審 査 の 結 果 の 要 旨

氏　　名　　小川 周吾

本論文では高性能かつスケーラブルなマルチコアプロセッサを実現するために、各コアに共有キャッシュを動的に配分することでコア間の競合ミスを削減する方式であるHistory-Free Cache Allocation (HFCA)を提案している。多くのマルチコアプロセッサが持つ共有キャッシュでは、コア間の競合ミスの発生による性能低下が問題である。また競合ミスの頻度はコア数に比例して増加するため、プロセッサのコア数に比例した性能の実現は困難である。よって共有キャッシュの競合ミス削減は、コア数に比例した性能が得られる高性能かつスケーラブルなマルチコアプロセッサを実現するために重要な課題である。

提案されているHFCAは従来の多くのキャッシュパーティショニングとは異なり、キャッシュを各コアの占有する領域と各コアが共有する領域に分割する。HFCAはキャッシュ上の共有領域における競合を検出して、各コアの占有するキャッシュ領域の大きさを決定し、キャッシュヒット数の計測に用いる大量のハードウェアを不要としている。HFCAにより、IPC (Instruction Per Cycle)が、2コアで平均13.5%，4コアで平均17.2%の向上効果を確認している。さらに、HFCAは他の既存提案手法であるUCPに比べて、IPCが2コアで平均1.4%，4コアで平均1.5%低下し、かつ、追加ハードウェア量は、32-wayのキャッシュを4コアで共有する場合でUCPの0.1%未満に削減している。

本論文は7章から構成される。第1章では、本研究の動機、既存手法の問題点、研究の貢献が記載されている。第2章では、共有キャッシュに対する既存競合回避方式として、オペレーティングシステムによるPage Coloring、ハードウェアによるキャッシュパーティショニングおよびDead Block Eliminationに関して紹介し、既存手法の問題点を明確にしている。HFCAは、著者の既存研究の成果であるThread-Set Selection方式ならびにPartial Trial-Based Runtime Partitioning方式の知見の上に考案されている。そこで本論文では、最初にこの2つの方式について述べている。

第3章では、プロセッサ上で並列実行するスレッドを選択することで競合ミスを回避する手法としてThread-Set Selection (TSS)を提案している。TSSは実行した各スレッドの組に対してキャッシュミス頻度を記録し、キャッシュミス頻度が低くなるように各コアで次に実行するスレッドを決定する。またTSSは複数のプロセッサに対して、ミス頻度が高くなるスレッドを異なるプロセッサで実行することで競合ミスを回避する。TSSをLinuxカーネルに実装して、マルチスレッドの行列乗算プログラム及びSPEC CPU2000を用いて評価を行っている。行列乗算プログラムによる評価では平均で5%の性能向上が確認できているが、SPEC CPU2000による評価ではTSSによる性能向上は確認されなかった。TSSの競合ミス削減効果は限定的であり、競合を直接的に回避するキャッシュ管理方式の必要性が判明した。

第4章では、ハードウェアによる競合ミスの直接的な回避を行うために、キャッシュパーティショニング方式の一手法としてPartial Trial-Based Runtime Partitioning (PTRP) を提案している。PTRPは共有キャッシュの一部を用いてパーティション変更の試行を行う。試行の前後でキャッシュミスが減少した場合に、試行したパーティションをキャッシュ全体に適用してミスを回避する。この試行を繰り返してキャッシュミスを最小化するように各コアのパーティションを決定する。PTRPをSPEC CINT2000のワークロードに対するミス削減効果およびハードウェア量で評価している。評価の結果PTRPは従来のキャッシュパーティショニング方式と比べて2コアで平均1.5%、4コアで平均3%のミス増加に対して、追加ハードウェア量は従来方式の1%未満であることを確認している。 PTRPは試行の繰り返しに伴うソフトウェアオーバーヘッド、キャッシ

ュの増加に応じてミスが減少しないスレッドに対して最適なパーティションが得られない、という2つの問題点がある。これら問題を解決する手法としてHistory-Free Cache Allocation (HFCA)を第5章で提案している。

HFCAは共有LLCを各コアのPrivate partition (PP)とShared partition (SP)に分割して、各コアのPPにキャッシュヒットしたラインが格納されるようにway数の制御を行う。この制御によりヒットしないdead blockをSPに分離することで、パーティショニングにおけるdead blockの影響を排除する。各コアのPPに割り当てるway数の変更は各ラインのキャッシュヒット時に行う。そのためキャッシュ各部のアクセス履歴の保存、ヒット数の計測は不要であり、結果として少ないハードウェア量で実装可能となっている。HFCAをSPEC CPU2006を組み合わせたワークロードに対するIPCの向上効果で評価した結果、2コアで平均13.5%，4コアで平均17.2%の向上効果が確認されている。またHFCAは従来方式と比べてIPCが2コアで平均1.4%，4コアで平均1.5%の低下である一方、追加ハードウェア量は4コアが共有する4MB、32-wayのLLCに対して従来方式の1%未満であることが確認されている。

第6章は既存手法との違いをまとめ、第7章で本論文の結論を述べている。

本論文の成果は、高性能かつスケーラブルなマルチコアプロセッサ実現の課題となる共有キャッシュの競合回避方式に対して、従来方式が持つ問題を解決したHistory-Free Cache Allocation （HFCA）を提案したことにある。HFCAは共有キャッシュをコア毎のprivate partitionとshared partitionに分けてdead blockを排除するキャッシュパーティショニング方式である。少量のハードウェア追加で実装が可能であり、従来方式と同等の競合回避効果を持つことをベンチマークプログラムの評価で示した。本研究の成果により、高性能かつスケーラブルなマルチコアプロセッサの発展に顕著な貢献をしたといえる。

よって本論文は博士（情報理工学）の学位請求論文として合格と認められる。