Vol.2009-ARC-184 No.6
2009/8/4

キャッシュメモリを有するベクトルプロセッサのためのプログラム最適化手法

佐藤義永[†1]　永岡龍一[†1]　撫佐昭裕[†2]
江川隆輔[†1]　滝沢寛之[†1]
岡部公起[†1]　小林広明[†1]

近年，ベクトルプロセッサにおいて演算性能に対する相対的なメモリバンド幅 (B/F) が低下しており，実行効率の低下が懸念されている．B/F 低下の影響を緩和するために，高いメモリバンド幅を有するキャッシュメモリを搭載することが検討され，その有効性が明らかになっている．そこで，キャッシュの性能をさらに引き出すためのプログラム最適化手法の確立が本報告の目的である．本報告では，キャッシュと性能の関係を解析するために，ルーフラインモデルを用いてキャッシュメモリを有するベクトルプロセッサの性能モデルを構築する．そして，実アプリケーションにプログラム最適化を施し，プログラム最適化の効果を性能モデルを用いて評価する．

Performance Optimization Techniques for Vector Processors with Cache Memory

Yoshiei Sato,[†1] Ryuichi Nagaoka,[†1] Akihiro Musa,[†2]
Ryusuke Egawa,[†1] Hiroyuki Takizawa,[†1] Koki Okabe[†1]
and Hiroaki Kobayashi [†1]

Since the ratio of memory bandwidth to computational performance(B/F) recently decreases, it is concerned that the sustained performance of future vector processors degrades. To reduce the performance degradation due to the decrease in B/F, vector cache memory with high memory bandwidth has been proposed and evaluated. The purpose of this paper is to establish the optimization techniques to further exploit the vector cache memory performance. To analyze the relationship between the vector cache memory and the sustained performance, this paper first presents a performance model of vector processors with vector cache memory based on the roofline model. Then, several optimization techniques are applied to real applications, and their effects are assessed with the performance model.

1. はじめに

大規模な科学技術計算におけるベクトル型スーパーコンピュータの優位性は，その高い実効メモリバンド幅によって支えられていが，今まで以上のメモリバンド幅の向上は CPU に実装可能な入出力ピン数の物理的限界により困難になってきている．その一方で，CPU の演算性能は半導体チップの微細化による集積度の向上により，メモリバンド幅以上に向上している．このため，メモリ性能と演算性能の比である B/F は減少傾向にある．その結果，演算性能に見合ったデータ供給がなされず，理論最大性能に対する実効性能の比である**実行効率**が大きく低下する[1)]．

低下する B/F の影響を軽減するために、ベクトルプロセッサにもキャッシュメモリを設けるという提案がなされている[2)]．プロセッサ内部にキャッシュメモリを設けることで，キャッシュメモリとベクトルレジスタ間のメモリバンド幅は，メインメモリのメモリバンド幅よりも高く保つことが可能となる．現在，ベクトルプロセッサ向けのキャッシュメモリは実用化されており，Cray X1 や NEC SX-9 などのスーパーコンピュータで採用されている．これらのスーパーコンピュータの性能評価に関する研究はすでに多数報告されており，その高い演算性能が明らかとなっているが，キャッシュメモリと性能の関係を定量的に評価するには至っていない[4)5)6)7)]．ベクトルコンピュータは大規模な問題を扱うことから，限られたキャッシュメモリ容量を用いて，高い実効性能を実現するためには，キャッシュメモリ容量を効率よく利用する最適化手法が重要な鍵となる．そこで，キャッシュメモリがベクトルプロセッサの性能に与える効果の定量的な解析と，それに基づくキャッシュメモリの効果を引き出す最適化手法の確立が求められている．

本研究の目的は，キャッシュメモリを最大限に活用するためのプログラム最適化手法を確立することである．本報告では，ベクトルプロセッサ向けキャッシュメモリであるベクトルキャッシュ[3)] を対象とし，その有効活用を実現するプログラム最適化方針を明らかにする．まず，ベクトルキャッシュを有するベクトルプロセッサの性能モデルを示す．その後，実アプリケーションにプログラム最適化を施し，得られる実効性能と性能モデルを比較すること

†1 東北大学
Tohoku University
†2 日本電気株式会社
NEC Corporation

でプログラム最適化の効果を評価する．また，ベクトルキャッシュやプログラム最適化利用時の消費エネルギも評価し，実効性能と消費エネルギの両面からプログラム最適化の効果を検証する．

2. 関連研究

2.1 ベクトルプロセッサにおけるキャッシュメモリ

現在，キャッシュメモリは 2 種類のベクトルプロセッサにおいて実用化されている．一つは Cray 社の Cray X1 や Cray Black Widow[10] に搭載されているベクトルプロセッサ，もう一つが NEC 社の SX-9 に搭載されているベクトルプロセッサ[9] である．

Cray X1 では，ECache と呼ばれる 512KB の容量を持つキャッシュメモリを備えており，4 つのベクトルプロセッサで 4 つの ECache を共有する構成となっている．その結果，メインメモリとベクトルレジスタ間は 2.7B/F であるのに対して，ECache とベクトルレジスタ間を 4B/F とすることが可能であり，これにより高い実効メモリバンド幅を有するシステムとなっている．Cray X1 に関する研究では，これまでシステム全体の性能評価はなされているものの，キャッシュメモリの有効性について定量的評価はなされていない[4][5][6]．

また，NEC SX-9 は Assignable Data Buffer(ADB) と呼ばれる 256KB のオンチップメモリをプロセッサ内部に備えており，プログラマが任意にデータを ADB に格納することが可能である[9]．その結果，メインメモリとベクトルレジスタ間は 2.5B/F であるのに対して，ADB とベクトルレジスタ間を 4B/F にすることが可能である．SX-9 の性能評価では，流体解析などの実アプリケーションにおいて高い実効性能が得られるという評価が得られている．また，地震解析プログラムや FDTD 法を用いたアプリケーションにおいて ADB の利用により 1.2 倍から 1.7 倍という高い ADB の効果が示されている[7]．

以上の関連研究から，ベクトルプロセッサ向けのキャッシュメモリは性能向上に貢献することが分かるが，キャッシュメモリが性能に寄与する詳細な解析は行われておらず，キャッシュメモリの利用法の指針は未だ不明瞭なままである．したがって，キャッシュメモリによって得られる効果を解析するために，キャッシュメモリを備えたベクトルプロセッサの性能モデルが必要である．また，性能モデルを用いてキャッシュメモリの効果を解析することで，キャッシュメモリの利用を考慮したプログラム最適化指針を得ることが可能となる．

2.2 ルーフラインモデル

ベクトルプロセッサで利用される多くの科学技術アプリケーションでは，大量のメモリアクセスが行われることから，実効性能はメモリバンド幅に大きく左右される．そのことから，ベクトルプロセッサの性能モデルはメモリバンド幅を考慮する必要がある．メモリバンド幅を考慮した性能モデルとして，ルーフラインモデルが Williams らにより提案されている[11]．アプリケーションが必要とするメモリバンド幅から，実効性能は演算性能とメモリバンド幅のどちらか一方に律速される．アプリケーションに含まれる演算量とメインメモリから転送されるデータ量の比を *Operational Intensity(Flops/Byte)* として定義した場合，Operational Intensity が低いアプリケーションではメモリバンド幅が実効性能を制約し，Operational Intensity が高いアプリケーションでは演算性能が制約する．そこで，Operational Intensity から得られる理論実効性能をルーフラインとして表現し，また，プロセッサ各々の特徴によって律速される性能を上限 (シーリング) としてルーフラインモデル中で表現する．実際にアプリケーションの性能解析を行う際には，アプリケーションの実効性能と Operational Intensity をルーフラインモデルと比較することにより，ボトルネックの解析が可能となる．ルーフラインモデルは，アプリケーションの特徴やプロセッサが有する演算性能・メモリバンド幅の関係に基づいた性能モデルであるため，本研究で扱うキャッシュメモリの実効メモリバンド幅向上の影響や，プログラム最適化の効果をアプリケーションの特徴に基づいて解析可能である．そこで，本報告ではルーフラインモデルをベクトルプロセッサに適用し，プログラム最適化の効果を評価する．

3. ベクトルキャッシュのルーフラインモデル

3.1 ベクトルキャッシュを有するベクトルプロセッサの概要

本報告で対象とするプロセッサモデルについて述べる．ここで，本報告で用いるプロセッサのキャッシュメモリと他のプロセッサで利用されているキャッシュメモリを区別するために，本報告で用いるキャッシュメモリを**ベクトルキャッシュ**と呼称する．ベクトルキャッシュを備えるベクトルプロセッサの構成を図 1 に示す．ベクトルキャッシュは，各メモリポートに対応させてサブキャッシュとして分割する．メモリポート数を N とすると，N 個のサブキャッシュでベクトルデータを分散して保持する．その後，ベクトルキャッシュに保存したベクトルデータが再利用され，ベクトルキャッシュからベクトルレジスタに短いメモリアクセスレイテンシと高いメモリバンド幅でデータを供給する．また，ベクトルキャッシュにはバイパス機構と Miss Status Handling Register(MSHR)[8] を設けることでキャッシュメモリの利用効率を高めている．

バイパス機構とは，キャッシュメモリを経由せずに，メインメモリからベクトルレジスタへ直接データを転送する機構である．バイパス機構を用いることで，ベクトルキャッシュに

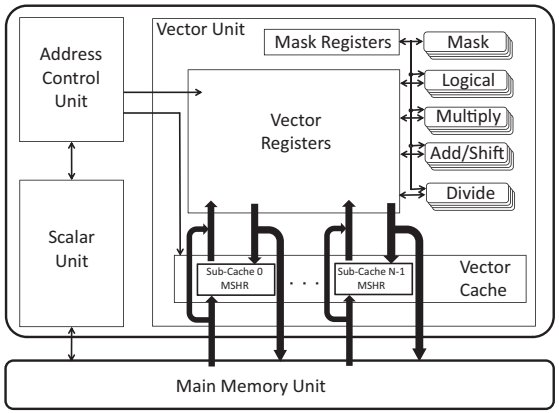

図 1 ベクトルキャッシュを有するベクトルプロセッサ

格納するデータをプログラマが選択することが可能となり，時間的局所性の高いデータのみを効率的にキャッシュに格納することで，データの再利用性が高まりキャッシュヒット率の向上が期待できる．

MSHR とは，キャッシュミス時のアドレス情報管理を行うレジスタであり，ノンブロッキングキャッシュを実現する機構である．さらに，MSHR のアドレス情報管理により，キャッシュミス時の冗長なメモリアクセスを回避することが可能となる．先行するメモリアクセスがキャッシュミスし，後続のメモリアクセスも同じキャッシュラインであった場合，先行のメモリアクセスが完了していなければ後続のメモリアクセスもメインメモリへ要求を送り冗長なメインメモリアクセスが生じる．しかし，MSHR を導入することにより，後続のメモリアクセスでキャッシュミスが起こった場合，MSHR を確認して既にメインメモリへアクセス中のキャッシュラインであればメインメモリへのアクセスは行わず，データの到着を待ってメモリアクセスを完了させる．このように MSHR を利用することで，冗長なメモリアクセスを回避することが可能となる．

本報告では，将来 B/F が低下することを仮定し，メインメモリと CPU 間のデータ転送性能は 2B/F とする．一方で，ベクトルキャッシュを CPU のチップ上に設けることで，ベクトルキャッシュとベクトルレジスタ間は 4B/F を維持することができ，ベクトルキャッシュからデータを転送することで性能向上が期待できる．

3.2 ベクトルキャッシュを有するベクトルプロセッサのルーフラインモデル

Williams らによって提案されているルーフラインモデルは，メインメモリとキャッシュメモリ間の実効メモリバンド幅からルーフラインを構築している．しかし，ベクトルキャッシュは実効メモリバンド幅を向上させる機構であるため，本報告ではベクトルキャッシュの最大メモリバンド幅をルーフラインとする．そして，メインメモリのメモリバンド幅は，キャッシュメモリを利用できない場合の性能の上限 (シーリング) のひとつとしてルーフラインモデル中で表現される．また，本研究で想定するプロセッサの演算器は，加算器と乗算器が独立していることから，最大演算性能は双方の演算器が稼働する場合と定義する．したがって，どちらか一方のみしか稼働できない場合，ベクトルプロセッサは最大演算性能の 50 %しか発揮できない．さらにベクトル化率やベクトル長によっても演算性能は低下するため，これらの条件もシーリングとして本性能モデル中で考慮される．ベクトルキャッシュを有するベクトルプロセッサにおけるルーフラインモデルを図 2 に示す．なお，図中における V.Op.ratio はベクトル化率，V.Len はベクトル長，balanced mul/add は加算と乗算の演算割合の均一さを示している．

図 2 より，Operational Intensity が 1/2 以上であればメモリバンド幅はボトルネックになることはない．Operational Intensity が 1/4 以上では，メインメモリがボトルネックとなるが，ベクトルキャッシュの利用でき，加算器と乗算器を同時に稼働可能なアプリケーションの場合には，高い実行効率が達成可能である．Operational Intensity が 1/4 以下では，ベクトルキャッシュを利用しても最大演算性能を達成することは不可能であるが，ベクトルキャッシュの利用により，利用しない場合と比較して最大で 2 倍の性能向上が期待できるため，ベクトルキャッシュを有効に利用するプログラム最適化が性能向上の鍵となる．

4. プログラム最適化

問題サイズが大きく, 再利用するまでの間隔が長いプログラムでは，データを再利用する前に他のデータによって置換され，データの再利用ができない恐れがある．特にベクトルプロセッサで利用されるアプリケーションは一般的にキャッシュメモリサイズをはるかに越える問題サイズである場合が多い．そこで，限られたキャッシュメモリを有効に利用するために，局所性が存在するデータのみに限定してベクトルキャッシュに格納する選択的キャッシングとループ長を変化させることで局所性を高めるキャッシュブロッキングを利用する．また，ベクトルプロセッサを効率よく利用するための最適化であるループアンローリングによる最適化も考えられる．一方で，これらの最適化は一長一短の性質を持つ．そこで，ループ

Vol.2009-ARC-184 No.6
2009/8/4

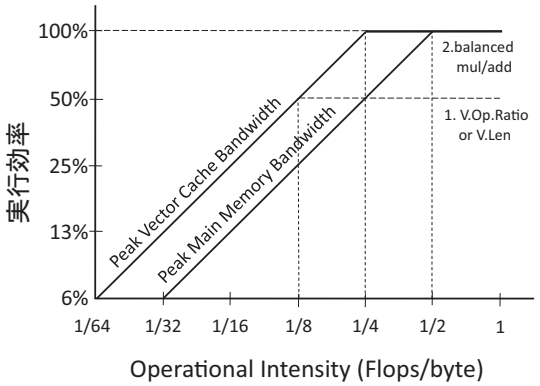

図 2 ベクトルキャッシュを有するベクトルプロセッサのルーフラインモデル

アンローリング，選択的キャッシング，キャッシュブロッキングの特徴を考察し，これらの最適化手法を組み合わせてプログラムの最適化を行う．

4.1 ループアンローリング

ループアンローリングはループを展開することで複数の繰り返し演算を，一度のループ内で処理する手法である．複数回実行されるループを 1 つのループに展開することで，ループ間で共通するデータを 1 度のメモリアクセスのみで同時に利用可能になるため，メモリアクセス数が削減できる．したがって，ループアンローリングを行うことにより，演算数に対するデータ転送量が減少するため，Operational Intensity が向上する．Operational Intensity が向上した結果，メモリバンド幅のボトルネックが緩和され，実効性能が向上する．

一方で，ループアンローリングによる時間的局所性に着目すると，外側のループが展開されることにより，内側ループで行われるベクトルロード数は増加する．その結果，外側ループ間に存在する時間的局所性が向上する一方で，内側ループの局所性は低下する恐れがある．よって，ループアンローリングを施した結果，再利用可能なデータは減少するためキャッシュヒット率が低下し，有効にベクトルキャッシュを利用できない恐れがある．

4.2 選択的キャッシング

データ参照の局所性が高いメモリアクセスと，局所性が低いメモリアクセスが混在するプログラムの場合，局所性が低いメモリアクセスによって局所性が高いデータが上書きされキャッシュヒット率が低下する．そこで，時間的局所性が高いデータに限定してベクトルキャッシュへ格納する手法が選択的キャッシングである．選択的キャッシングでは，局所性が高いデータのみをキャッシュメモリに格納し，一方の局所性が低いデータはベクトルキャッシュをバイパスさせる．それにより，再利用が期待できる局所性の高いデータを，より多くキャッシュメモリに格納可能になるため，結果としてキャッシュヒット率が向上する．

4.3 キャッシュブロッキング

キャッシュブロッキングは，ループを分割することでループ長を短くし，時間的局所性を高める最適化である．キャッシュブロッキングにより時間的局所性を高めることで，再利用性のあるデータをより多くベクトルキャッシュに格納することができ，キャッシュヒット率の向上が期待できる．その一方で，キャッシュブロッキングを行うためのループ分割によりループ長が短くなるため，ベクトル長が短くなる．ベクトル命令で一括して行う要素数が少なくなるため，メモリアクセスレイテンシの隠蔽が困難になる．このため，キャッシュブロッキングによる最適化では，ブロッキングするループ長の長さに応じて，キャッシュヒット率とベクトル長にトレードオフの関係がある．したがって，最適化の対象となるプログラムが必要とするベクトル長を意識してキャッシュブロッキングを行う必要がある．

5. 実アプリケーションによる評価

実アプリケーションに 4 章で示したプログラム最適化を施して評価し，評価結果をルーフラインモデルと対応させることで，プログラム最適化の有効性を検証する．また，プログラム最適化の有無による消費エネルギへの影響も評価し，消費エネルギの面からもプログラム最適化の有効性を検証する．

5.1 評 価 環 境

本評価では，SX ベクトルプロセッサのタイミングシミュレータを用いて評価を行う．本評価で用いるタイミングシミュレータは，実行される各命令の実行タイミングや，実行時のサイクル数，メモリアクセスサイクル数等を計測することで，全実行時間やキャッシュヒット率を算出するシミュレータである．本評価におけるシミュレーションの手順を示す．初めに，SX ベクトル型スーパーコンピュータの実機において，評価対象プログラムを実行し，その際に実行される命令列をトレースデータとして記録する．このトレースデータを SX タイミングシミュレータの入力として，実行時間，キャッシュヒット率を得る．また，SX タイミングシミュレータに与えるパラメータを変更することで，ベクトルキャッシュの有無やプロセッサの構成，メモリバンド幅の変更も可能である．本評価で用いるシミュレーション

表 1 シミュレーションパラメータ

Base System Architecture	NEC SX-7
Main Memory	DDR-SDRAM
Vector Cache	SRAM
Vector Cache Size	1MB
Number of Sub-caches	32
Associativity	2-Way
Cache Policy	LRU，Write-through
Cache Bank Cycle	5 % of memory cycle
Cache Latency	15 % of memory latency
Line Size	8B
MSHR Entries(Sub-cache)	8192(256)
Memory-Cache bandwidth per flop/s	2B/F
Cache-Register bandwidth per flop/s	4B/F

パラメータを表 1 に示す．

5.2 評価アプリケーション

評価アプリケーションの一覧を表 2 に示す．姫野ベンチマーク以外は，東北大学サイバーサイエンスセンターのスーパーコンピュータで実際に利用されている実アプリケーションである．

Earthquake[12] は 3 次元プレート沈み込みモデルの数値解析シミュレーションプログラムである．特徴として，主要カーネルはグリーン関数の解法であり，全実行時間の 83 %がメモリアクセス時間で占められている．

Land Mine[14] は地雷探査を行うシミュレーションプログラムである．特徴として，主要カーネルである FDTD(Finite Difference Time Domain) の解法が全体実行時間の 80 %を占めている．また，全実行時間の 99 %がメモリアクセス時間で占められるため，メモリバンド幅の影響が大きいプログラムである．また，差分式であるという特徴から，繰り返し利用されるデータをベクトルキャッシュに保持することでメインメモリへのアクセス回数を大幅に削減できるプログラムである．

姫野ベンチマーク[13] は，ポアッソン方程式をヤコビの反復法で解く際の主要ループに対する性能を評価するベンチマークプログラムである．特徴として，全実行時間の 99 %がメモリアクセス時間で占められ，メモリバンド幅の影響が大きいプログラムである．したがって，B/F 低下の影響を緩和するためにベクトルキャッシュの利用が不可欠なプログラムと言える．また，姫野ベンチマークのカーネル部分は，複数の配列へのメモリアクセスが存在しており，時間的局所性のある配列とない配列が混在するという特徴がある．このことから，時間的局所性のある配列のみをキャッシュに載せる必要がある．

表 2 評価アプリケーション

評価アプリケーション	計算手法	問題サイズ	ベクトル化率	ベクトル長
Earthquake	Friction Law	$2047 \times 2047 \times 257$	99.5 %	256
Land Mine	FDTD	$1500 \times 1500 \times 50$	99.7 %	250
姫野ベンチマーク	Jacobi	$512 \times 256 \times 256$	99.5 %	256
Antenna	FDTD	$364 \times 100 \times 100$	99.9 %	247

Antenna は，FDTD 法に基づいて対せき型フェルミアンテナの数値解析シミュレーションを行うプログラムである[15]．特徴として，主要カーネルであるフーリェ変換の解法が全実行時間の 99 %を占める．メモリアクセス時間は全実行時間の 87 %である．

5.3 性能評価

各アプリケーションによる評価結果を表 3 に示す．また，ルーフラインモデルに各評価結果をプロットしたものを，図 3〜図 6 に示す．Antenna は Operational Intensity が 0.58 と非常に高く，メモリバンド幅がボトルネックにならないため，ベクトルキャッシュを利用することなく高い実行効率が得られる．また，姫野ベンチマークは高い参照の局所性を有することから，ベクトルキャッシュを利用するだけで飛躍的に性能が向上する．一方，Earthquake と Land Mine では，そのままのコードではベクトルキャッシュを有効利用することができないため，ベクトルキャッシュの高いメモリバンド幅を活用できず，実行効率は低い．そのため，これらのアプリケーションにおいてはベクトルキャッシュを有効利用するためにプログラム最適化が必要であり，最適化により実行効率が向上する．

ルーフラインモデルでは，Operational Intensity が 0.5 より小さい場合にはメモリバンド幅がボトルネックとなり，0.5 より大きい場合にはメモリバンド幅がボトルネックとならないことが示されている．本評価結果より，ルーフラインモデルに表現される通り，Operational Intensity に応じてメモリバンド幅がボトルネックになることがわかる．このことから，ルーフラインモデルを用いることで，アプリケーションの特徴を考慮したボトルネックの解析が可能となる．さらに，ベクトルキャッシュを利用することでメモリバンド幅が増加し，メインメモリのバンド幅を越える実行効率が得られることも，ルーフラインモデルで表現されている．したがって，ルーフラインモデルは，ベクトルキャッシュを有するベクトルプロセッサの性能モデルとして利用可能である．ルーフラインモデルの利用により，アプリケーショ

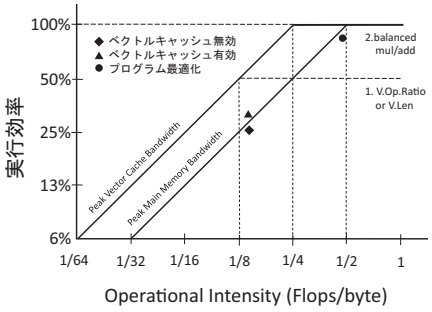

図 3 Earthquake の性能評価

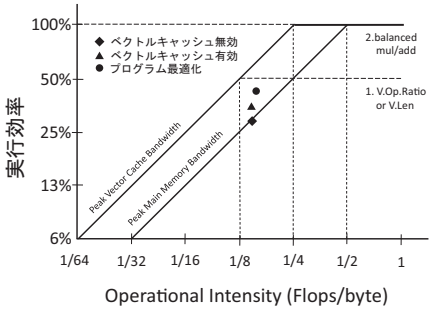

図 4 Land Mine の性能評価

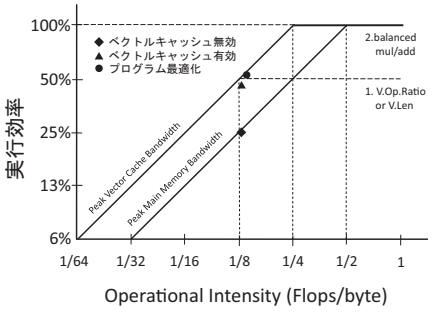

図 5 姫野ベンチマークの性能評価

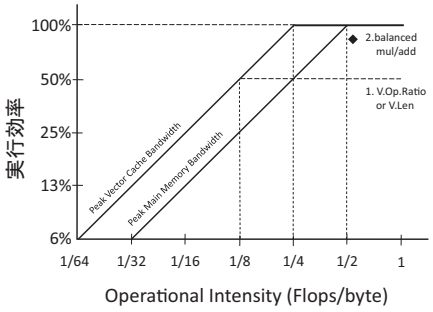

図 6 Antenna の性能評価

ンの特徴に応じて，Operational Intensity の増加を狙うプログラム最適化や，キャッシュヒット率の向上を促す最適化といった最適化方針を確立することが可能となる．3.1 節から 3.4 節において，各アプリケーションに適用する最適化の詳細と性能向上の理由を考察する．

5.3.1 Earthquake

Earthquake の Operational Intensity は 0.16 と低いため，ベクトルキャッシュを用いない場合にはメインメモリのバンド幅が制約となり，その実行効率はわずか 25 %である．Earthquake のカーネルループを図 7 に示す．プログラムの特徴として wk1_r(iq,km)，wk2_r(jq,km)，wk1_i(iq,km)，wk2_i(jq,km) の 4 種類の配列に時間的局所性が存在するため，ベクトルキャッシュによってそれらのデータを再利用することにより，実行効率は 28.6 %に向上する．しかし，Operational Intensity が低いために依然としてメモリバンド幅がボトルネックとなっている．そこで，Operational Intensity を高めるために Earthquake へループアンローリングによる最適化を施す．外側ループをアンローリングすることにより，wk1_r(iq,km)，wk1_i(iq,km) のロード/ストア回数が削減される．その結果，メモリアクセス数が減少し，Operational Intensity は大きくなるため，メモリバンド幅のボトルネックの解消が期待される．ループアンローリング段数ごとの実効効率と Operational Intensity を図 8 に示す．アンローリング段数が増加するに伴い，多くのメモリアクセスが削減されるため，Operational Intensity は増加する．一方で，実効性能は 8 段のループアンローリングで最大値を示し，以降は減少する．これは，ループアンローリングを行うことでベクトルレジスタ間でのデータ移動や，ループ分岐時のスカラ命令が増加するためである．これらの処理時間が，削減可能なメモリアクセス時間を越えるため，全体の実行時間が増加し実行効率が低下する．

まとめとして，Operational Intensity が低いアプリケーションではループアンローリングが有効であるが，ループアンローリング回数が多すぎると性能が低下するので，適切なアンロール回数を探りつつプログラムを最適化する必要がある．

表 3 各アプリケーションの実行効率

評価アプリケーション	ベクトルキャッシュ無効	ベクトルキャッシュ有効	プログラム最適化
Earthquake	25.0 %	28.6 %	85.8 %
Land Mine	30.0 %	36.4 %	39.2 %
姫野ベンチマーク	25.5 %	48.6 %	53.1 %
Antenna	83.6 %	83.6 %	-

```
1  do km = 1, nd
2   do jq = 1, nsum_dip
3    do iq = 1, nsum_dip
4     wk1_r(iq,km) = wk1_r(iq, km)
5                  + wk2_r(jq, km) * gd_dip(iq, jq, km)
6     wk1_i(iq,km) = wk1_i(iq, km)
7                  + wk2_i(jq, km) * gd_dip(iq, jq, km)
8    enddo
9   enddo
10 enddo
```

図 7 Earthquake のカーネルプログラム

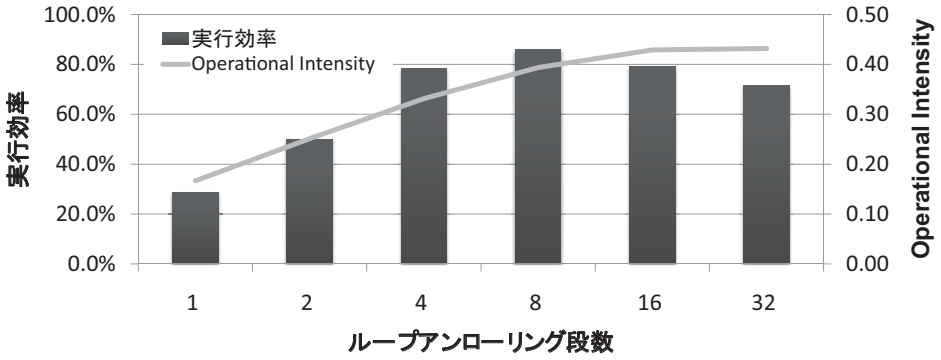

図 8 ループアンローリング段数ごとの実行効率と Operational Intensity

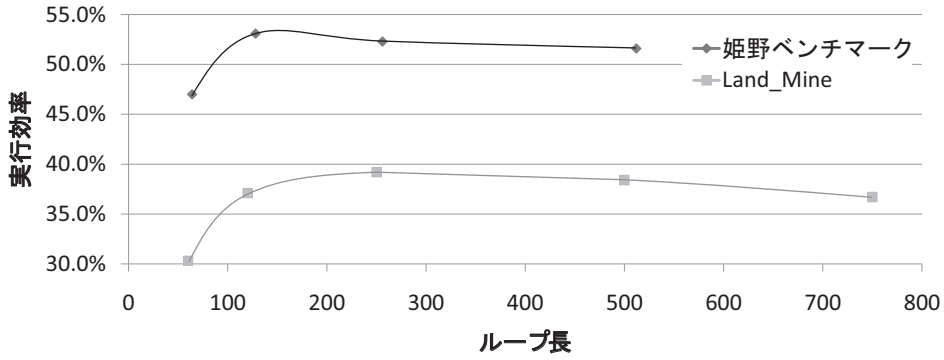

図 9 ループ長ごとの実行効率

5.3.2 Land Mine

Land Mine の Operational Intensity は 0.16 であり，キャッシュを用いない場合にはメインメモリのバンド幅が制約となり，図 4 に示す通り，実行効率が 30.0 %である．一方，キャッシュを用いた場合でも，キャッシュヒット率が 18.9 %しか得られず，実行効率は 36.4 %までの向上にとどまる．そこで，時間的局所性を高めるためにキャッシュブロッキングを検討する．図 9 より，ループ長を短縮するにつれ実行効率が向上していくことがわかる．これは，ループ長が長い場合では，1 つのループで扱うデータサイズが大きいため，ループ間の時間的局所性が小さく高いキャッシュヒット率は望めないが，ブロッキングを行いループ長を短縮することで時間的局所性が増加し，キャッシュヒット率が向上するためである．しかし，ループ長が 256 までしか性能は向上は得られない．これは，キャッシュブロッキングによって得られる性能向上よりも，ループ長低下によるベクトル演算の性能低下が大きいためであると考えられる．

これまでのベクトルプロセッサ向けの最適化では，ループ長を長くすることで性能を向上させてきた．しかし，ベクトルキャッシュを利用する場合には，逆にループ長を短くし，時間的局所性を高める方がより高い性能が得られることがわかった．ただし，ブロッキングによりループ長が 256 よりも短くする場合は性能が低下する恐れがあるため，適切なループ長を探りつつプログラムを最適化する必要がある．

5.3.3 姫野ベンチマーク

姫野ベンチマークの Operational Intensity は 0.13 と低いため，ベクトルキャッシュを用いない場合にはメインメモリのバンド幅が制約となり，図 5 に示す通り，実行効率は 25.5 %である．一方，キャッシュを用いることでキャッシュヒット率が 51.3 %となり，実行効率は 48.6 %まで向上する．この実行効率は Operational Intensity が 0.13 で得られる最も高い実行効率であるため，キャッシュメモリの利用を目的とした最適化ではこれ以上の性能向上は得られない．そこで，ループアンローリングによる最適化を施す．ループアンローリングを行うことにより，外側ループのデータをベクトルレジスタで再利用可能になるため，メモリアクセス数が削減され，Operational Intensity が向上する．ループアンローリングを行った結果，Operational Intensity は 0.14 に向上する．ルーフラインモデルより，Operational Intensity が 10 %増加することで，理論実効性能も同様に 10 %改善することが期待される．しかし，キャッシュヒット率は 46.2 %に低下し，実行効率は 6 %しか向上しない．これは，ループアンローリングによって最内ループのデータ参照数が増加し，ループ間での時間的局所性が低下するためである．そこで，ループアンローリングで低下した時間的局所性を増加させるために，キャッシュブロッキングを検討する．図 9 にキャッシュブロッキングを行った際のループ長ごとの実行効率を示す．ブロッキングによりループ長を短縮することで時間的局所性が増加し，徐々に実行効率が向上していき，ループ長 128 で最も高い性能が得られる．しかし，Land Mine と同様にループ長が短すぎる場合には性能が低下する．

ベクトルキャッシュの性能が十分に得られるプログラムでは，ループアンローリングによる Operational Intensity の増加で，より一層の性能向上が期待できる．しかし，ループアンローリングによりキャッシュヒット率が低下し，ベクトルキャッシュの性能が発揮できなく

なる場合がある．その場合には，キャッシュブロッキングを用いてキャッシュヒット率を増加させることで性能向上が実現できる．ただし，ループアンローリング回数に応じて時間的局所性が低下することから，十分なキャッシュヒット率を得るためには，アンローリング段数に応じてループ長を短縮する必要があると考えられ，その結果，ループ長短縮による性能低下が生じる恐れがある．したがって，ループアンローリングに加えキャッシュブロッキングを行う場合は，性能維持に必要な最低限のベクトル長を考慮して最適化を行わなければならない．

5.3.4 Antenna

Antenna の Operational Intensity は 0.58 であることから，他のアプリケーションと異なりメモリバンド幅がボトルネックにならなず，高い実行効率が期待できる．演算性能におけるボトルネックとして，ベクトル化率やベクトル長，実行される乗算回数と加算回数の演算割合が挙げられる．Antenna はベクトル化率 99.9 %，ベクトル長 247 と十分に高いため，これらはボトルネックにならない．しかし，乗算回数と加算回数を比較すると，加算回数は乗算回数の 7 割と少ないため，乗算器がボトルネックとなり，加算器は 7 割しか稼働できない．その結果，実行効率は 83.6 %となる．演算の割合を均等にするためには，演算種類の異なる複数のループを融合することにより可能であるが，Antenna は単一ループのみで構成されるため，ループ融合による最適化は不可能である．また，メモリバンド幅がボトルネックになっていないため，ベクトルキャッシュ利用による性能向上も得られない．

Operational Intensity が 0.5 よりも高く，演算性能が上限となるアプリケーションでは，ベクトル長やベクトル化率，演算の割合が実行効率を左右する．ベクトル長やベクトル化率はループ交換などの最適化によって向上可能であり，演算の割合は複数のループが存在する場合はループ融合により均等にすることが可能である．しかし，これらの最適化はプログラムのアルゴリズム依存であることから，より高い性能を得るためにはアルゴリズムの変更が求められる．

5.4 消費エネルギ評価

消費エネルギの観点から，プログラム最適化の有効性を評価する．ベクトルキャッシュを用いない場合とベクトルキャッシュを用いる場合の消費エネルギと，プログラム最適化を施した際の消費エネルギを比較することにより，プログラム最適化が消費エネルギに与える影響を評価する．評価方法は，メインメモリとキャッシュのアクセス数をシミュレーションにより計測し，得られた結果からそれぞれで消費される消費エネルギを算出する．メインメモリの消費エネルギは，スーパーコンピュータのメインメモリにおける消費エネルギが公開されていないことから，1 回のメインメモリアクセスにかかる消費エネルギを 50 μJ，100 μJ，200 μJ と仮定して実験を行った．その中から，本報告では，メモリアクセスにかかる消費エネルギを 100 μJ として評価した結果を用いる．また，キャッシュメモリの消費エネルギは CACTI5.1[16] を用いて算出する．

各アプリケーション実行時における消費エネルギを図 10 に示す．各消費エネルギはベクトルキャッシュを用いない場合の結果で正規化する．評価結果より，Earthquake と姫野ベンチマークではベクトルキャッシュを用いることで消費エネルギが 50 %削減でき，さらにプログラム最適化を施すことにより Earthquake ではさらに 18 %，姫野ベンチマークでは 4 %の消費エネルギが可能となる．Earthquake では，ループアンローリングによってメモリアクセス回数を削減することで，メモリアクセスに関わる消費エネルギが大幅に削減されている．姫野ベンチマークでは，キャッシュブロッキングによりキャッシュヒット率が増加することにより，消費エネルギの大きいメインメモリへのアクセスが，消費エネルギの小さいベクトルキャッシュへアクセスに置き換えられたため，アプリケーション全体の実行に要する消費エネルギも大幅に削減されている．Land Mine はキャッシュヒット率が低いため，ベクトルキャッシュを用いることによる消費エネルギの削減は 17 %程度であり，最適化によりさらに 3 %削減される．Antenna は，キャッシュを用いても性能向上は得られなかったが，消費エネルギは 33 %に削減される．このプログラムはメモリバンド幅がボトルネックにならないため，キャッシュを用いても実効メモリバンド幅が向上せず性能向上は得られない．しかし，キャッシュヒット率は十分に高く，キャッシュからデータ転送を行うことで消費エネルギが削減される．

以上の結果から，ベクトルキャッシュを用いることで消費エネルギを削減することが可能であり，ベクトルキャッシュを有効利用するように最適化を行うことで消費エネルギ削減の効果はさらに大きくなることが示された．

6. まとめと今後の課題

本報告では，ベクトルキャッシュを有するベクトルプロセッサにルーフラインモデルを適用した．ルーフラインモデルにおけるメモリバンド幅の性能上限をベクトルキャッシュのメモリバンド幅とすることで，ベクトルキャッシュによる性能向上が視覚化でき，キャッシュの有効性の検証が可能となった．

ベクトルキャッシュを利用して得られる性能向上を，実アプリケーションにより評価し，ルーフラインモデルを用いてプログラム最適化の効果を視覚化した結果，一部のアプリケー

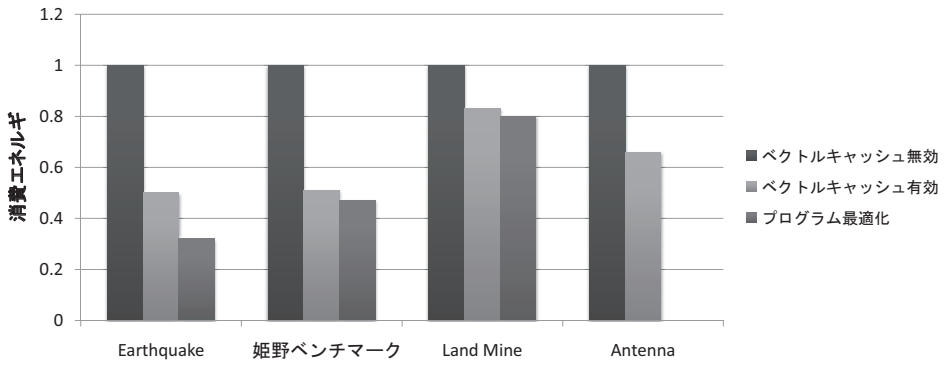

図 10 各アプリケーションにおけるプログラム最適化による消費エネルギの削減

ションでは十分にキャッシュの性能が引き出せていないことが分かった．そこで，これらのアプリケーションに最適化を施し，最適化の効果の解析を行った．その結果，Operational Intensity が 0.5 よりも低いアプリケーションでは，ループアンローリングを行うことで Operational Intensity が増加し性能向上が得られることがわかった．しかし，ループアンローリング回数が多い場合は，ベクトルレジスタ間のデータ転送やスカラ命令の増加，キャッシュヒット率の低下を招くため，適切なアンロール回数を探りつつ最適化を行う必要がある．また，キャッシュヒット率が低いアプリケーションでは，キャッシュブロッキングを行うことで性能向上が得られることがわかった．従来のベクトルプロセッサ向けプログラム最適化では，ベクトル長を伸ばすことで性能向上を目指してきた．一方，ベクトルキャッシュを有するベクトルプロセッサでは，逆にベクトル長を短くし，時間的局所性を高める方が高い実行効率を得ることが可能である．しかし，ループ長が短すぎる場合では，ベクトル長低下による急激な性能低下が起こりうるため，性能の維持が可能な最低ベクトル長を意識してブロッキングを行う必要がある．キャッシュの性能が十分に引き出せているプログラムにおいて，さらに性能向上を達成するためには，ループアンローリングによる Operational Intensity 増加に加え，アンローリングによるキャッシュヒット率低下を防ぐためのキャッシュブロッキングも行う必要がある．ただし，多すぎるループアンローリング回数によるキャッシュヒット率低下を防ぐには，ブロッキングによってループ長をより短縮させる必要があるため，その結果，ベクトル長が低下しすぎて性能を悪化させる恐れがある．したがって，必要最低限のループ長を確保したブロッキングを行い，そこで維持可能なキャッシュヒット率を考慮してループアンローリングを行う必要がある．

実アプリケーション実行時の消費エネルギの評価より，プログラム最適化を行わずベクトルキャッシュを利用するだけでも，消費エネルギの大きいメインメモリへのアクセスを，消費エネルギの小さいベクトルキャッシュへのアクセスに置き換えることができ，消費エネルギが削減できる．評価結果より最大で 50 %の消費エネルギが削減可能であことが示された．さらに，プログラム最適化を行うことにより，ループアンローリングによるメモリアクセス削減や，選択的キャッシング・キャッシュブロッキングによるキャッシュヒット率の向上によって，最大で 77 %の消費エネルギ削減ができる．したがって，プログラム最適化は実行効率の向上だけでなく，消費エネルギも大幅に削減できることが明らかとなった．

本報告では，基本的なプログラム最適化を行ったが，一部のアプリケーションでは実効メモリバンド幅がベクトルキャッシュのメモリバンド幅の上限に達していない．そこで，さらに高度な最適化についても定量的に検証し，ベクトルプロセッサ向けプログラム最適化の方針を明らかにすることが今後の課題である．

7. 謝　　辞

本研究では，日本電気株式会社の協力により，日本電気株式会社が開発したトレースドリブンシミュレータを使用した．関係各位に感謝する．実験に際し，東北大学サイバーサイエンスセンターのスーパーコンピュータシステムを利用した．

参 考 文 献

1) Hiroaki Kobayashi.: Implication of Memory Performance in Vector-Prallel and Scalar-Parallel HEC Systems. In: High Performance Computing on Vector Systems 2006, pp.21–50, Springer-Verlag (2006)
2) Hiroaki Kobayashi, Akihiro Musa, Yoshiei Sato, Hiroyuki Takizawa, and Koki Okabe.: The Potential of On-Chip Memory Systems for Future Vector Architecture. In: High Performance Computing on Vector Systems 2007, pp.247–264, Springer-Verlag (2007)
3) Akihiro Musa, Yoshiei Sato, Ryusuke Egawa, Hiroyuki Takizawa, Koki Okabe and Hiroaki Kobayashi.: An on-chip cache design for vector processors. In: MEDEA Workshop Memory Performance, Dealing with Applications, systems and architecture (2007)
4) Hongzhang Shan, and Erich Strohmaier.: Performance Characteristics of the Cray X1 and Their Implications for Application Performance Tuning. In: ICS'04 pp.175–183 (2004)
5) Leonid Oliker, Rupak Biswas, Julian Borill, Andrew Canning, Jonathan Crater, M.Jahed, Hongzhang Shan, and David Skinner.: A Performance Evaluation of the Cray X1 for Scien-

tific Applications. In: 6th International Meeting on High-Performance Computing for Computational Science, Valencia, June 20-30 (2004)
6) Thomas H.Dungian Jr, Jeffrey S.Vetter, James B.White III, and Patrick H. Worley.: Performance evaluation of the Cray X1 distributed shared-memory architecture. In: IEEE Micro, Vol.25, pp.30–40 (2005)
7) Hiroaki Kobayashi, Ryusuke Egawa, Hiroyuki Takizawa, Koki Okabe, Akihiro Musa, Takashi Soga, and Yoichi Shimomura.: First Experiences with NEC SX-9. In: High Performance Computing on Vector Systems 2008, pp.3–11, Springer-Verlag (2008)
8) David Kroft.: Lockup-Free Instruction Fetch/Prefetch Cache Organization. In: ISCA, pp. 81–88 (1981)
9) Nakazato Satoshi, Tagaya Satoru, Nakagomi Norihito, Watai Takayuki, and Sawamura Akihiro.: Hardware Technology of the SX-9 (1) - Main System. In: NEC Technical Journal, Vol. 3, No. 4, pp.15–18, (2008)
10) Dennis Abts, Abdulla Bataineh, Steve Scott, Greg Faanes, Jim Schwarzmeier, Eric Lundberg, Tim Johnson, Mike Bye, and Gerald Schwoerer.: The Cray BlackWidow: a highly scalable vector multiprocessor. In: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, pp.1–12 (2007)
11) Samuel Williams, Andrew Waterman, and David Patterson.: Roofline: an insightful visual performance model for multicore architectures. In: Communications of the ACM, Vol. 52, No. 4, pp.65–76 (2009)
12) Keisuke Ariyoshi, Toru Matsuzawa, and Akira Hasegawa.: The key frictional parameters controlling spatial variations in the speed of postseismic-slip propagation on a subduction plate boundary. In: Earth and Planetary Science Letters, Vol. 256, pp.136–146b (2007)
13) Himeno benchmark. http://w3cic.riken.go.jp/HPC/HimenoBMT
14) Takeo Kobayashi, Motoyuki Sato.: FDTD simulation on array antenna SAR-GPR for land mine detection. In: Proceedings of SSR2003, pp.279–283 (2003)
15) Yukiko Takagi, Hiroyasu Sato, Yoshihiko Wagatsuma, and Kunio Sawamura.: Study of High Gain and Broadband Antipodal Fermi Antenna with Corrugation. In: Proceedings of 2004 International Symposium on Antennnas and Propagation, Vol. 1, pp.69–72 (2004)
16) Shamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P.Jouppi.: Cacti 5.1. In: HP Laboratories, Palo Alto, HPL-2008-20 (2008)