< 概論 >

# 2 ソフトウェア自動チューニングの研究動向

直野　健　(株)日立製作所中央研究所



## 自動チューニング技術の生まれた背景

1990 年前後から，計算機アーキテクチャが複雑化してきたため，行列計算ライブラリでは，性能をチューニングする作業が膨大になった．そこで，当該作業の負担を軽減するため，性能チューニングを自動化する技術，自動チューニング（automatic tuning，AT）が研究され始めた．行列計算向け自動チューニングの最初の文献 PHiPAC[1] には，直接的に automatic tuning という用語は出てこないが，その要旨の 2 文にほぼそれを意味する表現が出てくる．

Modern microprocessors can achieve high performance on linear algebra kernels but this currently requires extensive machine-specific hand tuning. We have developed a methodology whereby near-peak performance on a wide range of systems can be achieved automatically for such routines.

すなわち，計算機個別向けの手動チューニング（machine-specific hand tuning）という状態を克服し，自動的にほぼピークに近い性能に達する方法論を開発したというわけである．

ほぼ時期を同じくして，データベースシステムでも自動チューニング技術が注目されるようになっている．データベースシステムの広範囲にわたるパラメータ設定を調整するのに，データベース管理者が多大な時間をかけていることを克服しようとする技術である．行列計算と異なる事情はあるものの，性能に影響を及ぼすパラメータ調整の工数を削減するという目的はほぼ同じである．また，両者とも，コンパイラとアプリケーションの中間に位置するミドルウェア階層のソフトウェアである．データベースでは，いわゆる Web 3 階層（Web サーバ／アプリケーションサーバ／データベースサーバという 3 種類のサーバで構成されるサーバシステム）での高性能化というシステムインテグレーション上の課題があり，これが行列計算の直面したメモリ多層化および多重並列化での高性能実現という課題に相当する．

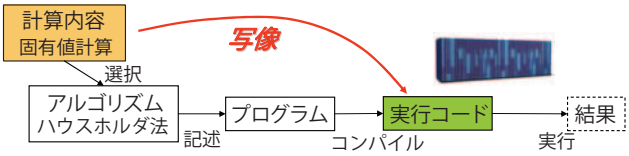


図 -1　計算内容から実行コードへの写像としてのチューニング例

本稿では，主として，行列計算ライブラリ向けに 2003 年までに発表された代表的な 6 つの自動チューニング技術について，比較論を展開する．また，2003 年以降の動向についての情報参照先として，自動チューニング研究会と国際学会 iWAPT について簡単に紹介する．

## 自動チューニング技術の比較の枠組み

### ■ 自動チューニングの例

自動チューニングの比較論に入る前に，図 -1 に固有値計算（行列計算の 1 つ）を実装する過程の例を示す[2]．まず，実行したい固有値計算のアルゴリズムを選択する．たとえば，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コードに変換する．そして，実行コードを計算機上で実行し，固有値計算の結果を得る．最近の高性能計算機上では，アルゴリズムやプログラムの記述方法の組合せの数が莫大であり，かつ実行コードの性能が大幅に変わる．このため自動化が必要になったわけである．

### ■ 自動チューニング比較における 2 つの軸

自動チューニングするプロセスの詳細を分析するため，

| | |
|---|---|
| ①ソルバ階層 | 連立 1 次方程式 $Ax = y$，固有値 $Av = \mu v$ |
| ②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） |
| ③ BLAS | BLAS1　$x = x + y,\ \alpha = (x, y)$<br>BLAS2　$y = Ax,\ y = Ax + A^t x$<br>BLAS3　$C = AB,\ A = A - yx^t - xy^t$ |
| ④ループ | DO J = 1,100<br>DO I = 1,100<br>Y(J) = Y(J) + A(I,J)*X(I)<br>ENDDO<br>ENDDO |

表 -1　行列計算でのソフトウェア階層とその例

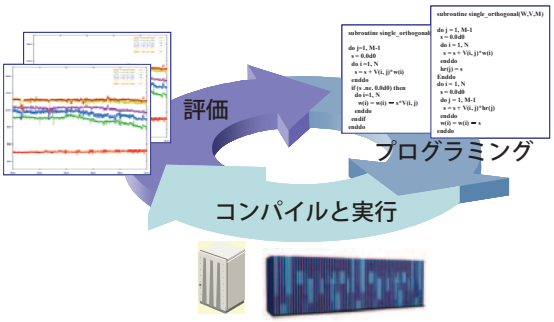


図 -2　ソフトウェア開発サイクルの例

本稿では，2 つの重要な軸を導入する．1 つの軸は行列計算におけるソフトウェア階層の軸であり，もう 1 つの軸は，ソフトウェア開発サイクルの軸である．

表 -1 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持つ，①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，②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．たとえば，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，③ BLAS（基本線形計算サブルーチン）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．最下位の階層は，④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．

ソフトウェア開発サイクルの視点を図 -2 に示す．計算システムを構築する際，ソースコードを書き（プログラミング），コンパイルして実行コードを計算機上で実行し（コンパイルと実行），そして性能を評価する（評価）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．この過程は，従来，人間の手作業による部分が多かったが，一連の流れとして自動化することが検討され始めている．

この 2 つの軸で比較する理由は次の通りである．ソフトウェア階層の軸は，コンパイラの上位化を意味する自動チューニングの基本の軸であり，より上位の計算内容の意味を鑑みたチューニングを行うために必須の軸である．もう 1 つのソフトウェア開発サイクルの軸は，チューニングの手順からくる自然な軸である．プログラムを書き，性能を計測し，プログラムのパラメータを制御する．また，性能を計測し，再度パラメータを制御する．このような過程を意味する基本的な軸である．

## 自動チューニング技術の主要動向

ここでは，6 つの代表的な自動チューニング研究を述べ，上記の 2 軸による分析を行う．

### ■ 6 つの自動チューニング研究

まず，上述の PHiPAC は Blimes ら [1] により提案された，ANSI C で記述された行列計算ループを高性能化するコーディング方法である．論文 [1] では，行列乗算ループ（DGEMM）をチューニングするスクリプト言語が提案され，PHiPAC を自動チューニング付きライブラリとして呼ぶ場合，このスクリプトを指す．

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（レジスタ，L1 キャッシュ，L2 キャッシュといった階層）に対応したブロックサイズ長（行列演算を行うある塊の長さ）による C のソースコードを生成する．たとえば，mm_gen という行列積のコードジェネレータは，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行列積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．

ABCLibScript は，片桐ら [3] 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（た

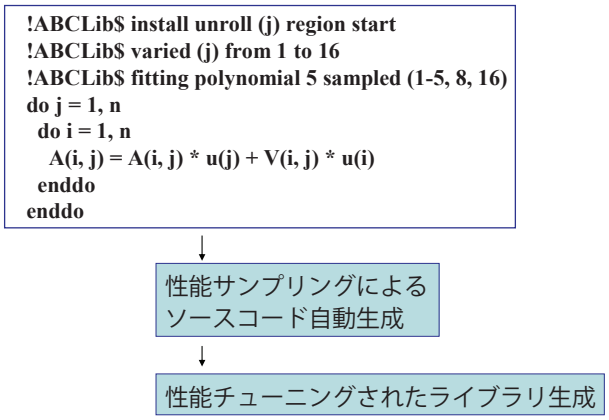


図 -3　ABCLibScript のサンプルとチューニング過程の概要

とえば行列ベクトル積のループなど）を実行する前に自動チューニングする．図 -3 に ABCLibScript によってチューニングする過程の概要を示す．このスクリプトは，多項式などで目的関数（図 -3 の fitting polynomial 5 とは 5 次の多項式での近似目的関数）を記述することで，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．

ATLAS は Whaley ら [4] により提案されたライブラリで，Pentium シリーズ，PowerPC 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．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ない．

ILIB は黒田ら [5] 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．GMRES(m)（代表的な反復型連立 1 次方程式解法の 1 つ）などに対し，主として以下 4 つのパラメータをチューニングする．

1) 行列ベクトル積のループアンローリング
2) 行列ベクトル積の通信方法
3) 直交化のアルゴリズム
4) 行列計算の前処理アルゴリズム

スーパーコンピュータ SR2201 および SR8000 上での GMRES(m) に対する数値実験では，ILIB は，同等の機能を有する PETSc（世界的に有名な反復型連立 1 次方程式解法ソフトウェア）に対し約 4 倍の高速化を達成している．

FIBER は片桐ら [3] 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，筆者らが提案した自動チューニングの定義 [2] をソフトウェア開発サイクルの 3 つの段階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 1）インストール時最適化，2）実行前最適化，3）実行時最適化に分類している（他の自動チューニング方法では，最適化パラメータがインストール時最適化に限られていた）．性能評価例として，固有値計算のランク 2 更新の部分では，インストール時最適化のみでのチューニングに対し，実行前最適化を適用することで 28.7% 性能向上したと報告されている [3]．

SANS は Dongarra ら [6] によって提案されたフレームワークで，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ている．SANS は，主として，1) 知的エージェント（自動データ分析），2) 履歴データベース，3) システムコンポーネント（グリッドへのアクセス制御），4) メタデータ言語（ユーザデータ，CCA に基づく性能プロファイラ），という 4 つのコンポーネントからなる．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（Common Component Architecture，共通コンポーネントアーキテクチャ）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．

### ■ 2 軸による分類と発展の傾向

上記 2 軸で 6 つの自動チューニング技術を評価した結果を表 -2 にまとめる．この表から自動チューニングの発展に，以下の 3 つの傾向が読み取れる．

第 1 に，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発展しているという傾向が読み取れる．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．ATLAS では対応範囲が BLAS 全体へと広がっているが，サブ

ソルバまでは対応していない．一方，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に研究が発展してきている．

第2にプログラム実装からソフトウェア開発サイクル全体へ発展しているという傾向が読み取れる．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．たとえば，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．なぜならば，計算性能は，実行時の行列データに大きく依存し，したがって，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，さらにその評価の結果，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．

第3に，単一CPUから並列計算機，グリッド環境へと発展しつつあると読み取れる．

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，ILIB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．さらに，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．

| 名称(年) | | PHiPAC (1997) | ATLAS (1998) | ILIB (1998) | ABCLibScript (2003) | FIBER (2003) | SANS (2003) |
|---|---|---|---|---|---|---|---|
| タイプ | | スクリプト | ライブラリ | ライブラリ | スクリプト | フレームワーク | フレームワーク |
| ソフトウェア階層（プログラミング段階） | サブソルバ | - | - | △ | | △ | - |
| | BLAS | △ | ○ | △ | △ | △（ABCLibScriptを一部利用） | △（ATLASを一部利用） |
| | ループ | △ | △ | △ | ○ | △（ABCLibScriptを一部利用） | △（ATLASを一部利用） |
| 開発サイクル | コンパイル | - | - | - | - | - | - |
| | 実行 | - | - | △ | - | - | △ |
| | 評価 | - | - | - | - | - | △ |
| 計算環境 | | 単一CPU | 単一CPU | 超並列計算機 | 超並列計算機 | 超並列計算機 | グリッド |

表-2　行列計算向けの従来の自動チューニング方法の比較

## 国内における研究活動状況

上記の通り自動チューニング技術が種々提案され始めた状況を受け，本格的な調査，研究を目的に「自動チューニング研究会」が2003年11月，電気通信大学（東京都調布市）に発足した．以来約5年の間，年間5回程度の研究会を開催している．2009年2月現在の研究会構成員数は17名である．その間，文部科学省科学研究費補助金費などへの応募と採択された研究課題の実施という形で，ソフトウェア自動チューニング技術の研究を進め，課題調査報告書[7]をまとめている．また，2006年以降，国際会議iWAPT（International Workshop on Automatic Performance Tuning）[8]を企画，実施している．

本稿では2003年までの動向についてまとめたが，その後の発展については，上記の報告書[7]や会議録[8]および本特集の各テーマを参照されたい．


**参考文献**

1) Blimes, J., Asanovic, K., Chin, C.-W. and Demmel, J. : Optimizing Matrix Multiply using PHiPAC : A Portable, High-performance, ANSI C Coding Methodology, Proceedings of International Conference on Supercomputing 97, pp.340-347 (1997).
2) 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).
3) Katagiri, T., Kise, K., Honda, H. and Yuba, T. : 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).
4) Whaley, R., Petitet, A. and Dongarra, J. : Automated Empirical Optimizations of Software and the ATLAS Project, Parallel Computing, 27, pp.3-35 (2001).
5) Kuroda, H., Katagiri, T. and Kanada, Y. : 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).
6) Dongarra, J. and Eijkhout, V. : 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).
7) 自動チューニング研究会編：自動チューニング技術に関する課題調査，自動チューニング研究会技術報告 (2008).
8) Proceedings of International Workshop on Automatic Performance Tuning (iWAPT), Vol.1 (2006)/ Vol.2 (2007) / Vol.3 (2008).


（平成21年4月6日受付）


**直野　健（正会員）** ken.naono.aw@hitachi.com

（株）日立製作所中央研究所 主任研究員．