メニーコアプロセッサにおけるコア間通信レイテンシ隠蔽手法の検討

高前田 伸也 †　森 洋介 †　吉瀬 謙二 †
東京工業大学 大学院情報理工学研究科 †

1　はじめに

メニーコアプロセッサにおいて，その性能を引き出すには十分な並列化を行うことが重要である．しかしながら，十分な並列化ができたとしても，コア間通信のレイテンシがボトルネックとなり，並列化の恩恵を受けることができない懸念がある．

本研究では，各コアにローカルメモリを持ち，明示的にコア間 DMA 通信を行うアーキテクチャを対象とし，ブロッキングな DMA 転送の実行中に，コア内処理をオーバラップ実行することでコア間通信のレイテンシ隠蔽を目指す．その為に，DMA 転送待ちのデータを保護するハードウェアを追加し，先行実行によるメモリアクセスと DMA 転送との間のデータハザードを防止する．

2　想定するアーキテクチャと提案手法

本手法は，各コアにローカルメモリを持ち，プログラムから明示的にコア間 DMA 通信を発行するアーキテクチャを想定している．本研究では，メニーコアプロセッサアーキテクチャM-Core[1] 上に提案手法を実装し，その評価を行う．

まずはじめに，M-Core アーキテクチャについて説明する．図 1 に M-Core アーキテクチャ全体の構成を示す．M-Core アーキテクチャは，小規模で均一なコアを多数並べる構成を採用し，システムソフトウェアとの協調によりチップの高性能化を目指す．

M-Core の各構成要素はノードと呼ばれる．各ノードにはノード識別子 (ノード ID) が割り当てられており，これを用いて通信を行う．各ノードは 2 次元メッシュネットワークで接続されており，DMA を用いて互いにデータ転送が可能である．タイル状に配置されたノードは，計算専用のコアを含む計算ノードである．

M-Core アーキテクチャの cycle-accurate なシミュレータとして SimMc[1] がある．評価にはこの SimMc を用いる．

次に，各計算ノードの仕様と提案手法について説明する．

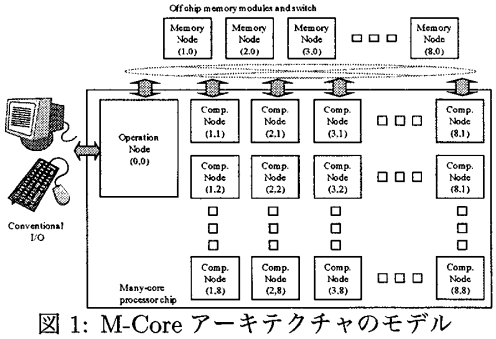

図 1: M-Core アーキテクチャのモデル

A Technique to Hide the Communication Latency between Cores on Many-core Processors
Shinya TAKAMAEDA†, Yosuke MORI† and Kenji KISE†
†Graduate School of Information Science and Engineering, Tokyo Institute of Technology

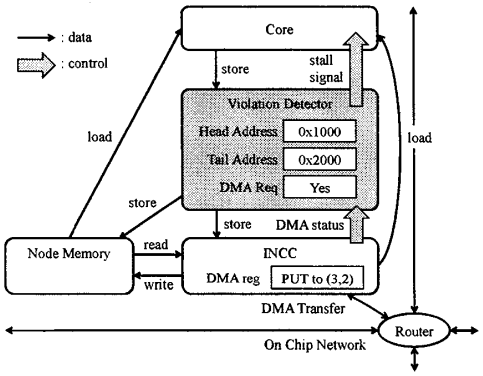

図 2: Violation Detector を追加した計算ノード

M-Core の計算ノードは，Core(演算処理ユニット)，ノードメモリ (各ノードが持つ小規模なメモリ)，INCC(Inter Node Communication Controller, DMA 転送の制御を行う)，オンチップルータとで構成される．さらに本手法では，Violation Detector というハードウェアを追加している．Violation Detector を追加した計算ノードの構成を図 2 に示す．各ノードは独立したメモリアドレス空間を持ち，他のノードあるいはメインメモリへのアクセスは INCC を介した DMA 転送により行う．Core は INCC に対して DMA 転送要求をストア命令を用いて発行し，INCC は DMA 転送要求に応じてノードメモリからデータを読み出し，適切なフォーマットでオンチップルータを介してネットワークに出力する．

通常の M-Core アーキテクチャにおいて，自ノードから他ノードへの DMA 転送には，Core が DMA 転送完了を待つ場合 (ブロッキング転送) と，転送完了を待たずに処理に戻る場合 (ノンブロッキング転送) の 2 種類が存在する．しかし本手法を適用した場合は，ブロッキング転送の場合でも，Core は完了を待たずに，先行的に転送完了後に行うはずの処理に進む．これにより DMA 転送と Core の処理をオーバラップさせ，通信レイテンシの隠蔽を目指す．しかし，DMA 転送の範囲と本来転送完了後に行われる処理でのメモリアドレスに重なりがある場合，Core のメモリアクセスと DMA 転送の間に Write after Read のデータハザードが発生する可能性がある．そこで本手法では，データハザードの発生を防ぐ為に，各計算ノードに DMA 転送待ちのローカルメモリのデータを保持し，Core のストア命令を監視するハードウェアである **Violation Detector** を Core と INCC，Core とノードメモリの間に追加する．

Violation Detector は，INCC から DMA 転送の情報を受け取り，それらを保持する以下の 3 つのフィールドから構成される．Head Address は自ノードから他ノードへのデータ転送が未完了なデータ範囲の先頭アドレス，Tail Address は末尾アドレスを示す．DMA Req は自ノードから他ノードへの DMA 転送要求があるかどうかを示

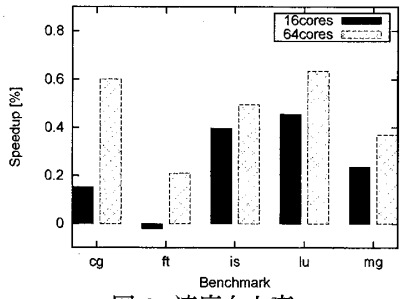

図 3: 速度向上率

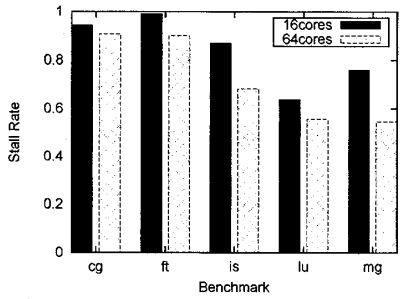

図 4: 全通信時間に対するストール時間の割合

している．Violation Detector は，Head Address と Tail Address で示される DMA 転送未完了範囲へのストア命令を検知し，データハザードを引き起こす場合には，Core をストールさせる．また，DMA 転送中に Core が新たな DMA 転送要求を発行する場合も Core をストールさせる．このように必要に応じて Core をストールすることで，コア内の Write after Read ハザードの発生を防ぐ．

3 評価・考察

SimMc 上の各計算ノードに提案手法を実装し，コア数を 16 コア，64 コアとして評価を行った．評価ベンチマークには，NAS Parallel Benchmark のうち cg, ft, is, lu, mg を用いた．M-Core の各パラメータは次のように設定した．各ノードにおける Core はシングルサイクルプロセッサとし，Core および INCC はノードメモリに対して 1 サイクルでアクセスできるものとする．また，オンチップルータはシングルサイクルのワームホールルータとし，Virtual Channel なし，入力バッファは 4flit 分としている．

図 3 は各ベンチマークにおける本手法を適用した場合の性能向上率を示している．これより，本手法適用による性能向上は最大で 0.6%程度に留まることがわかる．性能向上率が低い理由として，3 つの原因が考えられる．

1 つ目は，アプリケーション全体において通信が占める割合が小さいということが挙げられる．今回用いたベンチマークにおいて，全実行時間に対して通信が占める割合は実測で最大 5%程度である．その為，そもそも本手法適用により高速化されうるポイントが少ない．

2 つ目の原因は，複数の DMA 転送要求を発行する場合に，先行する DMA 転送実行中に後続の DMA 転送要求の発行が Violation Detector によりストールされることで，性能向上を妨げていることが挙げられる．提案手法では，DMA 転送中に新しい DMA 転送要求を Core が発行すると，その時点で Core をストールし，プログラムの正しい実行を保証している．その為，複数の DMA 転送を連続で発行する場合は，本手法による性能向上が難しい．この問題は，Core にストアバッファが実装されている場合に，アクセス違反を引き起こすストア命令だけをストールさせ，アクセス違反を引き起こさないストア命令は先行発行させることで回避できる可能性がある．解決方法の検討は今後の課題とする．

3 つ目の原因は，ソフトウェアレベルのコア間同期の利用である．コア間でデータの同期を行う場合，データを転送し，その後，あらかじめ宣言しておいた同期用の volatile 変数の値を DMA 転送によって変更することで実現される．同期待ちをしている側は同期用変数をポーリングすることによって同期完了を検知するため，Violation Detector による違反検知および，プログラムの先行実行ができない．これに対しては，同期対象となっているデータ範囲と同期用変数の監視をハードウェアがサポートし，Read after Write ハザードの検出を Violation Detector が行うことで，先行実行が可能となる見込みがある．関連研究して，スレッド間同期の HW によるサポート手法 Synchronization State Buffer[2] が提案されている．

一部ベンチマークで性能が低下している理由としては，ノンブロッキング転送に対しても Violation Detector が働き，Core をストールしていることが挙げられる．ノンブロッキング転送とブロッキング転送の違いはライブラリレベルでしか存在しない．その為，現状のアーキテクチャでは，Violation Detecotor がこれらの違いを知ることはできず，ノンブロッキング転送の場合は Violation Detecotor を停止する，という対処を取ることができない．この問題を解決するためには，両者の違いを明示的にハードウェアに伝達するようにアーキテクチャを変更する必要がある．

一方，ネットワークの通信時間に着目する．図 4 は各ベンチマークにおける本手法を適用した場合の総通信時間に対する Core のストール時間の割合を示している．最大 40%程度の通信レイテンシ隠蔽が達成できており，コア間通信レイテンシおよびプロセッサコアの構成によっては，大きな性能向上を達成できる可能性がある．

4 まとめ

本稿では，メニーコアプロセッサにおいて，コア間通信のレイテンシ隠蔽を目指し，ブロッキングな DMA 転送の実行中のコア内処理のオーバラップ実行をサポートするハードウェアである Violation Detector の追加を提案した．ソフトウェアシミュレータを用いて評価したところ，最大約 0.6%程度の性能向上に留まることがわかった．更なる性能向上には，コアのストアバッファのとの連携や，コア間同期をサポートするハードウェアの検討が必要である．

参考文献

1) Koh Uehara and et. al. A Study of an Infrastructure for Research and Development of Many-Core Processors. In *UPDAS held in conjunction with PDCAT'09*, pp. 414–419, 2009.

2) Weirong Zhu and et. al. Synchronization state buffer: supporting efficient fine-grain synchronization on many-core architectures. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pp. 35–45, New York, NY, USA, 2007. ACM.