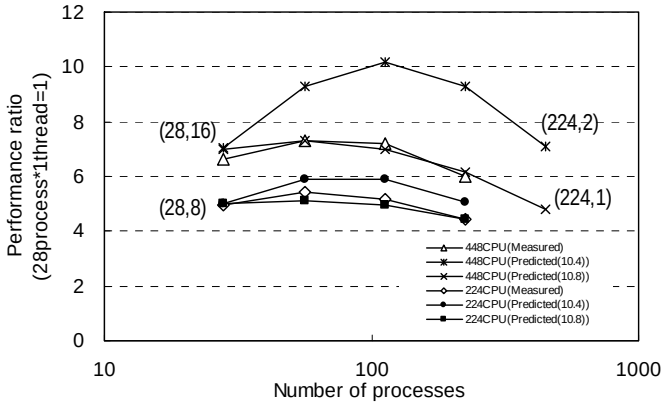


図 10.11 コード P3 における推定値と実測値の比較

10.4 おわりに，性能評価のまとめと課題

本章では，JAXA における並列 CFD コードの性能測定結果を示すとともに，コードの特性と並列性能の関係について論じた．また，アムダールの法則を拡張したハイブリッド並列における簡易な性能推定法を提示し，その推定精度を検証した．JAXA のハイブリッド並列 CFD コードの特性を分析した結果，通信が少ない場合は拡張アムダールの法則，通信が多い場合は高精度拡張アムダールの法則で性能向上比の推定が可能であることがわかった．ここで示した性能推定法は，特殊なパラメータを引用しているわけではないので，JAXA の並列システムに限らず，一般の並列システムに適用できるものである．

しかし，今回実施したような系統的性能評価は一般には困難と考えられるから，拡張アムダールの法則(10.3)(10.4)や(10.7)(10.8)の基本になっている並列化率や通信コストを如何に簡便に見積もるかがこの推定法の鍵である．例えば，コード P1 のような通信量が少なく線形の性能の場合には，プロセス×スレッドの組み合わせとして，16×1, 1×16 のように，2 ケースでプロセス並列化率とスレッド並列化率を採取すれば，式(10.3)(10.4)により精度良い性能推定が可能である．しかし，コード P3，P4 のように通信量が多い場合には，式(10.7)(10.8)を使う必要があり，しかも，その場合には，プロセス数に比例するかどうか等の通信の中身まで把握する必要がある．通信量の全体はプロファイラで知ることができるが，通信の中身を簡単に測る方法については今後の検討課題である．

第 4 章より NS-III の基本特性，CeNSS における JAXA 実アプリの処理性能，コードのチューニング事例，簡易な性能推定法について述べてきた．基本性能や JAXA アプリの性能とその類型，SMP クラスタ上で性能を出すハイブリッド並列プログラミング技術，ハイブリッド並列の性能評価・性能推定等について多くの有意義な知見が得られた．その一方で，SMP クラスタ特有の問題やハイブリッド並列で性能を出す複雑さ・難しさも明らかになった．

基本性能に関しては，**演算実効性能の低さ，メモリ性能・インターコネクト性能のアンバランス**が気になった．大規模並列性能は，**JAXA 実アプリに対しても 1,000 並列程度まではスケールする**ことが確認できた．一方，**ハイブリッド並列**に関しては，アプリによって性能特性に差があり，場合によっては，**同じ並列度でも 2 割程度は性能が変化する**（得をする）ことがわかった．その場合，**本章で示したような性能モデル**を使えば，流そうとするコードの特性（メモリコスト，通信量）を調べることにより，最適なプロセス数とスレッド数の組み合わせを知ることができる．また，JAXA アプリのスレッド並列特性からは，**2 とか 4 スレッドまでならスレッド並列による性能向上はそれなりに得られる**ことがわかった．ただしかし，何倍もの CPU 数を使ってのスレッドによる性能向上が思ったほどでなければ，大規模なスレッド並列を使うことを敢えてするだろうか．また，自動並列ならまだしも OpenMP によるスレッド並列化等の手間をかけるぐらいだったら単純にプロセス数を増やすという選択をするのが普通ではないだろうか．一方，**性能チューニング**は，事例で示したように，技術は無論のこと，**経験と根気の要る作業**である．その上で，プロセスもスレッドも両方ということになると，ハイブリッド並列は手間に比べてメリットが少ないという印象を与えることにならないだろうか．運用上の話はまた別であるが，性能や並列モデルからみた疑問・課題への対応とそれを踏まえた次期のシステムの在り方に関しては第 14，15 章で述べることとしたい．

参考文献

[1] Matsuo, Y., Tsuchiya, M., Aoki, M., Sueyasu, N., Inari, T. and Yazawa, K.: Early Experience with Aerospace CFD at JAXA on the Fujitsu PRIMEPOWER HPC2500, *Proc. SC'04*, Pittsburgh, USA (Nov. 2004).

[2] Matsuo, Y., Sueyasu, N. and Inari, T.: Performance Evaluation and Prediction on a Clustered SMP System for Aerospace CFD Applications with Hybrid Paradigm, *Parallel Computational Fluid Dynamics, Parallel Computing and Its Applications*, Elsevier 2007.

[3] Cappelo, F. and Etiemble, D.: MPI versus MPI+OpenMP on IBM SP for the NAS Benchmarks, *Proc. SC'00*, Dallas, USA (Nov. 2000).

第 11 章 CeNSS の運用分析と課題

11.1 はじめに

NS-III の中核システムたる CeNSS は，ベクトルからスカラーへの方式変更という点もさることながら，NWT（NS-II）に比べ，CPU 数が 14 倍弱，メモリ量が 80 倍強，ストレージ量がディスクだけで 200 倍弱と，規模・物量の面で前システムに比べ相当に拡大したため，当初は未経験によるパラメータ設定の困難や実運用面の課題の噴出が懸念された．2007 年 10 月で導入して丸 5 年を迎えたが，初期不良で最初は少しバタバタしたものの，懸念された更新後の大きな混乱もなく，その後も含めて試行錯誤ではあるがそれまでの経験と勘を頼りに何とかやって来られた．

ここでは，その間に取得した統計情報を用いて CeNSS の導入後 5 年間の運用の経緯を振り返るとともに，いくつかの論点から運用を総括・分析し，課題などに言及してみたい．

11.2 ジョブ運用

運用データの分析に入る前に，管理側の体制等を含めて具体的にどのような運用を行って来たかに触れる必要がある．なぜなら，ジョブ[25]の動きやユーザ利用は，システムの形態というよりはジョブ運用方針やシステムのパラメータ設定に強く依存するからである．CeNSS システムにおいては，ジョブスケジューラを自主開発（11.6.1 参照）し，パラメータを自由に変えられるので，通常のシステムではできないような柔軟な運用も可能となっている．ジョブとしては一般に，バッチジョブとインタラクティブジョブがあるが，ここでは特に断らない限り，ジョブといえば「バッチジョブ」を意味するものとする．また，バッチジョブとは，バッチファイルを計算機にサブミットすることにより，バッチキューイングシステムによって起動される一括処理のことを指す．

CeNSS の基本的な運転形態は，24 時間運転とし，年度当初に決めた計画停止期間（お盆，年末年始，年度末）以外は運転することとしている．これは，ユーザからの処理要求が常に十分あるというのが大きな理由ではあるが，停止・起動にある程度時間がかかるため，それらを頻繁に繰り返すのはむしろ非効率的であるという判断に因る．ただし，長期の連休や年末年始などの休日が続くと，ジョブが枯渇する恐れがあり，また，大きなシステムトラブルが発生すると対処できないということもあり，基本的にはユーザサービスは停止（計画停止）としている．また，毎月の第 2 月曜日の午前中は定期保守時間とし，ユーザサービスを停止している．この間に，修正・予防等の保守作業は無論のことバグ等の修正，運用ソフトの更新，パラメータの変更などを実施する．

CeNSS は，基本的にはバッチシステムなので，バッチ処理がシステムとしての最も重要な仕事である．ジョブには，前述のように，バッチ処理の他にインタラクティブ処理やファイル処理などがあるが，バッチ処理のパラメータは，システムの基本であり，イコール運用方針と言っても過言ではなかろう．**表 11.1** に，バッチジョブのパラメータを示す．パラメータの細かな変更は随時行っているが，稼動した期間の中では最も長い間使用した数値を掲げる．ここで，通常ノードとは，ノードあたり 32CPU のノードであり，FAT ノードとは，ノードあたり 128CPU のノードである．

表 11.1 CeNSS のバッチジョブパラメータ

キュー名	ジョブの種類	タイプ	実行ノード
QTSS	会話型キューイングジョブ	TSS	通常ノード
QJOB	バッチ型ジョブ	DEBUG	
		LONG	
		FAT	FAT ノード

タイプ	起動判定 ｱﾙｺﾞﾘｽﾞﾑ	ﾌﾟﾛｾｽ数 (最大値)	プロセス資源使用制限値(MAX)		
			実効 CPU 使用時間	ｽﾚｯﾄﾞ数	ﾒﾓﾘ量
TSS	FIFO	128	300 秒	8	20GB
DEBUG	FIFO	384	500 秒	8	
LONG	優先順位	512	20000 秒	16	
FAT	FIFO	128	20000 秒	32	15GB

バッチ処理のパラメータを設定するにあたって，機会均等，透明性，利用効率などに留意した．また，CeNSS はチェックポイント/リスタート機能を持たないので，1 ジョブの最大 CPU 時間を 5.5 時間とした．チェックポイント/リスタートの必要性については議論のあるところであろうが，メモリ容量や仕組みを組み込むためのコスト等を考慮し，CeNSS では実装しなかった．CeNSS においては，保守等の計画停止時におけるジョブの絞込みや再開はジョブスケジューラが担当しているので，チェックポイントを取る必要性は必ずしも大きくない．

稼動期間中のジョブ運用に影響した事象としては，特別利用と呼ばれる優先処理がある．これは，プロジェクト支援や緊急処理のために，特定のジョブの優先順位を上げるというものであり，2005 年度から運用した．2005～2007 年度の実績では，全体の CPU 時間の約 3 割が特別利用に割かれた．特別利用は，CPU 数固定の定型処理が多いため，特別利用のジョブ形態（CPU 数，メモリ量）等が，ジョブ運用全体の統計に少なからず影響を与えていると思われる．しかしながら，JAXA のスパコンサービスは，一般的なスパコンセンターの提供するサービスとは異なるので，特別利用のミッション志向の利用も含めた利用形態は特別なものというよりは，今後の JAXA のスパコン運営においても有効であり継続されるであろうという前提の下に，以下に述べる統計処理の部分では，特別扱いはしていない．

共有メモリシステム特有の並列化にスレッド並列がある．スレッド並列は，自動並列または OpenMP によって起動される．スレッド並列は，ユーザ側からすれば，共有メモリの中で計算を速くしたいときに使う動機が生まれる．最初のう

[25] ここで，ジョブとは，コンピュータにさせる仕事の単位を指す．

ちは，スレッド並列ユーザの優先順位を高く設定した運用を行った．しかしながら，同じ並列数において，プロセス数とスレッド数を任意に変えられるようなジョブは少なく，プロセス数は一定で，スレッド数を 1 から 2 にするといった形態のジョブが大半なので，必要な CPU 数がスレッド数倍になってしまって，スレッド並列によって計算時間は短くなる反面，今度はジョブが動くまでの待ち時間が長くなってしまうといった問題が発生し，共有メモリにおけるメモリ競合（メモリコンテンション）の問題も相俟って，特に多スレッド並列の利用は広まらなかった（後述）．

11.3 運用のトピックス

NS-III ほどの大規模で挑戦的なシステムとなると，導入から運用終了まで問題もなく最良の状態で稼動し続ける（もちろんそれが理想ではあるが）ことはあり得ないと言っても過言ではない．経営層からの要求や稼動状態を見ながら，そこそこ大きな構成変更やシステム・パラメータの調整などを臨機応変に行ってきた経緯がある．ユーザ側も当然その影響を受ける．そこで，運用で発生した大きなトピックスを**表 11.2** に時系列で列挙する．ユーザの利用状態に影響を与えない程度の軽微な変更や調整，システム作業は常時行っているが，ここには挙げていない．

表 11.2 運用トピックス

2002 年	10 月	NSIII システムの稼動開始
2003 年	11 月	I/O プロセス用に Limited モードを標準化 管理用 CPU の設定
2003 年	8 月	新課金指標（実効 CPU 使用時間）によるジョブ打切り制限運用開始
2004 年	1 月	ログインノード長時間 TSS ジョブの運用開始
2004 年	7 月	共有 DTU の導入
2008 年	10 月	NSIII システム運用終了

大きなトピックスとしてまず挙げられるのは，XPFortranにおけるI/Oプロセスの問題がある．UNIXは，プロセス主体のOSなので，デーモンと呼ばれるプロセスがユーザには見えないところで沢山動いているいることは良く知られている事実であるが，XPFortranを使ったジョブにおいて，入出力が発生するとI/O用プロセスの割り込みが発生し，性能を劣化させるとか，常に（並列数＋１）個のCPU数を指定しなければならないという煩雑さを招いた．この課題を取り除くために，まずは実行時オプションとして，I/Oプロセスをユーザプロセスとマージして1CPUに収めるLimitedモードと，独立して1CPUを割当るFullモードの選択肢を用意した．当初はFullモードを無指定時の標準としていたが，2003年11月以降は，システム全体の稼働率が上がることから，Limitedモードを標準としてI/Oプロセスのみの余分なCPUを使わない運用に移行した．

また，この問題をきっかけに，そうした余分な管理プロセス（「OS ジッタ」と呼ばれることもある．）用に，**計算ノードあたり 1CPU を割り当てる（管理 CPU）**措置を講じた．一見，資源の無駄のようにも思えるが，ユーザジョブは管理プロセスの影響を受けなくなったので，結果的には効率化に繋がった．

次に，経過時間のばらつきの問題がある．運用当初より，あるジョブの実行開始から終了までの経過時間が，同時実行する他のユーザのジョブの影響を受けて予定より間延びし，実行が終了し得ないうちに経過時間切れで強制終了されるという状況が多発した．この理由として，SMP アーキテクチャの宿命として，あるジョブがメモリアクセスの多い他のジョブと同居した場合，メモリ負荷による性能変動の影響を激しく受けるためである．この障害を除去するため，メモリアクセスによる遅延を反映しない新課金指標として「**実効 CPU 使用時間**」を新たに設定し，これをジョブ打切り制限値とした．

CeNSS は，導入 2 年目にしてジョブの混雑状況が激しさを増してきた．この混雑緩和のため，インタラクティブ処理専用のログインノードの稼働率を高め計算エンジンとしての役割を担わせることを目的に，インタラクティブ処理の応答性を決して劣化させないように，システムが軽負荷の状況においては，バックグランド処理的に長時間の TSS 実行コマンドが実行できるようなシステムとした．この**長時間 TSS 実行コマンド**は，ログインノードの稼働率を向上させるとともに，バッチジョブ処理にない処理の柔軟性がユーザサービスレベルの向上にも役立った．

次のトピックスとして，**共有 DTU** の導入がある．32 個の CPU を有する計算ノードには，結合ネットワークへの出口として 1 個の DTU が付いているが，一度に 16 プロセスしか処理できないために，1 ノードに最大 16 プロセスしか起こすことができないという問題があった．スレッドを利用できれば良いのだが，スレッド並列性能が出ないコードも結構あり，CPU 資源が無駄になる．そのようなコードに対しては，ノードに CPU 数にあたるプロセスを詰め込めるように富士通に依頼して実装したのが共有 DTU である．ただし，スレッド性能が出ないコードははっきりしていたので，プロセス間の転送量が少ないジョブに共有 DTU を割当るスケジューリングを実施し，スレッド並列ジョブの実行に依存することなく 32 個の CPU が実行可能となったので，共有 DTU 機能は CPU 稼働率の向上に大きく寄与した．

こうした，トピックス（課題）は，システムの提供ベンダー（この場合は富士通）の問題に因るところもあるが，コストや時間的な制約等の中で，完全なシステムを開発することは不可能と言っても良いのである意味では避けられない．また，運用は，システムによって変えざるを得ないものであり，利用側の要求も時代時代によって変わって来るのが常であるがゆえに，実際に運用の結果として様々な問題が発生する．上記のとおり，我々は，これらの問題に対して，できるだけシステムを有効利用するというスタンスで対処した．

11.4 システム運用の統計情報

システムの運用で特徴的なのは，ISO9000 による QMS 管理を適用したことである．「QMS をスパコンの運用に」などという向きもあろうが，その是非は別な機会に論ずることにして，極めて有意義であったと思うのは，QMS の仕組みによって当初からの各種統計情報をきちっと取ることができたということである．(今でこそ QMS は定着しているものの，初期のころはベンダーをはじめとして JAXA 側の運用部隊にも建設的にご協力いただいた点には深謝したい．)

11.4.1 ハード/ソフト障害の推移

図 11.3 は，運用当初からのハード障害及びソフト障害の件数を月別（棒グラフ），累計（折れ線グラフ）で示したものである．最初は多く徐々に少なくなるという典型的な漸近カーブを描いているが，途中 1 年後ぐらいの時点でソフト障害が一時的に増加している時期があるのが見て取れる．これは，ユーザがシステムに慣れて来て，システムの負荷が非常に高まり，それまでの運用では現れて来なかった障害が表に出てきたことによるものと解釈している．「更新して 1 年後ぐらいの負荷が高まった時期は要注意」という教訓？は，今後の運用に役に立てて行きたいと思っている．

月別の発生をみると，初期動揺期→遷移期→安定期ぐらいに時期を分けることができるであろう．運用的には遷移の期間を短くすることがポイントと思われる．一方，累計をみるとハード障害の傾きが 0 近くに漸近して来ているのは直感的にも理解できる傾向である．これは予兆監視により運用への影響を最小限に留めている効果も大きい．しかし，ソフト障害については，ハード障害ほど傾きが小さくなっていない．これは，ソフト障害は基本的になかなか取れにくいことを意味しているのか，ソフト障害は基本的にある一定頻度で起こりやすいものなのか，ベンダーも交えた突っ込んだ解釈・分析が必要であろう．(ちなみに，ここで言っている「障害」とは，あくまで運用管理サイドで把握し定義しているものであって，ベンダーからの情報ではないことに注意．)

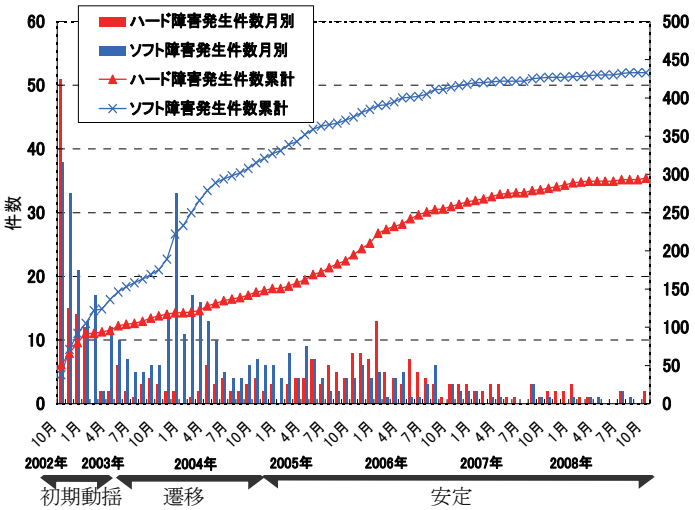


図 11.3 ハード/ソフト障害の推移

表 11.4 ハードウェア障害の年度別推移

	2002	2003	2004	2005	2006	2007	2008
SD	15	6	3	7	0	3	2
MD	65	33	25	15	24	19	7
保守	97	119	37	86	117	120	66
その他	0	0	4	37	30	0	0

SD: System down, MD: Machine down

表 11.5 ソフトウェア障害の年度別推移

	2002	2003	2004	2005	2006	2007	2008
OS	44	74	73	34	22	7	4
言語	44	47	14	9	3	0	0
運用	1	8	11	4	6	1	2
その他	3	1	7	11	2	1	0

障害の種類を年度別に挙げたのが以下の**表 11.4**, **11.5** である．ここで，2002 とあるのは，2002 年 10 月から 2003 年 9 月までの数字を示している．また，SD とはシステムダウン（全系停止），MD とはマシンダウン（一部停止），保守とは予防保守を表す．2006 年度にハード障害が一時的に増えている理由は不明である．

11.4.2 基本運用データの推移

運用時間，ジョブ処理件数，稼働率などのシステム運用の基本データの月単位の推移を**表 11.6** にまとめる．ここで，

CPU 割当時間　CPU を実行ジョブに割り当てていた時間．ジョブが使用していなくても割り当てられていれば加算される．

CPU 使用時間　utime + stime，ここで utime とはユーザプロセスに消費された時間，stime とは，システム関係に消費された時間をあらわす．

実効 CPU 使用時間　CPU 時間からメモリ待ちや DTU 転送待ち時間を除いたもの．CPU が実際の計算に使われていた時間をあらわす．

をあらわしている．最後の 1 年半ぐらいの期間については，月平均値で，ジョブ処理件数 8,000 件，ジョブ運用時間 1,000,000 時間，CPU 稼働率 90%以上，ただし実効 CPU 時間は 500,000 時間程度という数字が得られている．

表 11.6 CeNSS における運用の総括

月次	バッチジョブ処理件数	CPU 割当時間	CPU 稼働率	CPU 使用時間	実効 CPU 使用時間	ジョブ運用時間	電源投入時間	運用日数
2002 年 10 月	27,724	85,308	57.3%	64,009		148,800	824,656	5
11 月	30,115	293,042	57.9%	255,364		505,920	858,985	17
12 月	15,638	474,683	72.2%	394,853		657,696	1,033,525	17
2003 年 1 月	10,005	534,778	65.8%	474,324		812,448	929,143	21
2 月	10,453	462,522	54.3%	393,232		851,136	959,363	22
3 月	4,864	407,142	75.2%	335,911		541,632	1,030,450	14
4 月	11,295	776,419	77.2%	749,382		1,005,888	1,243,017	26
5 月	9,524	518,844	74.5%	493,894		696,384	1,086,001	18
6 月	12,268	819,403	78.4%	784,390		1,044,576	1,249,588	27
7 月	13,021	932,358	86.1%	865,076		1,083,264	1,268,287	28
8 月	9,593	836,167	86.5%	647,703		967,200	1,203,572	25
9 月	13,003	882,208	87.7%	641,238	378,308	1,005,888	1,219,041	26
10 月	10,000	871,920	86.7%	609,308	470,422	1,005,888	1,242,451	26
11 月	11,698	831,705	79.6%	584,132	442,933	1,044,576	1,206,040	27
12 月	13,107	769,290	82.9%	588,756	457,988	928,512	1,257,637	24
2004 年 1 月	10,464	865,273	89.5%	672,442	536,195	967,200	1,171,052	27
2 月	9,107	768,855	76.4%	602,484	477,613	1,005,888	1,163,899	29
3 月	8,252	864,746	77.5%	601,648	468,263	1,115,208	1,241,342	31
4 月	6,683	795,174	72.6%	587,958	433,019	1,095,065	1,019,591	30
5 月	6,392	799,577	78.2%	573,918	403,577	1,021,963	1,235,611	31
6 月	7,071	863,698	77.9%	632,304	462,128	1,108,297	1,199,412	30
7 月	9,631	1,023,768	89.5%	746,536	540,789	1,143,803	1,253,418	31
8 月	7,213	899,402	89.1%	612,541	485,711	1,009,067	1,171,749	31
9 月	7,597	1,014,837	88.1%	730,297	578,453	1,151,445	1,216,779	30
10 月	7,561	965,746	86.1%	682,654	501,909	1,121,485	1,266,194	31
11 月	7,737	981,669	86.7%	751,433	550,508	1,132,109	1,200,883	30
12 月	6,506	837,655	84.7%	628,233	446,005	979,738	1,100,073	27
2005 年 1 月	9,274	903,114	83.7%	683,409	518,150	1,077,645	1,089,229	27
2 月	11,066	963,801	88.2%	717,632	455,024	1,082,974	1,128,720	28
3 月	9,747	1,087,054	88.2%	829,770	487,909	1,232,740	1,265,859	31
4 月	6,916	1,058,924	89.6%	763,092	550,908	1,182,446	1,190,237	30
5 月	8,446	986,760	90.2%	744,533	529,940	1,093,670	1,204,578	31
6 月	8,537	1,128,756	95.6%	784,831	542,796	1,180,802	1,221,543	30
7 月	7,870	1,199,037	95.5%	892,740	618,449	1,255,824	1,264,016	31
8 月	7,588	943,621	92.0%	690,911	481,964	1,025,245	1,089,357	31
9 月	8,550	1,148,136	94.6%	834,972	596,940	1,213,140	1,227,422	30
10 月	7,652	1,150,604	93.6%	781,372	536,530	1,229,741	1,250,585	31
11 月	7,040	1,138,235	95.4%	795,796	520,985	1,193,472	1,202,992	30
12 月	7,297	938,062	91.8%	713,035	479,877	1,022,029	1,133,128	29
2006 年 1 月	7,443	1,003,819	92.3%	725,236	492,202	1,087,790	1,092,508	27
2 月	9,865	1,062,306	94.6%	779,963	563,887	1,123,280	1,134,856	28
3 月	7,404	1,160,357	95.3%	685,173	494,169	1,218,095	1,232,882	31
4 月	8,795	1,066,759	96.5%	751,545	522,886	1,105,648	1,150,463	30
5 月	10,790	1,144,923	96.0%	784,572	556,694	1,193,097	1,247,014	31
6 月	6,561	1,138,996	92.7%	825,026	538,852	1,228,382	1,227,368	30
7 月	7,260	1,141,966	91.5%	829,513	527,496	1,259,771	1,243,572	31
8 月	8,520	1,084,960	96.4%	797,450	545,801	1,125,297	1,173,176	31
9 月	8,424	1,177,382	97.3%	875,297	564,303	1,209,097	1,225,999	30
10 月	10,427	1,129,727	93.5%	876,811	592,576	1,208,513	1,244,599	31
11 月	7,426	1,132,877	97.0%	879,100	573,615	1,168,235	1,192,485	30
12 月	7,703	1,051,359	94.3%	787,073	536,595	1,114,851	1,125,650	28
2007 年 1 月	6,933	1,020,576	93.7%	702,824	490,350	1,089,612	1,089,194	27
2 月	7,103	1,057,639	94.5%	734,147	512,982	1,118,823	1,126,617	28
3 月	8,205	1,113,071	94.0%	759,364	541,429	1,184,194	1,241,548	30
4 月	8,240	1,089,524	94.7%	795,332	249,916	1,151,318	1,174,433	29
5 月	9,647	1,160,862	94.1%	872,717	589,079	1,232,152	1,247,298	31
6 月	7,630	1,091,595	91.0%	734,489	523,242	1,200,439	1,224,292	30
7 月	6,417	1,165,307	94.6%	831,570	831,570	1,232,372	1,258,080	31
8 月	6,172	1,050,810	90.4%	740,560	523,992	1,162,279	1,202,382	28
9 月	6,221	1,168,664	95.5%	811,020	561,855	1,224,223	1,227,530	30
10 月	7,929	946,913	89.7%	693,010	485,401	1,055,375	1,074,267	31
平均/月	9,854	934,142	79.95%	662,013	505,424	1,027,280	1,168,453	27.2

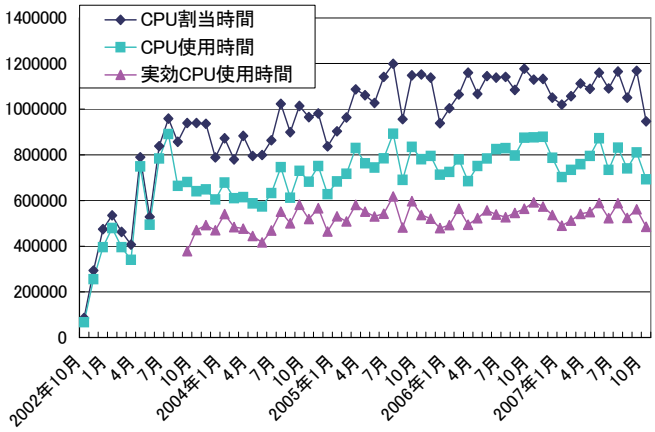


図 11.7　各種 CPU 時間の推移

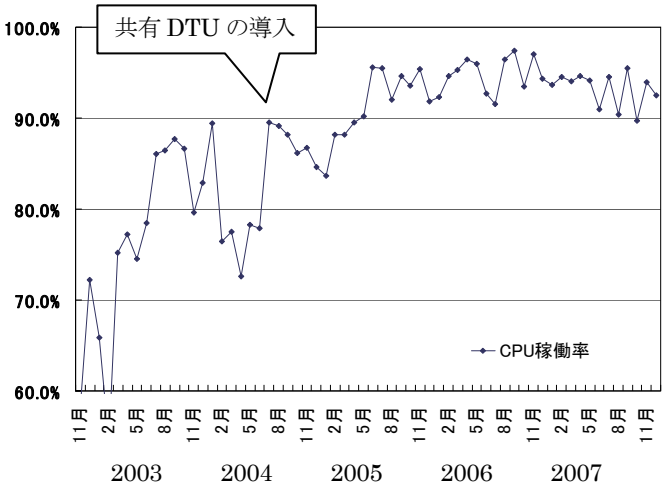


図 11.8　CPU 稼働率の推移

表 11.9　計画停電の日数

2002	2003	2004	2005	2006	2007	2008
4	4	7	7	6	9	8

表 11.10　計画外停電の状況

発生日時	経過
2003/1/31 06:54:55	30 分程度の停電（原因不明） 全系ダウン 13:00　運用再開
2003/8/23 14 時頃	17 時過ぎまで停電（原因は東電側の問題） 全系ダウン 23:57 運用再開
2004/7/15 16 時頃	16:43 頃瞬電　18:00 頃瞬電（原因は雷） 一部空調器故障

図 11.7 に各種 CPU 時間の推移をグラフで示す．図において，CPU 割り当て時間が少しずつ増大しているのは，まさに運用側のいろいろな努力による賜物である．2003 年 7 月以前に線が重なっているのは，きちっとしたデータが取れていなかったことによる．平均値でみると，CPU 割当時間：CPU 使用時間：実行 CPU 使用時間 ＝1：0.75：0.57 となっていて，CPU 割当時間に対する有効な CPU 稼働時間は約 6 割に留まっており，プログラムの作りの問題などもあるにせよ，ベンダーにはこの数字を真摯に受け止めていただきたいと思う．**図 11.8** に CPU 稼働率（＝CPU 割当時間／ジョブ運用時間）の推移を示す．**2005 年 5 月以降はコンスタントに 90%を越えていて，利用率としては限界に達しており，システムとしては更新すべき状態となっている**のがわかる．

空調設備及び自動運転システムに関しては，運用期間中は重大な障害は発生せず極めて安定に動作した．電力供給は非常に安定している．ただし，年に数回の計画停電がある．これには，電力供給設備の定期点検等が含まれる．**表 11.9** に計画停電の日数を示す．また，運用期間中，計画外の停電（「瞬電」を含む）が 3 回発生した（**表 11.10**）．NS-III システムは，その規模ゆえに，一部のネットワーク機器を除いて，無停電電源装置（UPS）は設置していないため，一瞬でも電力供給が停止すればシステムそのものも停止する．電力供給の突然の遮断は，ジョブ等がリセットされるだけでなく，大きなパルス電圧が発生するため，システムにとっても極めて危険である．しかし，幸いなことに，3 回の停電に対し，ハード・ソフトともトラブルは発生しなかった．近年の電力事情から推察して，瞬電等の発生は想定しにくいが，可能性としてはゼロではないので，災害等への対策も含め，そうした不可抗力的な事象への対応については今後とも検討が必要であると考えている．

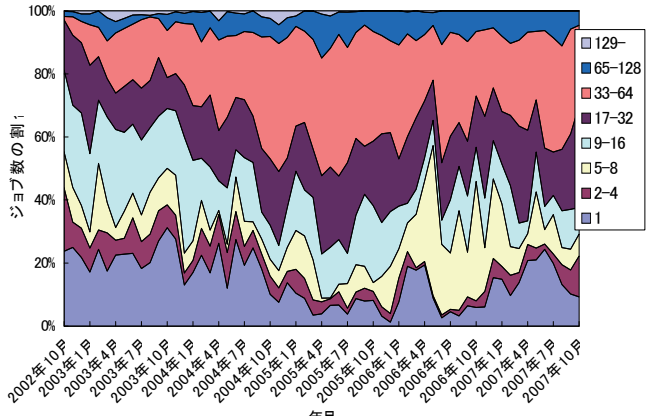


図 11.11(a) プロセス数の推移（ジョブ数）

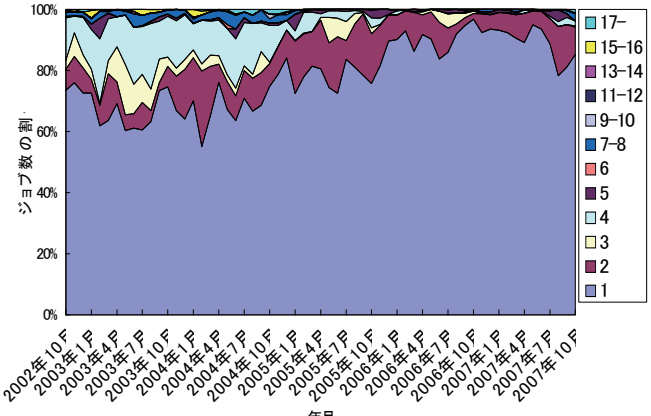


図 11.12(a) スレッド数の推移（ジョブ数）

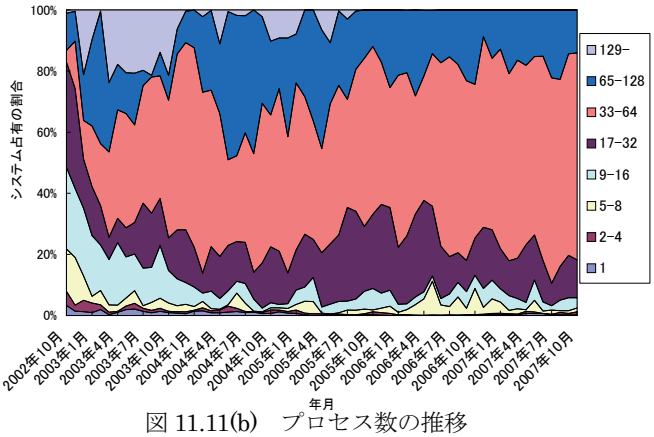


図 11.11(b) プロセス数の推移
（プロセス数×スレッド数×経過時間）

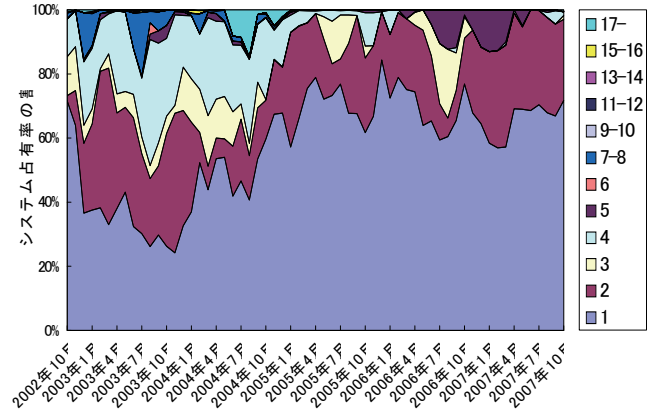


図 11.12(b) スレッド数の推移
（プロセス数×スレッド数×経過時間）

11.4.3 プロセス数，スレッド数の推移

図 11.11(a)及び**(b)**は，運用当初からのプロセス数の推移を，全ジョブ数に対する割合，プロセス数×経過時間でみたときの割合をそれぞれ示したものである．システム管理者，テストジョブユーザを除き，60 秒以上実行されたジョブを対象とした．ジョブ数，システム占有割合ともに，**33 プロセス以上の高並列ジョブが増加傾向にある**のがわかる．65-128 のジョブも常に一定程度は存在している．NWT のときは 64 以下がメインであったから並列度数は少しずつ増えて来ているといえる．

一方，**図 11.12(a)**及び**(b)**は，運用当初からの利用されたスレッド数の推移を，全ジョブ数に対する割合，プロセス数×経過時間でみたときの割合をそれぞれ示した．システム管理者，テストジョブユーザを除き，60 秒以上実行されたジョブを対象とした．最初の頃，ジョブ数としては多いが時間として少ないのは，一生懸命トライはしたということであろうか．その後は，非スレッドのジョブが増加する傾向にあり，**スレッド利用者は減っている**傾向が現れている．また，**5 スレッド以上の多スレッド並列がほとんどない**のもある意味特徴的である．

こうしたプロセス数，スレッド数の推移が示しているある一定の傾向は，あくまで，JAXA のジョブ運用の方針なりパラメータ設定なりに依存していることにまず注意しなければならない．たとえば，129 プロセス以上のジョブが極端に少ないのは，このようなジョブがいろいろな理由から現行の運用では流れにくくなっているからであって，高並列のジョブの実行に問題があるとか流す要求がないというわけではない．そうはいっても，（初期の 3 ヶ月ほどの試行期間を除けば）運用方針やジョブの設定パラメータの大きな変更はしていないので，こうした**並列度が全体として拡大している傾向**は，やはりパラメータ等の設定によって誘導されたものではなく，ユーザ利用の自然な動向として現れてきたものであると解釈できるであろう．

そのあたりを念頭に置いて**図 11.11, 11.12** を見直してみると，プロセス数が増えてスレッド数が減っているわけであるから，基本的にスレッド並列やハイブリッド並列が減って**MPI や XPFortran のみのプロセス並列が増加している**ことになる．その理由としては，1)マルチブロックジョブのようにプロセス数を変えられない場合，**スレッドを使うと利用 CPU 数が増え，ジョブ実行の優先順位が下がってしまう**ので，待ち時間を入れるとスレッド利用のメリットがなくなってしまう，2)**スレッド並列，特に自動並列の性能がそもそも上がらない**のでスレッドを使う気がしない，などが考えられ，ユーザにとってのスレッドの位置づけや使うメリットが明確でなくなっている傾向が表れている．

11.4.4 タスクタイプの推移

図 11.13 に，2005 年 3 月から 2007 年 10 月までの XPFortran と MPI の利用割合（ジョブ数）の推移をプロットした．**未だに XPFortran の利用が 4 割ほどある**が，多少のでこぼこはあるもののなんとなく MPI の利用が少しずつ増えて来ているように見える．

11.4.5 メモリ使用量の推移

図 11.14(a)及び**(b)**は，ラージページ使用量のジョブ数における割合，実際の使用量積算の割合の推移を示したものである．明らかに，**少しずつではあるが大規模ラージページの使用が増える傾向**にあるのがわかる．

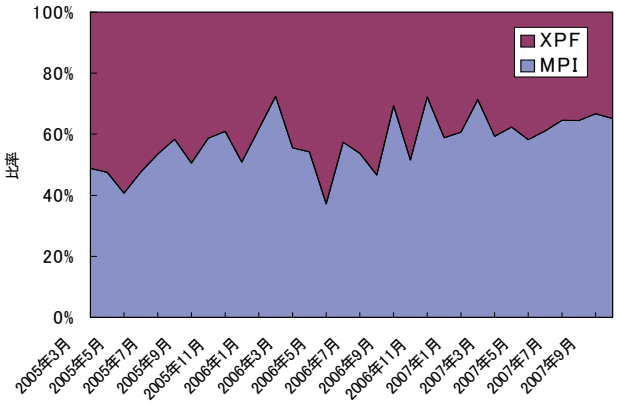


図 11.13 XPFortran 対 MPI の利用割合の推移（ジョブ数）

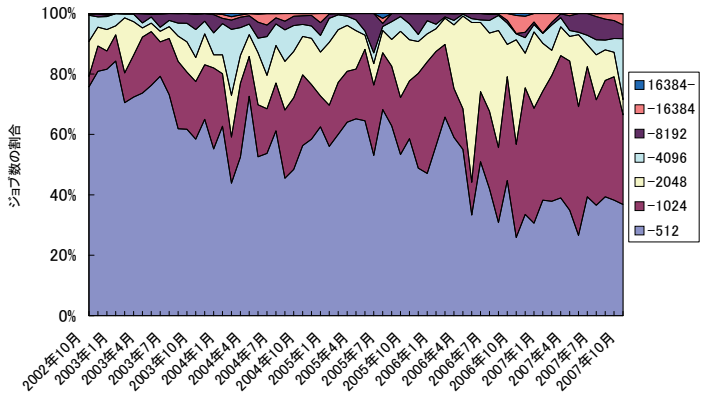


図 11.14(a) ラージページ使用量（ジョブ数）

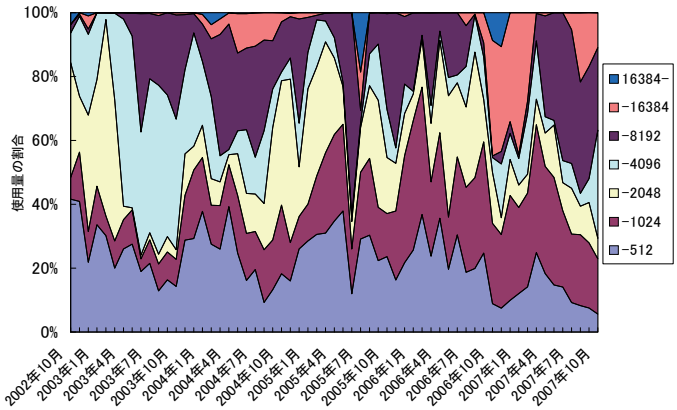


図 11.14(b) ラージページ使用量（積算における割合）

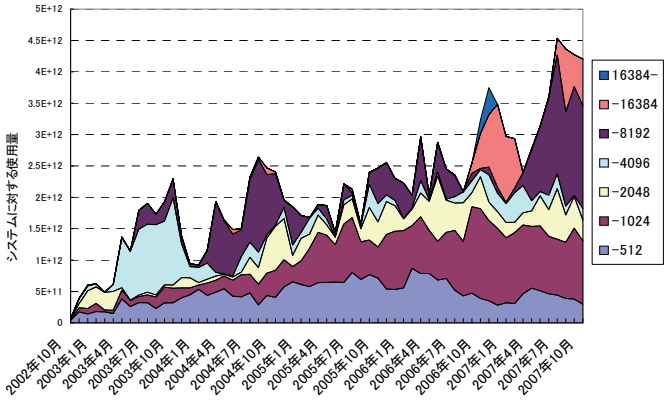


図 11.15 ラージページ使用量（積算）

また，**図 11.15** はラージページ使用量の積算値の推移を示す．積算自体も少しずつ増加しつつあるものの，運用からみると，**図 11.8** に示したように CPU 資源が逼迫しており，メモリ的にはまだ余裕がある状況にある．従って，メモリの有効利用に関しては今のところ運用の切迫した課題にはなっていないが，ユーザの要求メモリ量と実際にジョブで使用されるメモリ量の乖離が激しいなどの問題もあるので，今後は，動的なメモリ解放機構の実現等に取り組んで行く必要があるのかもしれない．

11.4.6 ストレージの利用状況

図 11.16 は，ディスク+テープのストレージの使用量の推移を示したものである．運用当初からのデータではなく，過去 3 年間分の推移であることに注意されたい．HSM による階層ストレージの運用をしているので，**ディスクはキャッシュとして利用され，データが一定量に達するとテープにアーカイブ**される．テープのデータ保管は，保全性を考慮し，ダブルコピーの運用としている．本システムの運用においては，ディスク＋テープのクオータは設定していないので，テープを含めればユーザは必要なだけストレージを利用できることになる．本グラフからいえるのは，**データ量は比較的リニアに伸びてきている**ということである．これは将来のストレージの使用量が比較的予測しやすいことを意味している．ファイル数としては数 100 万ファイルのオーダである．

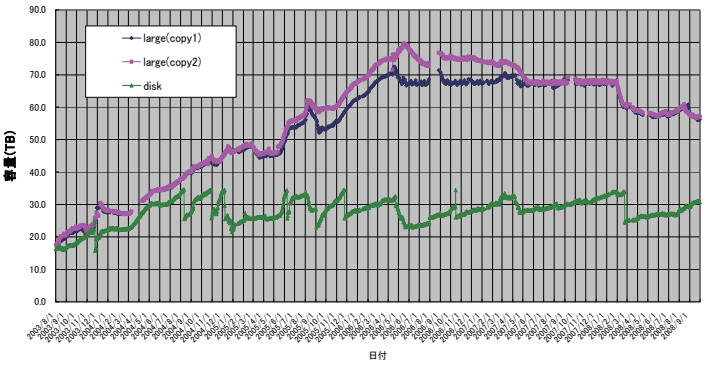


図 11.16 ストレージの使用量の推移

11.5　高度有効利用の工夫と対策[3]

CeNSS システムは，運用側にとってはかなりの部分が試行錯誤の連続であり，どうすればシステムの潜在能力をフルに引き出せるかは，ベクトルや汎用機のような確立されたものがあるわけではなかった．そういう中で，**資源の有効利用と利用性の向上**を中心に具体的な工夫と対策に取り組んだ．以下では，そのうちの幾つかを例として挙げてみたい．

11.5.1　ジョブスケジューラ NSJS の開発と利用

NSJS (Numerical Simulator Job Scheduler) とは，NQS[26] のセンター出口ルーチンとしてジョブの各種事象発生契機において，**図 11.17** に示すように，各種の OS 機能と連携し，CeNSS 独自のジョブスケジューリング機能を実現するプログラムである．NSJS 開発に際し，ジョブスケジューリングに必要となる新たなトリガーや多数のインターフェースが提供された．

NSJS 開発における基本的な思想は以下のとおりである．

- CeNSS 運用の最重要課題である「システムの高度有効利用を最大限に図る」運用システムの実現．
- 小規模／大規模ジョブが公平に混在実行できること．
- デバッグジョブの実行環境を最善なものとし，ユーザのプログラム生産性を向上し得ること．

また，NSJS の主たる機能を列挙すると以下のようになる．

- システム運用計画に従った運転開始／終了処理のスケジューリング
- CeNSS 独自アルゴリズムによるジョブ実行起動のスケジューリング
- ジョブ実行に必要となる各種システム資源の割当／解放等のスケジューリング

NSJS は，CeNSS が有する超高速処理性能を十分に引き出すために本質的な役割を果たしている．一方，センターの経営方針に則り，システムの運用規則を定め，秩序ある運用を実現するためには，各種の運用機能が必要である．実運用においては，規則どおりの運用では対処し得ない事態が多々発生する．そのような事態に即応できる柔軟性も要求されるスケジューラが必要となる．NSJS は多数有する各種パラメータ操作によりこれらの要求に応えている．

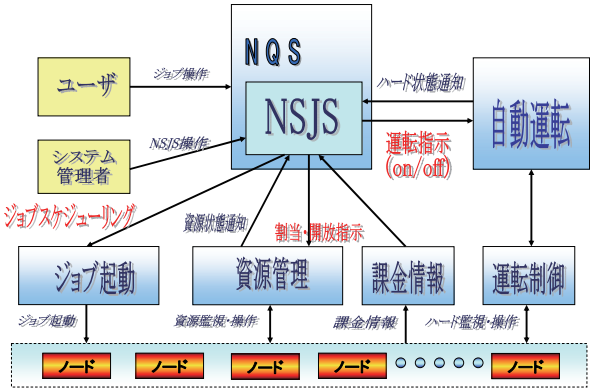


図 11.17　　NSJS の位置づけと基本概念

NSJS は，2003 年 4 月より実運用に供したが，運用期間中の障害発生も非常に少なく，高品質なシステムプログラムとして完成され，順調に安定稼動した．NSJS は，独自開発プログラムという点で，機能拡張や変更等ならびに保守は，必要な時点で容易に早期実施でき，ジョブスケジューラとして一層完成度を高めることが可能である．

NSJS のスケジューリング可能項目を以下に列挙する．

- 運用時間帯ごとの運用ノード，運用ユーザおよびジョブ制限値等の運用パラメータを定義する（運用パターン）．
- 1 年 365 日のカレンダーを定義し，それぞれの運用パターンを定義する．
- 不特定多数ユーザによるセンター運用であるが，画一的ではない柔軟なユーザ管理を実現する．
- プロジェクト開発ユーザグループをプロジェクトユーザとして一括管理する．
- 設備貸付ユーザを定義でき，ユーザのシステム利用を予算管理する．

NSJS のスケジューリングにおける主な特徴を以下に列挙する．

① CPU 資源割当方式

スケジューラが資源管理する資源の要素は，CPU，メモリ，ユーザ DTU コンテキスト，ノード内バリア，ノード間バリアである．CPU の割当方式は，最も CeNSS の性能を引き出すことができる割当方式 **UNPACK 方式を採用**している．これは極力多数のノードに分散してプロセスを割当てる方式である．**図 11.18** に CPU 資源割当方式示す．

② 限定ノード実行方式

他ユーザジョブに大きな影響を及ぼす**特定ユーザジョブを限定したノードで実行**する．

③ マルチブロックジョブの資源割当

マルチブロック構造格子用に**プロセスごとに CPU やメモリを実際に必要な資源量だけを割当てる**スケジューリング方式（使用しない資源は最初から割当ない）．

④ デバッグジョブのスケジューリング

デバッグジョブはレスポンスを保証するために最優先で実行処理する．実行時に割当可能資源が不足している場合には，直近に実行したジョブまたはデバッグジョブと同一ユーザ実行ジョブを**スワップアウト**させて実行する．このスケジューリング方式により，デバッグジョブのレスポンスは劇的に改善された．なお，大多数のデバッグジョブは 10 秒から 30 秒で実行起動できる．

[26] Network Queueing System，バッチ処理を司る標準的なスケジューリング・ソフトウェア．

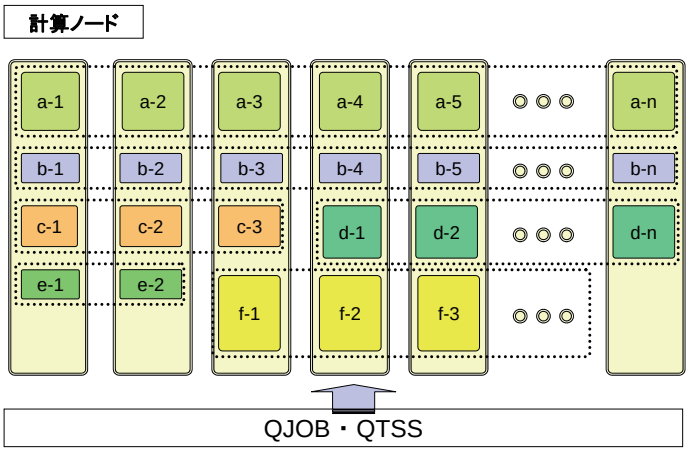


図 11.18　CPU 資源割当方式

11.5.2　ノードの有効利用

システム導入当初より，多数の計算ノード運用において，最も効率的なジョブの割当方式について調査，検討し，試行的に運用を実施した．計算ノードとして運用するノードの資源構成には以下（**表 11.19**）の二種類がある．

表 11.19　ノードの種類

ノード	数量	仕様
Thin	56	32CPU，2DTU，64GB メモリ
Fat	4	64CPU，1DTU，128GB メモリ

当初, Thin ノードと Fat ノードを混在して運用してみた．Fat ノードは, ジョブのプロセスは 2 倍割り当てられる反面, DTU 資源量が Thin ノードの半分なので, 通信やメモリアクセスの競合が Thin ノードより激しく，性能も悪い．このため，混在運用すると，システム全体は Fat ノードの性能になり，大多数の Thin ノードが低性能に合わされる状況が分かった．この結果, Thin と Fat は独立に運用することとした．

ジョブの要求する各ノードの CPU 資源をジョブに割当る方式として以下の 4 種類がある．

完全 PACK 方式：　リクエストを構成するプロセスすべてを同一のノードに割り当てる方式であり，すべてのプロセスを 1 つのノード に配置できない場合は，実行可能リクエストとして選択しない．

PACK 方式：　リクエストを構成するプロセスすべてを同一のノードに割り当てる方式であり，すべてのプロセスを 1 つのノード に配置できない場合は，配置できないプロセスを他のノード に割り当てる式．

完全 UNPACK 方式：　1 ノードに 1 プロセスを割り当てる方式であり，プロセス数が割り当て可能なノード 数より多い場合は，実行可能リクエストとして選択しない．

UNPACK 方式：　1 ノードに 1 プロセスを割り当てる方式であり，プロセス数が割り当て可能なノード 数より多い場合は, 1 ノード に 2 つ以上のプロセスを割り当てる．

CeNSS のジョブ形態および各割当方式の性能調査や試行運用を経て，導入初期から UNPACK 方式を採用している．

次に，ノード内のバリア資源について運用当初は，ジョブへの CPU 割当が自在に行えないという問題があった．当初, ノード内に 8CPU を実装するシステムボードにおいてバリアグループが 2 つ存在し，同一リクエストの各プロセスはバリアグループを跨いで CPU を割当てることができなかった．このため，CPU がアイドル状態でもプロセスに割当られない状況が発生し，稼働率低下を引き起こしていた．この問題は，**バリアグループ方式という垣根を取り除き解消**した．

システムの高度有効利用を阻む問題として，DTU 資源枯渇問題があった．並列プロセスを実行する場合には，プロセスごとに 1DTU コンテキストを割当てる必要がある．1 ノードは 16DTU コンテキスト有し，割当て可能な最大プロセス数は 16 個であった．このため，1 プロセス 1 スレッド構成のリクエストが多いと, 1 ノード内の半数の CPU が DTU 枯渇でアイドル状態となる．この問題は導入から約 2 年後に**共有 DTU 機能が提供され解消**した．共有 DTU とは，16 個のコンテキストのうち，1 つを複数プロセスで共有するというものである．この共有は CENSS における約 4 割強の MPI ジョブに有効な機能となった．

以上のような効率的運用を阻害する問題を解消し，平均 CPU 稼働率 90%を年間クリアしている．

11.5.3　NS アクション管理，WANS，CMS

NS アクション管理とは，センターで扱う障害/Q&A/要望を/そのやりとり/状態を含めてウェブインターフェースにて操作可能なデータベース化し，運用管理の手間を削減するものである．本ツールの運用により，日々発生する種々なインシデントは的確に処理・対応され，かつ，これらの処理･対応状況は逐一運用管理チーム全員に伝わるので，情報共有ならびに，それぞれのノウハウの蓄積にもなり，システム運用管理に非常に有効なツールとなっている．現時点では，NS アクション管理に独自機能の追加と大幅な機能改善の理由から，新たなツールとして **NS インシデント管理（NSIM）**を独自開発し，運用に供している．

主な機能として，NSIM は**図 11.20** にあるように，障害, Q&A, 要望ならびに定例業務等の発生インシデントの登録および対応状況の記録，**図 11.21** にあるような検索による状況表示が可能である．また，画面に検索・抽出した内容を CVS データにダウンロードと累計グラフを表示でき，会議資料として必要となる各種の定型帳表を 1 クリックで作成できる．

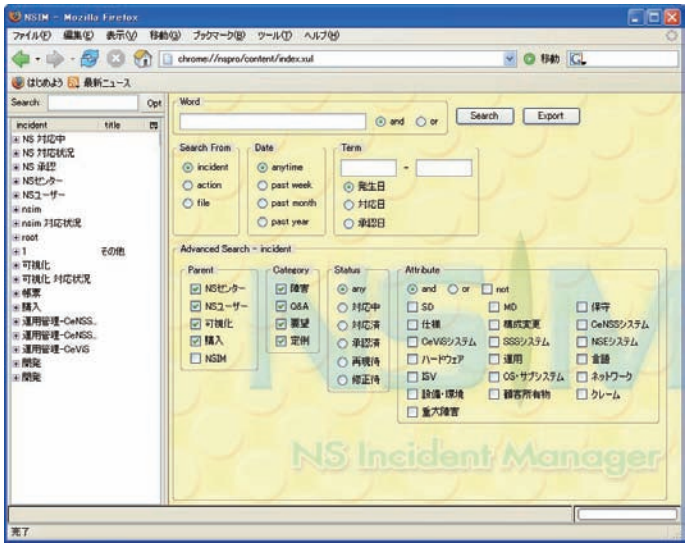


図 11.20 NS インシデント管理（NSIM）

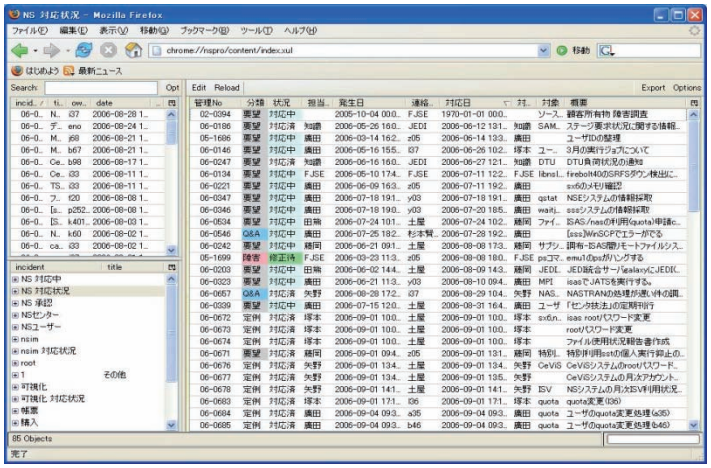


図 11.21 NSIM 検索画面

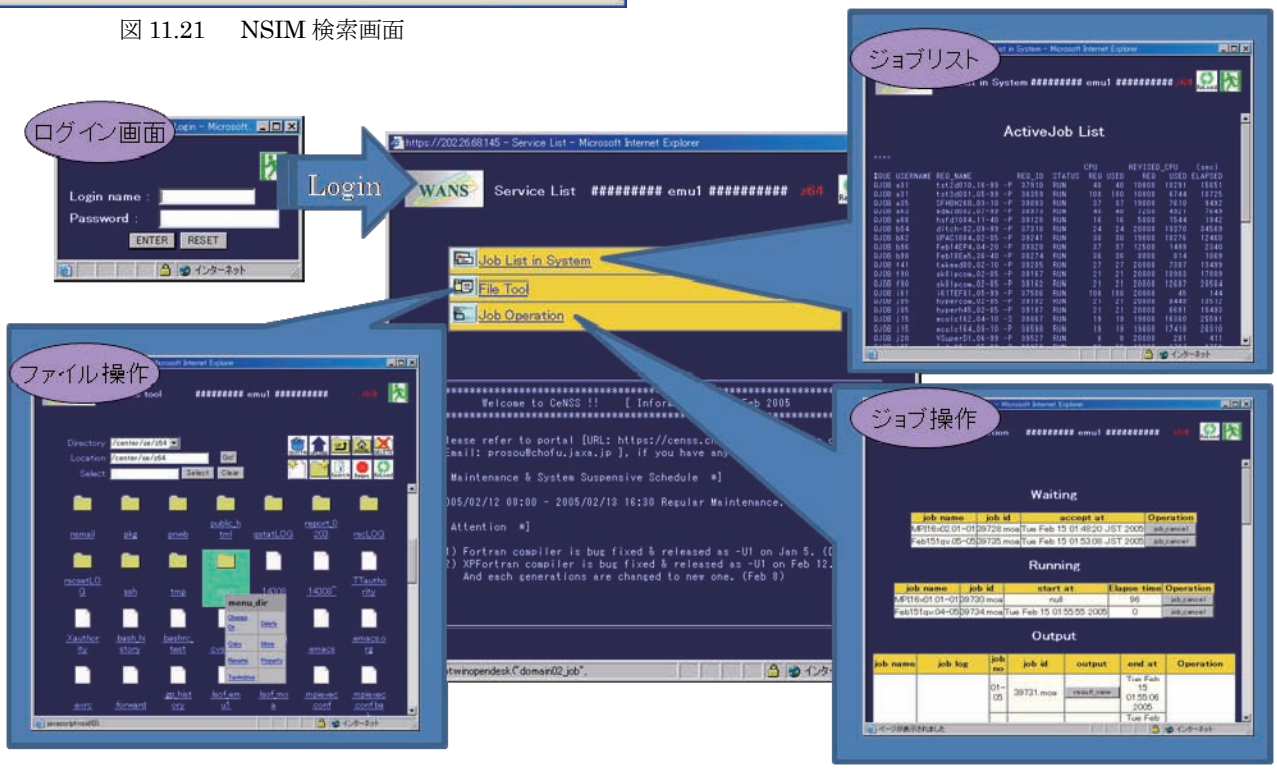


図 11.22 WANS のメニュー画面

WANS(Web Access to Numerical Simulator)とは，ウェブブラウザから CeNSS にアクセスするツールである．普及したウェブ技術を用いて，ユーザがジョブの状況確認や簡単なファイル操作を手軽に利用可能にすることを企図した．WANS の主な機能として，ユーザは CeNSS のシステム混雑状況確認，ファイル編集や操作ならびにジョブの投入から実行結果検索等のジョブ操作が行える．ウェブブラウザから WANS にアクセスし，ログイン名，パスワードが正しく入力されると，**図 11.22** の WANS のメニュー画面が表示される．同画面から，たとえば【File Tool】メニューをクリックすると，**図 11.23** に示すファイル操作画面が表示される．ファイル操作画面からは，ファイルのブラウズ，編集，複写等，各種のファイル操作がウィンドウズ的に行える．**図 11.24** はジョブ操作画面を示す．同図に示されるとおり，ジョブ操作画面からはユーザジョブ状態表示，ジョブキャンセル，ならびに結果表示や削除等の各種ジョブ操作が行える．最新の技術を使えばウェブからほとんどの必要な操作を行えるように作り込むことも可能であったが，メンテナンスの容易さやセキュリティを考えて限定的な操作のみを可能にした．

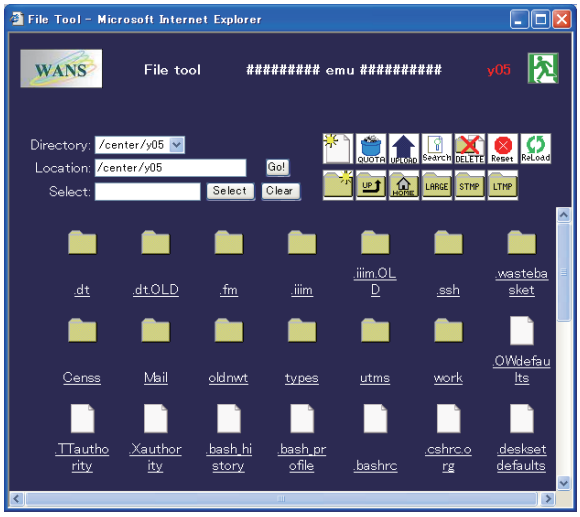


図 11.23　ファイル操作画面

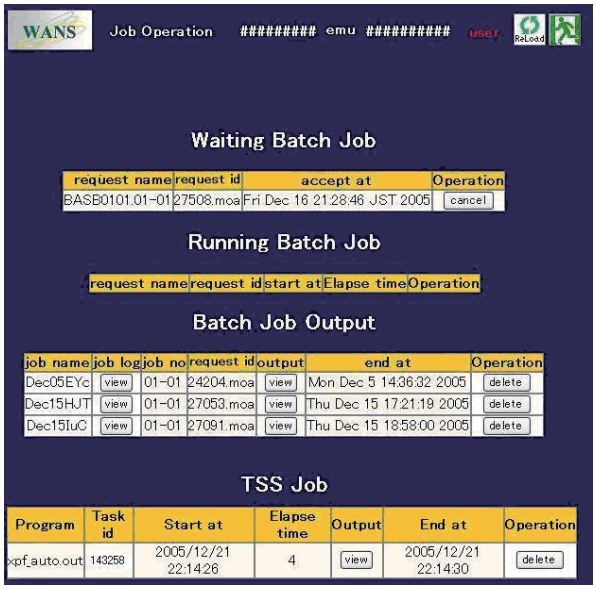


図 11.24　ジョブ操作画面

CMS（CeNSS Monitoring System）とは，CeNSS におけるシステムの利用状態をウェブブラウザによりグラフィカルに表示する管理者向けツール群（一部ユーザにも開放）の総称である．WANS 同様，普及したウェブ技術を用いて管理者が利用状況を手軽に随時確認できるようにすることを企図したものである．以前のシステムと異なり，今回のシステムは，管理すべき資源量が多く，また，どのような状態になるのかの予想もつかなかったので，多角的にシステムの状態がウオッチできるようにした．**図 11.25** に示すとおり，CeNSS System Usage(CPU 利用率)，CeNSS Resource Information（ノード資源利用率），CeNSS Job Map(ジョブ CPU 割当状況)，CeNSS System Monitor(資源利用状況の履歴)，CeNSS Access Log Analysys（ポータルサイトアクセスの解析）ならびに CeNSS CPU Information(CPU 稼動状況)の 6 つのメニューを有しており，詳細なシステム稼働状況をダイナミックにモニタリングすることができる．**図 11.26** の CeNSS System Monitor からは，直近 1 時間前，1 日前，1 週間前，1 ヶ月前のシステム資源（CPU，メモリ，DTU）利用履歴が確認できる．**図 11.27** には，CMS の処理の流れを示す．

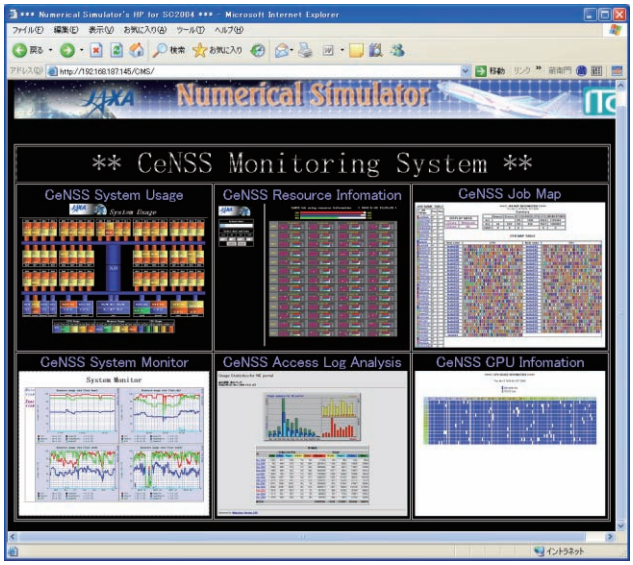


図 11.25　CeNSS Monitoring System のメニュー

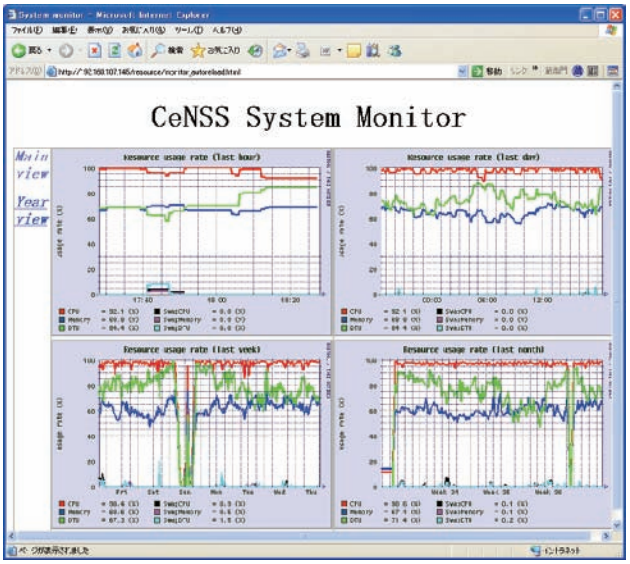


図 11.26　CeNSS System Monitor の UI 画面

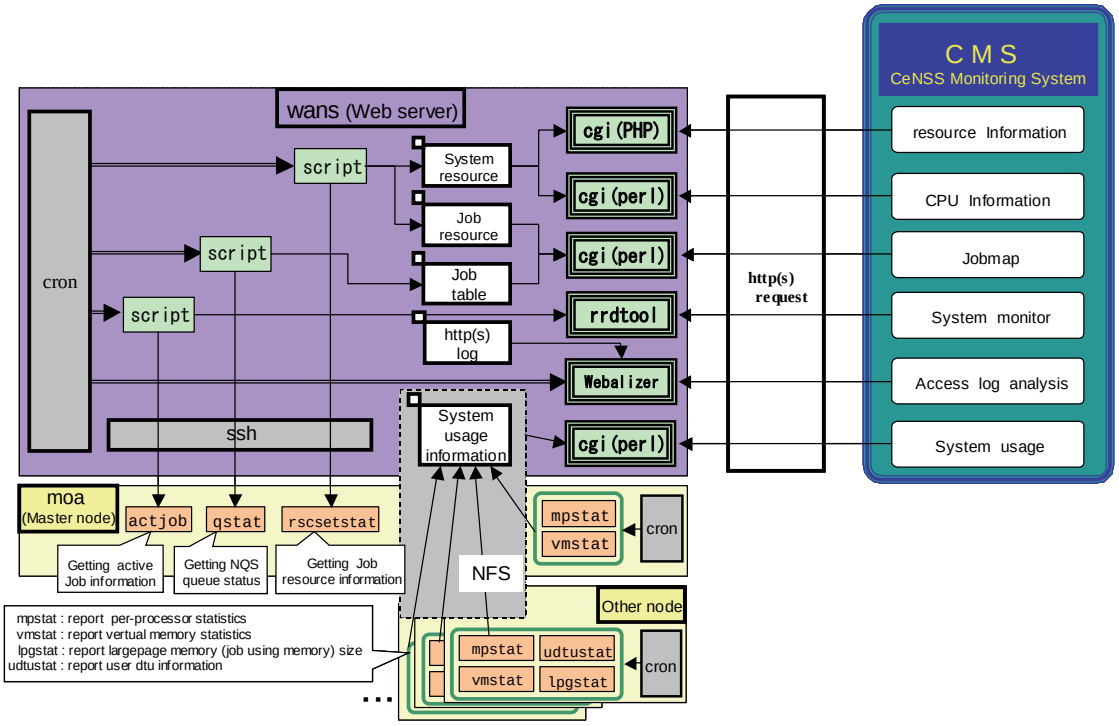


図 11.27　CeNSS Monitoring System の処理の流れ

11.5.4　オープンソースの利用

以下の観点でオープンソースの活用を方針として定め，運用環境の改善に努めた．

(ア) 研究開発環境の標準になりつつある
(イ) 利用者の利便性向上（ツールの追加）
(ウ) 管理者の利便性向上（ツールの追加）
(エ) カスタマイズ可能（使えるものは活用する）

OS として Solaris を採用したことにより，豊富なオープンソースを苦労なく使えるようになったメリットは大きい．NWT の特殊 OS の時に味わった労苦を考えると隔世の感がある．

導入した主な利用者向けオープンソースを以下に示す．

acrobatreader，a2ps，canna，gcc，ghostscript，ghostview，gmake，gnuplot，gv，gzip，g++，g77，ImageMagick，Java，kinput2，kterm，openGL，ruby，ssh，scp，screen，sftp，tcl/tk，tex，wnn xemacs，xv

また，オープンソースをベースにした高速コピーコマンドの実現や，管理者向けとして OpenLDAP，RRDtools，swatch を活用した大規模運用管理ソリューションの実現など，CeNSS では最大限にオープンソースを活用した．詳細は，**表 11.30** の運用工夫のために開発したツール類の一覧を参照されたい．

オープンソースの課題についても言及しておきたい．ユーザ認証という運用の根幹にかかわる部分に，自ら行った評価結果により，当時大規模クラスタの多数ノードに性能面で耐えられうる唯一のものは LDAP であると判断し，OpenLDAP の採用に踏み切った．RAS 性を考慮し，マスタ，スレーブと冗長構成を取ったものの，データの同期化，更新に失敗し，何度か運用を停止せざるを得ない重大障害を引き起こした．オープンソースがゆえベンダーのサポートはなく，結果としてなかなか根本解決には至らず，運用の工夫でなんとか回避せざるを得ない状況になった．

以前は，ツールに過ぎなかったが，今やオープンソースでなんでも揃う時代である．たとえば，

オペレーティングシステム	Linux，Open Solaris…
ファイルシステム	AFS，GFS，Lustre，XFS…
データーベース	PostgreSQL，MySQL…
認証	OpenLDAP，GSI
グリッド	Globus Toolkit…

ただし，上述のとおりイニシャルコストの安さという魅力だけで判断すると，**無料ほど高いものはないという**結末に結びつくこともあり，オープンソースの利用には慎重な判断が必要である．

11.6　システム運用のまとめと課題

CeNSS のシステム運用の 5 年間の推移は，主なところは以上の通りである．この間の変化としては，計算機システムとしては安定期・円熟期に入って行く中で，「**ジョブを確実に処理し安定性を保持しつつ稼働率を最大にする**」というシステム運用の本来的な要求の他に，独法化や宇宙 3 機関統合という大きな流れの下，「**計算需要を効率よく把握し，所望の成果をタイムリーかつ最大限に創出する．多様化しつつある処理要求に的確に応える**」といった運用に対する今までに

ない要求が徐々に出て来ているということである．ここでは，そのような観点も含めて，この 5 年間のシステム運用を総括するとともに，運用の課題についていくつかのポイントに絞って言及してみたい．

11.6.1 ベクトル機からの移行

全部で 2,304 個のスカラーCPU から成る大規模並列システムを，ベクトル機の経験しかない我々が導入し運用するのは，移行も含め，一つの大きな挑戦であったことは確かである．しかし，効率的な日程，スタッフの奮起，ユーザの協力，ベンダーの支援という関係者の相補的な尽力により何とか乗り切ることができたといえよう．特に，JAXA では，後述のようにジョブスケジューラを自前で開発しているので，システムの状態や多様な処理要求に臨機応変に対応することができたことが，実際，効果的だった．また，プログラム移行の観点からは，主アプリケーションである CFD では，プログラムは単純な多重ループ構造の組み合わせとして構成されていることが多いことから，当初，並列に関しては，「**ベクトルループを自動並列ループにマッピングさせる**」（**図 11.28**），配列のデータ構造に関しては，「**C 言語のプリプロセッサを用いて変換する**」というわかりやすい移行モデルを敷いたのが，円滑な移行を実現できた要因の一つと考えても良いかもしれない．無論，大口ユーザに対する個別移行サポートやセミナーの開催，利用手引き整備などによる教育・啓蒙の効果も大きかったと思われる．ただ，**自動並列・スレッド並列は外側ループしか並列化されないなど問題も多く**，**図 11.12** で見たように少しずつ利用者が減っているのを見ると，もう少し周到な初等教育が必要だったかもしれない．

```
        :
  !XOCL PARALLEL REGION
        :
  !XOCL SPREAD DO /IPN
       do 1000 n = 1, nblock
       do 1 l = 1, lmax
       do 1 k = 1, kmax
v      do 1 j = 1, jmax
v        di      = 1./q(j,k,l,1,n)
v        u(j,k,l) = q(j,k,l,2,n)*di
v         :
v        rmu(j,k,l,n) = (cc**1.5)*c2bp/(cc+c2b)
v        turmu(j,k,l,n) = 0.
v    1 continue
   1000 continue
  !XOCL END SPREAD DO
         :
  !XOCL END PARALLEL REGION
         :
```

```
        :
  !XOCL PARALLEL REGION
        :
  !XOCL SPREAD DO /IPN
       do 1000 n = 1, nblock
p      do 1 l = 1, lmax
p      do 1 k = 1, kmax
p      do 1 j = 1, jmax
p        di      = 1./q(j,k,l,1,n)
p        u(j,k,l) = q(j,k,l,2,n)*di
p         :
p        rmu(j,k,l,n) = (cc**1.5)*c2bp/(cc+c2b)
p        turmu(j,k,l,n) = 0.
p    1 continue
   1000 continue
  !XOCL END SPREAD DO
         :
  !XOCL END PARALLEL REGION
         :
```

図 11.28　並列プログラムの移行の例

11.6.2 大規模 SMP クラスタ上でのジョブ運用

SMP ということでまず問題になったのは，ジョブの割り付け方に関する問題である．標準では，プロセス・スレッドを 1 ノードにできるだけ集める PACK という方式と，できるだけ分散させる UNPACK という方式があるが（11.5.2 で詳述），自前のジョブスケジューラでやりくり可能であるし，分散させても空いている CPU にどんどん割り付けた方が最終的には効率が良いだろう，という判断から **UNPACK 方式**を選択した．**図 11.29** に UNPACK 運用におけるプロセスの割付状況を示す．

UNPACK 方式の問題は，性能の低いプロセスが一つあるとノード全体が足を引っ張られて性能が低下する危険があるということだが，幸い極端な性能低下はほとんど出現しなかった．（全体が性能低下していた可能性はある．）

次に問題になったのは DTU 資源不足である．1 ノード 32CPU に対して DTU 資源は 16 しかないので，スレッドを用いないプロセス並列のジョブだけの場合，16 プロセスまでしか入れることができず，結果として CPU が遊んでしまうという問題である．これに対しては，「**共有 DTU**」という機構を導入し，1DTU=1 プロセスの制限をはずすことで解決した．**図 11.8** の 2004 年 7 月以前の CPU 稼働率が低いのは，DTU ネックで CPU をフルに使いきれなかったせいである．ただし，共有 DTU は性能に問題があるので，MPI ジョブ専用とした．

システム構成は，初期の試行段階を除き大きな変更はしなかった．スレッド利用が増えていないので，32CPU/ノードという構成は，初期のままにした．ニーズに応じて，サービスノード，ログインノードにおける処理パラメータを少しいじった程度であろうか．ただ，ノードにユーザジョブのプロセスまたはスレッドをすべて割り付けてしまうと，OS のプロセスが立ち上げられずに性能が劣化する問題（これを「**OS ジッタ**」と呼ぶ）が現われたため，**計算ノード 1 ノードあたりの利用可能 CPU 数を 31 に制限**した．このあたりの資源の有効利用についてはシステム提供ベンダー側に何とかしてもらいたいところではある．

図 11.7 の **CPU 使用時間と実効 CPU 時間の間に乖離**がある問題，**他のジョブの存在により性能がブレる問題**の原因は，当初，SMP 特有のメモリの取り合いによるコンテンションだと思われた．しかし，分析の結果，**コンテンションの影響**

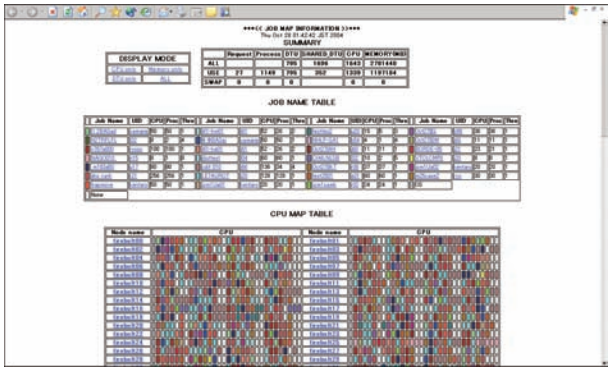


図 11.29 UNPACK 運用におけるプロセスの割付

も相当にある一方で，**DTU における通信待ち**がもう一つの大きな原因であることがわかった．これについては，ジョブ毎に通信のパケットサイズが不均等で大容量通信ジョブがあると待たされてしまうことから，**通信量の均等化**を行うことで通信待ちを多少は緩和することができた．SMP といえばメモリネックという先入観があったので，これはある意味で思わぬところに落とし穴があったという印象である．バンド幅を増やせば解決するのかもしれないが，スケジューリングと通信アルゴリズムによりブレ幅が大きくなっている可能性があり，コストとの兼ね合いもあるので次の世代においては解決すべき課題であろう．

11.6.3 システム管理

運用管理と空調などの設備との連携を完全な形で取り入れたのも，目立たないところではあるが，このシステムの大きな特徴である．大規模システムになると，個別の機器の電源のオンオフやそのシーケンスなどは大きな手間であるしオペミスも発生しやすい．そこで，今回のシステムでは，**図 3.27** にあるように，ほとんどすべての機器をネットワークで接続し，設備管理 PC を通じて監視・指令を送ることにより，電源のオンオフなどの処理を完全に自動化した．スパコンの昨今の電力問題は深刻の度合いを増しているが，空調なども含めた設備全体としての節約を考える時代に来ているのであろうと思われる．実際，**このシステムでは前回の数値風洞システムの電力使用の約 2/3 割程度**で済んでいる．このようなシステムの課題として，初期コストの問題は一番大きいと思うが，一方で，こういった仕組みを作るためのインターフェースが意外にきちっとしていないというのが，実際にやってみてわかったことである．

運用管理からみたときの **Solaris OS** は，UNIX OS としての成熟度も高いので（いわゆる「枯れた」OS なので），**信頼性という点ではかなり高い**のではないかと思われる．障害の遍歴は**図 11.3** の通りであるが，安定期に入ってからは，全系ダウンに繋がるソフトウェアの致命的障害はほとんど発生していない．ただし，サーバ OS であるので，**HPC との融和性の向上**（例えば IPL 時間の短縮など）については，今までのベクトルシステムで培ってきた技術・ノウハウを活かす意味でも，システム提供ベンダーには（実現困難かもしれないが）今以上に是非取り組んでいただきたいところではある．

11.6.4 入出力・ストレージ系

システム構成の観点からいえば，単体 I/O ノードによる入出力は，1 ファイルは物理的にも 1 ファイルにできる，ジョブスケジューラから見たファイル配置が均一でスケジューリングし易い，I/O ノードの実性能を推定し易い，などのメリットがあるが，**I/O ノードの障害 ⇒ 全系ダウン**[27]につながる点で，ハードウェア的には冗長構成にしているとはいえある種の弱点である．この弱点を克服するには，ファイルシステム（つまり，ソフトウェア）として冗長構成とすることが必要となるが，大規模かつ高速 I/O を必要とする HPC システムにおいて，このような冗長構成を採った例は知られていない．今後の課題と考えられる．

データの入出力性能，ディスク＋テープのストレージについては，容量に対する不満は少なく，**図 11.16** にあるように総量は徐々に増加しているが想定の範囲内である．ユーザジョブによる I/O 頻度は著しく増加しているが，入出力性能，安定性，信頼性に関する障害は少ない．ただ，C ライクなプログラミングスタイルや，（MPI の影響と思われるが）**小規模ファイルが非常に多く，ファイル総数は数 100 万に達している**．小規模ファイルの高速処理は全てのファイルシステムにとって不得意なものであるが，今後は小規模多数ファイルの処理系についても，性能向上の努力が必要になっていくと考えられる．また，もうひとつの課題としては，**システム外も含めたファイル共有の要望が出始めている**ので，何とか対応して行けないかと考えている．SRFS: Shared Rapid File System を拡張した SRFS on Ether の活用等が解決の一助となるかもしれない．

11.6.5 処理性能

性能面では，**特定の実アプリケーションではあるが，実効効率で 15%，スケールアップ性能で実効 1TFLOPS 越えを達成**した．実効 1TFLOPS 越えは，CeNSS の導入当初からの大きな目標であり，これを達成できたことは一つの成果だと考えられる．しかし，「実効効率がせいぜい 15%，しかも特定のアプリケーションで」というのは，「ベクトル機の半分程度の実効性能」というフレコミでスタートしたシステムとしては不満が残り，この点に関しシステム提供ベンダーの今後の奮起をぜひ期待したいところである．特に，スレッド並列については，前述の経緯のところでも触れたように，**現在ユーザのスレッド離れが進んでおり，**昨今の CPU マルチコア化路線においてスレッド利用が議論されている中で，どのように位置づけて行くかを早急に明確化する必要があるだろう．

[27] このような障害のパターンを「**シングルポイント障害**」（Single point of failure: SPOF）と呼び，可用性低下の原因となる．

11.6.6 利用性

CeNSS では，NWT 時代に指摘されていた性能面以外での使いにくさ・移植性の問題や近年の計算機のシステムとしての多様化やすそ野の広がりを視野に，旧来型スパコンからの脱皮，旧来型のトップダウン的なアプローチと PC クラスタに代表されるボトムアップのアプローチの融合を目指して，**標準化・汎用化の促進や PC 環境との連続性構築を図るという命題**があった．そのために，構成的な配慮（第 3 章で前述）もしたが，オープンソースの実装や ISV アプリケーションの導入，ウェブからの利用の構築などに積極的に取り組んだ．

オープンソースや ISV アプリへの親和性という点では Solaris OS は極めて有効で，移植性も高く信頼性も高い．ただし，**スパコンとしての性能を出そうとすると，ソースの再コンパイルが必要となり，ISV アプリの場合は問題が多かった**．しかし，スカラーシステムの使いやすさを今後とも活かして行こうと思ったとき，ISV アプリの移植の問題はやはり今後とも無視できないと考えており，HPC システムで ISV アプリを動かす確固たるメリットを構築して行く必要があると考えている．

11.6.7 プログラミングとチューニング

大規模並列プログラミングに対して，プロセス並列とスレッド並列を組み合わせるハイブリッド・プログラミングや性能チューニング技術等の「導入」は比較的スムーズに行われたと思われる．特に，性能評価やチューニングを通じて実性能面へのユーザの関心が集まるようになった（集まらざるを得なかった）のは，低性能なプログラムを少しでも減らして設備としての高度有効利用を図る意味でも有効である．しかし，ハイブリッド並列モデルの理想と現実の差は大きいというのが我々の実感である．今後このモデルをさらに推し進めて行くには，単にハードウェアを改善するだけでなく，環境や指導の充実ということも合わせた総合的な対策が必要なのではないかと思われる．また，チューニングによる性能向上が困難なプログラムが実際に存在し，どのような有効打を打ったらよいかわからない，という宿題も明らかになって来ており，今後の課題と捉えている．

経験によれば，チューニングの効果は，場合によっては非常に大きく出るが，そうでない場合もある．第 6 章で述べたようにチューニングの原則論や事例集もあるわけだが，ケースバイケースや問題規模によるということもあり，勘所を掴むのがなかなか難しい．「スカラチューニングは，ベクトルに比べて難しい．何をどうやってよいかわからない．」という声が多かったような気がする．**経験と勘以外の形式知を如何に積み上げるかが課題**である．また，実際にチューニングを行う場合に，他ジョブの影響を受けて（性能ぶれ等により）チューニングによる差分が明確に出ない場合もあり，**システムとしてチューニングしやすいような環境，有効なツールを如何に構築して行くかは課題**である．

11.7 おわりに

本章では，CeNSS の運用と課題について，統計情報や対処事例を中心に述べた．システム運用は，ユーザという相手があるがゆえにどうしても実務的な業務が多くなり，ある意味では実態を掴みにくく体系化しにくいところがある．スパコンに限らず設備の宿命といえるかもしれない．我々はそこに QMS という仕組みを持ち込み，徹底的にデータ集計した．

ここでは，そのデータを元に分析し，できるだけ**技術的側面**に光を当てることにより，**設備運用という暗黙知の世界を形式知に引き上げ可視化する**ことを試みた．主なところ（あるいは整理できた部分）は述べられたと思われるが，基本的にはここには書ききれない日々の課題との格闘（＝カイゼン）の積み上げが運用という業務の本質であり，従って（良く言われるような）単純なアウトソーシングではおそらく立ち行かないということを付言しておきたい．また，**2005 年 5 月以降は，CPU 稼働率は定常的に 90%を超えており**，ニーズに対して性能が飽和している状況であり，**システムとしては更新の時期を迎えているといえる**．

旧航技研の時代から，ベクトルスパコンの運用経験は 20 年近くあるのだが，スカラーシステムについては，まだまだ駆け出しである．スカラーシステムは多様性が大きいので，運用のセオリーや王道のようなものがあるわけではない．従って，今後とも試行錯誤を繰り返しながら，システム提供ベンダーやユーザとともに運用の経験と実績を積み上げて行くしかないと思われる．以前のようにステレオタイプ的なスパコンの定義がはっきりしなくなっている今日ではあるが，時代やマシンアーキが変わっても**スーパーコンピューティングの本質**は変わらないと考えている．

- **他の追随を許さないほどの超高速性**
- **科学技術の革新に寄与するアプリケーションの存在**
- **共有資源としてのインフラ性**
- **何らかの意味での先進性**

などがその中心的要素であろう．今後ともこの中心要素を常に念頭に置きながらシステム運用を構築して行くのが我々の指名であると考えている．

参考文献

[1] 松尾裕一，航空宇宙技術研究所共用計算機システム「数値シミュレータ III」－その構想とシステム概要，日本航空宇宙学会誌第 50 巻第 586 号，pp.262-269，2002.

[2] 松尾裕一，「数値シミュレータ III」－システム性能特性/航空宇宙への初期応用成果と今後の展望，日本航空宇宙学会誌第 52 巻第 606 号，pp.175-185，2004.

[3] 松尾裕一，土屋雅子 : JAXA 大規模 SMP クラスタにおける運用の課題と工夫, 大規模 SMP 運用 WG 成果報告書, サイエンティフィックシステム研究会, pp.58-84, 2006.

表 11.30 運用工夫のために開発したツール類の一覧

分類	名称	開発元	機能・目的	効果・備考
ｼﾞｮﾌﾞ	NSJS	JAXA	柔軟なジョブスケジューリングと資源割当を行い，稼働率の向上，適切なターンアラウンドを実現する． NQS 出口を利用して以下の機能を実現． 実行順序保障／マルチブロック／概括並列／プロジェクト管理／ プレステージ／年間カレンダ／運転制御など	センタ主導による柔軟なジョブスケジューリングを実現し，稼働率を向上． 他サイトで流用するにはカスタマイズが必要．
管理	NS アクション， 管理	JAXA FJ OSS	センタで取り扱う障害，QA，要望の情報を，そのやりとり，状態含めて Web UI にて操作可能なデーターベース化し，運用管理コストを削減する．いわゆるバグトラッキングシステム，リクエストトラッカー． OSS の RRDtool，PHP 等を利用	記入もれや間違い，確認忘れをなくし，対応をスピードアップ． 他サイトで流用するにはカスタマイズが必要．
管理	CMS	JAXA FJ	CeNSS Monitoring System の略． ジョブの割当状況や稼動情報を容易に把握するため，Web ベースでジョブ状態を表示する．用途に応じて，5 種類開発．	稼動状況の容易な把握，問題点の早期発見が可能． ユーザへの運用状況の提供． イベントなどでの展示． 他サイトで流用するにはカスタマイズが必要．
管理	NSLDAP	JAXA OSS	LDAP によるユーザ情報の一元管理を行うことにより，運用管理コストを削減する． OSS の OpenLDAP を利用．	一般に行われる UID 管理だけではなく，失効管理，quota 情報，使用期間，課金情報，住所，連絡先を一元管理し，かつそれらを GUI 操作可能とし管理業務を効率化． 他サイトで流用するにはカスタマイズが必要．
管理	ログ監視	JAXA OSS	/var/adm/messages 等に出力される重要なメッセージを監視し，即時通報（メール，オペコール），即時アクション（コマンド発行）を行う． OSS の Swatch を利用．	重大障害の早期発見，早期対応が可能． 他サイトでも容易に流用可能．
管理	unyo_check	JAXA FJ	保守前後での確認すべき運用状態などのチェックを自動化（スクリプト化）し，手順ミス，考慮漏れを防ぐ．	環境の戻し忘れなど，保守作業後の単純ミスを撲滅． 他サイトで流用するにはカスタマイズが必要．
利用	WANS	JAXA	Web Acess to NS の略． Web ベースで特別なクライアントを必要としない統一的な GUI で，JAXA 内外からのでシステム利用を可能とする． ジョブ投入／ファイル操作／簡易可視化など	場所を問わず，システムの利用が可能． 海外からの利用実績もある． 他サイトで流用するにはカスタマイズが必要．
利用	iWANS	JAXA	WANS のサブセットとして，携帯電話からシステム利用を可能とする．現状，i-mode のみ対応．	ちょっとした空き時間に場所を問わず，システムの利用が可能． 他サイトで流用するにはカスタマイズが必要．
利用	コンパイラ 世代管理	JAXA	コンパイル環境について，新機能重視なら最新版，安定性重視なら 1 世代前の版，というようにユーザがコンパイラ選択できる環境を提供する．	安全なコンパイラバージョンアップが可能． 他サイトでも容易に流用可能．
利用	TUTRO	JAXA FJ	TUning TRaining Online sysytem の略． 利用者のスキル向上とそれによる稼働率向上を目指し，ユーザフォーラムの内容やその他有益な教育を，e-lerning（ストリーミング）として提供． 富士通製 INTERNET NAVIGWARE を利用．	利用者のスキル向上．いつでも都合の良いときに利用可能． 他サイトで流用するにはカスタマイズが必要．
性能	ﾌﾟﾛﾌｧｲﾗ 診断ツール	JAXA	ジョブの規模が大きくなるとプロファイラの出力情報が大きくなり，解析するのに時間，手間がかかる．そのため，プロファイラの診断結果を分析し，問題箇所を指摘する機能を実現する．	開発中．
利用	NS コマンド	JAXA	利用者の利便性向上のため，簡単にジョブを投入する機構やジョブ状態の整形出力などのコマンド群を NS コマンドとして提供． 例えば以下． 1)acctjob sysrscstat，rscsetstat *1 の情報を 80x24 の画面で参照可能な情報に整形出力する． 2)lsize LPG を考慮したセクションバイト数の出力し，プログラムの使用メモリの概算がわかります．ただし，スタック，動的配列などは含まれない． 3)ns-shell qsub スクリプトを簡素化．Shell を知らなくてもジョブ投入を可能とする．	システムの利用性向上． (簡単な NS コマンドだけ覚えれば，システムが使える)． ns-shell を除き他サイトでも容易に流用可能． *1：複数ユーザから同時実行された場合の効率を考慮し，sysrscstat，rscsetstat の情報は定期的(1min)にログ保存し，その情報を整形している．
利用	高速コピー	JAXA OSS	通常の cp コマンドは I/O 長が小さく(8K,256K,etc)，SRFS に適していない．そのため，ファイルシステムと認識し，最適な I/O 長を選択する高速な cp コマンドを実現する． GNU の fileutil を改造．	SRFS 間でのコピー性能が１１倍 （23.4MB/s→258.9MB/s）に改善． 他サイトで流用するにはカスタマイズが必要．

第 12 章　CeViS の運用分析と課題

12.1　はじめに

可視化システム CeViS は，2001 年 4 月に導入された．それまでは，センターマシンとしての CRAY YMP M92 とグループ規模で利用するワークステーションが可視化プラットフォームの中心であったが，性能の陳腐化及び機能不足と計算システムからのデータの流れの問題から導入が決められたものであった（文献[1]，**付録 F** 参照）．本章では，導入後の運用と利用状況についてまとめ，課題を抽出する．

12.2　CeViS を含む可視化環境に対する基本的考え方

CeViSシステム導入当時の可視化環境に対する考え方は，当初は「**データの大きさやニーズによって可視化資源を適切に使い分ける**」という作戦を基本にした．それは，2001年当時の手元端末（パソコンやワークステーション等）では高性能スパコンからの大規模データを処理するのは困難で，特殊な可視化装置が必要だったからであり，さらには，そうした特殊可視化装置によって提供される画像と可視化の研究開発の動機のシナジー効果を期待したからであった（**付録F**参照）．

可視化資源の「使い分け」に関しては，最上層の大規模データの可視化や特殊な可視化は「中央可視化システムCeViS」を利用することとし，一番ニーズの多い通常の可視化には，運用側で用意した数台の可視化専用端末や研究者の手元のPC等を利用する．ここで，「通常の可視化」とは，汎用可視化ソフトの扱える範囲の可視化を意味している．汎用可視化ソフトとしては，**AVS/Express**，**FIELDVIEW**，**Tecplot**を導入した．FIELDVIEWは，航空宇宙の可視化に特化したもっとも古くから使われている可視化ソフトであり，機能は特定されるもののそれなりに洗練されている．一方，AVS/Expressも古くからある可視化ソフトではあるが，航空宇宙に特化したものではない．モジュールを組み合わせることにより，ユーザ特有の可視化を実現できるところに特徴がある．Tecplotは，実験データの処理も含めたポスト処理用のソフトであり，論文に載せられるグレースケールの絵図を簡単に作ることができるので根強い人気がある．

12.3　可視化システムの運用・利用の実態と課題

以上のような考え方の下に，CeViS の運用を決めて行った．CeViS の可視化ソフトウェアとしては，多画面出力に対応している市販の **EnSightGOLD** と **AVS/MPE** を利用することとした．両方ともステレオ表示にも対応している．また，plot3D をはじめとする各種のデータ形式にも対応しているので，FIELDVIEW 等と同様の使い勝手で利用することができる．ボリューム表示などのツールを一部自作した[2]が，運用で利用するには至らなかった．さらに，マニュアルの整備やソフトウェアのメンテナンスに課題がある．

運用については，一度に沢山のユーザが使えないという特性を配慮して，時間制限を設けることとし，1 日 3 交代（1 回 2 時間午前 1 コマ，午後 2 コマ），ウェブからの予約制とした．予約画面を**図 12.1** に示す．

利用内容からいえば，**システム紹介や計算内容の紹介などのプレゼン用**が圧倒的に多く想定以上に使われた．可視化サーバがなくなった時点（2008 年）でも，見学者や幹部等へのプレゼン用として使われる機会があることを踏まえると，一定の役割を果たして来たといえる．また，そういった用途へのこの種のシステムの必要性をある程度説明できるであろう．ただし，その場合には**コストとのバランス**（プレゼン用にどこまで投資するか）が合理的に説明される必要がある．ちなみに，見学者とか訪問者の年度別の総数を**表 12.2** に掲げる．また，**図 12.3** には，見学者等へプレゼンで使われている例である．

図 12.1　CeViS の予約画面

表 12.2　CeViS の見学者数

年度	2003	2004	2005	2006	2007
見学者数	649(注)	1,270	1,500	1,038	355

注）　2003 年の 10 月からの数字

図 12.3　CeViS の利用されている様子

一方で，研究のための可視化に使われた頻度はあまり多くなかった．その理由を幾つか考えると，

① そもそも必要とするユーザが少なかった

② 新しいユーザの開拓ができなかった

③ そこに行かないと使えない等使い方が馴染めなかった

④ 手元の端末が急速に進歩し，使う必要がなくなった

などが挙げられ，この種のシステムの運用の難しさをある意味で露呈することとなった．ただ，周知の通り，GPU やグラフィックスボードをはじめとする可視化コモディティ製品は，急速な高性能化，低廉化，標準化が進んでおり，他にもまして④の要素の影響が大きかったと思われる．将来的には，大規模データの可視化が特殊な装置を用いずとも手元の端末や PC サーバでいとも簡単にできるような環境が整ってくる可能性さえあるのではないだろうか．

研究のために作られた可視化画像としては，**図 12.4** の平行平板間乱流の DNS における壁面付近の大規模構造のボリューム可視化の画像であるとか，**図 12.5** の浮き上がり火炎の DNS における空間の流れ構造の等値面表示の画像などが例として挙げられる．これらのシミュレーションは非常に大規模であり，パソコンのメモリや表現力では能力的に不十分であるため CeViS システムが利用された．もともとこの種の大規模計算の数自体が多くないため，利用者数が増えなかったともいえる．

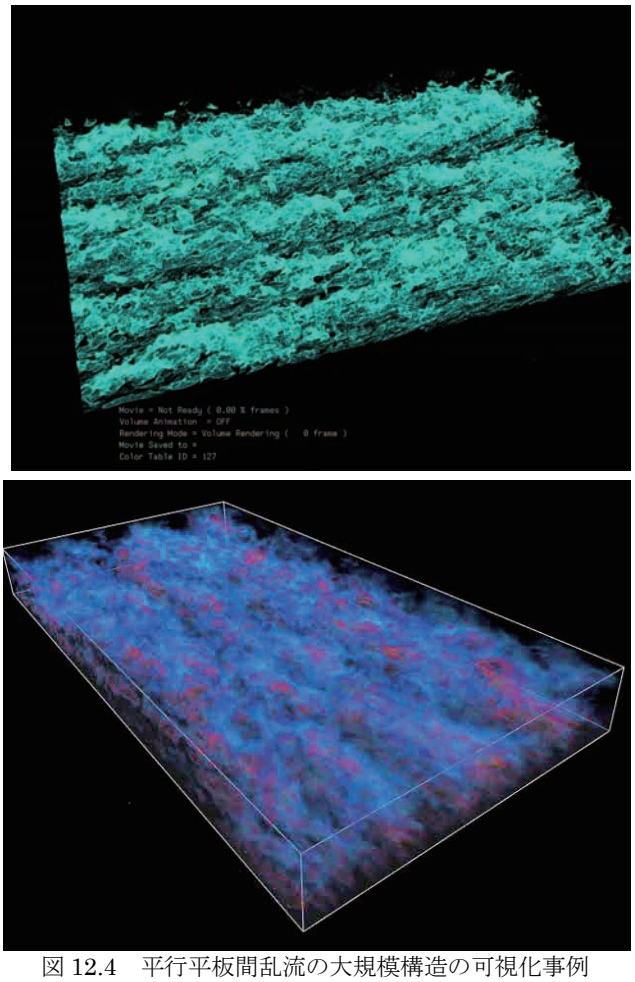


図 12.4 平行平板間乱流の大規模構造の可視化事例

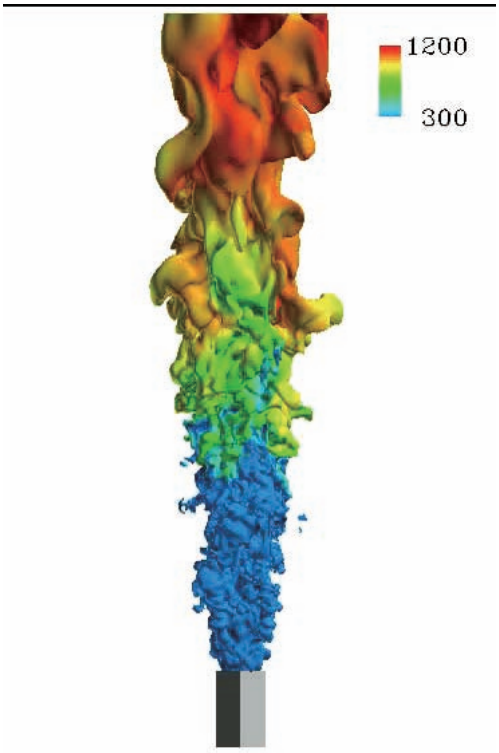


図 12.5 浮き上がり火炎の空間温度分布の可視化事例

可視化手法の研究や開発という側面からは，近年，技術的にはかなり成熟して来た感がある．線画や色塗り画，パーティクル追跡などの通常良く使う基本的な機能のほとんどは汎用可視化ソフトにすでに取り込まれており，GUIなどの充実ぶりを考えると，今さら可視化ソフト/ツールを一から自作する動機は乏しい．ただし，ハードの進歩を利用した処理の高速化であるとか，新たな表現法の開拓などといった研究開発要素は未だ残っており，機会は少ないとはいえ，LIC（Line Integral Convolution）法や高品質ボリューム表示法等を開発して来た経緯がある．しかし，CG等に関する専門的な知識や技術が必要とされるようになって来ており，**JAXAとして，可視化に関する研究開発と実用のバランスをどのあたりに置くかが課題**と思われる．

可視化の利用という側面から見てみると，工学CFD解析における可視化では，工学レベルで見られれば良い，あるいは，必要な部分を必要な精度で迅速に可視化することが利用者の興味の中心であり，特に新しい手法とか見方へのニーズはあまり強くないように見受けられる．この場合の最大の課題は，普通の可視化を如何に高速かつ柔軟に行えるか，ということであり，逆に言えば，設計や開発に役に立たない可視化は必要とされない．具体的には，マルチブロック，非構造といった各種データ構造への対応や，パラメータスタディによる多数のデータセットの効率良い可視化，設計開発に役立つ新たな使い方の模索などが課題となっている．

CeViSの導入，運用から見えてきた課題の一つに，大規模データの可視化の考え方・あり方がある．ある程度以上の規模のスーパーコンピュータからは大規模データは必ず出てく

るが，そのようなデータを出すユーザ数は少ないのが常であり，**レアなケースのために運用側としてどこまで可視化環境を整備するか**ということである．I/Oなどと同じく大規模データ用の可視化装置は，オーバーヘッドや同期機構のため，小規模データの可視化にはむしろ向かない．結果として，稼働率や使い勝手に課題が出てしまう．CeViSは，中央可視化システムを置くという考え方に対する究極のチャレンジであったといえないだろうか．結果的には，その導入は十分な成功だったとはいえないかもしれないが，いろいろな教訓やノウハウを得ることができた．純技術的な側面からは，**付録F**に示したように，**大画面の可視化技術や3次元表示に対する構築技術**を取得することができた．

12.4 おわりに

本章では，中央可視化システム CeViS の運用・利用の実状と課題を俯瞰した．CeViS 自体は，可視化サーバが陳腐化した時点（2007 年 10 月）で，導入時の姿ではなくなっている．我々は，CeViS の導入を通じて，当時の技術としては最高の可視化システムを構築し運用した．性能面はともあれ，これ以上の機能を持ったシステムはおそらく当分はできないであろう．しかし，それでもいろいろな課題が出て，ユーザの満足の行くものとはならなかった．さらなる分析は必要だが，**ユーザインターフェース部分の構築の難しさや後処理のユーザニーズに対する適切な対応は如何にあるべきか**は今後の課題であり，注意していく必要がある．

参考文献

[1] 松尾裕一：航技研新中央可視化システムの概要とその戦略， 航技研特別資料 SP-53，2001，pp.149-154.

[2] 上村周平，松村誠，柿本正憲，松尾裕一：Pre-Integration 法を用いた高品質ボリューム可視化ライブラリの開発，第 33 回可視化情報シンポジウム講演論文集，(2005)，pp.263-266.

第 13 章　NS-III の利用，航空宇宙への初期応用成果と将来展望

13.1　はじめに

本章では，NS-III の（ユーザ側からみた）利用の概要および航空宇宙分野への初期応用成果，NS-III を使った数値シミュレーションの将来展開も含めた可能性展望や技術課題について論ずる．

13.2　NS-III の利用概要

分野別の利用割合の年度ごとの変遷を**図 13.1** に示す．ここで言う分野とは厳密なものではなく，ユーザ毎に業務（分野）を振り分け，分野ごとの利用時間の積算を百分率で表したものである．また，NWT 時代からの変遷を追っている．これを見ると，数値シミュレーションが大局的にどのようなところに利用されて来たかがわかる．NS-II 時代における平成 11 年頃の宇宙利用の増大は，宇宙往還機プロジェクトによるものである．平成 13 年に航空の利用が増えているのは，小型超音速機プロジェクトによる．**最近になって，宇宙利用が増えだしている**のは，JAXA になって，数値シミュレーションがロケットエンジン等の宇宙関連に利用されてきだしていることによるし，航空利用が堅調に推移しているのは，国産航空機や国産エンジンのプロジェクトが動いていることによる．反面，基礎分野が最近少しずつ減りつつあるが，これは数値シミュレーションがプロダクション的に実問題に利用されるようになってきている証しでもあり，JAXA という組織の性質上，やむを得ないところかもしれない．一般に **Capability Computing** と呼ばれるシステムリソースの大半を使うような種類の計算はここの範疇に含まれるが，数値シミュレーションがプロジェクトにおける設計や開発の支援に使われるようになった今日，パラメータ解析などの規模は小さいが重要な計算（**Capacity Computing**）が常に動いているという状況を変えることは困難であり，将来のテーマの芽だし玉だしの役割を担う Capability Computing への配慮は別途必要になると思われる．

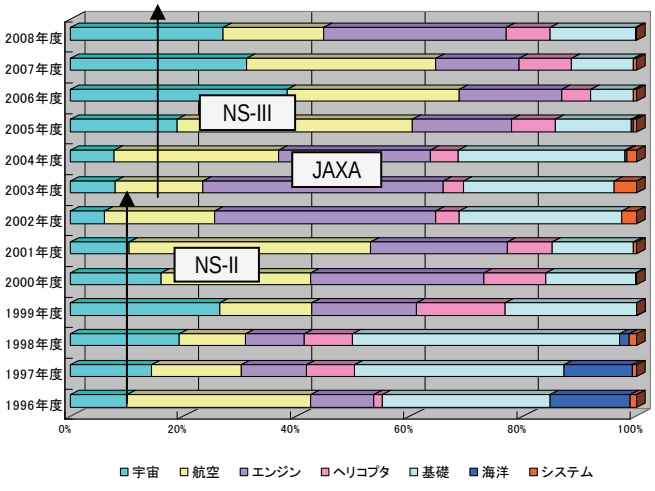


図 13.1　分野別利用割合の変遷

一方，2003 年度からは，「**特別利用**」という形で，ジョブの優先利用形態を実施した．特別利用とは，毎月行われる特別利用審査会で，テーマを選択し，そのテーマに係るジョブを優先実行させるものであり，主にプロジェクト対応のために結果の時期が決まっているような場合に利用される．特別利用として，次（**表 13.2**）の 2 種類の形態を考慮した．

表 13.2　特別利用の形態

種類	内容	選定方法	課題数
年間（長期）特別利用	年間を通じて計画的に実施するもの	運営委員会で年度当初に選定し計画する	数課題
一時（短期）特別利用	月単位で短期に計画的に実施するもの	利用審査会で利用戦略に基づき選定する	必要に応じて

優先実行の方法は，次の 2 通りとし，どちらにするかは，ジョブの内容，混み具合，期限などによりその都度判断して決めることとした．

① 通常運用時間帯において，必要数のジョブを優先的に実行させる．ただし，ジョブの制限は通常ジョブと同じとし，同時実行数は原則として 1 とする．（優先実行）

② 通常運用時間帯以外（夜間・週末）の時間帯において，必要な数のジョブを優先的に実行させる．ジョブの制限や同時実行数の制限は必要に応じて別途定める．（特別実行）

必要な手続きは，短期利用の場合は**表 13.3** とし，透明性と公平性を確保した．

表 13.3　特別利用（短期）に必要な手続き

ｽﾃｯﾌﾟ	手続き	補足
1	課題の申請・受付	申請書：　様式 1（NS システム HP より DL 可能） 受付窓口：調布 IT センターソフトウェア利用係 期限:　実施前月の 20 日まで
2	課題の審査と選定	利用審査会
3	運用計画の策定	担当部署
4	運用計画の通知	NS システム HP に公開
5	実施	
6	利用報告書の提出	報告書：様式 2（NS システム HP より DL 可能）

13.3　航空宇宙における数値シミュレーションの動向と NS-III の初期応用成果

JAXA におけるスーパーコンピュータを用いた数値シミュレーションは，旧航技研時代から含めて 30 年近い歴史がある．その利用は，いうまでもなく，流体・空力を中核とする **CFD (Computational Fluid Dynamics)** が中心であったが，最近，その様相は少しずつ変わって来ている．ここでは，そのあたりを念頭に入れながら，現行の CFD の一般動向も踏まえ，NS-III の初期応用成果について報告する．なお，NS-III

の成果ついては，各年度の利用成果報告書を参照されたい．

数値シミュレーションの航空宇宙の‘基礎’へ適用という意味で従来から続いている代表選手は，**「乱流」の DNS（Direct Numerical Simulation）**である．全ての渦を解像するという DNS の主旨から必要な格子点数（空間解像度）はレイノルズ数の 9/4 乗に比例する．従って，特にメモリを必要とする大規模計算になる．NS-III において，Re_τ=1,020 のチャネル乱流の DNS を，2,048×448×1,536（14 億点）の格子により実施し，数ヶ月に及ぶ計算の末，統計的に妥当な結果を得た[1]．**図 13.4** は，壁近傍の渦構造（高速・低速領域）のスナップショットを示したものであり，大規模構造が中心付近まで達しているのがわかる．**図 13.5** は，統計処理した平均速度分布を示す．このレイノルズ数になると対数速度分布の直線部分がはっきりと現れているのがわかる．

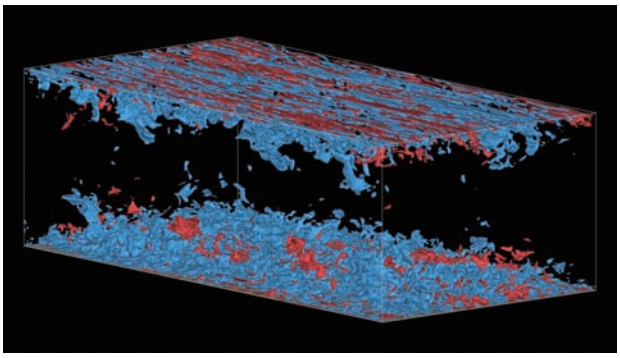


図 13.4　Re_τ=1,020 チャネル乱流 DNS の壁近傍渦構造

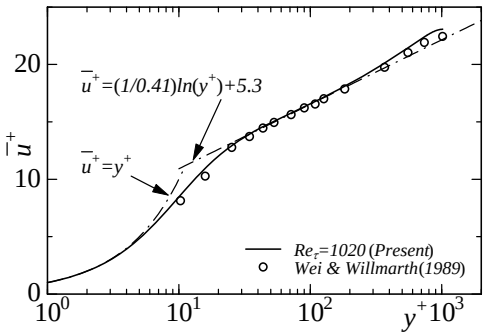


図 13.5　Re_τ=1,020 チャネル乱流 DNS の平均速度分布

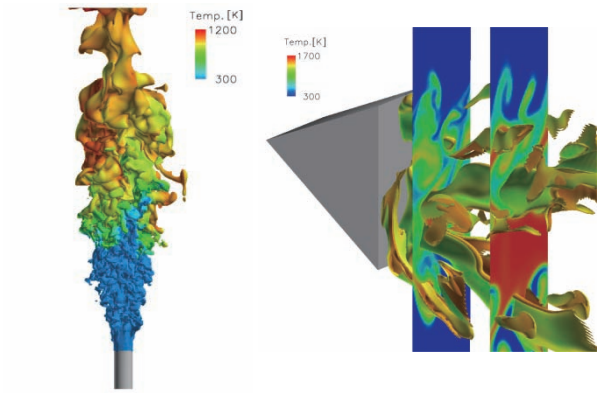


図 13.6　DNS による水素浮き上がり火炎，メタンガス予混合火炎の温度分布

乱流 DNS と並んで学術ツールとしての地位を固めつつあるのが**「燃焼や反応流」の DNS** である．ただし，乱流的な挙動に加えて化学反応も解かなければならないため，乱流以上に大規模計算（メモリ，速度とも）が必要となり，実在火炎への挑戦は，始まったばかりである[2][3]．**図 13.6** は，ノズルから噴出する水素の浮き上がり火炎及び燃焼器後方のメタンの希薄予混合火炎の DNS で得られた温度の瞬時分布を示したものである．メタンの DNS では，20 の化学種と 64 反応を 1 億点以上の格子により解き，希薄予混合燃焼の保炎・消炎・再着火機構を明らかにしようとしている．

LES（Large-Eddy Simulation）といえば，従来，DNS の補完としての現象理解などの基礎分野での利用が主だったと思われるが，近年，LES の実用問題への適用が増えている[4]．これは，計算機性能の向上によって従来難しかった大規模 LES が卑近になって来たこともさることながら，剥離や遷移，振動，物体移動などを伴う場合にアンサンブル平均に基づく RANS（Reynolds-Averaged Navier-Stokes）解析の適用限界が見えて来たこと，手法の進歩（ダイナミック SGS[5]モデルや流入条件等）や検証の実施により LES の適用能力が向上したこと等に因る．とはいっても，航空宇宙で扱う高レイノルズ数流れに LES を適用するには格子要件が厳しすぎるので，特に最近，壁近傍を RANS で近似し，遠方場を LES で解く **DES（Detached Eddy Simulation）**と呼ばれる手法が注目を浴びている[6]．**図 13.7** は，DES により，NACA64A-006 翼まわり高迎角流れを解析した結果[7]である．

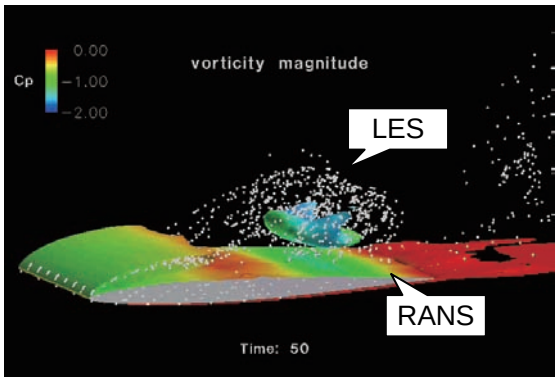


図 13.7　DES による NACA64A-006 翼まわりの高迎角流れ

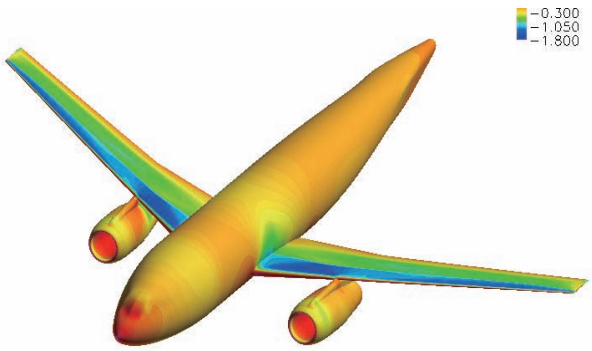


図 13.8　DLR-F6 翼胴エンジン結合体形状の解析

NS-III は，前述の入出力性能とストレージ容量により，こういった非定常計算に強みを発揮する．このような条件の流れに RANS を適用すると，渦粘性が過大になり剥離域が小さく出る傾向があり，CFD の弱点とされて来たが，LES や DES の利用によりそのあたりの懸案に何らかの解決策，改善策を与えられる可能性がある．

RANS 等の CFD の‘実用的な利用’は，計算機の進歩，市販コードの充実などと相俟って急速な発展を遂げている．設計などへの一般的な利用の範囲では，RANS 系統のオーソドックスな定常解析が主であったが，NS-III の登場により，実用複雑形状解析，大規模非定常解析，数値風洞，多分野統合解析，各種最適化などが確実に我々の手中に納まりつつある．NS-III は，元来この種の応用を目途に設計されていることは前述した（第 2 章）．

複雑形状解析は，CFD のプロジェクト等実開発への適用という意味ではある種自然な流れといえる．複雑形状問題の典型は，航空では翼胴エンジン結合体，高揚力装置など，宇宙では，往還機や分離の問題などである．特に，高揚力装置は，現在進められている小型航空機プロジェクトにおいても重要なテーマの一つとなっている．NS-III を利用した 1,000 万点規模の翼胴エンジン結合体あるいは高揚力装置の大規模解析が始まっている．**図 13.8** は，AIAA の第 2 回抵抗予測ワークショップの課題形状 DLR-F6 のマルチブロック格子による NS 解析の例[8]である．

複雑形状解析で，現在，非常に高い障壁となっているのが「**格子生成**」である．航空宇宙分野において格子生成が難しいのは，壁面境界層や衝撃波が局所的な現象であるがゆえに格子点の適切な配分や集中が求められること，要求精度が極めて高いので直交性や滑らかさに配慮した良質の格子を作成しなければならないこと等に起因している．構造格子系では，マルチブロック法やオーバーセット法が洗練されて来ており，JAXA では，UPACS[9]と呼ばれるマルチブロック構造格子汎用 RANS コードを開発し，実用複雑形状へ対応している．**図 13.8** の例では，200 以上のブロックが使われている．ただし，現状，良質の格子の作成は専門スタッフの腕に頼っており，高い解析精度は得られるものの自動化や柔軟性に難がある．そのあたりを踏まえ，非構造格子や直交格子を用いる手法の検討・開発にも着手している．

非定常解析は，NS-III の力量が真に発揮される領域の一つである．**図 13.9** は，重合格子を用いて，遷移飛行するヘリコプタの回転ブレード及び胴体まわりの流れを 200 万点程度の格子を用いて非定常的に解いたもの[10]である．物体表面は圧力分布を示し，パーティクルは，ブレードと後流渦の干渉状況を表している．ブレードに固定した回転格子，胴体に固定した移動格子と静止した背景格子の間で情報交換している．**図 13.10** は，ジェットエンジン内の多段圧縮機内の非定常流れを解いたもの[11]であり，瞬時のマッハ数分布を示す．この種のターボ回転機械の解析では，従来，回転方向の流れの周期性を仮定し，静翼・動翼のセットから成る単段の解析を行ってきたが，ここでは周期性を仮定せず，翼枚数

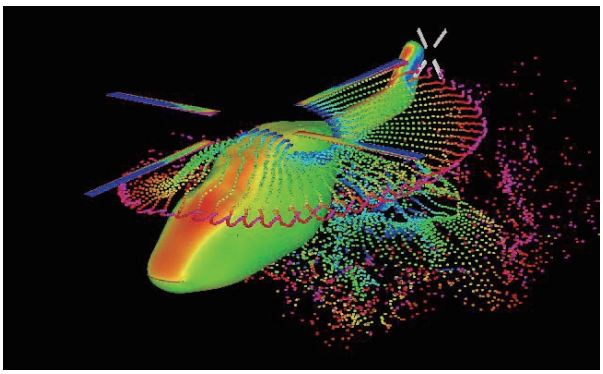


図 13.9 遷移飛行するヘリコプタのブレード胴体干渉流れ解析

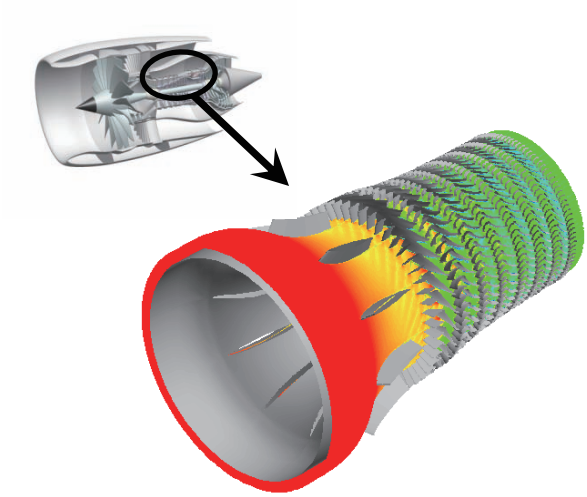


図 13.10 多段圧縮機内の非定常流れ解析

等を実態に合わせて複数段を同時に解く手法を採用している．格子点数は 7,000 万近くなるが，NS-III の持つ大容量メモリと並列計算により可能となっている．なお，この種の非定常解析は，Unsteady RANS ということで，URANS と呼ばれることがある．

「**数値風洞**」とは，前システム NWT のコンセプトだったが，NS-III になってようやく名実ともに真の数値風洞が実現されつつある．**図 13.11** は，JAXA 遷音速風洞の試験部（模型付）を，抽気やスティングを考慮に入れて解いたものであり，それらの影響の定量化が解明されつつある．

多分野統合解析関連では，フラッタや静的空弾，あるいは空力加熱といった伝統的な空力-構造，空力-化学反応といった連成問題に加えて，空力-熱伝動等の新たな分野の連成への取り組みが始まっている．**図 13.12** は，タービンブレードの冷却の問題を，流体と熱伝導を組み合わせて解いたもの[12]である．

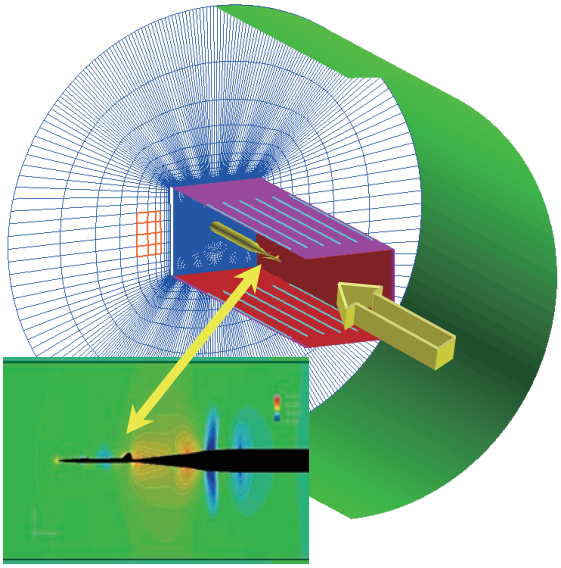


図 13.11　風洞の試験部を解析した例

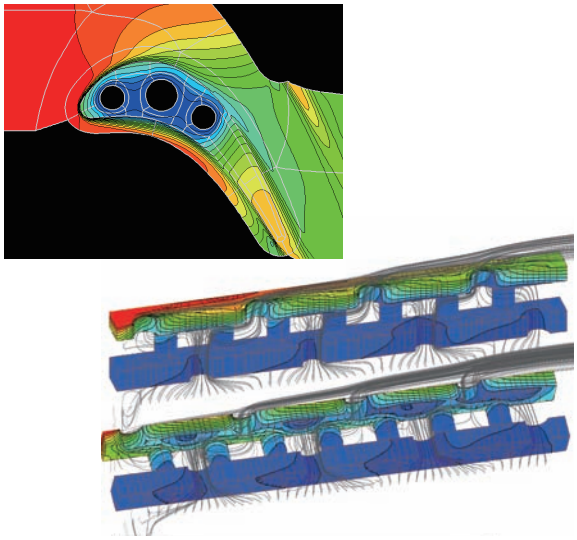


図 13.12　タービンの冷却に関する流体ー熱伝導連成解析

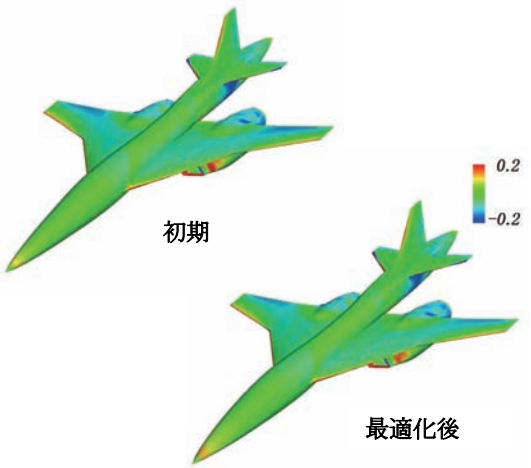


図 13.13　SST 機体形状への逆問題設計の適用

最適化は，SST プロジェクトにおいて取り上げられて以来，急速に進んだ分野[13]である（**図 13.13**）．いろいろなレベルがあるので，全ての場合に計算機資源を要求するわけではないが，基本ソルバーに RANS や LES を使ったり，ループをまわすとかパラメータを振ったりすると計算量は急速に増え，本格的な利用は NS-III のようなシステムにおいてはじめて可能になるものである．

以上，大規模問題や大きな流れに沿うものを中心に説明して来たが，NS-III の昨今の状況を概観してみると，

i) 多分野統合解析等に代表される複雑多様なシミュレーションが増加している．

ii) コード開発や可能性提示のためのランより，実際の設計開発で利用するためのプロダクションランが増えている．

といった利用状況の変貌がうかがえる．

一方で，前述の格子生成の他に，DNS/LES への適用を睨んだコンパクトスキーム等の空間高次精度スキームの開発，非構造格子ソルバーのさまざまな改良，実用乱流モデルの検証といった課題への取り組みもみられる．計算技術的には，並列計算への対応や大規模データの可視化が現在の主題と思われる．並列計算は，構造・非構造を問わず，領域分割に基づいて MPI によりプロセス並列化する手法が現時点で主流を占めている．大規模 SMP システムで，ノード間でプロセス並列，共有メモリのノード内でスレッド並列というハイブリッド並列モデルを採用している例も数は少ないがある．

13.4　NS-III およびポスト NS-III による数値シミュレーションの今後の展開の可能性展望[28]

次に，NS-III における数値シミュレーションは，今後どのように展開して行くのか，あるいは，どこにどう適用しどう展開して行けばどう有用となり得るか，といった観点について幾つかの考察や適用シナリオを展開してみたい．あくまで私見・想定であり，JAXA の数値シミュレーションの方向性（第 14 章で記述）とは異なる面もあることに注意する．ただし，このような多様な方向性の探索は，今後のシステムの在り方や要求要件の検討に有効と考えられるので，ここに敢えて記載することとする．

乱流や燃焼の **DNS** は，当面は現象理解などの**学術利用**が主であろうと思われる．レイノルズ数の壁[29]があるので，どうしてもその時代にできる最大規模の計算を狙うことになるが，もはや「ただ計算してみました」的に結果を出しただけで済まされる時代ではなく，十分な分析と考察の下に実際にそこから新たな知見，学術的成果を引き出すことがより重要となって来るであろう．

一方，**RANS** などの適用先として考えるべきは，今後ともやはり設計開発などの**工学分野**が第一であろう．CFD の実適用という意味では，前述のように，現実は可能性を示す段階を超えて，如何に実際に使えるか（**精度**），どこまで使えるか（**適用範囲**）という，技術としての『**信頼性**』を厳しく

[28] 2004 年の時点での考察であることに注意する．
[29] 必要な格子点数はレイノルズ数の 9/4 乗に比例する．

問われる時代になっている．今後ともその方向はますます加速して行くであろう．このとき，スパコン利用による大規模シミュレーションの使い道として有効と考えられるのは，データベース構築とその信頼性検証への活用である．従来からの DNS データを用いた乱流モデルの構築や計算結果の相互比較による CFD コードの検証といった活動はみられる．欧米では，NPARC[14]，EUROVAL[15]，ECARP[15]，ERCOFTAC[15]等のプロジェクトが進められている．一方，我が国では，細々とした活動[16]が見られるのみであり，スパコン大国を自負して行こうとするなら世界に伍するデータの蓄積とその有効利用が望まれるところであろう．その際，データマイニングや分析技術が重要になってくる．

次に，計算機能力がここまで向上した今日，**DNS/LES を設計開発の工学実用面に活用**できないかという発想はごく自然であろうし，その可能性を検証するのは NS-III の，もっと言えば JAXA のような研究機関の挑戦すべき良い課題であろう．LES については少なくともそのような動きが既に見え始めていることは前述した．そうはいっても必要な計算量を勘案すると，当面は RANS ではできない場所・条件の課題や実際に課題解決が必要な部分に適用するのが妥当であろう．例えば，航空機において「翼」は最も基本的かつ中心的な構成要素であるが，この流れを CFD で精度良く解析することが（方法を問わず）意外にできていない[17]．遷移位置の特定，失速角・最大揚力の予測，遷音速での衝撃波・境界層干渉を含む特性把握となると，既存の RANS では限界がある．こうした場合への DNS/LES 手法の適用と進展に期待が持たれる．ただし，LES は，非定常シミュレーションであって統計処理が入って来るので，結果の解釈には特に注意する必要があり，**統計処理の標準化・ツール化**は喫緊の課題と思われる．また，他にあまり有効な方法のない**燃焼問題**への DNS/LES の適用と確立は，素現象解明やモデル改良といった基礎面への寄与に加えて，航空エンジンやロケットエンジンの効率，安全性や環境負荷に対する要求が日々厳しくなっている今日，特に実用面での発展に期待が持たれるところである．

CFD の設計開発への利用ということでは，航空機の「**全機体解析**」が今後の展開の一つの起点となる可能性がある．全機体解析は，既に空力性能解析や設計点・非設計点での流れの調査/確認といった利用に供されているが，これがプロダクションベースでできるようになれば，航空機のコンフィギュレーションスタディ（取り付け位置や部位形状の変更）や空力舵面の効き検証，風洞試験の代替と補完（数値風洞），風洞試験模型と実機とのレイノルズ数効果の補間，機体の動安定特性解析等の広範な課題に利用することが可能になるだろう．ただし，ここでいう「全機体」とは，いわゆる（胴体＋主翼＋尾翼）のクリーン全機ではなく，エンジンやパイロン，高揚力装置，脚などの構成要素も伴っているものであって，ストール時等の非設計点や離着陸時の解析が CFD としての威力を発揮する部分である．全機体 NS 解析は，手法的には RANS が主となるであろうが，剥離を伴う場合や非定常的な挙動を追う場合には，LES や DES の利用も視野に入れる必要がある．また，デジタルモデル作成や格子生成についても現実的な手法を見出す必要があろう．

この方面で大きな可能性と期待を秘めるのは，**多分野統合解析と最適化**である．多分野統合解析の今日的意味は，単に空力弾性のような実践的課題に対応するということだけでなく，CFD の利用や研究開発の地平を広げられること，CFD としてどこまで精度が必要かという究極の課題に一定の答えを与えられることにある．空力ー構造といった 2 分野間の連成に止まらず，制御まで含んだような 3 分野以上の統合解析も今後は必要になってくるであろう．そのとき，流体的には非定常解析への対応，一般的には**分野間の情報連携**が特に重要になる．分野によって解法も違えば特性（応答）時間も違うし，有効な情報の伝達法自体まだ確立されていない．安易に連携させるだけでは済まされない点に留意する必要がある．最適化については，**進化的計算法による多目的最適化**に，大規模 CFD としての先見性と有望性を見出る[18]．計算技術的には，大規模な計算を一発打つのではなく，小規模な計算をたくさん実施する類（**パラメータスタディ**）のものであり，ジョブ管理，データ管理などの違った観点での大規模処理が求められる．

CFD や NS-III の**宇宙方面への広範な適用と展開**は，宇宙 3 機関統合を契機により最も活性化されるべき分野であろう．実態に即した適用分野は種々あるだろうが，敢えて新たな構想的なものを列挙してみると，ロケットエンジンシミュレーション，打ち上げ・回収シミュレーション，宇宙環境シミュレーション，リバースシミュレーションなどが考えられよう．このうち，ロケットエンジンシミュレーションと宇宙環境シミュレーションについては，具体的な進め方についてすでに担当チームでの打ち合わせが始まっている．打ち上げ・回収シミュレーションとは，ロケットの打ち上げから回収にいたる各プロセス（エンジン燃焼，上昇飛行，分離，再突入，回収）における解析を総合的に行い，実地試験が難しい条件下での解析や複合システムの評価を行い，信頼性や安全性の事前検討に寄与しようとするものである．また，リバースシミュレーションとは，問題や事故が発生したときに，原因を特定するとか，事故発生にいたる経緯を突き止めるようなタイプのシミュレーションのことを言っている．構想の域を出ていないが，圧力分布を与えて形状を探る逆問題設計の方法論を適用することはできないだろうか．

NS-III の新たな適用先として次に考察したいのが，**安全，環境，運航**などの分野である．安全性にからんで重要なテーマとして気象関連の課題が挙げられる．例えば，航空機は「ウインドシア」に遭遇すると危険にさらされる．（ウインドシアは，局所的な気象現象であり，積乱雲からの吹き降ろしによるマイクロバースト，台風や前線の通過に伴う低層ウィンドシア，山岳波によるシビアウインドシアに分けられる．）特に，マイクロバーストは，離着陸時に遭遇すると大事故につながる危険性があり，実際，統計データによれば，1964～1984 年に 600 人以上が亡くなっている．「タービュランス

（乱気流）」も，安全に係わる卑近な課題である．富士山山頂付近における BOAC 機の墜落に象徴される山岳波などによる晴天乱気流（CAT）はあまりにも有名であるが，乱気流による小さな事故のニュースは例年しばしば耳にするところである．2001 年以前過去 6 年間のわが国の大型機による人的被害を伴う航空事故の約 60%が乱気流を主因としているというデータもある．ジェットストリームによるもの，積乱雲によるものから自動操縦のはずれ（乱気流ではないが）に至るまで原因は多様である．同時多発テロの直後に起きたニューヨーク近郊でのエアバス墜落事故の原因が，先発航空機のウェークタービュランスに大きく関係している（らしい）こと[19]は，本課題が侮れないものであることを象徴している．最近の大型機の就航やダイアの過密化に伴って本課題の重要性は増している．このような気象現象に遭遇したときの，気流の動き，航空機の振舞いや脱出法，あるいは事前に察知し回避する方法等が具体的な課題となる．この場合，手法的には，まず全機解析が必要になる．さらには，機体だけでなく他の事象との関連がむしろ論点となるので，気象シミュレーションとの連携や空港などの地形条件や気象条件の CFD への取り込み方法が課題となるであろう．また，非常に大規模な非定常シミュレーションを可能にする計算機インフラが必須となってくる．また，課題の性質から，設計・開発との技術基準の違い，実際の安全管理や運航現場との連携に留意する必要がある．一方，環境問題にからんだ騒音や CO_2，NO_x の低減も重要な課題である．ICAO の規制強化の動きもあり重要性は高まっている．

DNS/LES に関連する高次精度計算で最近注目されているのは，**CAA（Computational Aeroacoustics）**と呼ばれる「**空力音響**」の分野である．JAXA でも，ヘリコプタの BVI 騒音の解析が従来より行われている．最近では，DNS により音波を精度良く直接捉え，音の発生と伝播のメカニズムを明らかにしようとするアプローチが行われつつある[20]．計算領域を広くとるため計算機への負荷は高く，流体音の微小な圧力変動（音波）を捉えるための高精度計算スキームや無反射境界条件の開発が不可欠となる．

以上の数値シミュレーションの適用局面としてベースとなるのは，実験や理論解析の支援や補完ということであろうが，より効果を発揮するのは「条件的な事情により実験や理論解析が困難な場合」であり，そうした場面で適用されてこそ数値シミュレーションの真価が発揮されるであろう．ここで，条件的な事情とは，扱う対象が極めて小さいとか大きいとかで実験が高価につくとか，極限的な条件（たとえば宇宙とか原子炉）のため実験ができないとか，非常に多くの回数が必要になるといった場合を想定している．そのあたりの状況を俯瞰して数値シミュレーションの適用シナリオを整理してみると次の 4 種類程度に分類され得るであろう．

第 1 は，一つの現象なり状況なりを，**より精密なレベルまで理解**しようとするものである．これは，ある意味では旧来型のスーパーコンピューティングの方向性でもある．乱流の DNS とかがそれにあたる．ただしここで注意すべきは，たとえば乱流の解像のためには $Re^{9/4}$ に比例するメッシュ数が必要であり，シミュレーションとして意義のある結果を出すための綿密な準備が必要であるということである．

第 2 は，**複合的な複雑な系への適用**である．これは，従来，単一の要素・場面に対して適用していたものを，複数の要素・場面に対して適用するものであり，流体－構造連成といった多分野融合解析や，翼＋胴体といった複合形態への適用を意味する．スパコンの性能評価とのアナロジーで言うと，第 1 がスピードアップ性能，第 2 がスケールアップ性能に相当するものである．

第 3 は，**設計開発への適用**であり，特に実験・試験ではできないほどの数のパラメータ解析や設計空間探査，逆問題解析や最適化が含まれる．この範疇では，解析の規模よりターンアラウンドやデータ処理が重視される．

第 4 は，**リアルタイムのシミュレーション**である．観測データをもとに，気象予測をするとか危険予測をするといったものが含まれる．従来，数時間かかっていた RANS を 1 秒以下で行う必要がり，計算性能の相当の向上が求められる．

最後に，現在の数値シミュレーションや CFD が直面している技術課題に触れておきたい．DNS/LES における最大の課題は，大規模出力データを有効に処理する後処理環境であろう．可視化とともに大規模データの蓄積・評価がスムーズにできれば，膨大な数値データの中に潜む新たな物理・知見を引き出すことも容易になる．RANS 関係では，文中で指摘した通り表面データ生成，複雑形状に対する格子生成が大きな課題である．非定常シミュレーションに対しては，精度良い効率的な時間積分法が求められるだろう．良く使われているルンゲクッタ法は，精度は良いが時間刻みを大きく取れない難点がある．陰的陽的を組み合わせた時間積分法などが提案されている[21]．非定常データの扱いも，ただストレージがあれば良いというだけでなく，必要なデータをすぐ取り出せるとか，アニメーションを容易に作れるとかの機能が求められる．非定常問題や大規模問題に適した乱流モデルというのも重要である．定式化が複雑で現象を追い過ぎるモデルより，絶対に発散しないモデルとか，形状や格子の分布に敏感でないモデルが必要である．航空機特有の遷移や遷音速に強いモデルも求められる．このように大規模 CFD をまともにやろうとすると，計算パワー以外のところで意外と穴が多いというのが実情ではあるまいか．

13.5 おわりに

本章では，利用の概要と現状の初期応用成果，ならびに今後の可能性と将来展望について論じてきた．初期応用成果については，代表的な大規模問題や大きな流れに沿うものを中心に説明し，利用状況が少しずつ変化してきていることを指摘した．最後に，NS-III における数値シミュレーションは，今後どのように展開して行くのか，あるいは展開していったら良いのかについて私見を展開した．なお，NS-III の運用期間中における成果については，各年の利用成果報告を参照さ

れたい．[30]

数値シミュレーションや CFD に関する昨今の論点の一つは，CFD サイドの嗜好でただ大規模/複雑/高精度にするということではなくて，色々な課題やニーズがある中でそれに基づいた適切な展開や戦略を如何に設定するかということである．大切なのは，言うまでもなく，数値シミュレーションや CFD をどこにどう使うかである．現状，市販 CFD コードの流通に代表されるように，CFD 技術の汎用化とツール化はどんどん進みつつあり，計算機の性能向上と相俟って，技術としての CFD の成熟度・完成度は（航空宇宙に限っては特に）日々高まっているといえよう．もはや可能性や先進性だけを示して済まされる時代ではないことに議論の余地はない．一方で，材料，バイオ等の分野の数値シミュレーション技術の発展も著しい．しかしながら，航空宇宙分野の数値シミュレーションが計算機と関連技術の発展を促し，計算機の発展が新たな数値シミュレーションの R&D 展開を生むという好循環をもたらして来たのも事実であって，そのような関係は航空宇宙における数値シミュレーションを魅力あるものにしている大きな要素であると思われる[22].

参考文献

[1] 阿部浩幸, 松尾裕一, 河村洋： Re_tau=1020 の平行平板間乱流の DNS による乱流構造の解析, 第 17 回数値流体力学シンポジウム講演論文集, （2003）, A3-2.

[2] Mizobuchi, Y., Tachibana, S., Shinjo, J., Ogawa, S., and Takeno, T.: A Numerical Analysis of the Structure of a Turbulent Hydrogen Jet Lifted Flame, Proceedings Combustion Institute, **29**（2002）, pp. 2009-2015.

[3] Shinjo, J., Mizobuchi, Y., and Ogawa, S.: LES of Unstable Combustion in a Gas Turbine Combustor, High Performance Computing, LNCS 2858（2003）, pp. 234-244.

[4] Mellen, C. P., Frohlich, J., and Rodi, W.: Lessons from the European LESFOIL Project on LES of Flow around an Airfoil, AIAA Paper 2002-0111（2002）.

[5] Germano, M., Piomelli, U., Moin, P., and Cabot, W. H., :A Dynamic Subgrid-Scale Eddy Viscosity Model, Physics of Fluids A, Vol. 3（1991）, pp.1760-1765.

[6] Spalart, P. R.: Strategies for Turbulence Modeling and Simulations, International Journal of Heat and Fluid Flow **21** （2000）, pp.252-263.

[7] 向井純一，山本一臣，高木亮治 : DES による薄翼失速の非定常数値解析，第 16 回数値流体力学シンポジウム講演論文集（2002），E27-4.

[8] Yamamoto, K., Ochi, A., Shima, E., and Takaki, R.: CFD Sensitivity of Drag Prediction on DLR-F6 Configuration by Structured Method and Unstructured Method, AIAA Paper 2004-398（2004）.

[9]山根敬，山本一臣，榎本俊治，高木亮治，山崎裕行，牧田光正，山本武，岩宮敏幸，中村孝 : CFD コード共通化プロジェクト UPACS の現状，航空宇宙数値シミュレーション技術シンポジウム 2000 論文集，航空宇宙技術研究所特別資料 SP-46（2000），pp. 45-50.

[10] 青山剛史，梁忠模，近藤夏樹，齊藤茂，松尾裕一，末松和代 : ヘリコプタの遷移飛行シミュレーション，日経サイエンス 2004 年 1 月号，（2004）p.108.

[11] 浜辺正昭，児玉秀和，山本一臣，松尾裕一，野崎理 : 多段翼列の非定常大規模シミュレーション，第 17 回数値流体力学シンポジウム講演要旨集（2003），C8-2.

[12] 山根敬，山本一臣，吉田豊明 : タービン翼の流体・熱伝導連成解析，航空宇宙数値シミュレーション技術シンポジウム 2003 論文集，宇宙航空研究開発機構特別資料 JAXA-SP-03-002（2004）, pp.190-195.

[13] Makino, Y., Iwamiya, T., and Zhong, L.: Fuselage Shape Optimization of a Wing-Body Configuration with Nacelles, AIAA Paper 2001-2447（2001）.

[14] Nelson, C. C., and Power, G.D.: CHSSI Project CFD-7: The NPARC Alliance Flow Simulation System, AIAA Paper 2001-0594（2001）.

[15] Vos., J. B., Rizzi, A., Darracq, D., and Hirschel, E. H.: Navier-Stokes solvers in European aircraft design," *Progress in Aerospace Sciences*, **38** （2002）, pp.601-697.

[16] 廣瀬直喜 : 飛行機の空気力学の基礎的課題，ながれ第 22 巻第 1 号（2003），pp.23-28.

[17] Levy, D. W., Zickuhr, T., Vassberg, J., Agrawal, S., Wahls, R. A., Pirzadeh, S., and Hemsch, M. J.: Summary of Data from the first AIAA CFD Drag Prediction Workshop, AIAA Paper 2002-0841（2002）.

[18] 大林茂： CFD 利用の新段階－数値最適化,"日本機械学会誌 Vol.105, No.999, （2002）pp.64-69.

[19] 杉江弘 : 機長の失敗学，講談社，東京（2003）.

[20] 井上督 : ながれから出る音の直接数値シミュレーション, ながれ 第 20 巻 第 3 号（2001），pp. 187-195.

[21] Men'shov, I., Kaneko, M., Nakamura, Y.: A Hybrid Explicit-Implicit High-Resolution Method for Non-Linear Advection Equation, 航空宇宙数値シミュレーション技術シンポジウム論文集 航空宇宙技術研究所特別資料 SP-44（1999），pp. 315-321.

[22] 松尾裕一,「数値シミュレータ III」－システム性能特性/航空宇宙への初期応用成果と今後の展望, 日本航空宇宙学会誌第 52 巻第 606 号，pp.175-185，2004.

[30] http://www.jss.jaxa.jp

第 14 章　JAXA 統合スーパーコンピュータの導入に向けての準備的考察

14.1　はじめに

本章では，前章までの議論や分析を踏まえ，従来からの流れでいえば第 4 世代数値シミュレータ，一方では宇宙 3 機関統合の一つの象徴として位置づけられている「JAXA 統合スーパーコンピュータ」に関して，導入のための要求要件やシステム描像，設計に際しての留意点等を考察してみたい．

14.2　JAXA 統合スーパーコンピュータ

JAXA は 2004 年 10 月に発足したが，スパコンについては，レンタル期間の途中ということもあり，第 1 章で述べたように，調布，角田，相模原の各事業所でそれぞれ継続して運用することとされた．しかし，2005 年 10 月の情報・計算工学センターの発足を機に，運用の効率化とサービスの向上を目途に，まずは運用を統合させ，スパコン運用・利用技術チームで一元的に管理することとされた（2007 年 4 月以降）．

その後，調布システムと角田システムのレンタル終了時期が半年しか違わないこと，第 1 期中期計画の終了に更新時期を合わせる必要があることなどから，関係者・委員会等で議論を進めた結果，運用のさらなる効率化と JAXA 事業への貢献のために，設置場所を一箇所（調布事業所）に集約し，「**JAXA 統合スーパーコンピュータ**」（**JAXA 統合スパコン**）として更新する（2008 年 4 月以降）こととされた．

14.3　第 2 期中期計画における利用の方向性と主な要求要件

前章では，いろいろな数値シミュレーションの可能性について前広に論じたが，ここではもう少し焦点を絞り，今後の JAXA 事業に照らした第 2 期中期計画[1]中におけるスパコン利用の方向性とそこから導出される JAXA 統合スパコンの主な要求要件について述べる．

今後の JAXA 統合スパコンの利用の方向性として，従来からの航空分野に加えて，宇宙分野への積極的展開が挙げられる．その中心となるのは，「**JAXA 長期ビジョン**[2]」に沿った流れである．長期ビジョンによれば，次期中期計画を睨んだ今後 5 年間程度（2008－2013）の事業ロードマップとして，HIIA を中心とする基幹ロケットの信頼性向上や HTV の運用，国産航空機開発支援の加速等を挙げている．こうした目標や開発計画をみたとき，**スパコンによる数値シミュレーションは，不具合対策や課題の事前検討等のプロジェクト等支援を通じて，信頼性向上，開発の短縮化・効率化，経費削減，新規アイディアの創出といった様々な観点で JAXA 第 2 期中期計画事業に十分な貢献を果たすことができるものと期待**され，以下のような幾つかのアプリケーションが想定される．

ロケットおよびロケットエンジン開発は我が国の科学技術基本計画において国家基幹技術として位置付けられており，信頼性の高い宇宙輸送システムの開発は JAXA として極めて重要な事業となっている．今後，数値シミュレーションによって課題の事前検討や不具合対策を実施し，ロケットおよびロケットエンジンの信頼性向上および開発コスト削減に寄与することが期待されている．これを実現するには，**大量のパラメトリック解析**を行う必要があり，少なくとも**従来の 10 倍以上の演算能力が必要**とされる．特に，エンジンプルームに起因する打ち上げ時の音響振動を把握することが重要とされているが，今後エンジンやモータのクラスタ化，新射点など従来よりも音響環境を的確に把握する必要性が出てきており，数値シミュレーションを用いた課題解決が求められている．

惑星探査を強力に進めるためには効率のよい宇宙航行システムが不可欠である．JAXA では「はやぶさ」の小惑星探査にて電気推進器による惑星間航行の実績をあげているが，この発展として電気推進器の更なる性能向上や信頼性の向上が強く求められている．地上実験では再現不可能な宇宙環境そのものや機器の耐久性能評価については数値シミュレーションによる評価が非常に有効であると考えられており，そのためには宇宙機と周辺プラズマ環境の相互作用の数値計算が基本となる．また，電気推進器の技術は，惑星探査のみならず静止衛星の姿勢制御や低高度衛星の軌道変更にも応用が期待されており，電気推進系を持つ宇宙機の開発において，将来的に**プラズマシミュレーション**が果たす役割は大きいと考えられる．また，プラズマコンタクターによるアクティブな衛星電位制御技術，衛星から放出されたガスによる衛星搭載機器への汚染解析，磁気プラズマセイルなどの将来惑星間航行システムの開発などでもプラズマシミュレーションの応用が期待されている．

NEDO の国産旅客機開発プロジェクトについては，事業化判断後に基礎設計・詳細設計・製作・飛行試験へとフェーズ移行していく計画とされている．JAXA ではそのスケジュールに合わせ，脚の低騒音化のための解析など離着陸時の機体**騒音の予測**，機内騒音低減化のための飛行中のエンジンのファン，ジェット騒音源の予測技術の開発や機内への騒音伝播解析などを本格化する必要がある．また，詳細／維持設計での空力解析を実施していく必要があり，これらのためには**従来の 10 倍以上の大量の解析を必要**とする．

静粛超音速研究機計画では，実験機の設計・製作，MDO などのコンピュータ設計技術の高度化，将来航空機概念への技術展開となっている．その中で特に実験機の設計，飛行試験のために大量の解析を行う必要がある．詳細な**ソニックブーム解析**，離着陸時ジェット騒音解析，複合材構造－空力連成などをはじめとした**多分野統合最適化**，エンジン内部（インテーク，ノズルを含む）と機体全体の解析や非定常運動の解析（動微係数，運動解析）等々がある．

このようなロードマップ・将来計画から導き出される JAXA 統合スパコン・システムへの要求要件としての第一は演算処理性能の向上であり，最低でも**従来の 10～20 倍以上の性能**が必要とされる．ロケットエンジンの解析や音響解析

等においては，**ラージエディシミュレーション（LES）**等による**本質的に非定常な解析が必須**となるため，大規模処理能力に加え，三次元かつ時系列の膨大なデータを高速かつ多量に処理する**入出力性能やデータ蓄積管理能力**，あるいは**処理の双方向性**，**ポスト処理性能**などの計算機システムとしての総合処理能力が求められる．また，シミュレーションを設計開発に有効に活用して行くためには，市販アプリケーションとの親和性，オープンソースの受け入れやすさ，ソフトウェアとしての標準化度などの**ユーザから見た使い勝手の良さ**，柔軟かつ機敏なジョブスケジューリングに代表される運**用の柔軟性・信頼性**などの機能が重要となる．特に，多分野統合解析や最適化といった類の設計開発にリンクした処理も必要となるため，パラメトリックスタディや最適化エンジンとの連携を可能とするようなミドルウェアやシステムとしての柔軟性などが要求される．さらには，調布地区，角田地区，相模原地区，つくば地区の 4 つの事業所からのネ**ットワーク利用を踏まえたシームレスな利用環境の構築**，**セキュリティ機能**などの項目も必要である．また，プロジェクト支援に代表されるスパコン活用事業の停滞を極力少なくするため，**前システムからの円滑な移行とフル稼働への迅速な遷移**が必須となる．

14.4 様々な側面・角度からみた JAXA 統合スパコンの計算機システム像

上記要件は，主にアプリケーションや利用者から見たトータルな計算機システムへの要求であるが，無論ほかに考慮すべき視点も多い．そこで以下では少し見方を変えて，JAXA 統合スパコンの計算機システムとしての描像について，様々な角度から検討し考察してみる．

JAXA としては，今後 5 年程度（少なくとも第 2 期中期計画期間中）の運用に耐えられるとともに，きちんと使えるシステムであって，当然ある程度の費用対効果が期待できるものが望ましい．また，ユーザアプリケーションが最大限に重要であり，業務の継続性を担保しつつ，汎用計算機と差別化可能である必要があり，ピーク性能だけとか話題性ありき・計算機ありきでないことが求められる．このようなことを念頭に置き，JAXA の計算機としてどのようなものが相応しいか（具備すべき条件）について，①アプリケーション（利用）サイド，②センター（管理）サイド，③計算機技術，④それ例外，の 4 つの観点から検討してみよう．

14.4.1 アプリケーション（利用側）からみたシステム像

アプリケーション（利用サイド）からみた計算機の描像（イメージ）としては，おおまかには上記 14.3 で述べたものとなるが，もう少し細かくみたときに考慮すべき点として，第一にアプリケーションの特性モデルがある．JAXA アプリケーション（JAXA アプリ）の特性（計算量，メモリアクセス量，通信量）がどのあたり（中心特性，ばらつき）にあって，それに基づいてどのような計算機を選ぶべきか（演算性能重視，メモリ性能重視，通信性能重視）といった話である．

最近の JAXA アプリの傾向として，

1） **エンジニアリング系**
2） **サイエンス系**
3） **ミッションクリティカル系**

の 3 種類に分けることができる．エンジニアリング系は，スループット重視，機能（入出力系，ジョブ周り）重視のものであり，いわゆる Capacity 計算あるいはプロダクションランの類に分類されるものである．一方，サイエンス系は，実効性能や規模が重視されるものであり Capability 計算の類である．また，ミッションクリティカル系は，政治的な理由による対応（迅速性，セキュリティ性等）が求められる類のものであり，アプリの特性というよりジョブ運用に影響を与える．これらのアプリはさらに，

a） **計算系**
b） **メモリアクセス系**
c） **通信系**

に分類することができる．上記の傾向と組み合わせれば JAXA アプリの特性は定性的には**図 14.1** のようになる．なお，横軸，縦軸とも全 CPU 時間に対する（メモリアクセス，通信の）割合を表していることに留意されたい．正確な数字は**表 5.7** や**図 5.8** を参照されたい．このような特性の理由としては，大雑把には，「サイエンス系は FFT といった全体全通信や反応計算などの大量の通信・計算を含む場合がある」のに対して，「エンジニアリング系では形状に対応するための非構造格子等の複雑なデータ構造が災いしてメモリアクセスが多くなる傾向が強いから」，といった感じで説明できる．また，この特性は，今後現れてくるアプリケーションによっても変わって来ることも考えられるが，流体分野としての特徴（基礎方程式，アルゴリズムなど）が大きく変わることはないので，このような構造がまったく違ってくるとは考えにくい．この図で，右上のアプリケーションに合わせた計算機ほど特殊なものとなり，最も TFLOPS 単価の高いものとなるので，どのあたりを設計中心とするかでノードや通信系の考え方が変わって来る．割合的には，近年はエンジニアリング系やミッションクリティカル系のジョブ数が増えて来ている．

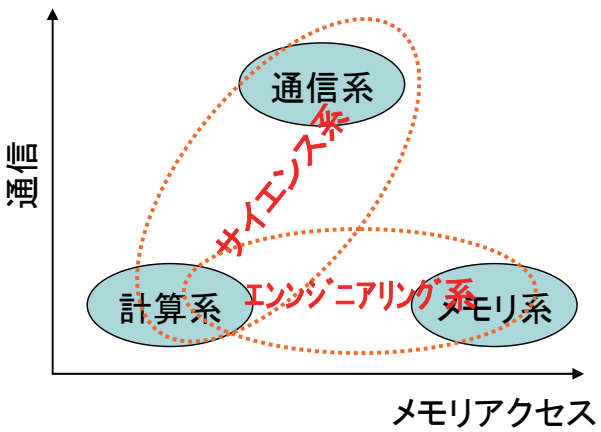


図 14.1 JAXA アプリケーションの特性

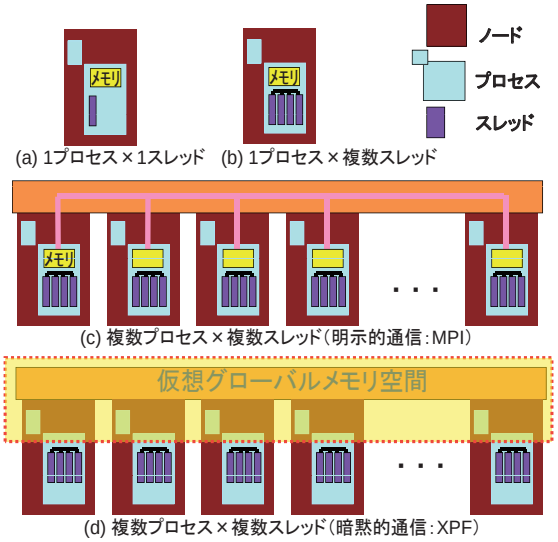


図 14.2　JAXA アプリケーションの並列化モデル

アプリケーションからみたとき，もう一つ考慮すべきは，並列化（ジョブ）モデルである．原則論として，並列化形態の急激な変更は，利用者側に大きな負荷を強いることになり，業務の継続性という点からは好ましくない．（技術動向に合わせて徐々に並列化形態を変更して行かざるを得ないのは仕方のない面もある．）JAXA で現在用いられている並列化形態は，スレッド，プロセスという観点から見れば，

i)　単一プロセス単一スレッド（逐次）
ii)　単一プロセス複数スレッド（自動並列，OpenMP）
iii)　複数プロセス（明示的通信：MPI）単一スレッド
iv)　複数プロセス（暗黙的通信：XPF）単一スレッド
v)　ハイブリッド並列（複数プロセス×複数スレッド）
vi)　パラスタ並列（通信なし）

と分類される．**図 14.2** に幾つかのスタイルの事例を示した．CeNSS におけるプロセス数，スレッド数の実際の利用割合の変遷（2002 年 10 月～2007 年 10 月）は，**図 11.11** 及び**図 11.12** に既に示したところであり，プロセス数の利用中心が 50-150，一方，スレッド数の利用は最近ではせいぜい 2 に留まっているのがわかる．MPI と XPFortran の利用割合の最近の傾向は**図 11.13** のようになっていて，MPI のジョブ数が少しずつ増えてきているが，XPFortran 利用者も少ないわけではない．一方，ハイブリッド並列の性能評価や性能推定の困難さ・面倒さは第 10 章で示した通りであり，スレッド並列があまり広まっていないことを考慮すると，今後ハイブリッド並列を主たる並列化手段として全面的に採用するのはあまり得策でないように思われる．

そこで，CeNSS ユーザに関して最近のプログラミングスタイルの調査を行った．結果を**表 14.3，表 14.4** に示す．これは，2005 年 2 月～2008 年 3 月までの全ジョブのコンパイルリスト，実行リストから拾ってきた数字である．ここで，**表 14.4** において，割合 1=（その欄の人数/プロセス並列形式の人数），割合 2=（その欄の人数/利用者総数）を示す．**表 14.3** からは，並列利用が 8 割以上，MPI 利用が 6 割以上，

表 14.3　プログラミングスタイル調査（2005/2～2008/3）

総利用者	159	1.000
逐次利用	25	0.157
並列利用	134	0.843
XPF 利用	49	0.308
MPI 利用	101	0.635
自動並列利用	80	0.503
OpenMP 利用	37	0.237
プロセス並列のみ利用	49	0.308
スレッド並列のみ利用	8	0.050
ハイブリッド並列利用	77	0.484

表 14.4　ハイブリッド並列の利用状況

プロセス並列	スレッド並列	回数	割合 1	割合 2
XPF	自動	39	0.348	0.245
XPF	OMP	6	0.054	0.038
XPF	自動＋OMP	20	0.179	0.126
XPF	なし	47	0.420	0.296
MPI	自動	30	0.211	0.189
MPI	OMP	5	0.035	0.031
MPI	自動＋OMP	10	0.070	0.063
MPI	なし	97	0.683	0.610
なし	自動	38	0.186	0.239
なし	OMP	12	0.059	0.075
なし	自動＋OMP	10	0.049	0.063
なし	なし	144	0.706	0.906

ハイブリッド利用が 5 割，スレッド並列のみの利用は希少であることがわかる．また，**表 14.4** からは，スレッド並列の利用で多いのは自動並列であることがわかる．このことから，**表 14.3** のハイブリッド並列は，おそらく自動並列利用がそのほとんどと思われる．

一方，並列化形態は別の観点から，

A)　領域分割並列
B)　ループ並列
C)　ジョブ並列

に分けられる．領域分割並列は，分割された領域（ブロック）同士のエッジ部分の情報交換を MPI で通信するスタイルのものである．この場合，並列数が問題に依存することが多く，並列数の変更が一般には困難である．また，領域のサイズがばらつけばロードバランスが悪くなる可能性もある．ループ並列は，自動並列，OpenMP，XPFortran を利用する場合の形態である．並列数が比較的自由に変えられる場合が多い．ジョブ並列は，上記 vi)のパラメータスタディを行う場合に出現する．

上記の JAXA アプリケーションや並列化形態の調査分析から，（ここに掲げていないデータも含めて）主なシステム要件として以下が導出される．

① 従来（NS-III）に比べ，最低でも 10-20 倍の全体性能の向上（100-200TFLOPS の性能）が必要．
② 並列化はプロセス並列が中心で，大規模なスレッド並列は基本的には不要．従って，ノードは SMP 等の大規模ノードである必要はない．プロセス並列の並列数は，ノ

ミナルは数 100，大規模は数 1000 程度と予測．

③ アプリケーションの並列化形態を大きく変えない範囲で，ノードの演算性能，メモリ性能は高い方が良い．

④ 通信は，領域分割並列への対応のために，近接通信は速いほうが良い．

⑤ ベクトル適合ジョブへの何らかの配慮が必要．

14.4.2 管理側からみたシステム像

次に，管理サイドとか運用からみた計算機像について考えてみる．ユーザ個々人からすると，好きなときに好きなだけ使える，というのが望むべき姿かもしれないが，運用管理側としては，そのようなサービスを如何に提供するか，システムの運用管理を如何に効率よく行うか，といった観点からの検討である．

管理サイドの一般的な要件としては，

1) 安定稼動（トラブルが少ない，障害復帰が速い）

2) 運用管理が容易，運用コストが適正

3) 設置性（スペース，電力，冷却）

4) 各ユーザに同一サービス内容を提供（公平性）

5) ソフトウェアの継続性・移植性・汎用性

6) システムの拡張性

などが挙げられる．従来は，1)とか 2）の要素が最も重要であるとして来たが，近年は，3）の要素の重要性やそこから来る制約が高まっている．基本的には，半導体の微細化によりチップの熱密度が高まり（実装密度が高まり），冷却をきちんとやらないといけなくなったこと，省スペースとか省電力のスピードに比べて低コスト化のスピードが速いため，結果としてスペース，電力が増加傾向にあることが原因として挙げられる．**表 14.5** は，2005 年 11 月の Top500[31]リスト中の幾つかのシステムの単位電力あたりの性能を示したものである．おおまかな傾向として，一世代古いシステムは 10 以下，最近のシステムは 100 以下，最近のやや特殊なシステムで 100 以上となっている．JAXA 統合スパコン（JSS）の目標を入れてみた．

表 14.5 主なスパコンシステムの単位電力あたりの性能

System name	Linpack GFLOPS	Power kW	MFLOPS/W	Top500 Rank
BlueGene/L	367,000	2,500	146.8	1
ASC Purple	77,824	7,600	10.24	3
Columbia	60,960	3,400	17.93	4
Earth Simulator	40,960	11,900	3.44	7
MareNostrum	42,144	1,071	39.35	8
Jaruar XT3	24,960	1,331	28.75	10
ASC Q	20,480	10,200	2.01	18
ASC White	12,288	2,040	6.02	47
CeNSS	**5,408**	**600**	**9.01**	**-**
JSS	**100,000 -150,000**	**1,500 - 2,000**	**60-80**	**-**

[31] http://www.top500.org

従来，この種の問題はあまり重要視されていなかったように思われ，どちらかといえば，誰か（設備担当者）がやってくれる，後付けの対応でどうにかなる，といった判断がされてきた感があるが，今後は**計算機運用と設備運用との密接な連携**や，**双方が一体となった施策（自動運転，電力・空調制御など）**が求められるようになって行くだろうと思われる．

14.4.3 技術的側面からみたシステム像

一方，技術的な側面からみたとき，アプリや運用の要求を満足するためには，先進的すぎて実績がなかったり，維持管理（手間，コスト）が大変なものは採用すべきではなく，リース期間中の需要と将来動向を見据えた確実な技術を採用する必要がある．たとえば，ノード形態としては，メモリ性能や電力・コストを勘案すると小規模ノードが有利となる．また，結合ネットワーク（インターコネクト）は，伝統的なクロスバの採用は物量の点でほぼ不可能であり，ファットツリー等の実績のある多段結合網が必須となる．ソフトウェアは，最新のものというよりは，場合によっては一世代前の枯れて安定したものの方が，結果的には良かったりする．独断と偏見で，スパコン関連技術の今後の動向を整理してみれば，次のようになるだろう．

- **CPU 性能は 2010 年頃（あるいはもう少し先）までは Moore の法則により性能向上（5 年で 10 倍），ただしクロック向上ではなくマルチコア化**
- **メモリバンド幅は長期的には低落傾向（CPU と DRAM の性能差は 48%/年で拡大中）にあるので，メモリバンド幅の確保は重要**
- **部品の低コスト化により，システム性能は CPU 性能以上のペースで向上→システムとしての並列度数の増大**
- **省電力，省スペース技術は，今後は良いものが出てくる可能性あり**
- **コモディティ製品は，さらに台頭する**

このような動向を念頭に入れると，今後のシステムとしては，「**何でもできる計算機から，解くべきアプリケーションに相応しい計算機**」へと転換して行かざるを得ないと思われる．IBM の BlueGene 等はそういった流れから生まれてきたものであると考えられ，我々としては，単に「ベクトル計算機」「スカラー計算機」という二分法的な話ではなく，航空宇宙分野のアプリケーションに相応しい計算機像は如何にあるべきかを真剣に模索していく必要がある．

14.4.4 それ以外の側面からみた計算機像

上記以外に考慮すべき論点として，例えば，JAXA 業務への適合性とか，我が国のこの分野（HPC）の方向性に対してどうか，といった項目がある．JAXA 業務適合性とは，セキュリティ対策とか課金情報の取得とかいった話であるが，（詳細は触れないが）意外に重要である．また，我が国のこの分野の方向性に対して，という点では，「次世代スーパーコンピュータプロジェクト」が見ておくべき対象であろう．大規模並列ジョブなどの経験・ノウハウは，共通性があって

活かして行けるものと思われる．

14.5 JAXA 統合スパコンの基本設計における注意点

以上の分析を念頭に，JAXA 統合スパコンが持つべき性能や機能についてさらに進んで検討してみる．

14.5.1 CPU まわり

・ JAXA アプリの中心である流体アプリは演算量が多いので，プロセスあたりの性能はできるだけ高くすべき
→単体 CPU の性能をできるだけ活かすような作り
→アプリの並列化形態を大きく変えない作り
・ 計算ノードに大規模 SMP は性能面で不利 (電力，コスト，メモリ性能)
・ スカラーCPU を採用する場合，マルチコアを如何に有効に利用するかに留意
・ スレッド並列採用時は，オーバーヘッド，フォルスシェアリング，同期処理などに留意

14.5.2 メモリまわり

・ 現状運用をみると，良く言われているユーザ領域として RAM 比（TB/TF） = 1 は必要なく，システム領域（OS, バッファ，スワップ等）を考慮して，RAM 比=0.6～0.8
・ ノードのメモリとして，数 10GB は確保したい
→PC サーバとの差別化
・ メモリバンド幅は，B/F 比=1 程度以上は必要
（参考）B/F 比： Byte/FLOP の略で，計算性能 (GFLOPS) に対するメモリ転送性能（GB/s）の比
・ 後処理や非並列ジョブのために，ある程度の規模の共有メモリ（数 100GB）ノードが別にあると便利

14.5.3 結合ネットワークまわり

・ ジョブ状況によれば, 全システムを使うジョブはほどんどなく，最大でも 1/3 システム程度で，通常は 1/20～1/4 が多い．従って，速く機能的な通信はこの範囲（全体の 1/4 まで）で行われれば良い．この範囲を超える通信や全対全通信，リダクション等の集合通信については，別途実装を工夫するのが合理的

14.5.4 システム構成

・ 単一の大規模システムは，高性能は出しやすいが，機能面で不足する．また，サブシステムに分割し過ぎると運用の手間や使いにくさが増大する．よって，バランスの良い構成に留意
・ 各拠点には，ローカルサーバがあった方が良い
→ファイルサーバ，データ処理サーバの役割
・ 各拠点間の接続については，superSINET の利用が前提
・ ベクトルジョブへの配慮という意味では, 統合前の各拠点間の性能差程度の小規模ベクトルシステムがあった方が便利かも→ベクトルからスカラーへの完全移行は手間

14.5.5 ソフトウェア，プログラミング

・ ソフトウェアやアプリケーションの寿命はハードウェアやシステムより長いため，システムが変わった場合の継続性に配慮する必要あり
・ システムは，マルチコア/CPU，マルチ CPU/ノード，マルチノード/システム，マルチシステム/サイトというような階層構造になる可能性があるが，並列プログラミングの観点からは階層構造はできるだけ少ない方が良い

14.5.6 その他

・ 大規模システムを考える上で, かなり重要なものの一つがファイルシステムである．ファイルシステムが持つべき特性として，単発 I/O の速度性能，複数 I/O の速度性能（スループット），十分な容量，耐久性，障害に対する堅牢性と復帰の早さ，などがある．このうち容量については，ディスクだけというのはコスト的に無理なので, テープとの組み合わせが有利．また，ディスクとテープについては，HSM として運用するのが望ましい
・ ジョブスケジューラは, システムの効率運用の観点からは重要．いろいろな状況に対応するためには，この部分は自前での開発が望ましい

14.5.7 構成検討

以上述べた性能や機能を盛り込んだシステムの構成イメージを（CeNSS と同様に）作ってみると，**図 14.6** のようになるであろう．

14.6 おわりに

本章では，JAXA 統合スーパーコンピュータに関して，導入のための要求要件やシステム描像，設計に際しての留意点等を考察した．なお，これらの結果は，個人的な検討の産物であり，JAXA 統合スパコンの調達方法や最終スペックは，検討委員会や調達委員会等での正式な議論・手続きを経て決められるものである．

参考文献

[1] 独立行政法人宇宙航空研究開発機構の中期目標を達成するための計画，宇宙航空研究開発機構，2008年4月．
[2] JAXA 長期ビジョン-JAXA 2025-, 宇宙航空研究開発機構, 2005 年 3 月．

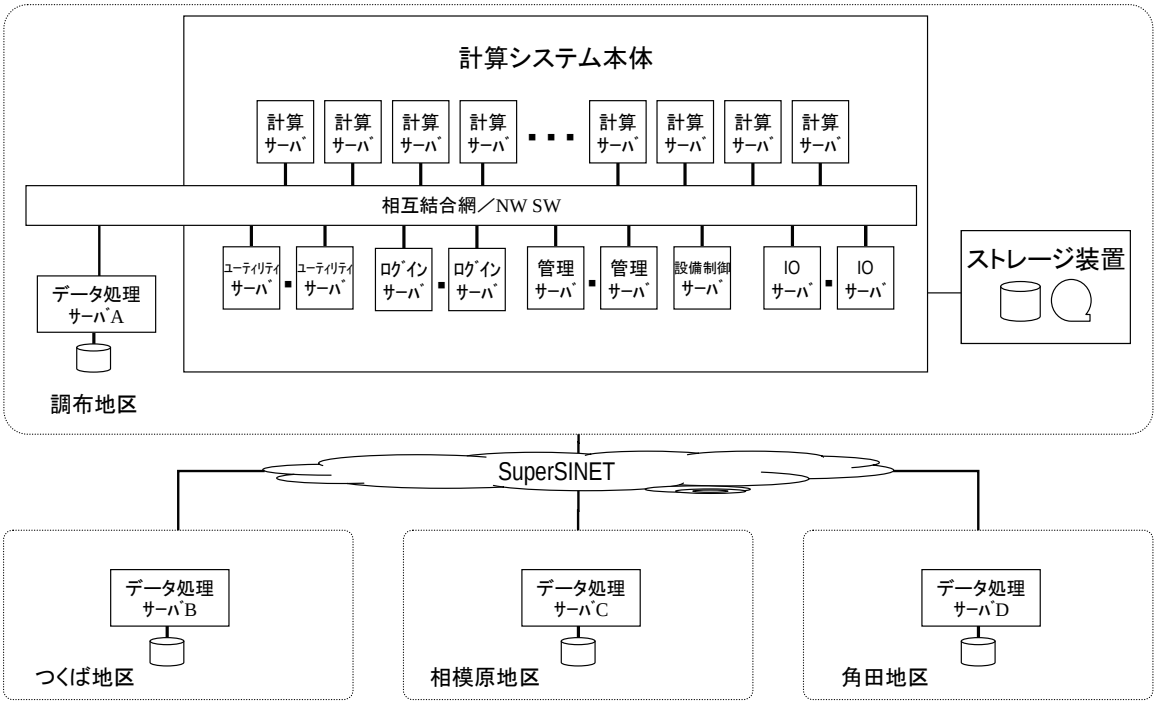


図 14.6 JAXA 統合スパコンの構成イメージ

第 15 章　次世代スーパーコンピューティングへの課題と展望，あとがき

15.1　次世代スーパーコンピューティングへ向けての技術課題

本章では，次の世代，さらに次々世代に向けたスーパーコンピューティングに関する技術課題のうち，JAXA として注視すべきものについて考えてみたい．

前章で述べた JAXA 統合スパコンにまつわる要求要件の背景や長期ビジョンに謳われている宇宙航空開発の今後の発展の方向性，さらには第 1，2 章で述べた宇宙航空分野における技術開発とスーパーコンピューティングのここ 20 年来の良好な関係，等の論拠から外挿・推察すれば，JAXA におけるスーパーコンピューティングに対する性能向上要求は，JAXA 統合スパコン以降も当分の間は継続するものと予想される．従って，当面は 5 年で 10-20 倍程度（ムーアの法則の 2 倍程度）の性能向上が必要（JAXA として必要な性能を確保）という前提で以下考えるものとする．

次世代への課題としてまず挙げられるのは，**設置性**（**電力，スペース，冷却**）の問題である．我が国の大型設備の場合，国家プロジェクト等の特殊な理由がない限り，1 事業所として，電力では数 MW 程度，スペースでは 1,000m² 程度の利用が上限であろう．無論，JAXA とて例外ではなく，最近では CO_2 の排出量規制も制約条件に加わりつつある．将来にわたってこの範囲・制約下でやりくりせざるを得ないのは我々にとっての大前提である．

比較的最近まで，汎用 CPU の性能向上はクロック（動作周波数）の上昇に大きく依存していた．この場合，性能はクロックに比例するが，消費電力はクロックの 3 乗に比例し増加する．最近のスパコンはいずれにせよ並列計算機の形態を採り，非常に多くの CPU を使うので，クロックアップは消費電力の増大につながる．**図 15.1** は，CPU チップのトランジスタ数，クロック，電力等を年代順にプロットしたものである[1]．いわゆるムーアの法則により，半導体の集積度[32]は 18 カ月で 2 倍，5 年で 10 倍の速度で増加する．一方で，電力増加を抑えるためにクロックの増加は頭打ちになっていることを示している．結果として進んでいるのが 1 つのチップ（ダイ）上に複数のプロセッサコアを搭載する「**マルチコア**」の流れである[2]．マルチコアをどう使うかについては，方式技術の選択に係わる重要な問題である．現状でも既に，全て同じアーキテクチャのコアを載せる方法（ホモジニアス・マルチコア，**Homogeneous multicore**，Intel や AMD 等の汎用 MPU）と，違ったアーキテクチャのコアを載せる方法（ヘテロジニアス・マルチコア，**Heterogeneous multicore**，Cell B.E 等）の 2 つの流れが出てきている．2009 年の時点でコア数として，4 コアは実現済み，8 コアは実現

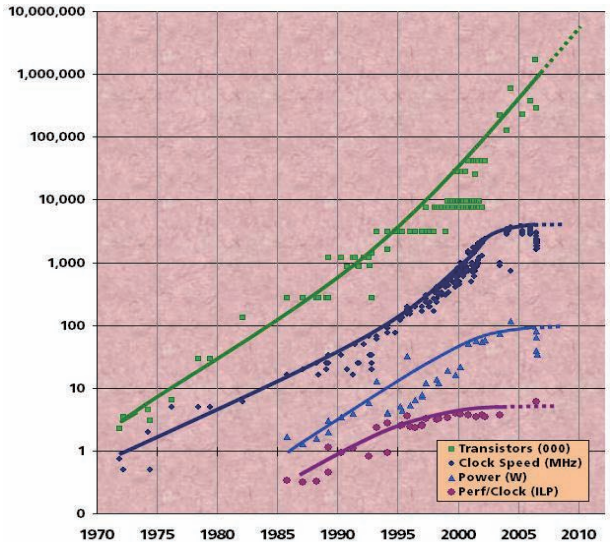


図 15.1　要素技術の性能向上

が見え，16 コアやもっと数の多いメニーコアと呼ばれるチップは数年後の実現が予想されている．

一方，冷却については，全体の電力消費が変わらなくても，半導体の微細化とともにリーク電流により発生する熱の密度（局所性）が高くなっていることに注意する必要がある．このリーク電流は，トランジスタの接合部温度に比例するので，接合部を冷やすほど電力消費も減る．結果として，CPU 等の熱い部分への局所的な「**水冷**」の採用が始まっている．しかし，我々の NS-II（NWT）の経験からすれば，水冷設備の維持管理は大変に手間がかかる．万一，水が漏れたら大変なことになるので，地震等へのリスク対策も含めると安全管理には神経を使わざるを得ない．ただ，CO_2 削減の話もあり，今後とも電力供給量は増やせない状況において，JAXA においても水冷の採用は将来的には不可避と思われる．CPU のように熱密度が高い部分だけ局所的に水冷にしてラジエータで除熱し，残りの部分は基本的に空冷というような何らかの形での**空冷/水冷ハイブリッド**という方法が現実的と思われる．

方式技術は，スーパーコンピューティングにおいて鍵となる技術の一つである．CPU をどう繋げてノードを構成し，ノードをどう繋げてシステムを構成するか，というような観点である．まず，JAXA ほどの規模のシステムとなると，システムとして他に参考とすべき事例は少ないので，NWT から NS-III，JAXA 統合スパコンへと性能デモンストレーターからプロダクションシステムへと発展して来た（今後もその方向で進化するであろう）JAXA スパコンとしては，（後述の信頼性の意味でも）**先端的すぎる方式，実績のない技術の採用は避けるべき**であろう．**業務やソフトウェア資産の継続性**が重要であり，過度のリスク回避のためにも**利用技術・運用技術の継続性**が求められる．その論点を含め，CPU のマルチコア化が進んだときの方式を考えてみると，マルチコア CPU を相互接続する従来型の大規模 SMP は，構成の複雑さの点でも電力消費の点でも不利なことが多い．（運用の困難

[32] ITRS2005 のロードマップ(http://www.itrs.net)によれば，微細化技術のトレンドは，2004 年-90nm，2007 年-65nm，2010 年-45nm，2013 年-32nm，2016 年-22nm となっている．

さは第 11 章で述べた通り.）SMP 以外（たとえば NUMA）の構成でも事情は同じであろう．従って，CPU を多数搭載した大きなサイズのノード（Fat ノード）を相互接続する形態より，少数 CPU のノード（Thin ノード）を多数接続する**超並列形態**の方が今後の選択肢としては望ましい姿のように思われる．ただし，システムの形態が JAXA アプリケーションの並列化形態に整合する（極端に並列度を上げる等の必要がない），ノードあたりのメモリ量が性能とバランスする（ノードのメモリ量が性能に比べ少なすぎない），結合ネットワークが複雑かつ高価になり過ぎない等の条件を満たす必要がある．

最近の方式技術として，**専用チップ**あるいは**グラフィックスチップ（Graphical Processing Unit,GPU）**の利用という路線も出てきている．専用チップとは，ヘテロジニアスな CPU のコアの一部に特有の SIMD 回路を搭載する，といった考え方であり，特定分野で GRAPE[3]等の実例も既にある．GPU は，元来はグラフィックス処理のために開発されたものであるが，演算性能やメモリスループット性能が高い（ただし基本は単精度）ので，これを HPC にアクセラレータとして利用しようというものである[4]．現状，アプリケーションや利用・開発環境が絞られ，性能もピーキーであるため，可能性提示に留まっている．単に計算性能や電力消費のことだけを考えたときには，一つの方向性であるかもしれないが，システムバランスや信頼性を考えると，JAXA のようなプロダクションシステムには，安易には採用しにくい方式であると思われる．信頼性が高まる，周辺環境が充実する等の条件が揃ってくれば可能性も出てくるので，動向は見守っていく必要がある．

マルチコア化で重要なことは，CPU チップ面積は変えられない→ピン数も変えられない，という状況において，演算性能に見合ったメモリからの**データ供給能力（=メモリ性能）**を如何に確保するかということである．CPU とメモリの間にチップセットを置く形態から，メモリコントローラを CPU 内に内蔵する形態になりつつあるが，チップ性能の向上（年 55%）に比べて DRAM の性能向上（年 7%）は鈍く，差は開く一方（年 48%）である．JAXA の主たるアプリケーションが CFD を中心とする流体関連という図式は当面は大きく変わらないであろうから，B/F 比で 1 程度のメモリ性能の確保はしばらくは至上命題となる．メモリの 3 次元実装などの新技術も提案されている[5]ようではあるが，HPC 専用のハードウェアを作るベンダーは今後ますます少なくなる可能性も危惧され，物理的な性能の確保が難しくなれば，アルゴリズムの見直しという事態もいずれは迫られるのではないだろうか．

結合ネットワークについては，ノードの性能・コストとのバランスにおいて決まるものではあるが，近年の大規模システムにおいてはノード数は増える傾向にあり，今後もその傾向は続くであろう，従って，採用できるトポロジは限定され，概ね**表 15.2** のようになるだろう．

表 15.2 結合ネットワークのトポロジの例

ノード数	種類	トポロジ
500 以下	動的網	クロスバ
5,000 以下	動的網	多段結合網，ファットツリー
5,000 以上	静的網	メッシュ，トーラス

クロスバやファットツリーにより，どのノード間もフルバンドで等距離のネットワークが構成できるが，ノード数の 2 乗で物量が増え，スイッチが介在するので非常に高価になる．メッシュやトーラスでは，スイッチがない分コストは安くなるが，近くのノードは速く，遠くのノードは遅いというように通信が等距離ではなくなる．また，途中のノードに障害が起きたときの耐障害性に課題が残る．ノード数が増えたときに集合通信の通信量の増加へも配慮が必要であり，システム全体における結合ネットワークのコスト増加に注意が必要である．今後，専用の結合ネットワークを提供するベンダはどんどん少なくなって行くと思われる状況において，**インフィニバンド（Infiniband）**や**ギガイーサ（Gigabit ethernet）**のような汎用のネットワークを用いて，必要なバンド幅（B/F 比=0.1 程度），レイテンシ，耐障害性等の技術条件を如何にして確保するかは困難な課題である．

ソフトウェアやアプリケーションに絡んだ課題として，問題の定式化や並列スケーリングがある．**問題の定式化**とは，今のような前処理（形状定義，格子作成），決定論的なジョブの実行，後処理（可視化，データ分析）という手法でいったいどこまで行けるかということである．たとえば，ある程度以上の問題規模になると前処理，後処理も並列処理にする必要があるし，データ分析の方法論も確立して行く必要がある．特に，エンジニアリングにおける格子生成の問題は，現在のようなせいぜい半自動の（人手に頼った）方法では，いずれボトルネックになる可能性がある．また，複雑な問題が解けるようになったとき，今のような決定論的な境界条件，初期条件の設定の仕方で有効な解が得られるのか，さらに，現実的な計算時間内に収束解が得られるのか，という課題に対しても答えは自明ではない．

一方，**並列化スケーリング**とは，マルチコア化やクラスタ化が進んで極端に並列度が上がった時に，どうやって並列性能を維持向上させるか，という課題である．第 10 章で述べたように，並列性能（スピードアップ性能）はアムダールの法則により決まる．たとえば問題規模一定で 10 万の並列度数で逐次時の 10 万倍の並列性能を出そうと思ったら 99.999%の並列化率を達成しないといけない，99.99%だったら 1 万倍以上にはならないということを理論的には意味している．もちろん，実際の状況は若干違っているが，100 コアの時代になったときには逐次部分が 1%でもあると 50 倍にしか高速化しない．こうした事情も考慮すると，各コアで安易に MPI プロセスを走らせるのはナンセンスである．

ソフトウェアとしては，**システムウェア**あるいは**ミドルウェア**と呼ばれる部分（コンパイラ，通信ライブラリ，スケジューラ，ファイルシステム，管理ツールなど）の維持発展を

どう管理して行くかも重要な課題である．これらは，いわゆるスーパーコンピューティングあるいは並列処理に不可欠のものではあるが，市場が狭いがゆえに意図的に開発側に働きかけて行かないと適切なもの・良いものがなくなってしまう危険がある．特に，スケジューラとファイルシステムについては，プロダクションシステムとして，システムを効率的に使うため，CFD を中心とするアプリケーション[33]をきちんと実行するためのキーでもある．一方，可視化やネットワークなども重要ではあるが，汎用的な外部の技術を適用できる点では救いがある．全てを自立開発して行くのは無理にしても，自立開発して行く部分と外部技術を採用する部分のトレードオフには常に注視して行く必要があると思われる．

JAXA 特有の課題として，**Capability と Capacity のバランス**の問題が挙げられる．第 14 章で最近の傾向を分析したが，Capability Computing とは，システムリソースの大半を使うような種類の計算を指し，Capacity Computing とは，多数の小さな解析を同時実行するような計算を指す．Capability の場合，世界最大規模とか誰もトライしたことがない計算を行う意味でサイエンス系のジョブが多くなるであろうし，一方，Capacity では，設計パラメータを多様に振るエンジニアリング系のジョブが多くなるであろう．JAXA 長期ビジョンなど見る限りでは，当面は両者が実行可能なシステムづくりが必要である．この問題は，システムの設計ポイントをどこに置くかに係ってくるので，システム構成や方式技術には大きな影響を与えるものであることは常に頭に入れておく必要がある．

また，遠隔利用環境を含む**ユーザの利用環境**の問題も看過できない．世界一の成果のみを目指すのであれば，利用環境は犠牲にして性能だけに投資して何がなんでもやるという取り組み方で良いのであろうが，JAXA のようなプロダクションシステムの場合には，利用まで含めたトータルのターンアラウンドが重視されるからである．グリッドコンピューティングやメタコンピューティングという考え方が一時期流行ったが，我々の場合，大規模可視化であるとか統計処理であるとか，センターとローカルあるいはサーバとクライアントの強い連携が必要な場合が多いので，常に実質的な利用環境の構築が求められる．

システムの信頼性も極めて重要な観点であり技術課題である．「**信頼できる（Credible）**」とはどういうことか，から考えて行く必要があるが，技術要素としては，いわゆる RAS と呼ばれる信頼性（Reliability），可用性（Availability），保守性（Serviceability）の他に，耐障害性（Fault torelance），セキュリティ（Security）などが挙げられる．これらの技術要素を念頭に置きつつ JAXA としては，銀行の基幹システムほどの信頼性は必要ないまでも，プロダクションジョブやミッションクリティカルジョブは問題なく走るといった程度の信頼性は求められる．第 14 章で述べたように，こうしたジョブは今後とも増える傾向にあるので，信頼性に対する要求はもっと高くなると思われる．これらの技術要素を総称して「デペンダビリティ，Dependability」と呼ぶこともあり，ディペンダブルなシステム構築が要求される．また，システムの規模がさらに大きくなったときに，障害から如何に回復するか[34]にも気を配る必要がある．

これらの他に，素子技術，実装技術，パッケージング技術，OS，プログラミング言語，開発環境，ストレージなどを含め，課題は枚挙に暇がないが，動向を見極め，事業の継続性に注意しながら，採用・不採用を判断していくというのが妥当な一つの姿かと思われる．

15.2 スーパーコンピューティングの将来展望

Top500 の予測[6]によれば，世界 No.1 のスパコンシステムの Linpack 性能は，2011 年に 10PFLOPS，2015 年には 100PFLOPS, 2019 年には EFLOPS[35]に達するとされている（**図 15.2**）．米国においては，2009 年 6 月の時点で Jaguar(ORNL, 2.33PFLOPS), Roadrunner(LANL, 1.38PFLOPS)と呼ばれる 1PFLOPS 以上のシステムが既に実働し[7], Blue Waters(DARPA, 10PFLOPS, 2011 年[8]), Sequoia(LLNL, 20PFLOPS, 2011 年[9]), Pleiades(NASA Ames, 10PFLOPS, 2012 年[10])といった 10-20PFLOPS 規模のシステムの導入も計画されている．また，EFLOPS に向けての系統的な技術検討[11]は既に始まっている．韓国，中国，インド，サウジアラビアなどのアジアの HPC 新興国においても数 100TFLOPS 級のシステムが導入（計画）されている[12]．而してスーパーコンピューティングの大規模化・高性能化の世界的な流れは，今後ともしばらくは続くように見える．ただし，現在の PFLOPS システムのうち，LANL の Roadrunner と呼ばれるシステム[13]は，アクセラレータを用いた非常に特殊なシステムであり，使い方は必ずしも簡単ではないらしく，このようなシステムであっても我々の目指すべきところかは疑問が残る．いずれにしても，JAXA が「JAXA 統合スパコン」で目指した 100TFLOPS 級のシステムに対して，性能でもコア数でも 10 倍を超えるシステムが既に現実に存在し，100 倍のシステムの導入計画も既にされているということを認識する必要がある．

我が国においては既に，「次世代スーパーコンピュータ開発プロジェクト」が動いており，2012 年 6 月には，10PFLOPS 級のシステムが稼働することになっている[14]．当該プロジェクトにおいて計画されているシステムは，128GFLOPS の 8 コアのチップ 1 個を搭載したノードの 80,000 個弱を 6D トーラスで相互結合した超並列型のシステムである．3D でなく 6D の結合ネットワークや I/O ノードの接続の仕方等などに特徴はあるが，汎用として開発していることもあり，Roadrunner ほどの特殊性はない．コア数では，60 万コア以上になるので，MPI だけの並列化では限界があるのは明らかであり，ノード間はプロセス並列，ノード内はスレッド並列

[33] 非定常問題では，入出力回数が多くデータ量も莫大．

[34] 回復能力をレジリエンシー（Resiliency）と言うこともある．

[35] Exa FLOPS（エクサフロップス）=10^{18}FLOPS

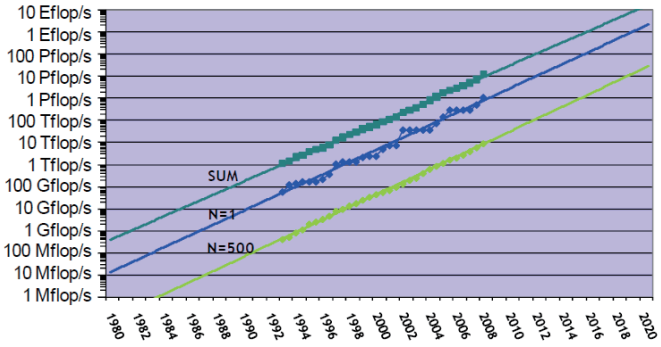
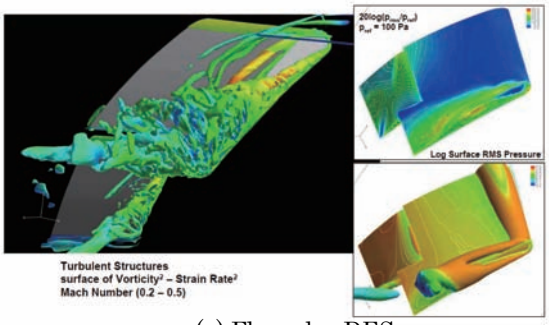


図 15.2 Linpack 性能の変遷と今後の行方

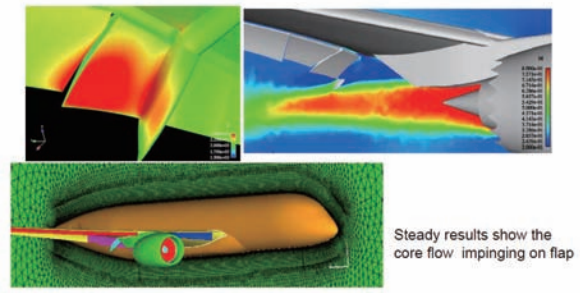
という方法を採らざるを得ないであろう．もっとノードが増えマルチコア化が進んだときのことも考えて，ノード内外で並列化の線を引くという考え方が当面は合理的であろうと思われる．このプロジェクトにより，我が国においても PFLOPS 級のシステムの成立性は実証されるであろう．アプリケーション，利用，運用が次の目標となろうが，2009 年 4 月の時点で 100TFLOPS 以上のシステムが国内でも幾つか動いている現状から察知すれば，PFLOPS の壁は我が国においても，早晩，克服できるものと思われる．

そうした大規模化，高性能化の流れの中で，将来どのようなアプリケーションが想定されているのだろうか．航空宇宙分野については，第 13 章において一部私見を述べたが，Boeing[15]や Airbus[16]は**図 15.3**，**15.4** にあるような多分野融合最適化や実用的な空力課題・開発時間の短縮というような路線を提唱している．また，NASA の R. Biswas[17]は，方法論やアプリケーションでの将来展望を提案している（**図 15.5**）．他の分野の例では，米国 DOE の下で，Scientific Grand Challenge Workshop Series において，キラーアプリケーション（Climate science, High energy physics, Nuclear physics, Fusion energy science, Nuclear energy, Basic energy science, Biology, National security）の検討が進められている[18]．我が国では，次世代スパコン開発プロジェクトの中では，グランドチャレンジ・アプリケーションとして，次世代ナノ統合シミュレーションソフトウェアと次世代生命体統合シミュレーションソフトウェアが，また，戦略 5 分野として，分野 1：予測する生命科学・医療および創薬基盤，分野 2：新物質・エネルギー創成，分野 3：防災・減災に資する地球変動予測，分野 4：次世代ものづくり，分野 5：物質と宇宙の起源と構造，が設定されている[19]．防災や医療のようにこれからのもの，あるいは未着手や未開発のものも含め，スーパーコンピューティングにおけるアプリケーションは，これからもひたすら発展していくような気配である．ただし，前節で述べたように，問題設定やデータ処理は複雑になって行く一方なので，その周辺の技術課題が同時に解決されないと，行け行けドンドンの楽観主義だけではどこかで行き詰まる懸念もある．

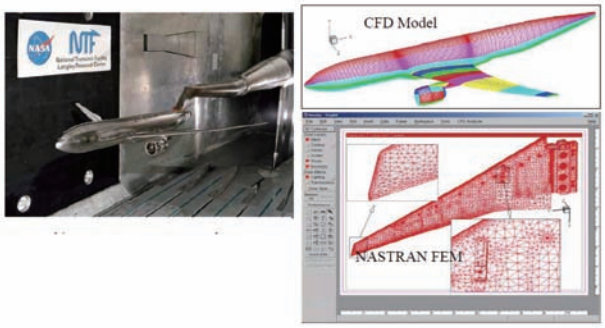
スーパーコンピューティングの拡大基調の中で，並列処理はその中心をなすが，並列度は一層莫大な数に上って行くものと予想される．その場合にどのような並列言語や通信ライブラリを使うのかが問題である．MPI だけでは限界があることは明らかであり，新しい言語についての検討がされている．米国では，DARPA が主導する HPCS プログラムの中で，Chapel（Cray），X10（IBM），Fortress（Sun）といった言語が提案・研究されている[20]．これらは，PGAS（Partitioned Global Address Space）と呼ばれる言語モデルの一種で，米国では CAF(Fortran)，UPC(C)，Titanium(Java)が既に使われている．日本では XcalableMP が提案されている[21]．



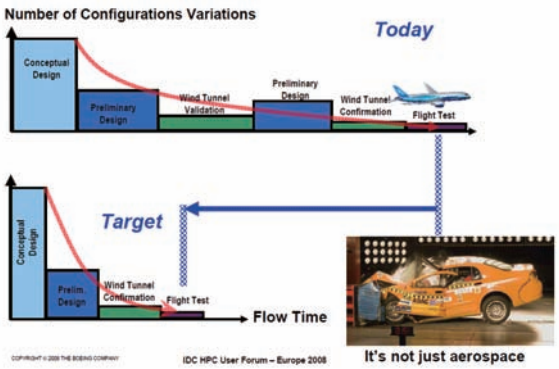
(a) Flap edge DES



(b) Jet impingement loads



(c) Aeroelastic modeling



(d) Shortened product development cucle time

図 15.3 Boeing の HPC 将来展望

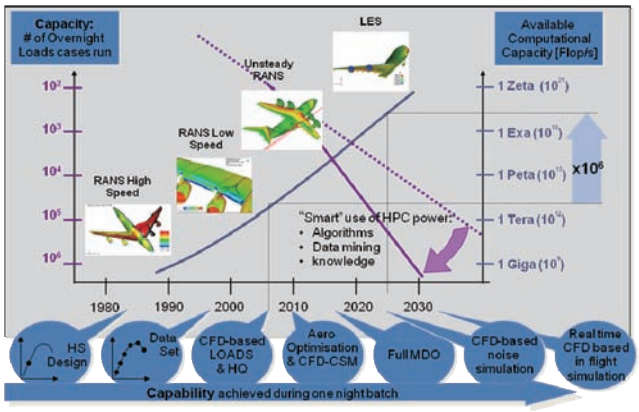
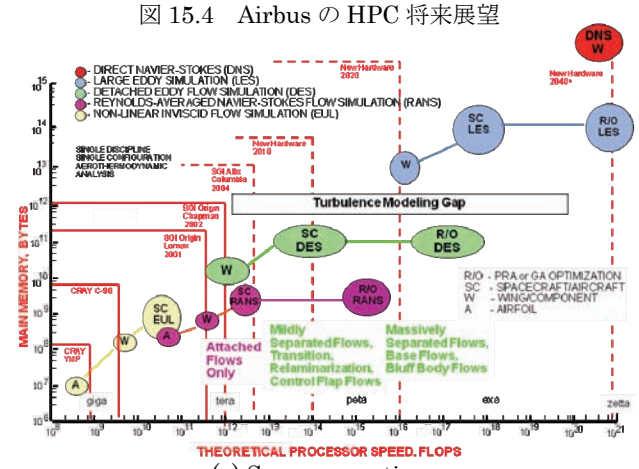
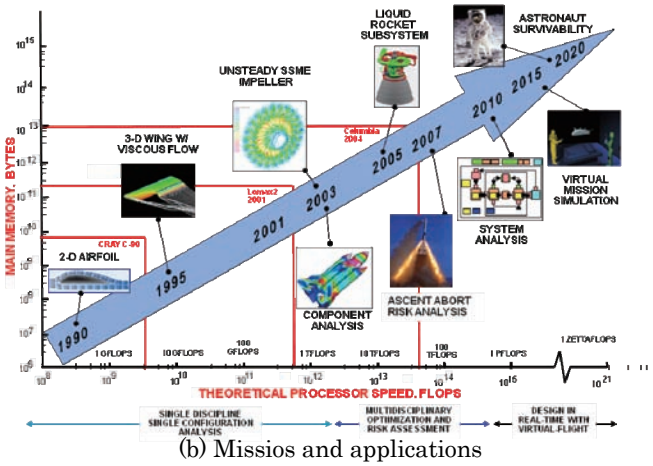


図 15.4　Airbus の HPC 将来展望



(a) Supercomputing



(b) Missios and applications

図 15.5　NASA の HPC 将来展望

15.3　あとがき

本報告では，第 3 世代の数値シミュレータ（NS-III）の導入から運用までの経緯を俯瞰するとともに，性能や特性について分析し，NS-III とはどんな計算機だったのかについて細かく考察した．また，運用データを分析することにより，運用技術に関する暗黙知を形式知として洗い出すとともに，技術的課題を抽出した．さらに，次世代への課題を論ずることにより，航空宇宙分野におけるスーパーコンピューティング及びシステムのあり方を考察した．

スーパーコンピュータシステムの導入/運用を効率良く継続的に行うには，本報告で明らかにしたように，多くの暗黙知，ノウハウに当たる部分が極めて重要であり，またそれらを具現化する関係者が献身的に係わってこそ始めて実現するものである．逆にいえば，そのような係わりの必要のないシステムで良ければわざわざ導入する価値はなく，極論すれば人まで含めたシステム丸ごとのアウトソーシングでも十分であろう．

無論，本報告で記述できたのはスーパーコンピューティングに係る活動・技術の一部分に過ぎず，全ての活動・技術を俯瞰できたわけではない．ただ，事実を正確に記述し，精緻に分析することにより，「設備とか関連技術の持続性（Sustainability）」を高めることを一つの狙いとした．その時代の状況に強く影響を受け，ユーザの要求も時事変わるものなので，基本的には柔軟な対応が求められるものではあるが，ある種の原則・勘所のようなものを提示できたのではないかと思う．本報告が，スーパーコンピュータや設備運用に携わる諸賢の参考に供すれば幸いである．

参考文献

[1] J. Dongarra: Current Trends in High Performance Computing and Challenges for the Future, SS 研 HPC フォーラム 2009.

[2] プロセサはマルチ×マルチへ：日経エレクトロニクス　2007 年 10 月 8 日号，pp.47-62.

[3] http://grape-dr.adm.s.u-tokyo.ac.jp/

[4] 松岡聡：アクセラレータ技術の影と光 - ペタ～エクサの次世代 HPC の中心的な躍進技術へ, 情報処理 Vol. 50, No. 2, pp.95-99, 2009.

[5] 傳田精一：3 次元チップ積層のためのシリコン貫通電極（TSV）の開発動向，表面技術，Vol. 58，p.712, 2007.

[6]http://www.top500.org/lists/2009/11/performance_development

[7] http://www.top500.org/lists/2009/06

[8] http://www.hpcwire.com/features/17883339.html

[9] http://www.hpcwire.com/features/Lawrence-Livermore-Prepares-for-20-Petaflop-Blue-GeneQ-38948594.html

[10] http://www.nas.nasa.gov/Resources/Systems/pleiades.html

[11] http://www.exascale.org/iesp/Main_Page

[12] http://small-servers.com/top500asia

[13] http://www.lanl.gov/roadrunner/

[14] http://www.nsc.riken.jp/index_j.html

[15] D. Ball: Recent Applications of CFD to the Design of Boeing Commercial Transports, HPC User Forum, 2009. (http://www.hpcuserforum.com/presentations/April2009Roanoke/BOEINGApplicationsofCFD.pdf)

[16] P. Messina: IESP and Applications, IESP BoF, SC09, 2009.(http://www.exascale.org/mediawiki/images/2/21/IESP-BOF-SC09-Applications-Messina.pdf)

[17] R. Biswas: NASA’s Science and Engineering Applications in the Future, ZettaFLOPS Forum, Frontires of Extreme Computing, 2005.

[18] http://extremecomputing.labworks.org/index.stm

[19] http://www.mext.go.jp/b_menu/houdou/21/07/1282265.htm

[20] E. Lusk, and K. Yelick: Language for High-Productivity Computing: The DARPA HPCS Language Project, Parallel Processing Letters, Vol. 17, No. 1, pp.89-102, 2007.

[21] http://www.xcalablemp.org/

付　録

付録A NS-III の導入根拠資料

NS-III の導入に際して,その根拠となった平成 11 年(1999 年)当時の主要な報告等の資料を掲載する.

諮問第 25 号「未来を拓く情報科学技術の戦略的な推進方策の在り方について」に対する答申

平成 11 年 6 月 2 日
科学技術会議

第１章 情報及び情報科学技術に関する基本認識

１．物質資源依存型社会から情報資源依存型社会へ

一般に，事物は，物質の要素と情報の要素から成ると考えられる．人類の歴史は，物質及び情報との係わりの歴史であると言うことができる．近代に入って，物質の利用に関する科学技術が急速に進歩し，産業革命等を通して，人類社会は大きな変革を遂げた．他方，情報の利用については，それに比肩するような急速な進展は見られなかった．近代文明は，人類の物質を利用する能力の急速な進歩により，地球規模の資源・環境の制約を顕在化させつつ，かつてない物質的な豊かさを実現した．

近年に至り，情報の生成，処理，伝達，蓄積，保存，利用等に関する科学技術，即ち情報科学技術の急速な発展により，情報をコンピュータにより処理し，ネットワークにより伝達するなど，人類の情報を利用する能力が飛躍的に高まった．このことは，情報が物質とともに，ある面では物質に代わって，文明を形作る主要な要素となりうる状況が到来したことを意味している．

このような情報を巡る新たな情勢に積極的に対応し，情報が有効適切に利用される望ましい社会の構築，特に，これまでの物質を大量生産，大量消費する物質資源依存型社会の，持続可能な発展を可能とする情報資源依存型社会への転換に取り組むことが必要である．ここで，情報資源依存型社会においても，物質の重要性がなくなるのではなく，情報を活用することによって，物質がより効果的・効率的に使われるものであることに留意することが必要である．

情報は，情報一般に共通する性質を有すると同時に，それが由来する事物又は分野によって異なる，固有の性質を有している．現在，多様な分野において膨大な情報が生成されているが，それらを総合的に理解することが困難になっている．また，コンピュータ等の情報を扱う手段は，自然科学のうちの理工系の分野から発展してきたが，自然科学のうちの生命科学系及び人文社会科学において，それらの手段を大いに活用していくことによって，新たな展開が期待される．情報に関するこのような状況において，多分野に亘る情報を体系的に理解し，活用するための情報学という学問分野を確立することが学術の課題となっている．

２．情報科学技術による社会の変革

上述のような課題を内包しつつも，人類の知的・創造的な所産である情報科学技術の発展は，既に，社会のあらゆる分野を急速に変革しつつある．コンピュータとネットワークを中心とする近年の情報科学技術の発展と社会への浸透は，情報を，より多くの人が，より大量に，より高速に，より高度な方法で，より容易に，より低コストで，より長い距離を越えて活用することを可能にしつつある．これは，社会における情報の活用のされ方や情報が果たす役割について，単に量的ではなく，質的な変化を引き起こすものであり，２１世紀には，社会システムから，個人の生活，科学の在り方に至るまで，社会全体が情報科学技術に依存することになる．その際，情報科学技術の発展が社会にもたらす多様な可能性を広く認識し，その実現を目指していくという視点がまず必要であり，同時に，情報化による社会の変革に伴って生ずる新たな課題について十分に認識し，それに適切に対処していくという視点が必要である．

（１）情報科学技術がもたらす多様な可能性

情報科学技術の進展は，まず，個人の選択と行動の自由度を著しく拡大し，生活の豊かさの実現を可能にする．コンピュータと結びついたネットワークの発達による時間と距離を越えたコミュニケーションの拡大は，人と人の結びつきを深め，個人が多様な情報の中から必要なものを入手すると同時に自ら情報を発信することを可能にし，個人が地理的，身体的等の制約を越えて活動することを支援する．情報化の進展は，仕事を効率化し，人がより創造的な仕事に携わることや生活を楽しむ余裕を持つことを可能にする．情報科学技術を活用した新しいサービスの創出や利便性の向上は，個人の生活における様々な面において，それぞれの価値観に応じた多様な選択を可能にする．

また，情報科学技術は，経済社会の在り方を望ましい方向に転換する大きな可能性を有している．すなわち，情報科学技術の活用は，既存の産業の生産性を大きく向上させるとともに，新しい技術の実用化や異分野の融合により新産業を創出することが期待される．これは，より付加価値の高い，物質資源を大量に消費しない，持続可能な方向への経済構造の転換にも繋がることが期待される．

さらに，情報科学技術は，研究開発の手法に大きな変革をもたらし，科学技術の進歩を加速すると期待される．

（２）情報科学技術による社会の変革に伴って生ずる新たな課題

情報科学技術による社会の変革に伴い，まず，情報を使いこなす能力（情報リテラシー）が不可欠な社会になるので，そのような能力を多くの人が身に付けられるように配慮するとともに，情報リテラシーが不十分な人を取り残したりすることがないように配慮することが必要である．

また，ネットワーク上をあらゆる種類の膨大な情報が流通するようになるため，その中から有用な情報を選択できるよ

うにするとともに，有害・不適当な情報の氾濫に対応することや，個人情報の流通に関してプライバシーを保護することが課題となっている．個人の生活，行政や経済の重要な活動がネットワークを介して行われるようになるため，不正な行為等に対するセキュリティの確保や，災害に強く信頼性の高いシステムの構築が必要である．

さらに，金融を初め経済の諸システムが急速に情報化されることにより，これまで予期しなかった現象が見られるようになっており，それに対する十分な理解が必要である．いわゆる 2000 年問題は，社会のあらゆる活動が情報化されることに伴って発生する思いがけない重大な現象の一例と考えられる．

これらの社会に関する課題を含めて情報科学技術を捉えるためには，倫理学，法学等を含む人文社会科学の視点が不可欠となっている．

３．社会の知的・創造的基盤としての情報科学技術

これまで述べてきたように，これからの社会にあっては，研究開発により生み出される知的・創造的な所産である情報科学技術なくしては，高度化する社会のあらゆる活動を支えることは不可能であり，情報科学技術は，社会の知的・創造的な基盤と位置づけられる．

また，情報科学技術が，社会の知的・創造的な基盤として，実際に社会のために役立つためには，情報を活用して２１世紀に向けて社会のどのようなニーズに重点的に対応すべきか，それを実現するための手段として情報科学技術をいかに活用すべきか，そのためには情報科学技術の研究開発はどうなければならないか，という広い視野に立って情報科学技術を推進することが必要である．

情報科学技術の推進に当たっては，以下のような情報科学技術の特徴を十分認識する必要がある．

まず，情報科学技術については，新しい手法等が，長期間をかけて確立され，定着するという現象がある一方で，ソフトウェアに見られるように，基礎的な研究の成果が直ちに実利用に結びつく傾向があり，従来の，基礎から応用，利用へといった，研究開発のプロセスを一方向の段階的なものと見なす認識がそのまま当てはまらない．

また，ソフトウェアやコンテンツに見られるように，成果が個人の独創性に依存する傾向が著しい．言い換えれば，研究開発において個々の人材が大きい要素となる．したがって，優れた人材が能力を発揮できる環境を整えることが重要となる．

さらに，情報科学技術自体についても，それに対するニーズについても，多様性と変化の速さが著しいので，柔軟な対応が必要である．

４．情報科学技術の研究開発を巡る情勢と課題

（１）情報科学技術の研究開発を巡る情勢

海外においては，社会の情報化の重要性の認識が高まり，その基盤としての情報科学技術に対する国家的な取組みが活発化している．例えば，米国においては，雇用の創出，国民が等しくサービスを享受する機会の提供及び産業の国際競争力の向上を目的として，ハードウェア，ソフトウェア，人材等を含む情報基盤の整備を進めようという NII (National Information Infrastructure)構想を進めており，それに基づき，計算，ネットワーク等の広範な分野を対象とする，産学官連携による研究開発計画として HPCC (High Performance Computing and Communications)計画，その後継の CIC (Computing, Information and Communications Research and Development)研究開発計画を実施している．欧州においては，欧州各国の計画に加え，ＥＵが実施している産学官連携の研究開発計画である第 4 次フレームワーク・プログラムの中で，ESPRIT (European Strategic Programme for Research and Development in Information Technologies)等の情報科学技術の研究開発計画を実施している．

また，G7 諸国間の協力として，広帯域ネットワークの相互運用性，電子図書館，ヘルスケアのアプリケーション，電子政府等に関する１１のテーマから成る G7 情報社会パイロット・プロジェクトが実施されている．

我が国においては，「我が国の高度情報通信社会の構築に向けた施策を総合的に推進するとともに，情報通信の高度化に関する国際的な取組みに積極的に協力するため」，平成６年８月に内閣に高度情報通信社会推進本部が設置され，「高度情報通信社会推進に向けた基本方針」（平成 10 年 11 月改定）が決定されるなど，情報科学技術の研究開発を含む，高度情報通信社会の構築に向けた施策について，政府としての取組みが強化された．

（２）情報科学技術の研究開発に関する課題

情報科学技術の研究開発について，民間における活発な活動に加え，国による取組みが活発化しているが，以下に示すように，克服すべき課題が顕在化していると認識される．

まず，諸外国における情報科学技術への取組みが強化される中，我が国は情報科学技術において総体的に遅れを取りつつあるのではないかと懸念される．特に，ソフトウェアの分野における立ち後れが顕著である．また，新しい発想が，速やかに，ソフトウェア等として具体化し，社会で使われるようになるという活力が不足していると考えられる．

また，我が国は，情報科学技術の中で国際的に見て進んでいる分野もあるが，情報科学技術の基礎的・基盤的な領域において，特に米国と比べ，弱さがあると認識される．基礎的・基盤的な情報科学技術の研究開発については，国の研究開発機関が主要な役割を果たすことが期待されるが，国の各機関の情報科学技術部門の規模が小さいなど，体制が十分には整備されていないと考えられる．

また，情報科学技術の研究開発及び活用においては，優れた人材（研究者，研究支援者等）の確保が特に重要であるが，我が国は，人材の層が薄いと認識される．さらに，研究者や研究開発機関の活性化，成果の展開・活用のためには，産学

官の組織間の連携，特に，研究開発と利用との間の連携が重要であるが，現状では十分ではないと考えられる．

５．科学技術情報の流通を巡る情勢と課題

（１）科学技術情報の流通を巡る情勢

上述したように，社会における情報の役割が増大しているが，このことは，自然科学から人文社会科学に亘る広範な科学技術に関する情報を含む科学技術情報について特に顕著である．科学技術が一般社会に深く浸透している現在では，科学技術情報は，社会・経済活動になくてはならないものになってきており，科学技術情報の利用に対する需要はますます増大している．科学技術情報は，人類が直面する種々の問題の解決や創造的な研究開発活動の展開のために必要不可欠なものであると同時に，それ自体が人類共通の知的資産であり，人類全体の重要な資源である．

ネットワーク時代を迎え，科学技術情報の形態及びその流通の在り方が大きく変化している．

まず，科学技術情報の形態は多様化しており，従来では，実験，観測，計算等から得られる各種データ及び事実（以下「ファクトデータ」という），文献，記事等，各種の情報が個別のものとして存在していたが，今や，複数の種類の情報が一体化，複合化した形で存在する割合が，著しく増大している．また，静動画像，音声など新たな形態のものがますます増大している．

また，科学技術情報の流通については，従来，紙の媒体が主体で電子媒体が補完的なものであったが，現在は，電子媒体が主流となり，ネットワークを通じて流通する時代に変わっていく過渡期にあると考えられる．インターネットなどのネットワークとワールドワイドウェブ（WWW）の普及に伴い，情報の流れが一方向から双方向になり，発信源がその数を著しく増加しつつ広く分散し，発信から利用までの時間が短縮され，専門家以外の一般の利用者のニーズが高まるなど，科学技術情報の流通が大きく変貌している．

さらに，ネットワーク時代の到来は，研究開発活動の進め方自体を様変わりさせつつある．例えば，研究者と研究者がネットワークを通じ，電子メールや電子会議によって討論したり，ソフトウェアや実験データを共有するなど，離れた場所に居ながら一つの研究チームとして研究を行うことが可能になりつつある．

このような状況において，科学技術情報を円滑に流通させ，研究開発の情報化を進めるための基盤整備は，重要な公共財を提供するものであるとともに，新産業を創出する源となるものである．

海外の先進国における科学技術情報の流通に係わる取組みの状況を見ると，学協会，出版業界等が中心となって，活発に電子的な情報発信を進めてきているだけでなく，電子出版された雑誌や既存のデータベースをリンクした情報の収集，提供も積極的に行われている．特に，米国では，前述のCIC 研究開発計画の中で革新的アプリケーション開発の一環として地球観測データを初めとした各種データベースの整備及び流通を推進している．ディジタルライブラリ・イニシアティブとして，マルチメディアデータ等のデータベースの整備，ユーザインタフェース技術の開発など産学官の連携による研究開発プログラムが活発に推進されている．また，ドイツでは，ディジタルライブラリ及び電子出版を柱とする情報流通政策を展開している．さらに，他のＥＵ諸国でも遠隔医療，遠隔教育等といったネットワークを活用した情報流通の研究開発や，電子商取引の利用に関する研究開発が進みつつある．

（２）科学技術情報の流通に関する課題

我が国としても，ネットワーク時代に対応した科学技術情報の円滑な流通の実現に向けて取り組むことが急務であるが，そのためには，以下に示す諸課題の解決が必要となる．

まず，我が国の研究開発活動は，その研究成果を世界に向けて広く発信することにより，初めて，世界に認められ，また，アジア地域の中でも我が国が中心的な役割を果たすことができると考えられる．しかし，我が国では，国際的に通用する学協会誌の発行や研究者自身の発信能力に十分ではない面がある．また，所在情報の不足，言語や制度の壁等により，研究コミュニティが発信した情報が海外から十分に利用される状況になっていない．

また，研究開発活動が，データベース等の科学技術情報にますます依存するようになっているにもかかわらず，国内におけるデータベースの整備が十分でなく，一般に公開される情報が少ない．特に，ファクトデータベースについては，研究開発基盤の充実に対する全体的な認識の不足，研究開発機関等におけるデータベース整備に対する業績評価の低さ，研究開発機関と情報流通関係機関との協力関係の弱さなどから，国際的に通用するものが国内には少なく，海外への依存度が高い．

一方，文献情報については，電子媒体の長所を生かすための電子化，電子図書館化等，電子的な流通環境の整備が不十分であるほか，情報を有効に活用する上で重要な抄録，所在等の総合的なデータベースの整備についても，国全体としては必要であるにもかかわらず，十分なものとなっていない．また，情報流通の多様化，国際化に対応するために，情報や関連するソフトウェアの仕様の標準化が必要であるが，部分的な取組みに留まっている．

また，国の研究開発活動についての情報の整備と公開は，国が研究開発に関して国民に対する説明責任（アカウンタビリティ）を果たす上で重要であるが，十分には進んでいない．

科学技術情報の収集と蓄積については，科学技術に関する資料の数の増加と価格の高騰により，国内の情報流通関係機関等における収集が限定的なものとなりつつあり，国として網羅的に情報を蓄積することができない状況になっている．また，科学技術情報のマルチメディア化が進む中，これに対応した電子的な蓄積・保存の状況は十分ではない．

科学技術情報の利用のしやすさという面では，まず，ネットワーク上に多種多様な情報が氾濫，分散する中，必要な，

信頼できる情報を入手することが困難になっている．また，専門家でない利用者が科学技術情報のデータベースを簡便に利用できる環境になっていない．

科学技術情報の流通については，関係機関の連携が重要であるが，ネットワーク時代の課題の解決に向けた連携関係が構築されるに至っていない．科学技術情報の流通に係わる人材については，情報流通関係機関や研究開発機関における情報流通を担う専門家の育成・確保が十分でなく，科学技術の各分野の研究者等についても情報を発信・活用する能力が十分でないと考えられる．

さらに，電子化された科学技術情報の流通に関しては，料金及び決済について，柔軟な料金体系や電子商取引を導入することが課題であり，また，著作権について，簡便な権利処理ができるよう，権利情報の提供を始め，権利処理体制を整備することが課題となっている．

第２章　情報科学技術の戦略的な推進の考え方

21 世紀に向けて情報科学技術を戦略的に推進するに当たっては，情報科学技術の研究開発及び科学技術情報の流通に関して，以下の考え方により進めることが適当である．

１．情報科学技術の研究開発の推進

「社会のニーズを明確に指向した，基礎・基盤の強化」

情報科学技術は，社会の知的・創造的基盤として，21 世紀に向けて社会の将来を左右するものであり，我が国として，情報科学技術における世界のフロントランナーを目指して，情報科学技術の強力な研究開発力を確立するとともに，その成果が社会で広く利用される環境を整備することが必要である．

情報科学技術の研究開発を効果的に推進するためには，以下に述べるように，情報科学技術の特徴に即して行うことが重要である．

まず，情報科学技術は，社会の基盤であり，様々な分野において重要な手段として利用されるものであるので，社会のニーズに対応して研究開発が行われ，その成果が速やかに利用に供されること，また，幅広い分野で共通的に利用できる，波及性のあるものに重点的に取り組むことが重要である．ここで，ニーズに対応することを強調する趣旨は，情報科学技術の研究開発において，方向性あるいは目的意識を明確にすることにより，有用な成果を生み出そうということであるので，ここでいうニーズは，目先の，個別具体的なニーズに限定するものではなく，長期的な視点に立った，社会や科学技術を広く捉えた方向性を主としたものである．ニーズとしてどのようなものを重視すべきかということについては，後述する．

このように，情報科学技術は実際に社会で活用されることが重要であることから，研究開発を進めるに当たっては，単に科学技術として高度なものを取り上げるだけでは十分でなく，技術標準，市場動向等も広く視野に入れて，実際に使われる情報科学技術を生み出すことを目指すという視点が必要である．

また，情報科学技術は，ソフトウェアに見られるように，基礎的な研究の成果が直ちに実利用に結びつく傾向があるので，基礎的な研究から利用に至る取組みが相互に密接に連携できる環境が必要である．

さらに，情報科学技術においては，ソフトウェアに見られるように，一人又は少数の専門家により，革新的な成果が生み出されることが少なくないので，競争的資金の活用や柔軟な研究管理により，そのような可能性を有する個人やグループに，必要な研究資源や研究環境が提供される仕組みが必要である．

情報科学技術の研究開発においては，急速な技術革新や社会のニーズの多様化に柔軟に対応していくことが重要であり，自由な競争の中で民間の能力が十分発揮されることが期待される．特に，社会の具体的なニーズに即した応用的な研究及び開発については，民間が主要な役割を果たすことが期待される．

国の役割は，国全体として重要であるが民間に委ねることにより十分に行われることが期待できない情報科学技術の研究開発を推進することと，人材育成，産学官の連携，ベンチャー・ビジネスの育成等，研究開発のための環境整備を行うことである．特に，ベンチャー・ビジネスの育成については，情報科学技術の分野における民間の活動を活性化するために重要であり，既に，種々の施策が講じられつつあるが，それらを実効あるものとするための一層の取組みが必要である．

（１）基礎的・基盤的な情報科学技術の強化

我が国が情報科学技術における世界のフロントランナーとなるためには，革新的なシーズを自ら生み出す能力を涵養することが不可欠であり，そのためには，長期的観点に立った，基礎的な情報科学技術の強化が必要である．基礎的な情報科学技術としては，新しい機能や格段に高い機能を有する素子等の画期的なハードウェアの探求，新しい概念や手法によりこれまでにない機能を実現する画期的なソフトウェアの探求，認知・学習・言語等の人の知的機能や脳・ゲノム等の生命の情報機能の解明とその知見を活用した画期的な情報処理手法の探求といった，情報科学技術の新たな展開を切り開く可能性を有するものに取り組むことが重要である．

また，情報科学技術は社会全体の基盤として，社会のあらゆる活動のための手段を提供するものであることから，様々な分野で共通的に利用される基盤的な情報科学技術の強化が必要である．基盤的な情報科学技術としては，高度化する多様な用途を可能にするネットワークを構築・運用するための科学技術，言語や音声・画像を含む幅広い情報に高度で多様な処理を行うための科学技術，情報システムをより使いやすくするための人と情報システムのインタフェースに関する科学技術，研究開発等の幅広い分野の重要な課題を解決するための計算科学技術，安心して情報システムを使えるよう

セキュリティを確保するための科学技術といった，社会の情報化を支える柱となるものが重要である．

情報科学技術の研究開発において我が国の民間の活動は活発であるが，基礎的・基盤的な情報科学技術については，市場原理を基本とする民間のみに期待することはできず，国の積極的な取組みが必要である．

基礎的・基盤的な情報科学技術の研究開発は，その成果が円滑に社会の利用に結びつくことが必要であり，そのため，自らニーズを有する利用機関や，研究成果を利用へと展開していく民間と連携しつつ進めることが必要である．

（2）社会のニーズに対応するための情報科学技術の推進

情報科学技術を社会の知的・創造的基盤として実際に社会のために活用するためには，社会のニーズに対応するための情報科学技術の推進が必要である．

上述の基礎的・基盤的な情報科学技術の強化と社会のニーズに対応するための情報科学技術の推進は，密接に関連している．すなわち，基礎的・基盤的な情報科学技術の研究開発の優れた成果が活用されることが，社会のニーズに十分対応するために必要であり，同時に，社会の重要なニーズに対応するために高い目標を掲げた研究開発を行うことを通して，基礎的・基盤的な情報科学技術も高度化される．

社会のニーズへの対応については，情報科学技術により社会の諸分野における重要課題を解決することと，その前提条件として，情報科学技術が使いこなされる高度情報通信社会の環境を整備することの２つに分けて考えることが適当である．以下の社会のニーズは国全体として重要と考えられるものであり，国の情報科学技術の取組みにおいて，それらのニーズへの対応に重点を置くことが必要である．なお，以下，個々のニーズについて，より具体的な内容及び主な研究開発課題の例を挙げているが，それらは，各ニーズについての理解を容易にするために主要なものを例示したものである．

1）社会の重要課題を解決するための情報科学技術の推進

(a) 創造的な科学技術の創出

科学技術創造立国の実現に向け，情報科学技術を活用して，より高度な研究開発手法を実現し，より生産的な研究環境を整備することにより，我が国の研究開発を強化していくことが必要である．その際，研究開発の分野で生まれたネットワークがインターネットに発展したことに見られるように，研究開発のための先端的な取組みの成果が研究開発以外の諸分野に幅広く波及することの重要性について留意することが必要である．

具体的には，まず，あらゆる分野の研究開発にとって科学技術情報は不可欠なものであり，ネットワーク時代に対応した，高度で利用しやすい科学技術情報流通を実現することが必要である．そのための主要な研究開発課題の例として，ネットワーク上に散在するデータベースを統合検索する技術，科学技術情報のマルチメディア化に対応するための画像検索技術・可視化技術，あいまい検索等知識処理，検索用類義語辞書（シソーラス）等がある．

また，分散した研究開発機関等がネットワークを介して共同して研究開発を行える環境の実現や，ライフサイエンス，地球環境等の科学技術の各分野への計算科学技術の活用等，情報科学技術の手法を積極的に活用して，研究環境を高度化していくことが必要である．そのための主要な研究開発課題の例として，ネットワーク上に分散した情報システムを構築・運用する技術，幅広い分野の重要課題を扱う計算科学技術等がある．

(b) 個人の能力の向上・発揮

21 世紀に向けて，個人が能力を十分に向上し，発揮できる社会を構築することは，個人の自己実現のために必要であるとともに，少子・高齢化が進行する中で，我が国が活力ある社会を実現していくために必要である．

具体的なニーズを挙げれば，まず，学校，家庭，職場等において，場所，時間等の制約を超えて，良質の教育を受けることを実現することがある．そのための主要な研究開発課題の例として，学校教育においてコンピュータ等を活用して質の高い授業を行うためのアプリケーション，ネットワークを活用して地理的な制約を超えて教育を受けられるようにするための学習システム等がある．

また，高齢者，障害者等が身体的な制約を超えて能力を発揮し，活動範囲を広げられるように支援するシステムを実現することが必要である．そのための主要な研究開発課題の例として，音声認識等を活用したインタフェースにより情報システムを障害者等にも使いやすくする技術，身体の損なわれた機能を情報科学技術の手段により支援・代行する技術等がある．

(c) 質が高く安心できる国民生活の実現

社会の情報化が国民生活全体のために真に役立っていくためには，個々の国民が質の高い生活を実感しながら，日々安心して生活できる環境を実現することが必要である．

具体的なニーズとしては，まず，質の高い国民生活を実現するために，労働，行政サービス，余暇等の多様な場面において，国民にとって利便性が高く，個々の国民のニーズに応じた多様な情報サービスが提供されることがある．そのための主要な研究開発課題の例として，ネットワークを活用して自宅や職住接近のサテライトオフィス等で仕事をすることを可能とするためのテレワーク関連技術，情報システムの活用により行政事務を効率化するとともに国民に対する情報提供等のサービスを向上させるための行政情報システム等がある．

また，国民が安心できる社会を構築するには，全ての国民が質の高い医療を受けられるようにすること，地震等の災害に強い社会をつくること等が必要となる．そのための主要な研究開発課題の例として，ネットワークを活用して自宅，遠隔地等においても優れた診断等を受けられるようにするための遠隔医療システム，地理情報システムも活用した防災情報のデータベースや災害に強いネットワークを実現する防災情報通信システム等がある．

(d) 活力ある経済の実現

我が国が活力と国際競争力のある経済を維持していくためには，情報科学技術を活用して，第１次産業から第３次産業に至る既存の各産業の生産性を高めると同時に，新しい技術の実用化や異分野の融合により，新産業とそれによる雇用を創出していくことが必要である．その際，情報科学技術の研究開発とその成果の活用を担う情報通信関連産業が高い活力を有していることが重要である．

より具体的には，幅広い経済活動のための商取引，物流等に関するインフラあるいは環境を情報化することにより高度化していくことが必要である．そのための主要な研究開発課題の例として，電子認証，電子決済，電子マネー等を始めとする電子商取引を推進するための諸課題に関連する技術，情報科学技術を活用して道路と車両を一体のシステムとして構築し，渋滞，交通事故等の道路交通問題の解決，物流の効率化等を実現するための高度道路交通システム関連技術等がある．

(e) 経済社会の持続可能な発展

今後の経済社会の持続可能な発展を図るためには，顕在化しつつある地球規模の資源・環境の制約を踏まえて，現在の大量生産・大量消費文明からの転換のための取組みを具体化することが急務となっている．

具体的には，まず，物質資源依存型社会から情報資源依存型社会への転換のため，情報を積極的に活用することにより，製造，物流，個人消費等のあらゆる分野において，物質利用の高効率化，省資源化を進めるとともに，人・物の移動をネットワーク上のコミュニケーションにより部分的に代替したり，実験・試作等の研究開発のプロセスを計算により部分的に代替するなど，物質の利用を情報により代替していくことが必要である．そのための主要な研究開発課題の例として，遠隔地とのやりとりを要する用件をネットワークを利用して処理するための技術，研究開発等の幅広い課題を計算の手法で解決するための科学技術等がある．

また，そのような取組みを進める上で，地球環境問題についての理解を深めることが重要であるため，地球環境の観測や地球温暖化等の地球変動の予測を推進することが必要である．そのための主要な研究開発課題の例として，情報科学技術を活用して地球環境に関する観測，データの伝送等を高度化するための科学技術，計算科学技術の手法を用いて地球変動の予測を行うためのシミュレーション等がある．

2) 高度情報通信社会の環境整備のための情報科学技術推進

(a) 高度な情報インフラが広く利用できる環境

高度情報通信社会においては，高度な情報インフラが社会全体において広く利用できるようにならなければならない．

具体的には，コンピュータとネットワークを中心とする高度な情報インフラが，家庭や教育部門を始めとする公共部門を含め広く社会活動の全体を対象として，相互運用・相互接続性と一定の品質を確保しつつ，妥当な価格で，安定的に利用できるように整備されることが必要である．そのための主要な研究開発課題の例としては，インターネットを始めとする多様なネットワークを高速化，高機能化するとともに相互に円滑に接続し安定的に運用するための技術，計算機のシステムの違い等を意識せずに利用するためのシームレス・コンピューティング関連技術等がある．

(b) 全ての国民が情報インフラを使いこなせる環境

高度な情報インフラが実際に活用されるためには，国民が情報を使いこなす能力，即ち，情報リテラシーの向上が必要である．

具体的には，情報教育の機会が学校や社会で十分に提供されることが必要となる．そのための主要な研究開発課題の例として，初等中等教育等においてコンピュータやネットワークの利用を進めるためのアプリケーション，社会人を含めて幅広い人々が情報システムを活用した学習を体験できる電子図書館等がある．

また，同時に，情報インフラの側がより使いやすいものとなることが必要である．そのための主要な研究開発課題の例として，コンピュータ等をより使いやすくするためのミドルウェアやインタフェースに関する技術，必要な情報を選択し，不要な情報を取り除くための技術等がある．

(c) 信頼できる情報インフラを安心して使える環境

情報インフラが社会全体を支えるものとして機能するためには，高い信頼性を有し，安心して使えるものであることが必要である．

具体的には，まず，情報インフラが，システム自身の故障・誤作動，地震等の災害等に対し，頑健で，信頼性が高いことが必要である．そのための主要な研究開発課題の例としては，高信頼性のシステムを構築するための技術，システムの信頼性を評価するための技術，障害発生時に瞬時に原因を特定しバックアップ等に切り替えるシステムの技術等がある．

また，情報インフラ上を流通する個人情報についてのプライバシーの保護や，情報に対する不正アクセス，コンピュータウィルス等に対するセキュリティの確保が必要である．そのための主要な研究開発課題の例として，暗号技術等のセキュリティ技術，それらの有効性等を評価するための技術等がある．

（3）情報科学技術の活用を進めるための取組み

情報科学技術の研究開発が社会に真に有用なものとなるためには，研究開発の計画策定の段階で，人文社会科学の観点から，当該研究開発の成果が社会に及ぼすであろう影響について検討・評価が行われ，適切な目標と内容を持つものとして研究開発計画が策定されることが必要である．人文社会科学がそのような役割を果たすためには，情報科学技術の社会的諸側面に関する倫理学，法学等の人文社会科学の観点からの研究を活発化し，知見を積み重ねていくことが重要である．

また，情報科学技術が社会で活用されるには，研究開発により優れた成果を生み出すだけでは十分でなく，それを受け入れる側の社会の制度的な条件を整えることが必要である．

特に，情報科学技術の研究開発にインセンティブを与えると同時に，その成果の利用を促進する上で，著作権等の知的財産権の扱いは重要な問題である．既存の諸制度の中には，情報科学技術の発展と利用の現状にそぐわないものもあり，情報科学技術の活用を進めるためには，既存の規制の見直し等，制度面の取組みが重要である．

さらに，情報科学技術の活用を進める上での行政の役割としては，上述の制度面の取組みに加え，行政自らが情報科学技術を積極的に活用し，行政の情報化を進めることが重要であり，政府としては，行政情報化推進基本計画（平成９年１２月改定）に基づき推進しているところである．同計画に基づいて行政の情報化をさらに進めることにより，行政サービスを向上し，国民の利便を高めるとともに，行政情報の提供の推進により国民に対する説明責任を果たしていくことに加え，情報通信需要の大口の利用者として，ニーズを高度化し，高度な情報科学技術の創出と活用を牽引することが期待される．さらに，社会全体が情報化され，ネットワーク化されていく上では，一部門の遅れが他部門に制約を与えることがあるので，行政の情報化が着実に進められることは，民間分門の情報化のためにも不可欠である．

２．科学技術情報の流通の推進

「ネットワーク時代に対応した円滑な流通の実現」

科学技術情報が円滑に流通し，価値が生み出されるためには，利用者が，必要十分な情報を，必要な時に，必要な場所で，容易かつ速やかに，できる限り安い適正な価格で利用できる環境が必要である．その際，個々の機関による個別の取組みに加え，海外も含めた産学官の連携・協力が不可欠である．

また，国全体として必要であるが，民間に委ねることによって十分な取組みが期待できないものについては，国の資金を投入して実施するなど国の関与が必要である．特に，国としては，海外資料の収集を始めとする情報の網羅性の維持，基盤的データベースの整備，情報資源の信頼性を保った案内，科学技術情報の流通に関係する基礎的・基盤的な研究開発等を推進する必要がある．なお，その際，関係する市場形成が十分成熟するまでは，国が何らかの関与をしつつ，育成することが必要である．

さらに，国費を投じて科学技術情報の流通を推進するにあたっては，国民に対する説明責任を果たすため，国がより広く科学技術情報を公開することを念頭において施策を進めることが必要である．

当面の具体的な進め方についての考え方を以下に示す．

（１）世界に向けた科学技術情報の発信

研究成果等に関する科学技術情報については，学協会，研究者等の研究コミュニティが活発に発信できるよう支援する必要がある．特に，研究者がネットワークを介して，研究速報，会議の予稿，研究論文等多様な研究成果を電子的に発信することを支えるための環境を整備することが必要である．また，研究コミュニティが研究論文集等を電子的に編集，出版するための環境整備及びこのためのソフトウェア開発等の支援を強化することが重要である．

また，国内外の研究コミュニティと情報流通関係機関の間，あるいは情報流通関係機関相互の協力体制を強化するとともに，国内で発生する論文，研究報告等について，国際的な流通が円滑に行われるよう，総合的なデータベースとして整備し，英文化した上で海外に発信する必要がある．

さらに，研究開発機関等に蓄積されている研究データ等を，高品質な形で集積し，データベースとして整備，更新し，広く内外への公開を促進するための支援を充実することが必要である．

（２）情報資源の蓄積と流通環境の整備

科学技術情報の収集・蓄積・保存については，国立国会図書館を始め，情報流通関係機関等が連携を図りつつ，内外のより多くの科学技術情報の収集を図るとともに，情報の蓄積・保存に当たっては，電子化を図りつつ継続的な蓄積・保存を推進する必要がある．特に，一般には入手しにくい政府関係報告書，企業の技報等の流通促進のため，国立国会図書館の納本制度等を活用し，その収集を強力に推進する必要がある．

また，広く科学技術振興を図る上で重要な情報基盤である科学技術文献，記事等の情報については，信頼性の高い，網羅性のある，所在，抄録等の情報から成る総合データベースの構築の支援を強化する必要がある．特に，国内での利用を推進するためには，日本語による情報提供を図る必要がある．

一方，基盤的なファクトデータベースについては，我が国の研究開発の推進に寄与するものであるとともに，国際社会における我が国の役割を果たす上で重要なものと考えられる．特に，人類共通の地球的規模の問題の解決にも役立つような基盤的なファクトデータベースについて，整備のための支援を強化する必要がある．この際，その整備に係わる研究者等の業績に対する評価について配慮が必要である．

これらに加え，国費を投じた研究開発活動全般について，研究者，研究課題，研究資源等に関する総合的なディレクトリデータベースの整備を充実する必要がある．

ネットワーク時代の流通環境の整備については，まず，マルチメディアに対応した科学技術情報について，電子化，データベース整備とそれに対応するソフトウェア開発等への支援を強化し，電子図書館化を促進する必要がある．また，流通する科学技術情報が有用なものとなるためには，情報を表現する用語について，一定の共通的な理解があることが前提となるので，ネットワーク時代に即した類義語の整理等の継続的な取組みが重要である．さらに，広く分散する科学技術情報の流通を促進するため，個々の情報について発信時におけるフォーマットの標準化を促進するとともに，情報流通に関するソフトウェアの仕様の標準化を促進する必要がある．

また，ネットワークを介した円滑な情報流通のため，多様

な利用者，複雑な利用形態に対応して，より簡便な著作権処理ができるよう，各種情報に関する権利情報の提供と集中的な権利処理体制の整備が望まれる．さらに，研究者等に加え一般の者も含めた，より多くの利用がなされるよう，柔軟な料金体系にも配慮する必要がある．

（3）科学技術情報の利用しやすい環境の整備

ネットワーク上に広く分散して存在するデータベース等の科学技術情報を，あたかも一つのデータベースのように，統合的，横断的に各データベース中の個別のデータまで自動的に検索できるシステムの構築を図るとともに，そのような環境を構築するのに必要な情報流通関係機関とデータベース提供機関の間の連携を促進することが必要である．

また，電子図書館の実現を念頭におき，一般の利用者にとっても，簡便で利用しやすい案内機能，検索機能等のユーザインタフェースに関する環境を整備する必要がある．その際，国内での利用に対応するため，海外の情報も含めて日本語での情報提供の環境を実現する必要がある．

第３章　情報科学技術の戦略的な推進のための共通的方策

第２章において述べた考え方に沿って情報科学技術を戦略的に推進するに当たって，情報科学技術の各分野における研究開発や科学技術情報の流通について共通的に必要となる方策は以下のとおりである．

１．専門的な人材の育成

科学技術一般について人材は重要であるが，特に，情報科学技術の分野では，ソフトウェアやコンテンツに代表されるように，独創的な個人に負うところが大きく，人材育成は重要な課題である．また，情報科学技術の活用があらゆる分野に広がりつつある現在，それを担う人材の確保が各分野共通の課題となっている．

育成すべき専門的な人材については，情報科学技術の研究開発を担う人材と情報科学技術の活用を担う人材に大別して考えることができる．後者の人材は，科学技術情報の流通に加え，情報科学技術が重要な役割を果たす様々な分野で必要とされている．これらの人材全体について，我が国は層が薄いという認識がある．

活用について強い関心を持ち，よく理解していることが優れた研究開発を行う上で重要であることから分かるように，研究開発と活用については一体のものとして考えていくことが重要である．

このため，人材育成のための取組みを早急に強化することが必要であり，特に，大学における教育の質的かつ量的な充実が重要な課題である．

（1）情報科学技術の研究開発を担う人材

情報科学技術の研究開発を担う人材の育成については，まず，その供給を拡大するため，大学における情報科学技術分野の学生数を増やすことが課題である．特に，高い専門能力を有する人材の確保の観点から，大学院の拡充は緊急の課題である．また，進歩の速い情報科学技術について，より充実した内容の教育を行うために，ソフトウェア工学等の最近重要性を増している分野のカリキュラムの充実，産業界から大学への人材の流動による産業界の動向を反映した教育の充実等の取組みが重要になっている．

情報科学技術の研究開発には，大学等における教育の充実とともに，実際に研究開発を行うことを通しての資質向上が重要である．そのためには，研究開発機関やプロジェクトにおいて若手研究者等が主体的に活躍できる場を充実したり，機関間，分野間の流動，交流を促進することが重要である．社会の基盤として有用な情報科学技術の成果を挙げられる研究者等を育成するためには，研究論文だけでなくソフトウェア等の成果物により評価することが重要である．また，社会の具体的なニーズに対応して実際にものを作ることを通して，独創的な成果を生み出していくような，活力ある人材の輩出が望まれており，ベンチャー・ビジネスの育成等，そのような環境を醸成するための取組みが重要である．

（2）情報科学技術の活用を担う人材

情報科学技術の活用は既に幅広い分野において不可欠になっていることから，情報科学技術以外の分野を専門とする学生についても，それぞれの分野において必要となる情報科学技術の知識を習得させることが必要である．特に，研究者等となる者については，情報発信と情報利用を円滑に行えるよう，科学技術情報の利用に関する教育を充実することが必要である．また，大学において全ての学生を対象として情報に関する教育を行えるよう，施設の整備等を進める必要がある．

また，各分野における情報科学技術の活用の高度化に対処するためには，当該分野における情報科学技術の利用者一般の能力向上だけでは十分でなく，情報科学技術の専門的な能力を有する人材が利用の現場に入っていって，その能力を発揮することが必要である．

まず，科学技術情報の流通と，それと密接に関連する研究開発機関等の情報化のための人材の問題について考える必要がある．我が国の情報科学技術の研究開発や利用を行う機関においては，情報化を担う人材の不足が顕著である．例えば，高度な情報化を進めるために必要なデータベース開発，ソフトウェア開発等についての高度な能力を有した技術者等が不足している．また，研究者が情報システムの管理を担当するといった現状があり，そのような支援業務を行う人材の不足が顕著である．情報化を担う人材を育成し，確保するためには，組織内で十分な処遇を行うことを含めた組織的な対応が必要である．情報流通関係機関においても，情報流通の専門家を各機関独自に育成，確保することが困難な場合があり，機関間で人材交流を行うことにより人材の能力を向上することが望まれる．

さらに，情報科学技術の専門的な能力を有する人材は，ま

すます多くの分野で必要とされている．言うまでもなく，高い成長が期待される情報通信関連産業においては，情報関連の製品の製造，サービスの提供等を担う専門的人材の需要には大きなものがある．さらに，情報通信関連産業以外でも，金融を始めとするあらゆる分野で情報科学技術の専門的な能力が必要になっている．

２．実施体制の強化

（１）情報科学技術の研究開発体制

情報科学技術の研究開発体制の強化には，情報科学技術の高度化と様々な分野のニーズへの情報科学技術の活用という２つの視点があり，この２つの視点に立った取組みは，高度な情報科学技術を活用することによって重要なニーズが達成され，重要なニーズに取り組むことを通して情報科学技術の高度化が図られるという，相互に密接な関係にあり，一体的に進めることが必要である．

基礎的・基盤的な情報科学技術の研究開発において国が主要な役割を果たせるよう，国の情報科学技術の研究開発体制を強化するための方法としては，個々の研究開発組織（研究開発機関またはその内部部門）を整備・充実するとともに，同時に，各組織と国及び国以外の研究開発機関等との連携を強化することが必要である．

個々の組織の整備・充実については，情報科学技術及びそれを活用する幅広い分野の多数の研究開発機関において情報科学技術の研究部門及び支援部門を着実に整備・充実することが必要である．その際，小規模な組織が多数育成されるのみでは不十分であり，ある程度の規模を有し，組織間の連携において主要な役割を果たせる組織の育成が重要である．

特に，大学においては，情報分野の学術研究及び人材育成の強化等のため，各大学の情報関係の学科・専攻等を拡充するとともに，大学共同利用機関として情報分野の中核的な研究機関を設置することが適当である．その機関は，大学間の連携に留まらず，大学以外の機関とも密に連携するものとして体制整備を進めることが必要である．

（２）科学技術情報の流通体制

近年，ネットワークを介した科学技術情報の流通が主流になりつつある中で，ネットワークの活用を前提とした，立法，行政，大学，民間企業等の枠を越えた，情報の流通に係わる新たな形の協力が必要となっている．既に，一部の情報流通関係機関，図書館等において協力しているが，さらに強化する必要がある．

そのためには，産学官の関係機関が，機動的かつ効果的な連携協力を行えるような連絡調整の場を課題別に設け，産学官各々の機関のイニシアティブを相互に尊重する形で継続的に課題の解決を図ることが必要である．なお，具体的な構成としては，国立国会図書館，科学技術振興事業団，学術情報センター，日本特許情報機構，学協会のほか，情報流通サービスに係わる民間企業，団体等が挙げられる．その連絡調整の場で扱う課題としては，例えば，ネットワーク上に分散するデータベースの総合案内や総合検索の実現，文献・記事等全文の情報の蓄積・保存のための継続的な収集等が挙げられる．

また，主な研究開発機関等において，情報流通を始めとする情報化を進めるため，欧米諸国で一般的になっている CIO（Chief Information Officer : 情報統括担当役員）等を置くなど，情報化の中核的機能を置くことが望ましい．

３．研究開発基盤の整備

（１）研究情報ネットワーク

異なる組織や分野の間の連携により情報科学技術の研究開発を活発に進めていく上で，省際研究情報ネットワーク，学術情報ネットワーク等の研究情報ネットワークは不可欠な役割を果たしており，また，研究情報ネットワークにおける先端的な取組みは高度なネットワーク需要を牽引することが期待されるものであるので，産学官連携を円滑に促進することに留意しつつ，研究情報ネットワークの高速大容量化，研究情報ネットワーク相互及び内外の各ネットワークとの接続の推進等，その一層の充実を図ることが必要である．また，これらの研究情報ネットワークが，情報流通の基盤として，科学技術情報のマルチメディア化や情報流通量の著しい増大に対応するためには，基幹回線について，利用状況及びニーズに迅速に対応しつつ，より高速化，広域化を図るとともに，日米間など国際回線を高速化することが必要である．

また，ネットワークの高速大容量化のためのハードウェア，運用技術等，ネットワークの高度化のために必要な技術の研究開発と，高速なネットワークを活用するアプリケーションの研究開発には，高速大容量のテストベッドが必要である．そのため，省際研究情報ネットワーク，学術情報ネットワーク等の既存の研究情報ネットワークとともに，新たに整備される研究開発用ギガビットネットワークが，上述のような幅広い研究開発に活用できるよう運用されることが重要である．

（２）高速計算機

計算科学技術の手法は，研究開発の共通的手法として，今後，一層重要となると考えられる．その研究手段である高速計算機については，テラフロップスからペタフロップスへの更なる高速化を目指し，ハードウェア，基本ソフトウェア，応用ソフトウェア等の研究を行うことが重要である．また，ネットワークを介した分散環境に適したシステムの研究等，計算科学技術を，幅広い分野において，多様な環境の下で活用できるようにするための取組みが重要である．

（３）データベース

データベースは，情報科学技術を含め，あらゆる分野の研究開発に不可欠のものであるので，幅広い利用に供することに留意しつつ，その整備を推進する必要がある．　特に，ファクトデータベースは，研究開発の基盤としての重要性を増しつつあり，その整備を強化するとともに，継続的な取組

みを行うことが重要である．例えば，ライフサイエンスの分野において，近年，急速に発展しているゲノム科学及び脳科学については，実験等によって得られたファクトデータを整理し，それを解析することが，主要な課題の一つになっている．

また，高度な利用ニーズに応えるため，高次に構造化され，個別のデータに比べてより多くの情報を引き出せる基盤的なデータベースの整備に関する研究開発を強化する必要がある．例えば，ゲノム科学の分野においては，ＤＮＡ情報，タンパク質情報，個体情報等のデータベースを整備していくとともに，それら相互に関連したデータを統合したデータベースに関する研究開発を強化する必要がある．

さらに，情報科学技術の分野においても，例えば，自然言語処理のための音声・言語等の大量の文例のデータベースのように，ファクトデータベースの整備の強化と継続的な取組みが重要である．

４．国際的展開

国境を越えたネットワークの発達等による情報科学技術における国際的な交流・協力の拡大と同時に，各国の情報科学技術への取組みの強化に伴う国際競争の活発化が進展する中，我が国として情報科学技術を推進するに当たっては，戦略的な取組みにより国際的な競争力を確保した上で，積極的に国際協力，国際貢献を行うという視点が重要である．例えば，国際電気通信連合（ITU），国際標準化機構（ISO）等の国際的な標準化機関における標準化活動への積極的な参画や我が国発のデファクト・スタンダード実現への取組みは，国際競争と国際協力という視点から重要な課題である．

科学技術情報の流通に係わる国際協力については，従来，我が国は先進国から活発に情報を入手することが中心であったが，今後は，先進国との間では受信と発信がつりあうよう相互交流を行い，発展途上国に対してはこちらから積極的に提供していくという考え方に立って，前述したように，世界に向けた科学技術情報の発信を強化していく必要がある．

また，前述したように，情報科学技術の研究開発においては，優れた人材の確保が基本であるので，研究開発活動の国際的展開が進む中，我が国としては，広く世界に人材を求め，我が国の研究開発を強化・活性化するという考え方が重要である．そのため，有能な人材を引きつけうる優れた研究環境の整備と開かれた研究開発体制の構築が必要である．それと同時に，我が国が国際社会に貢献していくためには，国際的な場で活躍できる人材を我が国から輩出できるよう，そのような人材の育成に努めることが重要である．

さらに，情報科学技術の国際協力の基盤としては，研究情報ネットワークの国際的な整備・接続を進めるとともに，それをテストベッドとして活用し，国際協力によるアプリケーションの開発・実証を行うことが重要である．

５．国の投資

我が国として，今後，社会の知的・創造的基盤として情報科学技術を戦略的に推進し，国際的な競争の中，情報科学技術における世界のフロントランナーとなるためには，第２章に述べた戦略的な推進の考え方及び上述した共通的な方策に沿って，所要の施策を強力に実施していく必要がある．そのためには，国として，所要の投資を行っていくことが必要である．

第４章　情報科学技術の戦略的な推進のための先導的取組「ネットワーク時代の研究システムの構築」

第２章及び第３章において述べた情報科学技術の戦略的な推進の考え方及び共通的な方策に沿って，各界，各機関により精力的な幅広い取組みが行われ，我が国全体として情報科学技術への取組みが変革され，強化されていくことが期待される．他方，情報科学技術に対する取組みの強化の重要性に鑑みれば，各界，各機関の自主的な取組みを待つのみでは十分ではなく，情報科学技術への取組みの変革・強化を加速するための先導的な取組みに産学官連携により着手する必要がある．

１．先導プログラムの創設

先導プログラムは，情報科学技術の特徴に即した，ネットワーク時代に相応しい研究システムの実現に向けて，前述した戦略的な推進の考え方及び共通的な方策を具体化するための革新的な試みを積極的に実施する，いわば，情報科学技術の研究開発を革新するための実験場と位置づけることが適当である．

このプログラムの目指すべき研究システムの方向は，異なる組織や分野の間で，特に情報科学技術の研究開発と利用の間で個人と個人が結びつきを強めながら，個人の能力が最大限に発揮される，人本位のネットワーク型のシステムであり，そのためには，優れた人材を確保し，その能力が発揮される環境を整備すること，情報科学技術の研究者と利用者が一体的に取り組むこと，異なる組織の間で連携を強めた研究体制をネットワークにより実現することが重要な要素になる．

このプログラムを通して得られる経験は，産学官の関係機関で共有することにより，情報科学技術への取組みの変革・強化のために広く活用していくことが重要である．また，その経験は，情報科学技術以外の科学技術の分野における研究開発を改善していく上でも有用なものになると期待される．

先導プログラムは，社会のニーズを明確に指向して基礎的・基盤的な情報科学技術の重点領域を設定し，重点領域毎に研究開発のための諸施策を幅広く活用して研究プロジェクトを設定し，各プロジェクトについて，利用と研究が融合した人本位の研究チームを設置し，それを中心としてネットワークにより異なる組織の間で連携を強めた研究体制によりプロジェクトを実施する枠組みとすることが適当である．

２．先導プログラムの進め方

本プログラムは，基礎的・基盤的な情報科学技術の研究の戦略と計画の策定，研究体制の構築と研究の実施，成果の活用の三段階のプロセスに大別して考えることができる．それぞれの段階のプロセスについて重視すべき点及び本プログラムの推進に当たって留意すべき点は以下の通りである．

（１）社会のニーズを指向して，基礎・基盤の研究を重点化するプロセス

基礎的・基盤的な情報科学技術の研究の戦略と計画の策定の段階については，社会のニーズを指向して，基礎的・基盤的な情報科学技術の研究を重点化するプロセス，即ち，重点領域を設定し，その領域毎に研究プロジェクトを設定するプロセスを設けることが課題である．

重点領域の設定については，まず，社会のニーズの側から，基礎的・基盤的な情報科学技術の研究開発によるブレーク・スルーを必要とする，重要なニーズを汲み上げることが必要である．具体的には，利用者や利用機関からの提言を募る，情報科学技術の研究者が調査を行うといった方法が考えられる．利用者や利用機関からニーズに関する有用な提言が出されるためには，利用者や利用機関自身に情報科学技術の活用についての知見が必要である．そのためには，上述したように，利用機関において，情報科学技術を活用する専門的な能力のある人材を確保し，情報科学技術の活用を組織的に進める体制を構築することが重要となる．他方，情報科学技術の研究者の側も，利用の現場に飛び込んでいき，各分野の潜在的なニーズを把握するとともに，情報科学技術のもたらす可能性について利用者に伝えていくことが必要である．そのような研究者と利用者の相互の触発を促進することが必要である．

利用分野からのニーズを十分に汲み上げた後，それらからの基礎的・基盤的な情報科学技術に対する要請について，優れた研究成果，即ち，先端的で波及性の大きい研究成果が得られるかどうかという観点から評価を行い，重点的に取り組むべき研究領域を設定することが必要である．その評価においては，産学官から専門家の参加を得て，研究と利用に亘る広い視点から行うことが重要である．

重点領域の設定に続き，領域毎に研究プロジェクトを設定するに当たっては，優れた人材の発想と判断を活かして，有効なプロジェクトを設定することが重要である．具体的には，優れた個人に大きな裁量を与えて，プロジェクトを設計させる，あるいは，競争原理により，優れた提案を選定するといった手法を採用することが考えられる．

その際，当該プロジェクトにおいて，社会のニーズに対応するために，どのような具体的成果を実現するのかという点について，明確な目標設定をする必要がある．また，計画策定プロセスにおいて，計画の目標と内容の妥当性に対する人文社会科学の視点が反映されるようにすることが必要である．

（２）研究と利用が融合した，人本位のネットワーク型研究体制

研究体制については，情報科学技術のうちの多様な分野の専門的な能力と情報科学技術を活用する専門的な能力が相互に触発し，融合する体制とすることが必要である．そのような融合を実現するためには，いかに情報科学技術の研究者と利用者を結びつけるかが重要であり，物理的に結集する方法とネットワークにより結ぶ方法を適切に活用する必要がある．具体的な研究体制としては，少数の有能な人材が優れた研究環境の整った場所に集まって集中的にプロジェクトに取り組むための研究チームと，そのチーム以外の物理的に離れている有能な人材をもネットワークにより結んだ研究体制の二重の仕組みを考えることが適当である．まず，中心となる研究チームには，研究者と利用者が参加して，一体的に取り組むことが必要である．また，高度なネットワークによって研究開発機関と利用機関の両方を含む産学官の参加機関を結び，必要に応じ海外の機関とも連携させ，異なる組織の間で結びつきを強めて一つの研究所のように研究を行える研究体制を構築することが必要である．ネットワークを介した研究は，一部の研究分野では既に行われつつあるが，本プロジェクトは情報科学技術の専門家が行うものであり，ネットワークを研究の基盤として使いこなした上で，高度な情報科学技術を適用して，これまでにない進んだネットワーク環境の実現を図っていくことが重要である．そのように取り組むことにより，本プログラムを通して，研究情報ネットワーク等の研究開発基盤の高度化を進め，ひいては，社会の情報化を牽引するという認識が重要である．

研究の実施については，まず，研究に携わる研究者や研究責任者の専門的な知見や発想を活かして柔軟に研究を進めるとともに，事務的な負担を過度にかけないため，最大限の裁量を与えることが重要である．

人材の面では，研究プロジェクトにおいて，優秀な人材が能力を発揮して成果を挙げることと，若手の研究者等が存分に研究を行い，能力を向上させることが必要である．そのため，まず，中心となる研究チームについて，世界的な水準の研究者等が集まって来るような処遇と研究環境を実現するよう，思い切った措置を講ずることが重要である．また，若手の研究者等が，適切な処遇を得て，雑務から解放されて，研究に専念できる環境を作ることが重要である．

（３）本プログラムの成果の幅広い活用を促進するプロセス

本プログラムから期待される成果としては，各研究プロジェクトの研究成果とプログラムの実施を通して得られる経験がある．研究プロジェクトは，上述のように，計画の策定から研究の実施を通して，利用機関からの参加を得て，ニーズと密接に関連付けながら進めることに加え，研究により得られた成果を，積極的に幅広い利用につなげていく必要がある．その際，新しい情報科学技術を実用に供する上で，既存の規制等の諸制度が制約になることがあるので，制度面を担当する関係機関に対して問題提起したり，関係機関の参加を

得て対応方策を検討することが重要である．

研究プロジェクトの評価については，プロジェクト設定のプロセスとしての事前評価に加え，中間及び事後の評価を行う必要があるが，特に，本プログラムにおいて新規に採用した研究開発の手法等について十分に評価し，その結果を公開し，得られた経験を広く産学官で活用し，情報科学技術への取組みの強化に資することが重要である．

（4）本プログラムの推進体制の整備

以上述べた考え方により本プログラムを実施するに当たっては，情報科学技術を巡る情勢の変化を踏まえつつ，情報科学技術の戦略的な推進を図るという広い視点から本プログラムを進められるよう，科学技術会議が適切に関与して，具体的な仕組みの検討を含めた推進体制の整備を行い，情勢の変化に対応した柔軟な運用を図ることにより，本プログラムが所期の成果を挙げ，我が国の情報科学技術への取組みの変革・強化を先導していくことが期待される．

以上

情報科学技術先導プログラムの重点領域の設定について

平成 11 年 7 月 5 日
科学技術会議
情報科学技術委員会

I. 総論

コンピュータやネットワークをはじめとする情報科学技術は，21 世紀において，社会システムから，個人の生活，科学の在り方に至るまで大きな影響を及ぼす重要な分野である．特に，情報科学技術は，様々な分野の活動において共通的に必要とされる基盤的技術といった側面を有しており，こうした分野の活動を促進していく観点からも情報科学技術の研究の推進は極めて重要であり，我が国として戦略的に取り組んでいく必要がある．

この情報科学技術分野について，科学技術会議第 25 号答申（平成 11 年 6 月 2 日）においては，社会のニーズを明確に指向した情報科学技術の基礎・基盤の強化という観点から，情報科学技術への取組みの変革・強化を加速するための先導的な取組み（以下「先導プログラム」という．）の創設が提言されており，この先導プログラムの実施に当たっては，基礎・基盤の情報科学技術の中から重点的に取り組むべき研究領域（重点領域）を設定することが重要とされている．以下に，重点領域設定の考え方及び設定した重点領域について述べる．

１．重点領域設定の考え方

重点領域の設定に当たっては，戦略としての方向性を明確にするため，情報科学技術に対する社会の要請を，いくつかのまとまりのある技術的な目標として捉え，その目標を目指す上での重要な技術項目と，その技術項目について達成すべき課題という３つの要素の組み合わせとして重点領域を設定することとした．なお，重点領域の設定に当たっては，世界的に見て十分に先端的なレベルに到達することを狙いつつ，概ね１０年以内に達成することを念頭に置いた課題を設定している．

２．設定された重点領域

上述の考え方により３つの重点領域を設定した．その目標及び技術項目，並びに，その設定の主旨は以下のとおりである．

（1） 安全で豊かなネットワーク社会の構築

【重点技術項目：フレキシブル・ネットワーク技術，モバイル・コンピューティング技術，セキュア・ネットワーク技術，ネットワーク・サービスプラットフォーム基盤技術，先導的ネットワークアプリケーション，ネットワーク社会の経済的・社会的影響に関する総合的研究】

インターネットの普及に象徴される今日のネットワーク社会においては，電子情報の流通とその活用による新産業の創出，人の創造的活動の拡大等による豊かな生活をもたらす社会基盤としての安心して利用できるネットワークという要請とともに，ネットワーク上におけるハッカー等による犯罪の防止や安全なネットワーク社会の実現ということが強く要請されている．例えば，数十億の端末をサポート可能なフレキシブル・ネットワーク技術や，耐故障，自己修復，動作監視等に優れた高信頼性のセキュア・ネットワーク技術に，情報科学技術としてのブレークスルーを求める要請が大きい．かかる観点から，「安全で豊かなネットワーク社会の構築」を目標とする重点領域を設定することとする．

（2） 人にやさしい情報システムの実現

【重点技術項目：バリアフリー情報システム技術，人間重視ヒューマンインタフェース技術，人にやさしいソフトウェア開発技術】

パーソナルコンピュータやインターネットの急速な普及等様々な情報システムが我々の日常生活に深く根ざしてくる社会においては，子供から高齢者，障害者に至るまで，あらゆる人にとって十分に使いやすい情報システムを実現することが社会の要請となっている．例えば，バリアフリーな情報システムを実現するための様々なデバイス技術や，多言語に拡張可能な日本語対話型コンピュータ技術，人の五感を巧みに活用するマルチモーダルインタフェース技術，ユーザがコンピュータを抵抗なく使いこなせるようなユビキタスコンピューティング技術等に対する社会の要請は大きく，将来における情報家電等への技術の波及も期待されている．かかる観点から「人にやさしい情報システムの実現」を目標とする重点領域を設定することとする．

（3） 先端的計算によるフロンティアの開拓

【重点技術項目：統合シミュレーション技術，可視化技術，並列分散ソフトウェア技術，アーキテクチャ技術】

近年のコンピュータ技術の急速な発達の結果，従来，理論的なアプローチや，実験的なアプローチでは対応が困難であった複雑な諸問題の解決に当たって先端的な計算科学技術が有効な手法として注目されており，その手法を用いたフロンティアの開拓が様々な分野から要請されているところである．例えば，ものの製作に取りかかるまでの全ての作業をコンピュータ上で行うようなバーチャルエンジニアリングの手法は大規模設計の画期的な効率化，製造コストの削減に大きく資するものであり，また，生命科学分野，地球環境分野等２１世紀の国民生活にとって重要な様々な分野で，先端的なシミュレーション手法やデータマインニング手法の確立が求められている．かかる観点から「先端的計算によるフロンティアの開拓」を目標とする重点領域を設定することとする．

３．留意すべき事項

（省略）

II. 各論

１．安全で豊かなネットワーク社会の構築

（省略）

２．人にやさしい情報システムの実現

（省略）

３．先端的計算によるフロンティアの開拓

新たな物質材料・医薬品・生体分子等の検索・設計・創製，長期的・短期的な気象変動予測・異常気象対策，遺伝子解読，航空機・自動車産業・電機産業などにおける自動設計等の計算科学技術や，多量のデータから情報，仮説，知見，課題，法則性等を見つけ出すデータマイニングなども含め先端的計算（ハイエンド・コンピューティング）に係る技術の開発が，こうした各種分野で不可欠になっており，更なる技術開発が強く求められている状況にある．

ハイエンド・コンピューティングは，２１世紀の豊かな国民生活にとってのフロンティアを拓くと同時に，これにより多くの新しい産業が創出され，製造業における製造コストの削減，設計サイクルの短縮，安全性の向上などとともに，金融・証券・流通・サービス等の分野における意思決定への支援などにより，産業競争力の強化の実現に資するという意味で産業界のフロンティアをも拓くものである．

このような先端的計算を実現するに当たっては，例えば，より高精度のシミュレーションを行うためには，モデルも線形から非線形，静的解析から動的解析，非粘性から乱流へと次第に複雑になっており，地震等災害時の対策などにおいてはリアルタイムであることが要求される．また，より大量なデータから複雑な因果関係やモデルを発見するためには，膨大なデータに対して多くの演算を施さなくてはならない．かかる観点から，高速大容量のコンピュータ技術が不可欠な状況にある．

また，先端的計算は膨大な計算能力を要求するが，これを実現するにはプロセッサレベルの高速化，複数のプロセッサやメモリを結合する相互結合網の高速化，これらを故障なく長時間稼働させる高信頼化技術，これらのハードウェアを調和的に機能させる並列分散ソフトウェアなどの開発が重要である．更に，対象とするシステムの複雑な動きをモデル化しコンピュータに乗せる統合シミュレーション技術，結果を目に見えるようにする可視化技術なども開発しなくてはならない．このような先端的ソフトウェアはわが国が遅れている分野の一つであり，戦略的に推進することが必要である．さらに，長期的な展望に立って，次世代のコンピュータの基礎技術を開発しておくことが重要である．

しかし，これらの先端的計算の分野は，世界的に見ても利潤を研究開発投資に回す拡大再生産のメカニズムが成り立ちにくい状況にあり，国の施策として，先端的計算に継続的に取り組み，計算科学技術の発展を促す必要がある．具体的な課題としては，2010 年を目途として，ペタフロップス・ペタバイトクラスの計算機システムの実現を目指すことが重要である．

このような認識の下，「先端的計算によるフロンティアの開拓」という目標を掲げながら，具体的には次のような技術項目に重点的に取り組むことが適当である．

（1） 統合シミュレーション技術

国民生活に役立ち，また産業界にとって有益な実用的な問題について，信頼性の高いシミュレーションを行うためには，単に大規模であるだけでなく，ミクロとメゾとマクロ，気体と液体，流体と構造，流体と熱・化学反応など，異なる物理法則を連成させ，複雑な系の統合的なシミュレーションを行うことが必要になる．また多くの問題では，異なる時間スケール・空間スケールの現象が共存し，シミュレーションが困難になるが，こうした困難を解決し，統合的な解析を行うためには，膨大な計算速度，記憶容量を実現することが重要であると同時に，モデル化手法・アルゴリズム・計算手法・並列化手法などの開発が重要である．このような技術は，物質科学，生命科学，医療，環境，安全，製造業など多くの分野で必要とされ，社会的な重要性は極めて高いものである．

（2） 可視化技術

２１世紀のフロンティアを拓くような先端的計算は膨大な計算結果を生むが，データの羅列だけでは役に立たない．そのデータから意味のある情報を引き出し，有効に活用するためには，先端的な可視化技術の開発が重要である．多くの分野では，最終結果だけではなく計算途中のリアルタイムの可視化が必要とされ，例えば，産業界においては，これにより設計を効率化でき，危険の除去が容易になるものと期待さ

れる．また，データマイニングにおいても，直観的な把握と新たな発想を支援するために，可視化が必要である．この技術の波及効果は，設計の高効率化，原子炉・宇宙開発・遠隔医療等におけるテレプレゼンスなどから，教育・教養・娯楽のためのグラッフィックスまで広がるものである．特に，先端的な３次元コンピュータグラフィックス技術は，物質科学，生命科学，医療，安全，環境，製造業のみならずエンターテインメントなど多くの分野で必要とされており，産業の競争力の強化に資することが期待される．

（３） 並列分散ソフトウェア技術

今後必要とされる先端的計算は非常に大規模となるため，単一のプロセッサでは対応が困難で，並列処理・分散処理によってはじめて実現できるものである．また，並列処理・分散処理はこれを支えるソフトウェアがあってはじめて実用化されるものである．このためには，並列オペレーティング・システム，並列言語，通信ソフトウェア，並列入出力，並列ライブラリ，並列処理環境ツール，並列アルゴリズム等の開発が重要である．これらはいずれも膨大なソフトウェアとなるところから，大規模で信頼性の高いソフトウェアを開発する手法を確立することが必要になる．このような技術により，高速ネットワークに結合された様々なコンピュータ群の形態や環境を気にせずに，並列処理により必要とされる計算パワーを引き出すこと （グローバルコンピューティング）を可能にし，誰でも使える並列コンピュータの実現を通して大きな波及効果を持つものである．

（４） アーキテクチャ技術

高速大容量の処理等の先端的計算を実現するには，その基盤となるコンピュータアーキテクチャ技術の開発が必要であり，先端的な，プロセッサ，相互結合網，データ入出力などの技術を開発し実現することが重要である．プロセッサ技術としては，ペタフロップス級への高速化とともに，新しいアーキテクチャ，特にマルチプロセッサオンチップやメモリ混載型プロセッサなどの技術が重要である．相互結合網技術としては，高速・大容量・低レイテンシであることが必要であり，また，今後は光によるインターコネクション技術，更に，それを制御するソフトウェア技術を開発することが重要である．データ入出力技術としては，得られるデータがテラバイトからペタバイトに及ぶ膨大な量となることから，これを保存し管理することが重要な技術的課題となる．またこれらの先端的計算を安定に実行するためには，高い信頼性を持つシステムである必要がある．これらの技術は，データサーバ，トランザクションサーバ，パーソナルコンピュータ等の要素技術としても応用できるものであり，その技術的波及効果は社会的にも極めて高いものである．

以上

NAL 計算科学ビジョン 21

－CFD 技術の飛躍と 21 世紀における中核的研究拠点の形成を目指して－

平成 11 年（1999 年）10 月
数値シミュレーション技術等検討委員会

1. はじめに

1.1 第 3 のツールとしての計算科学技術

『数値シミュレーション（計算科学技術）は，理論，実験に次ぐ第 3 の強力な研究開発ツール』との認識が最近の高性能計算機の発展とともに進展し，科学技術会議第 21 号答申「先端的基礎科学技術に関する研究開発基本計画について」（平成 6 年 12 月）において「未来を担う先端科学技術研究にブレークスルーを生む基盤分野としてその積極的振興が必要」と唱われ，25 号答申「未来を拓く情報科学技術の戦略的な推進方策の在り方」（平成 11 年 6 月）に基づいて最先端フロンティアの開拓を目指した重点領域が設定され，統合シミュレーション技術，可視化技術，並列分散ソフトウェア技術，アーキテクチャ技術が重点項目として選定されているところである．また，日本学術会議からは，「計算機とネットワークを中心とする情報技術の急速な発達によって人類社会の情報革命はかつてないピッチで進行中であり，健全な高度情報化社会を築くためには計算機科学研究を幅広い観点から推進すべき」との勧告が建議されている（平成９年５月）．

1.2 計算科学技術（特に CFD 技術）を取り巻く状況

航空宇宙に目を転ずれば，航空・宇宙それぞれの分野で，次世代超音速機（SST）や宇宙往還機（HOPE）等の飛行実証を中心とする大規模国家プロジェクトが進行中であり，計算科学技術は，空気力学を始め構造，制御，推進等のあらゆ分野でプロジェクト成就のキーテクノロジーと認識されている．特に，計算空気力学（CFD）技術は，実験機開発における設計支援ツールとして中核的な役割を期待されている．

CFD 技術は，計算科学技術の発展を牽引してきたのであるが，CFD が開発設計に利用されるまで技術が成熟したこと，経済産業構造がより効率性を要求してきていること，及び計算機が急速に発達普及してきたこと等により，CFD 技術を取り巻く状況は大きく変化し，CFD への要求要件，CFD の目的意識，CFD 研究開発の在り方などはここに来て見直しを余儀なくされている．こうした中で，航空宇宙技術研究所（以下「NAL」）は平成 8 年度，外部評価を実施し，CFD 技術については，高いレベルに到達してはいるものの空力部門と計算機部門の協力によるさらなる高度化の要請，研究成果の発信及びソフトの汎用化と公開の必要性，等が指摘されたところである．また，科学技術基本計画では，我が国の科学技術活動のレベルアップ，あるいは国の業務の効率化等の観点から，研究開発経費の重点的配分，柔軟な研究開発体制の整備，研究成果の社会への還元等が急務とされている．さらには，2001 年での独立行政法人への移行を控え，研究目

標の明確化や具体性のある研究計画の策定が急がれている．

1.3 ビジョン策定の必要性

以上の指摘及び状況を踏まえ，NAL における計算科学技術の研究開発の推進に当たっては，適切な目標を据えた具体的な研究計画を定め，柔軟な体制の下，資源の有効利用を含め重点的かつ効率的に事業を展開して行く必要があるとの観点から，また，計算科学技術の振興には，NAL における CFD 研究の着実な推進と拡大が不可欠との信念から，２１世紀初頭の NAL の CFD 研究の方向を示すものとして，本ビジョンを提案するものである．

2. 航空宇宙における CFD 技術の発展経緯

まず，CFD 技術の歴史的な発展経緯を一般的な見地から概観するとともに，CFD 技術を含む情報科学技術分野の動向を展望し，現状理解の共通化を試みる．

2.1 CFD 技術の実用化は 80 年代から

CFD 技術の実用化は 1980 年代に始まったと言って良い．おそらくその端緒となったのは，1980 年前後の Steger の仕事であろう．そこには，一般曲線座標系，衝撃波捕獲法，陰的時間積分法，局所時間刻みなど現在でも使われている CFD 技術コンセプトのほぼ全てが包含されている．いわるゆ CFD 解析コードが世に現れたのもこの時代であり，NASA Ames の ARC2D／ARC3D，Jameson の FLO シリーズ，内部流では Denton コードなどが有名である．中でも Jameson は，自身のコードを売り出して産業界に流通させるという，今日のソフトウェアに関するデファクトスタンダードの姿をいち早く実現させていた点で極めて興味深い．

世界レベルで技術的にみたときの 1980 年代の特徴は，研究の中心がスキームと計算術に特化していたという点であろう．無論そのことは計算機の発達と無縁ではなかったが，計算機の発達以上に解析そのもののスピードアップをもたらした．その結果，上記スキームと計算術の研究成果により，現状では当たり前の技術となったレイノルズ平均ナビエストークス（RANS）解析の端緒が切られ，その後の実用化の出発点となった点で特筆されよう．しかし，研究の中心がスキームと計算術に偏るという体質は今日まで温存され，格子生成，乱流モデル等は世界共通の課題となってしまっている．

2.2 混沌とした 90 年代

1990 年代に入り，計算速度が 10 年間で約 1000 倍という計算機性能の劇的な向上も手伝って，CFD コードを「より複雑で大規模な問題へ応用しよう」とする動きが強まり，マルチブロック法，重合格子法などの開発により解析対象が広がった．一方で，乱流モデルの検討やスキームの高精度化など「技術の信頼性を高める活動」も起こったが，現象の理解や数学的な解明が相対的に進まず，現状でも CFD 技術のボトルネックの一つとなっている．

90 年代のもう一つの特徴は，RANS 解析が定着するとともに CFD 技術が設計フェーズに広く利用されるようになった点であろう．それは，ある意味では技術発展の自然な姿であり，旧来型の研究開発が成熟域に達したことを意味する．とりわけ，CFD 技術の空力設計への応用は，多分野融合解析や最適設計などといった CFD をツールとして自在に使いこなす新分野を生みつつある．また，乱流・燃焼などの物理現象そのものの解明に CFD を使うという方法論が確立されたのもこの時代の特徴である．

2.3 情報科学技術の現状と展望

近年，CFD 技術のインフラを支えるコンピュータとネットワークを中心とする情報科学技術の発展と社会への浸透は目覚ましく，社会・経済・生活の全般に大きな変革を起こしつつある．このような流れに対応して我が国の科学技術会議は，「未来を拓く情報科学技術の戦略的な推進方策の在り方」を答申（第 25 号答申）し，社会ニーズを明確に指向した情報科学技術の基礎・基盤の強化という観点から先導的取り組みへの必要性を提言したところである．これを受けて，先に述べたように，先端的フロンティアの開拓を目指した重点領域が設定され，統合シミュレーション技術，可視化技術，並列分散ソフトウェア技術，アーキテクチャ技術が重点項目として選定された．

３．NAL における CFD 研究の現状と課題

NAL における CFD 技術の発展も基本的には上記の世界的流れと連動してなされて来た．このことは，我が国の航空宇宙産業が欧米主導で動いているという事実と突き合わせて考えてみれば容易に理解できよう．そういう中で NAL は，常に先進的な計算機ハードウェアを維持しながら CFD 技術の研究開発の先頭を走り，RANS 解析を民間へ根づかせる原動力の役割を果たして来た．また，実用開発プロジェクトあるいは研究開発現場との密接な係わり合いを持ちながら技術の有効性や利用可能性を示して来た．これは，現場の要請が CFD の発展を促し，CFD の発展が現場の技術課題を解決する糸口を与える，という相互啓発のシステムが一定程度有効に機能したからに他ならない．これらのことはまさに NAL の内外に誇れる成果であるといえる．しかしながら，我が国の場合は自国主導の航空機開発が少ない分，技術の実用化に向けてのインセンティブが十分とはいえず，個々の要素技術レベルは高いが，技術の統合（ツール化），応用分野の開拓，産業への移転という面では，世界に比べて必ずしも進んでいないのが実情である．

以上の認識をもとに，ここでは NAL における CFD 研究の現状と課題についてもう少し詳細な分析を試みる．なお，問題点や課題は「CFD を何にどう使うか」という視点によって変わる場合もあるが，ここでは航空宇宙における実用技術という観点から論ずる．

3.1 要素技術研究は第 2 フェーズへ

CFD を構成する個々の要素技術の研究開発は，第２フェ

ーズに入ったと考えて良い．つまり，スキーム，計算術等の基本的な要素技術の開発は一通り終わり，ボトルネックとなっている技術が顕在化し，その技術自身をどうにかしなければ CFD 技術全体のレベルが上がらない，他の方法では代替がきかない，という分野がある程度明らかになった．その代表格が「乱流モデル」と「格子生成」である．課題の技術的詳細についてはここでは触れないが，これらの分野は，論文にならない，個人の力ではどうにもならない，といった部分の活動を含み，ボトルネックからの脱却には研究スタイルまで含めた見直しが必要である．

3.2 応用技術（ツール）としては未熟

応用技術として総合的にみたときの CFD 技術はというと，ツールとしての本格的利用はまだ緒に就いたばかりであり，以下のような基本的問題を内包している．

① 精度に関する問題

精度が足りないというとき，「絶対的な精度が低い」というのと，「精度に関する情報が足りない」という２種類の形態があるが，現状では，精度に関する情報を取るのさえままならない状況にある．これは，技術開発側と応用側との連携が足りないゆえに，精度の達成基準が明確でなく，精度の客観評価ができないことに主に起因する．

② 意識・価値観の問題

CFD 技術をツール化するという作業は，CFD を応用分野で使いこなすためには必須のプロセスである．それは，CFD 技術に，システム性，ポータブル性，インターラクティブ性，メンテナンス性など，従来の研究開発における価値観（＝新規性，独創性）以外の要求を突きつける．しかし，開発側と利用側との連携不足により，そのような要求に対する開発側の対応は必ずしも十分とは言えなかった．従って，現状では「必要となる毎に一から用意しなければならない」ようなイメージが強く，利用側に「CFD を使えばそれなりに良い結果が出るのはわかっていても敢えて使う気がしない」という消極的雰囲気を作り出してしまっている．

③ 技術開発のアンバランスの問題

上記の発展経緯でも見たとおり，要素技術研究の中心がスキームと計算術に偏るという傾向があり，実用 CFD 計算のための道具（ツール）としてみたときの各要素技術間の成熟度にアンバランスが生じている．このことは，ボトルネック課題がなかなか解決されないまま残るという CFD 技術における構造的問題を引き起こし，ツールとして利用するときに鍵を握る「スループットの向上」が思うように図れない原因を生み出している．

④ 運用の問題

論文以外の成果物（例えば解析コードや解析データ）をどう使って，どう移転するのかについてのガイドラインや方策が見えない．例えば，今の仕組みでは，民間は NAL のコードを自由には使えない．今日のように混沌とした経済状況の中では民間も NAL だけを相手にしているわけにはいかないので，市販コードの利用とか大学との共同開発といった方向へ流れるのは必然である．また，前提となるソフトウェアの維持管理や情報共有についても，それを行うインセンティブが働かず，十分な対策が取られているわけではない．

3.3 技術活用のジレンマ

CFD 技術が空力設計を通じて航空機の性能向上に大きな役割を果たして来たことは誰もが認める事実であろう．しかし，今後とも同じ筋書きが成り立つかというとそうともいえない．例えば，航空機の空力性能の向上は燃費の改善に繋がるが，その向上のために莫大な開発費や時間がかかっていたらコスト的には意味がない．こうした場合に必要な視点は，ライフサイクルコストや多分野融合（Multidisciplinary）といった総合的な考え方である．総合的な視点が欠落し，とにかくレポートになる結果が出れば良いという成果中心主義で組み立てられた今日のNALのCFDコードではそのような考え方にはなじみにくく，手軽さと精度要求とを同時に満たすべく方向転換を迫られるのは必然であろう．また，航空機や宇宙機の設計や開発における CFD の利用が進んだ今日，NAL が大型高性能計算機を使って最先端の研究成果を着実に上げていくためには，利用者の CFD に対するニーズは何なのか，CFD 技術の利用は如何にあるべきか，について有効な答えを持っている必要がある．

3.4 組織的活動の欠如

NAL の CFD 研究は，プロジェクトなどの応用先の現場と密接に関連して発展して来たがゆえに CFD 研究者が各部門に散在する形となり，時として縦割り組織の弊害にさらされる中で，CFD 技術それ自体については，標準化，汎用化などの有効な組織的活動を立ち上げることができないまま今日に至っている．結果として，開発したソフトウェアが個人のレベルに止まり有効に相互利用されない，などといった問題を引き起こしている．

4. CFD 研究に関する NAL の果たすべき役割

2001 年における「独立行政法人」という新体制への移行を控え，上記の課題を着実に克服し，CFD 技術分野の研究開発における強力なリーダーシップを発揮して今後とも計算科学技術の発展を牽引して行くために NAL の果たすべき役割は以下の４点に集約される．

4.1 先駆的 CFD 技術研究開発への挑戦（CFD 技術における「知」のフロントランナーを目指して）

過去においてNALは，最先端の計算機システムを導入することで，時代を先取りするCFD研究課題に挑戦することが可能になり，その成果が新たな計算機を作る原動力を生み出す，という計算機の性能向上とCFD研究の発展における良好な関係を実現させて来た．今後もその関係を維持すべく，規模や斬新さの点でリスクが大きく民間では取り組めないような先端的・先導的技術課題や，共通基盤的なCFD基礎技術の研究開発へ大規模かつ集中的に取り組むことにより，計算科学技

術分野におけるリーダーシップを堅持することを目指す.

4.2 実用設計に耐え得る「数値風洞」技術の確立

国際共同開発が一般的になっている今日，NAL の計算機「数値風洞（NWT)」は空力分野において我が国が世界に誇り得る唯一の設備であると言っても過言ではない．航空宇宙分野の超大型計算において NWT を用いた研究が世界的な成果をあげていることは周知であり，この分野の充実なくしては世界の研究開発における日本の発言力はあり得ないだろう．しかしながら，現状では「数値風洞」の名に相応しいものに到達しているかといえばまだまだ不十分なところがある．精度の明確化や使いやすさの向上は当然のことながら，少なくとも定型メニューに従えば期待される結果が出てくる，という正に風洞試験のイメージがそこに実現されることが重要である．数値風洞を名実ともに充実させることで，次に進むべき方向が明確になる．

4.3 CFD 技術の研究拠点たること

我が国における CFD 技術の運用の現状を見ると，研究者間，組織間の連携が必ずしもうまく行われていない．個々の研究者は，優れた計算法，プログラムを開発してはいるものの，そのノウハウ，技術がまさに関係部門のみに止まっている側面がある．そのために，この分野に新しく参入して来た者は，自分で一からプログラムを作り始めなければならない．今後，CFD 技術をより一層高度化させ発展させるためには，技術基準，参照形状，計算コードや計算結果，実験データ含む検証用データを統一的に蓄積し発信できるような研究拠点の形成が必要であり，その役割に相応しい機関は NAL をおいて他にはないだろう．

4.4 利用方法，応用分野の開拓と実用性の実証

CFD のトータルな技術レベル向上を図る上での要素技術研究が重要なことは言うまでもないが，それもこれも使うあ・てがあっての話である．しかし，「CFD をどこでどう使うのか」という「利用・運用の問題」については十分語られていないのが現状ではあるまいか．NAL は大規模プロジェクトを抱えているだけでなく，航空宇宙「技術」研究所の名前が示す通り，NAL の保有する「技術」は我が国の航空宇宙産業の技術に直結すべきものであり，そういう意味で，CFD をどこにどう使うのが最も有効かという問いを常に投げかけ，それについて回答を与えて行くというスタイルを取っていくことが肝要である．

5. NAL-CFD 研究の重点テーマ

以上の論点を踏まえ，NAL が 21 世紀において中核的研究拠点となるための CFD 技術研究についての今後の目標として以下の４点を掲げる．

5.1 CFD におけるボトルネック技術課題への挑戦と克服

NAL は，CFD 技術の発展を加速し信頼性の高い工学的ツールとして実用の場に供して行く上で世界的にもボトルネックとなっている技術課題の克服に重点的かつ全力を挙げて取り組む.

(1)複雑格子生成

複雑な形状まわりの大規模な格子の生成能力は，数値シミュレーションの生産性を左右する重要な因子となっている．それを短時間で可能にすること，及び，格子の品質を迅速に評価できることが必要である.

(2)乱流モデル等のモデリング技術の高度化

CFD 技術の定量性を高めるには高精度な乱流モデルなどのモデリング技術の確立が必須である．例えば乱流モデルの場合，今後は直接数値シミュレーション（DNS）を利用したモデル構築・モデル検証が主流となって来るであろう．乱流のモデリング技術を飛躍的に向上させるには，より精緻な DNS データ，あるいはより現実的な流れに対する DNS データを取得する必要がある．それには，従来に比べ２倍以上の空間格子解像度，すなわち 10 億点規模の DNS が求められ，それを実現するには計算処理能力にして最低でも現行の数十倍，主記憶容量にして数テラバイト規模の高性能計算機が必要である．また，得られた DNS の結果をチャンピオンデータのデータベースとして発信するに足る十分な容量のデータ蓄積・管理能力が求められる.

(3)非定常 CFD 技術の確立

CFD 技術のボトルネックを解消するもう一つ重要な要素は，本格的な非定常計算技術及び非定常計算対応型システムを確立することである．従来の非定常計算の使命は，定常解析を補足する，あるいは定性的な非定常性を把握する程度の受動的なものであったが，もっと積極的かつ能動的に捕らえることで，従来の定常計算では精度良い解析が困難であった高揚力装置まわりの剥離流やジェットエンジンの内部流などの複雑現象の予測や，フラッタなどの流体関連振動あるいはヘリコプタの空力騒音などの診断や制御が可能となる．そのために，従来の枠に捕らわれないセミインプリシット時間積分法や解の局所補正法などの解法や，ラージエディシミュレーション（LES）などの新技術に果敢に挑戦する.

5.2 信頼性の高い標準解析ツールの整備

(1)高性能「数値風洞」等の構築

航空機の場合，安全性の確保は絶対条件であるから風洞試験の回数を減らせることはあってもをなくすことはできない．従って，CFD と風洞試験は共存しなければならない．コスト，精度などの条件を比較し使い分ければ良いのである．風洞試験の役割は，３分力測定などからバルクな量を取得し模型の空力性能を把握すること，圧力計測やシュリーレンからローカルな情報を得て現象に考察を加えること，の２点に集約されるだろう．このような風洞試験がまさにそこに実現されることを目指し数値風洞をイメージする．そこから全てがスタートすると考える．精度や使いやすさについては一貫性（標準化）がなければならない．ただし，現実問題として「格子生成」など現状ではクリアし難い部分もあるから，当

面はメニュー形式で対応することとし，少なくとも定型問題に対しては風洞実験に対応して１日で設計に有用なデータが得られることを目指す．具体的に例えば対象毎に以下の数種類の数値風洞を考え，最終的には，風洞試験を凌駕する本格的スループット性能と高信頼性を確立し，数値風洞の名にふさわしいデータ生産性とシステム運用性を追求する．

① 航空宇宙機数値風洞： 三次元翼などの構成要素に対しては半日で特性曲線（たとえばポーラカーブ）が描けるようにする．また，付属物付きの完全全機については，１日で性能特性が把握できるようにする．

② エンジン数値風洞： 圧縮機，タービンなどの構成要素に対しては半日で性能曲線が描けるようにする．また，エンジン全機については，１日で性能特性が把握できるようにする．

③ ヘリコプタ数値風洞： ブレードに対しては半日で性能曲線が描けるようにする．また，ヘリコプタ全機については，１日で性能特性が把握できるようにする．

④ 乱流数値シミュレータ： 航空宇宙分野で遭遇する乱流形態は，境界層，伴流など限られているので，境界層，伴流，チャネル流などの基本的な流れ形態についてのデータベース「デジタル乱流図鑑」を整備することを念頭に，１日で１ケースのデータが得られるようにする．

⑤ 数値宇宙エンジン： ロケットエンジン，スクラムジェットエンジンなどに関して，構成要素については半日で性能曲線が描けるように，システム全体については１日で性能特性が把握できるようにする．

(2)精度検証と NAL 基準の確立

従来の「数値風洞」では，NWT を NAL における新風洞と位置づけて，そのハード性能を活かすためのソフトを開発するという意味合いが強かった．だから，各航空機メーカがそのソフトを使うには NWT 自体を使うのが前提であった．しかし，今後は次世代の計算機で使えるような標準ソフトウェアを整備するという観点で考える．10 年後の普及計算機で使われているであろうソフトウェアを照準にソフト（NAL コード）開発を推進する．例えば NS コードの開発は，10 年前に着手していたからこそ今日の隆盛があるわけである．

標準化について議論して行くためには，多くの人がそれについて議論するためのプロトタイプが必要である．まずは NAL がそのプロトタイプを提案すべきであり，それが，NAL 基準，NAL 形状，NAL コード，NAL データという考え方であり，多くの人が CFD 技術についての各種議論を行うための共通の土俵を持つことに相当する．このプロトタイプを提供することで，NAL は研究拠点としての第一歩を踏み出すわけであり，同時にそれが次世代標準ソフトウェアへ向けてのたたき台となる．

5.3 次世代統合シミュレーション技術の構築

現代の航空機及び宇宙機開発環境の中では，従来のように個別の分野の設計改善や最適化のみで複雑かつ多様な要求仕様を満足させて行くのはもはや不可能になっている．このような問題を解決する有力な手段は，「多分野融合解析(Multidisciplinary Analysis: MDA)」や「多分野融合最適化(Multidisciplinary Design Optimization: MDO)」である．航空分野ではフラッターなどの空力と構造のカップリングによって起こる連成問題，宇宙分野では再突入時の空力加熱や熱防御といった空力と熱の連成問題が従来からしばしば扱われてきたが，ここでいう MDA ではさらに幅広い分野の融合を目指している．空力，構造，熱の他に，音，制御，推進等の主要なテーマから，生産，廃棄，再利用，さらには環境適合性，安全性といった周辺的・付随的な点に至るまで，いろいろな領域及び分野の MDA を含んでいる．

一方，MDO は，イメージ的には例えば空力だけでなく構造その他の諸条件を配慮して最適な機体形状を求めるような問題，数学的には適当な制限下である目的関数を最大最小にする変数値を求める問題として定義される．応用例としては，次世代超音速旅客機や宇宙往還機の機体形状，再使用型宇宙輸送機のエアロスパイクエンジン性能などが挙げられる．これらの対象が取り上げられる要因は，いずれのシステムもその実現のためには極めて厳しい性能的なブレークスルーが求められているからである．

5.4 高速高機能計算機システムの実現

統合シミュレーションに関する研究開発を推進して行くためには，極めて大規模で統合的な，かつユーザフレンドリーな計算環境の整備が必要である．それには，ホスト計算機の劇的な性能向上が不可欠であり，テラフロップスからペタフロップスへの更なる高速化を目指す．また，ネットワークを利用した情報の相互利用などが柔軟に行え，研究開発の現場の要請に的確に応えられる計算機環境が整備されることが必須である．例えば，共通データモデルの構築，データとソフトウエアの標準化と品質管理，大規模データの蓄積や検索，インテリジェントな可視化法の開発，ユーザインターフェース及びミドルウェアの整備，といったことが鍵になる．特にこの分野については，我が国は欧米に遅れをとっている．その効率的な推進に当たっては，計算科学技術分野に止まらず，数学，計算機科学まで含めた広範囲の分野の協同，あるいはそのような協同を可能にする組織の柔軟運営に至るまで，まさに今までの仕切り・区分けに捕らわれない Interdisciplinary な考え方が必要である．

６．今後の研究の進め方

現状の課題を打破し重点目標を実現するために NAL が今後進めるべき研究開発の進め方について論ずる．

6.1 重点研究テーマの設定

CFD の発展の黎明期にあっては，計算機の発達に呼応して CFD の要素技術をアプリで使える道具として体系化（コード化）する必要があったために，その要請が大きな求心力を発揮し，CFD 研究活動における自然な流れを形作ってい

た．しかし，今日の CFD 研究は，計算法，後処理等の各分野毎の研究が進められているが，設計ツールとしては十分に整っていない．そして，活動の求心力となるような大きな流れがなく，さらにいろいろな方向に出口を模索しなければならないことから，まとまった研究の方向性が自然発生的には現れにくい状態になっている．そこで，特定の課題におけるブレークスルーを積極的に誘発する，あるいは各分野を融合することで新たな展開を生起させる原動力／推進力と成り得るような集中的に取り組むべき重点研究テーマを設定する必要がある．

6.2 工学（エンジニアリング）への寄与

CFD の研究活動には３つの側面がある．すなわち，CFD それ自体をレベルアップさせる，自然科学に寄与する，工学（エンジニアリング）に寄与する，という側面である．例えば，計算スキームの研究は第１の範疇，乱流・燃焼等の CFD は第２の範疇に，実機開発プロジェクト等のための CFD は第３の範疇に含まれよう．航技研の CFD 研究もこの構図の中でなされてきた．乱流・燃焼等の現象理解に関する CFD は，個別の成果が科学への寄与に直接繋がるために CFD の成果として受け入れられやすいし，また実際に成果も挙げて来ている．ところが，工学に寄与する CFD は，各部門あるいは各プロジェクトにおいてそれぞれの活動の枠組みの中で実機開発に近いところで実施されて来たために，その具体的成果とか活動内容がプロジェクトの中に埋もれて見えにくく，プロジェクトにおける成果は常に CFD 技術の視点で整理していく必要がある．先般の外部評価で航技研の CFD に期待されていること（計算コードのツール化，成果の移転，等）は，工学に寄与する CFD の研究活動を実体化させ，目にみえるような形で社会に還元することと考えられる．従って，航技研の CFD（計算科学）の研究活動を大括りするテーマとして工学に寄与するテーマを積極的に前面に押し出して行くことが重要である．

6.3 プロジェクト的研究体制（分業化のすすめ）

工学に寄与するテーマを設定するとすれば，それは多くの専門分野の人間が係わってその協調の下に技術を総合するプロジェクトとなろう．工学の重要性や意義はむしろそうした「協調に基づく総合性」にある．このことは，成果は個人の業績である感の強い科学研究と対比してみればより明確である．CFD 研究に新たな息吹を与え，まとまった活動の求心力に成り得るようなプロジェクト的研究体制を取る必要がある．

6.4 ガイドライン（数値目標）の設定

技術的数値目標を立てることの第１の意義は，その計画が技術動向，社会的要請，投入する資源に対して妥当なものであるかの事前判断を可能にするところにある．第２に，その結果，成果の利用関係者が，何時，どのような技術を利用出来るかが分かり，それを前提として自己の計画をたてることが出来る．第３に，上記状況をふまえれば，資金的，人的，時間的な投入資源に対して最大の成果を生み出す計画を策定することが可能となる．全体の数値目標を設定すれば，それをブレークダウンして個別分野に数値目標を割り当てることにより，個別分野での効率的な開発が期待できると共に，個別分野間の協調も合理的に進められ，全体としても最大の技術成果を生み出すことが可能となる．これは，現在の NAL の散在した CFD 研究者のどちらかというと個別的・分散的な研究のやりかたを，より体系的な研究環境に変えていく契機にもなるものである．また，妥当性のある数値目標を設定することにより，乱流モデルや格子生成などの難しい問題に対しても段階的かつ着実な進歩を期待することが出来るようになる．

７．おわりに

本報告書は，CFD 研究を中心に NAL の計算科学技術の研究開発における将来ビジョンについて提言したものである．本ビジョンは，21 世紀初頭の計算科学技術の将来動向を見据えて策定した．本ビジョンに基づいて．具体的研究開発計画を策定し実施することが，次の課題である．

以上

付録 B　スーパーコンピューティングをめぐる内外の情勢

2000 年～2001 年時点でのスーパーコンピューティングに関する国内外の動向・情勢を俯瞰する．

B.1　スーパーコンピューティング技術の一般的動向

「**スーパーコンピューティング（Supercomputing）**」の厳密な定義があるわけではないが，スーパーコンピュータを使って汎用機やパソコン，ワークステーションではできない規模・速度の計算やシミュレーションを行うこと，というぐらいが現実の姿に近い説明であろう．昨今では，**高性能計算（High Performance Computing; HPC）**や**高度計算（High End Computing; HEC）**と呼ばれることもある．スーパーコンピュータ（以下，スパコン）とは，政府調達手続き上は，2007 年 10 月時点で **1.5TFLOPS 以上**の理論最高性能を有するものが対象[1]となるが，一般には現存するあらゆるコンピュータのうちで最高性能にランクされるものの一群を指す．

スーパーコンピューティング技術（以下，HPC 技術）の活躍する科学技術分野として，従来，高エネルギー物理学，原子力工学，化学，材料科学，航空宇宙工学，気象・天文，環境科学，人口知能などがよく知られている．対象がミクロ（または巨大）あるいは危険なために実験が困難であったり，膨大な費用がかかる分野，問題が複雑多様過ぎて理論解析が不可能な分野に相当し，主に「数値シミュレーション」あるいは「数値実験」「数値解析」という形で用いられる．昨今では，生化学（たんぱく質），医療（薬），ゲノム，核実験などの解析・シミュレーションにも適用されている．HPC 技術は，もともと米国クレイ社のコンピュータを使っていた機関（例えば米国の NASA や NCSA など）が中心となって発達させた経緯がある．その後，我が国の計算機メーカが開発発展の一翼を担い，自動車と並んで日米貿易摩擦の対象となるまでに至り，連日マスメディアを賑わせていたのは未だ我々の記憶に新しい．しかし，そのことは，とりもなおさずスパコンや HPC 技術が，科学技術の最先端のもの・テーマに深く係わっていた証拠でもある．

HPC 技術は，その規模の大きさや特殊さ故に国家戦略や科学技術政策としばしば結びついて発展してきた．最近では，米国における ASCI 計画（後述），我が国における地球シミュレータ計画（後述）などの HPC 技術に関する国家プロジェクトが注目されている．いずれも先進アプリケーションを実践するために高性能スパコンを開発するのが大目標とされ，ASCI では 2004 年までに 10TFLOPS のスパコンが，地球シミュレータでは 2002 年に 40TFLOPS のスパコンが開発されることになっている．その中で，高速計算アルゴリズム，問題解決環境，並列計算プラットフォーム，大規模データマイニング，大規模可視化，ネットワーク利用技術などの HPC 関連ソフトウェア技術も開発されることになっている．

B.2　ASCI 計画[2]

スーパーコンピューティングに関して有名なプロジェクトは，米国の ASCI（Accelerated Strategic Computing Initiative）計画である．その計画の全貌はここに書ききれないが，その目標は，一口で言えば「核実験のシミュレーション」である．核兵器の安全性，信頼性，性能維持の確認をするために，今後は核実験を行わずに数値シミュレーションによって検証して行こうとするものである．ASCI では，2004 年までに 100TFLOPS の高性能コンピュータを開発するとしており，1995 年にサンディア国立研究所に ASCI Red と呼ばれるインテルの 1.8TFLOPS のスーパーコンピュータが，ローレンスリバモア国立研究所には 1999 年までに ASCI Blue-Pacific と呼ばれる IBM の 3.2TFLOPS のマシンが，ロスアラモス国立研究所には 1999 年に ASCI Blue-Mounain と呼ばれる SGI の 3TFLOPS のマシンが，すでに導入されている．2000 年度にはローレンスリバモア国立研究所に ASCI White と呼ばれる IBM の 10TFLOPS のマシンが導入されている．また，ASCI ではかつてないほどの大規模なアプリケーションプログラムを動かすために，問題解決環境 PSE と呼ばれる統合的なシミュレーション環境が構築されることになっている．表 B.1 に ASCI 計画で導入された（される予定の）システム概要を記す．

表 B. 1　ASCI 計画で導入されたシステム

システム名	年度	機関	性能 TF	CPU 種類 /数
Red	1996	SDSC	1.8	PenPro 200MHz /9,072
Blue Mountain	1998	LANL	3.1	MIPS 250MHz /6,144
Blue Pacific	1998	LLNL	3.9	PowerPC 332MHz /5,856
White	2000	LLNL	12.3	Power3 375MHz /8,192
Q	2002	LANL	20.5	Alpha 1.25GHz /8,192
Red Storm	2004	SNL	41.5	Opteron 2.0GHz /10,368
BlueGene	2005	LLNL	360	PowerPC 700MHz /131,072
Purple	2006	LLNL	92.8	Power5 /12,544

SDSC: San Diego Supercomputer Center
LANL: Los Alamos National Laboratory
LLNL: Lawrence Livermore National Laboratory
SNL:　Sandia National Laboratory

B.3　地球シミュレータ計画[3]

旧科学技術庁は，理論研究，観測，計算機シミュレーションの三位一体で地球環境変動予測研究を推進するプロジェクトを平成 9 年度より推進しているが，その一貫として大気大循環シミュレーションでピーク性能 40TFLOPS の超高速並列スパコン「地球シミュレータ」を開発することとしている．地球シミュレータでは，現在の気候・気象の代表的なシミュレーション速度 5GFLOPS の 1000 倍の 5TFLOPS の実効速度を実現することを達成目標としている．また，エルニ

ーニョや温暖化，マントル活動といった地球規模での現象の高精度シミュレーションを可能とする並列ソフトウェアを開発することになっている[36].

B.4 情報通信技術（IT 技術）との係わり

近年，ネットワーク，情報通信機器，低電力・モバイル技術などの進展により，HPC 技術をそれら IT 技術と関連させて発展させようとする試みが盛んである．米国では，クリントン政権下で 1990 年代前半より **HPCC（High Performance Computing and Communication）** 計画[4]や **CIC R&D（Computing, Information and Communications Research and Development）** 計画[4]と呼ばれるコンピュータとネットワーク，関連ソフトウェアの技術向上と利用促進を目指した大規模なプロジェクトが国家施策として実施されて来た．また，IT 技術と計算機技術との連携として，**NPACI（National Partnership for Advanced Computational Infrastructure）**[5]と呼ばれる，地理的に離れた計算資源を有機的に結合し，いつ・どこからでも同じやり方で高性能計算ができる新たな研究インフラストラクチャを築くプログラムが実施されている．さらに，グローバルコンピューティング（広域分散計算）技術として，「グリッド」と呼ばれるインターネットを利用して分散した計算資源を有効利用する考え方が注目されている．例えば，①世界中のスパコンを同時に複数台使用することでこれまで不可能であった超大規模計算を行う（メタコンピューティング），②世界中の PC などの余剰時間を使ってこれまでできなかった発見的計算を行う（スループットコンピューティング），③席にいながらにして遠隔地の共同研究者と同じ画面イメージ，同じ計算結果を共有して高度なコラボレーションを行う（アクセスグリッド），などが具体的応用として提案されている．

欧州では計算機及びネットワークを利用して産業の生産性を上げ期間短縮を図ることを目的とする **ESPRIT（European Strategic Program for Research and Development in Information Technologies）** プロジェクト[6]や通信技術を向上させる **ACTS（Advanced Communications in Technologies and Service）** プロジェクトが推進された．現在は，**第 5 次総合計画（Fifth Framework Program; FP5）**[7]が実行されている．

我が国では，上述の地球シミュレータ計画の他に旧科学技術庁が **ITBL（IT-Based Laboratory）**[8]と呼ばれる各スパコンサイトのアプリケーションを共有し相互利用に供することを目的としたプロジェクトを 2001 年より 5 年計画で実施している．また，旧通産省系の **RWC（Real World Computing）** 計画[9]が，クラスタ型の高速コンピュータを開発することを目的に走っていた．

平成 9 年 7 月 28 日，内閣総理大臣から科学技術会議に対して諮問第 25 号「未来を拓く情報科学技術の戦略的な推進方策の在り方について」が提示された．科学技術会議は，答申に盛り込むべき施策等を策定するため，その下に情報科学技術部会を設置し，鋭意検討を進め，平成 11 年 6 月 2 日答申を行った．その中で，先導プログラムを設けることが謳われ，これを受けて科学技術会議 情報科学技術委員会では重点領域を設定し，先端的計算によるフロンティアの開拓として，統合シミュレーション技術，可視化技術，並列分散ソフトウェア技術，アーキテクチャ技術などを進めるべきことが提言されている（**付録 A** 参照）．

航技研でも，このような動向を受けて，「航空宇宙分野における情報科学技術の戦略的な推進方策の在り方」を模索するために，各方面の有識者から成る研究会を組織し，精力的に検討を重ね，大規模シミュレーションと高速ネットワークを用いた航空機及び宇宙機のコンカレントデザインシステムの構築の提案を含む調査・検討報告書を提出した．

B.5 航空宇宙におけるスーパーコンピューティング

航空宇宙分野におけるスーパーコンピューティング（HPC）技術の歴史は古く，今のベクトル型スパコンの原型である米国クレイ社のスーパーコンピュータ CRAY 1 が導入されたのも NASA の研究所が最初である．我が国においても，ベクトルスパコン FACOM 230-75AP を日本で初めて導入したのは航技研である．このように，スーパーコンピュータの最初のアプリケーションは航空宇宙から始まったと言っても過言ではなく，その意味で，航空宇宙は「HPC 発祥の地」と言うこともできる．以来，航空宇宙は HPC 技術における最前線の適用分野である．現在に至っても機体設計，開発，製造などの様々なフェーズで活躍している．特に，設計フェーズでは，近年，性能確認，パラメータ評価などのための解析ツールとして，また，それを使って形状自体を追求して行く設計ツールとして一般的に用いられている．HPC が航空宇宙においてこれほどまでに多用されて来た理由は，航空分野において，翼や機体のまわりの流れのシミュレーション（計算流体力学；Computational Fluid Dynamics; CFD）が，極めて多大な計算機リソースを要求したことが大きい．

航空宇宙の CFD は，HPC 技術分野における主要なアプリケーション分野の一つである．その特徴として，機体の空力特性を期待される精度で求めるために極めて高いシミュレーション精度が要求されること，流体現象の非線形性と硬直性のために直感的な近似や簡略化などが認められず，方程式系を忠実に解かなければならないこと，開発フェーズではパラメータ依存性を調べるためにパラメータを振った数多くのシミュレーションを行う必要があること，などが挙げられる．CFD のこのような特徴から莫大な計算機リソースを要求するため，スパコンに頼らざるを得なかったわけである．このような要求要件が急転することは考えにくいので，航空宇宙の CFD は今後とも HPC の主要なアプリケーション分野として生き残り続けるであろうことは想像に難くない．また，単にコストの問題の域を超えて，計算機利用者が計算機開発者に新しい開発のインセンティブを与え，開発者は利

[36] 地球シミュレータは，2002 年 3 月に NS-III より約半年早く導入された．8CPU16GFLOPS のベクトル共有メモリノード 640 台がクロスバで結合されたピーク性能 40TFLOPS，総メモリ量 10TFLOPS のシステムである．

用者に新しい利用のインセンティブを与える，というように開発者と利用者が一体となって技術を発展させて来たのも航空宇宙の CFD 分野の大きな特徴であり，上記の課題がありつづける限りはこの相互啓発の良好な関係は今後とも維持されるであろう．

米国では，NASA の **CAS (Computational Aerosciences)** プロジェクト[10]が，航空宇宙の HPC 技術分野のプロジェクトとして有名である．そこでは，産業界と連携し，航空機の設計にかかる時間とコストを削減することを目指している．複数の専門分野の視点から航空機全体の設計を最適化するフレームワークやエンジンの全機能をシミュレーションする**数値推進システムシミュレーション（Numerical Propulsion Simulation System; NPSS)** [11]などが検討されている．また，NASA のグランドチャレンジ問題として，航空機に影響する気象予測，ハリケーンの予測などが検討されている．我が国では，航技研の数値シミュレータ計画が，航空宇宙の HPC 関連技術開発計画として 1987 年より進められている．（第 2 章参照）

B.6 航空宇宙の CFD 大規模ソフトウェア開発

近年, オブジェクト指向や構造化によるソフトウェア開発の流れを汲んで，CFD の大規模ソフトウェアパッケージを開発する試みがなされている．ドイツの航空研究所 DLR では，**MEGAFLOW**[12]と呼ばれる輸送機の CFD 解析を対象とした形状定義から格子生成，ソルバー，後処理まで含む総合ソフトウェアパッケージを開発した．フランスの航空研究所 ONERA では **elsA**[13]と呼ばれる CFD ソルバーパッケージを開発している．米国では，国策レベルでソフトウェアをパッケージ化する試みは早くからなされ（例えば **OVERFLOW**[14]や **NPARC**[15])，民間への技術移転の結果，幾つかの市販品が既に出回っている．

B.7 オープン化，標準化の流れ

ビジネス分野は別として，研究開発の分野で，オープン化とか標準化という考え方・方法論が注目を浴びている．現在，オープンソースソフトウェアは Linux や FreeBSD といった OS から Apache，Perl，Tcl などの記述言語，GNU 系のコンパイラに至るまで各種そろっており，ソースコードをオープンにすることによる仕様の標準化，情報の共有，開発の共同化などをねらっている．無論，オープン化の裏にはメール転送やインターネットによる閲覧が自由かつ柔軟にできるようになったことがあることは言うまでもなく，今後ともこの流れは加速されよう．

B.8 インターネットとセキュリティ

情報通信技術や通信インフラとしてのインターネットの発展は HPC 界にも大きな影響を与えている．スパコンへのインターネットと通じたリモートからのアクセスが現実的になって来た．上述のグリッドなどの考え方はまさにインターネットの発展に因っている．しかしながら，HPC で扱わなければならないデータ量は，通常のインターネット上で往来するデータ量に比べ依然として膨大であり，HPC 用のリモートアクセス環境をどう整備していくかは今後の課題である．また，もともと汎用 UNIX と違った特殊 OS の世界で発展してきたスパコンシステムは，セキュリティ面での特殊性（ある面は強いが別の面では弱い）があり，今後想定されるオープン化や標準化の流れに乗りつつ必要なセキュリティ技術やセキュリティの仕組みをどう組み込んでいくかは，やはり今後の課題である．

参考文献

[1] http://www.kantei.go.jp/jp/kanbou/19tyoutatu/
[2] http://www.lanl.gov/asc/index.shtml
[3] http://www.jamstec.go.jp/esc/about/index.html
[4] 情報先進国の情報化施策と我が国の情報技術開発における重点分野の選択指針 II，先端情報技術研究所　技術調査部調査レポート　H12-6，平成 13 年 3 月．
[5] http://www.npaci.edu
[6] http://cordis.europa.eu/esprit/home.html
[7] http://cordis.europa.eu/fp5/
[8] http://www.itbl.jp/index.html
[9] http://keima.la.coocan.jp/rwcp/
[10] T. L. Holst, M. D. Salas, and R. W. Russell: The NASA Computational Aerosciences Program - Toward teraflops computing, AIAA　Paper 92-0558, 1992.
[11] J. K. Lytle: The Numerical Propulsion System Simulation: A Multidisciplinary Design System for Aerospace Vehicles, NASA TM-1999-209194, 1999.
[12] N. Kroll and R. Heinrich, MEGAFLOW-A Numerical FLOW Simulation System for Aircraft Design
[13] http://elsa.onera.fr/
[14]http://rotorcraft.arc.nasa.gov/cfd/CFD4/New_Page/Overflow-D2.htm
[15] http://www.grc.nasa.gov/WWW/wind/

付録 C スーパーコンピュータの技術動向

2000 年～2001 年時点でのスーパーコンピュータの技術動向を俯瞰する．

C.1 方式技術

スーパーコンピュータ（スパコン）の定義は，**付録 B** で述べたようにかなり曖昧であるが，一般には現存するあらゆるコンピュータのうちで最高性能にランクされるものの一群を示す．その意味で，**スーパーコンピュータのピーク性能は普通の計算機に比べ格段に高い**．しかし，その高速性を引き出すためには，アルゴリズム選択やプログラミング上の工夫が重要であり，用いるスパコンの特徴を十分に把握している必要がある．

スパコンには，よく知られているように，要素計算機の演算処理方式の差異によりベクトル型とスカラー型がある．

ベクトル型は，ベクトル要素計算機(ベクトルプロセッサ)を一つまたは多数並べることで高速性を実現するものである．プロセッサあたりの性能が高いので，単一プロセッサでもスパコンと呼ばれていたが，最近では，ベクトルプロセッサを多数並べたベクトル並列型が主流となっている．ベクトルプロセッサでは，複数のレジスタ上に並べられたベクタ間の演算を，多重化されたパイプライン演算器にデータを次々に流し込むことにより高速に実現する．近年，パイプライン多重度の増加やクロックの向上により，さらなる高速化を狙っている．

一方，**スカラー型**は，高性能マイクロプロセッサ（MPU）を多数並べることにより高速性を実現するものである．従来，単一 MPU の性能がそれほど高くなかったので，（最近はそうでもないが），基本的にスカラー並列型しか存在しない．性能を上げるためには数多くの MPU を結合する必要があり，MPU の数が特に多い場合には超並列計算機と呼ばれることもある．MPU は，キャッシュメモリを持ち，処理に必要なデータをできるだけ演算部に近いところに置いて，メモリとプロセッサ間のデータ転送を少なくする工夫をしている．結果としてコスト的にも有利となる．スカラー並列型スパコンは，メモリ（主記憶）をプロセッサに対してどう配置するかにより，共有メモリ型と分散メモリ型に分かれる．**共有メモリ型**は，さらに**対称型マルチプロセッサ型**と**分散共有メモリ型**に分類される．対称型マルチプロセッサ（Symmetric Multi Processor）は，**SMP** と略称され，このタイプの並列計算機は，各プロセッサ からみて相互結合網に接続されているすべてのメモリがハードウェア的に対等に接続されているものを指す．したがって，各プロセッサからは，一つの大きな大域（グローバル）メモリがあるようにみえる．最近のネットワークサーバ向きに開発されている高性能並列計算機では，数十台までのプロセッサとメモリ間を複数の高速バスや高速のスイッチで接続することにより，対等なメモリアクセスを実現している．高速かつ均質なメモリ空間を実現できる反面，メモリやデバイスへのアクセスの衝突により予期せぬ性能低下を招くことがある．

共有メモリ型のもう一つの形態は分散共有メモリ型である．この並列計算機では，各プロセッサ毎に分散配置された局所メモリがあり，このメモリに対してほかのプロセッサから大域メモリ管理機能を介してアクセスすることによって，実効的に共有メモリを実現する．分散共有メモリ型の並列計算機では，各プロセッサに接続された局所的なメモリへのアクセスに比較して，ほかのプロセッサに付属した共有メモリへのアクセスの遅延が大きくなるという問題がある．そのために，キャッシュ技術を利用することによって，共有メモリへのアクセスへの遅延を実質的に低減する工夫や，プロセッサ間のキャッシュの一貫性を保持する工夫がされている．

表 C.1 は，各観点からベクトル型とスカラー型のスパコンの特徴を比較したものである．最近の傾向として，ベクトル型スパコンはアーキテクチャ的にも要素技術的にも定着した感があるが，スカラー型については開発途上であり，逆に言うとこれはという決め手がない．**表 C.2** は，分類毎の各ベンダーの代表的機種を示す．また，**図 C.3** は，スーパーコンピュータの性能発展の経緯を時代を追ってプロットしたものである．

表 C.1 ベクトル型とスカラー型のスパコンの比較

	ベクトル型	スカラー型
ｱｰｷﾃｸﾁｬｲﾒｰｼﾞ	古典的	現代的
並列処理	単一命令複数データ	命令レベルの並列
メモリバンド幅	大	小
性能のポイント	メモリバンド幅 パイプライン数	キャッシュ上のデータの局所性
コンパイラ	ほぼ確立	発展途上
性能効率	高い（30%）	低い（10%）

表 C.2 スパコンの分類

分類	代表的機種
ベクトル型	CRAY 1, FACOM VP400, NEC SX-2, HITAC S-810
ベクトル並列型	共有メモリ型： CRAY YMP, NEC SX-3, HITAC S-3800 分散メモリ型： FUJITSU VPP500
スカラー並列型	共有メモリ型： SGI Origin3000, 分散メモリ型： CRAY T3D, Intel Paragon

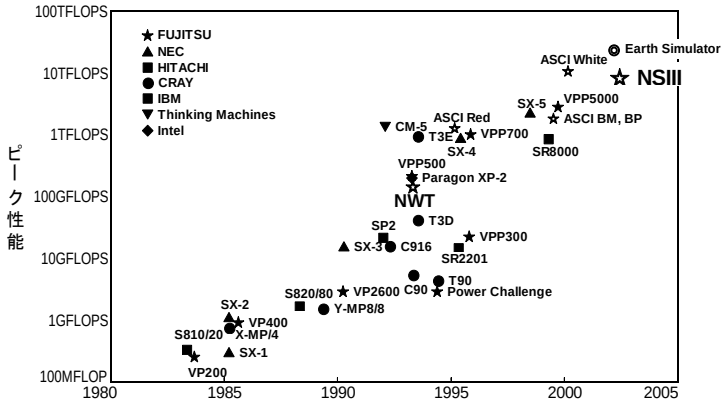


図 C.3　スーパーコンピュータの性能発展の経緯

かつて，スパコンといえばベクトル型が中心であり，ベクトル型といえば我が国の独壇場であった．航技研のスパコンシステムも代々ベクトル型であり，現在使われている「数値風洞」もベクトル型である．ベクトル型は，このようにスパコン市場で長らく主役の座を占めている．この方式が普及した理由として，高速性が簡単に引き出せる（実効性能が高い），過去のプログラム資産を有効に活用できる，多様な分野で高速性が引き出せる，各社のハードウェア構造が似ておりどのシステムでも高速実行できる，などが指摘されている．ただ，性能に見合う高速メモリの開発が技術的にもコスト的にも苦しくなりつつあり，クロック向上，低電力化，低廉化なども難しく，性能限界が見え始めているとも言われている．

しかし，最近になってそのようなベクトル中心の構図が変わりつつある．動きの中心になっているのが，スカラー並列型のスパコン，PC クラスタに代表されるクラスタ型システムである．キーワードはいずれも低コストであり，市場では低廉なシステムが求められる傾向が強くなって来ている．また，スカラー型の MPU の開発はすさまじい勢いで進んでおり，総合での実効性能がベクトル機を上回る時代もそう遠くはないであろう[37]．

C.2　半導体技術

米国半導体協会（SIA）の予測によれば，21 世紀初頭には 0.18μm の製造ルールが，2010 年には 0.05μm の製造ルールが使用可能になるとしている．これにより MPU の高密度化が可能になりより多くのトランジスタが実装できるようになり，2010 年頃には 1 チップ 10 億トランジスタの時代がやってくる．

クロックについては，PC のチップではすでに 1GHz の壁を超え 2GHz に達しようとしているが，RISC チップでも 21 世紀初頭には 500MHz を超え，2005 年頃には 1GHz が実現されると予想されている．

一方，メモリは，DRAM が主流になりコストは下がったが，MPU に比べると速度向上はそれほどでもなく，**MPU の速度向上が年率 60%**なのに対し，**DRAM のそれは年率 7%程度**である．プロセッサにいかに効率よく大量にデータを送り込むか，という **Memory Wall Problem** はますます深刻化するかもしれない．

C.3　結合ネットワーク技術

結合ネットワークは，**相互結合網**や**インターコネクト**とも呼ばれ，スーパーコンピュータまたは並列計算機を構成するプロセッサやメモリを互いに結合する．結合ネットワークは，並列計算機固有の技術であり，これをどう設計するかによって全体性能に大きな影響を与える．バンド幅と同時に**レイテンシ（スイッチ遅延時間）**が短いことが重要である．以前は，ハイパーキューブ，トーラス，メッシュなどの各種の結合形態（トポロジ）が議論されたが，最近のスーパーコンピュータでは，単段または多段の**クロスバ**がよく使われている．一方，PC クラスタでは，**ギガビットイーサネット（GigabitEthernet）**，**ミリネット（Myrinet）**[38]などが主に使われる（**表 C.4**）．結合ネットワークは，これといった決め手が無く，スーパーコンピュータに混乱を招く一因となっている．この部分は，性能を求めようとすると，一般に極めて高価につくので選択に注意が必要である．

表 C.4　主要結合ネットワークの諸元

結合ネットワークの種類	転送速度（片方向）	レイテンシ	特徴
クロスバスイッチ	4～8GB/s	10 マイクロ秒程度	動的網，高速だが高価
ミリネット	1.28Gbps	10 マイクロ秒程度	高速だが高価
ギガビットイーサネット	1Gbps	200 マイクロ秒程度	低速，安価

C.4　クラスタリング技術

最近注目されているのは，クラスタ技術とメタコンピュータ技術である．クラスタとは，同機種の計算機を一ヶ所に集め，ネットワークで接続することにより，仮想的に一つの計算機として作動させるものである．Beowulf[1]，PAPIA[2]などのプロジェクトがよく知られているが，低コストの反面，信頼性・可用性などを如何に向上させるかに課題がある．一方，メタコンピュータは，地理的に分散した異機種の計算機を高速ネットワークで接続することにより，ネットワークを計算機の一部として利用しつつ高性能計算資源を構築するというものである．米国では，Globus[3]，Condor[4]，わが国では Ninf[5]などのプロジェクトが精力的に進められているが，ユーザ認証，プロセス制御，異機種間通信などに課題がある．

[37] 並列度が高くなるとますます実効性能が下がるという「**発散問題（Divergence Problem）**」を克服する必要がある．

[38] Myricom 社が提供する結合ネットワーク．

C.5 ストレージ技術

スパコンにおけるストレージは，その容量の大きさもさることながら読書速度も重要である．スパコン用の高速ディスクとしては，ファイバチャネルをアクセス線とする RAID ディスクアレイが普及して来ている．ファイバチャネルの速度は 100MB/s だが，これを束ねる（ストライピングする）ことで，高速の転送速度を実現できる．

C.6 ソフトウェア技術

スパコンの OS としては，64 ビット UNIX という選択が一般的なようである．PC クラスタでは，32 ビットの OS を採用する例もあるようだが，OS あたり（ノードあたり）のメモリ空間がせいぜい数 GB に限られてしまう問題がある．米国のシステムで多いのは OS に Linux という選択である．オープンソースなのでカスタマイズしやすいというメリットがあるが，安定性や保守性を考えるとシステムまわりのスタッフが充実している必要があり，我が国においては採用しにくい OS である．

一般的なプログラム開発においては，C や C++，あるいは Java といった言語の使用が増えている．しかし，スパコンの世界では Fortran が主流である．ソフトウェアの蓄積があること，性能を引き出しやすいことなどが理由である．今後も Fortran の利用は必須である．ただ，C/C++の利用も今後は増えるであろう．並列処理関連のソフトウェアも重要であるが，これに関しては添付資料 E を参照されたい．

C.7 Top500 サイト情報

Linpack ベンチマークの数値をベースに，世界のスーパーコンピュータの動向についての情報を発信している Top500 というウェブサイトがある．URL は，

http://www.top500.org

である．2001 年 1 月における情報からいくつかを挙げてみる．**図 C.5** は，Linpack 性能の 1 位のシステム（N=1）及び 500 位（N=500），1-500 位の性能の総計（SUM）を年度毎にプロットしたものである．直近のトップは，LLNL の ASCI White（**付録 B** 参照）であり，Linpack 性能は 7.23TFLOPS であることがわかる．我々のシステムの目標である 10TFLOPS は，Linpack 性能が約 5 割としても，世界有数のものとなることがわかる．

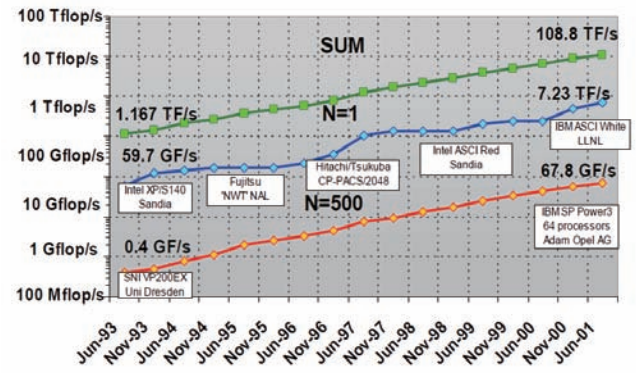


図 C.5 Linpack 性能の年度別変遷

Top500 サイトによれば，

- Linpack 性能の年向上率 1.82
- プロセッサ数の年向上率 1.30
- プロセッサ性能の年向上率 1.40
- ムーアの法則による年向上率 1.58

であり，プロセッサ性能の年向上率はムーアの法則を下回っているものの，Linpack 性能，すなわちシステム性能はムーアの法則を上回っている．これは真に並列計算技術，つまり，CPU を並列に沢山つなげることにより性能向上を図っている証拠でもある．ただし，これらの数字は技術的な可能性を言っているだけで，コスト一定とか面積・電力一定とかの制約下の数字ではないことに注意する必要がある．その意味で，注意が必要ではあるが敢えて数字をはじいてみると，NWT から NS-III までの 8 年間で，Linpcak 的には $1.82^8 = 120$ 倍，ムーアの法則では $1.58^8 = 39$ 倍の性能向上が技術的には可能であることがわかる．

参考文献

[1] http://www.beowulf.org/
[2] http://mbs.cbrc.jp/papia/
[3] http://www.globus.org/
[4] http://www.cs.wisc.edu/condor/
[5] http://ninf.apgrid.org/papers/papers.j.shtml

付録 D　並列処理の技術動向

D.1　はじめに

並列処理とは，多数のプロセッサ[39]に処理を分割して，処理全体の高速化を図ろうとするものである．最近のプロセッサの性能向上には目を見張るものがあり，数 GFLOPS の性能を持つプロセッサも出現している．しかし，CFD の実際の大規模アプリケーションにおいては，時として 100GFLOPS を超える性能が必要となる場合があり，このような高性能を要求する際には並列処理が必須となる．

D.2　並列処理の分類

命令レベルの並列

プロセッサの内部における処理のレベルでの並列処理である．最新のプロセッサでは複数の演算器（ハードウェア）が用意されており，命令を取ってきたときに並列に実行可能なように解読し，演算器に命令を出す**スーパースカラ（Super Scalar）方式**と，複数の同時実行可能な命令を一つ長い命令に格納して実行する **VLIW（Very Long Instruction Word）方式**がある．

SIMD 方式

SIMD(Single Instruction stream, Multiple Data stream)方式は，一つの命令が複数のデータに対する同一の処理を同時に制御する．これをハードウェアで実現するには，1 個の演算器にデータを流し込み，演算を実行させる時間並列（パイプライン）方式と，多数の演算器を並べて各々で担当するデータに対して演算を実行させる空間並列方式がある．SIMD の典型にベクトルスーパーコンピュータがある．

MIMD 方式

MIMD(Multiple Instruction stream, Multiple Data stream)方式は，各プロセッサが独自のプログラムを解釈しながら動作する．プロセッサを狭い空間に多数実装し，高速の結合ネットワークで結合させ，協調させながら動作させるマルチプロセッサ方式と，多数のコンピュータを比較的遅いネットワークで結合させるクラスタ方式がある．

D.3　並列システムの分類

D.3.1　プロセッサタイプによる分類

ベクトル並列型

ベクトルプロセッサを要素とするもので，最近ではプロセッサ自体が 10GFLOPS 程度の高い処理性能を有するため，数 10～数 100 個の結合形態が良く用いられる．メモリからのデータ供給能力が高く，実効性能が高いという特徴がある．

スカラー並列型

スカラープロセッサを要素とする．最新のスカラープロセッサは，すべてキャッシュメモリを内蔵している．キャッシュは，数 10KB－数 MB の容量を持つ高速メモリで構成される．現在のスカラー単体プロセッサの処理性能は高くて数 GFLOPS 程度であるが，安価なので，数 100～10,000 個程度の結合形態が用いられる．多くのプロセッサを用いる並列システムでは，プロセッサ間通信がボトルネックとならないような構成とする必要がある．

D.3.2　メモリ形態による分類

集中共有メモリ型　図 D.1(a)

同等の機能を持つ複数のプロセッサと，1 つのメモリ空間（共有メモリ）から成る．すべてのプロセッサは同じデータにアクセスすることができる．特に，プロセッサからメモリのアクセス遅延がプロセッサによらず一様であるものを **SMP（Symmetric Multi Processor）**と呼ぶ．

分散共有メモリ型　図 D.1(b)

各プロセッサにローカルにメモリを分散配置しているが，論理的には共有メモリとして参照できる．メモリに対するアクセス性能がプロセッサとの位置関係によって異なるため，**NUMA（Non Uniform Memory Access）**と呼ばれることがある．

分散メモリ型　図 D.1(c)

プロセッサとメモリから成るシステム要素が結合ネットワークで結合された形態．異なるシステム要素にあるデータへのアクセスは，プロセッサを介した通信によって実現される．

クラスタ型　図 D.1(d)

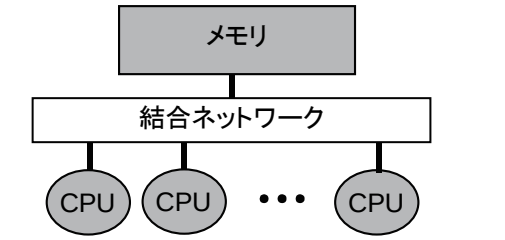
分散メモリ型のうち，特にそれ自体が一つの計算機であるパソコン（PC）やワークステーション（WS）が結合ネットワークで結合された形態をクラスタ型と呼ぶ．特に，独立のパソコンを多数結合したものは**PC クラスタ**と呼ばれる．

D.4　並列プログラミングと並列言語

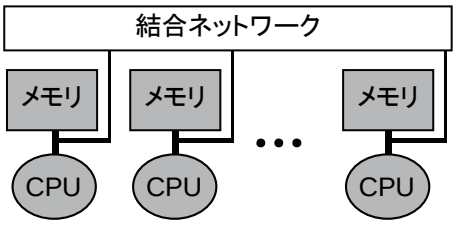
プログラムを並列処理する方法は，スレッド並列処理とプロセス並列処理に大別される．スレッド並列は，実行の制御**（スレッド；thread）**を複数持つが，データやファイルなどの資源はスレッド間で共有する並列実行の形態である．一方，プロセス並列では，各**プロセス（process）**はそれぞれ実行の制御と固有のデータ空間を持ち，プロセス間のデータのやり取りはプロセス間通信によって行われる（**図 D.2** 参照）．プロセス間通信は，通信ライブラリを用いて直接記述されるか，コンパイラによって生成される．共有メモリ型計算機では，スレッド並列が使われることが多く，分散メモリ型計算機ではプロセス並列が使われることが多い．

---

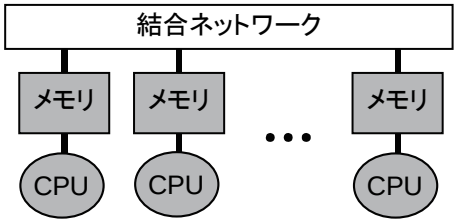
39 ベクトル計算機の要素プロセッサやパソコンの CPU を含め，演算処理を行う部品をプロセッサと呼ぶ．



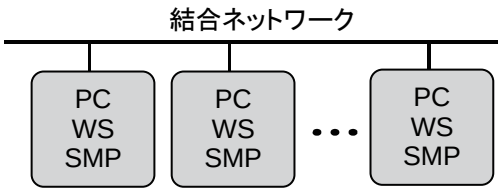
(a) 集中メモリ型



(b) 分散共有メモリ型



(c) 分散メモリ型



(d) クラスタ型

図 D.1 メモリ形態による並列システムの分類

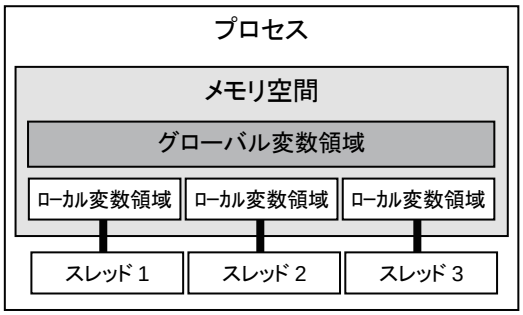
D.4.1 スレッド並列処理

スレッド並列では，あるスレッドが更新したデータの値は別のスレッドでも直接参照可能となる．複数のスレッドから同じデータに書き込みがある場合には，読み書きのタイミングに依存する問題を避けるために，適切な排他制御を行う必要がある．また，必要な場合には個々のスレッドに固有のデータを持つこともできる．例えば，並列 DO ループの実行中には, DO 変数は一時的に各スレッドに固有のデータとなる．スレッド並列処理を記述するための並列言語に **OpenMP** がある[1]．OpenMP には Fortran をベースにしたものと C, C++をベースにしたものがあり，どちらもベース言語を**指示文（directive）**によって拡張した言語仕様である．OpenMP は，ループやタスクの並列化，排他制御や同期処理，アフィニティ制御，スレッド固有変数の扱いなどに関する指示文を持つ．また，並列実行を制御したり，各種問い合わせのためのライブラリルーチンや環境変数も用意されている．並列性の検出が容易なループについては，自動的にスレッド並列化できる**自動並列**の機能を持つコンパイラ処理系もある．

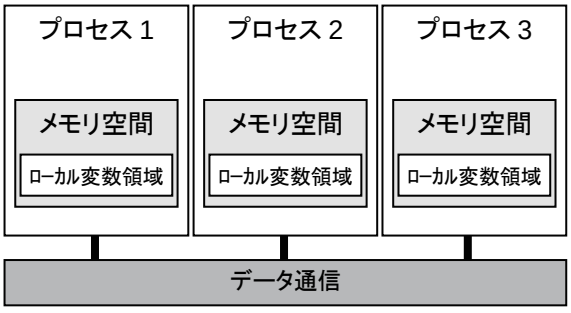
D.4.2 プロセス並列処理

プロセス並列は，同じプログラムに異なるパラメタ（プロセッサ番号など）を与えて同時実行することによって実現する場合が多い．このような実行方法を，**SPMD（Single Program/Multiple Data）**と呼ぶ．一方，異なるプログラムを複数プロセスで同時実行する方法を，**MPMD（Multiple Program/Multiple Data）**と呼ぶ．SPMD プログラムの記述には，主に **MPI（Message Passing Interface）**[2]が使われる．MPI は，Fortran または C から呼び出すことのできるライブラリであり，プロセッサ間の送受信，ブロードキャストや集計演算などの集合通信，プロセッサ間同期処理などを含む. MPI-2.0 では, MPMD プログラムの記述も可能である．MPI プログラムの作成においては, 利用者はすべての通信を明示的に記述するとともに，データ依存順序の保証やデッドロックの回避を利用者の責任で行わなければならない．

SPMD プログラムの作成の煩わしさを緩和するために, コンパイラにより SPMD コードを生成するための高級言語が使われる．利用者は，データをグローバルなイメージ，すなわち, プロセッサ間で共有される変数（仮想グローバル空間）であるかのように記述することができる．そのような言語として最も標準的なものが，**HPF（High Performance Fortran）**[3]である．HPF は，Fortran をベースとし，指示文によってデータの分散方法を指示したり，ループやタスクの並列化に関する情報をコンパイラに伝えたりすることができる．原則として，並列化はデータの分散に沿って自動的に行われる（データ並列）．HPF を拡張した HPF/JA 言語仕様[4]も実用化されている．その他に富士通が開発したものに **VPP Fortran**[5]や **XPFortran**[6]がある．



(a) スレッド並列



(b) プロセス並列

図 D.2 並列プログラミングのイメージ

```
program main
dimension dif(1000),u(1000)
  :
c = 2.0
do i = 2, 999
  dif(i) = u(i+1) - c*u(i) + u(i-1)
end do
  :
end program main
```

図 D.3　逐次プログラム

図 D.3 の逐次プログラムを，OpenMP, XPFortran, MPI により並列化した場合のプログラミングの実際を**図D.4**に示す．MPI のプログラムは非常に複雑であるが，現実には SPMD 型の記述が主であり，このようなループ単位の並列化は行わないことに注意する．

D.4.3　ハイブリッド並列処理

SMP クラスタ型などの**階層性**を持った並列システムでは，並列化手法を組み合せる方法が考えられる．例えば，SMP 間（クラスタ部）は MPI を使ってサブプログラム間などの粒度の大きい並列性を記述し，SMP 内は OpenMP を使ってループ並列等の細粒度並列を記述する，というハイブリッドな並列化手法を採ることができる．各々の並列化手段の組み合わせによって，柔軟なプログラミングが可能となり一層の高速化が期待できる．時にはハードウェアの性能を最大限に活かすこともできるが，全てを MPI によるプロセス並列で記述した場合に比べそれほど性能が上がらないことも多く，現実的にはプログラミングは複雑になりユーザの手間は増す．性能評価の難しさも加わり，まだ研究段階にある．

D.5　並列処理の性能

D.5.1　性能因子

並列処理の性能を左右する主要因子として，並列化率，並列化粒度，キャッシュ競合，プロセッサ負荷分散，通信コストなどがある．

①並列化率と速度向上

問題規模が一定の場合，並列処理できる部分の割合（並列化率）を a とし，n 台のプロセッサを用いて得られる理論速度向上率 $S(n)$ は，

$$S(n) = \frac{1}{(1-a) + a/n}$$

で与えられる．これを**アムダールの法則（Amdahl's Law）**という．例えば，逐次処理部分の割合が 10%あれば，並列部分の処理割合は 90%あるにも拘わらず，上式よりどんなに多くのプロセッサをつぎ込んでも得られる速度向上率は高々10 倍程度しかならないことを意味する．

一方，問題規模が変化する場合には，並列化率を a とすると，理論速度向上率 $S(n)$ は，

$$S(n) = 1 + a(n-1)$$

で与えられる．これを**グスタフソンの法則（Gustafson's Law）**という．これによれば，90%の並列化率があれば，$S(n) = 0.9\,n$ となり，高速化はほぼスケールすることになる．

```
    program main
    dimension dif(1000),u(1000)
      :
    c = 2.0
!$OMP PARALLEL DO
    do i = 2, 999
      dif(i) = u(i+1) - c*u(i) + u(i-1)
    end do
      :
    end program main
```

(a) OpenMP による並列化

```
    program main
!XOCL PROCESSOR P(4)
    dimension u(1000),dif(1000)
!XOCL INDEX PARTITION Q=(P,INDEX=1:1000)
!XOCL GLOBAL u(/Q(overlap=(1,1))),dif(/Q)
!
!XOCL PARALLEL REGION
      :
    c = 2.0
!XOCL OVERLAPFIX(u)(id)
!XOCL MOVE WAIT(id)
!XOCL SPREAD DO REGIDENT(u,dif) /Q
    do i = 2, 999
      dif(i) = u(i+1) - c*u(i) + u(i-1)
    end do
!XOCL END SPREAD
      :
!XOCL END PARALLEL
    end program
```

(b) XPFortran による並列化

```
program main
 include "mpif.h"
 real(kind=4),dimension(:),allocatable :: dif,u
 integer STATUS(MPI_STATUS_SIZE)
   :
 call MPI_INIT(ierr)
 call MPI_COMM_SIZE(MPI_COMM_WORLD,  npe,ierr)
 call MPI_COMM_RANK(MPI_COMM_WORLD,myrank,ierr)
 im  = 1000
 ilen =  (im + npe - 1 )/npe
 ist  =  myrank*250 + 1
 iend =  ist + ilen - 1
 allocate( u(ist-1:iend+1), dif(ist:iend) )
 nright = myrank + 1
 nleft  = myrank - 1
 if(myrank==  0) then
  nleft  = MPI_PROC_NULL
 else if(myrank==npe-1) then
  nright = MPI_PROC_NULL
 end if
 call MPI_SENDRECV( u(iend  ),1,MPI_REAL,nright,0, &
                    u(ist-1 ),1,MPI_REAL, nleft,0, &
                    MPI_COMM_WORLD,STATUS,IERR )
 call MPI_SENDRECV( u(ist  ),1,MPI_REAL, nleft,1, &
                    u(iend+1),1,MPI_REAL,nright,1, &
                    MPI_COMM_WORLD,STATUS,IERR )
 c      = 2.0
 ist_do  =  max( 2,ist )
 iend_do  =  min(999,iend)
 do i = ist_do, iend_do
  dif(i) = u(i+1) - c*u(i) + u(i-1)
 end do
   :
 call MPI_FINALIZE(ierr)
end program main
```

(c) MPI による並列化

図 D.4　並列プログラミングの実例

②並列化粒度

並列化の粒度とは，複数のプロセッサに割り当てられる処理の大きさを指す．粒度が小さくなれば並列化のためのオーバヘッドの割合が大きくなる．オーバヘッドには，プロセッサ間の通信や同期処理等がある．粒度を大きくすることにより，これらオーバヘッドを軽減することができる．例えば，ループを対象とする並列処理を行う場合，可能な限り外側のループで並列化したり，ループの繰り返し回数を多くしたりすれば良い．

③ロードバランス（プロセッサ負荷分散）

並列処理時に，特定のプロセッサに負荷が偏ると，全体の実行時間がそのプロセッサの実行速度に引きずられ，性能が悪くなる．各プロセッサの負荷（分担）は均等にするのが望ましい．

④キャッシュ競合

スカラー機では，スレッド間のキャッシュ競合がしばしば性能上の問題となる．一般の商用計算機では，プロセッサとメモリとの間に 2～3 階層のキャッシュが存在し，同じデータに対して頻繁に読み書きを行う場合の高速化を図っている．しかし，例えば他のプロセッサが同じデータに書き込みを行う場合などには，キャッシュ上のデータは無効になり，再度メモリからキャッシュにデータを取り込む必要が生じる．スレッド並列処理では，このようなキャッシュ競合をできるだけ避けることが性能向上のポイントとなる．

⑤通信コスト

プロセス並列処理では，プロセス間のデータ送受信，ブロードキャストや集計演算などの通信が必要となる．データ通信処理を計算処理と同時に行うことで，実質的にこれら通信コストを抑えるプログラミングが大きなポイントとなる．

D.5.2 並列計算機の性能評価

並列計算機を評価するときの指標の一つに，スピードアップ性能とスケールアップ性能がある．**スピードアップ性能**は，問題規模を同一にしたまま並列数を増加させていったときの並列システムの速度向上比を示し，処理のオーバヘッドや通信の立ち上がり速度（レイテンシ）などに関係する（**Strong Scaling** とも言う）．一方，**スケールアップ性能**は，プロセッサあたりの負荷を同一にしたまま並列数を増加させていったときの並列システムの速度向上比を示し，大量のデータ転送に対しての通信性能（スループット），メモリへのデータ供給能力などに関係する（**Weak Scaling** とも言う）．例えば，スピードアップ性能は，**図 D.5** に示したように，並列数の増加とともに線形（理想値）には増加しなくなる．プロセッサ数あるいは並列度数の向上とともに性能が向上する度合いを「**スケーラビリティ**」と言い，性能向上が線形に近いほど「**スケーラビリティが高い**」あるいは「**スケールする**」と言う．

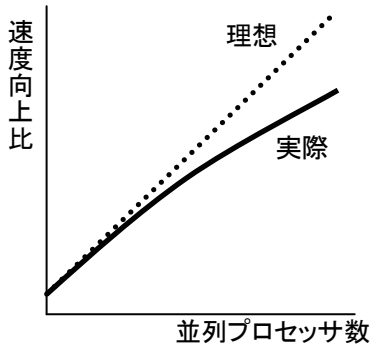


図 D.5 スピードアップ性能

D.6 CFD 計算における並列処理

数値流体力学（CFD）計算は，差分法や有限要素法などを用いた連続体モデルによるものと，直接シミュレーション・モンテカルロ法やセル・オートマトン法などを用いた粒子モデルによるものに大別される．ここでは，連続体モデルによる解析に適用される並列化手法について述べる．粒子モデルによる解析の並列化手法に関しては，セル間での粒子の移動によるロードバランスの変化に注意する必要がある．

D.6.1 領域分割による並列化

領域分割法とは，解析対象領域を複数の部分領域に分割し，その部分領域を各プロセッサに割り当てて計算する手法である．CFD により複雑な形状を扱おうとする場合，全空間に単一の構造格子を作成するのは困難なため，複数の部分的な構造格子（ブロック）を組み合わせざるを得ない．各ブロックを重なり合わないようにして空間を埋め尽くす方法をマルチブロック法と呼び，重なりを許す方法をオーバーセット法と呼ぶ．こうした方法では，もともと複雑な形状を扱おうとして計算領域を分割したのであるが，並列化のための領域分割として平行して利用することができる．ただしその場合，ロードバランスを均一にするように分割することが並列性能の観点からは重要である．

D.6.2 データ分割による並列化

乱流の直接シミュレーションなどのように大規模な単一領域を解析対象とする際には，配列構造やループ構造を念頭に細粒度の並列化戦略を考えた方が有利である．例えば，空間 3 次元問題を解く場合に，ループ長の大きい 3 重 DO ループ構造がプログラムの各所に現れる．3 重 DO ループのうち，どれか 1 軸は他の 2 軸とは独立に並列化軸とすることができる．ただし，並列化軸の入れ替えに伴う転置（transpose）処理や袖の処理が必要となる．

D.6.3　PC クラスタの利用

近年，CFD における PC クラスタの利用が増えつつある．PC クラスタ関連の技術動向としては，Intel，AMD を始めとするプロセッサやネットワークの性能向上，低コスト化は目覚ましいものがある反面，メモリ性能や I/O 性能がボトルネックとなりつつある．最近では，ブレードサーバと呼ばれる 19 インチラックへマウント可能ユニットを用いて集積度を高める試みがなされている．小規模構成なら低コストで性能も出しやすい．しかしながら，大規模構成になると，結合ネットワークが相対的に高価になったり，配線やインテグレーション，システム管理に手間がかかる．大多数で利用するセンター運用では依然として問題が多い．大規模システムの事例としては，国内では，新情報処理開発機構において SCore と呼ばれる PC クラスタ用のシステムソフトウェアを開発し[7]，これを用いて 1024 台の PC クラスタを構築している．国外では，Beowulf と呼ばれるシステムが広く利用されている[8]．これら SCore や Beowulf 等の PC クラスタでは, OS として Linux, 並列通信ライブラリとして MPICH[9] が一般に利用されている．

参考文献

[1] OpenMP Architecture Review Board: OpenMP: A Proposed Standard API for Shared Memory Programming, http://www.openmp.org/, 1997.

[2] Message Passing Interface Forum: MPI : A Message-Passing-Interface Standard, http://www.mpi-forum.org/, 1994

[3] High Performance Fortran Forum: High Performance Fortran Language Specification Version 2.0, 1997.

[4] 富士通，日立，日本電気訳：High Performance Fortran 2.0 公式マニュアル，付録：HPF/JA 言語仕様書, Springer, pp.311-386, 1999.

[5] 岩下英俊，進藤達也，岡田信：VPP Fortran: 分散メモリ型並列計算機言語，情報処理学会論文誌, Vol. 36, Vol. 7, pp.1542-1550, 1995.

[6] 富士通：XPFortran 使用手引書，2002.

[7] PC クラスタコンソーシアム, “SCore Cluster System Software 5.2 ドキュメント”, http://pdswww.rwcp.or.jp, 2002.

[8] Becker, D. J., et al., “Beowulf: A Parallel Workstation for Scientific Computation,” Proceedings of International Conference on Parallel Processing, 1995.

[9] Gropp, W., “A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard,” http://www-unix.mcs.anl.gov/mpi/mpich, 1994.

付録 E スーパーコンピュータの政府調達手続き

スーパーコンピュータや大型のコンピュータについては，「政府調達」として，定められた手続きの則って調達するものとされているので，ここでは，その根拠となる文書を参考に掲げる．

1 我が国の政府調達に関する規定

我が国の「国の機関」の政府調達手続については，法律では「会計法」(昭和22年法律第35号)，政令では「予算決算及び会計令」(昭和22年勅令第165号)及び「予算決算及び会計令臨時特例」(昭和21年勅令第558号)，省令では「契約事務取扱規則」(昭和37年大蔵省令第52号)が制定されている．（資料I-1）

さらに，「政府調達に関する協定」（以下，WTO政府調達協定）(平成7年条約第23号)その他の国際約束が適用される調達(注1)のうち，国(中央政府)の機関については，「国の物品等又は特定役務の調達手続の特例を定める政令」(昭和55年政令第300号)及び「国の物品等又は特定役務の調達手続の特例を定める省令」(昭和55年大蔵省令第45号)により，WTO政府調達協定その他の国際約束上の調達手続を国内法令上確保している．加えて，各省庁等においては，これらの規定に基づいた調達手続の細則を示す契約規則，資格審査規定などが定められている．

(注1) WTO政府調達協定の適用対象機関は，国(中央政府)の機関，地方政府の機関(地方公共団体)，その他の機関（特殊法人及び独立行政法人等）であるが，このうち，国の機関以外については，それぞれ地方自治法に基づく政令等，あるいは特殊法人または独立行政法人等ごとに定めている内規等に協定と適合した規定を設け，協定の国内における実施を確保している．

我が国は，これら会計法令上の調達手続に加え，内閣に設置されたアクション･プログラム(AP)実行推進委員会（資料I-2）が，WTO政府調達協定上の手続を上回る内外無差別・公正・透明な手続(注2)を自主的措置（資料I-3）として策定するとともに，そのフォローアップを着実に実施しているところである．

(注2) 例えば，「WTO政府調達協定」では，国の機関及びその他の機関による物品・サービスの調達につき，13万SDR以上の調達契約が対象となっているが，自主的措置では，10万SDR以上13万SDR未満の調達契約についても，「WTO政府調達協定」に準じて対処することとされている．また，同協定上，40日以上とされている応札期間については，自主的措置上，50日以上とされている．

2 政府調達に係る自主的措置の経緯について

アクション･プログラム実行推進委員会が策定している政府調達に係る自主的措置の経緯については次のとおり．

(1) 「市場アクセス改善のためのアクション･プログラム(骨格)」の段階

a. 「対外経済対策」の決定

昭和60（1985）年4月9日，経済対策閣僚会議は，対外経済問題諮問委員会の政策提言を踏まえて，「対外経済対策」を決定した．この中で市場アクセス改善のためのアクション・プログラムの策定・実施が決定され，その対象期間は原則として3年以内とすること，同年7月中に骨格を作成すること等の基本方針が定められた．

b. 「政府・与党対外経済対策推進本部」の設置

昭和60年4月19日，上記の「対外経済対策」を推進し，アクション・プログラムの策定・実施を行い，また，その他対外経済問題に関連する重要事項を推進するため，政府・与党首脳会議申合せにより，政府・与党対外経済対策推進本部(本部長：内閣総理大臣．全閣僚及び与党幹部を構成員とする．)が設置された．

c. 「市場アクセス改善のためのアクション・プログラムの骨格」の決定

昭和60年7月30日，同推進本部は，「市場アクセス改善のためのアクション・プログラムの骨格」を決定した．

その総論においては，アクション・プログラムの目標は我が国の市場が国際水準を上回る開放度を達成することにあるとされており，同プログラムの実施においては，同推進本部が強力なフォローアップを行い，その実効性を確保することとされた．

各論は6分野（(a)関税，(b)輸入制限，(c)基準・認証，輸入プロセス，(d)政府調達，(e)金融・資本市場，(f)サービス・輸入促進等）からなっており，政府調達はその1分野を構成していた．

d. 「アクション･プログラム実行推進委員会」の設置(資料I-2)

cに併せ，同日の同推進本部決定により，アクション・プログラム実行推進委員会が設置された．

e. フォローアップ継続の決定

昭和63（1988）年8月4日の第12回アクション・プログラム実行推進委員会において，アクション・プログラムによって各分野ごとに定められていた諸措置がほぼ完全に実施されたことを確認するとともに，基準・認証，輸入プロセス及び政府調達分野等に関しては引き続きフォローアップを行うこととし，このため，当分の間，同委員会を存続させることとした．以来，政府調達の分野については，内外無差別・透明・公正かつ開放的な競争の原則に基づく調達手続の確保を図るための種々の自主的措置を講じている．

f. 「政府調達に関する申合せ」決定

平成3（1991）年11月19日の第16回アクション・プログラム実行推進委員会において，我が国の市場開放努力の一環として，対象となる特定調達基準額の引下げ（政府調達

に関する協定上の義務である13万SDR（平成18年4月以降20年3月まで2,000万円）から10万SDR（同1,600万円）へ），大型政府調達予定案件の年度当初における官報公告，入札公告（公示）から落札までの期間の延長（協定上の義務である40日間から原則50日間へ），適用調達機関の拡大等の措置を我が国が自主的に実施することを内容とする「政府調達に関する申合せ」を行い，平成4（1992）年4月1日からこれらの措置を施行することとした．

g. 「アクション・プログラム実行推進委員会」を内閣に設置

平成5（1993）年8月の政権与党の交代に伴い，同月13日の閣議決定により，それまで政府・与党対外経済対策推進本部に設置されていたアクション・プログラム実行推進委員会の機能を引き継ぐものとして，同名のアクション・プログラム実行推進委員会が，当分の間，内閣に設置されることとなった．

(2)「政府調達に関するアクション・プログラム」の段階

a. 「政府調達に関するアクション･プログラム」の決定

平成6（1994）年2月3日に開催された第20回アクション・プログラム実行推進委員会において，政府調達の手続の抜本的改善等について，透明性，公正性及び競争性をより高める必要があるとの内外の要請に基づき，調達手続の抜本的改善，政府調達情報の公表手段の改善，政府調達情報の提供改善，苦情処理体制・手続の整備等を内容とする「政府調達に関するアクション・プログラム」が決定された．

b. 「物品に係る政府調達手続について(運用指針)」決定

平成6（1994）年3月28日，第21回アクション・プログラム実行推進委員会において，「政府調達に関するアクション・プログラム」に基づき，我が国政府として，政府調達における供給者の利便の向上，競争力のある内外の供給者の市場参入機会の拡大及び手続の透明性の徹底を図るガイドラインとして，「物品に係る政府調達手続について（運用指針）」を決定した．

c. 「政府調達(サービス分野)に関する申合せ」の決定

平成7（1995）年12月11日，第25回アクション・プログラム実行推進委員会において，「政府調達（サービス分野）に関する申合せ」を決定した．これは，平成8（1996）年1月1日，「政府調達に関する協定」(WTO政府調達協定)が発効し，GATTの下での旧政府調達協定(昭和56年発効)においては対象とされていなかったサービス分野についても対象として追加されたことから，「物品に係る政府調達手続について（運用指針）」において，その対象範囲を新協定で我が国がオファーしているサービス分野にまで拡大したものである．

3 個別分野毎の自主的措置について

特定分野の調達手続については，内外無差別・透明・公正かつ開放的な競争原則に基づく手続による調達をより一層推進するため，アクション・プログラム実行推進委員会において，物品一般に係る自主的措置の他，個別分野毎の自主的措置を定めている．（自主的措置の対象となる機関については，資料I-5参照）

個別分野毎の自主的措置一覧

- 「スーパーコンピューター導入手続（改正）」(平成2年4月19日AP決定)」
- 「非研究開発衛星の調達手続」(平成2年6月14日AP決定)
- 「日本の公共部門のコンピューター製品及びサービスの調達に関する措置」(平成4年1月20日AP決定)
- 「日本の公共部門における電気通信機器及びサービスの調達に関する措置」(平成6年3月28日AP決定)
- 「日本の公共部門における医療技術製品及びサービスの調達に関する措置」(平成6年3月28日AP決定)

(1) スーパーコンピューター

a. 経緯

スーパーコンピューターの調達については，昭和62年7月に開催された第10回アクション・プログラム実行推進委員会において決定された「スーパーコンピューター導入手続」に基づき実施されていたが，米国政府が同手続実施後も米国製スーパーコンピューターの我が国政府機関への納入実績がないことを問題視し，特に仕様の策定及び大幅値引き慣行の改善を指摘したことを契機として，平成2年4月19日に同手続の改正を第13回アクション・プログラム実行推進委員会において決定した．本改正手続の対象機関は，WTO政府調達協定の附属書Ⅰの付表1及び3に掲げられた機関(平成19年2月1日現在152機関)である．また，各省庁は，スーパーコンピューターを導入しようとする所管の特殊法人に対し，本手続の趣旨に則った導入手続をとるよう指導することとなっている．なお，本改正手続は，平成2年5月1日より適用されている．

b. 改正された導入手続の内容

改正された導入手続は，各調達機関がその導入目的に最も合致したスーパーコンピューターを導入できるよう定められたものであり，透明，公開かつ無差別な競争的手続を設けている．また，独占禁止法が定める不当廉売禁止に違反する入札に基いてスーパーコンピューターを調達することは，政府の政策に反する旨規定している．

（注）スーパーコンピューターの範囲

本導入手続が適用されるコンピューターの範囲（スーパーコンピューターの演算性能に関する基準値）は，当初「300MFLOPS以上」の理論的最高性能を有するスーパーコンピューターとされていたが，平成7年4月1日以降「5GFLOPS以上」，平成11年5月1日以降「50GFLOPS以上」，平成12年5月1日以降「100GFLOPS以上」，平成17年5月1日以降は「1.5TFLOPS以上」に引き上げられている．

(2) 省略

(3) コンピューター製品及びサービス

a. 経緯

公共部門におけるコンピューター製品及びコンピューターサービスの調達において，無差別待遇，透明性及び公正でかつ開かれた競争という原則に立脚した取引機会を拡大するために，平成4年1月20日の第17回アクション・プログラム実行推進委員会において「日本の公共部門のコンピューター製品及びサービスの調達に関する措置」が決定された．本措置は，平成2年に，米国政府より，我が国の公共部門によるコンピューター調達における外国製品の割合が恒常的に低く，民間分野における割合との間に乖離があることは，政府調達手続上の問題があるためと指摘されたことを契機として策定されたものであり，競争力のある外国系コンピューター製品及びサービスの調達拡大という目的を持ちつつ実施されることとなった．

b. 措置の内容

本措置の対象となる機関は，WTO政府調達協定の対象機関に加え，独立行政法人宇宙航空研究開発機構，商工組合中央金庫，関西国際空港株式会社，日本船舶振興会，日本放送協会及び日本勤労者住宅協会となっており(平成19年2月1日現在158機関)，これらの機関による，基準額10万SDR（平成18年4月以降20年3月まで1,600万円）を超える全ての特定調達契約（スーパーコンピューター導入手続の対象を除く．）が対象となっている．本措置は，製品の調達については平成4年4月1日より，サービスの調達については平成4年10月1日より（一部の機関については，平成5年4月1日までに適用対象となった．）適用されている．

なお，本措置上，評価方式は個々の調達機関の選択によることとされていたが，平成6年3月29日の「対外経済改革要綱」において，「コンピューター…について，平成6年度末を目途に総合評価落札方式を活用する際の評価基準を作成し，総合評価による調達を導入することとする．」とされたことを踏まえ，総合評価落札方式の導入に向けて準備が進められた．平成7年3月27日の第24回アクション・プログラム実行推進委員会において，平成7年7月1日以降，80万SDRを超える全ての調達に総合評価落札方式を適用することを決定するとともに，同方式導入のため，調達機関の事務処理効率化のための評価項目等を含む手引き書として，平成7年3月28日に，「総合評価落札方式の標準ガイド」を関係省庁の申合せにより作成・公表した．

(4) 省略

4 その他

(1) 政府調達に係る苦情処理制度について

a. 旧政府調達協定下の体制

GATT政府調達協定(昭和56年に発効)では，内外無差別，内国民待遇等の方針の下，種々の規定が定められていたが，苦情処理については全く規定されていなかった．他方，アクション･プログラム実行推進委員会で決定した各自主的措置では，苦情処理体制に係る規定が設けられており，これらの規定に基づき苦情処理の手続が実施されていた．

b. WTO政府調達協定下の体制(資料I-6)

ウルグアイラウンドにおいて締結されたWTO政府調達協定(平成8年に発効)では，旧協定にあった内外無差別，内国民待遇等の方針に基づく規定の他，苦情処理に関し，「第20条苦情申立ての手続」が定められることとなった．これを受け，日本政府は平成7年12月1日の閣議決定「政府調達苦情処理推進本部の設置について」により，総理府に政府調達苦情処理推進本部を設置し，同本部において政府調達苦情検討委員会を開催し苦情を処理することが決定され，アクション・プログラム実行推進委員会で決定していた各自主的措置に基づく苦情処理手続は新体制に引継がれることとなった．なお，平成13年1月6日の中央省庁再編に伴い，同本部は内閣府に設置されることとなった．

c. 省略

スーパーコンピューター導入手続の改正について

第13回アクション・プログラム実行推進委員会決定

平成2年4月119日
ｱｸｼｮﾝ･ﾌﾟﾛｸﾞﾗﾑ実行推進委員会

昭和62年7月16日，当実行推進委員会において，「スーパーコンピューター導入手続」を決定し，同年8月1日よりその着実な実施に努めてきたところであるが，これまでの同手続の実施状況等を踏まえ，スーパーコンピューターの導入について，一層透明，開放的かつ無差別な競争的手続を確保するとともに，各対象機関が導入の目的に最も合致したスーパーコンピューターを導入することを一層容易にするため，同手続を別紙の通り改正することとする．

（別紙）

スーパーコンピューター導入手続（改正）

スーパーコンピューターの導入に当たっては，透明，公開かつ無差別な競争的手続を設けるとともに，各機関がその導入目的に最も合致したスーパーコンピューターを導入することを確保することが政府の政策である．以下の手続は，この政策を十分かつ効果的に実施するために定めたものである．ここに定める手続は，スーパーコンピューターの調達をめぐる最近の事情の変化を踏まえたものである．競争的手続を行うことにより，国内外のいかなる企業も，各利用者の情報処理の要求に応える機種を提供する際，意図するとせざるとを問わず，優遇され，阻害され又は拒絶されない．私的独占の禁止及び公正取引の確保に関する法律（以下「独占禁止法」という．）が定める不当廉売禁止に違反する入札に基づいてスーパーコンピューターを調達することは，政府の政策に反

する．本手続は，平成2年5月1日から実施する．なお，本手続の実施にあたっては，政府調達協定（以下「協定」という．）の要件との整合性を確保しつつ行う．

Ⅰ．適用範囲

1． 本手続は，我が国の政府調達協定対象機関によるスーパーコンピューターの導入（購入及び借入れ）を適用対象とする．

2． 各省庁は，スーパーコンピューターを導入しようとする所管の特殊法人に対し，本手続の趣旨に則った導入手続をとるよう指導する．

3． この手続は300MFLOPS以上の理論的最高性能を有するスーパーコンピューターの導入に適用されるが，この対象範囲は必要に応じ見直すこととする．

Ⅱ． 手続

スーパーコンピューター調達に関しては，協定及びアクション・プログラムの手続に従って調達を行う．全ての手続は自由競争の理念に基づき，内国民待遇及び無差別待遇を確保するような方法で実施する．この観点から協定の手続を補完し，その原則を的確に実施するため，スーパーコンピューターの導入を今後計画する機関（ 以下「機関」という．）は，下記の手続により導入を行う．

1．市場調査手続

1.1 資料提供招請

(1) 機関は，スーパーコンピューターの導入を必要と判断する場合には，実際に必要とされる最低限の要求要件（機関が利用形態に基づき予想する作業負荷のパラメーターを含む）を策定する．これらのパラメーターは，機関が必要とするスーパーコンピューターの処理性能を実証する受容可能な最低限のベンチマークテストの結果を含むが，主及び補助記憶容量のような要件も含み得る．一定時間内における一定水準の処理性能が必要な場合には，その水準を要求要件に含めることはできるが，潜在的な供給可能者の参入を排除する形でこれを要求要件に含めてはならない．

(2) 機関は，実際に必要とされる最低限の要求要件の策定を確実に行うため市場調査を行う．機関は可能な限り民間分野において類似の使用形態にあるスーパーコンピューターシステムの取引実例価格に関する情報を収集する．右調査は，透明な形でこれを行い，かつ予定価格の算定及び当該システムの導入に十分な予算の要求の基礎とする．

(3) 機関は，供給者に対し情報提供を要請するに当たり，一部の供給者を優遇する形で情報の提供又は拒絶を行ってはならない．機関は，下記(4) に定める手続による場合を除いて，供給者となる見込みのある者に対して，スーパーコンピューターの導入計画に関する事前の情報を与えない．機関は，必要とされる最低限の要求要件の策定に直接関与した供給者を入札手続に参加させない．

(4) 機関は，上記（ 1）で策定された実際に必要とされる最低限の要求要件に従い，スーパーコンピューターの導入計画及びこれに関する供給者からの一般的な参考資料及び基本的な要求要件に関するコメント（仕様，技術資料等を含む．以下同じ．）の提供招請につき官報による公表（ 以下「公表」という．）を行うとともに，これを補完するものとして機関が承知する内外の供給者（スーパーコンピューターの供給に関心を表明した者を含む）に対し同様な内容の資料提供招請状（ 以下「招請状」という．）を送付する．公表に基づき応募する供給者と招請状に基づき応募する供給者は，同等に取り扱う．

(5) 公表及び招請状の送付は，上記資料及びコメント提供の受付期限の前日から起算して少なくとも40日前までに行う．

(6) 公表及び招請状には，次の事項を記載する．

（イ） スーパーコンピューターの導入計画及び実際に必要とされる最低限の要求要件

（ロ） 資料及びコメント提供の受付期限

（ハ） 公表又は招請状に基づき応募する供給者から要求があった場合には，導入説明書を送付する旨の注記並びに導入説明書の入手先及び期間

（ニ） 導入説明会を開催する場合にはその旨の注記

1.2 導入説明書

(1) 機関は，公表又は招請状に基づき応募する供給者に対して導入説明書を交付する．

(2) 導入説明書には，少なくとも次の事項を記載する．

（イ） 資料の提供先（担当窓口）

（ロ） 追加情報の照会先

（ハ） 資料提供の受付期限

（ニ） 導入を計画しているスーパーコンピューターに関する詳細な要求要件（求められる性能，運用業務内容等）

（ホ） 機関が想定する代表的な作業負荷を示すベンチマーク・テストに関する資料

（ヘ） 導入説明会を開催する場合にはその日時及び場所

（ト） 各必須要件の評価及び入札書の技術的要件の評価に関する客観的基準

1.3 導入説明会

機関は，必要に応じ導入説明書に関する説明会を開催する．導入説明書に日時，場所が明記されていない場合には，機関は公表又は招請状に基づき応募した全ての供給者に対して情報を検討するための十分な猶予期間を確保しつつ案内状を送付する．

1.4 照会

(1) 機関は，公表，招請状及び導入説明書の内容に関して供給者から照会のあった場合には，速やかに応じることと

する.

(2) 機関は，導入説明書に関する修正又は追加の情報がある場合には，当該情報を公表又は招請状に基づき応募する全ての関係供給者に同時に提供するとともに供給者がその情報を検討し，これに対応することができるように，追加資料提供のために十分な猶予期間を設ける.

(3) 機関は，提供された資料に関し，提供者に対し質問・照会を行うことができるが，一部の供給者を優遇する形で質問・照会を行ってはならない．また，機関は，必要な場合には，提供された資料に関し，性能及び機能に関する確認等を含む調査を行うことができる.

(4) 機関は，供給者から提供を受けた資料及び情報を当該供給者の同意を得ずに公表し，第三者に開示しない.

1.5 ベンチマーク・テスト

(1) 定義 この手続においてベンチマーク・テストとは，使用者が指定する代表的な作業負荷（一連のコード）に基づいてスーパーコンピューターの達成性能を計測することをいい，計測に当たっては米国と同様ウォール・クロックの経過時間を用いる.

(2) ベンチマーク・テストはいずれの調達においても下記に従って行われる.

（イ） 機関は，上記Ⅱ 1.1(1)の「資料提供招請」に従い，実際に必要とされる最低限の要求要件に合致するスーパーコンピューターの処理性能を特定する.

（ロ） 機関は，上記Ⅱ 1.2(ホ)の「導入説明書」の項に従いベンチマーク･テストに必要な書類を全て提供する.

（ハ） ベンチマーク・テストの選択に当たっては，下記Ⅱ 2.1(2)の「仕様書作成」の項に従い，機関が想定する作業負荷の中で代表的なものを選ぶ.

（ニ） 下記Ⅱ 3.6 の「技術審査」の手順に従って行われたベンチマーク・テストの結果は入札書に加え機関に提出されなくてはならない.

（ホ） 機関は，提出された試験結果を実証するため入札書提出後にベンチマーク・テストを実施することができる.

２．仕様書の作成

2.1 仕様書の作成

(1) 機関は，市場調査段階で策定した実際に必要とされる最低限の要求要件に基づき，仕様書を作成する．当該調達が現有システムを置換するもの又はこれと相互接続するものである場合には，仕様書は供給者が現有システム供給者と有効に競争し得るように作成されなければならない．必要な処理業務を実行する観点から必須でないような機能は要求してはならない.

(2) 機関が想定する代表的な作業負荷に基づくベンチマーク・テストは，機能的性能がスーパーコンピューターの性能評価の基礎として用いられるよう仕様書において最終的に定められる．機関は，供給者にベンチマーク・テストの準備及び実行に当たり必要とする全ての文書を提供する．機関が求めるシステムにとって必要なオペレーティング・システム及びその他のソフトウェアは，仕様書に明記されなければならない.

(3) 供給者が機関の実際の要求要件を満たすことに専心できるようにするため，機器仕様よりもシステム全体の性能が重視される．機関が求める機種の能力についての目安を示すため，Ⅱ1.1(1)にいう処理性能を仕様書に含めることができる.

(4) 仕様書には，潜在的な供給者が機関の実際に必要とする最低限の要求要件を理解するために必要な全ての情報が含まれていなくてはならない．機関は，仕様書の作成に直接携わった供給者を入札手続に参加させない.

2.2 仕様書の説明

(1) 機関は，上記Ⅱ2.1 の仕様書を作成したときは，公表又は招請状に基づき応募した全ての供給者に対して，案内状を送付し，当該仕様書に関する説明を行う．また，機関は，互換性を要件として随意契約を行う方針を固めた場合には，協定上の根拠を合せて説明する.

(2) 機関は，現有のハードウェア又はソフトウェアとの相互運用性を要求する場合には，供給者が自ら応ずる場合又は権限に基づく場合を除き独自のオペレーティング・システム，インターフェース又はプロトコルの開示を求めてはならない.

2.3 照会及び提案

機関は，Ⅱ2.2 の説明の後，少なくとも５０日以上の期間を設けて，機関及び供給者が当該仕様書に関して照会を行い，また供給者が当該仕様書に関する提案の申し出及び提案の修正を行う機会を与える.

３．入札手続

スーパーコンピューターの調達に関しては，調達機関は下記の手続に従う.

3.1 基本手続

(1) 購入

協定及びアクション・プログラムの手続に従って，調達を行う.

(2) 借入れ

借入れを対象とする協定の規定に従って調達を行う.

3.2 入札期限及び情報提供

(1) 調達機関は，競争入札によりスーパーコンピューターを調達する場合には，入札書が受領される期間を少なくとも入札公告後40日以上確保する.

(2) 調達機関は，互換性を要件として随意契約によりスーパーコンピューターを調達する場合には，上記Ⅱ 2.3 の期間経過後，契約締結の少なくとも４０日前に当該調達計

画を互換性の要件とともに官報により情報提供する．機関は，当該情報提供に基づき供給者より照会がある場合には，速やかに関連情報を提供する．

3.3 仕様書の確定

調達機関は，上記Ⅱ 1．4 及び2．3 に従い，仕様書に関する照会又は提案が行われ，全ての供給者に対して変更点が通知された後，仕様を確定し，入札に参加を希望する供給者に提供する．

3.4 入札手続

(1) 入札においては，価格に加え技術的性能，機能的性能の要因をも考慮して，調達機関にとって総合的に最も有利なものが評価される．評価のための基準は下記Ⅱ 3.7 に定めたとおりとする．

(2) 調達機関はスーパーコンピューター導入のための予定価格を決めるに当たっては，可能な限り，民間分野での類似の使用形態にあるスーパーコンピューターシステムの取引実例価格を基礎とする．

(3) 一又は複数の入札が予定価格の範囲内で，かつ，調達機関が策定した最低限の機能的性能を満たす場合には，再度入札を行ってはならない．機能的性能の要件は上記Ⅱ 2.1 に従って定められる．予定価格は，適切な予算に基いて上記Ⅱ 3.4(2)に従って決定される．

(4) 調達機関は，ただ一人の供給者が入札した場合であって，その入札者が上記Ⅱ2.1 に従って策定された機能的性能要件を満たし，かつ，上記Ⅱ 3.4 (2) に従って決定された予定価格の範囲内で入札する場合には，再度入札は求めない．

3.5 入札説明会

調達機関は，必要に応じて，最終仕様書及び要求要件に関して説明会を開催する．この場合，入札に参加を希望する全ての供給者に対して同一の情報が提供される．

3.6 技術審査

(1) ベンチマーキング 調達機関は，事前に決定され，かつ，内容が特定された，代表的な作業負荷を用いたベンチマーク・テストを行う．技術評価に当たり調達機関は，仕様書に示されたベンチマークテストのみを行う．ベンチマークの選択は，上記Ⅱ 1.1 に従い当該機関が想定する作業負荷に基づいて行う．調達機関は，ベンチマーク・テストを行う場合には，各入札者に対し適切な事前通告を行う．テストで用いられる基準は，上記Ⅱ 2 で特定されたものに限られる．調達機関は，一部の供給者のみが有利となる条件でベンチマーク・テストを行ってはならない．

(2) ベンチマーク・テストは，下記の条件を満たす場合を除き，全ての調達に関し，実存するシステムを使用してこれを行う．

（イ） 供給者が未だベンチマーク・テストを行い得ない新型の第 1 号機を提示する場合．

（ロ） 上記(イ) の要件を満たす供給者がいる場合には，他の供給者も当該調達において未だベンチマーク・テストを行い得ない新型機を提示することができる．

（ハ） 落札した当該供給者は，機種を指定する納期までに納入しなければならない．もし，当該供給者が納入を行うことができない場合には，調達全体は再度公告入札に付されるものとする．

（ニ） 落札したシステムは，納入前にベンチマーク・テストを行い，予測性能値と同等又はそれ以上の結果を示すと共に仕様を満たさなくてはならない．

(3) 調達機関は，入札者から照会があるときは，システムが納入された後，予測性能値とベンチマーク・テスト結果の双方を明らかにする．

(4) 入札者はベンチマーク・テストの実施方法及び結果に関連してⅢに定める調達審査委員会に対して苦情を申し立てることができる．

3.7 評価基準

(1) 入札の総合的評価は，全ての入札が公平に取り扱われ，かつ透明性を確保する形で行わなければならない．入札の総合的評価においては，調達機関は，システム全体の性能を重視しながら性能及び価格を考慮する．供給者からの入札書のうち，Ⅱ ． 3.4(2)に従って決定される予定価格の範囲内であり，かつ，上記Ⅱ3.3 にいう仕様書に定める必須の各要求要件を満たしているものを評価の対象とする．

(2) 調達機関は，必須の要件とそれ以外の要件を特定する．必須の各要件についての入札の評価基準は合格又は不合格という形で示し，Ⅱ．1.2(2)(ト）で定める導入説明書及びⅡ．3.3 に定める最終仕様書に定める．全ての必須要件を満たす入札のみが更に審査される．

(3) 調達機関は，入札の技術的要件についての客観的な評価基準をⅡ．1.2(2)（ ト）の導入説明書及びⅡ．3.3 の最終仕様書において定める．この客観的評価基準は，技術的要件ごとに得点方式で示すこととし，必須の要件に付いては，当該要件を超える部分に対して評価する．必須でない要件に対して特別の評価を与える場合には，上記客観的評価基準に基づき評価する．最終仕様書に書かれていない機能は評価の対象とはしない．入札者は，照会の段階で評価基準についてその変更を提案できる．入札の技術的要件に関する総合的評価は，各要件に対する得点を総計して行う．

(4) 利用可能なソフトウェアについても考慮の対象とする．

(5) 調達機関は，入札の技術的要件と入札価格を評価し，最も有利な入札を行った者を契約の相手方とする．

3.8 最終契約価格

最終契約価格は，Ⅱ．3.7 に定める評価基準に従って決定される．

3.9 不公平な入札

(1) 不当廉売の禁止を含む独占禁止法の規定に違反する入札に基づきスーパーコンピューターを調達することは政府の政策に反する．

(2) 価格又はその他の条件において不法に公平な競争を阻害する入札が行われた場合には，この入札は，無効と見なされ，調達機関はこれを当該スーパーコンピューター調達において落札の対象としてはならない．

(3) 上記(2) に当該する入札を行った者は，原則として，当該スーパーコンピューター調達の再度入札に参加する資格がないと見なされ，かつ，当該入札者の氏名は公表することとする．

(4) 落札決定後，Ⅲに定める手続きに従って苦情が申し立てられ，公正取引委員会又は裁判所が当該入札が不法に公正競争を阻害したと認定する場合には，調達機関はⅢ．4.4 に定める最も適切な措置をとる．

４．落札に関する情報及び調達結果の説明

調達機関は，協定第６条に準じて，落札に関する情報を公表するほか，資料提供を行い，選定されなかった供給者より照会があった場合には，その供給者が選定されなかった理由に関する適切な情報（選定された機種及び当該機種の相対的利点に関する情報を含む．）を速やかに提供する．但し，特定の供給者の正当な商業上の利益を害することとなる情報を除く．

Ⅲ．苦情処理機構

１． 目的及び実施時期

スーパーコンピューターの調達に当たっては，公正，かつ，開かれた競争及びこの手続との整合性を確保するために，次の苦情処理手続がこの手続の実施の日の約30日後から実施される．

２． 調達審査委員会

2.1 この手続に基づくスーパーコンピューターの調達に関する潜在的な供給可能者からの苦情を審査するための中立の調達審査委員会（ 以下「委員会」という．） が組織される．委員会は，審査の対象となるスーパーコンピューターの調達に関して実質的な利害関係を持つものであってはならない．

2.2 委員会は，苦情を文書で受理し，機関によるスーパーコンピューターの調達に関するいかなる事項に関しても事実関係を調査し，提案を行う．

2.3 委員会は，公的分野の調達に関する有識者で構成する．苦情に関する審査に当たり利害関係を有する委員は参加できない．

３． 調達審査手続

3.1 潜在的な供給可能者は，この手続の精神又は条項に反する形で調達が行われたと判断する場合には，委員会に対し，苦情を申し立てることができる．また，潜在的な供給可能者は，独占禁止法に違反する入札を行った者が落札したとの判断する場合も苦情を申し立てることができる．潜在的な供給可能者が，本手続の違反があると考える場合には，まず当該調達を行った機関との間で解決を求めることが奨励される．

3.2 苦情申し立ての時期

(1) 苦情は，調達手続のいずれの段階であっても申し立てることができるが，苦情の要因が判明した時又は判明し得る状態になった後10日以内に申し立てなければならない．潜在的な供給者は，委員会に苦情を申し立てた後１日以内にその写を調達機関に提出する．

(2) 委員会は，適時に申し立てられなかった苦情であっても正当な理由があるもの又は本手続の目的上重要な意味を持つものであればこれを受理できる．

3.3 委員会は申し立て後７日以内に苦情を審査し，次の各号に該当する場合には，その理由を付して，文書で却下することができる．

(1) 申し立てが適時に行われなかった場合
(2) この手続の対象外の調達の場合
(3) 軽微で無意味な申し立ての場合
(4) 潜在的な供給者からの申し立てではない場合
(5) その他の場合であって，委員会が審査するのが適当でない場合

3.4 委員会は，苦情が正当に申し立てられたと認める場合には，当該調達に関係する全ての潜在的な供給者に対して一日以内に文書で通知する．

3.5 落札又は調達手続の停止

(1) 委員会は，落札に至る前の段階で苦情申し立てを受理したときは，苦情処理に係る期間内は調達手続を停止する旨の要請を当該苦情の申し立て後１０日以内に文書で行う．

(2) 委員会は，落札後に苦情申し立てを受理したときは，苦情処理に係る期間内は契約執行を停止する旨の要請を当該苦情の申し立て後10日以内に文書で行う．

(3) 調達機関は，委員会からの要請を受けたときには，原則として調達手続又は契約執行を停止する．ただし，当該機関の長が，緊急かつやむを得ない状況にあるため委員会の要請に応じることができないと判断し，かつ，その旨を委員会に通知する場合にはこの限りではない．

3.6 調査

(1) 委員会は，申し立て者及び機関による説明，要請及びその他の文書の提出等を通じて，苦情に関する調査を行う．

(2) 委員会は，申し立て者若しくは機関の要請により，又は委員会の判断により，苦情に関する公聴会を開くことができる．

3.7 機関の報告

(1) 調達機関は，苦情の写の送付を受けた後25日以内に，委員会に対し，次の事項を含む苦情に関する完結した文書による報告を提出する．
   （イ） 要求要件に係る文書（苦情に関連する仕様を含む）
   （ロ） その他苦情に関連する文書
   （ハ） 機関の有する全ての事実関係，調達機関の行為及び提案が明記され，かつ，全ての苦情申し立て事項に十分応えている説明文
   （ニ） 苦情を解決する上で必要な追加的事実関係又は情報

(2) 委員会は，上記(１）の報告を受領した後，速やかに関係文書の写を申し立て者に送付するとともに申し立て者に対し，関係文書の受領後７日以内に，委員会に対しその意見を提出するか又は当該文書に基づき決定が行われるべき旨の要望を提出する機会を与る．委員会は，意見を受領した後，速やかにその写を調達機関に送付する．

3.8 参加者

調達機関及び当該調達に直接の経済的利害を有する潜在的な供給可能者は，苦情処理手続に参加できる．

４．事実関係の認定及び提案

4.1 委員会は，苦情が申し立てられた後90日以内に，認定した事実関係と提案に関する報告書を作成する．事実関係の認定において委員会は，苦情の全て又は一部を認めるか又は却下するかを明らかにするとともに，調達の手続又は落札がこの手続の精神又は一部の条項に反して行われたものかどうかを明らかにする．

4.2(1) 不当廉売を禁ずる独占禁止法の規定に違反して入札を行った者が落札した可能性が高いと委員会が認定する場合には，委員会は，当該調達につき公正取引委員会に通報し，独占禁止法違反の有無を認定すること及び適切な措置をとることを要請する．

(2) 委員会は，調達機関に対し，上記の通報に係る行為について公正取引委員会が最終的な結論を出すまでの間，調達機関に対して当該契約の執行を停止するよう要請する．調達機関は委員会からの要請を受けた場合には，原則として契約の執行を停止する．公正取引委員会の通知を受けた後，委員会は，苦情に関する審査を終了するが，公正取引委員会が独占禁止法違反があると認定した場合には，委員会は，当該調達機関に対し，4.4に掲げる措置を取るよう提案する．

4.3 委員会は，事実関係の認定と提案を行うに当たり，調達手続に係る瑕疵の程度，全ての供給可能者に対する取扱いの差異の程度，この手続との整合性及びその有効性の程度，参加者の誠意並びに当該契約がこの手続に関連している程度等，調達と落札に関する全ての事実関係を考慮するものとする．

4.4 委員会は，この手続の精神又は条項に違反するとの認定に至った場合には，次に掲げる一又は複数の適切な是正策を提案する．
（1）新たに入札手続を行う．
（2）入札条件は変えず再度入札を行う．
（3）入札を再審査する．
（4）他の供給者を落札者とする．
（5）契約を破棄する．

4.5 委員会は，報告書の作成後１日以内に事実関係の認定を文書の形で提案するとともに，苦情申し立て者，当該調達機関及び他の潜在的な供給可能者に送付する．認定結果に関し外国関係者からの照会がある場合には，外務省がこれを扱う．

4.6 調達機関が委員会の提案を受け入れない場合には，調達機関は，報告書の作成後１日以内にその決定と理由を委員会に送付する．この機関の決定に関し，外国関係者からの照会がある場合には，外務省がこれを扱う．

4.7 委員会がその審査の過程で法令に違反する行為の証拠を見出した場合には，委員会は，関係当局が適切な措置を取り得るよう当該証拠を関係当局に提出する．

５．迅速審査

5.1 苦情申し立て者又は機関が文書で苦情に対する迅速な処理の要請を行う場合には，委員会は，本項に定める手続（ 以下「迅速審査」という．）に従い，苦情処理を行うことを考慮する．

5.2 委員会は，迅速審査の要請を受領した後2日以内に迅速審査を適用するか否かを決定し，苦情申し立て者及び機関に対しその旨を通知する．

5.3 迅速審査が適用される場合に期限と手続は，次のとおりとする．

(1) 調達機関は，委員会により迅速審査適用の通知を受けた後10日以内に3.7に定める報告書を委員会に提出する．委員会は報告書を受領後，速やかに苦情申し立て者に関係書類を送付する．委員会は，苦情申し立て者に対し，関連書類の受領後5日以内に委員会に意見を提出するか又は当該関係書類に基づき決定が行われるべき旨の要望を提出する機会を与える．委員会は，意見書を受領後速やかにその写を調達機関に送付する．

(2) 委員会は，苦情に関する事実関係認定及び提案を苦情申し立て後45日以内に文書で行う．

日本の公共部門のコンピューター製品及びサービスの調達に関する措置について

第17回アクション・プログラム実行推進委員会決定

平成4年1月20日
アクション・プログラム実行推進委員会

先般行われたコンピューター製品及びサービスの調達に関する合衆国政府との協議の結果を踏まえ，我が国政府としては，公共部門のコンピューター製品及びサービスの調達に関する措置を，別紙のとおり実施することを決定する．

日本の公共部門のコンピューター製品及びサービスの調達に関する措置

Ⅰ．全般的政策

A．公共部門におけるコンピューター製品（注）（周辺機器及びパッケージソフトウェアを含む．）及びコンピューターサービス（コンピューターの運用及びメインテナンス，コンピューターデータ入力，コンピューターシステム開発（ソフトウェアの開発及びシステムインテグレーションを含む．）コンピューターソフトウェアのメインテナンスその他の関連サービス），（以下，「コンピューター製品及びサービス」と総称．）の調達において，無差別待遇，透明性及び公正でかつ開かれた競争という原則に立脚した取引機会を拡大するために，日本政府（以下，「政府」．）は，公共部門の調達手続の一層の改善に積極的に努める．そのために，政府は，競争力のある外国系コンピューター製品及びサービスの調達拡大という目的を持ちつつ，ここに示す「日本の公共部門のコンピューター製品及びサービスの調達に関する措置」（以下，「本件措置」．）を実施する．

B. 政府は関税及び貿易に関する一般協定（以下，「GATT」．）及び政府調達に関する協定（改正を含む．）（以下，「コード」）の義務に対するコミットメントを再確認する．本件措置の実施に当たっては，コード（今後の改正点を含む．）に規定する条件との整合性が確保される．

C．これらの政策を完全かつ効果的に実施するため，本件措置は，10万SDR又はコードの基準額のいずれかの低い方の金額を超えるすべてのコンピューター製品及びサービスに関して附属書Ⅰに示すコード対象機関及び附属書Ⅱに示す追加的機関（以下，「機関」．）の調達を対象とする．スーパーコンピューターの調達は，引き続き1990年の「スーパーコンピューター導入手続」の対象であり，本件措置の対象とはならない．

D．政府は，更に，1985年の「市場アクセス改善のためのアクション・プログラム」で政府調達について示された政策と措置を再確認し，競争力のある外国系コンピューター製品及びサービスの調整の分野において，かかる調達政策を引き続き実施することを確認するとともに，外国系コンピューター製造業者の日本の公共部門市場における販売拡大努力を歓迎する．

（注）コンピューター製品には，製品の供給に付随するサービスの価額が当該製品の価額を超えない場合の当該サービスの調達を含む．

E．実施

本件措置は，附属書Ⅰ，Ⅱ－A，Ⅱ－B及びⅡ－Cに掲載されている機関によるコンピューター製品の調達について1992年4月1日から，附属書Ⅰ，Ⅱ－Aの機関によるコンピューターサービスの調達について1992年10月1日から実施される．日本政府は，コンピューターサービスの調達について，附属書Ⅱ－B及びⅡ－Cに掲載されている機関が，1993年4月1日までに本件措置の対象となるよう措置を講じる．

Ⅱ．政策及び手続き

政府は，ここに公共部門のコンピューター調達に関する既存の政策及び手続を明確化するとともに，新しい政策及び手続きを策定し，実施する．政府は，競争力のある外国系コンピューター製品及びサービスの政府調達の拡大という目的を持ちつつ，無差別待遇，透明性及び自由でかつ開かれた競争機会を十分に確保するために，これらの政策及び手続を実施する．

（招請前段階）

1. 招請前情報が入手可能な場合には，内外のすべての潜在的供給業者に対して当該情報への平等なアクセスが保障されるとともに，かかる招請前段階に参加する機会が等しく与えられる．いかなる潜在的供給業者に対しても，事前情報に係る利点を与えられない．

2.調達機関は，調達が計算されるコンピューター製品及びサービスの技術，予算，仕様，機能その他の側面について話し合われる技術委員会，諮問グループ，研究会その他同様の会合が設置される場合には，全ての潜在的供給業者に平等に参加する機会を与えることを確保する．

3.招請前段階で提供される情報は，特定の潜在的供給業者を排除したり，事前に適格とするために用いられてはならない．

（仕様）

4.仕様は中立的な方法で策定される．調達が既存システムの代替又は既存システムとの連接のために行われる場合の仕様は，競争を排除するように策定されてはならない．業務目的のために不可欠でない内容は要求されない．

5.最終的な調達仕様作成に直接関与した供給業者は，関与したことによって競争上の不公正な利点を享受する場合には，入札過程に参加することを認められない．但し，調達機関が

仕様の準備又は仕上げの過程を管理し，公正かつ無差別に進めているという状況の中で潜在的供給業者が調達機関に情報若しくは支援を提供する場合及び供給業者が調達機関の要請に応じて，自らの製品に関する仕様若しくはデータを提供する場合は，例外とする．このような場合，すべての潜在的供給業者に，参加する機会又は製品に関する仕様若しくはデータを提供する機会が与えられる．

6.政府は，機関の調達担当官の仕様書作成の努力に関連する情報提供及び研修を統括し促進するプログラムを策定する．

（説明会）
7. 機関は，必要に応じ，コンピューター製品及びサービスの調達に関する説明会を開催する．これには，潜在的供給業者と調達機関とが技術面及び管理面に関して直接やりとりを行う機会が含まれる．

（入札及び応札手続き）
8. すべての潜在的供給業者に対し，調達機関の要求に対応するための公正かつ平等な機会が入札及び応札の過程において，与えられる．

9.競争的調達が政府調達に係る政策及び慣行の基礎となっていることから，随意契約及び単一入札はコード手続によって認められる例外的な場合に限り用いられ，国内のコンピューター製品及びサービス供給業者を優遇するようには用いられない．機関は随意契約の利用を縮減する．

10.入札説明書及び評価基準は，すべての潜在的供給業者に平等な機会が無差別に提供されることが確保されるよう，公平に作成される．

11.指名入札を含む入札制度は，国内のコンピューター製品及びサービス供給業者を優遇するように用いられない．調達機関は，無差別な方法でのみ，調達に入札する供給業者の数を制限することができる．

（入札の評価）
12.入札の評価は，全ての入札者に対する平等な取扱いが確保されるよう，透明性のある方法によって行われる．

13.入札の過程において，技術評価及びシステム性能評価が適用される場合における当該評価は，すべての潜在的供給業者に対して同一の条件の下で実施される．如何なる検査基準についてもすべての潜在的供給業者に対して同一のものを用いる．

14.全ての評価項目は，入札説明に明記される．入札の評価はコードと整合した手続に従って行われ，以下の手続を含み得る．個々の調達機関は調達の目的と性格に応じて，入札手続を選択する．
（a）入札は，仕様に示された特定の技術及び他の評価基準を満たすか否かが評価され，標準基準を満たすもののうちで最低価格の応札を行った者が落札する．
又は，
（b）評価基準を満たすとともに，技術・機能及び価格／コストの要件に照らして最適の入札を行った供給業者が落札する．必要な場合には，入札説明書に明記された評価基準に相対的加重が適用される．価格／コスト評価は，調達の全ライフサイクルコストに基づいて行うことができる．

（落札に関する情報）
15.最終選定が行われた後，調達機関は，落札に関する情報を公表し，落札しなかった供給業者からの要請がある場合には，落札しなかった理由について，落札したシステムの名称と相対的利点の情報を含む関連情報をその供給業者に対して早急に提供する．但し，特定の供給業者の正当な商業上の利益や供給業者間の公正な競争を阻害するような情報はこの限りでない．

（将来の計画に関する情報）
16.予算要求に関してある潜在的供給業者にとって利用可能とされた情報は，無差別に利用可能とされる．調達機関は，80万SDRを超える金額のコンピューター製品及びサービスの導入計画を，年度の可能な限り早い時期に官報で告示し，潜在的供給業者が右計画に関し文書及びコメントを提出できるよう一般的な招請を行う．

（機関毎の計画）
17.本件措置に従って，各調達機関が本措置によって示された政策と手続を実施するために行っている努力あるいは将来行う努力を示す調達機関毎の計画を策定することが勧奨される．右計画は，毎年度毎に改定されることが勧奨される．

（入札苦情申立て制度）
18.本件措置の対象となるコンピューター製品及びサービスの潜在的供給業者に対して，平等，適時，透明かつ効果的な入札苦情手続を提供するため，付属書Ⅲ(略)に掲載された公平な苦情処理制度が維持される．

（地方公共団体）
19.政府は，地方公共団体に本件措置を通報し，本件措置と整合した完全に競争的な調達政策及び手続の趣旨に則った協力を要請する．

（マルチベンダ・オープン・システム）
20.各省庁間の組織がマルチベンダ・オープン・システムのための環境を促進す
る作業を行うために設立される．内外のコンピューター企業に対し，マルチベンダ・オープン・システムの環境整備の支

援を行うために公正，無差別に招請が行われる．
Ⅲ．不公正な入札
　不当廉売の禁止を含む独占禁止法規定に整合的な入札に基づいてコンピューター製品及びサービスの調達を行うことが政府の政策であることに鑑み，調達機関は，反競争的慣行に対処する適切な措置を講ずる．

A．価格又はその他の点に関し，公正な競争を不法に阻害する入札が行われた場合には，この入札全体が無効とみなされ，調達機関は，落札に当たって当該入札を考慮の対象としてはならない．
B．前記Ⅲ．A．に言及される入札を行った者は，原則として，当該コンピューター製品及びサービスの調達に再度入札する資格はないものとみなされ，右入札者の氏名が公表される．

C．調達機関が，その調達（調達仕様書の作成を含む．）に関連し，不当に公正な競争を阻害する慣行の存在を示すような情報を得た場合は，当該調達機関は，公正取引委員会が適切と判断する措置を発動することができるよう，かかる情報を適時に同委員会に対し提供する．

D．前記の目的のために，調達機関は，公正取引委員会との間で，独占禁止法違反の可能性のある行為に関する情報の発見及び交換の手続を容易にするための連絡担当者を指名する．
附属書Ⅰ

（WTO政府調達協定対象機関）
省略

以上

付録 F　新中央可視化システム（CeViS）の導入とその概要

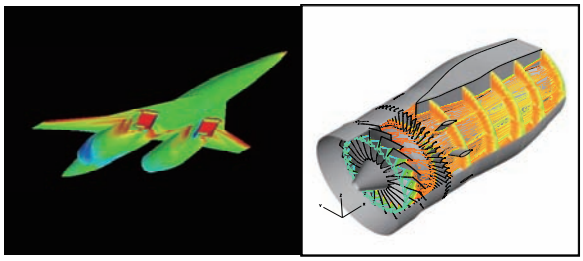
F.1　はじめに

航技研は，高速スーパーコンピュータを用いて行った主に計算流体力学（CFD）の大規模数値シミュレーションの結果を，迅速かつ効果的に可視化する研究並びに技術開発に従来より取り組んで来たが，2000 年度，それまで使っていた CRAY を中核とする可視化システムが陳腐化したのを契機に，大画面表示装置と高速可視化サーバから成る新中央可視化システムを導入し，2001 年 4 月より本格稼働に供した．

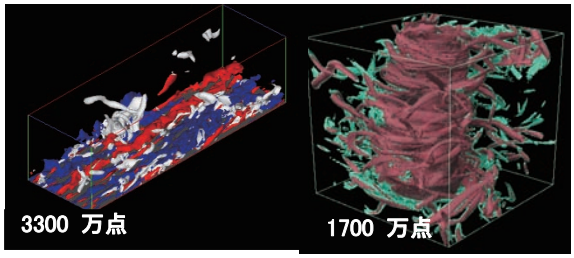
ここでは，その中央可視化システムのねらい，構想，技術，特徴，課題などについて概説し，可視化技術の可能性，問題などについて言及する．

F.2　航技研における可視化を取り巻く状況

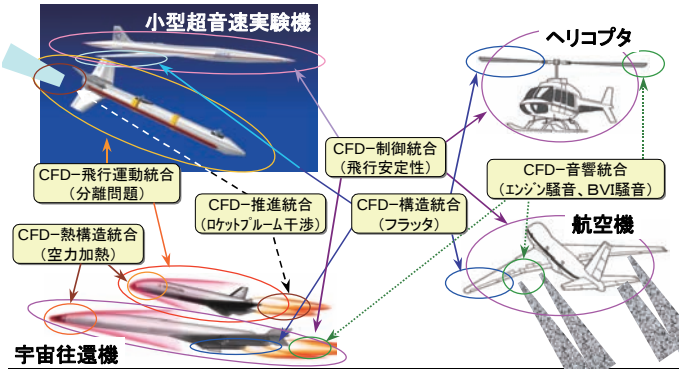
航技研では，ピーク処理性能 280GFLOPS，メモリ 44.5GB を有するスーパーコンピュータ（数値風洞：NWT）を用いた大規模 CFD 解析を先駆的に実施し，流体基礎現象の解明や航空宇宙の研究開発に適用して来た．近年の傾向は，工学系の解析では，設計開発への高度適用に係る実形状を見据えた複雑形状への対応が進んでおり，マルチブロック構造格子や非構造格子を用いることにより，エンジンナセルやフラップのついた航空機機体まわりの流れ解析やエンジン内部の複数段の流れ解析が可能になって来ている（**図 F.1**）．また，時間とともに流れが変化したり，物体が移動する非定常（過渡的）問題や，流体－構造などの多分野連成問題が扱われるようになって来ている．一方，学術系の解析では，マルチスケール・マルチフィジクスといった現実の現象をできるだけ正確に取り扱う High-Fidelity（高忠実度）解析の方向へと進展しており，乱流や燃焼流のシミュレーション（**図 F.2**）では，数 1000 万点のメッシュ，出力データは量にして時に 100GB を超えるようなケースも現出している．



図F.1　工学CFD解析の例



図F.2　学術CFD解析の例



図F.3　多分野統合シミュレーションの適用例

これに伴って，可視化についても，設計開発において必要とされる可視化技術と学術研究に必要とされる可視化技術は自ずと異なるものとなって来ており，適材適所の可視化技術が求められるようになっている．

一方，航技研では，1999年，NAL計算科学ビジョン（**付録A**参照）を策定し，今後のCFD研究開発の方向性を示すとともに，独立行政法人化を契機に中期研究計画を作成し，CFDを中核とする多分野統合シミュレーション技術に関する研究開発を行っていくこととした[1]．すなわち，「CFD（流体）を中核として異種分野との連成（流体－構造，流体－制御など）シミュレーションを統合的に実施可能なソフトウェア及び計算環境基盤を整備する．例えば**図F.3**に示したプロジェクトや機体・エンジン開発における技術課題が解けるようにコード，プラットフォーム，ミドルウェア等のソフトウェアを整備する．併せて，プログラムやライブラリの標準化・共用化を推進するとともに，コード検証の技術確立を行い，高品質のプログラムを開発し提供する．また，ボトルネックとなっている技術課題の克服のために，乱流／燃焼のモデリング，複雑格子生成，並列プログラミング環境などの課題に重点的に取り組む」とし，その推進のために鍵を握る要素の一つとして，可視化を含む統合的なユーザフレンドリーな計算環境の整備を挙げている．

また，科学技術会議 25 号答申「未来を拓く情報科学技術の戦略的な推進方策の在り方」（1997 年 6 月）に基づいて，最先端フロンティアの開拓を目指した重点領域が設定された（1997 年 7 月，**付録 A** 参照）．統合シミュレーション技術，可視化技術，並列分散ソフトウェア技術，アーキテクチャ技術が重点項目として選定されており，可視化などの技術の重要性が国としても認識され始めているところである．

このように状況が変わって行く中で，航技研では，可視化処理専用のサーバとして，CRAY Y-MP M92（700MIPS, 8GB メモリ）から成る可視化システムを運用して来たが，経年による性能の陳腐化，ジョブが終わるまで計算結果を扱えない[40]などの処理の非効率性が顕在化していた．また，従来の21インチ程度のディスプレイを有するデスクサイドやデスクトップ型のグラフィックスワークステーションでは，大規模デ

[40] ストレージシステムの問題．

ータの可視化に対して，メモリが足りない，解像度が足りない，使い勝手が悪いなどの問題点や限界が指摘され，新たな可視化システムに対する需要や期待が高まりつつあった．

一方，可視化手法の研究や開発という側面からは，近年，技術的にはかなり成熟して来た感がある．線画や面塗り画，パーティクル追跡などの通常良く使う基本的な機能のほとんどは汎用可視化ソフトにすでに取り込まれており，GUIなどの充実ぶりを考えると，現時点で可視化ソフト/ツールを一から自作する動機は乏しい．また，特殊なソフトの利用は，使い勝手やメンテナンスに問題が生じさせる．

可視化の利用という側面から見てみると，工学CFD解析における可視化では，工学レベルで見られれば良い，あるいは，必要な部分を必要な精度で迅速に可視化することが利用者の興味の中心であり，特に新しい手法とか見方へのニーズはあまり強くないように見受けられる．この場合の最大の課題は，普通の可視化を如何に高速かつ柔軟に行えるか，ということであり，逆に言えば，設計や開発に役に立たない可視化は必要とされない．具体的には，マルチブロック，非構造といった各種データ構造への対応や，パラメータスタディによる多数のデータセットの効率良い可視化，設計開発に役立つ新たな使い方の模索などが課題となっている．

他方，学術CFD解析における可視化では，大規模データをとにかく何とかして可視化する必要があり，大規模データの取り扱いが大きな課題となる．特に，非定常解析やLES解析から出てくる時系列データの扱いが一つのポイントであろう．また，大規模データの中から，データを壊すことなくその流れの特徴を捉えて（Feature detection），現象の本質を理解し，新たな発見を誘発するのに有効な表示法などの模索も重要な要素である．

F.3 新中央可視化システムの目標と要求要件

以上の整理・分析の下に新システムの性能や機能を絞り込んで行く中で，純可視化システムとしての性能/機能の他に，独立行政法人化に伴う効率化や説明責任の重要性高揚等に伴い，研究開発スタイルの革新による業務効率の改善やアピール効果の向上に関する可能性も検討対象とした．結果として，新中央可視化システムが持つべき目標は**表 F.4** に示す３点に集約された．

その目標を典型的な技術要求要件として絞り込み，それに対して現状与えられ得るソリューションを検討した．大規模データとしては将来への展開も見据え，1G 点（10 億点，1000 の３乗）を高速に処理する（数分以内に全てメモリに読込み，1 分以内に可視化する，表示画像に対して拡大縮小移動回転などの視点変換操作がリアルタイムに行える）ことを想定し，大規模共有メモリを備えた高性能可視化サーバが必要と判断し，グラフィックスエンジンの仕様やメモリ量を決定した．可視化力とアピール効果については，「何を見るか，どう見るか」という可視化の本来的な目的の他に，「何を見せるか，どう見せるか」という視点からの考慮を加え，ステレオ表示や3Dポインティングの機能を備えた大画面表示を採用することとした．

表F.4 新中央可視化システムが持つべき目標

1) CFD解析の複雑大規模化及び統合化に対応した可視化処理技術の高度化

キーワード ⇒ リアリティ

- 解像度，写実性，没入感の向上による可視化力の増強
- 空間的な位置検出や位置関係の把握の容易化

2) CFD解析と実験／設計プロセスとの融合による協調的研究開発環境の確立

キーワード ⇒ コラボレーション

- 計算結果，実験結果，設計構想の同一画面上での比較によるデータ信頼性及び生産性の向上
- 大画面表示による利用者コミュニケーションの能率化

3) 研究開発成果のアピール効果の向上と広報的利用の推進

キーワード ⇒ コミュニケーション

- 成果を多くの人に効果的に伝えるコンテンツ作成力
- 教育用，広報用として利用

表F.5 新中央可視化ｼｽﾃﾑへの要求要件とｿﾘｭｰｼｮﾝ概要

要求要件	ソリューション
1G点のデータ処理能力	高性能可視化サーバ（要ｸﾞﾗﾌｨｯｸｽｴﾝｼﾞﾝ，64GBﾒﾓﾘ）
可視化力，ｱﾋﾟｰﾙ効果	大画面表示，ｽﾃﾚｵ表示，３Dﾎﾟｲﾝﾃｨﾝｸﾞ
計算ｻｰﾊﾞのﾃﾞｰﾀを直接読む	GSNによる結合，専用ライブラリ
ユーザ対応力	階層的可視化，ウェブ可視化
コンテンツ作成力	ノンリニアビデオ編集

計算サーバから可視化サーバへのデータ転送が処理のボトルネックにならないためには，① 十分な転送速度，② スパコンのディスク上のデータを直接参照できること（ftp を使わない）が必要であり，①については 1GB/秒程度のデータ転送速度を確保する必要がある．その際プロトコルやソフトウェアの整合性を保証する必要がある．②については，スパコンシステムと可視化システム間でのいわゆるファイル共有機能を実現する必要がある．

また，サーバの持つ能力を有効に活用し，ユーザにとって最適なシステム利用性を構築するために，ユーザを3ランク（特別パワーユーザ，普通パワーユーザ，リモートユーザ）に階層化し，資源に応じた可視化利用環境を提供することとした．これらのソリューションの概要を**表F.5**に示す．

F.4 新中央可視化システムの調達と導入までの経緯

可視化システム（可視化サーバを含むシステム全体を指す．）は，スパコンではないが，やはり政府調達に「コンピュータ調達」（**付録 E** 参照）という分類があり，金額規模的

には含まれるものであるので，安全を期すためにこの手続きに則って調達することとした．**図 F.6** に落札までの手続きと経緯を示した．10 月の落札の後，可視化のためのスペースが必要であることから，検討の結果，**図 F.7** に示したように計算科学 3 号館の隣接地にプレハブの可視化センターを建設することとした．

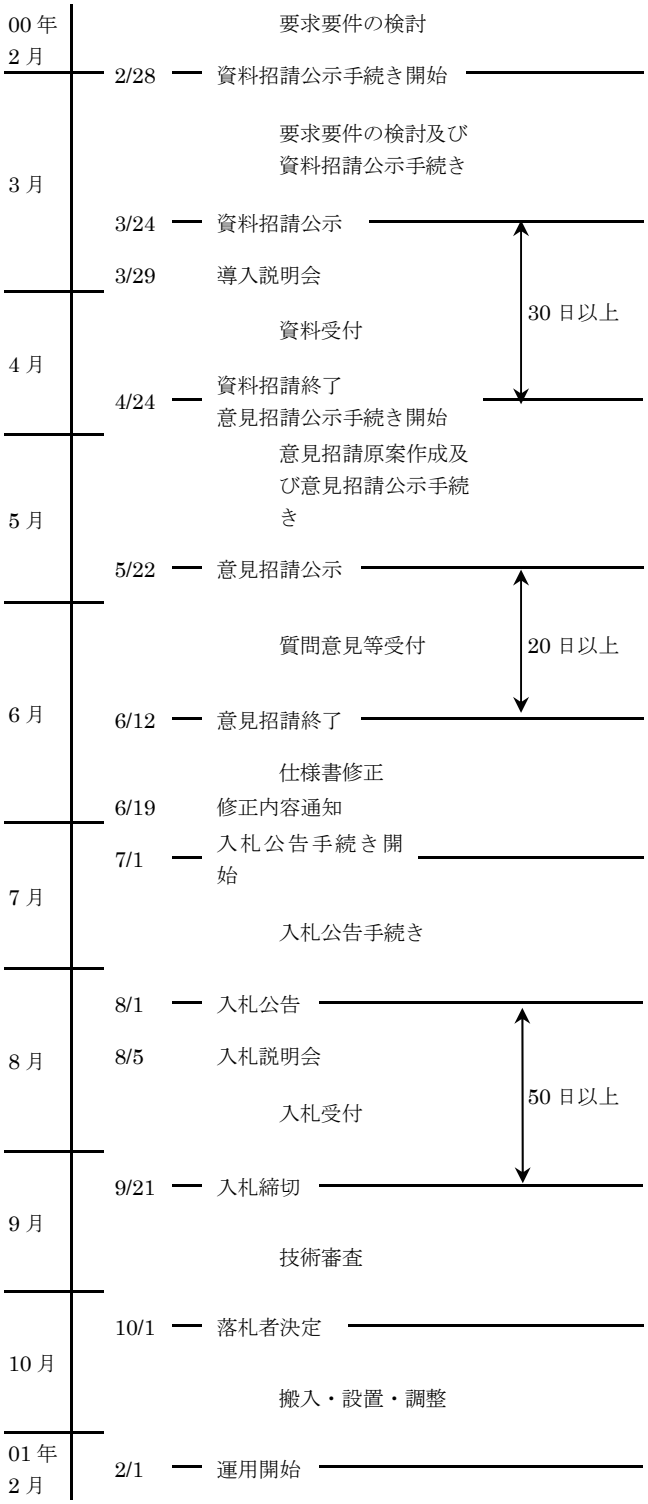


図 F.6　可視化システム調達経緯

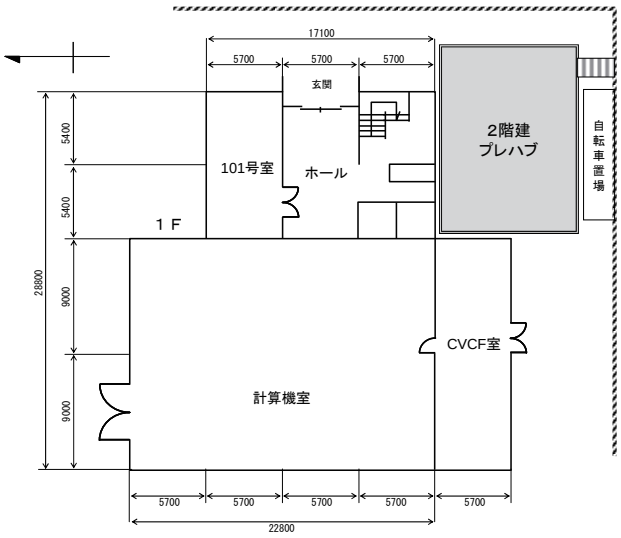


図 F.7　可視化センターの建設地

F.5　新中央可視化システムの構成概要

導入された新中央可視化システムは CeViS（Central Visualization System）と呼ばれ，**図 F.8** のように可視化サーバ，大型三次元表示装置，高性能グラフィックス端末，ノンリニアビデオ編集装置などから成る．**図 F.9** に CeViS の構成概要を示す．以下に，CeViS の構成上，特徴的なものについての技術的な内容を述べる．

図 F.8　新中央可視化システム（CeViS）の概要

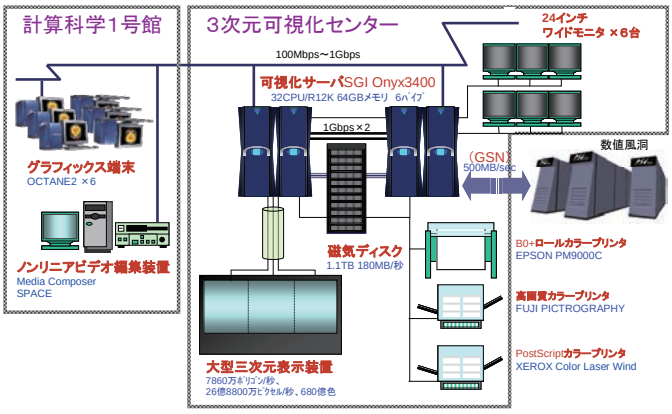


図 F.9　CeViS の構成概要

F.5.1 可視化サーバ

画像生成装置としての可視化サーバの役割は重要である．新システムでは，SGI 社の Onyx 3400 を採用した（**図 F.10**）．CPU は，MIPS R12000A 400MHz を 32 個有し，64GB の共有メモリ空間を提供する．データ一時保管のためのキャッシュディスクとして約 1.1TB の容量を搭載している．ジオメトリエンジンは Infinite Reality 3，グラフィックスパイプを 6 本有している．パイプあたり 256MB のテクスチャメモリ，756MB のフレームバッファ，1,310 万ポリゴン/秒の描画性能，680 億色の表示性能を持っており，高解像度表示やステレオ表示，動画表示に十分対応可能である．

図 F.10 可視化サーバ SGI Onyx3400

F.5.2 大型三次元表示装置（エアロビジョン）

可視化サーバと並んで CeViS システムの中核となるのが大型三次元表示装置（エアロビジョン）である．前記の技術要件を満たすために，以下のように仕様を決めた．全体の方式を決める上でスペースは重要なファクターであるが，今回の場合，スペース的余裕がなかったので，CAVE 型やドーム型は排除した．表示の不正確さから曲面スクリーンも対象外とし，想定人員（20 人程度の小規模），機能性，拡張性などからウォール型平面スクリーンが残った．画面の大きさは，実験機（6m 程度）やエンジンが実物大で表示できるようなサイズ（4.6m×1.5m）とした．ステレオ表示が必要ということから，プロジェクタは 3 管式とし，できる限り明るくしたかったので，BARCO 社製の REALITY 812 という 12 インチ管を選定した．エッジブレンディングには，SEMU と呼ばれる BARCO の内蔵ユニットを利用した．投影方式は，スクリーンに影が映るのを避ける等の理由からリア投影方式を選んだ．その場合，バックヤードスペースが必要になるが，映像を鏡で反射させることでバックヤード部分を最小化する方式を採用し，プロジェクタや反射鏡の取付けのためにボックス型剛体構造とした．スクリーンは，映像の正確さを勘案し，ハードスクリーンとした．拡散コーティングは，客席側ではなく裏側に施すことにより，拡散面の保護，ボックス内での光乱反射の防止を図った．大画面の構成は，3 面を横に結合させる方式とし，解像度は単面で現在の最高解像度（SXGA）に耐え得るようにした．以上から決められたスペックを**表 F.11** に示す．**図 F.12** は，大型三次元表示装置の前面写真であり，**図 F.13** は内部構造の概略，**図 F.14** に主な寸法を示した．**図 F.15** には，スクリーン裏側のプロジェクタの設置状況を写真を示した．

表 F.11 大型三次元表示装置の仕様概要

画面形状，ｻｲｽﾞ	ウォール型平面，4.6m×1.5m
投影方式	リア投影，ﾊｰﾄﾞｽｸﾘｰﾝ（裏面ｺｰﾃｨﾝｸﾞ）
解像度	3,300 ドット×1,028 ドット
明るさ	500ANSI ルーメン
機能	ｴｯｼﾞﾌﾞﾚﾝﾃﾞｨﾝｸﾞ，ｽﾃﾚｵ表示，保守容易

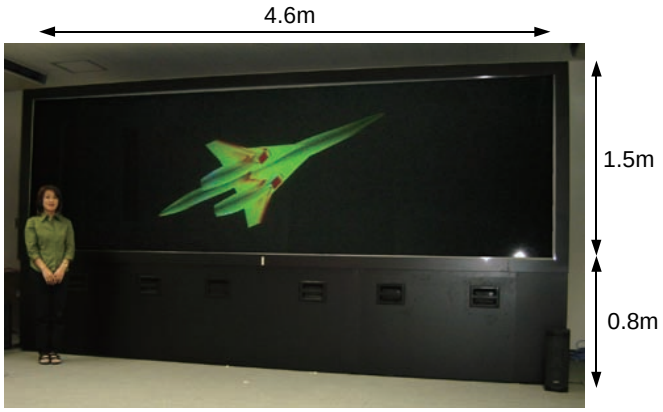


図 F.12 大型三次元可視化装置の前景

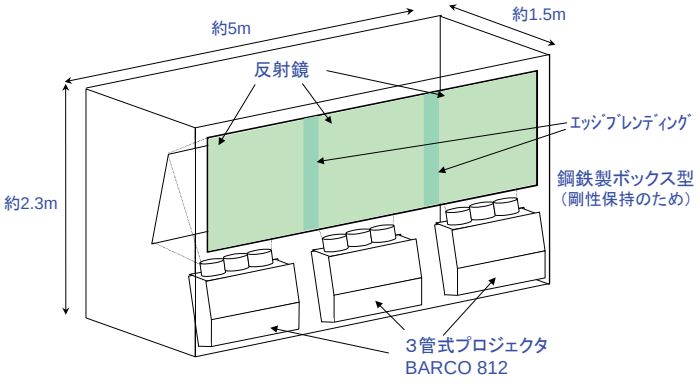


図 F.13 大型三次元表示装置の構造

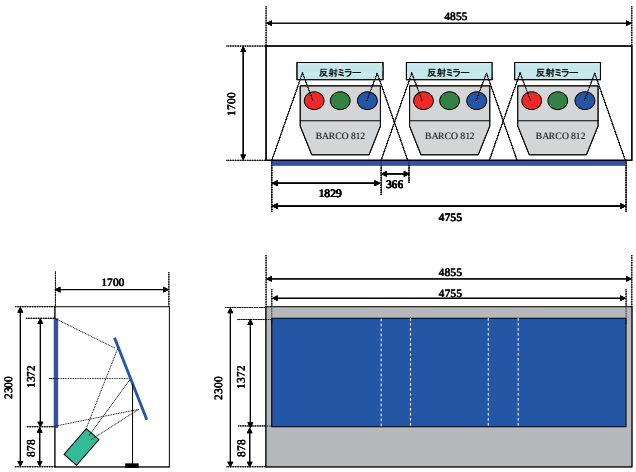


図 F.14　大型三次元表示装置の寸法

図 F.15　スクリーン裏側の様子

一方で，実際に使えるシステムを組むためには，制御方式，ソフトウェア，付加機能，維持管理などを別途検討する必要がある．システムの制御は，**図 F.16** に示したように，可視化室内に専用の制御用のデスク（制御卓）を用意し，オペレータが制御を行うような形式にした．プロジェクタの信号制御，画像ソースの切替えなどは，タッチパネルによりワンタッチで切り替えられるようにし，初心者の利用性にも配慮した．ソフトウェアは，制御ソフトウェアと可視化アプリケーションが要る．可視化アプリは，専用ソフトを作り込んでしまうと表示バリエーションがなくなる，またメンテナンスにも苦労するので，できるだけ流通しているものを使えるように設計した．大画面表示に対応する可視化アプリとして，AVS/MPE，EnSightGOLD を導入した．FIELDVIEW などのアプリも使えるように 1 パイプで表示するモードを設けた．付加機能として，3D ポインティング，会議用，パソコン画面表示などを考慮した．3D ポインティング用に，COVISE システムを導入した．会議用として，液晶プロジェクタを内蔵させた．

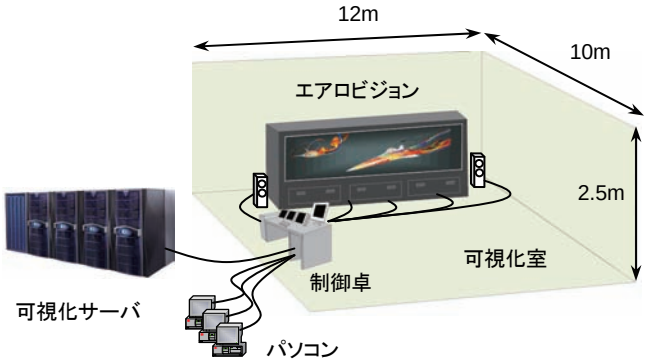


図 F.16　可視化スペースとシステム構成

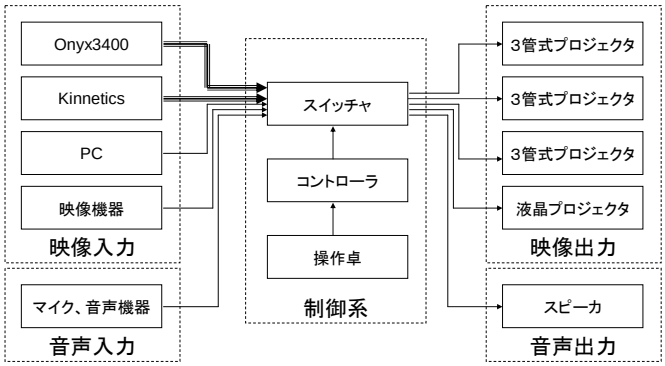


図 F.17　制御系の回路

F.5.3　GSN リンク

計算システムと可視化システム間の目標データ転送速度 1GB/秒を実現するための接続手段としては，イーサネット系はレイテンシが悪く，また，CPU 負荷が高い割りに実効速度も低いので排除された．この時期よく使われたプロトコルとして HiPPI（ピーク 100MB/秒）があり，1998 年には HiPPI-6400 と呼ばれるピーク 800MB/秒の規格が出ていた．別名，GSN（Gigabyte System Network）と呼ばれるものであり，スパコンへの適用を対象として設計されていた．2007 年の時点で考えれば，ファイバーチャネルや Infinibannd による高速接続の複数の選択肢が可能であるが，当時としてはこれが最高速の接続形態であったこともあり，また，HiPPI 系は実績もあり CPU 負荷も低いことから，GSN による接続を採用することとした．ただし，1 本だけで実効で数 100MB/秒の実効速度を出すのは困難であるので，何本か束ねることとし，1GB/秒の目標についても，現実を見据えて 500MB/秒と設定し，最終的には 4 本の GSN を束ねて接続することとした．これを GSN リンクと呼ぶこととした（**図 F.18**）

次に利用者からみた使い勝手について考えると，まず，可視化アプリからストレージシステムのファイルを直接読み込めることが望ましい．そこで，富士通株式会社の協力を得て，STF と呼ばれるライブラリを整備し，C 言語のプログラムから STF の read を呼び出すことにより，ストレージシステムのファイルを直接読み込めるような実装とした．また，

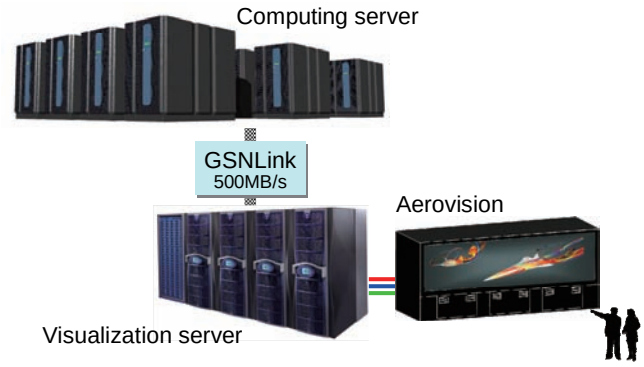


図 F.18 GSN リンクの概念図

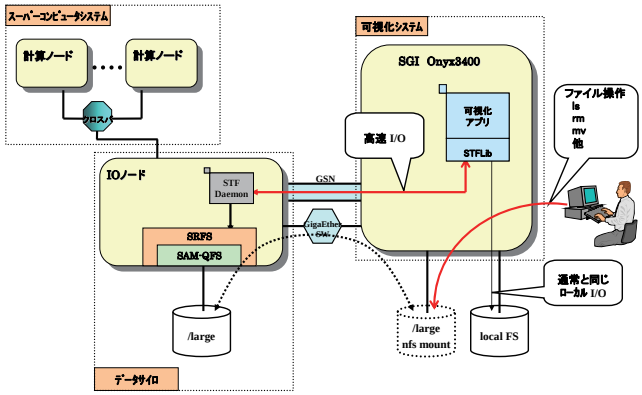


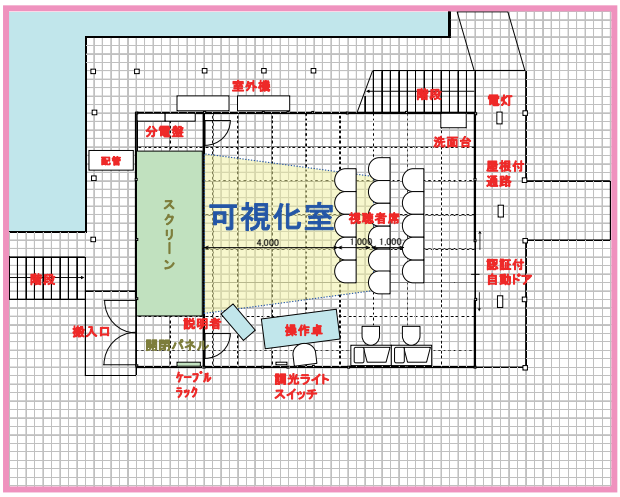
図 F.19 GSN リンクの運用概念

可視化システム側から GSN リンクを使って高速にデータをコピーするコマンドアプリケーション *gsncp* を用意し，大規模データを高速に CeNSS から CeViS に持って来られるようにした．**図 F.19** に GSN リンクの運用概念を示した．

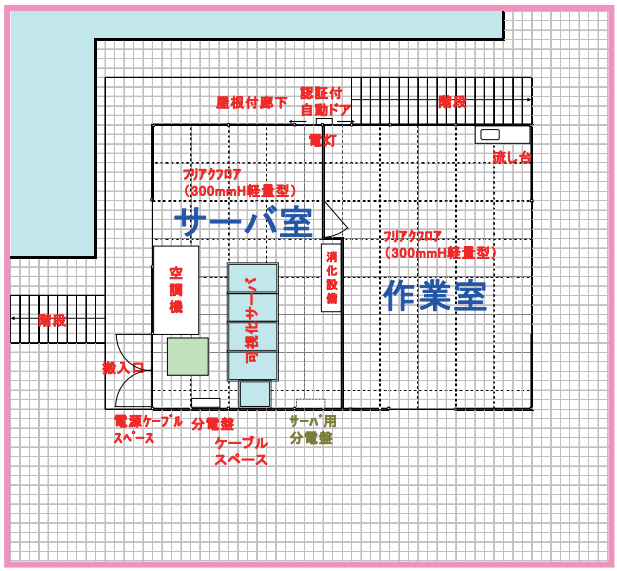
F.5.4 3 次元可視化センター

F.4 で述べたように，可視化システム（大型三次元表示装置と可視化サーバ）を収容する建物として新たに 3 次元可視化センターを建設した（**図 F.20**）．この建物の 1 階に，**図 F.21** のスペースを有する暗室タイプの可視化室を確保した．**図 F.22** に，可視化室において 3 次元可視化を行っているときの様子を示した．

図 F.20 3 次元可視化センターの外観



(a) 1 階の平面図



(b) 2 階の平面図

図 F.21 3 次元可視化センターの平面図

図 F.22 可視化室における 3 次元可視化の様子

F.6　新中央可視化システムよる可視化事例

ここでは，CeViS による幾つかの可視化事例を紹介することにより，CeViS システムの特質，特に大画面表示の長短所について論じてみたい．

大画面表示の特徴の第 1 は，複雑形状の全体像を一画面で見ることができることである．**図 F.23** は，航空機全機まわりの CFD 解析結果から表面の圧力分布を示したものである．

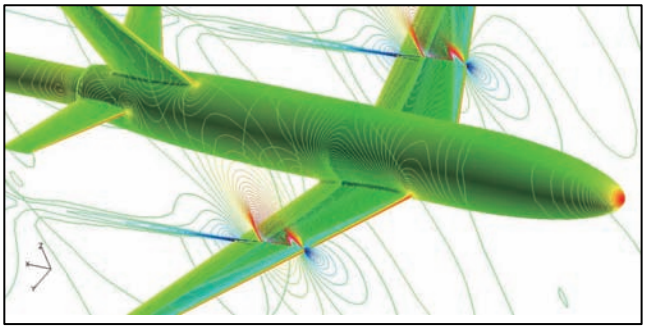


図 F.23　航空機全機まわりの CFD 解析結果（表面圧力）

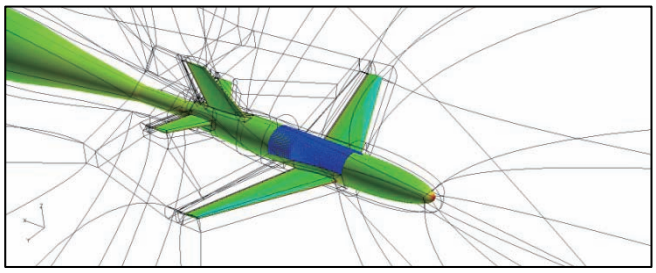


図 F.24　航空機全機 CFD における空間ブロック分割

この解析では，**図 F.24** に示したように空間内に 87 ブロック，650 万点が用いられている．これ程の規模のメッシュ分割が用いられるのは，境界層や衝撃波をきちんと解像し，数カウントの解析要求精度に応えようとしているためである．ブロック分割することで，単一格子では不可能な格子点の適切な分布を達成することができる．しかし，このブロック分割を，通常のデスクトップディスプレイで処理しようとすると，それを作成する場合も含め，ブロック同士の相互関係や接合状態を把握するのが困難になり，専門のスタッフに任せざるを得なくなる．しかし，大画面に表示すると，全体把握が可能になるので，ミスや混乱が軽減されるとともに確認も容易になり，時間や経費の節減になる．航空宇宙の機体形状は細長い場合が多いので，横長画面は全体表示に適する．また，物体が移動していくような場合にも大画面表示は都合良い．ただし，全体表示はサーバに負担をかけるので，データの間引きなどの工夫が要るかもしれない．

大画面表示の特徴の第 2 は，複雑形状の細部まで見ることができることである．**図 F.25** は，ジェットエンジン内の翼列段を過ぎる流れを 5,000 万点以上の格子点数を用いて解析したもので，翼列の半径一定のある断面で切断したときのエントロピの分布を表示している．対象がこれほど細かくなると，デスクトップディスプレイの場合には表示が小さくなりすぎるが，これを大画面表示で見ると，細部の状況まで比較的自然に理解することができる．

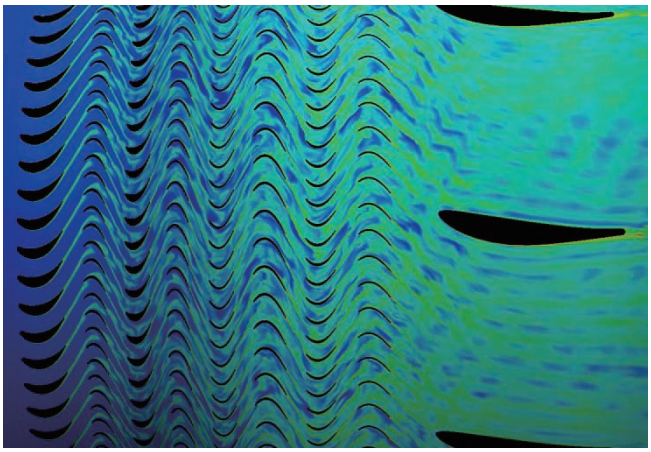


図 F.25　エンジン内の半径一定断面上のエントロピ分布

第 3 の特徴として，感覚・印象の違いを指摘することができる．**図 F.26** は，水素ガスの燃焼解析をボリューム可視化したものであり，動画にもなっている．計算自体は，2,000 万点以上を使った極めて大規模なものである．同じコンテンツでも，例えばノートパソコン上で見るのと，大画面上で見るのとでは感覚的にかなり違った印象がある．アピール度，写実感，鮮明度など，これほど違うものかと驚かされるものがある．単に慣れの問題なのかもしれないし，個人差もあるだろうが，スケール効果，体験感とでも分析することができるだろうか．

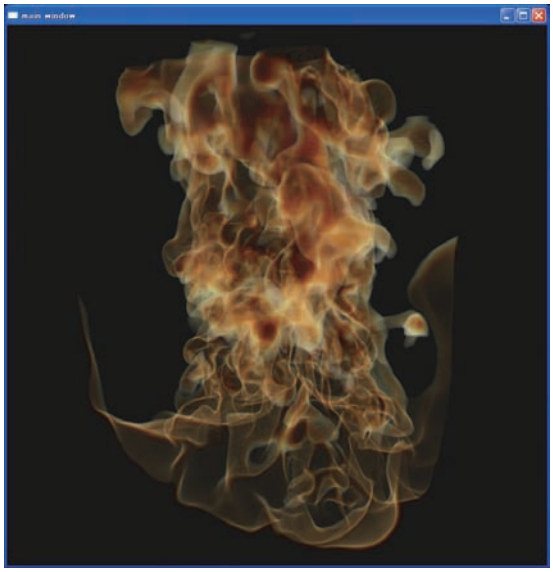


図 F.26　水素ガス燃焼のボリューム可視化

CeViS システムでは，3D ポインティング機能を有している．このような機能を導入したのは，複雑形状に対し，通常のマウス操作だけでは，オブジェクト操作や空間位置把握に限界があるとの指摘があったからである．ここでは，COVISE と呼ばれるシステムを導入した．ペンシル型のポインティングデバイスとステレオ表示を組み合わせることで，

表示対象に対する回転，移動，断面カット等の操作，あるいはウォークスルーなどをペンシルの回転，移動により直接的に行うことができる（**図 F.27**）．従って，通常のマウスによる操作環境で複雑形状を可視化するときに直面していたオブジェクト操作に対するもどかしさはかなり緩和される．適用例が未だ少なく未知な部分も多いが，新たな可視化方式として期待が持たれる．

図 F.27　3D ポインティングの実際

F.7　エンジニアリング可視化

近年，CFD などの数値シミュレーション技術を，設計や性能評価，環境アセスメントなどの場に積極的に利用して行こうとする動きが盛んである．航技研でも，小型超音速実験機や宇宙往還機の実験機プロジェクトにおける機体設計や性能確認に，CFD 解析や CFD による最適設計技術を主体的に用いるようになって来ている．その場合に要求される可視化の内容は，学術研究，現象理解などに求められる可視化（サイエンティフィック可視化）の内容とは自ずと異なったものになる．そのような場における可視化ニーズ，可視化技術を包括できないかという観点から，我々は，「エンジニアリング可視化」という発想・括りを提案している．ここで，サイエンティフィック可視化との対比で連想してみると，両者の相違は**表 F.28** に掲げたキーワードとして整理することができる．

表 F.28　ｻｲｴﾝﾃｨﾌｨｯｸ可視化とｴﾝｼﾞﾆｱﾘﾝｸﾞ可視化

サイエンティフィック 可視化	エンジニアリング 可視化
見えないものを見る	見たいものを見る
発見的	意志的
自然現象対象	人工物対象
何を見るか，どう見るか	何を見せるか，どう見せるか
正確さ，写実性	生産性，機能性
再現性，確実性	再利用性，拡張性
個別的，客観性	協調的，アピール性

このように整理してみると，エンジニアリング可視化の概念が明確になって来る．この観点で，今回導入した CeViS システムや大画面表示を捉え直してみると，例えば，大画面表示というのはエンジニアリング可視化の極めて自然な解の一部であると考えることができる．つまり，「何を見せるか，どう見せるか」の一手法として大画面というものがある．最近の表示技術の進歩や計算機性能の向上により，可視化研究や技術開発に対する成熟感が募っているように思われるが，エンジニアリング可視化という視点で考えてみると，表示方式の持つべき機能や，我々に必要な可視化の方向なり地平なりを，より鮮明にすることができる．さらに，今まで見過ごしていたもの，画一的な価値観の中に埋もれていたもの，が見えてくる可能性もある．

大画面表示による可視化の今後の課題としては，技術的なものと利用に係わるものがあり，

1)デスクトップ可視化環境ではわかりにくくなってしまう粒子追跡や等値面表示も，大きく見せることによりわかりやすくなる．大画面で表示は，通常のデスクトップ可視化の延長線上にはない．どのような表現法が最も効果的か？
2)大画面表示は，高い CPU 負荷や転送負荷を伴う．負荷分散やデータの間引き，高速レンダリングの手法などの開発がより強く求められる？
3)大画面表示にはどのようなコンテンツが最も相応しいか？　キラーコンテンツはどのようなものか？
4)多人数での利用や，遠隔利用などのバリエーションが考えられるが，どのような利用法が最も相応しいか？

などが挙げられる．

F.8　おわりに

以上，航技研新中央可視化システムの戦略，概要，可視化事例，展望，課題などを紹介した．この種の方式の宿命として，どうしても設備が大仕掛けとなり，経費もそれ相応の規模が必要となる．それだけに，政策的必要性や学術的重要性，技術的先進性を十分に説明する必要がある．本稿で背景や戦略に言及したのは，その辺りの我々の構想・経験が，この種の設備の導入に際し何かしらお役に立てるかもしれないと思ったからである．

大画面表示装置を実際に構築して映像を見てみると，その迫力，写実性，臨場感などの可視化システムとしての能力（＝可視化力）は予想以上であったといっても過言ではない．どの分野でもそうだと思うが，シミュレーション技術は，計算機性能やグラフィックス性能の向上と相俟って発達してきたため，行き過ぎや過度の期待が必ずある．CFD でも，Colored Fluid Dynamics と言って揶揄されたこともあるが，人間の持つ特に優れた感覚である「視覚」に訴えるために，可視化において，色彩，パターン，表現力，動画などを使うことは必要でありさえすれ，今後ともそれなしで済まされるということは決してないだろうと思われる．もちろんその場合，計算結果を如何に忠実に可視化するか，計算結果の特徴を如何にして出すか，というような話題とは少し趣きが違う

ということを，誤解を避ける意味で敢えて最後に言及しておきたい．

参考文献

[1] 永安正彦：航技研の数値シミュレーション技術に関する今後の取り組み，航技研特別資料 SP-46，p7，2000.
[2] 松尾裕一：航技研新中央可視化システムの概要とその戦略，　航技研特別資料 SP-53，2001，pp.149-154.

【参考】　大画面表示システムの技術要素

ここでは，参考のために大画面表示システムを構築する際の技術要素について述べる．

図 F.29 は，大画面表示装置の構成要素と，それぞれの要素間の連関を示したものである．主たる要素としては，

① どのような投影形態を選ぶか，
② どのようなスクリーン材質にするか
③ 投影方式，技術にどんなものを選ぶか
④ プロジェクタに何を選ぶか
⑤ 画像を作成するのに何を選ぶか

がある．また，この順番に技術要素を選んで行くことにより，システムとして完成する．

まず，どのような投影形態を選ぶかについては，目的，観客数，会場の大きさなどによって決まって来る．**図 F.30** は，典型的なスクリーン形態を示したものである．ここで，CAVE[41]とは，米国イリノイ大学が開発した没入型の表示システムを指す．我が国の場合，新たに建物を作る場合を除いて，場所の制約がかなりあるので，取れるスペースにより選択肢が限られる．また，以下に示したもののバリエーションもかなりある．画面の大きさは解像度とも関係してくるので注意が必要である．つまり，画面が大きいほど解像度が必要ということであり，プロジェクタの選択にも係ってくる．

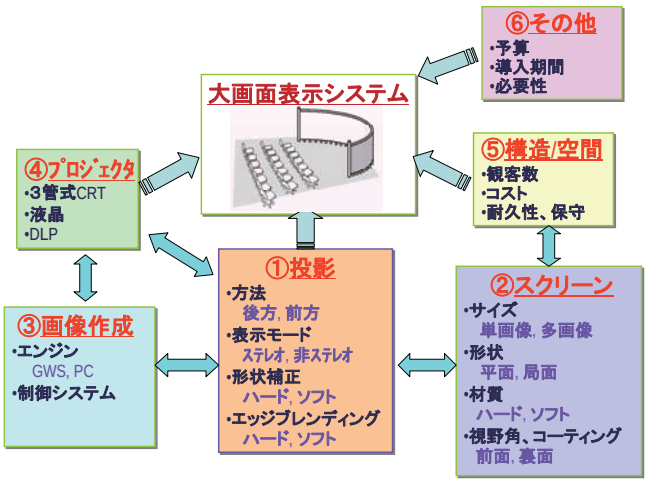


図 F.29　大画面表示システムの構成要素

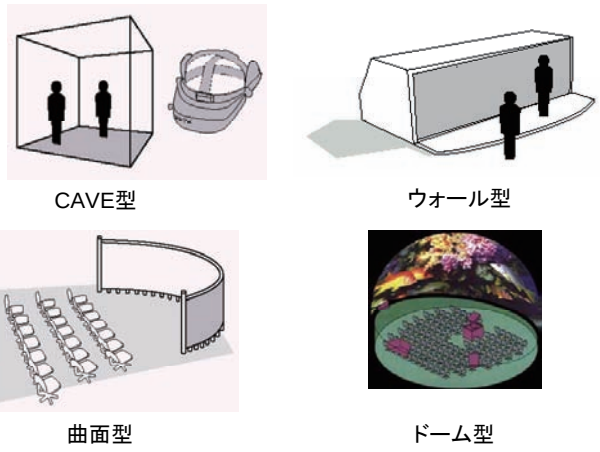


図 F.30　スクリーンの形態

次に，スクリーン材質を選ぶ．スクリーンには，ガラスのような硬いタイプ（ハードスクリーン）と，布のような柔らかいタイプ（ソフトスクリーン）があり，それぞれ**表 F.31** のような特徴がある．また，投影方式には，リア方式とフロント方式がある．**表 F.32** に示した方式のものであり，それぞれに一長一短がある．

表示方式については，いくつかの論点があるが，ここでは，ステレオ表示と大画面表示のためのエッジブレンディングについて述べる．ステレオ表示は，立体視とも呼ばれ，人間の目の錯覚を利用して平面上のディスプレイ上に擬似的に立体的な画像を作り出す技術を指す．**表 F.33** に，ステレオ表示を実現する手段とそれぞれの特徴を示す．

エッジブレンディングは，2 つの画面を滑らかに接続しあたかも 1 枚の画像のように見せる技術であり，大画面表示に特有の技術である．低解像度のまま画像を拡大するとピクセルが目立つようになり見苦しくなるので，画面の大きさの応じて解像度を上げる必要がある．しかし，1 台であまり高い解像度のプロジェクタはないので，複数の画面を接続して表示する必要が出てくる．横に長い画面の場合には，2 面なり 3 面なりの画像を横に連結する必要がある．この場合，光の性質から画面の境界部分にラインがくっきりと出てしまい，両画面のラインをきっちり合わせるのは至難となる．これを解消するために，境界部分の画面を 10 - 20%程度重ね合わせ，その部分の輝度を徐々に落としながら表示し，全体として明るさを一定に保ちつつ境界部分を目立たなくさせるのがエッジブレンディングと呼ばれる技術（**図 F.34**）である．ただし，正面から見るときれいに連続して見える場合でも，斜めから見ると，画面毎に光の来る方向が異なるので，画像の連続性は劣ることに注意する．エッジブレンディングには，ハードウェアとソフトウェアで実現する方法がある．BARCO 社のプロジェクタは，SEMU と呼ばれるハードウェアのエッジブレンディングユニットを内臓している．ソフトウェアでエッジブレンディングを実現しているものとしては AVS/MPE がある．

[41] http://www.evl.uic.edu/index2.php

表 F.31　スクリーンの材質

ﾊｰﾄﾞｽｸﾘｰﾝ	・アクリルをベース ・コストパフォーマンス良好 ・広視野角実現可能 ・画像歪みなし
ｿﾌﾄｽｸﾘｰﾝ	・軟質塩化ビニル等 ・低コスト ・ﾊｰﾄﾞｽｸﾘｰﾝでは実現できない大画面にも対応可 ・画像歪み出る

表 F.32　投影方式

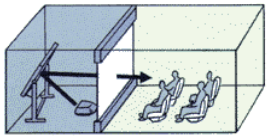
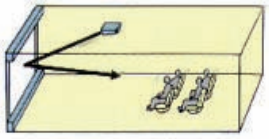
リア方式	フロント方式
	
● 外光の影響を受けにくいので，部屋を明るくできる ● スクリーン近傍まで視聴エリアを確保できるので，広視野角が得られる ● 設置にはバックヤードが必要．また，明るいプロジェクタが必要 ● 視点自由度の高いシステムに必須	● 外光の影響を受けやすいので，部屋を暗くする必要がある ● 映像の遮蔽を避けるために視聴エリアが制限され，広視野角を得にくい ● 設置にバックヤードが不要なので省スペース ● 視点固定システム，大型ドームに有効

表 F.33　ステレオ表示の実現手段と特徴

方式	液晶シャッタメガネ	偏光メガネ	
	時分割		ツインスタック
プロジェクタ	3 管式	3 管式 ＋ 偏光スクリーン	液晶 2 台 ＋ 偏光ユニット
長所	・ｽｸﾘｰﾝを選ばない ・一面につき，ﾌﾟﾛｼﾞｪｸﾀ 1 台と 1 系統の映像信号	・ランニングコスト小 ・メガネが小型・軽量 ・プロジェクタ 1 台で偏光立体視方式実現 ・視野角に最適なメガネを選定できる	・ﾌﾟﾛｼﾞｪｸﾀを選ばない ・ランニングコスト小 ・映像を明るくできる ・視野角に最適なメガネ選定可 ・高解像度 ・メガネが小型・軽量
短所	・映像が暗い ・メガネ装着の違和感 ・ランニングコスト大 ・広視野角映像がﾒｶﾞﾈで遮られる ・解像度が下がる	・映像が暗い ・偏光を崩さないｽｸﾘｰﾝ素材を選定する必要あり ・偏光フィルタの影響により色が変化	・一面につきﾌﾟﾛｼﾞｪｸﾀ 2 台と 2 系統の映像信号が必要 ・偏光を崩さないｽｸﾘｰﾝ素材を選定する必要あり ・偏光フィルタの影響により色が変化

次に，大画面表示に重要な技術要素としてプロジェクタがある．**表 F.35** に，主なプロジェクタの種類と特徴を示す．プロジェクタは，極めて進歩の速い技術であり，**表 F.35** は 2001 年レベルでの技術に基づく比較であることに注意する．総合的に見て，液晶や DLP(Digital Light Processing)は，明るさの点では有利であるが，色や機能に問題がある．3 管式（CRT 方式）は，多機能であり色の正確さや微妙な調整が可能というメリットもある反面で，明るさの点で不十分でありコスト的にも高い．また，これらの他に，導入コストやスペースの観点もある．液晶プロジェクタは急速に技術開発とコスト低下が起こっており，将来的には有望と思われる．

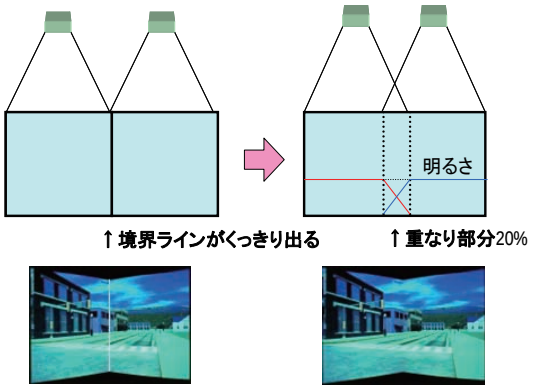


図 F.34　エッジブレンディング

表 F.35　各種プロジェクタの特徴

3 管式	・光出力 150～230ANSI ルーメン（暗い） ・最大 500ANSI ルーメン ・光出力は内蔵 CRT の大きさに依存（8 インチ CRT で 230ANSI ルーメン，12 インチ CRT で 500 ルーメン程度） ・入力信号の対応性に優れる（帯域が広い） ・時分割式，偏光方式の立体視に対応可能 ・エッジブレンディング対応可能 ・ジオメトリ補正機能に優れているため湾曲したスクリーンに映像提示可能 ・画面中央部と周辺部で輝度の差が大きいためホットスポットが発生しやすい ・ランニングコスト大（CRT は消耗品，定期的な映像調整が必要）
液晶	・光出力 1000～6000ANSI ルーメン（明るい） ・3 管式にくらべ 1 桁明るい ・垂直周波数が 3 管式に比べ上限が低いため時分割式の立体視は不可 ・偏光方式の立体視に対応可能 ・エッジブレンディング用ハードウェアが高価で性能に限界あり ・ジオメトリ補正機能に乏しく映像調整に難あり ・画面中央部と周辺差が小さいためホットスポットが発生しにくい ・ランニングコスト小（消耗品は光源のみ）
DLP	・光出力 1000～5000ANSI ルーメン（非常に明るい） ・最大 12000ANSI ルーメン ・コントラスト比　500:1 ・時分割式および偏光方式の立体視に対応可能 ・エッジブレンディング用ハードウェアが高価で性能に限界あり ・ジオメトリ補正機能に乏しく映像調整に難あり ・画面中央部と周辺差が小さいためホットスポットが発生しにくい ・ランニングコスト小（消耗品は光源のみ）

付録 G　NS-III システム運用設計書　第 2.1 版

平成 15 年 5 月 30 日
CFD 技術開発センター
計算情報基盤技術開発室

１．はじめに

航技研は，1987 年より数値シミュレータ（Numerical Simulator，NS システム）と呼ばれるスーパーコンピュータを中核に据えた共用計算機設備の運用を行っている．NS システムは，2002 年 10 月より，新システムが稼働し第 3 世代（NS-III）の運用となる．

本書は，NS-III システムの運用開始にあたり，運用方針と必要なシステム設定について述べる．

２．システム概要

(1)システム構成概要

NS-III システムの構成概要を下図 G2.1 に示す.

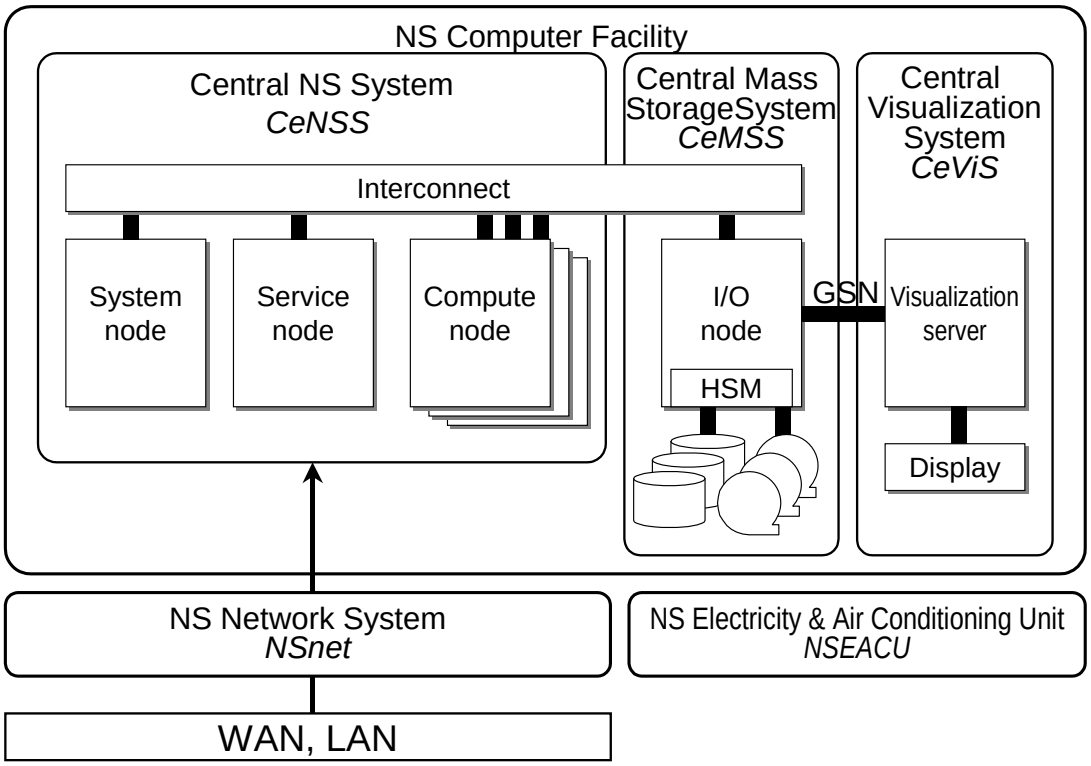


図 G2.1　NSIII システムの概要

(2)システム構成表

NS システムは，NS 計算機設備，NS ネットワーク，NS 電気・空調設備から成る．それぞれの構成要素の名前，略称などを下表 G2.2 に示す.

表 G2.2 ＮＳシステムの構成一覧

システム名	英語名	略号
NS システム	Numerical Simulator	NS
NS 計算機設備	NS Computer Facility	NSCF
中央 NS システム	Central Numerical Simulation System	CeNSS
計算ノード	Compute Node	CPN
サービスノード	Service Node	SVN
I/O ノード	I/O Node	ION
ログインノード	Login Node	LIN
システムノード	System Node	SYN
結合ネットワーク	Interconnect	CBN
GSN ネットワーク	GSN Link	GSNLink
ネットワークディスク	Network Attached Disk	NR1000
中央マスストレージシステム	Central Mass Storage System	CeMSS
ディスクサブシステム	Disk Storage Subsystem	
テープサブシステム	Tape Storage Subsystem	
中央可視化システム	Central Visualization System	CeViS
可視化サーバ	Visualization Server	VisServ
大型三次元表示装置	Large-Sized 3D Display System	Aerovision
NS ネットワーク	NS Network	NSnet
NS ｼｽﾃﾑｴﾘｱﾈｯﾄﾜｰｸ	NS System Area Network	NSSAN
基線	Gigabit Ethernet	
ルータ	Router	Alpine
連携ネットワーク	Cooperation Network	COOPnet
ファイアウオール	NS Fire Wall	NSFW
暗号化サーバ	SSH Server	
NS 電気・空調設備	NS Electricity & Air Conditioning Unit	NSEACU
NS 設備	NS Equipment	NSequip
電気設備	Electricity Equipment	
空調設備	Air Cooling Equipment	
NS 空調自動運転システム	NS Active Zone Control Air Cooling Sys.	NAZCA
自動運転システム		

3．システムの詳細

3.1 中央 NS システム（CeNSS）

中央 NS システムは，NS システムの中核をなす計算機設備の主要部分である．Central Numerical Simulation System を略して CENSS と呼ぶ．高速のスカラーCPU を多数結合した共有メモリの SMP ノードを，高速なクロスバーネットワークで結合した分散主記憶パラレルアーキテクチャの計算機である．

◆ 主要ハードウェア

【計算ノード】

・ピーク性能	9.3TFLOPS
・総メモリ容量	3.6TB
・CPU アーキテクチャ	SPARC64 V
・CPU 数	1,792
・計算ノード数	56
・計算ノード構成	SMP(32CPU)

【その他】

・IO ノード数	1（128CPU）
・サービスノード数	4（64CPU）
・ログインノード数	2（64CPU）
・システムノード数	1（PP650 8CPU）
・結合ネットワーク	クロスバ（4GB/s×2）
・占有面積	410m²
・電気量	800KVA

◆ 主要ソフトウェア

・オペレーティングシステム	Solaris 8 (64bit UNIX)
・ バッチ処理	NQS＋航技研独自 NSJS
・ ファイルシステム	BLASTSRSF, BLASTBAND
・コンパイラ	C, C++, F77, F90
・並列化サポート	XPFortran, MPI，自動並列, OpenMP
・数学ライブラリ	SSLII/VP, LAPACK/VP, BLAS/VP, C-SSLII/VP
・プログラミング支援ツール	Parallelnavi Workbench
・アプリケーション	NASTRAN, VisLink, WANS, UPACS, GNU

◆ その他特記事項

・可視化システムとの高速リンク	GSNLink
・可視化システムとの高速リンク用ライブラリ	STF

3.2 中央マスストレージシステム（CeMSS，中央データサイロ）

中央 NS システムのうち，特にストレージ部分を中央マスストレージシステムと呼ぶ．Central Mass Storage System を略して CeMSS，または中央データサイロと呼ぶ．ディスクとテープをファイバーチャネルで階層的に接続したものである．

◆ 主要緒元

・総ストレージ容量	677TB
・ディスク容量	57TB
・テープ容量	620TB
・総ユーザ領域	640TB（ディスク 40TB， テープ 600TB）
・運用・管理形態	HSM
・使用ソフトウェア	SAM-FS, QFS

◆ ディスクストレージサブシステム

・接続種類	ファイバーチャネル(FC1)
・接続本数	80 本

◆ テープストレージサブシステム

・接続種類	ファイバーチャネル(FC1)
・接続本数	40 本
・ドライブ数	40
・テープ媒体	LTO
・テープ 1 巻当たりの容量	100GB

3.3 中央可視化システム（CeViS）

中央可視化システムは，NS システムの中核をなす計算機設備のもう一つの主要部分である．Central Visualization System を略して CeViS と呼ぶ．32 個の高速のスカラーCPU を多数結合した共有メモリアーキテクチャの計算機である．

◆ 主要ハードウェア

・ピーク性能	24GFLOPS
・総メモリ	64GB（共有）
・CPU	R12000(400MHz)
・CPU 数	32
・ディスク容量	1.4TB

◆ 主要ソフトウェア

・オペレーティングシステム	IRIX6.4. (64bit UNIX)
・ バッチ処理	LSF
・コンパイラ	C, C++, F77, F90
・並列化サポート	MPI, OpenMP
・数学ライブラリ	SSLII/VP, LAPACK/VP, BLAS/VP, C-SSLII/VP
・プログラミング支援ツール	SpeedShop, perfex, ja
・可視化ライブラリ	OpenGL, OpenGL Volumizer, IRIS Performer
・可視化アプリケーション	AVS/MPE, EnSight, FIELDVIEW, COVISE, VisLink
・解析アプリケーション	ANSYS, GASPex

3.4 NS ネットワーク（NSnet）

NS システムのネットワーク部分を NS ネットワーク（NSnet）と呼ぶ．スイッチやネットワークディスクから成る NS システムエリアネットワーク（NS System Area Network: NSSAN）と，外部への接続をコントロールする連携ネットワーク（Cooperation Network: COOPnet）から成る．

◆ NSSAN の主要緒元

・基線	Gigabit Ethernet
・ルータ	Alpine
・ネットワークディスク	NR1000(1TB)

◆ COOPnet の主要緒元

・基線	Ethernet
・FW	Sidewinder V4.
・VPN サーバ	cisco VPN3030
・SSH サーバ	Sun Fire280R
・ルータ	cisco34.4.0

3.5 NS 電気・空調設備

電気・空調設備部分を，NS 電気・空調設備（NSEACU）とよぶ．電気・空調の機器部分（NSequip）と，空調機器を自動制御する空調自動運転システム（NAZCA）からなる．

◆ NSequip の主要緒元

・電気設備	計算機用電力供給能力	2050KVA
	空調用電力供給能力	1050KVA
・空調設備	空調機（20 馬力）×19 台	
	冷却能力 807,500kcal/h（700KVA）	

◆ NAZCA の主要緒元

・空調機設備とスーパーコンピュータをネットワークで接続させ，室内を仮想８ブロックに区分し，各ブロック内スーパーコンピュータの稼動状況と連携し，最適な空調機だけを稼動させる空調省エネ容量制御を行う．スーパーコンピュータの稼動率に合わせて，空調機も稼働率を変化させ省エネ運転を行う．さらに，天井に取付けた 25 個の赤外線センサーで，スーパーコンピュータ周囲環境を敏速に判断させ，スーパーコンピュータへ最適な空調環境を提供する．

４．運用方針

4.1 運用時間（2003.4.1～2004.3.31）

(1) NS 計算機設備

運用時間

①窓口業務（受付，クレーム対応，プログラム相談）

平日 9 時 30 分～12 時 00 分，13 時 00 分～17 時 00 分

②アーカイブ室

平日 9時00分～20時00分

保守点検日 13時30分～20時00分

（注）上記時間外におけるアーカイブ室の利用については以下の通りとする．

・20時に建屋出入口自動ロック

・20時に夜間操作員による各種点検及び残業作業者依頼
・20時以降の入館は北側通用口のインターホンにて解錠依頼
・24時より夜間操作員仮眠
・入館は24時以降は原則として禁止する

③NSシステムの運用時間帯

平日　9時00分～翌9時00分
閉庁日の前日　9時00分～翌7時00分
保守点検日　13時30分～翌9時00分

（注）平日18時00分以降において，
・中央ＮＳシステム（CeNSS）のジョブが途絶えるとCeNSSは停止する．
・中央可視化システム（CeViS）のジョブが途絶えるとCeViSは停止する．

④システム作業が必要であるとCFD技術開発センターが判断した場合は，運用を一時停止することがある．この場合は，原則として停止予定時刻の6時間以上前に運用停止予定時刻と運用再開時刻をユーザにメールする．ただし，障害発生時は除く．

休館

⑤年末，年始休館　平成15年12月26日（金）～平成16.年1月5日（月）
（注）平成16年1月5日は保守点検を実施するため月間定期保守は省略する．
⑥年度末休館　平成16年3月31日（月）
⑦運用不可能な状況(機器の導入，撤去，システム関連工事等)の場合は休館とする．その他，CFD技術開発センターが運用不可能と判断した場合は休館とする．

保守

⑧定期保守　毎月第2月曜日　8時00分～13時30分
（注）保守点検日が祝・祭日と重なった場合は，その次の日に行う．

(2) NSネットワーク

①窓口業務（申請受付，クレーム対応など）
平日　9 時 30 分～12 時 00 分，13 時 00 分～17 時 00 分
②定期保守　毎月第2木曜日　8時00分～10時00分
（注）保守点検日が祝・祭日と重なった場合は，第3木曜日に定期保守を行う．

(3)NS 電気・空調設備

①運用方針は NS 計算機設備と同一とする．
②定期保守は NS 計算機設備に合わせて行う．

4.2 運用体制

・NS システムの運用に関する最終権限は，CFD 技術開発センター長にある．
・計画管理室長は，対外機関との調整，所内センター間の要望処理などを行う．
・NS システムの運用管理は，計算情報基盤技術開発室において行う．
・運用管理する所掌範囲により以下の担当者を置く．

NS 計算機：　○○，(△△)
NS ネットワーク：　○○，(△△)
NS 設備：　○○，(△△)　（ ）内は権限代行者

各担当者は，担当する機器範囲の運用計画の策定，機器の管理，障害対応などを行う．

・NS システムの特別利用に関する担当を置く．

NS システム特別利用　○○，(△△)　（ ）内は権限代行者

特別利用担当者は，システムの特別利用に関する要望の吸い上げ，要望の調整，計画の策定，報告などを行う．

・NS 計算機及び NS ネットワークに関する研究開発担当を置く．

NS システム技術開発　○○，(△△)　（ ）内は権限代行者

研究開発担当は，NS システムの計算機及びネットワークに関して，企画立案，研究開発などを行う．また，障害対応には運用担当者と連携して当たる．

・保守はアウトソーシングにより実施する．保守業者との対応は，各担当者が責任を持って行うものとする．

4.3 運用・技術支援者

(1) 作業内容

・NS システムの円滑な運営を維持するために運用支援者及び技術支援者を置く．Q&A，要望，障害などへの対応，システム設定・更新，ソフトウェア作成・改善支援，ユーザ教育支援などを行う．

(2) 作業場所

・運用・技術支援者は，原則として CFD 技術開発センター 1 号館にて作業を行なう．

(3) 常駐時間

・運用・技術支援者の常駐時間は，通常運用時間帯を原則とする．
・夜間運用支援者の常駐時間帯は，夜間・休日運用時間帯を原則とする．

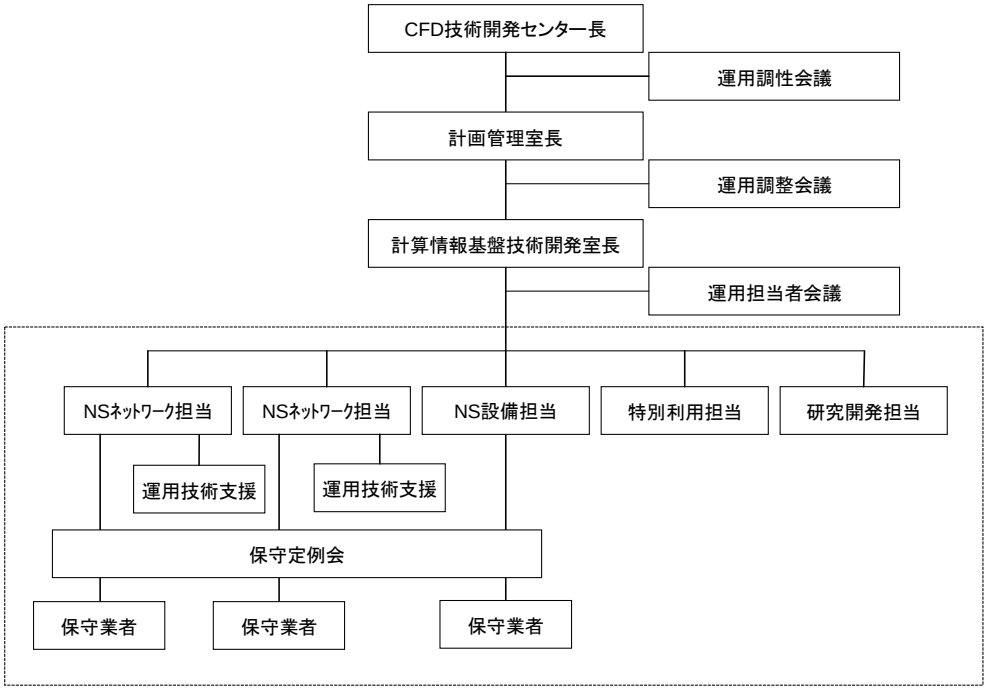


図 G4.1 NSIII システム運用体制

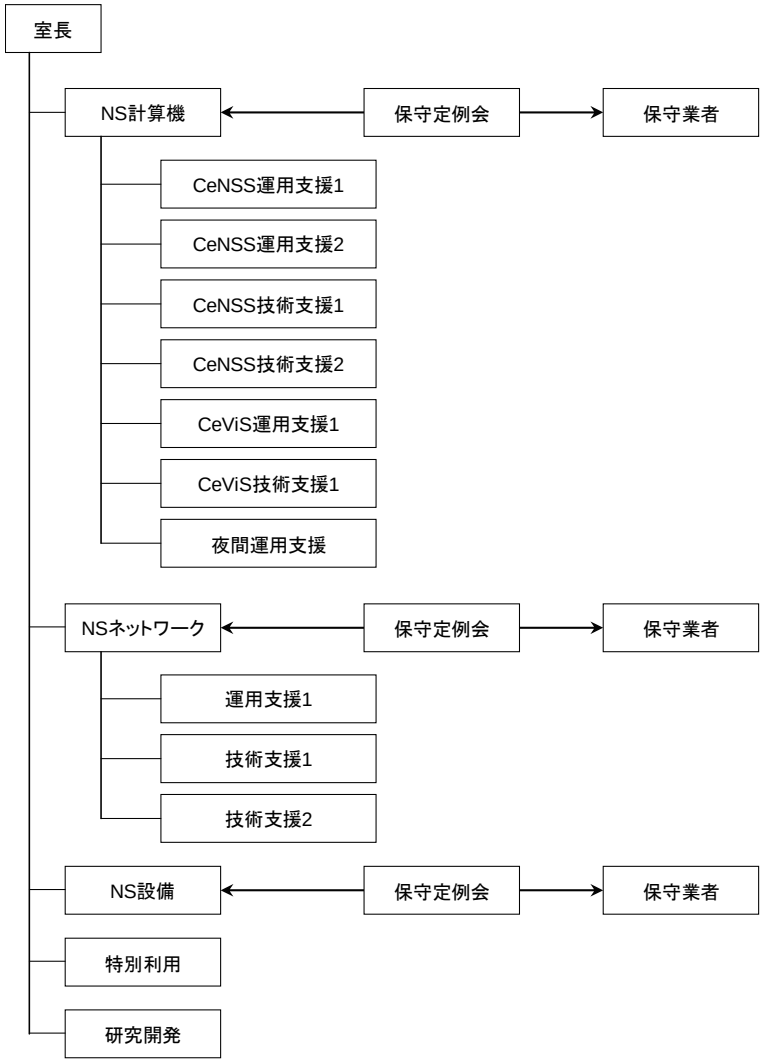


図 G4.2 運用・技術支援者の体制及び保守体制

4.4 ユーザ支援体制

(1) 受付

直接窓口： センタ受付

受付対応者： 運用支援者
受付方法： 直接受付，電話
受付時間： 通常運用時間帯
受付内容： ユーザ登録・抹消，ディスク利用申請，
Q&A，障害，要望など
処理： 受付対応者が内容の切り分けを行ない，必要に応じて関連対応者へ連絡し対処する．

電子窓口： ***@nal.go.jp

受付対応者： 運用支援者
受付方法： 電子メール，ウェブ
受付時間： 随時
受付内容： ディスク利用申請，設備利用申請，
Q&A，障害，要望など
処理： 運用支援者が内容の切り分けを行ない，必要に応じて関連対応者または保守業者に連絡し対処する．

(2) プログラム相談

・運用支援者は，ユーザからのプログラムならびにシステム利用に関する問い合わせ，相談に応じる．
・必要に応じて，保守業者に連絡し，すみやかに対応する．

(3) 管理

・ 障害，Q&A 等の情報管理は，当面運用支援者が行なう．

4.5 運用会議

(1) 主旨

ユーザの要求をシステム構成・運用に適切に反映させるとともに NS システムの運用を円滑に行うため，運用会議を開催する．

(2) 会議体

・運用会議は，運用調整会議，技術連絡会，運用担当者会議から成り，以下の業務を行う．

会議体名	内　容	開催頻度
運用調整会議	運用に関する調整案件の審議，決定	月 1 回
技術連絡会	外部機関との調整，所内センター間の調整	年 1 回程度
運用担当者会議	運用業務の報告，運用実施計画の策定，懸案事項の審議	月 3 回

・各会議は別途定める要領に従って開催する．各会議の位置づけは図 G4.1 を参照のこと．

4.6 保守定例会

(1)主旨

保守業者との連携を密にし，システム運営の効率化を図るとともに，技術情報の交流を円滑に行うために，保守定例会を行う．

(2) 実施頻度

保守定例会は月 1 回開催することとする．日時は随時調整とする．

(3) 参加者

参加者は NAL 担当者，運用・技術支援者，保守業者とする．

(4) 議題

・1 ヶ月間の障害・Q&A・要望対応の状況報告（運用支援者）
・統計情報の報告（運用支援者）
・障害情報の報告（保守業者）
・その他（有益な情報交換等）

(4) 議事録

運用支援者が作成し，関係者の承認を得て，運用支援者が関係者にメールで配布する．

4.7 NS 計算機の定期保守

(1) 定期保守時間帯

毎月第 2 月曜日　8:00～13:30

(2) 作業内容

・システムバックアップ（ミラーリングまたはテープ，頻度は別途定める．）
（注）ユーザデータのバックアップは，基本的に行わない．

・システム再起動，動作確認
・パッチ適用（必要に応じて）
・定期クリーンナップ（対象ディレクトリは運用開始後決定する.）

(3) パッチ適用

パッチは，運用に直結する重大な障害以外は集積パッチ（PTF，一括修正）を待つこととし，定期的（３ヶ月間隔程度）に適用する.

4.8 ユーザ管理

(1) 一般ユーザ管理
・ユーザの利用資格，遵守事項等は利用規約（添付資料１）に従う.
・アカウントの登録・抹消，ホームディレクトリの作成，初期環境の作成，初期パスワードの設定はセンター側で行う.
・CeNSS と CeViS は同一のアカウント及びログイン名を使用するものとする.
・NS システム内のアカウント管理は NIS により行う.
・ログイン名は，従来のナンバー方式（a01, b01,…）の他に，（センターの認める範囲で）任意のアカウント名を認める.
・１人１アカウント所有/システムを原則とする.
・グループユーザは，利用法やセキュリティなどを十分に検討してから利用に処する.

(2) 特殊ユーザ管理
・CFD センター長が認めた特殊ユーザの利用を認める.
・特殊ユーザは，センター全体の運用ポリシーに従わなければならない.

(3) スーパーユーザ管理
・root のパスワード管理は，NAL 運用担当者が行なう．ただし，運用担当者の管理の下に運用支援者にパスワード管理を行わせても良い.
・root のパスワードは各ホストに異なるものを設定する.
・root のパスワードは頻繁に変更する.

(4) その他
・グループ ID は，職員を cls1，職員外を cls2 とする.
・CeViS システムには，デモ用にデモアカウントを作成する.

4.9 NS 計算機設備の運用

(1) CeNSS システム

1) ノード構成/CPU の運用
・当面はシステムを一般ジョブ用及びシステム開発用の 2 分割して運用する．システム開発用には，サービスノード 2 ノード（64CPU×2）を充当する.
・CPU は，SIMPLEX モードで運用する.
・使用時間の制限は，ジョブの経過時間を以て行う.

2) ジョブの運用
・運用の具体的計画は，別途定める運用案に従う.
・NAL 独自スケジューラ NSJS（2003.4 から運用開始）.
・制御則概要：
－できる限り 1 ノードに 1 プロセスで割り当てる．割当てが不可能な場合は，1 ノードに 2 つ以上の並列プロセスを割当てる(UNPACK 方式).
－ジョブ実行ノードでは，ノード内実装 CPU の用途を分割定義(JOB 実行処理，TSS 処理)する.
－全てのバッチ型は SIMPLEX モード実行とし，会話型処理におけるインタラクティブジョブは SHARE モード実行とする.
－同一キュー・同一ユーザ環境下では，リクエストの命名規約に従い，そのリクエスト名単位で，センタ規約に沿った FIFO 制御(シーケンシャル制御：リクエスト順序の保証）を行う.
－同時投入可能ジョブ数および同時実行可能ジョブ数の制限を行う.
同一ユーザ最大同時ジョブ受付可能数 ：22（LONG=22，DEBUG=1，TSS=1）
－他ユーザの実行可能ジョブが存在する限り，同一ユーザジョブは連続実行しない.
－ジョブスワップ運用（DEBUG，TSS ジョブを優先実行）.

(2) ファイルシステム

1) ファイルシステムのソフトウェア
・ノード間ネットワークファイルシステムとして BLASTSRFS を適用する.
（注）SRFS は，クロスバを利用するノード間分散ファイルシステムを提供する.
・HSM ソフトウェアとして SAM-QFS を適用する.

2) ファイルシステムの構築
・ユーザ領域は，ホーム及びジョブテンポラリは NR1000 上に，データはデータサイロ上に構築する.
・ディレクトリは，ホームは/home，ユーザ用ジョブテンポラリは/jobtmp，データは/large，一時ユーザ領

域は/stmp，一時データ領域は/ltmp とする．
・quota 制限を実施する．
・/home，/large は申請期間使用可能な期限付き貸出しファイルとして運用する．
・/stmp，/ltmp，/jobtmp は短期間の期限付き貸出しファイルとして運用する．
・セキュアノードには，それぞれに別ファイルシステムを割り付ける．

ファイルシステム	ファイル容量制限				ファイル数制限				保存期間
	標準		最大		標準		最大		
	soft	hard	soft	hard	soft	hard	soft	hard	
/large	80MB	100MB	40GB	50GB	4KB	5KB	8KB	10KB	システム利用申請期間
/ltmp	10GB	10GB	10GB	10GB	2KB	2KB	2KB	2KB	未参照 60 日
/home	40MB		10GB		4KB		10KB		システム利用申請期間
/jobtmp	500MB		500MB		10KB		10KB		未参照 30 日
/stmp	500MB		500MB		2KB		2KB		未参照 60 日

3) データの冗長性
・/home については，RAID4 による冗長を行う．
・/large については，ディスク上のデータには RAID5，テープ上のデータには 2 コピーによる冗長を行う．
4) HSM の運用
・IO ノードーディスク間はピーク実効速度 1GB/s で運用する．
・テープは 4 ストライプ（非圧縮でピーク 60MB/s）で運用する．
・アーカイブ実行は，4 時間毎に行う．
・ディスクファイル混雑時で一定期間アクセスされないファイルはテープへ自動的にアーカイブされる．
(3) CeViS システム
1)CPU の運用
・可視化利用を主とするが，バッチ処理も受け付ける．
・三次元表示装置使用時は，バッチ処理を退避させる．
2)ジョブの運用
・運用の具体的計画は，別途定める運用案に従う．
・LSF によるバッチ運用を行う．

4.10 CeNSS システム異常時の対応
(1) 通報経路
・ISO9000 文書 L3A470R2 に従い，所定の手続きを踏み，必要と判断される場合には速やかに保守業者に連絡する．
(2) 復帰方法
・障害からの復帰については，管理者の指示に従い実施する．
(3) 部品のリザーブ
・CPU，メモリ，システムボードについては，障害からの復旧時間短縮のために，センター内にリザーブするものとする．

4.11 構成品管理
(1) CeNSS 及び CeViS
・構成品は，対象範囲を決定し，NAL 及び保守業者が管理する．
（注）消耗品及びマニュアルの管理はセンターが行なう．
(2) アーカイブ端末
・センター側でユーザがデータをアーカイブ・リストアするための端末を用意する．
・必要な機能は DAT（DDS3 対応），DTL，LTO 及び CDROM とする．
(3) プリンタ
・センター側で，ユーザが利用できるネットワークプリンタを用意する．
・本プリンタは，CeNSS，CeViS からのログ情報やデバッグ用リストが出力可能な運用とする．
(4) ユーザ端末
・センター側で，ユーザが CeNSS 及び CeViS システムを利用するための端末を用意する．

4.12 情報の管理
障害や Q&A 等の情報を共有するため運用支援サーバで一括管理し，Web ブラウザで参照可能とする．公開レベルを設定し，下記内容を一般ユーザにも積極的に情報発信する．
(1) 管理内容
・障害管理

・Q/A管理
・要望管理
・FAQ
・ログ情報
・ディスク使用状況
・アカウント情報
・その他

(2) ログ情報の管理
・情報収集/保存管理
・情報解析処理
・各種報告書作成処理

4.13 メール
・システム利用ユーザは，システム停止時においても連絡可能とするため，システム外にメールアカウントを持つこととする．所内ユーザは，NAL メールサーバ（asuka）のアカウントを利用し，所外ユーザは，独自のメールアカウントを登録することとする．
・ユーザの.forword ファイルには，メール転送先として通常使用するメールアドレスを定義するように指導する．
・CeNSS 及び CeViS 宛のメールがある場合は，上述のアカウントへフォワードする．
・メーリングリストを以下の通り作成し運用する．

アドレス	用途	所属者
nsadm@nal.go.jp	CeNSS 管理者用 （システムエラー通知など）	CeNSS 管理者 運用・技術支援者
nsusr@nal.go.jp	CeNSS ユーザ用 （定期メンテ案内など）	CeNSS 利用者 CeNSS 管理者
visadm@nal.go.jp	CeViS 管理者用 （システムエラー通知など）	CeViS 管理者 運用・技術支援者
visusr@nal.go.jp	CeNSS ユーザ用 （定期メンテ案内など）	CeViS 利用者 CeViS 管理者
prosou@nal.go.jp	システムに対する問い合わせ プログラム相談	NAL 管理者 運用・技術支援者

4.14 セキュリティポリシ
(1) ユーザ
・登録ユーザ以外はログインさせない．
(2) NFS
CeNSS 及び CeViS のファイルシステムはユーザシステムへの NFS マウントは原則として行わない．なお，ユーザシステムのファイルシステムについては，CeNSS または CeViS に NFS READ マウントのみを許可する場合がある．
(3) r コマンド
r コマンドを使用したアクセスを許可しない．
(4) 通信
・NS システム外における通信は，基本的に暗号化する．
・所内イントラネットからのアクセスについては制限しない．所内イントラネットへの通信については，特定のサービスのみを通す設定とする．

4.15 ユーザ環境
(1) ログイン環境
1) ログインのための環境はセンタ側で用意する．
・共通環境 /etc/profile
・基本ドットファイル（.profile, .bash_profile, .bashrc）
・基本キー設定
2) 基本環境はセンタ側で用意する．
・初期環境変数設定
・初期シェル変数設定
・資源（CPU 時間，ファイル容量など）設定
→corefile サイズのみ 1MB に設定．スタックサイズ等は運用をみて変更．
3) その他のログイン環境はユーザ自身で設定する．
・エリアス，他のシェル，フリーウェアなどのドットファイル（.alias, .cshrc, .emacs）

4) メールは，CeNSS(CeViS)以外へのメールアカウントの.forward 設定を行い，CeNSS(CeViS)にはスプールしない.

(2) シェル環境

1) ログインシェル

・bash とする．（csh は仕様上，共用環境の定義ができずセンタ運用には適さないためログインシェルにはしない.）

・ログインシェルの変更は禁止とする.

2) センタのサポート方針

・sh 用の設定ファイル（/opt/NS/skel/local.profile）を用意する．（シェルスクリプト用，bash と共通の設定）

・bash 用の設定ファイル（/opt/NS/skel/local.bash_profile, /opt/NS/skel/local.bashrc）を用意する.

・センタ側で設定が必須と考えるもの（強制したいこと）は，/etc/profile へ記述する.

・センタとして必要であるが必須ではない設定は, /opt/NS/skel/local.bash_profile, /opt/NS/skel/local.bashrc に記述する．変更があった場合には，メールまたはウェブなどでその内容をユーザに通知し，変更の適用はユーザにまかせる.

・センタ提供のシェルスクリプトなどは sh をベースとする.

3) 他のシェル

・ログイン後の他シェルの利用については，ユーザの自由選択とする.

・（センタ側に余力がある場合のみ）他のシェル用の情報も提供する．ただし，責任は持たない.

(3) パス

・コマンドパス，マニュアルパス，ライブラリパス，その他運用上必要なパスは予め通しておく．NS シェル，NS コマンド用のパスは切っておく．→ /etc/profile 内に記述

・フリーウェア用のパスは通しておく．設定や利用はユーザにまかせる．→ .bash_profile 内に記述

(4) ファイル

・個人用 /home/x/x00, /large/x/x00

・個人用の標準 umask は 077 -rwx------

・その他のアクセス可能な dir

一般ユーザコマンド，センタ提供環境（/opt/NS），man ページ，フリーウェアの dir

・ファイルシステムは HSM 運用を行う．当面，テープ装置の quota 管理は行わない.

IO システム	ファイル システム	ユーザ使用領域	Quota および 期限 制限	階層型 ﾃﾞｰﾀ管理	用 途
CeMSS	/large	/large/x/x01	○	○	大規模ファイルの格納
	/ltmp	/ltmp/x/x01	○		大規模ファイルの一時的な格納
NAS	/home	/home/x/x01	○		ホーム領域 小規模ファイルの格納
	/jobtmp	/jobtmp/spool/x/x01 /jobtmp/cntl/x/x01	○		Nsub 実行結果の一時的な格納
	/stmp	/stmp/x/x01	○		小規模ファイルの一時的な格納

(5) コマンド系

・当面は UNIX の標準コマンド（正式には Solaris8 のコマンド）のみをサポート.

・新システム特有のコマンド（高速コピーhcp など）をサポートする予定.

・NS コマンド及び NS シェルをサポート.

(6) エディタ

・センタ側でサポートするエディタは vi のみとし，初期設定を行っておく.

・emacs は，ユーザが多いので自分で設定すれば使えるようにしておく.

(7) プログラム開発・実行

・当面はコマンドベースでの開発，実行環境を提供する．→ f90ns, qsub など

・FORTRAN バッファ

・スレッド並列数

・Parallelnavi Workbench による開発環境は，準備が整い次第，順次提供する.

(8) フリーウェアの利用

・当面は，センターとしてはサービスしない．サービスの仕方，維持管理の仕方などについて検討し，目処がついた後に順次提供する.

・現状で，問い合わせを受けた場合は，置いてある場所だけ教える．自分で使うのはユーザの自由とする．相談には応じない.

(9) リモート端末
・teraterm（vt100 用）などの基本端末エミュレータから telnet での利用環境をサポート．vt100 用の環境設定（TERM vt100）を予めしておく．
・キーマップの指定は行わない．
・TERMCAP, TERMINFO の設定は行わない．
・ssh の利用に対する設定
(10) X ウィンドウシステム
・xterm, kterm をサポートする．起動できるようにパスを切っておく．
・当面，CDE はサポートしない．
・motd による初期画面表示がきかない．
(11) プリント
・管理者用，センタ用として何台かネットワークプリンタを用意する．
・一般ユーザ向けに何台かのプリンタは用意しない．また，プリントファイルを作るためのツール（a2ps, dvi2ps とか）を用意する．
・ユーザが手元の WS，PC でプリントするための方法を指示する．
(12) 日本語
・当面は表示のみをサポートする．
・入力はサポートしない．

4.16 NS システムの遠隔利用
(1) 所内利用のサービス
・所内からは，telnet，ftp，X ウィンドウ，WANS によるアクセスをサポートする．
・ただし，r コマンドの利用は認めない．
(2) 所外利用のサービス
・所外からは，ssh 及び WANS によるアクセスをサポートする．
・ただし，セキュアノード利用者は，VPN によるアクセスをサポートする．
・ポートフォワーディング機能による X の利用は認めない．
(3) アクセス可能なシステムとホスト名
・所内外からアクセス可能なシステムは，CeNSS 及び CeViS とする．
・telnet，ftp によるアクセスは，emu または gogh とする．
・WANS によるアクセスポイントは，https://censs.nal.go.jp/jpn/webaccess.html とする．
(3) セキュリティ
・セキュリティは基本的に別途定めるセキュリティポリシーに従う．
・NS システムへアクセス権のあるユーザが，所外から hawk 経由で所内の WS にログインし，そこから CeNSS または CeViS にログインする方法は排除しないが，積極的には勧めない（アナウンスしない）．
(4) 認証
・サーバ認証及びクライアント認証は，PKI により行う．
・ユーザ認証は，Solaris UNIX 標準のパスワード認証により行う．

《以下省略》

（添付資料１）

数値シミュレータ（NS システム）の利用に関する規約等について

2002 年 6 月
CFD 技術開発センター

１．利用資格
（１）航技研職員（ただし，特別研究員，非常勤職員 A を含む.）
（２）航技研職員外の者で，非常勤職員 B，派遣職員，共同研究者，研修生等の当研究所と何らかの契約関係にある者（利用には申請者の所属する機関長の許可と印を必要とする.）
（３）CFD 技術開発センター長が認めた者

２．利用期限
（１）航技研職員の利用期限は特に設けない.
（２）航技研職員外利用者のシステム利用期限は当該年度内とし，年度を超えた利用継続希望者は申請書の再提出を必要とする．年度末に申請書の再提出を受けた場合のみ登録の継続，それ以外は削除を行う．年度途中に航技研での身分を失った場合には，その時点で利用登録を抹消する．また，利用者遵守事項及び責務に違反した場合にも利用登録を抹消する場合がある.

３．利用者の遵守事項
（１）NS システムを利用して得られた研究成果は，航技研 NS システム利用による研究成果である旨を論文等に必ず記載する．また，数値シミュレーション結果を視覚化した著作物を外部に公表する際には，航技研の著作物であることを明示するために文字ないしロゴを必ず添付する.
（２）NS システムの資源は有限であり，多くの利用者が共同で利用することを念頭に置き，お互いが快適な利用環境を維持できるよう心がける.
（３）他人（共同研究者を含む）の登録名（利用者 ID）を借用したシステムの使用は固くこれを禁ずる．また，利用者 ID の貸借は絶対に行わない.
（４）NS システムおよびネットワーク（外部ネットワークを含む）で接続される他の全ての計算機に対する不法行為はこれを固く禁ずる.
（５）ファイル領域等の長期確保や独占的利用は行わない.
（６）セキュリティ確保のためパスワードの管理は慎重に行い，安易なパスワードの使用は避ける.
（７）NS システム利用時は，航技研のネットワークを利用することになるので，ネットワークセキュリティポリシー，ネットワークセキュリティガイドラインの記述事項に従わなければならない.

４．利用者の責務
（１）システムを利用して得られた研究成果は，論文雑誌名等を記して別刷りを CFD 技術開発センターへ提出する．過去３年に渡り NS システムを利用しつつ一編の論文の提出もない場合には，登録の更新を行わないなどの処置を講ずる.
（２）年度末に NS システムの利用について CFD 技術開発センターへ報告書を提出する．特別利用者は，特別の事情を除いて航技研が主催するシンポジウムで成果を発表する義務がある.

５．利用申請
（１）NS システムの新規利用者は，NS システム利用申請書（添付１）に必要事項を記入の上，CFD 技術開発センター受付窓口に提出する.
（２）職員外の者は，利用申請書（添付２）の他に，航技研共用システム利用誓約書（添付３）及び利用の裏付けとなる契約書の写し（手続き中の場合は案でも可）を提出する．ただし，契約書の保管及び運用側の求めに応じた提出を，情報セキュリティ担当者の責任において保証する場合はこの限りではない.
（３）添付資料は，NAL イントラネット計算センターのウェブページからダウンロード可能.

６．利用登録，登録変更
NS システムの利用は，申請後，CFD 技術開発センターでの登録作業が終了した時点で可能とる．また，登録作業の完了は申請者に書面を以て通知する．利用登録は利用希望システム毎に申請する必要がある．利用期限内に利用者の身分に変更が生じた場合には，その旨を CFD 技術開発センターに速やかに報告し，登録変更を行う.

以上

付録 H 中央 NS システム（CeNSS）性能チューニングガイド Ver 1.0

平成 15 年 10 月
宇宙航空研究開発機構
情報技術開発共同センター
富士通株式会社

1．性能チューニングとは

CeNSS は，計算用 CPU が 1792 個，計算用ノードが 56 ノードから成る大規模システムです．とりわけ大規模プログラムや複雑なプログラムを高速に実行するためには，CeNSS のハードウェア・ソフトウェアの性能を最大限に引き出す作業が必要となります．このような作業を性能チューニングと言います．

1.1 プログラミングモデル

計算用ノードの中で，プログラムはプロセスを単位として動作します．XPFortran や MPI を使用すると，複数のプロセスに分割して実行することができます(これを「プロセス並列化」と言います)．また，自動並列や OpenMP を使用すると，プロセス内部の処理を複数のスレッドに分割して実行することができます（これを「スレッド並列化」といいます)．

プロセス並列化とスレッド並列化は組み合わせて使用することができます．図 1.1 にプログラミングモデルを示します．

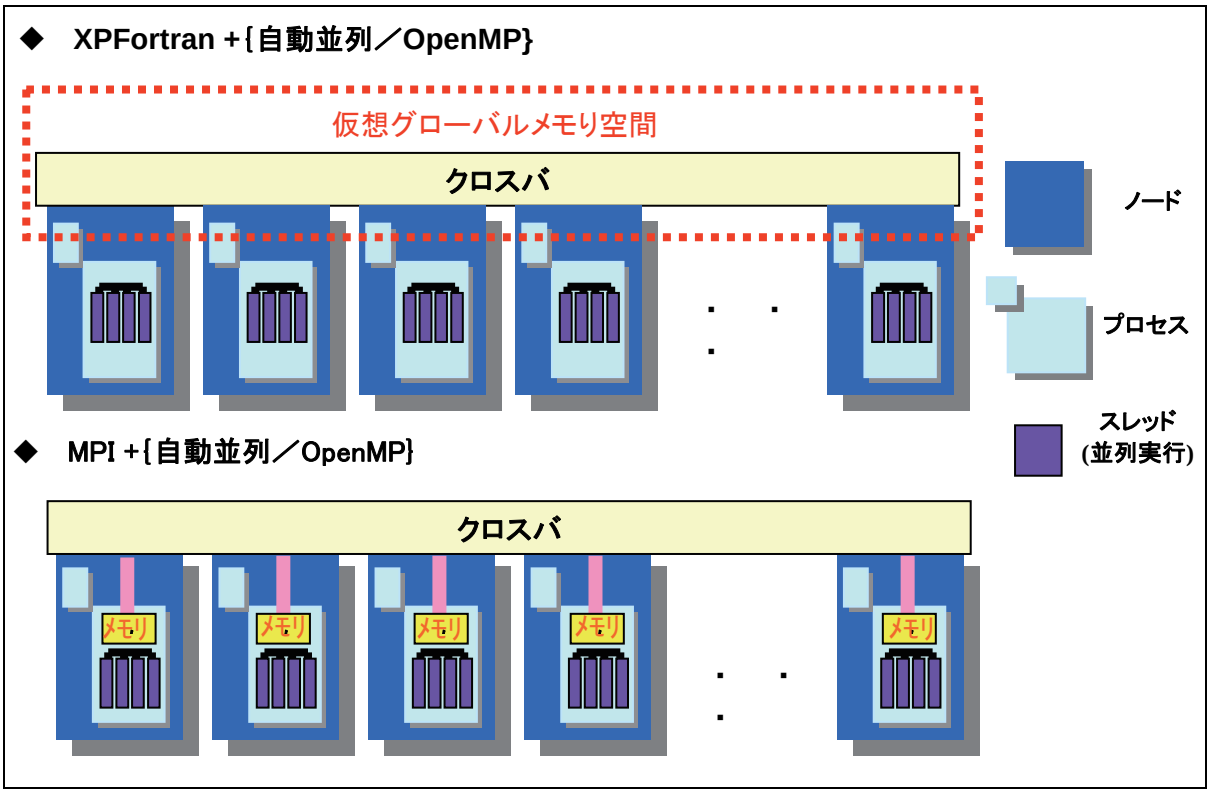


図 1.1 CeNSS のプログラミングモデル

1.2 性能チューニングの概要

性能チューニングは，前述のプログラミングモデルのどの部分を対象にするかによって，大きく 3 種類に分けられます．図 1.2 に性能チューニングの体系を示します．

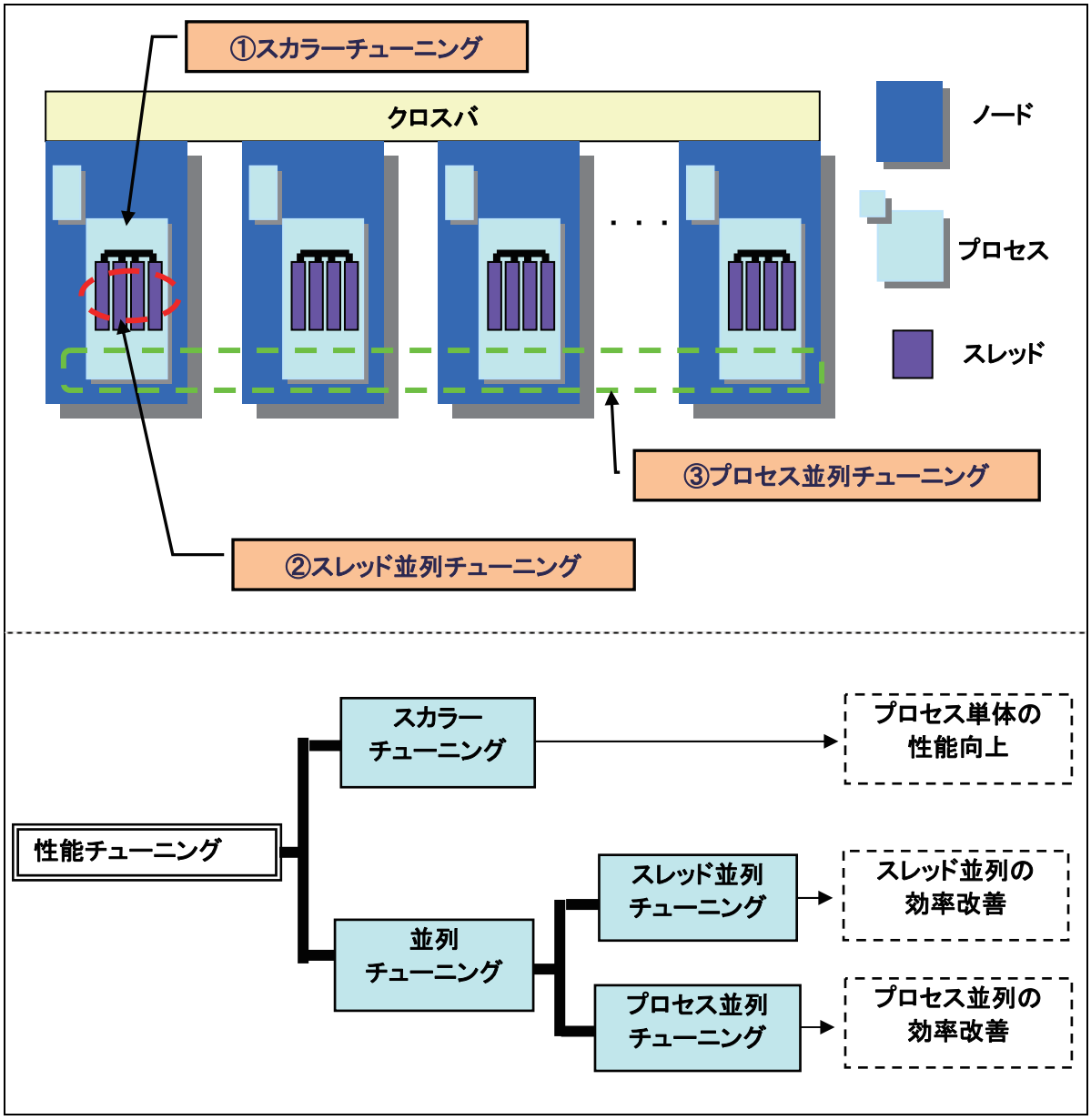


図 1.2　性能のチューニング体系

2. 性能チューニングのポイント

性能チューニングの種類ごとのポイントについて，以下に説明します.

2.1 スカラチューニングのポイント

① キャッシュミス率を下げる.

キャッシュメモリは，プロセッサと主記憶の間に置かれた比較的容量の小さな高速メモリです．主記憶から読み込んだデータをキャッシュメモリに保存することで，同じデータを再び利用する場合に，高速に読み込むことが可能となります．キャッシュメモリを階層的に組み合わせることにより，主記憶へのアクセスを高速化することができます．キャッシュメモリが有効利用されず，主記憶へのアクセスが発生することをキャッシュミスと言います．キャッシュミス率とは実行命令回数に対するキャッシュミス回数の割合の事です．キャッシュミス率が低いほど，性能としては高くなります．従って，キャッシュミス率を下げ，性能向上させるには，プログラムのデータアクセスを局所化し，主記憶に対してではなく，キャッシュメモリに対して行うことがポイントとなります.

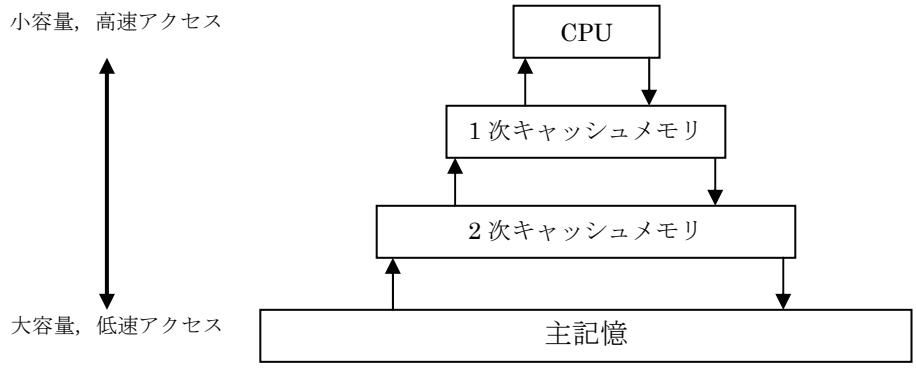


図 2.1 CPU とキャッシュメモリと主記憶の関係

② TLB ミス率を下げる.

プログラムの命令実行時には，論理的な仮想メモリアドレスから実メモリアドレスへの変換を行うために，アドレス変換テーブルを参照します．しかし，アドレス変換テーブルは主記憶上に存在するため，アクセスには時間がかかります．そこで，高速アクセスするために TLB（アドレス変換バッファ：Translation Lookaside Buffer）が使用されます．従って，TLB ミス率を下げ，性能向上させるには，仮想メモリアドレスから実メモリアドレスへの変換キャッシュ（TLB）を多用し，主記憶にアクセスせずに，高速なアドレス変換をさせることがポイントとなります．なお，大きな配列の全領域に渡ってデータアクセスが行われると，TLB ミスが多発し，性能劣化する場合があるため注意が必要です.

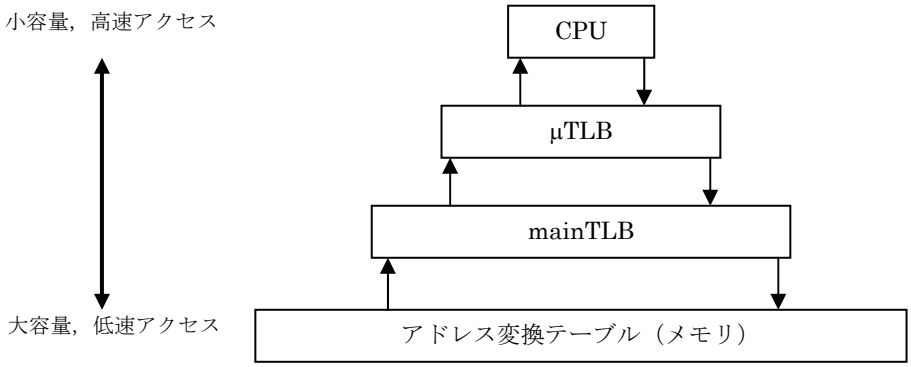


図 2.2 CPU と TLB とアドレス変換テーブルの関係

2.2 並列チューニングのポイント

●スレッド並列チューニング
- ・並列化率を上げる
- ・並列化粒度を上げる
- ・複数スレッド間のキャッシュ競合を回避する
- ・ロードバランスを均等化させる

●プロセス並列チューニング
- ・並列化率を上げる
- ・並列化粒度を上げる
- ・ロードバランスを均等化させる
- ・プロセス間のコストを削減させる

① 並列化率を上げる.

プログラムの並列化対象範囲が少ない状態で並列化部分の性能チューニングを行なっても，アムダールの法則などで言われる通り，あまり性能向上の効果は望めません.

従って，並列化率を上げ，性能向上させるには，並列化対象範囲を拡大し，その他の性能チューニングによる性能向上効果を最大限にすることがポイントです.

② 並列化粒度を上げる.

プログラム並列化を行なうことで，並列化処理のための同期処理のオーバヘッドが発生します.

従って，性能向上させるためには，並列化粒度を大きくすることにより相対的にオーバヘッドを小さくすることがポイントとなります.

③ 複数スレッド間のキャッシュ競合を回避する.

スレッド並列プログラムでは複数のスレッド間でのキャッシュ競合によりキャッシュミスが発生することがあります．このキャッシュミスが頻繁に発生すると性能が劣化します.

従って性能を向上させるためには，スレッド間のキャッシュ競合を少なくすることが重要になります.

④ ロードバランスを均等化させる.

並列プログラムにおいて，特定のプロセスに負荷の偏りが発生するとそのプロセスの処理時間が長くなり，結果的には，同期処理の時点でプログラム全体の経過時間延伸が発生することになります.

従って，性能向上させるためには，特定プロセスへの負荷集中（ロードバランス）はなるべく均等になるようにすることがポイントです.

⑤ プロセス間通信のコストを削減する.

並列プログラムにおいて，プロセス間通信は比較的時間のかかる（コストの高い）処理です.

従って，性能向上させるためには，プロセス間通信の回数およびその量を削減したり，演算処理と重ね合わせるなど，通信コストを削減することがポイントです.

3. 性能チューニングの流れ

性能チューニング作業の流れについて説明します.

性能チューニング作業は以下のフェーズを繰り返して実施することによって行います.

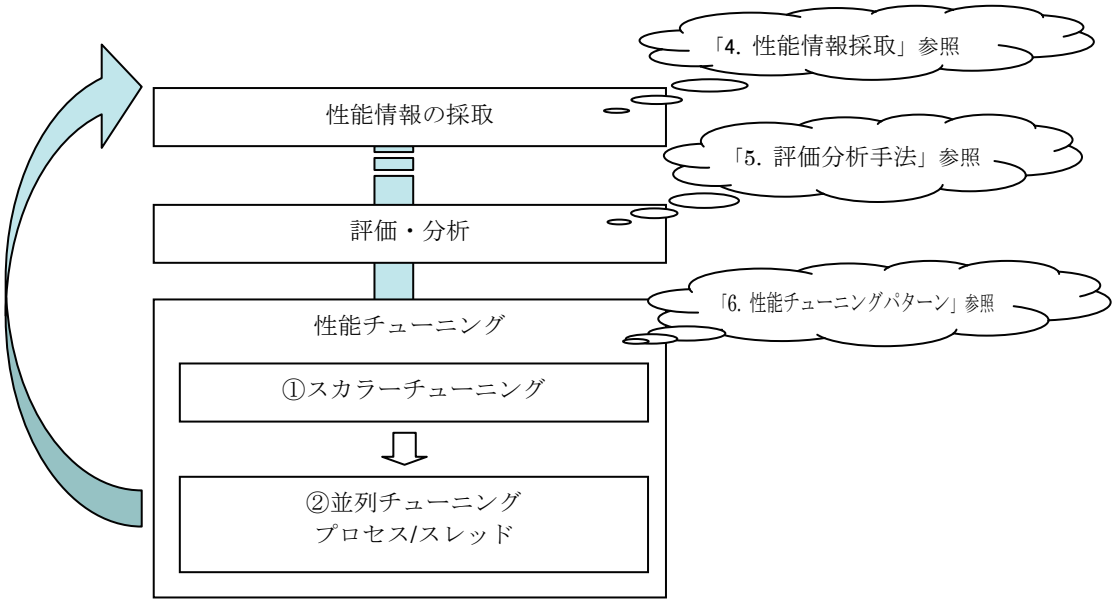


図 3.1 性能チューニングの流れ

3.1 性能情報の採取フェーズ

性能情報の採取では，以下の作業を行います.

①プロファイラと呼ばれる性能分析ツールを使用して，プロファイラ情報を採取します.

③ ロファイラ情報から，性能情報を抜き出します.

詳細は「4 性能情報採取」で説明します.

3.2 評価・分析フェーズ

評価では，以下の作業を行います.

①十分に性能が出ているか，後述する性能チューニングの目安値と，性能情報を見比べて評価します.

②十分に性能が出ていると判断されれば性能チューニングは終わりです．性能が出ていないと判断されれば，ソースコード，プロファイラ情報から原因を分析し，性能チューニング作業を行います.

詳細は「5 評価分析手法」で説明します.

3.3 性能チューニング

性能チューニングでは，以下の作業を行います.

①実行の効率化を図ることを目的として，そのソースコードを修正します．修正は情報分析で調べたホットスポット（実行時間の多い箇所）に対して行います.

詳細は「6. 性能チューニングパターン」で説明します.

4. 性能情報採取

4.1 プロファイラの概要

性能分析を定量的に行うため有用なツールとして，プロファイラと呼ばれるソフトウェアが提供されています．

ここでは，プロファイラの概要について説明します．

なお，プロファイラには GUI 版とコマンドライン版の 2 種類が提供されていますが，今回はコマンドライン版をベースに説明します．GUI 版については，「プログラミング支援ツール使用手引書」をご参照ください．

(1)機能概要

プロファイラとは，プログラミング支援ツールの性能チューニング機能の一部で，逐次および並列プログラムの実行性能解析を支援するツールです． プロファイラには，以下に示す機能があります．

- ◆ サンプリング方式による CPU 負荷解析
- ◆ ハードウェアモニタ情報
- ◆ プロセス間通信データ転送状況

サンプリング方式とは，プログラムの実行中に一定 CPU 時間間隔で割込みを発生させ，その時のプロセス，スレッド，手続き，ループおよび行の状況を測定するものです．

ハードウェアモニタ情報とは，プログラムの実行中にハードウェアカウンタで測定する CPU の詳細な性能情報です．

プロセス間通信データ転送状況とは，プロセス並列実行におけるプロセス間通信時間を測定するものです．

(2)動作概要

プロファイラは，ユーザが以下の手順を段階的に行うことにより使用します．

【翻訳】対象プログラムをプロファイラ向けに翻訳し，実行モジュールを作成します．

【実行】対象プログラムを実行し，収集した性能情報をプロファイラ情報ファイルに格納します．

【解析】解析コマンドを起動し，プロファイラ情報ファイルの解析結果を出力します．

4.2 プロファイラによる情報採取手順

コマンドライン版プロファイラで性能情報の解析結果を採取するまでの手順を以下に説明します.

(1)翻訳

プログラムを翻訳するとき, 通常の翻訳時オプションに加えて, オプション"-Ktl_trt"を指定してください. このオプションは, 実行ファイルを作成するときに指定する必要があります.

```
例）f90ns -Uxpf -Ktl_trt test.f90
```

(2)実行

プロファイラ用の環境変数を設定して, プログラムを実行してください.
主な環境変数の一覧を以下に示します.

表 4.1 プロファイラの環境変数

環境変数名	内容
TRT_ENV	プロファイラ情報を収集することを指示する. "PMP=on"を指定する.
PROF_STATS	性能チューニング情報の収集項目を指定する. コマンドライン版プロファイラの場合, 以下の項目の情報を収集可能. 収集する項目(複数指定可能)に対応する値の合計値を指定する. 1 : プロセス間通信情報, ユーザ DTU 動作情報 2 : ハードウェアモニタ情報 4 : サンプリング情報 256 : PA サービスサブルーチンによるハードウェアモニタ情報 プログラム全体の性能情報をまとめて採取する場合には, 7 を指定する.
PROF_PA	ハードウェアモニタ情報の収集項目を指定する. 指定可能な値は以下の通り(複数指定可能). sta : CPU 時間, 実行命令数, 命令実行効率, 2 次キャッシュミス率, 浮動小数点演算命令実行効率, 命令 main TLB ミス率, データ main TLB ミス率 性能チューニング指標値の分析を行う場合は"sta"を指定する.
PROF_SAMPINTVAL	プロファイラ情報を収集する時間間隔[単位:ミリ秒]を指定する. 省略値=10 ミリ秒. 実行時間が 10 分以上の場合は, 目安として 100(ミリ秒)を指定する.
PROF_MEMORY_SIZE	プロファイラの作業領域[単位:KByte]を指定する. プロファイラ作業領域は, PROF_SAMPINTVAL で指定した時間間隔あたり 128bytes 必要. 実行時間が 10～20 分の場合は, 目安として 262144(256MB)を指定する.

一般的な環境変数の指定例は以下の通りです.

```
TRT_ENV="PMP=on"                ;export TRT_ENV
PROF_STATS=7                    ;export PROF_STATS
PROF_PA=sta                     ;export PROF_PA
PROF_MEMORY_SIZE=262144         ;export PROF_MEMORY_SIZE
```

実行が完了すると, 実行時のカレントディレクトリにプロファイラ情報ファイルが作成されます.

表 4.2　プロファイラ情報ファイル名

ファイル名	概要
GProf_*jobid.file_id*.pri	ジョブ全体情報 ジョブごとに1個作成される.
DProf_*jobid.taskid.file_id*.pri	サンプリング情報，ハードウェアモニタ情報 ジョブのプロセス数だけ作成される.

jobid ：ジョブ番号
taskid ：プログラム実行時のタスク番号
file_id ：ファイル識別番号

※プロファイラ情報を取得するためにジョブを実行する場合，通常よりも実行時間が増大します.

※スレッド並列の情報を収集する場合，プロセスあたりの割当て CPU 数を 1 個増やす必要があります.
（プロセス並列だけの場合，増加は不要です）

例）8 プロセス並列×4 スレッド並列実行の場合

```
#!/bin/sh
/bin/qsub   << !CeNSS
#@¥$-s    /bin/sh
#@¥$-r    censsjob.01-01
#@¥$-q    QJOB
#@¥$-lM   512mb
#@¥$-lP   8
#@¥$-lp   5
#@¥$
#!/bin/sh
cd /home/x/x01
PARALLEL=4                      ; export PARALLEL
OMP_NUM_THREADS=4               ; export OMP_NUM_THREADS
TRT_ENV="PMP=on"                ; export TRT_ENV
PROF_STATS=7                    ; export PROF_STATS
PROF_PA=sta                     ;export PROF_PA
PROF_MEMORY_SIZE=262144   ;export PROF_MEMORY_SIZE

/home/x/x01/a.out
!CeNSS
```

プロセスあたりの割当て CPU 数を
実行スレッド数+1 で指定

図 4.3　8 プロセス 4 スレッド並列の実行シェル

(3)解析

(2)で収集したプロファイラ情報を解析するためのコマンドとして，mpprof コマンドを使用します.
使用方法は以下の通りです.

形式：`mpprof` *`profiler-data-file`* `[options]`
profiler-data-file：実行時に生成されたファイルのいずれか 1 つを指定.
※解析結果はテキスト形式で標準出力に出力される.

例）`mpprof DProf_14764.000.prof.pri >prof.out`
プロファイラ情報ファイルを解析し，結果を `prof.out` というファイルに格納する.

4.3 プロファイラ採取情報の見方

コマンドライン版プロファイラで採取した性能情報の分析手順を以下に説明します.

(1)プロファイラ解析結果の出力形式

mpprof コマンドで出力される，一般的なプロファイラ解析結果の構成は以下の通りです.

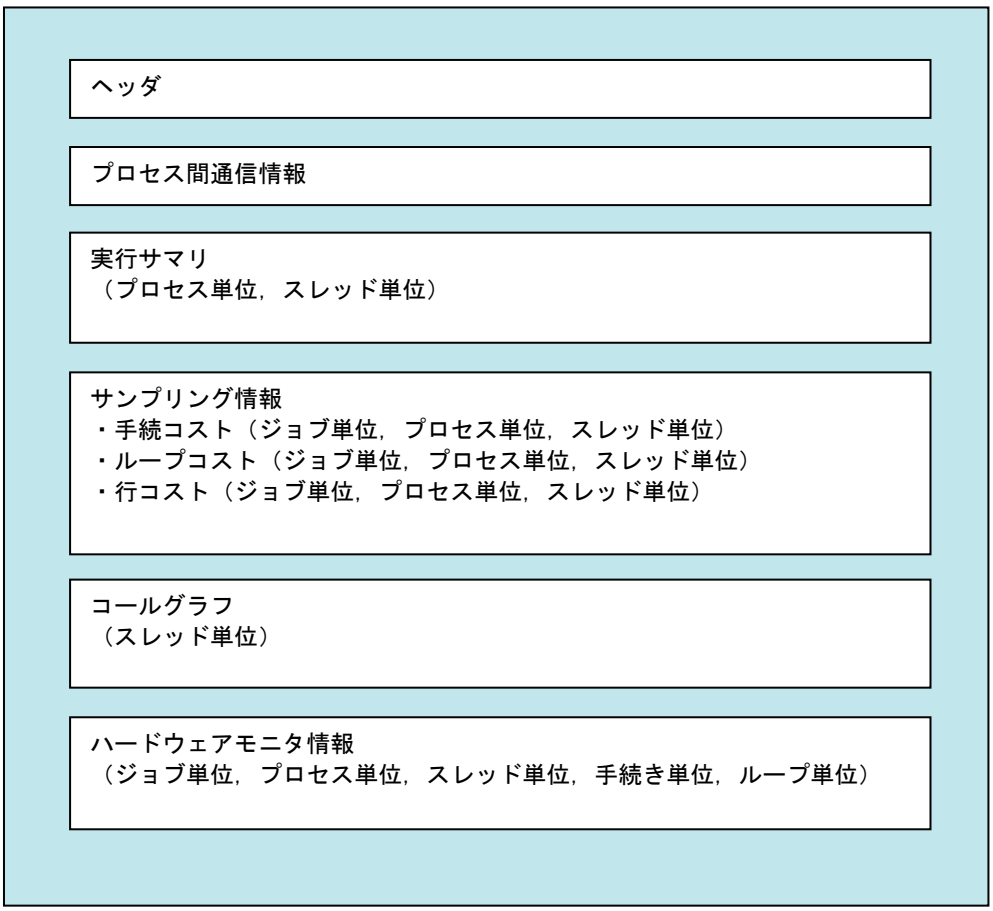


図 4.4 プロファイラ解析結果の構成

表 4.5 プロファイラ解析結果の構成

種別	概要
ヘッダ	実行日付，ジョブの種別
プロセス間通信情報	各プロセスの送信，受信および通信完了待ち時間
実行サマリ	経過時間，CPU 時間などの集計結果
サンプリング情報	サンプリング数を手続き，ループおよび行ごとに詳細化した結果
コールグラフ	手続きの呼出し経路のコスト(サンプリング数)をスレッドごとに編集した結果
ハードウェアモニタ情報	ハードウェアカウンタ情報を，手続きおよびループごとに詳細化した結果

(2)性能情報について

プロファイラの解析結果のうち，性能チューニングの指標となる代表的な性能情報は以下の通りです.

表 4.6　性能情報

性能情報	内容
MIPS 値	命令の実行効率 （命令実行数÷CPU 時間÷1e+6）
MFLOPS 値	浮動小数点演算命令実行速度 （浮動小数点演算命令数÷CPU 時間÷1e+6）
２次キャッシュミス率	2 次キャッシュのミスヒット率 （2 次キャッシュミス回数÷命令実行数）[%]
TLB ミス率	データメイン TLB のミスヒット率 （データメイン TLB ミス回数÷命令実行数）[%]

これらの情報はハードウェアモニタ情報から参照できます．以下に出力例を示します.

Hardware Monitor

Statistics Performance

CPUS	Commit	MIPS 値 MIPS	MFLOPS 値 MFLOPS	２次ｷｬｯｼｭﾐｽ率[%] L2-miss	mTLB-ir	ＴＬＢﾐｽ率[%] mTLB-or	
1.005655e+02	1.170257e+11	1.163677e+03	5.336309e+02	0.5409	0.0000	0.0000	Process 0
1.023455e+02	1.275961e+11	1.246719e+03	5.721569e+02	0.5274	0.0000	0.0000	Process 1
1.023942e+02	1.278711e+11	1.248812e+03	5.739716e+02	0.5222	0.0000	0.0000	Process 2
1.024329e+02	1.177806e+11	1.149831e+03	4.761365e+02	0.4848	0.0000	0.0000	Process 3
1.094746e+02	4.902736e+11	4.478424e+03	2.007461e+03	0.5190	0.0000	0.0000	Process Total

図 4.7　ハードウェアモニタ情報出力例

また，ハードウェアモニタ情報およびサンプリング情報は，手続き単位，ループ単位などでも出力されます．

サンプリング情報出力例

```
Execution Profile

                                          ソース上の開始／終了行番号
*** Global Profile
コスト分布
                        Sampling Cost
  ------------------------------------------------------
        Samples   % Run       Barrier     Start      End
  ------------------------------------------------------
    4.539000e+03   10.69  1.532000e+03      1811     2141  tvd._OMP_3_
    3.889000e+03    9.16  2.191000e+03      1445     1771  tvd._OMP_2_
    3.221000e+03    7.58  1.211000e+03      1082     1435  tvd._OMP_1_
    2.472000e+03    5.82  4.500000e+02       431      777  xy_cg1._OMP_1_
    2.434000e+03    5.73  4.710000e+02       318      777  xx_cg1._OMP_1_
    2.372000e+03    5.59  0.000000e+00         -        -  _LowLevelSpinMulti
    2.358000e+03    5.55  4.500000e+02       524      777  xz_cg1._OMP_1_
    2.321000e+03    5.47  0.000000e+00         -        -  prbar_probe
    1.299000e+03    3.06  0.000000e+00         -        -  _LowLevelWhileNotEqual
    8.060000e+02    1.90  0.000000e+00       207      219  qcx_cg1._PRL_1_
```

ハードウェアモニタ情報出力例

```
Hardware Monitor   By Symbols

Statistics Performance
       CPU         Commit          MIPS         MFLOPS      L2-miss   mTLB-ir   mTLB-or
  ------------------------------------------------------------------------------------
  4.478298e+01  3.540519e+10  7.905948e+02  2.991396e+02    0.2721    0.0000    0.0000  tvd._OMP_3_
  3.833003e+01  4.506079e+10  1.175600e+03  5.703916e+02    0.4144    0.0000    0.0000  tvd._OMP_2_
  3.176930e+01  2.493328e+10  7.848231e+02  2.290272e+02    0.5423    0.0000    0.0000  tvd._OMP_1_
  2.422000e+01  1.973925e+10  8.149980e+02  5.214672e+02    0.4548    0.0000    0.0000  xy_cg1._OMP_1_
  2.379628e+01  1.703009e+10  7.156618e+02  4.308993e+02    0.6560    0.0000    0.0000  xx_cg1._OMP_1_
  2.374581e+01  3.461685e+10  1.457808e+03  7.957734e+00    0.0196    0.0000    0.0000  _LowLevelSpinMulti
  2.309820e+01  1.890118e+10  8.182970e+02  4.952270e+02    0.4573    0.0000    0.0000  xz_cg1._OMP_1_
  2.279917e+01  2.307417e+10  1.012062e+03  3.111546e+02    0.4030    0.0000    0.0000  prbar_probe
  1.281160e+01  1.641253e+10  1.281067e+03  3.504462e+02    0.2491    0.0000    0.0000  _LowLevelWhileNotEqual
  7.858402e+00  1.029958e+10  1.310645e+03  5.077087e+02    2.2860    0.0000    0.0000  qcx_cg1._PRL_1_
```

図 4.8　ハードウェアモニタ情報とサンプリング情報

性能情報と行番号を組合わせて，ソース上の位置を特定できるようになっています．

5. 評価分析手法

5.1 性能分析の概要

性能分析の目的は，プログラム中のホットスポットを探すことにあります.

以下のアムダールの法則から明らかなように，効率よく性能を向上させるには，ホットスポットを集中的に性能チューニングする必要があります.

アムダールの法則(Amdahl's law)

プログラム全体の実行時間の a（割合：0 □a□ 1）にあたる部分を n 倍向上させた場合の
プログラム全体の性能向上比は以下の通り.

$$\text{性能向上比} = \frac{1}{(1-a)+a/n}$$

【例】

・プログラム全体時間の **90%**にあたる処理を **10 倍**性能向上 → 性能向上比＝約 **5.26 倍**

・プログラム全体時間の **30%**にあたる処理を **10 倍**性能向上 → 性能向上比＝約 **1.37 倍**

図 5.1 アムダールの法則

したがって，性能チューニングを行うにあたって，プログラムの性能情報を採取し，ホットスポットの特定および分析を行うことが重要になります.

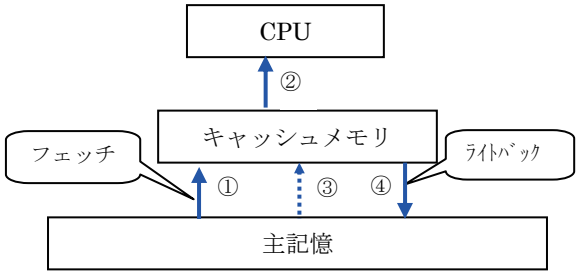
5.2 スカラー性能の分析手法

スカラー性能分析では，キャッシュミス率および TLB ミス率の状況を把握し，性能チューニングの目安値と比較し，状況に応じたコンパイルオプションやソース変更による性能チューニングを行なうことになります．

状況分析および対処を行なうためには，以下の問題発生原因を理解しておく必要があります．

キャッシュミスの主な発生原因

（1）キャッシュメモリが容量不足の場合



・データがキャッシュに無い場合，主記憶からキャッシュへデータが転送され（①），キャッシュからＣＰＵへデータが転送される（②）．
・キャッシュに別のデータがあってもデータが更新されていなければ，主記憶からキャッシュへデータが転送され（①），キャッシュから CPU へデータが転送される（②）．
・キャッシュにデータがある場合は，主記憶からキャッシュへのデータ転送は行われず（③），キャッシュからのみデータがＣＰＵへ送られる（②）．
・キャッシュに別のデータがり，かつデータが更新されていれば，キャッシュから主記憶へのライトバックが発生する（④）．その後，主記憶からキャッシュへデータが転送され（①），キャッシュから CPU へデータが転送される（②）．

図 5.2 キャッシュメモリが容量不足の時の主記憶へのアクセス

例）以下のような演算に対してデータアクセスが多い場合，キャッシュミス率が高くなります．
次の例では，nmax の値が極端に大きいとキャッシュミス率が高くなります．

```
do j=1,nmax
  do i=1,nmax
    a(i,j) = a(i,j) * b(i,j) + c * d(i,j)
  end do
end do
do j=1,nmax
  do i=1,nmax
    a(i,j) = a(i,j) * b(i,j) + e * f(i,j)
  end do
end do
```

図 5.3 演算に対するデータアクセスが大きいパターンでのキャッシュミスの例

（2）キャッシュメモリの使用に偏りがある場合

ループ構造によってはキャッシュメモリの使用に偏りが出る場合があります.

キャッシュメモリは，64 バイト毎の領域(キャッシュライン)に分けられており，領域単位で更新が行われます．使用される領域に偏り(集中)があると，キャッシュミス率が高くなります.

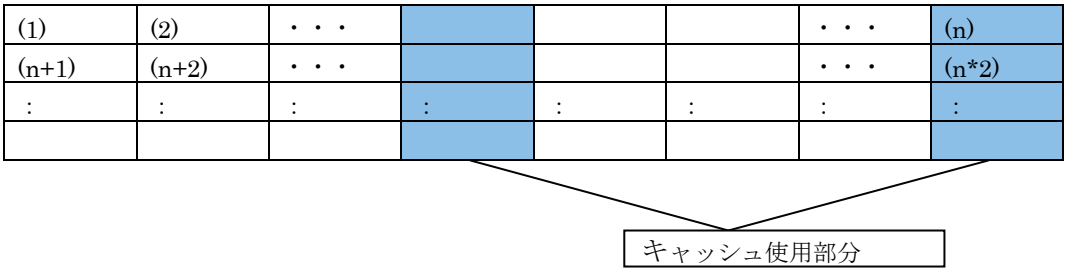


図 5.4　キャッシュメモリイメージ

TLB ミスの主な発生原因

（1）ループ内の大きなストライドアクセスの場合

ループにおいて大きなストライドアクセスになっていると，ＴＬＢミスが発生しやすくなります.

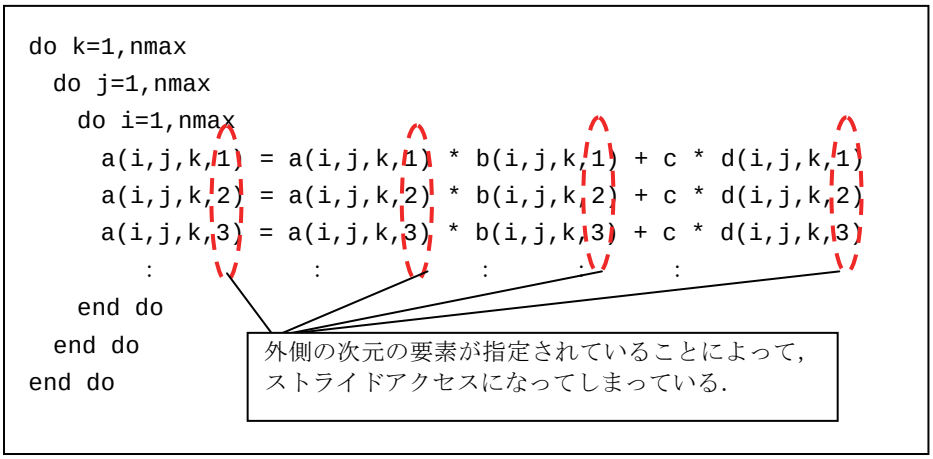


図 5.5　ストライドアクセス幅が大きいパターンでの TLB ミスの例

5.3 並列性能の分析手法

並列性能分析では，並列化対象範囲，並列実行時の演算粒度・ロードバランス・プロセス間通信コストなどの状況を把握し，性能チューニングの目安値と比較し，状況に応じたコンパイルオプションやソース変更による性能　チューニングを行なうことになります.

状況分析および対処を行なうためには，以下の問題発生原因を理解しておく必要があります.

自動並列化が阻害される主な原因

- 並列実行しても実行時間が短縮されないと予測される場合
- ループスライスの対象とならない型の演算を含む場合
- 外部手続き，内部手続きまたはモジュール手続きの引用を含む場合
- ＤＯループの型が複雑な場合（飛込みや飛出しがある場合など）
- 入出力文または組込みサブルーチンを含む場合
- データの定義引用順序が逐次実行の時と変わるおそれが有る場合

プロセス／スレッド並列効果が阻害される主な原因

- 特定のプロセスもしくはスレッドに処理量が偏る場合（IF 文や I/O 処理などで偏る場合など）
- プロセスもしくはスレッドにおける分割並列内の演算量（演算粒度）が少ない場合
- 複数スレッド間でキャッシュ競合が発生する場合（False Sharing）
- プロセス間通信の回数および１回辺りの通信量が多い場合

なお，False Sharing の発生原因を以下に示します.

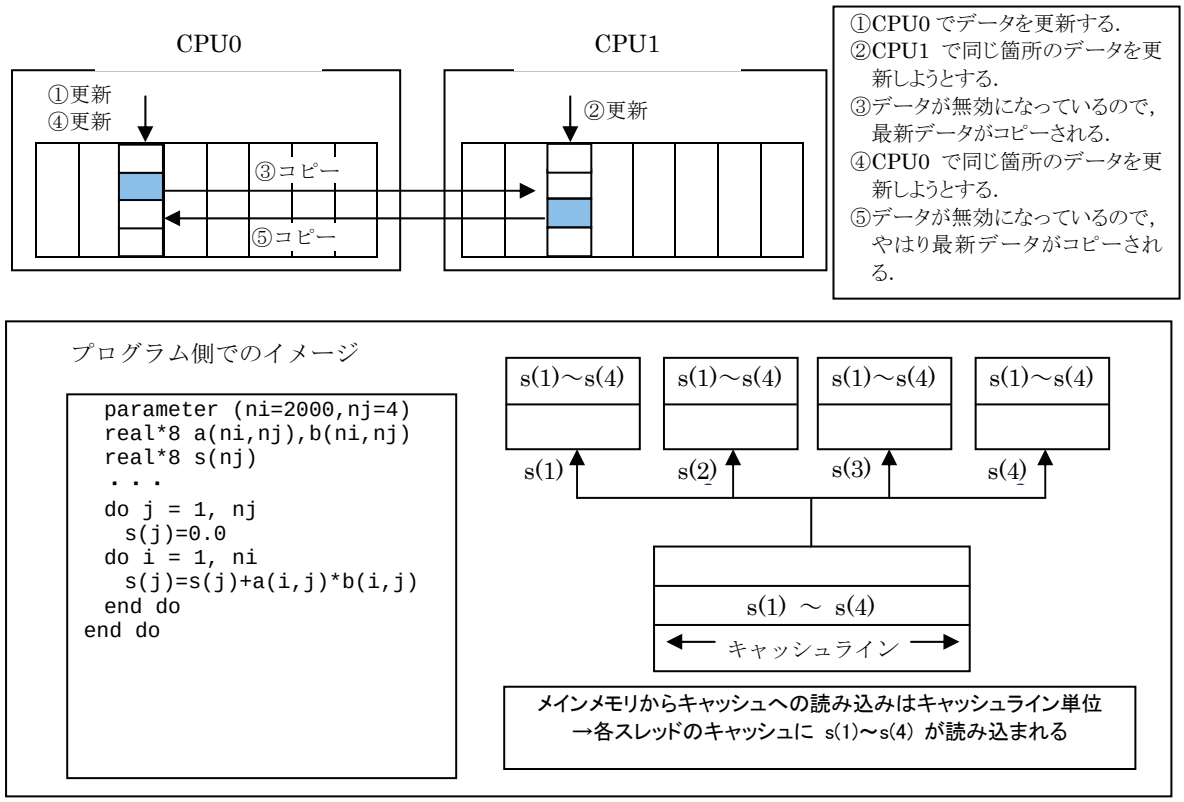


図 5.6　他の CPU で更新されたデータに対する主記憶へのアクセス

5.4 性能チューニングの目安

性能チューニングの際の目安を下記に示します．下記の値をクリアしていれば性能は十分でていると判断してよいでしょう．

表 5.7　性能チューニングの目安

指標	目安	備考
2 次キャッシュミス率	1.0 以下	
TLB ミス率	0.02 以下	限りなく 0 に近い
MIPS 値	1,000 以上	1CPU あたり
MFLOPS 値	400 以上	1CPU あたり
スレッド並列における台数効果	4 並列なら 1 並列の場合の 3.2 倍以上 2 並列なら 1 並列の場合の 1.6 倍以上	台数効果＝1 スレッド時の経過時間÷経過時間

6. 性能チューニングパターン

本章では，CeNSS システムにおいて有効な性能チューニングのパターンについて，代表的な例を取り上げて紹介します．

一般的に，ソース性能チューニングの種類は，以下の 3 つに大別できます．

- スカラー(非並列)チューニング
- スレッド並列チューニング
- プロセス並列チューニング

性能チューニングは，プログラムの実行性能を向上させることですが，実際は実行性能を阻害する要因を探し，それを除去していくという作業が中心となります．それぞれの性能チューニングパターンについて，代表例を紹介します．

6.1 スカラーチューニング

スカラーチューニングとは，非並列(1CPU)で実行した場合の性能向上を狙ったプログラムの高速化です．スカラー性能を阻害する最も大きな要因は，キャッシュミスや TLB ミス等の主記憶へのアクセス効率の悪化によるものです．

スカラー性能を上げるための性能チューニング手法と，主な性能チューニング内容は以下の通りです．

1．2 次キャッシュミスの削減
2．TLB ミスの削減

それぞれについての，代表的な性能チューニング例を紹介します．

6.1.1 2次キャッシュミスの削減

２次キャッシュミスの削減には，データアクセスパターン効率の改善が有効です．具体的改善方法としては，ループの再構成や，配列次元入れ替え等が挙げられます．

以下は，２次キャッシュミス改善に関する例です．

(1)ループ交換（loop interchange）

ループの交換（interchange）とは，多重ループに対して，do 文の順序を入れ替えることです．

ループ交換により，データアクセスが連続になり，結果的に主記憶へのアクセスの効率化を実現できます．

以下に，ループ交換の例を示します．

変更後	変更後
do 500 n = 1, neq do 500 k = 1, kmax(m) do 500 l = 2, lmax(m)-1 do 500 j = 2, jmax(m) q1F(n,j,k,l) = qoF(n,j,k,l) + alp*gts* & rhsF(n,j,k,l)*voli(j,k,l) 500 continue	**do 500 l = 2, lmax(m)-1** **do 500 k = 1, kmax(m)** **do 500 j = 2, jmax(m)** **do 500 n = 1, neq** （連続アクセスとなるように，ループを交換） q1F(n,j,k,l) = qoF(n,j,k,l) + alp*gts* & rhsF(n,j,k,l)*voli(j,k,l) 500 continue

図 6.1 ループ交換

●注意事項

・コンパイラによるループ交換

コンパイラの最適化により，自動的にループ交換される場合があります．

コンパイルオプション –Et により， コンパイラによるループ交換を診断メッセージで確認することができます．

以下に，翻訳例と，診断メッセージの出力例を示します．

（翻訳例）

```
% f90ns  -Uauto  -Et  Sample.f90
```

（診断メッセージの例）

```
jwd8211i-i  "Sample.f90", line 33: ループを入れ換えました.(行:34)
```

(2)ループ展開（loop unrolling）

多重ループの場合，外側のループを展開することにより，主記憶への参照回数が減少し，性能が向上する場合があります.

以下に，多重ループの外側のループ展開を行う例を示します.

展開前

```
do j = 1, n
  do i = 1, n
    a(i) = a(i) + b(i) * c(i,j)
  enddo
enddo
```

展開後

外側ループを4回展開

```
!ocl unroll(4)
   do j = 1, n
   do i = 1, n
     a(i) = a(i) + b(i) * c(i,j)
   enddo
 enddo
```

展開後の処理イメージ

```
do j = 1, n, 4
  do i = 1, n
    a(i) = a(i) + b(i) * c(i,j)   .... (1)
    a(i) = a(i) + b(i) * c(i,j+1) .... (2)
    a(i) = a(i) + b(i) * c(i,j+2) .... (3)
    a(i) = a(i) + b(i) * c(i,j+3) .... (4)
  enddo
enddo
```

図 6.2　ループ展開

展開前では，a(i) のロード，b(i) のロード, c(i,j) のロード，そして a(i) のストアで，1 文あたりのロードストア数は，3 ロード, 1 ストアとなります.　外側ループ 4 回展開後では,

(1)　a(i) ，b(i) ，c(i,j) のロード
(2)　c(i,j+1) のロード　　[a(i)，b(i) は (1) でロード済み]
(3)　c(i,j+2) のロード　　[a(i)，b(i) は (1) でロード済み]
(4)　c(i,j+3) のロード，a(i) のストア　　[a(i)，b(i) は (1) でロード済み]

となり，4 文で計 6 ロード，1 ストアとなります.
したがって 1 文あたりの平均ロードストア数は，1.5 ロード，0.25 ストアとなり，主記憶への参照回数が減少します.

●注意事項

・コンパイラによるループ展開

コンパイラの最適化により，自動的にループ展開される場合があります.
コンパイルオプション -Eu により，コンパイラによるループ展開を，診断メッセージで確認することができます.　以下に，翻訳例と，診断メッセージの出力例を示します.

（翻訳例）

```
% f90ns  -Uauto  -Eu  Sample.f90
```

（診断メッセージの例）

```
jwd8202i-i  "sample.f90", line 55: ループを展開数 4 回でアンローリングしました.
```

・展開が不適当なループについて

分岐を含んでいたり，サブルーチン呼び出しがあったりする 複雑なループは，ループ展開には不適当です.

(3)ループ融合（loop fusion）

ループの融合（fuse）とは， 複数のループを一つのループにまとめることです.

その目的の１つはループの展開の場合と同様です．さらに重要なのは，多重ループを融合させることによる，データアクセスの局所化です.

以下にループ融合の例を示します.

融合前	融合後

do i = 1, n
  a(i) = a(i) + b(i) * c
enddo
 	
do i = 1, n
  d(i) = a(i) + b(i) * e
enddo
	do i = 1, n
	  a(i) = a(i) + b(i) * c
	  d(i) = a(i) + b(i) * e
	enddo

図 6.3　ループ融合 1

ループ中でワーク用の配列を使用していて，ループ融合により配列次元を共通化できる場合，不要になった次元を削除することにより，データアクセスも局所化できます.

融合前	融合後
dimension A(JMAX,KMAX) ・・・ do 10 k=2,km do 10 j=1,jm ・・・ A(J,K) = R1*A(J,K)+R2*A(J+1,K) ・・・ 10 continue do 20 k=2,km do 20 j=1,jm ・・・ S(J,K,L) = R0*A(J,K) ・・・ 20 continue	dimension A(JMAX) ・・・ do 100 k=2,km ←kのループを共通化 do 10 j=1,jm ・・・ A(J) = R1*A(J)+R2*A(J+1) ・・・ 10 continue do 20 j=1,jm ・・・ S(J,K,L) = R0*A(J) ・・・ 20 continue 100 continue

図 6.4　ループ融合 2

さらにループ融合を進めて，全ての配列次元を共通化できる場合，配列次元を全て除去してスカラ変数化することにより，さらにデータアクセスの局所化を促進できます.

融合前	融合後
dimension A(JMAX,KMAX) ・・・ do 10 k=2,km do 10 j=1,jm ・・・ A(J,K) = R1*A(J,K)+R2*A(J,K) ・・・ 10 continue do 20 k=2,km do 20 j=1,jm ・・・ S(J,K,L) = R0*A(J,K) ・・・ 20 continue	・・・ do 10 k=2,km } j,k のループを共通化 do 10 j=1,jm ・・・ A = R1*A+R2*A ・・・ S(J,K,L) = R0*A ・・・ 10 continue

図 6.5　ループ融合 3

●注意事項

・コンパイラによるループ融合

コンパイラの最適化により，自動的にループ融合を行う場合があります.

コンパイルオプション -Ef により，コンパイラによるループ融合を，診断メッセージで確認することができます.

以下に，翻訳例と，診断メッセージの出力例を示します.

（翻訳例）

```
% f90ns  -Uauto  -Ef  Sample.f90
```

（診断メッセージの例）

```
jwd8210i-i  "Sample.f90", line 6: ループが融合されました.(行:10)
```

・融合が不適当なループについて

融合前の複数のループのなかに，共通する配列参照がない場合は，性能向上は望めません.

(4)配列の次元入れ換え

データアクセスのストライド幅が広い配列を扱う場合，配列の次元入れ換えによりメモリアクセスの効率を上げ，L2 ミスを削減できます．配列の次元入れ換えは，TLB ミスの削減にも有効であり，詳細は 1.5.1.2 の TLB ミス削減の章で説明します.

6.1.2 TLB ミスの削減

TLB ミスとは，仮想アドレスから物理アドレスへの変換時に，ページを管理するテーブルのキャッシュ(TLB)がヒットしないことを指します.

CeNSS システムでは，ラージページ機能を備えていますが，ストライド幅の大きな配列を一度に多く参照すると，TLB ミスが多発する場合があります．このようなケースでは，配列次元入れ替えにより TLB ミスを削減することが出来ます．また，同様の手法で２次キャッシュミスも削減できます.

以下に，配列次元入れ換えによる TLB ミスの削減方法について説明します.

以下の例では，q1(jp,k,lp,1)～q1(jp,k,lp,15)への参照が同時に行われているため，データアクセスのストライド幅が広く，主記憶へのアクセスの効率が悪いことが分かります．このような場合，

ql(a, b, c, d) □ ql(d, a, c, b)

という配列の次元入れ替えを行うことにより，主記憶参照の局所性を高めることができます.

配列の次元入れ替えを行う際，プリプロセッサの機能を利用することで，ソース修正の手間を省くことが出来ます.

プリプロセッサの機能を用いて，#define 文により，配列の次元を変更します.
以下に，配列の次元入れ替えの手順を示します.

・define 文による次元の定義
以下のような内容の define 文をソースに記述，又はインクルードして使用します.

```
#define  q1(a,b,c,d)  q1F(d,a,b,c)
```

・翻訳時オプションの設定
CeNSS システムでプリプロセッサの機能を用いる場合，翻訳時にオプションに”-Cpp”を指定して翻訳します.
以下に，指定例を示します.

```
f90ns  -Uxpf,auto  -Cpp  Sample.f90  -o  Sample.exe

  -Cpp に加えて, -P オプションを指定すると, #define で指定した内容を置き換えた結果が,
  ファイル（File.cpp.f 等）に出力されます.
```

●注意事項
プリプロセッサの機能は完全ではありませんので，-P オプションにより置き換えた内容を確認した上で，正式な翻訳を行います.

以下は, TLB ミスが発生するプログラムと, TLB ミスを回避したプログラムの例を示します. ソース変更後は, データアクセスの局所性が向上しています.

変更前

```
do l =          ! Z方向
do k =          ! Y方向
      :
  do j =        ! X方向
        jp = j + 1
     :
        s5jp =.5*( q1(jp,k,lp,10)/q1(jp,k,lp,1) + q1(jp,k, l,10)/q1(jp,k, l,1) )
        s6jp =.5*( q1(jp,k,lp,11)/q1(jp,k,lp,1) + q1(jp,k, l,11)/q1(jp,k, l,1) )
        s7jp =.5*( q1(jp,k,lp,12)/q1(jp,k,lp,1) + q1(jp,k, l,12)/q1(jp,k, l,1) )
        s8jp =.5*( q1(jp,k,lp,13)/q1(jp,k,lp,1) + q1(jp,k, l,13)/q1(jp,k, l,1) )
        s9jp =.5*( q1(jp,k,lp,14)/q1(jp,k,lp,1) + q1(jp,k, l,14)/q1(jp,k, l,1) )
        s10jp =.5*( q1(jp,k,lp,15)/q1(jp,k,lp,1) + q1(jp,k, l,15)/q1(jp,k, l,1) )
    :
  enddo
enddo
enddo
```

図 6.6　TLB ミスが発生するプログラム

●変更前：１回で参照するデータの間隔が広い　⇒　TLB ミス，及び２次キャッシュミスの増大

ql(1,1,1,1)	ql(2,1,1,1)	～	ql(1,2,1,1)	ql(2,2,1,1)	～	ql(1,1,2,1)	ql(2,1,2,1)	～
ql(1,1,1,2)	ql(2,1,1,2)	～	ql(1,2,1,2)	ql(2,2,1,2)	～	ql(1,1,2,2)	ql(2,1,2,2)	～
≀	・・・	～	・・・	・・・	～	・・・	・・・	～
ql(1,1,1,15)	ql(2,1,1,15)	～	ql(1,2,1,15)	ql(2,2,1,15)	～	ql(1,1,2,15)	ql(2,1,2,15)	～

図 6.7　TLB ミス改善前のキャッシュイメージ

変更後

```
include  'define.inc'
do l =          ! Z方向
do k =          ! Y方向
      :
  do j =        ! X方向
        jp = j + 1
     :
        s5jp =.5*( q1F(10,jp,k,lp)/q1F(1,jp,k,lp) + q1F(10,jp,k, l)/q1F(1,jp,k, l) )
        s6jp =.5*( q1F(11,jp,k,lp)/q1F(1,jp,k,lp) + q1F(11,jp,k, l)/q1F(1,jp,k, l) )
        s7jp =.5*( q1F(12,jp,k,lp)/q1F(1,jp,k,lp) + q1F(12,jp,k, l)/q1F(1,jp,k, l) )
        s8jp =.5*( q1F(13,jp,k,lp)/q1F(1,jp,k,lp) + q1F(13,jp,k, l)/q1F(1,jp,k, l) )
        s9jp =.5*( q1F(14,jp,k,lp)/q1F(1,jp,k,lp) + q1F(14,jp,k, l)/q1F(1,jp,k, l) )
        s10jp =.5*( q1F(15,jp,k,lp)/q1F(1,jp,k,lp) + q1F(15,jp,k, l)/q1F(1,jp,k, l) )
    :
  enddo
enddo
enddo
```

```
#define  q1(a,b,c,d) q1F(d,a,b,c)
により，次元入れ換えを行ったイメージ
```

図 6.8　TLB ミス改善後のプログラム

●変更後：１回で参照するデータは，隣り合っている　⇒　主記憶参照の局所性向上

ql(1,1,1,1)	**ql(2,1,1,1)**	**ql(3,1,1,1)**	**ql(4,1,1,1)**	**ql(5,1,1,1)**	～	**ql(13,1,1,1)**	**ql(14,1,1,1)**	**ql(15,1,1,1)**

図 6.9　TLB ミス改善後のキャッシュイメージ

6.1.3 その他のスカラーチューニング

上記で説明した性能チューニング以外に，オーバヘッドの削減や，冗長実行箇所の見直し等により，性能向上が期待できます．以下に，そのような例を紹介します．

(1)インライン展開

サブルーチンや関数の呼び出しには，レジスタを確保したり，引数のリストを用意したりするオーバヘッドがあり　ます．さらにコンパイラによる静的なデータ依存関係の解析処理を遮断し，各種の最適化を阻害する可能性もあります．したがって，小さなサブルーチンを頻繁に呼び出す場合はそのオーバヘッドが表面化するため，性能劣化の要因となります．このような場合，インライン展開により，サブルーチンの定義を呼び出し側のモジュールに埋め込むことで，性能劣化を削減することができます．インライン展開の機能を有効にする場合，翻訳オプションに，-x を指定して翻訳します．また，-Ei オプションを指定することで，インライン展開が行われたことを示す診断メッセージを確認することができます．

以下に，翻訳例と，インライン展開が行われたことを示す診断メッセージの例を示します．

（翻訳例）

```
% f90ns  -Uauto  -x-  -Ei  Sample.f90
```

インライン展開に関する基本オプション
- -x-　　：　実行文の数が 30 以下の利用者定義の手続きをインライン展開の対象とする．
- -x*num*　：　実行文の数が *num* 以下の利用者定義の手続きをインライン展開の対象とする．

（診断メッセージの例）

```
jwd8101i-i  "Sample.f90", line 35: 利用者定義の手続'sub1'をインライン展開しました.
```

以下に，インライン展開を実施したコンパイルリストの例を紹介します．

インライン展開前

```
s u        DO J=2,NY-1
m u          ROR = QR(J,1)
s u          UNR = QR(J,2)
s u          UTR = QR(J,3)
s u          USR = QR(J,4)
s u           PR = QR(J,5)
               ・・・
s u      CALL SHUS_CG1(ROR,UNR,UTR,USR,PR,ER,
          &  ROL,UNL,UTL,  USL,PL,EL,GAMM,
          &  H1,H2,H3,H4,H5)
s u              F4(J,1) = H1 * DS(J)
s u              F4(J,2) = H2 * DS(J)
s u              F4(J,3) = H3 * DS(J)
s u              F4(J,4) = H4 * DS(J)
s u              F4(J,5) = H5 * DS(J)
p u        END DO
```

jwd5208i-i "backg41.f", line 358: 定義引用の順序が分からないため,引用順序が逐次実行と異なる可能性があり,このDO ループは並列化されません.(名前:h1)

インライン展開後

```
p u        DO J=2,NY-1
p u          ROR = QR(J,1)
p u          UNR = QR(J,2)
p u          UTR = QR(J,3)
p u          USR = QR(J,4)
p u           PR = QR(J,5)
               ・・・
piu      CALL SHUS_CG1(ROR,UNR,UTR,USR,PR,ER,
          &  ROL,UNL,UTL,  USL,PL,EL,GAMM,
          &  H1,H2,H3,H4,H5)
p u              F4(J,1) = H1 * DS(J)
p u              F4(J,2) = H2 * DS(J)
p u              F4(J,3) = H3 * DS(J)
p u              F4(J,4) = H4 * DS(J)
p u              F4(J,5) = H5 * DS(J)
p u        END DO
```

インライン展開を示す"i"が表示される

jwd5001i-i "backg41.f", line 334: このDO ループは,並列化されました.(名前:j)
jwd8101i-i "backg41.f", line 358: 利用者定義の手続'shus_cg1'をインライン展開しました.

図 6.10　インライン展開

(2)冗長演算の削減

冗長演算を先行評価することにより，性能が向上する場合があります．以下の例では，log 関数が j [jmax(m)] 倍の冗長計算を行なっていたため，1000 のループ外に移動し，冗長処理を削減しています．

変更前

```
      do 1000 j = 1, jmax(m)
      do 1000 i = 1, ii

       RKFTj(j,i)=EXP(log(PAR(1,i))+PAR(2,i)*
      &             ALOGT(j)-PAR(3,i)/ttj(j))
            :

1000 continue
```

変更後

冗長な計算を j ループの外に移動

```
      do 200 i = 1, ii
       palog(i  )=log(PAR(1,i  ))
200   continue

      do 1000 j = 1, jmax(m)
      do 1000 i = 1, ii

       RKFTtmp      =palog(i  )+PAR(2,i  )*
      &               ALOGT(j)-PAR(3,i  )/ttj(j)
       RKFTjF(i  ,j)=EXP(RKFTtmp )
            :

1000 continue
```

図 6.11　冗長演算

6.2 スレッド並列チューニング

スレッド並列性能を阻害する最も大きな要因は，スカラー性能劣化と同様，キャッシュミス等のメモリアクセス効率の悪化によるものです．スレッド並列時には，複数のスレッドによる，キャッシュラインの奪い合い（false sharing）等の発生により，キャッシュミスが発生しやすくなります．

スレッド並列性能を向上させるには，以下のような方法が挙げられます．

1． 並列化率の向上
2． 並列化粒度の拡大
3． False sharing の除去
4． ロードバランスの改善

6.2.1 並列化率の向上

非並列処理の箇所を，OpenMP 指示行等により並列化させ，並列化率を向上させることができます．
以下の例では，ループ中にサブルーチン呼出しが含まれるため，自動並列では並列化されませんが，OpenMP により並列化を行っています．

変更前	変更後
do j = 1,nj do i = 1,ni a(i) = a(i) + x(i, j) call sub(a, j) y(i, j) = y(i, j) / a(i) enddo enddo	!$omp parallel do p do j = 1,nj p do i = 1,ni p a(i) = a(i) + x(i, j) p call sub(a, j) p y(i, j) = y(i, j) / a(i) p enddo p enddo !omp end parallel do

図 6.12 OpenMP による並列化

6.2.2 並列化処理粒度の拡大

並列化処理粒度の拡大により，性能を向上させることが出来ます．具体的な性能チューニング方法は以下の通りです．

（1）外側ループを並列化の対象にする

外側ループを並列化の対象とすることで，並列化範囲の拡大，オーバヘッド削減等により，性能が向上します．以下に，外側ループで並列化を行う例を示します．自動並列化では，内側ループしか並列化されませんが，OpenMP 指示行により外側ループでの並列化を行うことができます．

変更前

```
                do 25 l = 2, lmax(m)-1
                do 25 j = 2, jmax(m)-1
      c
      chemkin
                call ckvisy(m,j,l,NO,RMCWRK(NETA),vmyu)
                call ckcony(m,j,l,NO,RMCWRK(NLAM),vkap)
      chemkin
      c
p               do 20 k = 1, kmax(m)-1
:
p         20    continue
:
p               do 23 k = 2, kmax(m)-1
:
p         23    continue
      c
          25 continue
```

自動並列化では，
内側ループのみが並列化

変更後

```
      !$omp   parallel default(private)
      !$omp& shared(m,NO,NETA,NLAM)
      !$omp&shared(RMCWRK,PATMOS,SMALL)
      !$omp& shared(kmax,jmax,lmax,voli,t)
      !$omp& shared(xynF,q1F,rhsF)
        :
      !$omp   do
p           do 25 l = 2, lmax(m)-1
p           do 25 j = 2, jmax(m)-1
      c
p             call ckvisy(m,j,l,NO,RMCWRK(NETA),vmyu)
p             call ckcony(m,j,l,NO,RMCWRK(NLAM),vkap)
      c
p               do 20 k = 1, kmax(m)-1
:
p         20    continue
:
p               do 23 k = 2, kmax(m)-1
:
p         23    continue
      c
p         25 continue
      !$omp end do
      !$omp end parallel
```

OpenMP により，
外側ループも並列化

図 6.13　外側ループの並列化

(2)XPFortran 分割ループの自動並列化

標準では抑止されている XPFortran 分割ループでの自動並列化を有効にすることで，並列化対象範囲を拡大することが出来ます．有効にするには，翻訳時オプション(-Wx,-iteration=N)を指定してソースを再翻訳する必要があります．

（翻訳例）

```
% f90ns  -Uxpf,auto  -Wx,-iteration=2  -Wx,-Rsd  Sample.f90  -o  Sample.out
```

-Wx,-iteration=N : SPREAD DO 文で分割される DO ループの分割後の回転数が，N 以上であるときに自動並列化を行うことを示します．

以下は，XPFortran 分割ループで自動並列化したプログラムのコンパイルリスト例です．

解除前	解除後
!XOCL SPREAD DO /IPL	!XOCL SPREAD DO /IPL
DO 130 L = 2, LM	**p DO 130 L = 2, LM**
s u DO 130 K = 2, KM	p u DO 130 K = 2, KM
p u UXL(K,L,1) = VXL(K,L,1)	p u UXL(K,L,1) = VXL(K,L,1)
p u UXL(K,L,2) = VXL(K,L,2)	p u UXL(K,L,2) = VXL(K,L,2)
p u UXL(K,L,3) = VXL(K,L,3)	p u UXL(K,L,3) = VXL(K,L,3)
p u UXL(K,L,4) = VXL(K,L,4)	p u UXL(K,L,4) = VXL(K,L,4)
p u UXL(K,L,5) = VXL(K,L,5)	p u UXL(K,L,5) = VXL(K,L,5)
p 130 CONTINUE	p 130 CONTINUE
!XOCL END SPREAD	!XOCL END SPREAD
jwd5143i-i "wing2.f": DO ループの繰返し数が少ないため,この DO ループは並列化されません.	jwd5001i-i "lhsuw.f": この DO ループは,並列化されました.(名前:I)

分割ループで自動並列化

図 6.14 自動並列化したプログラムのコンパイルリスト

6.2.3 False Sharing の除去

スレッド並列プログラムでは，逐次プログラムで発生するキャッシュミスのパターンに加えて，False Sharing に　よって発生するキャッシュミスのパターンがあります．スレッド並列実行時の２次キャッシュミス改善には，スカラーチューニング同様，主記憶へのアクセスの効率化を行います．

※スレッド並列数を大きくするに従ってキャッシュミスが増大する場合, False Sharing が発生している可能性があります．

以下に，False Sharing の回避例を示します．
以下の例では，OpenMP を使用し，外側ループでの並列化により，キャッシュミスを回避しています．

変更前	変更後
do 160 l = 1, lmax(m) do 160 k = 1, kmax(m) do 100 j = 1, jmax(m)-1 : : rr(j) = rhol*dd uu(j) = (ul + dd*ur)/(1.0 + dd) vv(j) = (vl + dd*vr)/(1.0 + dd) ww(j) = (wl + dd*wr)/(1.0 + dd) hh(j) = (hl + dd*hr)/(1.0 + dd) s1(j) = (s1l + dd*s1r)/(1.0 + dd) s2(j) = (s2l + dd*s2r)/(1.0 + dd) s3(j) = (s3l + dd*s3r)/(1.0 + dd) s4(j) = (s4l + dd*s4r)/(1.0 + dd) : : 160 continue	**!$omp parallel default(private)** **!$omp& shared(m,lmax,kmax,jmax,xyn,t)** **!$omp& shared(q1,pp,phrgas)** **!$omp& shared(rhs,voli)** **!$omp do** do 160 l = 1, lmax(m) do 160 k = 1, kmax(m) do 100 j = 1, jmax(m)-1 : : rr(j) = rhol*dd uu(j) = (ul + dd*ur)/(1.0 + dd) vv(j) = (vl + dd*vr)/(1.0 + dd) ww(j) = (wl + dd*wr)/(1.0 + dd) hh(j) = (hl + dd*hr)/(1.0 + dd) s1(j) = (s1l + dd*s1r)/(1.0 + dd) s2(j) = (s2l + dd*s2r)/(1.0 + dd) s3(j) = (s3l + dd*s3r)/(1.0 + dd) s4(j) = (s4l + dd*s4r)/(1.0 + dd) : : 160 continue **!$omp end do** **!$omp end parallel**

図 6.15　外側ループの並列化によるキャッシュミスの回避

6.2.4 ロードバランスの改善

（1）並列化する DO ループの回転数を大きくする

データアクセスが連続であっても，並列化するループの回転数が小さい場合，ループのオーバヘッドが表面化し，性能が出ない場合があります．このような場合，並列化するループの回転数を大きくすることで，オーバヘッドが削減され，ロードバランスや実行性能が向上します．

以下に，ロードバランス向上の例を示します．

交換前	交換後
!$OMP DO do i = 1, imax ..... imax は小 do j = 1, jmax ..... jmax は大 do k = 1, kmax ..... kmax は大 ・・・ a(k,j,i) = a(k,j,i) + b(k,j,i) * c ・・・ enddo enddo enddo !$OMP END DO	!$OMP DO do j = 1, jmax ... jmax は大 do k = 1, kmax .... kmax は大 do i = 1, imax ... imax は小 ・・・ a(i,k,j) = a(i,k,j) + b(i,k,j) * c ・・・ enddo enddo enddo !$OMP END DO

図 6.16　DO ループの回転数を大きくすることによるロードバランスの改善

※注意：この場合，主記憶へのアクセスパターンを劣化させないため，配列の次元入れ替えも行っています．

6.3 プロセス並列チューニング

プロセス並列性能を阻害する代表的な要因は，同期処理，転送待ち等によるオーバヘッドが上げられます.

プロセス並列性能を向上させるには，以下のような方法が挙げられます.

1．並列化対象範囲の拡大
2．ロードバランスの改善
3．プロセス間通信コストの削減

6.3.1 並列化対象範囲の拡大

スカラーチューニングでも有効であった「ループ融合」を実施することで，並列化対象範囲を拡大することが出来ます.

XPFortran の場合，SPREAD DO を融合することで，プロセス間同期処理等のオーバヘッドの削減により，性能向上が期待できます.

以下にループの融合の例を示します.

融合前

```
      !XOCL SPREAD DO /IPL
p               DO 130 L = 2, LM
p                DO 130 K = 2, KM
                      ・・・
p                 UXL( K,L,1 ) = VXL( K,L,1 )
p                 UXL( K,L,2 ) = VXL( K,L,2 )
p                 UXL( K,L,3 ) = VXL( K,L,3 )
p                 UXL( K,L,4 ) = VXL( K,L,4 )
p                 UXL( K,L,5 ) = VXL( K,L,5 )
p       130     CONTINUE
      !XOCL END SPREAD
      !XOCL SPREAD DO /IPL
p               DO 150 L = 2, LM
p               DO 150 K = 2, KM
p                 SS   = 0.
                      ・・・
p                 VXL( K,L,1 ) = SL( JS,K,L,1 )*FX1
p                 VXL( K,L,2 ) = SL( JS,K,L,2 )*FX1
p                 VXL( K,L,3 ) = SL( JS,K,L,3 )*FX1
p                 VXL( K,L,4 ) = SL( JS,K,L,4 )*FX4
p                 VXL( K,L,5 ) = SL( JS,K,L,5 )*FX5
p       150     CONTINUE
      !XOCL END SPREAD
```

融合後

```
      !XOCL SPREAD DO /IPL
p               DO 130 L = 2, LM
p                DO 130 K = 2, KM
                      ・・・
p                 UXL( K,L,1 ) = VXL( K,L,1 )
p                 UXL( K,L,2 ) = VXL( K,L,2 )
p                 UXL( K,L,3 ) = VXL( K,L,3 )
p                 UXL( K,L,4 ) = VXL( K,L,4 )
p                 UXL( K,L,5 ) = VXL( K,L,5 )
p                 SS   = 0.
                      ・・・
p                 VXL( K,L,1 ) = SL( JS,K,L,1 )*FX1
p                 VXL( K,L,2 ) = SL( JS,K,L,2 )*FX1
p                 VXL( K,L,3 ) = SL( JS,K,L,3 )*FX1
p                 VXL( K,L,4 ) = SL( JS,K,L,4 )*FX4
p                 VXL( K,L,5 ) = SL( JS,K,L,5 )*FX5
p       130     CONTINUE
      !XOCL END SPREAD
```

図 6.17　並列化におけるループ融合

6.3.2 ロードバランスの改善

一部のプロセスだけ動作するような SPREAD DO ループが複数存在する場合，それらをの処理を融合することで，全体のロードバランスが改善できる場合があります.

以下に，ループ融合による書き換え例を示します.

修正前

```
C======================================== FOR J1
!XOCL SPREAD DO /ISC
      DO 601 J=1,1
       IF (J.EQ.1) THEN
       DO 201 K=1,KG
       DO 201 I=1,IG
         ・・・
  201  CONTINUE
       ENDIF
  601 CONTINUE
!XOCL END SPREAD
C======================================== FOR J2
!XOCL SPREAD DO /ISC
      DO 602 J=2,2
       IF (J.EQ.2) THEN
       DO 202 K=1,KG
       DO 202 I=1,IG
         ・・・
  202  CONTINUE
       ENDIF
  602 CONTINUE
!XOCL END SPREAD
      ・・・
C======================================== FOR JA
!XOCL SPREAD DO /ISC
      DO 611 J=10,10
       IF (J.EQ.10) THEN
       DO 211 K=1,KG
       DO 211 I=1,IG
         ・・・
  211  CONTINUE
       ENDIF
  611 CONTINUE
!XOCL END SPREAD
```

J=1 に対応するプロセスのみ動作(残りは完了待ち)

J=2 に対応するプロセスのみ動作(残りは完了待ち)

J=10 に対応するプロセスのみ動作(残りは完了待ち)

修正後

```
!XOCL SPREAD DO /ISC
      DO 601 J=1,10
C======================================== FOR J1
       IF (J.EQ.1) THEN
       DO 201 K=1,KG
       DO 201 I=1,IG
         ・・・
  201  CONTINUE
C======================================== FOR J2
       ELSE IF (J.EQ.2) THEN
       DO 202 K=1,KG
       DO 202 I=1,IG
         ・・・
  202  CONTINUE
      ・・・
C======================================== FOR JA
       ELSE IF (J.EQ.10) THEN
       DO 211 K=1,KG
       DO 211 I=1,IG
         ・・・
  211  CONTINUE
C
       ENDIF
  601 CONTINUE
!XOCL END SPREAD
```

1～10 の各プロセスが同時に動作

このほか，!XOCL SPREAD REGION による書換えも可能です.

図 6.18 ループ融合によるロードバランス改善

6.3.3 プロセス間通信のコスト削減

(1)MPI のノンブロッキング通信の使用

MPI プログラムの SEND 転送処理を，ノンブロッキング転送に変更することでコストを改善できます.
以下に，ノンブロッキング転送に書き換える例を示します.

修正前	修正後

m = myrank + 1
if(m.le.npe-npl) then
mm = m+npl
call **MPI_SEND**(Buf1s,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,IERR)
end if
 	
m = myrank + 1
if(m.ge.1+npl) then
mm = m-npl
call **MPI_RECV**(Buf1r,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,status,IERR)
 	
MPI_SEND → MPI_ISEND (Immediate**SEND**)
	m = myrank + 1
	if(m.le.npe-npl) then
	mm = m+npl
	call **MPI_ISEND**(Buf1s,jmax1*lmax1*14*2,
	& MPI_DOUBLE_PRECISION,mm-1,
	& itag1,MPI_COMM_WORLD,ireq1s,IERR)
	end if
	
	m = myrank + 1
	if(m.ge.1+npl) then
	mm = m-npl
	call **MPI_IRECV**(Buf1r,jmax1*lmax1*14*2,
	& MPI_DOUBLE_PRECISION,mm-1,
	& itag1,MPI_COMM_WORLD,ireq1r,IERR)
	end if
	
	m = myrank + 1
	if(m.ge.1+npl) then
	call **MPI_WAIT**(ireq1r,status,IERR)
	end if
	
	m = myrank + 1
	if(m.le.npe-npl) then
	call **MPI_WAIT(ireq1s,status,IERR)**
	end if

図 6.19 MPI のノンブロッキング通信

（2）XPFortran の演算と通信のオーバーラップ

XPFortran プログラムで，演算処理と通信処理を同時に行わせることで，実行時間を短縮することが出来ます．以下に，演算と通信をオーバーラップさせる例を示します．

修正前

```
!XOCL SPREAD MOVE /IPK
      DO 800 K = 1, KMAX
      DO 800 L = 1, LMAX
      DO 800 J = 1, JMAX
        S(J,K,L,1) = SK(J,K,L,1)
        S(J,K,L,2) = SK(J,K,L,2)
        S(J,K,L,3) = SK(J,K,L,3)
        S(J,K,L,4) = SK(J,K,L,4)
        S(J,K,L,5) = SK(J,K,L,5)
  800 CONTINUE
!XOCL END SPREAD (MOV4)
!XOCL MOVEWAIT (MOV4)
C   ROTATION TERMS
C
!XOCL SPREAD DO /IPL
      DO 40 L = 2, LM
      DO 40 K = 2, KM
      DO 40 J = 2, JM
        RJ          = 1./QL(J,K,L,6)
        SL(J,K,L,2) = SL(J,K,L,2) -OMEGA*QL(J,K,L,3)*(-DT)*RJ
        SL(J,K,L,3) = SL(J,K,L,3) +OMEGA*QL(J,K,L,2)*(-DT)*RJ
   40 CONTINUE
!XOCL END SPREAD
```

修正後

```
!XOCL SPREAD MOVE /IPK
      DO 800 K = 1, KMAX
      DO 800 L = 1, LMAX
      DO 800 J = 1, JMAX
        S(J,K,L,1) = SK(J,K,L,1)
        S(J,K,L,2) = SK(J,K,L,2)
        S(J,K,L,3) = SK(J,K,L,3)
        S(J,K,L,4) = SK(J,K,L,4)
        S(J,K,L,5) = SK(J,K,L,5)
  800 CONTINUE
!XOCL END SPREAD (MOV4)
C   ROTATION TERMS
C
!XOCL SPREAD DO /IPL
      DO 40 L = 2, LM
      DO 40 K = 2, KM
      DO 40 J = 2, JM
        RJ          = 1./QL(J,K,L,6)
        SL(J,K,L,2) = SL(J,K,L,2) - OMEGA*QL(J,K,L,3)*(-DT)*RJ
        SL(J,K,L,3) = SL(J,K,L,3) + OMEGA*QL(J,K,L,2)*(-DT)*RJ
   40 CONTINUE
!XOCL END SPREAD
!XOCL MOVEWAIT (MOV4)
```

演算部を転送処理とオーバーラップ

図 6.20　演算と通信のオーバーラップ

以　上