IV. 並列処理の応用

——回路シミュレーションを 中心として——

特集解説

中 田 登志之 (日本電気(株) C&C 研究所)

1. はじめに

近年、LSI 技術の進歩と共にマイクロプロセッサの処理能力が飛躍的に向上してきた。その結果、複数のマイクロプロセッサエレメント(PE)を用いることにより、大規模な演算を行うことが可能になってきた。

特に、アメリカでは HPCC というプロジェクトが推進されており、その中で「大いなる挑戦 (Grand Challenge)」として、以下のような超並列コンピュータの応答範囲が列挙されている。

- 磁気記憶技術研究・新薬研究・高速輸送手段(超高速機)研究
- 計算分子科学・燃焼解析・海洋モデリング・オゾン層解析
- 計算解剖学・公害解析
- 分子生物学 (DNA, タンパク質構造解析)・惑星地形解析

また、国内でも、種々のメーカ、研究機関で並列処理の応用に関する研究が行われている。特に毎年6月頃行われる JSPP (Joint Symposium on Parallel Processing) と、研究会の連合の形で8月に行われる SWoPP (Summer Workshop on Parallel Processing) は、国内の並列処理の最先端の動向を知る、貴重なシンポジウム/ワークショップとなっている。

各並列応用を紹介しても、あまりにも膨大になるため、本稿では二、三個の、比較的本学会の会員の方の興味に近いと思われる応用について、主に筆者が並列マシン Cenju/Cenju-2 で得た経験を元に紹介する。

なお、国内の並列処理の応用などについては文献(1)(2)などを参照されることをおすすめする。

IV. Application of Parallel Processing

——Focusing on Circuit Simulation——

By Toshiyuki Nakata (C&C Research Laboratories, NEC Corporation)

キーワード：並列処理、MIMD、LU 分解、共役勾配法、回路シミュレーション、有限要素法

2. 並列処理の実現方法と考慮すべき問題点

2.1 並列プログラムの実現方法

並列プログラムの実現方法としては、

- ①逐次型プログラムからコンパイラが並列性を自動的に抽出する方法
- ②プログラマが並列性を意識してプログラムする方法

近年並列化コンパイラの研究が盛んに行われており、並列化 Fortran 言語 (HPF: High Performance Fortran) の普及とともに、このアプローチでの成功例も増加しつつある。

一方、特に数十台以上の並列プロセッサを稼働させる場合、プログラマが並列性を意識して、アルゴリズムを並列プログラム用にチューンしないとなかなかそのような並列性を活用できない場合も存在する。

2.2 アルゴリズムの並列化の方法

ある応用を並列化する方針としては、

- ①アルゴリズムを実現するものを機能的なブロック/プロセスに分割してその間をデータの流れて結ぶ方法 (いわゆる機能的分割)
- ②問題が扱うデータを分割して、分割したデータを各プロセス/プロセッサに割り当てる方法 (いわゆるデータ分割: divide and conquer)
- ③両者の混合

などが考えられる (図 1)。

実際にはプロセッサの台数を増加させる場合、純粹に機能的な分割だけを用いることはほとんどない。何らかの方法でデータ分割を扱う必要がある。

2.3 SIMD 対 MIMD

次に考えるべきことはプロセッサを SIMD (Single Instruction Multiple Data stream) 作動させるか MIMD (Multiple Instruction Multiple Data stream) で作動させるかということである。

SIMD の利点としては、

図 1 並列処理の実現方法

- プロセッサ間の同期が不必要
- 命令供給が比較的容易なため、超並列処理に適している*

一方、MIMDの利点は、より複雑なアルゴリズムが記述可能**なことである。

近年はMIMDの一種である、SPMD (Single Program Multiple Data Stream) モデル (各プロセッサで同じプログラムが作動するもの) が一般的になりつつある。

2.4 並列効果を妨げる要因

並列処理による効果を妨げる要因としては、

- ①逐次処理部分のオーバーヘッドで有名なアムダールの法則³⁾によれば逐次処理部分の割合が全体の $s(0 \leq s \leq 1)$ 存在すれば n 台のプロセッサを用いて得られる速度向上は、

$$\frac{1}{s + \frac{1-s}{n}}$$

になる例えば1台実行時に逐次部分の割合が5% あればたとえ1,000台のプロセッサを用いても得られる速度向上はたかだか20倍である。

- ②負荷分散のばらつき (最も遅いものに抑えられる)
- ③通信/同期のオーバーヘッド

などがあげられる。

この中でも特に①の逐次処理のオーバーヘッドは、台数を増やせば増やすほど深刻になる。

*実際に数百万台以上のプロセッサを並べたシステムは全てSIMDである。

**機能分割などはこちらでしかできない。

表 1 Cenju で得られた速度向上

応用	性質	速度向上
回路シミュレーション	科学技術計算	25
モンテカルロプラズマシミュレーション	科学技術計算	48
LSI配線問題	横方向有線探索問題	38
有限要素法	科学技術計算	48/36

2.5 Cenju/Cenju-2 のハードウェア構成

Cenju は 64 台のプロセッサから構成される。各プロセッサは 8 台ごとに 1 本のクラスタバスで結合される。プロセッサ (以後 PE) には、MC 68020 (20 MHz), MC 68882, WTL 1167 ならびに分散共有メモリを実現するために 2 ポート化した 4 MB のメモリを用いることとした。

Cenju-2 は、多段網 (ペースライン網) によって要素プロセッサ (PE) が結合された構成をとっている。

PE は、ユーザプログラムの実行を行うアプリケーションプロセッサ (AP) とプロセッサ間通信を司る通信プロセッサ (NP) の二つの CPU と、64 MB の DRAM、二つの DMA コントローラから構成される。CPU としては、VR 3000 (25 MHz) + VR 3010 を用いており、Cenju [MC 68020 (20 MHz) + WTL 1167] に比べ約 10 倍の処理能力をもつ。

2.6 Cenju で得られた速度向上

表 1 に種々の応用に関して Cenju で得られた速度向上 (64 台時) を示す。

速度向上を阻害している原因としては、回路シミュレーションではネットワークに関する線形方程式の求解 (逐次ネック)、モンテカルロプラズマシミュレーションでは、負荷分散の問題、有限要素法/LSI 配線問題ではデータ転送があげられる。

3. 応用 1: 回路シミュレーション⁽⁴⁾

回路シミュレーションは基本的には非線形常微分方程式を陰解法と Newton-Raphson 法を用いて解く問題である (電力潮流計算問題と特性は似ている)。

シミュレーションの核となる部分は Newton-Raphson 法に基づくループで、

- ①素子モデルを評価して、Jacobian 行列 (連立方程式) を作成する

- ②連立方程式を解く

からなる。前者は、比較的容易に並列化できる。ただし、その結果として得られる Jacobian 行列は非零要素率が 2~3% と小さく、この連立方程式を解く部分を並列化しても得られる速度向上は小さい。この問題

を解決するために筆者らは回路シミュレーションをモジュール分割法を採用した。モジュール分割法では、シミュレーションの対象となる回路を結合網で結合された部分回路に分割する。各部分回路ならびに結合網はそれぞれ PE に割り当てられる (図 2 参照)。

シミュレーションアルゴリズムの核の部分は以下のように変更される。

並列フェーズ 1 各プロセッサは内部ノードの疑似解を求め、その Norton 等価値を結合網を担当するプロセッサに転送する。

逐次フェーズ 結合網に対応するプロセッサが結合網に関する縮退行列の線形方程式を解いて、境界変数の解を求め、各部分回路を担当するプロセッサに分配する。

並列フェーズ 2 各プロセッサは内部変数の真の解を境界変数の解を用いて連立方程式を解くことにより求める。

3.1 回路シミュレーションでの速度向上

図 3 に 2 種類の回路、回路 1、回路 2 のシミュレーションをプロセッサ数を変化させて実行させたときの速度向上比を示す。回路 1 は Cenju で実行するには

図 2 回路分割の概念

図 3 回路シミュレーションでの速度向上 (1)

小さな回路であり、Cenju で実行しても 16 台程度で速度向上は飽和すると予想される。一方、回路 2 は Cenju の仕様を設定したときに想定した (5,000-10,000 トランジスタ) 程度の大きさの回路である。部分回路数は 191 である。

回路 1、回路 2 共に速度向上としては 15-16 倍程度の速度向上が得られた。当初の予想に反して回路 1 で回路 2 であまり速度向上に差がなかった理由としては、

- ① 回路 2 の部分回路数が 191 個と PE 台数 64 の 3 倍程度となっており、その分接続ネットワークの行列自体が大きくなったこと
- ② 回路 2 の DRAM のモデルの特性のために、接続ネットワークの行列の非零要素率が 20% 程度と予想していたより大きくなったこと

の 2 点のために、逐次フェーズの占める割合が当初想定していたものより大きかったことがあげられる。このために、回路 2 では 1 台で実行する場合、逐次部分の全体に占める率が約 4% 弱になると推定される。

3.2 逐次部分の並列化-LU 分解の並列化

3.1 節では、部分回路数が増加すると逐次部分のオーバーヘッドが大きいことが判明した。本システムの逐次部分の処理の大半は境界解に関する総形方程式の求解処理における LU 分解の部分である。そこで、Do-Loop Level の並列処理を用いて、この部分の並列化に着手した。図 4 に外積法の LU 分解の並列アルゴリズムを示す。この部分に関しては、同期や通信のオーバーヘッドの及ぼす影響が他の部分よりはるかに大きい。本システムでは、図 4 で示した全 PE での同期などを不要にするなどの工夫を施した。図 5 に回路 2 について、LU 分解を 1 台で実行した場合と並列に実行した場合との速度向上を示す (並列 LU 分解は 8 台のプロセッサで実行している)。64 台時に 1 台のときの 23.9 倍、さらに同期回数の削減により、1 台の

```

for k: 1 to n do
  if (PeContains PivotRow){
    Broadcast  $a_{kj}$ , ( $j=k$  to  $n$ ) .....(a)
  }
Barrier() /*全 PE での同期*/
for All  $i$  in  $k+1$  to  $n$  which concerns
  this PE do .....(b)
     $a_{ik} := a_{ik} / a_{kk}$ 
    for  $j := k+1$  to  $n$  do
       $a_{ij} := a_{ij} - a_{ik} * a_{kj}$ 
    end
  end
Barrier() /*全 PE での同期*/
end.

```

図 4 LU 分解の並列化アルゴリズム

図 5 逐次部分の LU 分解を並列化した場合の速度向上

25.8 倍の速度向上を達成した。

なお、LU 分解でのより細粒度の並列性を活かすためには、Cenju のような粗粒度の並列性を扱うシステムよりも、より細粒度並列処理に適したマシンの方が適している。実際に笠原らは、並列マシン OSCAR 上で、L/U 行列の各要素を求める計算を 1 タスクとして、8 台で約 4.1 倍の速度向上を得ている。

4. 応用 2：有限要素法における連立方程式の並列化

一般に有限要素法の実線形非定常問題は、①剛性行列の作成、ならびに②剛性行列の線形求解からなる。剛性行列の作成に関しては文献(7)で述べたように、PE 台数 64 台のときに 1 台のときの約 49 倍の速度向上を得た。

本章では、後者の線形求解の並列化の手法に関して述べる。並列化の手法として、共役勾配法の一つである SCG 法⁽⁸⁾の検討について述べる。

4.1 SCG 法の検討

SCG (Scaled Conjugate Gradient) 法⁽⁸⁾は速水らによって提案された対角項によるスケーリングを施した共役勾配法の一つであり、その主な処理は以下に示すようになる。

- ① $D^{-1} = \text{diag}[1/a_{11}, 1/a_{22}, \dots, 1/a_{nn}]$ を求める。
- ② $r_1 = b - Ax_1$
 $p_1 = D^{-1}r_1$
- ③ i に関して反復計算

$$\alpha_i = \frac{r_i, D^{-1}r_i}{p_i, Ap_i}$$

$$x_{i+1} = x_i + \alpha_i p_i$$

図 6 SCG 法の並列アルゴリズム

図 7 大規模行列に対する速度向上(その 1)

$$r_{i+1} = r_i - \alpha_i A p_i$$

$$\beta_i = \frac{r_{i+1}, D^{-1}r_{i+1}}{r_i, D^{-1}r_i}$$

$$p_{i+1} = D^{-1}r_{i+1} + \beta_i p_i$$

ただし Ap , $D^{-1}r$, 内積などは反復あたり 1 回計算すればよい。演算量の大半は Ap の行列×ベクトル→ベクトルの計算に費やされる。

4.2 SCG 法の並列化

SCG 法の並列化にあたっては、基本的には行ごとにプロセッサに割り当てることにより並列化した。このときの並列化アルゴリズムを図 6 に示す。このときのプロセッサ間通信は以下の 2 点で発生する。

- (1) $A \times p = s$ を行うのに必要な p の転送。各プロセッサでは p のベクトルの要素のうち、割り当てられた要素に対して新しい値を得た後、担当する要素の値を全プロセッサに伝えないといけない。
- (2) ベクトル対ベクトルの内積を演算する時に部分和の段算、後述するように、(1) の転送が深刻なオーバーヘッドとなった。

4.3 大規模行列に対する SCG 法の評価(1)

8904*8904 (非零要素 184,776 個) のデータで評価

図 8 p が影響を及ぼす範囲

図 9 大規模行列における速度向上(その 2)

を行った。そのときの速度向上を図 7 に示す。10 台程度で 7 倍の速度向上で飽和している。そこで、1 反復の各演算部分の処理時間を計測した。その結果、18 台時には通信時間の方が $A * p$ に要する時間よりも時間がかかっていることが判明し、これが処理のあい路になっていることがわかった。

4.4 通信量の減少

4.3 節の評価では p の要素の通信時間が速度向上を妨げることがわかった。ところで、 p のベクトルの各要素が S に影響を及ぼす範囲は S の全領域ではなく基本的には、図 8 に示すように A の行列の形状によって定まる（これは LU 分解におけるバンド幅の概念と似ている）。この影響範囲は前もって計算できるので、各行の割り当てを帯状にすると、 p の各要素を通信するのはその要素が影響を及ぼす範囲を担当するプロセッサだけですむ。

4.5 大規模行列における SCG 法の評価(2)

図 9 に、データのプロセッサへの割り当てを変更し、 p の各要素は影響を与えるプロセッサにしかデータを転送しないように変更したプログラムでの速度向上を示す。

64 台時に 1 台のときの 38 倍の速度向上を達成した。また、同じ問題を VR 3000 ベースの並列マシン Cenju-2 で実行したところ、Cenju に比べて PE 1 台の場合 5.8 倍、PE 16 台の場合では 3.2 倍の性能とな

った。

5. おわりに

本解説では、筆者の並列マシン Cenju/Cenju-2 での並列処理の経験について述べ、並列処理の応用の紹介とさせていただいた。特に、「逐次処理部分の最少化」「プロセッサ間通信の削減」が、高い性能向上を達成する場合に不可欠な場合について述べた。

前者は、並列処理アルゴリズムを、ある程度プログラマが検討して、はじめて可能となる。これはある程度、並列処理が種々の経験を積んだ上で、ノウハウとして蓄積されるものであろう。

一方、後者のプロセッサ間通信の削減は、並列化コンパイラが活躍する余地が十分あると考えられる。

今後のこれらの分野での研究がより一層進み、並列処理が、「当たりまえの技術」として使われて行くようになることを祈念するものである。

(平成 5 年 10 月 19 日受付)

文 献

- (1) 矢川元恭編集：「計算力学に於ける並列コンピューティング」，コンピュータール，No.40(平 4-10)
- (2) 「並列/分散処理論文特集」，信学論，J75-D-1, 8, (平 4-8)
- (3) G. M. Amdahl: "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities", *Proc. SICC*, 30 (1967-4)
- (4) 中田，他：「並列回路シミュレーションマシン Cenju」，情報処理，31, 5, 593~601 (平 2-5)
- (5) Nakata, et al.: "A Multiprocessor System with a Distributed Shared Memory Scheme for Modular Circuit Simulation", *Proc. International Symposium on Shared Memory Multiprocessing*, p. 82~90 (1991)
- (6) 吉成・中山・田村・前川・笠原・成田：「直接解法による電子回路シミュレーションの並列化」，電子情報通信学会春季大会予稿集，p. 6-164 (平 5)
- (7) 加納，他：「並列シミュレーションマシン Cenju 上の有限要素法一剛性行列の作成部分の並列化」，情報学会第 42 回全国大会予稿集，p. 6-72~6-73 (平 3)
- (8) 速水・原田：「対角項スケーリングを施した共役勾配法のベクトル計算機における有効性について」，情報処理学会論文誌，30, 11, 1364~1375 (平 1)
- (9) Yamauchi, et al.: Proton: "A Parallel Detailed Router on an MIMD Parallel Machine", *Proc. ICCAD'91*, p. 91~95 (1991)

中 田 登志之

昭和 55 年京都大学工学部情報工学科卒業，60 年同大学大学院工学研究科情報工学専攻博士後期課程満期退学。同年 NEC 入社 C&C 研究所勤務，現在 C&C 研究所研究課長。入社以後，並列計算機のアーキテクチャ／応用の研究に従事。