電気通信大学紀要 20 巻 1 ・ 2 合併号 pp.1-14（2007）〔解説〕

数値計算応用を対象とする自動性能チューニング技術

弓 場 敏 嗣

Automatic Performance Tuning for Numerical Software

Toshitsugu Yuba

Abstract

Long-life software available during a long range period is required in high-performance computing applications. There appear two remarkable tendencies in the field; increase of software development cost and decrease of hardware development cycle. Rapid cycle of hardware development forces to make long-life software indispensable. As a result, automatic performance tuning (auto-tuning, in short) is in the spotlight, which leads to keep numerical software long-life by automatically adjusting its performance to various computing environments. Performance tuning in high-performance computing generally needs professional knowledge on target applications, libraries and computing systems to obtain the maximum performance. Here, performance means speed, accuracy and stabilization of computation. Auto-tuning makes it possible to automatically tune performance to multiple computing systems.

In this paper, auto-tuning technology for numerical software is overviewed. First, auto-tuning is formalized as "optimized mapping" from source programs to object programs, which can be executed efficiently on a target computing system. Secondly, a framework of optimized mapping is introduced and the auto-tuning mechanism for numerical software is described. Then, the effectiveness of auto-tuning is shown by using experimental results based on a practically implemented numerical library with an auto-tuning mechanism. Finally, related work and future problems are remarked.

Keywords: Automatic tuning, performance tuning, numerical software, numerical library, optimization.

1．はじめに

1．1　背景

近年、高速化を目指す高性能コンピューティングの世界では、自動性能チューニング（Automatic Performance Tuning）技術が注目を集めている。半導体技術の進歩により､高性能コンピューティングを支えるプロセッサ（マシン）の技術革新は速度を速め、その開発サイクルは著しく短くなっている。一方、ハードウェア複雑度は増し、ソフトウェア開発に要する費用と時間の増大化傾向を強めている。

高性能マシンを提供する事業者とその性能を享受する利用者にとって、旧世代のマシン環境で構築されたソフトウェア資産を、新世代マシン環境のもとで継承する必然性がある。事業者側では、オペレーティングシステム（OS）、コンパイラなどのシステムソフトウェアについて、対象マシンアーキテクチャの性能を最大限に利用すべく、新世代マシンのソフトウェア環境の整備に努力が払われている。しかし、応用ソフトウェアの優先度は、システムソフトウェアにくらべ相対的に低い。応用ソフトウェアを開発する場合、高性能マシンが保持する性能を有効利用するためには、応用、ライブラリ、プロセッサアーキテクチャについての知識と実装のスキルが不可欠である。しかしながら、そうした高度なプログラミング能力

Received on October 16, 2007.
Professor Emeritus, Formerly Professor in the Department of Infomation Network Science
電気通信大学名誉教授（元情報ネットワーク学専攻教授）

をもつ人材は多くない。ソフトウェア資産の再利用は、ソフトウェア生産性の向上を企図するソフトウェア工学への期待と合せて、高性能マシンの事業者にとって急を要する課題となっている。

高性能マシンの利用者の立場からは、長寿命ソフトウェアへの強い要請がある。数値計算ライブラリなどの応用ソフトウェアの利用者にとっては、永年の資産であるソフトウェアを新世代マシン環境下で再利用できることが第一義的に重要である。当然ながら、ソフトウェアの再利用においては、従来のマシン環境におけるよりも高性能が実現できることが最低限必要とされる。

このような状況下で、自動性能チューニング技術は、数値計算ライブラリ（以下、数値ライブラリ）などの旧世代ソフトウェア資産を活用する手段として提案された。ソースプログラム表現の旧世代ソフトウェアに変更を加えることなく、新しいマシン環境下でコンパイルするだけで、その高性能を自動的に発揮させる。例えば、旧世代マシン環境向けに開発された数値ライブラリを新世代マシン環境に移植（インストール）するとき、新マシン環境のもつ高い性能を自動的にひきだすことが、自動性能チューニング技術の狙いである。

歴史的に見ると、自動性能チューニング技術は、高性能コンピューティング、とくに数値計算応用の分野で生まれた [1,2,3,4]。過去において自動チューニングをもっとも必要とし、かつその適用が有効であった分野が数値計算応用であった [17]。しかし、マシン環境の開発サイクルの速さは、スーパコンピュータによって代表される高性能マシン系だけではない。例えば、多品種化と高性能化が著しい組み込み系システム分野において、昨今、高性能コンピューティング分野におけると同じ状況がもたらされている。組み込み系に用いられる低消費電力の汎用マイクロプロセッサ用ソフトウェア開発は、ユビキタスコンピューティング社会の到来とともにますます緊急度を増している。ASIC、システム LSI、FPGA を用いて開発される専用プロセッサは多種多様であり、それらを用いた専用システムのソフトウェア開発も開発費用の増大化が問題となっている。

また、広域分散コンピューティングシステムとして注目を浴びているグリッドコンピューティング分野では、広域ネットワークにつながる多種類のマシン環境のそれぞれに適合する応用ソフトウェアを、いかに効率よくかつ低費用で開発するかという要請がある。ソフトウェア開発費用の低減化が不可欠な情報技術分野において、自動性能チューニングは、今後ますます重要となることが予想される技術パラダイムである。

1.2　定義

自動性能チューニング技術は、以上の３つの観点から説明される。

- ●利用者：マシン環境のハードウェア性能を、人手を介さず最大限にひきだすことを可能とするソフトウェア技術
- ●事業者：新世代のマシン環境用応用ソフトウェアを、新たに開発することを不要にする方法論
- ●技術者：プログラムを新しいマシン環境へ移植(写像)するとき、その過程で自動的に最適化を行う技術

本稿では、自動性能チューニングを技術者の観点からとらえ、その現状と課題について議論する。すなわち、与えられた新しいマシン環境のもとで、プログラムとして表現される問題の解を得ようとするとき、マシン環境および問題の特性を考慮して、もっとも高性能な写像が自動的に得られる方法について考察する。ここで、写像という言葉は、プログラムとして記号表現された問題を対象マシン上で実行可能な表現に変換することを意味する。例えば、コンパイラは、ソースプログラムを目的プログラムに静的に写像する。写像の過程でいくつかの段階が存在する。ある段階では、ソースプログラムからソースプログラムへの写像のように、プログラム自身の記述水準は変わらない。また、最適化は、ある評価基準に基づき、もっとも高性能を得るための変換操作をいう。写像の過程で、最適化が繰り返される。この意味で、写像の過程は最適化の過程でもある。なお、以下では、チューニングの対象は性能に限定されるので、自動性能チューニングを、単に自動チューニングとよぶ。

自動チューニングを定式化して表現すると、次のように定義される。マシン環境の特性を表すパラメタ（特性パラメタ）と、問題を記述するプログラムによって規定される問題パラメタがあたえられたとき、速度や精度などの性能に影響を及ぼす調整可能なパラメタ（調整パラメタ）を自動的に最適化することを＜自動チューニング＞と定義する。ここで、マシンの特性パラメタとは、使用するプロセッサの速度、記憶容量、キャッシュサイズなどであり、あたえられたマシン環境によって先験的にきまる。並列システムの場合は、上記の他、プロセッサ数、結合網の通信性能などが含まれる。問題パラメタは、問題の行列サイズなどであり、実行するプログラムがあたえられるときまる。また、調整パラメタは、自動チューニングすべき性能パラメタであり、ループ展開段数、データ分割のブロック幅などが典型的である。高性能化のためのチューニングの選択対象には、後述するように、アルゴ

リズム要素、プログラム部品も含まれる。用意されたこれらの候補の中から最適なものを１つ選択することも、調整パラメタの最適化と考える。

自動チューニング技術が最適化対象とする性能には、以下のようなものがある。

- 高速性能
- 精度保証
- 性能可搬性
- 性能安定性

一般的には、最適化すべき性能は、あたえられたマシン環境のもとでのプログラム実行の高速性能である。この場合、最適化の評価基準として、プログラムの実行時間が用いられる。数値計算応用においては、速度に代わって計算精度の保証が性能の基準となる場合がある。評価対象とするマシン環境の範囲を広げ、それらの間で性能の可搬性が保たれることを評価基準にする場合も考えられる。さらに、問題パラメタの変化によってもたらされる、実行時間の変動の程度を性能安定性とよび、それを最適化基準とする場合もある [5]。その他、消費電力量、使い勝手のよさを評価基準とすることも可能である。

自動チューニングが対象とする応用ソフトウェアは、数値計算応用の分野に限らない。数値計算応用の分野を対象とした場合でも、数値ライブラリに対してのみ適用範囲を限定する訳ではない。関連研究の節で述べるように、今まで開発された自動チューニングの実装事例の多くが数値ライブラリであった [1,3,6,7,12,16,17]。その理由も同じく、もっとも適用が有効であったからに他ならない。数値ライブラリは、多種類のマシン環境で使用され、ソフトウェアとして枯れた存在であり、標準化されたインタフェースをもつ。さらに、新世代マシン環境への移植、すなわち、ソフトウェア資産継承の必要性が高い。狭義には、自動チューニングは、数値ライブラリ構築技術と捉えることも可能である。しかし、技術パラダイムとしての自動チューニングの枠組みは、もっと広義に解釈すべきである。

１．３　計算機科学における関連領域

自動チューニング技術と類似した計算機科学の分野として、コンパイラ、オペレーティングシステム（OS）、並列処理が存在する。静的な最適化写像を行うコンパイラでは、命令の先取り、ループ展開、タイリング、パイプライン化などの最適化技法が存在する。しかし、これらが適用されるのは最小のプログラム単位に対してであり、あたえられたプログラム（問題）全体から見れば、極めて局所的な部分最適化が行われるに過ぎない。一方、自動チューニングでは、もとのプログラムと同じ実行結果をもたらすことが保証される限り、最適化のために命令を全域的に並べかえることを許す。ただし、全域的な最適化を行うためには、部分的な最適化段階の適切な適用順序を決定しなくてはならない。あらゆる組み合せに従って適用順序を決める必要があり、結局は試行実験によってしか決定できない。さらに、自動チューニングでは、最適化の段階でアルゴリズム水準の置換えが可能である。自動チューニングは、コンパイラ最適化ではあつかえない問題についても、対応可能な枠組みを提供する。コンパイラとの重要な相違点として、後述するように、自動チューニングでは動的最適化写像、すなわち、プログラム実行時の最適化段階も有する。

自動チューニングは、OS の所持するスケジューリング機能とも関係する。OS のスケジューリングは、タスク、プロセス、スレッド、ジョブなどの制御単位に対して、システム全体のスループット、当該プログラムの優先度などの評価基準から、最適なマシン割当てを動的最適化写像として行う。自動チューニングでは、プログラムに明示的に現れない制御単位については、調整パラメタとは考えない。

並列処理においては、自動並列化によって、あたえられたプログラムを対象とする並列マシン環境にどのように写像すれば、最大限の性能をひきだすことができるかという問題がある。この問題は、多数のプロセッサを用いて高性能を獲得することを目的とする並列処理技術において、いわば、並列処理基本問題ともいうべきものである。この意味において、並列マシン環境を対象とする自動チューニングは、並列処理基本問題を解決する１つのアプローチといえる。並列処理における並列化コンパイラ、マルチスレッドのスケジューリングなどは、上述の自動チューニングとコンパイラ、OS との密接な関係を示す共通技術である。

2. 最適化写像の一般モデル

２．１　最適化写像の枠組みと要件

一般モデルとしては、最適化対象のプログラム（問題）は何であってもよい。プログラムをマシンに実行可能な形に写像する過程で、対象マシン上で当該プログラムが効率よく実行されるように変換、すなわち、最適化写像が行われる。最適化写像の適用局面を**図１**で示す。応用問題の事例は、数値計算、実世界シミュレーション、各種のメディア処理、データベース検索、ゲームなどを含む。マシン環境の事例は、それらの応用ソフトウェアが実行される多様なマシン環境をさす。

最適化写像において、対象プログラムをソースプログラムの形で入力し、ある性能基準に対して最適化されたソースプログラムが出力される。最適化写像の処理は、

問題が対象マシンに写像される過程に存在する。この過程は、最適化写像の各段階において、マシン環境上での処理によって進行する（図２）。このとき対象プログラムに最適化写像を記述した情報が、専用言語による注釈として付加される。写像機構をになう主体は、OS、ミドルウェア、コンパイラ、ライブラリ、プログラミング支援ツール、シミュレーションツールなど、写像過程に介在するシステムソフトウェアである。

最適化写像の要件として、以下のことが求められる。

１）軽量性
２）低オーバヘッド
３）実時間性（動的最適化の場合）
４）操作性
５）可搬性
６）言語記述性
７）有効性検証手段

最適化写像システムとして実現されるとき、システムが静的にも動的にもコンパクトで、軽量であることが求められる。入力プログラムやコンパイラへの負担が小さいこと、すなわち、最適化に要する空間的、時間的オーバヘッドが小さくなくてはならない。動的最適化を行う場合は、最適化処理に実時間性が要求される。システムには、最適化情報の付与などにおいてプログラマ（利用者）の介在が想定されている。その場合、操作性がよく使いやすいこと、かつ対象マシンに依存する部分が少ないこと、さらには、得られる性能に可搬性があることなどが求められる。プログラミング支援ツールとして具備される最適化写像の記述言語については、書きやすさ、読みやすさの観点から利用者インタフェースを考慮した記述性が求められる。さらに、最適化の有効性を評価し、検証する手段が用意されなくてはならない。

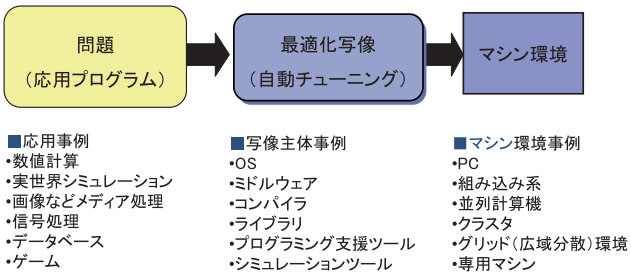

図１　最適化写像（自動チューニング）の適用局面

Fig.1 Optimized mapping (Automatic tuning) model

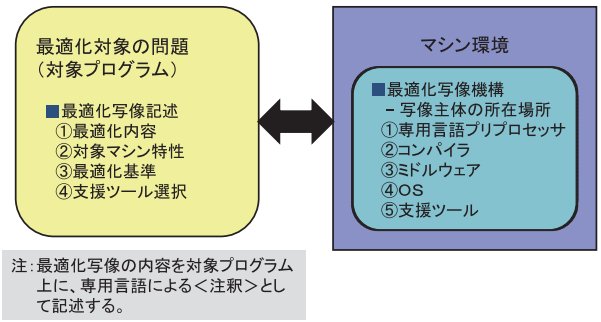

図２　最適化写像の記述とその機構の所在

Fig.2 Optimized mapping description and its target machine

２．２　最適化写像機構の構成

最適化写像機構は、以下の２つの基本となる機能から構成される。

１）最適化写像の判定
プログラマによって入力された情報、性能モニタによって得られた測定データ（プロファイル）、および知識ベースに置かれた情報を用いて、最適化基準に従って写像の是非を決定する。最適化の効果を予測し、調整パラメタ値あるいは選択対象を決定する。最適化基準関数は、プログラマによって、性能ポリシとしてあらかじめ入力される。

２）最適化写像の実施
最適化写像の有効性が予測されると、その最適化写像をあたえるパラメタ値が設定される。写像対象がアルゴリズム要素、あるいはプログラム部品

脚注：
自動チューニングという用語は、語感として普遍性、一般性をもつ。例えば、FM ラジオのチューナ、自動車のチューンナップ、楽器の音合せなどに対して、チューン（tune；調整、調律、調和）という言葉を用いる。三省堂国語辞典によると、「具合の悪いところを整える」という矮小化された部分修正の意味合いが説明されている。学術論文で用いられる英語表現には、tuning の他に、adjusting、adaptive、learning、self-organizing、harmonizing などがある。
技術用語として見た場合、＜自動チューニング＞には、技術の隙間を追究し、丁寧に磨き上げる感覚がある。科学というより、職人の技術領域を指し示す地味な用語である。本稿では、自動チューニング技術を、形式化という科学と有効性という工学の視点から、検討する。また、自動チューニングは対象独立な技術用語であり、チューニングの対象によって意味をもつ。例えば、シミュレーション、解析評価などと同様に、対象があたえられて語義が明確となる用語の１つである。チューニングする対象を明示して使用すべき言葉である。

である場合は、最適なものが写像対象として組み込まれる。最適化写像を実施するタイミングは複数存在する。最適化写像過程のそれぞれの段階で、可能な最適化の対象と適用される最適化手法が異なる。

最適化写像の過程で、あらかじめ用意されたあるいは同過程で追加された多様なアルゴリズム要素、プログラム部品が知識ベースとして参照される。最適化写像によって高性能を得るためには、性能データを測定しそれを用いて最適化を行う。すなわち、最適化写像判定と最適化写像結果の評価のために、対象プログラムの性能をモニタリングする機構が不可欠である。モニタリングは、周期的あるいは要求駆動的に行われ、所要の性能データが知識ベースに格納される。測定対象となるデータ（プロファイル）は、最適化写像判定に必要な実行時間、所要記憶量、通信速度、通信回数などである。

さらに、最適化写像において効率的な結果を得るためには、プログラマに対するプログラミング支援ツールが必要とされる。最適化写像機構への入力として、対象プログラムとともに、最適化に有用な情報があたえられる。同情報を記述する言語が必要であり、それを用いて最適化の内容、評価基準の指定などが行われる。入力される問題を記述したソースプログラムには、適宜、指示文が注釈として挿入される。指示文は、対象マシンのアーキテクチャ特性、最適化適用領域、最適基準などの最適化に必要な情報を示す。指示文つきの入力プログラムはプリコンパイルされ、所要の最適化機能を付加したソースプログラムに変換される。指示文の言語処理系（プリコンパイラ）は、最適化写像を支援するプログラミング環境であり、自動チューニング支援ツールといえる。

２．３　最適化写像のタイミング

入力されたソースプログラムを、最適化写像によって、最終的に対象マシン上で実行可能な目的プログラムに変換する。変換後、同マシン上で性能チューニングされた目的プログラムとして効率よく実行され、所期の結果を得る。この最適化写像過程で、さまざまな最適化が適用される。上流から下流に向けて、以下の順序で、最適化写像が実施される [7,11]。なお本稿では、コンパイラがコンパイル時に行う最適化写像、OS がプログラム実行時に行う動的スケジューリングによる最適化写像は含めない。

１）インストール時：主として、対象マシン依存の最適化
２）関数コール時：主として、問題サイズ依存の最適化
３）実行時：主として、負荷状況依存の最適化

インストール時は、例えば、数値ライブラリを新しいマシン環境に移植（インストール）するときの最適化である。主として、キャッシュサイズなどの対象マシンの特性パラメタを考慮する。インストール時では、対象マシン特性が指示文によってあたえられ、それに適合する最適化が行われる。関数コール時は、入力プログラムが確定して、問題サイズなどのプログラム変数が定数として処理可能なプログラム実行前の時点である。ライブラリ関数をコールするときに確定している引数のいくつかについて、最適化情報として利用できる。実行時自動チューニングでは、対象プログラムの実行が開始されたのち、性能モニタによって得られた動的な負荷状況などを考慮して最適化する。並列スレッド割り当て、プロセス移送、通信路選択など、動的な負荷状況プロファイルを利用する場合は、実行時最適化写像においてのみ可能となる。これらの各段階で、調整パラメタ、ブロック幅、データ配置、プログラム部品、アルゴリズム要素、処理粒度などが最適化の対象となる。どの段階でどの最適化を適用するかについては自由度があり、現在のところは、発見的方法で選択されている。

疎行列を扱う問題においては、実際にプログラムを実行するまで行列の構造が決定しない。このような問題に対しては、プログラムの実行時に自動チューニングを行う必要がある [14]。数値ライブラリの調整パラメタに対して、パラメタのとり得る値から複数の標本点を選択する。それらの標本点ごとの値を設定して数値ライブラリをコンパイルし、実行する。得られた実測データをもとに、実行時間を最小にする調整パラメタの最適値を推定する。実行時自動チューニングの特徴は、以下の通りである。

１）行列サイズなどの問題の規模、疎行列の構造などの問題の特性が確定しているので、調整パラメタの推定範囲を狭めることができる。
２）プログラムの実行時に行うため、自動チューニングの所要時間がそのままプログラムの実行時間の増加につながる。同所要時間の短縮が重要となる。

２．４　最適化写像手法の分類

最適化写像システムに用いられる手法を分類すると、以下の通りとなる。これらは、最適化写像の考え方を自動チューニング機構として実装するときの選択肢である。選択肢の多さは、自動チューニングの設計思想の多様性を示すものである。これらを取捨選択して、自動チューニング機構の設計方針が決定される。

１）調整パラメタ値選択型：最適化写像の過程で、あらかじめ指示文で与えられた調整パラメタについ

て最適値を決定し、選択する。

2）知識ベース型：最適化写像の過程で、プログラム部品、アルゴリズム要素の集合、性能データのプロファイルなどを知識ベースとして利用する。

3）実行時間予測型：対象マシンおよびプログラム実行をモデル化し、実行時間を予測し、それを最適化写像情報として利用する。

4）プロファイル利用型：対象マシン上での仮実行によりプロファイル（性能データ）を取得し、それを最適化写像の過程で利用する。

5）対話型と非対話型：最適化写像の過程（途中）で、プログラマが対話型に関与するか否かの区別である。障害発生時、異常事態での人の関与をどのように行うかは、システムの設計方針として重要である。

6）適応型と非適応型：最適化写像の過程で、手順の繰り返し適用をフィードバック的に行うか否かの区別である。性能モニタからのプロファイルをもとに、フィードバックの是非を判断する方式などがある。

7）動的と静的：最適化写像の過程での実行環境変化に対して、実時間対応を行う場合を動的、行わない場合を静的と区別する。

これらのうち、数値計算応用における最適化写像、すなわち、自動チューニングにおいて、主要な役割を果たす1）と2）について、説明を補足する。まず、調整パラメタ値選択型自動チューニングでは、マシンごとに異なるアーキテクチャ特性をパラメタ化する。例えば、プロセッサのキャッシュサイズを考慮して、行列のブロック幅を調整パラメタ化する。適切な調整パラメタ値の選択により、プログラムの実行速度が向上する。対象となるマシン特性は、データキャッシュサイズ、命令キャッシュサイズ、命令実行パイプラインの長さ、浮動小数点演算パイプラインの長さ、記憶容量などである。

つぎに、知識ベース型自動チューニングでは、1つの入力プログラムに対して、手作業で最適化したプログラムを複数、プログラム部品として用意する。チューニングでは、それらのプログラム部品の集合から、もっとも適切なものを自動的に選択する。また、1つの問題について複数の解法アルゴリズムが存在するとき、与えられたマシン環境にもっとも適切なものを選択する。これらの選択においては、あらかじめ測定された性能データや、性能モニタによる実時間情報などのプロファイルを知識ベースに格納しておき、最適化写像の過程でそれを参照して、最適化基準をもっとも適切に満足するものに決定される。

3.　数値計算応用を対象とした最適化写像

3.1　自動チューニングの対象となる数値ライブラリ

本章では、最適化写像の具体例として、数値ライブラリを対象とした自動チューニングを考える。2章では、一般的あるいは抽象的表現としての最適化写像という用語を使用したが、3章以降では、当該分野で一般に使用される用語である自動（性能）チューニングを用いる。対象プログラムを数値ライブラリとする理由は、以下の通りである。

1）現実的な必要性が高い。
2）自動チューニングの効果が期待できる。
3）有意義な研究成果が得られている。

数値計算応用の根幹をなす数値ライブラリの開発は、高性能化に高度な専門性を必要とするという理由から、誰もが容易に行える作業とはなっていない。現状では、チューニングのための調整パラメタの探索領域が膨大なものとなり、従来の手作業によるチューニング自体が困難な状況にある。このチューニング作業は、高度な専門性を有する人的資源を占有し、かつ時間を浪費する。このような現実的要請から、数値ライブラリを対象とした自動チューニング技術の確立が求められている。

後述するように、米国テネシー大学において、自動チューニング機能をもつ数値ライブラリ ATLAS[17] の開発が行われ、基本線形計算副プログラム BLAS（Basic Linear Algebra Subprograms）を対象として、その有効性が実証されている。数値ライブラリには、直接解法、反復解法、密行列解法、および疎行列解法などの解法が含まれる。既存の自動チューニングは、BLAS や高速フーリエ変換 FFT（Fast Fourier Transform）[4] などの解法の高速化に限定されており、数値計算応用における広範な問題を解決できていない。

一般に数値計算応用分野では、BLAS などの基本演算をもとにした階層（基本演算階層）での処理を、基本構成要素として包含する。数値計算応用における高性能化、高安定性の要請を考えると、対象を基本演算階層に限定しない、より高い演算階層での高性能化や高安定性を達成できる方式が必要となる。このような数値計算応用ソフトウェアの階層性を考慮するとき、従来の自動チューニング技術は、適用されているアルゴリズム階層が低水準に制限されているといえる。

3.2　数値ライブラリの自動チューニング機構

数値ライブラリに対する自動チューニング機能は、図3で示す2つの機構、調整パラメタの最適値を決定する

機構と最適アルゴリズムを選択する機構で実現される。プログラム実行の性能ポリシとしてプログラマによって入力されるデータには、最大実行時間、使用する最大記憶量、計算結果の要求精度、許容する性能安定性などがある。また、マシン環境の特性パラメタとして、記憶容量、キャッシュサイズ、プロセッサ数、通信性能などが入力される。出力は、選択された最適アルゴリズムと決定された調整パラメタの値である。

1）調整パラメタ値決定機構：最適な性能をあたえる調整パラメタを決定するために、最適な標本点を決定する標本点自動決定機構を内部にもつ。補間法を用いて自動的に標本点を決定する方式のほか、プログラマからの最適化基準定義関数の指定に基づき決定する方式がある。
2）最適アルゴリズム選択機構：調整パラメタ値決定機構によってあたえられた最適パラメタ値を利用し、最適なアルゴリズム要素を決定する。アルゴリズム要素の選択では、統計手法を用いた自動分類方式、λ式を用いた実装導出方式などを適用する。また、すべての可能性を全探索する方式をプログラマが指定した場合は、全探索により最適アルゴリズム要素を決定する。

これらのチューニングにおいて、プログラム実行中のマシン性能を、インストール時にモニタする必要がある。インストール時に、実行時間、キャッシュミス率、性能安定性などのデータを測定し、知識ベースに格納しておく。それを用いて、チューニングの効果を定量的に予測し、予測結果に基づいてチューニングの判定を行う。判定基準については、速度と精度、速度と記憶量、速度と記憶量と精度などのトレードオフが存在する。対象が並列マシンの場合、並列処理オーバヘッドに起因するプロセッサ数と速度のトレードオフがある。これらのトレードオフを勘案して、効果を予測し、調整パラメタ値およびアルゴリズム要素を決定し、チューニングを実施する。

数値ライブラリ向け自動チューニング機構のアルゴリズム選択の枠組みとして、アルゴリズム階層を利用する方法がある [10]。数値計算アルゴリズムに対して、次に示す３階層が設定される（図３）。

1）数値計算応用における各種ソルバ
2）各ソルバを構成する要素ソルバ
3）行列演算基本ルーチン

アルゴリズム階層ごとに、選択すべきアルゴリズム要素を定義し、その性能を決定する調整パラメタと有効範囲を指定する。最適アルゴリズム選択機構では、プログラマが指定した性能ポリシと、自動チューニングシステムが自動的に取得したマシン環境情報が入力され、各階層における最適なアルゴリズム要素が出力される。階層化されたアルゴリズム集合が置かれた知識ベースが参照され、同情報をもとに、実際に実行時間を測定しつつ、下位の階層から順に各階層での最適なアルゴリズム要素が決定される。

数値計算応用として、利用頻度の高い固有値ソルバと連立１次方程式ソルバをとりあげる。それぞれについて、要素ソルバまでをアルゴリズム階層化すると、以下のようになる。各ソルバに共通する行列演算基本ルーチンは、BLAS1、BLAS2、BLAS3 である。

1）固有値ソルバ
　①各種ソルバ：密行列ライブラリ、疎行列ライブラリ
　②要素ソルバ
　　●直交化ルーチン：Gram-Schmidt 法、QR 法、Givens 法
　　●相似変換ルーチン：Householder ３重対角化法、Householder 帯行列化法
　　●３重対角行列求解ルーチン：２分法＋逆反復法、Givens 法、QR 法、MRRR 法、分割統治法
　　●疎行列解法ルーチン：Arnordi 法
2）連立１次方程式ソルバ
　①各種ソルバ：密行列ライブラリ、疎行列ライブラリ
　②要素ソルバ
　　●密行列 LU 分解ルーチン：内積形式、外積形式、クラウト法
　　●帯行列 LU 分解ルーチン
　　●疎行列反復解法ルーチン：CG 法、GMRES 法
　　●前処理ルーチン：Jacobi 法、ILU 法

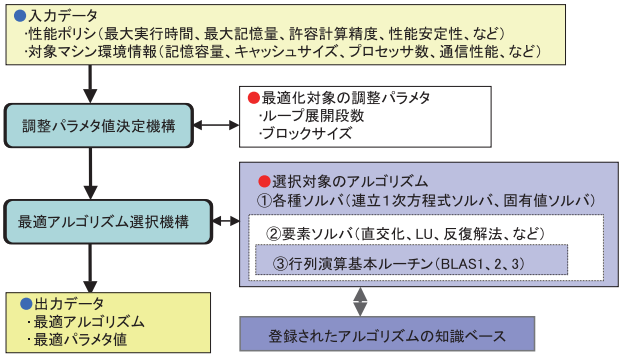

図３　数値ライブラリの自動チューニング機構

Fig.3 Automatic tuning mechanism for numerical library

３．３　自動チューニング機能をもつ数値ライブラリの開発と利用

３．３．１　自動チューニング機能をもつ数値ライブラリの開発

自動チューニング機能をもつ数値ライブラリの開発と利用に関して、数値ライブラリ開発者と開発された数値ライブラリの利用者の２つの立場が存在する。数値ライブラリ開発者は、自動チューニング機能ついての知識とスキルをもち、それを活用して自動チューニング機構を数値ライブラリに実装する。実装には、自動チューニング支援ツールに含まれる専用の記述言語が用いられる。プログラム中のチューニング対象部分、選択すべきアルゴリズム集合、調整パラメタ、最適基準定義関数、対象マシンの特性パラメタ、チューニングのタイミングなどの指定が、専用言語によってあたえられる。

図４に、数値ライブラリの開発者からみた開発手順を示す。作成された自動チューニング機能をもつ数値ライブラリでは、対象マシン環境は特定されていない。すなわち、この段階ではマシン独立な数値ライブラリである。その利用者が対象マシン環境情報（特性パラメタ値）をあたえ同マシンにインストールすることで、同マシンに最適化された数値ライブラリが実現される。図５に、最適基準定義関数、対象マシンの特性パラメタ、選択されるアルゴリズム候補の設定を、専用言語 ABCLibScript[8] で記述した例を示す。図６に、調整パラメタとしてループ展開変数とその段数を指定する固有値計算プログラムの一部を示す。同プログラムにおいて、!ABCLib$ で始まる文は、コンパイラは注釈として処理する。最適基準定義関数および性能モニタリングすべき標本点情報、最適化の対象領域（region start から region end まで）が、専用言語 ABCLibScript を用いて記述されている。さらに、これらの情報を用いた最適化が、インストール時に行われることが指定されている。

専用言語によって記述された自動チューニング機能が、プリコンパイラによって変換されて自動チューニング機能付きのソースプログラムとなる。これが、マシン独立な数値ライブラリとして登録される。ライブラリの使用者は、同ライブラリのインストール時に最適化情報をあたえることで、対象マシンに最適化された数値ライブラリを得ることができる。図７に、プリコンパイラが生成した最外殻ループの展開候補の１部を示す。ここで、ループ展開段数を表わす変数 j_unroll は調整パラメタであり、インストール後、使用者が同ライブラリを利用するときに自動的に最適値が決定され、最適なプログラム部品が選択される。

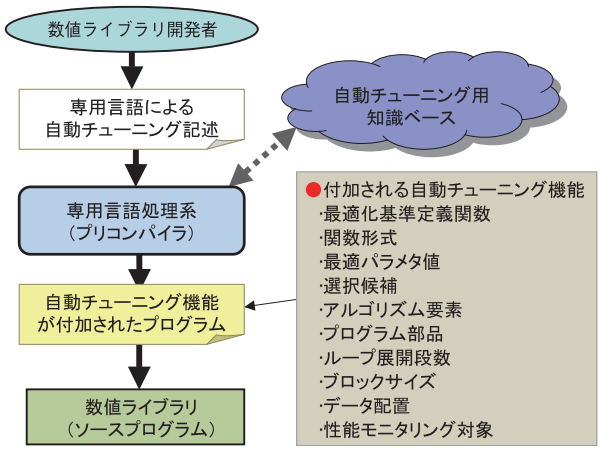

図４　数値ライブラリの開発者の立場からみた開発手順

Fig.4 Implementation process of numerical library with automatic tuning features

```
!ABCLib$  static  select region start
!ABCLib$  parameter (in CacheS, in NB, in NProc)
!ABCLib$     select sub region start
!ABCLib$     according estimated
!ABCLib$         (2.0d0*CacheS*NB)/(3.0d0*NProc)
                     Algorithm 1
!ABCLib$     select sub region end
!ABCLib$     select sub region start
!ABCLib$     according estimated
!ABCLib$         (4.0d0*CacheS*dlog(NB))/(2.0d0*NProc)
                     Algorithm 2
!ABCLib$     select sub region end
!ABCLib$   static select  region  end
```

注：!ABCLib$は、自動チューニング機能を記述する専用言語による記述子である。注釈として解釈される。

図５　専用言語によるマシン環境パラメタと最適基準定義関数の記述

Fig.5 Description of machine characteristics and cost definition function

```
!ABCLib$  install  unroll (j) region start
!ABCLib$  varied (j) from 1 to 16
!ABCLib$  fitting  polynomial 5 sampled (1-5, 8, 16)
    do j=0, local_length_y-1
       tmpu1 = u_x(j);    tmpr1 = mu*tmpu1-y_k(j)
    do i=0, local_length_x-1
       A(i_x+i, i_y+j) = A(i_x+i, i_y+j)+u_y(i)*tmpr1-x_k(i)*tmpu1
    enddo  enddo
!ABCLib$  install  unroll (j)  region end
```

注：　EigenSolverプログラムの１部分である。
Unroll(j):　ループ展開変数、jは段数
Install:　　インストール時の指定

図６　チューニング領域と最適基準定義関数の記述

Fig.6 Description of automatic tuning region and cost definition function

```
if (j_unroll .eq. 1) then
      {the "as is" code}
endif
if (j_unroll .eq. 2) then      /* j is even.*/
   do j=0, local_length_y-1, 2
        tmpu1 = u_x(j);      tmpr1 = mu*tmpu1-y_k(j)
        tmpu2 = u_x(j+1);   tmpr2 = mu*tmpu2-y_k(j+1)
   do i=0, local_length_x-1
       A(i_x+i, i_y+j) = A(i_x+i, i_y+j)+u_y(i)*tmpr1-x_k(i)*tmpu1
       A(i_x+i, i_y+j+1) = A(i_x+i, i_y+j+1)+u_y(i)*tmpr2-x_k(i)*tmpu2
     enddo  enddo
endif
  …
```

注：ループ展開段数を示す変数j_unrollは関数コール時、実行時に確定する。

図７　最外殻ループを展開して生成されたプログラム部品（部分）

Fig.7 Program components generated by unrolling outer-most loop

３．３．２　自動チューニング機能をもつ数値ライブラリの利用

数値ライブラリ利用者（プログラマ）は、数値計算応用プログラムを作成するとき、数値ライブラリ開発者が用意した自動チューニング機能をもつ数値ライブラリを利用する。プログラマは、対象マシンの特性情報、問題サイズなどの数値計算応用プログラムおよびそれを実行するマシン環境の情報を保持している。これらの情報を性能ポリシとして入力することで、実行時間の最小化などの性能の自動チューニングが行われる。自動チューニングが適用されるタイミングは、先に述べたように、対象マシンへのライブラリのインストール時、プログラム実行前の関数コール時、さらに同プログラムの実行時の３段階存在する。プログラマは最適化が有効なタイミングを指定する。

図８で示すように、ライブラリ利用者（プログラマ）は、まず自動チューニング機能をもつ数値ライブラリを、自分のマシン環境に最適化してインストールする。出力されたライブラリは、性能ポリシによって定まる範囲で対象マシンに最適化されている。このインストール時自動チューニングの過程で、ループ展開段数、ブロック幅、アルゴリズム要素、プログラム部品などが適切に選択され、それらを含むソースプログラムが生成される。同ソースプログラムは、その後の関数コール時、実行時自動チューニング機構への入力となる。変換過程では、自動チューニングシステムが管理する知識ベースに置かれる種々の選択候補、プロファイリングされた性能データなどが利用される。

関数コール時自動チューニングでは、行列サイズ、ループ回数など問題の特性を示すプログラム変数が、利用者によって定数としてあたえられたあとのプログラムに対してチューニングを行う。例えば、プログラマがあたえたプログラムの問題サイズに基づき、最適化候補の中から適切な選択が行われる。この過程で、必要に応じて知識ベースに置かれたアルゴリズム要素、プログラム部品、プロファイルなどが参照される。その後、コンパイルが行われ、数値ライブラリと結合後、目的プログラムとして実行される。実行時自動チューニングでは、入力されたデータの内容によってきまる特性（例えば、密行列、疎行列、帯行列など）、対象マシンおよび通信網の動的負荷状態などの実行時情報を考慮して最適化を行う。これら２つの自動チューニングについて、性能モニタリング情報を含む知識ベースが適宜利用される。

適切な最適化を行うためには、プログラマの介在が不可欠である。自動チューニング機能をもつ数値ライブラリの利用者（プログラマ）は、与えられた問題に対して、もっとも効率よく計算結果を得るために、以下の手順で問題解決を行う。この手順を支援するツールとして、自動チューニングのためのプログラミング支援ツールが用意される。

１）対象プログラムの記述
２）対象プログラムの性能に影響を及ぼす調整パラメタ、プログラム部品およびアルゴリズム要素の集合を抽出
３）最適化基準をあたえる関数を定義
４）同関数を最小とする調整パラメタ値、プログラム部品、アルゴリズム要素を選択
５）得られた調整パラメタ値、プログラム部品およびアルゴリズム要素を用いて対象プログラムを実行

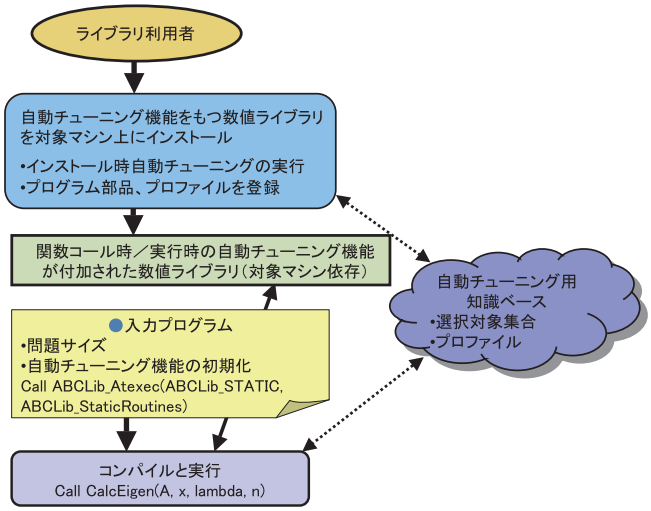

図８　自動チューニング機能つき数値ライブラリの利用

Fig.8 Use of numerical library with automatic tuning features

３．４　自動チューニングのためのプログラミング支援ツール

自動チューニング機能をもつプログラム（数値ライブラリを含める）を作成するとき、プログラミング支援ツールが必要である（図９）。自動チューニング記述専用言語は、プログラムに注釈のかたちで、自動チューニング機構への制御情報をあたえる。同専用言語 ABCLibScript によるプログラム記述の例は、図５、図６に示した。ABCLibScript の言語処理系（専用言語プリコンパイラ）は注釈を前処理して、必要な自動チューニング用ソースプログラムをもとのソースプログラムに埋め込む。ABCLibScript の現在の版では、並列マシン環境に特化したものとなっており、Fortran90、MPI プログラムの生成を前提としている。なお、ABCLibScript を用いて、密な対称実数行列を対象とする固有値ソルバ用数値ライブラリが開発されている。ABCLibScript の設計方針は、以下の通りである [8]。

１）数値計算に特化した自動チューニング指定
２）プログラマが注釈形式で自動チューニング内容を記述
３）注釈を前処理しコンパイラに渡すプリコンパイラ

その他の自動チューニングのプログラミング支援ツールとして、専用言語入力用テキストエディタ、最適化支援 GUI、有効性評価のための測定環境などが必要である。これらの支援ツールが整えられてはじめて、使いものになる自動チューニング利用環境が構築されたといえる。

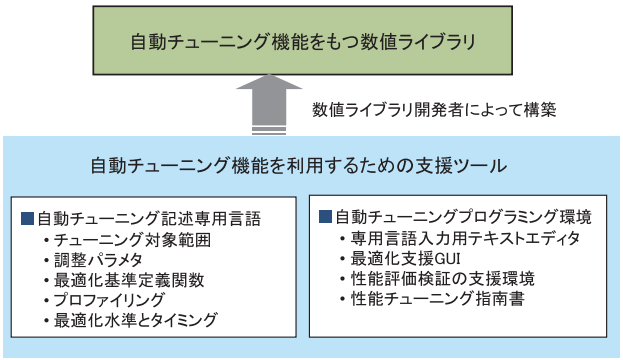

図９　自動チューニング機能を利用するための支援ツール

Fig.9 Development support system for automatic tuning programs

４．　数値計算応用における自動チューニングの有効性

４．１　評価マシン環境

数値ライブラリを素材として、自動チューニングの有効性を示す２つの適用実験の結果を示す。合せて、自動チューニングの適用が求められる性能安定化の実験結果を示す。これら３つの実験に用いたマシン環境は、以下の通りである。

●実験１　Hitachi SR8000/MPP: 16 nodes（128PE）
●実験２　Hitachi SR8000/MPP: 1 node（8PE）
　　Fujitsu VPP800/63: 8 nodes（8PE）
　　Intel PC Cluster: Intel Pentium4, 2.0GHz × 4PE
●実験３　Hitachi SR8000/MPP: 1 node（8PE）

日立製作所の Hitachi SR8000/MPP については、東京大学の情報基盤センターに設置されているマシンを使用した。同マシンの仕様は、以下の通りである。

●ノード数：144 nodes (8PE/node)
●記憶容量：16GB/node
●計算速度：14.4GFlops/node
●通信速度：3.2GB/sec/switch

富士通の Fujitsu VPP800/63 は、京都大学学術情報センターに設置されているマシンを使用した。同マシンの仕様は、以下の通りである。

●ノード数：63 nodes (1PE/node)
●記憶容量：8GB/node
●計算速度：8GFlops/node (Vector), 1GFlops/node (Scalar)
●通信速度：3.2GB/sec/switch

４．２　アルゴリズム選択の効果

実験１では、ベンチマークとして 10,000 次元フランク行列の固有値ソルバを用いる。６種類のよく知られた解法アルゴリズムを、並列マシン環境 Hitachi SR8000 で実行する。図 10 に、プロセッサ（PE）数を 8、16、32、64、128 と変化させた場合について、各アルゴリズムの実行時間と直交精度を示す。横軸はアルゴリズムの種類、縦軸は実行時間と直交精度を示す。

アルゴリズム NotOrt（非直交化）は他にくらべ、実行時間は短いが精度が極端にわるい。プログラマが性能ポリシとして指定した要求精度に達していないので、自動チューニングにおいて選択されることはない。他の５種類については、アルゴリズムによって実行時間が７～20 倍の差異がある。つまり、適切なアルゴリズム選択機能をもつ自動チューニングによって、最適化がはかられる余地が大きい。プロセッサ数によっても実行時間は異なるが、必ずしもそれが多い方が小さい訳ではない。例えば、プロセッサ数 128 では、64 の場合にくらべて性能が低下している。これはあたえた問題サイズが、プロセッサ数 128 に対して小さすぎるからであると考えられる。実行する問題サイズおよび使用するマシン環境によって、適

切なプロセッサ数とアルゴリズムが存在することを示している。

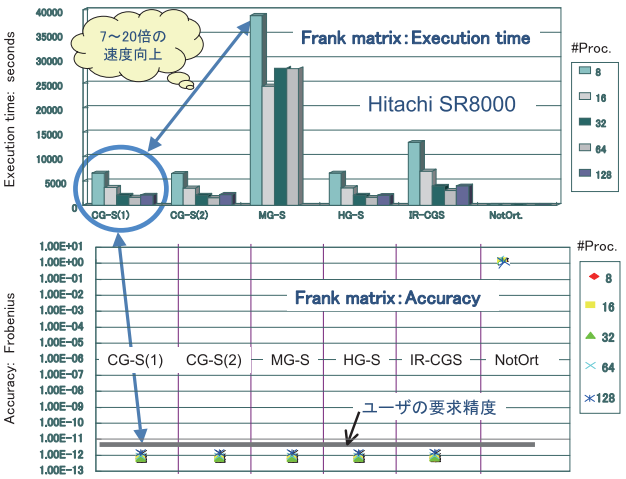

図 10　フランク行列の固有値計算におけるアルゴリズム選択の効果

Fig.10 Effect of algorithm selection in Frank matrix eigensolver

本実験結果によって、インストール時自動チューニングは、多くの場合で非常に有効であること明らかとなった。関数コール時自動チューニングについては、インストール時ほどではないが、事例によっては有効であることがわかる。

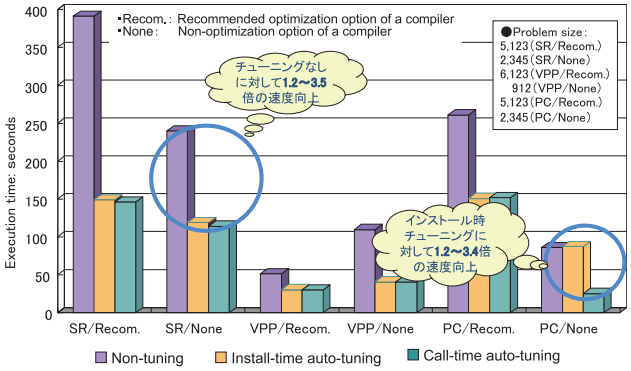

図 11　3 つのマシン環境（SR/VPP/PC）におけるループ展開の効果

Fig.11 Effect of loop unrolling potimization on three machine environments

4．3　ループ展開の効果

実験 2 では、3 種類のマシン環境について、行列計算におけるループ展開の効果を測定した [7]。図 11 に示すように、実験では、①自動チューニングを適用しない場合、②インストール時のみ適用した場合、③インストール時に加えて関数コール時にも適用した場合について、実行時間を測定した。ループ展開による最適化は、通常のコンパイラの最適化オプションに含まれているので、それを適用した場合としない場合について、自動チューニングによる最適化の効果を調べた。問題サイズは図に示す通りである。

3 種類のマシン環境において、自動チューニングを適用した場合は、適用しない場合にくらべて 1.2 ～ 3.5 倍の高速化が得られている。PC クラスタを除くマシン環境では、自動チューニングの効果はインストール時最適化であり、関数コール時のそれはそれ程有効ではない。PC クラスタでは、関数コール時自動チューニングにより、3.4 倍の高速化が得られている。SR8000 や VPP800 のようなスーパコンピュータでは、インストール時最適化により対象マシンに適応させる効果は大きい。しかし、関数コール時の問題サイズを考慮したループ展開については、並列化コンパイラの能力が高いためそれ程有効ではない。スーパコンピュータの並列化コンパイラに対して、PC クラスタの場合は MPI 並列化ライブラリを用いているので、関数コール時に行うループ展開段数の最適化の効果が増す。コンパイラの最適化オプションの効果については、SR8000 は指定なしの方が、VPP800 は推薦指定の方が実行時間は小さい。コンパイラの最適化能力は、機種および問題により分散が大きいことが経験的に知られている。

4．4　性能安定化の期待

図 12 に、行列積の行列サイズを変化させ、それによって速度性能がどのように変化するかを計測した結果を示す [5]。マシン環境は Hitachi SR8000 である。横軸は行列サイズ、縦軸は MFLOPS を単位とする速度性能である。ループ展開の段数を 3 ～ 15 まで変化させている。この実験から、つぎのことが観察される。

1）行列サイズ 4,096 近傍で、速度性能が大幅に低下する。
2）ループ展開段数 3 の場合は、行列サイズによる速度性能の大きな変動はない。
3）ループ展開段数ごとに速度性能が異なる。

これらは、キャッシュの衝突がもたらすスラッシング現象による。行列サイズが4,032のときの衝突回数は、ループ展開段数が 3 の場合で 6,628K 回、15 の場合で 2,796K 回発生している。行列サイズが4,096のときは、同じくループ展開段数が 3 の場合で 4,233K 回、15 の場合で 2,704M 回発生している。4,096 近傍での大幅な性能低下は、同近傍で衝突発生頻度が 3 桁増加しているのが原因である。ループ展開段数 3 の場合は、衝突発生頻度は減少しており、4,096 近傍ではもっともよい速度性能をあたえる。すなわち、行列サイズによる速度性能の変動が少ない安定した性能が得られている。キャッシュの衝突頻度が変わらない範囲で、ループ展開段数の大きい方が高速である。しかし、性能が著しく低下する状況があるので、それを自動的に回避し性能の安定性を確保する自動チューニングが必要とされる。

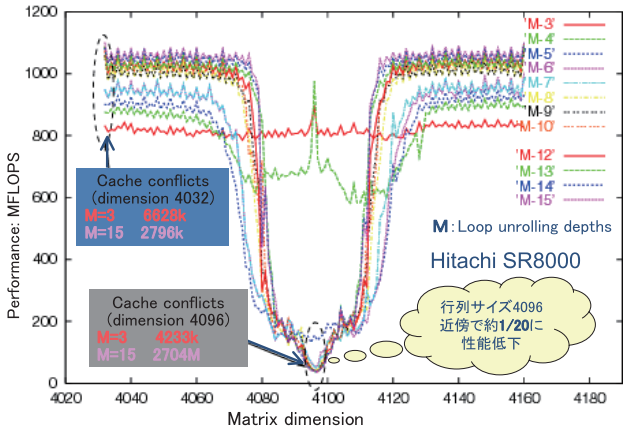

図 12　キャッシュスラッシングがもたらす性能の不安定性

Fig.12 Performance instabilization caused by cache thrashing

5.　おわりに

5.1　関連研究

自動チューニング技術の研究は、10 年程前から米国を中心として取り組みが活発化している。これまでに実現されている自動チューニング技術の適用は、多くの場合、スーパコンピュータなど高性能計算機環境における数値計算応用に限定される。自動チューニング機能により、数値計算応用における調整可能な性能パラメタの最適化を行う。それらの中で、2001 年米国テネシー大学で開発された ATLAS（Automatically Tuned Linear Algebra Software）[17] がもっとも有名である。ATLAS は基本線形計算副プログラム（BLAS）を最適化する。しかし、ATLAS のパラメタ選択方式は、BLAS 以外には適用できない。

高速フーリエ変換（FFT）を自動チューニングするソフトウェアとしては、1999 年米国 MIT で開発された FFTW（Fastest Fourier Transform in the West）[4] が著名である。高速フーリエ変換（FFT）を対象としている。FFT の基数についてのプログラム部品（要素演算）を選択する。FFTW は汎用的なパラメタ分類方式の枠組みがなく、FFT 以外の処理に適用できない。

疎行列反復解法を対象として、2000 年東京大学で開発された自動チューニング機能をもつ数値ライブラリ（Intelligent Library; ILIB [6]）がある。ILIB は、同解法の前処理アルゴリズムを自動選択する。アルゴリズム選択に際しては、想定されるすべての状況に対して、候補となるプログラム部品、アルゴリズム要素を準備しておく。ILIB は、自動チューニングの汎用的な枠組みを考慮していない。

自動チューニング機能をもつ疎行列計算ライブラリとして、2002 年米国カリフォルニア大学バークレイ校で開発された Optimized Sparse Kernel Interface (OSKI) [16] がある。OSKI は、疎行列ベクトル積のレジスタとキャッシュの調整パラメタを自動チューニングできる。しかし、OSKI のパラメタ分類方式は経験的手法に基づいており、汎用的な処理に適用できる方式ではない。

電気通信大学では、2003 年に、汎用的に性能に影響をおよぼす調整パラメタをモデル化し、自動チューニングを行う一般的枠組み FIBER[7] を提案している。その中で、自動チューニング記述用専用言語 ABCLibScript[8] は、数値ライブラリ開発者がライブラリ中に指示文を注釈として埋め込み実装の多様性を記述することによって、ライブラリに自動チューニング機能を付与する。この指示文に従って、ライブラリを新しいマシン環境にインストール時あるいはライブラリ関数コール時に、最適な調整パラメタ、プログラム部品、アルゴリズム要素などの選択が自動的に行われる。現状では、選択可能なアルゴリズム集合が小さい範囲に限定されているなどの問題点がある。

5.2　研究課題

プログラムの最適化写像、数値計算応用に向けた自動チューニングに関する研究課題は、以下の通りである。

1）最適化写像の観点から、コンパイラの最適化、OS のスケジューリングなどにおける最適化の枠組みをとらえ直す。
2）自動チューニング、コンパイラ、OS などそれぞれが担う最適化写像機能を、全体最適化の観点から再配置する。
3）要素分割された最適化写像機能の適用順序について、全体最適化を行う。
4）最適化基準とその組み合わせの多様化をはかる。
5）プロファイルに対して、統計的手法を用いて最適パラメタ値の推定を行う。
6）自動チューニング手法とコンパイラ最適化手法を統合化する。
7）プログラマの介在を絶った完全自動チューニングを実現する。
8）数値計算応用以外のソフトウェアについて、自動チューニングパラダイムの適用を拡大する。

これらの研究課題について、取り組みの戦略を考えねばならない。現在までに提案されている自動チューニング技術の枠組みには、まだ決定版がない。枠組みは、プログラミング言語がそうであるように、多数、存在してよい。その中からよいものが残り、標準化への道をたどればよい。数値ライブラリを素材として、自動チューニング研究の展開を考えてみる。今後、以下の過程を経て、情報社会における有用な基盤ソフトウェアとなることが

期待される。

1）自動チューニング機能をもつ数値ライブラリの枠組みを提案し、モデル上でその有効性を評価する。
2）提案した枠組みに基づき、数値ライブラリの作成支援システムを構築する。
3）同支援システムを利用して自動チューニング機能付き数値ライブラリを構築し、その有効性を評価する。1から3までの過程を繰り返し、枠組みと実装のチューニングを行う。
4）同数値ライブラリの普及をはかり、標準化を行う。

5.3　将来展望

自動チューニング技術の将来を展望する。今後の研究開発において考慮すべき観点は、以下の通りである。

1）自動チューニングパラダイムの適用範囲を拡大する。
2）自動チューニング技術の実用化を推進する。
3）ソフトウェアの生産性をたかめる技術との統合をはかる。
4）自動チューニング技術の標準化とオープンソースソフトウェア化を進める。

自動チューニング技術は、従来、その適用分野として数値計算応用を想定してきた。しかし、考え方の枠組みとしては、数値計算分野以外の応用に対して適用可能である。その意味において、自動チューニング技術の研究は次世代基盤ソフトウェアの構築に向けた重要な研究の1つといえる。今後、自動チューニングパラダイムの適用が有効な結果をもたらす分野を積み上げる努力が必要である。高性能マシン環境に対する数値ライブラリから始まり、大規模科学技術計算応用、最終的にはシステムインテグレーション企業が対象とする大規模ソフトウェアシステムへの適用が期待される。ライブラリのように閉じたソフトウェアでない場合は、セキュリティ、障害対策などチューニングすべき課題の複雑度が増す。その場合、＜自動＞の程度に幅をもたせ、人が介在する部分を内包する形で進展が図られねばならない。パラダイムの適用範囲の拡大によって、情報技術インフラストラクチャに多大の貢献がもたらされる。

現状では、数値計算応用に限定しても、現場からの要求を高いレベルで満足する自動チューニングの実用的な枠組みは存在しない。一度開発したソフトウェアを長期間にわたって利用するためには、ソフトウェア開発過程と自動チューニング過程が連動して最適化を行い、マシン環境の変化に柔軟に対応できる完成された自動チューニング方式が要請されている。多様化するマシン環境に、柔軟に適応する自動チューニング技術の実用的な枠組みを構築しなくてはならない。

高性能化するマシン環境の急速な進歩にあわせて、システムソフトウェア、応用ソフトウェアが開発されねばならない。ソフトウェア技術者の不足が叫ばれる中で、自動チューニングパラダイムは、ソフトウェア構築の生産性を高めるものとして期待される。ソフトウェア工学におけるソフトウェア部品化とその最適組合せによるソフトウェア構築方法論は、自動チューニングの対象領域である。今後、マシンの技術革新に対して柔軟な適応能力をもつソフトウェアが構築され、それにより効率的で高度な高性能コンピューティング環境が容易に実現できることが期待される。

自動チューニング技術は、新しく開発するマシンの機種ごとにライブラリを新しく作りなおす手間を削減するという意味において、産業技術的観点からも重要である。高性能コンピューティング技術の分野で、外国企業との競合に勝ちぬくために不可欠な、国際戦略性の高い情報基盤技術と位置づけることができる。そのためには、自動チューニング技術の標準化に力をいれる必要がある。合せて、オープンソース化によって、多くの人たちの知の集積が企図されねばならない。これらは多様化するマシン環境それぞれについて求められている。

最後に、「自動チューニング研究会」の活動について、言及しておく。新しい技術パラダイムの検証には、同じ目的に対して多様なアプローチをもって取り組むことが不可欠である。そのためには志を共有する研究者の議論の場が必要であるとの認識のもとに、電気通信大学、東京大学、名古屋大学、京都大学、筑波大学、理化学研究所、日立製作所の研究者約10名が、2003年から、「自動チューニング研究会」の活動を開始している。研究会の開催頻度は、現在のところ、2ヶ月に1度程度である。研究会では、各自の研究成果や国際会議参加報告などの議論の他に、外部研究費獲得のための研究課題の調査を進めている。現在は特定の学会との結びつきはないが、いずれ正式の学会研究会になると予想される。同研究会のもとで、2007年9月、昨年に引き続いて東京大学において第2回国際ワークショップiWAPT（International Workshop on Automatic Performance Tuning）を開催し、50名の参加を得た。今後のさらなる発展が期待される。

謝辞

本稿の執筆に際して、自動チューニング研究会のみなさまに、多くの示唆をいただいた。とくに、当該分野の開拓者の1人である片桐孝洋東京大学准教授（前電気通信大学大学院情報システム学研究科助手）は、電気通信大学における研究・教育のよき同僚であった。今村俊幸電気通信大学准教授、須田礼二東京大学准教授は、上記

研究会の中心メンバとして、絶え間ない研究の刺激をいただいた。直野健氏、田中輝雄氏はともに株式会社日立製作所社員の立場で、自動チューニング技術に関する学位論文をまとめ、筆者の所属した電気通信大学大学院情報システム学研究科から博士（工学）の学位を取得された。これらのみなさまとの議論の中で、自分なりの自動チューニング技術の枠組みを考え、解説論文としてまとめた。なお、4 章で示した実験データは、片桐孝洋氏、今村俊幸氏から提供いただいた。あらためて、常日頃の研究交流に感謝したい。

参考文献

1. Fran Berman, Rich Wolski, Silvia Figueira, Jennifer Schopf and Gary Shao. Application-level scheduling on distributed heterogeneous networks. Proc. Int. Conf. Supercomputing, IEEE/ACM, 1996.
2. J. Blimes, K. Asanovic, C.-W. Chin and J. Demmel. Optimizing matrix multiply using PHiPAC: A portable, high-performance, ANSI C coding methodology. Proc. Int. Conf. Supercomputing, IEEE/ACM, pp.340-347, 1997.
3. Jack Dongarra and Victor Eijkhout. Self-adapting numerical software for next generation applications. Journ. High-Performance Computing and Applications, Vol.17, No.2, pp.125-131, 2003.
4. Matteo Frigo. A Fast Fourier Transform compiler. Proc. ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI), pp.169-180, 1999.
5. Toshiyuki Imamura. C-Stab: Cache stabilizing algorithm for a numerical library. Proc. Int. Conf. Parallel and Distributed Computing and Networks (IASTED), pp.638-643, 2005.
6. 片桐孝洋 , 黒田久泰 , 大澤清 , 金田康正 . ILIB: 自動チューニング機能付き並列数値計算ライブラリとその評価 . 並列処理シンポジュウム（JSPP）論文集 , pp.27-34, 2000.
7. Takahiro Katagiri, Kenji Kise, Hiroki Honda and Toshitsugu Yuba. FIBER: A generalized framework for auto-tuning software. Proc. Int. Sympo. High-Performance Computing (ISHPC), Springer, LNCS 2858, pp.146-159, 2003.
8. Takahiro Katagiri, Kenji Kise, Hiroki Honda and Toshitsugu Yuba. ABCLibScript: A directive to support specification of an auto-tuning facility for numerical software. Parallel Computing, Elsevier, Vol.32, No.1, pp.92-112, 2006.
9. Ken Naono and Toshiyuki Imamura. An evaluation towards automatically tuned eigensolvers, Proc. 5th Int. Conf. Large-Scale Scientific Computations (LSSC), Springer, LNCS 3743, pp.422-429, 2005.
10. Ken Naono. Automatic tuning for high performance implementation of eigenvalue computations, 電気通信大学・博士（工学）学位論文 , 2006.
11. 直野健 , 猪貝光祥 , 木立啓之 . 行列計算における自動チューニング研究動向について , 情報処理学会研究報告 , 2006-HPC-107, pp.181-186, 2006.
12. M. Püschel, B. Singer, J. Xiong, J. M .F. Moura, J. Johnson, D. Padua, M. M. Veloso, and R. W. Johnson. SPIRAL: A generator for platform-adapted libraries of signal processing algorithms, Journ. High Performance Computing and Applications, Special Issue on Automatic Performance Tuning, Vol.18, No.1, pp.21-45, 2004.
13. Randy L. Ribler, Huseyin Simitci and Daniel A. Reed. The Autopilot performance-directed adaptive control system. Journ. Future Generation Computer Systems, Elsevier, Vol.18, No.1, pp.175-187, 2001.
14. 田中輝雄 . 数値計算ライブラリを対象としたソフトウェア自動チューニングにおける性能パラメタ推定法に関する研究 , 電気通信大学・博士学位（工学）論文 , 2007.
15. Cristian Tapus, I-Hsin Chung and Jeffery K. Hollingsworth. Active Harmony: Towards automated performance tuning, Proc. High Performance Networking and Computing Conference, Baltimore, 2002.
16. R. Vuduc, J.W. Demmel, K.A. Yelick, S. Kamil, R. Nishtala and B. Lee. Performance optimizations and bounds for sparse matrix-vector multiply, Proc. Int. Conf. Supercomputing, IEEE/ACM, Baltimore, 2002.
17. R. Clint Whaley, Antoine Petitet and Jack Dongarra. Automated empirical optimizations of software and the ATLAS project, Parallel Computing, Elsevier, Vol.27, pp.3-35, 2001.