AT研究コミュニティで開発されたソフトウェア群と今後の展望

「ソフトウェア自動チューニング:
科学技術計算のためのコード最適化技術」
出版記念講演

名古屋大学情報基盤センター
片桐孝洋

第13回 自動チューニング技術の現状と応用に関するシンポジウム(ATTA2021)
日時： 2021年12月13日（金）10:40-11:00
場所：オンライン開催

本発表の流れ

- 書籍の紹介
- AT研究会関係者が開発されたソフトウェア
- 今後の展望

本発表の流れ

- 書籍の紹介
- AT研究会関係者が開発されたソフトウェア
- 今後の展望

書籍の紹介

森北出版

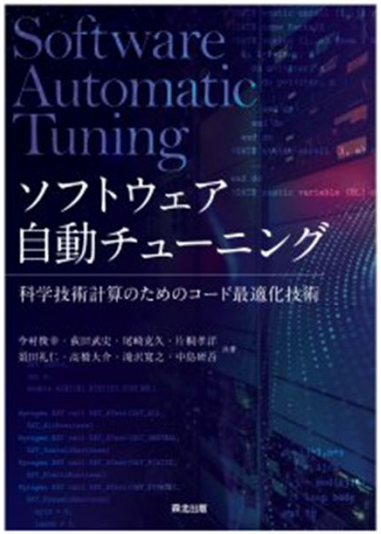
本を探す

HOME › 情報工学・コンピュータ ― 数値計算　情報工学・コンピュータ ― プログラミング言語

ソフトウェア自動チューニング

科学技術計算のためのコード最適化技術

紙版　電子版



試し読み

理化学研究所チームリーダー Ph.D. 今村俊幸 (共著)
東京女子大学教授 博(情報科学) 荻田武史 (共著)
芝浦工業大学教授 博(工) 尾崎克久 (共著)
名古屋大学教授 博(理) 片桐孝洋 (共著)
東京大学教授 博(理) 須田礼仁 (共著) 筑波大学教授 博(理) 高橋大介 (共著)
東北大学教授 博(情報科学) 滝沢寛之 (共著)
東京大学教授 博(工) 中島研吾 (共著)

定価　¥5,720

ページ　320
判型　菊
ISBN　978-4-627-87221-9
発行年月　2021.09

参照 : https://www.morikita.co.jp/books/mid/087221

書籍概要

- タイトル
  - ソフトウェア自動チューニング：
    科学技術計算のためのコード最適化技術
- 著者
  1. 理化学研究所チームリーダー Ph.D. 今村俊幸
  2. 東京女子大学教授 博(情報科学) 荻田武史
  3. 芝浦工業大学教授 博(工) 尾崎克久
  4. 名古屋大学教授 博(理) 片桐孝洋
  5. 東京大学教授 博(理) 須田礼仁
  6. 筑波大学教授 博(理) 高橋大介
  7. 東北大学教授 博(情報科学) 滝沢寛之
  8. 東京大学教授 博(工) 中島研吾
- ページ320
- 判型菊
- ISBN978-4-627-87221-9
- 発行年月2021.09

書籍目次

- 第1章
  ソフトウェア自動チューニングとは
- 第2章
  自動チューニングツール
- 第3章
  自動チューニング機能付き数値計算ライブラリ
- 第4章
  数値計算ライブラリと自動チューニング機能の展望
- 第5章
  これからの数値計算：計算結果の保証
- 第6章
  ポストムーア時代における自動チューニング

自動チューニングの手順

1. プログラム確定

```
do i=1, n
  do j=1, n
    do k=1, n
      C( i, j ) = C( i, j ) + A( i, k ) * B( k, j )
    enddo
  enddo
enddo
```

2. 性能パラメタ抽出

$$F = (x_1, x_2, \cdots, x_n)$$

3. 性能定義

$F(X)$

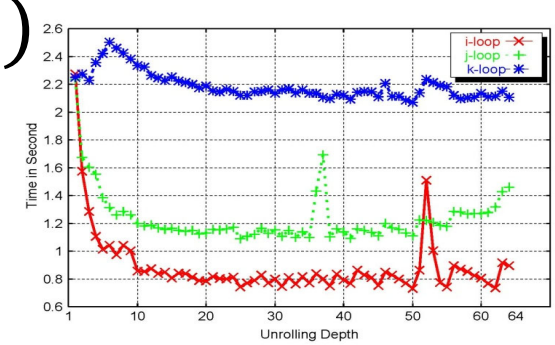
```
!oat$ install unroll (i,k) region start
!oat$ name MyMatMul
!oat$ varied (i,k) from 1 to 8
do i=1, n
  do j=1, n
    do k=1, n
      C( i, j ) = C( i, j ) + A( i, k ) * B( k, j )
    enddo
  enddo
enddo
!oat$ install unroll (i,k) region end
```

4. 性能チューニング

$\min F(X)$

X^*

を見つける



5. 最適実行

X^*を用いて

プログラムを実行

このコストが大
＝自動化による
工数削減

本発表の流れ

- 書籍の紹介
- AT研究会関係者が開発されたソフトウェア
- 今後の展望

1. プログラム確定

```
do i=1, n
  do j=1, n
    do k=1, n
      C( i, j ) = C( i, j ) + A( i, k ) * B( k, j )
    enddo
  enddo
enddo
```

2. 性能パラメタ抽出

$$F = (x_1, x_2, \cdots, x_n)$$

3. 性能定義

$F(X)$

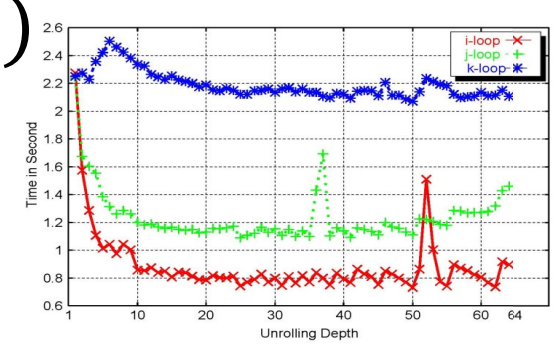
```
!oat$ install unroll (i,k) region start
!oat$ name MyMatMul
!oat$ varied (i,k) from 1 to 8
do i=1, n
  do j=1, n
    do k=1, n
      C( i, j ) = C( i, j ) + A( i, k ) * B( k, j )
    enddo
  enddo
enddo
!oat$ install unroll (i,k) region end
```

4. 性能チューニング

$\min F(X)$

X^*

を見つける



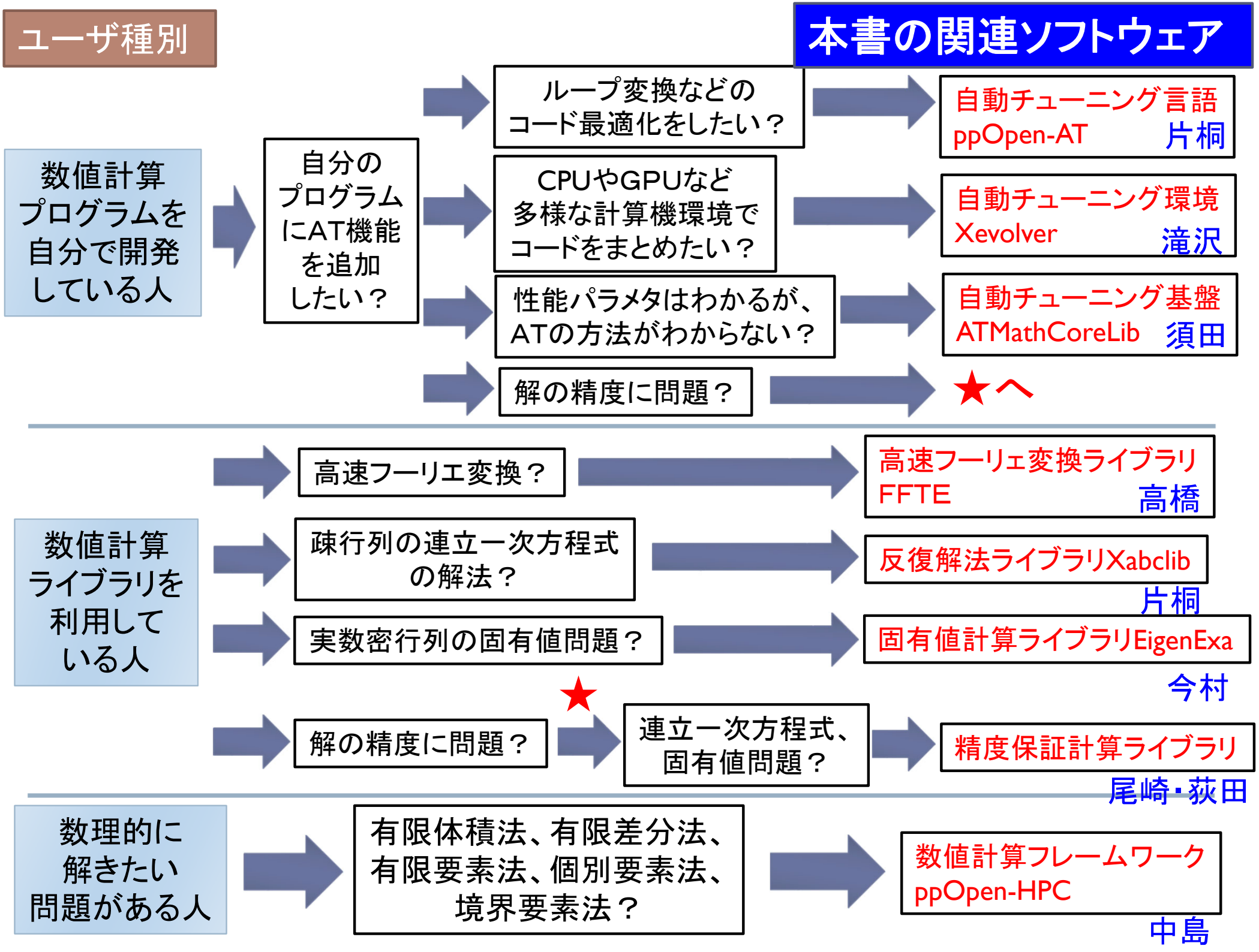
5. 最適実行

X^*を用いて

プログラムを実行

本書の関連ソフトウェア（AT研究会関連者が開発）

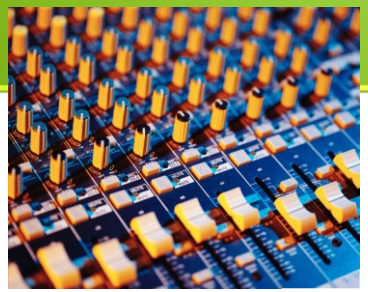
- ppOpen-AT　片桐
- Xevolver　滝沢
- ATMathCoreLib　須田
  (数値計算ライブラリ)
  - FFTE　高橋
  - Xabclib　片桐
    - ppOpen-HPC (一部)　中島


ユーザ種別
本書の関連ソフトウェア
数値計算プログラムを自分で開発している人
自分のプログラムにAT機能を追加したい？
ループ変換などのコード最適化をしたい？
自動チューニング言語
ppOpen-AT 片桐
CPUやGPUなど多様な計算機環境でコードをまとめたい？
自動チューニング環境
Xevolver 滝沢
性能パラメタはわかるが、ATの方法がわからない？
自動チューニング基盤
ATMathCoreLib 須田
解の精度に問題？
★へ
数値計算ライブラリを利用している人
高速フーリエ変換？
高速フーリェ変換ライブラリ
FFTE 高橋
疎行列の連立一次方程式の解法？
反復解法ライブラリXabclib
片桐
実数密行列の固有値問題？
固有値計算ライブラリEigenExa
今村
★
解の精度に問題？
連立一次方程式、固有値問題？
精度保証計算ライブラリ
尾崎・荻田
数理的に解きたい問題がある人
有限体積法、有限差分法、有限要素法、個別要素法、境界要素法？
数値計算フレームワーク
ppOpen-HPC
中島

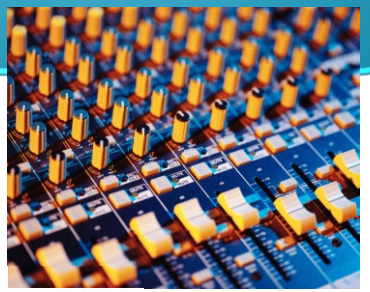
AT機構とは

•計算機アーキテクチャ
•計算機システム

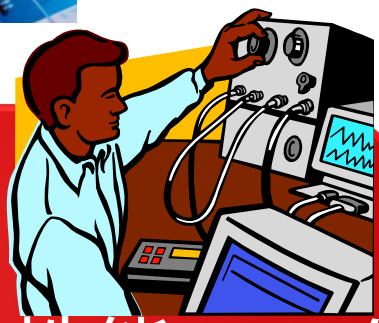
•プログラム
•アルゴリズム



性能調整つまみ
（性能パラメタ）



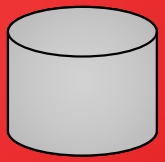
調整機構
•最適化
•パラメタ探索
•学習／自己適応



性能モニタ機構

•プログラム
•アルゴリズム

つまみ自動生成機構



AT性能データベース

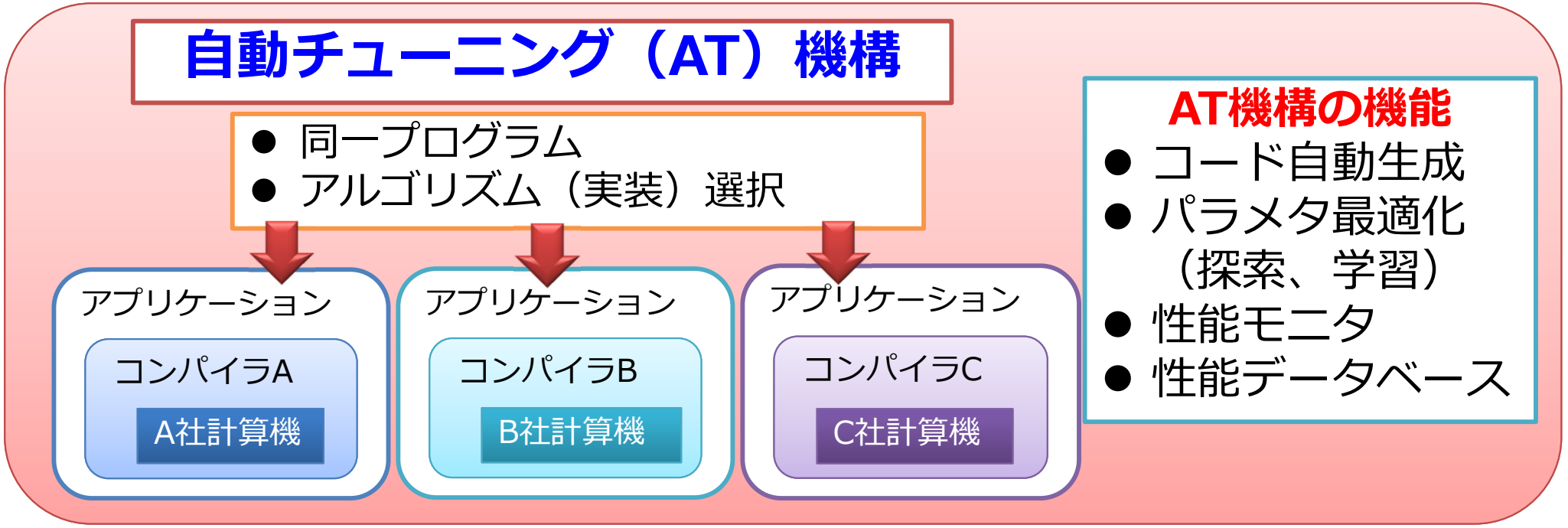
自動チューニング機構

性能可搬性(Performance Portability)とは？

- 複数計算機環境での最適化を提供するパラダイム

（HPCI 技術ロードマップ白書、数値計算ライブラリのための自動チューニング2012 年3 月）

- 2000年頃から日本で使われている技術用語
- 同一プログラムで**計算機環境**が変わっても高性能を維持



GPU

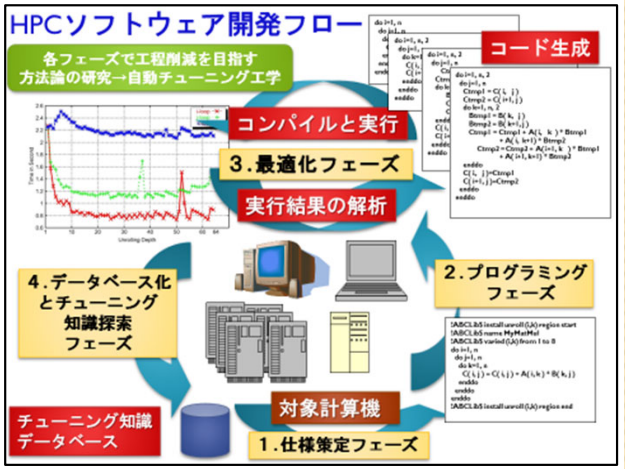
マルチコア CPU

FPGA？

量子コンピュータ？

ポストムーアに向けた多様なハードウェア環境に対応

2.1節　AT専用言語ppOpen-AT
ソフトウェア開発手順



ソフトウェア
開発者

●計算機環境に非依存
●コード、計算機資源、
電力、精度、の最適化方式を記述

ppOpen-ATによる
自動チューニング記述

```
#pragma OAT install unroll (i,j,k) region start
#pragma OAT varied (i,j,k) from 1 to 8
for(i = 0 ; i < n ; i++){
 for(j = 0 ; j < n ; j++){
  for(k = 0 ; k < n ; k++){
   A[i][j]=A[i][j]+B[i][k]*C[k][j]; }}}
#pragma OAT install unroll (i,j,k) region end
```

専用言語処理系
（プリプロセッサ）の起動

AT機構が付加された
プログラム

最適化候補とAT機構が
付加された実行可能コード

■自動生成
される機構
●最適化候補
●性能モニタ
●パラメタ探索
●性能モデル化

コンパイラ
ではできない
最適化

ループアンローリングの例（行列-行列積、Fortran言語）

- i-ループ および j-ループ 2段展開 (nが２で割り切れる場合)

```
do i= 1 , n,  2
  do j= 1 , n,  2
    do k= 1 ,  n
      C(i     , j     ) = C(i     , j     ) +A(i     , k) * B(k, j     )
      C(i     , j+ 1 ) = C(i     , j+ 1 )+A(i     , k) * B(k, j+ 1 )
      C(i+ 1 , j     ) = C(i+ 1 , j     )+A(i+ 1 , k) * B(k, j     )
      C(i+ 1 , j+ 1 ) = C(i+ 1 , j+ 1 )+A(i+ 1 , k) * B(k, j + 1 )
enddo;  enddo;  enddo;
```

- A(i, j)、A(i+ 1 , k)、B(k, j)、B(k, j+ 1)をレジスタに置き高速化
- 何段が最適かは計算機依存（段数が＜性能パラメタ＞）

FIBERフレームワーク[Katagiri et.al., 2003

高度な知識が必要

ライブラリ開発者

ユーザ知識

①

オリジナルコード

ライブラリ公開前

ディレクティブによる記載

②

自動コード生成

公開ライブラリ

知識不要！

選択

候補 *n*

3

2

1

オートチューナー

実行時

⑥

自動チューニングされたコード実行

⑤

③

④ 実行時間

ライブラリ呼び出し

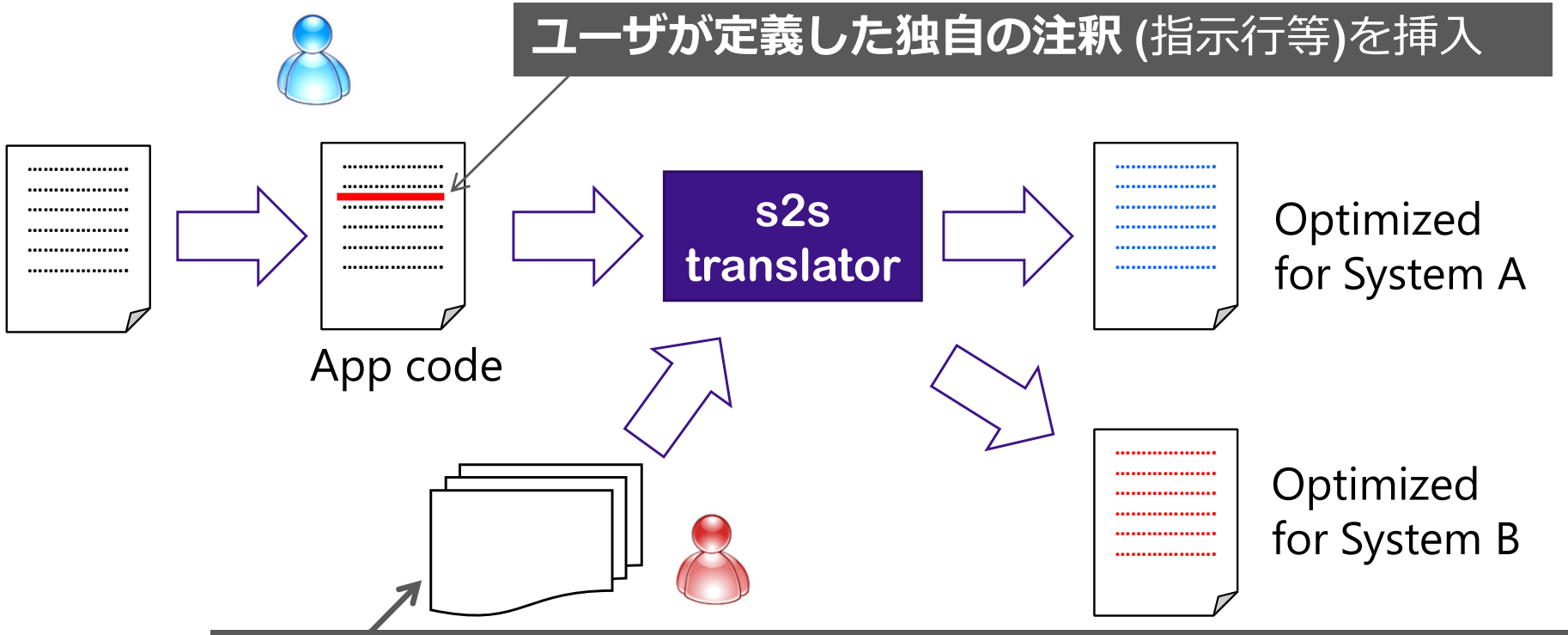
Xabclib、ABCLib、ppOpen-AT (ABCLibScript) のAT方式

ライブラリユーザ

：対象計算機

2.2節 Xevolverフレームワーク

任意のコード修正を置き換えるはためには多種多様のコード変換が必要
　　＝すべてのコード変換を事前に用意することは不可能
→ Xevolver：ユーザが独自のコード変換を定義して利用するためのフレームワーク

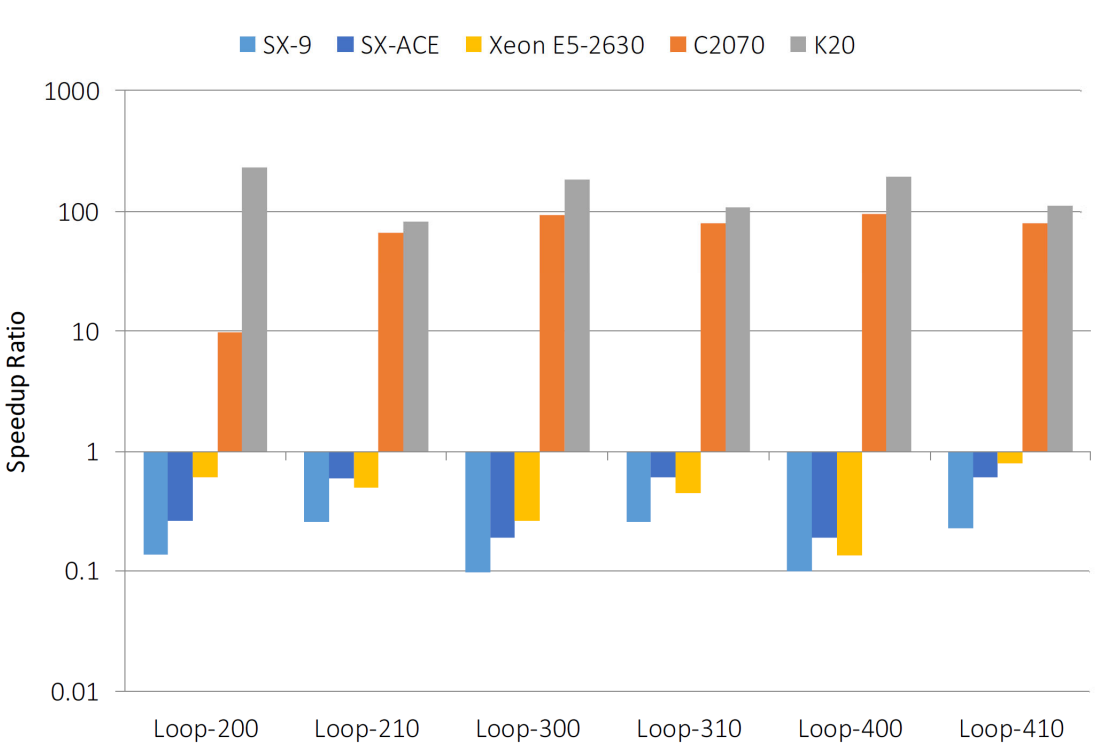


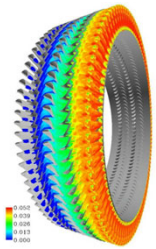
Xevolver 適用事例

Xevolver：ユーザ定義コード変換のためのフレームワーク

手作業によるコード修正を**コード変換パターン**として表現

→ GPUで実行時には変換、 SXで実行時には無変換
＝GPUとSXの両方で高性能を達成可能

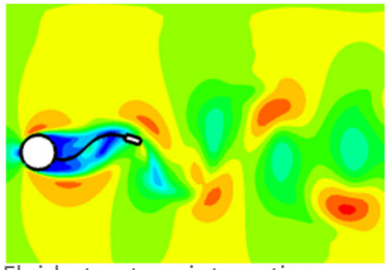




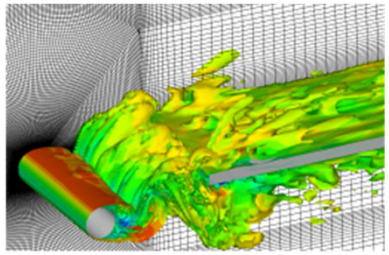
変換前のコードパターン

```
program nt_opt
!$xev tgen var(i1,i2,i3,i4,i5,i6,if) stmt
!$xev tgen list(body) stmt
!$xev tgen var(lstart,lend,II2,IIF) exp
!$xev tgen condef(has_doi) contains stmt begin
  DO I=II2,IIF
!$xev tgen stmt(if)
!$xev tgen stmt(body)
  END DO
!$xev tgen end
!$xev tgen list(stmt_with_doi) stmt cond(has_doi)
!$xev tgen src begin
  DO L=lstart,lend
!$xev tgen stmt(stmt_with_doi)
  END DO
!$xev end tgen src
!$xev tgen dst begin
  DO I=1,inum
    DO L = lstart, lend
      IF (I .GE. IS(L) .AND. I .LE. IT(L)) THEN
        EXIT
      END IF
    END DO
!$xev tgen stmt(if)
!$xev tgen stmt(body)
  END DO
!$xev end tgen dst
end program nt_opt
```

変換後のコードパターン



Fluid-structure interactions



3D flow around cylinder-plate configuration

国際共同研究(DFG ExaFSA)における適用事例

Xevolver-C

■ ユーザ定義コード変換を記述するためのDSL

- C言語＋特殊な変数と文法

```
int i,j;
xev_stmt* any_stmt;

int main(int argc, char** argv){
#pragma xev stmt src("label1")
  {
    for(i=0;i<10;i++){
      for(j=0;j<10;j++){
        any_stmt;
      }
    }
  }
#pragma xev stmt dst("label1");
  {
    for(int ij=0;ij<100;ij++){
      i=ij%10;
      j=ij/10;
      any_stmt;
    }
  }
```

もともとのループ構造

ループ内のすべての文がコピーされることを指示

変換後のループ構造

「コードパターン」として定義 → **個別事例に応じてカスタマイズすることも容易**

2.3節　ATMathCoreLib とは

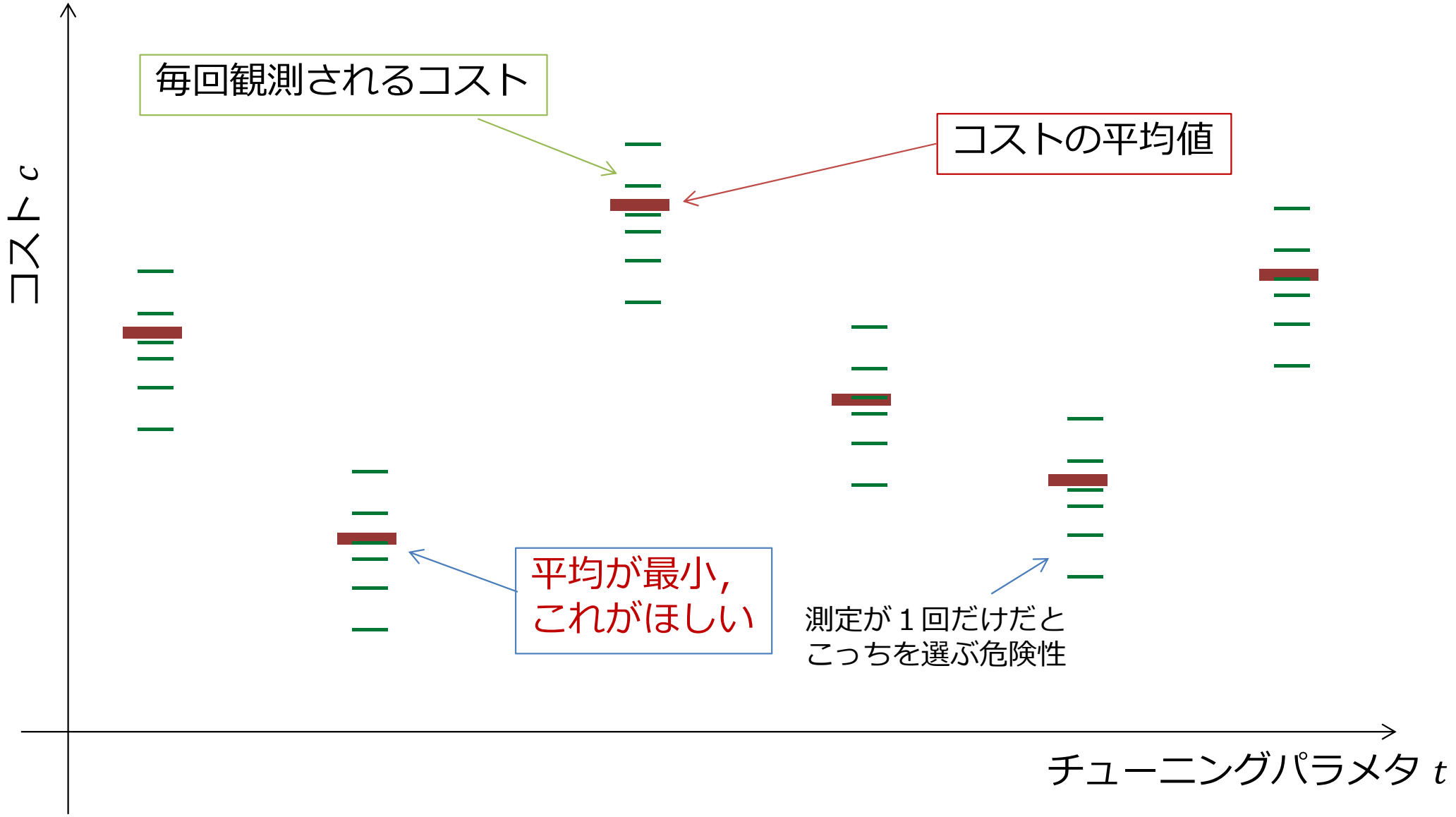
- 自動チューニングのための数理技術として開発してきたアルゴリズムを実装

- 一般公開，利用自由
- http://olab.is.s.u-tokyo.ac.jp/~reiji/atmathcorelib/

- Scilab バージョン，C バージョンがある
- C の 2012/12/25 版の使い方を説明します

須田礼仁 教授提供

ATMathCoreLib が解く問題



須田礼仁 教授提供

ATMathCoreLib の使い方

もとのユーザーコード

```
// func_param は動作を決める
// tune_param は性能を決める
// 0 <= tune_param < M とする
void function(func_param, tune_param) {
  主要な計算;
}

void main() {

  for (i = 0; i < K; i++) {
    function(fp_value, tp_value);
    // cost を測る手段 get_cost() がある
  }

}
```

改変後のユーザーコード

```
void function(func_param, tune_param) {
  主要な計算;
}

void main() {
  x = new_exdesign(M, 0.0);
  for (i = 0; i < K; i++) {
    // 残り実行回数 K – i – 1
    // 次に選ぶ選択肢 tp
    tp = exdes_nomodel_online(x, K-i-1);

    function(fp_value, tp);
    // コストは get_cost() で得られる
    update_exdes(x, tp, get_cost());
  }
}
```

3.1節 FFTE の概要

高橋大介 教授提供

- FFTEは高速フーリエ変換（Fast Fourier Transform, FFT）を計算するためのライブラリである.
- 2002年5月にver. 1.0を公開し，現在のバージョンは2020年8月に公開したver. 7.0である.
- FFTEはFortranで記述されており，逐次，共有メモリ（OpenMP），分散メモリ（MPI），GPU（CUDA），ハイブリッド並列（OpenMP+MPI，CUDA+MPI）を用いた以下のFFTをサポートしている.
  - 1～3次元の複素数FFT
  - 2～3次元の実数FFT
  - 1～3次元の並列複素数FFT
  - 2～3次元の並列実数FFT
  - 2次元分割を用いた並列複素数3次元FFT
  - 2次元分割を用いた並列実数3次元FFT
- http://www.ffte.jp/ よりソースコードを入手可能.

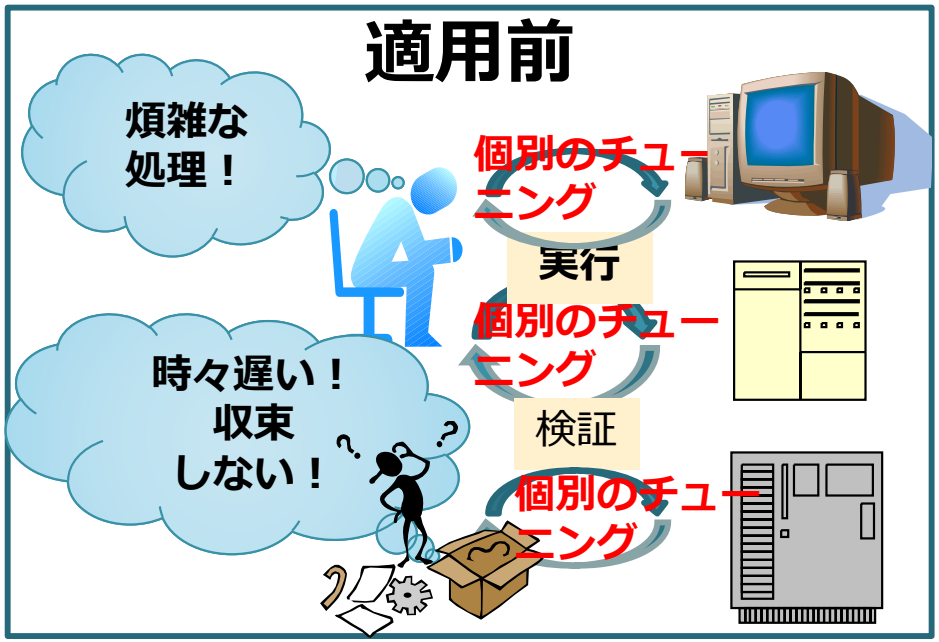
FFTEにおいて用いられている自動チューニング手法

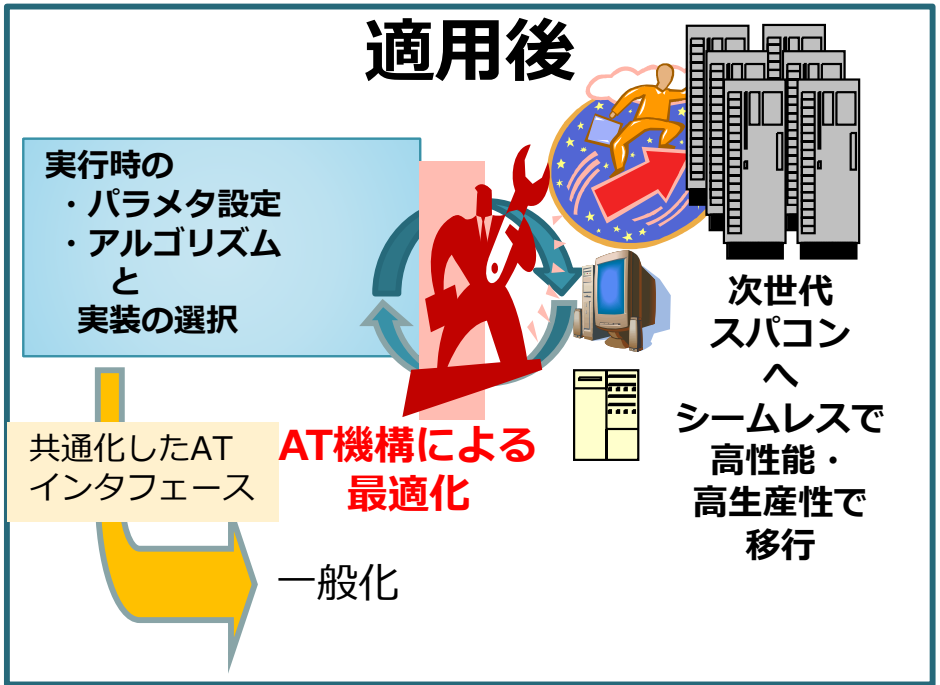
- FFTEにおいて自動チューニングを行っている性能パラメータは，以下の四つである.
  1. 基底（1次元FFTのみ）
  2. ブロックサイズ
  3. 全対全通信方式
  4. 演算と通信をオーバーラップする際の通信メッセージサイズの分割数
- 上記の1および2はMPIプロセス内の性能に関するパラメータであり，3および4はMPIプロセス間通信に関するパラメータである.
- したがって，1および2のチューニングと3および4のチューニングは，独立に行うことが可能である.

3.2節 疎行列反復解法ライブラリXabclib

e-Scienceプロジェクト「シームレス高生産・高性能プログラミング環境」、高性能ライブラリ（文部科学省）（2008FY から 2011FYまで）

- 問題
  - 職人技術に依存
  - 生産的でなく移植性もない
  - 煩雑でコストがかかる
  - パラメタの定義域外での指定、数値特性の悪化による収束失敗
- 目標
  - 高性能でかつ性能移植性のある数値計算ライブラリの提供
- 疎行列の非ゼロ成分構成を考慮し**以下の機能の実行時最適化を提供**
  1. **計算カーネル選択**
  2. **数値アルゴリズム選択**
  3. **並列実装選択**
  4. ***一般化したATの APIの提供 (OpenATLib)***





OpenATLibの設計方針

- 実行時に行う自動チューニング
  インターフェースを（疎行列を扱う）
  ATライブラリ開発者やユーザに提供
- OpenATLibでは、以下の機能を提供
  1. アルゴリズム選択処理
     - Krylov部分空間のリスタート周期選択
     - 直交化方式選択
  2. 実装方式選択
     - 疎行列-ベクトル積 (**SpMV**)
       - 負荷バランス改良、ベクトル計算機向け、並列化向き
  3. 計算機資源選択
     - 実行時のメモリ量・コア数などを考慮した
       SpMVとリスタート周期
- ユーザポリシ設定機能
  - 実行速度、メモリ量、演算精度

OpenATLibの命名規則

第１文字	● S：単精度 ● D：倍精度
第２文字と 第３文字	● 補助関数の場合：AF ● 計算関数の場合：第２文字に行列の種類、第３文字に行列格納方式 　● 第２文字 　　● S： 対称 　　● U：非対称 　　● D：対角 　　● T：三重対角 　● 第３文字 　　● R：CRS形式（※） 　　● C：CCS形式
第４文字と 第５文字	処理の種類 ● MV： 行列-ベクトル積 ● RT： リスタート周期

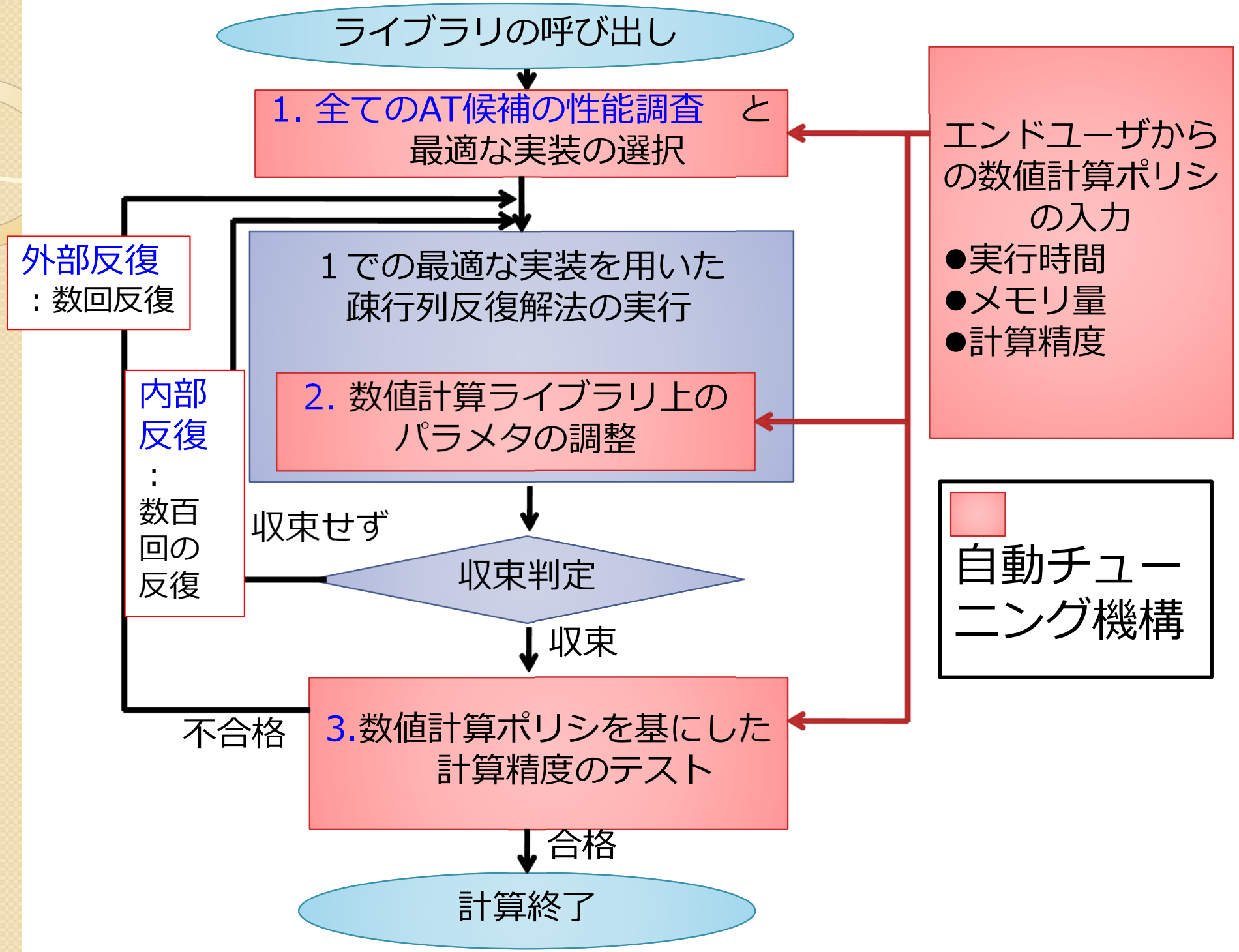
※現在、CRS形式のみサポート

OpenATLib の提供関数

	関数名	説明
1	OpenATI_INIT	OpenATLib と Xabclib　の初期パラメタを設定
2	OpenATI_DAFRT	Krylovサブスペースでのリスタート周期の増加を判断する
3	OpenATI_DSRMV	CRS形式での倍精度対称SpMVの最適な実装を判断
4	OpenATI_DURMV	CRS形式での倍精度非対称SpMVの最適な実装を判断
5	OpenATI_DSRMV_Setup	OpenATI_DSRMVのセットアップ関数
6	OpenATI_DURMV_Setup	OpenATI_DURMVのセットアップ関数
7	OpenATI_DAFGS	4種の実装によるGram-Schmidt再直交化
8	OpenATI_LINEARSOLVE	数値計算ポリシを適用した連立一次解法ソルバのメタインタフェース
9	OpenATI_EIGENSOLVE	数値計算ポリシを適用した固有値ソルバのメタインタフェース

注) OpenATLib2011で提供した新関数（スレッドセーフのため）

OpenATLibにおける実行時のAT機構



尾崎克久教授提供

第5章 精度保証付き数値計算

- 連立一次方程式の近似解と誤差上限（最大値ノルム）を出力する
- LINSYS_VR: Verified Solution of Linear Systems with Directed Rounding
  - 丸めのモードの変更を用いる精度保証
  - 精度保証に必要なメモリは行列4つ分（入力行列含む）
- LINSYS_V: Verified Solution of Linear Systems
  - 丸めのモードの変更を用いない精度保証
  - 精度保証に必要なメモリは行列3つ分（入力行列含む）
- http://www.math.twcu.ac.jp/ogita/post-k/

でコードを公開中

本発表の流れ

- 書籍の紹介
- AT研究会関係者が開発されたソフトウェア
- 今後の展望

名古屋大学
NAGOYA UNIVERSITY

自動チューニング研究の今後の展望

- 最新計算機環境へのたゆまぬ適用と評価
  - マルチコアCPUやGPU技術も、どんどん進化し、留まることがない
  - 人工知能、ビッグデータなどのSociety5.0への対応
  - ポストムーアを指向した、いくつかの動向と対応
    - 特に「**量子コンピュータ**」、もしくは「**量子コンピューティング**」への対応
- 「ソフトウェア工学」への昇華
  - 方法論やツール開発はソフトウェア生産性から重要
  - 数値計算ライブラリの開発は今後も重要
  - それらの知見を学術的にまとめて発展させる

iTC
名古屋大学
NAGOYA UNIVERSITY

性能意識工学
（パフォーマンス・アウェア・エンジニアリング、ＰＡＥ）

性能意識ソフトウエア建築論
（パフォーマンス・アウェア・ソフトウエア・アーキテクチャ、**ＰＡＳＡ**）

- ソフトウエア自動チューニング
- ＡＴソフトウエア構成論
- ＡＴモデル化と最適化理論

構築論へ昇華・最適化理論

方法論・開発効率・ツール開発

性能意識ソフトウエア工学
（パフォーマンス・アウェア・ソフトウエア・エンジニアリング、**ＰＡＳＥ**）

- 自動チューニング ソフトウエア工学
- ＡＴソフトウエア方法論
- ＡＴツール

適用評価

現実問題モデルの提示

性能意識計算
（パフォーマンス・アウェア・コンピューティング、**ＰＡＣ**）

- ＡＴ付き数値計算ライブラリ
- ＡＴ付き数値シミュレーション

理論の実践

現実問題の提示