姫野ベンチマークによるItanium®2とXeon™の性能比較とその分析

山本 昌生，山村 周史，久門 耕一
（株）富士通研究所
E-Mail: {masao, yamamura, kumon}@flab.fujitsu.co.jp

本論文では、Itanium 2 や Xeon の性能評価や比較および最適化ポイントについて、HPC(High Performance Computing)向けベンチマークの実測結果に基づいて述べる。ベンチマークには HPC 分野の性能評価で広く利用されている姫野ベンチマークを使用した。また、各 CPU の性能モニタリング機能を利用して、CPU やメモリのアーキテクチャレベルでの性能分析も行った。実験の結果、両 CPU とも HPC 向けにはプリフェッチによる最適化が非常に効果的であることがわかった。とくに、Itanium 2 ではコンパイラが行うソフトウェアプリフェッチのメモリ最適化効果によって、その性能が 0.5GFLOPS～1.4GFLOPS と大きく変化し、コンパイラの最適化により性能が大きく左右されることがわかった。

キーワード: Itanium 2，Xeon，姫野ベンチマーク，性能評価，プリフェッチ

Performance comparison and analysis of Itanium® 2 and Xeon™, using Himeno-BENCHMARK

Masao Yamamoto，Shuji Yamamura，and Kouichi Kumon
Fujitsu Laboratories，LTD.
E-Mail: {masao, yamamura, kumon}@flab.fujitsu.co.jp

In this paper, we describe the performance evaluation, the comparison and the optimization of Itanium 2 and Xeon, based on the result of the benchmark for HPC(High Performance Computing). For the evaluation, we used the Himeno-BENCHMARK, widely used in the HPC field for the performance evaluation. Moreover, we analyzed the performance from the viewpoint of microarchitecture by using the performance monitoring counters built into the processor. The experimental results showed that memory prefetch operations are very effective for HPC on the both CPUs. Especially on Itanium 2, the performance has changed widely from 0.5GFLOPS to 1.4GFLOPS by the effect of a memory optimization with prefetch instructions which are generated by the compiler. So, we found that the performance of Itanium 2 is more strongly controlled by the compiler's optimizations than the performance of Xeon.

Key Words: Itanium 2，Xeon，Himeno Benchmark, Performance Evaluation, Prefetch

1. はじめに

2002 年 7 月、IA-64(64bits Intel® Architecture)プロセッサの最新版である Itanium 2[1]の出荷が開始された。Itanium 2 は主にエンタープライズ分野や HPC 分野向けとして開発されたプロセッサである。一方、本来デスクトップ分野を主なターゲットとする IA-32(32bits Intel Architecture)プロセッサもまた、近年の動作周波数の向上によって性能を大幅に向上しており、その最新版である Xeon[2]は HPC 向けとして利用できるレベルに達している。しかし、これら 2 つの各アーキテクチャでの最新 CPU は、いずれも市場に投入して間もない製品であり、未だその性能についての評価例は少なく、とくに、Itanium 2 の実システムでの性能評価や分析が不足している。

そこで、本報告では、Itanium 2 と Xeon の各性能評価および比較を行う。とくに、HPC の観点からの性能評価を行い、CPU やメモリアーキテクチャレベルでの性能分析もあわせて行うこととする。それにより、Itanium 2 や Xeon の実システム上で HPC 向けアプリケーションを動作させる場合の性能面での留意点や最適化ポイントについて明確にする。

そのために、本実験では、HPC 分野におけるコンピュータシステム性能評価に広く利用されている姫野ベンチマーク[3]を使用して、Itanium 2 と Xeon の性能を実測し、比較を

行う。

以降、まず第2章では各 CPU のアーキテクチャについて簡単にまとめ、第3章では姫野ベンチマークの特徴について述べる。続いて第4章では実験環境について、第5章では実験結果について報告を行い、Itanium 2 と Xeon の比較を行う。そして第6章ではプリフェッチの効果について考察し、第7章でまとめとする。

2. CPU アーキテクチャの特徴

2.1. Itanium® 2

Itanium 2 は、VLIW ライクな EPIC(Explicitly Parallel Instruction Computing)という、明示的に命令の並列実行を行うアーキテクチャを採用している。そのため、コンパイラが十分にソースコードから並列性を抽出し、最適化が行える場合には、命令レベルの並列性の向上やソフトウェアパイプライン等の効果による性能向上が期待できる。また、メモリ上のデータ配置やメモリアクセスパターンが静的で固定されている場合などで、コンパイラが十分に把握できる範囲であれば、ソフトウェアプリフェッチによるメモリ最適化が行われ、高い性能が得られる。しかし、コンパイラによる以上のような最適化が十分でない場合には、高い性能は期待できない。つまり、Itanium 2 の性能はコンパイラの最適化に大きく左右される。

また、Itanium 2 では、ロード命令やストア命令も並列実行可能であり、その効果を得るために、2次キャッシュが 16 バイト単位の 16 バンク構成[4]になっている。

2.2. Xeon™

Xeon は、スーパースカラによる並列実行を行うアーキテクチャを採用し、かつパイプラインを細分化して動作周波数を高くすることにより演算性能向上を狙った CPU である。また、メモリ上の連続するアドレスにアクセスする場合には、ハードウェアプリフェッチが動作し、メモリアクセス遅延を隠蔽することによる性能向上も期待できる。この他、Out Of Order 実行もサポートしている。この様に、Xeon は、ハードウェアによる性能最適化部分が大きい。

Xeon では2次キャッシュは1バンク構成になっているので、2次キャッシュのスループットは Itanium 2 よりも小さくなる。

3. 姫野ベンチマークの特徴

3.1. プログラム概要

姫野ベンチマーク(以下、姫野ベンチ)は、理化学研究所の姫野龍太郎氏が非圧縮流体解析コードの性能評価のために開発したベンチマークプログラムである。このプログラムは、ポアッソン方程式解法をヤコビ反復法で解く場合の主要なループの処理速度を測定するベンチマークである。主要ループ内では、配列への連続アクセスとその配列要素に対する浮動小数点演算が行われている。そのため、姫野ベンチは計算機のメモリやキャッシュ性能による影響が大きい。

また、測定結果として対象システムの実測速度(MFLOPS値)を出力する。この MFLOPS 値を性能指標として、システム間の性能比較を行うことができるので、HPC 分野での性能測定用ベンチマークプログラムとして広く利用されている。

姫野ベンチには、新バージョン(姫野ベンチ Xp)と以前のバージョン(姫野ベンチ 98)があり、それぞれ、ソースプログラムの記述言語によりC版と Fortran 版がある。よってソースプログラム的には4種類存在する。本実験ではこれら4種類全てについて測定を行う。

3.2. メモリアクセスパターン

姫野ベンチは種類により、メモリ上の配列データへのアクセスパターンが異なる。姫野ベンチ 98(C版)だけが、他の3つの姫野ベンチと、主要ループ内でのアクセスパターンが異なる。

```
for(k=1 ; k<kmax-1 ; k++){
    s0 = a[i][j][k][0] * p[i+1][j  ][k  ]
       + a[i][j][k][1] * p[i  ][j+1][k  ]
       + a[i][j][k][2] * p[i  ][j  ][k+1]
       + …
}
```

(a) 姫野ベンチ 98(C 版)… シーケンシャル

```
for(k=1 ; k<kmax-1 ; k++){
    s0 = a[0][i][j][k] * p[i+1][j  ][k  ]
       + a[1][i][j][k] * p[i  ][j+1][k  ]
       + a[2][i][j][k] * p[i  ][j  ][k+1]
       + …
}
```

(b) 姫野ベンチ XP(C 版)… ストライド

図 1. 配列アクセス・パターンの実装例

姫野ベンチ 98(C版)は、図1(a)のようにメモリ上の連続したアドレスに続けてアクセスする(シーケンシャルアクセス)。対して他の3つの姫野ベンチでは、図1(b)のようにメモリアクセスが連続アドレスではなく飛び飛びにアクセスしている(ストライドアクセス)。

4. 実験環境

姫野ベンチによる性能測定を行う各 CPU の実験環境を表1および表2に示す。コンパイラは両CPU環境とも、以下の2種類を用いた。

- Windows 版 Intel C++ コンパイラ 7.0
- Windows 版 Intel Fortran コンパイラ 7.0

また、アーキテクチャレベルの分析には、各 CPU 内部に備えられた性能モニタリング機能[5][6]を利用して、姫野ベンチ実行時の実行命令数や浮動小数点演算命令数、キャッシュミス数等の各種イベントデータを採取した。

表 1. Xeon 実験環境(self-made machine)

CPU	Xeon™ 2.0 GHz × 1CPU (HyperThreading オフ)
L1/L2 Cache	8 KB / 512KB
Memory	512 MB
OS	Windows® 2000 Server
最適化オプション	/G7 /O3

表 2. Itanium 2 実験環境(hp server rx2600)

CPU	Itanium® 2 1GHz ×1CPU
L1/L2/L3 Cache	16KB+16KB / 256KB / 3MB
Memory	1 GB
OS	Windows® .NET Server 2003 (64bit) EnterpriseServer RC2 JP
最適化オプション	/G2 /O3 /Qprof_use /Qipo

姫野ベンチは以下の4種類を用いた。

- 姫野ベンチ 98 (C 版)
- 姫野ベンチ 98 (Fortran 版)
- 姫野ベンチ Xp (C 版)
- 姫野ベンチ Xp (Fortran 版)

それぞれの種類において、使用する配列のサイズとして"SMALL"、"MIDDLE"、"LARGE"等があるが、本測定では、各実験環境の物理メモリ量に適した共通のサイズとして「MIDDLE (配列要素数＝256 x 128 x 128)」を用いた。また、配列確保を動的に行う版と静的に行う版があるが、コンパイラの能力を最大限に引き出すために、本実験では静的確保で実験を行った。

5. 実験結果

5.1. 性能

表3に、各CPUの性能測定結果およびXeonを1とした場合のItanium 2性能比を示す。また、図2に、各CPUのFPC(Floating Per Cycle,1サイクルあたりの浮動小数点演算数)を性能データとあわせてグラフにしたものを示す。

表 3. Xeon および Itanium 2 の姫野ベンチマーク性能比較

			Xeon [MFLOPS]	Itanium2 [MFLOPS]	性能比
姫野ベンチ(M)	98	C	537	802	1.49
		Fortran	413	1400	3.39
	XP	C	392	1468	3.75
		Fortran	393	1650	4.20

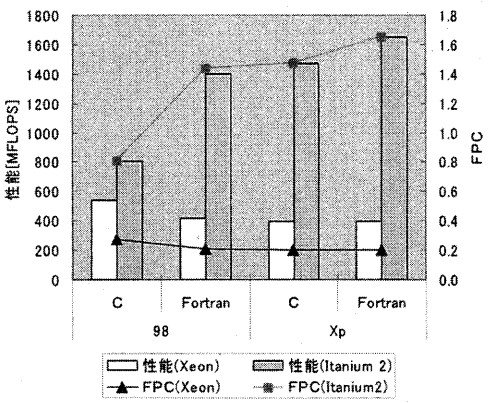

図 2. Xeon および Itanium 2 における性能と FPC

まず表 3 を見ると、姫野ベンチ 98(C 版)を除いて、Itanium 2 は Xeon に対して 3.39～4.20 倍の性能を示している。

つぎに図2を見ると、Itanium 2の場合、姫野ベンチ98(C版)を除けば、FPCは1.44～1.66という良好な値を示している。対して、Xeonでは0.20～0.27に留まっている。Itanium 2のFPC即ち並列実行性能が良いのは、第2章で述べた通り、Itanium 2がEPICアーキテクチャを採用しており、かつ姫野

ベンチは、主要ループ部のソースコードのサイズが小さく、コンパイラによる命令レベル並列性の抽出が比較的容易だからである。

続いて、各 CPU 個々の性能について見てみる。

まず、Itanium 2 は姫野ベンチ 98(C版)を除けば 1.4～1.6GFLOPS という非常に良好な実測結果が得られている。また、この性能は市販のコンパイラによる最適化のみで実現できることに注意されたい。しかし、姫野ベンチ 98(C版)のみ極端に性能が悪く、他の姫野ベンチ結果の約5～6割程度の性能しか出ていない。

一方、Xeon では、逆に姫野ベンチ98(C版)のみ他より性能が 1.3～1.4倍高くなっている。姫野ベンチ 98(C版)以外は、Itanium 2 同様、ほぼ同性能という結果となっている。

5.2. キャッシュミス率

各 CPU において、性能傾向は逆にしろ、姫野ベンチ 98(C版)のみ、残り3つの姫野ベンチの結果と異なっている。本節では、姫野ベンチの性能を大きく左右するキャッシュミスの観点からさらに分析してみる。

図 3 に、命令あたりのキャッシュミス率(キャッシュミス回数／総命令数)および性能のグラフを示す。なお、Xeon は 3 次キャッシュメモリを装備していないので該当するデータは示していない。

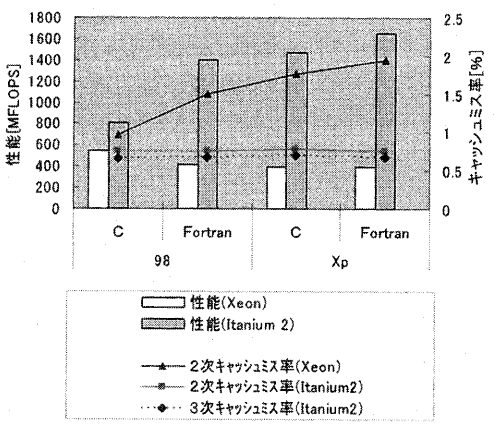

図 3. Xeon および Itanium 2 における性能とキャッシュミス率

図 3 を見ると、姫野ベンチ 98(C 版)を除いて、両 CPU で 1%以上の差が見られる。これが、CPU 間での性能差の一因と考えられる。

個々の CPU について見てみると、Xeon では、前節でも述べた通り、その高い動作周波数のためにキャッシュミスによるペナルティの影響が大きく出てしまう。そのため、キャッシュミスが大きな性能阻害要因の一つとなる。よって逆に、図3の結果ではキャッシュミス率の最も低い姫野ベンチ 98(C版)の性能値が一番高くなっていて良い。即ち、Xeon ではこの姫野ベンチ 98(C版)が、キャッシュに関する最適化の効果がもっとも良く現れているといえる。これは、姫野ベンチ98(C版)のメモリアクセスがシーケンシャルであることに起因する、ハードウェアプリフェッチによる効果だと考えられる。

一方、Itanium 2 に関しては、図3をみると、キャッシュミス率はすべての姫野ベンチで 0.75%前後と低く、ほぼ一定な値となっている。唯一性能が悪い姫野ベンチ 98(C版)でも、キャッシュミス率は他と同じで 0.75%と低い。そこでさらに2次キャッシュメモリについて調査を行う。その際、2次キャッシュミスに起因する L2_FORCE_RECIRC[5]というイベントを指標として用いてみることとした。

この L2_FORCE_RECIRC というイベントは、2次キャッシュへの再要求回数を示すイベントである。具体的には、2次キャッシュミスが発生した後、再度リード要求を行ってその時まだデータが来ていない場合にカウントされる。従って、より現実的なキャッシュミスによるペナルティを表しているものと考えることができる。本論文では、この2次キャッシュミスに起因する L2_FORCE_RECIRC イベントを「2次キャッシュ再要求回数」、それを実行総命令数で割った値を「2次キャッシュ再要求率」と定義する。それらをグラフ化したのが図4である。

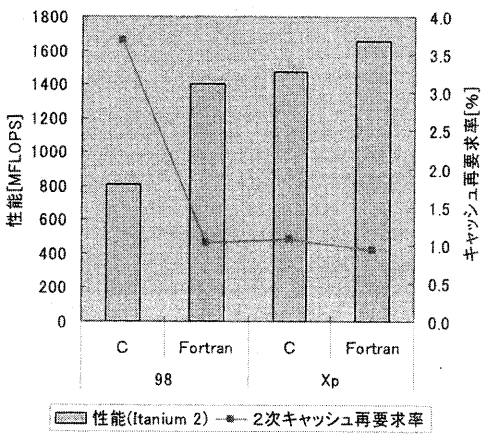

図 4. Itanium 2 における性能と2次キャッシュ再要求

図 4 を見ると、姫野ベンチ 98(C 版)のみ、2次キャッシュ再要求率が高く、「Itanium 2 では、姫野ベンチ 98(C版)のみ性能が悪い」ことへの大きな要因となったと考えられる。従って、Itanium 2 では姫野ベンチ 98(C版)だけが、キャッシュに関する最適化がうまく効いていないといえる。これはプリフェッチが間に合っていないか、その実行コード内にソフトウェアプリフェッチが含まれていないことが考えられる。

結局、両 CPU とも、プリフェッチによるキャッシュミスへの影響が性能を左右していると考えられる。そこで次章では、XeonとItanium 2それぞれにおける、プリフェッチによる最適化の効果について明らかにする。

6. プリフェッチによる最適化の効果

6.1. Xeon™での効果

本章では、姫野ベンチ性能にとってプリフェッチが有効な最適化要因となっているのかどうか、どの程度の効果を実際に発揮しているのかどうかについて実験してみる。また、なぜ姫野ベンチ 98(C版)だけが、他の3つの姫野ベンチと性能が異なり、Xeon では良く、逆に Itanium 2 で悪くなるのかについても検証する。

姫野ベンチのプログラムは、3.1 節で説明した通り、4種類ある。そのうち、姫野ベンチ 98(C版)だけがメモリ上のデータに連続アクセスするという特徴がある。これにより、姫野ベンチ 98(C版)の実行時のみ、ハードウェアプリフェッチが動作し、それが効果を発揮して、図 3 で見た様に、キャッシュミス率が他の3つよりも低く、性能も良くなっていると考えられる。逆に他の3つの姫野ベンチでは、プリフェッチは動作していないため、キャッシュミス率も高く、性能もその分低下している。このことを確認するために、CPU の設定で実際にハードウェアプリフェッチを無効にして性能測定を行ったところ、4種類の姫野ベンチの性能差はほとんどなくなった。よって、Xeon における姫野ベンチ実行時のハードウェアプリフェッチの効果が高いことが確認できた。

6.2. Itanium® 2 での効果

Itanium 2 におけるプリフェッチについては、2つの手段がある。

1. **ロード/ストア命令のポストインクリメントによる暗黙的プリフェッチ**
2. **lfetch 命令による明示的なプリフェッチ**

これらが実際に姫野ベンチ実行時に有効に動作しているのかどうか、ここでは姫野ベンチ Xp(C版)からコンパイラにより生成したアセンブラソースによる実験を行った。

まずは、1について、アセンブラソースにおいて、姫野ベンチの最内ループ内のload命令のポストインクリメント(約20個)を、add 演算によるベースアドレスのインクリメント処理に置きかえた。ソース上では、処理追加によるバンドル数及び命令グループ数はともに高々1個である。よってそれによる性能への影響はほとんどない。実測定してみたところ、性能は全く変わらなかった。よって、姫野ベンチ実行時の暗黙プリフェッチによる効果は全く無いことが分かった。

つぎに、2について、最内ループ内の lfetch 命令を無効にする実験を行った。姫野ベンチ Xp(C版)では最内ループ内に4つの lfetch 命令が挿入されている。その4つすべてを無効にして、性能測定した結果を図 5 に示す。

その結果を見ると、2次キャッシュ再要求率が、1.1%から4.2%に大幅に上昇し、性能は約69%の大幅ダウンとなっていることが分かる。

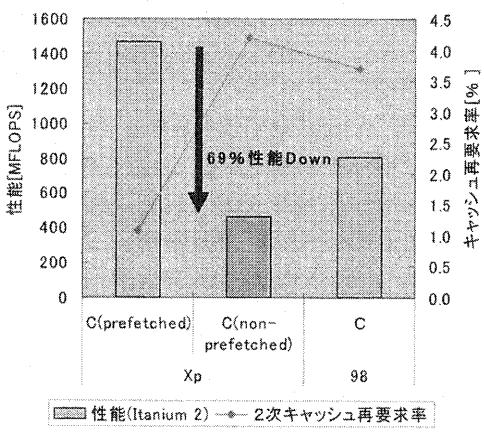

図 5. lfetch 命令によるプリフェッチの効果

また、残りの3つの姫野ベンチについても調べてみた。その結果、lfetch命令はすべての姫野ベンチに含まれており、かつ4種類の姫野ベンチとも lfetch 命令がなければすべて同程度の性能であった。よって、lfetch 命令による最適化を行った結果、姫野ベンチの種類によってその効果に差があったといえる。即ち、最低でも 1.5 倍以上の性能向上が得ら

れたが、4種類のうち3つは3倍以上の性能向上になっていた。

以上より、Itanium 2 で姫野ベンチの様な HPC 向けアプリケーションプログラムを動作させる場合、lfetch 命令による明示的なプリフェッチの効果が非常に大きいことが分かった。

7. まとめ

本論文では、最新の IA-32/IA-64CPU である、Xeon/Itanium 2 について、HPC の観点から姫野ベンチマークを用いた性能評価を行った。その結果、コンパイラによる命令レベル並列性の抽出が比較的容易で、かつメモリアクセスの最適化が行える場合、Itanium 2 は Xeon と比較して最大 4.2 倍もの性能を発揮した。しかし、コンパイラによる最適化がうまくいかない場合には、Xeon と同等の性能となることもあった。

また、実際に姫野ベンチの様な HPC 向けプログラムを使用する場合、両 CPU ともプリフェッチによる最適化が有効であることも今回の実験で確認できた。とくに Itanium 2 の場合は、プリフェッチによる性能向上が約3倍と非常に大きい。しかしプログラム内容によって、その効き具合の差が極端であったり、扱い難い一面もある。一方、Xeon の場合は、Itanium 2 より最適化と性能の関係は素直であり扱いやすい。最適化については、動作周波数に見合った性能を得るために、キャッシュミス率をいかに抑えることができるかが重要ポイントとなることが分かった。

今回用いたのは HPC 向けベンチマークであり、プログラム自体も小さかったので、コンパイラにとっては最適化が行い易かった。しかし、ビジネス向けアプリケーションではコンパイラレベルでのメモリアクセスの最適化が難しく、Itanium 2 にとっては不利となり、Xeon との性能差も少なくなると思われる。今後、評価する予定である。

参考文献

[1] Intel® Itanium® 2 プロセッサ,
http://www.intel.co.jp/jp/developer/design/itanium2/index.htm

[2] Intel® Xeon™ プロセッサ,
http://www.intel.co.jp/jp/developer/design/xeon/index.htm

[3] Himeno BENCHMARK,
http://w3cic.riken.go.jp/HPC/HimenoBMT/index.html

[4] William Jalby and Christophe Lemuet. Exploring Optimizing Itanium®2 Cache(s) Performance for Scientific Computing. In *Proceedings of the 2nd Workshop on Explicitly Parallel Instruction Computing Architecture and Compilers*, November 2002.

[5] *Intel® Itanium® 2 Processor Reference Manual For Software Development Optimization* (251110-001), June 2002.

[6] *IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide* (245472-009), 2002.