自動チューニング処理記述用ディレクティブ ABCLibScript の設計と実装

片桐孝洋[†,††] 吉瀬謙二[†,††]
本多弘樹[†] 弓場敏嗣[†]

この論文では，FIBER フレームワークによる自動チューニング処理記述用ディレクティブである ABCLibScript の設計方針と実装について述べる．ABCLibScript では処理対象を数値計算処理に限定し，ブロック化アルゴリズムのブロック幅調整，アンローリング段数調整，およびアルゴリズム選択，といった自動チューニング機能のディレクティブを提供する．このことで自動チューニング処理の付加を実際に行うライブラリ開発者において，自動チューニングライブラリ作成の際の労力の削減とその開発支援をねらう．

Design and Implementation of ABCLibScript: A Directive to Support Specification of Auto-tuning

Takahiro KATAGIRI,[†,††] Kenji KISE,[†,††] Hiroki HONDA[†]
and Toshitsugu YUBA [†]

In this paper, we describe the design and implementation of ABCLibScript, which is a directive to support the addition of auto-tuning processes based on the framework of FIBER. ABCLibScript limits the function of auto-tuning to numerical computations. For example, the functions of the block-length adjustment for blocked algorithms, loop unrolling depth adjustment, and algorithm selection, are crucial functions in ABCLibScript. We focus on the reduction and support of development for auto-tuning library to innovate the limited functions of ABCLibScript.

1. はじめに

近年，PHiPAC[1)]，ATLAS[2)]，FFTW[3)]，および ILIB[4)] に代表される，いわゆる「自動チューニング機能付き」の数値計算ソフトウエア（以降，自動チューニング機能付きソフトウエア，Software with Auto-Tuning Facilities (SATF) とよぶ）が多数開発されるようになってきた．現在 SATF の性能評価が多くなされており，自動チューニング機能の有効性が周知のものとなりつつある[4)]．

ところが性能の観点で SATF は有効であるのにもかかわらず，ライブラリ開発者が SATF を自分の開発ライブラリに付加する場合，現状では個別に SATF の機能を開発して付加するしかない．例えば数値計算ライブラリの場合，一般的に行列の直接解法ルーチン，反復解法ルーチン，密行列用解法ルーチン，および疎行列用解法ルーチンが含まれているが，それらのルーチン全てに容易に SATF を付加できる基盤ソフトウエアが存在しない．

そこで本論文では機能を数値計算処理に限定し，自動チューニング機構の付加が容易なディレクティブである ABCLibScript を設計，実装する．

本論文の構成は以下の通りである．2 章で，ABCLibScript で前提とする自動チューニング方式の FIBER について説明する．次に 3 章で，ABCLibScript の設計方針について述べる．4 章は，API (Application Programing Interface) を中心とする ABCLibScript の実装について説明する．5 章では，ABCLibScript を用いたプログラム例を示す．6 章では関連研究を述べ，さいごにこの論文での知見をまとめる．

2. FIBER による自動チューニングの概要

FIBER は，SATF に幅広く適用できるソフトウエア枠組の確立を目指し提案された自動チューニングの枠組である[5)~7)]．具体的には SATF の構成方式を，適用対象のアプリケーションの幅を増やし，かつパラメタ推定精度を高める目的で，(1) ライブラリインストール時に行う**インストール時最適化階層** (Install-time Optimization Layer, IOL)，ソフトウエア開発者が指定した特定パラメタ (たとえば問題サイズなど) を，ソフトウエア利用者が固定した時点で行う**実行起**

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
†† 科学技術振機構 戦略的創造研究推進事業 さきがけプログラム「情報基盤と利用環境」領域
“Information Infrastructure and Applications”, PRESTO, Japan Science and Technology Agency (JST)

動前最適化階層 (Before Execute-time Optimization Layer, BEOL)，そして実際にライブラリが実行された時点で行う**実行時最適化階層** (Run-timeOptimization Layer, ROL)，の 3 種の最適化階層に分けて構成する枠組である．この SATF のためのソフトウエア構成を，FIBER (Framework of Install-time, Before Execute-time, and Run-time optimization layers, ファイバー) と呼ぶ．

2.1 2 種のユーザと FIBER のソフトウエア構成

FIBER では，数値計算ソフトウエアにおける自動チューニング処理を主なターゲットとしている．そのため FIBER におけるユーザは，以下の 2 種が想定されている．

- ソフトウエア開発者
- ソフトウエア利用者 (エンドユーザ)

ソフトウエア開発者は，FIBER が提供する自動チューニング指定子である ABCLibScript を利用して，自分が作成したソフトウエアに自動チューニング機能を付加するユーザである．一方で，ソフトウエア利用者 (エンドユーザ) はソフトウエア開発者が開発した FIBER による自動チューニング機構を付加したソフトウエアを利用するユーザである．**図 1** に，上記 2 種のユーザからの観点での FIBER での処理手順をのせる． 図 1 では，2 種のユーザごとに処理手順が異なる．各ユーザにおける，FIBER の利用シナリオを以下に示す．なお以降,「最適化」と「自動チューニング」という言葉が混在する．ここで,「最適化」とはパラメタを手動や自動にかかわらず調整する処理のことを指し,「自動チューニング」とは最適化を用いてパラメタを自動的に調整する処理のことを指す．

図 1(1) **ソフトウエア開発者の処理手順：** ソフトウエア開発者は，FIBER ツールキット開発者が提供するディレクティブ (ABCLibScript) を用いて自動チューニングを施したい箇所 (自動チューニング対象領域, AT 領域とよぶ) の指定と，どのような自動チューニング機能を適用するかを指定する．その後，FIBER ツールキット開発者が提供するプリプロセッサを起動する．プリプロセッサ起動後，以下の 3 種の構成要素がもとのプログラムに自動付加される．

- **パラメタ最適化コンポーネント：**ソフトウエア開発者が指定したパラメタを最適化する処理を含む部分．最適化方式は多種考えられるが，現在の実装では全探索 (brute-force search)，ソフトウエア開発者が指定する関数 (**コスト定義関数**) による探索空間の絞り込み，および最適パラメタ推定法指定 (最小二乗法) の機能を有する．

  なお本パラメタ最適化コンポーネントには，最適化対象の AT 領域についてパラメタ定義領域全てについて当該 AT 領域を実行しながら自動チューニングを行うモード (**パラメタ自動チューニングモード**) と，既に最適化されたパラメタを参照し当該 AT 領域を実行するモード (**パラメタ実行モード**) がある．これらモードは，以降説明する 3 種の自動チューニングモードごとに使い分けられる．
- AT **領域選択コンポーネント：**現在の最適化の前に行われた最適化済データが蓄積しているファイル (パラメタ情報ファイル) 上の情報と実行時情報から，AT 領域内にある最適な自動チューニング対象を選択する部分．
- AT **領域ライブラリコンポーネント：**ソフトウエア開発者が指定した AT 領域を，FIBER 内で識別可能なサブルーチンとして蓄積する部分．

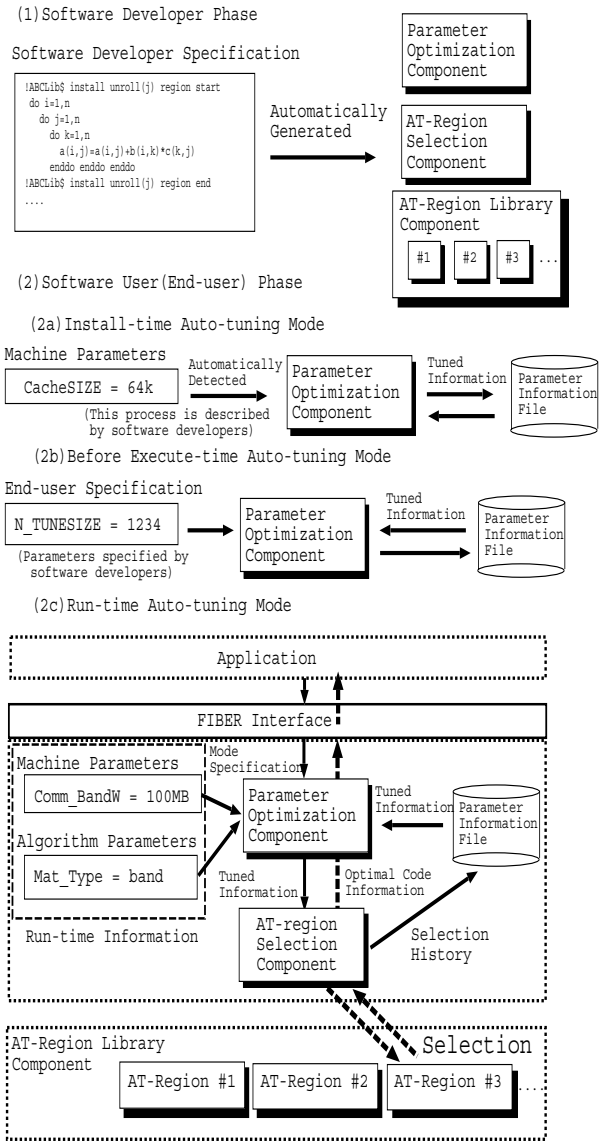

図 1 FIBER での処理手順

この処理のあと，ソフトウエア開発者は自動生成されたプログラムのソースコード，もしくはコンパイル後

のオブジェクトコードをエンドユーザに公開する．

図 1(2) エンドユーザの処理手順：エンドユーザは，ソフトウエア開発者が公開したソフトウエアを利用する．この時，以下に示すタイミングに起因するモードで自動チューニングがなされる．

図 1(2a) インストール時自動チューニングモード：公開されたソフトウエアのインストール時に，**インストール時最適化**を用いた自動チューニングを行うモードである．このモードでは，ソフトウエア開発者が記述した処理に従い，キャッシュサイズなどのエンドユーザが利用している計算機環境情報を自動取得し，それを最適化に利用する．パラメタ最適化コンポーネントのモードは，パラメタ自動チューニングモードである．この自動チューニングは，エンドユーザが特に意識することなく行われる．

図 1(2b) 実行起動前自動チューニングモード：ソフトウエア開発者が指定したパラメタについて，エンドユーザがその値を固定した時を実行起動前時とよび，その時に行われる最適化が**実行起動前最適化**である．この実行起動前最適化を用いて自動チューニングするモードである．このモードでは，インストール時自動チューニングと異なり，エンドユーザが積極的にFIBER ツールキット開発者が提供する API を利用し，自動チューニングのための情報をシステムに通知する．パラメタ最適化コンポーネントのモードは，パラメタ自動チューニングモードである．

図 1(2c) 実行時自動チューニングモード：ソフトウエア開発者が指定した AT 領域が実行された時に，**実行時最適化**を用いて自動チューニングするモードである．このモードでは，実行時におけるシステム情報や，この実行より前に行った当該 AT 領域での自動チューニング情報 (インストール時最適化情報，実行起動前最適化情報，および現実行より前に実行された当該 AT 領域の最適化情報) から，ソフトウエア開発者が指定した自動チューニングを行う．

パラメタ最適化コンポーネントのモードは，ソフトウエア開発者が指定する自動チューニング処理に依存する．当該 AT 領域について，実行時最適化を一切行わない場合はパラメタ実行モードになるが，実行時自動チューニングをする場合はパラメタ自動チューニングモードになる．

3. ABCLibScript の設計

3.1 設計目標

ABCLibScript は，以下の 4 点に考慮して設計がなされている．

1. 数値計算処理に特化した自動チューニング機能を用意することによるチューニング指定の容易さ
2. ソフトウエア開発者がディレクティブの形式でプログラム中に指示を与えることで，もとのプログラムの実行を阻害しないチューニング指定方式
3. ソフトウエア開発者の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
4. 性能に影響する 2 種のパラメタである性能パラメタ（Performance Parameter，PP）と基本パラメタ（Basic Parameter，BP）の概念の導入によるチューニング (パラメタ最適化) 処理の単純化

ABCLibScript は，数値計算処理を指向した以下の機能を有する．

- 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
- 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
- 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替えるアルゴリズム選択機能

ソフトウエア開発者によるディレクティブを解釈し，ソフトウエア開発者が解釈できるプログラムを自動生成するプリプロセッサを提供する．このことで，自動チューニング処理の付加後のプログラムさえもソフトウエア開発者自身がとり扱える．またソースコードが生成され，かつソフトウエア開発者がそれを所有することで，自動付加された自動チューニング処理もソフトウエア開発者に透過性がある (処理のホワイト・ボックス化)．

ABCLibScript では，FIBER の特徴である性能に影響する 2 種のパラメタ (PP と BP) の概念を導入することにより自動チューニングの定義をしている．すなわち自動チューニングとは，AT 領域におけるコストの挙動を定義した関数（コスト定義関数 F）について，パラメタ BP を固定した上で，PP を 3 種のタイミング（インストール時，実行起動前，および実行時）においてコスト定義関数 F を最小化する処理と定義する．(文献 5)~7) を参照．) 本論文で提案する ABCLibScript は，ソフトウエア開発者に PP と BP の定義をさせた上で，最適化処理である自動チューニング処理を記述できるディレクティブといえる．

3.2 指定形式

3.2.1 ソフトウエア開発者による性能パラメタ指定

FIBER では概述のとおり，ソフトウエア開発者が性能パラメタ PP，基本パラメタ BP☆，およびソースプログラム中のチューニング範囲 (AT 領域) の指定に関与する．一方でエンドユーザは，実行起動前最適化において，パラメタ BP の値の指定のみ関与できる．FIBER における自動チューニングとは，ソフトウエア開発者もしくはエンドユーザが固定した BP の値をもとに，PP を自動的に推定する処理である．

ソースプログラムにおいて，`!ABCLib$`で始まる行

☆ 基本パラメタ BP については，行列サイズ n がデフォルトの BP パラメタとなっている．ただし ABCLibScript の仕様では，ソフトウエア開発者が基本パラメタを追加する API の `ABCLib_BPset`(後述) を用意している．

は ABCLibScript による自動チューニングのための指示行とみなす．この表記法の概略を**図 2** に示す．

```
!ABCLib$ ⟨ 自動チューニング種類 ⟩ ⟨ 機能名 ⟩
        [(対象変数)]  region  start
 [!ABCLib$ ⟨ 機能詳細 ⟩ [sub region start]]
              AT 領域
 [!ABCLib$ ⟨ 機能詳細 ⟩ [sub region end]]
!ABCLib$ ⟨ 自動チューニング種類 ⟩ ⟨ 機能名 ⟩
        [(対象変数)]  region  end
```

図 2　ABCLibScript における自動チューニング指示形式

図 2 における ⟨ 自動チューニング種類 ⟩ や ⟨ 機能名 ⟩ のことを指定子とよぶ．また ⟨ 機能詳細 ⟩ のことを補助指定子とよぶ．

3.2.2　指定子および補助指定子の説明

2004 年 1 月における ABCLibScript の仕様である指定子一覧を**図 3** に，および補助指定子一覧を**図 4** にのせる．

- ⟨ 自動チューニング種類 ⟩ :: =
  （install | static | dynamic | ⟨ 式 ⟩）
  install：インストール時チューニングの指定
  static：実行起動前チューニングの指定
  dynamic：実行時チューニングの指定
  ⟨ 式 ⟩ :: =処理対象の計算機言語の文法に従う式
- ⟨ 機能名 ⟩ :: =
  （define | variable | select | unroll）
  define：パラメタ定義処理
  variable：変動パラメタ処理
  select：複数 AT 領域からの選択処理
  unroll：ループアンローリング処理

図 3　ABCLibScript における指定子

3.3　処理対象の計算機言語

ABCLibScript の仕様の本質は，ソフトウエア開発者が所有するソフトウエアが記述されている計算機言語に依存しない．しかし ABCLibScript を処理するプリプロセッサの実装を考慮すると，対象となる言語を特定する必要がある．

ここでは並列数値計算ソフトウエアを指向し，ソフトウエア記述言語は Fortran90，および通信ライブラリとして MPI (Message Passing Interface)-1 が利用されていることを前提とする．この ABCLibScript の実装を考慮した仕様を，$ABCLibScript(Fortran90, MPI-1)$ と表記する．

3.4　プリプロセッサ利用方法

FIBER フレームワークでは，ABCLibScript で記述したディレクティブを解釈し，自動チューニング機能を付加したコードを自動生成するプリプロセッサを提供する．このプリプロセッサ名は `ABCLibCodeGen` である．

$ABCLibScript(Fortran90, MPI-1)$ の仕様に基づき，並列プログラム (test.f) に対し，実際の自動チューニング機能付き並列数値計算プログラムのコード（Fortran90 + MPI-1）を生成するには，以下のコマンドを実行する．

```
>ABCLibCodeGen test.f
```

またプリプロセッサ`ABCLibCodeGen` は，以下の実行時オプションを指定できる．

【実行時オプション】`-debug`
`OFF`：デバック用コードを生成しない（デフォルト値）
`ON`：`ABCLib_DEBUG` 変数で指定されるデバックレベル x のデバック用コードを生成する

【実行時オプション】`-visualization`
`OFF`：自動チューニング軌跡ファイルを出力しない（デフォルト値）
`ON`：自動チューニング軌跡ファイルを出力する

[使用例]

```
> ABCLibCodeGen -debug ON
   -visualization ON test.f
```

ABCLibScript による指示子が記入されているプログラム test.f に対して，デバック用コードを生成し，かつ自動チューニングの軌跡をファイルに残すことを指定する．この生成されるファイルを参照することで，開発予定のビジュアライザを用いれば，自動チューニング経過を閲覧できる．

4.　ABCLibScript の実装

4.1　ABCLibScript の API

ABCLibScript では，ディレクティブによる自動チューニング指定の補助を行い，より緻密な自動チューニング処理が記述できるようにするため，ユーザインタフェース (API) を用意している．なお本 API は，主にソフトウエア開発者が必要とする機能を提供するものであるが，エンドユーザも必要に応じ全ての API を利用することができる☆．

まずはじめに自動チューニングを実行するには，ABCLibScript が提供する以下の API を利用する．

```
【API 1】 ABCLib_ATexec
( ABCLib_ATkinds, ABCLib_ATroutines )
```

`ABCLib_ATexec` 手続きは，`ABCLib_ATkind` で指定した自動チューニングを，`ABCLib_ATroutines` で指定した AT 領域について実行する．引数`ABCLib_ATkinds` は，自動チューニングの種類を指定するための引数である．ヘッダファイル`ABCLibScript.h` で定義された以下の 4 種の定数が指定できる．

☆ エンドユーザはソフトウエア開発者が提供するソフトウエアを利用できればよい立場のユーザなので，これら API の詳細を知りたくないし，API を実際に利用したくないはずである．エンドユーザが本 API を利用しなくてはならない状況は，実行起動前自動チューニングを利用する場合のみである点に注意されたい．

- `name` ⟨ 文字列 ⟩ : AT 領域の名称を記述.
  （全ての機能で利用可能）
- `parameter` （ ⟨ 属性指定 ⟩ ⟨ 変数名 ⟩, [⟨ 属性指定 ⟩ ⟨ 変数名 ⟩, …]）
  : パラメタ情報ファイルに出力か，パラメタ情報ファイルから入力するパラメタを指定する．
  もしくは，基本パラメタであることを宣言する．
  ⟨ 属性指定 ⟩ :: = [`in` | `out` | `bp`]
  `in` ： 入力される（外部で定義され，この AT 領域で参照される）パラメタ,
  `out` ：出力する（この AT 領域で定義される）パラメタ,
  `bp` : 基本パラメタであることの宣言.
  (全ての機能で利用可能)
- `select sub region` （ `start` | `end` ）: 選択する処理であることを指定.
  (機能名 select 指定時)
- `according` （ ⟨ 条件式 ⟩ | `estimated` ） : 以降に指定する基準で，AT 領域を選択することを指定.
  ⟨ 条件式 ⟩ :: =[（`min` ⟨ 変数名 ⟩ | `condition` ⟨ 条件 ⟩）⟨ 接続演算子 ⟩]
  ⟨ 接続演算子 ⟩ :: = [`.and.` | `.or.`] ⟨ 条件式 ⟩
  `estimated` ⟨ 数式 ⟩: ユーザが定義した選択に関するコスト（数式）を基に，最適な処理を選択する処理であることを指定.
  (機能名 select 指定時)
- `varied from X to Y` : 指定するパラメタの変動範囲（X から Y まで）の指定.
  （機能名 variable, unroll 指定時）
- `fitting` ⟨ 方式 ⟩ `sampled` ⟨ 範囲 ⟩ ：パラメタの推定に利用する方式を指定.
  ⟨ 方式 ⟩ :: =[`least-square` ⟨ 次数 ⟩ | `user-defined` ⟨ 数式 ⟩ | `auto`]
  `least-square` ：多項式による最小二乗法でパラメタ推定することを指定.
  ⟨ 次数 ⟩ で，多項式の次数を設定.
  `user-defined` ：ユーザ指定による数式を用いて最小二乗法で推定.
  `auto` : システム側にパラメタの推定を任せる.
  ⟨ 範囲 ⟩ :: = [⟨ 数値 ⟩ | `auto`]：パラメタ推定に必要な範囲を指定.
  なお ⟨ 方式 ⟩ =`auto` 指定時は省略可能.
  ⟨ 数値 ⟩：具体的なパラメタの数値を指定.
  `auto` ： パラメタのサンプリング間隔を自動決定．fitting 補助指定子省略時，varied 補助指定子で指定した範囲全てを測定（＝全探索）して最適パラメタを決める．
  （機能名 variable, unroll 指定時）
- `pripro sub region` (`start` | `end`)：AT 領域を呼び出す前に適用する処理を指定.
  (全ての機能で利用可能)
- `postpro sub region` (`start` | `end`)：AT 領域を呼び出した後に適用する処理を指定.
  (全ての機能で利用可能)
- `debug` (⟨ 変数 ⟩, [⟨ 変数 ⟩, …])：AT 領域が実行されるときに，デバック表示する変数を指定.
  （全ての機能で利用可能）
  ⟨ 変数 ⟩ :: = [`bp` | `pp` | **任意変数**]
  `bp` ： 基本パラメタ情報を表示する,
  `pp` ： 性能パラメタ情報を表示する,
  任意変数 ： 記述された変数情報を表示する.

図 4 ABCLibScript における補助指定子

ABCLib_INSTALL：インストール時自動チューニング；ABCLib_STATIC：実行起動前自動チューニング；ABCLib_DYNAMIC：実行時自動チューニング；ABCLib_ALL：すべての自動チューニング；

引数ABCLib_ATroutinesは，対象となるAT領域について指定する引数である．この引数はヘッダファイルABCLibScript.hで定義されている型ABCLib_ATnameを用いてユーザが宣言するか，ABCLibScript.hでcommon定義されている大域変数，ABCLib_AllRoutines：すべてのルーチン用；ABCLib_InstllRoutines：インストール時自動チューニングルーチン用；ABCLib_StaticRoutines：実行起動前自動チューニングルーチン用；ABCLib_DynamicRoutines：実行時自動チューニングルーチン用；を利用して指定する．

[使用例]

```
!ABCLib$ call ABCLib_ATexec
  (ABCLib_INSTALL, ABCLib_InstallRoutines)
```

：インストール時自動チューニングについて，インストール時最適化を指定した全てのAT領域について行う．

【API 2】ABCLib_ATset
(ABCLib_ATkinds, ABCLib_ATroutines)

ABCLib_ATset手続きは，ABCLib_ATroutinesで指定したAT領域名を，ABCLib_ATkindで指定する自動チューニングの種類として設定する．

【API 3】 ABCLib_ATdel
(ABCLib_ATroutines, DelName)

ABCLib_ATdel手続きは，ABCLib_ATroutinesで指定したAT領域名情報が入っている変数から，DelNameで指定するAT領域名を削除する．引数DelNameに削除したいAT領域名を記述する．なおAT領域名は，name補助指定子を用いて各AT領域に記述しておく．

[使用例]

```
!ABCLib$ call ABCLib_ATdel
  (ABCLib_InstallRoutines,"MyMatMul")
```

：インストール時最適化を行う候補から，”MyMatMul”と名付けたAT領域を除外する．

【API 4】ABCLib_ATInstallInit
(ABCLib_InstallRoutines)

ABCLib_ATInstallInit手続きは，ABCLib_InstallRoutinesで指定されるAT領域名のインストール時自動チューニングを未実行とする．

【API 5】ABCLib_BPset (BPvalName)

ABCLib_BPset手続きは，引数BPvalNameに指定したパラメタ名を，新たな基本パラメタBPとする．

[使用例]

```
!ABCLib$ call ABCLib_BPset( "nprocs" )
```

：変数nprocsを基本パラメタ（BP）とする．

【API 6】 ABCLib_BPsetName
(Kind, BPvalName, Name)

ABCLib_BPsetName手続きは，引数Kindで指定される自動チューニングを行う際の基本パラメタBPの固定に関する変数について，引数BPvalNameに指定した基本パラメタ名に関する名称を引数Nameとする．ここで引数Kindは，以下のとおりである．Kind ::= [STARTTUNESIZE | ENDTUNESIZE | SAMPDIST]. また，これら変数の意味は以下の通りである．STATTTUNESIZE：基本パラメタBPvalNameに関するサンプリング開始点情報；ENDTUNESIZE ： 基本パラメタBPvalNameに関するサンプリング終了点情報；SAMPDIST：基本パラメタBPvalNameに関するサンプリング間隔情報．

[使用例]

```
!ABCLib$ call ABCLib_BPsetName("STARTTUNESIZE",
   "nprocs","ABCLib_NprocsStartSize")
```

：基本パラメタnprocsにおける自動チューニング開始点を指定する変数をABCLib_NprocsStartSizeとする．

【API 7】 ABCLib_BPsetCDF
(BPvalName, CDFKind)

ABCLib_BPsetCDF手続きは，引数BPvalNameに指定した，基本パラメタに関するサンプリング点以外のパラメタ値の推定方式を，引数CDFKindで指定される種類のコスト定義関数とする．ここで，引数CDFKindは，以下のとおりである．CDFKind ::= [least-square ⟨ 次数 ⟩ | user-defined ⟨ 式 ⟩ | auto]. また，これら変数の意味は以下の通りである：least-square ⟨ 次数 ⟩： 多項式による最小二乗法でパラメタ推定することを指定する．⟨ 次数 ⟩で多項式の次数を指定する；user-defined ⟨ 式 ⟩： ユーザ定義による式を用いて最小二乗法でパラメタ推定する；auto：システム側にパラメタ推定の方式決定を任せる；

なおデフォルトでは，AT領域で定義されたコスト定義関数と同一の方式を用いて推定する．さらにユーザがAT領域でのコスト定義関数指定を省略した場合には，3次多項式を用いた最小二乗法による推定方式が選択される．

[使用例]

```
!ABCLib$ call ABCLib_BPsetCFD("nprocs",
   "least-square 5")
```

：基本パラメタnprocsに関するパラメタ推定方式を，5次多項式を用いた最小二乗法で行うことを指定する．

4.2 API記述例

例えば，ソフトウエア開発者が所有するサブルーチンEigenSolverに関して，ソフトウエア開発者が記述する自動チューニング処理手順を記述したサブルーチ

ンfoo は，ABCLibScript の API を用いて以下のように記述できる．

```
subroutine foo(...)
include (ABCLibScript.h)
  …
C ===AT 領域の登録など初期化
!ABCLib$ call ABCLib_ATset
!ABCLib$ & (ABCLib_ALL,ABCLib_AllRoutines)
!ABCLib$ call ABCLib_ATset(ABCLib_INSTALL,
!ABCLib$ & ABCLib_InstallRoutines)
!ABCLib$ call ABCLib_ATset(ABCLib_STATIC,
!ABCLib$ & ABCLib_StaticRoutines)
!ABCLib$ call ABCLib_ATset(ABCLib_DYNAMIC,
!ABCLib$ & ABCLib_DynamicRoutines)
C ===インストール時自動チューニングの実行
C  (以下はソフトウエア開発者が記述)
C  !全体を通して 1 回のみしか実行できない
!ABCLib$ call ABCLib_ATexec(ABCLib_INSTALL,
!ABCLib$ & ABCLib_InstallRoutines)
C        ===インストール時最適化のみ実行済
!ABCLib$ call EigenSolver(…)
  …
C ===実行時自動チューニングの実行許可
C  (以下はソフトウエア開発者が記述)
C  !この時点では実行されない．
C   対象の以下の EigenSolver コール後の
C     指定箇所実行時に行われる
!ABCLib$ call ABCLib_ATexec (ABCLib_DYNAMIC,
!ABCLib$ & ABCLib_DynamicRutines)
C    === 呼び出し後，対象箇所でチューニング
C    (インストール時，実行起動前 (エンドユーザ
C    が行う)，および実行時最適化が実行済)
!ABCLib$ call EigenSolver(…)
…
return
end
```

ソフトウエア開発者が提供したソフトウエアの機能である EigenSolver の利用に関して，エンドユーザが記述する実行起動前最適化処理を実行するサブルーチンpooh は，ABCLibScript の API を用いて以下のように記述する．

```
subroutine pooh(...)
include (ABCLibScript.h)
  …
C ===実行起動前自動チューニングの実行
C  (以下はエンドユーザが記述)
C  !この時点で実行される
C      ===ソフトウエア開発者定義の BP の固定
       N_TUNESIZE_START=1234
       N_TUNESIZE_END=1234
       call ABCLib_ATexec(ABCLib_STATIC,
  &            ABCLib_StaticRoutines)
C    ===インストール時，実行起動前最適化実行済
       call EigenSolver(…)
  …
return
end
```

このように実行起動前自動チューニングの記述は，エンドユーザにとって負担が大きい．この負担を削減するため，専用 GUI の開発が必要である．

5. ABCLibScript によるプログラム例

以下は，ソフトウエア開発者が ABCLibScript を用いて自動チューニング処理指定を行う場合の，プログラミング事例について述べる．

5.1 インストール時最適化

以下のプログラム例 1 は，インストール時最適化において行列積コードのアンローリング段数調整処理を記述した例である．

【プログラム例 1】行列積コードのアンローリング

```
!ABCLib$ install unroll (i) region start
!ABCLib$ name MyMatMul
!ABCLib$ varied (i) from 1 to 16
!ABCLib$ fitting least-square 5
!ABCLib$ & sampled (1-5, 8, 16)
do i=1, n
  do j=1, n
    da1=A(i,j)
    do k=1, n
      dc=C(k,j)
      da1=da1+B(i,k)*dc
    enddo
    A(i,j)=da1
  enddo
enddo
!ABCLib$ install unroll (i) region end
```

プログラム例 1 では，BP について特にソフトウエア開発者が指定していないことを前提とするので，デフォルトの基本パラメタ BP を問題サイズに関する変数 n としている．またunroll 指定子を用い，かつループ変数 i を指定しているので，性能パラメタ PP は対象となる行列積ループの最外側ループ変数 i に対するアンローリング段数となる．

varied 補助指定子から，アンローリングは 1 段から 16 段まで行うことを定義している．また，fitting 補助指定子から最適化の対象となるコスト定義関数は 5 次多項式となる．unroll 指定子が使われているので，対象となる AT 領域の実行時間の挙動に関するモデル化を指定したことになる．

システムとしては，パラメタ BP を API で指定した情報を用いて固定し，パラメタ PP を推定する．このパラメタ PP については，sampled 補助指定子に

よるサンプリング点 (＝アンローリング段数) で指定されている，1 段から 5 段，8 段，および 16 段の点において実行時間を実測する．この実測結果を用いて指定された 5 次多項式モデルによる PP の挙動を決定する (＝ 5 次多項式の係数を決定する) が，その方法として最小二乗法を指定 (least-squre 補助指定子から) している．

以下のプログラム例 2 は，ブロック化アルゴリズムを用いて実装されているカーネルサブルーチンをコールする場合における，ブロック幅調整処理を記述した例である．

【プログラム例 2】ブロック幅調整

```
!ABCLib$ install variable (MB) region start
!ABCLib$ name BlkMatMal
!ABCLib$ varied (MB) from 1 to 64
do i=1, n, MB
  call MyBlkMatVec(A,B,C,n,i)
enddo
!ABCLib$ install variable (MB) region end
```

プログラム例 2 では，variable 指定子により性能パラメタ PP がブロック幅変数 MB であることを指定する．このブロック幅変数 MB の範囲はvaried 補助指定子により，1 から 64 であることを指定している．

5.2 実行起動前最適化

以下のプログラム例 3 は，アルゴリズム選択処理に関するプログラミング例である．アルゴリズム選択基準として，ソフトウエア開発者が定義するコスト定義関数を利用する．

【プログラム例 3】ソフトウエア開発者が定義したコスト定義関数によるアルゴリズム選択

```
!ABCLib$ static select region start
!ABCLib$ name TestSelect
!ABCLib$ parameter (in CacheS,in NB,in NPrc)
!ABCLib$   select sub region start
!ABCLib$   according estimated
!ABCLib$ & (2.0d0*CacheS*NB)/(3.0d0*NPrc)
              AT 領域 1
!ABCLib$   select sub region end
!ABCLib$   select sub region start
!ABCLib$   according estimated
!ABCLib$ & (4.0d0*CacheS*dlog(NB))
!ABCLib$ &  /(2.0d0*NPrc)
              AT 領域 2
!ABCLib$   select sub region end
!ABCLib$ static select region end
```

プログラム例 3 では実行起動前最適化において，アルゴリズムの選択処理をすることを指定している．性能パラメタ PP として，AT 領域 1 もしくは AT 領域 2 の実行指定がパラメタ化される．

ソフトウエア開発者が定義するコスト定義関数は，according estimated 補助指定子で定義されている．この例でのコスト定義関数では，この最適化が行われる前に定義された浮動小数点変数 (CacheS, NB, NPrc) が参照されている．AT 領域 1 のコストは $(2.0d0 * CacheS * NB)/(3.0d0 * NPrc)$ で見積もられ，AT 領域 2 のコストは $(4.0d0 * CacheS * dlog(NB))/(2.0d0 * NPrc)$ で見積もられる．

これらコスト評価は実行起動前自動チューニングモードで行われ，実行する AT 領域が 1 つ決定される．その後対象となる AT 領域の実行時において，決定された AT 領域のみが選択的に実行される．

5.3 実行時最適化

以下のプログラム例 4 は，AT 領域実行時において，AT 領域 (アルゴリズム) 中で定義参照される変数 (eps, $iter$) をもちいて最適な AT 領域 (アルゴリズム) を選択する処理の例である．具体的には，eps が最小となるような AT 領域を条件 $iter < 5$ 以内で決定する．このプログラム例 4 は，反復解法における前処理方式の自動選択を指向している．

【プログラム例 4】反復解法における前処理方式の実行時選択

```
!ABCLib$ dynamic select (eps,iter)
!ABCLib$ & region start
!ABCLib$ name PricondSelect
!ABCLib$ parameter (in eps, in iter)
!ABCLib$ according min (eps) .and.
!ABCLib$ &  condition (iter<5)
!ABCLib$   select sub region start
             AT 領域 1（前処理 1）
             ...
             eps = ...
!ABCLib$   select sub region end
!ABCLib$   select sub region start
             AT 領域 2（前処理 2）
             ...
             eps = ...
!ABCLib$   select sub region end
!ABCLib$ dynamic select (eps,iter)
!ABCLib$ & region end
```

プログラム例 4 では実行時において，浮動小数点変数 eps と $iter$ に基づきアルゴリズムの選択処理をすることを指定している．性能パラメタ PP として，AT 領域 1 もしくは AT 領域 2 の実行指定がパラメタ化される．

ソフトウエア開発者が定義するコスト定義関数は，対象となる AT 領域が実行される前に定義されている変数 eps と $iter$ を用いて定義される．このコスト定義関数は，$iter$ が 5 より小さい場合については各 AT 領域における eps の値を保存しておき，$iter$ の値が 5 以上になったとき eps の値が最も小さい AT 領域を検索して選択する関数である．

6. 関連研究

パラメタ自動チューニングに関する枠組やソフトウエアの研究は，以下の 2 種に分類できる．

まず計算機システムにおける実行時最適化の枠組をもつソフトウエアがある．たとえば，パラメタ (I/O バッファサイズなど) の決定を実行時に行うソフトウエアとして Active Harmony[8]，および Autopilot[9] がある．

つぎに数値計算ソフトウエアにおけるインストール時最適化の枠組をもつソフトウエアがある．例えば，PHiPAC[1]，ATLAS[2] および経験的手法を用いた枠組の AEOS[10], および FFTW[3] が知られている．

自動チューニングの定式化では，SIMPL[11] においてインストール時最適化に関する定式化を行った．この SIMPL において，性能パラメタ (PP) と基本パラメタ (BP) の概念がはじめて提案された．

一方，今村と直野[12] は，対象となるコードのレジスタ数などを算出し，固有値ソルバ中の特定処理に関するアンローリング処理の最適段数を決定する式を導出した．その結果パラメタ空間の探索に関し，大幅なコストの削減に成功した．今村と直野により導出された式は，固有値計算におけるアンローリング処理の振舞いを実行時間において定義したコスト定義関数と解釈でき，ABCLibScript での記述や ABCLibScript で用いる新しい最適化方式として実装が期待できる．

また Brewer[14] はソースコードの構成情報や実行時間計測から，対象となるプログラムの実行時間に関する挙動を自動的にモデル化する方式を提案し，これをソフトウエアのアルゴリズム選択に利用した．Brewer の自動モデル化方式は，ABCLibScript でのユーザ定義によるコスト定義関数指定の自動化を行う方式であると解釈できる．

しかしながらこれらの研究のいずれも，本論文で示した ABCLibScript のように，自動チューニング処理自体の容易な記述を目指した言語処理系開発に関連するものではない．

7. おわりに

この論文では，FIBER による自動チューニング処理記述用ディレクティブである ABCLibScript の設計方針と実装について述べた．ABCLibScript は，処理対象を数値計算に限定した自動チューニング機能を提供することで，自動チューニングの付加を行うソフトウエア開発者において，自動チューニングソフトウエア作成の際の労力の削減とその開発支援をねらう．

FIBER は，2 種のユーザであるソフトウエア開発者およびエンドユーザの知識を自動チューニングに利用する枠組であるといえる．すなわちソフトウエア開発者の観点では，彼らが有する開発ソフトウエアに関する知識をディレクティブである ABCLibScript を用いて記述することで，(1) 有効となるパラメタ抽出，(2) 対象となる領域，および (3) 必要な経験値 (ループアンローリング段数やブロック幅の上限など) を，自動チューニング機構に周知させることができる．一方エンドユーザの観点では，実行起動前最適化において，実行すべき問題サイズや有効となるアルゴリズムなどの知識情報を，自動チューニング機構に周知させることができる．

現在，FIBER フレームワークに基づく自動チューニング機能付き並列数値計算ライブラリ *ABCLib* を公開している．この *ABCLib* は，科学技術振興機構さきがけプログラム「情報基盤と利用環境」領域の研究成果の一部として開発された．また ABCLibScript の機能の一部 (アンローリング処理等) を実装した試用版を開発済である．発表の際には，この試用版 ABCLibScript のデモを行う．なお *ABCLib* のソースコード，および ABCLibScirpt 試用版は，科学技術振興機構の助成による WWW サーバ (`http://www.abc-lib.org/`) 上でマニュアル等を含め公開される予定である．

今後の課題として，以下の事項があげられる．

- **入れ子指定子に対する処理：** 指定子の中に指定子を記述する場合，どのように実行がなされるのか，もしくは入れ子が可能な指定子を仕様として定める必要がある．また，入れ子の数の上限を決める必要がある．一般的に指定子を入れ子にすると，パラメタの探索空間が指数関数的に大きくなる．したがって，全探索をもちいた最適化の実行が極めて困難となる．ゆえに，効率の良い (もしくは手を抜いた) 入れ子指定子に関するパラメタ探索手法の検討，およびその手法を利用するための仕様決定を行う必要がある．
- **コスト定義関数の自動設定に関する仕様拡張：** 現在の ABCLibScript の仕様では，コスト定義関数の記述や，その関数の推定精度などは全てソフトウエア開発者が責任をもつことが前提となっている．もしソフトウエア開発者が取り扱うプログラムの特性について熟知していない場合は，自動チューニング機能を付加できない．そのような場合でも自動チューニング機能の付加ができるように，コスト定義関数が自動設定できる手法の検討，およびそれに伴う仕様決定が必要である．
- **性能安定化機能指定子の追加：** 今村と直野が提案している，性能安定化に関する実行時最適化機構[13] を本仕様に拡張することも重要な今後の課題である．
- **Grid 環境や PDA 環境を考慮した仕様拡張：** 現在の ABCLibScript の仕様は，スーパーコンピュータ環境や均質 PC クラスタ環境を想定して，仕様が決められている．FIBER プロジェクトでは，Grid 環境や PDA 環境での自動チューニングもターゲットとしている．したがって，これらの環

境特性を考慮したABCLibScriptの仕様拡張も今後の課題である．

謝辞 有益なコメントを頂いた査読者の各位に感謝いたします．なお本研究は，科学技術振興機構さきがけプログラムの助成による．

参 考 文 献

1) Bilmes, J., Asanović, 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) ATLAS project; http://www.netlib.org/atlas/index.html.
3) Frigo, M.: A Fast Fourier Transform Compiler, *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation*, Atlanta, Georgia, pp. 169–180 (1999).
4) 片桐孝洋, 黒田久泰, 大澤清, 工藤誠, 金田康正: 自動チューニング機構が並列数値計算ソフトウエアに及ぼす効果, 情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム, Vol.42, No. SIG 12 (HPS 4), pp. 60–76 (2001).
5) Takahiro, K., Kise, K., Honda, H. and Yuba, T.: FIBER: A Framework of Installation, Before Execution-invocation, and Run-time Optimization Layers for Auto-tuning Software, *IS Technical Report, Graduate School of Information Systems, The University of Electro-Communications*, UEC-IS-2003-3 (May 2003).
6) Takahiro, K., Kise, K., Honda, H. and Yuba, T.: FIBER: A General Framework for Auto-Tuning Software, *Proceedings of The Fifth International Symposium on High Performance Computing*, Springer Lecture Notes in Computer Science, No. 2858, pp. 146–159 (2003).
7) 片桐孝洋, 吉瀬謙二, 本多弘樹, 弓場敏嗣: FIBER: 汎用的な自動チューニング機能の付加を支援するソフトウエア構成方式, 情報処理学会研究報告, Vol. 2003-HPC-94, pp. 1–6 (2003).
8) Tapus, C., Chung, I.-H. and Hollingsworth, J. K.: Active Harmony : Towards Automated Performance Tuning, *Proceedings of High Performance Networking and Computing (SC2002)*, Baltimore, USA (2003).
9) Ribler, R. L., Simitci, H. and Reed, D. A.: The Autopilot Performance-Directed Adaptive Control System, *Future Generation Computer Systems, special issue (Performance Data Mining)*, Vol. 18, No. 1, pp. 175–187 (2001).
10) Whaley, R., Petitet, A. and Dongarra, J. J.: Automated Empirical Optimizations of Software and the ATLAS Project, *Parallel Computing*, Vol. 27, pp. 3–35 (2001).
11) 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, No. 2001-HPC-87, pp. 25–30 (2001).
12) 今村俊幸, 直野健: 性能安定化を目指した自動チューニング型固有値ソルバーについて, SACSIS2003 論文集, pp. 145–152 (2003).
13) 今村俊幸, 直野健: キャッシュ競合を制御する性能安定化機構内蔵型数値計算ライブラリについて, HPCS2004 論文集, pp. 173–180 (2004).
14) Brewer, E. A.: Portable High-Performance Supercomputing: High-Level Platform-Dependent Optimization, Technical report, Ph.D Thesis, Massachusetts Institute of Technology (1994).