5F−5

超並列システム用OS「超流動OS」における アプリケーションソフトウェアの動的最適化

須崎 有康　栗田 多喜夫　田沼 均　平野 聡

電子技術総合研究所

1　はじめに

超並列計算機上でのソフトウェア開発では、計算機アーキテクチャとアルゴリズム及び処理するデータの適合性を考慮せずに効率の良いプログラムを記述することはできない。しかし、こうした適合性を調べるためには計算機アーキテクチャの特性とアルゴリズムの特性を理解し、動的な振舞いを推論できなければならない。このような推論は超並列計算機のように大規模複雑になると人間の手に負えるところではない。

我々は先に、逐次計算機上でデータの性質に合わせ最適なアルゴリズムを選択実行する AASM [1] を提案した。AASM ではまず同一機能を実現する複数のアルゴリズムを用意し、事前テストとして多種データに対し各アルゴリズムの実行時間を測定する。この結果から図 1 のようなデータの性質とそれに最適なアルゴリズムの関係をニューラルネットで学習する。通常の実行では、ニューラルネットがデータの性質から最適アルゴリズムを推論し、実行することで効率化を図る。

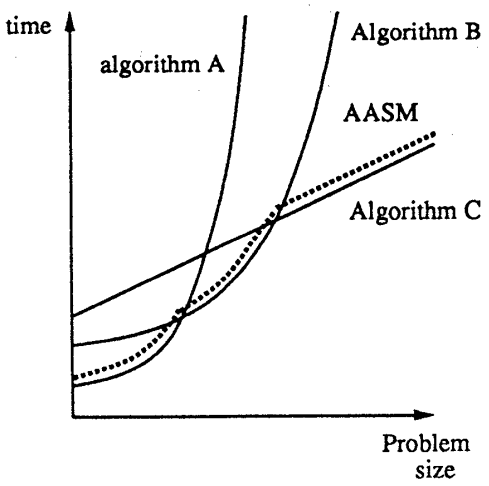

図 1: 逐次版 AASM

超並列システム上のソフトウェア開発では、並列アルゴリズムの動的な振舞いを詳細に理解することは難しくなるため AASM のような動的最適化手法が必要となる。本論文では分割統治法に基づく並列アルゴリズムに対して、AASM を並列版に拡張した動的最適化手法 p-AASM を提案する。

2　p-AASM

p-AASM では AASM を並列アルゴリズムが扱えるように拡張する。並列アルゴリズムは、データ領域を再帰的に分割しつつ、個々の領域を並列に処理し、結果を統合する「並列版分割統治法」を対象とする。

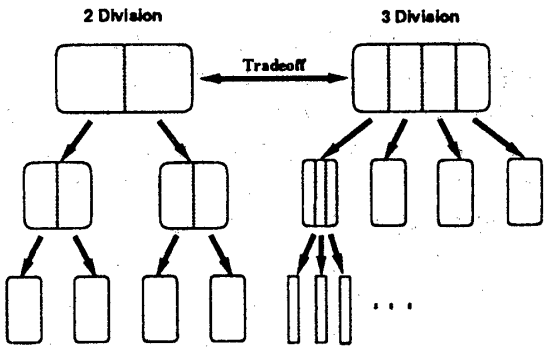

図 2: 分割法ごとの実行時間のトレードオフ

並列版分割統治法ではデータの分割法が多数存在する。図 2 に示すように二分割、三分割など、問題の並列度がある限り分割することができる。しかし、通信のオーバーヘッド(スレッドの生成、データの転送、同期)を考慮して、問題のサイズに応じて分割する数の最適値を求めなければならない。また、図 2 に示すように分割統治法では再帰が基本的手法である。ある問題のサイズ以上では 1 台のプロセッサで分割処理を集中するよりも再帰の段数を増やし、ネズミ算的に分割したほうが効率が良い。このため一回に分割する数と再帰の段数のトレードオフを求めなければならない。

また、再帰的に問題を分割していくと、ある問題のサイズで並列処理する場合より 1 台のプロセッサで逐次処理したほうが効率の良い時点がある。このサイズを最適な粒度とする。最適な粒度は計算機アーキテ

"Fluid", An OS for Massively Parallel Systems: Dynamic Tuning of Application Software

Kuniyasu SUZAKI　Takio KURITA
Hitoshi TANUMA　Satoshi HIRANO
Electrotechnical Laboratory

クチャやアルゴリズムやデータの性質によって変化する。効率良く実行するためには、如何に分割して最適な粒度に適合させるかを決定しなければならない。

並列アルゴリズムでは上記のように動的に変化する不確定要素を含むため、p-AASM では AASM で用いた事前テストからデータの性質と最適アルゴリズムの関係を簡単に求めることができない。このため 3 章に示す学習方法を用いる。

3 並列アルゴリズムの学習方法

AASM での学習データの作成では、各データに対して各々のアルゴリズムを実行することにより、データの性質と最適なアルゴリズムとの関係を容易に求めることができた。しかし分割統治を基本にした並列処理では、データと実行時間の関係が分割する数や動的に変化する再帰の段数によって無限に存在し、すべて調べ尽くすためには膨大なテストを行なう必要がある。そのため提案する p-AASM ではこのような膨大なテストを回避するために、学習データを作りながら学習するインクリメンタル学習を用いる。

3.1 インクリメンタル学習

インクリメンタル学習では並列アルゴリズムの動的振舞いとして次の三つを想定する。まず、問題のサイズが小さいときは逐次のアルゴリズムのほうが効率が良いとする。このため並列分割統治法で再帰的に分割する途中で逐次アルゴリズムに切り替わる。二つ目は、再帰の段数は問題のサイズとともに多くなるとする。従って問題のサイズの小さいものからテストすれば、再帰の段数が推論できることになる。三つ目は、一回に多数に分割する時間と小数で再帰的に分割する時間はトレードオフであるため、一回に分割する数は一定数以上にならないとする。これにより分割する数のテストを定数回で終了できる。

上記の想定に基づいて並列アルゴリズムの動的振舞いを問題のサイズが小さいものからインクリメンタルに学習する。分割する数は一定数以上大きくならないためプログラムで固定し、分割する数が異なるプログラムを複数用意する。並列アルゴリズムの中では分割されたデータの処理として p-AASM を呼び出す。このため、逐次アルゴリズムのみを対象とした p-AASM 用のニューラルネットをあらかじめ AASM の手法で作成しておく。p-AASM のニューラルネットの構成は図 3 のようになり、最初は逐次実行のアルゴリズム (algorithm A,B,C) のみ選択され、並列アルゴリズム (2 division, 3 division) は選択されない。

学習の方法としてはまず、問題のサイズの小さいものから並列アルゴリズム (2 division, 3 division) の性能を測定する。その結果を逐次アルゴリズム (algorithm A,B,C) の実行時間と比較し、最適となった場合には p-AASM のニューラルネットを再学習する。ここで、p-AASM で並列アルゴリズムが選ばれる機会が生じる。さらにテストを続けると、あるサイズの問題で p-AASM 内での並列アルゴリズム (2 division, 3 division) が p-AASM を呼び出したとき、そこで再び並列アルゴリズムが選択される。この結果を学習することで再帰が生じる。再帰の段数は問題のサイズとともに増加する。分割する数も問題のサイズによって 2 division と 3 division のうち適する分割が選択されるようになると考える。さらに再帰により、相互に呼び出し合うことも考えられる。

上記のようにインクリメンタル学習では再帰の少ない段数から多い段数を学習することで、テストの回数を抑えて分割する数と再帰の段数を学習することが可能となる。

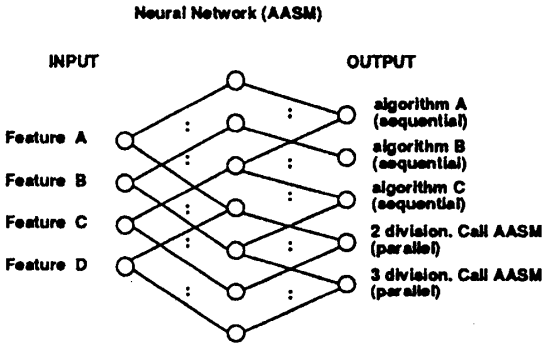

図 3: インクリメンタル学習のニューラルネット

インクリメンタル学習を用いた p-AASM では再帰の段数の数だけ p-AASM が起動されるためオーバーヘッドが大きくなると考えられるが、分割する数と再帰の段数の最適調整を可能であるため、オーバーヘッドに見合う効率が得られると考えられる。

4 おわりに

並列アルゴリズムの動的最適化手法である p-AASM を提案した。今回の提案では、並列アルゴリズムとして並列版分割統治法に絞った。分割統治法は最も簡単な並列化手法であり、その応用範囲は広いと考えている。今後は p-AASM を実計算機上に実装し、アーキテクチャごとの効率向上を調査する予定である。

謝辞

本研究の一部は RWC 計画の一環として「超並列システムアーキテクチャに関する研究」で行なわれたものである。関係各位に感謝いたします。

参考文献

[1] 須崎, 栗田, 田沼, 平野: 適応的アルゴリズム選択法による動的最適化, 第 34 回プログラミングシンポジュウム報告集, pp.177-184 (1993)