連続・離散時間制御システム シミュレーションの並列処理

非会員 山 本 裕 治 (早稲田大)

非会員 鳥 居 宏 行 (N H K)

非会員 前 川 仁 孝 (早稲田大)

非会員 田 村 光 雄 (松下電器)

正 員 笠 原 博 徳 (早稲田大)

正 員 成 田 誠之助 (早稲田大)

Parallel Processing of Continuous/Discrete-Time Control Systems Simulation

Yuji Yamamoto, Non-member (Waseda University), Hiroyuki Torii, Non-member (Japan Broadcasting Corporation), Yoshitaka Maekawa, Non-member (Waseda University), Mitsuo Tamura, Non-member (Matsushita Electric Industrial), Hironori Kasahara, Member, Seinosuke Narita, Member (Waseda University)

This paper proposes a parallel processing scheme for continuous- and discrete-time control systems simulation on multiprocessor system. The scheme allows ordinary users to easily simulate control systems represented by S domain and Z domain transfer functions, state space equations and block diagrams on a shared memory multiprocessor system in real time. A control system simulator based on the proposed scheme has been implemented on a multiprocessor system named OSCAR. The simulator automatically generates a simulation program from a block diagram input on X-Window, compiles the program into parallel machine codes, executes them on a multiprocessor and outputs the results graphically.

キーワード：並列処理，マルチプロセッサ，制御系シミュレーション，常微分方程式，差分方程式

1. ま え が き

制御システムの設計・解析⁽¹⁾を効率良く行うためには、被制御対象の動特性あるいはシステム全体の動特性のシミュレーションが不可欠である。このようなシミュレーションは、最近の半導体の集積技術の進歩に伴う計算機の高速化とダウンサイジング化のため、ワークステーション上で行われるのが一般的となりつつある。このようなワークステーション用 CAD システムはマン・マシンインタフェースがよくユーザフレンドリーであるが、原子炉のダイナミクスシミュレーションや航空機の動特性解析といった大規模なシステムのシミュレーションを高速に行う必要がある場合、リ

アルタイム計算が困難であるという問題がある。ここでのリアルタイムとは、制御におけるサンプリング時間を時間制約として、その時間制約と演算精度を満足させるための積分刻み幅を選び、その積分刻み幅より短い時間で処理を行うことを意味する。このように、ワークステーション単体の処理能力ではリアルタイム計算が不可能な場合、CAD システムがシミュレーションプログラムを FORTRAN などの高級言語で生成し、そのプログラムをネットワークなどを介して、汎用大型計算機あるいはミニスーパーコンピュータなどに送り実行するというアプローチがとられている。しかし、これらのアプローチは汎用大型機、ミニスーパーコンピュータなどを占有して使用しないとリアルタ

イム計算が困難であるため、コストなどの点で問題が生じる。

このような問題を解決し、ユーザフレンドリネスを保ちながら高速なシミュレーションを可能とするために、安価なマイクロプロセッサを用いたマルチマイクロプロセッサシステム上での並列処理⁽¹¹⁾が注目されている。しかし、従来提案されているマルチプロセッサシステム上での並列処理手法⁽⁵⁾⁻⁽⁷⁾では、任意数のプロセッサを用いて並列処理することができない、連続時間系のシミュレーションしか扱えないなどの問題があったが、本論文で提案する並列処理手法では、任意数のプロセッサを用いて連続系 (S 領域)、離散系 (Z 領域) システムの両方が混在したシステムの並列シミュレーションを可能とする。本手法を用いて試作したマルチプロセッサ制御システムシミュレータでは、ユーザが S 領域伝達関数、Z 領域伝達関数、状態空間方程式表示などを含むブロックダイアグラムを X-Window 上のグラフィックエディタを用いて入力すると、自動的にシミュレーションプログラムを生成し、それを並列化してマルチプロセッサ上で実行後、

シミュレーション結果を出力する。

2. マルチプロセッサシステム OSCAR のアーキテクチャ

ここでは、今回制御システムシミュレータの試作に使用したマルチプロセッサシステム OSCAR のアーキテクチャについて述べる。

OSCAR (Optimally Scheduled Advanced Multiprocessor) のアーキテクチャは、図 1 に示すように、最大 16 台のプロセッサエレメント (PE) と 1 台のコントロール & I/O プロセッサ (CIOP) とが、三つの集中型共通メモリ (CM) と各 PE 上の分散共有メモリに、3 本のバスを介して接続された共有メモリ型のマルチプロセッサシステムである。各 PE は、5 MFLOPS の処理能力をもつ 32 ビット RISC プロセッサであり、

- (i) 64 個の 32 ビット汎用レジスタ (REG)
- (ii) 32 ビットの整数演算ユニット (IPU)
- (iii) 32 ビットの浮動小数点演算ユニット (FPU)
- (iv) データメモリー

図 1 OSCAR のアーキテクチャ
Fig. 1. Architecture of OSCAR.

- (v) プログラムメモリー
 - (vi) 分散共有メモリー(デュアルポートメモリー : DPM)
 - (vii) スタックメモリー(SM)
 - (viii) DMA コントローラ
- から構成されている。

OSCAR は、コンパイル時のスタティックスケジューリング結果を使用し、細粒度タスクの並列処理を効果的に行うように設計されている。例えば、OSCAR の RISC プロセッサは浮動小数点の加算、減算、乗算を含むすべての命令を 1 クロックで実行する。また他の浮動小数点演算も固定クロック数で実行できる。実際のマルチプロセッサシステム上でスタティックスケジューリングを行う場合、タスクの実行時間の正確な推定は一般的には困難であるが、OSCAR では 1 命令の処理を 1 クロックあるいは固定クロック数で実行するので、その推定を確実に行うことができる。

また OSCAR には、

- (i) DPM を使用した 1 PE 対 1 PE の直接データ転送
- (ii) DPM を使用した 1 PE 対全 PE のブロードキャストデータ転送
- (iii) CM を使用した 1 PE 対複数 PE の間接データ転送

の 3 種類のデータ転送モードが用意されている。DPM を使った PE 間直接データ転送は、送信側 PE が受信側 PE 上の DPM 上に共有データを 1 回書き込むだけで、データ転送を行うことができる。1 PE から他の PE へのブロードキャストデータ転送は、ある PE のデータを他のすべての PE の DPM にデータを書き込む方式をとることにより実現されている。これに対し、通常のマルチプロセッサで使用される CM を使った間接データ転送では、ある PE が CM へデータを書き込み、他の PE が CM からデータを読み込む必要があるため、2 回のデータ転送が必要となり、DPM を使用したデータ転送と比べオーバーヘッドが大きい。以上のことから、データ転送のオーバーヘッドを減らすためには、スタティックスケジューリングの結果を用いて 3 種類のデータ転送を使い分けることが重要となる。

3. 連続・離散時間制御システムシミュレーションの並列処理

〈3・1〉 制御システムシミュレーションの並列処理の問題点 制御システムの動特性は、一般的にエクスプリシットな連立一階常微分方程式でモデル化され

る。これらの方程式の求解は、通常オイラー、トラペゾイダル、三次・四次のアダムスバッシュフォース法および四次のルンゲ・クッタ法などといった数値積分法を用いて行われる。しかし、これらの積分法では、1 積分ステップ中の計算が算術代入文集合の処理となり、また積分ステップ間にもデータ依存 (loop carried dependence) が生じてしまうために、現在のマルチプロセッサシステムで使用されている do all や do across⁽³⁾ といったループレベルの並列処理手法では、効率良く並列処理を行うことができない。また、システム中に離散時間の伝達関数 (Z 領域) が含まれる場合、サンプリング周期ごとに離散時間システムの計算を要するため、連続時間システムのように単純に同じ積分計算の繰返しが行われるとは限らない。

本論文では、このような積分計算をマルチプロセッサシステム上で並列処理するために、以下の並列処理コンパイレーション手法を提案する。

〈3・2〉 並列化コンパイレーション手法

(1) 制御対象の入力 まずユーザは、ワークステーション上のグラフィックエディタを用いて、シミュレーション対象となる制御システムのブロック線図入力を行う(図 2)。

この構成図入力では、各種演算要素(積分、加減乗除および各種非線系要素を含む)から、S 領域および Z 領域の伝達関数や状態方程式で記述されるものまでを対象としている。

ここで、入力されたブロック線図をもとに、

- (i) 各ブロックの演算要素の種別
- (ii) ブロック間の接続関係
- (iii) 各ブロックに必要なパラメータ
- (iv) 積分・単位遅延要素の初期値
- (v) 連続・離散時間システムの区別

などを記述した接続テーブルを生成する。

(2) タスク粒度の選択 前述のように、連続時間システム単独の場合のシミュレーションにおいては、 $dx/dt=f(x, t)$ の連立一次微分方程式を数値積分法を用いて求解する計算となる。数値積分を行う場合、各数値積分ステップごとに導関数の計算、積分法の計算および外部との入出力を繰り返すことになるので、並列処理の対象は各積分ステップ内の計算となり、この計算のタスク分割および割当てについてのみ

図 2 ブロック線図の入力例
Fig. 2. An example of block diagram.

考えればよい。このとき問題となるタスクグラニュラリティーの選択では

- (i) 方程式レベルの分割 (Coarse Granularity)
- (ii) 演算要素レベルの分割 (Fine Granularity)
- (iii) その中間レベルの分割 (Medium Granularity)

の三つが候補として考えられる。これらのタスクグラニュラリティーの選択については、問題の並列性、使用するマルチプロセッサシステムの処理性能、データ転送能力、同期機構、スケジューリングの能力を考慮し、そのシステムごとに適切に選ばなければならない。本並列処理手法では、使用するマルチプロセッサシステム OSCAR 上での各種性能、コンパイラのスケジューリング機能およびグラニュラリティーの調整機能（タスク融合機能）を考慮して、1 演算要素（加算、積分など）を 1 タスクとした比較の細かいグラニュラリティー（Near Fine Granularity）によるタスク生成を行っている。

(3) タスクの生成 データ依存解析の結果得られたテーブルをもとに、1 演算要素を 1 タスクとし、コンパイラは並列中間言語で記述された並列化プログラムを自動生成する。伝達関数や状態空間モデルのように、複数の演算要素からなるブロックは、演算レベルのブロックに自動的に展開される（図 3）。

また、制御対象プラントが通常のアナログモデルで、それをデジタルコントローラで制御するといった連続・離散時間の混在するシステム全体をシミュレ

ーションする場合、ディジタルシステム内の S 領域伝達関数は通常連続時間システムの複数積分ステップに 1 度の割合で計算され、それ以外のステップではデータホールドにより保持された計算結果が連続時間システムの入力となる。CSMP (Continuous System Modeling Program) などのシミュレーション言語では、FORTRAN 言語の条件文を用いてこの部分の計算制御を行っている。しかし、条件分岐の方向が規則的に定まっていることを考慮し、またシミュレーションの積分ステップ間の並列性を得るため、本手法では連続時間システム内の積分ステップをアンローリング⁽²⁾し、離散時間システムのサンプリング周期を 1 シミュレーションステップとしている。連続時間システム部分の計算のアンローリングにより、各積分ステップごとに行われる離散時間システムの計算の有無に関する条件判断を除去することができ、また連続・離散時間システム間での並列性や連続システムの積分ステップ間の並列性を抽出することも可能となる。

(4) ブロック間のデータ依存解析およびタスクグラフ生成 入力されたブロック線図の接続テーブルに記述された接続関係をもとに、タスクの演算順序を決定する。制御システムのダイナミクスシミュレーションは前述のように、通常連立 1 階微分方程式、または差分方程式の求解となる。従って、各積分ステップの計算は、先行制約（実行順序関係の制約）は演算要素レベルで考えると、連続時間システムにおける積分要素、または離散時間システムにおける単位遅延要素を含んだサイクルの組合せとなる。ここで、積分または遅延要素には初期値が与えられているため、各積分ステップの計算はこれらの初期値から始まり導関数の評価、積分法固有の計算、差分方程式の評価を行う計算となる。このような計算のデータ依存解析を行うと、タスク間データ依存グラフは、積分、遅延要素の直接の後続ブロックを入口とし、積分、遅延要素を出口ノードとする無サイクル有向グラフとなる。ただし、コンパイラが生成する中間語で記述されたシミュレーションプログラム内では、上述のデータ依存解析で除去した積分、遅延要素とその後続ブロック間のデータ依存関係を保持する必要がある。従って、データ依存解析終了後は、積分要素、遅延要素の出力値を接続テーブルにおける積分ブロックおよび遅延ブロックの後続ブロック（データ依存グラフの入口ノード）の入力値として登録する。

また、タスク間データ依存グラフ生成後、コンパイラはグラフに各タスクの OSCAR 上での推定実行時間、タスク間のデータ転送時間などを付加したタスク

図 3 ブロックの展開例

Fig. 3. Expansion of blocks.

図 4 タスクグラフの例
Fig. 4. An example of task graph.

図 5 タスク融合パターン
Fig. 5. Task fusion pattern.

グラフ⁽¹¹⁾を生成する。

(5) タスク融合 演算要素レベルのタスクは、浮動小数点演算命令レベルの細粒度タスクなので、使用するマルチプロセッサのデータ転送および同期性能を考慮し、ヒューリスティックなタスク融合手法を用いて、適切なグレインまでタスクを融合する。図 5 に、このヒューリスティックタスク融合手法が自動的に適用されるパターンを示す。

タスクグラフ中に図 5 のような部分パターンをもつタスクのうち、その最小処理時間がシーケンシャル処理時間より大きくなる場合、それらのタスクを一つのタスクに融合する。例えば図 6 に示すように、最小並列処理時間 $\min\{(t_1+t_3+t_4), (t_2+t_{23}+t_4)\}$ がシーケンシャル処理時間 $(t_1+t_2+t_4)$ よりも大きい場合にはこれらのタスクを一つに融合する。

これらの手法をタスクグラフに繰返し適用することにより、タスク間のデータ転送時間に対するタスク処理時間の割合に応じて複数のタスクが一つのより粗粒度のタスクに融合される。このタスク融合を行うことにより、融合されたタスクは同一プロセッサに割り

図 6 タスク融合の適用例
Fig. 6. Task fusion.

当てられるので、これらのタスク間のデータ転送をなくすとともに、各タスク間で PE 上のレジスタを有効に利用することができる。

(6) タスクスケジューリング 制御システムのシミュレーションは、タスクグレインが比較的小さく、タスク間のデータ転送量も大きいので、データ転送を考慮したヒューリスティックなスケジューリングアルゴリズムである CP/DT/MISF 法 (Critical Path/Data Transfer/Most Immediate Successors First)⁽⁹⁾ を用いて各 PE にタスクを割り当てる。そのアルゴリズムは次のようである。

(i) 各タスクのレベルを求める。レベルとは、タスクグラフ上での、タスク N から出口ノードへの最も長いパス長である。

(ii) プロセッサ間のデータ転送を考慮したリストスケジューリングを行う。

(a) あるスケジューリング時刻において、実行待ちタスクとアイドルプロセッサを見つける。そのうちで最もレベルの高いタスクを、アイドルプロセッサの一つへ割り当てる際に生じるデータ転送時間を、考えられるすべての組合せに対して推定する。

(b) データ転送時間が最小になるようなタスクのプロセッサへの割合を一つ選び出し、その時点までの部分スケジューリングにこの割当てを加える。このときデータ転送時間が最小となるタスク割当てが複数存在するときには、直接の後続タスク数が最も多いタスクを優先する。

アイドルプロセッサがなくなるまで、手順 (b) を繰り返す。

(c) データ転送時間を考慮し、次のスケジューリング時刻を算出する。これは、少なくとも一つのプロ

セッサが、既に割り当てられたタスクの実行を終了してアイドル状態になる最も早い時刻である。そして実行待ちタスクとアイドルプロセッサを見つける。

実行待ちタスクがなくなれば終了し、それ以外の場合は手順(a)へ戻る。

(7) マシンコード生成 スケジューリング結果により得られた情報、すなわち各PEで実行されるタスク番号、タスクの実行順序、他のPEに割り当てられたタスクからのデータ転送の待ち時間の推定値、同期の有無などをもとに、各PE用のマシンコードを生成する。各PE用マシンコードは、各タスクに対応するマシンコード、タスク間同期のためのコード、積分ステップ間の同期コードからなる。この際、待ち時間を考慮したデータ転送の最適化、冗長なフラグセットチェックの削除を行っている。

また本手法では、データ転送時に共有メモリーへの書き込みおよび読み出しという2回の共有メモリーアクセスを要する間接データ転送は用いず、分散共有メモリーへの1回のストアのみでデータ転送が行える直接データ転送(1PE対1PE直接転送と1PE対全PEブロードキャスト転送)を中心としてマシンコード生成を行う。また、1PE対1PEの直接データ転送と1PE対全PEのブロードキャストデータ転送では、32bitデータの転送に要するクロック数が4と等しいので、現状では受信プロセッサが2台以上の場合にはブロードキャスト転送、受信プロセッサが1台の場合、直接データ転送を利用するように並列マシンコードを生成する。

4. 性能評価

ここでは、提案する連続・離散時間制御システムシミュレーションの並列処理手法の性能を、OSCAR上で評価した結果について述べる。

図7、図8は、OSCAR上でシミュレーションプログラムを実行したときの、使用プロセッサ台数に対する並列処理時間をプロットしたものである。グラフ中の実線はタスク融合を行った場合、破線はタスク融合を行わない場合の並列処理時間を示している。

図7(a)はグラマンF14ベンチマーク制御問題のブロック線図である。この制御系は複数のS領域伝達関数ブロックをもつもので、シミュレータの性能評価に用いられる標準的な問題である。(b)図から、タスク融合によりプロセッサ間データ転送が削減され、実行時間が短縮されることがわかる。タスク融合をしない場合の処理時間は、PE1台で0.372s、4台で0.191s、8台で0.125sであり、タスク融合をした場

(a) ブロック線図

(b) 並列処理時間

図7 グラマンF14ベンチマーク制御問題とその並列処理時間

Fig. 7. The Grumman F14 benchmark control problem and its parallel processing time.

合の処理時間は、PE1台では0.348sであったのが、4台で0.185s、8台では0.115sと、PE1台の場合と比較して、約1/3に短縮されている。

この問題のタスク数は607個で、1タスク当たりの平均実行時間は9.4クロックであった。1回のデータ転送にかかる時間が9クロックであるから、タスクの実行時間とデータ転送時間との比は約1である。

図8(a)は製粉機制御シミュレーションのブロック線図である。これは、連続および離散時間システムを含む制御システムの問題である。タスク融合を行わない場合の処理時間は、PE1台で0.432s、3台で

(a) ブロック線図

(b) 並列処理時間

図 8 製粉機制御システムとその並列処理時間
Fig. 8. The feed drive position control system and its parallel processing time.

0.377 s, 5 台で 0.210 s であり, タスク融合をした場合の処理時間は, PE 1 台では 0.392 s, 3 台で 0.241 s, 5 台で 0.173 s と短縮されている。

この問題のタスク数は 252 個で, タスク融合前の 1 タスク当たりの平均実行時間は 10.2 クロックであり, タスク融合後の 1 タスク当たりの平均実行時間は 18.0 クロックであった。タスク融合により 1 タスク当たりの平均実行時間が大きくなっているため, 融合による実行時間短縮の効果が大きく表れている。

従来の並列処理手法ではループレベルの並列性しか抽出できないため, ループ間, ループ内の並列性がほとんどないこのような制御モデルでは, PE 数を増やしてもほとんど並列処理効果が得られなかった。本手法では, 比較的細かい粒度のタスクを生成し, そのタスク間の並列性を抽出する事により, 実マルチプロセッサ上で並列処理時間の短縮を達成している。

5. む す び

本論文では, 連続・離散時間制御システムシミュレーションのための自動並列処理手法を提案し, その手法をマルチプロセッサシステム OSCAR 上でインプリメントすることにより, 実用性の検証を行った。その結果, 従来並列処理が困難であった, 多項式伝達関数や状態空間モデル, 連続・離散時間が混在するために数値積分ステップ内, または積分ステップ間に複雑なデータ依存が存在する制御系シミュレーションの並列処理を, 実マルチプロセッサシステム上で低オーバーヘッドで行えることが確認できた。また本並列処理システムは, ユーザが UNIX ワークステーション上でブロック線図をグラフィック入力すれば, 専用目的並列化コンパイラが自動的にシミュレーションプログラムおよびその並列化オブジェクトコードを生成し, OSCAR 上での並列実行後, 結果をワークステーション上でグラフィック表示するユーザフレンドリーなシステムであり, 従来の並列処理システムのようにユーザが並列言語を用いて並列化プログラムを作成するという困難な作業を除去することを可能としている。

(平成 4 年 4 月 12 日受付)

文 献

- (1) C. L. Phillips & H. T. Nagle, Jr.: Digital Control System Analysis and Design (1984) Prentice-Hall, Inc.
- (2) A. V. Aho, R. Sethi & J. D. Ullman, Compilers: Principles, Techniques, and Tools (1988) Addison Wesley
- (3) D. A. Padua & M. J. Wolfe: "Advanced Compiler Optimizations for Supercomputers", *C. ACM*, **29**, No. 12, 1184 (1986)
- (4) MATRIXx System Build Version 2.01 Core Module User's Guide, Editions 8 (1990) Integrated Systems Inc.
- (5) E. O. Gilbert & R. M. Howe: "Design Consideration in a Multiprocessor Computer for Continuous System Simulation", *Proc. National Computer Conf.*, p. 385 (1978) AFIP Press, Reston
- (6) R. Yoshikawa, T. Kimura, Y. Nara & H. Aiso: "A Multi-Microprocessor Approach to a High-speed and Low-Cost Continuous-system Simulation", *Proc. National Computer Conf.*, p. 931 (1977) AFIP Press, Reston
- (7) S. Koyama, K. Makino, N. Miki, Y. Iino & Y. Iseki: "On the Parallel Processor Array of Hokkaido University High-speed System Simulator 'Hoss'", *Proc. 8th IFAC World Cong.*, p. 1715 (1981) Pergamon Press, Oxford
- (8) S. G. Tzafestas (ed.): *Microprocessors in Robotic and Manufacturing Systems* (1991) Kluwer Academic Pub.
- (9) H. Kasahara, H. Honda & S. Narita: "Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR", *Proc. IEEE Trans. Conf. on Supercomputing*, p. 856 (1990)
- (10) 笠原・成田・橋本: 「OSCAR のアーキテクチャ」, 信学論, **J71-D**, 1440 (平成元-8)
- (11) 笠原博徳: 並列処理技術 (平 3) コロナ社
- (12) 成田誠之助: システム工学の手法 (昭 45) コロナ社

山 本 裕 治 (非会員)

平成 4 年早稲田大学理工学部電気工学科卒業。5 年同大学大学院修士課程在学中。マルチプロセッサシステムにおける各種アプリケーションおよびコンパイラに関する研究に従事。

鳥 居 宏 行 (非会員)

平成 4 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了。同年 4 月日本放送協会入社。

前 川 仁 孝 (非会員)

平成 4 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了。5 年日本学術振興会特別研究員。現在、早稲田大学大学院博士課程在学中。各種アプリケーションの並列処理に関する研究に従事。情報処理学会会員。

田 村 光 雄 (非会員)

平成 4 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了。5 年松下電器産業(株)入社。情報処理学会会員。

笠 原 博 徳 (正員)

昭和 60 年早稲田大学大学院博士課程修了。58 年同大学理工学部助手。60 年学術振興会特別研究員。61 年早稲田大学理工学部電気工学科専任講師、63 年同電気工学科助教授、平成 3 年同情報学科助教授、現在に至る。平成元年～2 年イリノイ大学 Center for Supercomputing Research & Development 客員研究員。工学博士。62 年 IFAC World Congress 第 1 回 Young Author Prize 受賞。電子情報通信学会、情報処理学会、シミュレーション学会、ロボット学会、IEEE、ACM 会員。

成 田 誠之助 (正員)

昭和 37 年早稲田大学大学院修士課程修了。37 年アメリカ・パデュー大学大学院留学(フルブライト留学生)。38 年早稲田大学理工学部助手、以後、講師・助教授を経て、48 年教授、現在に至る。工学博士。分散計算機制御システム、並列処理、産業用ロボット制御、ディジタル制御理論、CIM などの研究に従事。41、45 年電気学会論文賞受賞。計測自動制御学会、ロボット学会、IEEE 会員。