4H−4　階層メモリマルチプロセッサシステムのための データプレローディング及びポストストアアルゴリズム

藤原 和典，白鳥 健介*，鈴木 真**，笠原 博徳
早稲田大学理工学部電気工学科　*NTT データ通信　**三菱重工

1. はじめに

ローカルメモリを持つ複数のプロセッサを共有メモリに結合したアーキテクチャの階層メモリ・マルチプロセッサシステム上で科学技術計算の並列処理を効率良く行なうためには、共有メモリとローカルメモリ間のデータ転送のオーバーヘッドが大きなネックとなる。この問題を解決するための手法としては、データプレローディング及びデータポストストアの導入、及びCPU の実行を共有メモリとローカルメモリ間のデータ転送とオーバーラップさせ、オーバヘッドを軽減させる方法が考えられる。本稿では、データ転送量が大きく、かつプログラム自体の並列性の高い航空流体シミュレーションコードをサンプルとして、データ・プレロード、ポストストアを可能にするデータ転送を考慮したスケジューリングアルゴリズムを提案すると共にシミュレーションによる性能評価結果について述べる。

2. 階層メモリ・マルチプロセッサシステムのモデル

今回対象とするマルチプロセッサシステムのモデルは、図1のように、ローカルメモリを持つプロセッサを共有メモリへ複数パスにより接続した階層メモリマルチプロセッサシステムである。PE 間通信は共有メモリ (CM) を通して行ない、PE-CM 間通信は各 PE の持つ MAC (Memory Access Controller) によって CPU のプログラム実行と同時にオーバーラップして転送することが可能である。LM は各 PE のプログラム及びデータをストアするローカルメモリであり、PE 内の CPU と MAC(DMA) から同時アクセスが可能である。また MAC は DMA を含んだメモリ及び BUS アクセスを管理をするコントローラである。

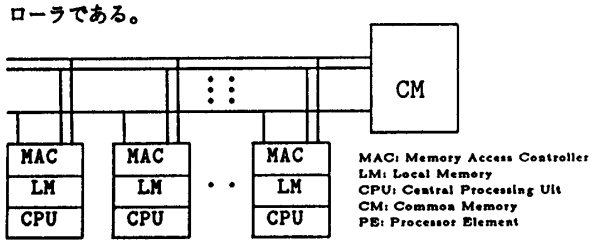

図1. 対象システムモデル

3. プレロード、ポストストアを考慮したスケジューリング

将来各 PE 上で実行される演算のために必要なデータをその実行前に前もって LM 上にロードしておくことをデータのプレロード、実行が終ったあとにすぐデータを転送せずそのデータが他 PE で使用されるまでのパスが空いている時間にストアすることをポストストアと呼ぶ。この際実行とデータ転送をより多くオーバーラップさせるためには、データとプログラムの分割、配置及びデータ転送タイミングの最適化が重要となる。

3.1. プレロード、ポストストアの概念

図2のプログラムを PE 2 台で実行する簡単な例を用いてデータプレロード、ポストストア、実行とデータ転送のオーバーラップについて説明する。図2の DOALL ブロック3つ (MT1,MT2,MT3) からなる Fortran プログラムを図3のように PE に分割し、スタティックスケジューリングを用いて処理したとすると、一般的には図4(a) のように処理される。図中 S は CM へのストア、L は CM からのロードを表わす。この図から単純にすべてのデータを CM 上に割り当てると CM に対するデータ転送オーバーヘッドが非常に大きいことがわかる。

このとき、各配列を分割し、PE1,PE2 のローカルメモリ上に置き、PE 間で転送の必要なデータのみを共有メモリに書き、プレロード、ポストストア、実行とデータ転送のオーバーラップを行なう様子を表現したのが図4(b) である。この場合、MT1,MT2,MT3 間や MT1',MT2',MT3' 間ではローカルメモリを通じてデータの受け渡しを行なうため、ストア、ロードが不要になる。また、MT3 の I=100 の場合、A(J,101) へのアクセスがあり、MT3' の I=101 の場合、A(J,100) へのアクセスがあるため、PE 間のデータ転送が発生するが、図中 PS(PostStore),PL(PreLoad) のようにプログラム実行 E とオーバーラップされる。

```
      COMMON A(100,200),B(100,200)

|       DO 1 I=1,200
MT1     DO 1 J=1,100
|     1   A(J,I)=A(J,I)/MAXA

|       DO 2 I=1,200
MT2     DO 2 J=1,100
|     2   B(J,I)=0

|       DO 3 I=2,199
|       DO 3 J = 1,100
MT3   3   B(J,I)=B(J,I)
|           +0.5*(A(J,I+1)
|           -A(J,I-1))
```

図2. サンプルプログラム

```
          PE 1                                PE 2
|       DO 1 I=1,100                |       DO 1' I=101,200
MT1       DO 1 J=1,100              MT1'      DO 1' J=1,100
|       1   A(J,I)=A(J,I)/MAXA      |       1'  A(J,I)=A(J,I)/MAXA

|       DO 2 I=1,100                |       DO 2' I=101,200
MT2       DO 2 J=1,100              MT2'      DO 2' J=1,100
|     2   B(J,I)=0                  |     2'  B(J,I)=0

|       DO 3 I=2,100                |       DO 3' I=101,199
|       DO 3 J=1,100                |       DO 3' J=1,100
MT3   3   B(J,I)=B(J,I)             MT3'  3'  B(J,I)=B(J,I)
|           +0.5*(A(J,I+1)          |           +0.5*(A(J,I+1)
|           -A(J,I-1))              |           -A(J,I-1))
```

図3. 図1のプログラムを二分割したもの

Data-preloading and Data-poststoring Algorithum for Hierachial Memory Multiprocessor System
Kazunori FUJIWARA , Kensuke SHIRATORI* , Makoto SUZUKI** , Hironori KASAHARA
WASEDA University, *NTT DATA COMMUNICATIONS SYSTEMS CORPORATION, **Mitsubishi Heavy Industries, LTD.

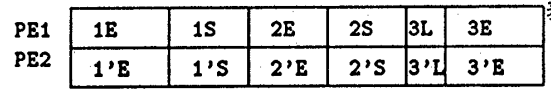

図4(a) 従来のスケジューリング結果

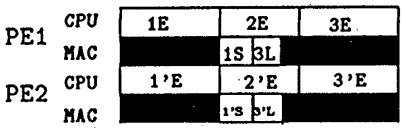

図4(b) プレロード、ポストストアによるスケジューリング結果

次のスケジューリングアルゴリズムをシミュレーションによって評価した。

データプレロード、ポストストア、データ転送と実行のオーバーラップなどのような機能はコンパイル時のスタティックスケジューリングと組み合わせて使用するのが望ましいため、ここでは以下のようなスケジューリングアルゴリズムにこれらの機能を付加し、シミュレーションにより評価した。

3.2. スタティックスケジューリングアルゴリズム

(a) ランダムスケジューリング
実行待ちタスクの中からランダムにタスクを選びだし、空きPEに割り当てる。

(b) CP/MISF [1]
実行待ちタスクの中から出力ノードまでの距離の長い順、同じ距離の場合は直接後続タスク数の多い順に割り当てる。データ転送を考慮しない場合の準最適解を得る方法である。

(c) CP/DT/MISF [2]
実行待ちタスクの中から出力ノードまでの距離の長い順にデータ転送量が最小となる組合せで空きPEに割り当てる。出力ノードまでの距離とデータ転送量が同じ場合は直接後続タスク数の多いタスクを優先する。

(d) DT/CP/MISF
実行待ちタスクと空きPEの組合せのなかでデータ転送量が最少になるものを順に割り当てる。データ転送量が同じになった場合、CP/MISFに準じた優先を行なう。

データ転送量は、すでにローカルメモリに存在してロードする必要がない場合やプレロード、ポストストアによって実行とオーバーラップできる部分を差し引いて計算する。

3.3. プレロード、ポストストアの挿入法

プレロードは共有メモリにデータが転送された時刻からデータを参照される時刻の間にバスとPEバスの空いている時間を利用して行なわれる。図4(b)で示すとMT3のロード(3L)は、MT1'のストアの終ったあとからMT3の実行の始まるまでの時間に行なわれている。プレロードできなかった残りは割り当て後にロードを行なう。また、ポストストアはデータが必要になったときに共有メモリにデータが存在しない場合とローカルメモリがあふれそうになったときに行なわれ、データが定義された時刻からデータを参照されるまでの時刻のあいだにバスとPEバスが空いていればPEの実行とオーバーラップして行なわれる。図4(b)で示すと、MT1のストア(1S)はMT1の実行後でMT3の実行(ロード)前に行なわれている。オーバーラップしてストアできなかった場合、ロードを待たせてストアを行なう。
スケジューラ上でのプレロード、ポストストアの実現であるが、割り当てがプレロードする時点において決まっていないので、過去のPE、BUSの状態をすべて保存しておき、そのデータを使って必要な時に過去に遡って操作を行なう。プレロードの場合は、必要になったときに過去に遡ってデータをロードしたことにするということになる。データが共有メモリに準備されてから現在までの時刻のうち、BUSとPEのデータ転送パスが空いていて、さらにローカルメモリにそれだけの空き容量があるときプレロードを挿入できる。ポストストアであるが、これは、他のプロセッサでのロード要求が発生してストアが必要になってからストアするということで実現している。この場合、ストアが終るまでロードができないというオーバーヘッドが生じることがある。ここで、さらにそのストアを実行とオーバーラップさせる。すなわち、過去に遡り、BUSとPEのデータ転送パスが空いてるときにストアを挿入するわけである。

表1 PE2台BUS2本　　単位[unit time]

	random	cp/misf	cp/dt/misf	dt/cp
	388064	336170	329390	328339
PL	362689	327852	335218	334573
PS	223537	175734	165757	163587
PS,PL	179097	175893	173699	160468
1PE実行のみ	320884			

表2 PE4台BUS2本　　単位[unit time]

	random	cp/misf	cp/dt/misf	dt/cp
	291280	273029	251515	214092
PL	274274	252695	250330	229183
PS	143600	139329	118710	92833
PS,PL	122384	111865	96249	82720
1PE実行のみ	320884			

4. 性能評価

性能評価に使用したプログラムは並列性が非常に高い航空流体シミュレーションコードであり、すべてのDOブロックがDOALL可能、IF分岐無しである。シミュレーションによる各アプリケーションの性能評価の結果を表1,2に示す。表1はPE台数2台、表2はPE台数4台でBUS2本とした場合のランダム、CP/MISF法、CP/DT/MISF法とDT/CP/MISF法をプレロード有り無し、ポストストア有り無しの場合について表している。

PLはプレロードを行なった場合、PSはポストストアを行なった場合を示している。また、ポストストア無しのものはすべてのデータを共有メモリにストアする。

この例では一台での実行に320[kut]かかる。処理がPE 2台では最高160[kut]、PE 4台で82[kut]で処理されるというようにデータ転送オーバーヘッドが最小化され、効率の良い実行が行なわれることがわかる。

5. 結び

本稿ではデータプレロード、ポストストアを効果的に用いるスケジューリング手法を提案した。そしてその有効性は航空流体シミュレーション用の実プログラムの並列化シミュレーションにより確認された。

尚、本研究の一部は(株)富士通との共同研究に基づくものである。

6. 参考文献

[1] H.Kasahara and S.Narita :"Practical multiprocessor scheduling algorithm for efficient parallel processing", IEEE Trans. Comput.,c-33,11,pp.1023-1029(Nov,1983)

[2] H.Kasahara, H.Honda and S.Narita :"Parallel processing of near fine grain tasks using static scheduling on OSCAR ",IEEE Supercomouting'90 Nov 12-16,1990,