CeNSS における大規模シミュレーションの性能向上

松尾裕一、土屋雅子
宇宙航空研究開発機構

Performance Enhancement of Large-Scale Simulations at CeNSS

by
Yuichi MATSUO and Masako TSUCHIYA

ABSTRACT

In this paper, we discuss the issues concerning performance evaluation and its enhancement with tuning technique on the CeNSS system at JAXA NS-III. First, we show the real performances for the JAXA CFD codes on the CeNSS, and point out the importance of the performance evaluation and its enhancement with tuning. Currently, there is no effective indicator nor explicit strategy of the performance improvement on the CeNSS. Then, we propose the concept of effective peak performance which comes from the number of ideal floating point operations, suggesting the potential improvement by the performance tuning. With this concept, we can estimate the possible improvement. From analysis using the 250 codes, we found this effective peak performance important for the performance enhancement.

１．はじめに

近年、CeNSS（JAXA スーパーコンピュータシステム NS-III の計算エンジン Central Numerical Simulation System を略して CeNSS「センス」と呼んでいる。）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出てきているが、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、あるいはそれをどうする、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。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、きちんと議論されていないように見受けられる。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。こうした問題意識が本研究の背景にある。

本稿では、JAXA 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チューニングの方法論について論ずる。実効ピーク性能という考え方を提示し、その有効性やチューニング法について論ずる。

２．CeNSS の構成とプログラミングスタイル

図 1 は、JAXA 総研本部に導入されているスーパーコンピュータシステム 数値シミュレータ III（NS-III）の全体構成を示したものである。CeNSS とは、このシステム全体の中で計算部分を担当しているサブシステムの呼称であり、中央 NS システムとも呼ばれている。単に、NS システムと呼んだときには、主にこの部分を指している。

図 2 は、I/O などの部分を除いた CeNSS の構成イメージを示したものである。ノード内は、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。ここで、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。DTU あたり、16 プロセスを同時に処理することができる。表 1 に、CeNSS の構成緒元を示した。4 ノードで 1 筐体を構成しており、計算筐体は全部で 14 筐体、ノード数では 56 ノードある。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 クラスターと呼ばれることがあり、どの CPU からみても厳密に対称な配置を有している。[1-4]

図 3 は、CeNSS における Fortran の並列プログラミング体系をイメージ的に示したものである。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 4 に並列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 5 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。

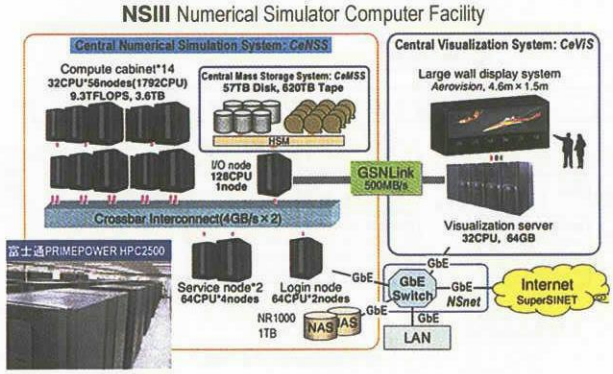

図 1　JAXA 数値シミュレータ III（NS-III）の構成概要

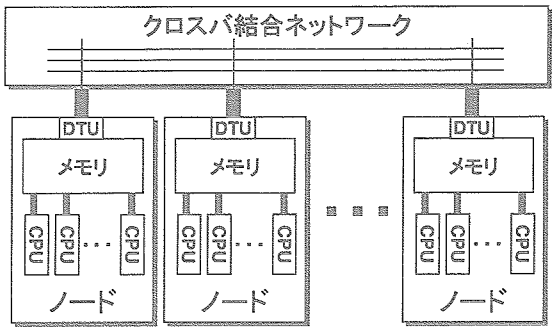

図2 CeNSSの構成イメージ

表1 CeNSSの緒元

演算処理性能	9.3TFLOPS
メモリ	3.6TB
計算用ノード数	56
ノード内CPU数	32
ノード内メモリ	64GB
ノード構成	SMP
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

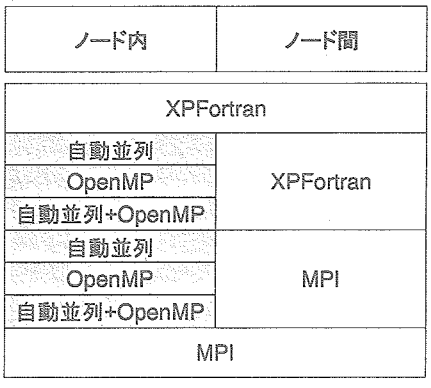

図3 CeNSSにおけるFortranプログラミング体系

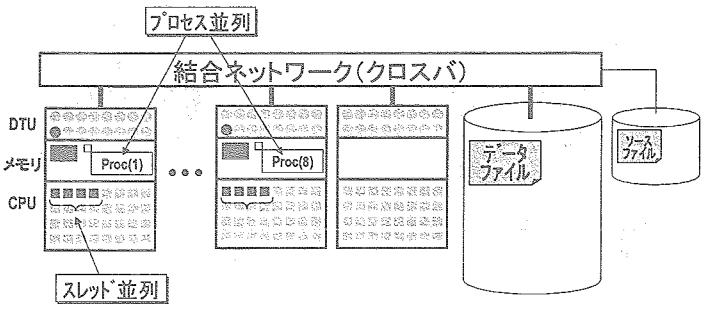

図4 並列ジョブ実行イメージ

3．性能向上の背景と課題

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で作られたものであり、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意外なほど多い。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、問題は、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、ベクトル機に比べて性能がかなり落ちてしまうコードがあるということ、しかも、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がある、ということである。図5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で走らせた場合の性能を比較したものである。CeNSSでは、どのコードもチューニング後には性能は向上している。特にコード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が、チューニング後にはベクトルと同様の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、その困難さをも示唆している。

現状、性能評価やチューニングに対する指標や方針は、キュッシュミスを1%以下にするであるとか、ループ操作やメモリアクセスの局所化のような一般的な例・法則を示すことはできる。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、システム運用側の大きな課題であると考えている。

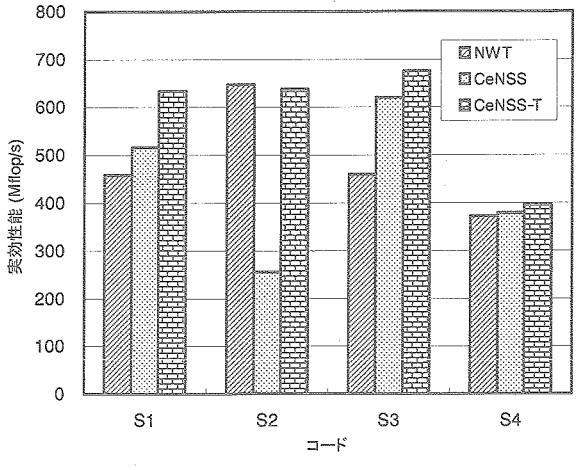

図5 チューニングによる性能向上の例

４．CeNSS における単体 CPU 実効性能の現状

富士通製 PRIME POWER HPC2500 に基づく CeNSS システムの性能は、CPU あたりでいうと、ピークで 5.2GFLOPS ということになっている。これは、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）の場合に、

1.3GHz（クロック）×4FLOPS = 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。実際のコードでは、M&A 命令の出現頻度はどれほどなのだろうか。また、実際のコードの性能詳細はどうなっているのだろうか。

そこで、JAXA における幾つかの実コードの特性や単体 CPU 性能を調べてみると以下の表 1 のようになった。これをみると、実際には M&A 命令の出現割合はせいぜい 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 を使っていることになる。とすれば、表 1 でいうところの実効効率（=実効性能/5.2GFLOPS）というのは、単なる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と表 2 のようになる。（ここで、CPU コスト、メモリコストとは、実行時間に占める CPU 処理時間、メモリアクセス時間の割合を示す。）このように、バルクの値としての実効効率の意味は別として、細かく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。M&A 命令の効果は、あまり出ていない。

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、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 2 では実効効率は 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？また、コードＰ6 は、実効性能は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、徒労に終わるかもしれないしやる気にも影響する。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、あるいは方針がほしいものである。

表 2　JAXA 実コードの特性と単体 CPU 性能

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

表 3　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	2.922	2.852	2.714	2.7721	2.748	2.774
実効効率	22.3%	22.7%	8.9%	15.2%	5.8%	4.1%

５．実効ピーク性能

5.1 提案と意義

そこで我々は、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 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、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。

以上のような実コード性能の評価分析活動から、図 7 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。「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
```

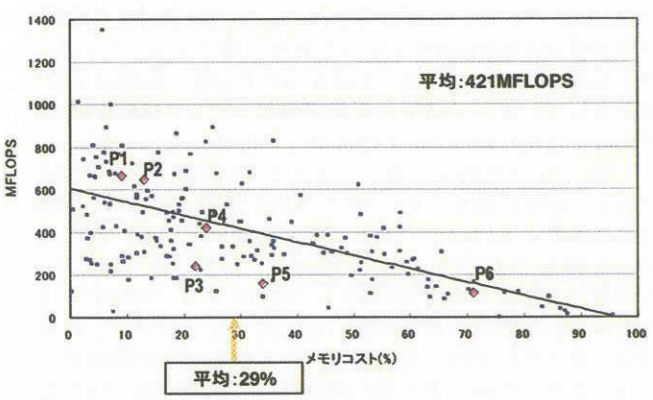

図 6　実効性能の調査結果

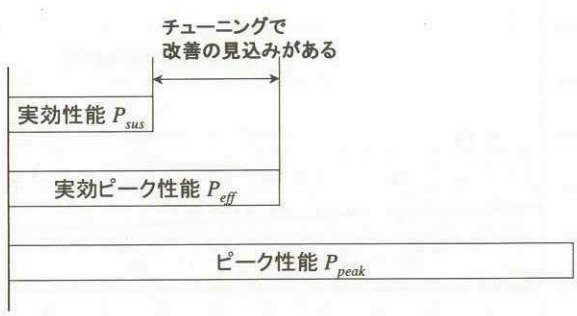

図 7　実効ピーク性能の考え方

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）

$$P_{eff}\ (\mathrm{MFLOPS}) = \frac{\text{浮動小数点演算数}}{\text{理想実行時間}} \times 10^6 \quad (1)$$

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 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(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 8 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(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。

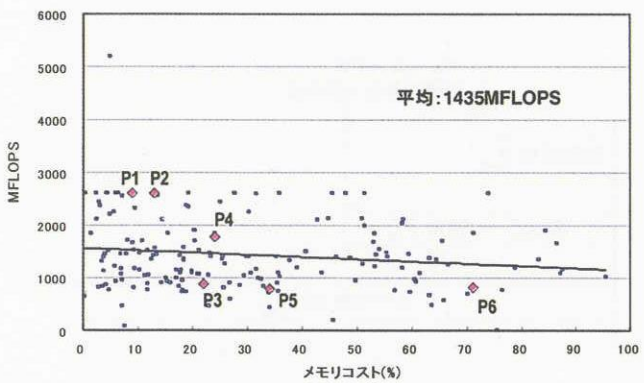

図 8 実効ピーク性能の調査結果

5.2 評価と課題

表 4 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 9 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、P6 を除きどのコードも平均値を下回っている。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 4 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
実効ピーク性能	2600 (3218)	2600 (3077)	873	1764	777	810
P_{sus}/P_{eff}	25.61%	24.91%	27.55%	23.90%	20.65%	14.07%

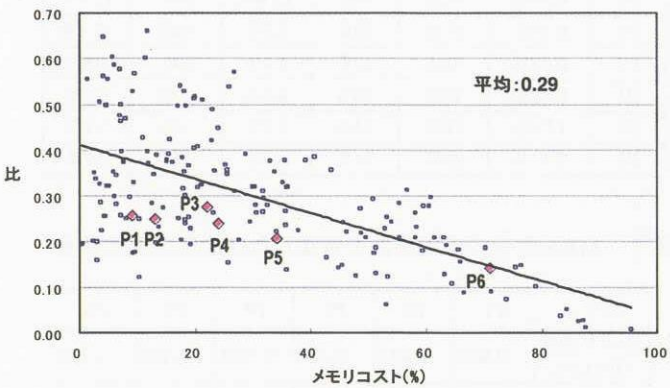

図 9 実効ピーク性能比の調査結果

5.3 実効ピーク性能によるチューニングの指針

以上の分析、考察から、実効ピーク性能を用いたチューニングについて以下のことがいえる。

1) 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。

2) 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、まさに一つの指標にはなるであろう。現在、富士通壁式会社と協力して、システムへの実装を考慮しているところである。

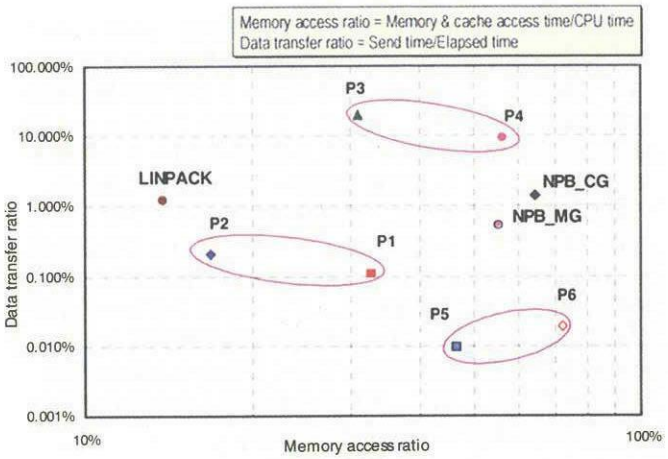

図 10　JAXA コード P1～P6 の特性

表 5　JAXA コードの特性総括

コード	P1, P2	P3, P4	P5, P6
ﾒﾓﾘｱｸｾｽ特性	軽	中	重
ﾃﾞｰﾀ転送特性	軽	重	軽

6．並列チューニング

ここまでは単体 CPU の性能評価とスカラーチューニングについて説明してきたが、大規模シミュレーションにおいては、当然のことながら、並列実行が主となる。上記の 6 本の JAXA コードは全て並列コードであり、その特性は、横軸にメモリアクセス比、縦軸にデータ転送比を取ってプロットすると図 10 のようになる。これから、表 5 に示すように、メモリアクセス特性とデータ転送特性でいうと、3 つのタイプに分類されることがわかる。このことからプロセス間（ノード間データ転送）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、コードの特性を知ることも性能向上の指針を得るには重要である。

図 11 は、性能チューニングの内容を分類したものである。本稿で議論したのは、主にスカラーチューニングの部分であった。CeNSS では、図 3 のように、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・スタイルを採用しているので、スレッドを使っている場合には、スレッド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。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、管理側からの支援も重要であると考えている。我々は、チューニングに関する情報を共有し、チューニング作業を体系的に効率よく行うために、「チューニング・フレームワーク」という仕組みを提案・構築している。図 12 にその概念と構成を示した。ここでのポイントは、チューニング作業のワークフロー化と、単なるガイド等の参考書以外の事例集とか FAQ 集とかの知識ベースの提供であり、初中級者をターゲットに提供していく予定にしている。

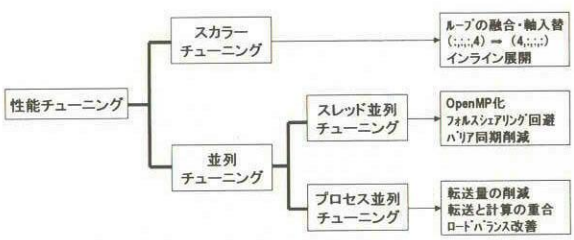

図 11　並列計算機における性能チューニング

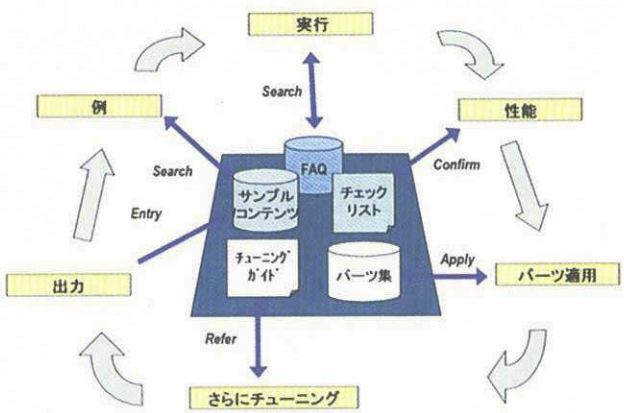

図 12　チューニング・フレームワークの概念

7．まとめ

本稿では、JAXA 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。

はじめに述べたように、性能評価やチューニングというのは、単に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ると、その必要性・重要性は高まって行くであろうと思われる。本稿の内容や考え方が、来たるべきそのような時代に向けての皆様のご参考になれば幸いである。

謝辞

本研究において、コードの性能測定やデータ提供に関して、富士通株式会社の関係者の方々に全面的にご協力いただいた。特に、青木正樹氏、稲荷智英氏には大変有益なご助言、ご支援をいただいた。ここに記して謝意を表します。

参考文献

1) 松尾：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) 藤田：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.
3) 高木：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.
4) 大川：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.
5) 松尾：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.