論 文 の 内 容 の 要 旨

論文題目　　A Study of Auto-tuning Parallel Library
（自動チューニング機能付き並列ライブラリに関する研究）

氏　名　　黒 田 久 泰

近年、あらゆる産業分野・学問分野で計算機の利用が急速に広まり、より大規模な問題をより短時間で解くために、並列計算機が利用される機会が増えつつある。しかしながら、利用者が直接、全てのプログラムコードを記述することは少なくなり、代わりに数多くのライブラリが提供されるようになってきた。これらのライブラリを利用することで、利用者はプログラムの作成にかかる時間を大幅に削減することができる。現在、数値計算ライブラリとして、逐次版では Netlib templates が有名である。並列数値計算ライブラリとしては、PETSc などが挙げられる。基本線形計算ライブラリ(BLAS)については各ハードウェアメーカーが独自にライブラリを提供している場合が多い。しかしながら、従来のライブラリの利用は、利用者のプログラム作成にかかる時間を減らすことはできても、並列計算機を有効に利用させることにはならない場合が多い。さらに、連立一次方程式を例に取れば、利用者はたくさんある解法やいろいろな種類の前処理の中から、問題を解くのに適した方法を自ら選択しなくてはならず、利用者に高度な知識を要求する場合も多い。そして、正しく選択できなかった場合には、解が求まらないこともあり得る。本研究の目的は、並列ライブラリにおける利用者による設定項目を少なくする、および高い実行性能を出すという相反する 2 つの要求を満足することである。それには、実行速度を向上させるために自動的にライブラリ自身が振る舞いを変えるという自動チューニングの機構が必要となる。似たような研究としては、PHiPAC(C コンパイラが最適化を行いやすいソースコードの生成)、ATLAS プロジェクト(BLAS3 のブロック幅の自動調節など)などがあるが、本研究はライブラリ実行開始後に自動チューニングを行うという点で、大きく異なっている。これにより、実行して初めてわかる情報を元にチューニングを行うことが可能となる。その情報とは、入力として与えられる問題や実行プロセッサ台数などである。これらの情報と実行される並列計算機のプロセッサ性能や通信性能から、最適な解法の選択を行う。

これまでは、問題を解く場合のアルゴリズムの計算量や収束性に研究の主眼が置かれがちであった。こういった場合、計算機による数値実験では、一機種のみで十分であった。しかし、本研究では、実行時間の最も短くなる解法を常に優先し、計算量や反復回数の増

加については問題にしない。また、一機種に限らず、いろいろなタイプの計算機上で高い性能が得られるようにする。つまり、アーキテクチャの違いや問題の性質の違いを吸収して性能向上を図るアルゴリズムの構築が必要となってくる。そして、アルゴリズムの選択やパラメータの調整を利用者が行うのではなく、できるだけ高い実行性能を引き出すようにライブラリ自身が自動的に決定するという仕組みを導入する。

例えば、すでにコンパイル済みのライブラリでは、アンローリング段数を変更するといった性能向上のための調整ができない場合が多い。また、ソースコードで提供されているライブラリにおいては、利用者自身が用いる解法、アンローリング段数、通信方式といった項目を設定し、その後コンパイルを行いライブラリを構築するという場合が多い。この方法だと、高い性能を引き出すためには問題が与えられるたびにコンパイルが必要になってしまい、大変な作業を伴う。

本論文では、自動チューニングを適用する例として連立一次方程式の反復解法である共役勾配法と一般化最小残差法の２つを取り上げた。特に係数行列が疎行列の場合には、与えられた問題の行列の非零要素の分布の仕方に応じて、行列格納形式や行列ベクトル積で使われる演算カーネル及び最適な通信方式の選択が必要になる。また、係数行列の固有値に応じて、効果的な前処理も変更する必要がある。共役勾配法では、比較的単純な反復式が繰り返されるため、前処理の効果が特に大きく影響する。そこで、前処理の自動選択を行うライブラリの構築を行った。一般化最小残差法においては、さらに直交化アルゴリズムの選択やリスタートをどのように行うかという要素も加えて自動チューニングを行うライブラリを構築した。

疎行列では、非零要素の値だけをメモリ上に保持することでメモリの使用量を減らすということが一般的である。その際に、非零要素のインデックスだけを格納したインデックス配列を利用する。ある位置にある非零要素の値を取り出すには、この配列を参照する必要があるために、メモリアクセスは常に間接参照にならざるを得ない。そこで、間接参照を含むループをアンローリングしたコードを多数用意しておいて、最も高速に計算できるコードを自動選択して性能を向上させるという機構を採り入れることで、速度向上を達成した。

さらに、分散メモリ型並列計算機上においては、疎行列・ベクトル積の際に多くの通信が発生する。これは、行列やベクトルが全プロセッサに分散されて保持されていることによる。密行列・ベクトル積では、通常、ベクトル要素の全対全通信を行う以外に方法はないが、疎行列・ベクトル積においては非零要素の分布を予め集計しておき通信を必要最小限に行うようにスケジューリングすることが可能である。しかし、通信にかかる立ち上がり時間が無視できないため、小さいサイズのデータを数多く通信するより、無駄なデータが含まれていても通信回数を少なくする方が結果的に早い場合もある。ここでは、係数行列が与えられてから実際に疎行列・ベクトル積にかかる時間を計測し、最適なコードを選択するという方法を用いることで、全体の実行時間を減らすことに成功した。

また 2 つの反復解法に共通することであるが、並列計算機においては、自動チューニング機能は逐次計算の場合に比べて特に重要になってくる。利用者が問題を解く際には、利用するプロセッサ数を変更する場合も当然考えられる。また、少ない数のプロセッサ上で十分なデバックをした後に、より多くのプロセッサで実行するというのが典型的な例である。しかしながら、少ない数のプロセッサで十分な実行性能が得られていても、プロセッサ数が多くなった場合に十分な性能、つまり台数効果が得られるという保証はない。そこで、本論文では具体的にグラムシュミットの直交化に関して、実行するプロセッサ数に応じて計算方法を変更すれば高い性能向上が得られることを示した。

また、逐次計算機上では計算量の増加のためにほとんど利用価値のない計算手法が、並列計算機上では高い台数効果が得られ結果的に速度向上につながる場合があることを述べる。この例として行列の逆行列の近似としてノイマン級数展開で得られる行列多項式を考え、これを前処理として適用する例を示し実際にプロセッサ数が多くなった場合に効果が上がることを示した。

自動チューニング機能付きライブラリの問題点としては、ライブラリのコードサイズが大きくなってしまうことと、チューニングのための余分なオーバーヘッドがかかることが挙げられる。前者については、ここ数年、大きなサイズの問題を解くため、プロセッサ性能の向上に伴ってメモリ容量も増えてきた。しかしながら、プログラムの実行コードが格納されるメモリ量はほとんど一定である。つまり、実行コードが占めるメモリ使用量は問題データを格納するメモリ容量に比べて十分に小さいので、実行コードのサイズを気にする必要はほとんどなくなっている。後者については、ライブラリ構築の際に、自動チューニングを行う項目を本論文で示す指針に基づいて正しく設定することで、無駄なオーバーヘッドを少なくすることができる。また、問題が難しくなるほど、自動チューニングのオーバーヘッドにかかる時間の割合が小さくなるため速度向上率が大きくなり、実用上はほとんど問題がないと言える。

このように自動チューニング機能付きライブラリは、利用者の利便性を向上させるだけではなく、並列計算機の資源を最大限に活用することにつながる。また、ライブラリを実行している最中に問題の性質を分析しているため、ライブラリの実行が終了するまでに、予測時間についても知ることが可能である。一般化最小残差法においては、自動チューニング機能付きライブラリの一切のパラメータを設定することなしに利用した場合でも、現在並列数値計算ライブラリとして広く使われている PETSc ライブラリと比べて 4 倍程度の高速化が達成される例も確認できた。今後、自動チューニングを備えたライブラリが増えていくことは間違いなく、本研究が役立つものと思われる。