計算機アーキテクチャ 149－8
（２００２．８．２２）

HPC向けオンチップメモリプロセッサアーキテクチャ SCIMAのSMP化の検討と性能評価

高橋睦史[†1] 近藤正章[†2,†3] 朴泰祐[†4]
高橋大介[†4] 中村宏[†2] 佐藤三久[†4]

SCIMAはプロセッサとオフチップメモリの性能格差に起因する問題の解決を目的とするHPC向けアーキテクチャであり，オンチップメモリを利用した明示的なデータ転送のプログラミングが可能となっている．一方，近年のHPC向けアーキテクチャではSMP構成が不可欠となっているが，それはオフチップメモリへのアクセストラフィックの増大をもたらし，深刻な性能低下を招く恐れを生じている．そこで本稿では，SCIMAのSMP化への対応を検討し，その構成について性能評価を行った．その結果，SCIMAによる大きな粒度でのデータ転送とバストラフィックの削減によりスケーラビリティが得られることが分かり，SMP構成においてもSCIMAが有効であることが分かった．

SMP configuration and performance evaluation of SCIMA - on-chip memory processor architecture for HPC

Chikafumi TAKAHASHI,[†1] Masaaki KONDO,[†2,†3]
Taisuke BOKU,[†4] Daisuke TAKAHASHI,[†4]
Hiroshi NAKAMURA[†2] and Mitsuhisa SATO[†4]

SCIMA is a processor architecture equipped with addressable on-chip memory to solve the memory-wall problem caused by processor/memory performance gap. In this paper, we propose to modify current SCIMA for SMP-enabled configuration and present preliminary performance evaluation on SMP-SCIMA system. As a result, it is shown that SCIMA's feature to reduce the off-chip traffic and optimize the data transfer granularity improves the performance on several benchmarks even with SMP configuration compared with traditional cache-only architecture.

1. はじめに

SCIMA（Software Controlled Integrated Memory Architecture）[1),2)] は従来のキャッシュの代わりに，データ転送をプログラミング可能な高速オンチップメモリを導入することにより，HPCアプリケーションにおいて最も深刻であるプロセッサ・オフチップメモリ間のデータ転送量を最小化し，かつデータ転送粒度を最適化することを目的としたプロセッサアーキテクチャである．これまでに，SCIMAに関しては単一プロセッサ構成を基本とした性能評価が行われ[3)]，キャッシュのみを持つプロセッサに対する優位性が示されてきた．

一方，近年のHPC向けアーキテクチャでは，相互結合網上のノード規模を抑えつつプロセッサ数を増やすため，SMP構成を取ることが不可欠となっている．SMP化された計算ノードでは，ネットワークインタフェースの共有化やシステムのコンパクト化が図られるが，その反面，オフチップメモリへのアクセストラフィックの増大により，深刻な性能低下を招く恐れがある．SCIMAでは，このようなアクセストラフィックを最小化することがその本質であるため，これをSMP化することにより，通常のキャッシュ型アーキテクチャ（以降，CACHE only）に比べ，ノード内プロセッサ数に対する高いスケーラビリティが確保できるのではないかということが予想される．

そこで，SCIMAの基本アーキテクチャをSMP化に対応させ，そのような構成における予備的な性能評価を行うことが本稿の目的である．

†1 筑波大学 大学院 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology, The University of Tokyo
†3 科学技術振興事業団 JST
†4 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

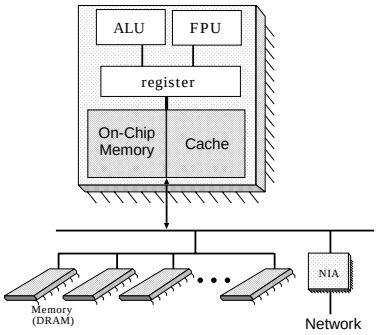

図 1 SCIMA の構成図

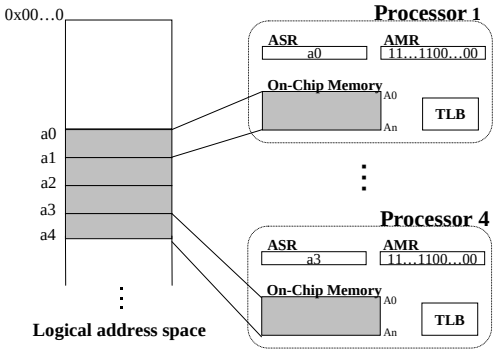

図 2 SMP 構成によるアドレス空間

2. SCIMA

図 1 に，SCIMA の構成を示す．SCIMA では，プロセッサチップ上にキャッシュに加えてオンチップメモリを搭載する．キャッシュはハードウェア制御により暗黙的にデータ転送が操作されるのに対し，オンチップメモリはソフトウェアにより明示的に転送を指示することが可能である．また，単純な連続アドレスアクセスだけでなく，ストライド転送等を指定できるようになっている．

SCIMA では，論理アドレス空間上にオンチップメモリをマッピングする．このアドレス空間はオフチップメモリのそれの一部となっており，これを管理するためにオンチップメモリの開始アドレスを保持する ASR（On-Chip Address Start Register）とオンチップメモリ容量を保持する AMR（On-Chip Address Mask Register）の 2 種類のレジスタが導入されている．オンチップメモリ以外のアドレスへのアクセスは通常のプロセッサ同様，キャッシュを通じてアクセスされる．

また SCIMA では，オンチップメモリ・オフチップメモリ間のデータ転送を指定するために，page-load/page-store 命令（以下，p-load/p-store）が拡張命令として加えられる．この命令は，page と呼ばれる比較的粒度が大きなオンチップメモリの管理単位に対するデータ転送を指示するものである．また，この命令はブロックストライド転送機能を備え，例えば多次元配列に対するアクセスにおいて，キャッシュで生じるような無駄なデータ転送を極力抑えることが可能となっている．さらに，必要に応じてキャッシュラインよりもはるかに大きなデータブロックを転送することにより，データ転送オーバヘッドを相対的に小さくすることも可能になっている．

対象アプリケーションによってはデータアクセスが複雑で明示的なデータ転送命令を記述するのが困難な場合も想定される．これに対処するため，SCIMA ではキャッシュとオンチップメモリをハードウェア的に統合し，way 単位でキャッシュ・オンチップメモリを切り替えることが可能である[1)]．

3. SMP 化の検討

3.1 アドレス空間

従来の SCIMA では，オンチップメモリとオフチップメモリを，同一のアドレス空間における排他的な領域として定義してきた．SMP 化においては，当然オフチップメモリの領域についてはアドレス空間がプロセッサ間で共有化される．しかし，オンチップメモリを共有することは，データの局所性を最大限に生かすという SCIMA の基本概念に反することになる．したがって，オンチップメモリ空間についてはプロセッサ毎に独立とすべきである．このため，以下の 2 つの方針が考えられる．

(1) 各プロセッサにおけるオンチップメモリにマップされた空間のアドレスは同一とし，その領域に対するアクセスはそのプロセッサのオンチップメモリに対するアクセスとなる．

(2) アドレス空間上にプロセッサ台数分のオンチップメモリ領域を確保し，各プロセッサにはその一部がそれぞれ割り当てられる．他のプロセッサのオンチップメモリ領域にマップされた領域にアドレスは割り当てられるが，その領域へのアクセスは不可能である（segmentation violation となる）．

（1）の方法はシンプルではあるが，SPMD プログラム上で混乱を生じやすい．我々はユーザが意識的にオンチップメモリへのアクセスを記述することを想定し，概念的にわかりやすい（2）の方法を選択することにした．各プロセッサには従来と同様に ASR 及び AMR が存在するが，その内容はプロセッサ毎に異なる（オンチップメモリサイズの異なるプロセッサが混在することも，理論的には可能である）．4 プロセッサによる SMP 構成の場合の構成とアドレス空間の様子を**図 2** に示す．

3.2 オンチップメモリ間データ転送

前節で述べたアドレス空間構成では，あるプロセッサから他のプロセッサの持つオンチップメモリに直接アクセスすることができない．オンチップメモリは高

速にアクセスすることのできるプロセッサ上のメモリであり，仮にプロセッサ間においてオフチップメモリを介さない直接のデータ転送を行うことが可能であるならば，さらなる性能が得られる可能性がある．しかし，これを実現するハードウェアを実装することは，メモリアクセス機構を非常に複雑にする．また，SCIMA自体がデータセットの非常に大きなHPCアプリケーションをターゲットとしているので，オンチップメモリ間で小さなデータセットを交換できることがどの程度有効であるかという点で疑問が残る．したがって現時点では，あるプロセッサは自プロセッサ内のオンチップメモリにのみアクセスできることとし，他プロセッサ内のオンチップメモリに対する直接のデータアクセスは，各種アプリケーションの要求などを考慮し今後検討していく予定である．

4. オンチップメモリプログラミング

本章では，SCIMAを適応したSMPマシン向けプログラムの例として，行列積演算とNAS Parallel Benchmarks[4]（以下NPB）のKernel CGを取り上げ，SCIMAを利用したプログラミングと，それによって得られる利点を示す．

4.1 行列積演算

まず，ここでは$C = A \times B$で表される，N次の倍精度浮動小数の行列積演算について取り上げる．行列積演算を単純に記述すると3重ループ構造となるが，キャッシュ・ヒット率を考えた場合に時間的局所性を生かすことができない．そこで最適化手法のひとつとしてキャッシュブロッキング[5]がしばしば用いられる．

並列化としては，行列Cについての1次元もしくは2次元のブロック分割が考えられる．ここで1次元ブロック分割を考えるとすると，行列Cを行方向に分割することで，1つのCPUあたりのデータセットは，分割されたCと同様に分割されたA，加えてB全体となる．この場合，分割後の各ブロックに関しては前述のキャッシュブロッキングが適用できる．これは，行列をいくつかの小行列に分割し，各フェーズのワーキングセットをキャッシュに収めるようにする方法である．

しかし，行列積演算をキャッシュブロッキングを用いた場合，同じキャッシュラインに連続してアクセスしてしまうことでラインコンフリクトが発生する可能性がある．

SCIMA向けアルゴリズムでは，CACHE onlyで行ったキャッシュブロッキングされたA, B, Cすべての行列の領域について，p-load/ p-storeを用いてオンチップメモリとのデータ転送を記述してやればよい．この場合，オンチップメモリにはラインという概念が存在しないのでラインコンフリクトは発生しない．さらに，SCIMAではストライドアクセスに対応した明示的なデータ転送ができるので，意図しないデータ転送は発生せず，転送量を必要最小限に抑えられる．

4.2 NPB Kernel CG

NPB Kernel CGは，正値対称な大規模疎行列の固有値をCG（Conjugate Gradient）法によって求めるベンチマークである．この固有値を2重ループで収束させながら計算するが，その計算に必要な実行サイクル数のうち大部分は疎行列とベクトルの積を求めることに費やされる．この行列・ベクトル積は$q = \Sigma i(A[i] \times p[colidx[i]])$という形で，ベクトル$q$の一要素を求めるのに，行列$A$への連続アクセスと，ベクトル$p$の要素への間接アクセスを必要とする．Class Wのベンチマークにおいては，Aのサイズは7000×7000で非零率1%の疎行列，pは$colidx$によってランダムに間接参照される7000要素のベクトルである（精度は倍精度浮動小数点）．

並列化に関しては，行列Aに着目して，1次元，もしくは2次元のブロック分割が行える．本評価では比較的少数のプロセッサのSMP構成を考えるため，1次元行ブロック分割のみを考える．

キャッシュのみを用いてこの演算を行うと，順次アクセスではあるが再利用性のないAと，$colidx$を用いた間接参照によるランダムアクセスのため局所性が活かせないpのキャッシュミスが多発する（このオリジナル版を以下C-orgと呼ぶ）．この時，ベクトルpをブロック分割し，その分割したブロックにアクセスするようAと$colidx$をあらかじめpのブロックにあわせて並び替え，pの参照に局所性を発生させることによって，pについてはキャッシュブロッキングが可能となり，キャッシュミスを軽減できる（この手法を以下C-optと呼ぶ）．しかし，Aに関してはこのようなキャッシュブロッキングは行えない．

ここでNPB CGにSCIMAの適用を考えると，先ほどキャッシュブロッキングを行ったpをオンチップメモリに転送することができる．加えて行列Aや，$colidx$もオンチップメモリに転送することができる（この手法を以下OCM-apcと呼ぶ）．CACHE onlyではブロッキングを行ったpと連続アクセスとなるAを比較的小さな粒度であるラインサイズ単位でデータを転送しているのに対し，SCIMAではページという大きな粒度の単位で転送できるので，SCIMAの適応によりメモリアクセスのレイテンシによるオーバーヘッドの影響が，キャッシュと比較して相対的に小さくなる．また，明示的にオンチップメモリ・オフチップメモリ間のデータ転送を指定できるので，不要なデータを転送することなく効率的にバスを使用することが可能である．加えて，キャッシュのようにラインコンフリクトが起きる心配もない．

5. 性能評価環境

SMP化されたSCIMAの性能を評価するため，計算機シミュレーションによる評価を行う．本稿では，

SMP 化された SCIMA に対する並列プログラミング環境として，スレッドプログラミングの代表的なライブラリである Pthreads によるプログラミングを考える．将来的には OpenMP や自動並列化コンパイラ等によるコード生成も検討する予定であるが，先述のアドレス空間においてオンチップメモリに対するデータ転送を記述する都合上，スレッドによる明示的な並列プログラミングが現時点では有効である．

評価用のターゲットプログラムには，前章で示した行列積演算と NPB Kernel CG を用いる．行列積演算は HPC 分野に置いて多用される処理であり，また，NPB は非常に多くの並列計算機で評価がなされていることから，この両者は性能評価の指標として有効であると考えられる．

シミュレーションは，MIPS IV の ISA（命令セット）を拡張した ISA を用いる既存の SCIMA シミュレータ[1] を元に，バス結合型 SMP マシンのシミュレーションが行えるような変更を加えたものである．キャッシュは L1 キャッシュをシミュレートし，キャッシュコヒーレンスには MSI プロトコル[6] を採用する．並列プログラムに関しては，基本的な Pthreads ライブラリに対応する．また，命令キャッシュは常にヒット，分岐予測は常に成功という条件でシミュレーションを行う．

p-load/p-store のような SCIMA 専用の命令は関数の形で擬似的にプログラムに挿入し，評価には既存の MIPS 用コンパイラを用いる．拡張命令と Pthreads に関する命令は，プリプロセッサを用いてアセンブラコードレベルで ISA 中で使用していない擬似命令に変換，バイナリにコンパイルする．シミュレータはこのバイナリコードを読み込み，擬似命令をフックすることでシミュレーションを行う．

シミュレータに与える共通のパラメータについて，特に指定が無い場合は以下の値を使用する．

- レジスタ数：INT=32, FP=32
- 演算ユニット数：INT=2， FP(add, sub, mult, multiply-add 用)=1
- reservation station エントリ数：INT, FP, LS 用各 32
- load/store 命令レーテンシ：2cycle
- バスバンド幅：4B/cycle
- オンチップメモリページサイズ：4KB
- キャッシュラインサイズ：32B

また，キャッシュ・オンチップメモリ容量に関しては，SCIMA と CACHE only の比較をするため，**表 1** の 2 つの構成のパラメータを与え，チップ上のメモリ

表 1 シミュレーションパラメータ

	Cache size	OCM size
CACHE only	32 KB (4 way)	0 KB
SCIMA	8 KB (1 way)	24 KB

OCM: On-Chip Memory

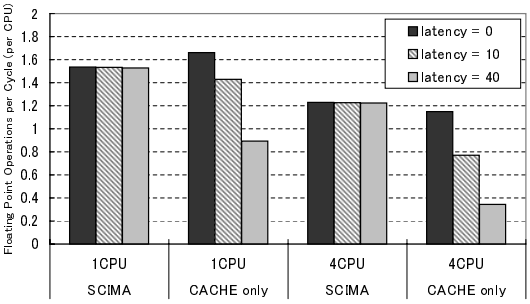

図 3 行列積におけるオフチップメモリアクセスレイテンシによる影響

を 8KB × 4way とし，SCIMA ではそれぞれの way をキャッシュもしくはオンチップメモリに切り替えて演算を行うことを想定する．キャッシュの連想度は，表 1 で示した，キャッシュとして利用する way 数と同一とする．また，オフチップメモリへのアクセスレイテンシは重要なパラメータのため，これはシミュレーション条件として別途与える．ある CPU でこのアクセスレイテンシによるストールが発生した場合には，他の CPU もオフチップメモリにアクセスする事はできない．

6. 性能評価結果

6.1 行列積演算

図 3 に，行列サイズ 250 のとき，オフチップメモリへのアクセスレイテンシを 0，10，40 cycle と変化させた評価について結果を示す．まず，CACHE only に注目するとオフチップメモリへのアクセスレイテンシが大きくなるにつれて大きく性能が下がっていることが分かる．さらに 1CPU 時よりも 4CPU 時のほうがその影響が大きい．これはレイテンシの増大によりデータ転送 1 回あたりのオーバーヘッドが増大するが，SCIMA では大きな粒度でのデータ転送により転送回数自体が少ないので，影響を受けにくいことが理由と考えられる．逆に CACHE only では大きく影響を受け，CPU 数が増大した場合にはオーバーヘッドでバスが混雑し，演算がストールしてしまう．

表 2 に，行列サイズ別の総バストラフィック量を示す．SCIMA ではストライド転送を行えることや意図しないデータ転送がほとんど発生しないので，CACHE

表 2 行列積における総バストラフィック量

Matrix size	CACHE only	SCIMA
50	0.13 Mbyte	0.16 Mbyte
100	0.97 Mbyte	0.80 Mbyte
150	3.98 Mbyte	2.70 Mbyte
200	8.24 Mbyte	5.44 Mbyte
250	14.6 Mbyte	9.00 Mbyte
300	28.2 Mbyte	17.2 Mbyte

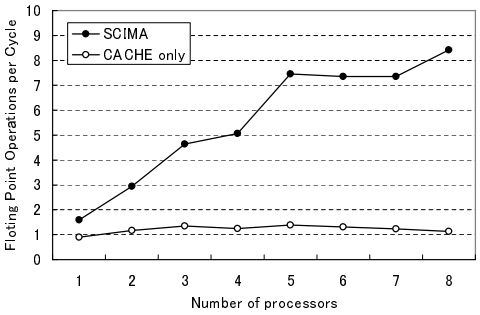

図 4 行列積におけるスケーラビリティ

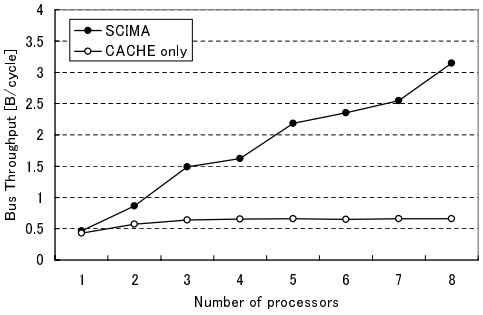

図 5 行列積におけるバススループット

only と比較してデータ転送量が抑えられている．

図 4 に，アクセスレイテンシ 40 cycle，行列サイズ 300 のとき，SMP 上のプロセッサ数増加に対する 1cycle あたりの浮動小数点演算量を示している．これを見ると，SCIMA では並列度の増加にしたがって性能向上が見られるが，CACHE only では並列化による性能向上がほとんど得られていない．このときの 1 cycle あたりのバススループットを**図 5** に示す．SCIMA では CPU 数の増加に従ってトラフィック量が増えているにもかかわらず，CACHE only では約 0.66 B/cycle に達するとそれ以上トラフィック量は増加しないことが分かる．これは，CACHE only ではオフチップメモリへのアクセスレイテンシの影響でバス帯域が飽和してるものと考えられる．キャッシュラインサイズは 32 B でスループットが 4 B/cycle，オフチップメモリへのアクセスレイテンシが 40 cycle であるから，1 ラインの転送に必要な時間は $(32/4) + 40 = 48$ cycle であり，1 cycle あたりのデータ転送量が $32/48 \approx 0.667$ B/cycle と理論的に求まることからこの推測が裏付けられる．以上の結果より，CACHE only ではスケーラビリティが得がたく，逆に SCIMA では大きなスケーラビリティが得られることが分かる．

6.2 NPB Kernel CG

4CPU 実行時でオフチップアクセスレイテンシが 40 cycle のとき，3 種類のプログラムについてキャッシュラインサイズを変化させてシミュレーションを行った結果について示す．**図 6** には，全実行時間を、オフチップメモリのスループット不足に起因するストール

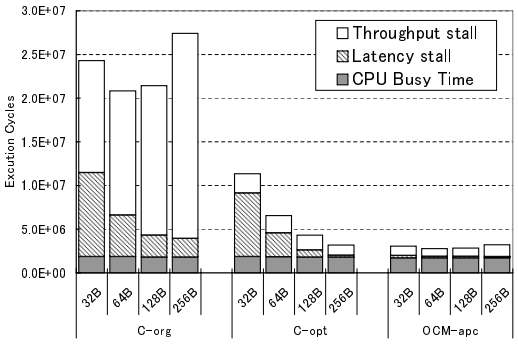

図 6 CG におけるキャッシュラインサイズ別実行サイクル数

(Throughput stall)、オフチップメモリへのアクセスレイテンシによるストール (Latency stall)、及び実際に CPU が処理を行っている時間 (CPU Busy Time) に分類して示す。図 6 から分かるように，すべてのラインサイズにおいて SCIMA は CACHE only で最適化を行ったプログラムよりも良好な性能を示している．ここでラインサイズが 32 B の時に注目すると，C-org と C-opt を比較した場合，約 15%の性能の向上が見られる．これはキャッシュブロッキングの効果によりトラフィック量が減少したためと考えられる．次に，C-opt と OCM-apc を比較すると，実行サイクル数が約 73%減少している．これはメモリブロッキングされた p をオンチップメモリに転送したことで配列間干渉によるトラフィックを減少させると共に，連続アクセスとなるが再利用性の無い行列 A とベクトル $colidx$ をキャッシュラインよりも大きな粒度でデータをロードし，latency stall を軽減できたためだと考えられる．しかし，キャッシュラインが大きくなるにつれて後者の理由による優位性は少なくなる．

また，C-org や C-opt では，ラインサイズの増減によってデータ転送の回数と意図しないデータの転送回数が変化することにより latency stall と throughput stall が大きく変動している．しかし SCIMA では主要なデータをオンチップメモリに転送することで，キャッシュラインサイズに左右されず，ほぼ一定に良好な性能を保っている．

図 7 を見ると，C-opt と OCM-apc はデータトラフィック量がほとんど変わらないことから，キャッシュブロッキングによる効果が大きいことが分かる．しかし，OCM-apc ではラインサイズの増加に伴い，トラフィック量が若干増加している．これは，オンチップメモリに載せなかった，ループを制御するための配列と解を格納するベクトル q 等について，8KB・連想度 1 のキャッシュにロードされる際にキャッシュコンフリクトを発生したものと考えられる．

7. おわりに

これまでの評価結果において，SCIMA は大きな粒

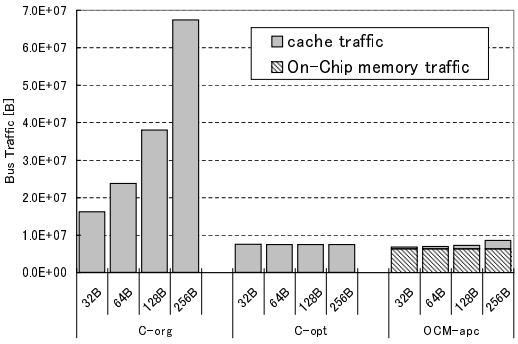

図 7 CG における総バストラフィック量

度でのデータ転送を行うことから，オフチップメモリへのアクセスレイテンシによる影響を受けにくいことが分かった．また，ユーザの意図しないデータ転送を引き起こさないことによるバストラフィック量の削減が，SMP 構成においても有効であることを示した．加えて，キャッシュを用いた SMP 構成の計算機では，シングルプロセッサ構成の計算機と比較してアクセスレイテンシに敏感に反応して性能を低下するが，SCIMA を適用した場合，この影響を大きく軽減できることが分かった．

図 6 の C-opt の結果からも分かるように，CACHE only ではラインサイズの増減により大きく性能が変化する．また，ラインサイズによるアプリケーションの最適化を考えた場合，移植性の低下や，多階層キャッシュへの最適化は困難であるという問題も発生する．この点，SCIMA ではラインサイズという概念は存在せず，大きな粒度によるバスの使用効率の良いデータ転送が行えるという点で優位性がある．その結果，SCIMA は CACHE only と比較して大きなスケーラビリティが得られていると考えられる．

キャッシュを用いたアーキテクチャでも，キャッシュプリフェッチ[7] やメモリインタリーブといった方法を用いることにより，これまで示してきたような事例において一定の性能改善は見込めると予想される．しかし，バス帯域が圧迫されている SMP マシンの現状ではデータプリフェッチによるレイテンシ隠蔽は見込めないと考えられる．また，メモリインタリーブによりメモリバンド幅を向上すれば，データプリフェッチによりさらに効果的にレイテンシを隠蔽できるが，その場合でも意図しないデータ転送が発生する可能性が残るので，明示的にオフチップメモリ・オンチップメモリ間のデータ転送が行える SCIMA は有効性は変わらないと推測される．

以上の結果から，SMP 構成の計算機において SCIMA を適用することは非常に有効であると考えられる．

また，SMP 化された SCIMA では，不必要なデータの load/store を避ける事ができるため，false sharing[8] の問題の解決が見込まれる．したがって，今後はこの問題についても評価を行い，また，より正確な評価を行うために，他のベンチマークや実アプリケーションによるプログラムでの評価も行う必要があると考えている．

謝辞 本研究に関して，御助言，御討論いただきました，筑波大学 計算物理学研究センターの関係者及び，東京大学 南谷・中村研究室の各位に感謝いたします．なお，本研究の一部は日本学術振興会未来開拓学術研究推進事業「計算科学」（Project No. JSPS-RFTF 97P01102）によるものである．

参 考 文 献

1) 中村宏, 他: ハイパフォーマンスコンピューティング向けアーキテクチャ *SCIMA*, 情報処理学会論文誌, Vol. 41, No. SIG 5(HPS 1), pp. 15–27 (2000).
2) Kondo, M. and et. al.: SCIMA: Software Controlled Integrated Memory Architecture for High Performance Computing, *ICCD-2000* (2000).
3) 近藤正章, 他: SCIMA における性能最適化手法の検討, 情報処理学会論文誌, Vol. 42, No. SIG 12 (HPS 4), pp. 37–48 (2001).
4) Bailey, D. and et. al.: The NAS Parallel Benchmarks 2.0, *NASA Ames Research Center Report*, NAS-05-020 (1995).
5) Lam, M. and et. al.: The cache performance and optimizations of Blocked Algorithms, *Proc. ASPLOS-IV*, pp. 63–74 (1991).
6) Culler, D.E. and et. al.: *Parallel Computer Architecture*, Morgan Kaufmann Publishers Inc., pp. 293–299 (1999).
7) Chen, T. and et. al.: A Performance Study of Software and Hardware Data Prefetching Schem, *Proc. of ISCA-21*, pp. 223–232 (1994).
8) Jeremiassen, T. E. and et. al.: Reducing false sharing on shared memory multiprocessors through compile time data transformations, *SIGPLAN Notices*, Vol. 30, Issue 8, pp. 179–188 (1990).