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)]．

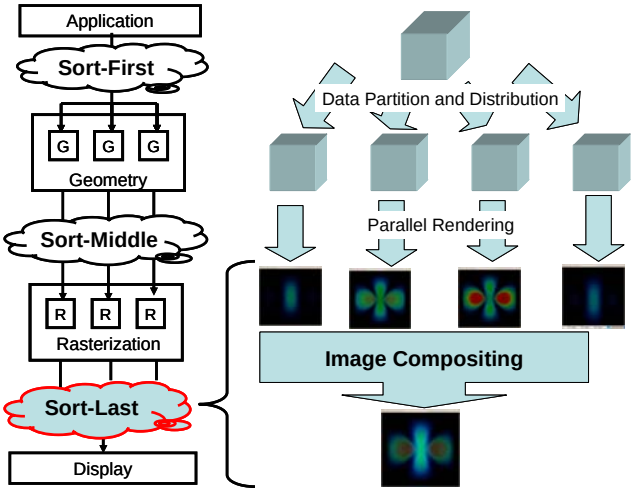


図 1　並列レンダリング方式

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．

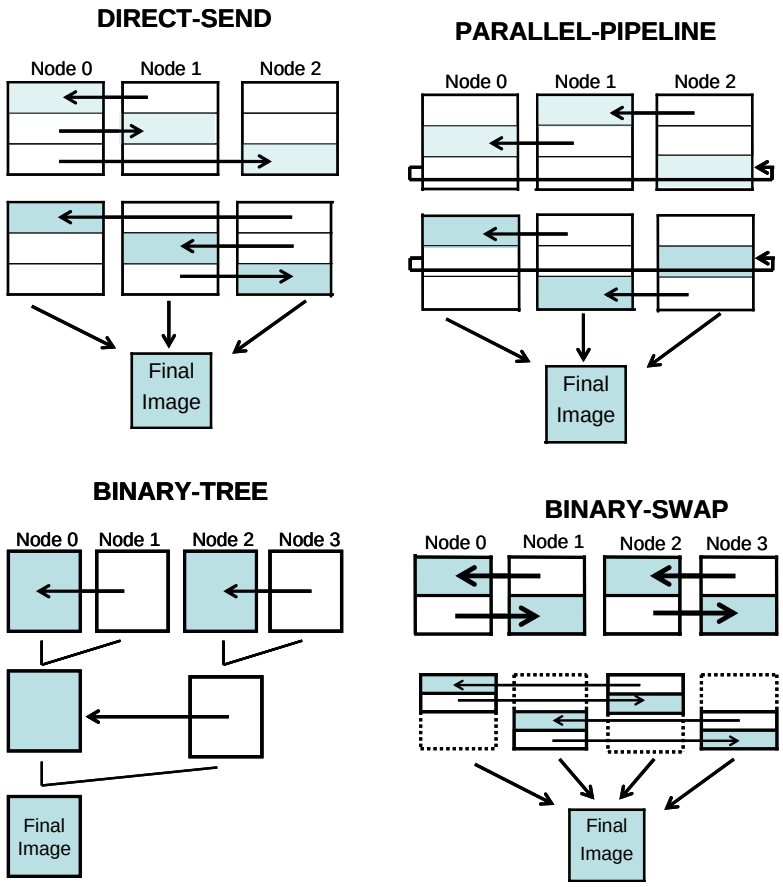


図 2　様々なイメージ重畳法

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．

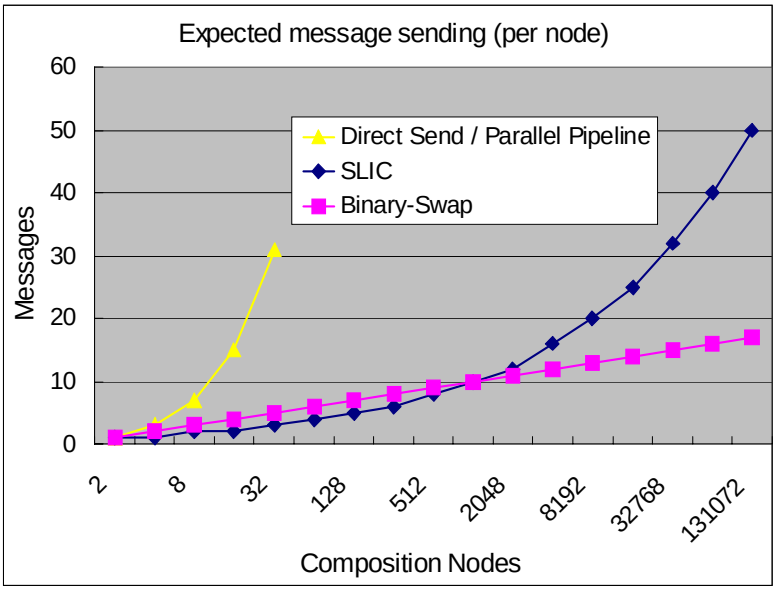


図３　画像重畳法の通信コストの比較

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．

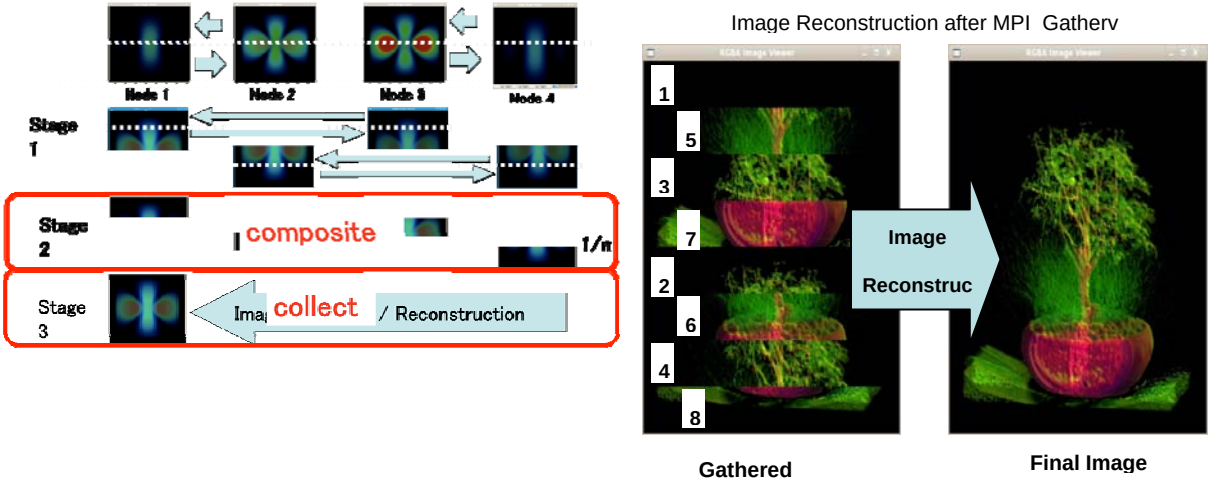


図 4 Binary-Swap イメージ重畳法のアルゴリズム

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）.

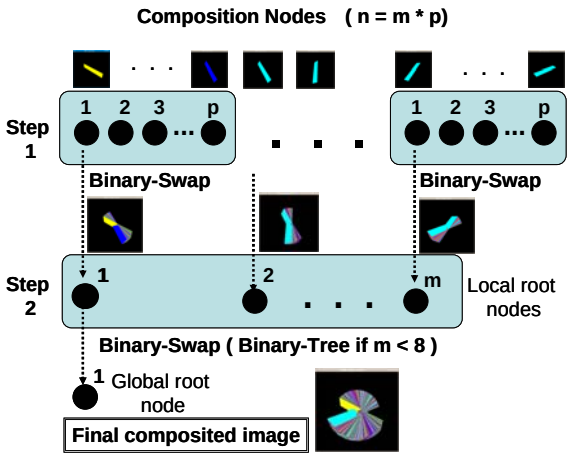


図５　Multi-Step イメージ重畳法のアルゴリズム

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

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.

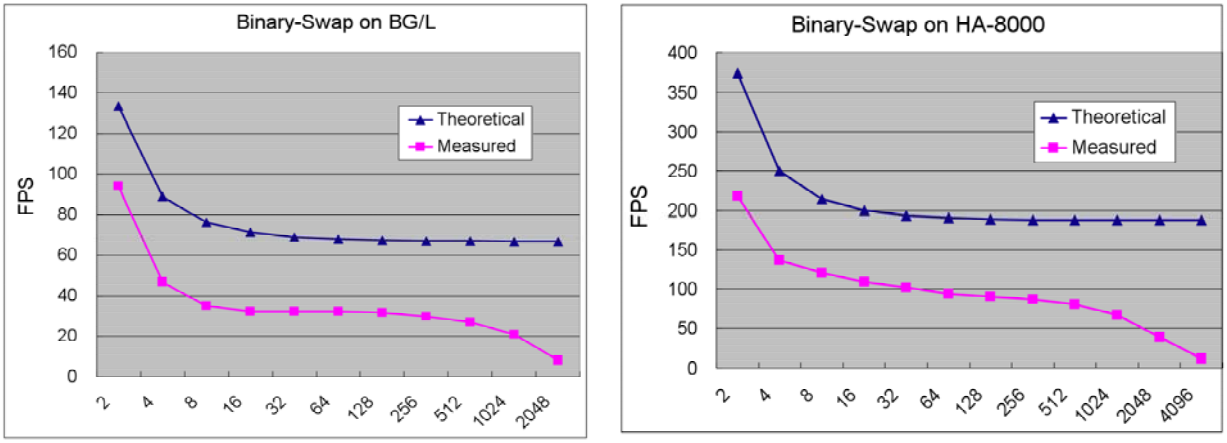


図 6　Binary-Swap 法によるイメージ重畳の性能

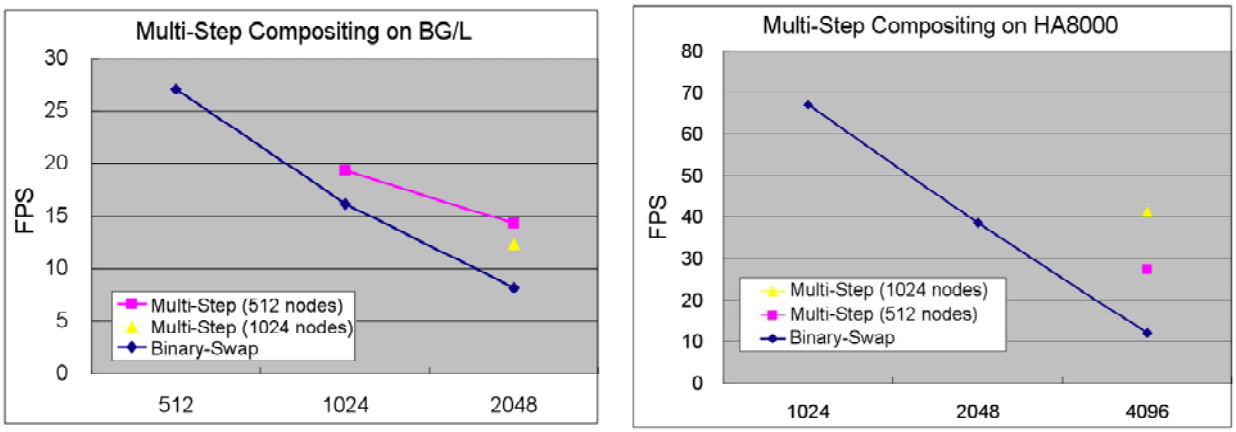


図 7　Multi-step 法によるイメージ重畳の性能改善

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．

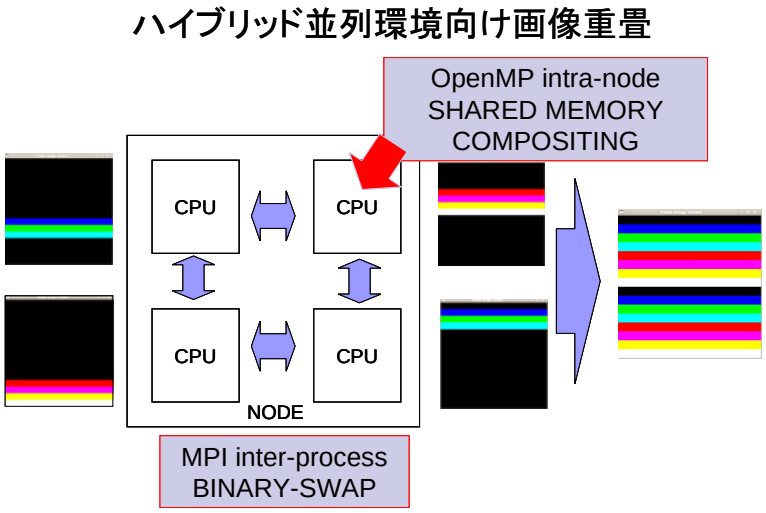


図８　共有メモリに対する画像重畳

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

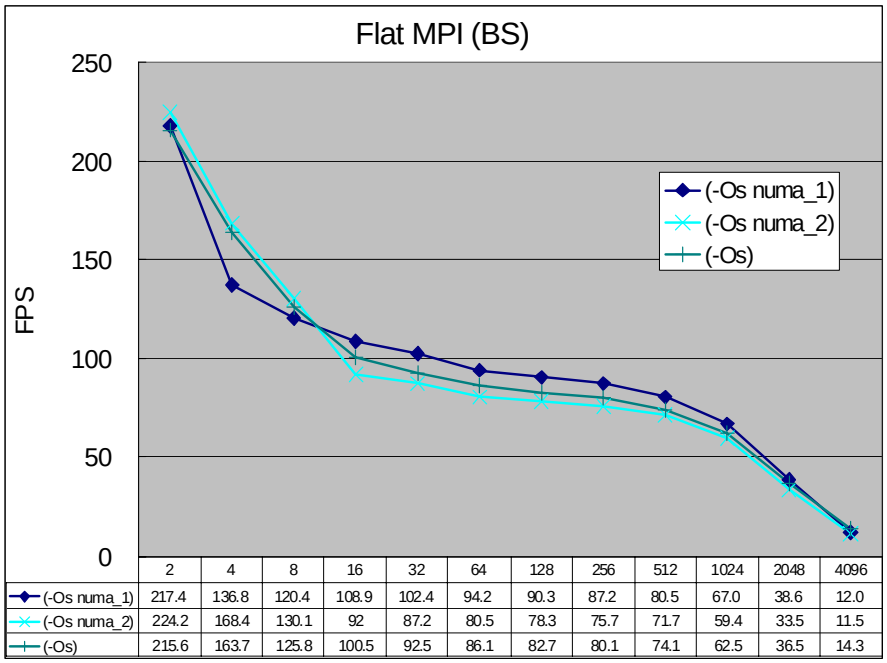
Flat-MPI	Hybrid MPI-OpenMP
#!/bin/sh	#!/bin/sh
#@$-r bswap	#@$-r bswap
#@$-q monthly	#@$-q monthly
#@$-N 256	#@$-N 256
#@$-J T16	#@$-J T4
#@$-lm 2gb	#@$-lm 7GB
#@$-lM 28GB	#@$-lM 28GB
#@$-lT 0:15:00	#@$-lT 0:15:00
#@$-e bswap.elog	#@$-e bswap.elog
#@$-o bswap.log	#@$-o bswap.log
#@$-lc 0MB	#@$-lc 0MB
#@$-nr	#@$-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

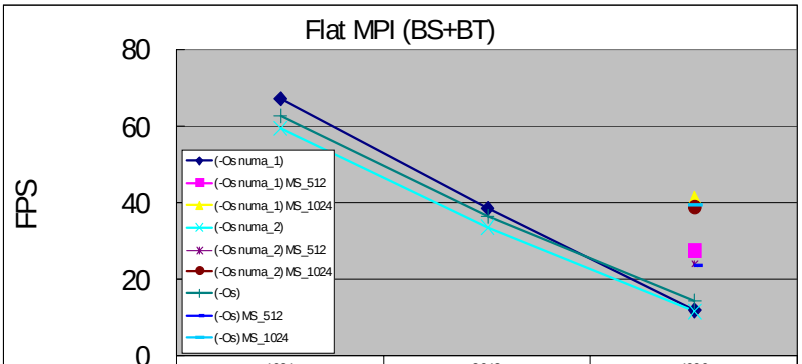
- Binary-Swap + Binary-Tree

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．

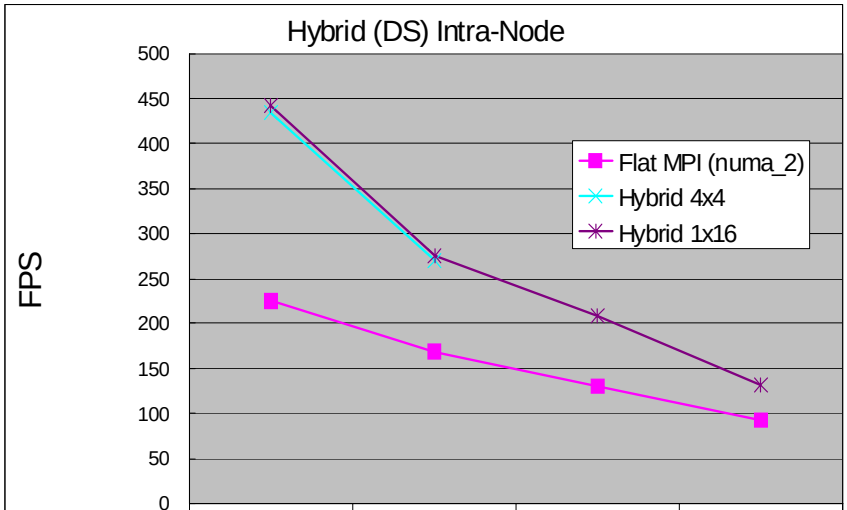
- Hybrid　MPI-OpenMP
  - Direct-Send + Binary-Swap

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．



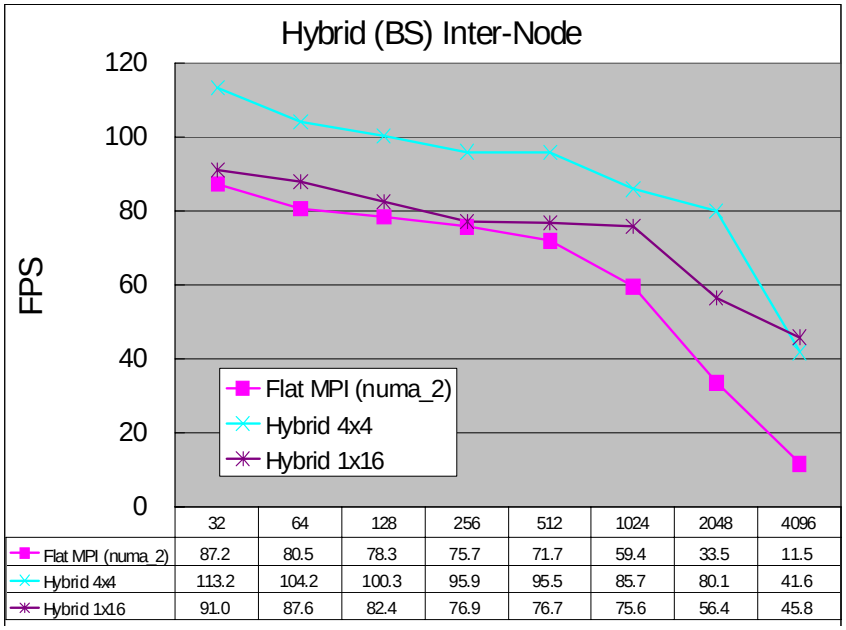
	1024	2048	4096
(-Os numa_1)	67.0	38.6	12.0
(-Os numa_1) MS_512			27.4
(-Os numa_1) MS_1024			41.3
(-Os numa_2)	59.4	33.5	11.5
(-Os numa_2) MS_512			24.0
(-Os numa_2) MS_1024			38.8
(-Os)	62.5	36.5	14.3
(-Os) MS_512			23.5
(-Os) MS_1024			39.4

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).