ISSN 1344—9567

スーパーコンピューティングニュース

Vol. 11 No. Special Issue 2 2009. 3

特集：試行期間限定T2Kオープンスパコン（東大）HPC特別プロジェクト

東京大学情報基盤センター
（スーパーコンピューティング部門）

目　　次

特集：試行期間限定 T2K オープンスパコン（東大）HPC 特別プロジェクト

特集号「試行期間限定 T2K オープンスパコン（東大）HPC 特別プロジェクト」発刊にあたって

中島 研吾

東京大学情報基盤センター

東京大学情報基盤センター（以下「本センター」）では 2008 年 6 月から新スーパーコンピュータ「T2Kオープンスパコン（東大）」の稼動を開始した。本システムは筑波大，東大，京大の 3 大学で定められた「T2Kオープンスパコン仕様」に基づき日立製作所が製作した 952 ノード，約 15,000 コア，ピーク性能 140 TFLOPSのクラスタ型コンピュータシステムである[1]。

本センターでは，2008 年 10 月よりの本運用に先立ち，9 月末までは，「試行期間」として無料でご利用いただいた。特に大量の計算リソースを必要とする利用者が優先的に利用できるように公募型「試行期間限定 T2K オープンスパコン（東大）HPC 特別プロジェクト」を実施した。本プロジェクトは最大 512 ノード（8,192 コア）を使用可能である。

全部で 29 件の応募があったが：

- 自作コードによる研究であること
- 当該コードについて 1,000 コア以上の利用実績があること
- 計算結果が科学的に有用，あるいは社会的なインパクトがあると考えられること
- 当情報基盤センターの運用，利用者にとって有用な情報を提供すること
- 512 ノードの利用を目標としていること
- 計画に実現性があり，短期間で効果を示すことが可能であること

等を考慮して，表 1 に示す 10 件を採択した。うち，「次世代スーパーコンピュータ」のターゲットアプリケーションに選ばれているものが 5 件，「SC-XY Conference Series」[2]でアプリケーションの最高性能を競うGordon Bell PrizeのFinalistに選出された経験のあるものが 2 件である。多くのグループは 10 月以降もT2Kオープンスパコン（東大）を利用していただいている。また，いわゆる科学技術シミュレーションだけでなく，「大規模テキスト処理」に関連した課題も採択されている。

---

[1] http://www.cc.u-tokyo.ac.jp/ha8000/

[2] http://www.sc-conference.org/　毎年 11 月にアメリカで開催されているIEEE主催による国際会議The International Conference for High Performance Computing Networking, Storage, and Analysisのこと

本特集号は，「試行期間限定 T2K オープンスパコン（東大）HPC 特別プロジェクト」各グループから提出いただいた報告書をまとめたものである。単に得られた知見だけでなく，「T2K オープンスパコン（東大）」への要望，不満も含めて忌憚の無い意見を記述いただいている。

なお，表 1 にも示したように「3 次元不均質場での地震波伝播の大規模シミュレーション（代表：古村孝志教授（東大・情報学環））」については，別途出版したスーパーコンピューティングニュース特集号「ペタスケール・シミュレーションの信頼性」に「地震波伝播と強震動の大規模並列 FDM シミュレーション」と題して掲載されているのでそちらをご覧いただきたい。

表 1　「試行期間限定 T2K オープンスパコン（東大）HPC 特別プロジェクト」採択課題

課題名・	代表者（所属）
全球雲解像正 20 面体格子非静力学大気モデル（NICAM）の開発	佐藤正樹（東大・気候システム研究センター）
海洋循環形成プロセスの高解像度シミュレーション	羽角博康（東大・気候システム研究センター）
高性能直接法 N 体計算ベンチマーク	似鳥啓吾（東大・理学系研究科）
T2K オープンスパコンへのインヤン地球ダイナモコードの移植	陰山 聡（海洋研究開発機構）
3 次元不均質場での地震波伝播の大規模シミュレーション[3]	古村孝志（東大・情報学環）
超並列計算によるマルチスケール・マルチフィジックス心臓シミュレーション	久田俊明（東大・新領域創成科学研究科）
革新的シミュレーションソフトウェア	加藤千幸（東大・生産技術研究所）
密度行列繰り込み群法と行列対角化による強相関量子系のシミュレーション、	町田昌彦（日本原子力研究開発機構）
プロセッサアフィニティ制御を組み込んだフレームワークによる実用大規模並列シミュレータの性能評価	小野謙二（理化学研究所）
GXP システムとそれを用いた大規模テキスト処理の実行	黒橋禎夫（京大・情報学研究科）

512 ノードジョブを実行する特別サービスは週末を中心に実施されるため，1 グループが 512 ノードを占有できる時間は概ね 1 日程度となった。従って，科学的な成果を求めるより

3　「スーパーコンピューティングニュース　特集：ペタスケール・シミュレーションの信頼性」（Vol.11 No. Special Issue 1, 2009.2）に掲載

はベンチマーク，性能評価が中心となっている。

このような限られた時間の中，またシステム稼動初期の非常に不安定な状態で，有用な知見を出していただいた各グループの代表者，メンバーの皆様には，東京大学情報基盤センタースーパーコンピューティング部門教職員一同，心から感謝の意を表したい。本センターにとっても，512 ノード，8,192 コアという大規模なシステムを継続的に運用することは貴重な経験であり，その後の円滑な運用に大きく貢献している。

今後，科学技術シミュレーションを中心として，計算の大規模化が進み，10 万コア，100 万コアを利用することが日常化することもそれほど遠い将来のことではない。今回は 512 ノード，8,192 コアが最大であるが，本特集号がペタスケール，エクサスケール規模の計算に向けての出発点として資することができれば幸いである。

超並列スカラー機での全球雲解像モデルNICAMの性能評価

山田 洋平[1)]　富田浩文[1)]　佐藤正樹[1) 2)]

(1)(独)海洋研究開発機構/地球環境フロンティア研究センター

(2)東京大学気候システム研究センター

１．はじめに

大気大循環モデル（AGCM: Atmospheric General Circulation Model）は大気現象の理解や日々の天気予報や気候変動予測に用いられており，気候研究には欠かせないものである。従来のAGCM では水平方向の格子解像度が数十～百 km であり，大気の鉛直方向の運動は水平方向の運動に対して十分に小さいので静力学近似を認めたプリミティブ方程式系が用いられていた。しかし静力学近似を適用した方程式系では深い対流による個々の積雲を表現することができない。静力学近似を用いた従来型のAGCMでは深い対流を統計的な手法によって再現していた。このような手法を積雲パラメタリゼーションと呼ぶ。こうした積雲パラメタリゼーションには不確実性が含まれるため，予測結果の信頼性を減少させてしまうおそれがある。AGCMによるシミュレーションの信頼性を向上させるための方法の一つとして積雲パラメタリゼーションによる不確実性を除去する必要がある。特に熱帯の積雲活動は地球規模の大気大循環に大きな役割を果たしており，個々の積雲対流システムを陽に表現することは重要である。これらの現象をシミュレーションするためには数 km スケールの水平格子を用いてシミュレーションをおこなう必要がある。

正20面体非静力学大気モデル(NICAM：Nonhydrostatic ICosahedral Atmospheric Model)は一つ一つの雲を表現するのに積雲パラメタリゼーションを用いずに陽に表現する全球雲解像モデルとして開発された。そのため従来のAGCMが有した不確実性を除去し，より精緻なシミュレーションが可能となり，図１に示すような現実的な大気現象を再現することが可能となった。NICAM を用いたシミュレーションではメソスケールの対流システム，クラウドクラスター，スーパークラウドクラスター，MJO（マッデンジュリアン振動）などの階層構造が精緻に再現された（Miura et al, 2007)。これらの現象は熱帯の対流活動に大きな寄与を持ち，シミュレーションの信頼性を向上させることができるようになった。それだけでなく全球を水平格子スケール数 km で覆ったシミュレーションは個々の積雲を陽に取り扱うことができるので現象の議論や理解を深めることが可能となる。

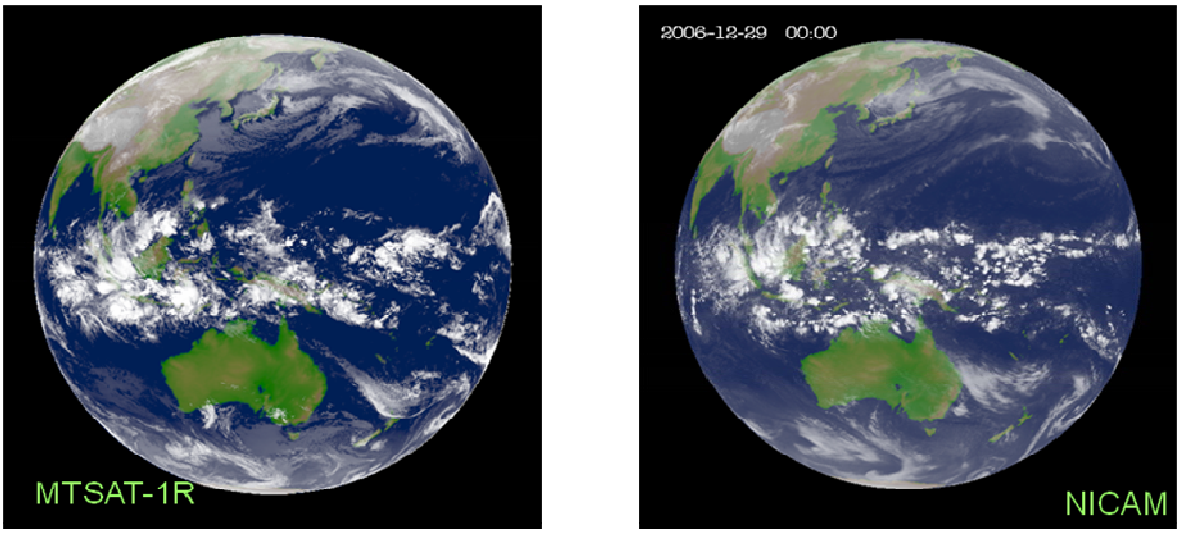


図１：衛星画像と NICAM によるシミュレーション結果

左図が MTSAT を示し，右図が NICAM によるシミュレーション結果を示す。

こうした全球の大気現象を高解像度でシミュレーションすることが可能になった一つの要因は近年までの大型計算機の発達が挙げられる。NICAMは東京大学気候システム研究センター(CCSR)と(独) 海洋研究開発気候/地球環境フロンティア研究センター(JAMSTEC/FRCGC) の協力の下で，(独) 海洋研究開発機構/地球シミュレータセンター(JAMSTEC/ESC)の地球シミュレータ上で開発がおこなわれてきた。地球シミュレータ はベクトル型演算機を有しているため，NICAMはベクトル型計算機向けに最適化が施されている。しかし近年のスーパーコンピュータの動向は膨大な数のスカラ型演算機をもつクラスタ型マシンが主流となっている。次世代のAGCMであるNICAMは物理的なパフォーマンスはもとより，計算効率においても有望でなければならない。ところがスカラ型計算機とベクトル型計算機は互いにアーキテクチャが異なるため，ベクトル型計算機に向けて最適化された現状のNICAMはスカラ型計算機上では高い計算効率での運用には適していないという問題があり，スカラ型計算機に向けた最適化を施す必要がある。

現在，次世代10ペタフロップスコンピュータ（京速計算機）の開発が進められている。NICAMは京速計算機のターッゲトアプリケーションの一つに選定されており，京速計算機上でさらなる知見を得ることが期待されている。我々は京速計算機の計算機資源を生かして全球で400m格子の実験を目標としている。また京速計算機上で数km格子の全球雲解像実験を現在おこなっている積分期間よりも長期にわたる実施やアンサンブル実験を実施する予定である。これらの実験を京速計算機のスカラ部分で高い計算効率で実施するためにはスカラ型計算機に向けた最適化は必要不可欠なものといえる。

本研究ではスカラ型計算機向けに最適化されていない現状のNICAMの計算効率をスカラ型計算機上で性能計測を実施する。そこで既存の大型スカラ型計算機(T2K東大・T2K筑波)を京速計算機と見做してNICAMの現状でのスカラ型計算機上での演算性能を測定した。また地球シミュレータでも同じ条件の実験を実施して演算性能を計測し，スカラ型計算機とベクトル型計算機での演算性能を比較した。

本稿では，2章でNICAMの計算手法の概要を述べる。3章では計測を実施した時の実行環境を述べる。4章ではそれぞれの計算機における計測結果とスカラ型計算機とベクトル型計算機の比較を示す。5章で計測結果のまとめと今後の展望を述べる。

２．NICAMの概要

NICAM の詳細は参考文献(Tomita and Satoh, 2004, Satoh et al, 2008, Tomita et al, 2008)に譲るがここでは簡単に概要を述べる。支配方程式は音波を含んだ完全圧縮な非静力学方程式系を用い，質量と全エネルギーの保存を考慮に入れたスキームを採用している。空間方向には有限体積法を用いて離散かを施している。時間積分には伝搬の遅い波と速い波を分けた time-splitting method を用いる。遅い波は 2 or 3 次のルンゲクッタ法を用いており，早い波は水平方向に陽解法を用い，鉛直方向には陰解法を用いる HEVI 法に基づき forward-backward 法を採用している。この手法では鉛直方向に 1 次元のヘルムホルツ方程式を解く必要がある。地形を表現するために地形に沿った座標系を適用し，鉛直方向にはローレンツグリッドを採用している。水平方向には修正型正二十面体格子系(Tomita et al, 2001, 2002)を用いている。

ここでは二十面体格子の生成方法を簡単に説明する。全球を 20 個の正三角形で覆った状態が図 2(a)に示す状態で G-Level 0 と称している。計算格子点は三角形の頂点に定義されている。20 個の正三角形を各々4 分割することによって生成される状態(図 2(b))が G-Level 1 と呼ばれ，さらに 4 分割することによって得られる格子系(図 2(c))が G-Level 2 と呼ばれる。上述の様に再起的に三角形を分割することにより水平格子解像度を向上させることができ，分割数に応じて G-Level n と定義している。表 1 に G-Level に対応した全球での格子点数と水平格子間隔を示す。

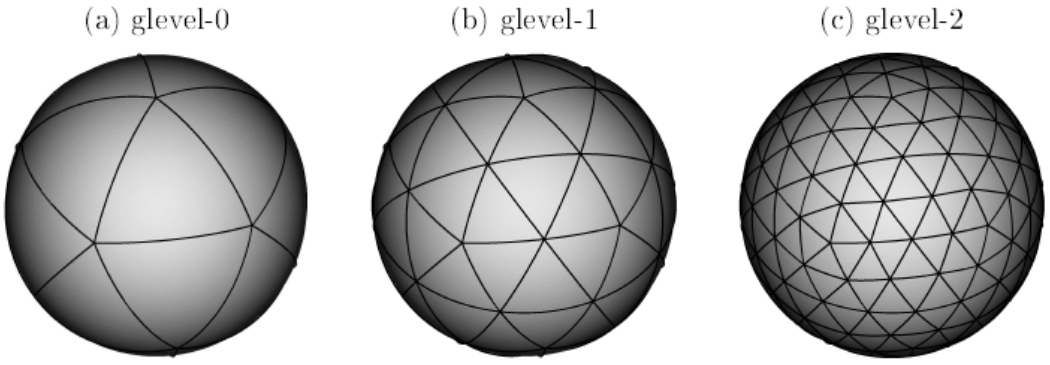


図２：水平格子点の概念図

(a)は G-Level 0 を示し，全球での水平格子点は 12 個存在する。(b)は G-Level 1 を示し，全球での水平格子点は 42 個存在する。(c)は G-Level 2 を示し，全体での水平格子点は 162 個存在する。

表１:G-Level と水平格子数及び水平格子間隔

G-Level	水平格子点数	水平格子間隔[m]
0	12	6000000
1	42	1500000
2	162	750000
.....	.....	.....
8	655362	28000
9	2621442	14000
10	10485762	7000
11	41943042	3500
..	..	..
14	2684354562	400

次に計算領域の形成方法について解説する。球面を図 3(a)に示されるように 10 個の矩形領域に分割する。この矩形領域を図 3(b)のように 4 分割し，さらに 4 分割することにより図 3(c)のような矩形領域を生成する。初期の状態(図 3(a))を R-Level 0 と称し，水平格子分割と同様に分割数ｎに応じて R-Level n と定義している。矩形領域がそれぞれ計算領域となり，通常は単一のコアが単一の計算領域を処理する。また各計算領域上の水平格子は 2 次元の配列ではなく 1 次元の配列として定義してループ長を得られるようなコーディングとなっている。R-Level に対応した計算領域の数を表 2 に示す。

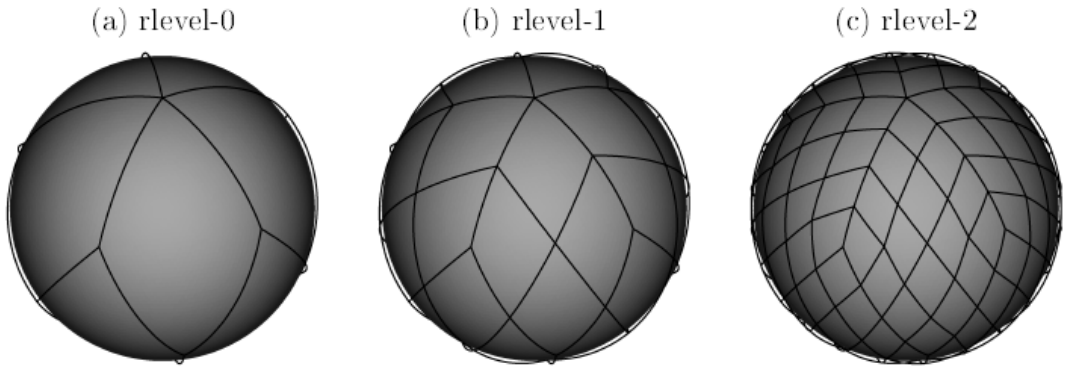


図３：計算領域分割の概念図

(a)は R-Level 0 を示し，計算領域は全球で 10 個存在する。(b)は R-Level 1 を示し，計算領域は全球で 40 個存在する。(c)は R-Level 2 を示し，計算領域は全球で 160 個存在する。

表３：R-Level と計算領域数

R-Level	計算領域数
0	10
1	40
2	160
3	640
4	2560

３．計測実行環境

今回の計測ではベクトル型計算機として(独)海洋研究開発機構/地球シミュレータセンターの地球シミュレータを用いた，スカラ型計算機は東京大学情報基盤センターの T2K オープンスーパーコンピュータ(東大)･HA8000 クラスタシステム(以下 T2K-東大)と筑波大学計算機科学研究センターの T2K-Tsukuba システム(T2K- Tsukuba)を利用した。

スカラ型計算機ではキャッシュメモリを有効に利用することによって計算効率を向上させる必要がある。そこで今回の計測では水平方向の解像度を G-Level7 で固定して，R-Level を 1 から 4 まで変化させることにより 1 コア当たりの問題規模を変化させた。利用可能な計算機資源を考慮した結果，１コア当たりの問題規模を上述のように決定した。各 R-Level に対応した利用コア数と水平方向の格子点数を表 4 に示す。ただし T2K- Tsukuba の場合はメモリ使用量の制約のため R-Level0 の計測は実施していない。

表４：G-Level 7 の時の R-Level に対応したコア数と水平格子点数

R-Level	0	1	2	3	4
コア数	10	40	160	640	2560
水平格子点数	16384	4096	1024	256	64

計算時間の計測には MPI の関数である MPI_WTIME を用いて，NICAM のソースコードの必要な部分に適宜挿入した。今回の計測では時間刻みを 30 秒として 121STEP(約 1 時間)の積分を実施して計測を実施した。ただし，T2K-Tsukuba にはステージング機能が存在しないためデータの入出力を除去して計測を実施する必要がある。NICAM の計算過程には毎 STEP 計算せず，数 STEP 度に計算を実施するプロセスがある。このプロセスを今回は 12STEP 度に計算するとしていたため初期条件の読み込みを含む 12STEP 分を 121STEP から除去することによって計測結果を評価した。

また T2K-東大では通信部分を取り除いたスカラ型計算機の単体性能を評価し，T2K-Tsukuba では通信部分を含んだ性能の評価を実施した。

４．計測結果

はじめに通信を除去した1step 分の計算時間のR-Levelへの依存性を地球シミュレータ と T2K-東大 についての計測結果を表５ に示す。表５は横方向に地球シミュレータのR-Levelを示し，縦方向にT2K-東大のR-Levelを示す。地球シミュレータがR-Level 0の時にT2K-東大がR-Level 0の時の計算時間は8.58倍を示し，T2K-東大がR-Level 4の時には0.06倍の計算時間を要したことを示している。同表より，地球シミュレータのR-LevelよりT2K-東大のR-Levelが2以上大きい時，つまり1プロセス当たりに割り当てられる問題のサイズが16分の1になった場合には1step当たりの演算時間は地球シミュレータよりもT2K-東大の方が短いということが示された。

表５：地球シミュレータとT2K-東大の1STEP当たりの計算時間の比率

	地球シミュレータ					
T2K-東大	R-level	0	1	2	3	4
	0	8.58	30.41	109.3	286.12	648.51
	1	2.97	10.53	37.85	99.08	224.57
	2	0.49	1.73	6.22	16.29	36.91
	3	0.20	0.71	2.54	6.64	15.06
	4	0.06	0.22	0.78	2.05	4.64

次にプロセス間の通信を考慮したT2K-Tsukubaの計測結果を示す。利用するコア数が40個の時の計算時間を基準としてR-Levelが上昇した時の計算時間の減少率を図4に示した。ここで R-Levelが上昇するごとに水平格子点の数は4分の1に減少するので理論的には計算時間の減少率は4分の1になることが期待される。図4より地球シミュレータは利用するコア数が640個を超えると演算効率が極端に悪化する様子がわかる。これベクトル型計算機である地球シミュレータのベクトル長(256)を有効に利用できていないためと考えられる。各R-Levelにおける平均ベクトル長を示す。

表６：G-Level 7の時のR-Levelに対応した平均ベクトル長

R-Level	0	1	2	3	4
平均ベクトル長	247.66	236.52	217.98	147.74	89.55

一方でT2K-Tsukubaの場合は利用するコアが2560個から演算効率が悪化する様子が見て取れるが，この原因の一つとして通信の影響が挙げられる。

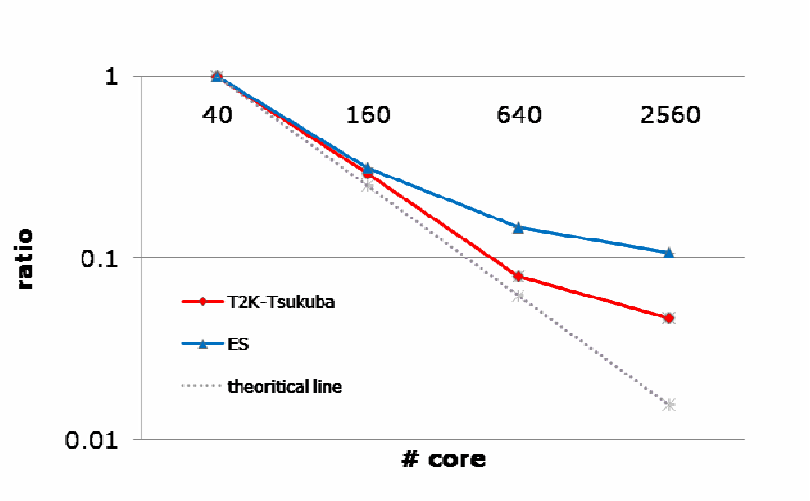


図４：計算過程全体のスケーラビリティ

縦軸はコア数40を基準とした場合の計算時間の減少率を示し(対数目盛)，横軸は利用したコア数を示す。赤線はT2K-Tsukubaのスケーラビリティを示し，青線は地球シミュレータ（ES）のスケーラビリティを示す。点線は理論値を示す。

NICAMの計算過程の中には通信を含まない過程が存在するので，力学過程(通信を含む)・放射過程・雲物理過程・乱流過程を抜き出して地球シミュレータとT2K-Tsukubaのスケーラビリティをそれぞれ図5と図6に示す。地球シミュレータの場合は利用するコア数が640個を超えると計算効率が悪化する様子が示されており計算過程全体の場合と同様の傾向が得られた。また T2K-Tsukubaの場合も利用コア数が2560個を超えると計算効率が悪化しており，計算過程全体のスケーラビリティと同様の傾向を示しているが，プロセス間の通信を含む力学過程より通信を含まない雲物理過程や乱流過程の計算効率が悪化している様子がわかる。一方で放射過程は利用するコア数が640個の時にわずかながら理論値より計算効率が向上していることがわかり，キャッシュメモリの有効利用がおこなわれていることが示唆される。しかし放射過程の計算効率も2560個のコアを使用した場合には効率が悪化していることがわかる。

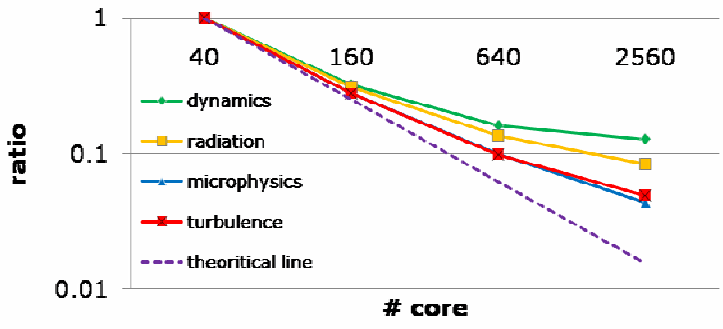


図５：地球シミュレータにおける各計算過程のスケーラビリティ

縦軸はコア数40を基準とした場合の計算時間の減少率を示し(対数目盛)，横軸は利用したコア数を示す。緑線は力学過程のスケーラビリティを示し，黄線は放射過程のスケーラビリティを示し，青線は雲物理過程のスケーラビリティを示し，赤線は乱流過程のスケーラビリティを示す。点線は理論値を示す。

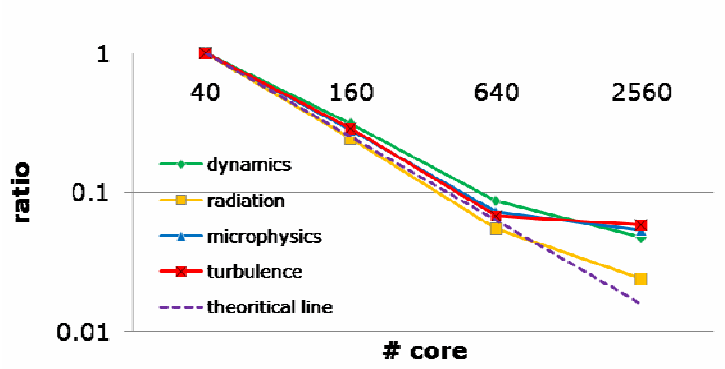


図６：T2K-Tsukubaにおける各計算過程のスケーラビリティ

縦軸はコア数40を基準とした場合の計算時間の減少率を示し(対数目盛)，横軸は利用したコア数を示す。緑線は力学過程のスケーラビリティを示し，黄線は放射過程のスケーラビリティを示し，青線は雲物理過程のスケーラビリティを示し，赤線は乱流過程のスケーラビリティを示す。点線は理論値を示す。

利用コアが2560個で計算効率が悪化する通信以外の原因のとしてハロリージョンの取り扱いが挙げられる。ハロリージョンとは計算格子の最外側に存在し(図7の赤丸)隣接する計算領域とのプロセス間通信によって収得されるが，通信の回数を減らすためにハロリージョン自体でも計算がおこなわれている。計算領域当りの格子点数が多い場合にはハロリージョンの量は無視することができるが，格子点数が小さくなるとハロリージョンが無視できなくなる。

計算領域当りの水平格子点数とハロリージョンの数には表7に示すような関係がある。表7に示すようにR-Levelが上がるごとに計算格子自体は4分の1に減少するのに対してハロリージョンの数は2分の1に減少する。このためG-Level7の場合にはR-Level4の時に計算格子に対するハロリージョンの比が2:1となり，ハロリージョンの数が無視できなくなるため計算効率が悪化すると考えられる。

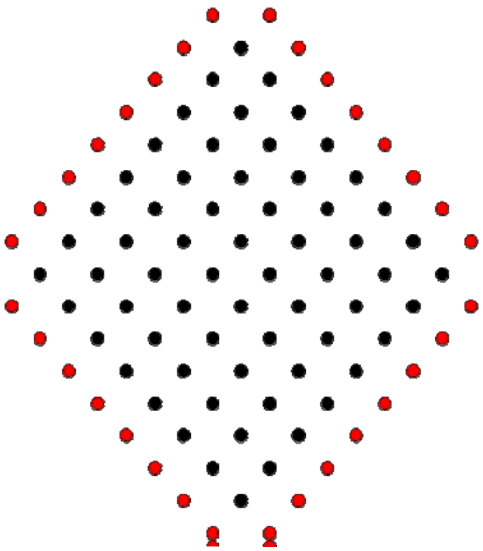


図７：計算領域上の構成の概念図

赤丸・黒丸が計算おこな う格子を示す。ただし、赤丸は隣接する計算領域との通信をおこなう格子(ハロリージョン)を示す。

表７：R-level と対応するハロリージョンの数

Rlevel	1	2	3	4
格子点数	4096	1024	256	64
ハロリージョン	258	130	66	34
計算格子	4354	1154	322	98

前述したとおりハロリージョンの取り扱いの他に計算効率を悪化させる原因として，1 プロセス当たりの問題規模の縮小により通信量は減少するにも関わらず，通信の回数は変わらないので通信のオーバーヘッドの比率が増加することが挙げられる。図８には計算時間に対するプロセス間の通信に要する時間の増加率を利用したコア数が 40(R-Level=1)の時を基準としてコア数への依存性を示している。図８より地球シミュレータの場合には平均ベクトル長の減少のために，演算量に対する通信に要する時間の比率の増加が隠されてしまうことが考えられが，T2K-Tsukuba と地球シミュレータのいづれの場合でも計算時間に対する通信時間の割合は増加していることがわかる。

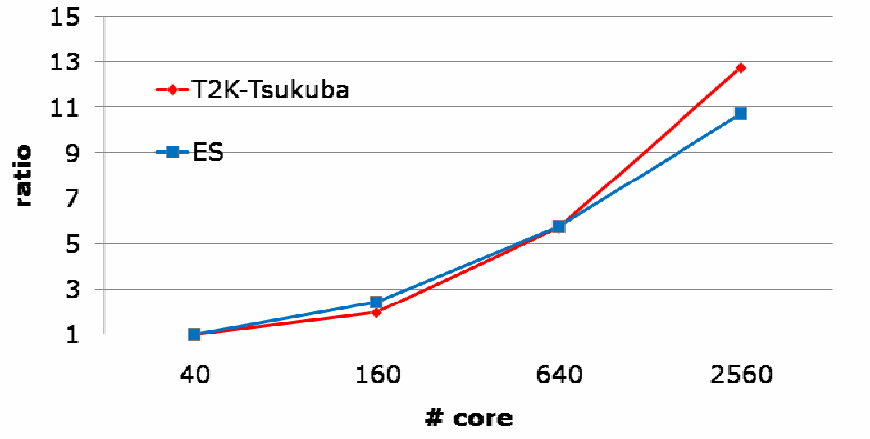


図８：計算時間に対する通信時間の上昇率

縦軸はコア数40を基準とした場合の計算時間の増加率を示し(対数目盛)，横軸は利用したコア数を示す。赤線はT2K-Tsukubaの結果を示し，青線は地球シミュレータ（ES）の結果を示す。

５．まとめと今後の展望

本研究では既存の大型スカラ型計算機である T2K-東大及び T2K-Tsukuba を京速計算機と見做して NICAM の計算時間を計測し，ベクトル型計算機である地球シミュレータでの計算性能との比較を行った。

今回の計測では単体性能だけを比較した場合，利用するコア数が同数でループ長がベクトル計算機のベクトル長に適切な場合には，スカラ型計算機の 1step 当たりの計算時間ははベクトル型計算機の 6 倍から 10 倍程度となる。しかしスカラ型計算機で利用するコアの数がベクトル型計算機の 16 以上になるとスカラ型計算機の計算時間がベクトル型計算機よりも短くなることがわかった。

またプロセス間の通信を含んだ計速結果を比較すると，スカラ型計算機では 1 プロセス当たりの水平方向の格子数が 256 個でキャッシュ利用による計算効率の向上が見られ，コア数が

2560 個で演算性能が悪化することがわかった。 コア数が 2560 個で計算効率が悪化する原因として，ハロリージョンが計算領域の格子数に占める割合が大きくなり，ハロリージョンの計算量が 1 プロセス当たりの計算量に対して無視できなくなっているということが挙げられることと小規模なプロセス間の増大による通信のオーバーヘッドの増加が原因の一部として挙げられる。

京速計算機では G-Level 14 の水平格子間隔 400m を用いた超高解像度実験が可能となる(佐藤他，2008)。そこで理論性能が 10 ペタフロップス計算機の京速計算機で半分のノードを使用したと仮定して NICAM がそのピーク性能比率の 10%をだせた場合，計算時間が 4 時間で一日分の積分が実行可能だと見積もることができる。

図 8 に現状の NICAM の地球シミュレータと T2K-Tsukuba の R-level に対応したピーク性能比率を示す。ベクトル型計算機（地球シミュレータ）では平均ベクトル長を稼ぐことができる R-Level の低い状態では 35%程度となっているが,スカラ型計算機(T2K-Tsukuba)では今回計測を実施した水平格子解像度の範囲ではピーク性能比率が 5%未満の値となっている。部分的にキャッシュメモリ有効利用しだしたコア数が 640 個の場合では 2.5%程度となっており，今回見積もったピーク性能比率 10%の 4 分の 1 程度となっている。しかし 1 計算領域当りの水平格子点数は 64 よりも少なくしてしまうとハロリージョンとプロセス間の通信の制約により計算効率を悪化させてしまう結果となることが分かっている。

スカラ型計算機では CPU の性能に依存するがベクトル型計算機よりも多くのコア数を使用して計算効率を上げる必要がある。この条件はキャッシュメモリを有効利用することとは整合性が取れているが，NICAM 固有のハロリージョンの制約とは逆のセンスとなっている。今後はハロリージョンの取り扱いとキャッシュメモリの有効利用を目指した最適化が必要といえる。

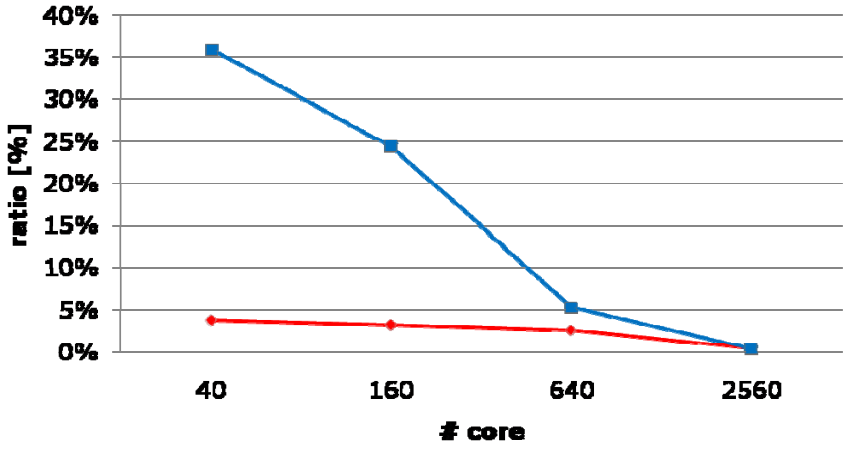


図８：計算時間に対する通信時間の上昇率

縦軸はピーク性の比率を示し，横軸は利用したコア数を示す。赤線は T2K-Tsukuba の結果を示し，青線は地球シミュレータの結果を示す。

６．謝辞

本研究の一部は独立行政法人科学技術振興機構(JST)の“戦略的推進事業（CREST)”の御支援を承った。 計速には海洋研究開発機構/地球シミュレータセンターの地球シミュレータと「HPC 特別プロジェクト」及び 「T2K オープンスパコン（東大）共同研究」のご支援の下，東京大学情報基盤センターの T2K オープンスパコン(東大)プロジェクトコード：H94 と筑波大学計算科

学研究センター学際共同利用プログラムのご支援の下，筑波大学情報科学センターの T2K-Tsukuba を利用させていただいた。 この場をお借りして感謝の意を表す。

参 考 文 献

佐藤正樹，富田浩文，準一様格子を用いた全球雲解像大気モデルの開発とそれによる熱帯対流雲集団のシミュレーション—2007年度日本気象学会賞受賞記念講演—，***天気***，55:451-456，2008

Satoh, M. Conservative Scheme for the Compressible Nonhydrostatic Models with the Horizontally Explicit and Vertically Implicit Time Integration Scheme. *Mon. Wea. Rev.*, 130:1227-1245, 2002

Satoh, M., Conservative Scheme for the Compressible Nonhydrostatic Models with Moist Process, *Mon. Wea. Rev.*, 131:1033-1049,2003

Satoh, M., T. Matsuno, H. Tomita, H.Miura, T.Nasuno, S.Iga, Nonhydrostatic icosahedral atmospheric model (NICAM) for global cloud resolving simulations, ***J. Comput. Phys.***, 227:3486-3514, 2008

Miura, H., M.Satoh, T.Nasuno, A.T.Noda, and K.Oouchi, A Madden-Julian Oscillation event realistically simulated by a global cloud-resolving model, ***Science,*** 318, 1763-1765

Tomita, H., M.Tsugawa, M.Satoh, and K.Goto. Shallow Water Model on a Modified Icosahedoral Geodesic Grid by Using Spring Dynamics, *J. Comput. Phys.*, 174:579-613, 2001

Tomita, H., M.Tsugawa, M.Satoh, and K.Goto, An optimization of the icosahedral grid modified by spring dynamics, *J. Comput. Phys.*, 174:307-331, 2002

Tomita, H. and Satoh. M, A new dynamical framework of nonhydrostatic global model using the icosahedral grid, ***Fluid Dyn. Res***. 34:356, 2004

Tomita, H., K.Goto,and M.satoh. A NEW APPROACH OF ATMOSPHERIC GENERAL CIRCULATION MODEL GLOBAL CLOUD RESOLVING MODEL NICAM AND ITS COMPUTATIONAL PERFORMANCE- *SIAM J. Sci. Comput.*, 30, 2755-2776; DOI. 10.1137/070692273.4

海洋循環形成プロセスの高解像度シミュレーション

羽角博康，草原和弥，松村義正

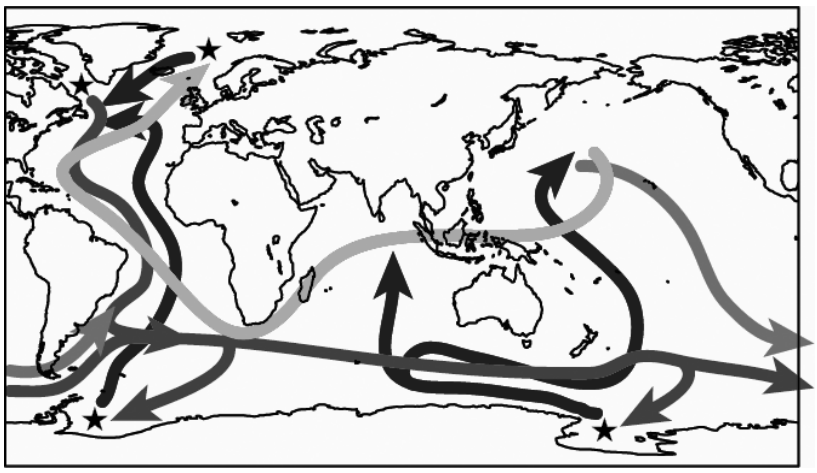
東京大学気候システム研究センター

１．はじめに

海に存在する流れは我々の生活環境に大きな影響を及ぼしている．日本の南岸の表層海洋に存在する黒潮を例に取ると，これは全世界を代表する強い海流のひとつであるが，低緯度から運ぶ熱が日本の気候や気象に多大な影響を及ぼすとともに，海洋生物の棲息環境や輸送の面から日本近海の水産資源にとっても極めて重要である．この黒潮は主に風によって駆動されるが，日本付近の風は直接的に重要ではなく，北太平洋全体の海上風分布が決める北太平洋全域の流れとの関係の中で形作られている．したがって，黒潮の性質や変動を解き明かそうとするならば，日本付近の流れとしてのみならず，大規模な海洋循環という視点から黒潮を捉える必要がある．また，沿岸付近には黒潮のような強い海流以外にも様々な流れが存在し，それらは時として黒潮以上に我々の生活と密接に関わる．そうした小規模な流れは，付近に存在する黒潮のような強い海流の影響を大きく受け，それを通して海洋大循環ともリンクしている．特に全地球的な気候変動のもとでは，沿岸の小規模な流れやその変動を考える場合においても，海洋大循環の変動との関わりを無視するわけにはいかない．

一方，海洋の深層にも流れは存在する．それは我々の生活環境からは非常に遠くに位置し，しかもその流れの速さは黒潮に比べれば何十分の一に過ぎないのであるが，実は根本的なところで我々の生活環境を大きく左右している．深層海洋の流れは深層だけに閉じておらず，表層海洋の流れとリンクしている．海水の密度は温度と塩分に依存し，海面付近で冷却されて低温化(もしくは蒸発等により高塩分化)した海水は，高密度のために深層へ沈む傾向が強くなる．現在の海洋の状態において，深さ数千 m の深海に存在する水はいたるところで 0℃近くの低温であり，これは高緯度の海面付近にある低温水が沈降して深層を占めていることを示す．第 1 図はそうした海洋の深層と表層をつなぐ循環の概略を模式的に示したものであるが，その大きな特徴は，海面付近から深層海洋への沈降が北大西洋高緯度と南極周囲の極めて限られた場所でのみ生じていることである．深層水形成と呼ばれるこの沈降過程は水平 1 km 程度のスケールで生じる対流現象に端を発している．その意味では，極めて限られた領域における微小規模の現象が全海洋規模の循環をコントロールしている．そしてこの循環は，地球上の熱を大規模に再配分する．その働きがあればこそ，例えばヨーロッパ北部は 70 度を越えるような高緯度にも関わらず人が居住できる環境にある．氷期等の過去に生じた大規模な気候変動はこの海洋大循環の変動と大きな関わりがあることが知られており，今後起こり得る気候変動の中でも，特に大規模かつ長期に及ぶものに関しては，この循環の振舞がひとつの焦点になる．その振舞を知るということは，すなわち，水平 1 km 程度のスケールにおける沈降現象と水平 10 万 km スケールに及ぶ海洋大循環がどのように相互に関わっているかを解き明かすということである．

我々は今，地球温暖化という全地球規模の長期気候変動に直面している．その中で海洋がいかに変化し気候変動をどのようにコントロールするか，またその結果が沿岸海況や水産資源への影響を通してどのようなインパクトを人類社会に及ぼすのかを知ることは喫緊の課題である．



第 1 図：海洋大循環の概略図.

ただし，黒潮のような各大洋内での水平的な循環は無視しており，大洋間のつながりに特化している．矢印は色の淡い順からそれぞれ，表層(海面付近)，中層(深さ 1000 m 程度)，深層(深さ 3000 m 程度)，底層(深さ 5000 m 程度)の流れを，星印は主な海水の沈降(深層水形成)領域を示す.

そのためには，上述のように幅広いスケールに渡る現象の相互作用として存在する海洋大循環を理解し，その物理プロセスを適切に表現した数値モデルによる海洋大循環シミュレーションが必要とされる．本稿においては，そのようなシミュレーションとスーパーコンピューティングの関わりについて概観した後，T2K オープンスパコン HPC 特別プロジェクトにおいて実施したシミュレーションによって得られた成果について紹介する.

２．海洋大循環シミュレーションとスーパーコンピューティング

海洋の循環を記述する方程式系はナビエーストークス方程式(線型粘性流体の運動方程式)と熱・溶存物質(塩分)の輸送・拡散方程式を基礎としている．しかし，当然のことながら分子粘性・拡散を直接表現する解像度を海洋という巨大系に対して適用できるわけはない．後述するように，現状で数値海洋モデルが解像可能な水平スケールは，海洋大循環全体を扱う場合ならば 10 km 程度，深層水形成のような特定の領域における重要プロセスのみを扱う場合でも 100 m 程度が限界である．特に前者の場合には，解像されない現象は乱流的と呼ぶには程遠く，その解像されないスケールの現象の物理的性質を解き明かした上で，解像されるスケールに与える影響をパラメータ化して表現する必要がある.

全海洋規模の循環を扱う数値モデルは海洋大循環モデルと呼ばれ，方程式系を構造格子上で差分化して解くのが一般的である(具体的な方程式系や差分化手法に関しては Hasumi (2006)などを参考にされたい)．海洋大循環の数十年以上に渡る変動を扱う場合，例えば地球シミュレータなどの現状で最高性能の部類に属するコンピュータを用いるならば，10 km という水平格子サイズは実現可能である．しかし，1 km となると現状では不可能であり，現在開発が進められている次世代スーパーコンピュータであっても非常に困難である．一方，深層水形成のような海洋大循環をコントロールする重要プロセスの一部や沿岸海況の適切な表現のためには，さらに 1 桁小さい水平格子サイズが必要とされ，それを全海洋に一様に適用することは次世代スーパーコンピュータでも実現不可能である．ただし，いま考えているような目的においては，特に高解像度が必要とされるのは沿岸や深層水形成領域など，固定された限定的な領域のみであ

る．それ以外の場所についても，あらゆる場所で 1 km という水平格子サイズが必要とされるわけではなく，数十 km という格子サイズですら足りる領域も存在する．ネスティング手法などを用いて必要な場所を重点的に高解像度化する連結階層化を行えば，例えば次世代スーパーコンピュータの許容範囲内で目的を達することは可能であると考えられる．ただしその場合には，特にサブメソスケールと呼ばれる水平 1 km スケールの現象に関して，それを陽に解像する特定の領域以外では適切なパラメータ化が必要とされる．いま目的とする多スケール間の相互作用を扱う連結階層シミュレーションを実現するためには，陽に表現すべき現象(対象領域と解像度)を特定するためのプロセス研究,そして解像しない現象のパラメータ化に関する研究を行うことがまず必要とされる．我々は今まさにそのような研究に取り組んでおり，本プロジェクトにおいてはその一環として，南極周囲の海氷生成に伴う高密度水の形成を取り扱ったシミュレーションを実施した．その結果を§３で紹介する．

一方，特定の小規模プロセスの表現のために特別な高解像度を適用する場合，そこに対しては大規模現象を扱う海洋大循環モデルとは若干異なる定式化に基づく数値モデルを使用する必要が生じる．大規模スケールを対象にする場合には，運動方程式の鉛直方向成分に対して静水圧近似を適用する．これは，大規模スケールでは海洋が基本的に安定成層(上方ほど低密度)であるため，鉛直方向の運動が抑制されるという事実に基づき，ある点での圧力をそれより上方に存在する海水の重さだけで決めるという近似である．例えば高密度化した海水の沈降を扱う場合にはこの近似は不適切である．静水圧近似を適用しない場合，海水を圧縮性流体として定式化すると，音波が表現されることになる．海洋中の音波の代表的速度は 1,500 $\mathrm{m\,s^{-1}}$ であり，本来のシミュレーション対象である流れが高々 1 $\mathrm{m\,s^{-1}}$ 程度であることに鑑みると，音波を陽に表現することは甚だ非効率的である．そのため，静水圧近似を適用しない数値海洋モデルでは海水を非圧縮性流体として定式化するのが通常であるが，この場合には 3 次元ポワソン方程式を解く必要があり，モデルのパフォーマンスはその計算効率に大きく依存する．海洋シミュレーションの場合には，海底地形という複雑な境界条件への対応や，対象とする領域および格子の縦横比が非常に大きいことなどについて，特別な配慮が必要となる．また，今後のスーパーコンピューティングの方向を念頭に置いた場合には，大規模並列において高スケーラビリティーを保証するようなアルゴリズム開発が必須である.我々はこの問題に対する解決策として，マルチグリッド法を前処理とする共役勾配法に基づく具体的手法を開発しており(Matsumura and Hasumi, 2008)，本プロジェクトにおいては先述の南極周囲における高密度水流出過程に関するシミュレーションに適用した．その結果を§４で紹介する．

３．東南極における海氷と高密度水形成のシミュレーション

（１）はじめに

冬季の海氷域にみられる周囲を海岸線と厚い海氷で覆われた開水域ないし薄氷域は沿岸ポリニヤと呼ばれ，沿岸から数～100 kmの広がりをもって存在する．沿岸ポリニヤにおいては比較的暖かい海水面が寒冷な空気に晒されるため，大量の熱が海から奪われる．これにより，沿岸ポリニヤでは高い海氷生成がみられ，それと同時に起こるブライン[1]排出のために，その直下の海水は高密度化する．こうして大陸棚上に形成された高塩分水は高密度陸棚水と呼ばれる．南

[1] 海水が凍結して海氷になる際，もとの海水に含まれていた塩分の大部分は海氷にとりこまれない．その結果として排出される高塩分海水をブラインと呼ぶ.

極大陸周囲には人工衛星観測によって多くの沿岸ポリニヤの存在が確認されている[2]. 沿岸ポリニヤで生成される高密度陸棚水は南極底層水[3]の形成に重要な役割を果たし, それを通して海洋大循環とも深く結びついている.

地理的・気候的制約のため, 南極大陸周囲に存在する沿岸ポリニヤの直接観測は, 時間的にも空間的にも十分でない. そうした限られた海洋観測からではあるが, 沿岸ポリニヤにおける高密度陸棚水の変動は南大洋の中層から底層の特性に大きく影響を及ぼすことが示唆されている. 沿岸ポリニヤにおける様々なプロセスを陽に表現した数値シミュレーションは, 南大洋の, そして全海洋の変動を理解する上で必要とされる. 過去に行われてきた低解像度の海洋大循環モデルによるシミュレーションでは, 沿岸ポリニヤが十分に表現できず, その海洋に対する影響を評価することができなかった. Marsland et al. (2004)は南極低層水の主要な起源領域のひとつと考えられるメルツ氷河ポリニヤに着目し, その周囲のみを特に高解像度で表現した海洋大循環シミュレーションを通して高密度陸棚水形成を論じた.

Tamura et al. (2008)は人工衛星データに基づく熱収支解析により, 南極周囲の沿岸ポリニヤにおける海氷生成量の空間分布を初めて明らかにした. その結果からは, メルツ氷河ポリニヤだけでなく, 多くの沿岸ポリニヤにおいて高密度陸棚水が生成されていることが示唆される. 本研究の目的は, 沿岸ポリニヤにおける海氷生成量を現実的にシミュレートし, 高密度陸棚水形成を通した海洋への影響を調べることである. ここでは特に, 数多くの沿岸ポリニヤが存在する東南極[4]に着目する. その中で, 高解像度シミュレーションで沿岸ポリニヤをいかに現実的に再現するか, および沿岸ポリニヤに伴ってどこでどれだけの高密度陸棚水が生成されているかを論ずる.

（２）数値モデルの設定

シミュレーションに使用するのは, 海洋大循環モデルCOCOである. 海氷モデルにおいては, 2カテゴリー厚さ表現[5], 0層熱力学, 弾粘塑性レオロジーを採用する. 海氷の塩分は5 psu[6]で固定する. 海洋モデルでは鉛直方向に31層が存在し, 格子間隔は最上層の5 mから最深層の500 mまで変化する. シミュレーション領域は全海洋である. 着目する領域を特別に高解像度で表現するために, 水平方向の曲線直交座標系[7]における 2 個の特異点を東南極上(東経 70 度・南緯70.5度, および東経143度・南緯68度)に置く. 東南極の沿岸における水平格子サイズは3～15 kmである. モデルの地形は標準的な地形データセットであるGEBCOに基づくが, アデリー海嶺周囲については最近の航行データを用いて, 海岸線については人工衛星観測データを用いて, それぞれ修正する.

---

[2] 南極大陸周囲では, 夏季にはほとんど海氷が存在しないのに対し, 冬季の最盛期には南極大陸沿岸から南緯55～60度程度にかけた広い範囲が海氷に覆われる.

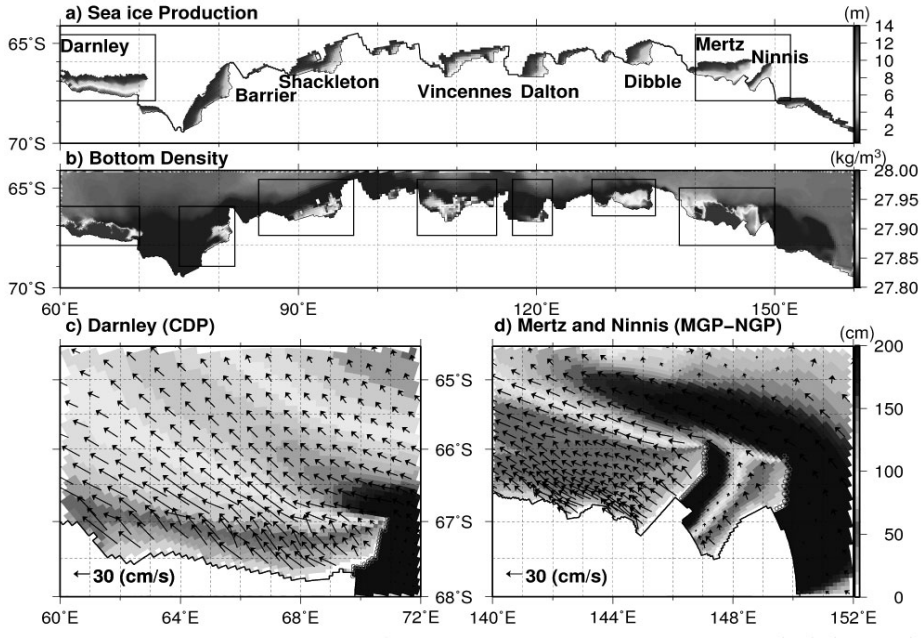
[3] 第1図における底層流が運び, 全海洋の最も深い部分を占める高密度水の名称.

[4] 南極大陸のうち, 経度が東経で表される領域. メルツ氷河ポリニヤもこの領域に存在する.

[5] 1格子内の海氷の存在量について, 格子の面積のうち海氷に覆われる部分の割合(密接度)と, 海氷に覆われている部分における平均的な厚さを変数として予報する方法のこと.

[6] 観測で実用的に用いられる塩分濃度の単位で, 通常は千分率と同一視してよい.

[7] 本モデルは水平2次元について一般曲線直交座標上で定式化されている. 地球上の水平座標系としては経度－緯度座標が最も自然であるが, 特異点を避ける目的や着目領域の解像度を選択的に高める目的等のため, 経度－緯度座標系を等角写像によって変換するなどして得られる様々な曲線直交座標系が適用される.



第 2 図:東南極の沿岸ポリニヤのシミュレーション結果.

a) 1 年間の海氷生成量(m yr^{-1}), b) 8 月の海底における海水のポテンシャル密度, c) 1998 年 8 月のダーンレー岬ポリニヤ周辺における海氷厚と海氷流速, d) 1998 年 8 月のメルツ氷河ポリニヤとニニス氷河ポリニヤ周辺における海氷厚と海氷流速.

モデルに与えられる海面境界条件のうち，塩分に関するものについては，淡水フラックスを与えるのではなく月平均気候値に時定数 10 日で緩和する．熱フラックスについては，放射フラックスデータに基づく下向き短波・長波放射，および海上気象要素データからバルク式を用いて計算される顕熱・潜熱を与える．北半球においては，全領域・深さに渡って，水温と塩分を月平均気候値に緩和する．気候値水温・塩分を初期状態として，海面境界条件を日毎気候値データによって与えて 20 年のスピンアップ計算を行った後，1990～2000 年の海面境界条件を適用したシミュレーションを行う．以下では 1991～2000 年の結果を解析する．

沿岸ポリニヤは海岸線や海上に流出した氷河[8]の近傍のみでなく，座礁した氷山の風下ないし下流にも形成されることが知られている．こうした座礁氷山による海氷堰き止め効果を表現するため，モデルにおいては座礁氷山が存在する地点をあらかじめ決めておき，その点における海氷流速を 0 にする．ここでは海氷は動かないが，その下に存在する海水は流れることができる．この効果をダーンレー岬ポリニヤ，メルツ氷河ポリニヤ，ニニス氷河ポリニヤについて考慮する(それぞれの位置については第 2 図を参照)．この海氷堰き止め効果を表現した実験をコントロール実験とし，CNTLと記述する．一方，この効果が海氷や海洋の場にどれだけの意味を持っているのかを調べるため，2 種類の実験を追加的に行う．実験NO-GIBにおいては海氷堰き

[8] 南極大陸上には大陸氷床と呼ばれる氷の塊が存在する．これは積雪が圧縮されて形成されるものであるが,長い時間をかけてゆっくりと流動する.この流動する氷塊を指して氷河と呼ぶ.南極においては氷河が海まで達する．大陸上の氷河からつながった状態で海上に浮いて存在するものは，氷河ないし棚氷と呼ばれる．一方，ある程度以上海上に突出した氷河や棚氷の先端は切り離されて海を漂流する．この状態になったものは氷山と呼ばれる．氷山も海氷も海を漂う氷塊という点では共通するが，前者は真水であって初期には 100 m以上の厚さを持つのに対し，後者は若干の塩分を含み，厚さが数mを超えることは稀である．

Polynya name	Ice Production	(km^3)	Total Volume of DSW	(km^3)	Formation Rate of DSW	(Sv)
Darnley	197.7 ±	24.13	5783 ±	704	0.72 ±	0.34
Barrier	46.5 ±	3.35	2709 ±	1062	0.24 ±	0.18
Shackeleton	81.1 ±	11.84	6077 ±	3768	0.39 ±	0.42
Vincennes	76.7 ±	14.69	7177 ±	1553	0.47 ±	0.37
Dalton	44.0 ±	8.21	595 ±	285	0.03 ±	0.01
Dibble	55.2 ±	8.61	5672 ±	1535	0.37 ±	0.34
Mertz	103.3 ±	9.37	9510 ±	1784	0.84 ±	0.16
Ninnis	30.7 ±	3.54				

第 1 表：各沿岸ポリニヤにおける海氷生成量・高密度陸棚水存在量・高密度陸棚水生成率.
ポリニヤの名称については第 2 図を参照.

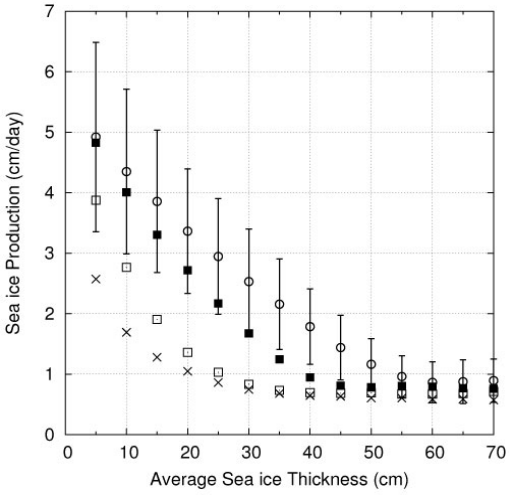
止め効果を導入せず，実験LANDにおいては座礁氷山を陸として扱って海水の流れも止める.

（3）シミュレーション結果

第 2 図に示されるシミュレーション結果によると，東南極の海岸線に沿っては，海氷生成が大きいバッチ状の領域が多数存在する．人工衛星観測にみられるのと同様に，突出した地形の主に西側(風下ないし下流側に相当)に沿岸ポリニヤが存在する．それぞれの沿岸ポリニヤにおける海氷生成量(第 1 表)は，人工衛星観測に基づく見積もりと概ね一致する．シミュレーション結果において最も高い海氷生成を示すのはダーンレー岬ポリニヤであり，第 2 番目はメルツ氷河ポリニヤである．このことも人工衛星観測に基づく見積もりと整合する.

第 2 図には 8 月における海底の海水ポテンシャル密度[9](σθ)分布も示されている. 沿岸ポリニヤ直下のポテンシャル密度は 27.88 以上に達し，これは観測の分類による高密度陸棚水の範囲に含まれるものである(低塩分陸棚水：$27.88 < \sigma_\theta < 27.91$；高塩分陸棚水：$27.91 < \sigma_\theta$)．このような海水は深海に到達するに十分な高密度を持っており，流出過程で周囲の海水と混合することで南極底層水を形成する. 高密度水生成時期には高密度陸棚水形成領域(第 2 図の四角で示されるコントロール体積)の内と外の間で高密度陸棚水の交換が存在しないものと仮定した場合に，それぞれの沿岸ポリニヤにおける高密度陸棚水の存在量と生成率を第 1 表に示す．高密度陸棚水が存在するのは，秋から冬にかけての生成時期のみである．海氷生成量の大きさに対応して，ダーンレー岬ポリニヤ，メルツ氷河ポリニヤ，ニニス氷河ポリニヤでは大きな存在量と生成率がみられる．メルツ氷河ポリニヤにおける高密度陸棚水生成の過去の見積もりと比較して，この結果は妥当な範囲に収まっている. ダーンレー岬ポリニヤ，メルツ氷河ポリニヤ，

[9] 海水には若干ではあるが圧縮性があり，同じ温度・塩分を持つ海水でも深海に行くほど高密度となる．それと同時に，圧縮(膨張)には温度上昇(低下)が伴う．このため，異なる深さに存在する海水の間で単純に密度や水温の高低を比較しても，あまり有用な情報をもたらさない．ある深さに存在する海水について，それを断熱的に海面まで移動させた場合に実現される水温(断熱膨張のために低下する)のことをポテンシャル温度と呼ぶ．また，海水の密度は水温・塩分・圧力(水深)の函数であるが，水温の代わりにポテンシャル水温を用い，圧力を海面として得られる密度をポテンシャル密度と呼ぶ．こうして海面に基準を置くことで，異なる深さにおける海水の間で密度や水温の大小を比較することに意味を持たせることができる．ポテンシャル密度は通常σθという記号で表され，その値は$kg\ m^{-3}$の単位で表された密度の値から 1000 を引いたものである．よほど特殊な状況(大河川の河口や外洋との交換が著しく少ない湾など)でなければ，22～28 の範囲の値をとる.



第 3 図:平均海氷厚と海氷生成量の関係.

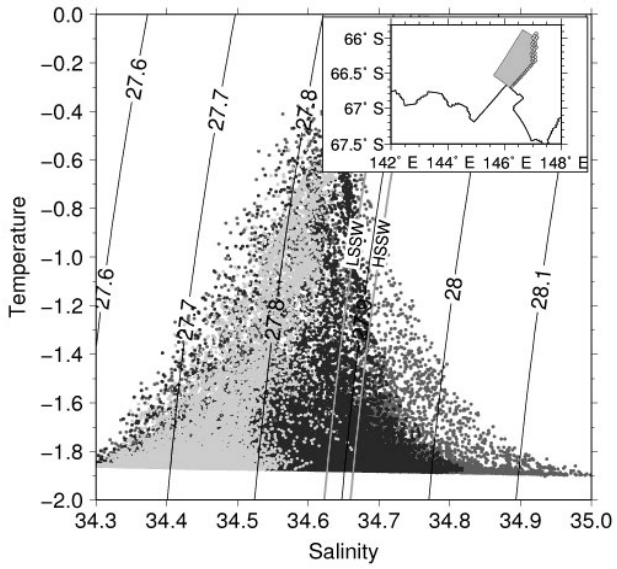
×, □, ■, ○はそれぞれ薄氷の閾値が 1, 10, 30, 50 cm の場合に対応. 50 cm の場合についてのみ, データのばらつきの標準偏差をエラーバーで表示.

ニニス氷河ポリニヤ以外の数多くの沿岸ポリニヤにおいても, 決して少なくない量の高密度陸棚水が生成されていることもわかる. なお, 座礁氷山の西側に形成されるポリニヤ以外における高密度陸棚水の存在量や生成率には大きな経年変動が存在し,沿岸ポリニヤが外力(海面熱フラックスや風)に大きく影響されることが示唆される.

第 2 図にはダーンレー岬ポリニヤ, メルツ氷河ポリニヤ, ニニス氷河ポリニヤ周囲における海氷の厚さと流速も示されている. 前述の通り, これらの領域においては座礁氷山による海氷堰き止め効果が考慮されている. 座礁氷山や海岸線の東側には, 厚さ 2 m を越える海氷が停滞している. 座礁氷山列の北側からは厚い海氷が舌状に伸びて存在している. 座礁氷山や海岸線の西側では, 50 cm よりも薄い海氷が発散的な流速場とともに存在している. メルツ氷河ポリニヤにおけるシミュレーション結果にみられる特徴は, 観測や過去のシミュレーション結果と整合的なものである.

2 カテゴリー厚さ表現の枠組みのもとでは, 開水域ないし薄氷域と厚氷域が 1 格子内に共存するのだが, 厚氷と薄氷を分ける閾値をモデルパラメータとして指定する必要がある. 第 2 図および第 1 表に示した結果は, この閾値を 50 cm とした場合である. この閾値の選択はモデルにおいて薄氷をどのように表現するのかと密接に関係するものだが, 過去の低解像度シミュレーションにおいては経験的に決められており, 今回のような比較的高解像度のシミュレーションにおいてどのような値を適用すべきかは明らかでない.

海氷密接度は薄氷の閾値に強く依存するため, 海氷生成量のシミュレーションはこの閾値に大きく依存するものと考えられる. 実際, この閾値を 1, 10, 30, 50 cmと変化させた場合の海氷生成量の違いを調べてみた. 第 3 図に示すのは, それぞれの閾値のもとでの 7 月(海氷生成が最も盛んな時期)における海氷生成と海氷厚の関係である. 閾値の値が大きいほど, 薄氷域における大きな海氷生成が得られる. どのケースにおいても, 海氷厚が 50 cmを越える場合には海氷生成は非常に小さい. 現場海洋観測によると, メルツ氷河ポリニヤでは最盛期に 5.8 cm day^{-1} で海氷が生成されている. これと比較すると, 沿岸ポリニヤで現実的な海氷生成をシミュレー



第 4 図:メルツ氷河ポリニヤの座礁氷山西側における 9 月の水温－塩分ダイヤグラム.

最も薄い色が NO-GIB, 中間の濃さが LAND, 最も濃い色が CNTL.背景にある等値線はポテンシャル密度を示し, 低塩分陸棚水(LSSW)と高塩分陸棚水(HSSW)の密度下限も示してある. 右上はデータサンプリング領域.

トするためには, 閾値として比較的大きな値を選ぶ必要があると言える. 沿岸ポリニヤ内部での海氷流動に関しては, その方向は閾値によらないものの, その大きさは依存し, 閾値が大きいほど強い発散場が形成される.

座礁氷山による海氷堰き止め効果について, 実験NO-GIBでは座礁氷山の西側における沿岸ポリニヤは消滅し, 海氷生成も著しく減少する(ダーンレー岬ポリニヤ: 165.8±24.74 km^3, メルツ氷河ポリニヤ: 68.9±6.74 km^3, ニニス氷河ポリニヤ: 24.7±2.38 km^3). 実験LANDにおいては, 海氷生成に関しては実験CNTLと同様である(ダーンレー岬ポリニヤ: 195.4±22.99 km^3, メルツ氷河ポリニヤ: 102.7±9.43 km^3, ニニス氷河ポリニヤ: 30.4±3.38 km^3). 実験間で水塊[10]特性を比較するために, メルツ氷河ポリニヤにおける座礁氷山の西側における水温－塩分ダイヤグラム[11]を第 4 図に示す.実験NO-GIBにおいては,沿岸ポリニヤの消滅に対応して,$\sigma_\theta > 27.88$ で定義される高密度陸棚水は存在しない. 実験CNTL, LANDでは高密度陸棚水が存在し, その量と生成率は互いに同じ程度である. しかしながら, 実験LANDにおける高密度陸棚水は実験CNTLに比べてかなり塩分が高く, したがって密度も高い. 実験CNTLの結果における最大塩分はメルツ氷河ポリニヤの冬季の観測値に近い. 実験LANDでは, 座礁氷山の地点において高塩分のブラインが周囲の水と混合することが抑制されてしまう. ダーンレー岬ポリニヤとニニス氷河ポリ

[10] 海洋物理学においては, ある特徴的な水温・塩分(もしくは溶存物質濃度)を持って比較的広い領域に分布する海水の総体を指して, 水塊と呼ぶ. 水温・塩分は海面においては大気との相互作用によって変化するが, 海洋内部では保存される量である. したがって, 同じ水温・塩分特性を持った海水が海洋中に広く分布することは, 特定の領域の海面付近でその特性を獲得した海水がその分布に沿って流れていることを意味する. 海洋観測においては, 水温・塩分にくらべて他の溶存物質の観測は容易ではなく,流速の観測はさらに困難である(とくに深層において).観測される量から海洋内部の流れの状態を知ることは海洋物理学の根源的な命題のひとつであり, 水塊の特性と分布を調べることはその古典的手法のひとつである.

[11] ある範囲からサンプリングされた海水の水温と塩分について, それらを座標軸として 2 次元プロットを行ったもの. 水塊の形成や混合のプロセスを見るために用いられる.

ニヤについても同様のことが生じている．座礁氷山の下流に形成される沿岸ポリニヤにおける高密度水陸棚水を現実的にシミュレートするためには，座礁氷山による海氷堰き止め効果を表現することはもちろんのこと，その下では流れが堰き止められることなく存在することが表現されることも必要である．

（４）議論

海氷のシミュレーションにおいて，沿岸ポリニヤにおける海氷生成は厚氷と薄氷を分ける閾値というパラメータに強く依存する．本研究の結果が示すところは，沿岸ポリニヤにおいて現実的な海氷生成と海氷流速発散場を再現するためには，比較的大きな閾値を選択しなければならないということである．現実においては，沿岸ポリニヤはできたての氷で満たされている[12]．このできたての氷は力学的強度をほとんど持たない．沿岸ポリニヤを維持するのに必要な大きな海氷流速発散は，この力学的強度の小ささに起因する．通常の海氷モデルにおいては海氷の力学的強度は開水域の割合に対して指数関数的に減少するように扱われるため，モデルにおいては沿岸ポリニヤを低密接度領域として表現する必要があり，そのためには比較的大きな閾値が求められる．こうした取り扱いは，観測事実と対照しても妥当なものである．観測によると，できたての氷は 10 cm程度になるまでは開水域で成長し，その後 40 cm程度までは主に乗り上げ[13]によって厚くなる．したがって，たとえ多カテゴリーの海氷厚表現を用いたとしても，こうした過程がモデルで妥当に表現されるのでなければ，今回と同様な考慮なくして沿岸ポリニヤを現実的にシミュレートすることはできない．

第 1 表に挙げた全ての高密度陸棚水が半年のうちに南極底層水へと変換されると仮定すると，これによる南極底層水の生成率は $2.4\times10^{6}\ \mathrm{m^{3}\ s^{-1}}$ と見積もられる．この大きさはウェッデル海[14]に対して見積もられているものと同程度であり，東南極の沿岸ポリニヤにおける高密度陸棚水生成の重要性を示すものと言える．

４．ウェッデル海での棚氷水の沈降に関するシミュレーション

（１）はじめに

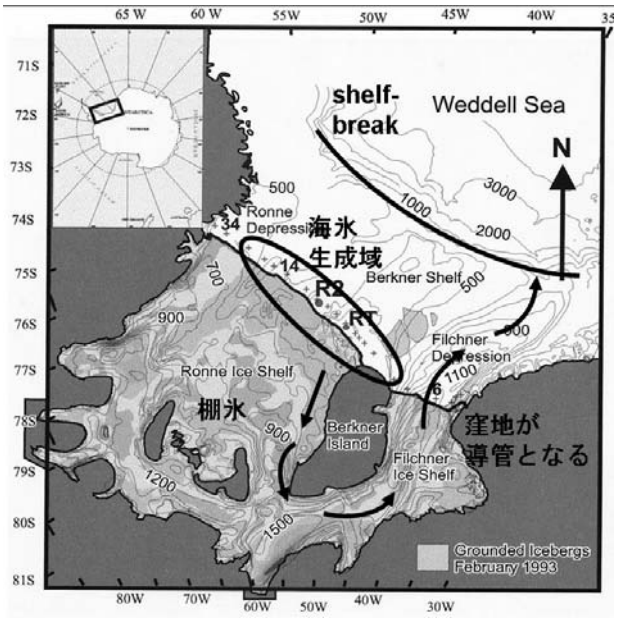
南極沿岸では活発な海氷生成によるブライン排出を起源とする高密度水が深層に沈み込んでおり，これが南極底層水の起源になっている．このような深層水形成は海洋大循環を駆動する要因でもあり，その定量的な理解は地球の気候を論ずる上でも重要である．

---

[12] 海水が凍結して海氷ができるとき，いきなり板状の氷が作られるわけではない．最初はフラジルと呼ばれる針状の結晶が析出し，融けかけのシャーベットのような状態にある．このような状態でも海氷の存在量を厚さに換算してしまえば数cm以上になってしまい得るが，実際には氷の塊として存在するわけではないので，水のように自由に流動する．その次の段階として，フラジルの凝集により，蓮葉氷と呼ばれる，文字通り蓮の葉状の薄い板状の氷が形成される．

[13] 板状の海氷の塊同士が衝突するとき，蓮葉氷のように海氷があまり厚くない状態のうちは，一方が他方に乗り上げるという現象が起こる．これは，海氷が力学的強度をほとんど持たないことに相当する．十分に厚い海氷同士がぶつかる場合には，衝突箇所において破壊が起こりつつ圧縮されて厚さが増加する．

[14] §４で取り扱われる海域．大西洋から南に進んだ突き当りに相当する部分で，南極半島と呼ばれる南極大陸の突き出した部分の東側に存在する，南極大陸が高緯度側に窪んだ場所．南極底層水の主要な形成箇所として古くから着目されてきた海域である．これに対して，ここで扱っている東南極は，これまではそれほど重要視されてこなかった．



第 5 図：ウェッデル海奥における高密度水の形成場所と経路.

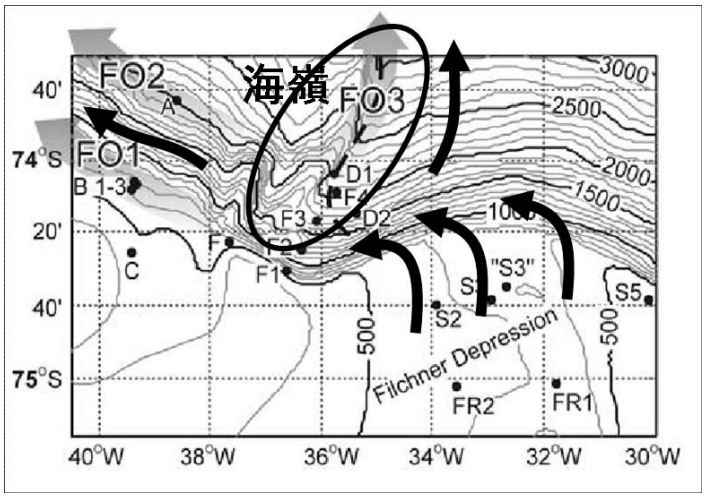
沿岸で形成された高密度水は，まず沿岸付近の大陸棚上の海底付近に溜まる．これが全世界の海洋の深層を占める高密度水になるまでには，大陸棚上から大陸斜面上へと流出し，大陸斜面上を深層まで下降しなければならない．このような高密度水が重力の作用によって斜面を下る流れのことを重力流と呼ぶが，地球という回転系においてはコリオリ力が作用するため，高密度水は重力流となって斜面を下るというよりも，重力とコリオリ力の斜面に沿う方向の成分がバランスすることによって斜面上の等深度を流れる傾向がある．したがって，ただ大陸棚上で高密度水が生成されるというだけでは，深層水が形成されるとは言えない[15]．重力流の沈降のためには，コリオリ力と重力の間のバランスを崩す何らかの作用が必要となる．

そのバランスを崩す要因としては，海底摩擦の効果，傾圧不安定[16]に伴う渦による輸送効果，斜面上の地形起伏の影響など，様々なものがこれまでに指摘されている．しかし，これまでの研究ではそれらの個々の要素が理論的あるいは理想化設定の数値シミュレーションとして扱われるばかりであり，実際の海洋においてどのプロセスがどの程度働くのかについてはほとんど調べられていない．

本研究では，ウェッデル海における深層水形成の主要な起源となっているフィルヒナー流出を対象に，現実的設定のもとで高解像度数値シミュレーションを実施する．ウェッデル海の奥部は棚氷によって覆われており，その棚氷の端には冬季に沿岸ポリニヤが形成される．ここでは活発な海氷生成に伴って高密度水が生成されるが，直下の大陸棚上に落ちた高密度水は即座に外洋に向かうのではなく，逆に棚氷の下にもぐりこみ，そこを反時計回りに循環して再び棚氷の外に出る(第 5 図参照)．棚氷の下面は結氷温度に保たれているが，海面よりも 100 m以上

[15] 大陸棚から遠い場所で高密度水が形成される場合には，このような過程を経ずとも深層水に直結することができる．例えば，カナダとグリーンランドの間にあるラブラドル海では，そのような深層水形成が生じている．

[16] 海洋や大気のような回転成層(密度が鉛直方向に変化して層を成すこと)流体に特徴的な力学的不安定．密度フロント(密度が水平的に急激に変化する領域)に伴うシア流の存在を原因として，密度フロントが持つ位置エネルギーを解放して渦運動エネルギーを生じる．



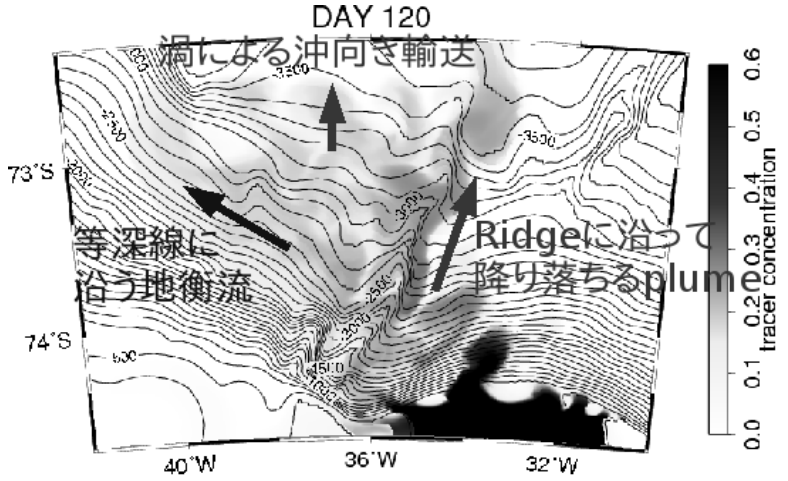
第 6 図：フィルヒナー流出の経路.

深くに存在するため，圧力の影響によってそれは海面で実現される結氷温度よりも低くなる[17]．その影響を受けるために，棚氷下から再び外に出た海水は非常に低温であるという特徴を持つ．こうして生成される低温の高密度水は棚氷水と呼ばれる．この棚氷水はフィルヒナー陥没と呼ばれる海底地形のくぼみに沿って大陸棚の端へと達し，大陸斜面上へと流出する．フィルヒナー陥没の出口においては，このようにして局所的に継続的な高密度水供給が存在する．流出した棚氷水はコリオリ力の作用のために大陸斜面上をほぼ等深度線に沿って流れるが，西経 36 度付近に存在する小規模な海嶺に当たって進路を北向き(沖向き)に曲げられ，等深度線を横切って沈降することが観測に基づいて知られている(第 6 図参照)．また，この高密度水の沈降においては，サーモバリック効果の働きが重要であることが指摘されてきているが，実際にどの程度の大きさをもってこの効果が沈降に寄与しているのかは明らかでない．本研究では海底地形起伏の影響が他の要素に比べてどの程度重要かについて，またサーモバリック効果が高密度水沈降にとってどの程度重要かについて，定量的に明らかにすることを目的とする．

（2）シミュレーションの設定と結果

用いる数値モデルは，§ 2 で述べた非静水圧海洋モデルである．シミュレーションの対象領域は，ウェッデル海内の西経 30-42 度，南緯 71-75 度の領域である(第 5 図および第 7 図等を参照)．水平解像度はおよそ 900 m，鉛直解像度は 50 m である．初期状態は，観測に基づく水温・塩分の鉛直分布を水平一様に与える．フィルヒナー陥没の中では深度 500 m より下を水温-2℃・塩分 34.7 psu に緩和し，継続的な棚氷水供給を表現する．また，沈降する棚氷水の流路および流量を追跡するために，この緩和領域において値を常に 1 とし，それ以外の場所では初期値を 0 とする仮想トレーサーを導入する．この仮想トレーサーは水温や塩分と同じ輸送・拡散方程式を用いて値が予報される．初期状態から 180 日の計算を行い，解析に用いる．

[17] 海水の結氷温度は塩分と圧力に依存する．典型的な海水塩分である 34～35 psuのもとでは，1 気圧における結氷温度は-1.8℃程度であり，海面付近の水温はこれより下がることは無い．海洋中には少なくとも冷却源は存在しないため，これよりも低い水温は通常は実現されない．ここで説明している棚氷下面との接触部は例外であり，-2℃を下回る水温が実現される．



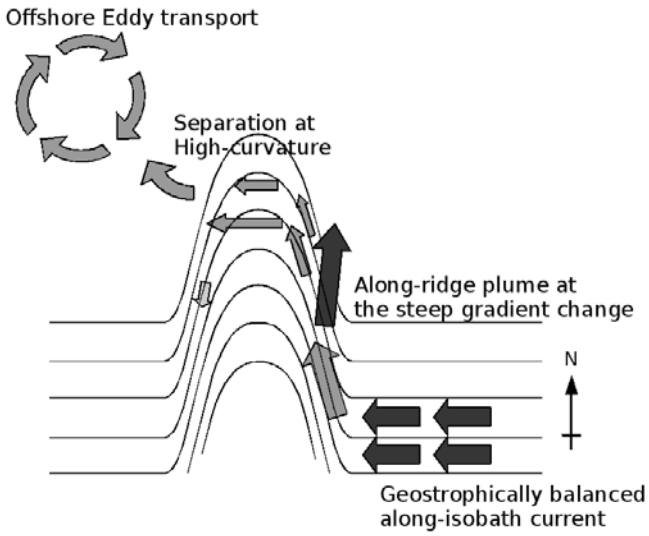
第 7 図：120 日目の海底付近の仮想トレーサー濃度.

実験 120 日目における海底の仮想トレーサー濃度を第 7 図に示す．フィルヒナー陥没の出口で供給された高密度水は，大陸斜面上をまず西向きに流れ，海嶺にぶつかる場所で手前にある谷線に沿って下降している．こうした全体的な様相に加え，直接観測が存在する場所での水温・塩分分布や流速は，観測とよく整合している(Foldvik et al., 2004)．仮想トレーサー分布からわかる通り，高密度水の主要な流路は海嶺の手前を下降するものだが，海嶺の先端を回り込んで等深度線に沿ってさらに西向きに流れる成分も存在する．また，海嶺の先端からは渦が切り離され，これもまた高密度水を沈降させる働きを持つことがわかる．

（３）海底地形の起伏の効果

斜面上において重力とコリオリ力の間のバランスだけを考慮する場合，コリオリパラメータ f，等深度線に沿う流速 U，斜面の勾配 α，重力加速度 g，高密度水の密度 ρ，高密度水と周囲水の密度差 $\Delta\rho$ に対して $fU = -g\alpha\Delta\rho/\rho$ が成り立つ．この場合，任意の f，α，ρ，$\Delta\rho$ に対して，バランスを満たす U が存在する．ただし，等深度線が曲率を持っている場合には，等深度線に沿う流れのバランスの中に遠心力の項が加わる．曲率半径を R とするとき(等深度線が沖に向かって凸の場合を正とする)，バランスの式は $fU + U^2/R = -g\alpha\Delta\rho/\rho$ となる．このバランスを満たす U が存在するためには，R に下限値 $R_c = 4g\alpha\Delta\rho/\rho f$ が存在する．これよりも小さい正の曲率半径を持つ場合，等深度線に沿う流れは斜面に沿うことができず，剥離して外洋側へと向かう．外洋に向かう過程では高密度水層が上下に引き伸ばされることになるため，ポテンシャル渦度[18]保存より高密度水層は f と同じ符号の渦度を獲得する．フィルヒナー流出における典型的な条件で R_c を見積もるとおよそ 4 kmとなり，海嶺先端の曲率半径 3-5 kmとよく対応する．したがって，第 7 図に見られた海嶺先端から切離する渦は，このような理由によってできたものだと考えられる．

[18] 地球の回転による渦度である f と回転系上で考えた渦度(相対渦度)の和を流体層の厚さで割ったもの．回転流体においてはこれが保存量となる．



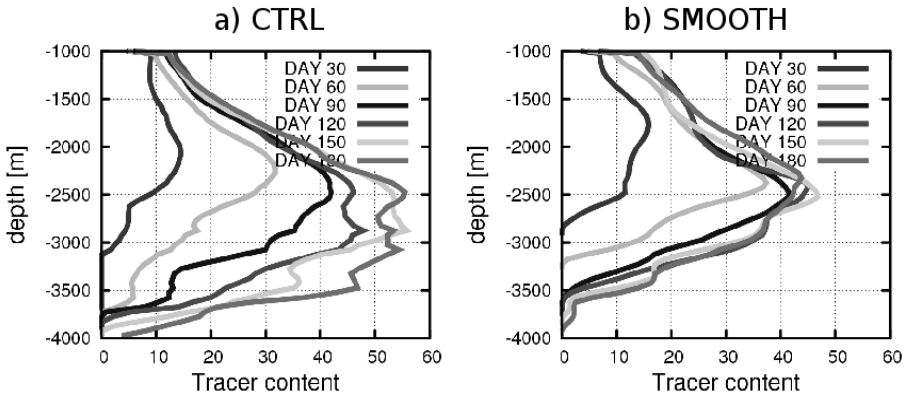
第 8 図：高密度水が等深度線を横切るメカニズムの概念図.

一方，海嶺の手前の部分は曲率が負であり，遠心力は流れを斜面に押し付ける方向に働くため，高密度水の沈降を阻害する要因として働く．その一方で，この部分では等深度線に沿って進むにつれて斜面の勾配が急になっていく．この場合には，バランスの式において，斜面勾配が空間的に変化する影響を表す項が新たに追加されることになる．ここでは具体的な式は割愛するが，その結果として，曲率・コリオリパラメータ・斜面勾配それぞれについて等深度線に沿って考えた微分が問題となり，それらの兼ね合いによって沈降が生じるかどうかが決まる．これについてもやはりフィルヒナー流出における典型的な条件で評価したところ，海嶺の手前は確かに高密度水の沈降が生じるべき場所にあたっていた.フィルヒナー流出の沈降について，地形起伏の力学的影響を概念的にまとめたものを第 8 図に示す.

仮想トレーサーの量を各深度で水平積分することにより，フィルヒナー流出で与えられた高密度水がどの深度に供給されるかを調べた結果を第 9a 図に示す．分布は 2500-3500 m にわたって幅広いピークを持ち，高密度水の一部は 4000 m の深さまで輸送されていることがわかる．海嶺を平滑化した行った実験の結果では，高密度水は深度 2500 m を中心に分布し，3000 m より下にはほとんど輸送されていない(第 9b 図).海嶺が存在しない場合に高密度水の沈降を担うものは，海底摩擦と傾圧不安定の影響によるものと考えられる．今回の対象領域においては，高密度水が深層に沈降する要因として，海底起伏の効果が海底摩擦や傾圧不安定の効果を大きく上回ることが示された.

（4）サーモバリック効果

海水の密度は水温・塩分・圧力の函数として決まるが，その状態方程式が持つ非線型性に起因したいくつかの特徴的な現象が知られている．サーモバリック効果はそのひとつで，海水の圧縮率が水温に依存することによって現れる効果である.例えば一様な水温(正確にはポテンシャル温度)の海水の中をそれよりも低温の水が沈降する場合,深くなるにつれて低温水の方が強く圧縮されるため，周囲の水との間の密度差は大きくなっていき，沈降が促進されることになる.



第 9 図：各深度で水平積分した仮想トレーサー量.

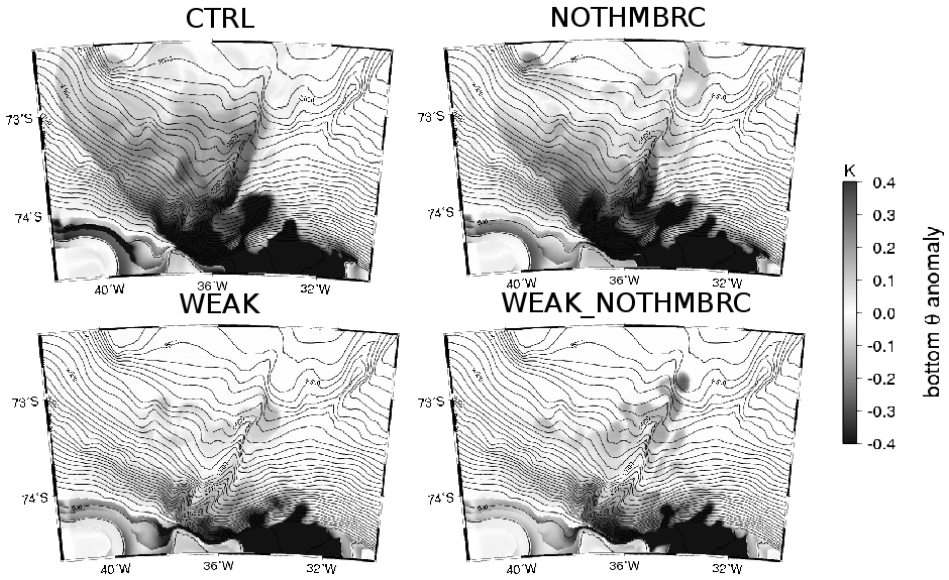
a）標準のシミュレーション結果，b）海嶺を取り除いたシミュレーション結果.

フィルヒナー流出についてこの効果を定量的に評価するために，サーモバリック効果を取り除いたシミュレーションを行った．これは，モデルで用いている状態方程式において，水温と圧力の積の項については水温を一定の参照温度にすることで行っている．その結果，高密度水の最大到達深度は 3400 m 程度となり，確かにサーモバリック効果が高密度水沈降において量的に無視できない寄与をしていることが示された．ただし，他に様々な感度実験を行う中で，サーモバリック効果が必ずしも高密度水の沈降を促進するばかりではないことも明らかとなった．例えば，フィルヒナー流出の大陸棚上からの供給量を減らした実験を行ったところ，サーモバリック効果を取り除いた方が逆に深くまで高密度水が沈降するという結果が得られた．

前述の通り，サーモバリック効果が沈降を促進するのは，高密度水が周囲の海水よりも低温の場合である．いま対象にしているウェッデル海では，沈降する高密度水の周囲にある海水の密度(正確にはポテンシャル密度)は鉛直方向にほぼ一定であるが，水温と塩分は鉛直方向に変化している．すなわち，上方では比較的高温・高塩分，下方では比較的低温・低塩分になっている．この中を低温の高密度水が通過するとき，上方で強い混合を受けてしまうと，高密度水があまり深くまで達しない段階で，沈降する高密度水の水温が周囲の水温よりも高くなってしまう(密度はあくまでも沈降する側の方が高くても)．このような場合にはサーモバリック効果は高密度水の沈降を阻害する．実際，上述の供給量が少ない実験においては，そのような状況が確認された(第 10 図)．

（５）議論

サーモバリック効果がどのように働くかについては，上述の通り，高密度水と周囲の水との混合がどこでどの程度生じるかが重要な要素となる．この混合の重要性は，サーモバリック効果の現れ方に限られたものではない．大陸棚上などで局所的に生成された高密度水が深層に達するまでの間には，様々な場所で様々な要因によって周囲の海水と混合する．これによって深層水は起源となる水より低密度化していく一方，その体積を増やしていく．この混合過程はエントレインメントと呼ばれる．第 1 図に表した海洋大循環では，10^7 m^3 s^{-1}を越える流量の海水が常に表層海洋と深層海洋をつないで巡っているのだが，それだけの量が実現されるのはエントレインメントが起こればこそである．エントレインメントがなければ，こうした海洋大循環の流量は 1 桁程度少ないものになるであろう．



第 10 図：周囲の海水を基準とした，高密度水の水温偏差.

CTRL は標準実験，NOTHMBRC は CTRL からサーモバリック効果を取り除いた実験，WEAK は高密度水供給量を少なくした実験，WEAK_NOTHMBRC は WEAK からサーモバリック効果を取り除いた実験.

このように，シミュレーションにおいてエントレインメント過程を適切に表現することはとても重要であるが，この点に関しては未だ多くの課題を抱えている．エントレインメントをもたらす混合過程を陽にシミュレートしようとするならば，格子間隔を水平方向には 100 m より十分小さく，鉛直方向には 10 m より十分小さくしなければならない．そのようなシミュレーションを実行してエントレインメント過程を定量化することには大きな重要性があり，多くの計算機資源を投入して実施する価値がある．しかしその一方で，あらゆる目的に対してエントレインメント過程を陽に表現したシミュレーションを行うというのは，甚だ非現実的である．LES やパラメータ化などを通して，限られた計算機資源の中ででもその効果を適切に表現していくようなモデル開発もまた，今後の重要な方向性である．

５．おわりに

全地球規模の環境変動予測のために，それに対する防止策や適応策の模索のために，スーパーコンピューティングが果たすべき役割は実に大きい．地球温暖化に代表される大規模な長期気候変動において海洋がどのような役割を果たすのか，そして結果として生じる海洋変動が人の生活環境にどのような影響を与えるのか．これらを明らかにすることができる連結階層型海洋シミュレーションシステムの開発を目指し，筆者らは研究・開発に取り組んでいる．その構築に必要となる海洋大循環のコントロールプロセスの抽出･定量化とパラメータ化等について，現在も着々と成果が得られているところであるが，より広い領域を対象とするより高解像度のシミュレーションがなおも必要であり，次世代スーパーコンピュータをはじめとする高性能計算機への期待は高い．ただし同時に，筆者らが開発している数値モデルが，未だ見ぬ超高並列のもとで十分な性能を発揮しうるのかについての不安も大きい.

本プロジェクトを通して少なからぬ HA8000 のリソースを使わせていただいた上で抱いた感想は，事前に覚えた不安に比べれば「意外と何とかなりそう」というものである．とはいえ，

今回経験した並列度は1,000のオーダーに過ぎず，また，入出力等の意味でヘビーな使い方をしたとも決して言えない．今後さらなる開発やチューニングを行い，今後の計算機資源を有効に活用できるよう，努力していきたい．

参 考 文 献

Foldvik, A., et al. (2004)：Ice shelf water overflow and bottom water formation in the southern Weddell Sea, *J. Geophys. Res.*, **109**, C02015.

Hasumi, H. (2006)：CCSR Ocean Component Model (COCO) Version 4.0, CCSR Report No. 25, 103pp. (http://www.ccsr.u-tokyo.ac.jp/~hasumi/COCO/coco4.pdf)

Marsland, S. J., N. L. Bindoff, G. D. Williams and W. F. Budd (2004)：Modeling water mass formation in the Mertz Glacier Polynya and Adelie Depression, East Antarctica, *J. Geophys. Res.*, **109**, C11003.

Matsumura, Y., and H. Hasumi (2008)：A non-hydrostatic ocean model with a scalable multigrid Poisson solver, *Ocean Modelling*, **24**, 15-28.

Tamura, T., K. I. Ohshima, S. Nihashi (2008)：Mapping of sea ice production for Antarctic coastal polynyas, *Geophys. Res. Lett.*, **35**, L07066.

HA8000 システムでの地球ダイナモシミュレーションと可視化

陰山 聡，大野 暢亮

海洋研究開発機構 地球シミュレータセンター

1. はじめに

HPC 特別プロジェクトの一つとして，今回我々は，地球シミュレータ向けに開発・最適化した地球ダイナモコードを HA8000 システムに移植した。このコードは，地球シミュレータ上の主に 512 ノード（4096 プロセッサ）を使った大規模な並列計算によるプロダクトランに用いていたものである。本研究の目的は，地球シミュレータに最適化された我々のコードが，そのまま HA8000 システムの多ノードで走るか？ HA8000 システムへの移植にはどの程度の作業が必要か？そのまま移植した場合，HA8000 システムの多ノードでどの程度の速度が出るか？限られた時間内で最小限の最適化によってどの程度の加速効果が見られるか，などの疑問に答えることであった。ここではその結果について報告する。なお，我々はベクトル計算機の利用とチューニングに関してはある程度経験をもっているが，スカラー型の，特に大規模な並列計算機を使うのは今回が初めてであり，スカラープロセッサへのチューニングについて知識も経験もほとんどない。従って，我々の技量不足からこのシステムの性能を十分に引き出していないことは間違いないので，この報告はその点を斟酌して読んでいただきたいと思う。また，今回のプロジェクトでは，シミュレーションデータをスーパーコンピュータ上で直接解析するために我々が開発している並列可視化ツールも移植し，最大 8192 コアまで使って様々な可視化も試したので，これに関してもあわせて報告する。

まず始めに，地球ダイナモシミュレーションについて簡単に紹介する。コンパスが北を指すのは，地球が双極子磁場に覆われているからである。その双極子磁場は，地球内部に流れる 10 億アンペア程度のリング状の電流が作りだしている。このリング状電流を生み出す発電のメカニズムが地球ダイナモである。この電流の直接のエネルギー源は，地球内部に存在する液体金属の流れである。地球の中心部，岩石でできたマントル層の内側には核と呼ばれる領域がある。核は鉄でできている。この核は二層に分かれており，外側は外核と呼ばれる液体領域，内側は内核と呼ばれる固体領域である。外核の液体鉄は対流運動しており，この対流運動が磁気流体力学（MHD）ダイナモと呼ばれる機構を通じて地球磁場を作り出している。磁場中を電気伝導性流体が流れると，誘導起電力により電流が生じ，その電流が元の磁場と同じ形の磁場を作っていればこれは正のフィードバックとなり磁場が強まる。これが MHD ダイナモの原理である。外核の対流の駆動源は未解明だが，我々のシミュレーションでは簡単のため熱対流のみを考える。基本方程式は MHD 方程式である。

シミュレーションモデルは以下の通りである：地球の外核を想定し，球殻状の容器に電気伝導性流体（MHD 流体）が入っているものとする。内側の球面は高温，外側の球面は低温に保たれている。球の中心方向に重力がはたらき，二つの球殻は同じ角速度 Ω で回転する。温度差が十分に大きければ内部の流体は熱対流運動し，MHDダイナモ機構によって，対流の運動エネルギーが磁場のエネルギーに変換され，磁場が生成される。比較的低い解像度で十分であればこの問題を数値的に解くのはそれほど難しいものではない。我々も含めて世界中のグループが

地球ダイナモシミュレーションに挑戦し，これまでに双極子磁場の自発的生成やその非周期的逆転現象など，地球磁場に特徴的な性質を計算機の中で少なくとも定性的には再現することに成功している。スーパーコンピュータの進歩に伴い，地球ダイナモシミュレーションも着実に進歩してきた。その進歩の指標の一つとしてよく使われるのは，この系を特徴づける無次元量の一つであるエクマン数 $Ek = \nu /2\Omega R^2$ である。ここで ν は動粘性率，Rは外核の半径である。地球外核のエクマン数は $O(10^{-15})$ 程度と見積もられる。エクマン数は粘性率に比例するので，その値が小さな系ほど，高い空間解像度を要求され，シミュレーションが難しくなる。現在のスーパーコンピュータでも$Ek = 10^{-15}$のシミュレーションは不可能である。1990 年代に我々が地球ダイナモシミュレーション研究を始めたとき，計算で用いたエクマン数は$O(10^{-4})$であった。スーパーコンピュータの進歩と共に，地球ダイナモシミュレーションで使われるエクマン数はゆっくりとではあるが着実に小さくなっており，我々は最近，$Ek = 10^{-7}$の計算に世界で初めて成功した[1]。

地球ダイナモシミュレーションで用いられる空間離散化手法としては球面調和関数を基底関数としたスペクトル法が長年主流であったが，この手法は大規模な計算には向かないので，近年では有限体積法や有限差分法，有限要素法などに基づく地球ダイナモシミュレーションコードが開発されている。 我々は長年，有限差分法に基づく地球ダイナモシミュレーションコードを使っている。基本計算格子として現在採用しているのは，図 1 に示したインヤン格子である。

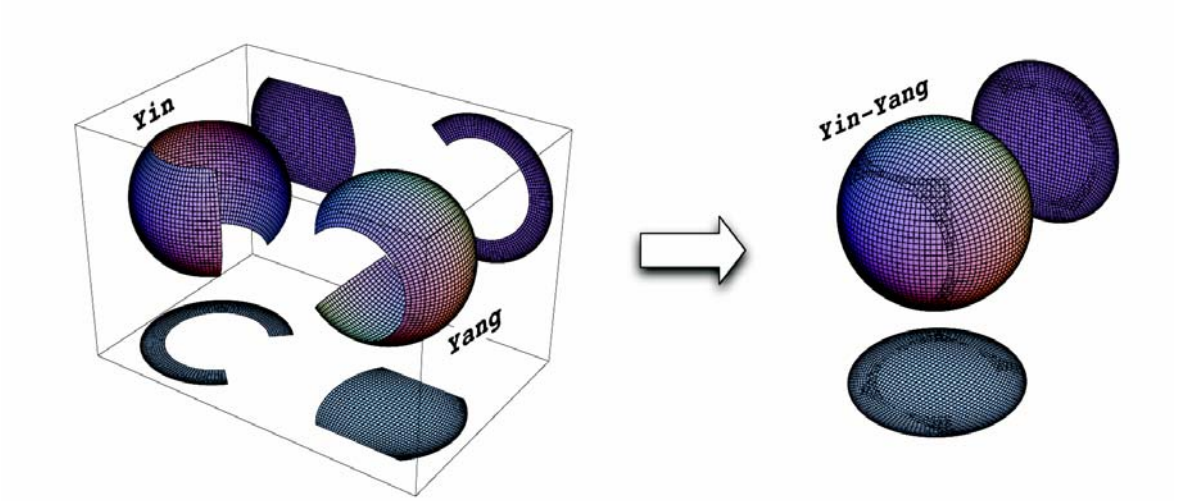


図 1. インヤン格子

二つの合同な要素格子を組み合わせて球面を覆うキメラ格子の一種。球座標の低緯度部分を切り出して一つの要素格子（イン格子）を作る。それを回転させると一つの要素格子（ヤン格子）ができる。

通常の球座標に基づく格子（緯度経度格子）は，高緯度部分，極の近くで格子間隔が不必要に集中するために計算格子として望ましくない。しかしながら低緯度部分の領域をだけを見れば，これは数値計算向きの「タチの良い」格子系である。直交格子であり，格子間隔はほぼ一定で，しかもメトリックは単純でよく知られているからである。通常の球座標上，緯度方向に赤道を挟んで南北 45 度ずつの領域を切り取り，経度方向には（360 度ではなく）270 度切り取った領域を考える。これは球座標の計算空間では長方形の領域である。これを一つの要素格子とし，まったく同じものをもう一つ用意して，実空間で回転させて組み合わせれば球面全体を覆う。これがインヤン格子である。我々のコードではこのインヤン格子上で MHD 方程式を解

いている。空間離散化は 2 次の中心差分，時間発展には 4 次精度のルンゲ＝クッタ法を用いている。

地球シミュレータ上の計算ではベクトル化は球殻の半径方向にかけていた。並列化は，イン格子とヤン格子のそれぞれで水平方向（緯度・経度方向）に 2 次元領域分割し，MPI を用いる。それぞれの要素格子で独立の MPI コミュニケータを構成することで MPI 通信のためのコーディングを簡略化させている。イン格子，ヤン格子それぞれの水平方向の境界上でのデータは，キメラ格子手法に基づき相互補間で設定する。

2. HA8000 システムへの移植

移植作業は順調であり，コードの変更は以下に述べるわずかな点のみで済んだ。

変更点 1 ：我々のコードでは，全エネルギーを計算する場合などに MPI_ALLREDUCE 関数の SUM 機能を使っている。この関数は（MPI では）第一引数と第二引数に与える送信バッファと受信バッファを個別に用意しなければいけなかったが，MPI2 では第一引数に MPI_IN_PLACE を与えることで余分なバッファが不要になるので便利である。HA8000 システムではデフォルトの MPI がこの MPI_IN_PLACE の機能に対応していなかったので，利用する MPI を MPI2 に切り替えようとしたが，それがなぜかうまくいかなかった。そこで今回は，我々のコード中の MPI_ALLREDUCE で MPI_IN_PLACE を使わないように変更した。なお，我々のコードでは，MPI_ALLREDUCE も含めてほとんどの MPI 関数は，mpiut.f90 という自作の wrapper モジュールで wrap している。これは Fortran90 に用意されている関数の多重定義機能を利用して，ソースコード中の MPI 関数の呼び出しを簡潔でわかりやすくするためである。従ってソースコード中の MPI_ALLREDUCE の変更はわずかで済んだ。

変更点 2：我々の元々の Fortran90 コードでは MPI ライブラリを使う各モジュールで MPI モジュールを use する（use mpi）ことで MPI ライブラリ使うようにしていたが，デフォルトではこの方法は使えないようであった。この問題も MPI2 を使えば解決できるらしいことがわかったが，上述したとおりなぜか MPI2 への切り替えに失敗したので，今回は ‘include mpi.f’ とすることで回避した。

以上，述べたとおり，地球シミュレータから HA8000 システムへの移植は簡単で，短時間のうちに終了することができた。

3. Numacntl の仕方

移植した我々コードを flat MPI で計算した場合，numactl を以下の指定で行うときに最も大きな加速効果（約 20%）が見られた。

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
NODE=$(expr $MYVAL % 4)
numactl --cpunodebind=$NODE --membind=$NODE $@
```

以下に述べる計算は全てこの numactl で計算したものである。また，

```
numactl --cpunodebind=$NODE --interleave=$NODE $@
```

としたときも（わずかながら遅かったが）ほぼ同様の効果が見られた。

4. 高速化に関するいくつかの試み

ここでは計算時間短縮のために行ったいくつかの試みと，その結果について述べる。今回は時間が限られていたので，網羅的で系統的なテストにはなっていない。なお，今回試した並列化手法は，flat MPI だけである。

コンパイラによる速度の違い：日立のコンパイラ（オプション –Oss –noparallel）と比較して，インテルのコンパイラ（オプション –O4）の方が約 10%ほど高速であった。

配列サイズ依存性：我々のコードでは，物理量は（半径，緯度，経度方向に対応する）3 次元配列で表されている。ある物理量 p を p(N1,N2,N3)と書いたとき，第 1 次元のサイズ N1 は，地球シミュレータの場合，ベクトルレジスタの数から決めていた。一方，第 2 次元と第 3 次元方向には領域分割をかけるので，利用するノード数（MPI プロセス数）によってそれぞれの大きさは変動する。分割する前の第 2 次元と第 3 次元方向のサイズをそれぞれ（我々のダイナモシミュレーションのプロダクトランで典型的な値である）514 と 1538 に固定し，第 1 次元のサイズ N1 を以下の表のように変化させて速度を測定した。これは 128 ノード，2048 コアでの計算である。

N1	507	508	509	510	511	512	513
GFLOPS	502	535	531	531	528	543	482

これから分かるとおり，N1=512 のときに最も早く，それから 1 グリッド増やしただけで（N1=513）速度は約 11%落ちた。この結果から，今回は N1=512 の計算を主に行った。

ホットスポット：128 ノードでの計算結果を見ると，コード中で，MHD 方程式のソルバに約 80%の計算時間費やされ, ルンゲ＝クッタ法による時間積分ルーチンに 10ー15%の時間が費やされていた。これは地球シミュレータにおける割合とほぼ同じである。そこで基本アルゴリズムは変えずに，MHD ソルバのソースコードを変更して加速する試みをいくつか行った。その中で，意外にも効果がなかったものと，逆に意外にも効果があったものを紹介する。

効果のなかったもの： MHD 方程式には基本物理量（3 次元配列）が 8 個ある。これらをひとまとめにして，第 1 次元のサイズが 8 の 4 次元配列を定義すれば，キャッシュを有効利用できると期待される。この方法は，スカラー型スーパーコンピュータでの MHD シミュレーションでは効果的な加速手法であることが知られている [2]。今回，我々のコードに対してもこの方法を試してみたが，残念ながら効果はなく，むしろ遅くなってしまった。

効果があった方法： MHD ソルバの基本部分は 3 次元配列に対する 3 重 do loop である。ある物理量を p(i,j,k) とすると, 3 重 do loop の再内側ループのインデックス i が走る範囲は 2≦i≦N1-1 である。この第 1 次元方向に一種の領域分割をかけ，配列を無理矢理 4 次元化してみた。つまり，第 1 次元方向を NDIV 個に分割し，4 次元の新しい配列 pp(ii,j,k,n)を作る。ここで 2≦ii≦N1D-1, 1≦n≦NDIV, i=(n-1)*(N1D-2)+ii である。そして，MHD ソルバにこの変数を渡すには，分割したそれぞれの領域毎に以下のように渡す。

```
do d = 1 , NDIV
```

```
        call mhd_solver(pp(:,:,:,d))
      end do
```

サブルーチン mhd_solver は，3 次元配列を受け取る MHD ソルバであり，基本的には元々の 3 次元 MHD ソルバと同じものである。この方法により，20%ほどの高速化がみられた。ただし，以下に述べる 512 ノードの計算では，この加速手法は使っていない。

5. HA8000 システム 512 ノードでの性能

この章では，地球シミュレータ用に最適化したコードに，第 2 章で述べた軽微な変更のみを加えたコードをそのまま 512 ノードで走らせた結果についてまとめる。これほど大きな規模の並列計算の実行では，何らかのトラブルが起きるものと覚悟していたが，ジョブ投入から実行まで，全て順調に進んだ。

ノード当たり 16 コア，計 8192 コアで約 40 分間実行したところ，プログラムの初期化処理のルーチンに全体の計算時間の 14%もかかった。これは異常に大きい。原因は不明である。この初期化処理では，メモリアロケーションや計算格子のメトリック設定，計算初期条件の設定など，シミュレーション開始時に一度だけ行う比較的複雑な処理を集めている。初期化処理の部分を除いて評価した計算速度は 1.8 TFLOPS であった。1 コアあたりの性能を 9.2 GFLOPS として計算すれば，8192 コアで 1.8 TFLOPS という性能は理論ピーク性能の約 2.4%に相当する。

次に同じ 512 ノードで，1 ノードあたり 8 個のコアだけを使った flat MPI を試してみた。その結果，初期化処理にかかる時間は全体の 4.7%に減少した。初期化処理の部分を除いて評価した計算速度は 1.5 TFLOPS であった。これは理論ピーク性能の約 4.0%である。

次に 512 ノードで，1 ノードあたり 4 コアを使った計算をしてみた。今度は初期化処理にかかる時間が 1.5%にまで下がった。地球シミュレータでの経験から言えば，この程度が通常の数値である。全体の計算速度は 0.87 TFLOPS であった。これは理論性能の 4.6%に相当する。

6. データ可視化について：インヤン 座標用可視化プログラム Armada

地球ダイナモシミュレーションの結果を可視化する上で，常に問題になるのは：(1) インヤン座標という特殊な座標系，(2) 出力されるデータサイズの大きさ，の 2 点である。

インヤン座標上のデータを直接可視化することが可能なソフトウェアは，我々の知る限り存在しない。ただし，インヤン座標は球座標の一部分の単純な組み合わせであるので，工夫すれば AVS/Express 等で可視化は可能であるし，球座標上にデータをマッピングすれば，多くの可視化ソフトウェアで可視化が可能となる。(2)については，我々のシミュレーションで出力するデータは単精度で出力しており，グリッドサイズはイン座標とヤン座標それぞれ 1 タイムステップ・1 変数ごとに約 3GB である。この規模のデータとなると，たとえインヤン座標上のデータを可視化できるソフトウェアが存在したとしても，インタラクティブに可視化するのは困難である。また，データ転送に多大な時間がかかる。

我々は，インヤン座標上の時系列データを直接可視化することが可能な，グラフィックス関係のハードウェアを一切必要としない並列可視化ソフトウェア“Armada”を開発している[3]。このソフトウェアは，C++で記述されていて，MPI および OpenMP で並列化されている。またグラフィックス関係のハードウェアを用いずに画像を描画するので，スーパーコンピュータ上で

も動作する。今回我々は，Armada を HA8000 上で動作させることにより，計算された結果をネットワーク経由で他のグラフィックスワークステーションなどの可視化用計算機に転送することなく,また球座標にデータをマッピングすることなく直接可視化を行う方針を立てた。今後，スーパーコンピュータの規模が拡大して計算結果がさらに大規模になると，これに類する可視化手法が一般的になると思われる。

Armada は地球ダイナモシミュレーション専用というわけではなく，汎用の CFD データ可視化ツールである。また，すべての可視化機能をソフトウェアレンダリングで実現している（グラフィックス関係のハードウェアを必要としない)。使用できる可視化機能は，スカラーデータでは，等値面・ボリュームレンダリング・カラースライス，ベクトルデータでは，流線・矢印表示である。スカラーデータの可視化については，すべての可視化手法にレイ・キャスティング法を用いている。レイ・キャスティング法でボリュームレンダリング以外の等値面やスライスを生成するのは，容易である。等値面生成は，2 つの連続するサンプリングポイント上の値を比較することで，レイが等値面を通過しかた否か判定できる。スライスについては，幾何学的にスライス面をレイが通過したか否か判定できる(使用するスカラーデータの可視化手法がスライスのみ場合は，幾何学的にレイとスライス面の交点を計算して描画する)。

スカラーデータとベクトルデータの可視化手法は，合計 5 種類存在するが，単独でも使用可能であるし，組み合わせて使用することも可能である。データは，複数読み込むことが可能で，例えば，温度の等値面と渦度の z 成分のスライス面，速度場の流線を同時に描くことも可能である。

Armada には，1 度の実行で，多数の可視化画像を生成する機能が備わっている。具体的には：(a) 各タイムステップ毎に複数，多数の視点を設置する機能，(b)各タイムステップ毎に複数の可視化パターンを適用させる機能，の二つを実装している。

まず(a)について詳しく述べる。Armada は時系列のデータを扱えるが，タイムステップ毎に 1 枚の画像だけでなくユーザーがあらかじめ設定した視点から，複数の画像を生成する機能がある(図 2 参照)。例えば，視点（位置，視線方向，視線上方の組）をあらかじめ 100 個用意した場合，データが 100 タイムステップあると 100x100 で合計 1 万枚の画像が生成される。

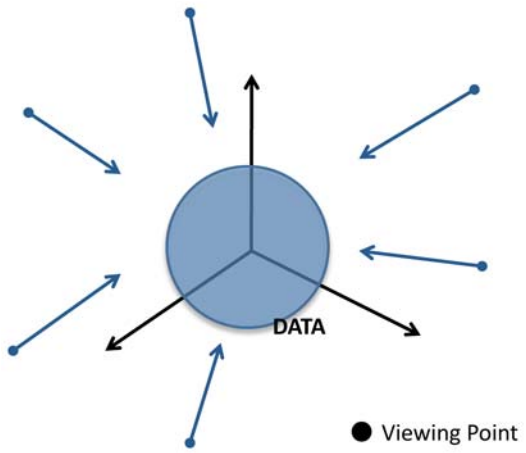


図 2. 複数視点からのデータ可視化の概念図

Armada ではあらかじめ設定した複数の視点からの可視化画像を作成することができる。

次に(b)について説明する。例えば,『可視化１：等値面（値 10.5)+スライス(z=10)，可視化 2：

等値面(値 6.8)，可視化 3：スライス（z=0)+流線 5 本』等の可視化パターンをユーザーが用意すると，各タイムステップで 3 種類ずつの可視化画像を生成する。視点が 1 つで 100 ステップのデータがあるとすると，合計 300 枚の画像を生成する。1. と 2. を組み合わせた場合，例えば，視点 100，可視化パターン 10，タイムステップ 100 あるとすると，合計 100x10x100=10 万枚の画像を生成し出力する。我々は，この大量画像生成機能を用いたリアルタイム可視化実験を将来おこなう予定である。

Armada の並列化については，ノード間並列を MPI，ノード内の並列化を OpenMP でおこなうハイブリッド並列化を採用している。

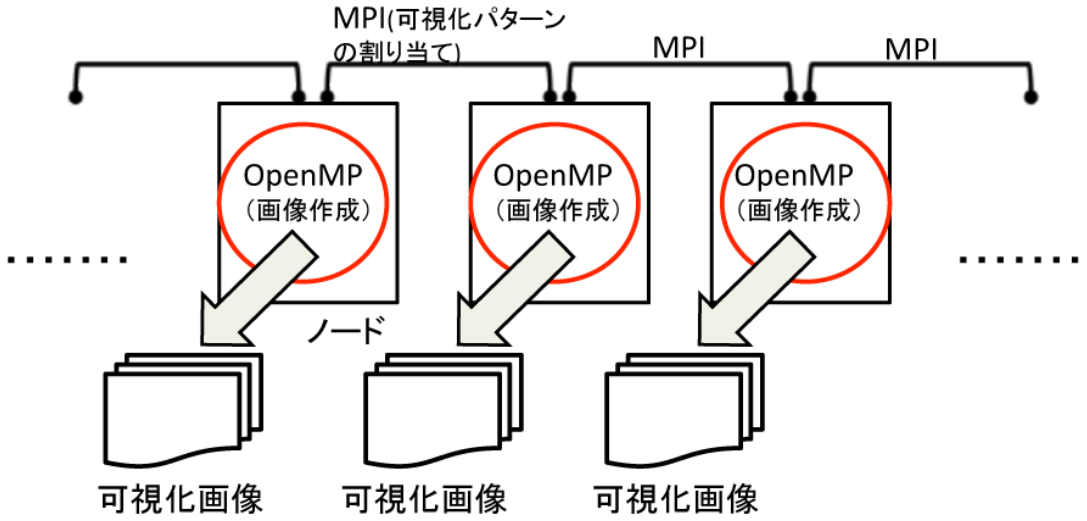


図 3. Armada の並列化

各ノードが独立して可視化をおこない画像を出力する。

Armada を実行すると，各ノード（正確には各 MPI プロセス）は基本的に独立して動作する。マスタープロセスから MPI 通信で割り当てられた“可視化”(可視化パターンとタイムステップが割り当てられる)を実行する（図 2 参照)。ノード内の並列化は，OpenMP を用いておこなわれており，可視化するデータは 1 ノード内に載ることが必要である。つまり 1 ノード内で走る可視化ソフトウェアが，複数のノードで同時に実行され，多数の可視化画像を出力するというイメージである。多数のノードに（領域）分割されたデータを読み込みこんで，１つの（１タイムステップの）巨大データを可視化する方式とは異なる。このソフトウェアの目的は，複数の可視化パターンの画像を多数の視点から可視化して大量の画像を保存することであること，スーパーコンピュータの 1 ノード内の共有メモリのサイズが増加傾向にあること，などの理由によりこのように並列化した。HA8000 の 1 ノード内の CPU コア数は 16，共有メモリは 32GB 搭載されているので，我々の 1 ステップ・1 変数あたり約 3GB のデータでも十分に可視化することが可能である。ただし，今後 1 ノード内に可視化すべきデータが載らなくなるような場合には，並列化を再検討する。

今回我々は Armada を HA8000 システムの 512 ノード，8192 コアでも動作することを確認した。以下に HA8000 で作成した地球ダイナモシミュレーションデータの可視化画像を紹介する。図 4 は比較的粗い解像度でのシミュレーションデータの可視化例である。図 5，6，7 は，１変

数約 3GB のデータを高解像度データを可視化した例である。

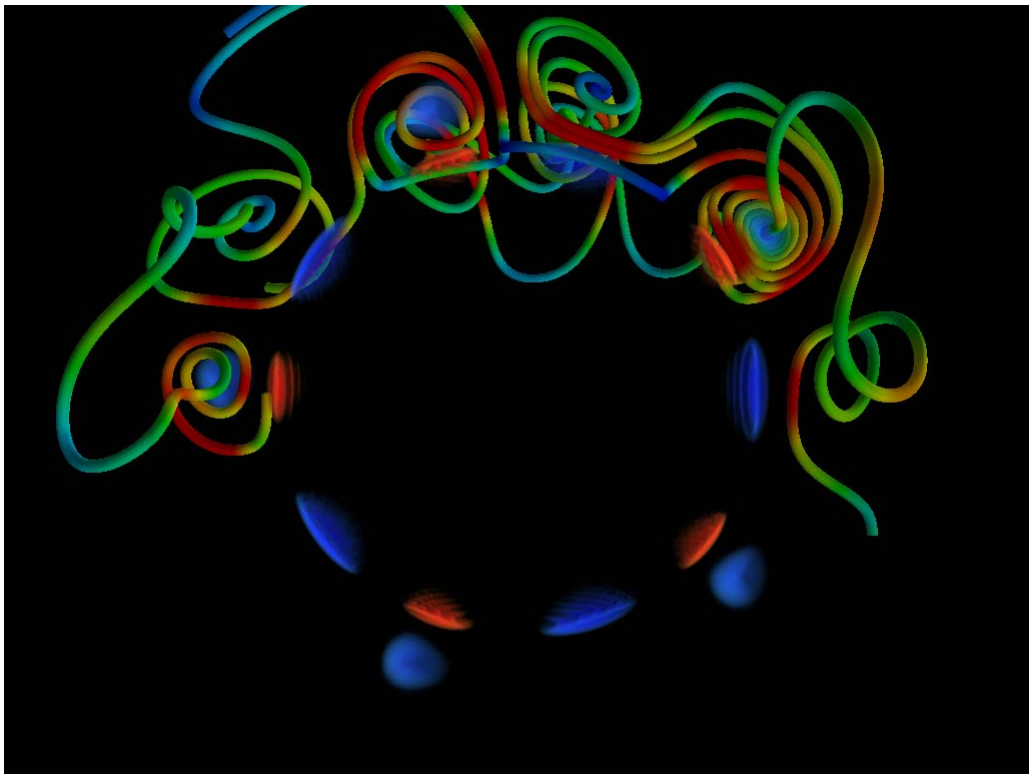


図 4. 速度場と渦度の z 成分

速度場を流線で可視化し，渦度の z 成分をボリュームレンダリングで可視化している。流線の色は，ベクトルの大きさを表している。

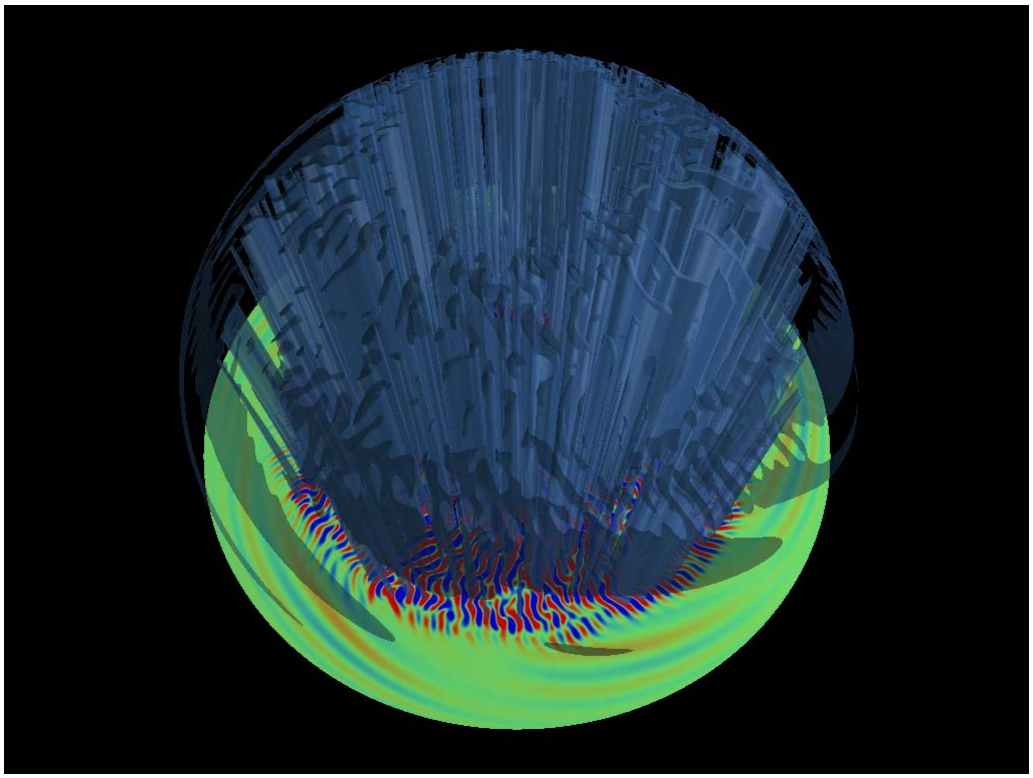


図 5. 渦度の z 成分の可視化

渦度の z 成分を赤道面のスライス，および半透明な等値面で可視化している。

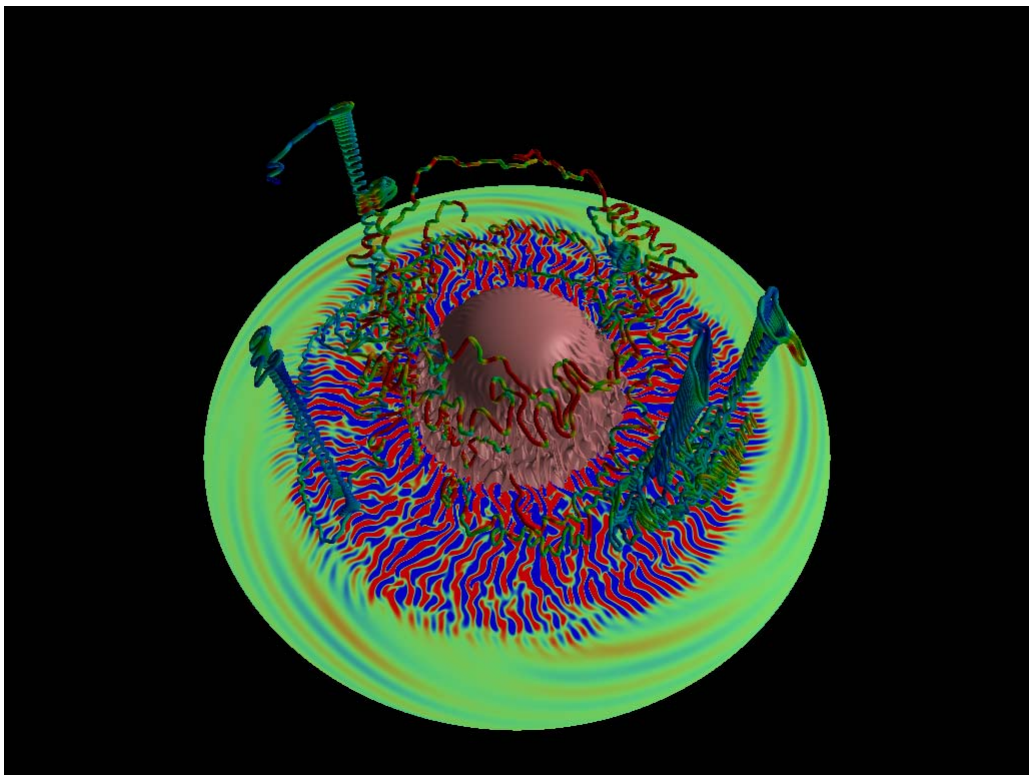


図 6. 温度，渦度の z 成分および速度場の可視化

温度を等値面，渦度の z 成分をスライス，速度場を流線で可視化している。

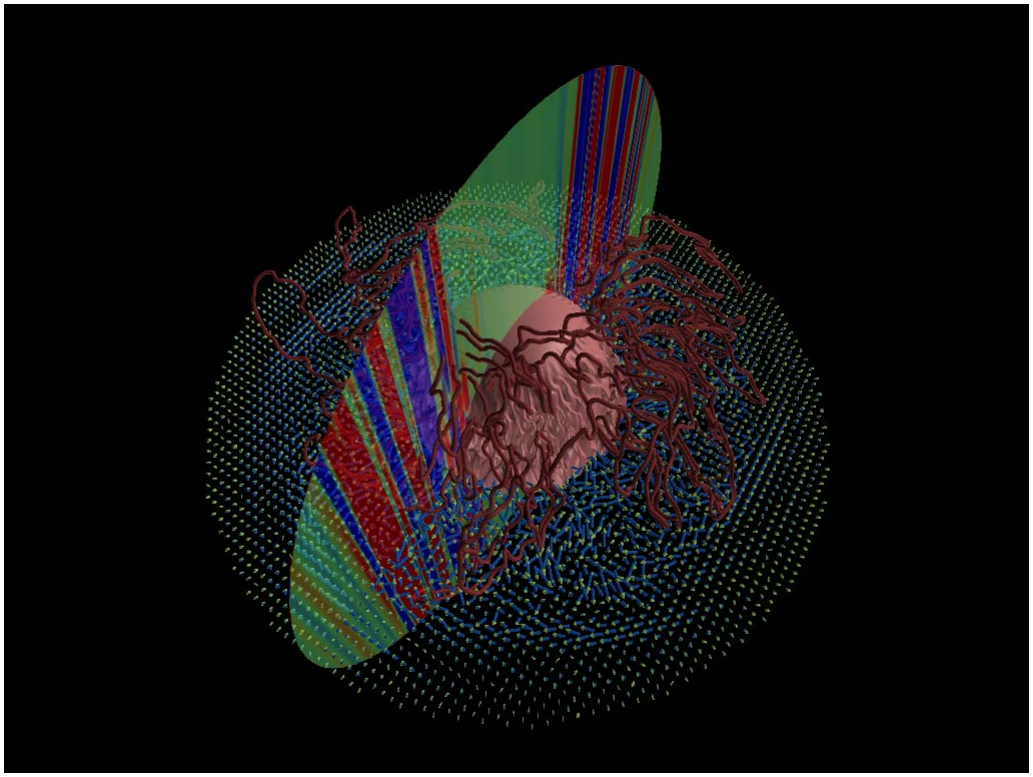


図 7. 温度，渦度の z 成分，速度場，磁場の可視化

渦度の z 成分に半透明なスライスを用いている。速度場を矢印，磁場を流線で可視化している。1 変数が約 3GB であるので，合計約 24GB のデータを可視化している。

7. まとめ

地球シミュレータ向けに開発・最適化した地球ダイナモシミュレーションコードを HA8000 システムに移植した。このコードは磁気流体力学（MHD）方程式を球ジオメトリ（インヤン格子）の下で解く流体系のコードである。ソースコードのわずかな変更で移植でき，2 次元領域分割による flat MPI 並列化の下，そのままのコードで最大 512 ノード，8192 コアまでの計算を問題なく行うことができた。

512 ノードでの計算性能は，1 ノード当たり 4 コアを使った場合，理論性能の 4.6%，1 ノードあたり 8 コアを使った場合は 4.0%であった。ベクトル計算機向けにチューニングされたコードを何もしないでそのままスカラー計算機で走らせたことを考えれば，512 ノード x 8 コア = 4096 コアの計算で 4%台のスピードが出たことは我々には驚くべきことのように感じた。しかしながら 512 ノード x 16 コアの計算では残念ながら 2.4%の性能しかでなかった。128 ノード x 16 コアの計算でも性能は 2.9%であったので，これは総コア数の問題ではなく，我々のコードがノード内の全てのコアを十分生かし切れていないことを意味する。いずれにせよ，この数字は悪すぎるように感じるので，何らかのチューニングが必要なのは明らかだが，我々の知識と技量では，今回の限られた時間内にこの性能を引き上げることはできなかった。

なお，今回移植した地球ダイナモコードのアルゴリズムと，その地球シミュレータ上での性能評価については，文献[4]に記述してある。

8. 謝辞

情報基盤センターの中島研吾教授には，HA8000システムの利用方法に関して様々な技術的アドバイスを頂きました。深く感謝します。本研究は科研費（17540404）と三菱財団助成金の補助を受けました。

参 考 文 献

1. A. Kageyama, T. Miyagoshi, and T. Sato, Formation of current coils in geodynamo simulations, Nature, vol. 454, pp. 1106-1109, 2008
2. 荻野竜樹，中尾真季，スカラー並列機を用いた高効率ＭＨＤコードの開発，名古屋大学情報連携基盤センターニュース，Vol.4，No.4，pp.284-299，2005
3. 陰山聡，大野暢亮，″固体地球シミュレーションの可視化″，可視化情報，Vol.28，No.110，pp.180-185，2008
4. A. Kageyama et al., A 15.2 TFlops Simulation of Geodynamo on the Earth Simulator, Proc. ACM/IEEE Conference SC2004 (Super Computing 2004), pp. 35-43, 2004

超並列計算によるマルチスケール・マルチフィジックス心臓シミュレーション

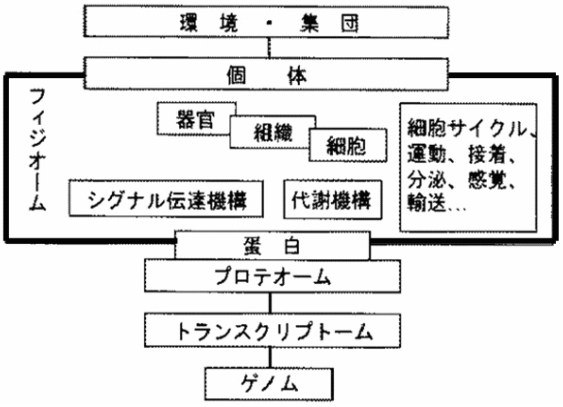
久田俊明

東京大学新領域創成科学研究科

１．はじめに

「ヒトゲノムプロジェクト」の完成をうけて生命科学の焦点は遺伝情報の解明からタンパク質および高次の生命単位の機能の解明へと移行しつつある。いわゆる「フィジオーム」とは，同時期に重要性が認識されたトランスクリプトーム，プロテオームなどの学問領域とならんでポストゲノムの医学・生物学の研究の中心的課題のひとつであると言われており，タンパク質から細胞内小器官，細胞，組織を経て臓器，そして個体までの各階層に属する膨大な数の機能モジュール間の相互作用として表現される複雑な生命現象を計算機シミュレーション(*in silico* 実験)によってモデル論的に再構成しようとする研究領域である(第１図)。そしてシミュレーションによって導かれた知見を *in vivo*, *in vitro* の いわゆる“wet”な実験によって検証するとともに，モデル，実験科学双方の欠落箇所を明らかにして補完し新たな発見に繋げる,というループの繰り返しにより全く新しい生命研究の領域が創成されると考えられている。

このような学術的背景に鑑み，本研究は，生命活動の中心をなす臓器である心臓について，細胞内のタンパク分子レベルの挙動から巨視的なポンプとしての血液拍出に至るマルチスケール・マルチフィジックス現象を可能な限り忠実に計算機上に再現し，フィジオームの概念を現実のものとしようとするものである。これにより，近年の分子生物学の進歩によって次々と明らかにされる膨大な情報の意味を明らかにし，医療や創薬に役立てることが出来ると考えられる。

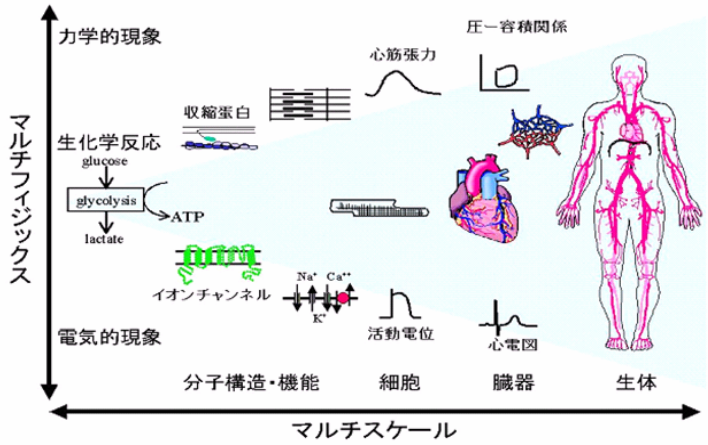


第１図　フィジオームと他の領域との階層関係

(Hunter&Borg, Nature Review 誌 2003 を梶谷文彦博士が改編)

心臓の活動の根元をなすのは純粋な生化学反応であるが，電気化学・力学の諸現象に広く派生するマルチフィジックス問題を構成する。また空間尺度としては，タンパク分子（ー数 10nm）

から細胞(ー100μm)，組織(ーmm)，臓器(ーcm)を経て血液拍出に至るマルチスケール問題を構成している（第２図)。我々がベッドサイドで計測する心電図や血圧などのマクロ現象については古くから多くの医学・生理学的研究がなされてきたが，一方で近年，分子生物学の進歩によって遺伝子，分子レベルの事象に関する膨大な知見が集積されるに至っている。しかし現状では，マクロ現象とミクロ現象の因果関係はその間に大きなスケールの差と複雑な相互作用によって混沌とし，専門家にとっても個々の知見を有機的に活用して理解し予測することが困難となっている。従って第２図のようなマルチスケール･マルチフィジックス問題を正面から取り扱うことの出来る計算科学の実現が医学・医療の現場から要請されている。



第２図　心臓におけるマルチスケール・マルチフィジックス現象

しかし従来の生体シミュレーションにおいては，細胞を一つの化学反応プールとして各種の代謝等の反応速度式をシステム的に構築する細胞シミュレーション，あるいは自動車衝突の数値ダミーに極論されるような巨視的な力学現象を解明するバイオメカニクス解析，のいわば両極が主に研究されてきた。そこで私たちのチームでは次のようなアプローチによりミクロからマクロまでの多階層の生命現象を統合した仮想のヒト心臓を再構成することを試みて来た。

（１）第２図に示されるように，細胞膜のイオンチャンネル，ポンプ，トランスポーターなどの作用に基づく細胞内イオン環境のダイナミックスを記述する電気生理学モデルやイオンによって制御される細胞内収縮タンパクであるアクチン・ミオシン間のクロスブリッジ運動を記述する興奮収縮連関モデルから出発し，組織，臓器を経て最終的な血液拍出に至る心臓機能の全過程を各レベルにおける要素間の相互作用を含めて合理的に数理化しシミュレートする。

（２）その際現れる，大規模な電気･化学･力学現象のマルチフィジックス問題を，流体構造連成解析などの有限要素法をベースとする最先端の計算科学手法により正面からシミュレートする。これによって根拠のない省略化，仮定を排除し生命科学からの疑問に厳密に解答を与えられるモデルを目指す。

（３）形ある細胞モデル（有限要素法により内部構造を再現した３次元数値細胞）を世界に先駆けて開発し，これを経由してミクロ現象とマクロ現象をシームレスにつなぐマルチスケールシミュレーションを達成する。

先ず上記（１），（２）に対して，ヒト心臓を心筋組織片レベルの構成則（応力・歪関係式）をベースに有限要素法によってモデル化した。以下ではこれを**マクロ構成則に基づく心臓シミュレータ**と呼ぶ。当シミュレータは，ＣＴから得られた断層データを３次元構成した全心臓(whole heart)モデルを使い，

(a) 細胞電気生理学に基づく細胞の興奮とその伝播

(b)心筋の興奮による心臓の力学的収縮運動

(c)内腔血液の流動と拍出といった

三つの物理現象を連成させて模擬できるようになっている。

(a) においては，細胞電気生理モデルとして国際的に広く受け入れられている Luo-Rudy model 並びに Noble model を選択できるようになっている。また細胞相互の接続については，細胞内同士のみが接続される mono-domain model 並びに細胞外の接続も加えた bi-domain model をの選択が可能であり，興奮伝播が精度良く模擬できるよう格子間隔が約 0.4mm のボクセル有限要素を用いる。細胞間の伝導率テンソルや刺激伝導系は解剖学的知見に基づき空間的に分布させる。各臓器や組織のセグメンテーションを行い，それぞれの伝導率を与えた胸郭モデルも別途準備して，これに心臓ボクセル有限要素モデルを埋め込むことにより，心電図を再現できた。

(b)においては(a)の計算結果である各細胞内のカルシウムイオン濃度を興奮収縮連関モデルに与え，アクチン・ミオシン間のクロスブリッジ生成を記述する時間発展方程式を解き収縮力を評価する。興奮収縮連関モデルは Negroni により提唱された４状態モデルを用いて，更にクロスブリッジ生成における協調性の効果を導入した独自のモデルを用いている。収縮力は心筋を離散化した各有限要素に与え，心室・心房の収縮運動を実現している。心筋の受動的構成式には線維方向と約４細胞毎に存在する cleavage plane を考慮した直交異方超弾性構成則を用いている。有限要素には 5/4c テトラ要素を，粘性は歪速度比例型を用いる。

(c)については，心臓壁すなわち構造の非線形運動方程式と，血液すなわち流体の Navier-Stokes(NS)方程式を強連成させた一体型の解法を用いる[1]。ただし血液領域境界は心臓壁の運動に伴って移動･変形するため NS 方程式は ALE(Arbitrary Lagrangian-Eulerian)表記し，流体 5/4c テトラ要素の節点座標は別途擬似弾性体の方程式を解くことにより制御する。流体解析には SUPG 安定化手法等を用いる。強連成解法をとることで安定性と収束性が確保される。心臓の前後には体循環や肺静脈などを模擬する回路モデルを接続し連成させる。なお有限要素解析で重要となる連立１次方程式のソルバとしては，電気現象解析用には並列マルチグリッドソルバ[2]の開発，力学現象解析用には並列 ILU 前処理付き反復ソルバ[3]を開発した。以上の「マクロ構成則に基づく心臓シミュレータ」の拍動シミュレーション結果は生理学的見地から定性的にも定量的にも妥当であり正常心を再現できていることを検証した。またイヌ心臓も別途モデル化し，電気的，力学的な諸量を比較した。

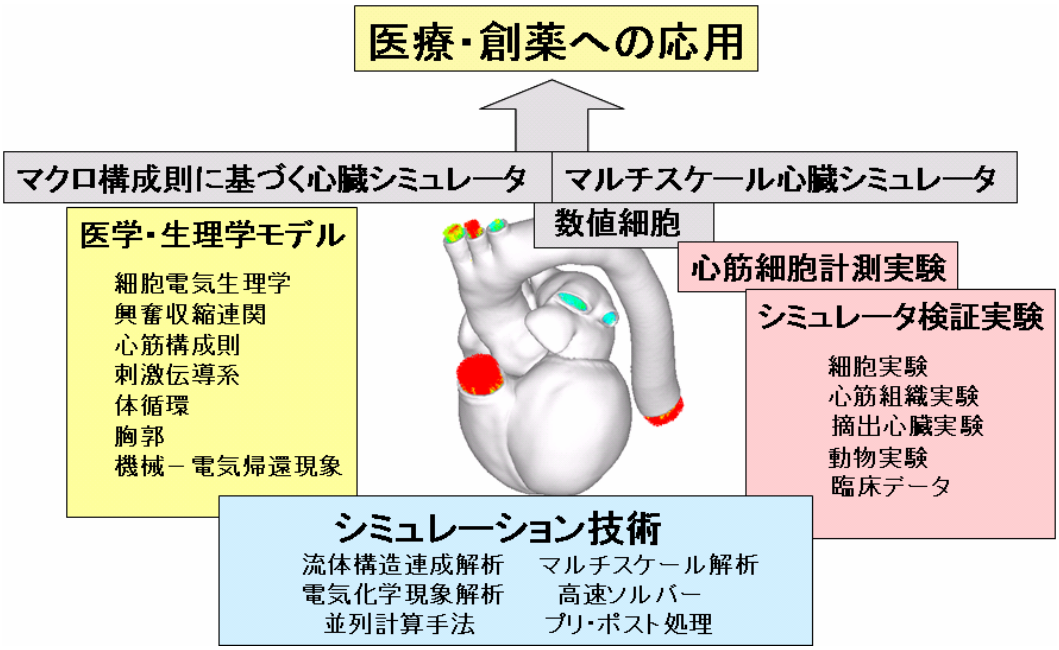
次に（３）に対しては次のような過程を経て**マルチスケール心臓シミュレータ**の開発を行った。先ず細胞の微細構造情報を得るため，表面形状について高い解像度をもち剛性情報も得られる原子間力顕微鏡と内部構造をタンパク特異的に解析できる共焦点顕微鏡の両者の利点を生かし同時観察および画像重ね合せを行うことによって細胞の主要な要素の３次元構造をサブミクロンの解像度で再構成した。そして，これに基づいて数値細胞モデルを開発した。即ち，主要なコンポーネントである細胞骨格，細胞膜，細胞質，筋原線維，筋小胞体，Ｚ帯などの機構

や相互作用を整理し，それぞれ適切な有限要素を用いてモデル化した。これにより細胞内の筋小胞体からのカルシウムイオンの放出・拡散に伴う筋原線維収縮を駆動力とする細胞収縮が３次元的効果を合理的に含んだ形で実現できた。また，生きたままの細胞を観察できるという当チームの実験手法の利点を活かし，数値細胞モデルの妥当性を検証した。このような数値心筋細胞の開発の試みは過去に前例がないが，既に米国生理学会誌に掲載されるに至っている。一方，心筋において細胞は局所的に見れば凡そ規則的に配置していることから，均質化法と呼ばれるマルチスケール解析手法の適用が原理的には可能である。すなわち，一つの有限要素内ではある定義された複数の細胞の組み合わせ（ミクロユニット）が無数に周期的に配置していると仮定し，その有限要素の力学的特性を数理的に導くことが出来る。しかしこの手法は非線形問題においては実行不可能な程に膨大な計算量となるため，現実にはほとんど用いられて来なかった。そこで当研究チームでは種々の数理的検討を重ねた結果，精度を確保しつつ大幅に計算量を削減できる手法を考案した[4]。

以上に基づき前記の**マクロ構成則に基づく心臓シミュレータ**におけるマクロ構成則部分を数値細胞モデルに基づく非線形均質化法で置き換えた。さらに本理論を大規模並列計算機上で効率的に実行できるようプログラム開発を行い，IBM JS22(Power6 Blade 型サーバー)240 コア上でシミュレーションを実施し良好な結果を確認してきた。最終的には神戸に建造中の我が国の次世代スーパーコンピュータを用いて数万から数十万コアの超並列計算を行い医学的に意味のある結果を得ようとしているが，その前段階として，本 HPC 特別研究プロジェクトでは 6000 コアを超える並列計算により自由度を大幅に減じた両心室モデルと細胞モデルを用いて試験的にマルチスケールシミュレーションを行った。

２．*UT-Heart* の概要

UT-Heart は前項に述べた**マクロ構成則に基づく心臓シミュレータ**と**マルチスケール心臓シミュレータ**の二つのシミュレータからなる。開発に必要な研究課題を含め第３図に *UT-Heart* の概念を示す。



第３図　*UT-Heart* を構成する二つのシミュレータとその開発に必要な研究課題

二つのシミュレータの計算負荷は大幅に異なるが，後者は前者をベースにしているので，先ず「マクロ構成則に基づく心臓シミュレータ」について概要を紹介する。

マクロ構成則に基づく心臓シミュレータ

マクロ構成則に基づく心臓シミュレータはヒト心臓を心筋組織片レベルでの構成則（応力・歪関係式）をベースに有限要素法によってモデル化したものである。心臓モデルは，第４図上段に示されるように，ＣＴから得られた断層データを３次元構成し，左右両心室・両心房，大動脈から成る全心臓(whole heart)モデルである。このシミュレータでは

(a)細胞電気生理学に基づく細胞の興奮とその伝播

(b)心筋の興奮による心臓の力学的収縮運動

(c)内腔血液の流動と拍出

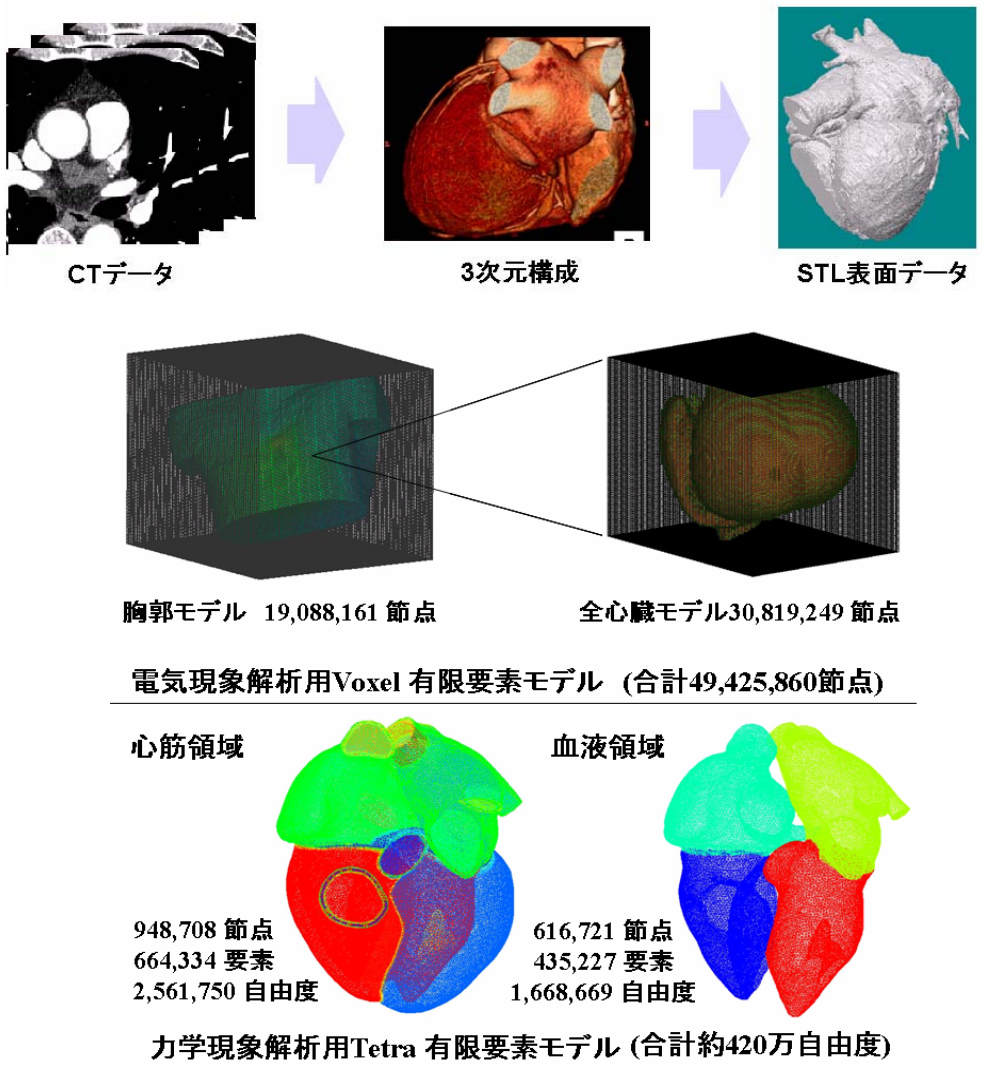
の三つの物理現象を連成させて模擬できるよう構成してある。

(a)における心室の細胞電気生理モデルとしては，国際的に広く受け入れられている Luo-Rudy model 並びに Noble model が選択できるようになっている。一方，心房にはその細胞電気生理モデルとして知られる Nattle モデルを用いている。また細胞相互の接続については，ギャップジャンクションを通じて細胞内同士が接続される mono-domain model 並びに細胞外の接続も加えた bi-domain model が選択できるようになっている。また細胞から細胞への興奮伝播が精度良く模擬できるよう，第４図中段に示すように格子間隔が 0.44mm のボクセル有限要素を用いる。総節点数は 30,819,249 節点である。また第５図（左）に示すように心筋には特徴的な線維方向（細胞長軸の方向）があるため，それを考慮して細胞間の伝導率テンソルは直交異方とし，生理学的知見に基づき空間的に分布させる。線維走向は左心室では心室壁内側(endo)から外側(epi)に向かって 90°（心室長軸方向）から-60°に捩れるように変化する。さらに刺激伝導系（Purkinje 線維）ネットワークもモデル化する（第５図（右））。このネットワークは Purkinje 線維細胞モデルとして知られる DiFrancesco-Noble モデルを直列に細胞内外のバイドメインで接続した生理学的根拠のあるものであり，ヒス束以降がモデル化されている。この刺激伝導系を通じて心筋細胞が刺激され心臓に興奮伝播が引き起こされる。またその結果微弱な電流が体表面まで伝わり心電図として計測される。なお心電図を再現するため，各臓器や組織をセグメンテーションしそれぞれの伝導率を与えた胸郭モデルも別途準備し，心臓ボクセル有限要素モデルを埋め込む（第６図)。この胸郭モデルは 19,088,161 節点のボクセル有限要素（1 ボクセルのサイズは 1.76㎜）によってモデル化されている。従って，心臓を埋め込んだボクセルモデルとの合計では約 4900 万節点となる。

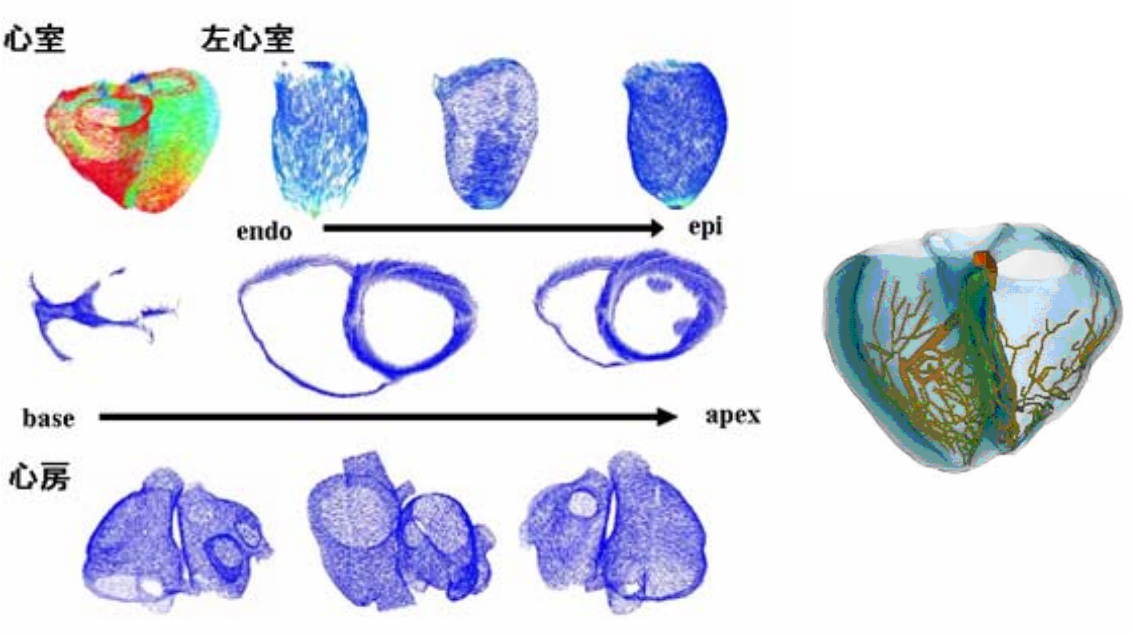
(b)においては(a)によって計算される各細胞内のカルシウムイオン濃度を興奮収縮連関（E-C Coupling）モデルに与え，アクチン・ミオシン間のクロスブリッジ生成を記述する時間発展方程式を解くことにより収縮力を評価する。この興奮収縮連関モデルはとしては，Negroni により提唱された Ca イオンとトロポニンの結合の有無，クロスブリッジ形成の有無により定義される４状態モデルに，クロスブリッジ生成における協調性の効果を導入した独自のモデルを用いる。収縮力は心筋を離散化した各有限要素の積分点に内力として与えることにより心室，心房の収縮運動を実現する。心筋の受動的構成式（応力・歪関係式）には線維方向と約４細胞毎に存在する cleavage plane を考慮した直交異方超弾性構成則を用いる。有限要素には微圧

縮性を考慮した混合型の 5/4c テトラ要素を，粘性は歪速度比例型を用いる。

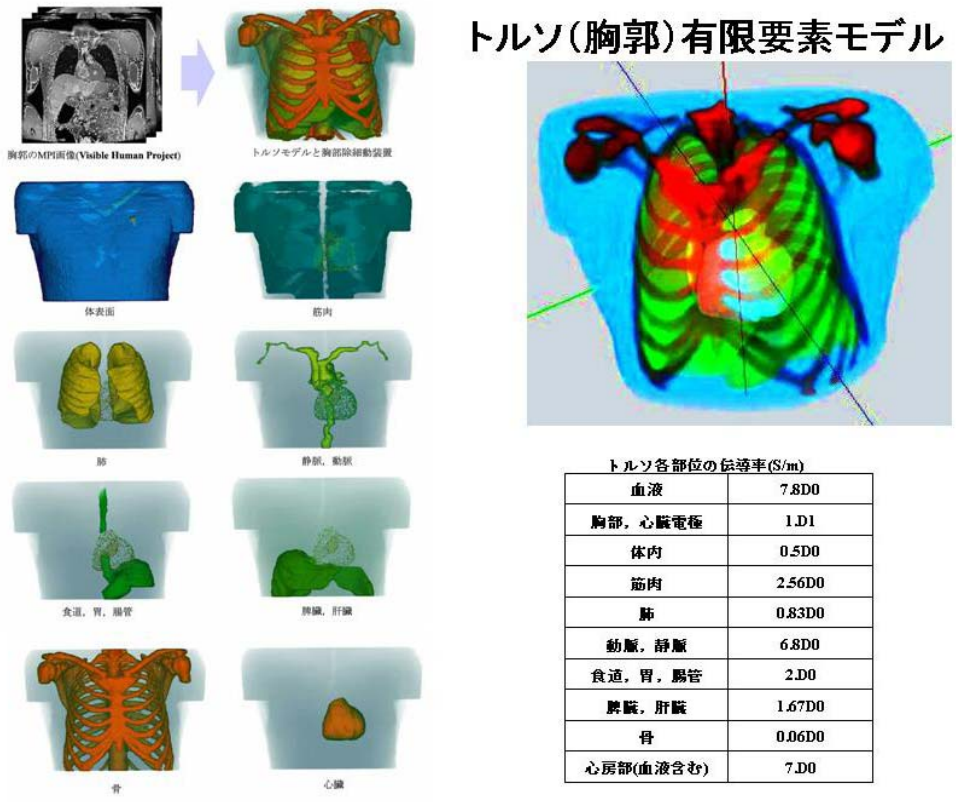
(c)については，心臓壁（構造）の非線形有限要素運動方程式と血液（流体）の Navier-Stokes(NS)有限要素方程式を強連成させた一体型の解法を用いる。ただし血液領域境界は心臓壁の運動に伴って大きく移動・変形するため NS 方程式は ALE(Arbitrary Lagrangian-Eulerian)表記し，流体領域を構成する 5/4c テトラ要素の節点座標は別途擬似弾性体の方程式を解くことにより制御する。流体解析には SUPG 安定化手法を用いる。強連成解法をとることで安定性と収束性が確保される。心筋部(両心室,両心房)は 948,708 節点,664,334 要素，2,561,750 自由度，また血液部（両心室内腔，両心房内腔）は 616,721 節点，435,227 要素，1,668,669 自由度によって構成される。合計の自由度は約 420 万自由度となる（第４図下段)。心臓の前後には体循環や肺静脈などを模擬する７つの回路モデル（windkessel model）を接続する（第７図)。この回路モデルと心臓モデルはやはり強連成させ一つの方程式系として解く。なお僧帽弁（第８図左）や大動脈弁などの心臓弁は境界面捕捉型の手法を使って連成させる。



第４図　CT データから構成した心臓モデル及び電気現象・力学現象解析用有限要素モデル



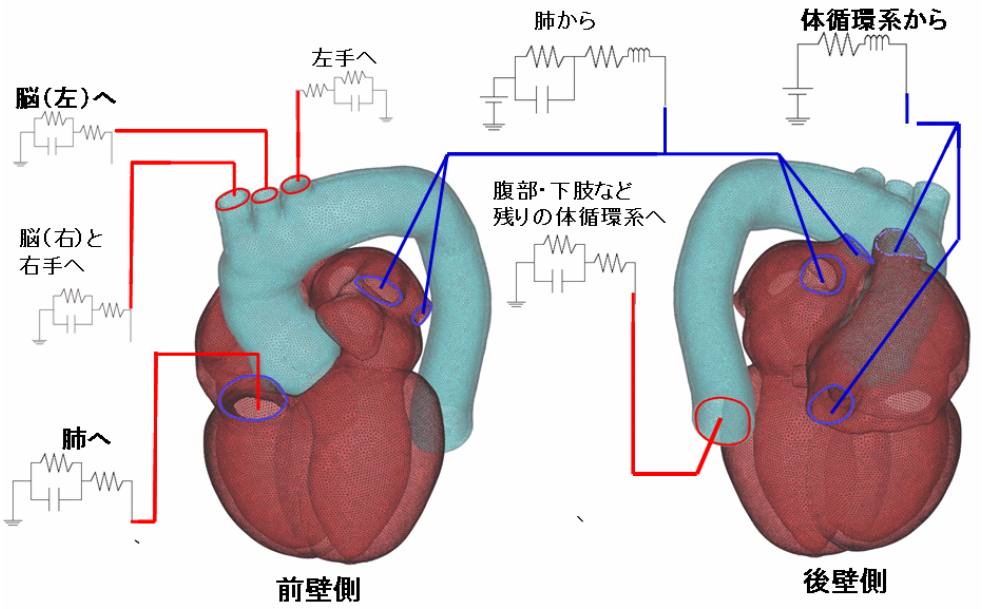
第５図　心臓シミュレータに埋め込まれた線維（細胞長軸方向）分布と
刺激伝導系（Purkinje 線維）の有限要素ネットワークモデル



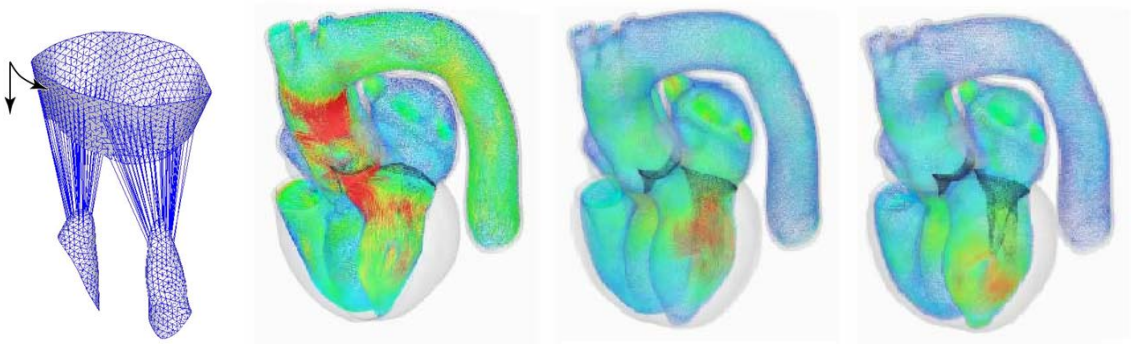
トルソ各部位の伝導率(S/m)

血液	7.8D0
胸部，心臓電極	1.D1
体内	0.5D0
筋肉	2.56D0
肺	0.83D0
動脈，静脈	6.8D0
食道，胃，腸管	2.D0
脾臓，肝臓	1.67D0
骨	0.06D0
心房部(血液含む)	7.D0

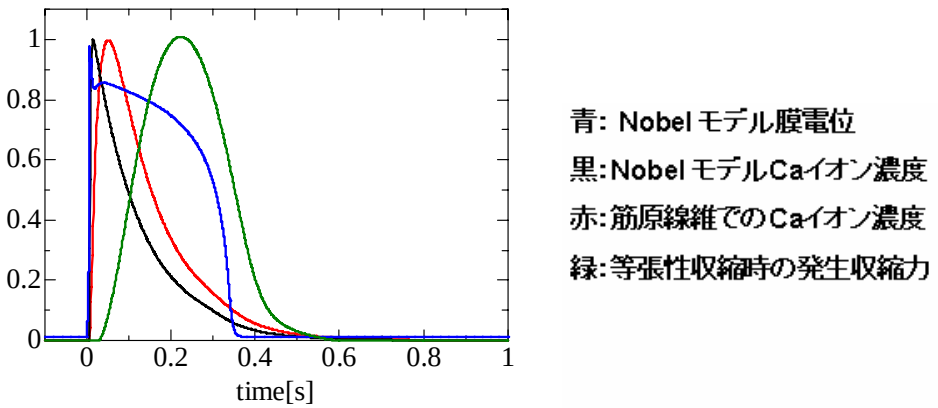
第６図　組織・臓器ごとに伝導率が区別されたトルソ（胸郭）有限要素モデル



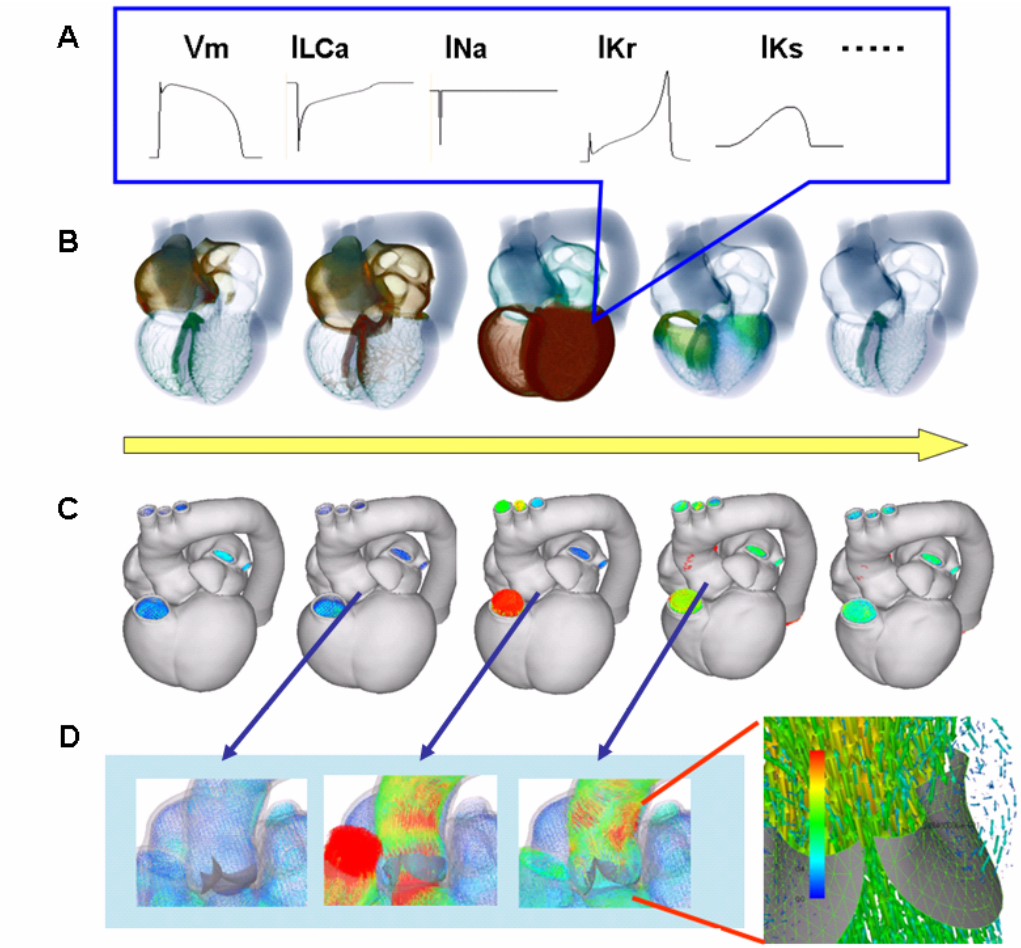
第７図　体循環モデル(7 つの windkessel モデル)と接続された心臓シミュレータ



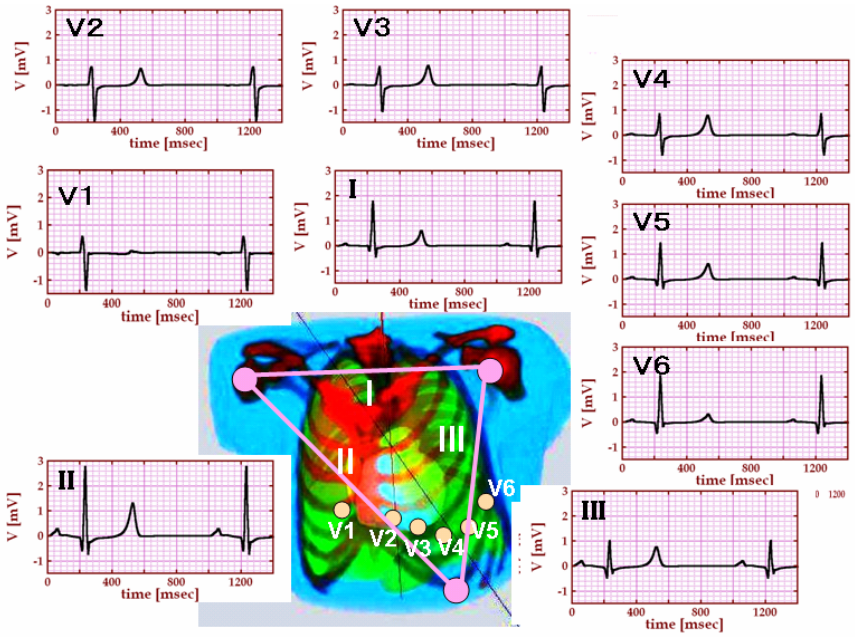
第８図　シェル要素とケーブル要素による僧帽弁モデル ならびに 僧帽弁・大動脈弁と血流，心臓壁との連成解析



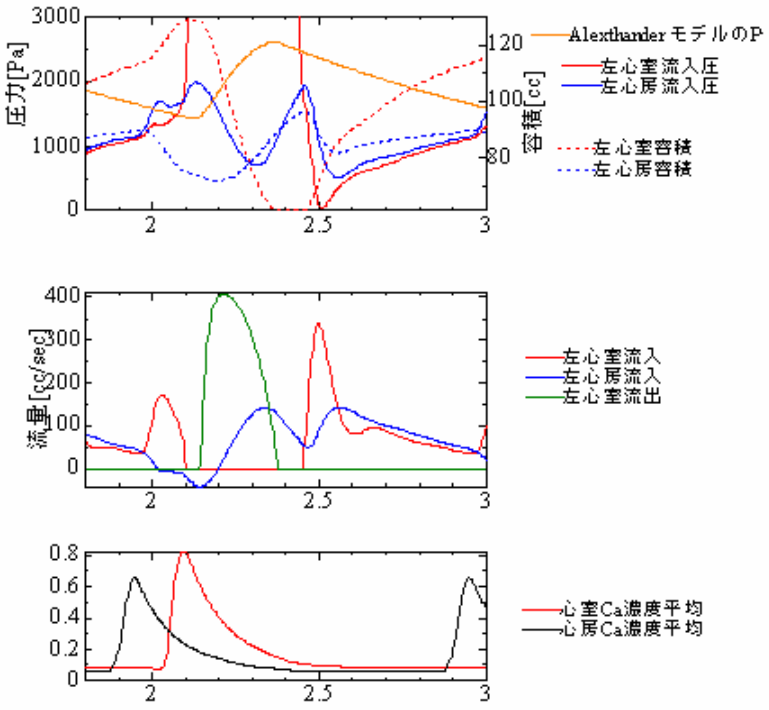
第９図　単一心筋細胞における正規化された膜電位，Ca イオン濃度，収縮力の推移



第 10 図　正常心臓の興奮伝播シミュレーションならびに拍動シミュレーション



第 11 図　心臓の電気的興奮伝播によって体表面で観察される心電図



第 12 図　正常心における心室，心房の圧，容積，流量関係，心室・心房 Ca 濃度平均の推移

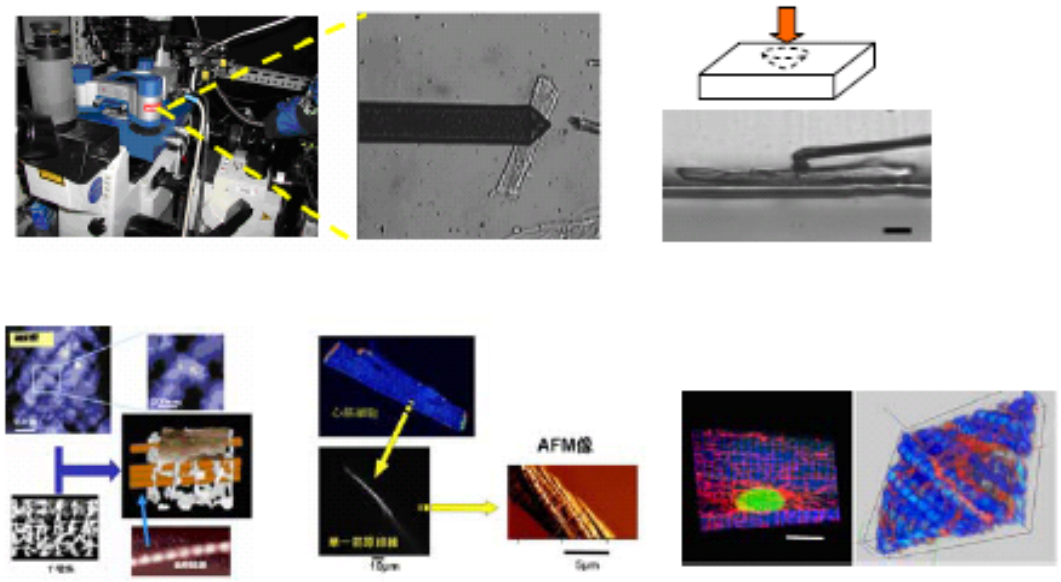
第 9 図に単一の細胞モデルにおける膜電位，Ca イオン濃度，収縮力の関係を示す。また，第 10 図にマクロ構成則に基づく心臓シミュレータの拍動解析結果を示す。図中 A は心室のある箇所における細胞膜電位(Vm)と各種イオン電流(ILCa:L 型カルシウムイオン電流，INa:ナトリウムイオン電流，IKr:r 型カリウム電流，IKs:s 型カリウム電流)の推移，B は心臓における興奮伝播の様子，C は拍動と血液の拍出，D は大動脈弁付近での流れ場を示す。大動脈基部にはバルサルバ洞と呼ばれる膨らみと大動脈弁があるが，バルサルバ洞では渦が観測され弁の開閉に寄与していることが分かる。僧帽弁での流れも妥当な値を示している（第 8 図右 3 葉)。体表面では第 11 図に示されるような心電図が計測される。また第 12 図には心臓各所の流量履歴を示す。これらはいずれも生理学的見地から定性的，定量的に妥当であり正常心を再現されている。

マクロ構成則に基づく心臓シミュレータは，基本的には実用レベルに到達しており，既に我が国初の植え込み型除細動装置（ICD）の開発における電極の最適化に用いられ，世界市場を独占する欧米製品の性能を大幅に上回る設計に成功した。また，その性能はイヌを使った実験で定量的に検証され，今後の貢献が期待されている。心臓外科領域においては，梗塞心の Dor 手術における切除領域最適化，マルファン症候群における大動脈基部置換術再現などの具体例により，術前のシミュレーションによって最適な手術計画を立てることが可能なレベルに到達している。内科的治療についても同様であり，合理的診断と最適な処方に心臓シミュレータを活用することが可能である。医療現場への供用は，今後急速に進むと見込まれる医療制度の改革によって，現実のものになると考えられる。創薬については，突然死に結びつく先天性 QT 延長症候群を再現し，L 型 Ca チャンネルブロッカーや Na チャンネルブロッカーを投与したシミュレーションを行って VT（頻脈）発生頻度を調べた。また併用効果についても明らかにした。こ

のような薬物実験をイオンチャンネルレベルから臓器レベルまでの範囲で容易に検証できるのは本シミュレータのみであり，動物実験を代替し新薬の開発を加速することが可能となる。

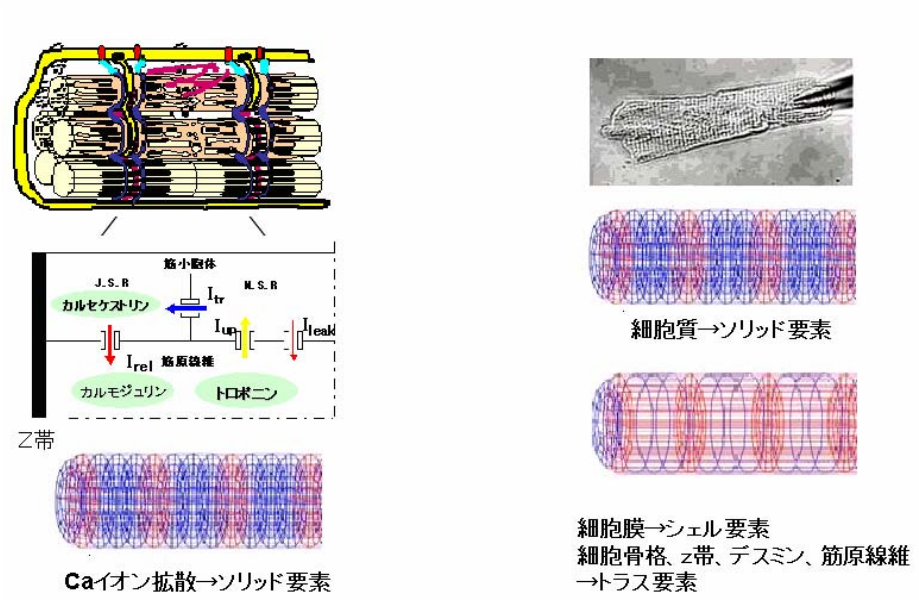
マルチスケール心臓シミュレータ

当研究チームでは細胞の微細構造情報を得るため，第 13 図に示すように，表面形状について高い解像度をもち剛性情報も得られる原子間力顕微鏡と内部構造をタンパク特異的に解析できる共焦点顕微鏡の両者の利点を生かし同時観察および画像重ね合せを行うことによって細胞の主要な要素の 3 次元構造をサブミクロンの解像度で再構成した。

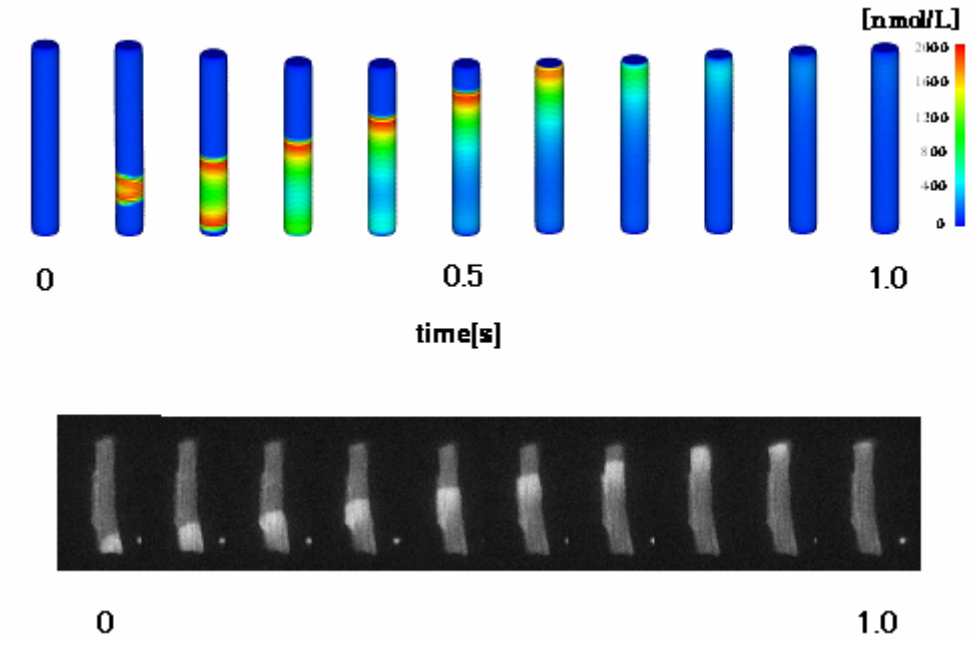


第 13 図　　原子間力顕微鏡と共焦点顕微鏡を組み合わせた細胞構造計測実験

そして，これから得られた各種のデータや知見に基づいて数値細胞モデルを開発した。即ち，第 14 図に示すように，主要なコンポーネントである細胞骨格，細胞膜，細胞質，筋原線維，筋小胞体，Ｚ帯などの機構や相互関係関係を整理し，それぞれ適切な有限要素を用いてモデル化した。これにより細胞内の筋小胞体からのカルシウムイオンの放出・拡散に伴う筋原線維収縮を駆動力とする細胞収縮が 3 次元的効果を合理的に含んだ形で実現できた。第 15 図上段は傷害された心筋細胞で起こりかつ不整脈発生との関係が注目されているカルシウムウエーブを再現したものである。下段は単離後約半日経過した心筋細胞で実験的に観察されたカルシウムウエーブである。このように生きたままの細胞を観察できるという当チームの実験手法の利点を活かし，数値細胞モデルの妥当性を検証した。以上のような数値心筋細胞の開発の試みは過去に前例がないが，既に米国生理学会誌に掲載されるに至っている[5]。



第 14 図　実際の心筋細胞の内部微細構造に従って興奮―収縮連関モデルを配置した数値心筋細胞モデル

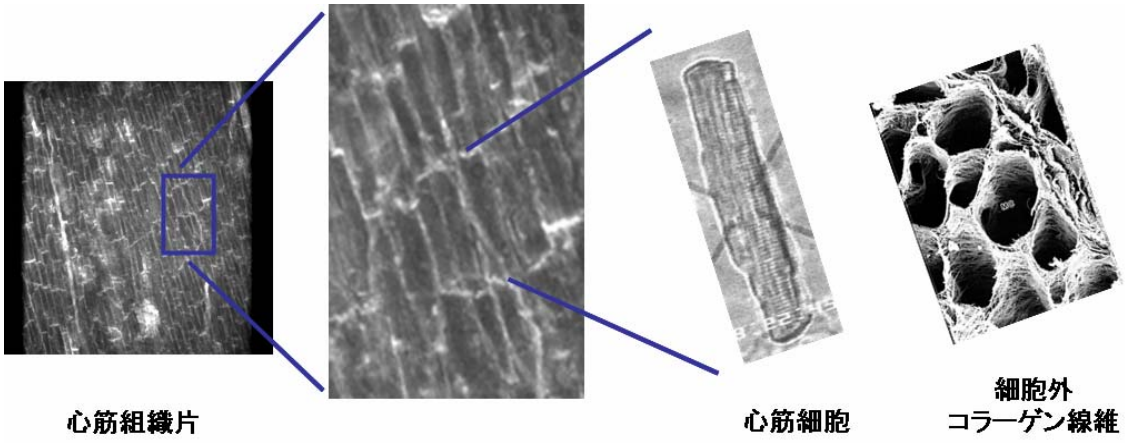


第 15 図　カルシウムウエーブ

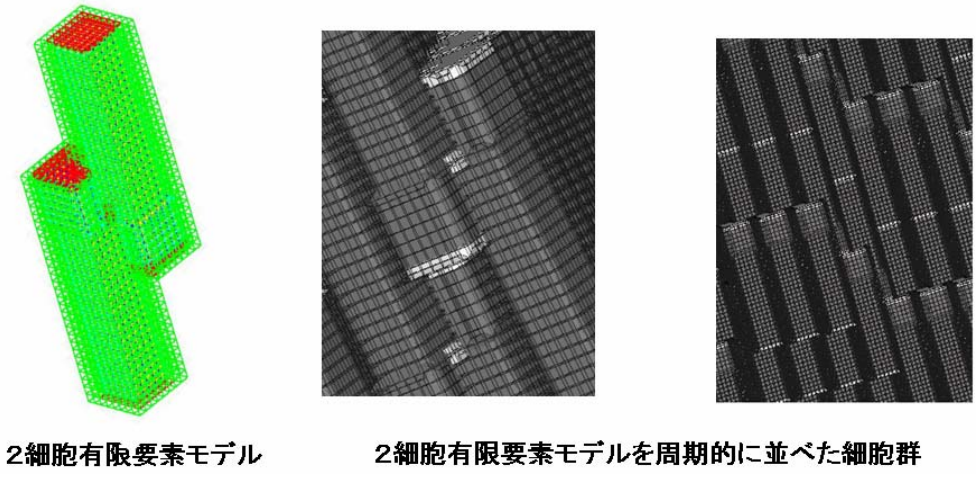
上段：シミュレーション　細胞内カルシウム濃度をカラーコードで示す。下段：実験　色の変化はカルシウム指示薬の発光を示す。

一方，第 16 図の顕微鏡写真に示されるように，心筋において細胞は局所的に見れば凡そ規則的に配置している。このような周期的配列は，均質化法（homogenization method）と呼ばれるマルチスケール解析手法の適用を可能とする。即ち心臓を構成する各有限要素内では，第 17 図に示すように，少数の数値細胞からなるミクロ構造が無限に周期的に配置していると仮定す

ることにより，その有限要素の力学的特性を数理的に導くことが出来る。この図では実際の配列を模擬した二つの数値細胞を周期的に接続した細胞群を表す。一般に細胞数を増せばより複雑な周期構造を表現できる。緑の領域は細胞外コラーゲン組織を表し，その内部には前記の筋原線維などで構成される細胞がモデル化されている。これにより心臓⇔数値心筋細胞の架橋，即ちマルチスケール解析を行うことが原理的には可能となる。しかし均質化法はこれまで主に線形問題に対して研究開発が行われてきており，非線形問題においては実行不可能な程に膨大な計算量となるため，現実の問題にはほとんど用いられて来なかった。そこで当研究チームでは種々の検討を重ねた結果「高精度低負荷均質化アルゴリズム」を新たに考案し，精度を確保しつつ大幅に計算量を削減できることを示した[4]。



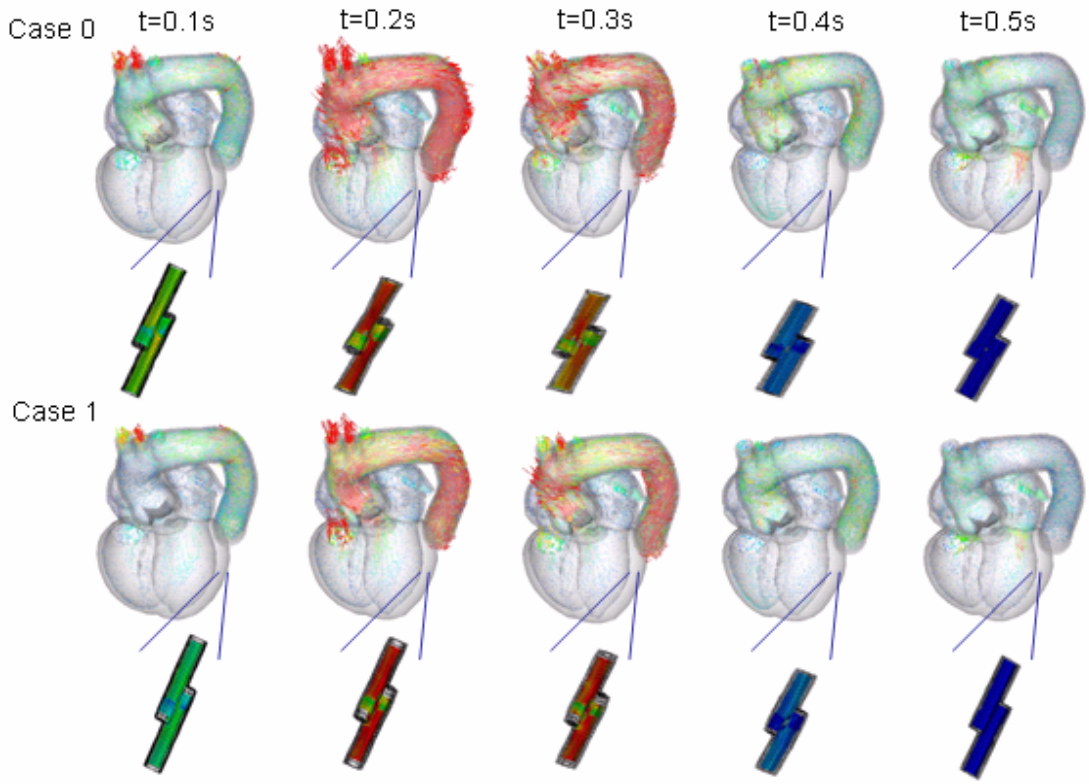
第 16 図　心筋組織片の顕微鏡写真と単離した細胞および細胞外のコラーゲン線維



第 17 図　二つの数値細胞が周期的に接続された細胞群

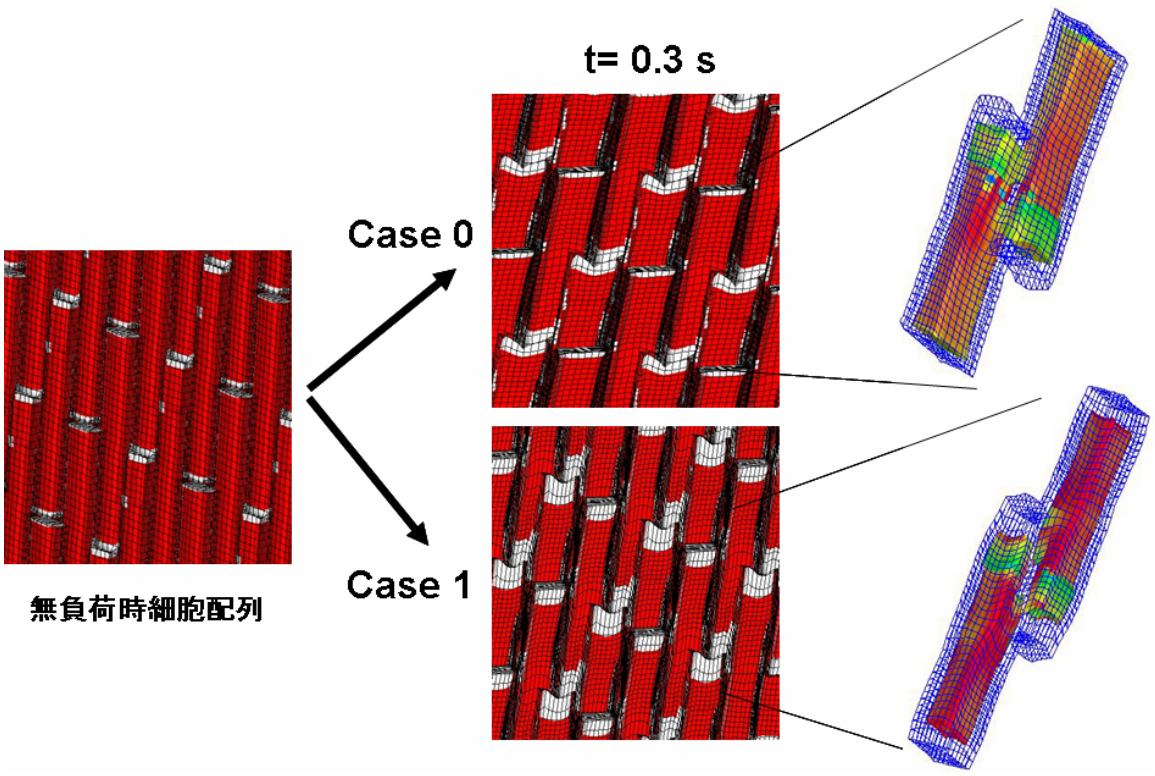
当チームでは以上の枠組に基づくマルチスケールシミュレーションを大規模並列計算機上で効率的に実行できるようプログラム開発を行い，研究室所有の IBM JS22(Power6 Blade 型サーバー)240 コアを用いて試計算を行ってきた。ここでは計算時間やメモリ容量の観点から自由度を大幅に削減したモデルを用いてはいるが，マルチスケールシミュレーションの意義を示す計算結果の一例を紹介する。

第 17 図左の有限要素モデルにおいて赤く示した箇所は細胞と細胞を繋ぐ介在板（gap junction）を現す。ここに存在する分子 integrin は細胞間をつなぐ接着タンパクであり，vinculin を含むタンパク複合体を介して収縮装置と細胞膜をアンカーすることが知られている。vinculin 遺伝子をノックアウトしたマウスの実験では 49%が突然死し，また突然死を免れた場合も拡張型心筋症を発症するが，因果関係については良く分かっていない。そこで vinculin 遺伝子のノックアウトを模擬して介在板での有限要素の物性値を大幅に低下させたマルチスケールシミュレーションを実施した。第 18 図は正常な場合と介在板の剛性を低下させた場合について心臓の拍動と左心室内膜側のある要素での細胞の運動を比較したものである。また第 19 図は t ＝0.3 s での細胞群の変形を両ケースについて等位置・等方向から観察したものである。同図右では２細胞の収縮力をカラー表示し比較している。筋原線維の収縮機能が同一であるにも拘わらず変形パターンに特徴的な違いが見られる。第 20 図は左心室圧と容積の１心周期における変化を比較したものである。興味深いことに介在板異常の場合，圧力では正常の 10%程度以下しか低下しないが，主要な指標の一つである拍出率(Ejection Fraction)で見ると正常の 54%から 34%まで低下することが分かる。このように，ミクロの変化がマクロにどう現れるかを定量的に評価でき，またここでは割愛するが原因を詳しく分析することにより医学的・生理学的に有用な知見を得ることが可能である。更に心臓の運動との相互作用の結果定まる細胞への詳細な負荷を知り，リモデリングの予測を行うことも出来る。本解析例では，ミクロモデル，マクロモデル共に分解能が十分ではないため議論には限界があるが，精密なマルチスケールシミュレーションを行うことの出来る計算機が利用可能になれば医療や創薬に新たな局面が展開されるものと期待される。



第 18 図　マルチスケールシミュレーションによる心臓の拍動

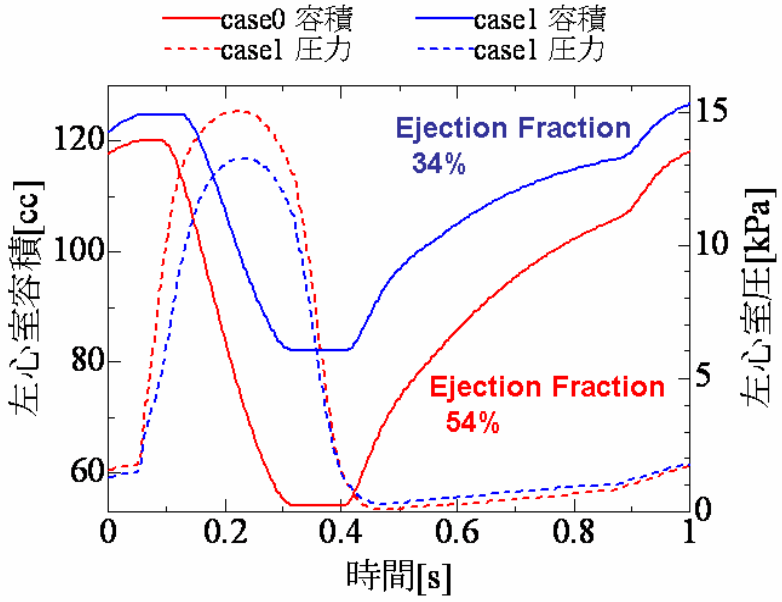
（上段：正常，下段：介在板接着タンパク喪失）



第 19 図　等位置・等方向から観察した t ＝0.3 s での細胞群の変形

(Case 0:正常, Case 1:介在板接着タンパク喪失)

2 細胞のカラーは筋原線維表面での収縮力を表示したもの



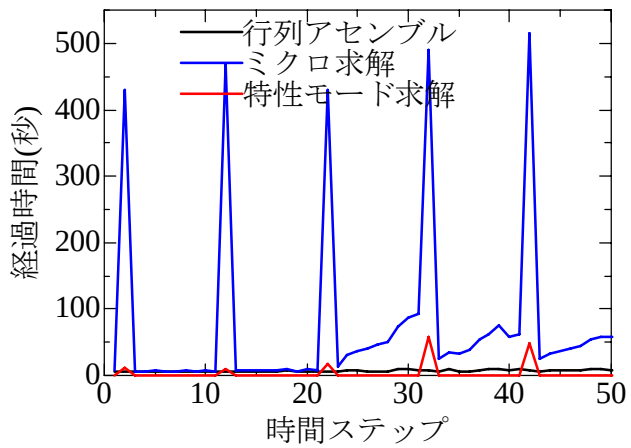
第 20 図　　マルチスケール解析から得られた左心室圧・容積の変化

(上段：正常, 下段：介在板接着タンパク喪失)

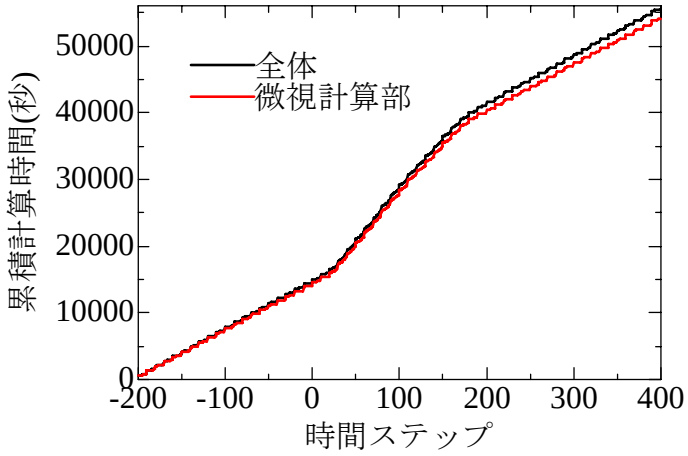
３．Ｔ２Ｋによるマルチスケールシミュレーション

前項の IBM Power6 を用いたシミュレーションでは 240 コアと小規模の並列計算機においてマルチスケール解析を行うため，有限要素心臓モデルに対して定義するミクロモデルの総数は数千とし，またミクロモデルの自由度も 5 千程度と大幅に削減した。しかし既述のように医学・生理学上精度の高い検討を行うには心筋細胞をより詳細にモデル化できる細かいメッシュで計算する必要があり，そのためには計算時間の制約のみならず，メモリ容量の観点からもより大規模な並列計算機が必要になる。これらのことを勘案し今回の HPC 特別研究プロジェクトではミクロモデルの 六面体要素数は 5,736，自由度は 26,121 とした。本解析では均質化アルゴリズム中のミクロモデルの解法においてスカイライン法を用いているため，LU 分解因子の成分数は 52,946,032 となり 1 ミクロモデルあたり 424Mbyte 程度のメモリ容量が必要となる。したがって，コアあたりのメモリ容量が 2Gbyte の場合には，1 コアに割り当てることができるミクロモデルの数は高々4 個までとなる。また 今回のテストではマクロモデルは心房を取り除いた両心室のみとし，これに 6144 個のミクロモデルを定義した。

第 21 図にミクロモデル総数に等しい 6144 コアを用いた場合の収縮初期(0～50 回目までの時間ステップ)でのミクロモデル計算における主な処理時間を,第 22 図 に計算開始時から 1 心周期半経過するまでの累積計算時間を示す。第 21 図中の「特性モード求解」とは均質化法においてミクロモデル内の詳細な変位分布とマクロモデルでの巨視的な歪を関係付ける計算を，また「ミクロ求解」とはミクロモデルの平衡をとる計算を意味する。特性モードの求解，およびミクロ求解におけるLU分解因子の計算は 10 タイムステップに一度のみ実行しており,図中のピークはそれら演算に要する時間に対応している。ここで求めた特性モードは，以後の Newton-Raphson反復におけるマクロスケールSchur complemet行列の近似行列として，ミクロ剛性行列のLU分解因子は､ミクロ求解時のGMRES反復法の前処理行列として再利用することにより大幅な計算量の削減を実現した[4]｡第22 図の累積計算時間のグラフで 30 ステップあたりから 160 ステップあたりまでの勾配がやや急になっている部分は，計算負荷が増す心臓の収縮期にあたる。これらの図からほとんどすべての時間をミクロモデル計算が占めていることが分かる。なお，1 周期半（1.5 心拍＝1.5 秒）のシミュレーションに 15.4 時間かかった。

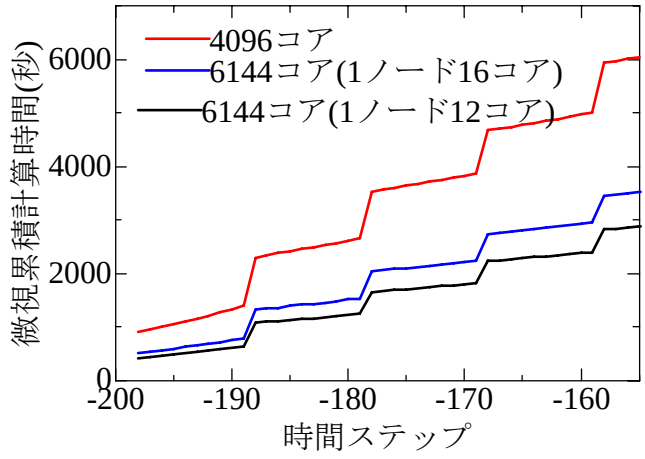


第 21 図 6144 コアによる収縮初期の各時間ステップにおけるミクロモデル計算時間

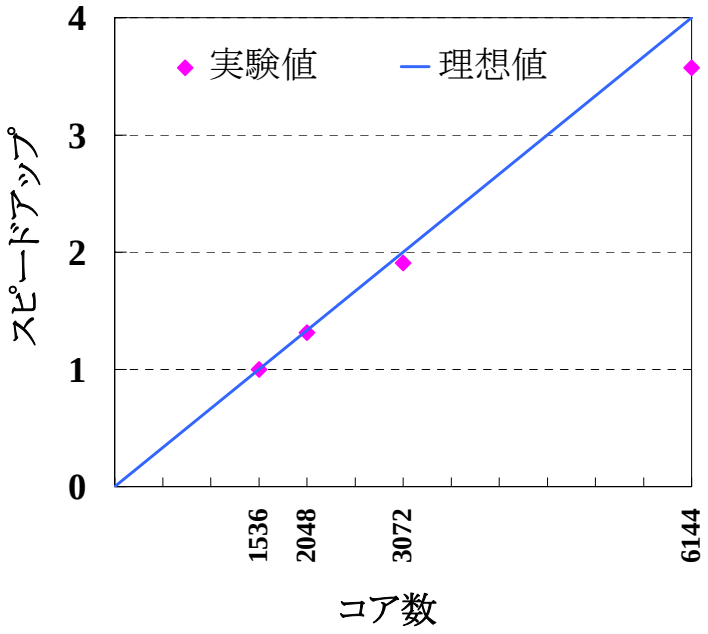


第 22 図　6144 コアによる 1 周期半（600 ステップ）における累積計算時間

次に並列化効率について調べた。第 23 図は計算初期におけるミクロモデル求解部での累積計算時間を比較したものである。ミクロモデル数に等しい数のコアを用いた場合(6144 コア)と T2K 総コアの半分にあたる 4096 コアを用いた場合を比較した。赤線と青線はともに，各ノードにおいて総てのコア(16 個)を用いた場合であり，黒は T2K の全ノード(512)を用い 1 ノードあたりのコア数を極力減らして 12 個とした場合である。4096 コアの場合は，ミクロモデルを 1 つまたは 2 つ受け持つコアが各ノードに混在し，1 つだけ受け持つコアは，他のコアの処理が終わるまで待つことになる。この待ち時間は無駄となるため計算時間は 6144/4096＝3/2 以上の比で 4096 コアが遅くなっている。ただしメモリアクセスの負荷が減少する分，2 ユニットを受け持つコアでの処理は速くなっていると考えられる。また，図中の青線と黒線を比較すると同じ総コア数でもノードあたりのコア数が少なくなるとかなり計算が速くなっていることがわかる。これはマルチコアを採用したシステムが共通して抱えるメモリアクセス競合時の性能劣化に関わる現象であり，今後何らかの対策が必要であると考えられる。第 24 図は総てのコアが 1 つ(6144 コア)，2 つ（3072 コア)，3 つ（2048 コア)，4 つ(1536 コア)のミクロモデルを均等に担当する場合の並列化性能を示したものである。1536 コアを基準にしたスピードアップで見ると，6144 コアの場合も 90%近い性能が得られていることが分かる。ミクロモデルの自由度が大きくなれば相対的に通信オーバーヘッドの比率が低下するため 6144 コアでも更に高いスピードアップが達成できると考えられる。

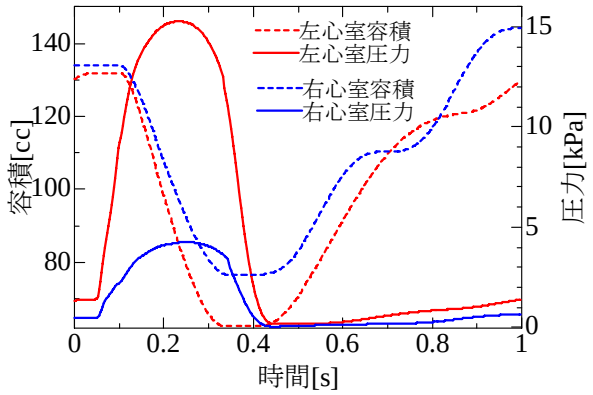


第23図　全コア数及び1ノード内コア数を変化させた場合のミクロ求解に要する計算時間の累積

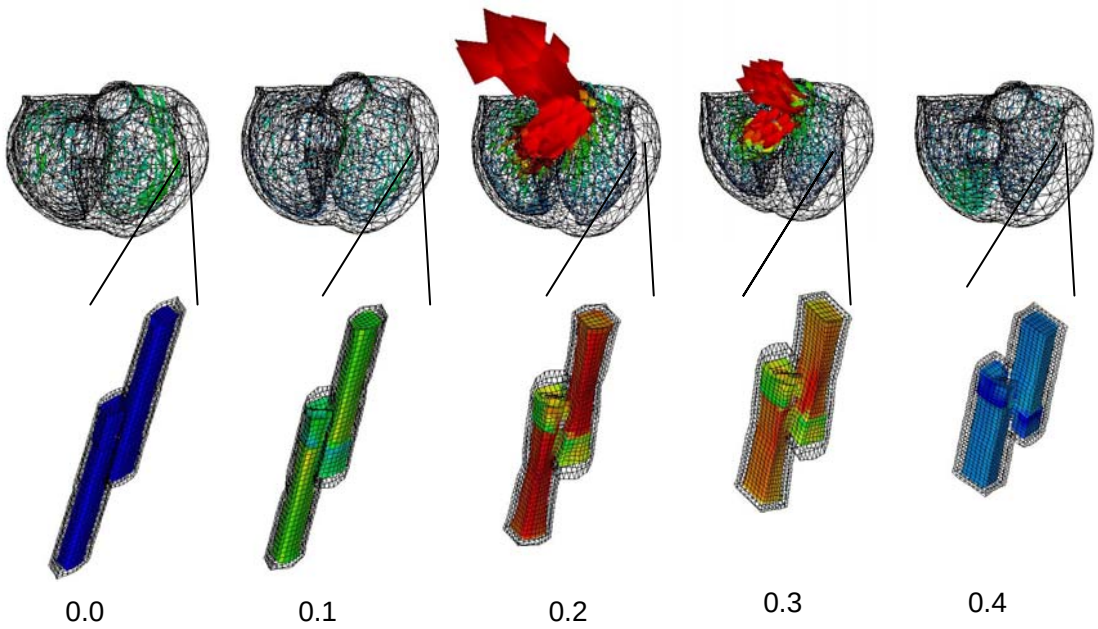


第 24 図 1536 コアを基準とした並列化効率

最後に計算結果を第 25 図，第 26 図に示す。第 25 図には両心室の圧，流量の関係を 1 心拍分示すが，これらは定性的，定量的に正常な心臓を再現している。第 26 図には収縮期をはさんだ幾つかの時刻における心室の変形と血液の流速ベクトル，および対応する時刻での心内膜側のある一つの有限要素に定義したミクロモデルの様子を可視化したものである。細胞の色は筋原線維の収縮力を表す。実験的に心筋細胞の生体内での（*in vivo*）計測を行うことは困難であり，その状態はシミュレーションによってのみ知ることが出来ると言っても過言ではない。また既述のように細胞内部の諸量も，細胞モデルを精密化すればそれだけ詳しく定量的に分析が可能となる。



第 25 図　HA8000(T2K)6144 コアによるマルチスケールシミュレーションにより得られた
両心室の圧・容積時刻歴



第 26 図　HA8000(T2K)6144 コアによるマルチスケールシミュレーションにより得られた
両心室の収縮と血液の拍出，ならびに心内膜側細胞の様子（数値の単位は s.）

４．終わりに

各種の検証を経て実用レベルに到達しつつある *UT-Heart* の概要とその背景を述べると共に，そのマルチスケール解析機能を HA8000（T2K）6144 コアを用いてテストした結果を紹介した。テストの結果，並列化効率と長時間に亘る安定稼動の両観点から予想通りの高い性能を示し，本心臓シミュレータが超並列計算の実用に耐え得ることを実証した。

現在，網羅的実験解析技術によって種々の条件下での細胞内のタンパク発現レベルの変化を容易に調べることが可能となっており，これらの知見を高度医療や創薬などに活かすために，今後は精密な細胞モデルに基づく短時間でのマルチスケールシミュレーションを現実のものと

する必要がある。このため当チームでは数万コアから数十万コアの並列計算に向けてマルチスケール心臓シミュレータ実用化のための準備を進めている。

UT-Heart は JST CREST・シミュレーション技術の革新と実用化基盤の構築領域（土居範久総括）における研究課題：「医療創薬のためのマルチスケール・マルチフィジックス心臓シミュレータの開発」のもとで集中的に研究が行われ発展してきた。現在その成果を引き継ぐ形で JST 産学共同シーズイノベーション化事業として富士通株式会社と共にペタフロップス級コンピュータの利用を前提とし実用化研究が推進されている。本 HPC 特別研究プロジェクトでのマルチスケールシミュレーションはその一環として東大チーム（筆者，杉浦清了，鷲尾巧，渡邉浩志，岡田純一，波田野明日可）と富士通チーム（門岡良昌，細井聡，清水香壱，渡邉正宏，平原隆生，山崎崇史，岩村尚，中川真智子）により行われた。また計算の実行に当たっては情報基盤センター・中島研吾教授に技術的ご支援を頂いた。これらの関係各位に感謝に意を表します。

参 考 文 献

[1] Zhang Q, Hisada T, Analysis of Fluid-Structure Interaction Problems with Structural Buckling and Large Domain Changes by ALE finite element method, Comput Methods Appl Mech Engrg, 190/48, pp 6341-6357, 2001.

[2] Washio T, Okada J, Hisada T, A parallel multilevel technique for solving the bidomain equation on a human heart with Purkinje fibers and a torso model, SIAM J Sci Comput, 30(6), pp. 2855-2881, 2008

[3] Washio T, Hisada T, Watanabe H, Tezduyar T E, A robust preconditioner for fluid-structure interaction problems, Comput Methods Appl Mech Engrg, 194, pp. 4027-4047, 2005

[4] Okada J, Washio T, Hisada T, Nonlinear Homogenization Algorithms with Low Computational Cost, JCST, 3(1) , pp. 101-114, 2009

[5] Okada J, Sugiura S, Nishimura S, Hisada T, 3D simulation of calcium waves and contraction in cardiomyocytes using the finite element method, Am J Physiol, 288, pp. 510-522, 2004

革新的シミュレーションソフトウェアの性能測定

加藤千幸

東京大学生産技術研究所
革新的シミュレーション研究センター

１．はじめに

革新的シミュレーション研究センター（以下、当センター）では，HA8000 クラスタシステムを活用し,ものづくりからバイオテクノロジ,ナノテクノロジまで幅広い分野のアプリケーションソフトに対するベンチマークテストおよび検証計算を実施した。使用したアプリケーションソフトは,文部科学省次世代 IT 基盤構築のための研究開発「革新的シミュレーションソフトウェアの研究開発プロジェクト」で開発された。これらのアプリケーションソフトウェアは，当センターがフリーソフトウェアとして公開しており，現在は、次世代 IT 基盤構築のための研究開発「イノベーション基盤シミュレーションソフトウェアの研究開発」プロジェクトのもと，開発が継続されている。本稿では，流体解析ソフト FrontFlow/blue，FMO 法に基づく量子化学計算プログラム ABINIT-MP,Kohn-Sham-Roothaan 法に基づく密度汎関数法プログラム ProteinDF,ならびに第一原理シミュレーションソフト PHASE に関して，それぞれ 2 章，3 章，4 章，5 章において紹介する。

２．流体解析コードのベンチマークテストおよび基礎検証

２．１　流体解析コード FrontFlow/blue の概要

FrontFlow/blue（FFB）は文部科学省次世代 IT 基盤構築のための研究開発「革新的シミュレーションソフトウェアの研究開発プロジェクト」において開発された流体解析ソフトウェアである。FFB の最大の特長は，乱流現象を高精度に予測できることにある。従来の流体解析では，時間平均に基づく Reynolds Averaged Navier-Stokes（RANS）が主に用いられていたため，流れの非定常性が現象の本質を支配する問題（例えば、振動や騒音）を取り扱うことができなかった。一方，FFB は非定常流れを高精度に予測することができる Large Eddy Simulation（LES）をベースとしているため，これまで予測することができなかった，機械内部の圧力変動スペクトルや空力騒音スペクトルの高精度予測が可能である。ここで，LES とは計算格子により解像することができる計算格子スケールよりも大きな渦（**Large Eddy**）の非定常の挙動を直接計算（**Simulation**）する手法である。計算格子スケールよりも小さな渦の挙動はサブグリッドスケールモデルによりモデル化される。

FFB のもうひとつの特長は“高速性”である。FFB はスカラマシン，ベクトルマシンにおいて，それぞれの特徴に応じて，マシンの性能を最大限に引き出すように設計されている。これにより大規模な解析を現実的な計算時間（例えば，ターボマシン 1 回転の計算が数時間）で実行することが可能となっている。現状では，スカラマシンにおいて，約 100CPU を用いて，1 千万点規模の六面体計算格子を用いた計算が可能である。また，ベクトルマシンにおいては約 4,000CPU を用いて，10 億点規模の六面体格子を用いた計算が可能である。LES は RANS と比較し，細かな計算格子が求められるため，実用上，その計算コストが問題になることがしばしば

ある。FFB が LES の実用的な問題への適用において多くの実績がある理由は FFB の“高速性”によるところが大きい。

２．２　流体解析コード FrontFlow/blue のベンチマークテスト

HA8000 システムに FrontFlow/blue ver.5.2 (FFB) をインストールし、六面体ソルバーles3c のベンチマークテストを行なった。使用したコンパイラは日立製コンパイラ mpif90 で、コンパイラオプションには-Oss を使用した。ノード内のコアの使用方法としては、全てのコアにプロセスを割り当てるフラット MPI と、CPU (4 コア) にプロセスを割り当て、CPU 内 (コア間) には自動並列化機能を用いるハイブリッド MPI の二通りをテストした。詳細は割愛するが、上記テストにおいて適切に NUMA 最適化を施す必要あることが確認された。テストデータは、ノードあたり百万要素に固定した。すなわち、全体のデータサイズは、ノード数に比例して大きくなる。ノード数として、1、2、4、8、16、32、64、128、256 をテストした。テストデータの全体要素数は最大で約 2.6 億点である。ステップあたりの計算時間を測定し、測定時間およびソルバーの演算回数より、実効性能を算出した。第 1 表および第 1 図に、ベンチマークテストの結果を示す。本ベンチマークテストの結果を以下にまとめる。

■単体の実効性能は約 3%であった。

■フラット MPI を用いたほうが、ハイブリッド MPI より若干性能が高かった。

■フラット MPI では 256 ノード(4096 コア)まで、ハイブリッド MPI では 128 ノード (2048 コア) まで高い並列性能を確認できた。

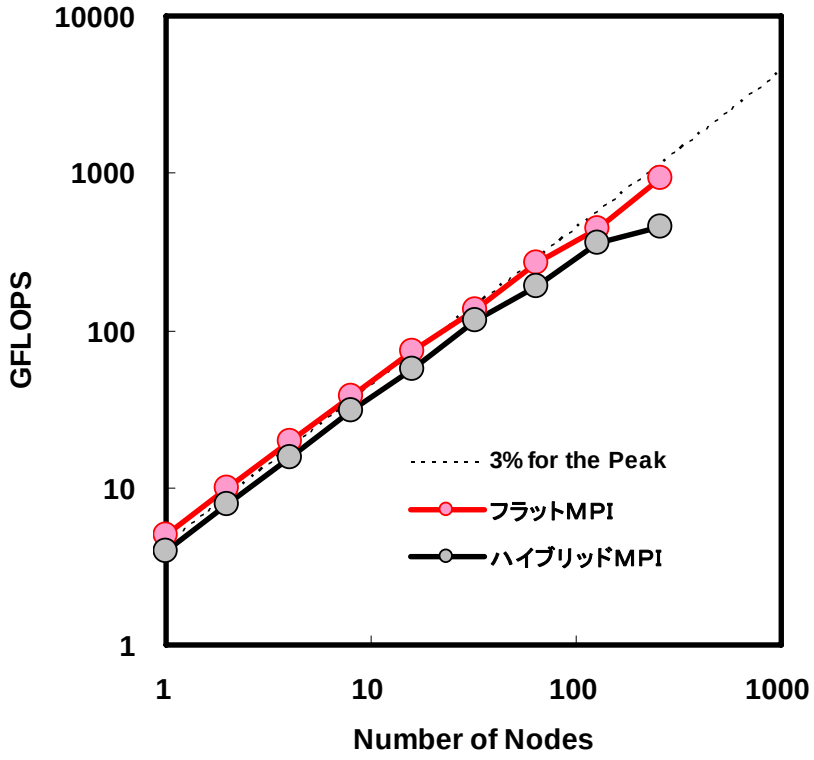
■プロセス数が多くなると，入出力に費やされる時間が 20 分程度となり，無視できなくなることが確認できた (第 2 図参照)。

今後の課題としては、単体実行効率および実用問題への応用が挙げられる。

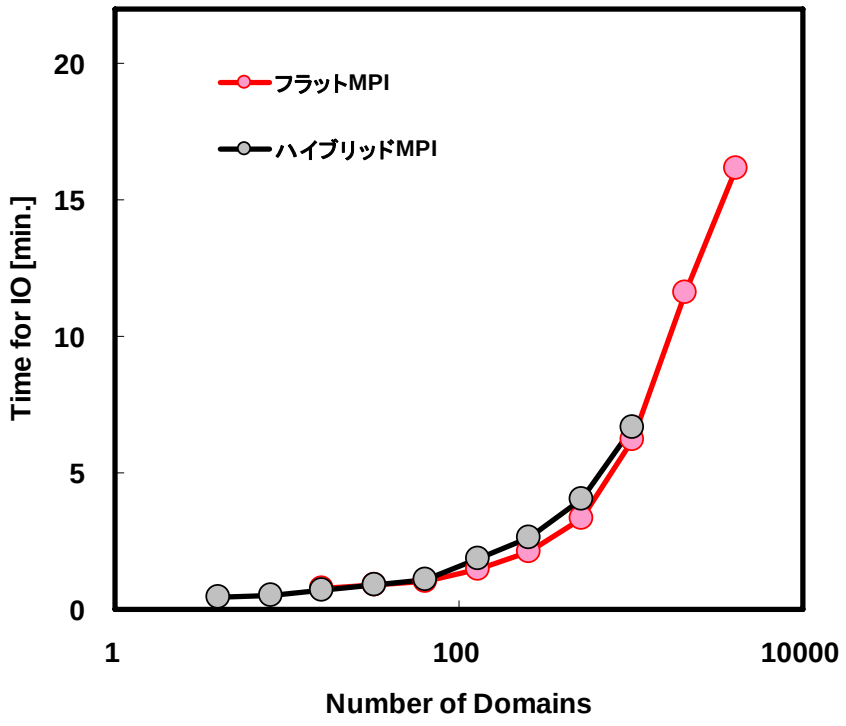
第 1 表：　HA8000 システムにおける FFB ベンチマークテスト結果

テストモード	ノード数	プロセス数	計算時間　(sec)		実行性能	
			0step	100step	GFLOPS	対ピーク (%)
フラット MPI	1	16	47	442	5.1	3.4
	2	32	55	454	10.0	3.4
	4	64	63	473	19.5	3.3
	8	128	87	509	37.9	3.2
	16	256	128	556	74.8	3.2
	32	512	201	678	134.2	2.8
	64	1024	374	855	266.1	2.8
	128	2048	698	1,277	442.1	2.3
	256	4096	968	1,525	919.2	2.4
ハイブリッド MPI	1	4	27	535	3.9	2.7
	2	8	30	540	7.8	2.7
	4	16	41	552	15.7	2.7
	8	32	53	577	30.5	2.6
	16	64	66	624	57.3	2.4
	32	128	111	666	115.3	2.4
	64	256	157	827	191.0	2.0

	128	512	244	964	355.6	1.9
	256	1024	399	1,525	454.7	1.2



第 1 図： HA8000 における FFB ベンチマークテスト結果



第 2 図： FFB ベンチマークテストにおいてファイル IO に費やした時間

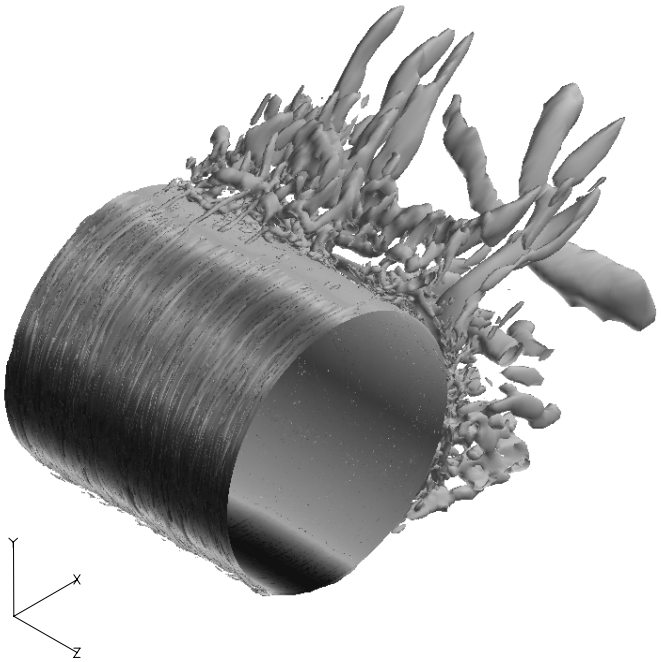
２．３　円柱周り流れ解析

HA8000 におけるFFBの検証計算のため，FFBを用いて円柱まわり流れ解析を実施した。円柱直径および主流流速をベースとするレイノルズ数は 8.5x10^{5}であり、このレイノルズ数では、流れははく離前に乱流遷移する。上記の高レイノルズ数における数値計算例はまれであり、流れの構造やメカニズムを理解するうえで本検証計算が有用である。本検証計算の計算条件をにまとめる。

第 2 表：　FFB による円柱周り流れ解析の計算条件

項目	内容
ソルバー	FFB 六面体ソルバー
計算格子	六面体　約 10 万要素
コンパイラ	日立製フォートランコンパイラ
使用ノード数	HA8000　16 ノード　（ハイブリッド MPI）
乱流モデル	Detached Eddy Simulation[(1)]

2000 ステップ計算するための計算時間は約 4 時間で，実効性能は約 28.5GFLOPS であった。計算結果の一例として、第 3 図に DES により計算された瞬時流れ場の渦構造を示す。渦構造は 2 次不変量の等値面として表現されている。



第 3 図：　レイノルズ数 8.5x10^{5}の円柱まわり流れ瞬時流れ場の渦構造（2 次不変量等値面）

２．４　空力騒音の直接計算

空力騒音の予測手法は分離計算と直接計算のふたつに大別される。分離計算では，音場が流れ場に影響を無視することにより流れ場と音場を別々に計算する。すなわち，はじめに音源データを得るために流れ場の計算を行い，その後に，音の伝播を計算する。一方，直接解法では，流れ場と音場を同時に計算する。FFB では分離計算による空力騒音予測を行う。分離計算は，直接解法と比べ計算コストが低いため，流れの速さが音速と比べ比較的小さい低マッハ数流れから発生する空力騒音予測では有効である。しかしながら，音場が速度場に与える影響が無視できないキャビティ音等の現象では，直接解法による空力騒音予測が必要となる。筆者らは，空力騒音の直接計算にも取り組んでいる。本節では，HA8000 システムを用いて計算を実施した空力騒音の直接計算の結果について紹介する。

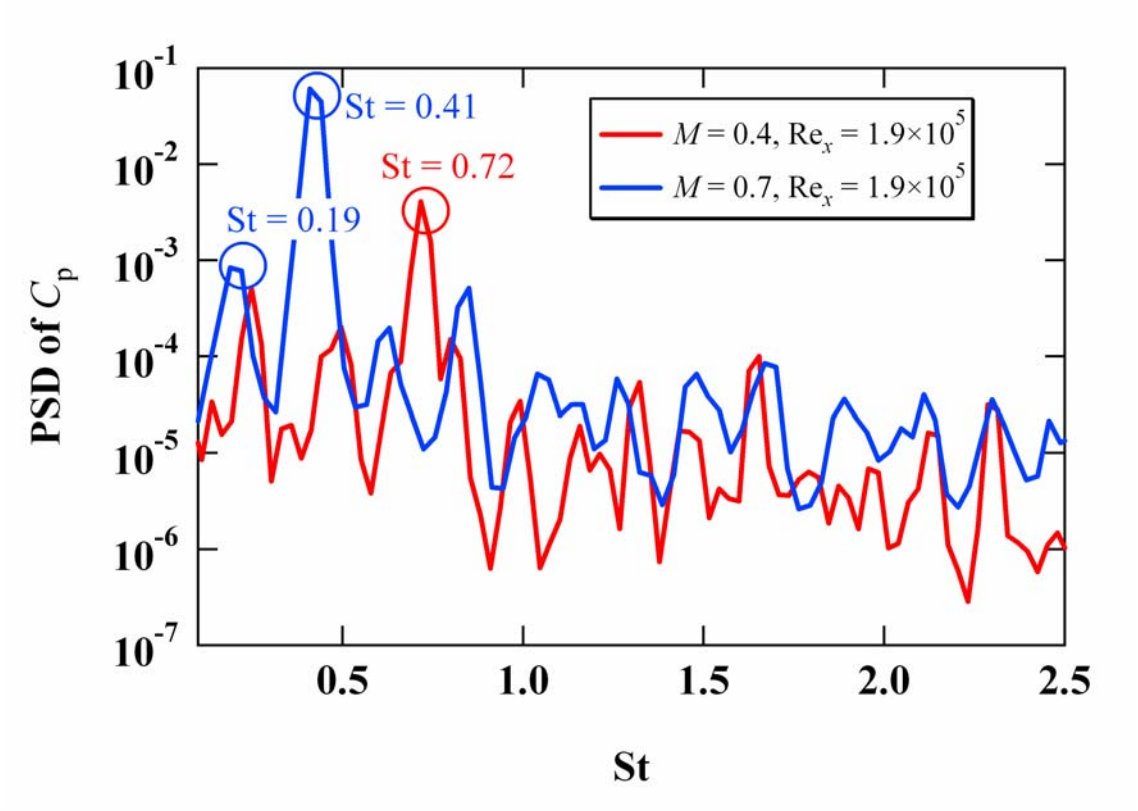
空力騒音は空気の流れ中の非定常運動から発生する音であり，音のパワーは速度の 4〜8 乗に比例し増大する。このため，近年の輸送機関の高速化において，空力騒音は大きな問題となっている。 特に，キャビティ上の流れでは，ピーク性の強い音が発生し，160dB に達する場合もある[(2)]。 こうしたキャビティ音は飛行機の着陸装置格納部や新幹線の車両車間部などから発生し，設計段階での予測が求められている。

キャビティ音の発生機構について，Rossiter[(3)] はキャビティ前縁で発生した渦が後縁にぶつかる際音が発生し，発生した音が前縁での渦形成を誘発するフィードバックループ(Fluid-acoustics interaction) を提案している。 Rossiter はこのフィードバック機構に基づくピーク音の周波数予測式を提案し，乱流境界層中のキャビティから発生するピーク音の周波数と一致することを確かめている。 また，Forestier は高速度シュリーレン法および Laser-Doppler Velocimetry (LDV) をもちい，乱流境界層中のキャビティのせん断層内に乱流の微小渦にくらべ大規模な渦が形成されることを明らかにしている。しかし，音波による大規模渦構造の形成機構や音波の発生機構は明らかにされていない。 本研究の目的は， 乱流境界層中のキャビティにおける Fluid-acoustics interaction を詳細に明らかにすることである。

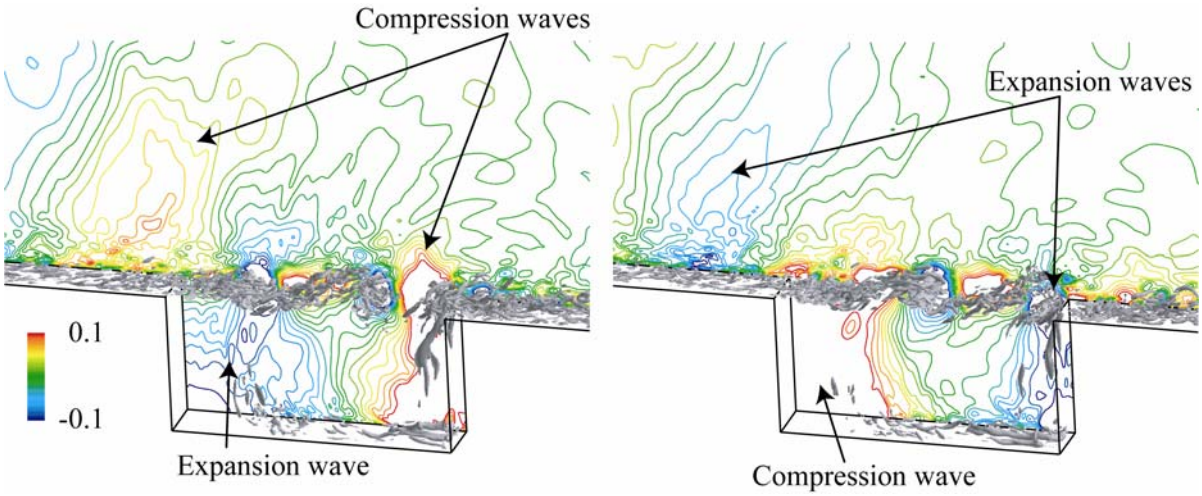
キャビティ底部におけるスパン方向に平均した圧力係数 のPower Spectral Density (PSD) をマッハ数 M= 0.4, 0.7 について第 4 図に示す.ここで，フーリエ変換にもちいたデータ長さは各マッハ数の主要なピーク周波数の 16 周期分とした。 各マッハ数において周波数 がSt = $fD/u_{1\infty}$ = 0.72 (M = 0.4)およびSt = 0.41 (M = 0.7)である強いピーク音が発生することがわかった。

第 5 図にマッハ数M = 0.4 における第 2 不変量の等値面および圧力係数Cp’の各地点における時間平均値と瞬時値の差を等高線にあらわす。乱流境界層中の微少な縦渦構造はキャビティ前縁においてはく離し，x_2方向の速度こう配により発生するx_3方向渦構造と重なり，渦構造は活発になる。また，せん断層内では，二次元的な大規模渦構造が生じおり，渦構造内では低圧領域が形成される.その結果，微小渦構造は大規模渦構造内で密になり，高圧領域では引き伸ばされリブ構造を形成する。

キャビティ内部のせん断層部以外での圧力変動は渦構造による変動は小さく，音波を表している。第 5 図より二次元大規模構造が後縁にぶつかる際に膨張波が発生することがわかる。 発生した音波はキャビティ内部を上流へ伝播し，前縁においてキャビティ外部へ放出される。



第 4 図：　圧力スペクトル



第 5 図：　圧力変動および渦構造

３．FMO 法に基づく量子化学計算プログラム ABINIT-MP の並列計算性能評価

３．１ はじめに

ABINIT-MPは「革新的シミュレーションソフトウェアの開発(RSS21)[1]」プロジェクトにおいて開発されたタンパク質-化学物質相互作用解析システムBioStationの中核となる量子化学計算プログラムである。ABINIT-MPの最大の特徴は，タンパク質-化学物質間の相互作用を量子力学に基づいて計算する点である。このため，古典力場では精度のよい計算が困難なタンパク質-化学物質間の相互作用を精密に計算できる。

通常タンパク質の量子化学計算は非常に困難であるが，非経験的フラグメント分子軌道法(Fragment Molecular Orbital Method:以下 FMO 法と略)[4-10]を採用しているため，ABINIT-MP はタンパク質の量子化学計算も現実的な時間(30 並列計算で半日～1 日程度)で可能となっている。FMO 法を採用したため，化学物質-タンパク質の結合様式をアミノ酸残基単位で解析できるようになっているのも ABINIT-MP の大きな特徴である。結合様式の詳細な情報は，医薬品等の分子設計において非常に有用である。このため，BioStation は量子論に基づく高精度なタンパク質-化学物質の相互作用解析システムとして創薬の現場において役立ちつつある[11,12]。

今回は，2008 年３月に公開された ABINIT-MP ver.4.1(最新版)の並列計算性能を調査するために，グリシン 128 量体(899 原子)に対して FMO-MP2/6-31G 計算を HA8000 上で実行した。その結果を報告する。以下,3.2 節「FMO 法に基づく相互作用解析」において FMO 法の概略と ABINIT-MP での並列計算方法ついて解説し，3.3 節「HA8000 における ABINIT-MP の性能測定結果」で今回の性能調査結果を報告する。3.4 節「まとめ」では，調査結果をまとめる。

３．２ FMO 法に基づく相互作用解析

３．２．１ FMO 法計算の流れ(FMO-Hartree-Fock 計算)

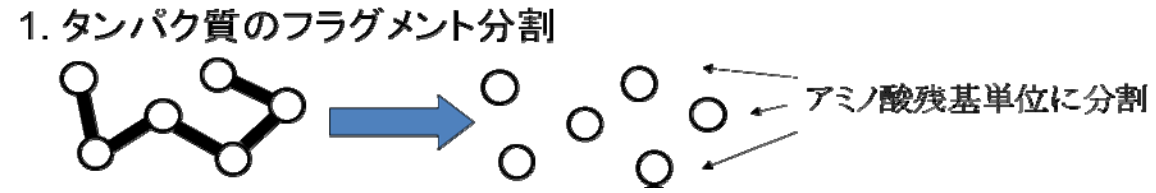
タンパク質の FMO 法計算は，3 段階に分けて行う。まず，タンパク質を構成要素であるアミノ酸残基単位に分割する(第 6 図参照)。

続いて，各フラグメントにおいて Hartree-Fock(HF)計算を実行する(Monomer Self-Consistent Field 計算;以下 Monomer SCF 計算と略)。この際，周囲フラグメントからの静電ポテンシャルを考慮していることに注意する。全てのフラグメントに対し分子軌道計算を実行し，分子全体の電子密度が自己無撞着になるまで iteration を行う(第 7 図参照)。

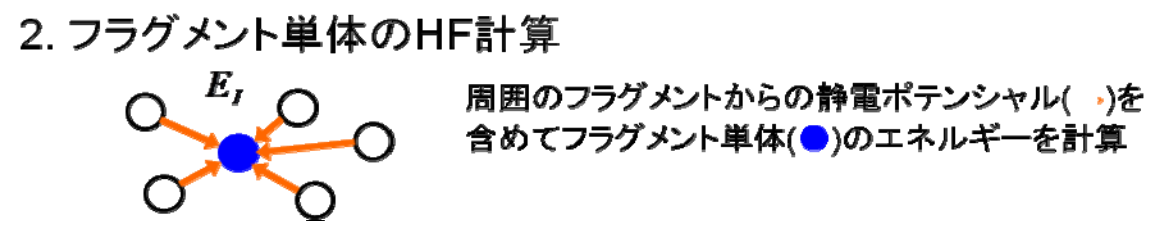
最後に，フラグメントペアに対しても HF 計算を行う(Dimer Self-Consistent Filed 計算;以下 Dimer SCF 計算と略)。この際，周囲のフラグメント(モノマー)からの静電ポテンシャルを考慮する(第 8 図参照)。

Dimer SCF計算は互いに近距離にあるフラグメントペアに対してのみ行い，遠く離れたフラグメント間ペアに対しては,静電相互作用項のみを考慮した近似計算(Dimer静電近似)を実行する。このため，FMO法全体としての計算時間は，ほぼO(N^1～N^2)と非常に高速になっている。

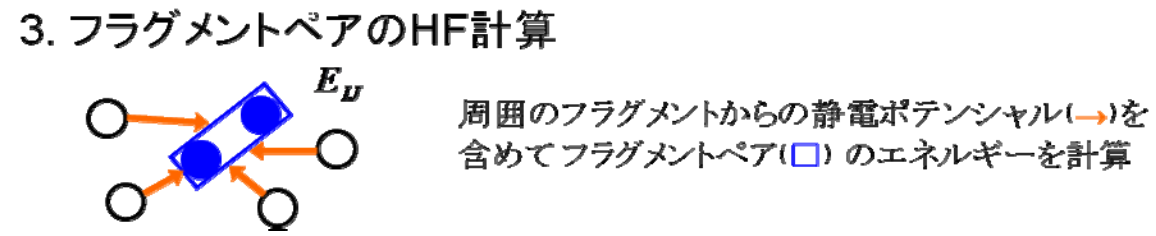
[1] http://www.ciss.iis.u-tokyo.ac.jp/rss21/



第 6 図:タンパク質のフラグメント分割



第 7 図:モノマーに対する SCF 計算



第 8 図:ダイマーに対する SCF 計算

３．２．２ MP2 計算による分散力の記述

今回並列性能調査を行う際に用いる計算手法は，MP2 (Møller-Plesset の２次摂動)計算である。MP2 計算とは，HF 波動関数を無摂動波動関数，静電相互作用から HF 平均場を引いたものを摂動項として取り扱った二次摂動計算のことである。FMO 法のフレームワークでは，フラグメントおよびフラグメントペア に対して MP2 計算を行う[13-15]。

タンパク質中にはトリプトファン，フェニルアラニンなどといった環状側鎖を持つアミノ酸残基が存在することが多い。このような環状側鎖中のπ軌道と化学物質との間には，π-π相互作用や CH/π相互作用といった分散力が働き，タンパク質-化学物質の結合において重要な役割を果たしている。分散力の記述は HF 計算ではできないため，電子相関を考慮した計算手法が必要となる。電子相関を考慮した計算の中でも MP2 計算は，計算コストが他の手法に比して軽いため，タンパク質という大規模分子における分散力の記述に有効であると考えられている。

３．２．３ ABINIT-MP における FMO 法の並列化実装

ABINIT-MP は，FMO 法の実装の際に二段階の並列化を行っている。即ち，フラグメント(もしくはフラグメントペア)に対する並列化とフラグメント(もしくはフラグメントペア)内の並列化である。ABINIT-MP では，各フラグメント独立に monomer SCF 計算および monomer MP2 計算を行うことが可能なので，非常に高い並列計算性能を出すことができる。また、各フラグメントペアに対しても同様に独立に dimer SCF 計算，dimer 静電近似計算を実行することが可能である。このため，FMO 法全体としても非常に高い並列計算性能を示す(3.3 節参照)。

３．３ HA8000 における ABINIT-MP の性能測定結果

３．３．１ 計算条件

ABINIT-MP のコンパイル方法、計算対象分子や計算条件について説明する。
ABINIT-MP のコンパイルには,日立製最適化フォートランを使用し,最適化レベル O4 でコンパイルした。また,並列化ライブラリは MPICH1.2 を用いた。コンパイル条件を第 3 表にまとめる。

第 3 表:使用したコンパイラ及びコンパイルオプション

コンパイラ	日立製作所最適化フォートラン f90
最適化オプション	O4
並列化ライブラリ	MPICH 1.2

計算対象分子としてグリシン 128 量体(全 899 原子:第 9 図参照)を採用し,計算手法/基底関数は MP2/6-31G 計算を採用した。フラグメント分割の際，フラグメントあたり 1 アミノ酸残基を割り当て，8 コアを割り当てた。また，1 コアあたり 1500MB のメモリ容量を使用した。第 4 表に計算条件をまとめる。

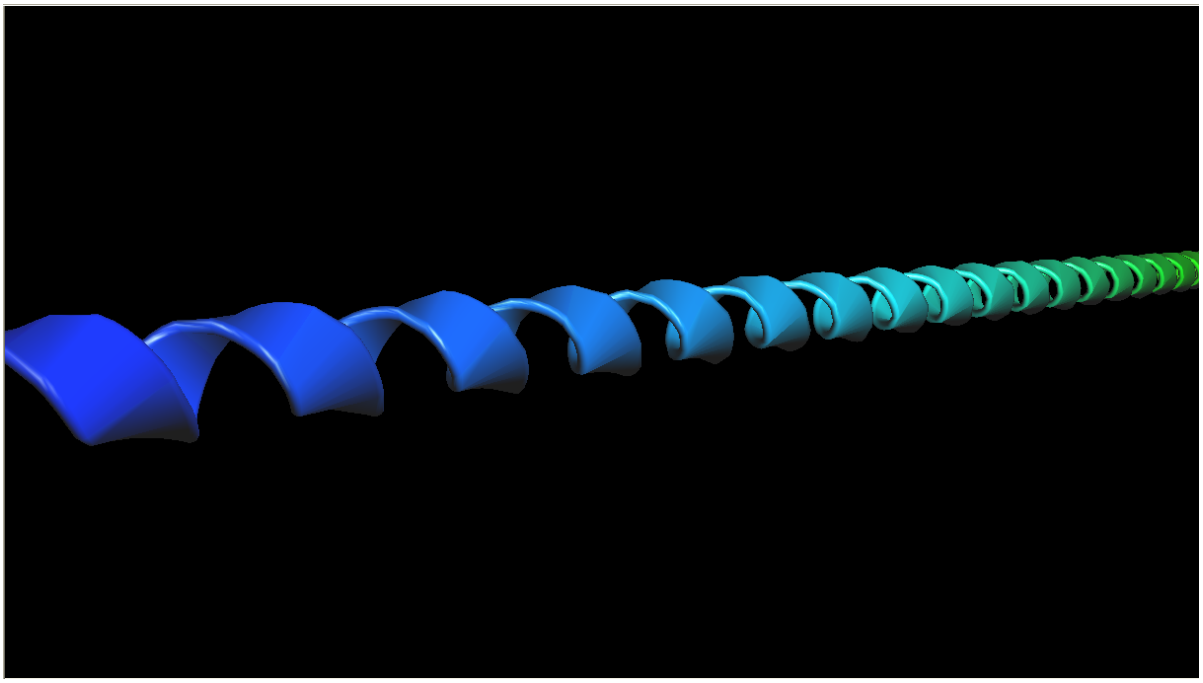
第 4 表: 計算条件

計算手法/基底関数	MP2/6-31G
フラグメントのサイズ	1 アミノ酸残基
コア数/フラグメント	8
メモリ容量/コア	1500MB

また,参考のため実際に使用したジョブ投入スクリプト例を以下(第 5 表)に示す。

第 5 表: 16 ノード 128 コア使用時の計算投入スクリプト

```
#!/bin.bash
#@$ -q b0n_16
#@$ -N 16  ←――――  16 ノード使用
#@$ -J T8  ←――――   8 コア/ノードを使用
#@$ -lT 1:00:00
cd /home/kobayasi/JOB
mpirun -np 128  abinitmp.exe < ./input/Gly128_MP2_6-31G.ajf > ./output/Gly128.out
```



第 9 図：性能調査に用いたグリシン 128 量体(α-helix)。リボン表示で図示する。

３．３．２ ABINIT-MP の性能測定結果

３．３．１の計算条件で 1,2,4,8 ノードを使用した計算を実行し，第 6 表の測定結果を得た。並列化率 α は，コア数 p の時の計算時間 $T(p)$ がアムダールの法則，即ち

$$\frac{T(p)}{T(1)} = 1 - \alpha + \alpha/p$$

に従うと仮定し，測定された計算時間を用いて最小二乗フィットした。この結果，得られた並列化率は $\alpha = 0.9994$ (99.94%)であった。

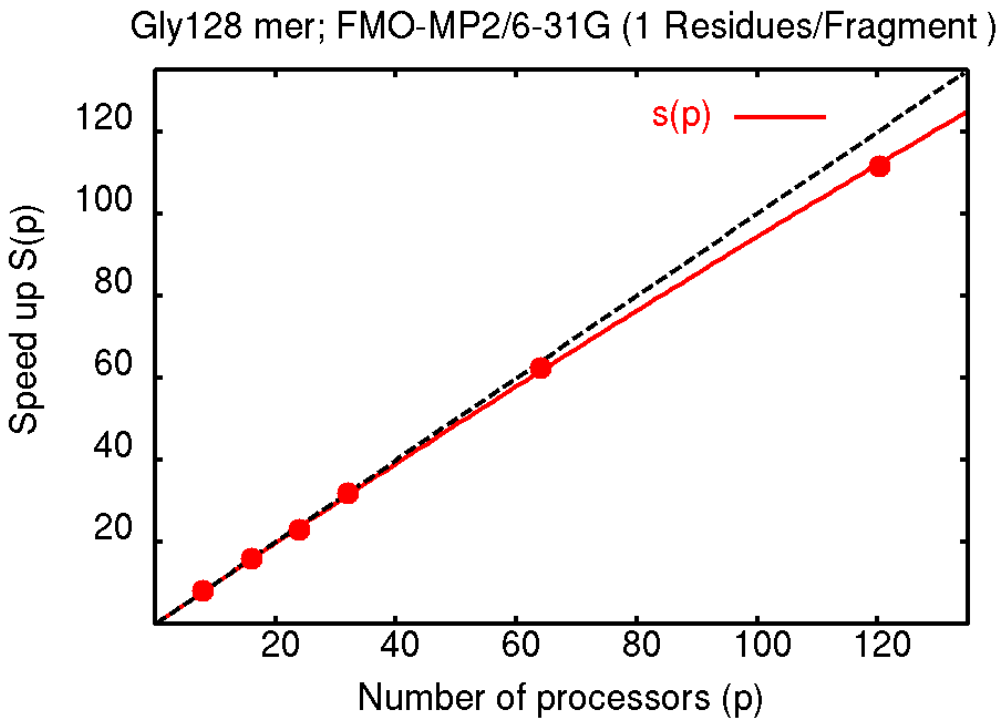
第 6 表：Gly128 量体の FMO-MP2/6-31G 計算

Gly128 残基(原子数=899,原子軌道=5389)の FMO-MP2/6-31G 計算		
ノード数	コア数	計算時間
1	8	11240.3
2	16	5638.9
4	32	2833.1
8	64	1444.8
16	128	740.0
ABINIT-MP の並列化率 99.94%		

また,測定結果から得られた高速化率を図示する(第 10 図参照)。図中には高速化 $S(p)$ 率もアムダールの法則に従うと仮定し，

$$S(p) = \frac{T(1)}{T(p)} = \frac{1}{1 - \alpha + \alpha/p}$$

となる曲線を赤字で示している。



第 10 図:HA8000 における ABINIT-MP の高速化率。横軸はプロセッサ数,縦軸は高速化率を示す。

また，測定されたFlops数を第 7 表にまとめる。ピーク性能比は，測定されたFlops数と理論性能(=9.2GFlops×コア数)との比から算出した。

第 7 表:ABINIT-MP の Flops 数とピーク性能比

Gly128 残基(原子数=899,原子軌道=5389)の FMO-MP2/6-31G 計算			
ノード数	コア数	GFlops	ピーク性能比(%)
1	8	3.69	5.02%
2	16	7.35	4.99%
4	32	14.63	4.99%
8	64	28.82	4.89%
16	128	56.36	5.02%

３．４ まとめ

ABINIT-MP の HA8000 上における並列計算性能を調査するため,Gly128 量体の FMO-MP2/6-31G 計算を 1(8 コア),2(16 コア),4(32 コア),8(64 コア),16(128 コア)ノードを用いて行った。
測定された結果から，ABINIT-MP の HA8000 上での並列化率は 99.94%であり,演算性能はピーク性能の 5%程度であることが判明した。

このことから，現在の ABINIT-MP は,1,000 並列(100 ノード規模の計算)という大規模並列計算においても十分な性能(並列化効率 50%以上)を出すと期待される。

今後 ABINIT-MP は，10,000～100,000 並列計算で十分な計算性能を発揮するため，更なる並列化率向上を目指したチューニングを行う予定である。

４．　ProteinDF

ProteinDF は，タンパク質全電子計算を目的とした Kohn-Sham-Roothaan 法に基づく密度汎関数法プログラムである。Kohn-Sham-Roothaan 法は分子積分演算，交換相関積分演算と一般行列演算からなる行列方程式である。巨大生体分子であるタンパク質の電子状態を高速かつ高精度にシミュレーションするために，ProteinDF では様々な高速化・並列化を施している。開発はオブジェクト指向言語 C++を利用し，行列演算ライブラリとして LAPACK，ScaLAPACK を利用している。

現在，生体分子の構造データベース PDB に登録されているタンパク質の 99%以上は 1,000 残基以内に収まる。すなわち，1,000 残基程度のタンパク質の全電子シミュレーションが可能になれば，ほとんどのタンパク質を網羅することができることになる。

このサイズのタンパク質の全電子計算を，スプリットバレンス型の基底関数を用いて実行すると行列次元数は約 100,000 になる。これを倍精度の密行列で保持するために必要なメモリサイズは，行列一つにつき 80 GB 程になる。行列演算，例えば行列積(A×B=C)には最低 3 つの行列が必要であるので，240 GB のメモリが必要ということになる。ProteinDF では RI 法を用いるため，さらに 2～3 倍次元程度の大きな補助基底関数を必要とする。到底，分散メモリ型並列計算機 1 ノードに収まるメモリサイズではない。

このような大規模メモリが必要な行列演算に対して，ProteinDF では巨大行列を並列計算機の各ノードへ分散保持させる戦略を採った。行列データの分配方法として ScaLAPACK で採用されている方式を利用した。この行列保持方式を採用にするにあたり，独自の分子積分演算リーチンの並列化方法を見直す必要があったが，一般行列演算には ScaLAPACK を利用することができた。ScaLAPACK は様々な計算機ごとに適したチューニングが行われ，ライブラリとして利用できるため，ScaLAPACK の採用は開発コスト・性能の両面から利点があった。

上記方法によりいくつかの巨大行列を並列計算機内で分散保持することが可能になったが，これでも量子化学計算で用いられる SCF 繰り返し計算に必要な行列を全てメモリ上に確保することはできない。そこで ProteinDF では，直近の演算に必要な行列以外はメモリ以外のデバイス，すなわちディスク上に退避することで巨大分子の量子化学計算を可能にしている。したがって，行列操作・演算が必要になるたびにディスクへの I/O と各ノード間通信とが発生することになり，計算効率はこれらの性能に大きく左右される。

ProteinDF プログラムは HA8000 システム上にてトラブルなくビルドし，シミュレーションを行うことができた。大小いくつかのデータセットにてシミュレーションを実行した結果、残念ながら今回はディスク I/O が足枷となり本システムのパフォーマンスを発揮するに至らなかった。一方，他システムの ProteinDF ベンチマークから，10,000 並列でも性能を発揮できると予想される良好な結果が得られている。HA8000 システムでは，各ノードに実装された高速な一時ディスク領域を利用することができる。今後，分散行列を各ノードの一時ディスク領域へ退避する等の仕組みを ProteinDF に導入することにより計算効率を大幅に改善できると考えられる。

５．第一原理シミュレーションソフト PHASE

密度汎関数法と擬ポテンシャル平面波手法に基づく第一原理シミュレーションソフトPHASE は様々な物質の性質の解析を可能にする。特に，半導体デバイス材料のシミュレーションにその威力を発揮してきた。たとえば，トランジスター向けの誘電体材料として期待されているア

モルファス・ハフニウム・アルミニウム酸化物(HfAlO)やセリウム酸化物の誘電物性解析を世界に先駆けて行ってきた[16,17]。また，半導体不純物系の 1 万原子を越えるモデルの大規模電子状態計算を地球シミュレータの 512 ノードを用いて行い，16.2 Tflop/sの性能を達成し，SC07のゴードンベル賞の最終選考に残った実績[2]がある。地球シミュレータとは異なるアーキテクチャーのHA8000 で，同様な大規模電子状態計算を行うためにはプログラムコードの最適化が必要である。今回は，コンパイラの選定とキャッシュチューニングの調整を行い，SR11000 との性能比較を行ったので，それらについて報告する。

PHASE コードは Fortarn90 で書かれており，HA8000 で利用できるフォートランコンパイラは日立製最適化フォートランコンパイラとインテルフォートランコンパイラである。日立製コンパイラを用いる場合は，高速フーリエ変換ライブラリとして MATRIX/MPP ライブラリを採用し，LAPACK ライブラリは情報基盤センターが提供するものを使用した。使用したメイクファイルは SR11000 向けものとほぼ同じであるが,MPI ライブラリのディレクトリの指定と C 言語の関数の名前変換方法を変更したので，変更した箇所を以下に示す。

```
F90FLAGS = -64 -Oss -i,P -msg=w -parallel -I/opt/mpich-mx/include
F77FLAGS = -64 -Oss -i,P -msg=w -parallel -I/opt/mpich-mx/include
CFLAGS = -O0 -DINTEL -64
```

インテルコンパイラを用いる場合は，EM64T/AMD64 向けのメイクファイルを書き換えて使用した。高速フーリエ変換ライブラリとして情報基盤センターが提供するFFTW3 ライブラリ[3]とインテルマスカーネルライブラリに含まれるLAPACKライブラリを用いた。変更した箇所を以下に示す。

```
F90FLAGS = -w90 -w95 -W0 -traceback -I/opt/itc/mpi/mpich-mx-intel/include
F77FLAGS = -w90 -w95 -W0 -traceback -I/opt/itc/mpi/mpich-mx-intel/include
LIBS = -Wl,-rpath,/opt/intel/mkl/10.0.3.020/lib/em64t
       -L/opt/intel/mkl/10.0.3.020/lib/em64t -Wl,--start-group
       /opt/intel/mkl/10.0.3.020/lib/em64t/libmkl_intel_lp64.a
       /opt/intel/mkl/10.0.3.020/lib/em64t/libmkl_sequential.a
       /opt/intel/mkl/10.0.3.020/lib/em64t/libmkl_core.a -Wl,--end-group
       -L/opt/itc/lib -lfftw3 -L/opt/itc/mpi/mpich-mx-intel/lib -lmpich
       -Wl,-rpath,/opt/mx/lib64,-rpath,/opt/mx/lib -L/opt/mx/lib64 -L/opt/mx/lib/
       -lmyriexpress -lrt -lfmpich -Bdynamic -lpthread
```

半導体デバイス材料として用いられるアモルファス酸化物$HfSiO_2$(第 11 図)を計算対象とし，HA8000 においてPHASEの性能調査を行った。比較的小規模である 192 原子系の計算における 1 回の波動関数更新に要する時間の計測結果を第 12 図に示す。日立製コンパイラとインテルコンパイラを用いて作成したプログラムをMPI並列で実行したが，日立製コンパイラの場合はMPIの

[2] http://sc07.supercomputing.org/schedule/event_detail.php?evid=11132
[3] http://www.fftw.org/

みの並列実行のほかにCPU内でコア間スレッド並列化を行うハイブリッド 4x4 並列実行も行った。MPI並列のみで1ノード(16MPI並列)で実行した結果を比較すると，日立製コンパイラよりもインテルコンパイラの方が約20秒速い。ノード数を増やすと両方とも実行時間が減少する。8ノードでの並列加速率は日立製コンパイラが3.6倍で,インテルコンパイラが3.9倍である。日立製コンパイラのハイブリッド4x4並列実行は1ノードでMPI並列のみの実行よりも約10秒遅いが,スケールが良いため8ノードでインテルコンパイラのMPI並列のみ実行よりも速くなり,その並列加速率は5.4倍である。

PHASEはベクトル計算機向けに開発されたが,公開されている最新コードではキャッシュチューニングが施されている。たとえば，波動関数$\phi_n(\mathbf{r})=\Sigma_j c(n,j)\exp(i\mathbf{g}_j\cdot\mathbf{r})$と擬ポテンシャルのプロジェクター関数$\beta_p(\mathbf{r})=\Sigma_j d(p,j)\exp(-i\mathbf{g}_j\cdot\mathbf{r})$の内積$\langle\beta_p|\phi_n\rangle$の計算はオリジナルコードでは以下に示すような3重ループで計算されている。

```
DO p = 1, Nprojector
        DO n = 1, Nband
                DO j = 1, Nplanewave
                        <βp|φn> = <βp|φn> + d(p,j) * c(n,j)
                END DO
        END DO
END DO
```

$N_{projector}$はプロジェクターの数,N_{band}は電子状態の数,$N_{planewave}$は平面波(基底関数)の数である。最内側ループはベクトル計算機ではベクトル化される非常に長いループである。キャッシュを使用するスカラー計算機では,このままのコードではキャッシュミスが頻発するためCPUの性能を十分に引き出せない。そこで，キャッシュ利用効率を上げて性能を引き出せるように最内側のループ長を調節した。部分配列d(p,j1:j2)とc(1:N_{band},j1:j2)がキャッシュに乗った状態で演算が行なわれるように変更したコードを以下に示す。

```
DO j1 = 1, Nplanewave, Ncache
        j2 = j1 - 1 + Ncache
        DO p = 1, Nprojector
                DO n = 1, Nband
                        DO j = j1, j2
                                <βp|φn> = <βp|φn> + d(p,j) * c(n,j)
                        END DO
                END DO
        END DO
END DO
```

N_{cache}は部分配列の合計サイズがキャッシュサイズの 3/4 になるように調整している。Itaniumプロセッサーでは実際のキャッシュサイズを用いるようにしてあるが，ベクトル機以外の他の

アーキテクチャーではキャッシュサイズを 1MBに固定している。キャッシュブサイズは入力ファイルのcontrolブロック内の変数cachesizeにkB単位で設定することができる。たとえば，キャッシュサイズを 1MBに設定するには，

```
Control{
    cachesize = 1024
}
```

とする。今回紹介した日立製コンパイラ向けメイクファイルでプログラムを作成すると，キャッシュサイズ変数 cachesize のデフォルト値はゼロになってしまうので，かならずキャッシュサイズ変数を適切に設定していただきたい。

HA8000 はコアあたり 512kB の L2 キャッシュをもつため，すべてのコンパイラと並列化手法の組み合わせでキャッシュサイズ変数を 512kB に設定した。しかし，それぞれの組み合わせに最適なブロッキングサイズがあるはずなので，8 ノードでプログラムを実行し，最適なブロッキングサイズの探索を行った。第 13 図にその結果を示す。MPI のみの並列化では日立，インテルともキャッシュサイズ変数が 256kB で実行時間が最小になる。一方，ハイブリッド 4x4 並列ではキャッシュサイズ変数を L2 キャッシュサイズ 512kB よりも大きく取った方が速くなる。キャッシュサイズ変数が 1MB と 2MB とで実行時間の差は殆どなく，キャッシュサイズ変数が 512kB と 1MB とで実行時間の差はわずか 1 秒である。従って，MPI 並列化のみの場合と異なりハイブリッド並列化ではブロッキングサイズにあまり依らずに良い性能が得られると結論できる。まとめると，HA8000 における最適な並列化手法は日立製コンパイラを用いたハイブリッド 4x4 並列化であり，キャッシュサイズ変数を 1MB 程度に設定すると良い。

擬ポテンシャル平面波法コードには平面波に関する長いループ構造があるため，ベクトル計算機に向いている。SR11000 はスカラー計算機であるが大きな L3 キャッシュを備えている上にバンド幅が太いため，あたかもベクトル計算機のように振る舞う。実際，HA8000 と SR11000 の 1 ノードあたりの理論性能値は同じであるにもかかわらず，96 原子系で HA8000 と SR11000 の性能を比較すると SR11000 の方が 2〜3 倍良い性能を示す（第 14 図）。ノード数が増えると性能差はわずかに縮まるが，SR11000 の優位性は変わらない。540 原子系では SR11000 はスケールが良く，2(8)ノードで HA8000 の 2.6(3.1)倍の性能を示す（第 15 図）。540 原子系で４k 点を用いる場合は，第 16 図に示すように HA8000 のスケールも良く，32 ノードを利用すれば，SR11000 の 8 ノード利用時の性能 231 Gflop/s よりも 83 Gflop/s 高い性能が得られる。

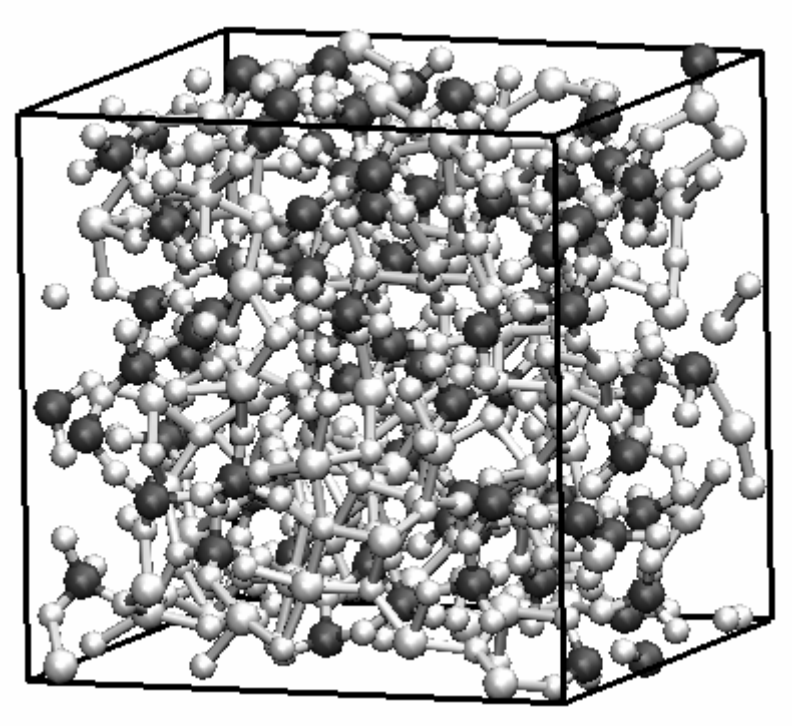
今回は個別ルーチンの性能評価まで踏み込んで行わなかった。また，詳細な報告は控えるが，数千原子系の大規模計算を HA8000 で実行するのは，現在のコードでは困難であることが判明した。地球シミュレータのようなベクトル並列計算機とは異なり，1000 MPI プロセスを超える並列実行を考慮したプログラム設計が必要であり，キャッシュ利用効率向上を目指して個別ルーチンの詳細な性能評価を行い，HA8000 のような超並列計算機向けにも PHASE コードを最適化していきたい。

最後に参考として， HA8000 ８ノードを用いてハイブリッド 4x4 並列で PHASE を実行するときのバッチスクリプトを紹介する。

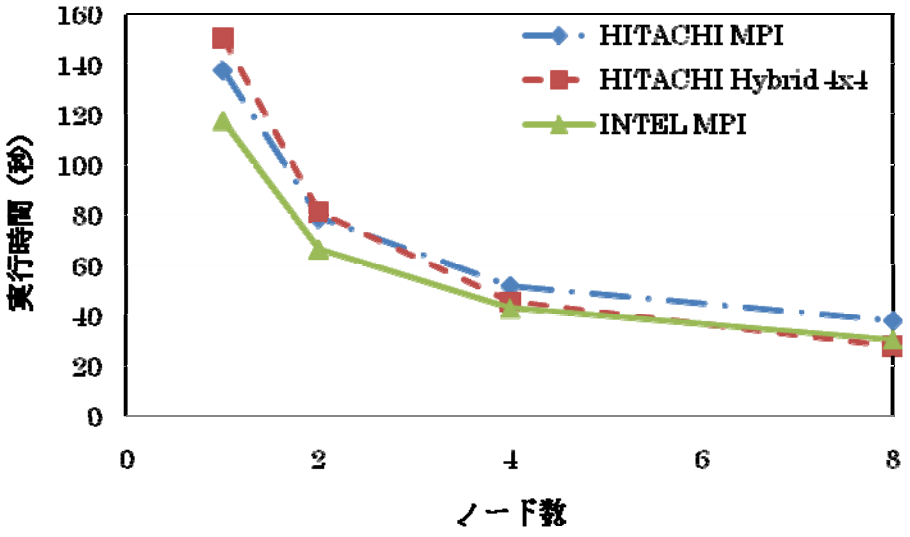
```
#!/bin/bash
#@$-q parallel
#@$-N 8
#@$-J T4
#@$-lT 8:00:00
export HF_PRUNST_THREADNUM=4

install=~/phase_v701/bin

idir=~/test/input
pdir=~/test/pp
odir=~/test/output
wdir=/tmp/work
bdir=/tmp/bin

cd $odir
cat <<here >stg.cfg
in $install/phase $bdir/phase
in $idir/nfinput.data $wdir/nfinput.data
in $idir/file_names.data $wdir/file_names.data
in $idir/continue.data $wdir/continue.data
here

echo 'start:' `date`
/opt/itc/bin/stagein $odir/stg.cfg
cp $pdir/* $wdir
cp $idir/continue_bin.data $wdir
cp $idir/nfchgt.data $wdir
cp $idir/zaj.data $wdir
cd $wdir
mpirun ~/numarun $bdir/phase
cp $wdir/* $odir
echo 'end:' `date`
```

このバッチスクリプトを用いれば，プログラム実行中の入出力はノード内で行われるので，ネットワークファイルシステムを用いるよりも良い性能が得られる。これをそのまま使うには，ディレクトリ input に入力ファイル nfinput.data と file_names.data を置き，ディレクトリ pp に必要な擬ポテンシャルファイルを置く必要がある。また，PHASE の実行ファイルは通常インストール位置~/phase_v701/bin になければならない。HA8000 利用の手引きに説明されている NUMA コントロールを行うスクリプト numarun がホームディレクトリ直下に置かれているとして

いる。継続計算を行うためには 4 つの出力ファイル continue.data, continue_bin.data, nfchgt.data,zaj.data を継続実行時に入力ファイルとしてワークディレクトリ$wdir に置かなければならない。上に示したバッチスクリプトにおいて，継続実行時の操作は太字で記述されているので，初期実行時には削除して使用していただきたい。PHASE を HA8000 で実行するときに，このバッチスクリプトを役立てていただければと思う。

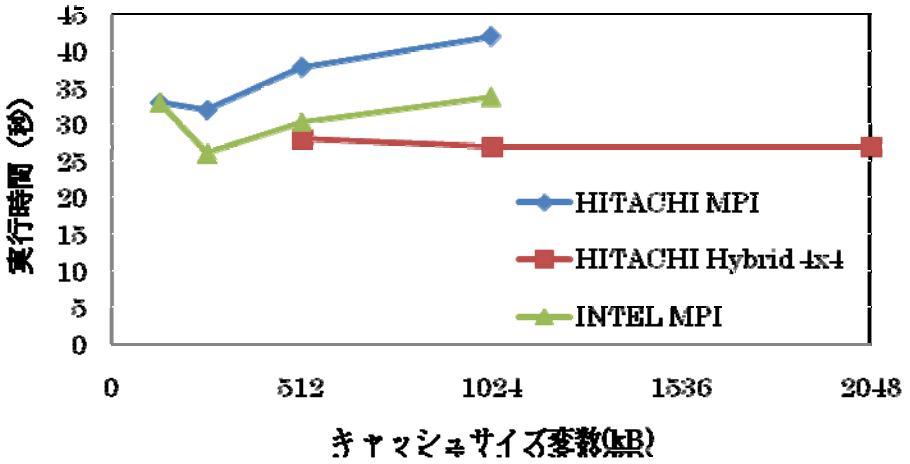


第 11 図　アモルファス$HfSiO_2$(540 原子)の構造モデル。
小さい白い球が酸素，大きい白い球がハフニウム，大きい灰色の球がシリコンである。

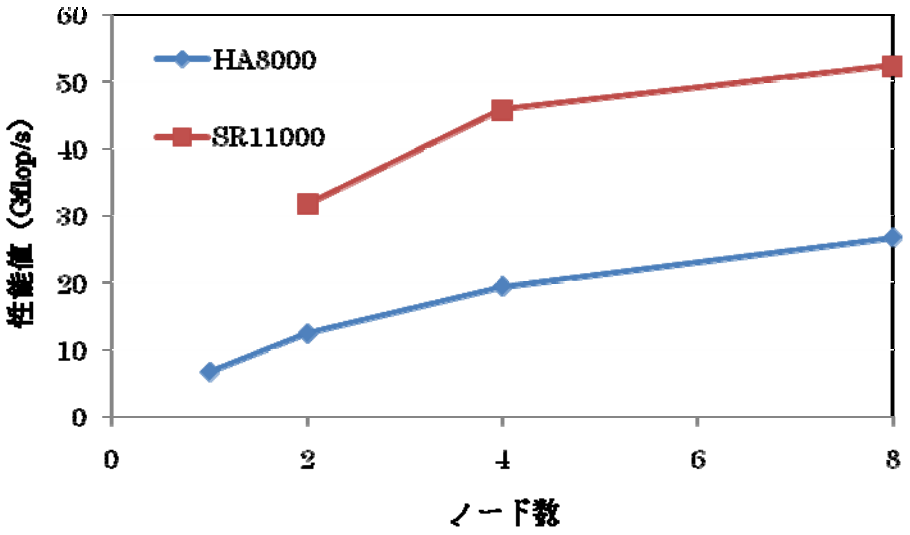


第 12 図：　$HfSiO_2$系(192 原子)の直線探索最小化の実行時間の並列度依存性。
コンパイラ(HITACHI と INTEL)，並列化方法(hybrid 4x4 と flat MPI)の組み合わせ毎にノード数に対する実

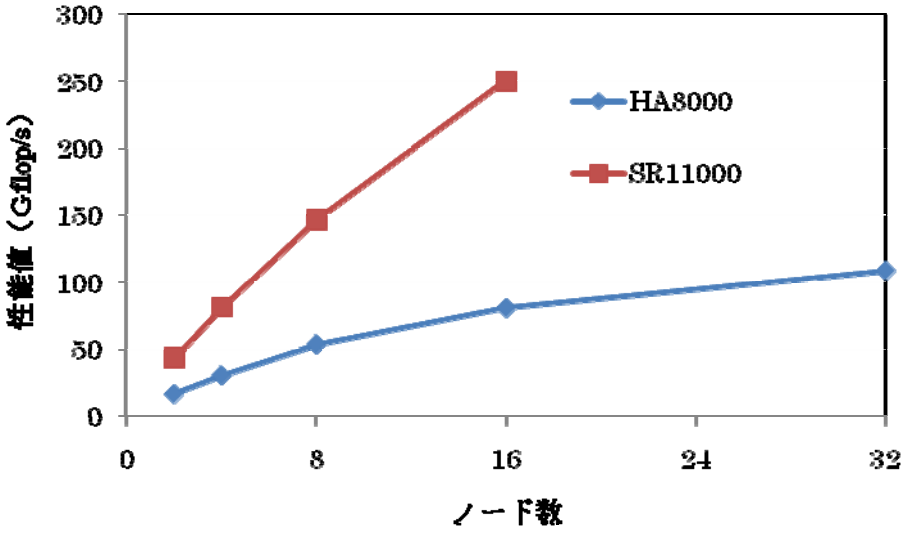
行時間の変化が示されている。キャッシュサイズ変数を 512kB に設定している。



第 13 図： $HfSiO_2$系(192 原子)の直線探索最小化の実行時間のキャッシュブロッキング依存性。
コンパイラ(HITACHI と INTEL)，並列化方法(hybrid 4x4 と flat MPI)の組み合わせ毎にキャッシュサイズ変数に対する実行時間の変化が示されている。８ノードで実行している。

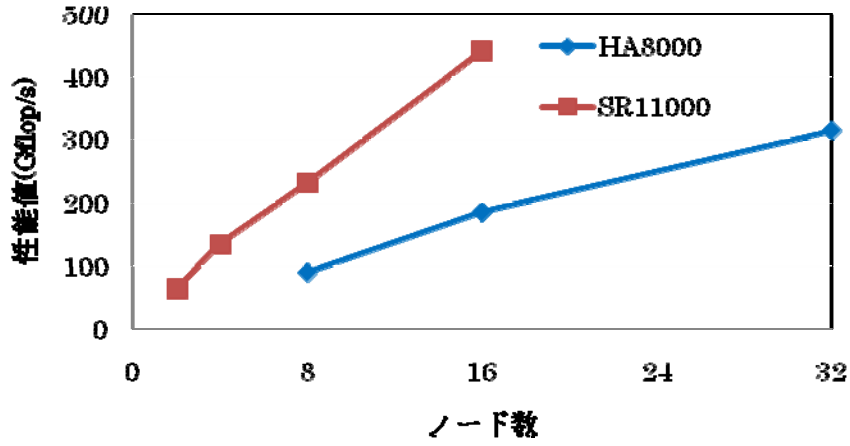


第 14 図　$HfSiO_2$系(96 原子，Γ点のみ)の電子状態計算の性能。
HA8000 と SR11000 の性能値を利用ノード数に対して示している。HA8000 では日立製コンパイラによるハイブリッド 4x4 並列化を採用し，キャッシュブサイズ変数を 1MB に設定している。



第 15 図： $HfSiO_2$系(540 原子, Γ点のみ) の電子状態計算の性能。

HA8000 と SR11000 の性能値を利用ノード数に対して示している。HA8000 では日立製コンパイラによるハイブリッド 4x4 並列化を採用し，キャッシュブロッキングのサイズを 1MB に設定している。



第 16 図： $HfSiO_2$系(540 原子, ４k点) の電子状態計算の性能。

HA8000 と SR11000 の性能値を利用ノード数に対して示している。HA8000 では日立製コンパイラによるハイブリッド 4x4 並列化を採用し，キャッシュサイズ変数を 1MB に設定している。

参 考 文 献

（１）Spalart, P. R., Jou, W.-H., Strelets, M., and Allmaras, S. R., 1997, “Comments on the Feasibility of LES for Wings, and on a Hybrid RANS/LES Approach”, In: Liu, C. and Liu, Z. (eds), Advances in DNS/LES, Proceedings of 1st AFOSR International Conference on DNS/LES, Ruston, Greyden Press, pp. 137-147.

（２）Krishnamurty, K., National Advisory Committee for Aeronautics Technical Report, No. 3487 (1955)

（３）Rossiter, J. E., Aeronautical Research Council, No. 3438 (1964)

（４）D. G. Fedorov, K. Kitaura, “Theoretical development of the fragment molecular orbital (FMO) method”, in Modern methods for theoretical physical chemistry of

biopolymers, E. B. Starikov, J. P. Lewis, S. Tanaka, Eds., Elsevier (2006) pp. 3-38.
（５）T. Nakano, Y. Mochizuki, K. Fukuzawa, S. Amari, S. Tanaka, “Developments and applications of ABINIT-MP software based on the fragment molecular orbital method”, in Modern methods for theoretical physical chemistry of biopolymers, E. B. Starikov, J. P. Lewis, S. Tanaka, Eds., Elsevier (2006) pp. 39-52.
（６）D. G. Fedorov, K. Kitaura, “Extending the Power of Quantum Chemistry to Large Systems with the Fragment Molecular Orbital Method”, *J. Phys. Chem.* **A 111**, 6904-6914 (2007).
（７）北浦和夫, “フラグメント分子軌道法の概要”, CBI 学会計算化学研究会ワークショップ(2008/5/22), http://www.cbi.or.jp/cbi/jigyou/FMO_gaiyo.pdf
（８）中野達也, 望月祐志, 甘利真司, 小林将人, 福澤薫, 田中成典, “フラグメント分子軌道法に基づいた生体巨大分子の電子状態計算の現状と今後の展望”, *J. Comput. Chem. Jpn.* **6**, 173-184 (2007).
（９）福澤薫, 中野達也, 加藤昭史, 望月祐志, 田中成典, “フラグメント分子軌道法による生体高分子の応用計算”, *J. Comput. Chem. Jpn.* **6**, 185-198 (2007).
（１０）佐藤文俊, 中野達也, 望月祐志 編著, “プログラムで実践する生体分子量子化学計算”,森北出版, 2008.
（１１）T. Ozawa, K. Okazaki, “CH/π Hydrogen Bonds Determine the Selectivity of the Src Homology 2 Domain to Tyrosine Phosphotyrosyl Peptides: An Ab Initio Fragment Molecular Orbital Study”, *J. Comput. Chem.* **29**, 2656-2666 (2008).
（１２）T. Ozawa, E. Tsuji, M. Ozawa, C. Handa, H. Mukaiyama, T. Nishimura, S. Kobayashi, K. Okazaki, “The importance of CH/π hydrogen bonds in rational drug design: An ab initio fragment molecular orbital study to leukocyte-specific protein tyrosine (LCK) kinase”, *Bioorg. Med. Chem.* **16**, 10311-10318 (2008).
（１３）Y. Mochizuki, T. Nakano, S. Koikegami, S. Tanimori, Y. Abe, U. Nagashima, K. Kitaura, “A parallelized integral-direct second-order Møller-Plesset perturbation theory method with fragment molecular orbital scheme”, *Theor. Chem. Acc.* **112**, 442-452 (2004).
（１４）Y. Mochizuki, S. Koikegami, T. Nakano, S. Amari, K. Kitaura, “Large scale MP2 calculations with fragment molecular orbital scheme”, *Chem. Phys. Lett.* **396**, 473-479 (2004).
（１５）Y. Mochizuki, K. Yamashita, T. Murase, T. Nakano, K. Fukuzawa, K. Takematsu, H. Watanabe, S. Tanaka, “Large scale FMO-MP2 calculations on a massively parallel-vector computer”, *Chem. Phys. Lett.* **457**, 396-403 (2008).
（１６）H. Momida, T. Hamada, Y. Takagi, T. Yamamoto, T. Uda and T. Ohno: ″Dielectric constants of amorphous hafnium aluminates: First-principles study″, *Phys. Rev. B* **75**, 195105-1-10 (2007).
（１７）T. Yamamoto, H. Momida, T. Hamada, T. Uda and T. Ohno, “First-Principles Study of Dielectric Properties of Cerium Oxide”, *Thin Solid Films* **486,** 136-140 (2005).

密度行列繰り込み群法と行列対角化による強相関量子系のシミュレーション：HPC 特別プロジェクト報告

町田昌彦，山田進，奥村雅彦

日本原子力研究開発機構・システム計算科学センター

CREST(JST)

今村俊幸

電気通信大学・電気通信学部

CREST(JST)

１．はじめに

現在，物理学において最も注目されている話題の１つに，フェルミ原子（Li 等）からなる極低温原子ガス（フェルミ原子ガスと呼ぶ）の物性が挙げられる（図１にそのイメージを示した）｛１｝。その理由は，フェルミ原子ガスでは，フェシュバッハ共鳴を利用することで原子間相互作用を任意に引力から斥力までコントロール可能であり，原子物理学の根本問題（原子間相互作用を制御すると何が起こりうるか？）から物理学一般の最大の難問（粒子間相互作用の制御により，どのように物質の様態変化を制御できるのか？）にまで解答を与えられる可能性を有するからである。最近，このフェルミ原子ガスにおいて，レーザーの干渉により作られる光学格子中(図２参照)で，超流動・絶縁体転移など，固体材料中で起こりうる様々な現象をほぼ完全に再現できることが示された｛２｝。この光学格子ではレーザー光や磁場などを連続的に変化させることが可能であるため，固体材料より遥かに容易にかつ系統的に物性を研究できることが分っており，固体物理学の最大の難問となっている高温超伝導体の超伝導発現機構｛３，４，５，６｝,最近現れた鉄系超伝導体｛７｝の発現機構，フラストレートした磁性体｛８｝，ランダムな強相関電子系｛９，１０｝などの様々な問題の理解に重要な貢献ができる可能性が指摘されている（図3 参照)｝。

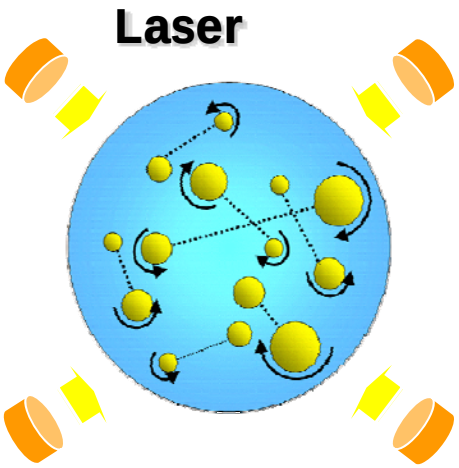


図1.　フェルミ原子ガスのイメージ。四方からレーザーを照射したり，磁場を印加する等により，その性質を制御できる。

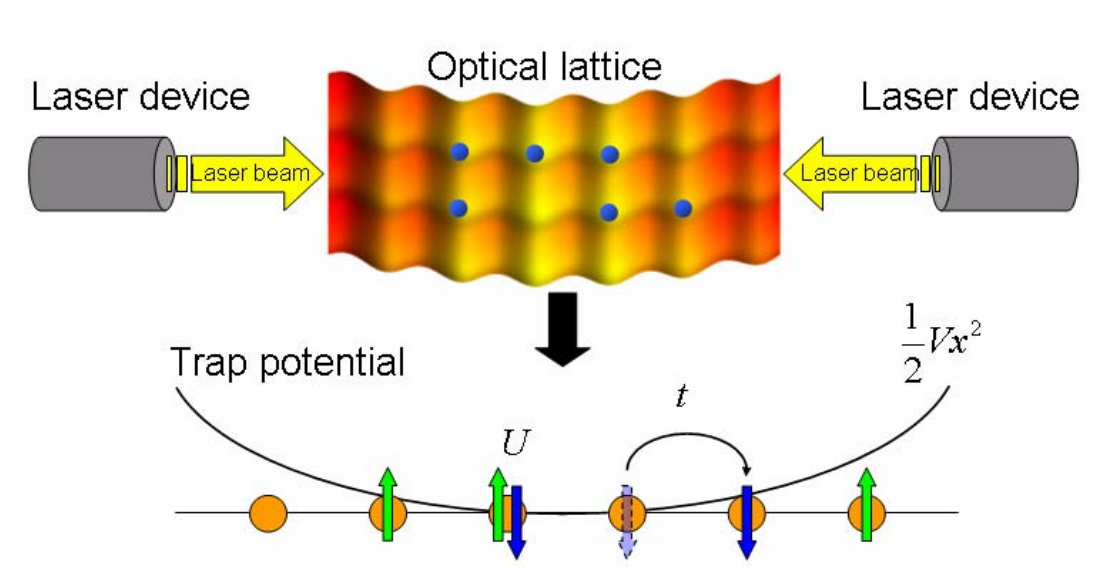


図2.　光学格子中のフェルミ原子ガスのイメージ（上図）。対向レーザーの干渉により格子（定在波）が形成される。フェルミ原子は，そのポテンシャルエネルギー極小点に位置することを好む一方，量子トンネル効果にて隣接格子点へと飛び移ることができる。この様子を模式的（1 次元にマップ）に示したのが下図である。

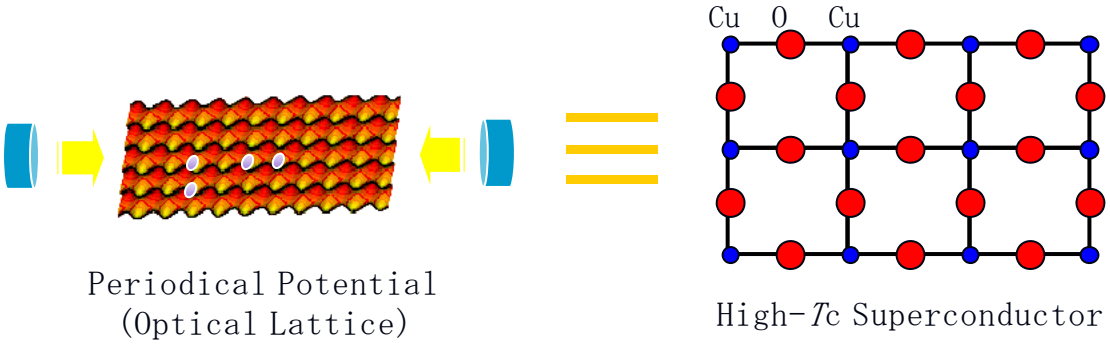


図3.　光学格子中のフェルミ原子ガスを上手に制御することで，高温超伝導体の CuO 面上の電子と同等の状態を作りだすことができる。高温超伝導と同じ現象（原子の場合は超流動）が観測されれば，高温超伝導機構の解明に大いに貢献できるかもしれない。

本記事にて紹介する HPC 特別プロジェクトでは，上記のように，原子物理学から固体物理学にまで広範囲に跨る物理学上の問題を厳密対角化法｛３，１１，１２，１３，１４，１５｝，一般の対角化法(全対角化法)｛１６｝と密度行列繰り込み群法｛１７，１８，１９，２０，２１，２２，２３，２４，２５，２６，２７，２８｝の二つの手法を用いて研究することを目標とした。これまで，厳密対角化法と全固有値を求める全対角化については，地球シミュレータでの最大ノード利用（512 ノード・4096ＣＰＵ）により計算科学上の成果（SC05 & SC06 における 2 年連続 Gordon-Bell 賞ファイナリスト選出｛１１，１２｝）も創出しており，本計算機 T2K を用いて，更なる世界最大規模の計算に挑戦する準備（次世代計算機が利用可能となった時）ができればと考えている。また，密度行列繰り込み群を用いたシミュレーションにおいては，本質的な超並列化に成功している研究グループは現在，世界において当グループのみ｛２１，２２｝

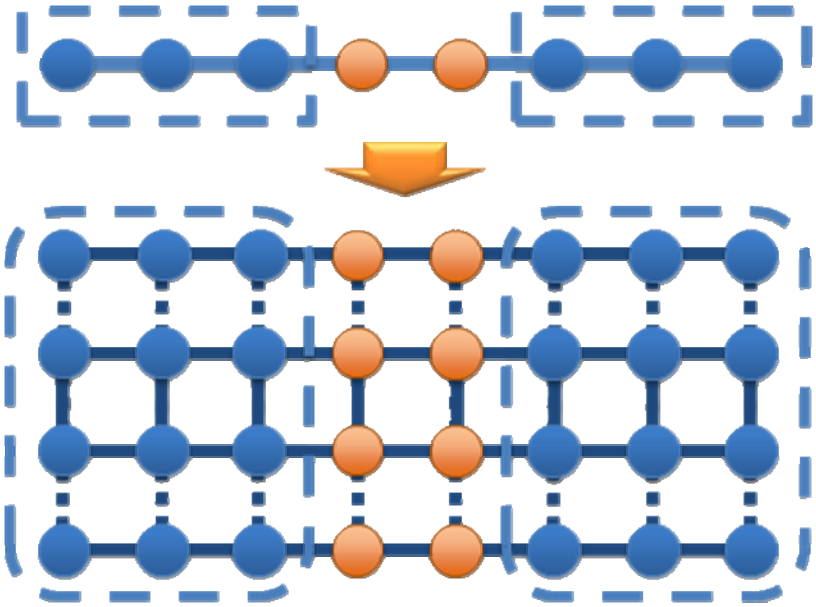


図4.　密度行列繰り込み群は，一次元のモデルに対して適用されてきたが，厳密対角化に相当する計算部分の並列化により，準２次元の梯子格子に対しても適用可能であることが分かっている。原理的には計算機資源が許す限り，梯子の足をいくらでも増やすことができる。

と自負しており，T2K を用いて従来の計算規模を遥かに超えた超並列計算を行い，準２次元系の基底状態の理解に対し，新たな成果の創出を目指すこととした（図４参照）。尚，本記事では，プロジェクトの限られた利用資源、利用時間の関係上，その大目標を達成するために，特別プロジェクトの期間内で得られた成果，発生した問題点，そして，今後への方策などを列記し，特に，私たちが直面した T2K 利用上での初歩的なミスから他のユーザーにとっても共通と思われる問題について率直に，研究の時系列に従って示していくこととした。この記事が T2K をこれから利用する研究者や，同じ悩みを抱く研究者の方々にとっての一助となれば幸いです。尚，以下の構成として，２章で厳密対角化法の T2K でのパフォーマンス評価結果について述べ，３章で，全対角化について同様の結果を記す。そして，４章にて，特に全対角化を移植し，チューニング時に出会った問題や特徴的な出来事を時系列に則して記した。

２．厳密対角化法の T2K 上でのパフォーマンス評価

我々の研究グループでは，2000 年ぐらいから光学格子上のフェルミ原子ガスやナノスケールの超伝導体の物性等が注目され始めたことを受け（しかも，対象とする系自体は有限系であり，計算資源が許せば計算可能である），それらの系の物性を理解する上で最も基本となるハバードモデル（図５参照）の厳密対角化法｛３｝の開発を 2004 年から開始した。また，同時に地球シミュレータ｛２９｝が稼動し，数テラバイトのメモリが使用可能となったことを受け，地球シミュレータ上で最大限度のメモリを使うことで，上記の物理問題解決に大きく貢献できるに違いないと考え，開発した厳密対角化法の超並列化および高速化の研究を同時に進めてきた｛３０｝。本章では，上記の経緯にて開発した地球シミュレータ用超並列厳密対角化コード｛１１，１２，３１｝を T2K に移植し，マルチコア性能の評価を行った結果について記す。

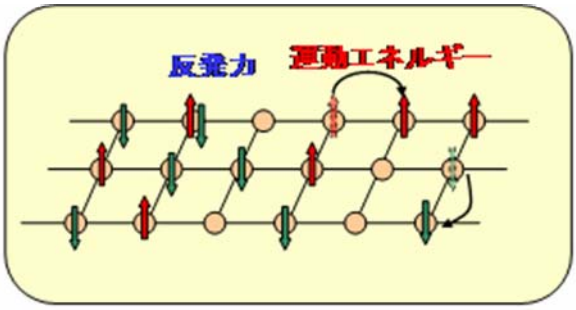


図5.　ハバードモデルの模式図。アップスピンとダウンスピン粒子(ともにフェルミ粒子)が格子点上に存在し,同じ格子点にアップスピン粒子とダウンスピン粒子が存在すると反発力によるエネルギーの損失が,粒子が移動できると運動エネルギーの利得がそれぞれ発生する。

T2K のマルチコア性能を評価し，地球シミュレータ・ベクトルプロセッサーとの比較評価を行うため，地球シミュレータ用に開発した厳密対角化のプログラムを用いて，MPI と OpenMP を用いたハイブリッド計算を行った。利用したプログラムは，地球シミュレータ上での最適チューンニングが済んでおり，地球シミュレータ上ではピーク性能の５０％を超える。尚，移植に際しては，ベクトルパイプラインを想定したループアンローリングを元に戻したことと，MPI_PUT，MPI_GET(MPI-2 命令)が利用できないため，MPI_isend，MPI_irecv，MPI_Wait を利用した通信に変更した。それ以外は支障が無かったので，そのまま T2K で実行した。

次にマルチコアの性能評価結果だが，その評価を行うために必要な情報として，本計算手法の特徴を以下に記す。厳密対角化法でキーとなる数値計算は行列（ハミルトニアンを行列で表現したもので超大規模の疎行列である）×ベクトル（ハミルトニアン行列の固有ベクトル）積の繰り返しであり，実は疎行列×ベクトル演算の並列化を論じるだけで十分である。実際の並列化に際しては，疎行列の並列化は効率が激減するため，一度，疎行列×密行列に計算対象の対称性を考慮して変換し，密行列を均等分割して並列化している｛１１，１２｝。これにより，分割単位毎のロードバランスが取れ，高い並列化効果が得られることを地球シミュレータ上で確かめている｛１１，１２｝。また，本手法の二つ目の特徴として，行列の転置を取る必要等から通信量が多い上，メモリーアクセスも頻繁に行われるため，マルチコアでの性能向上が難しいプログラムであることを付記しておく。

性能評価結果に移ろう。先の移植を実施した後，まず，マルチコアの特性を調べ，その効果を体感する意味を含めて，小さいモデル（16 サイトのハバードモデルを表現している約６４００万次元のハミルトニアン行列）でのシミュレーションとして

case1-1　４プロセッサ/ノード　×　１ノード　×２スレッド（４MPI x ２スレッド）
case1-2　４プロセッサ/ノード　×　２ノード　×２スレッド（８MPI x ２スレッド）
case1-3　４プロセッサ/ノード　×　１ノード　×４スレッド（４MPI x ４スレッド）

の３ケースを実行した。得られた結果を表１に示す。尚,スレッドは OpenMP で並列化している。同じシミュレーションであり，ノード数とコア数だけを変えて実行している。

表１：小規模モデルのシミュレーション結果

	コア数	プロセッサ数	１プロセッサあたりのコア数	計算時間（秒）	計算性能（ピーク比）
case1-1	8	4	2	184.49	2.70GFLOPS（3.67%）
case1-2	16	8	2	102.64	4.86GFLOPS（3.30%）
case1-3	16	4	4	160.45	3.11GFLOPS（2.11%）

表１の結果の case1-1 と case1-2 の比較から１プロセッサあたりのコア数が増加しなければ 1.80 倍の高い高速化を達成しているが，case1-1 と case1-3 の比較からコア数が増加したのでは 1.15 倍とほとんど高速化していないことが確認できた。この原因としては，本アプリケーションは本来地球シミュレータ（ベクトル計算機）用であり，データの再利用を全く考慮しないアルゴリズムを採用しているためであると考えられ，具体的には

1. データの再利用を考慮していないため，キャッシュメモリを有効的に利用できない
2. メインメモリへのアクセスが大量に発生するが，T2K のメインメモリは４つのコアで共有しているため，メモリーアクセスが競合し，十分な計算データを用意することができない
3. コア数が増えて計算の理論性能が向上しても，計算に必要なデータがないため，計算ができず，実際の計算性能は向上しない

という状況になっていると考えられる（正確に知るためには，メモリーアクセス等の情報を解析する必要があるが，今回は時間が限られていたため，解析は実行できず）。

次に，大規模なシミュレーション（21 サイトのハバードモデルを表現する約 135 億次元のハミルトニアン行列）の性能評価を行う。ここでは，大規模なシミュレーションとして

case2-1　４プロセッサ/ノード　×　128 ノード　×2 スレッド（512MPI x 2 スレッド）
case2-2　４プロセッサ/ノード　×　64 ノード　×4 スレッド（256MPI x 4 スレッド）

の 2 ケースを並列計算した結果を表２に示す。

表２：大規模モデルのシミュレーション結果

	コア数	プロセッサ数	１プロセッサあたりのコア数	計算時間（秒）	計算性能（ピーク比）
case2-1	1024	512	2	704.44	165.03GFLOPS（1.75%）
case2-2	1024	256	4	1135.85	102.31GFLOPS（1.09%）

この結果からも，総コア数が同じでも１プロセッサあたりの利用するコア数が少ないほうが約 1.6 倍高速であることが確認できる。このことからも，本アプリケーションをそのまま利用したのでは，メインメモリへのアクセスが多く，メモリバンドの競合のため，T2K の計算性能を有効に利用できないと推測できる。

以上の結果から，マルチコアのアーキテクチャを利用するためにはこれまで以上にキャッシュメモリを考慮する必要がある。そのため，本プログラムのように既存の計算機（特にベクトル機）用に開発されたアプリケーションをマルチコア計算機で実行する場合には，キャッシュ

メモリが有効に利用できているかを調査し，有効利用できていない場合には，キャッシュメモリを有効に利用できるアルゴリズムに変更することが必要であることが分かる。

３．全対角化法の T2K 上でのパフォーマンス評価

我々の研究グループでは，2 章で記した厳密対角化法の超並列化と共に，全ての固有値及び固有状態を得るために必要な全対角化法の超並列化も同時進行で研究開発を行ってきた。全対角化は，1 章にて記した物性物理や原子物理だけでなく，化学やバイオ等，あらやる分野にて利用可能な普遍的ルーチンであり，その研究開発は様々なアプリケーションが動作するだろう次世代計算機において重要であると考えている。計算は以下に記述するように，３つのルーチンから構成され，その一部は地球シミュレータ上で 70%以上のパフォーマンスを出している｛１２，３２｝。また，スカラー計算機上での並列化も進めており，パフォーマンスは十分に満足できる結果である。このように，全対角化は，2 章で記した厳密対角化と比べて，あまり，機種その他に依存せず，チューニングによる性能向上が目に見えて上昇するなど，マルチコアによる超大並列計算を行う予定の次世代計算機上でも，高い性能を出すと期待できる。本章では，これまでに開発してきた超並列全対角化コードを T2K に移植し，マルチコア性能の評価を行った結果を記す。

本プロジェクトでは，量子シミュレーションの中核となるハミルトニアン行列の全固有値固有ベクトル求解ルーチンを地球シミュレータより移植し，1000 コア以上の環境で高速にかつ安定，スケーラブルに動作するかの検証を行うことを目標として作業に着手した。まず，1 コア単体でハウスホルダー順変換 2.39GFLOPS，ハウスホルダー逆変換 2.94GFLOPS を記録した(尚，全体角化法は，ハウスホルダー順変換，分割統治法，ハウスホルダー逆変換の 3 つのルーチンに分割できる)。次に 1 ノード内の 16 コア使用による flat-MPI 実行での動作を検証し，1000 次元から 32000 次元まで問題なく動作することを確認した。性能は 1 コア時からチューニングを行い，32000 次元の計算でハウスホルダー変換 16.51GFLOPS，逆変換 60.55GFLOPS を記録した。尚，逆変換は 1 ノードの理論ピーク(147.2GFLOPS)の 41.1%であり，極めて高い性能を示している。次に，複数ノード間複数コアによる flat-MPI 実装での動作検証を行うため，32 ノードまでを利用して動作検証を行った。32 ノードまでの２ベキのノード数利用では全く問題なく動作した。性能は，32 ノード使用時 96000 次元でハウスホルダー順変換 428.35GFLOPS(ピーク 4710GFLOPS の 9%)，逆変換 2210GFLOPS(ピーク 4710GFLOPS の 47%)を達成した。全体の計算では概算で 781GFLOPS(ピークの 16%)に到達した。最後に 1000 コア以上での性能評価を実施した。各ノードで実施した最大規模の問題とその時の性能を次の表に記す。

ノード数 (コア数)	最大次元	ハウスホルダー順変換	ハウスホルダー逆変換	全体 (含む分割統治法)
128 (2048)	128000	1077GFLOPS (5.7%) 2594 秒	7562GFLOPS (40%) 554 秒	3298GFLOPS (17.5%) 3298 秒
256 (4096)	256000	2868GFLOPS (7.6%) 7799 秒	15373GFLOPS (40%) 2182 秒	5536GFLOPS (14%) 10241 秒
512 (8192)	192000	2393GFLOPS (3.1%) 3942 秒	24834GFLOPS (33%) 570 秒	5025GFLOPS (6.7%) 5025 秒

いずれの計算においても，全体でテラフロップスを記録できたが，512 ノード使用時が最も性能が悪い結果となっている。データの詳細を解析すると，通信時間の比率が 256 ノード以上に大きくなっていることが分かった。結論としては 512 ノードでは問題サイズが小さすぎたために十分な性能を活かしきれなかったと見ている。また，本結果は十分なチューニングを施しているわけではなく，十分な性能向上の余地を残したものであくまでも短期間での移植作業による結果である。したがって，固有値計算においては少なくともこれ以上の性能が見積もれる「指標」として考えてもらいたい。

４．全対角化法の T2K 上への移植と発生した問題点

本章では，2，3 章にて書くことのできなかった全対角化コードのテストで出会った事象（発生した問題点や解決策及びその他の出来事について）について，今後，新たに T2K 利用を始める方々に何らかの役に立つと考え，時系列に従い散文形式にて記した。

1. まず，今回のプロジェクトでは，移植作業の時間を多くとれなかったためインテルコンパイラは使わず日立コンパイラのみでの作業であったことを記す。

2. 移植に際して，コンパイル・リンクについて問題はなかったが，実際の動作において ScaLAPACK が複数ノード利用＋あるサイズ以上の問題でハングアップしてしまった。ScaLAPACK を実際にソースから構築し直し，ハングアップ部分を特定しようと試みたが，BLACS 通信関数でハングする傾向があることが分かったため，BLACS の build 時に SENDIS = -DSndIsLocBlk を有効にして build することでハングの傾向は減った。ただし，完全に減ったわけではなく ScaLAPACK の関数で BLACS により複数の送受信を同時に行っている部分を MPI のノンブロッキング通信関数に書き換え一時的にハングを解消した。しかし，それでもハイブリッド実行時には完全ではなかった。

3. ScaLAPACK ルーチンの性能は BLAS で決定するため，GotoBLAS を導入することで対応した。

4. MPI の交換型の通信パターンの際にハングすることがあったため，Isend->Recv または Irecv->Send を利用する代わりに Irecv->Isend->Waitall に変更した。

5. ハイブリッド実装の際，日立製コンパイラにてコンパイルした場合に実行時予期しない数のプロセスが立ち上がり，性能が全くでないという事態に遭遇した。マニュアル等を調べ実行時に -F'prunst(threadnum(4))' を指定すればよいことが分かり性能は改善したが，後で，調べた結果基盤センターの Web ページにも別の解決方法が示されていることが分かった。

次に，実際にプログラムを動作しパフォーマンス測定にまで至った作業について，何が問題となり，それらをどのように解決し，何を達成できたかを以下に示す。

1. Altix システム向けに開発したコードを HA8000 に移植：
   自作コード部分は特に問題なく移植完了。ScaLAPACK 部分は先に述べたとおり。

2. 1 コアでの動作検証：
   バッチ環境でのチューニング経験があまりなかったため，Intel Xeon プロセッサでチューニングしたものをもとにわずかなチューニング作業を行い，1 コア単体でハウスホルダー順変換 2.39GFLOPS，ハウスホルダー逆変換 2.94GFLOPS を記録した。逆変換部分はチューニングの余地は大きくあったが作業をこの程度に止めた。

3. 同一ノード内複数コアによる flat-MPI 実装での動作検証：
   1 ノード内の 16 コア使用による flat-MPI 実行での動作を検証し，1000 次元から 32000 次元まで問題なく動作することを確認した。性能は 1 コア時からチューニングを行い，32000 次元の計算でハウスホルダー変換 16.51GFLOPS，逆変換 60.55GFLOPS を記録した。逆変換は 1 ノードの理論ピーク(147.2GFLOPS)の 41.1%であり，性能面ではかなりよい成績を示している。

4. 複数ノード間複数コアによる flat-MPI 実装での動作検証：
   HPC プロジェクトでは通常 128 ノードのクラスが開放されていたが，通常の試験では 32 ノードまでを利用して動作検証を行った。結論としては 32 ノードまでの２べきノード利用では全く問題なく動作した。それ以外は検証していないが，地球シミュレータ等での経験から問題なく動作するものと判断する。性能は，32 ノード使用時 96000 次元でハウスホルダー順変換 428.35GFLOPS(ピーク 4710GFLOPS の 9%)　逆変換 2210GFLOPS(ピーク 4710GFLOPS の 47%)を達成した。全体の計算では概算で 781GFLOPS(ピークの 16%)に到達した。

5. 1000 コア以上での性能評価：
   128 ノード，256 ノード，512 ノード使用の機会を頂けたため，限られた時間ではあるが 4. の設定をそのまま使い大規模ノードでの実測を試みた。各ノードで実施した最大規模の問題とその時の性能については，3 章の表を参照されたし。いずれも計算全体でテラフロップスを記録できたが，512 ノード使用時が最も性能が悪い結果となっている。データの詳細を解析すると，通信時間の比率が 256 ノード以上に大きくなっている。具体的には順変換の 256 ノード(256K 次元)では通信 2557 秒，全体が 7799 秒であるので通信が 32.7%に相当する。512 ノード(192K 次元)では通信 2652 秒，全体で 3942 秒なので通信は 67.2%になり 512 ノードでは 256 ノードの倍以上の時間プロセッサがアイドル状態でいたことになる（図６に 256 ノードのときのみであるが実行ログを掲載する)。結論としては 512 ノードでは問題サイズが小さ過ぎたために十分な性能を活かしきれなかったということになる。また，通信性能が仮に 2 倍になっても本質はレイテンシにあるため，性能の改善はわずかなものになると予想される。測定データが少ないためスケーラビリティの議論はできないのだが，512 ノードで 360000 次元の測定が可能であれば全体で 10TFLOPS(ピーク 1/6)が達成できた可能性がある。ただし実行時間は 4 時間にも達するため、本プロジェクトのような

制限時間付ベンチマークとしては不適切になる（これは，来る次世代計算機にて問題になりそうである。次世代計算機で最大の計算サイズにてベンチマークした場合、結果が出るまでの時間を見積もると大変な占有時間となってしまうだろう)。

```
MATRIX_SIZE=          256000
NUM.OF.PROCESS=            4096 (              64              64 )
 calc (u,beta)          465.143042325973511
 mat-vec (Au)            6092.16216659545898           1835.93449432374246
   COMM1/2                781.987825393676758            730.568418025970459
 2update (A-uv-vu)      614.600288152694702           18198.5119146053039
 calc v                 187.255089759826660
 v=v-(UV+VU)u            428.345677375793457
 UV post reduction      1.29367899894714355
 COMM_STAT
    BCAST   ::       535.078449249267578
    REDUCE ::        1960.95371365547180
    REDIST ::      0.000000000000000000E+000
    GATHER ::        62.2501411437988281
TRD-BLK         256000        7799.53520894050598             2868.07107527270637         GFLOPS
D&C     259.870339870452881          ERRCODE=                0
TRDBAK-WY          256000
TRBAK=      2182.48651695251465
COMM=      424.347877740859985
          15374.4051747239555          GFLOPS
          18818.0385140817889          GFLOPS
          19674.9351032307859          GFLOPS
COMM_STAT
    BCAST   ::       120.342360734939575
    REDUCE ::        303.681238889694214
    REDIST ::      0.000000000000000000E+000
    GATHER ::      0.000000000000000000E+000
TRDBAK          256000     2182.56937813758850           15373.8210510624122          GFLOPS
```

図6.　256 ノードを使用して 256K 次元の全対角化実行時のログ（上記ログは内部の主要ルーチンや通信コストを示しているが詳細は省略する。また，３章で示した全体の FLOPS は(10/3N^3)/(全体の経過時間)で算出したものである。）

6.　MPI と OpenMP を併用したハイブリッド実装での動作検証：

ハイブリッド実装（日立コンパイラ）での実行は残念ながら 1. の問題が解決されなかったため（理由は不明），自前のルーチンのみ実行できることを確認するのみに終わった。通信が 1/4 以下になるため，性能面での改善は期待できたのであるが誠に残念である。こちらの Xeon クラスタでは問題なく動作しており，ScaLAPACK の問題がなければ，ハイブリッド版により計算全体でピーク性能の 20%は達成できたかも知れない。

５．本 HPC 特別プロジェクトのまとめと今後の展望

以上が本 HPC プロジェクトにおいて実際に行った作業，そして得られた知見だが，我々は，これまで，地球シミュレータなどのベクトル計算機を中心に固有値ソルバーの開発を行ってきたため，性能面，特に対ピーク性能で 10%を切るようなルーチンに遭遇するのは初めての経験であった（具体的には，全対角化のハウスホルダー順変換がその際たるものである)。厳密対角化では，抜本的アルゴリズムの改変が必要であることが判明した一方，全対角化については，

予想通りのパフォーマンスをほぼ得ることが出来たと考えている（ハウスホルダー逆変換で最大 40%の効率である)。また，問題点等については学会等ですでに議論されており，解決策の目星は大体ついている。今後はその解決策を取り込んだ新しいアルゴリズムの実装と評価を行うこと，さらには，今回実施できなかったハイブリッド実装の性能評価を行うことが目標となる。尚，本プロジェクトの範囲で，当初に密度行列繰り込み群法の移植や性能評価を予定したが，時間の都合で実行できなかった。その後，T2K の有償利用により，その計画を実行していることを付記する。

最後に，東京大学情報基盤センターの皆様には大変御世話になりました。厚く御礼申し上げます。今後もこうした機会がございましたら何卒，宜しくお願いいたします。

参 考 文 献

{１} I. Bloch, J. Dalibard, and W. Zwerger, Many-body physics with ultracold gases, Rev. Mod. Phys. **80**, 885 (2008).

{２} M. Lewenstein, A. Sanpera, V. Ahufinger, B. Damski, A. Sen(De), and U. Sen, Ultracold atomic gases in optical lattices: mimicking condensed matter physics and beyond, Adv. Phys. **56**, 243 (2007).

{３} E. Dagotto, Correlated electrons in high-temperature superconductors, Rev. Mod. Phys. **66**, 763 (1994).

{４} Y. Yanase, T. Jujo, T. Nomura, H. Ikeda, T. Hotta and K. Yamada, Theory of Superconductivity in Strongly Correlated Electron Systems, Phys. Rep. **387**, 1 (2003).

{５} P.A. Lee, N. Nagaosa, and X.-G. Wen, Doping a Mott insulator: Physics of high-temperature superconductivity, Rev. Mod. Phys. **78**, 17 (2006).

{６} J.R. Schrieffer and J.S. Brooks (ed), *Hand book of high-temperature superconductivity* (Springer, New York, 2007).

{７} Y. Kamihara, T. Watanabe, M. Hirano, and H. Hosono, Iron-Based Layered Superconductor La[$O_{1-x}F_x$]FeAs (x=0.05-0.12) with T_c=26K, J. Am. Chem. Soc. **130**, 3296 (2008).

{８} H.T. Diep (ed), *Frustrated Spin System* (World Scientific, Singapore, 2004).

{９} P.A. Lee and T.V. Ramakrishnan, Disordered electronic systems, Rev. Mod. Phys. **57**, 287 (1985).

{１０} F. Evers and A.D. Mirlin, Anderson transitions, Rev. Mod. Phys. **80**, 1355 (2008).

{１１} S. Yamada, T. Imamura, and M. Machida, 16.447 TFlops and 159-Billiondimensional Exact-diagonalization for Trapped Fermion-Hubbard Model on the Earth Simulator, Proc. of SC05 (2005).

{１２} S. Yamada, T. Imamura, T. Kano, and M. Machida, High-Performance Computing for Exact Numerical Approaches to Quantum Many-Body Problems on the Earth Simulator,

Proc. of SC06 (2006).
{１３} M. Machida, S. Yamada, Y. Ohashi, and H. Matsumoto, Novel Superfluidity in a Trapped Gas of Fermi Atoms with Repulsive Interaction Loaded on an Optical Lattice, Phys. Rev. Lett. **93**, 200402 (2004).
{１４}M. Machida, S. Yamada, Y. Ohashi, and H. Matsumoto, On-site pairing and microscopic inhomogeneneity in confined lattice fermion systems, Phys. Rev. A **74**, 053621 (2006).
{１５} S. Yamada, M. Machida, Y. Ohashi, and H. Matsumoto, Strong pairing and microscopic inhomogeneity of lattice fermion systems, Physica C: Superconductivity and its applications, **463**, 103 (2007).
{１６} S. Yamada, M. Machida, T. Kano, T. Imamura, and T. Koyama, On-site pairing interaction and quantum coherence in strongly correlated systems, Journal of Physics and Chemistry of Solids **69**, 3395 (2008).
{１７} S.R. White, Density matrix formulation for quantum renormalization groups, Phys. Rev. Lett. **69**, 2863 (1992).
{１８} S.R. White, Density-matrix algorithms for quantum renormalization groups, Phys. Rev. B **48**, 10345 (1993).
{１９} U. Schollwoeck, The density-matrix renormalization group, Rev. Mod. Phys. **77**, 259 (2005).
{２０} K.A. Hallberg, New trends in density matrix renormalization, Adv. Phys. **55**, 477 (2006).
{２１} S. Yamada, M. Okumura, and M. Machida, Direct Extension of Density-Matrix Renormalization Group toward 2-Dimensional Quantum Lattice Systems: Studies for Parallel Algorithm, Accuracy, and Performance, arXiv:0707.0159.
{２２} S. Yamada, M. Okumura, and M. Machida, High Performance Computing for Eigenvalue Solver in Density-Matrix Renormalization Group Method: Parallelization of the Hamiltonian matrix-vector multiplication,Proceedings of VECPAR’08, 448 (2008).
{２３} T. Yamashita, N Kawakami and M. Yamashita, Fermionic Atoms Trapped in One-Dimensional Optical Superlattice with Harmonic Confinement, Phys. Rev. A **74**, 063642 (2006).
{２４} M. Tezuka and M. Ueda, Density-Matrix Renormalization Group Study of Trapped Imbalanced Fermi Condensates, Phys. Rev. Lett. **100**, 110403 (2008).
{２５} M. Machida, M. Okumura, S. Yamada, Stripe formation in fermionic atoms on a two-dimensional optical lattice inside a box trap: Density-matrix renormalization-group studies for the repulsive Hubbard model with open boundary conditions, Phys. Rev. A **77**, 033619 (2008).
{２６}M. Machida, S. Yamada, M. Okumura, Y. Ohashi, and H. Matsumoto, Correlation effects on atom-density profiles of one- and two-dimensional polarized atomic Fermi gases loaded on an optical lattice, Phys. Rev. A **77**, 053614 (2008).
{２７} M. Okumura, S. Yamada, N. Taniguchi, and M. Machida, Hole Localization in Doped

One-Dimensional Anderson-Hubbard Model, Phys. Rev. Lett. **101**, 016407 (2008).

{２８} M. Machida, M. Okumura, S. Yamada, T. Deguchi, Y. Ohashi, and H. Matsumoto, Mott phase in polarized two-component atomic Fermi lattice gas, Phys. Rev. B **78**, 235117 (2008).

{２９} http://www.jamstec.go.jp/esc/index.html

{３０} 山田進, 今村俊幸, 町田昌彦,量子大規模固有値問題における共役勾配法の収束性：適応的シフト前処理の収束性の評価, 日本計算工学会論文集 Vol.2006, 論文番号 20060027, (2006).

{３１} 山田進, 今村俊幸, 町田昌彦, 荒川忠一, 共有分散メモリ型並列計算機における新規通信手法, 日本計算工学会論文集 2005 年号, 論文番号 20050010,

{３２} 山田進, 今村俊幸, 町田昌彦, 量子多体問題における自由度の壁とそれを超える並列対角化アルゴリズムの開発：地球シミュレータ上での超並列量子計算の現状, 数理解析研究所講究録 1573, 53 (2007).

プロセッサアフィニティ制御を組み込んだフレームワークによる実用大規模並列シミュレータの性能評価

小 野 謙 二・野 中 丈 士
理化学研究所 次世代計算科学研究開発プログラム 次世代生命体統合シミュレーション
生命体基盤ソフトウェア開発・高度化チーム

１．はじめに

大規模分散並列環境において，研究や設計に役立つ様々な物理現象のシミュレータ開発を効率化するため，アプリケーションミドルウェアと可視化システムを開発している．本プロジェクトでは，ミドルウェアを用いたベンチマークプログラムと並列可視化時の画像重畳性能について，T2K のマルチコア超並列環境において，FlatMPI，ノード内の NUMA アーキテクチャの OpenMP/MPI Hybrid，およびノード内のプロセッサアフィニティ制御を考慮した性能評価を実施し，大規模計算機利用技術の蓄積を図った．

２．研究計画と経過

大規模分散並列環境において，研究や設計に役立つ様々な物理現象のシミュレータ開発を効率化し，そのメンテナンスや運用を円滑にすることを目的として，連成解析や可視化処理も含めたアプリケーションミドルウェアを開発している．このミドルウェアは並列処理，並列入出力，線形ライブラリなどの機能をもち，開発者は最適化されたそれらの機能を利用して高性能なコードを構築する．このようなフレームワーク型のアプリ開発では，提供するモジュール自体の並列性能が非常に重要となる．大規模システムは MPP かつマルチコア化が一つの方向であり，アーキテクチャを考慮した高性能化が望まれる．このような背景のもと，ノード内の NUMA アーキテクチャの共有メモリ並列とノード間の MPI 並列のハイブリッド化，およびノード内のプロセッサアフィニティ制御を考慮した高性能化を行い，その性能を評価すること，および大規模計算機利用技術の蓄積を図ることを本プロジェクトの目的とする．

評価対象として以下の３つを予定した．

① 科学技術計算に多く現れる Poisson 方程式の求解ベンチマーク
② 三次元非圧縮熱流体の実用コード
③ 可視化処理における画像重畳コード

Poisson カーネルについては，これまで理研の計算機資源（RSCC，BG/L）等を用いて 1024 コア規模の性能評価を実施してきた．BG/L 上では現状で 95%の並列化効率である．実用コードは Altix 上では理論ピーク性能の 20%を超える性能を確認しており，実行効率のよいコードである．また，画像重畳コードは，BG/L1024 コア上で理論通信性能の 17%程度，11.5fps のインタラクティブ性能であることを確認している．

本プロジェクトでの目標として，項目①②については，フラット MPI，NUMA チューニング，ハイブリッド並列の評価を実施し，スケーラビリティを確認する．プロジェクトの進め方として，まず，7 月に単体性能の確認とチューニングとフラット MPI 性能測定，8 月に NUMA チューニング，9 月にハイブリッド並列性能測定の順で実施する．項目③については，7 月に単体性能

確認とフラット MPI での性能測定，8 月にマルチステージ Binary-Swap(MSBS)法の性能測定と NUMA 向けアルゴリズム変更とチューニング,9 月に MSBS 法のハイブリッド並列版の性能測定を実施する予定である．

上記の計画に対し，割り当てられた CPU 時間枠を使ったが，全ての項目についてデータをとることはできなかった．最終的に①と③についての結果を得た．本報告では主に③について報告を行う．

３．可視化における画像重畳コードの性能評価

可視化は数値計算データや測定データから有用な情報を抽出できる有効な手段として広く認められている．しかしながら，近年の計測装置や計算機システムの目覚しい技術の発達により可視化の対象となる 3 次元データ（以下、ボリューム・データ）は複雑化，大規模化の一途をたどっている．また，計算科学分野においては並列計算機上で生成される分散データも対象となる．近年，ハイ・パフォーマンス・コンピューティング（以下、HPC）分野では超並列計算機が主流となりつつあり，この問題が深刻化する傾向にある．HPCシステムの性能ランキング「Top500」[(1)] でも数千プロセッサ・コアの超並列計算機は珍しくなく，最高峰のシステムでは数万，数十万プロセッサ・コアを有する．国内でもT2Kオープン・スパコン[(2)] に代表されるように数千プロセッサ・コアのシステムも珍しくなくなってきた．この様な超並列計算機環境では分散データの保存や転送等の問題を軽減するため，この計算機自身を大規模データ可視化のプラットフォームとして利用することが注目されている[(3)～(8)]．

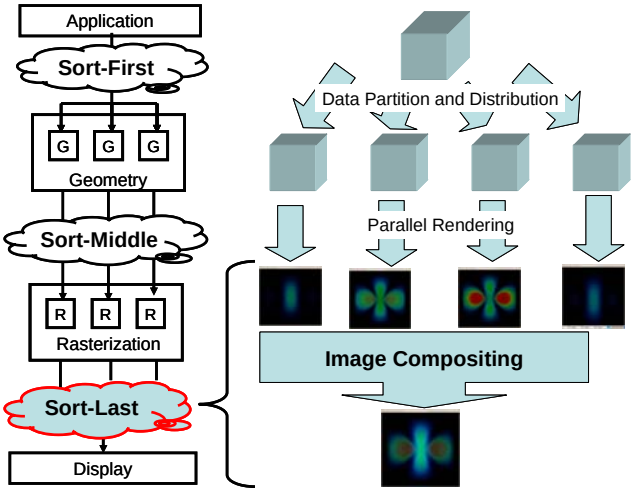


図 1　並列レンダリング方式

ボリューム・データの可視化手法は数多く存在する[(9)]が，その中でもボリューム・レンダリング[(10)] はボリューム・データの全領域を直接可視化することのできる手法として流体力学を含む多くの分野で広く用いられている．この手法は多大な演算量を必要とするが計算機能力の向上や並列処理の適用により身近な可視化手法になった．ボリューム・レンダリングの研究は盛んに行なわれ様々な手法が提案されてきた[(9), (10)] ．並列処理を適用した並列レンダリングの研究も盛んに行なわれ数多くの手法が提案されている．しかしながら，これらの手法の多くは

並列計算機ハードウェアへの依存度が高いことも挙げられる．Molnar [11] はソーティング処理に注目しこれらの手法を三グループに分類した：Sort-first，Sort-middle，とSort-last（以下、ソート・ラスト）式並列レンダリング．この中で，ソート･ラスト式（図 1）は計算機ハードウェアへの依存度が低く，尚且つデータ分割や負荷分散の面から注目を集め広く利用されてきた．

ソート･ラスト式では各レンダリング･ノードで出力されるレンダリング結果（2 次元画像）を最終的に一つの画像として合成する処理を行なう．この合成処理では用いたボリューム･データ分割法や視線情報から算出される奥行き情報を考慮し，用いたレンダリング手法に忠実な画像合成を行なう．この画像合成処理はImage Compositingや画像重畳とも呼ばれ，本報告では画像重畳として扱う．ボリューム・レンダリングでは不透明度（以下，アルファ値）を有する画像データが出力されるので通常Over [12] オペレーションと呼ばれる合成処理が適用される．これは遠い画像から順にアルファ・ブレンディング処理を施す手法である．

画像重畳処理は分散しているレンダリング結果を収集する必要があるため必然的にレンダリング・ノード間の通信が必要となり並列レンダリングのボトルネックとなりえる．この通信問題を軽減するためこれまでに様々な並列画像重畳手法が提案されてきた．これらはDirect-Send [13],[14]，Parallel Pipeline [15]，Binary-Swap [16] を含むBinary-Treeの三手法に分類できる．この三手法は小中規模並列計算機では有効性が認められ広く利用されてきたが，プロセッサ・コアの著しい増加が見受けられる近年の超並列計算機上での有効性は殆ど確認されていない [3]．本報告では，この様な超並列計算機上での並列画像重畳について考察する．

並列画像重畳

画像重畳処理はレンダリング画像を構成する最小単位であるピクセルに対する処理（per-pixel operation）が行なわれる．ピクセルは通常，RGB（赤、緑、青）の色成分の他，アルファ値（A）や奥行き情報（Z）を有する．これらの成分は利用用途別に様々な情報解像度や組み合わせが用いられる．本報告では一般的である 32 ビット RGBA ピクセルに注目した．これは，三原色情報と不透明度を各 8 ビットの情報量で表したものである．画像重畳処理では各ピクセルに対し奥行き情報を基に前後関係を考慮したアルファ･ブレンディング処理が行なわれる．式 1 に示されるアルファ･ブレンディング処理は単純な演算処理であるが全ピクセルの各色成分に対して行なう必要があるため，多大な演算処理が必要となる．また，アルファ値（A）は 8 ビットの情報量で表されるがアルファ･ブレンディング処理では浮動小数点型データに変換され処理される．アルファ･ブレンディング処理を高速にするため，計算精度を多少犠牲にした手法もいくつか提案されている．シフト演算を用いて 255 ではなく 256 で除算を行う手法や，丸め誤差を考慮しない LUT（Look Up Table）を利用する手法が挙げられる．

$$c = C_a + (1 - \alpha_a) C_b$$
$$\alpha = \alpha_a + \alpha_b (1 - \alpha_a) \qquad \text{(式 1)}$$

ここで c と α はそれぞれイメージの色と透明度を表す．

並列画像重畳処理はレンダリング・ノードに分散している画像を最終的に一つの画像として合成する処理である．考えられる一番単純な手法は一つの重畳ノードへ全ての画像を集め順次

重畳処理を行なう手法である．この手法はSequential手法とも呼ばれ，実装が簡単であるため小規模システムでは有効である．しかしながら，この手法では重畳ノード数に比例したデータ通信が発生するため超並列計算機向きではない．更に，一つしか存在しない重畳ノードがボトルネックになりえる． この問題を解消するため，同ノードでレンダリングと画像重畳処理を担当する手法が幾つか提案された． 図 2 に主流となっているDirect-Send [13]，Parallel Pipeline[14] [15]，Binary-Swap[16]を含むBinary-Tree手法を示す．

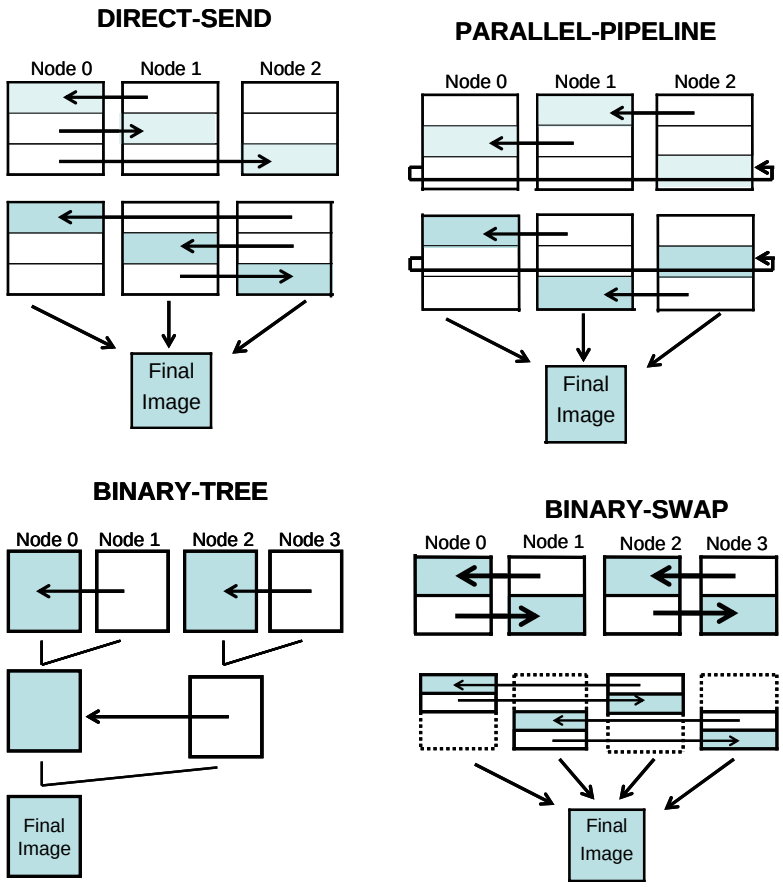


図 2　様々なイメージ重畳法

Direct-Send手法では画像はノード数(n)分に分割され，各重畳ノードはその内の一つのブロック（1 / nサイズ）を担当する．担当するブロックの部分画像を他のすべてのノードから受信しアルファ･ブレンディング処理を行なう．その為，この手法では各重畳ノードがワースト･ケースで（n － 1）の通信を必要とするため通信コストはn(n － 1）とみなされる．また，デッドロックや不必要な待ち時間を避けるためにデータ転送の順序を考慮しなければいけない.近年，同期回数を抑えた改良 [17] やアクティブ･ピクセル(有用な情報を保持しないバックグランド･ピクセル以外のピクセル）の重複度を考慮し，通信回数を削減する手法[18],[19] も提案されている．この中でも，Scheduled Linear image Compositing(SLIC)手法[19] はボリューム・データの 3 次元分割を考慮し,通信回数は分割数の最も多い辺に依存することに注目した方法である．ボリューム・データをn分割(各辺は$n^{1/3}$)した場合，ワースト･ケースで$n(n^{1/3})$の通信しか必要がない．しかしながら，この手法ではスケジューラー（データ転送順序）をアクティブ･ピクセルの重複度より算出する必要があるため超並列計算機では計算オーバーヘッドの課題が挙げられ

る.

Parallel Pipeline手法はDirect-Send手法を基にデッドロックが発生しないように通信順序を考慮した手法だといえる．従来のDirect Send手法と同じくワースト･ケースではn（n－1）の通信が発生する．この手法は実装が容易であることも含め小中規模並列計算機では有効だと考えられる．現在も商用並列可視化アプリケーションであるAVS Express PST（Parallel Support Toolkit）[20] やCEI Ensight DR（Distributed Rendering）[21] 等で利用されている．しかしながら，超並列計算機環境では通信がボトルネックになりえる可能性が挙げられる.

Binary-Tree手法では　二分木構造の走査を行なうため n($\log_2$n) の通信が発生する．Binary-Tree手法ではペアとなる重畳ノードの片方しか重畳処理を行なわないため,非効率的である．この問題に注目し，全ステージで全ノードを重畳ノードとして活用するBinary-Swap手法が提案された．通信コストが低く抑えられ，かつ計算ノードを効率よく利用するこの手法は最も広く利用され，研究された手法だといえる．この手法に対しは，数多くの改良が提案されている[22]~[26]．これらの多くは圧縮技術を含む通信データ・サイズの軽減を図る技術や重畳処理の負荷分散に注目したものである.これらの多くはBinary-Swap手法自体に変更を加えることなく，合わせて相乗効果を得る手法である.

図３にこれら三手法による各重畳ノードのデータ送信数を示す．1024 ノードを超える超並列計算機では Binary-Swap 手法が理論上優位にあると考えられる．本報告ではこの点に注目し超並列計算機上での Binary-Swap 式並列画像重畳の有効性を考察した.

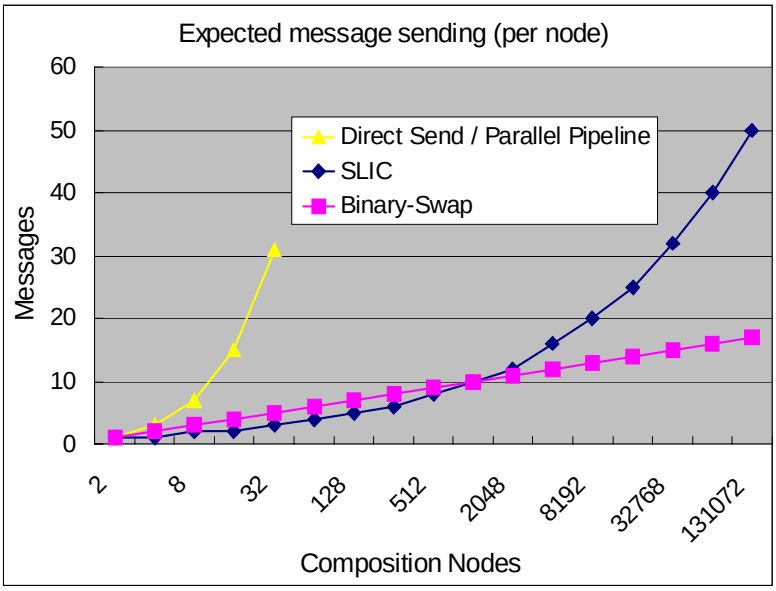


図３　画像重畳法の通信コストの比較

Binary-Swap 法

この手法は図４に示されるように各ステージにて重畳ノードペア間で画像の半分を交換し合い重畳処理を行なう．ステージ毎に担当する画像サイズが半減し，ステージ数（$\log_2$n）の最終ステージでは各重畳ノードに 1 / nサイズの重畳済み部分画像が存在することとなる．最終的にこれらの分散画像を収集し再構築することで最終重畳済み画像が出来上がる.MPI並列ライブラリを利用する場合,分散画像の収集にMPI_Gathervのような Collective関数の利用が考えられ

るが,MPIランク値の順にデータがルート･ノードへ集められるため最終構築作業が必要になる．画像を上下左右に交互分割した場合だけでなく上下方向のみに分割した場合も図４に示されるように画像再構築の作業が必要となる．超並列計算機環境ではこの最終工程がボトルネックとなる可能性が考えられる．最終画像は通常，表示のため表示装置を有する表示ノードへ送信するかポスト表示のためファイルに保存するかの処理が施される．

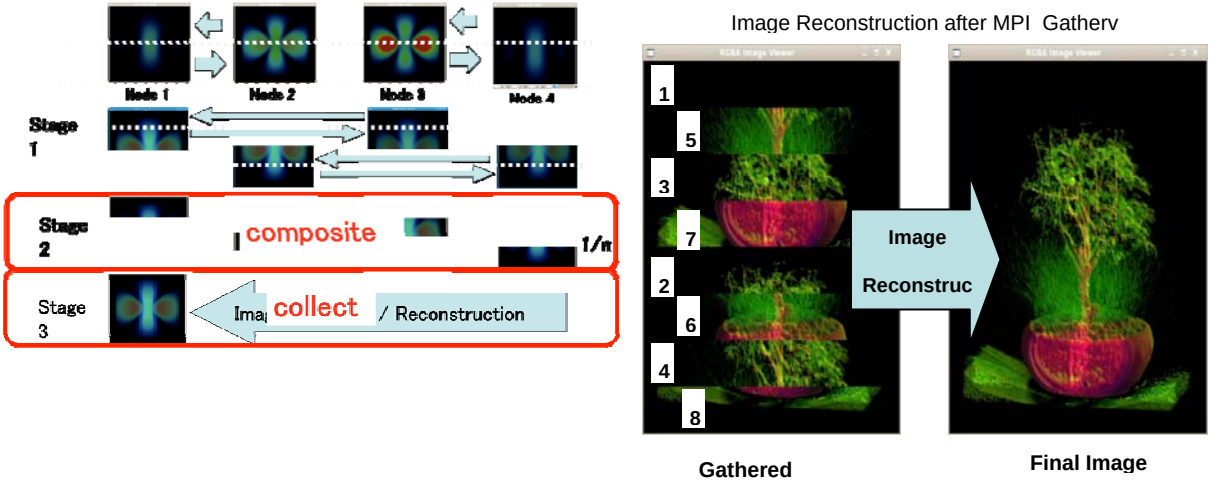


図 4 Binary-Swap イメージ重畳法のアルゴリズム

マルチ･ステップ画像重畳法

図 4 の Binary-Swap の擬似アルゴリズムから通信距離(ホップ数)はステージ毎に倍になることが伺える．これも最終ステージが近づくにつれ広域範囲でのデータ通信が行われることを表す．上記により Binary-Swap の最終ステージは性能低下を引き起こす可能性があると考えられる．本報告では予測される性能低下を軽減するため Binary-Swap の最終ステージに注目し，サブ･グルーピング化を行い Binary-Swap を多段階に分ける手法を提案する.複数ステップに分けて画像重畳が行なわれるためマルチ･ステップ画像重畳（Multi-Step Image Compositing）と名づけた．図 5 では画像重畳ノード（n）を重畳ノード(p）の数を持つ四つのサブ・グループに分けた例を表す．重畳ノード数(p)を持つ（m） のサブ・グループ（n = m ． p）に分けた場合，従来の Binary-Swap と比べ，分散画像収集および再構築のステージ数は増えるが重畳ステージ数は変わらないことが挙げられる．

Binary-Swap 式並列画像重畳は他の画像重畳手法同様にレンダリング･ノードが出力する 2 次元画像を入力データとして扱う．HPC 向けの超並列計算機では通常グラフィックス・ハードウェアを有しないため，レンダリング結果画像はメイン・メモリに既に保存されているため，読み込むみ作業を省く事ができる．Binary-Swap のデータ通信順序通りこれらの画像を重畳ノード間で交換し合い重畳処理を進めていく．各ステージで画像を半分に分割し交換し合うため最終ステージでは各重畳ノードは１ / n サイズの部分画像を交換する事となる．最終的に各重畳ノードに分散されている重畳済み部分画像を収集し再構築する作業が行なわれる．また，この最終画像は表示のため表示ノードへ送信されるか，ポスト表示のためにファイルへ書き込まれる処理が施される．これらの処理が Binary-Swap 手法が必要とする処理時間を決定する（式２）.

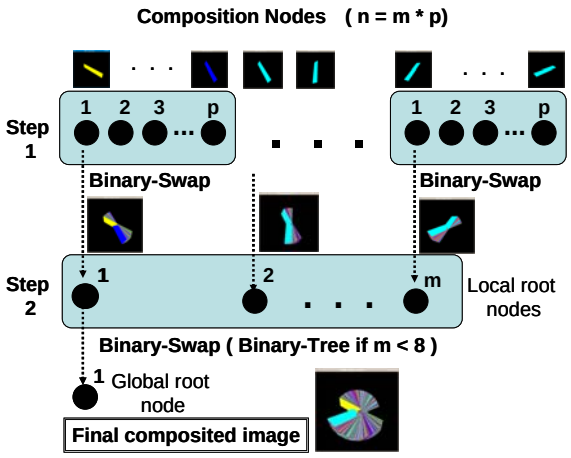


図５　Multi-Step イメージ重畳法のアルゴリズム

$$t_{total} = t_{read} + t_{compose} + t_{collect} + t_{write} \quad \text{(式 2)}$$

レンダリング･ノードからのデータ読み込み時間（ t_{read} ）はグラフィックス･ハードウェアを用いない超並列計算機では無視することができる．また，表示ノードへの画像送信やファイルへの書き込み時間（ t_{write} ）は今回性能評価の対象としていない．実際に表示されるまでの時間やファイルに書き出されるまでのタイム･ラグはいずれ考慮する必要があるが,本報告では連続的にどのぐらいのフレーム･レートで画像重畳が行なえるかという点に注目した．また，画像交換と重畳に要する時間($t_{compose}$)と重畳済みの分散部分画像の収集及び再構築時間($t_{collect}$)が最も処理時間を必要とするため，性能評価はこの二つに注目した．これら二つに注目した場合,処理コストを表す式2は式3の様に表すことができる.この式ではnは重畳ノード数を表す.また，$t_{compose}$は各ステージで画像データの送信(t_{send})および受信($t_{receive}$)，それとアルファ・ブレンディング処理(t_{blend})を行なう．双方向通信を可能とする近年の相互結合網では画像データの送信および受信はいずれかの最も時間の掛かる処理として表すことができる(t_{send_recv} = Max.{ $t_{receive}$, $t_{receive}$ })．これらの処理時間は画像サイズ(ピクセル数)およびピクセルサイズ（コンポーネント数、各コンポーネントのビット数）に存する．また，画像データの送受信時間は相互結合網のネットワークバンド幅やハードウェア遅延にも依存する．アルファ・ブレンディング処理は計算機の処理能力に直接依存する．

$$t_{BS} = \left(\sum_{i=1}^{\log_2 n} t_{compose_i} \right) + t_{collect_n} \quad \text{(式 3)}$$

並列画像重畳手法の性能予測は多くの文献[24]~[26]で紹介されている．　これらに沿って式３をBinary-Swapに当てはめると式４のように表すことができる.

式４では p_{xy} はレンダリング画像サイズを表す．これは画像の縦方向(I_y)および横方向(I_x)のサイズとピクセル・サイズ(pxl_{size})によって算出される．画像データの送受信時間(t_{send_recv})はハードウェア遅延($t_{net\_latency}$)および相互結合網のネットワークバンド幅($t_{net\_bandwidth}$)によって算出される．アルファ・ブレンディング処理時間(t_{blend})は計算処理遅

延($t_{cpu\_overhead}$)と計算処理能力($t_{cpu\_bandwidth}$)によって算出される．重畳済みの分散画像の収集および再構築処理時間(t_{gather})はIBM Blue Geneシステムの様にコレクティブ処理専用の相互結合網を持つハードウェアではハードウェア遅延は$t_{coll_net_latency}$，ネットワークバンド幅は $t_{coll\_net\_bandwidth}$ と表せる．通常の相互結合網であれば$t_{coll_net_latency}$および$t_{coll_net_bandwidth}$は通常通り$t_{net_latency}$と$t_{net_bandwidth}$で表される．

$$
\begin{aligned}
t_{BS} &= \left(\sum_{i=1}^{\log_2 n} t_{compose_i} \right) + t_{collect_n} \\
&= \left(\sum_{i=1}^{\log_2 n} \left(\frac{1}{2^i} p_{xy} \right) \left(t_{send\_recv_i} + t_{blend_i} \right) \right) + p_{xy} \left(t_{gather_n} \right) \\
&\cong p_{xy} \left[\left(1 - \frac{1}{n} \right) \left(t_{send\_recv} + t_{blend} \right) + t_{gather} \right] \\
p_{xy} &= I_x \times I_y \times pxl_{size} \\
t_{send\_recv} &= t_{net\_latency} + \frac{1}{net_{bandwidth}} \\
t_{blend} &= t_{cpu\_overhead} + \frac{1}{cpu_{bandwidth}} \\
t_{gather} &= t_{coll\_net\_latency} + \frac{1}{coll\_net_{bandwidth}} + t_{reconstruct}
\end{aligned}
\qquad \text{(式 4)}
$$

マルチ･ステップ画像重畳では重畳ノード数(p)を持つ（m） のサブ・グループ（n = m ． p）に分けた場合，式 5 のように表すことができる．

$$
\begin{aligned}
t_{MS} = \underset{[Blocks\ 1:m]}{Max} &\left\{ \left(\sum_{i=1}^{\log_2 p} t_{compose_i} \right) + t_{collect_p} \right\} + \\
&\left(\sum_{i=1}^{\log_2 m} t_{compose_i} \right) + t_{collect_m}
\end{aligned}
\qquad \text{(式 5)}
$$

この例では 2 段階のステップ数しか表していないが，千ノード･オーダーでサブ･グループを作成した場合，現在の超並列計算に十分対応できることが分かる．例えば，1024 の重畳ノードを一つのサブ・グループと考えた場合，最初のステップで各 1024 イメージを処理する 1024 の並行 Binary-Swap を行い，出力された 1024 の部分重畳済み画像を第 2 ステップで処理すれば 1,048,576 重畳ノード分を 2 段階で処理することに相当する．現在，もっともプロセッサ・コア数が多いHPC向けシステムが25万に満たないため2段階でも十分に処理ができると考えられる．この場合，サブ･グループのサイズを 512 としても十分処理できる事が分かる（ 512＊512 = 262,144 ）．

性能評価

Binary-Swap 式画像重畳処理の性能評価には 1024 ノードを超える二つの超並列計算機を利用した．東京大学情報基盤センターの日立 HA8000 クラスタ（以下，HA8000）と理化学研究所の IBM Blue Gene/L（以下，BG/L）．HA8000 は 2008 年 6 月版の Top500 ランキングでは 16 位にラ

ンキングされているスーパーコンピュータであり，BG/L はこのランキングで近年常に上位を占める Blue Gene シリーズのスーパーコンピュータである．

BG/L の最大の特徴はプロセッサ･チップを含むカスタム SoC（System on a Chip）チップと 3 次元トーラス相互結合網である．各プロセッサ・チップは二つの PowerPC 440　700MHz プロセッサ･コアと 512MB のメモリを持つ．このメモリはプロセッサ･コア間で共有されないのでピュアな分散メモリ型超並列計算機を実現している．相互結合網は point-to-point 通信用の 3 次元トーラスだけでなく，グローバル・ツリーとグローバル・バリアーも用意されている．グローバル・ツリーは MPI のコレクティブ通信に利用される．Blue Gene シリーズはスケーラビリティの高いシステムとして広く知られ，現在 212,992 プロセッサ･コアを有する Blue Gene/L スーパーコンピュータも存在する．今回，利用した BG/L は 2048 のプロセッサ･コアを有する．

HA8000 はT2Kオープンスーパーコンピュータ[(2)] の共通仕様に基づいた日立製の超並列計算機である．Quad-Core Opteron 2.3GHzを 4 個搭載する計算機ノードを 952 台持つ．これらのノードはMyrinet 10Gの相互結合網で接続されている．運営上，四つのクラスタ（512 台，256 台，128 台，56 台）に分割されているため最大で 8192 プロセッサ･コアまでしか同時利用する事ができない．もう一つの特徴として，各クラスタ内でフル・バイセクション・バンド幅が確保される方法でこれらのノードが接続されている事が挙げられる．今回，4096 プロセッサ･コアまでしか利用していない．

性能評価のため，32 ビット RGBA 画像を重畳するアプリケーションを作成した．Binary-Swap だけでなくマルチ･ステップ画像重畳向けに Binary-Tree 式画像重畳も実装した．これは重畳ノード数が少ない場合（例えば 8 以下），最終ステージ（分散重畳済み画像収集および再構築）が省けるため有利だと考えたからである．実装には C 言語と MPICH 通信ライブラリを用いた．画像重畳処理性能は対象となる画像に左右されるため，今回は画像重畳性能の下限値に注目した．そのため，アクティブとバックグランド･ピクセルの比率の差によるアルファ・ブレンディング計算処理時間の違いを防ぐためにフル･アクティブ･ピクセル画像を性能評価に用いた．また，画像圧縮等の高速化手法も一切利用していない．処理時間を計測するために MPI 関数の MPI_Wtime と MPI_Barrier を利用し，この処理時間より FPS（Frame per Second）を算出した．性能評価には多くの文献に利用されている 512x512 の画像サイズを用いた．

予測性能式（式 4）は画像重畳処理時間が重畳ノード数に対してどの様に変化するのか予測する目的で利用されている．ハードウェアの実性能ではなく理論性能を使用し，様々なファクタによって発生する処理オーバーヘッドを無視したものである．それにより，この予測性能はスケーラビリティ予測だけでなく重畳処理性能の上限値を表している．BG/L と HA8000 の予測性能は図 6 に示している．BG/L では 3 次元トーラス・ネットワークが双方向に 2.8Gbps で通信でき，ハードウェア遅延が 6μs としている．また，アルファ・ブレンディング処理が 0.7Gbps で行なえると想定した．HA8000 では Myrinet10G が 5Gbps で双方向通信でき，ハードウェア遅延が 8.5us としている．また，アルファ･ブレンディングの処理性能を 2.3Gbps とした．

BG/L と HA8000 上での Binary-Swap 式画像重畳の性能評価結果を図 6 に示す．この図から 512 ノードまで予測性能カーブ通りの挙動を確認できるが，それを超えたところから予測値から大きく外れていくことも確認できる．Binary-Swap の特徴である広範囲における通信が大きく寄与していると考えられる．これを考慮し マルチ･ステップ式ではサブ・グループのサイズを 512 と 1024 の二通りを試した．今回使用した重畳ノード数は最大で 4096 ノードしかないため，2

段階目では 8 重畳ノード以下となるため Binary-Tree を利用した.

マルチ･ステップ画像重畳の性能結果は図 7 に示す. この図からこの手法は超並列計算機上で画像重畳性能の性能低下を効率的に抑えることができる事が確認できる. BG/L ではサブ・グループが 512 ノードの場合,1024 重畳ノードの Binary-Swap＋Binary-Tree 画像重畳性能は 16FPS から 19FPS に向上した. また, 2048 重畳ノードでは 8FPS から 14FPS の性能向上がみられた. サブ・グループが 1024 ノードの場合, 2048 重畳ノードの Binary-Swap＋Binary-Tree 画像重畳性能は 8FPS から 12FPS に向上した. HA8000 ではサブ・グループが 512 ノードの場合, 4096 重畳ノードの Binary-Swap＋Binary-Tree 画像重畳性能は 12FPS から 27FPS に向上した. また, サブ・グループが 1024 ノードの場合, 12FPS から 41FPS に向上した. BG/L では 512 のサブ・グループ・サイズが最も良い結果を得られたが, HA8000 では 1024 を利用した場合であった.

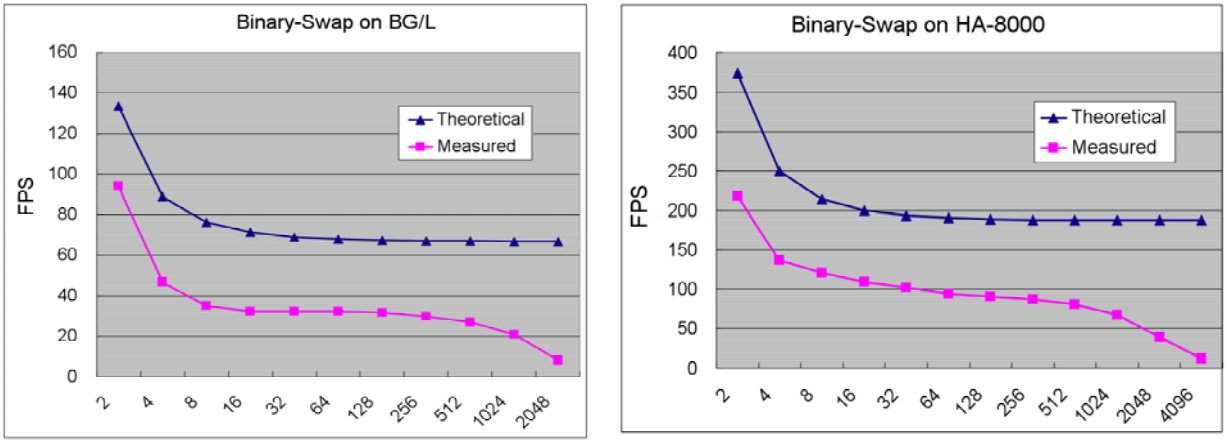


図 6　Binary-Swap 法によるイメージ重畳の性能

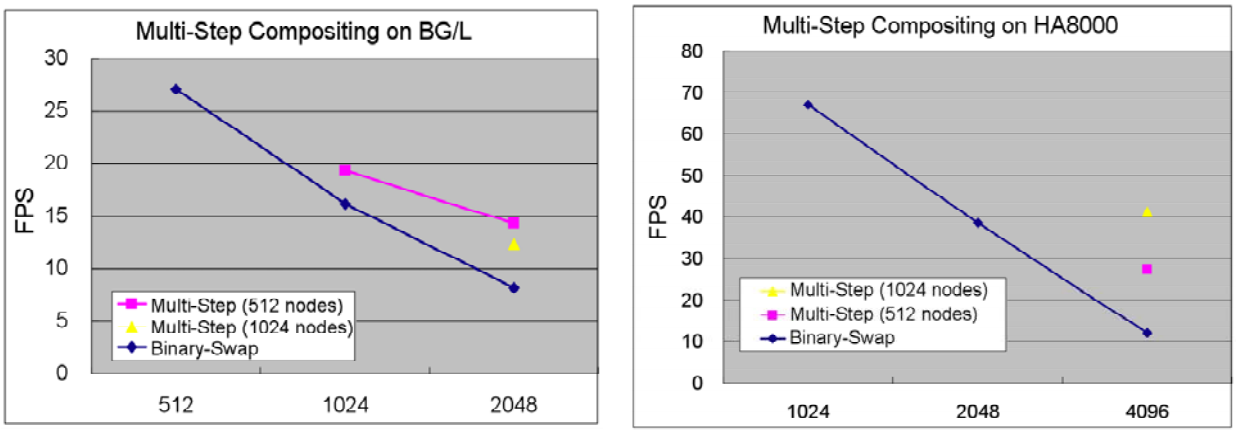


図 7　Multi-step 法によるイメージ重畳の性能改善

ハイブリッド性能

次に, マルチコア・アーキテクチャーの利点を活かした画像重畳について考える. オリジナルのBinary-Swapをはじめ多くの画像重畳アルゴリズムはハイブリッドプログラミングを念頭には考えられていない. 共有メモリについては, Shared-Memory Compositing(SMC)[30]が提案され

ている．SMCは図８において，最初の画像重畳に適用される．その後，ノード間にまたがりBinary-Swapにより更に重畳プロセスへと移る．

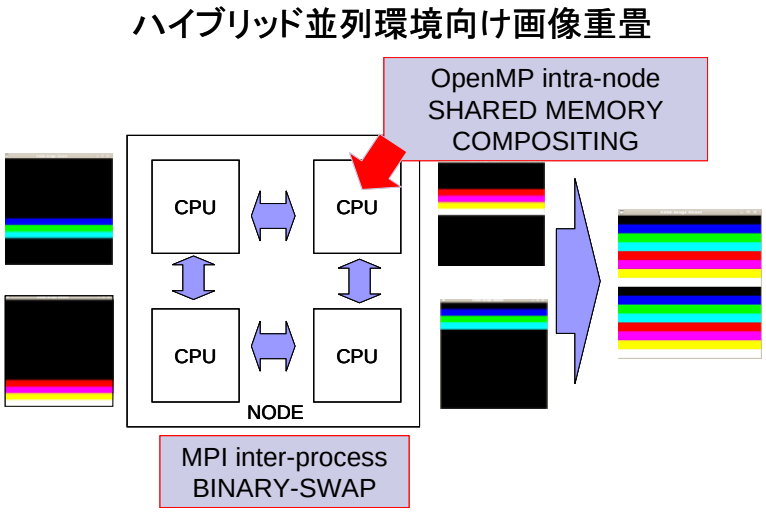


図８　共有メモリに対する画像重畳

Binary-Swap 式並列画像重畳は 2 のべき乗の重畳ノード数を必要とするため，性能評価には最低重畳ノード数を２とし最大利用可能ノード数(4096)まで２のべき乗毎に計測を行なった．表１に示す Flat-MPI 及び Hybrid 並列環境で以下の三画像重畳モードの性能評価を行った．

表１　性能測定パターン

	Flat-MPI	Hybrid MPI-OpenMP
BS	Binary-Swap	-
BS+BT	Binary-Swap + Binary-Tree	-
DS+BS	-	Direct-Send + Binary-Swap

コンパイルには日立製コンパイラを用い，表２のオプションを利用した．

表２　コンパイラオプション

Flat-MPI	Hybrid MPI-OpenMP
mpicc -Os -noparallel	mpicc -Os -parallel -omp

T2K オープンスパコンで利用可能の CPU-メモリ・アフィニティ効果を調査するため，表３のアフィニティ･モードを利用した．

表３　アフィニティモード

numa_1	numa_2
#!/bin/bash	#!/bin/bash
MYRANK=$MXMPI_ID	MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)	CPU=$(expr $MYRANK % 4)
CPU=$(expr $MYVAL % 4)	/usr/bin/numactl --cpunodebind=$CPU
/usr/bin/numactl --cpunodebind=$CPU	--membind=$CPU $@
--membind=$CPU $@	

バッチジョブ・スクリプトは表４のようなものを利用した(M256 キュー向け).

表４　バッチスクリプト

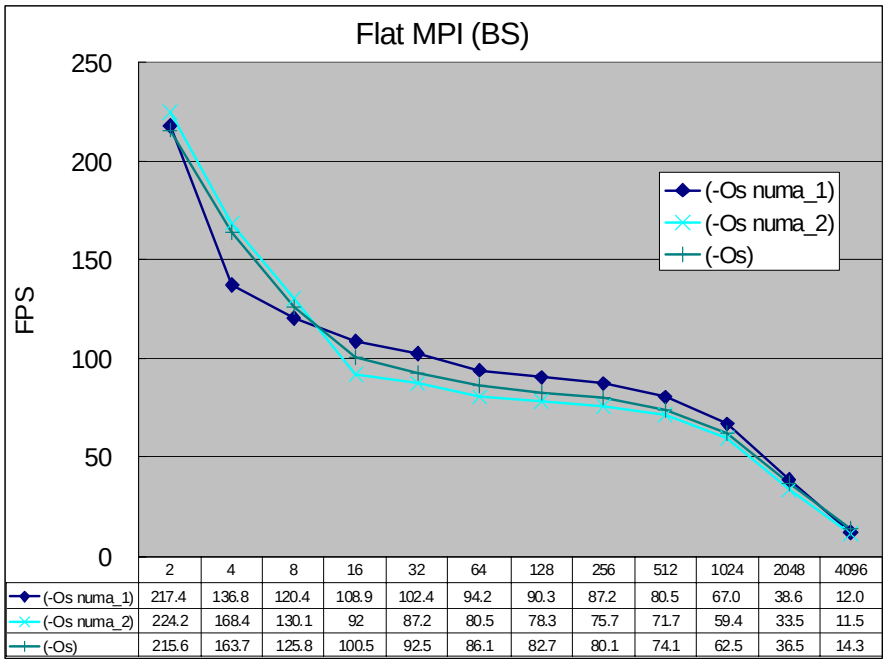
Flat-MPI	Hybrid MPI-OpenMP
#!/bin/sh	#!/bin/sh
#@$-r bswap	#@$-r bswap
#@$-q monthly	#@$-q monthly
#@$-N 256	#@$-N 256
#@$-J T16	#@$-J T4
#@$-lm 2gb	#@$-lm 7GB
#@$-lM 28GB	#@$-lM 28GB
#@$-lT 0:15:00	#@$-lT 0:15:00
#@$-e bswap.elog	#@$-e bswap.elog
#@$-o bswap.log	#@$-o bswap.log
#@$-lc 0MB	#@$-lc 0MB
#@$-nr	#@$-nr
#@$-s /bin/sh	#@$-s /bin/sh
#@$export MX_BONDING=4	#@$export MX_BONDING=4
cd $QSUB_WORKDIR	export OMP_NUM_THREADS=4
mpirun ./numa_1.sh ./bswap_flat	export HF_PRUNST_THREADNUM=4
	cd $QSUB_WORKDIR
	mpirun ./numa_2.sh ./bswap_hybrid

BS+BT のマルチ･ステップ並列画像重畳モードではサブグループ･サイズに 512 と 1024 重畳ノードを利用した.

4096 コア(重畳ノード)まで利用した計測を行なった. CPU-メモリ・アフィニティを利用しない場合と上記の２通り (numa_1,numa_2) 利用した三モードの性能評価を行った. 得られた実行処理時間から FPS (Frames per Second) を算出した結果は以下の通りである.

- Flat-MPI
  - Binary-Swap

図９に示すように Flat－MPI 並列環境での Binary-Swap は 512 重畳ノードを越えた辺りから著しい性能低下が見受けられた. CPU-メモリ・アフィニティを利用した場合, 良い結果が得られる事が確認できた.しかしながら重畳ノード数の増加に伴いその効果が見られなくなった.更に, 4096 ノードでは CPU-メモリ・アフィニティを利用しないモードよりも悪い結果となった.



	2	4	8	16	32	64	128	256	512	1024	2048	4096
(-Os numa_1)	217.4	136.8	120.4	108.9	102.4	94.2	90.3	87.2	80.5	67.0	38.6	12.0
(-Os numa_2)	224.2	168.4	130.1	92	87.2	80.5	78.3	75.7	71.7	59.4	33.5	11.5
(-Os)	215.6	163.7	125.8	100.5	92.5	86.1	82.7	80.1	74.1	62.5	36.5	14.3

図９　FLAT MPI におけるスケーラビリティ

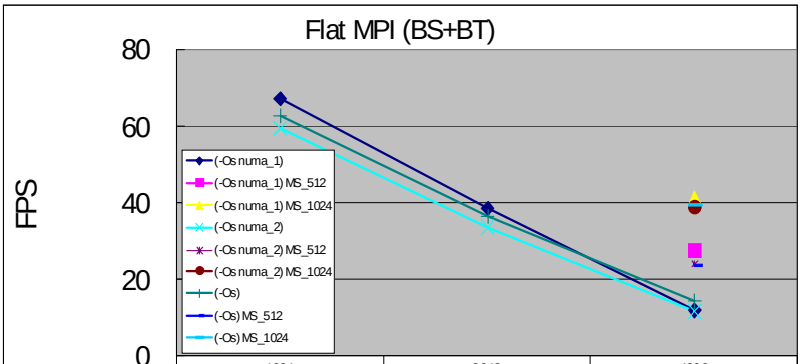
- Binary-Swap + Binary-Tree

4096 重畳ノードを用いたマルチ･ステップ並列画像重畳ではサブグループ･サイズに 512 と 1024 を用いた．サブグループ･サイズが 512 の場合，八つの 512 重畳ノードを用いた並行 Binary-Swap 式画像重畳が第一段階で行なわれ，得られた八つの画像の Binary-Tree 画像重畳が第二段階目で行なわれる．また，1024 の場合，四つの 512 重畳ノードを用いた並行 Binary-Swap 式画像重畳が第一段階で行なわれ，得られた四つの画像の Binary-Tree 画像重畳が第二段階目で行なわれる．

測定結果は図１０に示すように 512、1024 の両サブグループ・サイズを用いたマルチ･ステップ画像重畳を行なった方が 4096 重畳ノードの Binary-Swap を行なうよりも良い結果が得られた．また，512 よりも 1024 をサブグループ・サイズにした方がより良い結果が得られる事が確認できた．

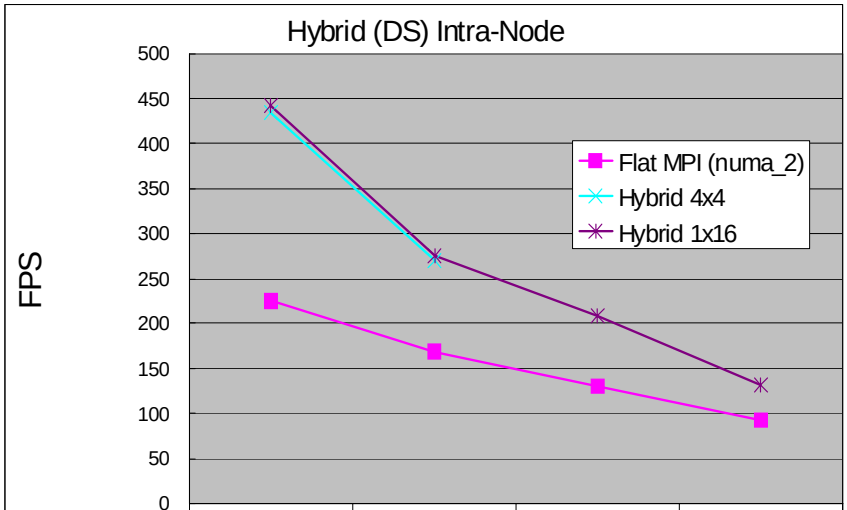
- Hybrid　MPI-OpenMP
  - Direct-Send + Binary-Swap

T2K オープンスパコンは 16CPU コアでメモリを共有するため 16 コアまでの共有メモリ型画像重畳 (Direct-Send) の性能評価を行った．4 コア (Hybrid 4x4) と 16 コア (Hybrid 1x16) での共有メモリを利用した際の性能評価は以下の通りである．両モードとも Flat－MPI より良い結果が得られた．共有メモリ Direct-Send の場合，Binary-Swap で必要とされる最終ステージ（重畳済み部分画像の収集および最終画像の構築）を省く事ができることより性能向上に貢献していると考えられる．



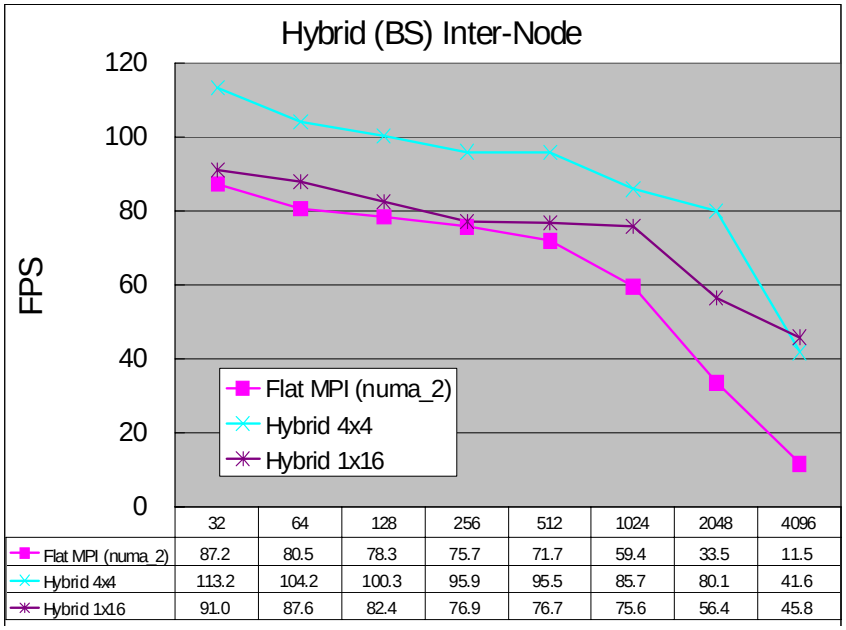
	1024	2048	4096
(-Os numa_1)	67.0	38.6	12.0
(-Os numa_1) MS_512			27.4
(-Os numa_1) MS_1024			41.3
(-Os numa_2)	59.4	33.5	11.5
(-Os numa_2) MS_512			24.0
(-Os numa_2) MS_1024			38.8
(-Os)	62.5	36.5	14.3
(-Os) MS_512			23.5
(-Os) MS_1024			39.4

図１０　Binary-Swap + Binary-Tree の測定結果



	2	4	8	16
Flat MPI (numa_2)	224.2	168.4	130.1	92.0
Hybrid 4x4	434.0	269.2		
Hybrid 1x16	441.5	275.0	208.0	131.1

図１１　Hybrid　MPI-OpenMP の測定結果



	32	64	128	256	512	1024	2048	4096
Flat MPI (numa_2)	87.2	80.5	78.3	75.7	71.7	59.4	33.5	11.5
Hybrid 4x4	113.2	104.2	100.3	95.9	95.5	85.7	80.1	41.6
Hybrid 1x16	91.0	87.6	82.4	76.9	76.7	75.6	56.4	45.8

図１２　共有メモリ重畳＋　Binary-Swap 重畳

共有メモリ Direct-Send 後,Binary-Swap 画像重畳を行なった結果を以下に示す.Hybrid モード 4 コア、16 コア・メモリ共有の両モードで Flat-MPI に比べ良い結果が得られた．4 コア･共有メモリ･モードの方が 16 コアよりも常に良い結果を得られていたが 4096 重畳ノードでは立場が逆転してしまった．これは Binary-Swap 重畳ノード数が大きく関係していると考えられる．このノード数では 4 コアの場合，1024 重畳ノードでの Binary-Swap 画像重畳が行なわれるが,16 コアでは 256 重畳ノードでの Binary-Swap 画像重畳が行なわれる．1x16 よりもでは 1024 重畳ノードから著しい性能低下が見受けられ，Hybrid4x4 では 2048 重畳ノードから性能低下が見受けられた.

まとめ

以上のように，T2K 上でのソート・ラスト式並列画像重畳性能について考察した．スケーラビリティの観点から Binary-Swap 式画像重畳手法に着目した．この手法は画像重畳処理が進むにつれ，データ通信が広範囲に広がっていく特長を持つ．これによってネットワーク上でのデータ衝突を招き，性能低下を引き起こすことが考えられる．実際に T2K 上でも，1024 ノードを超える超並列計算機上で Binary-Swap 式画像重畳処理の性能評価を行った際,千ノード･オーダーの重畳ノード数では性能低下が発生するのが確認できた．この問題に着目し，サブ・グルーピング化を用いて通信範囲を限定するマルチ･ステップ画像重畳手法を提案した.性能低下が発生する範囲の重畳ノード数をサブ･グループ･サイズとして利用した場合,マルチ･ステップ画像重畳手法は効果的に性能低下を制御できる手法であることを示した． Flat-MPI 並列環境では Binary-Swap + Binary-Tree を用いたマルチ･ステップ画像重畳の有効性が確認できた．また，Hybrid MPI-OpenMP 並列環境では共有メモリ Direct-Send + Binary-Swap 併用手法の有効性が確認できた．この提案手法は多段階に処理を行なうことから，更なる大規模な超並列計算機にも対応できる可能性を持つ.

本研究は,文部科学省 最先端・高性能汎用スーパーコンピュータの開発利用プロジェクト「次世代生命体統合シミュレーションの研究開発」の支援を受け行われたものである. また,「T2K オープンスパコン (東大) HPC 特別プロジェクト」によって実施した.

参 考 文 献

(1) Top500 Supercomputer sites: http://www.top500.org/

(2) T2K Open Supercomputer Alliance: http://www.open- supercomputer.org/.

(3) Ross, R., Peterka, T., Shen, H-W., Yu, H., Ma, K.-L., and Moreland, K., "Visualization and Parallel I/O at Extreme Scale," Journal of Physics: Conference Series 125, SciDAC (2008).

(4) Chen, L., Fujishiro, I., and Nakajima, K., "Optimizing Parallel Performance of Unstructured Volume Rendering for the Earth Simulator," Parallel Computing, 29 (2003), pp. 355-371.

(5) Tu, T., Yu, H., Ramirez-Guzman, L., Bielak, J., Ghattas, O., Ma, K.-L., and O'Hallaron, D. R., "From Mesh Generation to Scientific Visualization: An End-to-End Approach to Parallel Supercomputing," Proc. Supercomputing 2006, (2006).

(6) Ma, K. -L., Wang, C., Yu, H., and Tikhonova, A., "In-Situ Processing and Visualization for Ultrascale Simulations," Journal of Physics: Conference Series 78, SciDAC (2007).

(7) Peterka, T., Yu, H., Ross, R., and Ma, K.-L., "Parallel Volume Rendering on the IBM Blue Gene/P," Proc. Eurographics 2008 Symposium on Parallel Graphics and Visualization, (2008).

(8) Rao, A. R., Cecchi, G. A., and Magnasco, M. M., "High Performance Computing Environment for Multi dimensional Image Analysis," BMC Cell Biol. 2007; 8(Suppl): S9 (2007).

(9) Johnson, C. R., and Hansen, C. D., "Visualization Handbook," Academic Press (2004).

(10) Meissner, M., Huang. J., Bartz, D., Mueller, K. Crawfis, R., "A Practical Comparison of Popular Volume Rendering Algorithms," Proc. VolVis 2000, (2000) pp. 81-90.

(11) Molnar, S., Cox, M., Ellsworth, D., and Fuchs, H., "A Sorting Classification of Parallel Rendering," IEEE Computer Graphics and Applications, 14(4) (1994) , pp. 23-32.

(12) Porter, T., and Duff, T., "Compositing Digital Images," Computer Graphics (Proc. SIGGRAPH 1984), (1984) pp. 253-259.

(13) Hsu, W. M., "Segmented Ray-Casting for Data Parallel Volume Rendering," Proc 1993 Symposium on Parallel Rendering, (1993) pp. 7-14.

(14) Neumann, U., "Parallel Volume-Rendering Algorithm Performance on Mesh-Connected Multicomputers," Proc 1993 Symposium on Parallel Rendering, (1993) pp. 97-104.

(15) Lee, T.-Y., Rahavendra, C. S., Nicholas, J. B., "Image Composition Schemes for Sort-Last Polygon Rendering on 2D Mesh Multicomputers," IEEE Transactions on Visualization and Computer Graphics, 2(3) (1996), pp. 202-217.

(16) Ma, K. -L., Painter, J. S., Hansen, C. D., and Krogh, M. F., " Parallel Volume

Rendering using Binary-Swap Image Composition," IEEE Computer Graphics and Applications, 14(4) (1994), pp. 59-68.

(17) Strengert, M., Magallón, M., Weiskopf, D., Guthe, S., and Ertl T., "Hierarchical Visualization and Compression of Large Volume Datasets using GPU Clusters," Eurographics Symposium on Parallel Graphics and Visualization 2004, (2004) pp. 41-48.

(18) Eilemann, S. and Pajarola, R., "Direct Send Compositing for Parallel Sort-Last Rendering," Eurographics Symposium on Parallel Graphics and Visualization 2007, (2007).

(19) Stompel, A., Ma, K.L., Lum, E.B., Ahrens, J., and Patchett, J., "SLIC: Scheduled Linear Image Compositing for Parallel Volume Rendering," IEEE Symposium on Parallel and Large-Data Visualization and Graphics, (2003) pp. 33-40.

(20) Advanced Visual Systems: http://www.avs.com

(21) CEI Ensight: http://www.ensight.com

(22) Ahrens, J., and Painter, J., "Efficient Sort-Last Rendering using Compression-Based Image Compositing," Second Eurographics Workshop on Parallel Graphics and Visualization,(1998) pp. 33-40.

(23) Yang, D.L., Yu, J.C., and Chung, Y.C., "Efficient Compositing Methods for the Sort-Last-Sparse Parallel Volume Rendering Systems on Distributed Memory Multicomputers," Journal of Supercomputing, 18(2) (2001) pp. 201-220.

(24) Takeuchi, A., Ino, F., and Hagihara, K., "An Improved Binary-Swap Compositing for Sort-Last Parallel Rendering on Distributed Memory Multiprocessors," Parallel Computing, 29(11-12) (2003) pp. 1745-1762.

(25) Sano, K., Kobayashi, Y., and Nakamura, T., "Differential Coding Scheme for Efficient Parallel Image Composition on a PC Cluster System," Parallel Computing, 30(2) (2004) pp. 285-299.

(26) Lin, C.-H., Chung, Y. -C., and Yang, D. -L., "TRLE - An Efficient Data Compression Scheme for Image Composition of Volume Rendering on Distributed Memory Multicomputers," Journal of Supercomputing, 39(3) (2007) pp. 321-345.

(27) Reinhard. E., and Hansen, C., "A Comparison of Parallel Compositing Techniques on Shared Memory Architectures," Third Eurographics Workshop on Parallel Graphics and Visualization, (2000).

(28) Cavin, X., Mion, C., Fibois, A., "COTS Cluster-Based Sort-Last Rendering: Performance Evaluation and Pipelined Implementation," IEEE Visualization 2005, (2005) pp. 111-118.

(29) Tay, Y. C., "A Comparison of Pixel Complexity in Composition Techniques for Sort-Last Rendering," Journal of Parallel and Distributed Computing, 62 (2002) pp. 152-171.

(30) Reinhard. E., and Hansen, C.: A Comparison of Parallel Compositing Techniques on Shared Memory Architectures, *Third Eurographics Workshop on Parallel Graphics and Visualization*, Eurographics Organization (2000).

GXPシステムとそれを用いた大規模テキスト処理の実行

柴田知秀　姜ナウン　黒橋禎夫
京都大学大学院情報学研究科
田浦健次朗　Choi SungJun　Dun Nan　松崎拓也　辻井潤一
東京大学大学院情報理工学系研究科
河原大輔
情報通信研究機構
宇野毅明
国立情報学研究所

1　はじめに

本プロジェクトではのちに述べる二つの大規模な情報検索 (3 節), 情報抽出処理 (4 節) を実行した. それらのタスク自体の学術的な意義の他に, 本プロジェクトでは, HA8000 システムをデータ処理が中心のワークロードに適用すること, それを著者らが設計・実装した分散並列シェル GXP [6] [1]およびその上のワークフローシステム GXP make を用いて, 高い生産性で実行する事により, HA8000 の新しい適用分野を開拓する事を目指した.

本プロジェクトで実行した, データ処理中心のワークロードには一般に以下のような特徴がある.

1. 大量, 多数のデータの入出力を行う.
2. 処理時間がデータの性質に大きく依存し, 少数のパラメータからの計算量予想などがしにくい (稀なデータに対する実行時間やメモリ使用量などの「サプライズ」が頻繁に発生している).
3. 場合によっては, 新しいデータに対して処理が失敗することもあり, プログラムの修正と実行のサイクル (試行錯誤) を繰り返す必要がある.
4. ワークロード全体は, 複雑なデータ処理を行ういくつものプログラムを組み合わせて構築されており, 短いカーネルループの性能を向上させるような努力で, 全体性能が向上することはあまり期待できない. 性能を FLOPS 値で測ったり, ましてそれをマシンの最高理論 FLOPS 値に近づける, などといった評価基準を適用することは難しい.

[1] http://www.logos.t.u-tokyo.ac.jp/gxp/

全体として, プログラムの最高性能の向上を求めてプログラムをチューニングする努力は報われにくく, それよりもプログラムの生産性 (最初から失敗せずに並列化でき, 効果的な台数効果を得られること, 失敗の原因が容易に分かり, 並列化のためのプログラム統合に費やす時間が少ないこと) が重要である. また, 計算機の性能に対する要求としては, ファイルシステムの IO 性能がある程度高い (劣悪でない) ことが上げられる.

改めて, システム開発や評価に関連する本プロジェクトの目的を述べると以下のようになる.

1. GXP および, その上のワークフロー処理系 GXP make の HA8000 への移植を行い, HA8000 ユーザへ提供する準備を整えること.

2. GXP make のスケーラビリティを HA8000 規模 (数千〜15,000 並列度) のシステムで検証すること.

3. データ中心のワークロードに対する HA8000 の性能, 適用可能性を評価すること.

結果として, 本プロジェクトのために提供いただいた 8,192 CPU core 程度の並列性に対して, GXP make 自身は良好なスケーラビリティを有していることが確認されたが, HA8000 のファイルシステム HSFS の性能が非常に悪いために, 本プロジェクトで対象としたようなワークロードを効率的に実行するには, 現状の HA8000 には大きな問題があることが明らかになった.

もちろん, HSFS の性能が悪いことについて事前の情報と, 実験結果があったため, 我々はその条件下で, 極力 HSFS を経由せずにデータを受け渡すためのプログラムを開発するなど, できる限りの準備を行っていた. にもかかわらず残念ながら 8,192 CPU core の割り当て時間中には, 進行状況を監視するための数分おきのファイルアクセスを行うだけで, 計算がほとんど進行しない状態となってしまった. そのため本報告書で述べる実験結果には, 後日, 東工大 TSUBAME や, HA8000 上の通常の割り当て時間を用いて小規模に少しずつ実行した結果が含まれていることをお断りしておく.

2 GXP, GXP make

本プロジェクトで用いる処理系 GXP, およびその上のワークフロー実行系である GXP make について簡単に説明する.

GXP は並列・分散環境用のシェルであり, 本プロジェクトで用いた HA8000 は当然として, SSH でアクセスできる Beowulf クラスタでも, その他のバッチスケジューラ環境でも, それらの混在で有っても同じ対話的環境を提供する. 本節では, GXP の基本的なユーザインタフェースと, HA8000 のバッチスケジューラコマンドが規定する利用モデルへ GXP を適合させるために, 必要な概念について説明する. なお, GXP は以下の URL から入手可能である.

http://www.logos.t.u-tokyo.ac.jp/gxp/

2.1 GXP の基本ユーザインターフェース

GXP は, (1) 多数の計算ホストへ同時にログインし, (2) それらの任意のホスト上でコマンドを起動する, ことを基本とする. 例えば以下は, ホスト y001 から, ノード群 y002〜y031 に同時にログインし, それらの上で hostname コマンドを実行するための一連の操作である. ログインには SSH を用いるものとする.

```
$ gxpc use ssh y
gxpc: no daemon found, create one
[1/1/1]$ gxpc explore y[[002-031]]
reached : y007 (y007)
reached : y030 (y030)
reached : y019 (y019)
...
reached : y008 (y008)
reached : y025 (y025)
gxpc : took 1.653 sec to explore 29 hosts
[31/31/31]$ gxpc e hostname
y001
y013
...
y023
y009
[31/31/31]$
```

1 行目 *gxpc use ssh y* で, y で始まるホスト間 (正確には, 正規表現 y にマッチするホスト間) では, SSH コマンドを用いれば到達可能であることを指示し, 2 行目 *gxpc explore y[[002-031]]* で実際にそれを用いて, y001〜y031 のホストへの SSH ログインが行われる. 3 行目 *gxpc e hostname* は, それらすべてのホストで hostname コマンドを実行する.

e コマンドではオプションの指定により, 一つのノードや, 一部のノードの集合を指定して, それらでのみコマンドを実行することができる.

一般に GXP は,

- use コマンドを用いて, どのホストからどのホストへ, どういう手段を用いて直接ログインが可能かを指定し,
- explore コマンドを用いて実際にホストへ到達し,
- e コマンドを用いてそれらのホストへコマンドを投入する,

という手順で用いる. それらをどういう順序で何度用いてもよく, ひとたび explore コマンドでホストへ到達した後は, 個々の e コマンドはホスト数が多くても対話的な速度で高速に実行される.

2.2 GXP とバッチスケジューラ

HA8000 のように, 計算ノードへのアクセスが SSH ではなく, バッチスケジューラ (qsub コマンド) を介して行われる場合, *gxpc use ssh y* の部分で代わりのコマンドを指定する. qsub コマンドを, 何のオプションもなく用いればよい TORQUE 環境であれば, 以下で, バッチスケジューラ経由で一つのプロセスを立ち上げることができる

```
$ gxpc use torque y001 node
gxpc: no daemon found, create one
[1/1/1]$ gxpc explore node
```

gxpc explore node のところで実際に, qsub コマンドを用いて計算ノード上でプロセスが起動される. もちろんどの計算ノードで起動するかはバッチスケジューラ次第であり, explore は (調節可能な) タイムアウト以内にプロセスが起動しなければ, ノードの獲得に失敗したとみなす.

ここで, 計算ノードを複数 (N 個) リクエストするには, 通常 *gxpc explore node* の代わりに, *gxpc explore node N* のように指定すればよい. これは, qsub コマンドを N 回立ち上げることに相当する.

バッチスケジューラからは, N 個の独立なリクエストが来たように見えるため, 同時に実行可能なジョブ数に制限のある環境では, この方法を用いて同時に到達可能なノード数は, その数に制限される. HA8000 では, 1 ユーザが同時に走らせることができるジョブ数は 2 以内に制限されているため, この方法を用いて獲得できるのは 2 ノードまでとなる.

2.3 HA8000 上の GXP

HA8000 のように, 1 ユーザが同時に実行可能なジョブ数を少なく (2) 制限している環境で, 多数のノードを用いた計算を行うには, 1 ジョブで多数のノードをリクエストする必要が有る. HA8000 はバッチスケジューラとして日立製 NQS を利用しており, それは実際には TORQUE へのフロントエンドとして動作している. 1 ジョブで多数のノードを要求することが可能であり, その場合でもあくまでプロセスは一つだけ起動される. 残りのプロセスを起動するのはユーザの役割である. それには主に二つの方法が有る.

1. どのノードがそのジョブが割り当てられているかは環境変数 (PBS_NODEFILE) に記されたファイルを介して取得できる. これを参照し, それを元に SSH 経由でそれらのノードへ到達する.

2. TORQUE API を用いる. 残りのホスト上では, TORQUE API を用いてプロセスを起動することができる. この際ホスト名を直接指定することはできず番号で指定する. そのため, TORQUE から割り当てられていないホスト上でプロセスを起動することは原理的にできない.

HPC 特別プロジェクト実施時には方法 1 を用いていたが, プログラムの間違いなどで, 割り当てられていないノードへもログインできてしまうこと, SSH で起動されたプロセスには資源のアカウンティングが行われないこと, などにより, 方法 2 を開発した. これは本プロジェクトを通じて得た, GXP の改良である. もちろんこの方法は, TORQUE を用いたあらゆる環境で用いることができる方式である.

GXP を経由して HA8000 を用いることにより, バッチスケジューリング環境では難しい, 対話的なジョブの実行やデバッグが可能になる.

2.4 GXP make

GXP make は, GXP の e コマンドの上に構築された, make の分散並列処理系である. ユーザは通常の Makefile を記述し, 通常の make コマンドの代わりに, *gxpc make* コマンドを実行する. GNU make には, -j オプションをつけることで依存関係のないジョブがノード内で並列実行される機能があるが, GXP make では-j オプションで分散並列実行される (explore で到達したノードへジョブが分散される).

GXP make は, 内部で無変更の GNU make を起動している. GNU make はプロセス起動のためのシェル (デフォルトでは/bin/sh) を変更する機能を備えているため, それを用いてプロセス起動部分を intercept することができる. GXP make はその機能を用いてすべてのプロセス起動 (以下, make ジョブ) を intercept して, スケジューラプロセス (xmake) にキューイングする. xmake は実行すべき make ジョブと, 現在ジョブが割り当てられていないノードを管理しながら, make ジョブをディスパッチする.

GXP make は通常の make とまったく同じ記述で分散並列実行をサポートし, すべての GNU make の拡張構文や機能なども, GNU make を無変更で用いているが故に, 自動的にサポートする. 大規模な並列処理向けの機能として, ジョブの進行状況や並列度の時間推移などを表示する (html を生成する) などの機能を持つ.

2.5 GXP make のスケーラビリティ

GXP make は, 1 つのスケジューラプロセスがすべての make ジョブリクエストを受け取ってジョブをディスパッチするという構成をとるため, 高並列で実行する際にはこの部分がボトルネックになる可能性がある. 一方で, 依存関係の解析を並列に行うのは難しく, まして無変更の GNU make を用いるという制約下では, この部分の潜在的ボトルネックを完全に取り除くのは難しい. そこでまず GXP make が本タスクを実行するのに十分なスケーラビリティを持つかどうかを, 予備実験によって定量的に検証した.

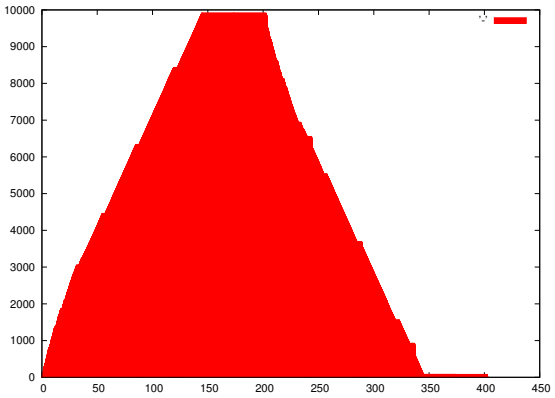


図 1: GXP make のジョブディスパッチ速度. 横軸=経過時間, 縦軸=並列度で, 開始 150 秒で頂上で並列度 10,000 に達している

図 1 は HA8000 システムにおいて, 200 秒 sleep するだけのジョブを 10,000 個投入したときの, 並列度の時間推移を示したものである. 約 150 秒ほどで 10,000 のジョブを投入し終え (並列度 10,000 へ到達), 200 秒経過後から, 徐々にジョブが終了している事が見て取れる. 言い換えれば, 60 ジョブ/秒程度のジョブディスパッチ性能を持っており, この程度の粒度以上のジョブが実行時間の大半を占めるようなタスクに対して, GXP make は 10,000 程度の並列性を十分に引き出せるということがわかる.

ただしこの速度は, GXP の処理系, および GXP を実行する Python の処理系がともにローカルファイルシステムにあったときの性能である. GXP 処理系や Python 処理系が共有ファイルシステムに置かれたときの性能は, 用いているファイルシステム, 設定 (PATH 環境変数など) で大きく異なっており, 条件により数倍〜10 倍程度のスローダウンが観測された. 同じ共有ファイルシステム (NFS) を用いていても大きく性能が異なっており, 今後詳細な調査を行う予定である. 本実験に置いては, GXP, Python をローカルファイルシステムに配置している.

3 生物医学テキストの, 深い自然言語処理を用いた索引づけ

3.1 タスクの目的と構成

本タスクは, 生物医学テキストの文献データベース MEDLINE[2] の抄録に対して, 高速な HPSG 構文解析器 [3] を核技術とした自然言語解析を行い, 構文情報を加味した索引づけを行う. 本タスクは, 東京大学情報理工学系研究科辻井潤一研究室により開発された.

この索引付けにより, 同じ意味を持つ構文的なバリエーションにとらわれない, 意味上の主語や意味上の目的語を指定した検索を行うことができる. この索引を用いた検索システム MEDIE[2]

[2] www.pubmed.gov

図 2: MEDIE 検索システム

が稼働しており, http://www-tsujii.is.s.u-tokyo.ac.jp/medie/ を通してアクセス可能である. そこで以下では本ワークフローのことを, **MEDLINE2MEDIE ワークフロー**と呼ぶことにする.

図 2 に, MEDIE で可能な検索と, 結果の一例を示す. 図では, Subject (意味上の主語) が p53, Verb (動詞) が, activate であるような文の Object (目的語) を検索しており, p53 遺伝子が活性化するものを検索する. 検索結果においては,

- . . . X which gets activated by p53 . . .
- . . . p53 . . . activates X . . .
- X, . . . , is . . . activated by . . . of p53

などの多様な構文的構造に対して, 正確に主語と述語をとらえた結果が返されている.

入力は, MEDLINE に収録されている, 1865 年から 2008 年 7 月 10 日までの全記事 (抄録が

```
<MedlineCitation Owner="NLM" Status="MEDLINE">
<PMID>17548109</PMID>
 ...
<ArticleTitle>Antimicrobial activity of truncated alpha-defensin
(human neutrophil peptide (HNP)-1) analogues without disulphide bridges.
</ArticleTitle>
 ...
<Abstract>
<AbstractText>Antimicrobial peptides play an important role in host defence,
 ...
 ...
 ...
use in therapeutic interventions.</AbstractText>
</Abstract>
 ...
</MedlineCitation>
```

図 3: 入力ファイルの抜粋

収録されているものと, いないものがある) で, 抄録数にして約 1,000 万, 文数にして約 1 億からなる.

それらが, いくつかの記事ごとに 1 ファイルにまとめられ, 入力ファイルとしては合計で 767 個のファイルからなる. 入力ファイルの一部分を抜粋したものを図 3 に示す. 出力は検索システム MEDIE が必要とする索引 (medie_db) であり, これを元に上記のような検索が実現される.

MEDLINE データベースに登録されている文献数は, 時間とともに指数関数的に増大しており, 検索システムの新鮮さを保つため, 計算能力にもそれに対応できるだけの向上が要求される. また, 抄録だけでなく論文の全文を検索の対象としたり, 全文から蛋白質の相互作用などのイベント抽出を行って新しい反応経路などを抽出・発見するなど, 処理能力に対する要求は今後も大きくなって行く.

異なるファイルに対する処理は独立に実行可能であるが, それで引き出せる並列度 (767) は十分ではなく, 一つのファイルをさらに分割して並列処理する必要がある. 特に時間を要する部分は構文解析部分で, ファイルごとに含まれる抄録数が大幅に異なること, 構文解析に要する時間が文により異なり, 事前に予測するのが困難であることなどから, この部分はある程度細かい粒度に分割する必要がある. 言い換えれば, 一つの入力ファイル自体をワークフローとして実行する必要がある. この記述は GXP make を適用する以前から, Makefile を用いて行われており, GXP make によって並列実行をするために施した変更はほとんどない. 図 4 に, 1 入力ファイルに対するワークフローを図式化したものを示す. ノードが一つのジョブ (逐次コマンド), 矢印はタスク間の依存関係を示し, それは同時にジョブ間で中間ファイルの形で受け渡されるデータを意味している.

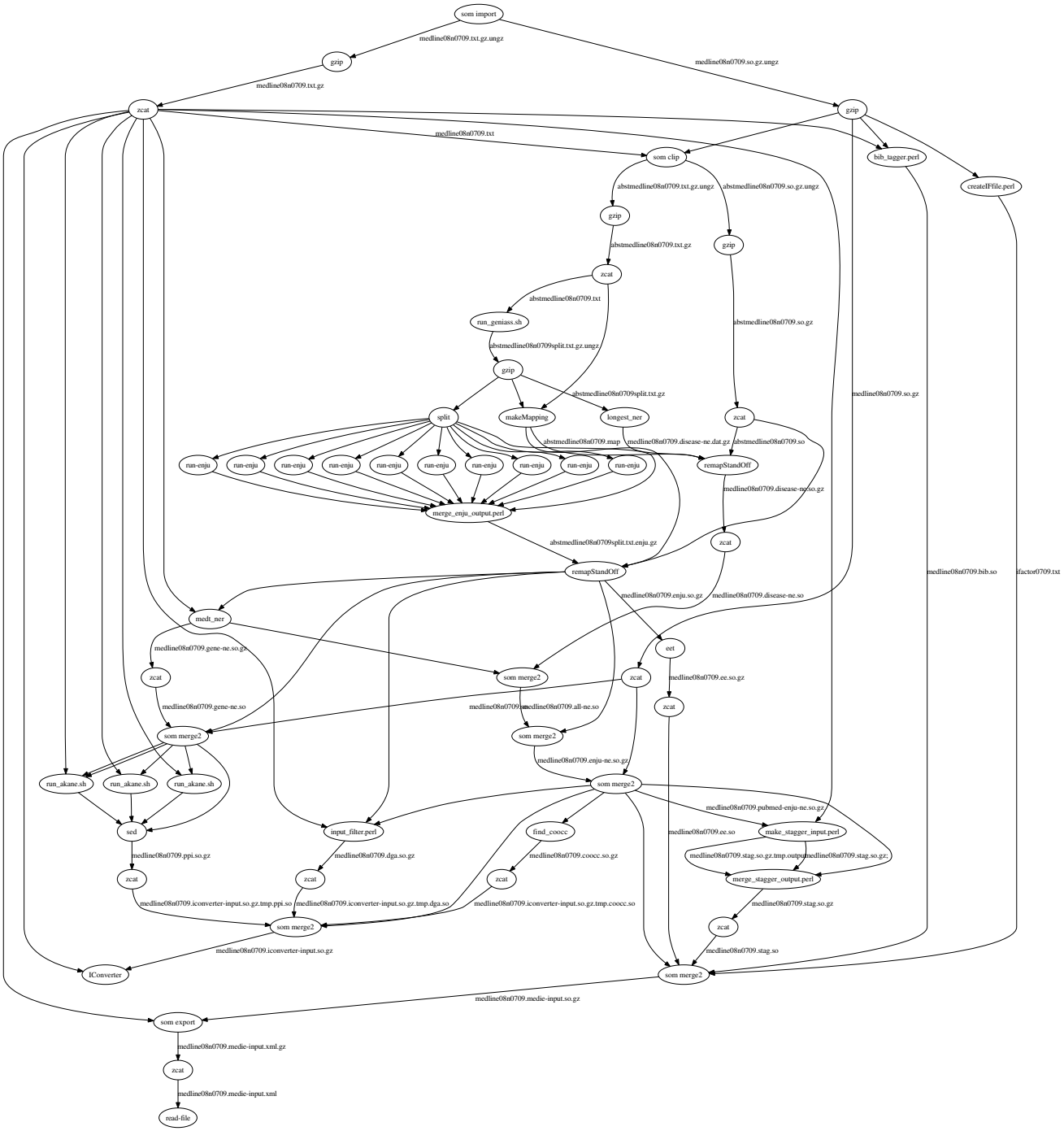


図 4: MEDLINE2MEDIE のワークフロー (1 入力ファイルにつき)

グラフの中央部で, “run-enju” という名前のノードが横に多数並んでいるが, それが構文解析を並列に実行している部分であり, 典型的には 40 個〜200 個程度である (図では見やすさのために少なくしている). CPU 時間の大部分はここで消費される. 左側に “run_akane.sh” という名前のノードが 3 つ程並んでいる部分では, 蛋白質-蛋白質の相互作用の抽出が行われており, ここでも 1 入力ファイルに対する並列化が行われている. CPU 消費量や並列度という観点から見ると, run-enju が占める部分が非常に多く, これ以外の部分は処理量という観点からは, 前処理・後処理と言ってよい.

中間ファイルは, 通常は共有ファイルシステムに置かれることを想定している. ユーザにとっての利便性や開発コストを考えると, そのようにして実行できることが望ましいのはもちろんであるが, 後に述べるように, 共有ファイルシステムとして HSFS を用いると, 大規模な実行を効果的に行うことは困難であったため, 今回の実験では次節で述べるような, ファイルを明示的にコピー (ステージング) する枠組みを追加して実験を行った.

3.2 準備

GXP make は, その make に準じた実行モデルから, 中間結果の受け渡しにファイルを利用する. これには, 長時間ジョブの中断と再開を自然に可能にするという, チェックポイントの効果もある. それらのファイルは共有ファイルシステムに置くのが簡便であるが, 転送性能を引き出し, ファイルサーバへの負荷の集中を避けるには, ノード間で明示的なコピーが望ましい場合もある.

HA8000 上においては 512 ノードの実行が割り当てられる以前の準備段階で, ワークフロー中の前処理・後処理に要する時間が他のシステムと比較して長く, それにより全体性能が悪化するという問題が明らかになっていた. 例えば図 5 は, 執筆時点で, 100 の入力ファイルを処理した際の, 最初の 2,500 秒程度の並列度の時間推移を, 二つのシステムで比較している. 左は HA8000 (16 CPU core × 8 ノード), 右は InTrigger システム[3]の筑波大学に設置されたクラスタ (8 CPU core ×16 ノード) を用いたものである. 前・後処理部分の比較のため, 構文解析部分を小さなデータに置き換えて比較している (したがって, 実際の性能比はここまでの差は出ない. あくまで前・後処理の比較が目的であることを断っておく). どちらも, 1 コアあたり 4 つまでのジョブを許容し, 合計で 500 程度の並列度を許容している. この様にした理由は, 我々が普段使える HA8000 環境の実コア数が 128 に限られているからである.

後者の共有ファイルシステムは NFS, ネットワークは 1000Base-T を 2 本トランクしたものである (計算ノード, ファイルシステム共). 前処理にかかっている時間は, グラフ上で並列度が最高に達するまでの時間で見積もれ, およそ 6 倍程度の開き (前者が 250 秒程度, 後者が 1,500 秒程度) がある.

[3] www.intrigger.jp

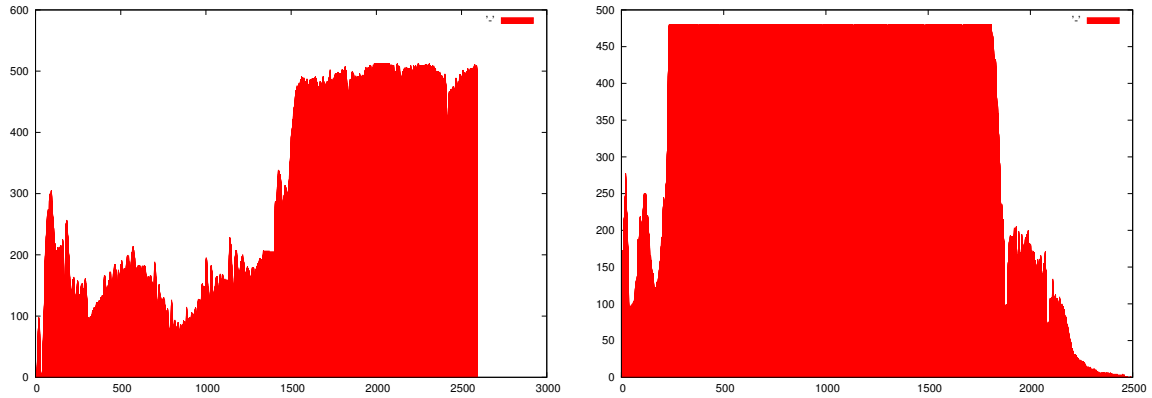


図 5: 100 ファイルを処理した際の並列度プロファイル (開始から 2,500 秒まで). HA8000 HSFS 使用 (左), InTrigger 筑波 NFS 使用 (右)

このまま大規模な実行を HSFS を用いて行うことは困難であると判断し, HSFS を中間ファイルの格納に用いない方式を採用した. 具体的には, ジョブの直前でファイルを明示的にコピー (転送) する補助システム FCP を実装した. FCP を用いた実行では, ファイルの読み書きはすべて計算ノードのローカルディスクに対して行われる. 個々の make ジョブ (make によって起動される個々のコマンド) の終了時は, ローカルディスクに書かれた実行結果をそのままにしておく. 逆に make ジョブの開始時は, そのジョブが読み込むであろうファイルを, それを作成した計算ノードから (scp コマンドを用いて) コピーする. ある make ジョブが読み込むファイルは, 一般には make ジョブ開始時に知ることはできないが, ここでは多くの Makefile で記述されたジョブに通用し, このタスクにもほぼ通用した経験則として, コマンドラインを見て, そこに現れる, パス名らしき文字列を入力ファイル (を包含する集合) とみなす, という方法をとった.

GXP make 自身には, コマンドラインをユーザが書き換えて, 前・後処理を挿入することができる枠組みを追加し, アプリケーション固有の前・後処理を自由に挿入できるようにした. 前処理で, 上記の方法で認識した入力ファイルと思しきファイルの作成ノードを突き止めて, それを scp でコピーする. あるファイルを作成したノードを見つけるために, メタデータサーバ (FCP サーバ) を走らせる. FCP サーバは, あるファイルに対する問い合わせを初めて受け取った際はすべての計算ノードに問い合わせを発行し, 返された結果を以降の問い合わせのためにキャッシュする. もちろん FCP サーバ自体がボトルネックになり得るが, ファイル作成時のオーバーヘッド (メタデータ更新) がないこと, 転送その物はサーバを介さずに行われること, 実装が単純であることなどから今回の目的に合致していると考える.

ファイル番号	実行時間 (秒)
001-199	20,664
200-299	36,032
300-399	48,615
400-499	54,695
500-599	45,824
600-767	29,784

表 1: HA8000 30 ノード (HSFS 不使用. FCP 使用) での実行時間

3.3 512 ノードでの実行結果

512 ノード (8,192 コア) が割り当てられた約 12 時間のスロットの後半を利用して, すべての入力ファイルを処理することを目標にした. 入力ファイルとノードを 2, 3 のグループに分割して, 一度に処理するファイル数を制限した. 中間ファイルはもちろん, GXP 処理系や Python を含めて, 極力 HSFS を参照しない工夫を行った. それでも構文解析に至るまでの前処理が数時間, 遅々として進まなかった. 実行の途中でそれは, 計算結果を蓄積するために時折行われている, ローカルディレクトリからホームディレクトリへのファイルのコピーに起因することが明かになった. そのコピーは, ジョブ終了後に計算ノードのローカルディスクへアクセスできない以上必須のものであり, 必要最低限のファイルアクセスとして実行していたが, これがスローダウンを引き起こすこととなってしまった. 調査のためそのコピーをするプロセスを殺すと, 並列度が急激に向上した. しかしながら割り当てられた時間の大部分を使いきっており, まとまった結果は残念ながら得られなかった.

3.4 事後処理

8,192 並列度を用いた処理の結果が芳しくないものであったため, その後 HA8000 の 30 ノード (480 コア. FCP 使用) を用いて全ファイルを分割処理した. 前者の結果の要約を表 1 に示す.

また, 東工大 TSUBAME (1024 コア, Lustre ファイルシステム使用) を用いて上記の 400-499 番の 100 個の入力ファイルを同時処理した結果が図 6 である. 実行時間は 35,000 秒程度で, HA8000 上での 480 コアの結果 54,695 秒と比較すると, Lustre を用いた TSUBAME と, HSFS を用いずに FCP を用いた HA8000 との間に大きな性能の差はないと言える.

3.5 まとめ

本タスクに関して現状で, 実証できたこと, 達成できなかったことは以下のとおりである

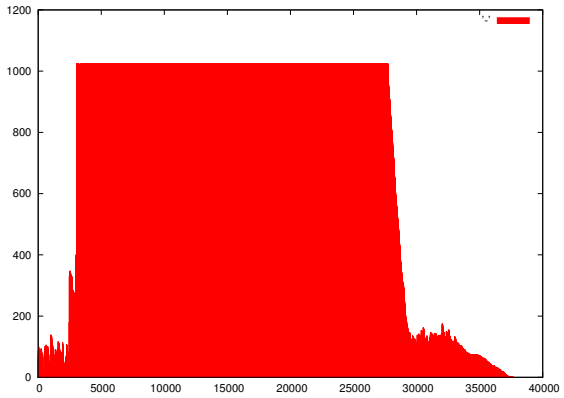


図 6: TSUBAME 上で 100 ファイルを 1,024 コアで処理した際の並列度プロファイル (Lustre ファイルシステム使用)

1. GXP make 自身は, GXP 処理系や Python をローカルディスクに置いた場合, 150 秒程度で 10,000 タスクを分散させるスループット (約 60 タスク/秒) を持っており, 一タスクが数分を要するワークフローであれば HA8000 規模の並列度を有効活用できることが実証された.

2. しかし HA8000 のファイルシステム (HSFS) 性能は, 多数のファイル作成や, 小さい単位での入出力を伴うワークロードに対して問題があり, はるかに廉価な選択肢 (NFS) に大きく劣っている.

3. Lustre ファイルシステムを用いれば 100 入力ファイルを 1,000 程度の並列度で, 効果的に実行できることが確認できた. NFS を用いた場合については, 500 程度の並列度でそれが確認できた. これ以上の規模については実験の機会を得ることができておらず, 実証できていない.

4. ファイルのステージングを行うツール FCP は有効に機能し, 500 程度の並列度で良好な性能を示したが, それ以上の規模にスケールすることは, まだ実験の機会を得ることができておらず, 実証できていない.

4 類似文字列検出による大規模ウェブページコレクションからの類似ページの発見

4.1 研究の背景・目的

本研究では日本語ウェブページ 1 億件という大規模ウェブページコレクションを対象として，あらゆるページのペア間に含まれる類似文字列の検出を行ない，その結果から，ミラーページ，

引用ページ，スパムページなどを同定する．本稿では類似文字列を含むページペアを類似ページと定義する．

類似ページを検出することは，大規模ウェブコレクションを検索エンジンの検索対象とする際に，以下の点で検索エンジンの質を向上させることができる．

- 同一・類似ページを検出し，それらをまとめて提示することにより，ユーザの検索結果把握を阻害させないようにできる．
- 他サイトの引用のみで記事を作ったようなスパムページを検索対象から除外する．

また，検索エンジンの質の向上だけでなく，実際のウェブに，盗作や，悪意のあるスパム的なものも含め，どのような引用関係がどのような割合で存在しているかなどの興味深い分析が可能となる．

本研究で対象とする類似ページを以下のように分類する．(図 7 を参照)

- **同一**: 2 つのページが同一のものである．例えば，ミラーページや，定型ページ (apache が生成するページなど)，盗作ページなどがある．
- **包含**: 1 つのページが他方のページに包含される場合である．例えば，ブログの月別ページと日別ページなどがある．
- **部分共有**: 2 つのページでその部分を共有するページである．例えば，引用ページと被引用ページ，文集合を共有するページ (例えば,「当Ｗｅｂサイトの手法，レイアウト，電子ファイル等は当社及び，グループ会社が権利を有します．以下の内容を無断で行う事を禁じます.」という文集合を共有するページ)，スパムページ，盗作ページと被盗作ページなどがある．

本研究では，まず，高速類似文字列検出アルゴリズム SACHICA を用いて，あらゆるページペア間に含まれる類似文字列の検出を行なう．その際には，ウェブコレクションを適当な大きさのデータを分割し，GXP を用いて並列に処理する．64 ノード (1,500CPU コア) を用いると，500 万ページにおける類似文字列検出を約 1 時間半で行なえることを確認することができた．1 億ページに対して全ての処理を行なうにはこの 400 倍の処理を行なう必要があるが，このうち，2,000 万ページにおけるあらゆる類似文字列を検出した．次に，この結果から，類似ページを検出し，同一ページ，引用ページなどに分類した．

関連研究として，Manku らは Web ページクローリングの際にクロールしたページがすでにクロールしたページの類似ページかどうかを判定し，類似ページであればクロールしないといった手法を提案している [1]．類似ページの判定はまず各ページを f ビットの fingerprint に変換し，2 つのページの fingerprint の違いが k ビット以内であるかをチェックすることによって行

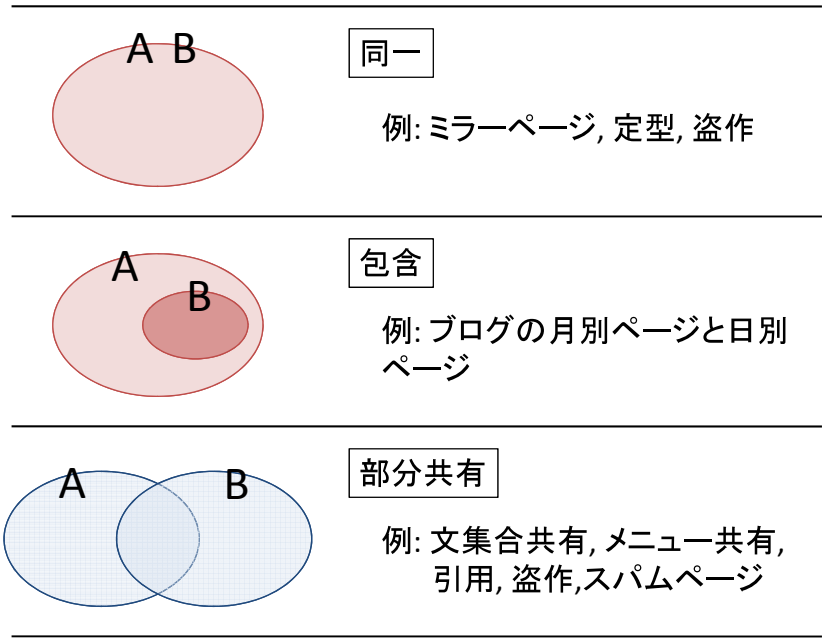


図 7: 類似ページの分類

なっている．この研究では上記の分類での同一もしくはほぼ同一ページしか検出することができず，ページの一部が類似しているページペアは検出できない．

4.2 ウェブコレクションと高速類似文字列検出プログラム SACHICA

ここでは，本研究で用いたウェブコレクション，ならびに，高速類似文字列検出プログラム SACHICA について簡単に説明する．

4.2.1 ウェブコレクション

ウェブページコレクションとして，検索エンジン基盤 TSUBAKI[5] で検索対象となっている 1 億ページを用いた．各ページは HTML タグの削除・文抽出などの処理が行なわれ，標準フォーマット [4] と呼ばれる XML 形式に変換されて管理されており，URL・文字コードなどのメタ情報，文集合，インリンク (このページにリンクしているページ集合)，アウトリンク (このページがリンクしているページ集合) などの情報が含まれている[4]．図 8 に標準フォーマットの例を示す．<RawString>タグで囲まれた部分が HTML から抽出された文を示す．

[4]実際には形態素解析結果・構文解析なども含まれている．

```
<?xml version="1.0" encoding="utf-8" ?>
<StandardFormat Url="http://www.kikuchinobuhide.com/assembly/10index.html" OriginalEncoding=
"shiftjis">
<Header>
<Title Offset="345" Length="15">
 <RawString>議会答弁集</RawString>
</Title>
<OutLinks>
 <OutLink>
  <RawString>平成１６年市議会答弁集</RawString>
  <DocIDs>
  <DocID Url="www.kikuchinobuhide.com/assembly/16index.html">050947710</DocID>
  </DocIDs>
  </OutLink>
 <Text Type="default">
 <S Offset="8248" Length="88" is_Japanese_Sentence="1" Id="1">
  <RawString>●平成１９年市議会答弁集</RawString>
 </S>
 <S>
  ...
 </S>
 </Text>
</StandardFormat>
```

図 8: 標準フォーマットの例

4.2.2 高速類似文字列検出プログラム SACHICA

SACHICA(Scalable Algorithm for Characteristic/Homogenous Interval CAlculation)[5] とは入力した文字列ファイルから，決まった長さの部分列の組でハミング距離が閾値以下のものを全て見つけ出すアルゴリズムである．この種の他のアルゴリズムに比べ，正確性を犠牲にせずかつより高速であり，主にゲノム相同性解析を目的として開発された．本研究ではこのプログラムをウェブページ中の類似文字列検出タスクに利用する．

4.3 ジョブの分割

4.3.1 前処理

簡単な URL の正規化でわかる同一ページは今回の処理からは除外した．表 2 に 5 つのタイプとそのページ数を示す．例えば 2 番目のタイプの場合,「http://.../%7E...」の「%7E」を「~」

[5] http://research.nii.ac.jp/~uno/code/sachica05.zip

表 2: URL の正規化

タイプ	ページ数
スラッシュの重複を削除	2,560,297
%7E をチルダに置換	802,902
ホスト名の最後のドットを削除	250,732
index.(html|html|cgi|php) を削除	1,405,483
「http://www.」の「www」の有無	5,604,769

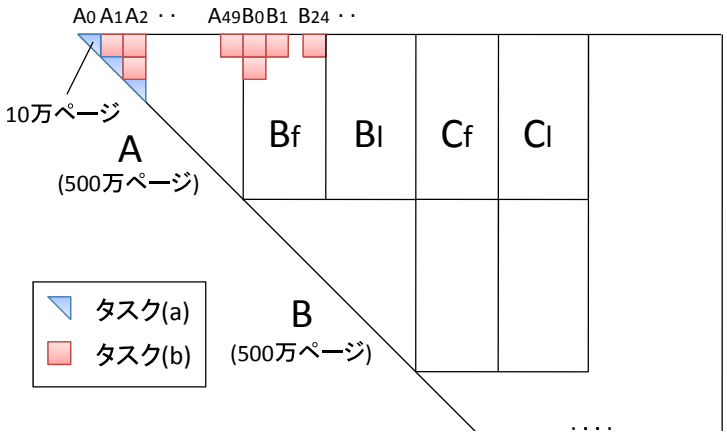


図 9: ジョブの分割

に置換したページが 1 億ページ中に存在する場合，このページを処理対象から外した．

4.3.2 ジョブの分割

以下で，類似文字列検出を行なうジョブの分割方法について述べる．分割方法を図 9 に示す．

1. 1 億ページを 20 分割し，500 万ページずつにする．これを **chunk** と呼ぶ．(順に，A, B, C, ... と名前をつける)．
2. chunk それぞれをさらに 50 分割し，1bin とする．(A を A_0,A_1,...,A_{49}，B を B_0,B_1,...,B_{49}，... に分割する．)

   1bin は約 10 万ページの集合からなる 1 ファイルであり，図 10 に例を示す．各ページから標準フォーマットの<RawString>タグで囲まれたものを文として抽出し，euc-jp に変換した後，各ページの文集合を空行で連結したものである．図 10 では,「●平成１９年市議会答弁集」から「Ｋｉｋｕｃｈｉ　Ｎｏｂｕｈｉｄｅ　Ｓｕｐｐｏｒｔｅｒｓ’Ａｓｓｏｃｉａｔｉｏｎｓ.」までが 1 ページ,「ｕｐｄａｔｅｄ／２００３／７／２９」から「戻る」までが 1 ページを示す．

●平成１９年市議会答弁集
●平成１８年市議会答弁集
...
ａ）現状と問題点。
ｂ）１１９番受信体制について。
ＡｌｌＲｉｇｈｔｓ　Ｒｅｓｅｒｖｅｄ．Ｃｏｐｙｒｉｇｈｔ　（ｃ）　２００５．
Ｋｉｋｕｃｈｉ　Ｎｏｂｕｈｉｄｅ　Ｓｕｐｐｏｒｔｅｒｓ’Ａｓｓｏｃｉａｔｉｏｎｓ．

ｕｐｄａｔｅｄ／２００３／７／２９
Ｔｈｅ　Ｉｎｔｅｒｎｅｔ　Ｇｕｉｄｅ　ｔｏ　Ｌｏｄｇｉｎｇ　ｉｎ　Ｊａｐａｎ
変更、抹消、登録データー用ＦＯＲＭ
。
変更の場合は施設名と変更箇所のみ記入してください。
変更、抹消、　データーの変更抹消
登録済みサイト
日本語英語、両方日本語のみ　英語のみ
お願い：施設名の頭にＬＧｘｘｏｏとある番号を必ずお書きください。
...
戻る

北海道の病院
Ｔｈａｎｋ　ｙｏｕ　ｆｏｒ　ｖｉｓｉｔｉｎｇ　ｍｙ　ｓｉｔｅ．
...

図 10: 入力ファイルの例

3. 2 種類のタスクを考える．図 9 において，以下のタスク (a) を青三角，タスク (b) を赤四角で表す．以下のいずれの場合においても，同一ページ内にある類似文字列は類似文字列とみなさないようにする．

   (a) 1bin 内の類似文字列を検出するタスク
   (b) 2 つの bin 間にある類似文字列を検出するタスク
   この場合，SACHICA には 2 つのファイルを引数として渡す．

4. 並列計算するにあたり，上記で設定したタスクの集合からなるジョブを以下の 2 種類考え，1 ジョブを並列計算の最小単位とする．

   (a) 1chunk 内の解析をするもの (例: A 内の解析をするもの)
   このジョブはタスク (a) とタスク (b) からなり，タスク (a) が 50 タスク (A_0,A_1,...)，タスク (b) が 50 * 49 / 2=1,225 タスク (A_0-A_1, A_0-A_2,..., A_{48}-A_{49})，計 1,275 タスクからなる．

pageid1	pos1	pageid2	pos2	length
010001159	89	011522466	100	75
010001179	733	010004257	960	71
010001179	734	010005184	916	77
010001179	734	010005625	216	76
010001179	88	010040879	205	76
010001179	734	010041164	1657	77

図 11: 出力の例

(b) ある chunk と別の chunk の半分の解析をするもの (例: A と B_f(B 前半) の解析をするもの．B_f は B_0,B_1,...,B_{24} からなる)
このジョブは，50 * 25 = 1,250 タスク (A_0-B_0, A_0-B_1, ..., A_{49}-B_{24} の解析) からなる．
同様に，A と B_l(B 後半)，A と C_f，...，B と C_f，... の解析を行なう．

このようにジョブを分割することによって，1 ジョブあたりの時間がなるべく均一になるようにする．

1 億ページに対しては，4-a タイプを 20 回，4-b タイプを，38 + 36 + ... + 2 = 380 回，計 400 回ジョブを投げる必要がある．

各並列計算処理の先頭では，入力ファイルを各ノードのローカルディスクにコピーすることによって，ファイルサーバへの負荷がかからないようにした．ローカルディスクへのコピーには GXP のコマンド `bcp` を用いた．また，結果は `bzip2` で圧縮しながらローカルディスクに書き出し，終わるとファイルサーバにコピーした．

SACHICA の出力例を図 11 に示す．一行が一類似文字列ペアに相当し，pos と length は euc-jp での文字数で表す．そして，ページペアごとにまとめ，URL と類似文字列の情報を付与したものを図 12 に示す．

4.4 実験・考察

4.4.1 GXP による並列処理

GXP を使って，1 億ページのうちの 2,000 万ページに対して類似ページ検出を行なった．SACHICA のパラメータとして以下を用いた．

- 類似文字列とみなす最小文字列長: 70
- 20 文字で 1 文字あたりの異なりを許す

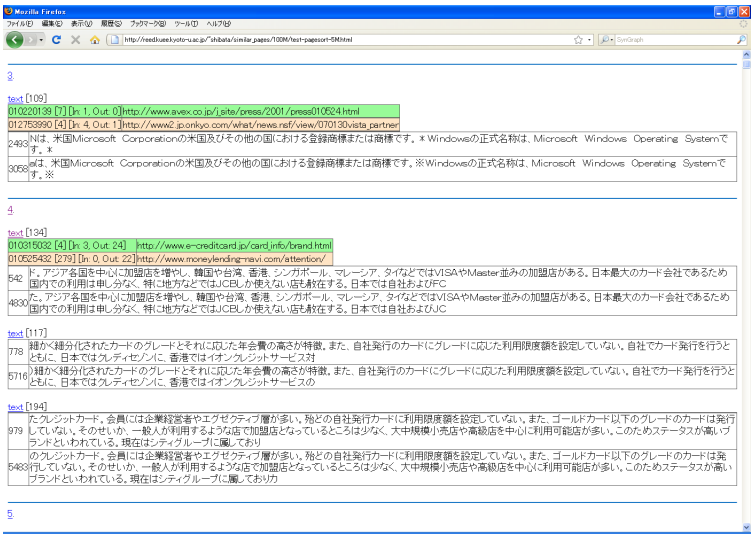


図 12: 類似ページの例 (各類似ページペアに対して，そのページペアに含まれる類似文字列と両ページの URL を表示している)

64 ノードを用い，ノードあたり 14CPU コアを利用したところ，4-a,4-b のどちらのタイプのジョブも約 1 時間半で終了した．図 13 に並列計算が進行する様子を示す．図中の横軸は経過時間 (秒)，縦軸は走っているタスク数を表す．1bin のサイズは約 300M(1chunk は gzip で圧縮して約 5.6G) であり，1 ジョブあたりの出力は bzip2 で圧縮して約 300M であった．

1 億ページの解析を行なうには 400 ジョブを走らせる必要があるが，そのうちの 20 ジョブを走らせることにより，2,000 万ページの解析を行なった[6]．

4.4.2 類似ページの分類

2,000 万ページの処理結果のうち 1,000 万ページにおいて，検出した類似文字列をもとに，1,000 万ページ中に含まれる類似ページの自動分類を行なった．出力された類似文字列のペアをページペア単位で集計すると，約 325 億の類似ページペアが得られた．

まず，あらゆるページペアにおいて，各ページの類似文字列重複率 (類似文字列とみなされた文字数/全文字数) を計算し，2 つのページの重複率をプロットしたものを図 14 に示す．重複率によって以下のように分類することができる．

ちょうど 1 2 つのページが同一である．この中には URL が類似したものだけでなく，定型ページ，IP アドレスとドメイン名・ホスト名のアドレスのページペア (例: `http://219.166.24.90/dainichi/rensai/furusato/furusato040728.html` と `http://ns.nnn.co.jp/dainichi/`

[6] 東京工業大学の TSUBAME も併用して解析を行なった．

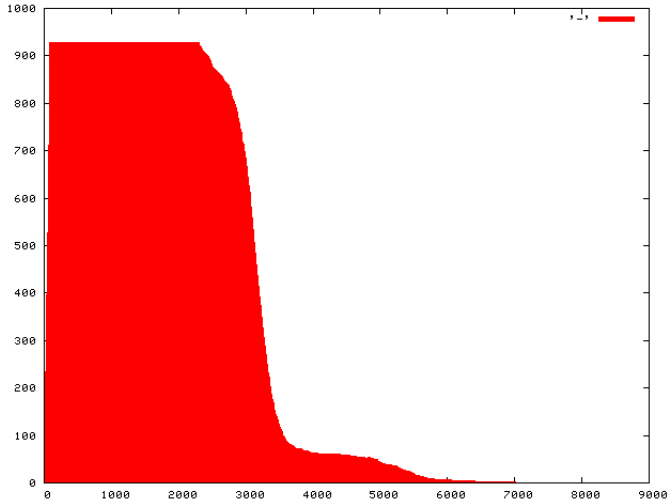


図 13: 並列計算における経過時間と走っているタスク数の関係

rensai/furusato/furusato040728.html)，ドメイン名が類似していないミラーページ (例: http://www.caj.co.jp/focus/security/spyware_list.htm と http://casupport.jp/focus/security/spyware_list.htm) が含まれていた．

1 より少し小さい ブログの「トップページ」と「月別アーカイブ」などといった包含ページであり，例えば，http://kroko.maxs.jp/~kroko/mt/archives/003687.shtml と http://kroko.maxs.jp/~kroko/mt/archives/003802.shtml などがあった．

0 より少し大きい 部分共有ページであり，ページの一部に定型文が含まれているものが多数あった．

重複率が 0.25 から 0.75 の場合は様々なページタイプが含まれていたが，標準フォーマットにうめこまれているインリンク，アウトリンクの情報を用いて，2 つのページがリンク関係にあるもののみに限定したところ，様々なページにリンクしているページ，様々なページからリンクされているページが得られ，以下のようなものであった．

様々なページにリンクしているページ いろいろなページをつなぎあわせたようなページ (図 15) などが得られた．いろいろなページをつなぎあわせたようなページは一種のスパムページであり有用なページではないので，今後，検索エンジンのインデックスから除外する予定である．

様々なページからリンクされているページ SEO 対策で様々なページ間でリンクしあっているものやトラックバックがはられているものがあった．SEO 対策で様々なページ間でリンクしあっているものは上記のスパムページと同様，検索エンジンのインデックスから除外する予定である．

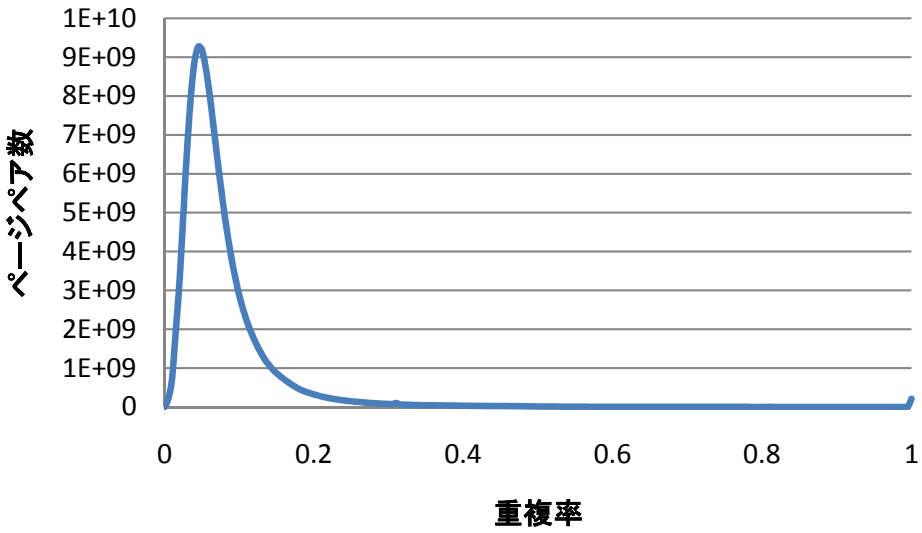


図 14: 重複率とページペア数

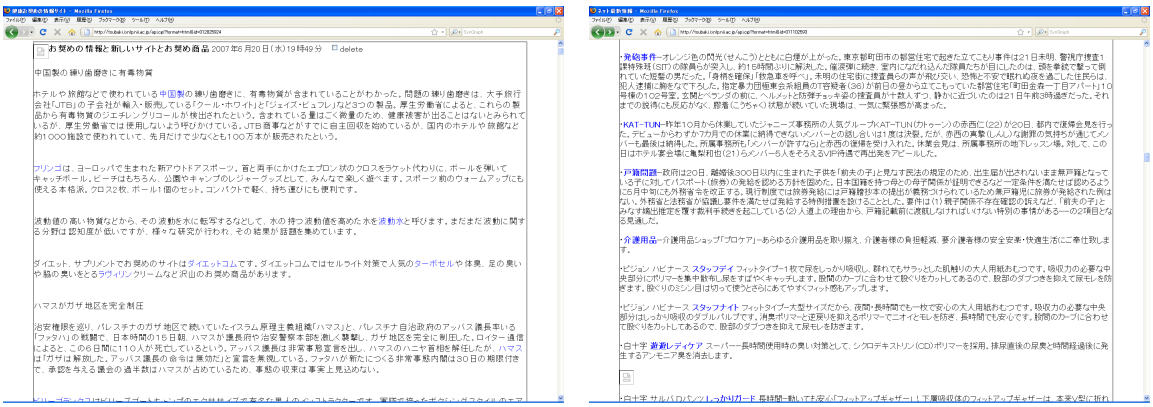


図 15: いろいろなページをつなぎあわせただけのページの例

4.5 まとめ

本研究では，大規模ウェブページコレクションを対象として，あらゆるページのペア間に含まれる類似文字列の検出を行ない，その結果から，同一・包含・部分共有関係にあるページなどを同定した．

今後の課題としては 1 億ページに対して処理を完了することや，類似ページ検出結果を検索エンジン基盤 TSUBAKI に反映させることなどがあげられる．

5 おわりに

本プロジェクトで実施したデータ集約的タスク, タスク間のデータの受け渡しをファイルを介して行うタスクは, もともとファイルシステムのスケーラビリティに対する要求, 負荷が大きく, チャレンジングなタスクである. また, 多くの逐次プログラムを, 少ない統合の手間で組み合わせて並列処理を行うワークフローシステムは, 統合の仕方を変更しては実行を繰り返す, 生産性に対する要求が大きいタスクで, その意味からもソフトウェア開発・実行環境にとってチャレンジングな課題を提供する.

本プロジェクトでは, 総じて, 500-1,000 並列程度の規模に対して, 複雑なワークフローを慣れた記述 (Makefile) で生産性高く実行できることを実証できた. また, それ以上の規模に対する実験の機会を与えていただいたことは, 有益な経験であり感謝したい. 今後も機会あるごとに, ぜひ 10,000 並列レベルのワークフローを簡単に, 効果的に並列化できる枠組みとして, 研究と実証実験を続けて行きたい. また, GXP を, 並列処理を手軽に実行できる仕組みとして, HA8000 ユーザの日常的ツールとして提供して行きたいと考えている.

参考文献

[1] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 141–150, 2007.

[2] Yusuke Miyao, Tomoko Ohta, Katsuya Masuda, Yoshimasa Tsuruoka, Kazuhiro Yoshida, Takashi Ninomiya, and Jun'ichi Tsujii. Semantic retrieval for the accurate identification of relational concepts in massive textbases. In *the Proceedings of COLING-ACL 2006*, pp. 1017–1024, July 2006.

[3] Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. Extremely lexicalized models for accurate and fast hpsg parsing. In *the proceedings of EMNLP*, pp. 155–163, 2006.

[4] Keiji Shinzato, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. A large-scale web data collection as a natural language processing infrastructure. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC08)*, 2008.

[5] Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. TSUBAKI: An open search engine infrastructure for developing new infor-

mation access methodology. In *Proceedings of Third International Joint Conference on Natural Language Processing (IJCNLP2008)*, pp. 189–196, 2008.

[6] Kenjiro Taura. Gxp: An interactive shell for the grid environment. In *Proceedings of the Innovative Architecture for Future Generation High-Performance Processors and Systems*, pp. 59–67, 2004.

東京大学情報基盤センター・スーパーコンピューティングニュース
Vol. 11 No. Special Issue 2 (2009.3)

編集・発行　　東京大学情報基盤センター
スーパーコンピューティング部門

〒113-8658　東京都文京区弥生 2-11-16
（電話）　03-5841-2717（ダイヤルイン）
（FAX）　03-5841-2708