SR11000 におけるソフトウェアプリフェッチ手法の評価

青木 秀貴[†]　処 雅尋[††]　本川 敬子[†††]　五百木 伸洋[††††]　齋藤 拡二[††]

SR11000 モデル H1 が採用する POWER4+はハードウェアによるデータプリフェッチをサポートするが，多数のロードストリームを含むループでは，ハードウェアですべてのストリームをプリフェッチすることができず，性能が低下する。本稿では，この問題を解消するソフトウェアプリフェッチ手法について紹介する。評価の結果，本手法の適用により，ストリーム数が増えた場合にも安定して高い性能を実現できることを確認し，ストリーム数を考慮したループ分割が不要なことを明らかにした。SR11000 モデル H1 向けの日立最適化 FORTRAN90 コンパイラは，本手法によるコード生成が可能である。

Evaluation of Software Prefetch Method for SR11000

Hidetaka Aoki[†]　Masahiro Tokoro[††]　Keiko Motokawa[†††]　Nobuhiro Ioki[††††]　Koji Saito[††]

The POWER4+ processor, which SR11000 model H1 adopts, employs hardware to prefetch data transparently to software. In a loop with many load streams, however, performance degrades since it is not possible to prefetch all the streams with the hardware. This paper describes the software prefetch method to prevent this performance degradation. The evaluation results show that stably high performance is achieved by applying the method even in a loop with many load streams and that loop splitting according to the number of streams is not required. The Hitachi Optimizing FORTRAN90 compiler supports automatic application of the method.

1　はじめに

2003 年 5 月に製品発表した SR11000 モデル H1 は，ベクトル・スカラ融合型サーバ SR8000 シリーズの後継機として開発を進めてきたスーパーテクニカルサーバ SR11000 シリーズの初代モデルである[11)~13)]。SR11000 モデル H1 は，1.7GHz 動作の POWER4+を 16CPU 搭載する主記憶共有ノードを基本構成単位とし，多段クロスバネットワークにより最大 256 ノード結合する構成を取る。最大理論演算性能 27.8TFLOPS を有し，大規模な科学技術計算にも対応可能である。

SR11000 モデル H1 が主な対象とする科学技術計算では，キャッシュメモリに入り切らない大規模データを扱うことが多い。このため，高性能を実現する上で，高い実効メモリバンド幅が重要となる。メモリレイテンシを隠蔽し，高い実効メモリバンド幅を実現する手法として，データプリフェッチが有効である。データプリフェッチとは，利用が見込まれるデータをあらかじめキャッシュメモリに登録しておく手法であり，ハードウェアプリフェッチとソフトウェアプリフェッチとがある[1)]。

ハードウェアプリフェッチとは，規則的なデータアクセスパターンを実行時に検出し，ハードウェアで自動的にプリフェッチをおこなう方式である。ハードウェアプリフェッチは，Intel の NetBurst アーキテクチャ[2)]（Pentium 4，Xeon）および Pentium M[2)]，AMD の QuantiSpeed アーキテクチャ[3)]（Athlon MP，Athlon XP），富士通 SPARC64 V[4)]などに実装されている。これに対しソフトウェアプリフェッチは，コンパイル時にデータアクセスパターンを解析し，ロードモジュール中にプリフェッチ命令を挿入することで実現する。SR11000 の先代にあたる SR8000 では，ソフトウェアプリフェッチによる高速化を実現している[5)]。

SR11000 モデル H1 が採用する POWER4+は，ハードウェアプリフェッチ機構を搭載し，ロード命令による読み出しが予想されるデータをプリフェッチすることで，キャッシュミスによる実効メモリハンド幅の低下を低減する。しかし，一般にハードウェアプリフェッチでは，ハードウェア資源の制約により，サポートできるストリーム数に限りがある☆。POWER4+では，各 CPU コアが最大 8 個のロードストリームを検出し，ハードウェアプリフェッチをおこなうことができる[6)]。しかし，8 個を超えるロードストリームを含むループでは，ハードウェアプリフェッチ機構がすべてのストリームをプリフェッチすることはできないことから，実効メモリバンド幅が低下する[7)]。

SR11000 モデル H1 では，ハードウェアプリフェッチにお

† （株）日立製作所 中央研究所
Central Research Laboratory, Hitachi, Ltd.
†† （株）日立製作所 エンタープライズサーバ事業部
Enterprise Server Division, Hitachi, Ltd.
††† （株）日立製作所 システム開発研究所
Systems Development Laboratory, Hitachi, Ltd.
†††† （株）日立製作所 ソフトウェア事業部
Software Division, Hitachi, Ltd.

☆ 各 CPU でハードウェアプリフェッチ可能なストリーム数は以下の通り。
Intel NetBurst:　8 ストリーム
Intel Pentium M:　forward 12 ストリーム，backward 4 ストリーム
富士通 SPARC64 V:　16 ストリーム

けるストリーム数の上限を解消するため，ソフトウェアプリフェッチ手法を導入している[14]。本稿では，SR11000 モデル H1 におけるソフトウェアプリフェッチ手法について紹介するとともに，その有効性を明らかにする。以下，2章では，POWER4+のハードウェアプリフェッチ機構と，SR11000 モデル H1 で採用するソフトウェアプリフェッチ手法について述べる。続く3章でソフトウェアプリフェッチ手法の性能を評価し，4章でその有効性について考察する。最後に5章で本稿をまとめる。

2 SR11000 のデータプリフェッチ

2.1 SR11000 モデル H1 のノード構成

SR11000 モデル H1 のノードは，1.7GHz で動作する POWER4+を 16CPU 搭載する。2CPU コアを持つ POWER4+チップ 4 個を高密度モジュール上に実装し，2 個の高密度モジュールにより 16CPU の主記憶共有ノードを構成する。SR11000 モデル H1 のノード構成を図 1 に示す。

SR11000 モデル H1 は，3 階層のキャッシュメモリを持つ。各 CPU コアに搭載された L1 キャッシュ（32KB/1CPU），POWER4+チップに集積され 2 個の CPU コアで共有される L2 キャッシュ（1.5MB/2CPU），および，全 16CPU で共有される L3 キャッシュ（256MB/16CPU）である。なお，キャッシュラインのサイズは 128B である。

2.2 POWER4+のハードウェアプリフェッチ機構

POWER4+のハードウェアプリフェッチ機構は，連続するキャッシュラインへの読み出しアクセス（昇順または降順）からロードストリームを検出し，3 階層のキャッシュメモリに対して階層的なプリフェッチをおこなう[6]。図 2 に，定常状態におけるハードウェアプリフェッチの動作の概要を示す。ロード命令が L1 キャッシュ中のキャッシュライン l_0 に初めてアクセスするのを契機として，次のキャッシュライン l_1 を L2 キャッシュから L1 キャッシュにプリフェッチする。それと同時に，5 番目のキャッシュライン l_5 を L3 キャッシュから L2 キャッシュにプリフェッチする。メモリから L3 キャッシュへのプリフェッチは 512B 単位でおこなわれ，ロード命令による 4 キャッシュラインへの参照に 1 回の割合で，キャッシュライン l_{17} から l_{20} の 4 キャッシュラインをプリフェッチする。POWER4+では，各 CPU コアが最大 8 個のロードストリームを検出し，上記のハードウェアプリフェッチをおこなうことができる。

計算機システム性能のベンチマークセットである LMbench[8] Ver. 2.0.4 のうち，メモリレイテンシを測定する lat_mem_rd コマンドを用いて，ハードウェアプリフェッチの

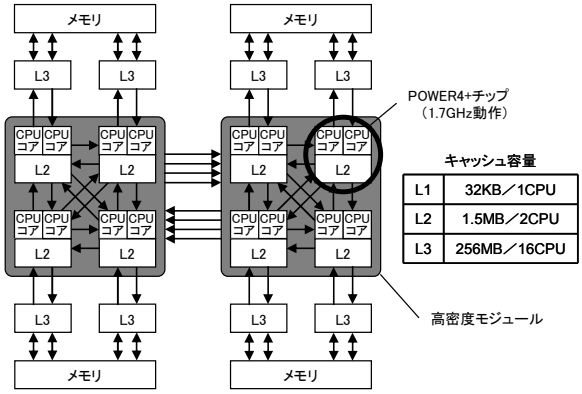

図1 SR11000 モデル H1 のノード構成

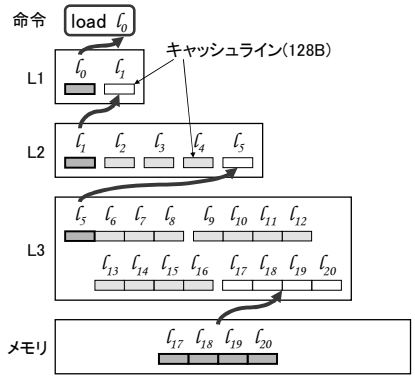

図2 POWER4+のハードウェアプリフェッチ動作

効果を評価した。lat_mem_rd コマンドは，読み出したデータをポインタとして次のデータを読み出す，ポインタアクセスを繰り返す。あるデータの読み出しと次のデータの読み出しにフロー依存があるため，ハードウェアプリフェッチが効かない場合には，各データ読み出しにメモリレイテンシ分のオーバヘッドがかかる。しかし，ハードウェアプリフェッチが効く場合には次のデータがプリフェッチされているため，メモリレイテンシの一部が隠蔽される。

SR11000 モデル H1 を用いて評価した結果，ハードウェアプリフェッチが効かない場合と比較して，ハードウェアプリフェッチが効く場合には，メモリレイテンシが見かけの上で 5 分の 1 以下に短縮されることがわかった。lat_mem_rd コマンドでおこなうポインタアクセスの繰り返し処理に対しては，コンパイル時の静的なデータアクセスパターン解析に基づくソフトウェアプリフェッチの適用も不可能であり，ハードウェアプリフェッチ機構による実行時のストリーム検出が有効である。

2.3 SR11000 のソフトウェアプリフェッチ手法

8 個を超えるロードストリームを含むループでは，POWER4+のハードウェアプリフェッチ機構がすべてのストリームをプリフェッチすることができないため，性能が低下

```
do i=1,m
   S=S+A1(i)+A2(i)+...+An(i)
end do
```

a) 適用前のコード

```
do i=1,m-U+1,U
   dcbt for A1(i+AHEAD)
   dcbt for A2(i+AHEAD)
   ...
   dcbt for An(i+AHEAD)
   S=S+A1(i)+A2(i)+...+An(i)
   S=S+A1(i+1)+A2(i+1)+...+An(i+1)
   ...
   S=S+A1(i+U-1)+A2(i+U-1)+...+An(i+U-1)
end do
do i=i,m
   S=S+A1(i)+A2(i)+...+An(i)
end do
```

b) 適用後のコードイメージ

図3 ソフトウェアプリフェッチ手法の適用例
（ロードストリーム数 n の場合）

する。この問題を解消するため，SR11000 モデル H1 ではソフトウェアプリフェッチ手法を導入している。

PowerPC アーキテクチャには，dcbt（Data Cache Block Touch）命令というプリフェッチ命令が用意されている[9]。本ソフトウェアプリフェッチ手法では，ループ中の全ロードストリームに対して dcbt 命令を発行し，全ストリームに対するプリフェッチをソフトウェアでおこなうことで，ハードウェアプリフェッチにおけるストリーム数の上限を解消する。dcbt 命令は，各ストリームに対し，少なくとも 1 キャッシュラインに 1 回の割合で発行する。

本手法の適用例を図 3 に示す。ここで AHEAD（≧0）とは，ロード命令に先行して dcbt 命令を発行するためのパラメータであり，メモリレイテンシを考慮して決定する。また，ループを U 倍（倍精度データでは U≦16）に展開することで，同一キャッシュラインに発行される dcbt 命令を 1/U に削減する。

SR11000 向けの日立最適化 FORTRAN90 コンパイラでは，本ソフトウェアプリフェッチ手法の自動適用をサポートしている。

3 ソフトウェアプリフェッチ手法の性能評価

以下の評価は，SR11000 モデル H1 の 1 ノードを用い，ページサイズを 16MB とするラージページモードでおこなった。ロードモジュール作成には日立最適化 FORTRAN90 を用い，コンパイルオプションによりソフトウェアプリフェッチ手法の適用あり／なしを制御している。

```
do i=1,m
   S1=S1+A1(i)
   S2=S2+A2(i)
   ...
   Sn=Sn+An(i)
end do
S=S1+S2+...+Sn
```

図4 n 配列の総和コード（倍精度データ）

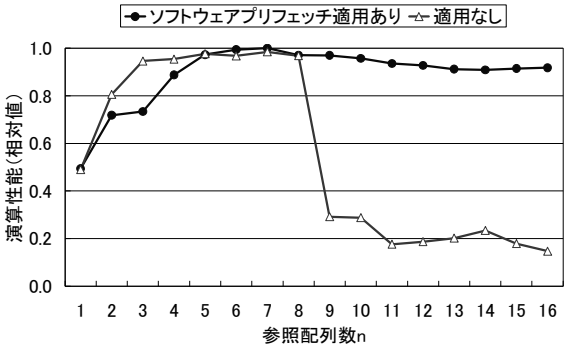

図5 n 配列総和の演算性能（1CPU 実行）

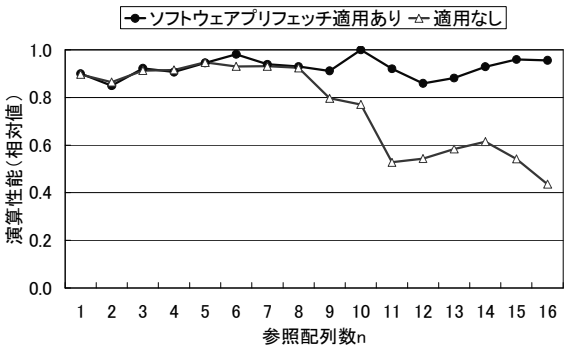

図6 n 配列総和の演算性能（16CPU 実行）

3.1 n 配列の総和

倍精度データの n 配列総和コード（図 4）を用いて，ソフトウェアプリフェッチ手法の効果を評価した。ループ長は十分に大きく，すべてのデータ読み出しがキャッシュミスする。配列数を 1～16 まで変化させ，SR11000 モデル H1 上で 1CPU 逐次実行／16CPU 並列実行した結果を，図 5 および図 6 に示す。それぞれの最高性能を 1 とする相対値で示した。

ソフトウェアプリフェッチ手法を適用していない場合／適用した場合とも 1CPU 実行でロードストリーム数（＝参照配列数）が少ない場合に性能が低いのは，on-the-fly のメモリ読み出し要求数が少なく，メモリレイテンシを十分に隠蔽できないためである。これに対し 16CPU 実行では，on-the-fly のメモリ読み出し要求が 16CPU 合計で十分な数となり，メモリバンド幅がボトルネックとなる。

ソフトウェアプリフェッチ手法を適用しない場合には，ハ

ードウェアプリフェッチ可能な 8 ロードストリームを超えると，1CPU 実行／16CPU 実行とも性能が低下している。

これに対しソフトウェアプリフェッチ手法を適用した場合には，1CPU 実行／16CPU 実行とも，ロードストリーム数が増えた場合にも安定して高い性能を実現している。参照配列数の増加に従って，アーキレクチャレジスタの不足により命令スケジューリングの自由度が失われ，性能を悪化させる危険があるが，本結果より，影響は小さいことがわかる。

3.2 n 配列の加算

倍精度データの n 配列加算コード（図 7）を用いて，ソフトウェアプリフェッチ手法の効果を評価した。ループ長は十分に大きく，すべてのデータ読み出しがキャッシュミスする。本コードでは，ループ中にデータの書き込みが含まれる。配列数を 1～16 まで変化させ，SR11000 モデル H1 上で 1CPU 逐次実行／16CPU 並列実行した結果を，図 8 および図 9 に示す。それぞれの最高性能を 1 とする相対値で示した。

1CPU 実行／16CPU 実行とも，ロードストリーム数が 8 を超えた場合にはソフトウェアプリフェッチ手法を適用した方が高性能となっており，ソフトウェアプリフェッチ手法の効果が現れている。ただし，1CPU 実行時には，ソフトウェアプリフェッチ手法を適用してもロードストリーム数が多い場合に性能が低下している。これは，3.1節の n 配列総和とは異なり，n 配列加算では 1 イタレーション中の演算間に依存性があることから，命令スケジューリングの制約による性能低下が起こっていると見られる。

また，16CPU 実行においても，n配列総和とは異なり，ロードストリーム数が少ないほど性能が低く，ロードストリーム数が多いほど性能が高くなる傾向が見られる。これは，ロードストリーム数が少ないほど全メモリアクセスにおけるストアの比率が高まり，L3－メモリ間のバス使用効率が低下するためである。16CPU 実行ではメモリバンド幅がボトルネックとなって，1CPU 実行時に見えたロードストリーム数が多い場合の命令スケジューリングの影響が見えなくなる。

3.3 ベクトルー行列積

倍精度データのベクトルー行列積コードを用いて，ソフトウェアプリフェッチ手法の効果を評価した。評価にあたっては，図 10 a）に示すコードを外側ループ do i で 1～16 倍に展開し，ロードストリーム数を変化させた。外側ループで U 倍展開した場合のコードイメージを図 10 b）に示す。

外側ループ do i で U 倍展開すると，内側ループ do j における配列 C のロードストリーム数は U 個に増えるが，ベクトル B(j) の読み出しが共通化されるため，浮動小数点演算あたりの必要データアクセス量（B/flop）が低減される。そのため，ロードストリーム数が増えても実効メモリバンド幅が変わらなければ，性能の向上を期待できる。外側ループ U 倍展開時の内側ループの特性を表 1 にまとめる。

```
do i=1,m
   A1(i)=1.0+A1(i)+A2(i)+...+An(i)
End do
```

図7 n 配列の加算コード（倍精度データ）

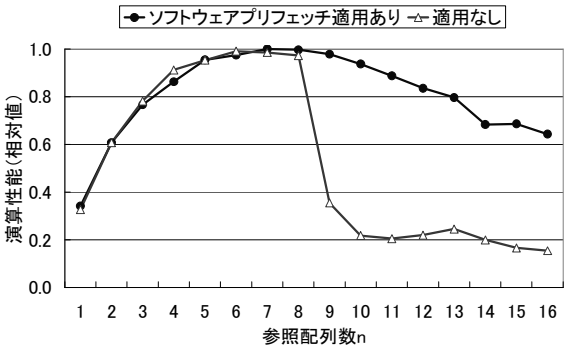

図8 n 配列加算の演算性能（1CPU 実行）

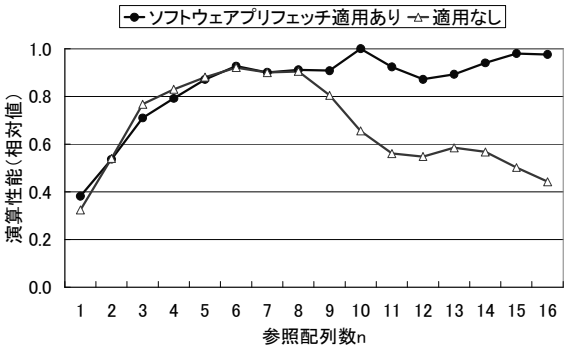

図9 n 配列加算の演算性能（16CPU 実行）

評価にあたっては図 10 a）のコードを用い，外側ループの展開数はディレクティブにより指定した。なお，本コードは外側ループで並列化される。

SR11000 モデル H1 の 16CPU で並列実行した結果を，図 11 に示す。結果は，最高性能を 1 とする相対値で示している。ソフトウェアプリフェッチを適用しない場合，外側ループ展開数 8 以上（ロードストリーム数 9 以上）で性能が大きく低下する。これに対しソフトウェアプリフェッチを適用することで，外側ループ展開数の増加に従って性能が向上し，16 倍展開時に性能が最大になる。

図 12 は，図 11 の結果を実効メモリバンド幅に換算したものである。16 倍展開時の実効メモリバンド幅を 1 とする相対値で示している。1～2 倍展開時には，外側ループの処理が進んでも内側ループでベクトル B が L2 キャッシュにヒットするため，実効メモリバンド幅が見かけの上で高くなっ

```
parameter(m=50000)
do i=1,m
   do j=1,m
      A(i)=A(i)+B(j)*C(j,i)
   end do
end do
```

a) 評価コード

```
do i=1,m,U
   do j=1,m
      A(i)=A(i)+B(j)*C(j,i)
      A(i+1)=A(i+1)+B(j)*C(j,i+1)
      ...
      A(i+U-1)=A(i+U-1)+B(j)*C(j,i+U-1)
   end do
end do
```

b) 外側ループで U 倍展開
[コード簡略化のため，m が U の倍数を仮定]

図10 ベクトルー行列積コード(倍精度データ)

表1 ベクトルー行列積コードの内側ループの特性

外側ループ U 倍展開時	
ロードストリーム数	U+1
必要データアクセス量	8(U+1)バイト
浮動小数点演算数(flop)	2U
B/flop	4(U+1)/U

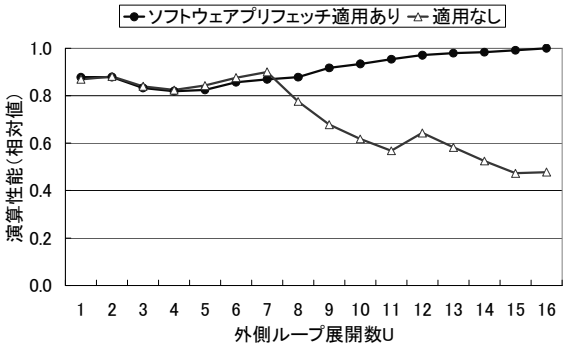

図11 ベクトルー行列積の演算性能

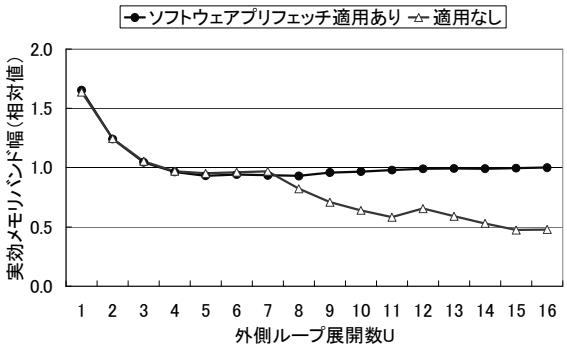

図12 ベクトルー行列積における実効メモリバンド幅

ている。しかし，3 倍展開以上ではすべてのロードストリー

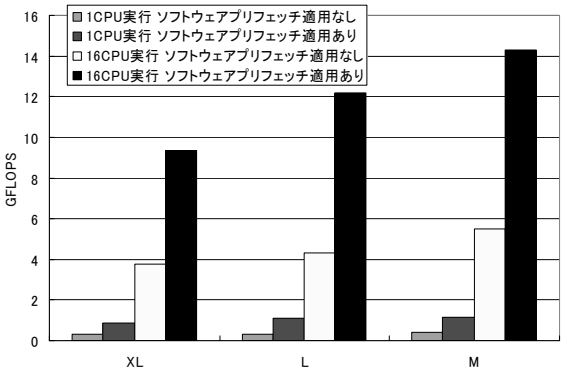

図13 姫野ベンチ結果

表2 姫野ベンチにおけるソフトウェアプリフェッチ効果

問題サイズ	並列	逐次
XL	14.1GB	2.48 倍	2.90 倍
L	1.77GB	2.84 倍	3.36 倍
M	0.22GB	2.59 倍	2.26 倍

ムがキャッシュミスするため，実効メモリバンド幅を正しく反映している。ソフトウェアプリフェッチを適用した場合，実効メモリバンド幅はほぼ一定であるが，外側ループ展開数の増大により B/flop が低減するため，図 11 の演算性能向上となっていることがわかる。

3.4 姫野ベンチ

非圧縮流体解析コードの性能評価プログラムである姫野ベンチ[10]で，ソフトウェアプリフェッチ手法の効果を評価した。姫野ベンチのカーネルは，演算部(ロードストリーム 21 個，ストアストリーム 1 個，34 演算)とコピー部(ロードストリーム 1 個，ストアストリーム 1 個，演算なし)から成る。演算部には 21 個のロードストリームがあるが，問題サイズによっては一部がキャッシュヒットする。

評価にあたっては，姫野ベンチのソースコードを修正せずに用いた。問題サイズ XL(1024×512×512)，L(512×256×256)，M(256×128×128)それぞれについて SR11000 モデル H1 の 1CPU 逐次実行／16CPU 並列実行した結果を，図 13 に示す。また，ソフトウェアプリフェッチを適用しない場合に対するソフトウェアプリフェッチ適用時の性能の比を，表 2 に示す。ソフトウェアプリフェッチを適用することで 2 倍以上の性能向上を実現しており，本ソフトウェアプリフェッチ手法の有効性を確認できる。

4 考察

3.1節および3.2節の結果は，ロードストリーム数の多いループで高い並列性能を実現するための重要な情報を含んでいる。

文献7)では，ハードウェアプリフェッチ機構の制約により

```
do i=1,m
   A1(i)=1.0+A1(i)+A2(i)+...+A8(i)
end do
do j=1,m
   A1(j)=A1(j)+A9(j)+...+A15(j)
end do
```

図14 15 配列加算のループ分割

表3 15 配列加算の性能

ソフトウェアプリフェッチ	ループ分割	相対性能
適用なし	適用なし	1.00
適用なし	適用あり	1.82
適用あり	適用なし	1.95

ロードストリーム数が 8 を超えると性能低下が起こることを紹介するとともに，その対策として 8 ロードストリーム以下のループに分割することを推奨している。例えば 15 配列の加算では，図 14 のようにループを二分割し，前半ループ do i と後半ループ do j のロードストリーム数をそれぞれ 8 とする。しかし，この場合，前半ループで書き込んだ配列 A1 を後半ループで再度読み出す必要があり，メモリアクセス量が増大する。

これに対し，ソフトウェアプリフェッチ手法を適用すれば，ロードストリーム数が多い場合でもループを分割する必要がない。ソフトウェアプリフェッチ手法の適用を前提とすることで，プログラム作成者およびコンパイラは，ロードストリーム数の増大による実効メモリバンド幅の低下を考慮する必要がなくなる。

ソフトウェアプリフェッチ手法を適用することで，例えば n 配列の総和では，ロードストリーム数によらずほぼ安定した性能となる。また，n 配列の加算では，ループ分割に伴うメモリアクセス量の増大を避けられるため，より高い性能を実現できる。15 配列の加算について，ソフトウェアプリフェッチおよびループ分割のいずれも適用しない場合の性能を 1 とする相対性能を，表 3 に示す。

5 まとめと今後の課題

本稿では，SR11000 モデル H1 におけるソフトウェアプリフェッチ手法の効果を評価した。POWER4+のハードウェアプリフェッチ機構は，8 個を超えるロードストリームを含むループでは実効メモリバンド幅が低下する。これに対しソフトウェアプリフェッチ手法を適用した場合には，ロードストリーム数が増えた場合にも安定して高い実効メモリバンド幅を実現できることを確認した。この結果，ソフトウェアプリフェッチ手法の適用により，ループ内のストリーム数を考慮したループ分割が不要なことを明らかにした。日立最適化 FORTRAN90 コンパイラは本ソフトウェアプリフェッチ手法の自動適用をサポートしており，安定した高い性能を実現する。

今後の課題としては，他の CPU を搭載する計算機において，本手法の効果の有無を確認することが挙げられる。

参考文献

1) Zhenlin Wang, Kathryn S. McKinley and Doug Burger: “Combining Cooperative Software/Hardware Prefetching and Cache Replacement”, IBM Austin Center for Advanced Studies Conference (2004).
2) “IA-32 Intel Architecture Optimization Reference Manual”, Intel Manual, Order Number 248966-011 (2004).
3) “QuantiSpeed Architecture”, AMD White Paper (2001).
4) 井上愛一郎: “UNIX サーバ用プロセッサ:SPARC64 V”, 雑誌 FUJITSU 2002-11 月号, pp. 450-455 (2002).
5) Yoshiko Tamaki, Naonobu Sukegawa, Masanao Ito, Yoshikazu Tanaka, Masakazu Fukagawa, Tsutomu Sumimoto and Nobuhiro Ioki: “Node Architecture and Performance Evaluation of the Hitachi Super Technical Server SR8000”, Proceedings of 12th International Conference on Parallel and Distributed Computing Systems, pp. 487-493 (1999).
6) J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le and B. Sinharoy: “POWER4 system microarchitecture”, IBM Journal of Research and Development, Vol. 46, No. 1, pp. 5-25 (2002).
7) Charles Grass: “Optimizing for Performance on IBM POWER4 Systems”, ScicomP7 Tutorial (2003), http://www.spscicomp.org/ScicomP7/Presentations/Grassl-ScicomP7-Optimizing_for_POWER4_tutorial.pdf
8) LMbench, http://www.bitmover.com/lmbench/
9) Joe Wetzel, Ed Silha, Cathy May and Brad Frey: “PowerPC Virtual Environment Architecture Book II Version 2.01”, IBM PowerPC Architecture Book (2003).
10) 姫野ベンチ, http://w3cic.riken.go.jp/HPC/HimenoBMT/
11) 中村友洋, 高山恒一, 青木秀貴, 松居昭宏, 助川直伸: “SR11000 モデル H1 におけるバリア同期の高速化手法”, 情処研報, ARC-155-10 (2003).
12) 青木秀貴, 高山恒一, 中村友洋, 松居昭宏, 助川直伸: “SR11000 モデル H1 のノード構成とスケーラビリティ評価”, 情処研報, ARC-155-11 (2003).
13) 松居昭宏, 助川直伸, 高山恒一, 青木秀貴, 中村友洋: “SR11000 モデル H1 における高精度性能分析手法”, 情処研報, ARC-155-12 (2003).
14) 青木秀貴, 處雅尋, 本川敬子, 五百木伸洋, 石原修: “SR11000 モデル H1 のソフトウェアプリフェッチ手法”, 電子情報通信学会 2004 年総合大会, D-6-11 (2004).