特集 科学技術計算におけるソフトウェア自動チューニング

<ソフトウェア自動チューニングを支える基盤>

4 ソフトウェア自動チューニング記述のための計算機言語

片桐　孝洋　東京大学

なぜ自動チューニング記述言語か?

本特集でも解説を行っているが，自動チューニング機能付き数値計算ライブラリが近年多数開発されている．ところが残念なことに，応用ソフトウェアの中には数値計算ライブラリを利用していないか，利用することができない場合がある．理由はさまざまである．独自開発コードしか利用できないという組織的制約もある．しかし，深刻な理由の 1 つは，提供機能が応用プログラムの機能要求を満たしていないことである．ライブラリの汎用手法が応用問題をまったく考慮していないことから，応用プログラム中で反復が収束しない場合がある．

かくして，少なからず応用プログラム開発者は独自の数値計算プログラムを個別に開発してきた．しかし近年の計算機アーキテクチャの複雑化により，性能チューニングがきわめて煩雑となった．このことから，性能自動チューニングにより開発コストを削減したい要求がある．数値計算ライブラリ開発者でさえ，自動チューニング機能の付加は重荷であり，自動化したい要求がある．

これらの問題を解決するため，所有コードに自動チューニング機能を付加する記述言語の開発が望まれる．本稿では，筆者らが開発してきた ABCLibScript (Automatically Blocking and Communication-adjustment Library Script) について紹介する．

自動チューニング記述言語が持つべき機能

コンパイラが提供する最適化のためのコメント行（ディレクティブ）など，これまでにもプログラム中にコメントを入れることで，計算機アーキテクチャに特化した最適化方針をユーザが与えることは広く行われている．このコメントを計算機言語と見なす場合，先行となる計算機言語は多数提案されてきた．本稿で対象とする記述言語は，このようなコメントの挿入作業，または，高速なコメント記述を探す作業（チューニング）を自動化するための記述言語である．したがって，従来の計算機言語が最適化のための直接的な記述を必要とするのに対し，本稿で取り扱う計算機言語は抽象度が一段高くなっている点に差異がある．また，どのようにチューニングを行っていくかという方針，および，チューニングを行うタイミングも計算機言語のフレームワークとして持つ必要がある．計算機言語の機能が多様である点も違いがある．

さて，このような自動チューニング記述言語には，以下の機能が必要である．

1. プログラム上の任意個所（ループや手続き呼び出し部分など)に，容易に自動チューニングが記述できること
2. 自動チューニングのタイミング指定ができること
3. アルゴリズム(コード)選択の機能があること
4. 自動チューニング機能が自動付加されたコードをユーザが直接操作(確認や編集)できること

要求機能 2 は，従来自動チューニングを行うタイミングの一般化がなされていないことから，実現不可能であった．そこで，自動チューニングのタイミングを (1) ソフトウェアのインストール時，(2) ユーザ知識を与えた時である実行起動前 (実行前) 時，そして，(3) ソフトウェア実行時，の 3 タイミングに規定した．この 3 タイミングを用いた自動チューニングのソフトウェア構築枠組みを FIBER（Framework of Install-time, Before Execute-time, and Run-time Auto-tuning，ファイバ）[1] と呼ぶ．一方，要求機能 4 であるが，応用プログラム開発者は自動チューニングにより，どのような処理がなされるのか性能向上の観点から確認したい場合がある．加えて，ユーザは計算機環境ごとに最適化性能が異なるコンパイラを利用している．これらコンパイラと自動チューニング機能がコード最適化の観点から協調をはかるため必要な機能である．

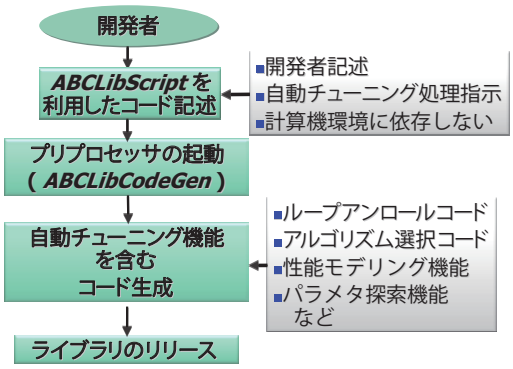

図 -1　適用シナリオ（開発者）

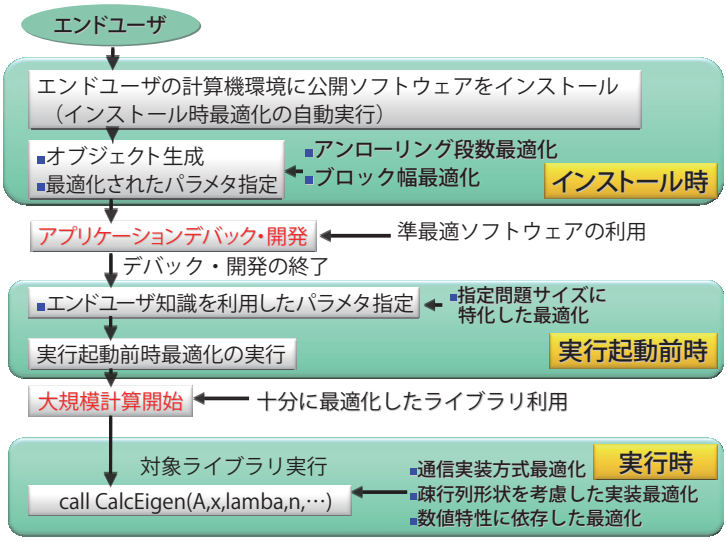

図 -2　適用シナリオ（エンドユーザ）

ABCLibScript ー FIBER を実現する言語

ABCLibScript[2] は，以下に考慮して設計された．

1. 数値計算処理に特化した自動チューニング機能の提供によるチューニング指定の容易さ
2. ソフトウェア開発者（以降，単に開発者と呼ぶ）がディレクティブ形式でプログラム中に指示を与えることで，ディレクティブによる指示を取り除いた場合に，もとのプログラムの実行順序を変更しない自動チューニング指定方式
3. 開発者によるディレクティブを解釈し，可読性の高いプログラムを自動生成するプリプロセッサの提供

ABCLibScript は，数値計算処理を強く指向し以下の機能のみを提供する．

1. ブロック化アルゴリズムに対応するブロック幅調整機能
2. エンドユーザが与える知識により最適なアルゴリズムを切り替えるアルゴリズム選択機能
3. ループアンローリングに対応するループアンローリング段数調整機能

ABCLibScript によるライブラリ開発シナリオ

数値計算ライブラリ開発者により，自動チューニング機能付加を ABCLibScript で行う開発シナリオを解説しよう．図 -1 に，開発者の立場での適用手順を示す．

図 -1 から開発者は，自分の所有するコードの任意個所に ABCLibScript で自動チューニング処理の指示をコメントとして記述する．共有メモリ型並列計算機での並列化をする際，OpenMP というコメント（ディレクティブ）を用いてループごとに並列化方式をユーザが直接指示していくプログラミングスタイルが容易であることから，広く知られている．ABCLibScript を用いた自動チューニング処理の指示は OpenMP での並列化指示に類似しており，容易に記述できるプログラミングスタイルであるといえよう．

ABCLibScript による自動チューニング指示は専用プリプロセッサで処理され，開発者が記述したコードの計算機言語と同一の計算機言語（たとえば C 言語や Fortran 言語）で，自動チューニング機能が自動付加されたコードが生成される．この自動生成コードをコンパイルすることで自動チューニング機能付きライブラリが開発・提供できる．

一方，ライブラリを利用するユーザ（エンドユーザと呼ぶ）は，図 -2 に示す手順で利用する．

図 -2 では，ABCLibScript が前提としている FIBER の 3 段階のタイミングで自動チューニングが実行される．

まず，エンドユーザの計算機環境にライブラリをインストールするとき，インストーラと連携する形でインストール時最適化がなされる．これは，基本線形計算ライブラリ BLAS を自動チューニングするソフトウェア ATLAS と同様のタイミングである．

次に，エンドユーザが必要とするタイミングで実行起動前時（実行前）最適化がなされる．たとえば，エンドユーザが解くべき行列を確定したとき，エンドユーザ自らが行列サイズを指定したうえで自動チューニング実行を，提供されているインタフェースを通して指定することに相当する．

```
!ABCLib$ ＜自動チューニング種類＞ ＜機能名＞ ［(対象変数)］region start
［!ABCLib$ ＜機能の詳細＞ ［sub region start］］

処理対象のプログラム(自動チューニング領域)

［!ABCLib$ ＜機能の詳細＞ ［sub region end］］
!ABCLib$ ＜自動チューニング種類＞ ＜機能名＞ ［(対象変数)］region end
```

- ＜自動チューニング種類＞::=（install | static | dynamic）
  - install　：インストール時最適化指定
  - static　：実行起動前時最適化指定
  - dynamic：実行時最適化指定
- ＜機能名＞::=（define | variable | select | unroll）
  - define　：パラメタ設定処理指定
  - variable：変動パラメタ指定
  - select　：選択処理指定
  - unroll　：ループアンローリング指定

図 -3　ABCLibScript の表記法

```
!ABCLib$  install unroll (i,j,k) region start
!ABCLib$  name  MyMatMul
!ABCLib$  varied (i,j,k) from 1 to 4
do i=1, N
  do j=1, N
    do k=1, N
      A(i, j) = A(i, j) + B(i, k) * C(k, j)
    enddo
  enddo
enddo
!ABCLib$  install unroll (i,j,k) region end
```

インストール時最適化
アンローリング処理指定

最大アンローリング
段数指定
（開発者知識）

対象領域（AT領域）
（開発者知識）

図 -4　unroll 指定子による記述例

最後に，エンドユーザがまったく関知しないタイミングで実行時最適化がなされる．この実行時最適化は，インストール時，実行起動前時においてもなされているが，開発者がこれらのタイミングと関連ないか，または同時に行っても問題ない自動チューニング機能を実行時最適化として実装する前提をおく．

■ 開発者によるディレクティブ記述

ソースプログラムにおいて，!ABCLib$ で始まる行は ABCLibScript の自動チューニング指示行と見なす．この表記法の概略を図 -3 に示す．

図 -3 の機能名で指定される命令を指定子と呼ぶ．図 -3 から指定子は，設計方針に従い機能が variable, select, unroll の 3 種のみである．ここで，define 機能はパラメタの設定のみ行うものであり，自動チューニング領域のコードを処理しないので，自動チューニングのための機能とは見なさない．

variable 機能は，変動するパラメタの定義域（変化範囲）すべてを変化させて自動チューニング領域を実行して実行ログを取ることで，最適なパラメタ値を探索する．その後，最適パラメタを設定する機能を提供する．この機能は，チューニング作業の自動化という意味で本質的な機能である．

select 機能は，自動チューニング領域に記載された任意のコード，サブルーチンコールの選択機能を提供する．数値計算においては，実行時に判明する入力行列の性質（非零要素の分布や数値特性）により最適なアルゴリズムが異なる．このような状況で最適なアルゴリズムを決定するには，実行時に分かる情報から候補となるアルゴリズムを実行して判断するしかないが，この処理を記述できる．

unroll 機能は，任意のループのアンロール段数最適化処理を記述することができる．この段数は，計算機アーキテクチャとループ中の演算におけるデータアクセスパターンにより異なる．従来は，コンパイラが最適化できないか，最適化が不十分な複雑なループについて，開発者が職人芸的な技能を用いてチューニングを行っていた．開発コスト削減の観点から，自動化が望まれていた．

次に開発者による具体的な記述例を説明しよう．図 -4 に unroll 指定子による記述例を示す．

図 -4 から，任意のプログラムにおける任意のループ（もしくはループ変数）のアンローリングによる自動チューニングが，たった 4 行で記述できる．どのコード部分がアンローリングをすると全体性能に大きく寄与するかの知識，およびアンローリングの最大段数の知識が必要となる．この知識は開発者が持っており，かつ陽に記述できる前提をおく．

一方，アルゴリズム（関数や手続き呼び出しなども含む）の選択処理を select 指定子により記述する例を図 -5 に示す．

図 -5 から，select sub region start 〜 select sub region end で囲まれた複数領域に任意のプログラム，関数・手続きの呼び出し，を記述する．一方，according estimated に続いた部分で記載される自動チューニング領域の選択基準（コスト定義関数）を陽に記述できる仕様となっている．この，コスト定義関数の記述を省略する場合，最初に select 指定子中に存在するすべての自動チューニング領域を実行し，実行時間のプロファイルをとる．次回以降は，実行時間が最も少ない個所を選択するコードが自動生成される．このような処理は，反復解法ライブラリ中でのアルゴリズム選択への適用を指向している．

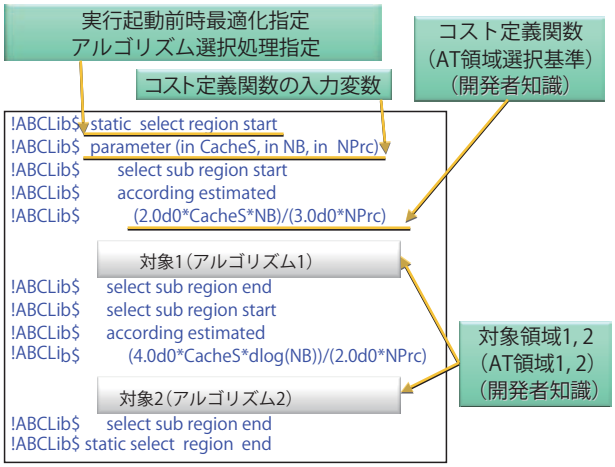

図-5　select 指定子による記述例

■ 性能評価事例

一般に，計算機言語の記述性，実行性能，およびそれを用いたアプリケーション開発コストを総合評価することはきわめて難しい．実行性能のみが議論されることが多く，性能評価として誤解を招く可能性を承知した上で，密行列の行列–行列積のチューニング（手によるコードの改編も含む）を対象とした．ターゲットマシンは Intel Pentium 4 2.0GHz, コンパイラは PGI Fortran コンパイラとする．コンパイラオプションを −O0（最適化なし）に固定する．開発者の最適化能力に実行性能が直接影響する状況下での性能評価を行うためである．最高性能を求めるのではなく，ソフトウェア高性能化のための開発期間と ABCLibScript による自動チューニング品質との評価を行うことが目的である．

手によるチューニング期間を 2 週間としこれを先行して行い，その後，ABCLibScript による自動チューニング機能の記述を 2 時間行う．両者のソフトウェアの最終性能を比較する．ABCLibScript の利用制約は，インストール時自動チューニング，unroll 指定子のみ指定可能とした．被験者は中程度の高速化技術を有し，ブロック化などのキャッシュ高速化手法を知っている．ABCLibScript 仕様は熟知していない．最終コードの実行性能を図-6 に示す．

図-6 から，ABCLibScript による自動生成コード（インストール時最適化を実行）が高速となった．ここで興味深いのは，問題サイズ 128 ～ 900 ではオリジナルコードは高性能を達成しているが，大規模になると性能の劣化を引き起こす点である．この理由は，ブロック化コードのブロック幅が固定で，かつアンローリング段数も固定となる実装をしたためである．被験者の手によるチューニングコードに ABCLibScript による自動チューニングが追加されることで，インストール時に自動設定されるサンプリング点（行列サイズに関係する）でアンローリング段数が動的変更できるコードを生成した．その結果，高性能を維持できるコードが低い開発コスト（開発工程）で実現できた．

行列積以外のプログラムで総合的評価をする必要があり，かつ，ブロック化など元となる高性能なプログラム開発のコストの評価もしなくてはいけないが，性能を維持できる高性能なコード開発の期間が劇的に短縮できる可能性がある興味深い事例といえよう．

汎用的な自動チューニング記述言語の確立を

自動チューニング記述言語について，たとえば PHiPAC[4] で提案しているスクリプトが機能として相当する．しかし，対象が行列–行列積の演算のみに特化されており，汎用的であるとはいえない．一方，ABCLibScript は，(1) 任意の演算・処理に対して適用できる；(2) まったく異なる処理であるアルゴリズムの選択機能がある；(3) FIBER の 3 段階の時間的タイミングにより，広範なソフトウェアに適用可能な自動チューニング方式をフレームワークとして提供している；という観点で，汎用性のある自動チューニング記述言語である．このような汎用的な機能を持つ自動チューニング記述言語は，筆者の知る限り諸外国で開発されておらず，ABCLibScript が国産かつ世界初の自動チューニング記述言語であると考えられる．

ABCLibScript の研究上の課題は，開発者が自分のプログラムの特徴を熟知していないと現実的な自動チューニングソフトウェアが開発できない点である．つまり，技術的に未熟な開発者が利用すれば，コード量増大，自動チューニング時間増大，および低効果の自動チューニングソフトウェアが容易に実現される．自動チューニング時間の増大については，離散的な補完方式を用いてサンプリングデータ数の削減を行う方式[3]や，実験計画法を用いて試行回数を削減する方法（本特集の「3. ソフトウェア自動チューニングの数理」参照）が研究されており，いずれも ABCLibScript の自動チューニング時間削減に資する．コード量削減は，インストール時自動チューニングで動的にコード生成する方式の実装をすれば，実用的なコード量に削減できると予想される．この一方で，高い自動チューニング効果を容易に実現するため，コスト定義関数の自動導出が本質的な課題である．そのため，機械学習分野の成果の適用が期待されるが，数値計算ライブラリを対象とした事例は筆者の知る限りほと

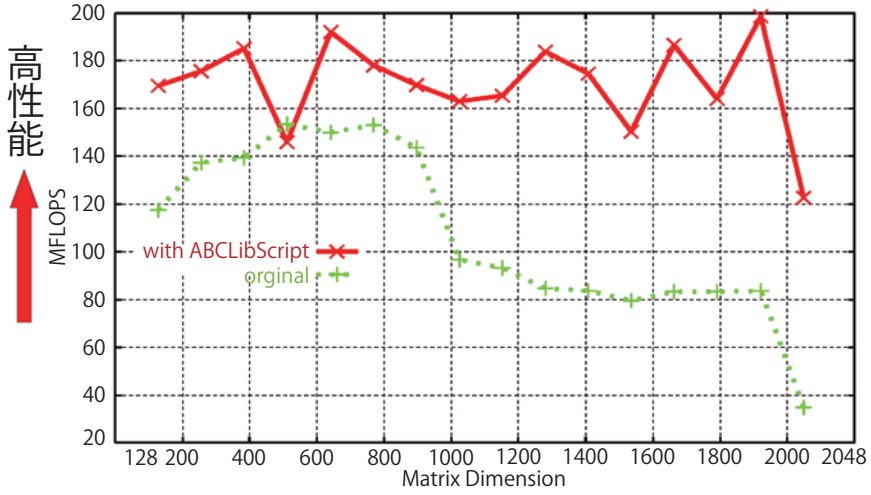

図 -6　被験者のコード性能（行列ー行列積コード，unroll 指定子，ブロック化コード）

んどない．一方で，ABCLibScript の組込みソフトウェアや電力最適化への適用が期待されている．しかし，これらの分野では C 言語が使われており，現在提供している Fortran90 言語用のプリプロセッサでは適用できない．C 言語版のプリプロセッサの開発と方式研究が必要である．

ABCLibScript は研究用ソフトウェアとして利用されており，現在，品質向上のための改良を適宜行っている段階にある．トライアル版が http://www.abc-lib.org/ で無償公開されているので，ご興味のある方は参照されたい．

参考文献

1）Katagiri, T., Kise, K., Honda, H. and Yuba, T. : Effect of Auto-tuning with User's Knowledge for Numerical Software, Proceedings of ACM Computing Frontiers 04, pp.12-25 (2004).
2）Katagiri, T., Kise, K., Honda, H. and Yuba, T. : ABCLibScript : A Directive to Support Specification of An Auto-tuning Facility for Numerical Software, Parallel Computing, Vol.32, Issue 1, pp.92-112 (2006).
3）Tanaka, T., Katagiri, T. and Yuba, T. : d-Spline Based Incremental Parameter Estimation in Automatic Performance Tuning, Selected Paper of Workshop On State-of-the-art In Scientific And Parallel Computing (PARA'06), Springer LNCS 4699, pp.986-995 (2007).
4）Blimes, J., Asanovic, K., Chin, C.-W. and Demmel, J. : Optimizing Matrix Multiply using PHiPAC : A Portable, High-performance, ANSI C Coding Methodology, Proc. International Conference on Supercomputing 97, pp.340-347 (1997).

（平成 21 年 3 月 24 日受付）

片桐　孝洋（正会員）　katagiri@cc.u-tokyo.ac.jp

東京大学情報基盤センター特任准教授．平成 8 年京都大学工学部卒業，平成 13 年東京大学大学院理学系研究科情報科学専攻修了．博士（理学）．平成 14 年電気通信大学助手，平成 17 年米国カリフォルニア大学バークレー校訪問学者を経て，平成 19 年より現職．平成 14 年度本会山下記念研究賞受賞．