2009 年 4 月 2 日 FS 専攻 修士論文中間発表会 発表資料

メモリバス競合に着目した CMP 向け実行フェーズスケジューリング手法の研究

高性能コンピューティング学講座 近藤研究室
0853009 久米正人
主任指導教員：近藤正章

1 概要

現在主流となっているプロセッサのアーキテクチャの 1 つに，複数のプロセッサコアを 1 つのチップ上に搭載したチップマルチプロセッサ (CMP) と呼ばれるものがある．CMP に搭載された複数のコアで並列処理あるいは並行処理を行うことによって，クロック周波数の向上に頼らず高い演算性能を達成することができる．

演算性能の向上を目指し，CMP に搭載されるコア数は年々増加傾向にある．ただし，キャッシュやバスなどの資源は複数のコアで共有することが一般的である．そのため，多くのコアを搭載した CMP においては，これら共有資源の競合が起こりやすいといえる．しかし，共有資源の競合が生じると，チップ全体の性能が低下し，期待される演算性能が十分に得られない．このため，CMP の持つ演算性能を最大限に引き出すためには，共有資源の競合をできる限り回避することが重要である．

このため，共有資源の競合に関する研究は従来より行われてきた．文献 [1] は，資源の占有率を考慮したプロセススケジューリング手法に関する研究である．しかし，さらに細粒度の視点から見ると，一般的には共有資源の占有率は 1 つのプロセス内でも変化する．このことを考慮すると，より細粒度な実行フェーズ単位でスケジューリングを行うことで，より効果的な実行が行えると考えられる．

そこで本研究では，このような共有資源の競合に着目しつつ，プロセスを実行フェーズで分割した細粒度でのスケジューリング手法を提案する．特に，メモリウォール問題が深刻化していることを考慮し，共有資源の 1 つとしてメモリバスの競合に焦点を当てて研究を行う．

2 提案手法

本研究では，プログラムを単位時間ごとに分割した実行フェーズ毎のメモリバンド幅占有率に着目する．実行フェーズの中から，占有率がメモリバスのバンド幅を超えないような組み合わせを選択してスケジューリングを行う．

本提案手法の具体的な手順は，まずは単位時間ごとに占有率が 100%になる実行フェーズの組み合わせを選択する．100%を超えてしまい，割り当てることができなくなった場合は，何も行わないアイドルの実行フェーズを選択する．このようにすることで，競合によるバンド幅の低下を防ぐことができる．

以下に例を示す．デュアルコア環境で 2 つのプログラムを実行する場合を考える．

プログラム 1，2 のバンド幅占有率の時間変化を図 1 に示す．ここで，プログラム 1 は A，B，C，D，E，プログラム 2 は F，G，H，I，J の部分に単位時間ごとの実行フェーズに分割する．実行フェーズ A，D，F，G，I，J は占有率が高くなっており，その他は低くなっている．この場合のスケジューリング結果を図 2 に示す．

これらのような 2 つのプログラムを単純に同時実行させると，占有率が高い部分が同時期に実行され，占有率が 100%を超えてしまうため各コアの性能が低下し，効率的でない．

それに対し本提案手法にて，占有率に着目してスケジューリングを行うと，占有率の高い部分が同時期に実行されることがなくなるために，提案手法の方がより高いスループットを得られると考えられる．

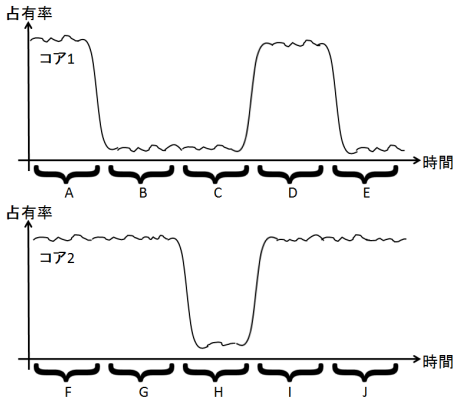

図 1: 各プログラム（コア）のバンド幅占有率の時間変化

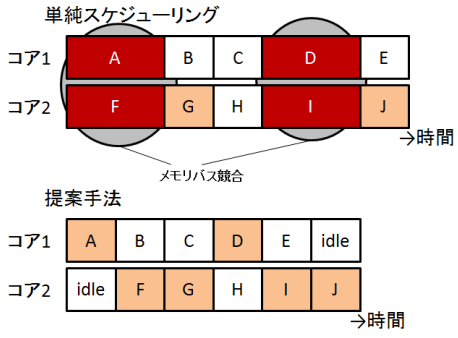

図 2: スケジューリング結果

3　研究の進捗状況

本提案手法の妥当性を確認するために，従来のスケジューリング手法による性能と本提案手法のスケジューリングを擬似的に再現した場合の性能の差を調べた．

具体的には，キャッシュミス時にメモリアクセスが発生することから，キャッシュヒット部とキャッシュミス部を交互に実行するプログラムを作成し，実行完了までの時間を計測した．そのプログラムをプログラム a として図 3 に示す．

また，ミス部とヒット部の順番を逆に，すなわちプログラム a の 4〜7 行目と 8〜11 行目を交換したものをプログラム b として用意する．計測に利用した CPU は，AMD Phenom X4 9850 である．ヒット部，ミス部を実行するのにかかる時間を同じになるようにした．

program_a.c

```
 1  char a = 0, *U;
 2  U = (char *)malloc(268435456 * sizeof(char));
 3  for(i = 0; i < 268435456; i++) U[i] = 1;
 4  for(k = 0; k < 10; k++){
 5      for(i = 0; i < 275; i++)
 6          for(j = 0; j < 268435456; j += 128)
 7              a += U[j];
 8      for(m = 0; m < 8; m++)
 9          for(i = 0; i < 1073741824; i++)
10              for(j = 0; j < 1073741824; j++)
11                  a += U[0];
12  }
13  free(U);
```

図 3: プログラム a（細部省略）

従来のスケジューリング手法として，プログラム a を 2 個それぞれのコアで同時に実行したものをケース A，提案するスケジューリングとしてプログラム a とプログラム b を同時に実行したものをケース B として評価した．この計測結果を図 4 に示す．ケース B は，ケース A と結果的に同じ処理を行ったにもかかわらず，約 1.2 倍高速に実行できた．また，ケース B の実行時間は，プログラム a を単独で動かしたとき（ケース C）と比較してほとんど同じであった．

この結果は，メモリバンド幅に着目した本提案手法が有効である可能性を示している．

図 4: 各実行環境での計測結果（10 回平均）

4　今後の方針

4.1　メモリバンド幅占有率の予測

この提案手法を実現するにあたって，それぞれのタスクのメモリバンド幅占有率がわからなければいけない．よって占有率を予測する必要があると考えられる．そのため，予測方法，予測のための指標を見つけることが今後の課題である．

4.2　スケジューリング機構の開発，評価

占有率の予測ができた後，実際にスケジューリングアルゴリズムとそれを実現するスケジューラの開発にとりかかる．現段階では，親プロセスを用意し，その子プロセスとして実際のプログラムを動かし親プロセスがシグナルを送ることにより子プロセスを制御することを考えている．

5　関連研究

メモリバスのバンド幅や共有キャッシュなど，共有資源の競合に関する研究は従来より行われている．文献 [2] では，共有資源の競合を緩和するために，各プロセスの実行スピードを制御するトラクションコントロール実行という手法がとられている．

また，文献 [3] では，メモリボトルネックを解消するためにプリフェッチのみを行うヘルパースレッドを用いる手法により CMP の性能低下を抑えている．

文献 [4] のように，バンド幅を有効利用するようなスレッド数を動的に制御する手法も存在する．

6　まとめ

CMP では，多くのコアが搭載されているために共有資源の競合が起こりやすく，チップ全体の十分な性能が期待できない局面が存在する．そこで，メモリウォール問題が深刻化していることから共有資源の 1 つとしてメモリバスに着目し，メモリアクセスが同時期に集中しないような CMP 向けのスケジューリング手法を提案した．

また，従来のスケジューリング手法による性能と本提案手法を擬似的に再現したスケジューリング手法による性能の差を比較し，本提案手法が有用である可能性があることを示した．

参考文献

[1] Wei Zhao, Krithivasan Ramamritham, and John A. Stankovic: Scheduling Tasks with Resource Requirements in Hard Real-Time Systems, *IEEE Transactions on Software Engineering*, Vol. SE-13, Issue. 5, pp. 564-577 (May 1987).

[2] 近藤正章, 佐々木広, 中村宏: トラクションコントロール実行: CMP 向けプロセス実行制御方式の提案, 情報処理学会論文誌: コンピューティングシステム, Vol. 1, No. 2, pp. 111-123 (Aug. 2008).

[3] 今里賢一, 福本尚人, 井上弘士, 村上和彰: 演算/メモリ性能バランスを考慮した CMP 向けヘルパースレッド実行方式の提案と評価, 電子情報通信学会技術研究報告, Vol. 108, No. 28, pp. 75-80 (2008).

[4] M. Aater Suleman, Moinuddin K. Qureshi, and Yale N. Patt: Feedback-Driven Threading: Power-Efficient and High-Performance Execution of Multi-threaded Workloads on CMPs, *Proc. of the 13 th Int'l Conf. on ASPLOS*, pp. 277-286 (2008).