Vol.2012-HPC-136 No.27
2012/10/4

HPC アプリケーションの性能可搬性に関する一検討

小松一彦[†1,†4] 江川隆輔[†1,†4] 安田一平[†2]
撫佐昭裕[†3] 松岡浩司[†3] 小林広明[†1]

近年 HPC システムの多様化が進む中で，様々な HPC システムにおいても性能を引き出すことが可能な性能可搬性の高い HPC アプリケーションの開発が重要になりつつある．しかしながら，一般的に，HPC アプリケーションは 1 つの HPC システムに高度に最適化されているため，他の HPC システムでも高い性能を引き出すのは難しい．本報告では，HPC アプリケーションの性能可搬性を調査するために，特定の HPC システム向けに適用された最適化手法を様々な HPC システムを用いて評価し，その効果と性能可搬性について議論する．

Performance Portability Issues on Modern HPC Systems

Kazuhiko Komatsu,[†1,†4] Ryusuke Egawa,[†1,†4] Ippei Yasuta,[†2] Akihiro Musa,[†3]
Kouji Matsuoka[†3] and Hiroaki Kobayashi [†1]

Since many types of HPC systems have been become avaiable recently, developing HPC applications that can exploit the potential of various HPC systems is getting very important. However, the HPC applications are not always the best ones for various HPC systems since HPC applications have been optimized for individual HPC system. This report discusses the performance portability of the basic optimization for individual HPC system through performance evaluations using 5 different HPC systems.

1. はじめに

近年，半導体微細加工技術の進歩によってプロセッサに搭載できるトランジスタ数が増加している．これによって，プロセッサの設計空間が広がり，様々なアーキテクチャのプロセッサが登場している．例えば，近年の汎用型スカラプロセッサには，複数の高性能コアと大容量キャッシュが搭載されている．さらには高性能コアだけでなく描画専用コアが搭載されている汎用型スカラプロセッサもある．

このようなプロセッサの多様化に伴い，先端科学や工学分野における大規模計算アプリケーション (HPC アプリケーション) を支える大規模計算システム (HPC システム) の多様化も同様に進んでいる．高い実効メモリバンド幅を有し，大量の要素を一括して演算可能なベクトル型スーパーコンピュータ[1),2)] や複数コアの汎用型スカラプロセッサを多数接続し並列計算を行うスカラ並列型スーパーコンピュータ[3)]，数百のシンプルなコアを持つアクセラレータと汎用型スカラプロセッサを混載するアクセラレータ型スーパーコンピュータ[4)] など，様々な種類の HPC システムが開発されている[5)]．

HPC システムの多様化によって，HPC アプリケーションは様々な環境で実行することが可能かどうかという**可搬性**だけでなく，様々な HPC システムの高い性能を引き出すことが可能かどうかという**性能可搬性**の重要性も急速に増している．

一般的に HPC アプリケーションのライフタイムは非常に長いと言われている[6)]．例えば，1 度作成された HPC アプリケーションがその時々の HPC システムに最適化・チューニングされながら，数十年間使われ続ける場合もある．しかしながら，このように HPC システムに合わせて進化し，高度に最適化された HPC アプリケーションが，必ずしも異なる HPC システムにおいて実行できる，または高い性能を引き出すことが可能であるとは限らない．ヘッダファイルや数値演算ライブラリ，プログラムの記述方法などのシステムに依存している箇所を修正することで実行することは

†1 東北大学サイバーサイエンスセンター
Cyberscience Center, Tohoku University
†2 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
†3 日本電気株式会社
NEC Corporation
†4 科学技術振興機構戦略的創造研究推進事業
Japan Science and Technology Agency, Core Research for Evolutional Science and Technology

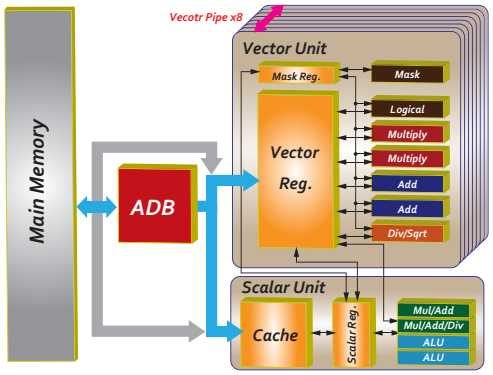

図 1 NEC SX-9 のアーキテクチャ

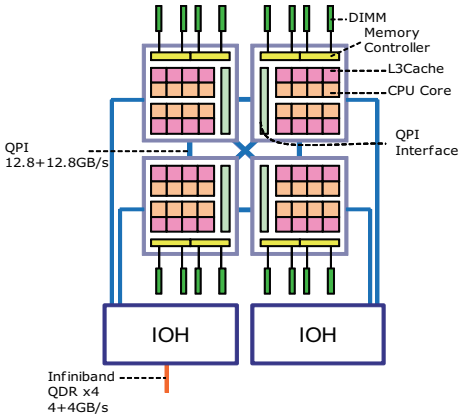

図 2 Intel Nehalem EX のアーキテクチャ

できても，HPC システムの本来の性能を最大限に発揮するためにはアルゴリズムから考慮し直すなどの大幅なコードの修正が必要になる可能性がある．

本報告では，HPC アプリケーションにおける性能可搬性を調査するために，特定の HPC システム向けに最適化された HPC アプリケーションを様々な HPC システムで評価する．HPC システムとしてベクトル型スーパーコンピュータ NEC SX-9，スカラ並列型スーパーコンピュータとして，Intel Nehalem EX Cluster, Fujitsu FX1，Fujitsu FX10，Hitachi SR16000 M1 の合計 5 つを取り上げ，各プラットフォームの特徴を考慮しつつ性能解析を行う．これによって，HPC アプリケーションにおいて，よく利用される最適化手法の性能可搬性について議論を行う．

2. 大規模計算システム

近年の HPC システムの性能向上は著しく，多種多様な HPC システムが登場している[5]．大量の計算を同時に処理することができるベクトルプロセッサを搭載するベクトル型スーパーコンピュータや汎用スカラプロセッサを搭載するスカラ並列型スーパーコンピュータ，シンプルなコアを多数搭載する描画処理用プロセッサ (Graphics Processing Unit, GPU) などを用いて計算を行うアクセラレータ型スーパーコンピュータなど，プロセッサのアーキテクチャやシステム構成によりその特徴が異なる．

本節では，本報告で取り上げる 5 つの HPC システムの概要とその特徴について述べる．

2.1 ベクトル型スーパーコンピュータ

NEC SX-9 は大規模な SMP(Symmetric Multi Processing) ノードから構成されるベクトル型スーパーコンピュータである[7]．図 1 に NEC SX-9 ベクトルプロセッサのアーキテクチャを示す．各 SMP ノードは理論性能 102.4Gflops/s のベクトルプロセッサを 16 個搭載しており，各ベクトルプロセッサでは 256 要素の同一演算を同時に処理することが可能である．SMP ノードは片方向 128GB/s の高速な専用クロスバースイッチ IXS により) 接続されている．

NEC SX-9 の特徴の 1 つとして，高いメモリバンド幅が挙げられる．メインメモリを 32768 個のメモリバンクに分割し，インタリーブにより複数のメモリバンクから同時にデータを転送することにより，1 ソケットあたり 256GB/s, 1SMP ノードあたり 4TB/s という非常に高いメモリバンド幅を実現する．メモリバンド幅で実効性能が律速される HPC アプリケーションも多く，この高いメモリバンド幅を活用することで高い実効性能を達成することができる．

また，NEC SX-9 は大規模共有メモリを有している．各ソケットとメモリがクロスバーで接続され，SMP ノード内の全てのプロセッサから全てのメモリへ直接アクセスすることができる．このような大容量かつ高いメモリバンド幅の共有メモリシステムによって，ノード内においてはプロセッサ間の通信がメモリアクセスと同等になるため，NEC SX-9 のアプリケーション開発者は容易に大規模 SMP 並列処理を行うことができる．

また，図 1 に示すように，NEC SX-9 にはオンチップメモリである ADB(Assignable Data Buffer) が搭載されている．ADB は 256KB の容量を持つソフトウェア制御が可能なオンチップキャッシュメモリである．HPC アプリケーションにおける再利用性の高いデータを ADB に保存し，演算に必要なデータを ADB とメインメモリから共に供給することができる．ADB を利用することで最大メモリバンド幅と演算性能の比が

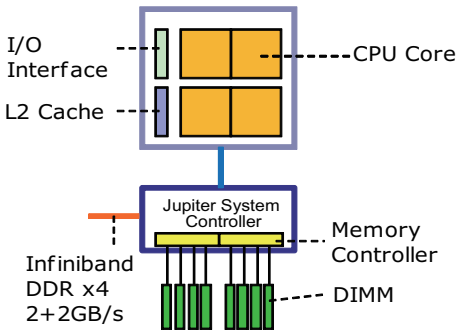

図 3 Fujitsu FX1 のノードアーキテクチャ

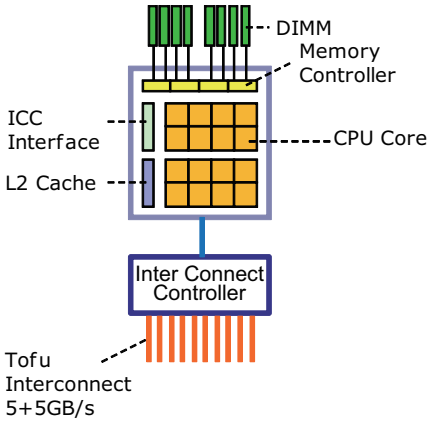

図 4 Fujitsu FX10 のノードアーキテクチャ

4Bytes/Flop と高くなり，メモリバンド幅への要求が高い科学技術計算においても高い実効性能を引き出すことが可能である．

2.2 スカラ並列型スーパーコンピュータ

Intel Nehalem EX クラスタ，Fujitsu FX-1, Fujitsu FX-10, Hitachi SR16000 M1 は，それぞれ Intel Nehalem EX, Fujitsu SPARC64VII, Fujitsu SPARC64IXfx, IBM Power 7 といった汎用型スカラプロセッサを多数搭載するスカラ並列型スーパーコンピュータである．複数のコアを持つ汎用型スカラプロセッサを 1 つまたは複数個用いて 1 ノードを構成し，このノードを Infiniband などの高速なネットワークで多数接続することで，数万から数百万のコアを搭載するスカラ並列型スーパーコンピュータを構築する．基本的には容易に並列化が可能なアプリケーションを膨大な数のコアで並列処理 (Massively Parallel Processing) を行うことで，高い演算性能を実現する．例えば，分子動力学や遺伝子解析，パラメータ探索などのアプリケーションは，各処理が独立しており，処理同士でのデータのやりとりが少ないため，大規模なスカラ並列型スーパーコンピュータに適している．

汎用型スカラプロセッサは，一般的に深いパイプライン段数で構成されるため，パイプラインをストールさせないように，投機的実行や out-of-order 実行を行う．また，浮動小数点演算性能を高めるために，複数のデータに対して同じ浮動小数点演算を実行することが可能な SIMD(Single Instruction Multiple Data) 演算器やメモリアクセスレイテンシを隠蔽するための大容量のオンチップキャッシュメモリを備えている．汎用型スカラプロセッサの性能を引き出すためには，SIMD 演算器や大容量キャッシュメモリを効率的に利用できるかが重要となる．

2.2.1 Intel Nehalem EX Cluster

図 2 に Intel Nehalem EX Cluster のアーキテクチャを示す[8]．1 ノードは 4 つの Intel Nehalem EX プロセッサで構成されており，ノード間は片方向 4GB/s の Infiniband で接続されている．各プロセッサは 8 つのコアを搭載しており，各プロセッサがそれぞれメモリを管理する NUMA アーキクチャである．プロセッサ間は QPI で接続されており，片方向あたり 12.8GB/s のバンド幅を持つ．

Intel Nehalem EX Cluster で高い実効性能を実現するためには，SIMD 演算器や大容量キャッシュメモリの活用の他に，自プロセッサが管理するメモリ領域へのファーストタッチを考慮したデータの保存やアフィニティの設定が重要となる．他のプロセッサが管理するメモリ領域へアクセスには，QPI を介したプロセス間通信が必要となり，メモリバンド幅やレイテンシなどメモリ性能が低下してしまい，HPC アプリケーションの実効性能を低下させる要因となりうる．

2.2.2 Fujitsu FX-1, FX10

図 3 に Fujitsu FX1 のアーキテクチャを示す[9]．1 つのノードに 4 コアを搭載する SPARC64VII プロセッサで構成されており，ノード間は片方向 2GB/s の Infiniband で接続されている．SPARC64VII は SIMD 演算器の他に，複数コア間の同期を高速に行うためのハードウェアバリア機構を備えている．これにより，複数スレッド間の同期オーバーヘッドを低減することで，スレッド並列性能を高めている．

理論メモリバンド幅と理論演算性能の比は 1Bytes/Flop と，他のスカラ並列型スーパーコンピュータと比べて高いが，STREAM benchmark により実効メモリバンド幅を測定すると，10GB/s と理論メモリバンド幅性能の約 1/4 となっている．

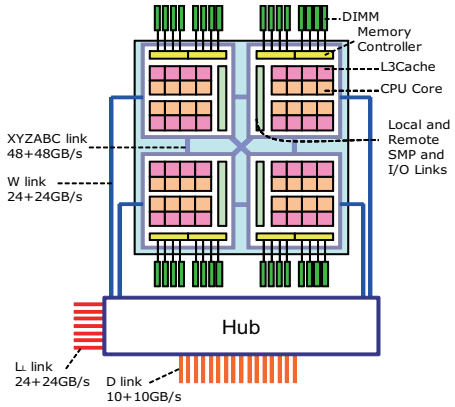

図 5 Hitachi SR16000 M1 のノードアーキテクチャ

図 4 に Fujitsu FX10 のアーキテクチャを示す[10]．1 つのノードに 16 コアを搭載する SPARC64IXfx プロセッサで構成されており，ノード間は 3 次元メッシュ/トーラスネットワークで接続されている．1Tofu グループである 12 計算ノード内は片方向 20GB/s, Tofu グループ同士は片方向 5GB/s で接続されている．SPARC64IXfx は，SIMD 演算の強化，レジスタの拡張，ソフトウェアで制御可能なキャッシュの搭載に加えて，メモリコントローラを内蔵することで，バンド幅を高めている．実効メモリバンド幅が理論メモリバンド幅の約 8 割程度と，SPARC64VII に比べて，実効メモリバンド幅効率が高くなっている．

2.2.3 Hitachi SR16000 M1

図 5 に Hitachi SR16000 M1 のアーキテクチャを示す[11]．ノード構成は 1 ノードは 4 つの IBM Power 7 プロセッサで構成されており，ノード間は片方向最大 24GB/s 階層型完全結合で接続されている．8 つのコアを搭載している IBM Power 7 は SMT(Simultaneous Multi-Threading) 機能をサポートしており，各コアに 2 スレッド割り当てることで，1 ノードで最大 64 スレッドまで SMP 並列処理が可能である．

また，Intel Nehalem Cluster と同様に，各プロセッサがそれぞれメモリ管理する NUMA アーキクチャであり，他のプロセッサが管理するメモリ領域へアクセスが必要な場合は，片方向あたり 48GB/s の XYZABC-link を介したプロセス間通信が必要となり，メモリバンド幅やレイテンシなどメモリ性能が低下する．

Hitachi SR16000 M1 は他のスカラ並列型スーパーコンピュータに比べ，メモリバンド幅が高く，比較的大きな SMP 並列処理が可能である．そのため，メモリ性能を引き出せるようにデータを配置することによって，並列処理が簡単な HPC アプリケーションだけでなく，依存関係や通信が必要な HPC アプリケーションにおいても他のスカラ並列型スーパーコンピュータに比べ高い性能を期待できる．

3. HPC アプリケーションにおける最適化手法

本節では HPC アプリケーションによく適用される代表的な最適化手法とその効果について概説する．

3.1 一時変数の利用

ループ中に同じ配列要素を複数回用いて演算を行う場合，同一データが複数回読み込まれ，冗長なメモリアクセスが発生する場合がある．一時変数にデータを保存することで，この冗長なメモリアクセスを削減することができるため，HPC アプリケーションを高速化できる．

しかしながら，一時変数の保存にはプロセッサ中のレジスタが使われるため，レジスタが不足してしまい，レジスタに保存されているデータをメモリに書き戻すレジスタスピルが発生してしまう恐れがある．その結果，メモリアクセスが増大してしまい，性能低下を引き起こす可能性がある．

3.2 ループ分散

ループ本体を分割し，複数のループに分けるループ分散を行うことで，ループ内の処理が大きい場合に発生するレジスタスピルを抑制することができる．さらに，ループ内に利用されるデータの参照局所性を高める効果がある．より多くの再利用性のあるデータがオンチップメモリに格納され，オンチップメモリが効率的に利用される可能性がある．

一方，ループ自体の数が増加するため，比較などのループ条件の判定や分岐などのループ制御が増え，性能低下を引き起こす可能性がある．このため，ループ分散では，利用レジスタ数の削減およびオンチップメモリの効率的な利用とループのオーバーヘッドによる性能低下とのトレードオフがある．

3.3 未定義変数の削除

ループ中の if 文などの条件文でのみ定義される変数が存在する場合，前のイタレーションで定義される変数を参照する可能性があるため，コンパイラによる自動並列化や自動ベクトル化ができない．ループのはじめに変数を定義し，ループ間の依存関係を解消することで，コンパイラによる自動並列化や自動ベクトル化を促進できる．

また，コードの可読性や保守性の観点からも変数を事前に定義し，未定義変数を削減する方が好ましい．

3.4 条件文のループ外への移動

ループ内にある if 文などの条件判定文をループの外に移動し，条件文中でループ処理を行う．これにより，

表 1 対象とする HPC システムのノード性能

HPC System	Peak Gflops/s	Sockets/node	Cores/socket	Memory BW GB/s	On-chip Memory	B/F
NEC SX-9	1676.8	16	1	256	256 KB ADB	2.5
Intel Nehalem EX	289.92	4	8	34.1	256 KB L2/core, 24 MB shared L3	0.47
Fujitsu FX1	41.28	1	4	40	6 MB shared L2	1.0
Fujitsu FX10	236	1	16	85	12 MB shared L2	0.36
Hitachi SR16000 M1	980.48	4	8	128	256 KB L2/core, 32 MB shared L3	0.52

コンパイラによる自動並列化や自動ベクトル化の阻害となる条件分岐処理がループ中になくなり，自動並列化や自動ベクトル化が促進される．しかしながら，条件判定文ごとに同じループを複数回記述する必要がある場合があり，コードの可読性や保守性が低下する可能性がある．

3.5 ループ展開

ループ展開は，複数の繰り返し演算を展開し，少数の繰り返しまたは 1 度で処理する手法である．これにより，ループの繰り返しに必要となるループ判定処理や分岐処理が削減できる．また，複数の繰り返しに含まれる同一のメモリアクセスが，1 回で済むためメモリアクセス数を削減できる．したがって，ループ中における演算の割合が増加し，演算器を効率的に利用することができる[12]．さらに，長さが短いループを展開することで，外側のループでの自動並列処理や自動ベクトル処理を促進する効果も見込める．

一方で，ループ展開によってメモリアクセス数が削減できない場合，ループに含まれるメモリアクセス数が増加する．その結果，外側ループにおける時間的局所性が減少し，キャッシュヒット率が低下する可能性がある．

3.6 ループ内不変量コードの移動

ループ内で毎回演算される同一結果の演算を，ループの前に移動し，1 度だけ計算する．ループ内ではその演算結果を参照することによって，冗長な演算を削減することができる．ループに依存しない計算を予めループの外に括り出すことで必要な演算が減るため，高速化が見込める．

3.7 ループの 1 重化

ループの 1 重化は複数のループを 1 つのループにまとめ，ループの長さを長くする．これにより，ベクトル処理において同時に演算できる要素数を増やすことができる．しかしながら，ループ展開と同様に，ループに含まれるメモリアクセス数が増加し，オンチップメモリの利用効率低下を招く可能性がある．したがって，ループ 1 重化は，ベクトル処理の演算効率向上とオンチップメモリの利用効率低下の可能性を考慮し適用する必要がある．

4. 各最適化手法の性能可搬性評価

本節では，3 章に挙げた代表的な最適化手法を NEC SX-9 向けに適用し，2 章で取り上げた表 1 の HPC システムを用いて，性能評価を行う．各 HPC システムの特徴に基づき，性能評価結果の考察を行い，性能可搬性について議論を行う．

4.1 性能評価環境

HPC アプリケーションとして，東北大学サイバーサイエンスセンターで実際に利用されている海流シミュレーションや飛行機周りの流体解析シミュレーション，ナノプラズマの生成シミュレーション，地震波の伝搬シミュレーション，ジェットエンジンの混合燃焼シミュレーションの 5 つの実アプリケーションを用いる．各 HPC アプリケーションにおける主要カーネルを抜き出し，それぞれのカーネルに対して NEC SX-9 向けの最適化を行う．特定の HPC システムに最適化されたカーネルを様々な HPC システムで評価することにより，性能可搬性について議論する．海流シミュレーションには一時変数の利用，流体解析シミュレーションにはループ分散，ナノプラズマ生成シミュレーションの 2 つのカーネルにはそれぞれ未定義変数の削除と条件文の条件文のループ外への移動，地震波の伝搬シミュレーションにはループ内不変量コードの移動とループ展開の両方，混合燃焼シミュレーションにはループの 1 重化を適用する．

4.2 マルチプラットフォーム環境における性能可搬性評価

図 6 から図 11 に各 HPC における最適化手法の速度向上率をそれぞれ示す．横軸に HPC システム，縦軸に最適化手法を適用することによる速度向上率を示す．

図 6 に一時変数の利用による速度向上を示す．評価用カーネルでは，ループ中に 2 度読み込んでいる同じ配列の要素を一時変数に代入し，計算結果を配列に書き戻している．図 6 を見ると，NEC SX-9 では約 1.9 倍の速度向上を得られている．SX-9 のメモリレイテンシが大きくため，最適化を適用する前には 1 回目のデータ転送が完了する前に，次の同一のデータ転送命令が発行されていたが，最適化適用後には一時変数を

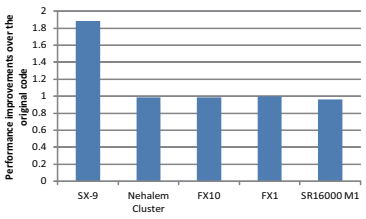

図 6 一時変数の利用による速度向上

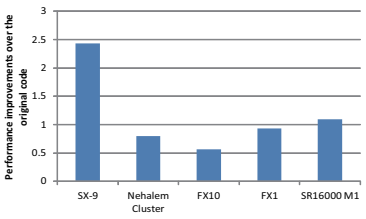

図 7 ループ分散による速度向上率

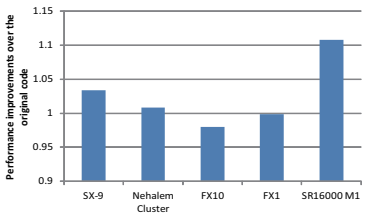

図 8 未定義変数の削除による速度向上率

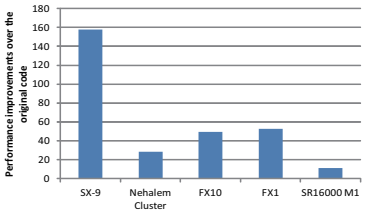

図 9 条件文の条件文のループ外への移動による速度向上

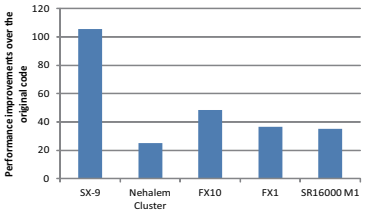

図 10 不変量コードの移動とループ展開による速度向上

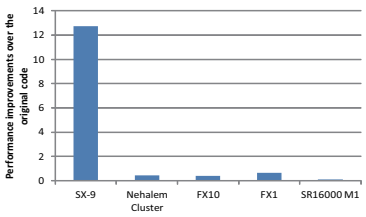

図 11 ループ 1 重化による速度向上

利用することで 2 回目のデータ転送を削減することが出来た．しかしながら，スカラ並列型スーパーコンピュータにおいては一時変数の利用による性能の変化はほとんど見られない．これは汎用型スカラプロセッサの大容量キャッシュに必要なデータが保存されるため，一時変数を用いなくとも冗長なメモリアクセスが発生しないためだと考えられる．以上より，一時変数の利用による最適化はメモリアクセスレイテンシが長いプロセッサで効果的であると考えられる．また，この最適化を適用しても性能が低下するなど悪影響は見られず，性能が向上する可能性があるため，性能可搬性は良いと言える．

図 7 にループ分散による効果を示す．評価用カーネルでは，ループ内の処理が大きくレジスタスピルが発生する可能性があるため，ループ分散によりレジスタスピルの抑制とオンチップメモリの効率的利用が見込める．図 7 を見ると，SX-9 ではループ内の処理を分割することにより演算に必要なレジスタ数を削減することができ，また再利用性のあるデータをより多く ADB に格納できたため，約 2.4 倍の性能向上が得られている．またスカラレジスタ数が比較的多い Hitachi SR16000 M1 でも約 10%の速度向上が得られている．一方，その他の HPC システムでは，ループ分散による利用されるレジスタ数の削減効果が表れていない．むしろ，ループの条件判定や分岐などの制御が増加し，性能低下を引き起こしている．特に Fujitsu FX10 や Intel Nehalem では，約 45%，約 20%もそれぞれ性能が低下している．1core での実験において同程度性能が低下を引き起こしている．ループ分散は HPC システムに応じて性能が 2 倍以上も向上する場合もあれば，性能が半分近くまで低下する場合もあり，今回の分割粒度においては性能可搬性が高くないと言える．

図 8 に未定義変数の削減による効果を示す．評価用カーネルではループ内の if 文が真の時のみ変数が定義されるため，ループ間の依存関係を解消できずにいた．ループ内の最初に同じ if 文を用いて変数を定義または初期化することにより，未定義変数を削減し，コンパイラによる並列化およびベクトル化を促進する．図 8 を見ると，最大で約 10%程度の速度向上が見られる．Hitachi SR16000 M1 や SX-9, Intel Nehalem EX では未定義変数を削除することにより，自動ベクトル化が促進され速度が向上する．意図的な未定義変数以外は基本的に削除すべきであり，性能可搬性が高いと言える．また，未定義変数に関してはコンパイラによる取り扱いの違いが性能に影響する可能性もあり，最適化手法だけでなくコンパイラ自体も HPC アプリケーションの性能可搬性を検討対象になりうると考えられる．

図 9 に条件文のループ外への移動の効果を示す．評価用カーネルでは，ループ内のいくつかの if 文で参照している変数が不変なため，ループの外に出すことにより，ループ内の並列化やベクトル化を促進する．図 9 を見ると，約 11 から 158 倍の速度向上と条件文の追い出しが全ての HPC システムにおいて非常に効果が高い．特にベクトル型スーパーコンピュータである NEC SX-9 において，効率的に条件分岐をベクトル処理するにはマスクを用いた演算が必要になるが，ループの外へ条件文を移動することによって，ループ内の処理が効率的にベクトル化されたため，非常に高い速

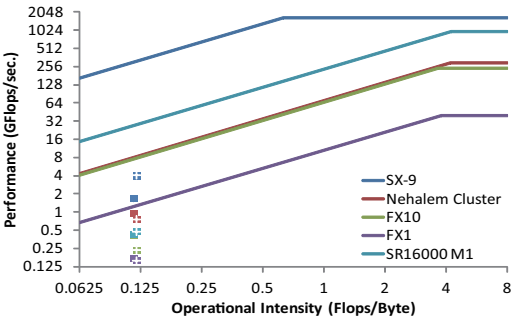

図 12 ルーフラインモデルによるループ分散の性能解析

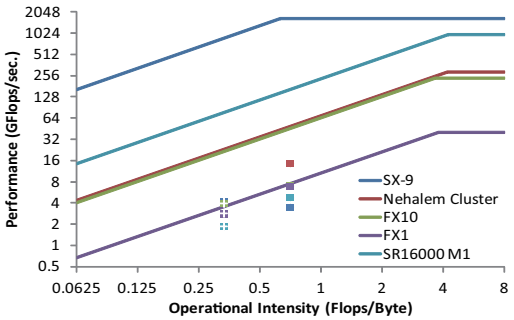

図 13 ルーフラインモデルによる不変量コードの移動の性能解析

度向上を達成している．以上より，条件文のループ外への移動は HPC システムによらず効果が高く，性能可搬性は非常に高く有用である．

図 10 にループ内不変量コードの移動とループ展開による効果を示す．評価カーネルではループ内に多くあるループに依存しない計算をループ外で出すことにより，演算の削減を行う．また，ループの長さが短いループを展開し，自動並列化や自動ベクトル化を促している．図 10 を見ると，約 24 から 105 倍と全ての HPC システムにおいて高い速度向上が得られている．ループ内不変量コードの移動は演算が大幅に削減することができるため，いずれの HPC システムも効果が高く，性能可搬性は非常に高く有用な最適化手法である．

図 11 にループ 1 重化による効果を示す．評価用カーネルでは，効率的なベクトル処理を実現するために，3 重ループを 1 重化してループ長を伸ばしている．これにより，より多くの要素を同時に計算することが可能になる NEC SX-9 ではベクトル長の増加により約 13 倍の速度向上が得られている．しかしながら，スカラ並列型スーパーコンピュータでは 4 から 9 割ほど性能の低下が見られる．これはループ内で利用されるデータが多く，キャッシュ内に収まらなくなり，キャッシュヒット率の低下を招いたためだと考えられる．特に，Hitachi SR16000 M1 では約 90%，Fujitsu FX10 では約 60%も性能が低下している．搭載されるコアの数が多く，コア当たりの共有キャッシュの容量が少ないため，影響が大きかったと考えられる．以上より，ループ 1 重化はベクトルプロセッサでは非常に有効であるが，汎用型スカラプロセッサでは性能低下を招きやすく，性能可搬性が低いことが分かる．

4.3 ルーフラインモデルを用いた性能解析

多くの HPC アプリケーションにおける実効性能はメモリバンド幅に依存していると言われており，Williams らはメモリバンド幅を考慮したプロセッサの性能モデルとして，ルーフラインモデルを提案している[13]．ルーフラインモデルはアプリケーションに含まれる演算量とメモリから転送されるデータ量の比 Flops/Byte を演算密度と定義し，演算密度が低いアプリケーションではメモリバンド幅が，演算密度が高いアプリケーションでは演算性能が実効性能を律速する性能モデルである．このルーフラインモデルを用いることで，各 HPC システムの特徴を考慮した性能解析が可能である．

4.2 節で性能可搬性が高くなかったループ分散と高かったループ内不変量コードの移動の効果をルーフラインモデルを使って示した物を図 12 と図 13 にそれぞれ示す．単色四角マークが最適化を適用しない場合，四角に白十字マークが最適化を適用した場合を示している．各色がそれぞれの HPC システムに対応する．

図 12 を見ると，ループ分散によってカーネルの Flops/Byte はほぼ変化がない．これはループ分散によって増加するのは条件判定や分岐であり，浮動小数点演算やメモリアクセスがほぼ変わらないためである．また，ループ分散による最適化後の実効性能はレジスタスピルを抑制した NEC SX-9 の実効性能が向上している．各 HPC システムにおける実効性能を見ると，メモリバンド幅による実効性能の上限までまだ余地がある．そのため，レジスタ数の少ない HPC システムにおいても，さらに細かくループ本体を分割し，レジスタスピルを解消することで性能向上をもたらす可能性がある．分割方法の粒度を変更し，より詳細に評価を行い性能可搬性を調査する必要がある．

4.2 節の結果からも冗長な演算を削減するループ内不変量コードの移動は，どの HPC システムにおいても有効であり，性能可搬性が高い．図 13 を見ると，ループ内の演算が削減されるため，最適化前と後で演算密度が小さくなり，ルーフラインモデル上のそれぞれの点は左側に移動する．したがって，メモリ性能によって実効性能が律速されやすくなるため，オンチップメモリの利用など，演算密度を高める効果がある最適化との相性が良いことが分かる．図 13 において，実効

性能がメモリ性能に律速されている Fujitsu FX1 では，これ以上演算性能を高めても実効性能が向上する余地はなく，演算密度を高める最適化が必要があるのが分かる．

5. おわりに

HPC システムの多様化が進むにつれ，可搬性および性能可搬性の高い HPC アプリケーションへの需要が高まっている．現在の HPC アプリケーションは 1 つの HPC システムに合わせて高度に最適化されており，必ずしも他の HPC システムにおいて高い性能を引き出すことができるとは限らない．このような背景のもと，本報告では，現状の HPC アプリケーションの性能可搬性を調査するために，ベクトル型スーパーコンピュータ向けの最適化手法を様々な HPC システムを用いて評価を行った．評価結果に基づきて解析を行うことによって，各最適化手法の性能可搬性について議論した．今後の課題として，スカラ並列型スーパーコンピュータ向けの最適化手法も対象として，スカラ並列型スーパーコンピュータ同士の効果の違いを明らかにすることが挙げられる．

6. 謝　辞

本研究は，北海道大学情報基盤センター，東北大学サイバーサイエンスセンター，東京大学情報基盤センター，名古屋大学情報基盤センターのスーパーコンピュータを利用することで実現することができた．本研究の一部は，文部科学省科研費研究 (S)(21226018) と科学技術振興機構 (JST) 戦略的創造研究推進事業 (CREST) 研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」研究課題「進化的アプローチによる超並列複合システム向け開発環境の創出」の助成を受けている．

参考文献

1) Takashi Soga, Akihiro Musa, Youichi Shimomura, Ryusuke Egawa, Ken'ichi Itakura, Hiroyuki Takizawa, Koki Okabe, and Hiroaki Kobayashi. Performance evaluation of nec sx-9 using real science and engineering applications. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pp. 28:1–28:12, 2009.
2) Thomas H. Dunigan Jr., Jeffrey S. Vetter, James B. White III, and Patrick H. Worley. Performance evaluation of the cray x1 distributed shared-memory architecture. *IEEE Micro*, Vol. 25, No. 1, pp. 30–40, January 2005.
3) Yukihiro Hasegawa, Jun-Ichi Iwata, Miwako Tsuji, Daisuke Takahashi, Atsushi Oshiyama, Kazuo Minami, Taisuke Boku, Fumiyoshi Shoji, Atsuya Uno, Motoyoshi Kurokawa, Hikaru Inoue, Ikuo Miyoshi, and Mitsuo Yokokawa. First-principles calculations of electron states of a silicon nanowire with 100,000 atoms on the k computer. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pp. 1:1–1:11, 2011.
4) Abtin Rahimian, Ilya Lashuk, Shravan Veerapaneni, Aparna Chandramowlishwaran, Dhairya Malhotra, Logan Moon, Rahul Sampath, Aashay Shringarpure, Jeffrey Vetter, Richard Vuduc, Denis Zorin, and George Biros. Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pp. 1–11, 2010.
5) Top 500 supercomputers sites. http://www.top500.org/.
6) D. Dig F. Kjolstad and M. Snir. Bringing the HPC Programmer's IDE into the 21st Century through Refactoring. In *SPLASH 2010 Workshop on Concurrency for the Application Programmer (CAP'10)*, Oct. 2010.
7) Sx-9 装置緒元 : Hpc ソリューション — nec:. http://www.nec.co.jp/solution/hpc/sx9/product/spec.html.
8) Express5800/a1080a - nec:. http://www.nec-itplatform.com/-Express5800-A1080a-.html.
9) Hpc ハイエンドテクニカルコンピューティングサーバ fx1 : 富士通:. http://jp.fujitsu.com/solutions/hpc/products/fx1.html.
10) Specifications : Primehpc fx10 : Fujitsu global:. http://www.fujitsu.com/global/services/solutions/tc/hpc/products/primehpc/spec/.
11) Sr16000：仕様：技術計算向けサーバ：日立:. http://www.hitachi.co.jp/Prod/comp/hpc/SR_series/sr16000/spec.html.
12) Steve Carr and Ken Kennedy. Improving the ratio of memory operations to floating-point operations in loops. *ACM Trans. Program. Lang. Syst.*, Vol. 16, No.6, pp. 1768–1810, November 1994.
13) Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, Vol.52, No.4, pp. 65–76, April 2009.