2006－HPC－107（31）
2006／8／1

行列計算における自動チューニング研究動向について

直野健，（株）日立製作所中央研究所
猪貝光祥，木立啓之，（株）日立超 LSI システムズ

要旨

高性能計算機のアーキテクチャが複雑になったため，高性能計算機向けに行列計算をチューニングするコストが多大なものになっている．そのコストを緩和するため，1990 年代から自動チューニング方法が行列計算向けに研究されている．本稿の目的は，代表的な自動チューニング方法について動向をまとめることである．自動チューニング方法を分類し，動向を分析するために，行列ソフトウェア階層とソフトウェア開発サイクルの視点を用いた分析を行った．その結果，自動チューニング方法はより抽象度の高いソフトウェア階層へと進展しつつあることが分かった．また，今後の課題として，ソフトウェア開発サイクルにおける性能評価データがより多く必要であることが分かった．

Research Trends on Automatic Tuning Methods for Matrix Computations

Ken Naono, Central Research Laboratory, Hitachi, Ltd.
Mitsuyosi Igai, Hiroyuki Kidachi, Hitachi ULSI Systems Corp.

Abstract

Cost of developing well-tuned matrix computation programs for high performance computers is increasing enormously because of complexity in the computer architectures. To alleviate the cost, the AT, automatic tuning methods, for matrix computations have been researched since 1990s. The aim of this paper is to review the automatic tuning methods. Two views of matrix software hierarchy and of software development cycle are used in order to classify the methods and to exploit the ongoing research trends. With the views, it is found that the AT research will approach the areas of higher matrix software semantics and that more data of evaluations will be necessary for software development cycle.

1． はじめに

行列計算プログラムのチューニングにおける自動化が必要になってきている．計算機アーキテクチャが複雑化し，高い性能を引き出すためのプログラムの書き方が複雑になり，チューニングに多大の工数がかかるようになったためである．このため，性能チューニングを自動化する，いわゆる自動チューニング方法の研究が盛んになりつつある．

しかし，行列計算向けの自動チューニングについて，以下の課題がある．第 1の課題は，性能チューニングの定義と，他の類似技術，たとえばコンパイラの性能チューニングとの差が不明瞭になっている点である．第 2の課題は，自動チューニング技術が盛んになりつつあるものの，発展性の方向が明確ではない点である．

そこで，本研究の目的は，行列計算の性能チューニングにおける2つの視点を導入し，行列計算における自動チューニング方法の位置付けを明確にすることである．また，従来の代表的な6つの行列計算向け自動チューニング研究を上記の 2つの視点から整理し，本研究分野の動向と今後の課題を述べる．更に，従来研究の課題の 1つを解決するため提案した数値計算ポリシーという新しいフレームワーク[1]について紹介する．

2． 自動チューニングの定義とチューニングにおける2つの視点

2.1 自動チューニングの定義

Fig. 1に固有値計算の一般的プロセスを示す．第一に，実行したい固有値計算のアルゴリズムを選択する．例えば，ハウスホルダ法というアルゴリズムを選ぶ．次に，そのプログラムを記述し，そのプログラムをコンパイルして実行オブジェクトに変換する．そして，実行オブジェクトを計算機上で実行し，固有値計算の結果を得る．このプロセスでは，定義すべき組み合わせの数は非常に多く，様々なアルゴリズムや記述方法から選択しなければならない．特に最近の高性能計算機上では，それぞれの実行オブジェクトの性能が大幅に変わるため，調整すべきパタンの数は非常に多くなる．

本論文では，自動チューニングを以下のように定義する[1]．

「自動チューニングとは，計算内容から実行オブジェクトの最適な写像の自動化である」．

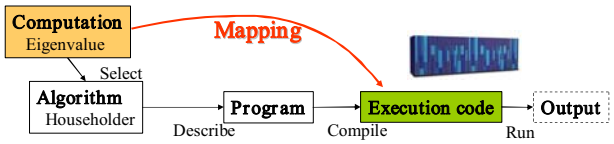

Fig.1 Tuning Process as Mapping.

2.2 チューニングにおける2つの視点

チューニングするプロセスの詳細を分析するため，本論文では，2つの重要な視点を提案する．ひとつの視点は行列計算におけるソフトウェア階層の視点であり，もうひとつの視点は，ソフトウェア開発サイクルの視点である．

Table 2にソフトウェア階層の視点を示す．最上位の階層は，

方程式を解く，という意味をもつ「ソルバ階層」である．次の上位の階層は，ソルバのコンポーネントとなる「サブソルバ階層」であり，行列の何らかの変換という意味を持つ．例えば，ハウスホルダ法による固有値計算においては，3重対角化がそれにあたる．3重対角化は，ある一定の処理の塊ではあるものの，何らかの方程式を解くまでの意味はない．その次の階層は，BLAS[2]である．これは，行列とベクトルの何らかの処理を行うという意味を持つ．BLASは更に3つに分かれ，BLAS1はベクトルとベクトルの演算，BLAS2は行列とベクトルの演算，BLAS3は行列と行列の演算である．最下位の階層は，ループである．

この4階層の定義により，自動チューニング技術とコンパイラにおける最適化技術の差異を次のように説明できる．コンパイラにおける最適化技術は，主としてループレベルの最適化情報として，ループ内における変数の相互依存関係を分析する．一方で，自動チューニング技術は，行列計算における最大性能を出すため，上記ソフトウェア階層の各々において最適なものを選択する．

Table 1. Matrix Computation Hierarchy.

ソフトウェア階層	例
ソルバ階層	連立一次方程式 $Ax = y$，　固有値 $Av = \mu v$
サブソルバ階層	3重対角化：A (実対称密行列) → T (3重対角行列)，再直交化：V (ベクトル群) → V' (正規化ベクトル群)
BLAS	BLAS1　$x = x + y$,　$\alpha = (x, y)$ BLAS2　$y = Ax$,　$y = Ax + A^t x$ BLAS3　$C = AB$,　$A = A - yx^t - xy^t$
ループ	DO　J = 1,100 　DO I = 1,100 　　Y(J) = 　Y(J) + A(I,J)*X(I) 　ENDDO ENDDO

Fig. 2に，ハウスホルダ法による固有値計算について，自動チューニングの対象となる行列計算ソフトウェア階層を示す．これは，Fig. 1の拡張版となっている．

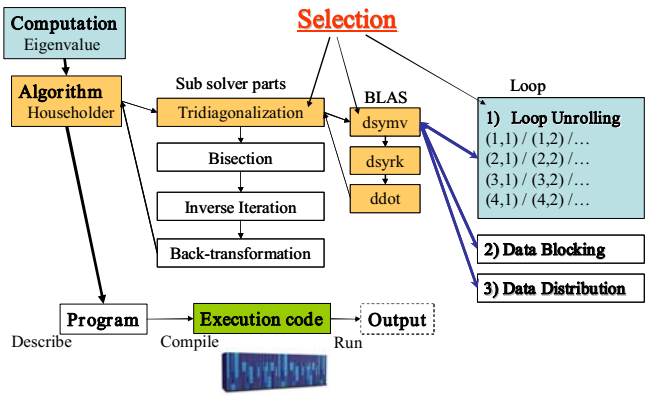

Fig. 2. Selection Points of Tuning in Eigenvalue Computations.

チューニングの選択の範囲はサブソルバの階層，BLASの階層，そしてループの階層と幅広くなっている．3重対角化のサブソルバ階層では，2種類の行列ベクトル積を統合するか，しないかを選択する．逆反復法のサブソルバ階層では，いくつかある再直交化のアルゴリズムから最適なものを選択する．BLASの階層では，各サブソルバに対する最適なBLASを選択する．ループの階層では，各BLASの実装に対して最適なループアンローリング，ブロックサイズ，データ分散方式を選択する．以上の各階層の選択は，想定する計算機上に応じて変わってくる．

ソフトウェア開発サイクルの視点をFig. 3に示す．計算システムを構築する際，ソースコードを書き，コンパイルして実行コードを計算機上で実行し，そして性能を評価するという手順となる．性能結果を検証した後，プログラミングの方法を改善する．この開発サイクルの各段階において性能に関するパラメータが種々あり，それらを調整することになる．従来は，人間の手作業による部分も多い，この一連の流れをトータルに自動化することを，自動チューニング研究では検討され始めている．

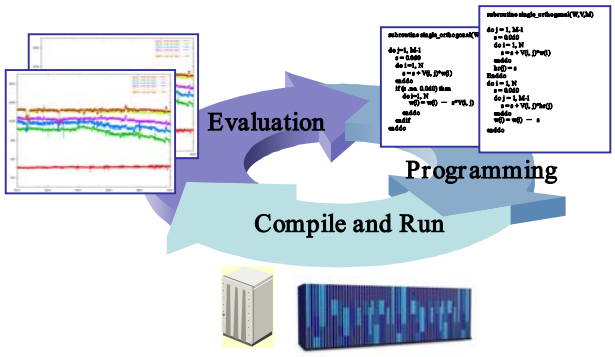

Fig. 3. General Process of Software Development Cycle.

3.　従来の代表的な自動チューニング方法

3.1 PHiPAC（Portable, High-Performance, ANSI C Coding Methodology）

PHiPAC[3][4]はCのループをチューニングするスクリプト言語として提供されている．主な特徴をTable 2にまとめる．PHiPACでは，パラメータ化されたコードジェネレータが，メモリ階層に対応する範囲のブロックサイズ付のCのソースコードを生成する．例えば，mm_genという行列積のコードジェネレータは，行列積のメモリ階層に見合ったいくつかのレジスタ数とブロック数付きのソースコードを生成する．サーチスクリプトは，生成されたソースコードの中で，最適なレジスタブロック数（＝アンローリング段数）を定め，次に最適なL1キャッシュブロック数，最適なL2キャッシュブロック数と順次定めていく．この順番で最適なパラメータ値を決定できる保証は得られていないが，経験上，良好な結果を得ている．

Table 2. Summary of PHiPAC Features.

行列ソフトウェア階層	主としてループ, およびBLASのDGEMM, DGEMV
ソフトウェア開発サイクル	インストール時
自動チューニングの概要, 構成要素	1) ソースコードジェネレータによる変数値つきCコードを生成. 2) レジスタからより低次キャッシュ(L1, L2)の順でアンローリングを決定. 3) FPUレイテンシを隠蔽.
実例	Sparc, SGI Indigo, HP712/80i, RS/6000-590 (但し単一プロセッサ上)

3.2 ATLAS (Automatically Tuned Linear Algebra Subprograms)

ATLAS[5]は, Pentiumシリーズ, Power PCシリーズ, Sun Sparcシリーズなど多くのCPUに対し個別にチューニングされたBLASプログラム集である. ALTASでのチューニング技術は, 主としてループアンローリングとキャッシュブロッキングである. いくつかのBLAS関数では経験に基づいたチューニングが含まれている. また, 各マシンへのインストールには幅広い性能サンプリングが必要とされる. つまり, システム運用管理者は, 新しい計算機上に対してBLASの全関数をチューニングする必要がある. しかしいったんチューニングされた後は, アプリケーションプログラム開発者は, BLAS関数をチューニングする必要は無い. 現在, ATLASは幅広い計算機上で利用可能であり, ウェブサイト[6]上で公開されている.

Table 3. Summary of ATLAS Features.

行列ソフトウェア階層	全BLAS関数のループレベル
ソフトウェア開発サイクル	インストール時
自動チューニングの概要, 構成要素	1) ループアンローリングおよびキャッシュブロッキングによって自動的にチューニングされたBLAS関数 2) 一部経験的な実装コードを含む 3) インストール時において幅広い範囲の性能サンプリングを行う
実例	PC (Pentium series), RS/6000SP, Sun, Onyx, など (但し単一プロセッサ上)

3.3 I-LIB (A parallel automatically tuned intelligent library)

I-LIBは黒田ら[7][8]により提案, 開発された自動チューニング型ライブラリである. 主として以下4つのパラメータをチューニングする.

1) Depth of loop unrolling for matrix-vector multiplications. (Depthの範囲は1, 2, 3, 4, 8などとされている)

2) Communication method for matrix-vector multiplications. (主として並列計算機のタイプごとに用意される)

3) Gram-Schmidt reorthogonalization. (MGS (Modified Gram-Schmidt) あるいはCGS (Classical Gram-Schmidt)の選択)

4) Preconditioning procedure. (いくつかの前処理が実行時に選択される)

SR2201およびSR8000上での反復解法GMRES(m)の自動チューニングに対する数値実験では, I-LIBは, 同等の機能を有するPETScに対し約4倍の高速化を達成している.

Table 4. Summary of I-LIB Features.

行列ソフトウェア階層	GMRES(m)およびハウスホルダ法に対するサブソルバ階層のチューニング
ソフトウェア開発サイクル	インストール時および実行時
自動チューニングの概要, 構成要素	1) 行列ベクトル積に対するループアンローリングの自動選択 (例えば, アンローリング段数を1,2,3,4,8など代表的な値から選ぶ) 2) 再直交化アルゴリズムの自動選択 (MGSかCGSを実行時に自動選択する) 3) 前処理アルゴリズムの自動選択
実例	SR2201 (MPPとして), SR8000 (SMPクラスタとして)

3.4 ABCLibScript (Automatically Blocking and Communication-adjustment Library Script)

片桐ら[9][10]によって提案されたスクリプト言語である. この言語はカーネルプログラムを実行する前に自動チューニングする. Fig.4にABCLibScriptによってチューニングする概要を示す. このスクリプトは, 多項式などで目的関数を記載することで, ループアンローリングを自動化することができる. 前述のPHiPACがマシンレベルでのBLAS (主として行列積など)チューニングを行うのに対し, ABCLibScriptでは多項式による最適化モデルを言語で与えられるようになっている.

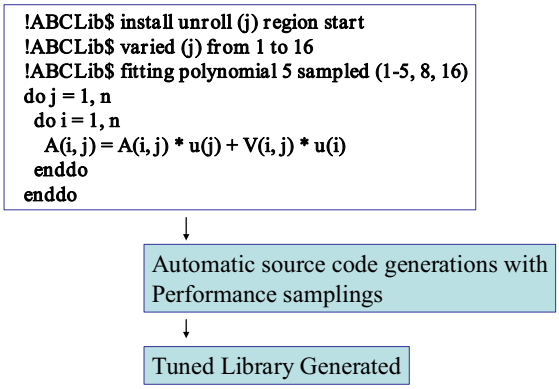

Fig. 4. Sample code with ABCLibScript and the Process.

Table 5. Summary of ABCLibScript Features.

行列ソフトウェア階層	ループレベル
ソフトウェア開発サイクル	インストール時, コンパイル時, および実行時
自動チューニングの概要, 構成要素	1) 任意のループの型に対するループアンローリングの自動選択 2) 多項式による最適化モデルの記述が可能 3) ループインデックスを指定し, そのインデックスに対する性能サンプリング点を指定が可能 4) 上記2)+3)のモデルに基づくソースコードを自動生成
実例	SR8000 (SMPクラスタとして), PC

3.5 FIBER （Framework of Installation, Before Execution-invocation, and Run-time optimization layers）

FIBERは片桐ら[9]によって提案された, ソフトウェア開発サイクルを意識した自動チューニングフレームワークである. これは, 著者らが提案した自動チューニングの定義([1])を3つのソフトウェア開発サイクルに拡張した位置付けになっている. FIBERでは, プログラム中にあり, 性能に影響を与えるパラメータをPerformance Parameters (PP)として定義し, PPを以下3つの種類に分類する.

1) インストール時最適化；Installation Optimization Parameters (IOP)
2) 実行前最適化；Execution-invocation Optimization Parameters (BEOP)
3) 実行時最適化；Run-time Optimization Parameters (ROP)

他の自動チューニング方法では, 最適化パラメータがIOPに限られていた. 固有値計算のランク2更新の部分では, IOPのみでのチューニングに対し, BEOPを適用することで28.7%性能向上したと報告されている([9]).

Table 6. Summary of FIBER Features.

行列ソフトウェア階層	ループ階層, BLAS階層, サブソルバ階層
ソフトウェア開発サイクル	インストール時, コンパイル時, および実行時
自動チューニングの概要, 構成要素	1) 以下の最適化フレームワークを有する. Find tuning parameters (ex, unrolling) satisfying given conditions (ex, matrix size). 2) チューニングパラメータの分類；IOP, BEOP, ROP（詳細は本文）
実例	SR8000 (SMPクラスタとして)

3.6 SANS（Self-Adapting Numerical Software）

数値計算ライブラリをグリッド環境下で管理する要望に対応するよう開発されたフレームワークである. SANSは, 主として以下4つのコンポーネントから成る.

1) An intelligent agent
   - An automated data analyzer
2) A history database
3) A System component
   - Controlling access to the Grid
4) A metadata vocabulary
   - User data and performance profile based on common component architecture, CCA, framework

このフレームワークでは, データベースとスケジューラが特にグリッドを意識したコンポーネントとなっている. システムコンポーネントには, 実行時適合型スケジューラが含まれており, 計算グリッドへのアクセス制御を行っている. CCA型メタデータ言語によって, 計算に使われたデータとアルゴリズムの実行履歴が記録される.

Table 7. Summary of SANS Features.

行列ソフトウェア階層	主としてループおよびBLAS
ソフトウェア開発サイクル	インストール時, コンパイル時, 実行時, および評価時
自動チューニングの概要, 構成要素	データベースとスケジューラによるフィードバック型のチューニング 1) 計算グリッドへのアクセス管理, アクセス制御を行う実行時の適合型スケジューラ 2) 計算対象のデータと計算アルゴリズムを記述したCCA型メタデータ言語
実例	NA (グリッド環境での検証を目指す, とされている. 最近の文献[]にはキャンパスLAN内での結果が報告されている)

CCA; common component architecture

4． 従来法の研究動向

Table 8に上記6つの代表的な自動チューニングについて, 行列ソフトウェア階層とソフトウェア開発サイクルの視点による比較結果を記載する. この表から以下の3つの動向が読み取れる.

1) ループやBLASなどの下位レベルからサブソルバを含めた上位レベルへ

PHiPACはANSI Cコンパイラのコーディングに対するガイドラインという位置付けであるが, そのチューニングをコンパイラで実現しようとするとコンパイル時間がかかりすぎ, 現実的ではない. PHiPACでは一部の行列演算の意味を変えない範囲で, ループの書き方を変えることでチューニングを実現する. また, ILIBではサブソルバを選択することでより高度なチューニングを実現する. このように, PHiPACからATLAS, ILIBとなるに従い, 徐々に上位の意味を鑑み, できるだけ少ない

コストで高度なチューニングを行う方向で研究が発展してきている.

2) プログラム実装からソフトウェア開発サイクル全体へ

上記の中で新しい研究であるFIBERとSANSではフレームワークを提案するようになっている. 以前の研究ではスクリプトやライブラリというソフトウェア形式を採用していた. このような変化は, 自動チューニング技術が, ソフトウェア開発サイクルをより意識したものになっていることを示している. 例えば, 疎行列の行列計算ライブラリに対する自動チューニングを検討する際には, 密行列の場合と比べて, 実行時の自動チューニングをより多く考える必要がある. 何故ならば, 計算性能は, 実行時の行列データに大きく依存し, 従って, フィードバックのある自動チューニングを考えねばならないからである. フィードバックするためには, 単にプログラムを実装するだけではなく, 性能を評価し, 更にその評価の結果, プログラムのパラメータを調整する, といったサイクル的な考え方を導入する必要がある. そのために, フレームワークが必要となり, 上記のような研究動向になっていると思われる.

3) 単一CPUから並列計算機, グリッド環境へ

PHiPACおよびATLASのターゲットは単一CPUの性能向上であったが, I-LIBやABCLibScript, およびFIBERではMPPやSMPクラスタをターゲットにしている. 更に, SANSではグリッドまでをターゲットにしている. このように, より幅広い計算環境をターゲットにするよう変わりつつある.

Table 8. Comparison of the Conventional Automatic Tuning Methods for Matrix Computations (P; prepared in detail, C; considered, -; not available).

AT name (Year)	PHiPAC (1997)	ATLAS (1998)	I-LIB (1998)	ABCLib Script(02)	FIBER (2002)	SANS (2003)
Type	Script	Library	Library	Script	FW	FW
Programming	P	P	C	C	C	C
Solver	-	-	-	-	-	-
Sub solver	-	-	C	-	C	-
BLAS	C	P	C	C	C (ABC.)	C (ATLAS)
Loop	P	C	C	C	C (ABC.)	C (ATLAS)
Compiling	-	-	-	-	-	-
Running	-	-	C	-	-	C
Evaluating (DB)	-	-	-	-	-	C
Platform	Single Proc.	Single Proc.	MPPs, Cluster of SMPs	MPPs, Cluster of SMPs	MPPs, Cluster of SMPs	Grid

FW; framework, XML; extensible markup language, CCA; common component architecture

5. 従来法の課題

5.1 長いインストール時間

従来の自動チューニング方法は, 長いインストール時間を要する. ATLASは, インストール時に広範囲のパラメータ値による多数の性能測定を要し, これが長時間インストール問題になっている. PHiPACも同様に非常に長い時間を要している. インストール時において, ATLASは, インストール先の計算機に適したアンロールパタンを探す. 探索中に, 多数のアンロールパタンのBLAS関数の性能を測定している. その結果, 多くの時間がかかってしまう.

上記の動向1)によれば, 自動チューニング方法は, この問題を緩和するためにより上位のセマンティックスを利用すべきである. これは, どの並列計算機においても, グリッド計算環境においても等しく基本的なアプローチである. この問題を解くことなしに, 自動チューニング方法の適用範囲は非常に限られた計算機のみになってしまうであろう.

5.2 性能の不安定性

行列計算では, パラメータ入力値をほんの少し変えるだけで, 性能が大幅に変わってしまう時がある. このように性能が保証できていないため, 最近の計算機アーキテクチャ上での実効性のある性能チューニングの自動化が出来ないという場合がある.

ある行列計算の性能を詳細に調査したところ, Fig. 5に示すような実効性能が不安定になる例を発見した. PHiPACおよびATLASによって得られた計算カーネルでも同様の性能不安定性が起きている. 最適なループアンローリングパタンを選ぶためには, このような不安定性を測定しておく必要がある. 不安定性の測定がない場合, 最適であるとして選ばれたパタンが, 実際には低い性能であったということも起こりえるのである. 従って, このような不安定性を取り除き, 性能保証を実現するためには, 予め性能劣化を考慮に入れた性能最適化モデルを考える必要がある.

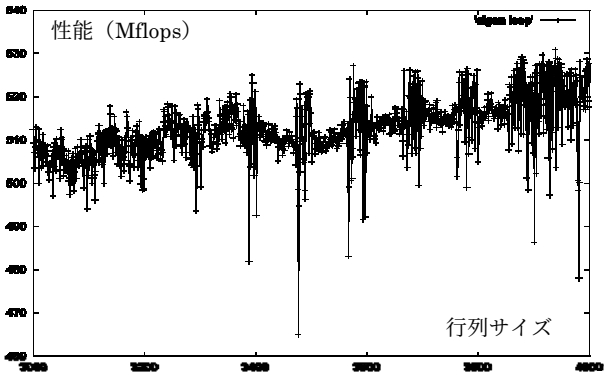

Fig. 5. Performance of some eigenvalue kernel.

既に, 上述の例に挙げた性能劣化はキャッシュコンフリクトによって生じることが知られている. そこで, 今村ら[12]では, キャッシュ上のデータアライメントを変えることによって, 固有値ループの性能を安定化させる方法を提案している. しかしながら, 一般に安定化する方法が得られるとは限らない. 計算機によってはキャッシュが多層になっており, かつ, 複数のCPUによってキャッシュを共有しているため, 性能が更に不安定になる可能性が高い.

5.3 計算時間と計算精度のバランスをとるインターフェイスがないこと

行列計算では計算精度や計算資源と計算時間のバランスを調整することが重要である. 特に反復解法におけるチューニングでは極めて重要である. しかし, これまでの自動チューニング研究には, そのようなバランスをとるという課題に対する答えが用意されていない. この問題は, 計算時間と計算精度のトレードオフを調整するパラメータが用意されていないことに現れている.

固有値計算での再直交化処理では, 上記のトレードオフが, どんなアルゴリズムが選択され, どんな計算が実行されたかによって大きく変わる. しかし, ライブラリの利用者があまり高精度な計算を必要としない場合にでさえ, 従来の自動チューニング方法では, このトレードオフを制御することができない.

I-LIBおよびFIBERでは, 実行時の段階でパラメータ制御の関数を与えることができる. この点では, 他の自動チューニング方法よりもよい性能チューニングとなる. しかし, このチューニングは, I-LIBでは予め用意されたパラメータから選択するか, あるいは, FIBERでは適切な最適化モデルをプログラマが定義して与えなければならない. たとえプログラマがそのような最適化モデルを与えることができたとしても, ソフトウェア開発サイクルにおいて徐々に計算時間と精度を評価しなければならない. このような場合, 非常に多くの時間をかけて, 本当は必要以上に高い精度になってしまう場合がある.

6. まとめ

行列計算の自動チューニングに定義を与え, ソフトウェア階層と開発サイクルの視点から位置付けを明確にした. また, 上記2つの視点から, 従来の代表的な自動チューニング研究を整理し, その課題を述べた. 今後は, 最近, 次々と現れ始めている各種の自動チューニング技術について調査を進めつつ, 5章に記載した課題に取り組んでいきたい.

謝辞：本研究に至る上で共同研究を通じ重要な数多くの寄与を頂いた電気通信大学の今村俊幸講師に謝意を表します. 自動チューニング研究に関し多大の御指導を頂いた電気通信大学の弓場敏嗣教授, 片桐孝洋助手, 東京大学の須田礼仁助教授, 名古屋大学の山本有作講師に謝意を表します.

参考文献

[1] 直野健, 山本有作, "単一メモリ型インターフェイスを有する自動チューニング並列ライブラリの構成方法, " 情報処理学会研究報告, 2001-HPC-87(SWoPP2001), pp.25-30(2001).

[2] BLAS Homepage, "http://www.netlib.org/blas/. "

[3] J. Blimes, K. Asanovic, C.-W. Chin and J. Demmel, "Optimizing matrix multiply using PHiPAC: a portable, high-performance, ANSI C coding methodology, " *Proceedings of International Conference on Supercomputing 97,* pp.340-347 (1997).

[4] PHiPAC Homepage, "http://www.icsi.berkeley.edu/~bilmes/phipac/. "

[5] R. Whaley, A. Petitet and J. Dongarra, "Automated empirical optimizations of software and the ATLAS project, " *Parallel Computing, 27,* pp.3-35 (2001).

[6] ATLAS project, "http://www.netlib.org/atlas /index.html. "

[7] H. Kuroda, T. Katagiri, and Y. Kanada, "Knowledge Discovery in Auto-tuning Parallel Numerical Library, " *Progress in Discovery Science, Final Report of the Japanese Discovery Science Project, Lecture Notes in Computer Science 2281 Springer 2002,* pp.628-639 (2002).

[8] Project I-LIB, "http://www.super-computing.org/~kuroda/nadia.html. "

[9] T. Katagiri, K. Kise, H. Honda, and T. Yuba, "FIBER: A General Framework for Auto-Tuning Software, " *Springer LNCS 2858, The Fifth International Symposium on High Performance Computing (ISHPC-V),* pp.146-159 (2003).

[10] ABC-LIB Homepage, "http://www.abc-lib.org/. "

[11] J. Dongarra and V. Eijkhout, "Self-adapting numerical software for next generation applications, " *The International Journal of High Performance Computing Applications, Vol. 17, No. 2, Summer,* pp.125-131 (2003).

[12] T. Imamura and K. Naono, "Automatic Tuning Technique Exploring within the Hardware-specific Constrained Parameters, " Springer LNCS 3743, Large-Scale Scientific Computing, 5th International Conference (LSSC 2005), Sozopol, Bulgaria, June 2005, pp.413-421 (2006).