並列数値シミュレーションの静的負荷分散法の検討

指導教官：市川 周一　　学籍番号：973722　田村 広志

1　背景と既存の研究

並列処理を用いて数値シミュレーションを高速化するために，多くの研究や試作が行われてきた．プロセッサ台数に応じた性能向上を得るためには適切な負荷分散が必須であるが，一般に静的負荷分散問題は計算困難であるため精度保証の無い近似解法が採用されてきた．

市川ら [1] は，並列 PDE 求解システム NSL において静的負荷分散問題を組み合わせ最適化問題として定式化し，分枝限定法を用いて実用的に最適解を求める方法を示した．この研究ではブロック数 m とプロセッサ数 n の間に $m \ll n$ の関係がある場合を扱っており，それ以外の場合は負荷が適切に分散できない．この問題を解決するため仮想プロセッサを採用する方式も提案された [2] が，この方式では性能が改善可能であることが示されただけで，具体的・実用的な解決法は未解決のまま残っている．

本研究では静的負荷分散を一種の箱詰め問題としてモデル化し，分枝限定法でこの最適化問題を実用的規模まで解くことが可能か検討する．

2　計算モデル

本研究では NSL の計算モデル [1] を単純化して扱う．計算モデルは m 個のブロックから構成されており，各ブロックは並列にデータの処理を行う．さらに各ブロックは隣接するブロックとデータの交換 (通信) を行う．ブロック B_j の計算時間を Ta_j，通信時間を Ts_j と置いたとき，全体の処理時間 T は以下の式で表されるとする．

$$
\begin{aligned}
T &= \max_i T_i \qquad (0 \le i \le n-1) \\
T_i &= \sum_{B_j \in g_i} (Ts_j + Ta_j) \\
Ts_j &= \sum_{B_l \notin g_i} (CtsSc_{j,l} + Dts) \\
Ta_j &= CtaSa_j + Dta
\end{aligned}
$$

ここで T_i はプロセッサ P_i の実行時間，g_i は P_i に割り当てるブロックの集合，B_l は B_j の接続先のブロック，$Cts \cdot Dts \cdot Cta \cdot Dta$ は定数，$Sc_{j,l}$ は B_j と B_l 間の転送量，Sa_j は B_j の格子点数 (計算量) である．この各定数は実装に依存する [1].

本研究では $m \ge n$ の場合を扱うので，m 個のブロックを n 個のプロセッサに分配する方法 (n^m 通り) の中から，T を最小にする方法を選ぶことが目的となる．

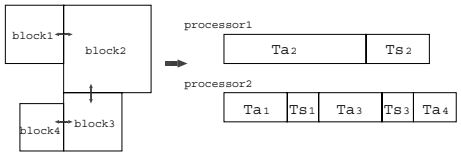


図 1：ブロック構造とプロセッサへの分配

3　実験方法

本研究は以下のような条件下で実験を行った．

- 分枝限定法では暫定解を用いて探索空間を制限するので，良い近似アルゴリズムを選ぶことが必須である．そこで本研究では以下の 4 通りの近似アルゴリズムについて検討する．
  1. B_j を P_k に割り当てて実行時間を計算する ($k \equiv j \bmod n$).
  2. ブロックを格子点数の大きさで降順にソートし 1. を適用する．
  3. 降順ソートし，現在最小負荷のプロセッサに順に割り当てていく．このとき計算時間 Ta_j を負荷として考え，ブロックを割り当てる毎に計算する．
  4. 3. の負荷に，通信時間 Ts_j を加味して見積もる．
- 与えるデータは図 1(左) のような木構造に接続されたブロックとする．各ブロックは正方形とし，大きさは格子点数を 1 から 10000 までの範囲でランダムに決める．
- ブロック数を変化させた時の各近似値の精度 (1)・最適解を求めるまでの分枝数 (2) を測定する．(1)・(2) において実行時間中の通信比を変えた場合も測定する (Cta を固定し Cts を変える)．

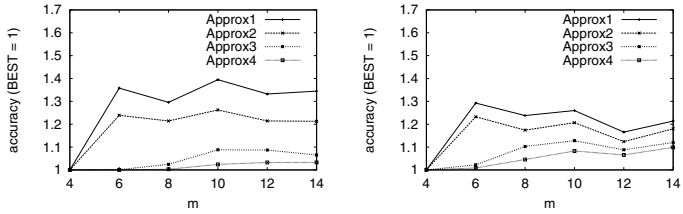
4　実験結果

各パラメータの数値は，$Cta = 1.0, Dta = 0.1, Dts = 0.1$ とした．Cts は 0.5 と 20 の二つについて測定した．また，$n = 4$ とした．

(1) の結果を図 2 に示す．図中で近似アルゴリズム 1 を用いたものを Approx1 としている．2・3・4 についても同様である．図 2 よりプロセッサへのブロック配置・通信時間を考慮する程に近似精度が良くなっていることが確認できる．また，近似アルゴリズム 1・2 はブロック数がプロセッサ数の倍数になるとデータのばらつきが小さくなるため近似精度がよくなる．

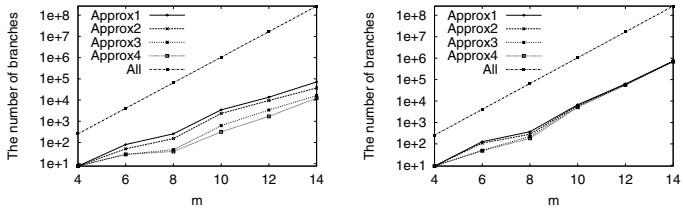
(2) の結果を図 3 に示す．図中で探索木の全分枝数を All としている．図 3 より近似アルゴリズムによって分枝数の指数的増加を抑えることに成功していることが確認できる．

$Cts = 20$ のように通信時間の占める割合が大きい場合，計算時間を基準に近似しても良い暫定解は求まらないので図 2(b) に示すように各近似アルゴリズムの差は小さくなる．このため図 3(b) でもブロック数が大きくなるにつれ分枝数の差がなくなる．



(a) $Cts = 0.5$　　(b) $Cts = 20$

図 2：ブロック数 m と近似値の精度



(a) $Cts = 0.5$　　(b) $Cts = 20$

図 3：ブロック数 m と分枝数

5　まとめ

最適解を求める計算時間は探索時の分枝数に比例している．従って図 3 より近似アルゴリズム 4 で最適化の計算時間が $\frac{1}{10}$〜$\frac{1}{10000}$ に短縮できることが分かる．しかしここで用いた各パラメータは実装依存であるため，実際に実用的規模まで解くことが可能かどうかは実機で確認する必要がある．また実行時間のうち通信時間が支配的である場合，本研究の近似アルゴリズムでは効果がほとんど見られない．今後，より応用範囲が広く精度のよい近似アルゴリズムの開発が望まれる．

参考文献

[1] 市川周一，川合隆光，島田俊夫：組合せ最適化による並列数値シミュレーションの静的負荷分散，情報処理学会論文誌，Vol.39，No.6 (1998)．

[2] Shuichi Ichikawa, Takamitu Kawai, Toshio Shimada： Enhanced Optimization Scheme for Parallel PDE Solver of NSL，並列処理シンポジウム JSPP'98，pp.143 (1998).