4T-6

OSCAR上でのインプリシット常微分方程式求解の並列処理手法のインプリメント

ﾌﾟﾚﾝﾁｬｲｻｳｧﾃﾞｨ ｳｨﾁｪﾝ、奥田　恒久、佐藤　東哉、
笠原　博徳、成田　誠之助
早稲田大学理工学部電気工学科

1.はじめに

本稿では、汎用目的マルチプロセッサ・システム上での、インプリシット常微分方程式求解の並列処理について述べる。

インプリシットな連立非線形微分方程式の高速な求解は、電子回路動特性解析等の様々な分野で要求されており、この要求を満たすために、最近ではマルチプロセッサを用いた並列処理が注目されている。しかし、その求解は、一般にスパース線形方程式求解計算を伴い、しかもその計算が全体の計算負荷の大半を占めるために、その効率良い並列処理は従来困難であった。これに対して筆者等は、スタティック・マルチプロセッサ・スケジューリング・アルゴリズム[1][2]を用いた、このスパース線形方程式求解の並列処理手法[3]及びそれを応用したインプリシットな連立非線形微分方程式求解の並列処理手法を既に提案している。ここではこの手法のOSCAR上でインプリメントについて述べる。

2.OSCAR のアーキテクチャ[6]

OSCAR(Optimally SCheduled Advanced multiprocessoR)のプロセッサクラスタは、図1に示すように、16台のプロセッサエレメント(PE)、3つのローカル共有メモリ(CM)、及びローカルコントロールプロセッサ(LCP)を3本バスで接続した平等型マルチプロセッサ・システムである。各PEは、小数の浮動小数点演算を含めた全てのインストラクションを1クロック(200 ns)で実行する32ビットRISCライクプロセッサであり、64個の汎用レジスタ、PE間データ転送用2ポートメモリ、256kwローカルデータメモリ、2バンクの64kwインストラクションメモリ、及びスタックメモリ等を持っている。

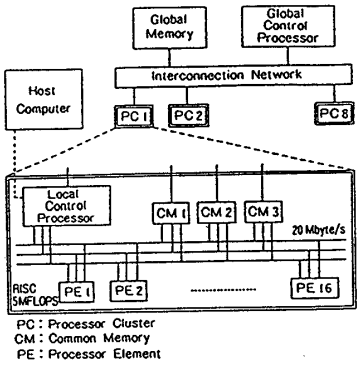

図1 ; OSCAR の構成

3.インプリシット非線形微分方程式の求解法

インプリシット非線形微分方程式の求解法には、スティッフな系に強い可変ステップ（刻み幅）可変次数のBDF(Backward Differential Formula)法を用いている。

$$f(x, \dot{x}, t) = 0 \qquad 0 \leqslant t \leqslant T \qquad (1)$$

但し、f及びXはベクタ

このBDF法は(1)式のような連立非線形微分方程式を次のような手順に分割して行なう。（但し、k：次数）

BDF1） ステップ h を決定する。

$$t_{n+1} = t_n + h \qquad (2)$$

BDF2） (3)式の線形方程式を解き、係数γ_iを求め、(4)式 より予測子X_{n+1}^{p} を求める。

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \left[\frac{t_{n+1}-t_n}{h}\right] & \left[\frac{t_{n+1}-t_{n-1}}{h}\right] & \cdots & \left[\frac{t_{n+1}-t_{n-k}}{h}\right] \\ \left[\frac{t_{n+1}-t_n}{h}\right]^2 & \left[\frac{t_{n+1}-t_{n-1}}{h}\right]^2 & \cdots & \left[\frac{t_{n+1}-t_{n-k}}{h}\right]^2 \\ \vdots & \vdots & & \vdots \\ \left[\frac{t_{n+1}-t_n}{h}\right]^k & \left[\frac{t_{n+1}-t_{n-1}}{h}\right]^k & \cdots & \left[\frac{t_{n+1}-t_{n-k}}{h}\right]^k \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_{k+1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad (3)$$

$$x_{n+1}^{p} = \sum_{i=1}^{k+1} \gamma_i x_{n+1-i} \qquad (4)$$

BDF3） (5)式の線形方程式を解き、係数α_iを求め、(6)式によりX_{n+1}を求める。

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & \left[\frac{t_{n+1}-t_n}{h}\right] & \left[\frac{t_{n+1}-t_{n-1}}{h}\right] & \cdots & \left[\frac{t_{n+1}-t_{n+1-k}}{h}\right] \\ 0 & \left[\frac{t_{n+1}-t_n}{h}\right]^2 & \left[\frac{t_{n+1}-t_{n-1}}{h}\right]^2 & \cdots & \left[\frac{t_{n+1}-t_{n+1-k}}{h}\right]^2 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \left[\frac{t_{n+1}-t_n}{h}\right]^k & \left[\frac{t_{n+1}-t_{n-1}}{h}\right]^k & \cdots & \left[\frac{t_{n+1}-t_{n+1-k}}{h}\right]^k \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad (5)$$

$$\dot{x}_{n+1} = -\frac{1}{h}\sum_{i=0}^{k} \alpha_i x_{n+1-i} \triangleq g(x_{n+1}) \qquad (6)$$

BDF4） (6)式を(1)式に代入してX(7)式のようなの非線形方程式を得る。その後(7)式を(2)で求めた予測子X_{n+1}^{p}を用いて、ニュートン・ラフソン法により解き、X_{n+1}を求める。このとき生成されるヤコビアン行列は、一般に大規模スパース行列になることが多く、筆者等が既に提案しているスパース線形方程式求解の並列処理手法はここに応用される。

$$f(X_{n+1}, g(X_{n+1}), t_{n+1}) = 0 \qquad (7)$$

BDF5） (8)式により、丸め誤差E_kを計算し、このE_kが許容誤差の範囲内であるならば次ステップのhとkを決定する。もし許容誤差より大きければ、hを小さくして手順BDF2に戻る。

$$E_k = \frac{h}{t_{n+1}-t_{n-k}} \| X_{n+1} - X_{n+1} \| \qquad (8)$$

BDF6） n＝n＋1とし、BDF1に戻る。

4.並列処理手法

4.1 タスク生成

計算を効率良く並列処理するためには、まず、処理

An Implementation of Parallel Processing of The Solution of Implicit Ordinary Differential Equations on OSCAR
Wichian PREMCHAISWADI, Tsunehisa OKUDA, Haruya SATOH, Hironori KASAHARA and Seinosuke NARITA
Waseda University

すべき全ての計算をタスク（プロセッサへの割当単位）に分割する必要がある。この分割レベル（タスク・グラニュラリティ）としては、演算要素レベル、代入文レベル、あるいは行・列単位の演算レベル等が考えられるが、OSCAR上で並列処理を行う場合は、PEの処理能力、PE間のデータ転送能力、同期のオーバーヘッド、あるいはタスク・スケジューラの性能等を考慮して、代入文レベルのタスク生成を行なう。なお、このタスク生成においては従来のタスクグラフと異なりニュートンラフソン法の部分が不定回繰り返し(REPEAT ループ)なるので、この繰り返し部分とその他の部分に分けたタスクグラフ(図2)の生成を行なうために中間語（従来の４つ組ライク）に do while, repeat until, if then else 等の制御命令を加えた特殊な中間語（図３）を導入した。

４．２　タスク・スケジューリング

生成されたタスク集合のプロセッサ上への最適割当て、実行順序決定問題は、実行終了時間最小マルチプロセッサ・スケジューリング問題に帰着されるが、本手法では筆者等が開発したスタティック・マルチプロセッサ・スケジューリング・アルゴリズムCP/MISF及びデータ転送を考慮したCP/MISF/DTを用いて、最適スケジューリングを生成する。なお、スケジューリングに際しては whileループ、 repeatループ、if構文のthen部とelse部に分割されたタスクグラフを個別にスケジューリングし、その後結合した形としている。

４．３　マシン・コード生成

次に４．２で得られたスケジューリング結果を基に、各プロセッサで実行するマシン・コードを生成する。このマシン・コードはタスク間同期オーバーヘッド・データ転送オーバヘッドを最小化し、プロセッサ内のレジスタを最適利用するように最適化が施されている。

５．結び

本稿では、スタティック・マルチプロセッサ・スケジューリング・アルゴリズムを用いたOSCAR上でのインプリシット常微分方程式求解の並列処理手法のインプリメントについて述べた。今後この並列処理手法を用いてOSCAR上で電子回路シミュレータ等を開発していく予定である。

参考文献

[1] H.Kasahara& S.Narita,"Practical multiprocessor scheduling algorithm for efficient pararell processing",IEEE Trans. Compt,c-33,pp 1023-1029 Nov,1984

[2] H.Kasahara& S.Narita,"An approach to supercomputing using multiprocessor scheduling algorithm",in Proc.IEEE 1st Int.Conf. on Supercomputing systems, pp 139 -148, Dec 1985

[3] H.Kasahara& Narita,"Pararell processing for the solution of sparse linear equations on OSCAR (Optimally Scheduled Advanced multiprocessoR), in CONPAR88,Sept 1988

[4] 中山 晴之、笠原 博徳 「スタティック・マルチプロセッサ・スケジューリング・アゴリズムを用いたインプリシットな常微分方程式求解の並列処理手法」情処 ３６全大、 ３Ｃ－９

[5] Leon O.Chua & Pen-Min Lin, " Computer-aided analysis of Electronic Circuit",pp 674-685

[6] 笠原、成田、橋本、「OSCAR (Optimally SCheduled Advanced multiprocessoR)のアーキテクチャ」、信学論(D),Vol.j71-D,No-8,pp. 1440-1445

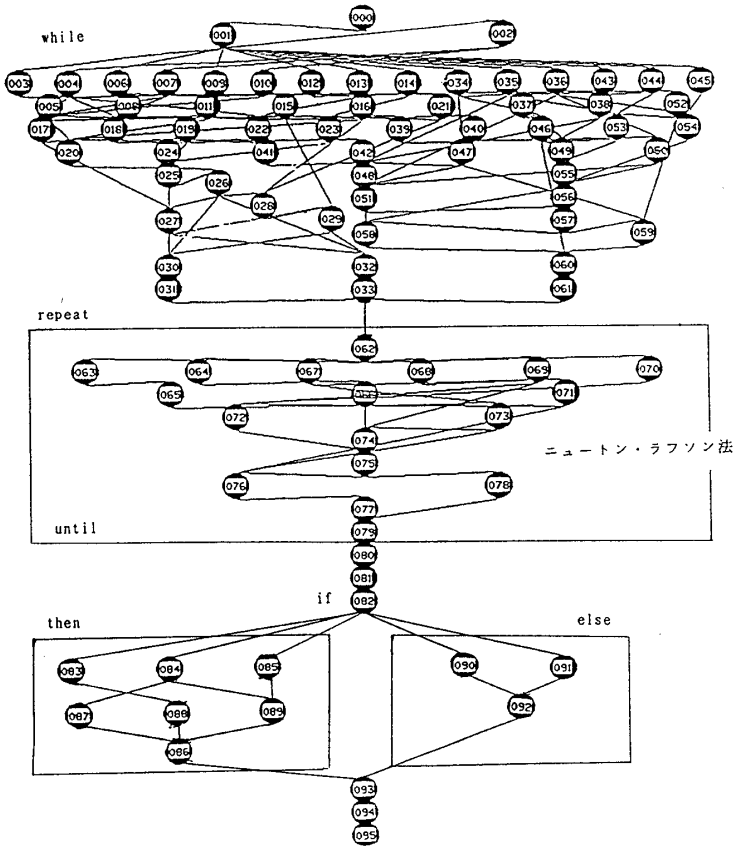

図２；タスクグラフ

```
while
>       v 5    v 1
do
:=      v25    v22
           :

repeat
+*      v48   v 9   v49   v10   v50   v11
/       t-1   v 6
*       t-1   v23
:=      v13   t-1
           :

until
>       v19   v51   v53
eou
-       v 9   v37
-       v14   v38
*       v 6   t-1
-       v 1   v 4
/       t-2   t-1
:=      v59   t-1
           :

if
>       v20   v59
then
*       v 6   v 7
:=      v 6   t-1
           :

eot
else
*       v 6   v 8
:=      v 6   t-1
           :

eoe
eow
end
```

図３；制御構造を持つ中間語