2007－HPC－110 (3)
2007／6／8

Cell Broadband Engine へのSPEソフトウェアデータキャッシュの実装

佐藤 芳紀　神酒 勤
九州工業大学 大学院生命体工学研究科

Cell Broadband Engine の持つ SPE はマルチメディアデータを高速でリアルタイムに処理することを目的に最適化されたプロセッサである。キャッシュミスによるリアルタイム性の低下を防ぐため SPE はキャッシュを持たず、ソフトウェアによる DMA でデータ転送をスケジューリングする。ただし、この手法では必要となるデータの事前予測が容易でない場合には、パフォーマンスが著しく低下する。この問題を解決するために SPE の持つスクラッチパッドメモリの一部をソフトウェア制御のデータキャッシュとして使用することを提案する。実験結果から、クイックソートプログラムにおいて 5 倍以上の速度向上が得られた。

Mounting a SPE software data cache on the Cell Broadband Engine

Yoshiki Sato　Tsutomu Miki
Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology

The SPE of the Cell Broadband Engine is a processor optimized for multimedia data processing at high-speed and in real-time. To prevent increasing of operation time caused by cache miss, the SPE schedules data transfers using DMA instead of a cache. In the case that the prediction of the required data is hard, however, the performance decreases remarkably. To solve this problem, we propose to use a part of the scratch-pad memory of SPE as a data cache controlled by software. As a result, in a quick-sort program, we achieved more than five times speed-up.

1．はじめに

図1に示すCell Broadband Engineは東芝、ソニー・コンピュータエンタテインメント及びソニー、IBM Corporationが共同で開発した高性能マルチプロセッサである[1),2),3),4)]。Cell Broadband Engineはマルチメディアデータを高速でリアルタイムに処理するために最適化された8個のSPE（Synergistic Processor Element）を持つ。SPEはデータ処理に特化している反面、OSのような制御系の処理には適していないので、SPEと既存の汎用マイクロプロセッサとを組み合わせた非対称マルチコアプロセッサ構成をとっている。汎用マイクロプロセッサ PPE（PowerPC Processor Element）には、組込み用途から高性能サーバまで幅広い用途で実績のあるIBM PowerPCアーキテクチャが採用されている。

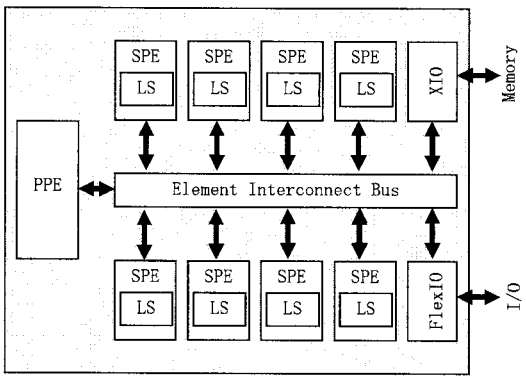

図1 Cell Broadband Engine の構成

SPEはキャッシュミスによるリアルタイム性の低下を防ぐためにキャッシュを持たず、代わりに256KBのローカルストレージ(LS)と呼ばれる専用のスクラッチパッドメモリを持つ。SPEは処理対象として、必要なデータの事前予測が容易なストリームデータを想定しているので、ソフトウェアによるDMAでデータ転送をスケジューリングし、将来必要になるデータを予めメインメモリからLSにデータ転送しておく必要がある。

2．SPE メモリアクセス体系の問題点

SPE では将来必要になるデータをソフトウェアによる DMA で LS 上に予めロードし、データ転送のレイテンシを隠蔽する方式をとる。しかし、将来必要になるデータが容易に予測できない場合には、データアクセスの度に DMA を発生させる必要があり、データ転送のレイテンシが悪化する。従って、必要なデータの事前予測が困難なプログラムは SPE には不向きであると考えられる。これを検証するために Cell リファレンスセット[5]を用いてクイックソートプログラムに対し実行時間の比較実験を行った。実験方法は、メインメモリ上に予め 4 MB 分の一様乱数を生成しておき、このデータのソーティングに要した時間を測定した。ここで、一様乱数の各要素は単精度浮動小数点型である。実験結果を図 2 に示す。図 2 において、1SPE は 1 個の SPE を用いた場合、PPE は PPE を用いた場合、また、参考として Pentium4 は Cell Broadband Engine と同一周波数の Intel Pentium4 プロセッサを用いた場合のクイックソートの実行時間をそれぞれ示している。図 2 に示すように、SPE 単体でのクイックソート実行時間は PPE での実行時間の 14.76 倍であった。また、図 3 に Cell ソフトウェア開発環境[6]に含まれるパフォーマンスモニタによる SPE 実行時間の内訳を示す。図 3 において ch stall は DMA 転送のために SPE がストールした時間の割合を示しており、この実験においては実行時間のおよそ 8 割が DMA 転送のためにストールしていることになる。以上から、クイックソートのように必要なデータの事前予測が困難なプログラムを Cell Broadband Engine 上で実行するには、SPE 上で実行すると DMA 転送によるストール時間が大きくなることが確認できた。

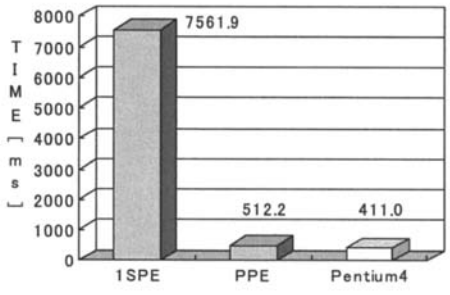

図 2　クイックソートプログラム実行時間の比較

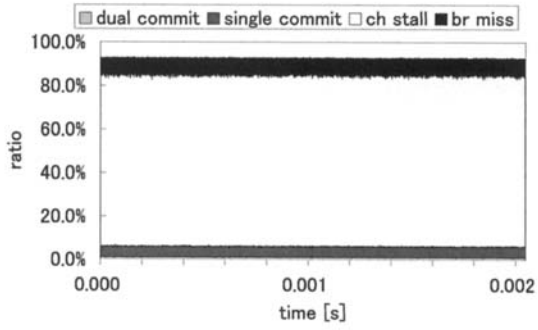

図 3　SPE によるクイックソート実行時間の内訳

3．SPE ソフトウェアデータキャッシュの実装

検証実験の結果から、アクセスパターンがランダムであり、必要なデータの事前予測が困難なプログラムは SPE に不向きであることがわかる。しかし、マルチプロセッサの恩恵を受けるには SPE を有効に使用することが不可欠である。一般に、アクセスパターンがランダムなプログラムであってもデータアクセスには局所性があることに着目し、SPE にソフトウェア制御のデータキャッシュを実装することを試みる。つまり、ソフトウェアによってキャッシュの動作をエミュレートし、SPE が持つ LS の一部をデータキャッシュとして使用する。

3．1　SPE ソフトウェアデータキャッシュの設計思想

SPE ソフトウェアデータキャッシュの実装にあたり、キャッシュコントローラの動作時間を小さくするためキャッシュの構成はできるだけシンプルなものを目指した。そこで、キャッシング方式にはダイレクトマップ方式を採用し、状態フラグは各エントリ内に有効なデータが格納されていることを示す valid flag のみを採用した。実装する SPE ソフトウェアデータキャッシュはプロセッサ間のコヒーレンシを保障しない。従って、ソフトウェアによってプロセッサ間のキャッシュの一貫性制御をスケジューリングする必要があり、各エントリを無効化する機能および各エントリの内容を強制的にメモリにライトバックする機能を持たせた。

3．2　SPE ソフトウェアデータキャッシュの仕様

SPE ソフトウェアデータキャッシュを使用するための関数群を図 4 に示す。SPE ソフトウェアデータキャッシュは必要なデータの事前予測が困難な配列変数をキャッシングすることを想定しており、メインメモリ上に格納されている配列変数の先頭アドレスと、配列のインデックスを元にデータを得る。プロセッサ間で共有される変数をキャッシングする場合には、SPE ソフトウェアデータキャッシュはキャッシュのコヒーレンシを保障しないので、ソフトウェアにより一貫性制御をスケジューリングする必要がある。プロセッサ間で共有される変数をキャッシングし、これを更新した SPE は wb_cache()関数により更新されたラインをメモリへ強制的に書戻す。その他の SPE は invalid_cache()関数により自身が持っているラインを無効化する。その後、無効化されたラインに含まれる変数のアクセスが要求されると、SPE ソフトウェアデータキャッシュは更新されたラインをキャッシュフィルし最新の値を得る。また、キャッシュの全ラインに対して書戻しや無効化が必要な場合には、それぞれ wb_cache_all()、invalid_cache_all()関数を用いる。

```
... = cache(x_addr, index);
4.a1　変数 x[index]の読出し

cache(x_addr, index) = ... ;
4.a2　変数 x[index]への書込み

wb_cache(x_addr, index);
4.b1　変数 x[index]が格納されているラインを書戻す

wb_cache_all();
4.b2　キャッシュに格納されている全ラインを書戻す

invalid_cache(x_addr, index);
4.c1　変数 x[index]が格納されているラインを破棄する

invalid_cache_all();
4.c2　キャッシュに格納されている全ラインを破棄する
```

図 4　SPE ソフトウェアデータキャッシュ関数群

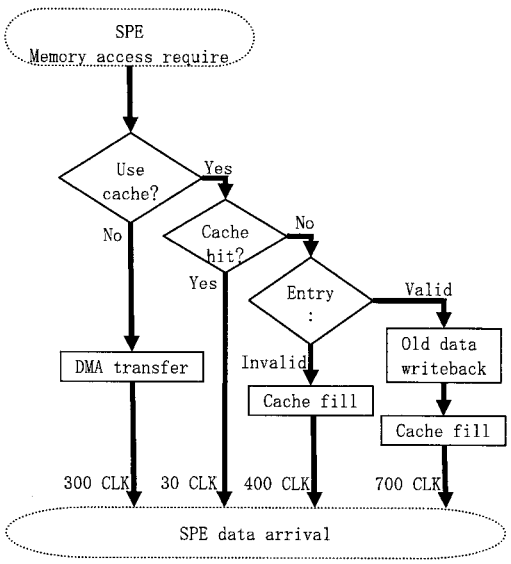

図 5　SPE ソフトウェアデータキャッシュの動作と動作時間の概算

図 5 に SPE ソフトウェアデータキャッシュへ読出し、または書込み要求が発生した場合の動作と動作時間の概算を示す。必要なデータの事前予測が不可能なプログラムにおいて、キャッシュを使用しなかった場合はデータアクセス要求の度に約 300 クロックを要している。一方、SPE ソフトウェアデータキャッシュを使用し、かつキャッシュヒットした場合には、約 30 クロックとなる。ただし、キャッシュミスの場合はキャッシュフィルが発生するためレイテンシは大きくなる。該当エントリに有効なデータが存在しなかった場合は約 400 クロックを要している。該当エントリに既に有効なデータが存在していた場合には、これをメモリに書戻した後に必要なデータをキャッシュフィルする必要があるので、約 700 クロックが必要となる。

SPE ソフトウェアデータキャッシュを搭載した場合の LS 圧迫量を表 1 に示す。プログラマは表 1 の LS 圧迫量の合計値を参考に、LS の空き容量に応じて適当な大きさのキャッシュを選択する。

表 1　SPE ソフトウェアデータキャッシュによる LS 圧迫量

容量[KB]	コード[KB]	データ[KB]	合計[KB]
0	0	0	0
4	1.068	4 + 0.125	4 + 1.193
8	1.068	8 + 0.250	8 + 1.318
16	1.068	16 + 0.500	16 + 1.568
32	1.068	32 + 1.000	32 + 2.068
64	1.068	64 + 2.000	64 + 3.068
128	1.068	128 + 4.000	128 + 5.068

4．SPE ソフトウェアデータキャッシュの評価

SPE ソフトウェアデータキャッシュを搭載し、2 章で述べた検証実験と同様の条件でソーティングに要する時間を測定した。この結果およびキャッシュヒット率を図 6 および図 7 にそれぞれ示す。SPE ソフトウェアデータキャッシュを搭載した場合、未搭載の場合と比較して 5.32～5.82 倍の速度向上が得られた。また、キャッシュの容量を大きくするに従い、キャッシュヒット率は上昇、実行時間は短くなっている。ここで、図 6 は 1 個の SPE を用いた場合の実行時間なので、複数個の SPE を用いて並列ソーティングした場合にはさらなる速度向上が期待できる。

一方、必要なデータの事前予測が容易なプログラムに対するキャッシュの影響を調べるために、4MB 分の単精度浮動小数点型変数で構成されるストリームデータの積和をとるプログラムの実行時間を測定した。この結果およびキャッシュヒット率を図 8 および図 9 にそれぞれ示す。積和演算プログラムではキャッシュ容量が 0KB、つまりキャッシュを搭載していない場合が最も高速であった。このプログラムでは必要なデータの事前予測が容易であるので、データアクセス要求が発生する前に予め DMA によってデータをロードできる。SPE ソフトウェアデータキャッシュを搭載しない場合では、キャッシュミスによるストールやキャッシュアクセスのオーバヘッドが発生しないのでキャッシュ搭載時よりも高速であった。また、ストリームデータ演算では 1 度アクセスされたデータが後に再びアクセスされる可能性は比較的低いため、キャッシュ搭載によるメモリアクセスの時間的局所性の恩恵が受けられず、キャッシュヒット率はその容量によらず 96.9%まで低下した。

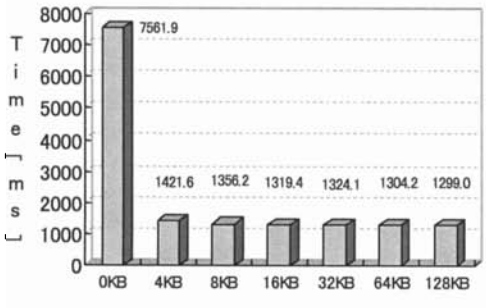

図6　ソフトウェアデータキャッシュ搭載時の
クイックソートプログラム実行時間

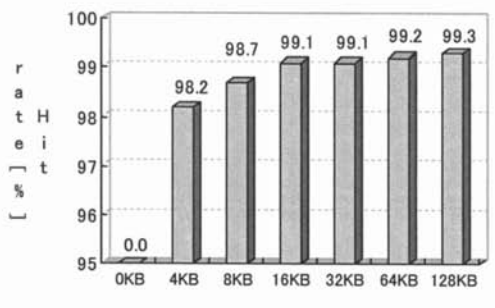

図7　クイックソートプログラムにおける
キャッシュヒット率

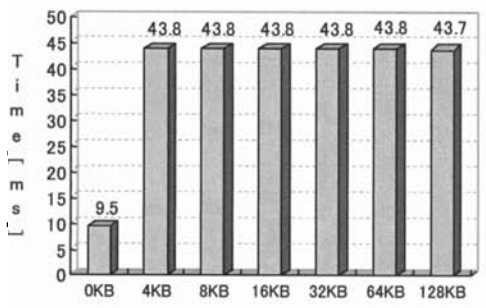

図8　積和演算プログラムの実行時間

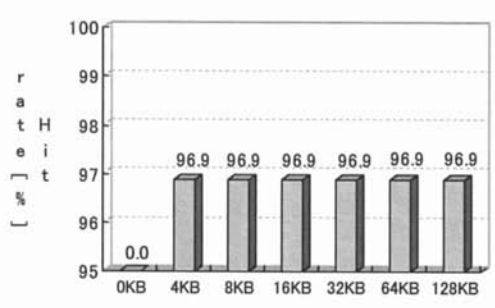

図9　積和演算プログラムにおけるキャッシュヒット率

5．まとめと今後の課題

Cell Broadband Engine の SPE はストリームデータを高速でリアルタイムに処理することを想定しているので、ランダムにアクセスされるデータには不向きである。そこで今回、データアクセスの局所性に着目し、ソフトウェアデータキャッシュの実装を試みた。その結果、キャッシュ未搭載の場合と比較して、キャッシュの容量に応じて 5.32 ～ 5.82 倍の速度向上が得られた。しかし、SPE 向きのストリームデータに対しては、ソフトウェアデータキャッシュを搭載しない方がキャッシュミスによるストールやキャッシュアクセスのオーバヘッドが発生しないので、より高速であった。ソフトウェアデータキャッシュを有効に使うには、データの性質によってキャッシュの使用、未使用を選択する必要がある。

今後は、作成したキャッシュの妥当性を検証し、SPE に最も適したキャッシュ構成を検討するとともに、キャッシュミス時のレイテンシ削減技術について検討する。

参　考　文　献

1) T. Chen et. al. : Cell Broadband Engine Architecture and its first implementation, IBM developerWorks (2005).

2)近藤 伸宏 ： CELL プロセッサに見るアーキテクチャ――次世代デジタルホームに向けて，東芝レビュー，Vol.60，No.7，pp.48-51 (2005).

3)林 宏雄，斉藤 光男，増渕 美生：Cell Broadband Engine の設計思想，東芝レビュー，Vol.61，No.6，pp.2-8 (2006).

4)黒澤 泰彦，渡辺 幸男，田胡 治之：次世代プロセッサ Cell Broadband Engine，東芝レビュー，Vol.61，No.6，pp.9-15 (2006).

5)上村 剛，大溝 孝，粟津 浩一：Cell リファレンスセット概要，東芝レビュー，Vol.61，No.6，pp.25-29 (2006).

6)大澤 諭，内川 貴幸，高野 秀隆：Cell ソフトウェア開発環境，東芝レビュー，Vol.61，No.6，pp.47-51 (2006).