CacheCore の動的最適化による積極的なデータ供給支援

森 洋介 † 吉瀬 謙二 ‡
東京工業大学工学部情報工学科 † 東京工業大学大学院情報理工学研究科 ‡

1 はじめに

今後、多くのコアを載せたメニーコアプロセッサの時代を迎えると考えられる。しかし多数のコアがメインメモリに集中的にアクセスすることで，通信レイテンシが増大する問題などから，性能がコア数の増加と共に線形に増加しない．このような背景から、我々はマルチコアプロセッサのコアの利用法として、他コアへのデータ供給支援を目的とするキャッシュコア (**CacheCore**)[1] の研究を進めている．

本稿では，Cell/B.E. の SPE に計算コアとキャッシュコアを実装する．そしてキャッシュコアに様々な最適化を施し，その性能向上を明らかにする．

2 キャッシュコアアーキテクチャ

図 **1** にキャッシュコアアーキテクチャを示す．図左に示すように，アプリケーションを動作させるコアを計算コア (ComputationCore) と呼ぶ．計算コアではソフトウェア L1 キャッシュも搭載する．図右に示すキャッシュコアは計算コアの L2 データキャッシュとして機能し，計算コアへのデータ供給を支援する．すなわち，通信レイテンシの大きいメモリアクセスの回数を減少させ．処理速度の向上を目指す．

キャッシュコアを利用するアプリケーションはデータアクセスにおいて，まず計算コアの L1 キャッシュを参照する．L1 キャッシュでミスが起こるとキャッシュコアの L2 キャッシュを参照する．計算コアは L1 キャッシュと，データ転送用のアドレスとフラグを含む．キャッシュコアは L2 キャッシュと，データ転送用のアドレスとフラグを持つ．これを互いに転送することで処理の要求や同期を実現する．

キャッシュ方式はソフトウェア処理の簡単化のため，ダイレクトマップを採用する．リプレース方式はメインメモリアクセスの削減のため，L1,L2 キャッシュ共にライトバック方式を採用する．

3 キャッシュコアの最適化

3.1 L2 キャッシュヒット時の最適化

図 **2** に L2 キャッシュヒット時のデータ転送の流れを示す．上部に示す最適化前は，L1 キャッシュミス (**1**) が起きるとキャッシュコアにリクエストを送り (**2**)，キャッシュコアからデータが送られてくるのを待つ (**3**) 方式である．下部に示す最適化したものでは，直接 L2 キャッシュからデータを取得する (**3**) 方式に変更する．これにより，通信に要するオーバーヘッドを DMA 転送 1 回分削減する．

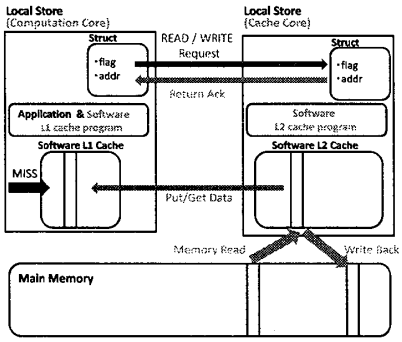

図 1: キャッシュコアアーキテクチャ

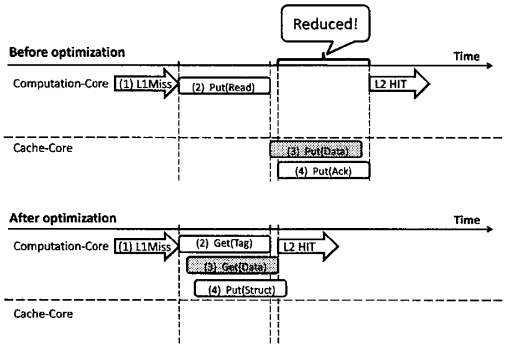

図 2: L2 キャッシュヒット時の最適化による通信オーバーヘッドの削減

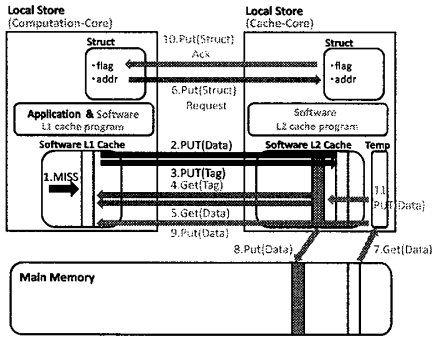

図 3: 今回最適化したライトバック処理を含むデータ転送

3.2 ライトバック処理の最適化

図 **3** にライトバック処理を含む実装のデータ転送の流れを示す．**矢印 2,3** は L1 ライトバック処理である．L2 キャッシュに直接書き込めるように最適化を施している．

The aggressive support of memory data access with dynamic optimization of the CacheCore.
Yosuke MORI, and Kenji KISE
† Department of Computer Science, Tokyo Institute of Technology
‡ Graduate School of Information Science and Engineering, Tokyo Institute of Technology

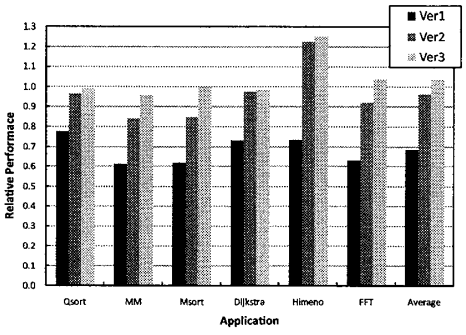

図 4: 各キャッシュコアの相対性能の平均

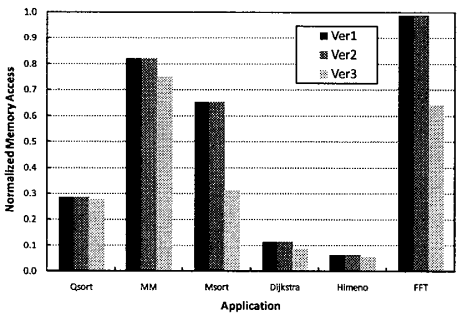

図 5: 各キャッシュコアのメモリアクセス数

また，キャッシュコアに受信バッファ(**Temp**) を設けることで L2 ライトバック処理のオーバーヘッドを削減できる．すなわち，この受信バッファにより，メインメモリからデータを取得する処理 (**矢印 7**) と平行してライトバック処理 (**矢印 8**) を行えるため，ライトバックのオーバーヘッドを隠蔽できる．

3.3 Victim キャッシュの追加

LS の容量の都合上，ダイレクトマップの容量が 128KB までしかとれない欠点がある．そこでキャッシュコアに **Victim** キャッシュ[3] を追加することで，キャッシュコアの性能向上を目指す．本研究では Victim キャッシュを 4way-SetAssociative で構成し，容量を 64KB とした．これによって 256KB の LS の容量を有効に活用できる．

今後，データサイズが小さく Victim キャッシュが埋まらないときは，Victim キャッシュのメモリをプリフェッチを行うバッファに動的に変更することを検討していく．

4 評価

4.1 評価環境

Cell/B.E. を搭載した実機 (PLAYSTATION3) を用いて最適化を行ったキャッシュコアを評価する．キャッシュコアを評価する基準として，計算コアに L1 ソフトウェアキャッシュを実装する版（利用するコアは 1 個）の性能を 1 とした際の相対性能を用いる．また，キャッシュコアを評価するプログラムとして次の 3 種類を用いる．

Ver1. 最適化前の [1] で用いたキャッシュコア．
Ver2. 3.1 節と 3.2 節の最適化を施したキャッシュコア．
Ver3. **Ver2** にさらに 3.3 節の Victim キャッシュを追加したキャッシュコア．

L1 キャッシュの容量は 2KB である．また，L2 キャッシュの容量は 128KB である．キャッシュのラインサイズは 128Byte である．

4.2 評価結果

図 4 に各キャッシュコアの相対性能を示す．Ver1 との比較から今回の最適化により大幅に性能が向上したことが分かる．キャッシュコアを使用することで Ver1 では 30%程の性能低下が見られたが，Victim キャッシュを追加した Ver3 では平均で 3%程の性能向上を達成している．また Ver3 の各アプリケーションの最大性能のみを見た場合では，Msort や FFT において 1.5 倍，平均で 1.3 倍の性能向上が得られる．データサイズがキャッシュコアのキャッシュ容量に収まる場合のキャッシュコアの有用性が確認できる．

また，**図 5** に各キャッシュコアを使用した場合のメインメモリアクセス数を示す．L1 キャッシュのみ使用した場合のメモリアクセス数の平均を 1 とした時の比を示している．キャッシュコアの使用により，メインメモリアクセス数の減少を達成している．特に Dijkstra や Himeno では 90%以上の削減を確認できる．

図 4 と**図 5** から Victim キャッシュの効果は大きく，性能向上とメインメモリアクセス数の両方に効果を現していることが分かる．

5 おわりに

本稿ではキャッシュコアに対し，L2 キャッシュヒット時の最適化，ライトバック処理の最適化，Victim キャッシュの追加を行った。その結果，初期実装よりも大幅な性能向上を達成した．

今後は，プリフェッチなどを用いてキャッシュコアを動的に最適化し，さらなるヒット率の向上を目指す．また、複数の計算コアと複数のキャッシュコアの構成における通信・同期機構の検討も行う．

謝辞

本研究の一部は，財団法人北九州産業学術推進機構 (FAIS) の支援による．

参考文献

[1] 森谷 章, 藤枝 直輝, 佐藤 真平, 吉瀬 謙二: メニーコアプロセッサに向けたデータ供給を支援する多機能キャッシュコア, SACSIS2008, pp.421-430(2008).

[2] 森 洋介, 森谷 章, 藤枝 直輝, 吉瀬 謙二: マルチコアにおけるオーバーヘッド解析を用いたキャッシュコアの最適化, 情報処理学会研究報告, ARC-181, pp.105-110(2008).

[3] Jouppi,N.P. :Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers, 17th ISCA,pp.364-373(1987).