2013 年 11 月 12 日 情報システム基盤学合同輪講発表資料

Memory-centric System Interconnect Design with Hybrid Memory Cubes

著者： Gwangsun Kim, John Kim, Jung Ho Ahn, Jaeha Kim
出典： *Proc. of Int'l conf. on Parallel architectures and compilation techniques, pp.145-156, 2013.*
発表者： 高性能コンピューティング学講座　本多・近藤研究室　1353014 佐々木沢

1　はじめに

CPU のマルチコア化・マルチスレッド化により、単位時間当たりの命令実行数が年々増加している。それに対して、メモリ帯域幅の向上は非常に緩やかであり、CPU がメモリアクセス待ちにより、高い性能を発揮できない問題がある。

複数の CPU を持つシステムでは一般的に、各 CPU がローカル DRAM を持つ一方で CPU 同士が接続され、他 CPU の DRAM へは当該 DRAM を持つ CPU を経由してアクセスする構成がとられている。これを CPU セントリックな接続方式と呼ぶ。この接続は、CPU 側と DRAM 側の接続が物理的に分割されており、バンド幅が固定して割り当てられるため、アプリケーションによっては合計バンド幅を有効活用できない。

本研究では、HMC を用いたメモリセントリックな接続方式を提案する。プロセッサの帯域幅を有効活用し、データ通信の効率を向上させることが目的である。

2　Hybrid Memory Cube(HMC)

Hybrid Memory Cube（HMC）は現在検討されている新しい DRAM 規格であり、メモリバンド幅不足を解消する次世代の DRAM アーキテクチャとして有力なものである [1]。HMC では、複数の DRAM チップが 3 次元積層され、配線長が短くなっている。また、ロジックチップも積層されており、メモリコントローラやルータの機能を搭載している。入出力にはシリアル伝送を用いることで、プロトタイプにおいて 320GB/s の高速なデータ通信を実現している。

3　Memory-centric Network

実行するアプリケーションによって、CPU-DRAM 間の経路と CPU 同士を接続した経路にかかる負荷は異なる。しかし、それぞれの経路の帯域幅は設計時に割り当てられた CPU ピンの本数により決まる。そのため、通信負荷に応じて CPU 間のバンド幅を増やしたり、別の経路を選択して負荷を分散するなどの柔軟性のある接続構成を利用できない。

HMC は、従来の DRAM アーキテクチャで用いられてきたパラレルデータ伝送方式による共有バス接続ではなく、CPU 同士の接続と同じポイントツーポイント接続で CPU-DRAM 間が接続される。そのため、CPU 同士の通信と DRAM アクセスのための IO ピンを統一することができ、帯域幅を接続先に応じて柔軟に割り振ることができる。また、各 HMC は目的のデータまでの経路を中継するルータとしても機能するため、HMC 同士を接続したメモリネットワークを構築することができこれにより主記憶の容量を拡張できる。これらの HMC の特性を利用し、HMC を組みあわせたメモリネットワークに各 CPU を接続し、CPU 同士の通信も当該ネットワークを介して行うメモリセントリックな接続方式を検討する。

なおこの際には、HMC 同士はそのルータを利用して多様なネットワークトポロジを構築することができる。図 1(a) は 16 個の HMC をメッシュ型に接続したメモリネットワークに、4 個の CPU を接続したものである。

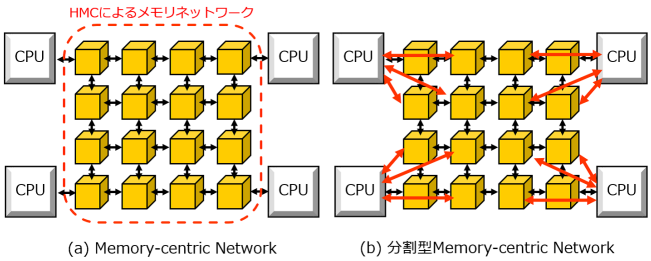

図 1: メモリセントリック接続

CPU は、HMC への通信経路をそのまま CPU 同士の通信にも利用できるため、今まで接続先をあらかじめ決められていたネットワークと比較して、CPU の帯域幅を活用できる。

4　提案手法

メモリセントリック接続の問題点として、CPU 間の通信経路が長くなることと、通信負荷の集中によるシステム全体のスループット低下がある。ここでは、それらの問題の解決を狙う 3 つの手法を提案する。

4.1　分割型のネットワーク設計

メモリセントリックな接続において、CPU は多量の HMC で構成されたメモリネットワークを介して通信する必要がある。そのため、CPU が直接通信できた CPU セントリックな接続と比較すると CPU 間の通信遅延が大きくなる。また、CPU からの接続経路が図 1(a) のように 1 つだと、その経路に対して負荷が集中しやすい。

そこで、CPU からメモリネットワークへの経路を分割

し、複数の HMC への経路を持つ分割型のネットワークを適用する。分割型ネットワークをメモリセントリックな接続に適用した例を図 1(b) に示す。各 CPU は接続経路を分割して、4 つの HMC への直接アクセスできる。

4.2 Adaptive Routing

CPU から複数の HMC モジュールに対して接続をし、CPU の帯域を複数の HMC に分割することにより、通信負荷の分散および通信経路の短縮による通信遅延の削減が期待できる。ここで、一般的には通信は最短経路を用いて行われる。そのため、一部の HMC にアクセスが集中すると特定の経路に負荷が集中することがある。分割型ネットワークにより生じた複数ある経路を活用する Adaptive Routing[2] を用いた、経路選択アルゴリズムを提案する。

最短経路と非最短経路を示した図を図 2 に示す。

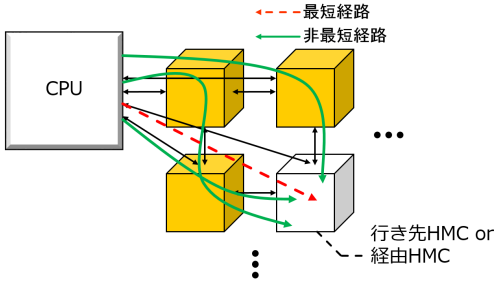

図 2: 分割型ネットワークによる最短経路と非最短経路

本アルゴリズムは、最短経路がパケット衝突やバッファあふれなどにより過負荷な時、複数ある非最短経路からランダムに 1 つ選択するものである。その後ここで、選択した経路と最短経路を比較して、より負荷の低い経路を選択して通信を行い、システム全体の負荷を均等化する。

4.3 Pass-through 機構

HMC では従来の DDR-SDRAM アーキテクチャとは異なりシリアル通信を用いる。シリアルリンクは実装コストも低く、高クロック化による転送速度向上が容易な一方で、転送時のシリアル化と受信時のデシリアライズ化を行う必要がある。HMC では信号送信時にシリアライズ回路、受信時にデシリアライズ回路を挟むため、通信ホップの度にその遅延時間が加算されてしまう。

Pass-through は HMC 内部に入力リンクと出力リンクを直接繋ぐ経路を増設することで、シリアル化による遅延を削減するものである。

Pass-through による遅延削減の様子を図 3 に示す。

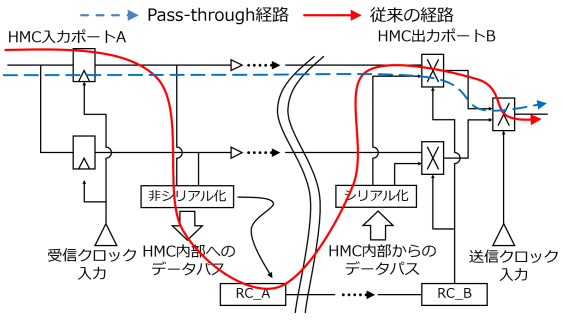

図 3: Pass-through による遅延削減

先行パケットにより後続パケットの行き先がこの HMC で無い場合は、内部の経路を増設した経路へと切り替え、必要の無いシリアライズ、デシリアライズを回避することができる。

5 性能評価

サイクル精度シミュレータの McSimA+、GEMS と Booksim を用いて評価を行った。異なる通信パターンを持つベンチマークを実行して、実行速度を測定した。ベンチマークは SPLASH-2 を用いた。

対象となる接続構成は、従来の CPU セントリックな接続とメモリセントリックな接続。さらに、提案手法をそれぞれ用いた場合について、プログラム実行速度とエネルギー効率を測定した。測定結果を図 4 に示す。

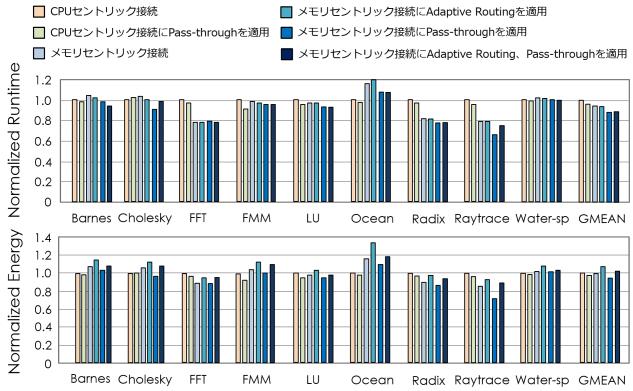

図 4: 実効速度およびエネルギー効率

それぞれの値は CPU セントリックな接続を用いた場合の測定値に正規化している。メモリセントリック接続に Pass-through 機構を用いた場合の実行速度は平均 12%向上し、エネルギー効率は平均 5.3%の向上を示した。このことから、メモリ中心の接続が CPU の帯域幅を活用できたことが分かる。Adaptive Routing に関しては、最短経路を選択しなかった場合の遅延などが表面化した一方で、負荷の分散による効果が発揮されなかった、ただし特定の経路にアクセスが集中することで、スループットが飽和するようなプログラムの場合に性能の向上が期待できる。

6 おわりに

本論文では、次世代 DRAM である HMC の持つ特性を活用し、CPU のバンド幅を柔軟に利用できるメモリ中心の接続を提案した。提案手法で増加した CPU 間の通信遅延を Pass-through 機構によって補うことで、実行速度およびエネルギー効率の双方で向上が見られた。

参考文献

[1] “Hybrid Memory Cube Specification 1.0,”[Online]. Available:http://www.hybridmemorycube.org/, Hybrid Memory Cube Consortium,2013.

[2] A. Singh,“Load-balanced routing in interconnection networks,”Ph.D.dissertation, Stanford University,2005.