2008 年 11 月 17 日 情報システム基盤学専攻合同輪講発表資料

Scheduling Threads for Constructive Cache Sharing on CMPs

著者： S.Chen, P.B.Gibbons, M.Kozuch, V.Liaskovitis, A.Ailamaki, G.E.Blelloch, B.Falsafi, L.Fix, N.Hardavellas, T.C.Mowry, C.Wilkerson

出典： *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 105-115, June 2007.*

発表者： 近藤研究室　0853009 久米正人

1 概要

近年ではチップマルチコアプロセッサ (CMP) が主流になってきている．共有型のキャッシュが搭載された CMP も珍しくない．このような CMP を活用することで高性能期待できるが，この並列性を効率よく利用するためには，以下の 2 つの観点に目を向けなければならない．

・搭載されたコア全てに処理を行わせること
・キャッシュの競合を減らすこと

搭載されたコア全てに処理が割り当てられることは，重要である．このようなスケジューリングを行う方式に Work Stealing (WS) がある．

また一方で，CMP はキャッシュの状態によって大きく影響を受ける．結果として，キャッシュ内データが再利用される，つまりキャッシュの競合をなくすことは，より高い性能を CMP から引き出すことにつながる．このような“建設的な”キャッシュ共有を行うようにスケジューリングを行うことも重要である．このようなスケジューリング方式に Parallel Depth First (PDF) がある．

これら 2 種類のスケジューリング方式について，実際のプログラムを利用しての比較，評価はなされていなかった．そこで本論文では，PDF，WS のアルゴリズムによるスケジューラについて，実際のベンチマークを利用し，ベンチマーク実行時間，キャッシュミス回数などのパフォーマンスの評価を行う．

2 スケジューリングアルゴリズム

2.1 DAG の導入

複数スレッドで動いているプログラムのタスク，スレッドとそれらの間の実行順序関係を表すために，有向非巡回グラフ (DAG) を使う．DAG の頂点をタスク，タスクの集合 (図 1 中の T_1,T_2,T_3) をスレッド，有向辺を実行順序の関係とする．

2.2 Work Stealing

Work Stealing (WS) は，コア全てに処理がなるべく割り当てられるよう考慮されたスケジューリングアルゴリズムである [1]．WS を実現するために，各コアに両端キューが存在することとする．

WS の方針は以下のようになる．

・コアが実行すべきスレッドは，キューの中に入っている
・コアは，キューの先頭からスレッドを取り出して実行する
・実行できるスレッドがないならば，他のコアのキューの末尾からスレッドを取る (*stealing*)

これらの方針により，処理が割り当てられていないコアの数が少なくなる．

WS は，並列性が高い場合には *stealing* はほとんど起こらず，よいアルゴリズムである．しかし *stealing* が起こるとワーキングセットが分散してしまう傾向があるため，建設的なキャッシュ共有を行うことができない．

2.3 Parallel Depth First

Parallel Depth First (PDF) は，建設的なキャッシュ共有を行うことを目的としたスケジューリング方式である [2]．PDF を実現するためには，1DF スケジュールを行わなければならない．

1DF スケジュールの方針は以下のようになる．

・タスクをスケジュールしたらそのタスクのすぐ次のタスクをスケジュール
・スケジュールできるタスクがなくなったら根に戻る

1DF とは，シングルコアでのスケジューリング方式であり，これを利用する PDF は逐次実行に則したスケジューリング方式とも言うことができる．最適化されたプログラムは，シングルコア上でよいキャッシュのパフォーマンスを引き出すように設計されていると考えられるため，PDF による並列実行でも良いキャッシュのパフォーマンスを得られる．

PDF の方針としては，実行できるタスクを 1DF でのスケジューリングの順にコア数ずつ選択していく．図 1 に 2 コアでの例を示す．タスクの円の外側の数字が 1DF，内側の数字が PDF でのスケジューリングである．

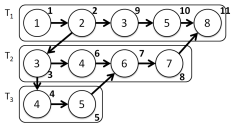

図 1: 1DF, PDF の結果

3 評価手順

種々の CMP の設計空間とベンチマークを用いて評価を行う．

CMP の設計空間に関しては，現実の CMP を想定しダイサイズは $240mm^2$ とし，ITRS 2005 Edition[3] に従いコア数，製造プロセスなどを決定した．製造プロセスを微細化させるに従いコア数を増やしたもの，プロセスを 45nm で固定しコア数を変化させたものの 2 種類の設計空間で評価を行った．それぞれの設定の一部を表 2，表 3 に示す．その他の仮定は表 1 に示す．

LU, Hash Join[4], Mergesort[5] の 3 種類を評価に利用するベンチマークとした．

4 評価

4.1 製造プロセスを変化させたときの比較

表 2 の設定で評価を行ったときの結果は図 2 のようになった．LU に関しては，PDF は WS に比べ L2 キャッシュミスを削減できているが，実行時間の速度向上という観点では PDF，WS 間に差はほとんどなかった．よって，PDF によって L2 キャッシュミスを削減できていてもパフォーマンスには影響がないということが分かった．これはキャッシュミスを減らせたといっても 1000 命令中で 0.05 回ほどであるためである．

表 1: 評価に使用した CMP 等の仮定

プロセッサコア	In-order scalar
L1 キャッシュ	Private, 64KB, 128-byte line, 4-way, 1-cycle hit latency
L2 キャッシュ	Shared, 128-byte line
主記憶	latency: 300; service rate: 30 (cycles)

表 2: 製造プロセスを変化させたときの設定

コア数	1	4	8	16	32
プロセス (nm)	90	90	65	45	32
L2 キャッシュサイズ (MB)	10	4	8	20	40
連想度	20	16	16	20	20
L2 ヒット時間 (cycles)	15	11	13	19	23

表 3: プロセス 45nm 固定での設定

コア数	1	4	8	16	26
L2 キャッシュサイズ (MB)	48	40	32	20	1
連想度	24	20	16	20	16
L2 ヒット時間 (cycles)	25	23	21	19	7

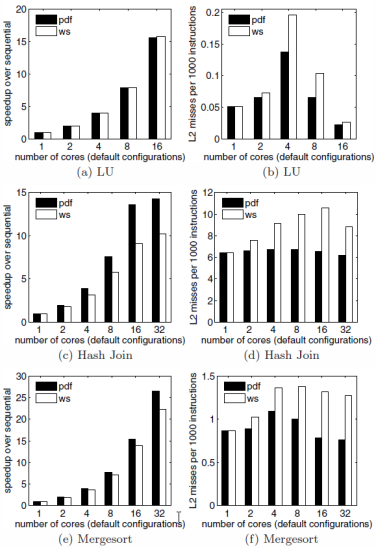

図 2: 製造プロセスを変化させたときの PDF, WS の評価

Hash Join に関しては，PDF は WS に比べ 13.2%～38.5%L2 キャッシュミスを削減できた．また，速度向上に関しては 1 コア時に対して WS は最大で 10.19 倍に比べ PDF は 14.28 倍と，PDF の方がよりよい数値を示している．Mergesort に関しては，PDF は WS に比べ 13.8%～40.6%L2 キャッシュミスを削減できた．これは Hash Join のときと似たような傾向になっている．速度向上に関しても，1 コア時に対し WS は最大で 22.30 倍に比べ PDF は 26.44 倍と，PDF の方がよりよい数値を示している．

以上より，大きなワーキングセットをもつアプリケーションに関しては，PDF は WS よりも大幅によいパフォーマンスを実現すると言える．WS，PDF 間にパフォーマンスの差がないため，これより先の評価では LU を省略する．

4.2 プロセス固定での比較

表 3 の設定で評価を行い，PDF と WS のパフォーマンス比較を行う．結果は図 3 のようになった．PDF は WS よりも良いパフォーマンスを実現すると言える．

4.3 ワーキングセットに関する比較

図 4 は，タスクのワーキングセットの容量を変えたときの実行時間の結果を表している．ベンチマークは Mergesort で，32 コアで動かしたときである．8MB のときには，PDF

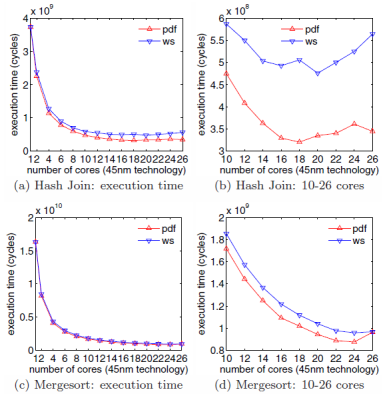

図 3: プロセス固定 (45nm) での PDF, WS の評価

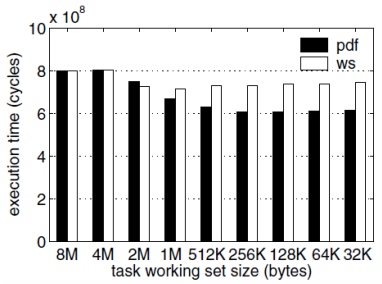

図 4: ワーキングセット容量に関する PDF, WS の評価

と WS の間に実行時間の差はほとんどなかったが，32KB とすると WS での実行時間は PDF での実行時間の 1.17 倍になっている．このことより，タスクのワーキングセットの大きさを制御し，タスクを適切な粒度にすることにより，PDF はより良い性能を発揮するということが分かる．

5 まとめ

本論文では，PDF と WS の 2 種類のスケジューラに関してベンチマークを用いて評価を行い，多くの CMP 構成において PDF が WS よりも良いパフォーマンスを発揮することを示した．

また，より効率的にキャッシュを利用することにより，PDF はプロセッサ設計者にとって設計空間を広げることにもつながる．

さらに，タスク粒度が CMP のキャッシュのパフォーマンスに大きな影響を及ぼすことを示した．

参考文献

[1] R. D. Blumofe and C. E. Leiserson. Scheduling Multithreaded Computations by Work Stealing. *JACM*, 46(5), pp. 720-748, 1999.

[2] G. E. Blelloch, P. B. Gibbons, and Y. Matias. Provably Efficient Scheduling for Languages with Fine-Grained Parallelism. *JACM*, 46(2), pp. 281-321, 1999.

[3] Semiconductor Industry Association. The International Technology Roadmap for Semiconductors (ITRS) 2005 Edition, 2005.

[4] S. Chen, A. Ailamaki, P. B. Gibbons, and T. C. Mowry. Inspector Joins. In *VLDB*, pp. 817-828, 2005.

[5] M. W. Weissman. Libpmsort. http://freshmeat.net/projects/libpmsort.