NICT サイエンスクラウドによる大規模シミュレーションデータ分散可視化処理

村田健史 [*1]，磯田総子 [*2]，渡邉英伸 [*1]，深沢圭一郎 [*3]，
山本和憲 [*1]，建部修見 [*4]，田中昌宏 [*4]，木村映善 [*5]

High Performance Visualization Processing of Large-Scale Computer Simulation Data via NICT Science Cloud

Ken T. MURATA[*1], Fusako ISODA[*2], Hidenobu WATANABE[*1], Keiichiro FUKAZAWA[*3],
Kazunori YAMAMOTO[*1], Osamu TATEBE[*4], Masahiro TANAKA[*4] and Eizen KIMURA[*5]

Abstract

Science cloud is a cloud system designed for scientific researches, and expected as a new infrastructure for big data sciences. Not only parallelization of CPU as in super-computers, but I/O and network throughput parallelization are crucial for the big data sciences. One of the typical structures of science cloud is a scalable cluster in which multiple clusters in a cloud are connected with high-speed network. In the present study, we examine performance of parallelization of both CPU and I/O in a cloud system as the first step to high performance scalable clusters. In case with few processes executed on each computational node (server), parallelization efficiency is almost 100%. This high efficiency is expected to maintain in larger-scale cluster systems such as those with 100 servers. On the condition of multi-processes on each node, the present parallelization does not show good performance due to the congestions of I/O. Parallelization efficiency (speed-up) is as low as 20.6%. New techniques of decentralization of I/O within each node are required in the next step.

Keyword: Science Cloud, Gfarm, Pwrake, Big data, Magnetospheric MHD simulation, I/O parallelization

概　要

現在，欧米や日本において，科学研究向けクラウドとしてサイエンスクラウドが提案され，システムや利活用についての議論が進められている．サイエンスクラウドはビッグデータ時代のデータ指向型科学研究基盤として提案されている．クラウド上でのビッグデータ処理では，CPU の並列分散化だけではなくネットワークや I/O の並列分散が重要である．本研究ではクラウド環境で高い計算効率を得るための基礎研究として，CPU および I/O を同時に分散処理するシステムを提案する．8 コアを有する 6 台のクラスタ計算機で基礎実験を行った結果，ノード内のプロセス数が少ない場合はほぼ 100%の並列化効率が達成され，100 台を超える大規模クラスタ環境でも高いスケーラビリティーが予測された．一方，ノード内で複数プロセスが処理を行う場合にはディスクアクセスの輻輳が発生し，必ずしも高いスケーラビリティーを得ることができない．8 コアを有する 6 台の計算サーバで最も高い並列化効率を得たのは 6 プロセス並列の場合であり，高速化率は 20.6 であった．今後は，同ノード内の I/O 処理を分散化することで，並列化効率をさらに向上させることが期待される．

1.　はじめに

多くの科学研究分野でのデータのデジタル化・大規模化を背景として，データ指向型科学は，実験科学・観測科学，理論科学，シミュレーション科学に続く科学研究手法の第 4 のパラダイムと言われている [1]．データ指向型科学の中でもビッグデータ科学は注目を浴びているが，近年，京コンピュータ [2] に象徴されるように第 3 の研究手法である数値シミュ

*1 情報通信研究機構 (National Institute of Information and Communications Technology)
*2 株式会社サイエンス・サービス (Science Service Co., LTD)
*3 九州大学情報基盤研究開発センター (Research Institute for Information Technology, Kyushu University)
*4 筑波大学計算科学研究センター (Center for Computational Sciences, University of Tsukuba)
*5 愛媛大学医学部 (Department of Medical Informatics Ehime University)

レーションの大規模化が進んでおり，スーパーコンピュータに代表される HPC（High Performance Computing）技術の進歩は目覚ましい．一方，数値シミュレーションが出力する数値データは同様に大規模化しているが，出力データのポスト処理技術については十分に発展しているとは言い難い．

宇宙物理学分野においても，流体系や粒子系の数値シミュレーションが行われている．その場観測（in-situ 観測）が容易ではない宇宙科学においては，数値シミュレーションの果たす役割は大きい．宇宙物理シミュレーションコード (プログラム) は，その多くが 3 次元化されており，大規模化が進んでいる．対象とする現象の複雑さに伴い，空間方向の大規模化と同時に時間方向の大規模化（長期シミュレーション）が進んでいる．

HPC の様に CPU リソースを極限まで活用して高速計算を行う計算指向型の研究に対して，生成されたデータを最も高速に処理するためのデータ指向型の技術開発が必要とされている．このような背景のもと，HPC に対する概念として MTC（Many-Task Computing）や HTC（High Throughput Computing）が登場した [3]．多種多様で大規模なデータを目的に応じて高速処理するための技術が MTC という概念である．MTC では，多様な計算機リソースを融合し，データ分散，並列データ処理やコンピュータとデータファイルをローカライズする工夫など，総合的なデータ処理環境をめざす．大規模数値シミュレーションデータ処理においては，HPC よりも MTC が有効である．また，同時に，HPC や MTC と言った計算指向（Computing Oriented）ではなく，データ配置や並列データ処理，データ局所性などに着目したデータ指向計算である Data Intensive Computing（DIC）と言う概念も提案されている[1]．

本研究では，10G ネットワークで接続された 6 台のクラスタ計算機環境において，MTC 型または DIC 型の並列分散データ処理システムを試験的に構築し，データ処理の並列化効率について調べる．これまでのクラスタ計算機では，計算指向型（HPC 型）の並列処理研究が多いが，本研究ではデータ処理時間に対してデータ I/O 時間が無視できない場合について考える．この場合，ネットワークを介して分散ファイルサーバ（たとえば NAS ストレージ）のデータファイルにアクセスする場合，スループットのオーバーフローが生じるため，一定以上の並列化分散処理は効率が極端に悪くなる．この場合に有効となるのは，I/O の分散化と CPU の並列化を両立するデータ処理システムである．

そこで本研究では，分散ファイルシステム Gfarm[4] とワークフローシステム Pwrake[5] を用いてクラスタ計算機環境に I/O と CPU を並列分散処理するシステムを構築し，実際の科学研究データ処理における並列化効率やスケーラビリティーについて基本性能を検討する．さらに検討結果をもとに実データファイルの処理を行い，その有効性を示すとともに，今後の NICT サイエンスクラウド [7] をはじめとする科学研究向けクラウドシステムで主流となると考えられるスケーラブルクラスタ環境での並列処理について議論する．

2. 実験の目的と実験方法

2.1. Gfarm/Pwrake

1 節で述べたとおり，MTC および DIC では，処理全体の効率向上のためには，HPC で考える CPU の分散化だけではなく，I/O やネットワークの分散化が必要である．本実験では，分散ファイルミドルウェア Gfarm[4] および Gfarm 用のワークフローシステム Pwrake[5] を用いた CPU 分散および I/O 分散により実際に科学研究で用いている数値シミュレーションデータ処理の並列処理を行う．

図 1 に Gfarm および Pwrake の機能について説明する．Gfarm は HPCI[5] などでも用いられている大規模分散ストレージに適した分散ファイルシステム（ミドルウェア）であり，メタデータサーバ（MDS）とファイルシステムノード（FSN）から構成される．複数のファイルシステムノードは L2 または L3 ネットワーク上で分散して配置され，すべてメタデータサーバにより管理される．クライアントノード（CN）は，分散ファイルシステムを利用するデータ処理ノードである．それぞれの CN は，複数の FSN から構成される分散ファイルシステムを共通ファイルシステムとして利用することができる．なお，Gfarm では特定のノード (計算機）がファイルシステムノードとクライアントノードを兼ねることができる．（図 1 は，FSN3 と FSN4 をクライアントノードとしても利用する場合である．）

Gfarm の特徴の一つは，データファイルの分散管理である．ユーザが分散ファイルシステムにファイル（たとえばファイル①）を保存する．Gfarm はあらかじめ設定した数の複製を作成し，異なる FSN に保存する．（図 1 の場合は，ファイル①は FSN1，FSN2 および FSN3 に保存される．）クライアントノードは，複製ファイルのうち，最も高速にアクセス

1 なお，HPC や MTC と並んで HTC（High-Throughput Computing）という用語が用いられることがあるが，HTC は HPC や MTC と比べると長期（数週間，数か月など）にわたってデータを処理する環境・技術を意味する．

できる最適データファイルを参照する．（たとえば，図 1 の場合にはファイル①は CN4 が処理し，ファイル②は CN5 が処理する．）データファイルごとの最適クライアントノードの選択は，ノード負荷やネットワークトラフィックによりメタデータサーバが行う．

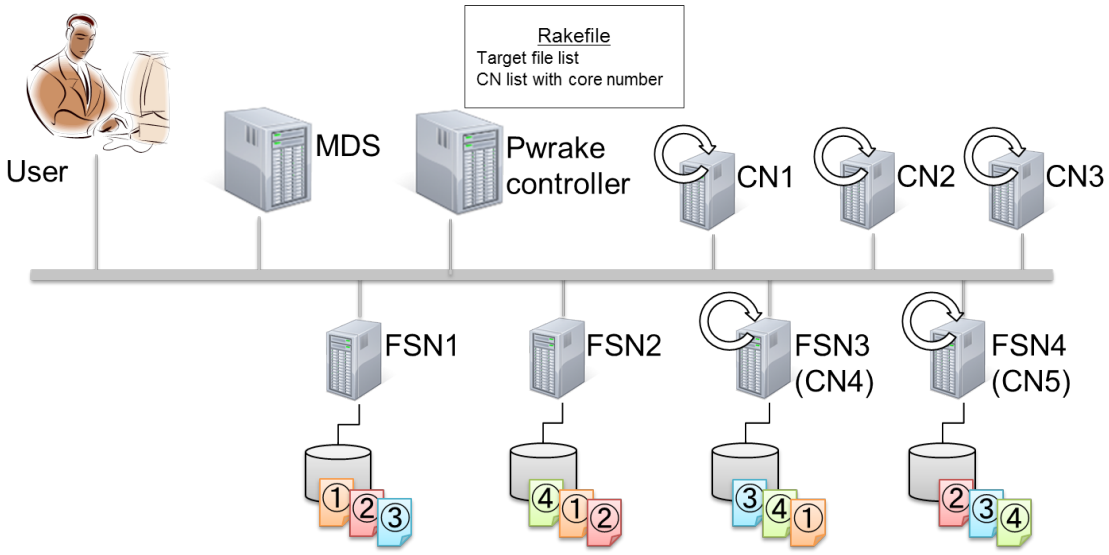

図 1　Gfarm と Pwrake の概念図：MDS（メタデータサーバ）/ FSN（ファイルシステムノード）/ CN（クライアントノード）/ Pwrake controller（Pwrake 制御サーバ）

Pwrake は，複数のクライアントノードを用いて高速な並列分散処理を行うことを目的に開発されたワークフローシステムである．ユーザは，まず，Pwrake コントローラが管理する Rake ファイルに処理したいファイル一覧を指定し，使用するクライアントノードとコア数のリストをノードファイルに記述する．Pwrake は処理の対象となるファイルリストを参照し，ノードファイルを読み込む．Pwrake は，参照した各データファイルについて，処理を行うクライアントノードとコアを指定する．その際，CN4 および CN5 のようにファイルシステムノードを兼ねたクライアントノードについては，クライアントノードがファイルシステムノードとして管理するファイルを優先的に処理に割り当てる．これにより，ネットワーク上にファイルアクセス（I/O）のトラフィックが流れることを回避できる．また，多くの場合は，ネットワークファイル転送よりもディスクアクセス速度の方が高速であるため，データ処理 (ファイル読み込みおよび書き出し) の高速化が期待できる．

Pwrake のワークフローの原則の例を図 2 に示す．図 2 は，2 台のクライアントノード（CN）がともにファイルシステムノード（FSN）を兼ねており，2 台の CN がそれぞれ 3 プロセスと 2 プロセスで①～⑳の 20 のファイル処理を実行する場合である．CN が処理を行うプロセス数は，図 1 の Rake ファイルが管理している．図 2 では簡単のため，2 台の FSN は 20 ファイル全てを管理しているものとする．図の右側が，各プロセスに割り当てられる処理対象ファイルである．図 1 の Pwrake コントローラはファイル処理が終わった各プロセスに，次のファイルを割り当てる．割り当てるデータファイルは図 1 の Rake ファイルが管理している．処理の順序はほぼデータファイルの順序となるが，処理時間が一定ではない場合には両者は必ずしも完全に一致はしない．

2.2.　実験の目的

1 節で述べたとおり，MTC および DIC では，処理全体の効率向上のためには，HPC で考える CPU の分散化だけではなく，I/O やネットワークの分散化が必要である．本実験では，Gfarm を用いた CPU 分散および I/O 分散により実際に科学研究で用いている数値シミュレーションデータ処理の並列処理を行う．

本研究では Gfarm/Pwrake を用いることで高い並列化効率を目指すが，そのためには CPU の分散と I/O の分散のバランスと最適化が必要である．Gfarm/Pwrake による詳細な処理効率に関する報告は少なく [5]，本研究ではまず予備実験，基礎実験により基本的な情報を収集する．それをもとに実際のデータ処理のためのモデルを作成し，本実験（実用化実験）を実施する．

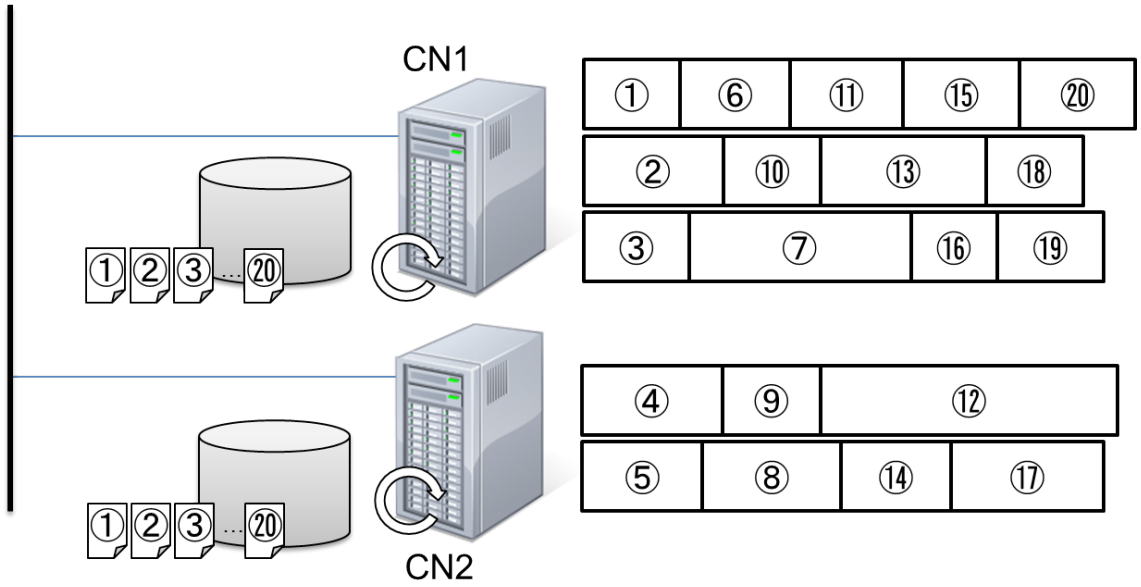

図 2 Pwrake によるワークフロー（クライアントノードが 2 台で，3 プロセスと 2 プロセスで処理する場合）：
処理が終わったノード（コア）に次のデータファイルの処理を割り当てる．当該ファイルが FSN として管理するローカルファイルでない場合には，そのファイルは割り当てない．

2.3. データセットと実験環境

本研究が対象とするデータセットを図 3 に示す．データはスーパーコンピュータにより計算された時系列 3 次元地球磁気圏グローバル MHD シミュレーションデータ [8] である．本シミュレーションコードは，450×300×300 グリッドの 3 次元時系列数値計算であり，0.5 秒の時間分解能で，各時刻 2.2GB の数値データを出力する [9]．出力データ形式は，数値シミュレーションで最もよく利用される形式の一つである HDF5 である．NICT が開発した可視化ツール（バーチャルオーロラツール）は HDF5 形式で記述されたこれらの数値データを読み込み，あらかじめ決めたパラメータにより可視化する．各時刻ステップのデータファイルの可視化には依存性がないため，本可視化処理を並列化する場合には可視化処理プロセス間でのメッセージ通信は発生しない．バーチャルオーロラツールは GUI を有するインタラクティブなアプリケーションであるが，あらかじめパラメータセットすることでコマンドラインからの処理が可能である．

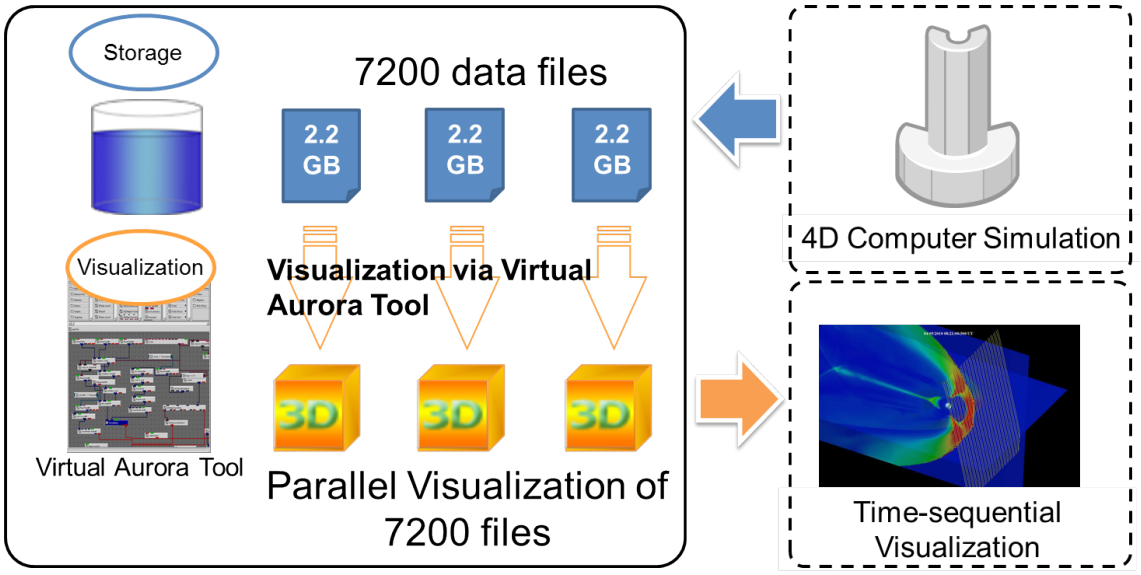

図 3 本研究が対象とする Global MHD シミュレーションデータ：時間分解能 0.5 秒で 1 時間分（7200 ステップ）の地球磁気圏現象のシミュレーションを行った結果である．実験では 7200 ステップデータのうちの一部を用いる．

図 4 に, 本研究で行う MHD シミュレーションの可視化出力例を示す．図 4 では, 磁力線（200 本）および xz 面と xy 面 (座標軸は GSM 座標系に準拠する) のプラズマ圧力のコンタープロットを出力している．本研究では，この数値データをすべてこの数値データ 1 時間分（7200 ステップ分）の一部を並列処理することを目標とする．[2]

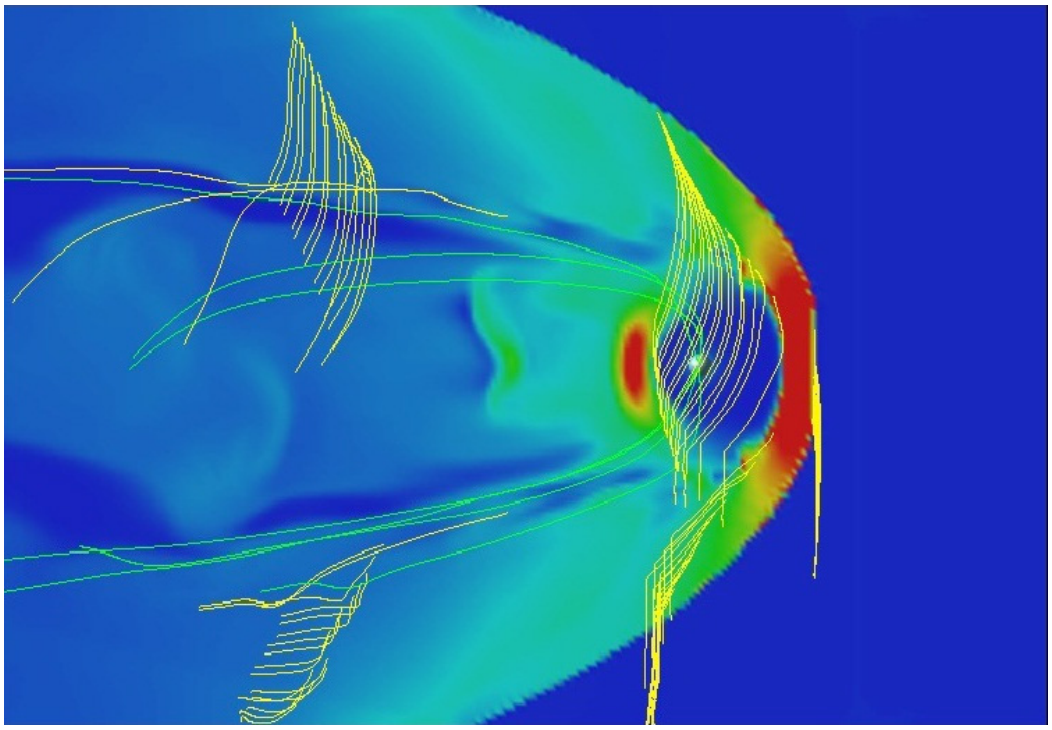

図 4　本研究の可視化結果（3D 可視化を特定の方向からキャプチャしたもの）：描画磁力線数は 200 本，コンターは 2 面（1 面は垂直面であるため表示されていない）でありプラズマ圧力を示している．

図 5 に本研究の実験システムの概要図を示す．6 台のデータ処理サーバが 10G スイッチによりネットワーク接続されている．各データ処理サーバは 10G ネットワークカードを有する．本実験システムは外部ネットワークから独立しており，本実験システムに外部トラフィックが流れることはない．なお，6 台のデータ処理サーバは図 1 の Gfarm システムのファイルシステムノード（FSN）とクライアントノード（CN）を兼ねている．システムにはメタデータサーバ（MDS）が 1G 接続されている．

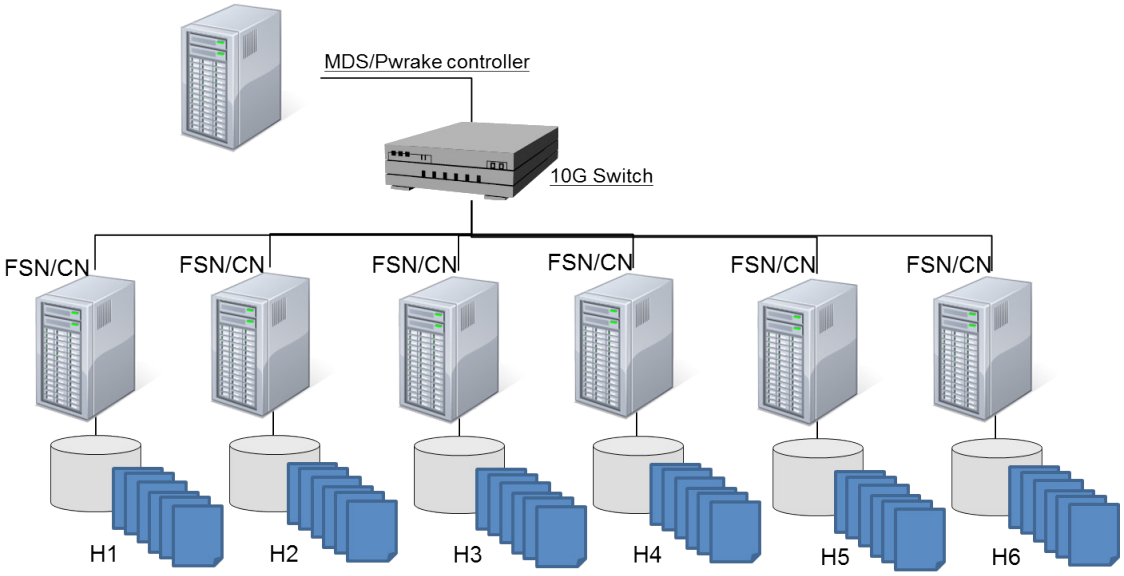

図 5　本研究の実験環境：並列処理サーバ・メタデータサーバ・10G スイッチから構成されている．

2　図 4 に示す各時間ステップの 3D 可視化結果を一つの時系列可視化データファイルに統合する後処理は本研究対象には含めない．

3. 予備実験

本節では，2 節で述べた実データ実験のための予備実験を行う．予備実験は，図 5 のシステムで Gfarm のクライアントノード上でバーチャルオーロラツールを用いて対象データを読み込み，2.3 節で述べた可視化処理を行う．図 6 に本研究のデータ処理時間の定義を示す．予備実験では，1 データファイルに必要となる読み込み時間（データ I/O 時間）とデータ処理時間（可視化処理時間）を比較する．なお，全処理時間は，これらにバーチャルオーロラツール（アプリケーション）の起動時間及び終了時間を加えた時間となる．

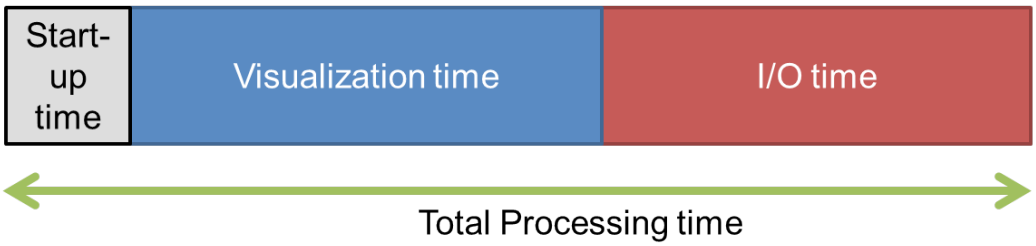

図 6 （1）データ I/O 時間（I/O time）と（2）可視化処理時間（Visualization time）：両者の合計にアプリケーション起動・終了時間（Start-up time）を加えた時間が全処理時間（Total Processing time）となる．

予備実験は，データ処理（可視化）に図 5 のうちの 1 台の計算機（クライアントノード）を用い，対象とするデータファイルと同じノードに保存する．これは，図 1（FSN3 および FSN4）で示した FSN が CN と一致する場合に該当する．本節の予備実験では処理データファイル数を図 3 に示すシミュレーションの時間ステップ番号 1 から 144 までの 144 とし，1 ファイル当たりの平均処理時間を求める．

図 7（1）に，本予備実験の結果を示す．図 7 は 144 回の測定の平均値を示している．これまで，多くの大規模数値データ処理ではデータ処理の並列化が重視され，データ I/O 時間は無視されることが多かった．しかし，図 7(1) では可視化処理時間（データ処理時間）とデータ読み込み時間（データ I/O 時間）は比較的同規模であり，I/O 時間が無視できないことが分かる．

予備実験として，図 7(1) と同様のデータに対してデータファイルを CN と異なる FSN に保存した場合についても調べる．すなわち，図 1 の CN1 と FSN1 のように，FSN が CN と異なる場合である．この場合の 1 データファイルに必要となる読み込み時間（データ I/O 時間）とデータ処理時間（可視化処理時間）を図 7(2) に示す．図 7(1) と図 7(2) を比較すると，両者の可視化処理時間はほとんど変わらないが，Gfarm のリモートストレージ（FSN）から参照する場合は，ローカルストレージを参照するよりも I/O 時間が約 1.6 倍大きくなる．

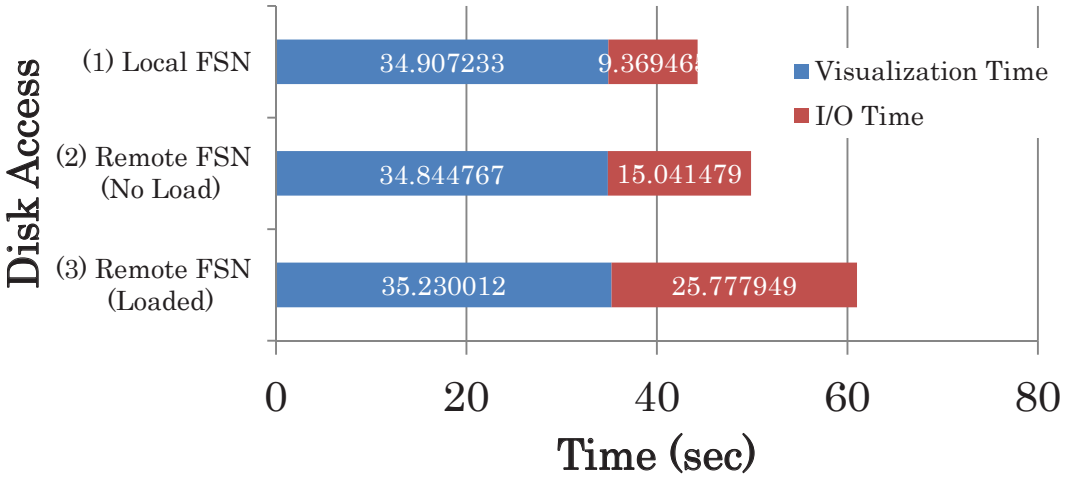

図 7 予備実験の結果：本研究の対象となるデータファイル（1 ファイル）を読み込み（I/O）後に可視化（Visualization）した場合のそれぞれの時間．（1）CN が FSN を兼ねる場合（ローカル FNS）（上図），（2）CN が異なる FSN からデータファイルを読み込む場合（リモート FSN）（中図），（3）リモート FSN に外部から他の負荷がかかっている場合（下図）．可視化処理時間はどの場合もほぼ一定であるが，データ I/O 時間はリモート FSN の場合は，外部負荷がない場合に 1.61 倍，外部負荷がある場合は 2.75 倍となる．

リモート FSN からアクセスする場合の I/O 時間増加の傾向は，計算の大規模並列化においてはさらに大きくなる．計算期間に流れる I/O データ量が増大し，ネットワークの輻輳が発生するためである．また，リモート FSN の負荷が上昇する場合にも，I/O 速度は低下する．FSN は任意の CN からのアクセスが常に起こり得るため，負荷を予測することができない．図 7(3) は，測定条件を図 7(2) と同様に設定し，さらにリモート FSN の全コアに外部から負荷をかけた場合の CN ノードの可視化処理時間および I/O 時間である．本実験では，CN からのアクセスとは別にリモート FSN に対して CPU コア数と同じだけのディスク I/O プロセス（データ読み出し）を外部から継続的に行うことにより，FSN を高負荷状態にした．その結果，図 7(2) と比して図 7(3) の I/O 時間はさらに長くなり，図 7(1) と比べると 2.75 倍の処理時間がかかっている．

本実験の可視化処理では図 6 に示すようにデータ読み込み（I/O）と可視化処理はシーケンシャルに実行されるため，同時に行われることがない．図 7(1) のように CN がローカル FSN のみを参照する場合には他の CN からのアクセスがないため，I/O 処理と可視化処理はプロセスごとに独立する．全処理時間の高速化のためには，図 7(3) のようにリモートアクセスがある場合は望ましくないことが分かる．本研究では，すべてのデータ処理を図 7(1) のような CN がローカル FSN のみを参照することとする．

2.3 節で述べたように，本実験の可視化処理はプロセス間通信が発生しないため，可視化並列処理のスケーラビリティー向上は難しくない．近年，コンピュータシミュレーションの出力データファイルサイズは増大化する傾向にあり，シミュレーションポスト処理（データ可視化）時間に比して I/O 時間が無視できない．全処理時間の短縮のためには I/O 時間の短縮が必須であり，図 7 で議論したようにローカルディスクを優先的に処理するアルゴリズムが有効である[3]．

4. 基礎実験

4.1. 基礎実験の目的

本節では，3 節の予備実験結果をもとに，Gfarm/Pwrake による並列分散処理で高いスケーラビリティーを達成するための基礎実験を行う．3 節の予備実験では，Gfarm のクライアントノード（CN）を用いてデータ処理を行う場合には，ローカル FSN と比較してリモート FSN アクセスによる処理速度の低下が指摘された．Gfarm/Pwrake では，2.1 節で述べたように，複数 FSN においてファイル複製が保存される場合にはローカル FSN 上のファイルアクセスが優先する設定が可能である．言い換えると，図 5 の様な CN と FSN がすべて一致するシステムの場合，すべての FSN に対象ファイルの複製が保存されている場合には，CN は常にローカル FSN のファイルにアクセスする．そこで，本実験ではストレージコストは高いがすべての FNS にすべてのファイルを配置することとする．（本研究では，最高ストレージコストファイル配置と呼ぶ．）

本基礎実験では，図 5 の全クライアントノード（すなわち全ファイルシステムノード）を用いてデータ処理を実験ごとに 1 回行う．プロセス数によるスケーラビリティーを調べるため，図 5 のクライアントノード数 N と各クライアントノードにおけるデータ処理プロセス数（用いるコア数）M を変更して処理速度の計測を実施する．なお，処理データファイル数を図 3 に示すシミュレーションの時間ステップ番号 1 から 144 までの 144 とする．

4.2. 基礎実験①（M=1 の場合）

基礎実験①では，複数のクライアントノード上でコア数 M を 1 と固定して 1 ファイル当たりの平均処理時間を測定する．データファイル配置は，4.1 節で述べた最高ストレージコスト配置である．図 8 に測定結果を示す．2.3 節で示したとおり本可視化処理にはプロセス間通信がないため，ノードごとにコア数（プロセス数）M を 1 と固定した場合，N=1 ～ 6 までほぼ完全なスケーラビリティーを達成していることが分かる．

本基礎実験結果から，最高ストレージコスト配置を行う場合には，CN 数を 10 台，100 台，1000 台と増やしても高い高速化が期待できる．並列化する場合にオーバーヘッドになるのは，メタデータサーバへのアクセストラフィックや CPU 負荷，DB 検索速度である．NICT サイエンスクラウドでは，Gfarm による広域分散ファイルシステムを実験的に運用しているが，2 億ファイルを超える場合でもメタデータベースの応答には影響がないことが分かっている．

3　なお，本研究における I/O 時間測定は，バーチャルオーロラツールのデータ読み込みモジュールの先頭と末尾で測定している．したがって，測定結果は厳密なディスク I/O 速度には一致しない．

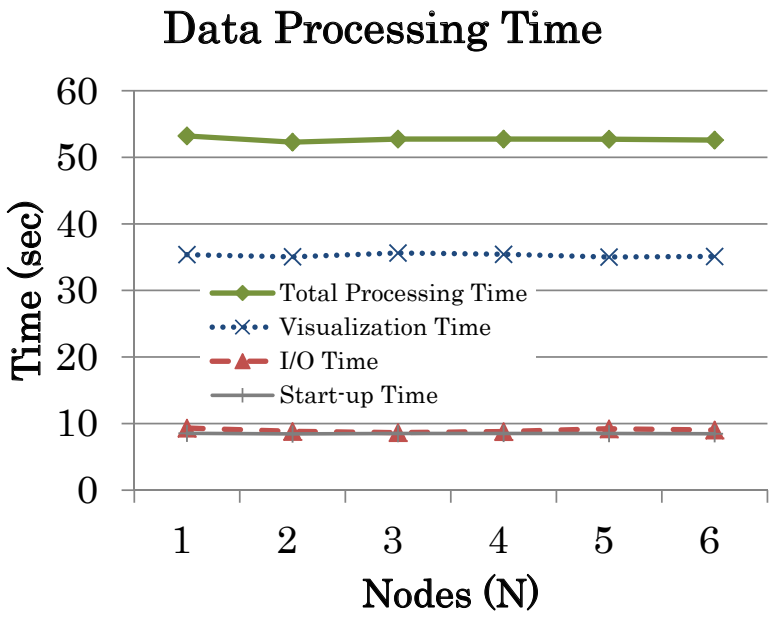

図 8 基礎実験①の結果：1ファイルのデータ処理時間（データ I/O 時間，可視化処理時間，アプリケーション起動・終了時間および全処理時間）のノード数 N 依存性（各ノードのプロセス数（コア数）M を 1 に固定した場合）.

4.3. 基礎実験②

基礎実験②では，基礎実験①の結果を踏まえてノード数 N を 1 および 6 に固定し，プロセス数 M を 1 から 12 まで変更して処理を行う．データファイル配置は，基礎実験①と同様に最もストレージコストが高い配置である．図 9 および図 10 に，データ処理時間のプロセス数 M への依存性を示す.

図 9 は，ノード H2 におけるプロセス数 M に対する 1 ファイル当たりのデータ I/O 時間，可視化処理時間，アプリケーション起動・終了時間および全処理時間を示している．図 9 はノード H2 の結果であるが，他のノードでも同様の結果である．可視化処理時間については，M=8 まではほぼ一定の並列化効率を保っているが，M=8 を超えると低下する．これは，表 1 より対象となる CN のコア数が 8 であることから予想されるとおりである．しかし，可視化処理時間はアプリケーション起動・終了時間（図 6 の Start-up time）とともにプロセス数 M への依存性は小さく，全処理時間の増加には寄与していない.

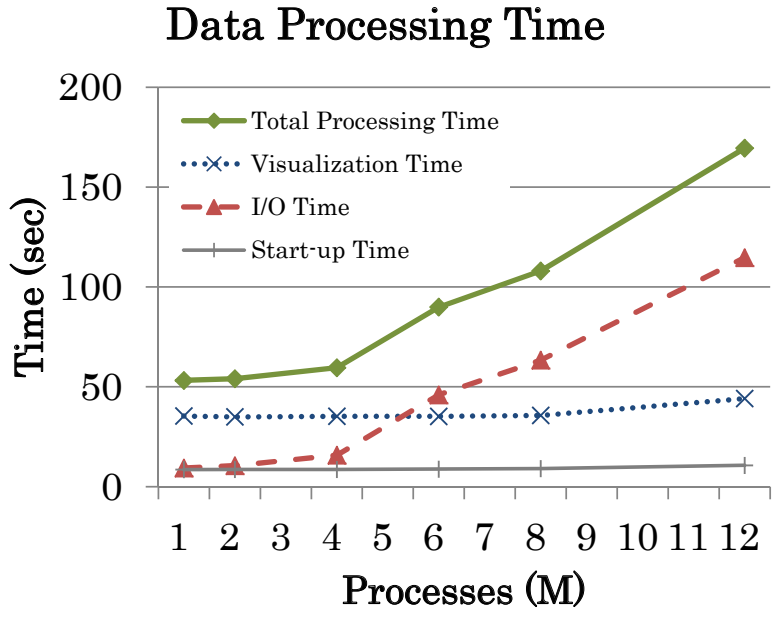

図 9 基礎実験②の結果：ノード数 1 のときの 1 ファイルあたりの平均データ処理時間（データ I/O 時間，可視化処理時間，アプリケーション起動・終了時間および全処理時間）のプロセス数 M の依存性.

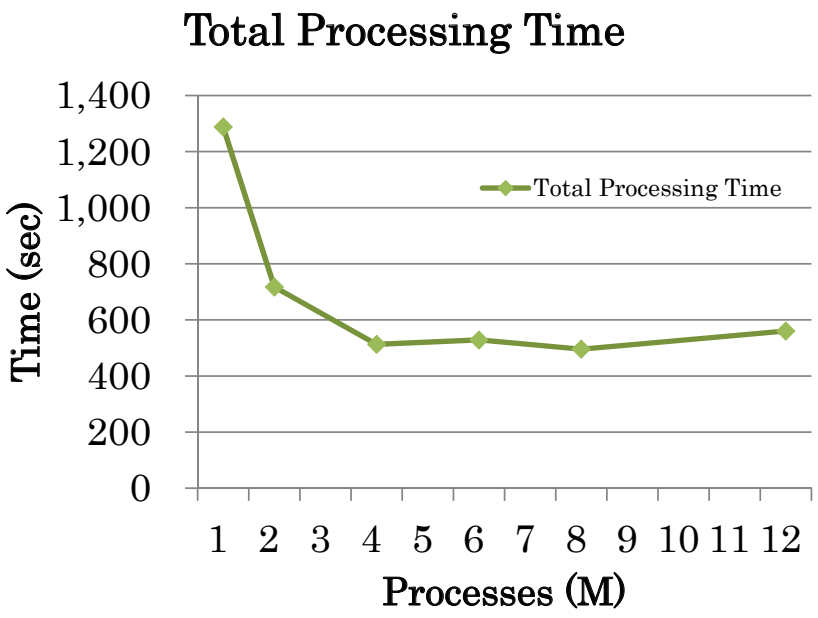

図 10　基礎実験②の結果（縦軸は全処理時間，横軸はプロセス数）：ノード数 6 の場合の並列処理による全処理時間.

表1　本研究のデータ処理サーバ緒元

CPU コア数	8
CPU	Intel Xeon X5550 2.67 GHz
メインメモリ	144 GB
オペレーションシステム	OpenSUSE 11.1 (x86_64)
ハードディスク構成 （Read および Write はカタログスペック）	SATA 3 × 4 (RAID5) Read: 371 MB/sec Write: 137 MB/sec
NIC	10GbE

一方，データ I/O 時間はプロセス数 M が増加するにしたがって増加する．これは，同じ CN で同時に複数のプロセスがローカルディスクを並列参照する場合にはディスクアクセス帯域の共有を行っていることによる．I/O 速度の低下は，特に M>4 で顕著である．一般に，SATA 系の HDD へのマルチアクセスにおいては，アクセス数が 2 ～ 8 以上に増加するほど I/O 性能は低下する傾向がある．

図 10 に，ノード数 N を 6 としたときの全データ処理時間の M への依存性を調べた結果を示す．図 10 の全データ処理時間は図 8 や図 9 のファイル単位での処理時間ではなく，6 クライアントノードを用いた全 144 ファイル全体の処理時間（処理開始時刻と処理終了時刻の差）を表している．処理時間が最も短かったのはプロセス数 M=8 の場合であり，1 ノード 1 コアで必要であった全処理時間 7741.7 秒に対して 495.7 秒で処理が終了した．図よりプロセス数 M=4 までは処理時間が短縮されるが，M>4 で処理時間は横ばいとなる．これは，図 9 に示すように M>4 では I/O の処理時間が増加することによる．

図 11 は，図 10 から求めた高速化率を示す．高速化率はノード数 N ＝ 1 でプロセス数 M=1 の場合の全処理時間を基準値として，図 10 の全データ処理時間よりノード数 N=6 における各プロセス数 M での処理時間から求める．その結果，最も高い高速化を達成したのはプロセス数 M=8 の場合であり，高速化率は 15.6 であった．

5.　実利用実験

5.1.　実利用実験の目的

前節の基礎実験では，6 台のクライアントノード（ファイルシステムノードを兼ねている）から構成される図 5 のシステムで Gfarm/Pwrake により 144 ステップ分の数値シミュレーションデータを並列処理する実験を行った．本節では，

このパラメータにより図5のシステムにおいて多ステップシミュレーションデータ処理を1回行い，その結果を評価する．処理データは，2.3 節で述べたデータファイルのうち，ステップ 1 からステップ 780 まで 780 ファイルを対象とする．本 Global MHD シミュレーションの 780 ステップ (時間分解能 0.5 秒) は，実時間で 390 秒（6 分 30 秒）に該当する．6 分 30 秒は地球磁気圏現象としては短時間であるが，0.5 秒は磁力線追尾 [9] などに必要な時間分解能である．

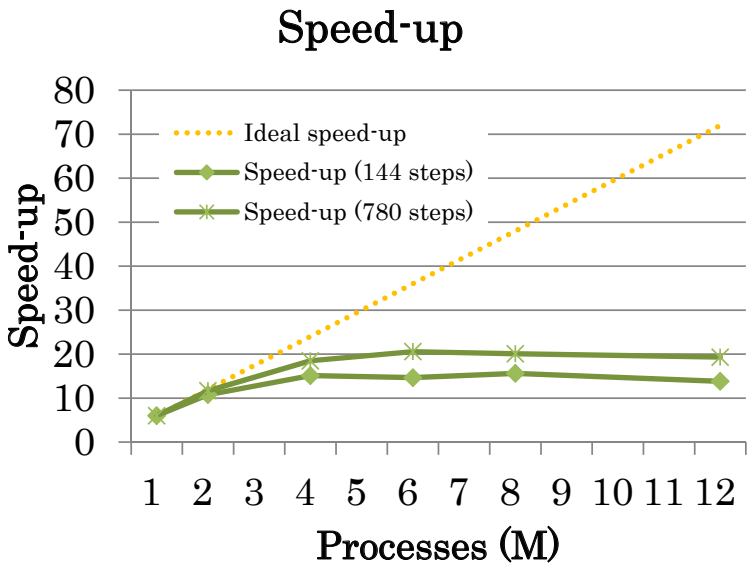

図 11 基礎実験②（144 ステップ）と実利用実験（780 ステップ）の結果：ノード数 N=6 の場合の並列処理による高速化率（Speed-up）．横軸はプロセス数．

5.2. データファイル処理時間と高速化率

図 12 に，実利用実験のファイルごとの処理時間を示す．図 12 は，処理を行った全 780 ファイルについて，それぞれの可視化処理時間とデータ I/O 時間（図 6）のデータファイルごとの時間変化を示している．横軸はファイル番号（計算機シミュレーションデータの時間ステップ番号）に対応する[4]．基礎実験②の高速化率結果である図 11 には実利用実験の高速化率も示されており，最高の高速化率は M=6 の場合で 20.6 である．実利用実験の高速化率は，図 12 の結果をもとに基礎実験②と同じ予備実験における 1 ノード 1 プロセスでの 1 ファイル処理時間から求めた．なお，図 12 の横軸は処理時刻ではなくデータファイルのステップ番号である．各処理時刻の可視化処理時間とデータ I/O 時間については，5.3 節において議論する．

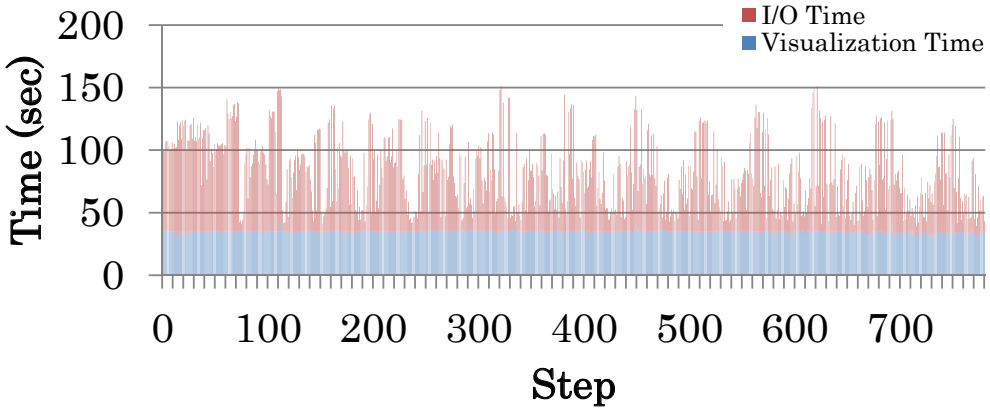

図 12 実利用実験のファイルごと処理時間：処理を行った全 780 ファイルについて，それぞれの可視化処理時間（青色）とデータ I/O 時間（赤色）を示している．

4 2.3 節で述べたとおり，本研究のデータ処理ではプロセス間通信は発生しない．また，ファイル配置は 4.1 節で述べた最高ストレージコスト（すべての時間ステップのシミュレーションデータが全ての FSN に配置されている）を採用している．2.1 節で述べたとおり，図 12 の横軸はファイル処理の順序とは必ずしも一致していない．

図 11 より，実利用実験の高速化率はプロセス数 M に関わらず基礎実験②の高速化率よりも高いことが分かる．実利用実験と基礎実験②との違いは処理したデータファイル数のみであり，その他の条件は同じである．基礎実験②の 144 ステップ処理よりも実利用実験の 780 ステップ処理の場合の方が高い高速化率を示す理由について，以下に検討・考察する．

5.3. 高速化率向上の検討と考察

図 12 によると各データファイルの全処理時間（図 6）において可視化処理時間はほぼ一定であり，780 ファイル処理の平均値は 35.07 秒である．これは, 予備実験である図 7(1) の値とほぼ等しい．一方, データ I/O 時間は先頭から 60 ステップ目までと 60 ステップ以降で傾向が異なっている．60 ステップまでは 60 〜 80 秒程度と大きいが，ファイル処理が進むにしたがってばらつきが大きくなる傾向が読み取れる．

これを確認するため，6 台のクライアントノード（図 5）のコアごとのデータファイル全処理時間（図 6）のヒストグラムを図 13 に示す. 図 13 の各ヒストグラムの色は処理したデータファイルの順序を示している[5]. 横軸は処理時刻であり，本図ではデータファイルごとにデータ I/O 時間と可視化処理時間をあわせた全処理時間を一つの色で示している．どのノードにおいても先頭または 2 番目のファイルについては全処理時間が各ノード内でほぼ一致しているのに対し，処理が進むにつれて全処理時間にずれが発生している[6]．また，全処理時間は処理が進むにつれて短くなる傾向にある．図 12 より可視化処理時間はほぼ一定であるので，全処理時間の短縮は I/O 時間の短縮を意味している．図 12 の後半の時間帯のように各コアでの処理時刻にずれが発生すると，ファイルアクセスを行う時刻がコアごとに異なってくるため，I/O の分散化が達成されたと考えらえる．

この効果を確かめるため，実利用実験において全クライアントノードの可視化処理時間とデータ I/O 時間の時間変化を示したのが図 14 である．図 14 では，図 12 に示した 6 ノードの全 6 プロセスについて，先頭から処理するファイル番号が等しい可視化処理時間およびデータ I/O 時間のステップごとの平均値を求めた結果である．たとえば，左から 2 つめの点は，各 36 プロセスが 2 番目に処理したデータファイルの可視化処理時間とデータ I/O 時間を示している．横軸の時刻は，36 プロセスの各処理時刻の平均で定義した[7].

図 14 では可視化処理時間はほぼ一定であり，図 12 の結果と一致している．一方，データ I/O 時間は横軸の処理時刻により異なっており，全体に処理が進むごとに短くなる．これは，処理が進むごとに I/O 効率が上がっていることを示唆している．この理由を以下に考察する．

図 9 より，クライアントノード内の I/O の並列数が増えるほど I/O 時間は増加し，I/O 効率は低下することが分かっている．図 12 の最初のステップでは各ノードにおいて 6 プロセス全てで同時にファイル読み込みが始まっており，ディスクへの同時アクセスが行われるため I/O 効率は悪い．後半のステップでは，図 12 から分かるようにプロセス間での処理のばらつきが発生するためディスクへの同時アクセス数が減少し，結果的に I/O 効率の向上の理由となっている．処理の後半（時刻 1500 秒以降）では，I/O 時間がさらに減少している．これは，すべての処理が終了したプロセス（コア）が増え始めていることを示している．

本研究では, 自発的なデータ処理時間のばらつきによる I/O の分散と効率化が実現した. 一般に, Global MHD シミュレーションでは時間ステップごとのデータサイズは一定であり，ファイルごとに I/O 処理と可視化処理は独立に行われることが多い（図 6）. Gfarm/Pwrake によるこのようなシミュレーションデータの並列可視化処理（ポスト処理）では，意図的にプロセスごとに I/O 処理を分散化するアルゴリズムを導入することで，さらに高速化が可能となることが示唆された．

5　図 13 のフォーマットは図 2 右側のフォーマットに該当する．

6　シミュレーションデータの可視化処理では，データサイズは全てのデータファイルで等しいが，描画する磁力線の長さや複雑さが変動する．そのため，データ可視化処理時間（データ処理時間）はファイルごとに異なってくる．5.1 節で述べたとおり実利用実験の 780 ステップは 6 分 30 秒に該当するが，この期間は地球磁気圏構造に大きな変化がないため，可視化処理時間はほぼ一定である．

7　2.1 節で述べたとおり，Gfarm/Pwrake のワークフローでは，シミュレーションデータファイル順にデータ処理が行われるとは限らない．そのためにグラフの点が横軸方向に逆転することがある．

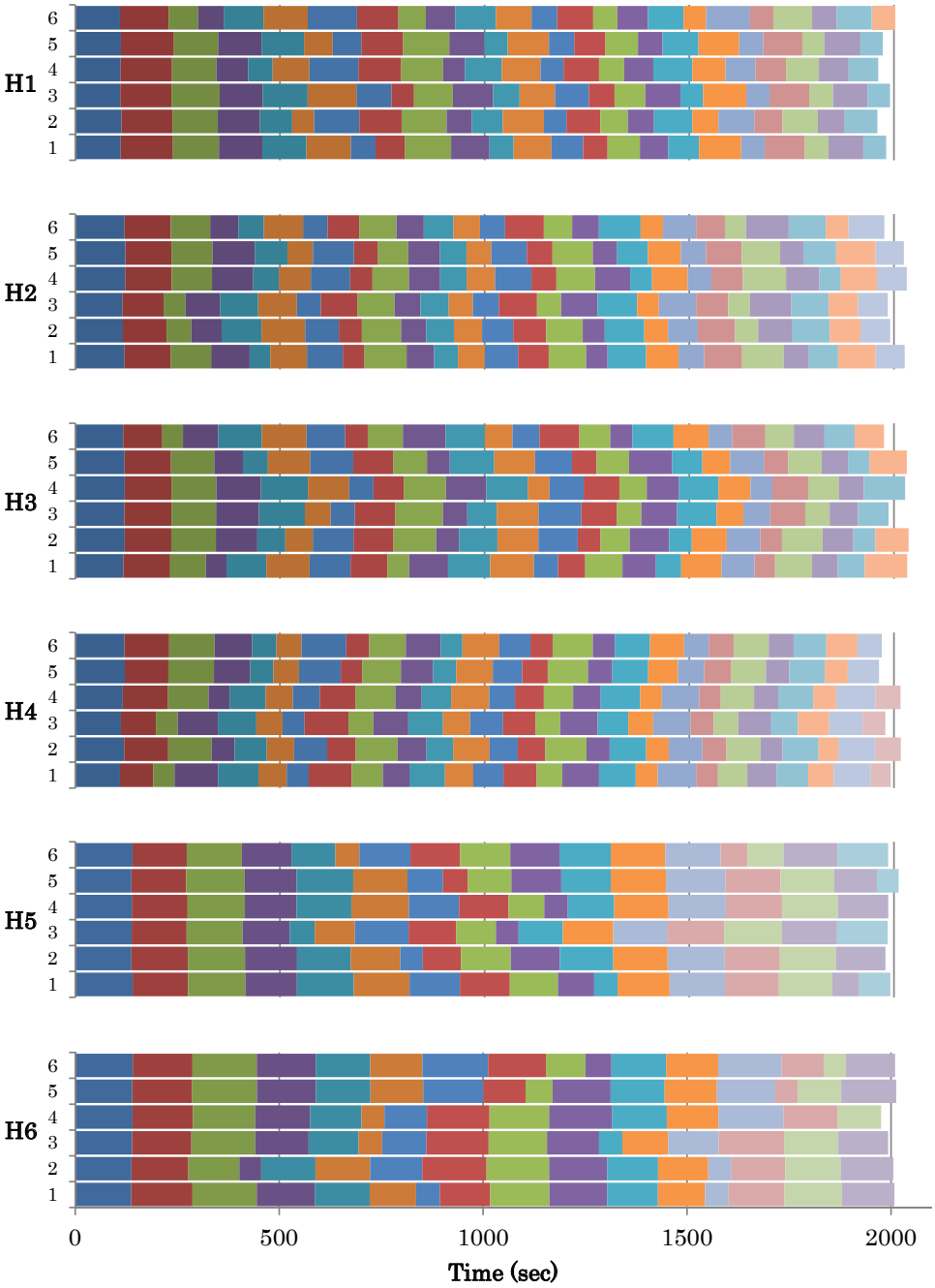

図 13 実利用実験の結果：各クライアントノード（H1 ～ H6）のコアごとのデータファイル処理時間のヒストグラム.

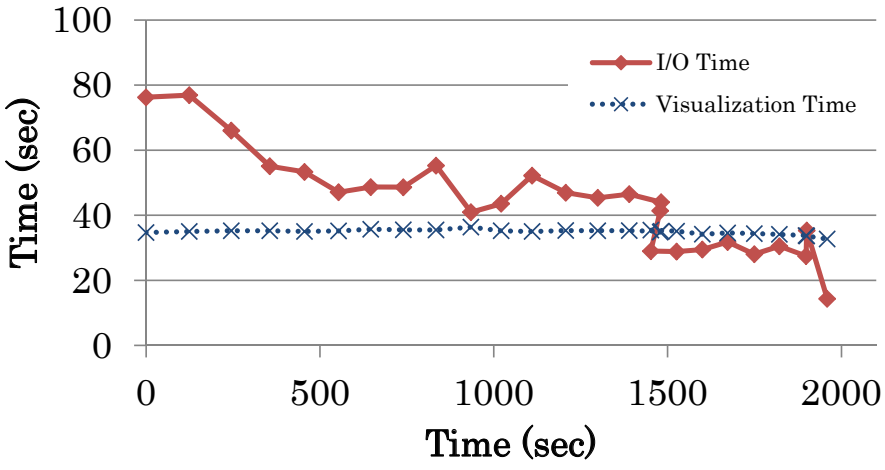

図 14 実利用実験の結果：全クライアントノードの可視化処理時間とデータ I/O 時間の平均値の時間変化

6. まとめ

計算指向型の HPC（High Performance Computing）は数値計算に有効な技術であるが，出力された大規模データのポスト処理に対して有効とは限らない．すなわち，大規模数値データを解析するための新しい科学研究手法が必要であり，Gray はデータ指向型科学研究手法を第 4 のパラダイムと呼んだ [1]．

Many-Task Computing（MTC）はこのような背景において，HPC と対をなす概念として提案された．MTC では，多様な計算機リソースを融合し，データ分散，並列データ処理やコンピュータとデータファイルをローカライズする工夫など，総合的なデータ処理環境をめざす．すなわち，スーパーコンピュータが HPC 的であるとすると，MTC 的であるのがサイエンスクラウドであると筆者らは考える．

MTC を実現するために我々が解決せねばならないサイエンスクラウド技術はまとめると次のようになる．① MTC を実現するために CPU，I/O およびネットワークスループットなどの分散化とその融合を行う技術，②クラウド内のヘテロ環境における分散データ処理のスケーラビリティーの達成，③目的達成のためのクラウドコストの最適化の解決手法の確立．

本研究の実験では①の CPU と I/O の分散を試みた．分散ファイルシステムと分散処理ワークフローツールにより，ネットワークスループットを最低にすることで，大規模ポスト処理に有効な並列分散処理を提案した．

一方で，③の視点からは本提案が最も有効であるとは限らない．本研究は閉じたネットワーク内において行ったが，これを複数ユーザのデータ処理が融合するサイエンスクラウドで行った場合には，コスト問題が発生する．たとえば，本研究ではストレージコストが最も高い方法（すべてのファイルシステムノードにすべてのデータファイルを配置する）を採用した．これは，例えば，100TB のデータを 100 サーバで処理する場合には合計で 10PB のデータ領域を必要とするため，現在のシステムでは現実的ではなくなる．また，分散ファイルシステムをクラウド内で複数ユーザが利用している場合には，データ処理ノード（クライアントノード）がファイルシステムノードとして機能することがボトルネックになる．すなわち，別ユーザがクライアントノードを兼ねたファイルシステムノードにアクセスすると，そのクライアントノードの負荷が上昇し，分散処理の最適化が崩れることがある．

②のヘテロ環境における分散データ処理を模式図にしたのが図 15 である．図 15 は複数のクラスタ（または HPC サーバ）がネットワーク接続されており，クラスタを随時追加することができる可用性の高いスケーラブルクラスタシステムのイメージ図である．NICT サイエンスクラウドを含む多くのサイエンスクラウドでは，このようなスケーラブルクラスタシステムであり，今後のサイエンスクラウドの主流となる．図 15 の様なヘテロ計算機環境においてスケーラブルなデータ処理を実現するためには，サーバ内の分散データ処理の高速化（HPC 的高速化），クラスタ内でのデータ処理の高速化（MTC 的高速化），クラスタ間での分散データ処理の高速化（HTC 的高速化）の段階を経ることが有効な手順であろう．本研究は，クラスタ内でのデータ処理の高速化を行う MTC 的高速化研究である．Gfarm/Pwrake を用いたクラスタ間での高速分散処理の研究もある [5] が，本研究成果を活かし，クラスタ内で最高の処理効率を拡張した MTC 的高速分散データ処理技術の発展が今後は求められるであろう．NICT サイエンスクラウドは図 15 の環境をすでに実現しており，MTC 研究と HTC 研究のテストベッドとしての機能が期待される．

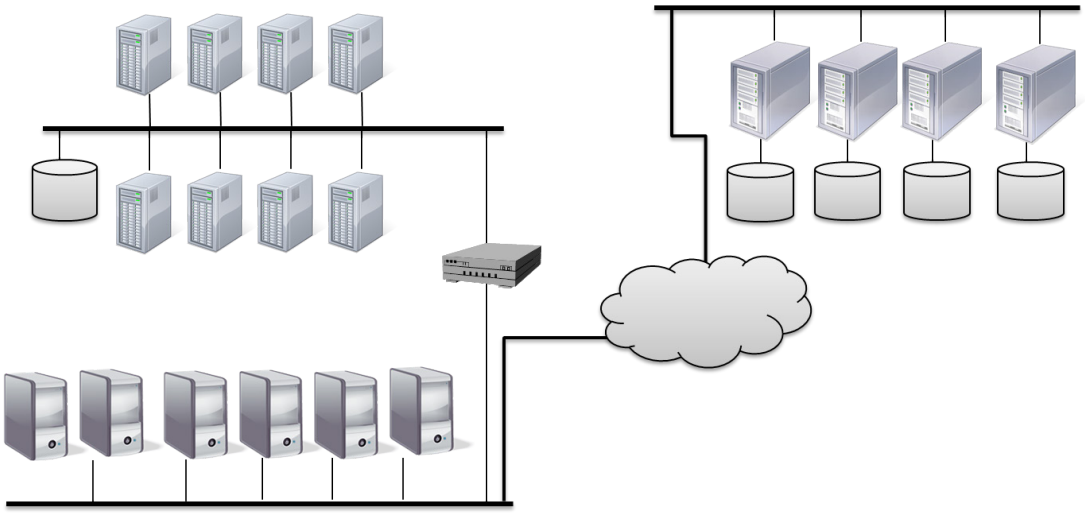

図 15　可用性の高いスケーラブルクラスタシステムのイメージ図：複数のクラスタ（または HPC サーバ）がネットワーク接続されており，クラスタ単位で随時追加することができる．

参考文献

1) Hey, T., S. Tansley, and K. Tolle. The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, Washington, 2009.
2) 平尾公彦，横川三津夫，京コンピュータと計算科学研究機構 (< 特集 > 次世代スーパーコンピュータ「京」: 動き出した大型プロジェクトの全体像)，NII 論文 ID(NAID) :110008686987，日本物理學會誌 66(7), 2011-07-05，524-528.
3) 2012 年第 3 回 ScienceCloud ワークショップ http://ceng.usc.edu/~simmhan/ScienceCloud2012/， 2012.
4) Tatebe,O., K. Hiraga and N. Soda. Gfarm Grid File System. New Generation Computing,vol.28, no.3, 2010， p.257-275.
5) 田中昌宏 , 建部修見 , 並列分散ワークフローシステム Pwrake による大規模データ処理 , 宇宙科学情報解析論文誌 , 1, 2012, p.67-75.
6) HPCI コンソーシアム Web サイト，http://hpci-c.jp/.
7) Murata, K. T., S. Watari, T, Nagatsuma, M. Kunitake, H. Watanabe, K. Yamamoto, Y. Kubota, H. Kato, T. Tsugawa, K. Ukawa, K. Muranaga, E. Kimura, O. Tatebe, K. Fukazawa and Y. Murayama, A Science Cloud for Data Intensive Sciences, Data Science Journal, Vol. 12, 2013, pp.WDS139-WDS146.
8) Fukazawa, K., Y. Aoyama, T. Ogino and K. Yumoto, Response of Cross Polar Cap Potential to IMF and Velocity of Solar Wind, J. Atmos. Solar-terr. Phys. 72, 2010, doi:10.1016/j.jastp.2010.06.002.
9) Kubota, Y., K. Yamamoto, K. Fukazawa and K., T. Murata, Visualization of the Flux Rope Generation Process Using Large Quantities of MHD Simulation Data, Data Science Journal, Vol. 12, 2013, pp. WDS134-WDS138.