様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２８　年　　６　月　　２　日現在

機関番号：　１２６０１
研究種目：　若手研究(B)
研究期間：　2013～2015
課題番号：　２５７３００２７

研究課題名（和文）次世代超低電力メニーコアにおける設計最適化のための性能評価フレームワーク

研究課題名（英文）Performance evaluation framework for design optimization of next generation low-power many core systems

研究代表者

中田　尚（Nakada, Takashi）

東京大学・情報理工学（系）研究科・助教

研究者番号：００４５２５２４

交付決定額（研究期間全体）：（直接経費）　　3,200,000円

研究成果の概要（和文）：メニーコアプロセッサの性能評価には共有キャッシュの性能予測技術が重要である。効率的な設計にはシミュレーション精度と速度のバランスを取ることが重要である。
この課題について検討を進めたが、高速化と高精度の両立は非常に困難な課題であった。競合が与える性能への影響は大きいため、安易な省略は精度の大幅な低下を招いてしまう。それに対応するために高精度な競合予測が必要となるが、その実現も非常に困難であり、結果として速度と精度を十分な領域で両立することは実現できなかった。今回の知見を元に新たなブレイクスルーを模索していく予定である。

研究成果の概要（英文）：Performance estimation techniques of shared cache are important for performance evaluation of many core processors. Tradeoff between evaluation speed and accuracy is a key.
To cope with this problem, I tried to develop fast simulator of shared cache. Simplification is a way to improve simulation speed but it also causes degradation of accuracy. I also tried to manage both speed and accuracy by speculative techniques. However, in this project, is was not possible to make compatible speed and accuracy with practical applications.

研究分野：　計算機アーキテクチャ

キーワード：　性能評価　メニーコアプロセッサ　共有キャッシュ

１．研究開始当初の背景

一般的なプロセッサでは 1 コアあたりの規模が大きくなるにつれ、性能は向上するがそれ以上に消費電力が増大することが知られている[1]。そのため、電力効率向上のためにはシンプルなコアを大量に接続したメニーコアが有望視されており、多くのメニーコアが登場してきている。多いものでは GPGPU のように 1000 コア以上を実装したシステムがすでに実用化されている。

メニーコアシステムの利用方法は、現在は科学技術計算のように 1 つのシステムを 1 つのアプリケーションで専有するものが中心であるが、今後メニーコアがさらに普及するとマルチタスク処理のように複数のアプリケーションを 1 つのシステムで同時に実行することが予想される。このような状況では、各アプリケーションに割り当てるコア数や、アプリケーションの組み合わせが多岐にわたり、全てのパターンについて網羅的に厳密な性能予測を行うことは設計空間の膨大さから非現実的である。そこで、本研究ではメニーコアを構成する各要素の特徴を利用することにより、高速かつ的確な性能予測を実現し、メニーコアの設計最適化問題を解決する。

高性能プロセッサ分野は、消費電力低減のために動作周波数追求から並列度追求へ大きく方向転換してきた。その結果、デスクトップ PC 向け CPU では今年度中には 50 コア程度のメニーコア環境が手軽に入手可能になると予想される。また、スマートホンにも複数のコアが搭載されているように、組込み機器も含めてほとんど例外なくメニーコアになっており、それぞれの分野における搭載コア数は着実に増加してきている。

その一方、共有キャッシュの競合、コア間ネットワークの遅延、分割損の増大など、性能向上を妨げるとともに、性能予測を困難にする要因が山積している。シングルコアの時代からシミュレーションによる性能予測が広く行われてきた。メニーコア環境においても、原理的には高精度なシミュレータの利用により厳密な性能予測を行うことは可能であるが、その実行時間が問題となる。数個のマルチコアのシミュレーションであってもその実行時間は実実行時間の 10,000 倍以上にもなる。コア数がさらに 100 倍になれば、少なくともこの実行時間は 100 倍になり、コア間の相互作用の増大によりさらに低速になることも十分に予測される。例えば 100 万倍低速なシミュレーションでは 1 週間で実行可能なシミュレーションはターゲット上ではおよそ 0.6 秒間のみとなり、大規模なシステムの性能予測にはとうてい十分とはいえない。このような問題を解決するためには、シミュレーションの高速化は必要不可欠である。

シミュレーションの高速化については多くの研究がされている。例えば、研究代表者らが提案した計算再利用[2] や時分割並列化[3] はシミュレーションの精度を維持したまま高速化を行う手法である。

また、SimPoint[4] では統計的処理により部分的なシミュレーションから全体の性能を予測する。これらの手法は精度を高く保ちつつ、高速化を目指す手法であるため、その高速化率は数十倍から 100 倍程度であり、メニーコアのシミュレーションにはさらなる高速化が必要である。特に初期設計の段階では、厳密な精度ではなくとも、大まかな絞り込みが迅速にできることが重要である。具体的には高速なシミュレーションにより最適な構成の候補となるいくつかのパターンを提示し、それらの中に正解が含まれていれば良いと考える。以上のことから、シミュレーション結果の絶対誤差よりも相対誤差を減らすことができれば、迅速かつ正確な設計空間の絞り込みが可能となる。

メニーコアシミュレーションの実行時間は大まかには、プロセッサコア自体のシミュレーション、共有キャッシュのシミュレーション、コア間ネットワークのシミュレーション、に大別される。プロセッサコアはシンプルなコアを利用するため、挙動の把握は容易であり全体のシミュレーション時間に占める割合は多くない。ネットワークシミュレーションについては、混雑が発生することはまれであるため初期段階では簡易的なシミュレーションで対応する。そして、本研究ではキャッシュシミュレーションを重点的に高速化する。研究代表者がこれまでに行なった種々のプロセッサに関する研究により、今後の超低消費電力高効率コンピューティングにおいてはメニーコアプロセッサが有望であるとの結論に至った。

その一方で、多くのメニーコアシステムが提案されているが、ソフトウェア開発者に対するサポートは十分とはいえず、どのようなメニーコアシステムを利用することが最適であるかを判断することは非常に困難である。本研究課題によりその問題を解決し、メニーコアシステムの有効性を高める。

２．研究の目的

・高速共有キャッシュシミュレータ

提案フレームワークにとって最も重要となる、キャッシュシミュレーションの高速に実行し、「メニーコア評価フレームワーク」に渡す。高速性と正確性は重要であるが、最終目標である設計空間の絞り込みにとって重要な相対誤差の最小化が最も重要な点である。

・メニーコア評価フレームワーク

指示された探索範囲内の各構成について、「高速共有キャッシュシミュレータ」の結果を用いて性能を予測する。各種パラメータに対する性能の変化を提示することにより、設計空間の絞り込みを可能にする。メニーコア利用者は必要に応じて厳密なシミュレーシ

ョンを行うことにより、最適な構成を決定する。もし、初回選択の構成では性能が不十分であったとしても、設計空間中の性能の変化はわかっているので、どのパラメータを改善したら良いかを判断することも容易である。

最終目標としては、詳細シミュレーションの 1000 倍の速度を目指す。すなわち、1 回の詳細シミュレーションと同じ時間で 1000 パターンの組み合わせをシミュレート可能なフレームワークを実現する。

システムを構成する各モジュールの特性を最大限に利用することにより、適切なシミュレーション手法を組み合わせることにより、初期検討に特化したメニーコア評価フレームワークの実現を目指す。

これにより、次世代の超低消費電力メニーコアプロセッサの可能性を多くのユーザが活用できることとなり、極めて大きな意義があると考える。

[1] Fred Pollack: “New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies,” Int’l Symp. on Microarchitecture (MICRO), Haifa Israel (1999)

[2] 中田尚, 中島浩: “高速マイクロプロセッサシミュレータ BurstScalar の設計と実装”, 情報処理学会論文誌コンピューティングシステム, Vol. 45, No. SIG6(ACS6), pp. 54-65 (2004)

[3] 高崎透, 中田尚, 津邑公暁, 中島浩: “時間軸分割並列化による高性能マイクロプロセッサシミュレーション”,情報処理学会論文誌コンピューティングシステム, Vol. 46, No. SIG12(ACS11), pp. 84-97 (2005)

[4] T. Sherwood et al.: “Automatically Characterizing Large Scale Program Behavior,” 10th Int’l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 45–57 (2002)

3．研究の方法

本研究で提案する性能評価フレームワーク全体の処理の流れを図 2 に示す。

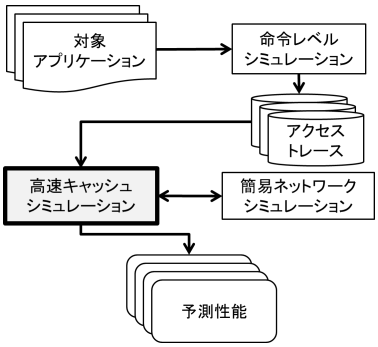

図 2：処理の流れ

メニーコアシミュレーションの実行時間は大まかには、プロセッサコア自体のシミュレーション、共有キャッシュのシミュレーション、コア間ネットワークのシミュレーション、に大別される。

プロセッサコア自体のシミュレーションについては、メニーコアに採用されているシンプルなコアは、性能予測が容易であることを利用する。具体的には実行時間は(全実行命令数）×（1 命令実行に必要なクロック数）+（キャッシュアクセス待ち時間）で求めることができる。すなわち、ある 1 回の実行の結果が得られれば、論理的な実行パターンが変化しない限り「全実行命令数」は変化しないため、「キャッシュアクセス待ち時間」の差分さえ求めることができれば、全体の実行時間を求めることができる。本研究ではプロセッサ自体のパラメータが変化しないパターン、具体的には入力パラメータと割り当てコア数が同一であるパターンについて、命令レベルシミュレーションを用いてアクセストレースを保存し再利用することにより、コア自体のシミュレーション回数を必要最小限とする。コア間ネットワークのシミュレーションについては、多くのネットワークは利用率が上昇すると徐々に性能が低下していき、ある点を超えると急激な性能低下が発生し輻輳状態に陥る。したがって、輻輳状態を回避するためには、余裕を持ったネットワーク設計とすることが重要であるので、そのような設計がされることを前提とする。本研究ではネットワーク上でのパケットの衝突は発生せず、全ての通信は理想的に行われることを仮定した、簡易ネットワークシミュレーションを用いる。

共有キャッシュについては、ネットワークの輻輳に対応する現象として、競合と呼ばれる現象がある。単一のアプリケーションであれば注意深く設計することで競合を回避することは可能であるが、複数のアプリケーション間の競合を予測することは困難である。そこでキャッシュの性能予測技術が重要になる。

共有キャッシュでは物理的に複数のキャッシュから構成されており、それらの間の一貫性（Coherence）を適切に管理することが必要である。この管理方針を決めるのがコヒーレンスプロトコルである。

実際のコヒーレントキャッシュでは、複数のコアからの要求が衝突することがある。例えば、複数のコアが同一のアドレスに対して同時に無効化要求を出すと、どちらかの要求のみが成功し他方は失敗となる。このような挙動を正確にシミュレートするためには厳密な時刻管理が必要であり、共有キャッシュのシミュレーションを複雑にしている要因である。しかし、このような衝突が頻発しないのであれば性能に与える影響は小さいと考えられる。

そこで、本研究では複数のリクエストの衝突は扱わず、個々のアクセスリクエストを独立にシミュレートする。つまり、全てのリク

エストを逐次的に処理することで、高速なシミュレーションを実現する。

共有メモリのシミュレーションに関する研究は古くから行われており、それらの知見を利用することも重要である。たとえば、共有メモリ環境においては大部分のキャッシュアクセスが 1 次キャッシュにヒットすることを利用し、確実にヒットすることが解析的に保証出来るアクセスを取り除く。フィルタキャッシュを配置し、1 次キャッシュでミスをする可能性があるアクセスのみを共有メモリシミュレーションを行うことで高速化する手法がある[5]。これは共有メモリを対象としているが、本研究で扱う共有キャッシュに対しても図 3 のようにキャッシュシミュレーションの前段に挿入することにより応用可能であると考える。

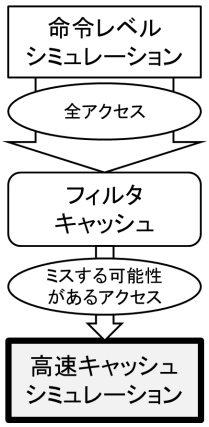

図 3 フィルタキャッシュ

加えて、投機的な手法についても採用を検討する。具体的には並列プログラムの実行中において真に共有しているメモリ領域を参照するタイミングは限定されていると考えられる。そこで、予測や学習によって競合が発生しない領域については共有キャッシュのシミュレーションを簡略化することにより、さらなる高速化を目指す。予測が外れた場合については、一定間隔のチェックポイントとロールバックを組み合わせることにより対応可能である。

キャッシュシミュレーションの高速化に加えて、フレームワーク全体の実行速度を十分なものにするためには、それ以外の部分の高速化が必要になることも予想される。

命令レベルシミュレーションについては、すでにバイナリ変換のような高速化手法が提案されているため、そのような技術の利用も必要に応じて検討する。また、命令トレースの取得方法についても最適化を検討する。ネットワークシミュレーションについては、さらに簡略化し「定数サイクルで通信が完了」というような仮定の導入を検討し、誤差と実行速度のトレードオフを調査する。

以上のような改良を取り込みつつ、フレームワークの完成を目指す。具体的には与えられたアプリケーション群に対して、設計空間における各種パラメータが性能に与える影響を一覧にするとともに、目標性能を達成可能な構成を複数個提示する。その後、詳細なシミュレーションを行うことにより、正確な性能を確認する。もし、性能が過小または過大であった場合には、設計空間中で隣接する構成の相対性能を参考に、微調整を行うことで対応できる。

[5] Haruyuki Matsuo, Shigeru Imafuku, Kazuhiko Ohno, and Hiroshi Nakashima. “Shaman: A Distributed Simulator for Shared Memory Multiprocessors,” In Proc. 10th IEEE/ACM Intl. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 347-355, (2002)

４．研究成果

メニーコアプロセッサの性能評価においては各コアの性能に加えて共有キャッシュの挙動を正しく解析することが重要である。特にアクセスの競合が性能に大きな影響を与えることが知られている。単一のアプリケーションであれば注意深く設計することで競合を回避することは可能であるが、複数のアプリケーション間の競合を予測することは困難である。そこでキャッシュの性能予測技術が重要になる。上記の課題に取り組むために、まずは実際のシミュレーション環境を構築し基本的なデータを取得した。

次に、高速なシミュレーションを実現するために、複数のリクエストの衝突は扱わず、個々のアクセスリクエストを独立にシミュレートすることを検討した。つまり、全てのリクエストを逐次的に処理することを検討した。衝突の確率が一定以下であれば、そのような近似を行っても精度は保たれ、高速なシミュレーションが可能となる。すなわち精度と速度のバランスを取ることが重要である。加えて、投機的な手法についても採用を検討した。具体的には並列プログラムの実行中において真に共有しているメモリ領域を参照するタイミングは限定されていると考えられる。そこで、予測や学習によって競合が発生しない領域については共有キャッシュのシミュレーションを簡略化することにより、さらなる高速化を目指した。

上記の方針について検討を進めたが、高速化と高精度の両立は非常に困難な課題であった。競合が与える性能への影響は大きいため、安易な省略は精度の大幅な低下を招いてしまう。それに対応するために高精度な競合予測が必要となるが、その実現も非常に困難であり、結果として速度と精度を十分な領域で両立することは実現できなかった。今回の知見を元に新たなブレイクスルーを模索していく予定である。

５．主な発表論文等

〔雑誌論文〕（計０件）
〔学会発表〕（計０件）

６．研究組織

(1)研究代表者
中田　尚（NAKADA TAKASHI）
東京大学・大学院情報理工学系研究科・助教
研究者番号：00452524