TOHOKU UNIVERSITY

研究課題

進化的アプローチによる超並列複合システム向け開発環境の創出

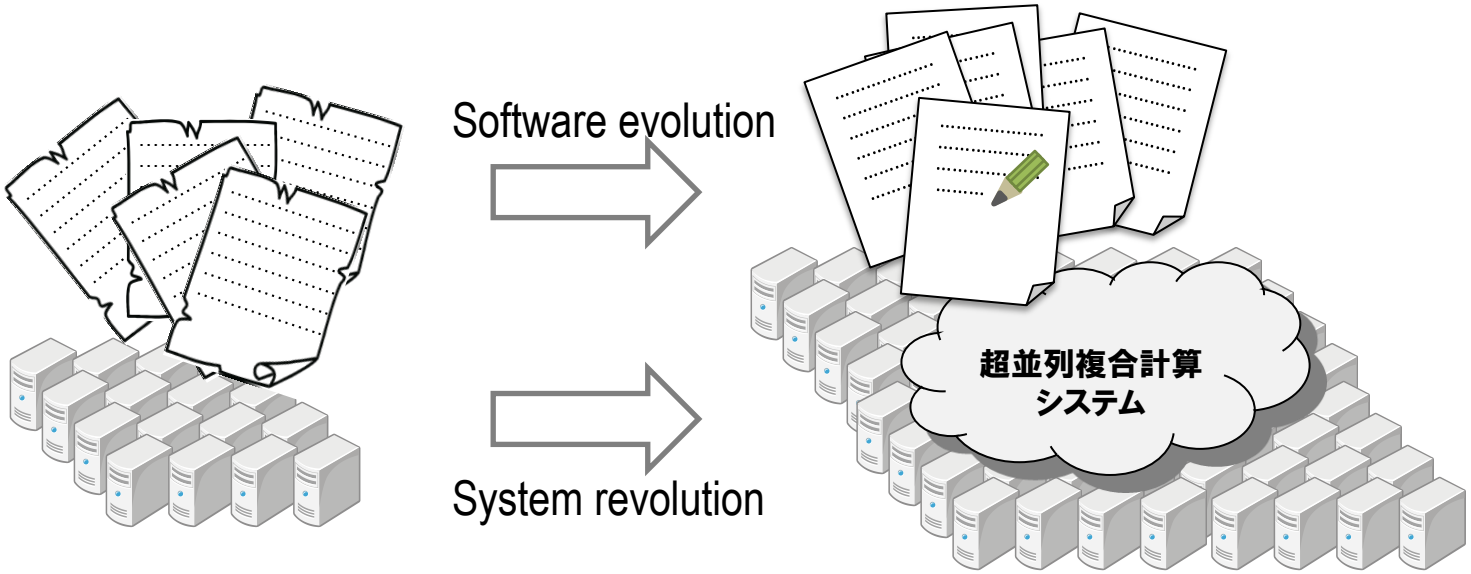
2013年12月25日 ATTA2013

研究代表 滝沢 寛之
東北大学大学院情報科学研究科

主たる共同研究者
須田 礼仁 (東京大学)
高橋 大介 (筑波大学)
江川 隆輔 (東北大学)

背景と目的

- 背景：HPCシステムアーキテクチャの劇的変化
  - 大規模化、複合化（ヘテロ化）、多様化、・・・
- 目的：超並列複合計算システム向け開発環境
  - システムの進化に対応可能なアプリ開発方法の確立
  - 既存アプリケーションの**進化**を支援



研究体制

チーム代表：滝沢寛之（東北大）

プログラミング	数値計算ライブラリ	ドメイン特化ツール	アプリ設計・開発支援
滝沢寛之(東北大) 伊野文彦(大阪大)	高橋大介(筑波大) 藤井昭宏(工学院大)	須田礼仁(東京大) 吉本芳英(鳥取大) 玉田嘉紀(東京大)	江川隆輔(東北大) 高橋桂子(JAMSTEC) 松岡 浩(東北大) Sabine Roller (GRS) 中田登志之(NEC) 松岡浩司 (NEC)

アドバイザリ委員：Wen-mei W. Hwu (UIUC)，Michael M. Resch (HLRS)，加藤千幸 (東大生研)

- **目的：超並列複合計算システム向け開発環境**
  - 性能可搬性の改善・維持を支援する方法論やツール
    - ツールによる支援方法を検討(**滝沢G**)
    - 既存アプリの分析と再設計の指針を検討(**江川G**)
  - 実装を抽象化して提供する仕組み
    - プログラミングインタフェースの観点から検討(**滝沢G**)
    - 数値演算ライブラリの観点から検討(**高橋G**)
    - ドメイン特化の観点から検討(**須田G**)

本研究の特色

- **既存コードを始点とした新世代システム向けアプリ開発**
  - 多くの関連研究では新規開発の性能・生産性に着目
  - 既存コードの移植には段階的な移行手段が重要
    - 新しいモデル・コンセプトへの移行には飛躍が必要
    - 継続的に動作確認しながら移行できる手段の需要

  → 既存コードから段階的に利用できる方法という制約の下で効果的なアプリ開発の形を模索
    - 性能可搬性向上によって将来の進化を支援

- **課題：性能と可搬性の両立**
  - **性能**　：システムごとに異なる性能最適化を適用する必要
  - **可搬性**　：特定のシステム向けの性能最適化を回避する必要

HPCプログラミング

- 専門の異なるプログラマたちのチームワーク

アプリ開発者（計算科学者）

- 表現したいもの：アルゴリズムとプログラムの関係

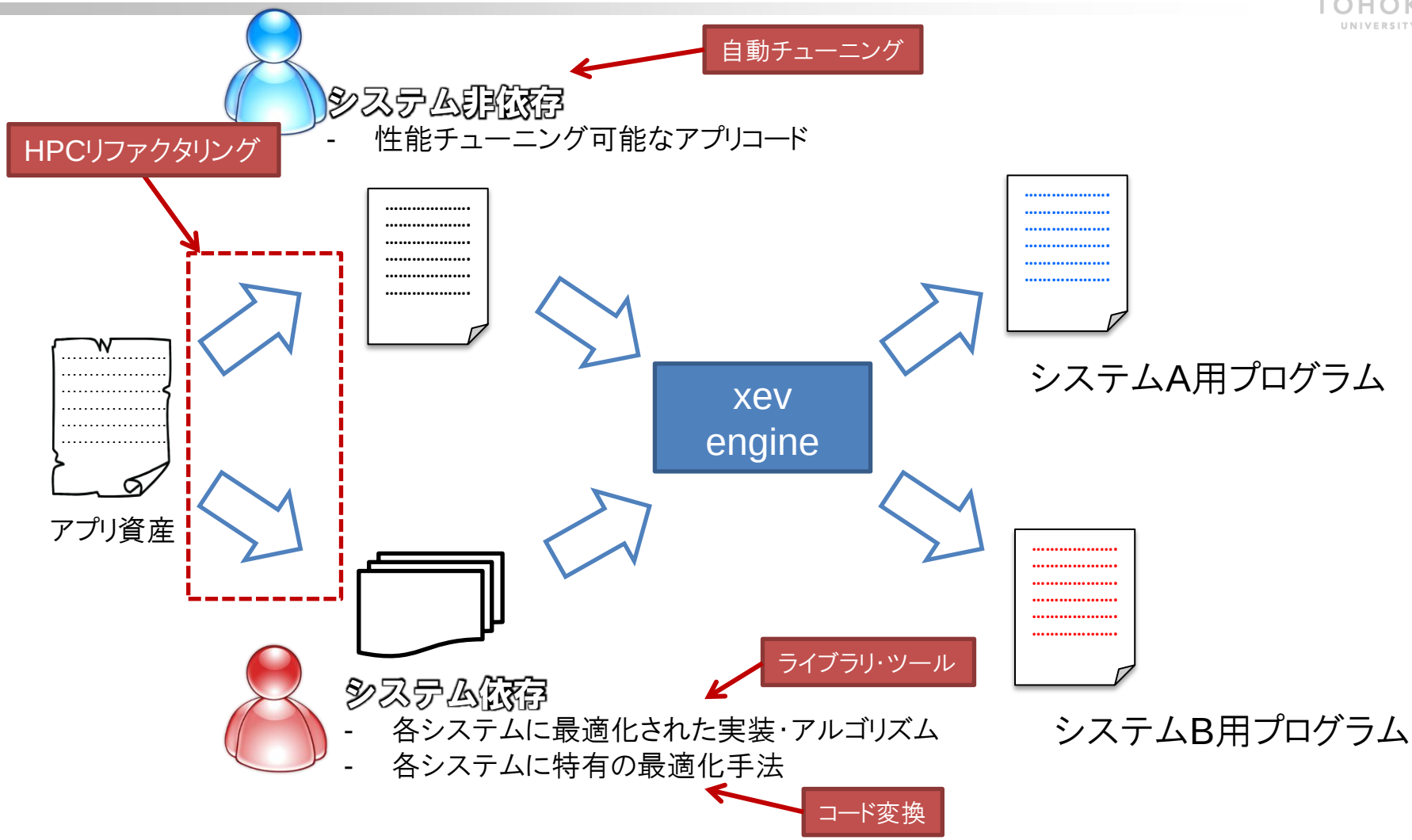
→ 正しい計算結果を出力するプログラムの作成

チューナー　（計算機科学者）

- 表現したいもの：プログラムとシステムの関係

→ 対象システムで高速動作するプログラムの作成

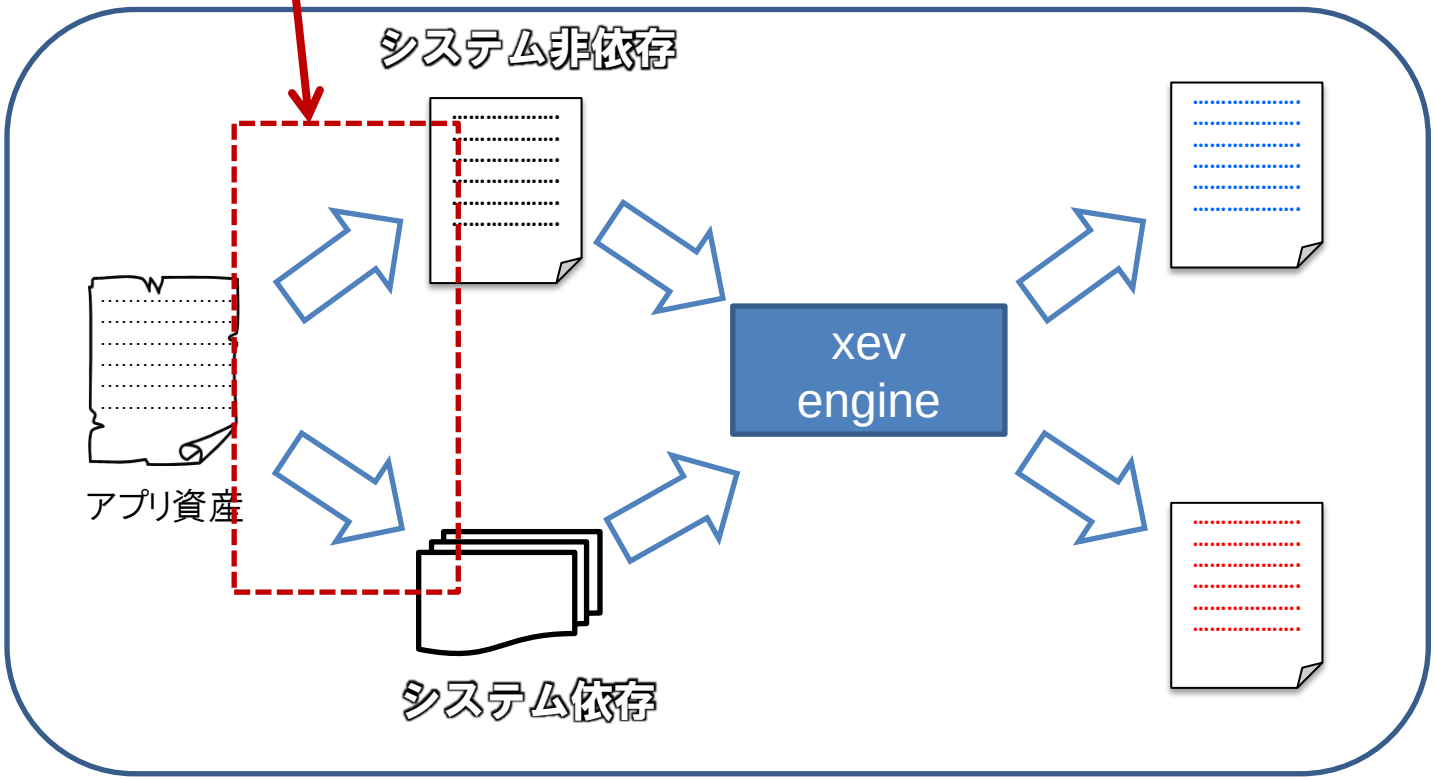
性能可搬性向上のアプローチ



システム依存性をアプリケーションから分離
→ システムごとに異なる組み合わせで高性能を実現

進捗報告

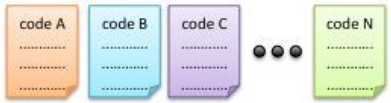
- HPCリファクタリング
- Xevolverフレームワーク
- ポストペタ向け実装・アルゴリズム

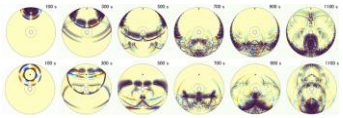


HPCリファクタリングカタログ構築

HPCコード

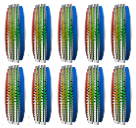
・既存最適化事例の収集



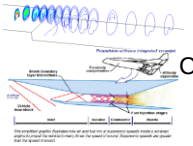


a seismic wave Propagation
X 109

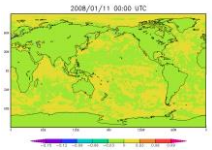
・新規コード最適化・並列化



Turbine
x11.91



Scramjet
Combustion
x20.6



Global Barotropic
Ocean Modeling
x8.31

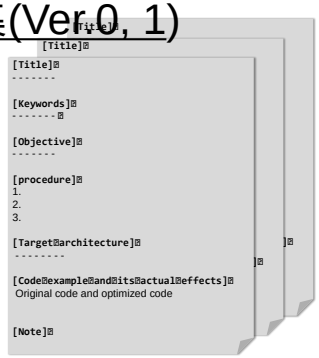
ターゲットアプリMSSGの整備

解析・体系化

最適化手法の開発

最適化・並列化事例集(Ver.0, 1)

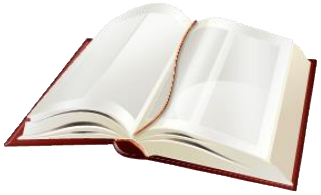
- コード特徴
- 目的
- 計算機環境
- 高速化手順
- 効果
- 実行可能な最適化前後のカーネル



解析

汎化・事例として追記

性能可搬性維持するためのガイドライン



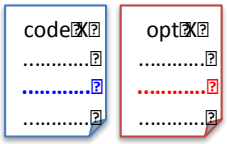
Refactoring Catalog

性能可搬性維持要件

性能可搬性の検討

・性能可搬性維持要件の調査

- マルチプラットホームにおける評価

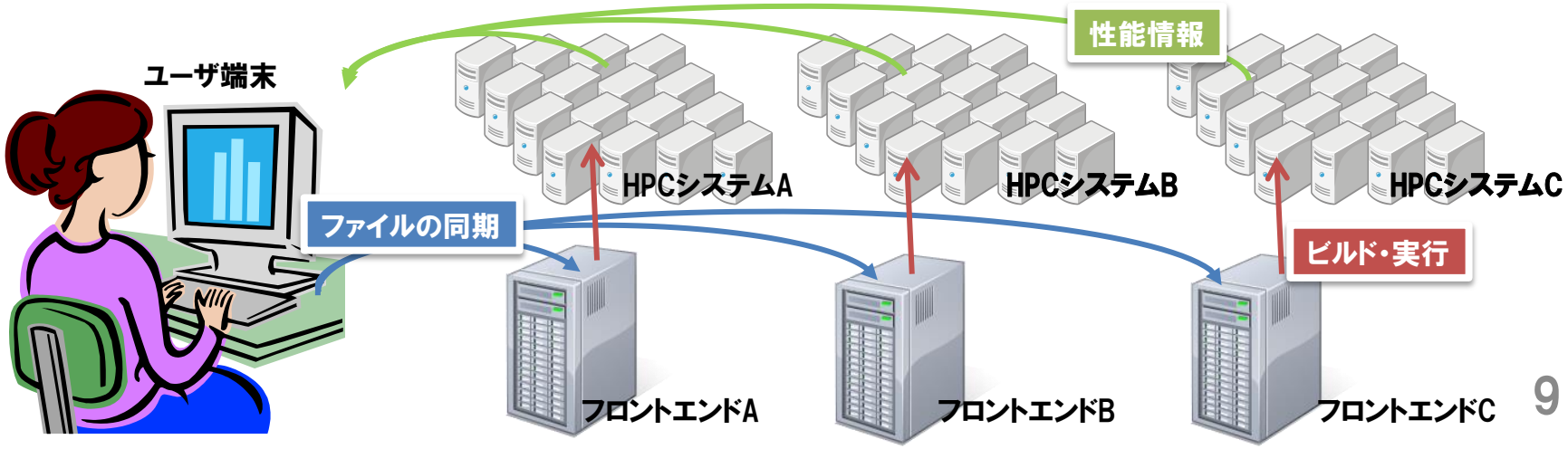


スカラ機の事例を検討追加
現在合計22件

リファクタリング支援ツール

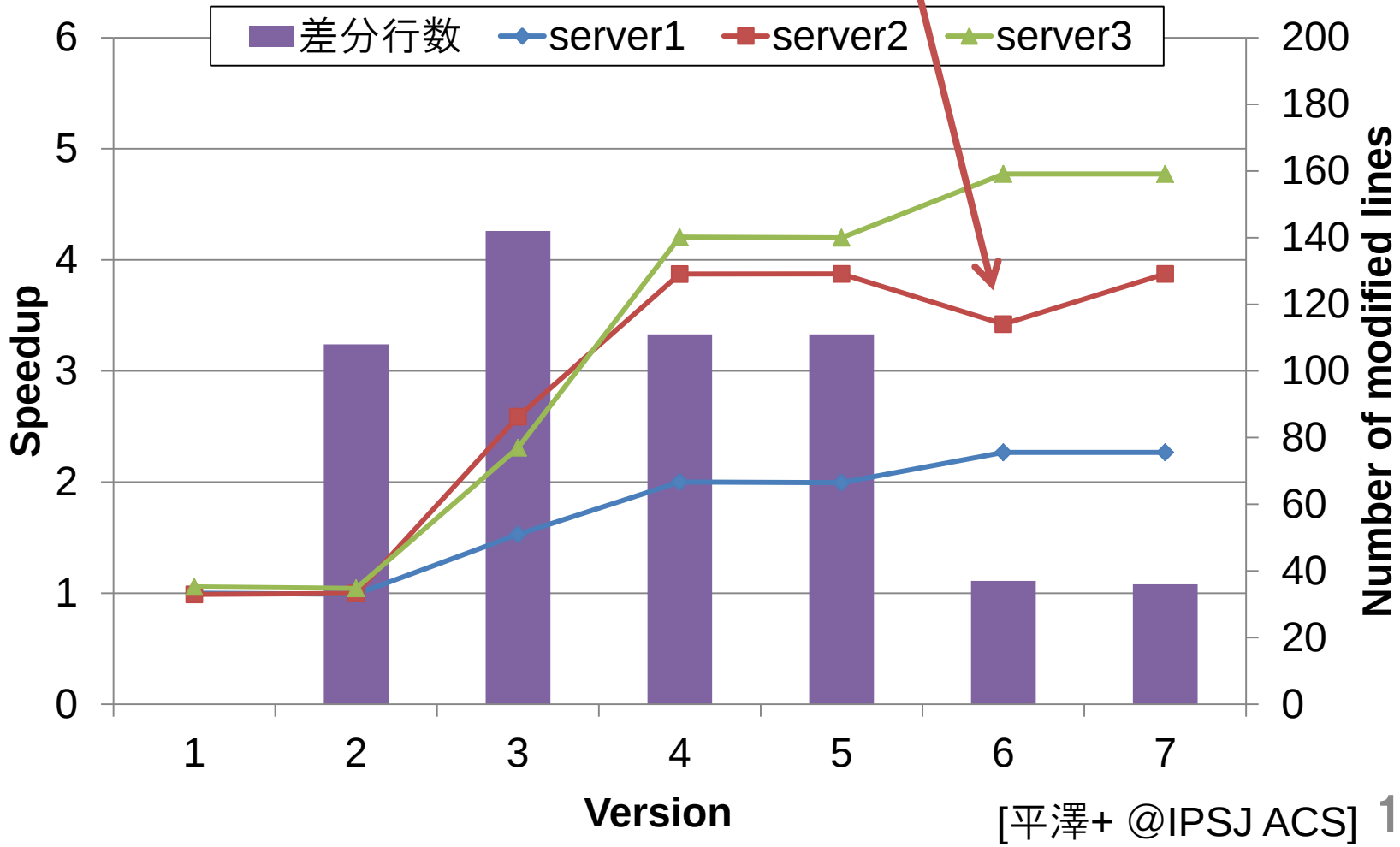
- 支援システムの機能
  - 指定されたすべてのサイトでソースコードを同期
    - ソースコード、ビルドファイル(Makefile)、その他リソースファイル
  - ビルド処理の遠隔実行機能
  - アプリケーションの自動実行機能
    - 性能の記録機能 ＝ コードと性能・実行情報の対応付け
    - バージョンを追跡して性能を記録する機能

→ 性能可搬性低下の要因検出・修正の提案と支援



コード変遷と性能変化の対応付け

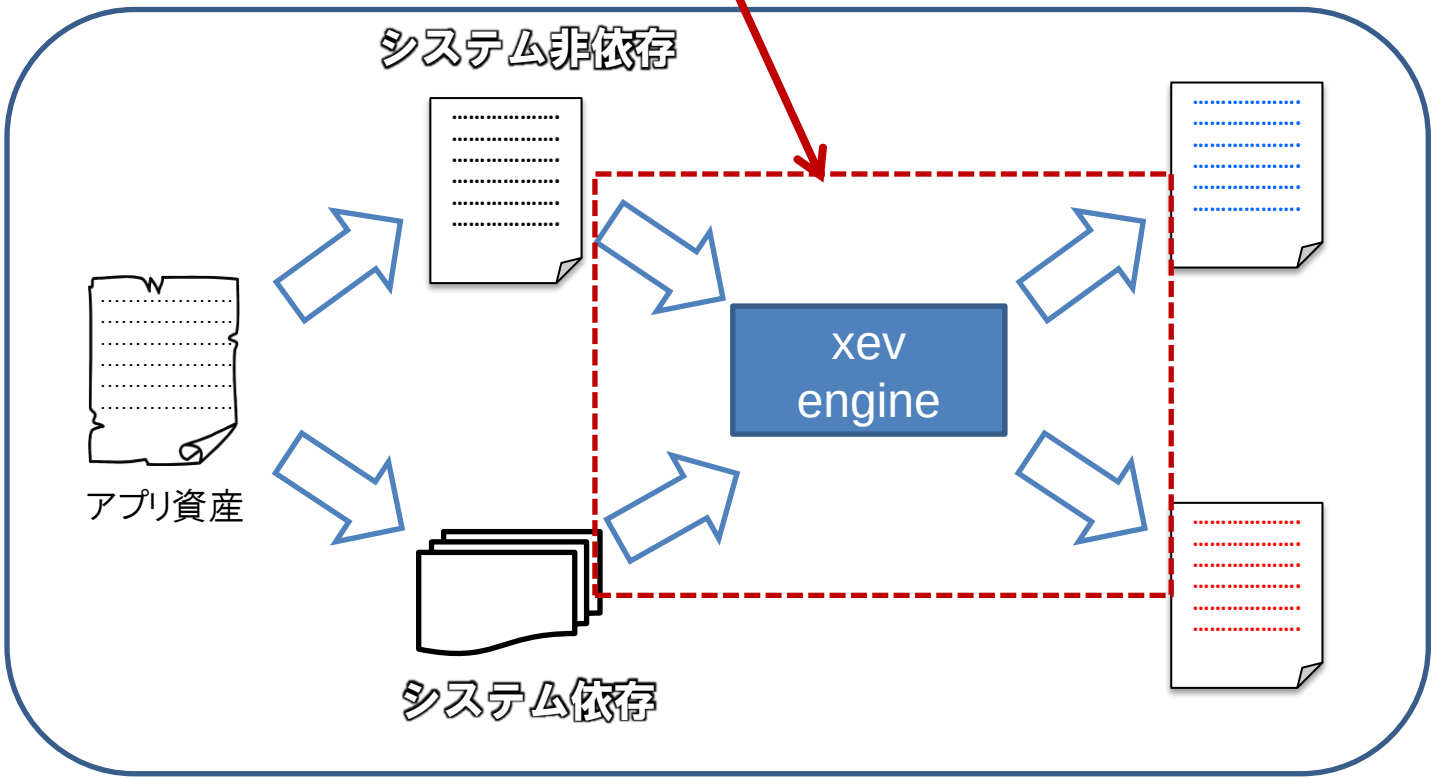
バージョン6でサーバ2に対する性能が低下
→ プログラマに性能低下を通知してコードの差分を提示



[平澤+ @IPSJ ACS]

進捗報告

- HPCリファクタリング
- Xevolverフレームワーク
- ポストペタ向け実装・アルゴリズム



コード最適化の抽象化

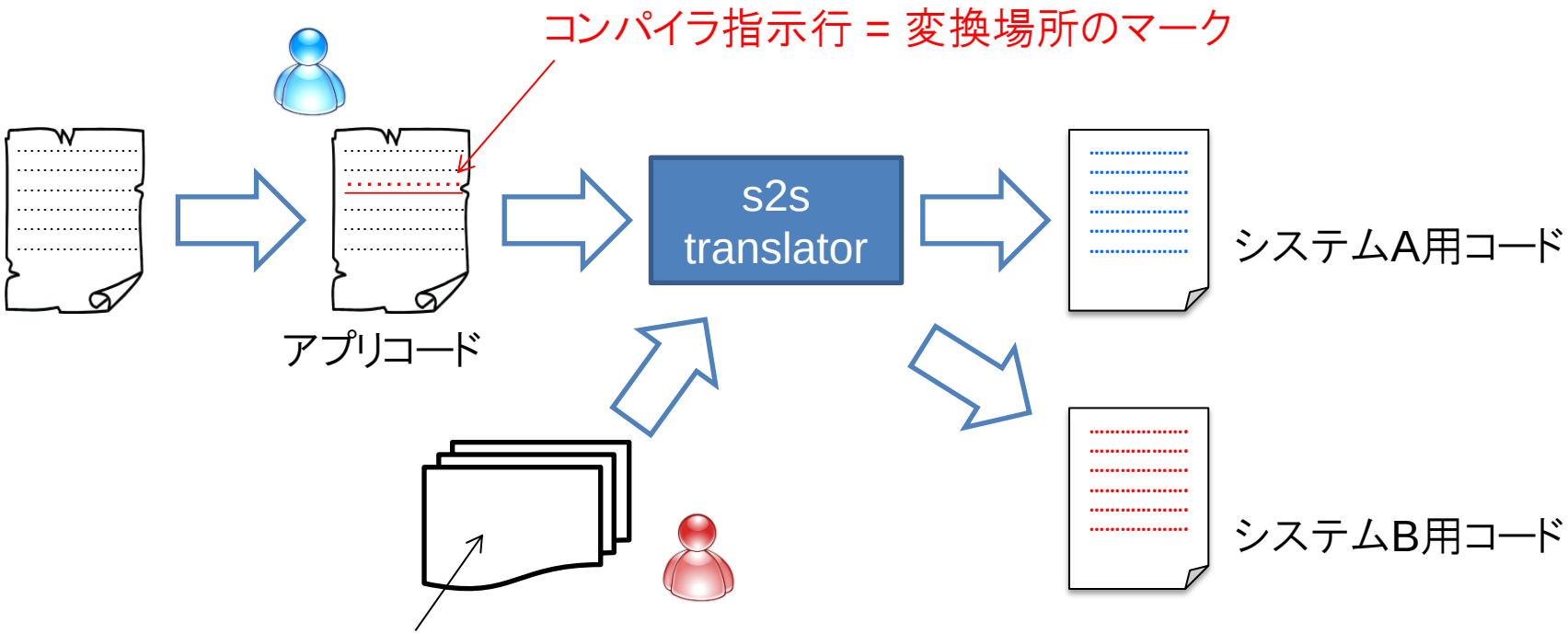
- **ディレクティブや変換スクリプト**
  - アプリコードを編集する代わりにディレクティブを追記
  - アプリコードとは別に変換スクリプトを記述
- **主にループ最適化の研究事例**
  - CHiLL (Univ. Utah)
  - POET (Georgia Tech)
  - LoopTool (Rice Univ)
  - Orio (OSU)
  - ROSE (CHiLL, POET, and ROSE)
  - ABCLibScript (Univ Tokyo)

名前がついているような基本的なループ最適化はすでに提供済み

提供されていない変換には対応困難
記述自体が特定のシステムに特化する傾向

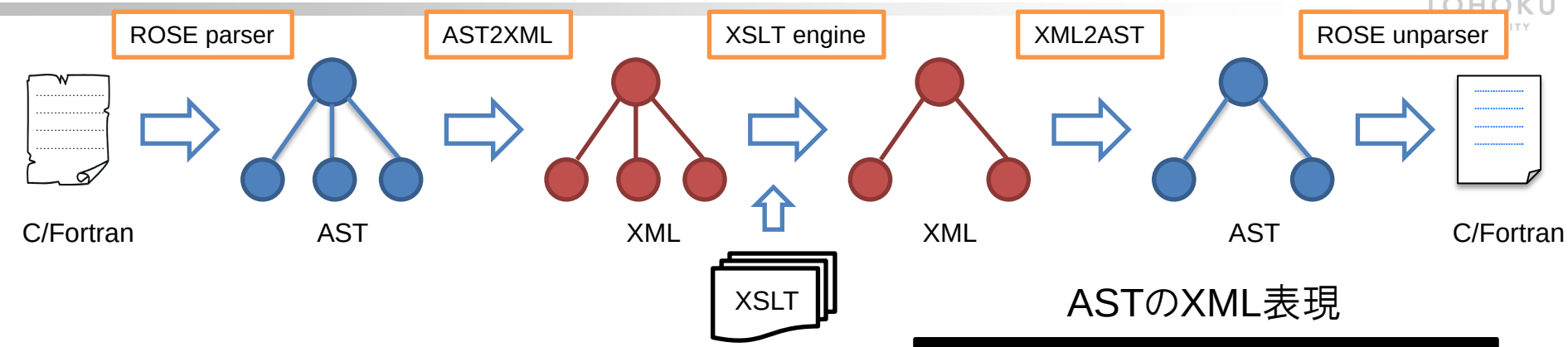
ユーザ定義コード変換の必要性

多様なシステムに適応するためには様々なコード変換が必要
→ 既存の変換の組み合わせだけでは表現不可



- コンパイラ指示行の実際の挙動を定義
- システムごとに異なるレシピを利用可能
- アプリケーション開発者が独自な指示行を定義可能

Xevolver フレームワーク



C/Fortranのプログラム

```
for(a=0;a<N;a++)
  {c=c+3;}
```

相互変換

ASTのXML表現

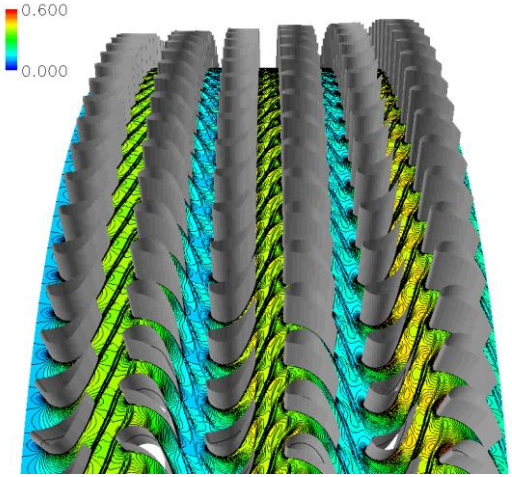
```
<SgForStatement address="0x7fc839c38010">
  <SgForInitStatement address="0x22b36a0">
    <SgExprStatement address="0x2301ab0">
      <SgAssignOp address="0x22e6520">
        <SgVarRefExp address="0x22ccdd0" name="a"/>
        <SgIntVal address="0x224b0a8" value="0" />
      </SgAssignOp>
    </SgExprStatement>
  </SgForInitStatement>
  <SgExprStatement address="0x2301b08">
    <SgLessThanOp address="0x2317370">
      <SgVarRefExp address="0x22cce38" name="a"/>
      <SgIntVal address="0x224b110" value="100" />
    </SgLessThanOp>
  </SgExprStatement>
  <SgPlusPlusOp address="0x2332900">
    <SgVarRefExp address="0x22ccea0" name="a"/>
  </SgPlusPlusOp>
  <SgBasicBlock address="0x7fc839d06120">
    <SgExprStatement address="0x2301b60">
      <SgAssignOp address="0x22e6590">
        <SgVarRefExp address="0x22ccf08" name="c"/>
        <SgAddOp address="0x234c070">
          <SgVarRefExp address="0x22ccf70" name="c"/>
          <SgIntVal address="0x224b178" value="3" />
        </SgAddOp>
      </SgAssignOp>
```

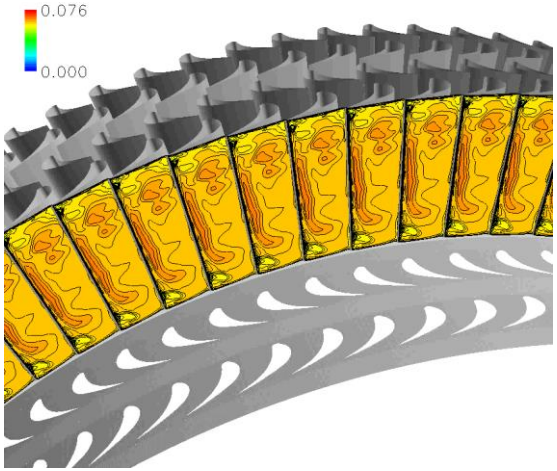
変換レシピから直接ASTを変更できるので
構文に基づく再利用性の高い変換規則を定義可能

XMLを介したプログラマとROSEの連携

評価アプリケーション

- 数値タービン
  - タービン多段流路内の非定常3次元流れをシミュレート
    - 東北大学大学院情報科学研究科・山本研究室
  - NEC SX-9@東北大学で2013年現在利用
    - SX-9向けにさまざまな最適化
    - 似たような形のループが多数出現 ＝ 同じような変換を繰り返す必要





東北大山本研究室 http://www.caero.mech.tohoku.ac.jp/research/NumericalTurbine/NumericalTurbine.html

ループ変更 + OpenACCディレクティブ

TOHOKU UNIVERSITY

SX version

```
DO 200 M=1,MF
  DO 200 K=1,KF
    DO 200 J=1,JF
      DO 200 L=lstart,lend
        II1 = IS(L)
        II2 = II1+1
        II3 = II2+1
        IIF = IT(L)
        IIE = IIF-1
        IID = IIE-1
        DO 200 I=II2,IIF
          IF (I.LE.II3.OR.I.GE.IIE)THEN
            STBC=0.0D0
          ELSE
            STBC=1.0D0
          END IF
            ⋮
```

OpenACC version

```
!$acc loop private(L)
DO 200 M=1,MF
!$acc loop gang
  DO 200 K=1,KF
!$acc loop gang,vector
    DO 200 J=1,JF
!$acc loop vector
      DO 200 I=1,inum
!$acc loop seq
        DO L=lstart,lend
          IF (I.ge.IS(L) .and. I.le.IT(L)) EXIT
        END DO
        IF (i.ne.IS(L)) THEN
           IF (I.LE.(IS(L)+2).OR.I.GE.(IT(L)-1)) THEN
             STBC=0.0D0
           ELSE
             STBC=1.0D0
           END IF
             ⋮
```

!$xev interchange

Before + !$xev

```
SUBROUTINE SAMPLE02

!$xev interchange depth1
    DO I = 1, 10
      DO J = 1, 25
        SUM = SUM + 3
      END DO
    END DO
```

After

```
SUBROUTINE SAMPLE02()
    DO J = 1, 25
    DO I = 1, 10
    SUM = SUM + 3
    END DO
    END DO
    RETURN
    END SUBROUTINE
```

XML before

```
….
<SgFortranDo address="0x9c5b1d0" style="0"  end="1"  nlabel=""  slabel="" >
  <SgAssignOp address="0x9c43ac0">
    <SgVarRefExp address="0x9c35ff8" name="I"/>
    <SgIntVal address="0x9c1c9a8" value="1" />
  </SgAssignOp>
  <SgIntVal address="0x9c1c9dc" value="10" />
  <SgNullExpression address="0x9c51588"/>
  <SgBasicBlock address="0x9ba3888">
    <SgFortranDo address="0x9c5b264" style="0"  end="1"  nlabel=""  slabel="" >
      <SgAssignOp address="0x9c43af8">
        <SgVarRefExp address="0x9c36030" name="J"/>
        <SgIntVal address="0x9c1ca10" value="1" />
      </SgAssignOp>
      <SgIntVal address="0x9c1ca44" value="25" />
      <SgNullExpression address="0x9c515b0"/>
      <SgBasicBlock address="0x9ba3910">
        <SgExprStatement address="0x9c98a48">
          <SgAssignOp address="0x9c43b30">
            <SgVarRefExp address="0x9c36068" name="SUM"/>
            <SgAddOp address="0x9c8af80">
              <SgVarRefExp address="0x9c360a0" name="SUM"/>
              <SgIntVal address="0x9c1ca78" value="3" />
            </SgAddOp>
          </SgAssignOp>
        </SgExprStatement>
      </SgBasicBlock>
    </SgFortranDo>
….
```

XSLT

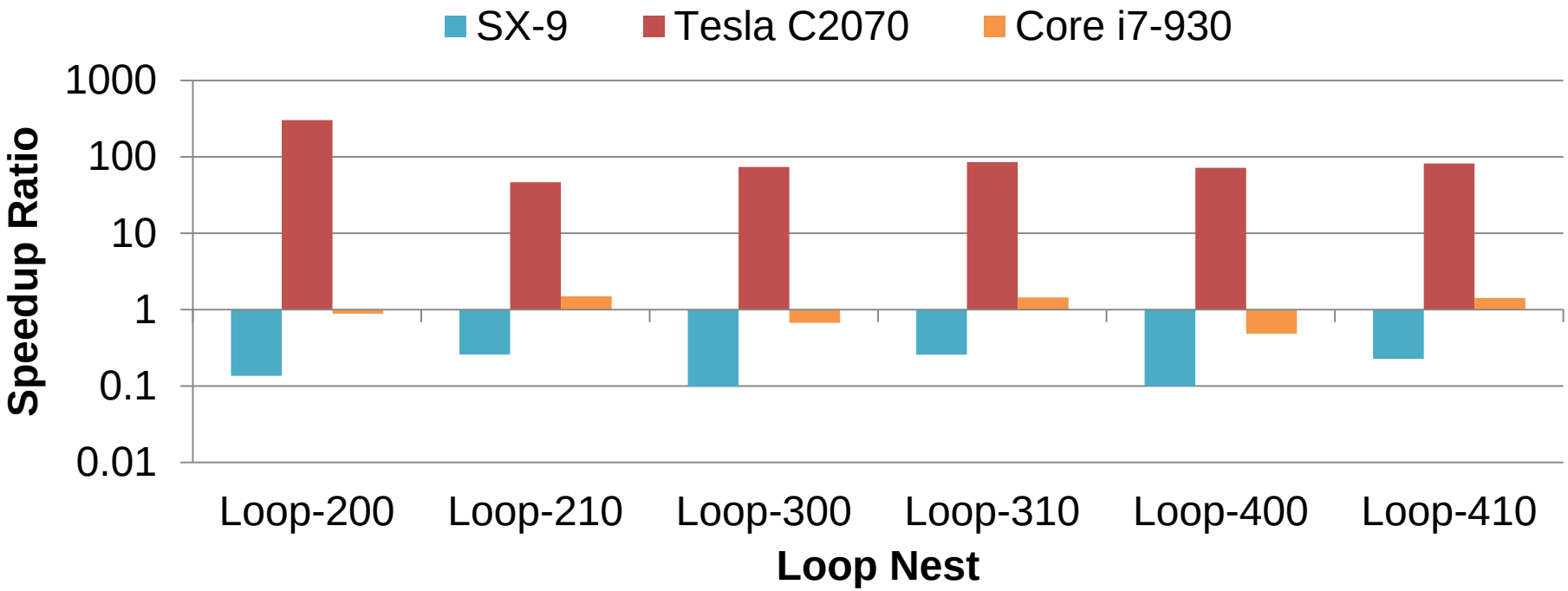
XML after(略)

XSLT for !$xev interchange

```
<xsl:template match="SgFortranDo">
    <xsl:variable name="doNum" select="count(./SgBasicBlock/SgFortranDo)" />
    <xsl:if test="count(child::PreprocessingInfo)=1"> // ディレクティブ検知
        <xsl:element name="SgFortranDo"> // 外側ループに、内側ループをコピー
            <xsl:copy-of select="SgBasicBlock/SgFortranDo/@*" />
            <xsl:copy-of select="SgBasicBlock/SgFortranDo/SgAssignOp" />
            <xsl:copy-of select="SgBasicBlock/SgFortranDo/SgIntVal" />
            <xsl:copy-of select="SgBasicBlock/SgFortranDo/SgNullExpression" />
            <xsl:element name="SgBasicBlock">
                <xsl:copy-of select="SgBasicBlock/@*" />
                <xsl:copy> // 内側ループに、外側ループをコピー
                    <xsl:copy-of select="@*" />
                    <xsl:copy-of select="./SgAssignOp" />
                    <xsl:copy-of select="./SgIntVal" />
                    <xsl:copy-of select="./SgNullExpression" />
                    <xsl:copy-of select="SgBasicBlock/SgFortranDo/SgBasicBlock" />
                </xsl:copy>
            </xsl:element>
        </xsl:element>
    </xsl:if>
</xsl:template>
```

性能評価結果

- SXとGPUでは異なる最適化が必要
  - SX向け数値タービンをOpenACC化した結果



変換レシピの切り替え → 各システムに適したバージョンを生成可能

カスタム指示行の定義

```
<DIRECTIVE name="interchange">
  <CLAUSE name="loop">
    <PLIST>
      <LI value="1"/>
      <LI value="2"/>
    </PLIST>
  </CLAUSE>
</DIRECTIVE>
```

独自のコンパイラディレクティブの定義

- ディレクティブ名(directive name)
- 宣言子(clause)
- パラメータのデフォルト値

独自のコンパイラディレクティブの利用

Xevolverでは指示行もXMLへ変換
特定の変換規則(XSLT)と対応

```
#pragma xev interchange loop(1,3)
for(i=0;i<100;i++){
  for(j=0;j<100;j++){
    for(k=0;k<100;k++){
      ...
    }
  }
}
```

特定のコードパターンの検出

XevolverではXMLでATを表現 → 様々なXML関連技術を利用可能

- XPath: the XML Path Language (XPath2.0, W3C 2007)
  - **XMLドキュメントから特定パターンのノードを選択するためのクエリ言語**
  - **XSLT でもXMLドキュメントの部分木を選択するために使用**

XML document

```
<A>
  <B>
    <C/>
  </B>
</A>
```

XPath form

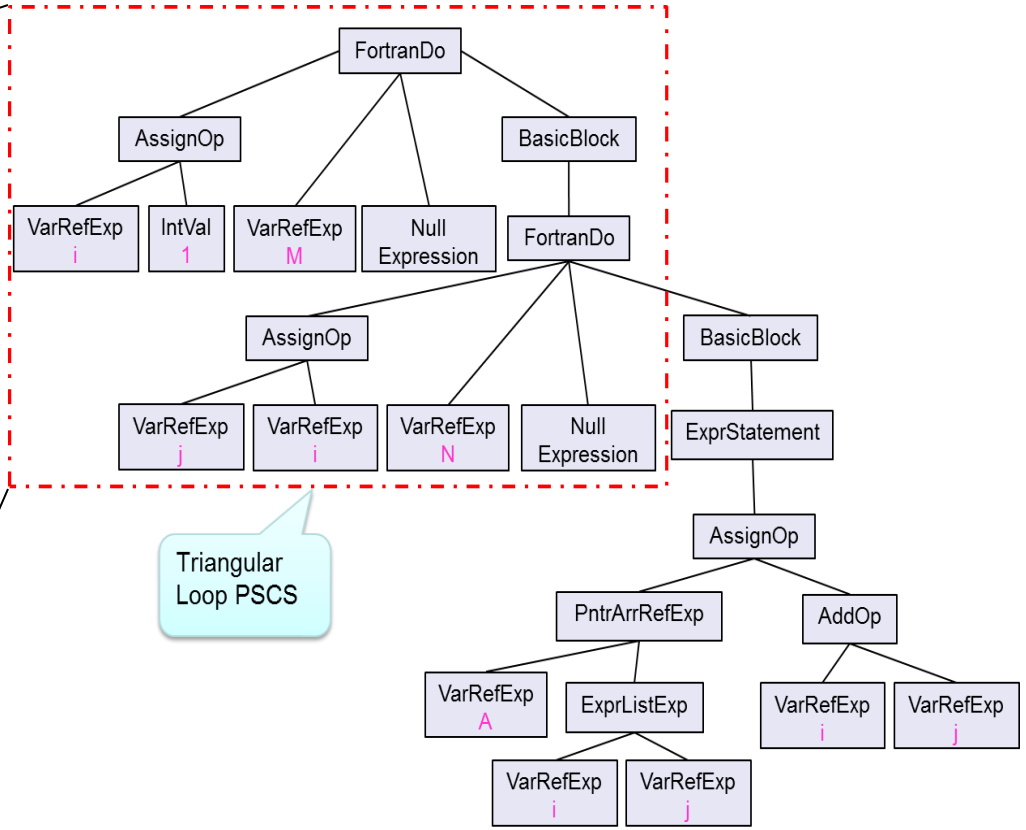
```
/A/B/C
```

Aの孫ノードかつBの子ノードとして定義されているCを選択

Triangular Loopパターン

```
!$acc region
do i=1,M
 do j=i,N
   A(i,j)=i+j
 end do
end do
!$acc end region
```

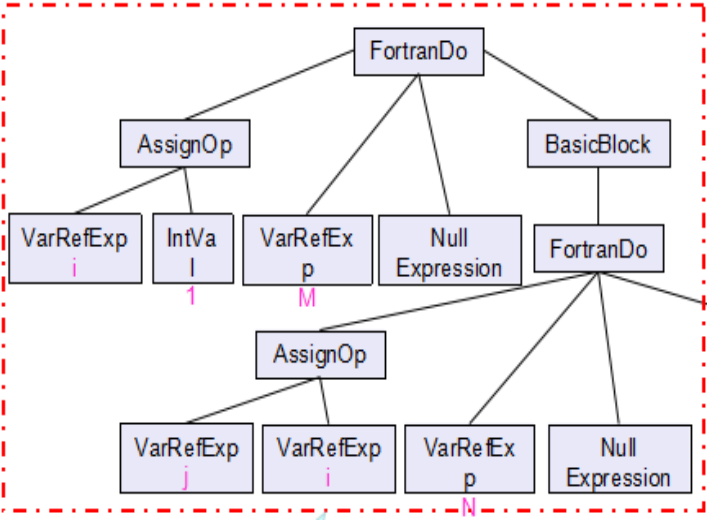
Triangular Loop



OpenACCコンパイラではベクトル化できないループのパターン

パターンマッチング

```
do i=1,M
 do j=i,N
   A(i,j)=i+j
 end do
end do
```



Use Xpath 2.0 expressions to do tree pattern matching

1. 外側のループのインデックス変数名

//FortranDo/AssignOp/VarRefExp[1]/@name

2. 内側のループの下限値

//FortranDo/BasicBlock/FortranDo/AssignOp/VarRefExp[2]/@name

3. 両者が同じならばTriangular Loop

//FortranDo/AssignOp/VarRefExp[1]/@name
=//FortranDo/BasicBlock/FortranDo/AssignOp/VarRefExp[2]/@name

膨大なソースコードの中から
特定のコードパターンを見つけるために有用

```
//FortranDo/AssignOp/VarRefExp[1]/@name=//FortranDo/BasicBlock/FortranDo/AssignOp/VarRefExp[2]/@name
```

Result
true

ROSE Autotuningとの連携

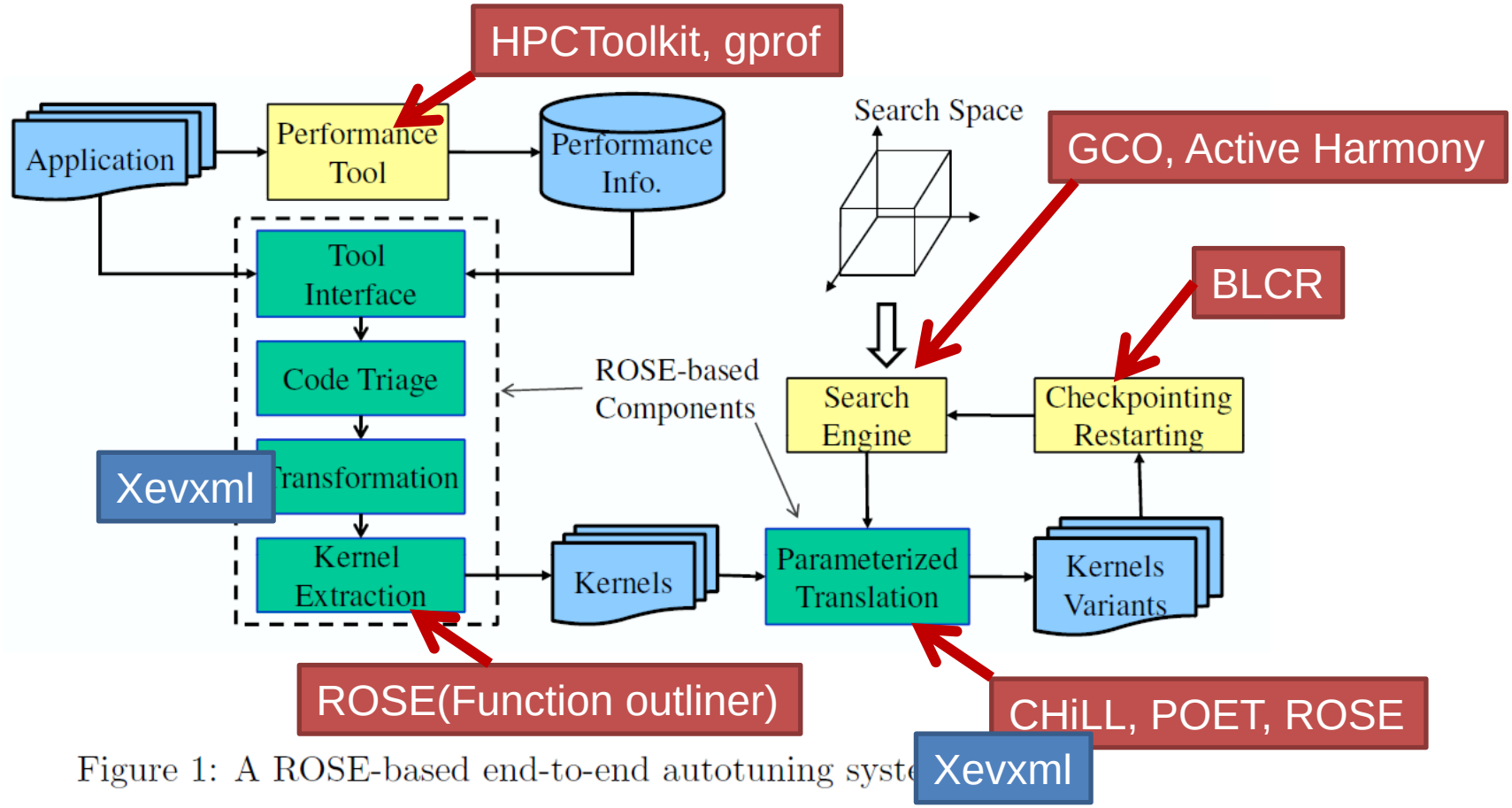


Figure 1: A ROSE-based end-to-end autotuning syste

XMLデータを介した連携

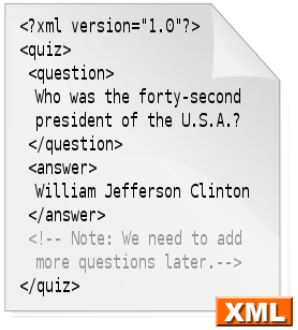
Empirical Tuning System

自動チューニング

HPCToolkit

性能情報

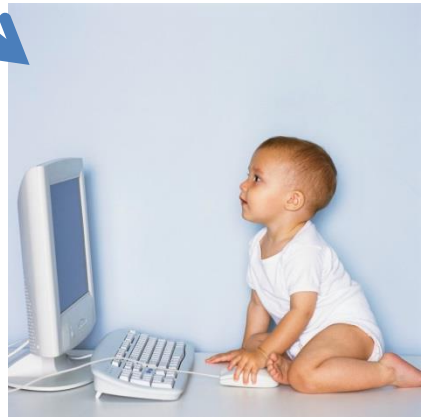
高度な変換
解析結果データの挿入



相互運用の実現

可視化
リファクタリング
(=対話的処理)

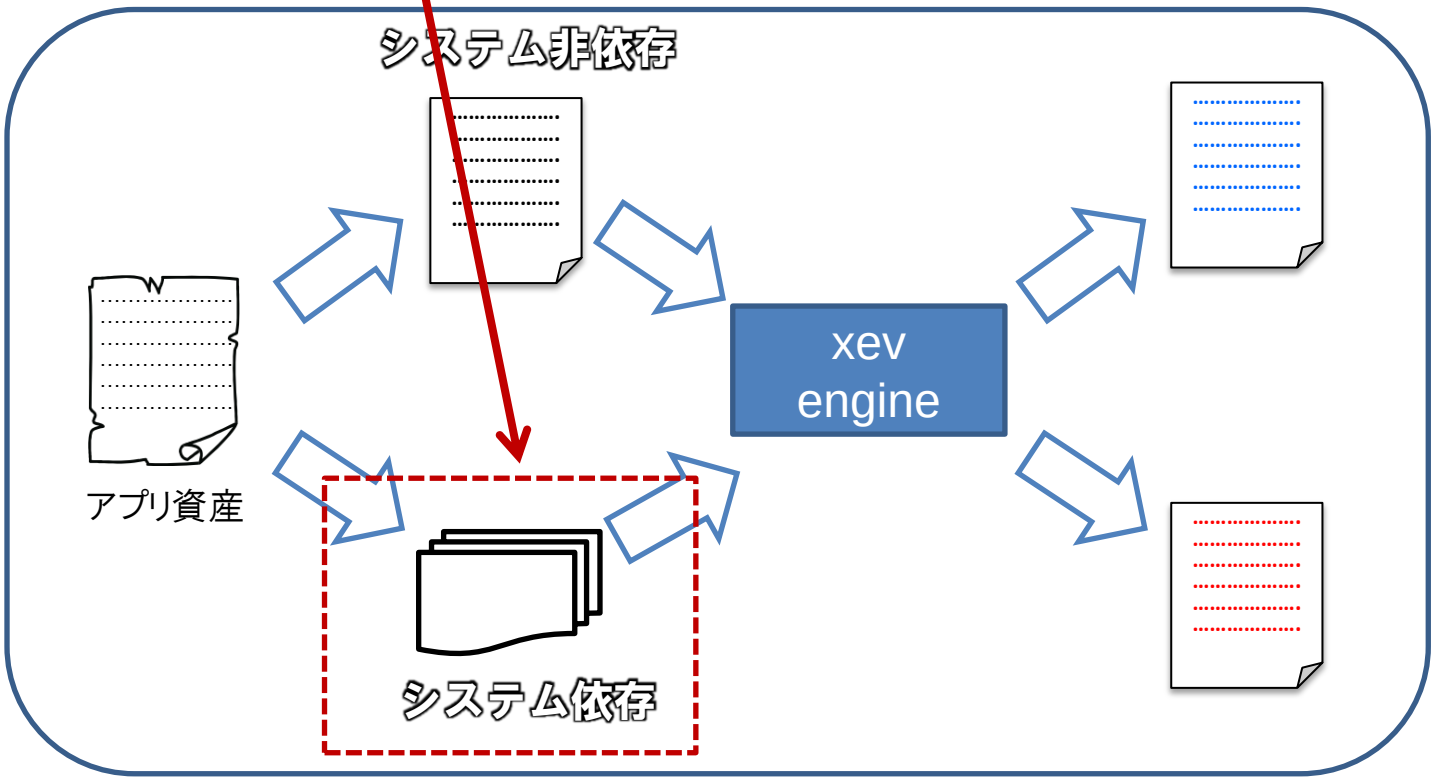
ユーザー定義の変換



他のコンパイラとの連携

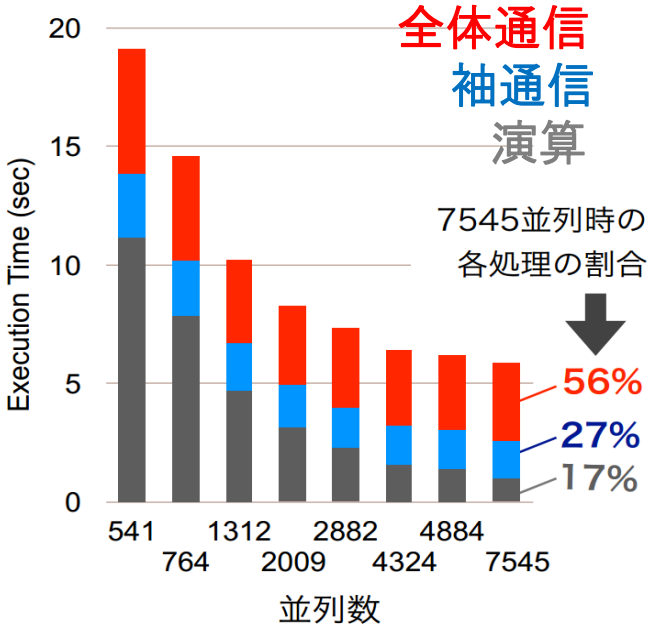
進捗報告

- HPCリファクタリング
- Xevolverフレームワーク
- ポストペタ向け実装・アルゴリズム



通信削減クリロフ部分空間法

- クリロフ部分空間法　solve $Ax = b$
- 並列数が増えると通信律速



[Nakao, Sato 2013]

- 既存研究 s-step [Hoemann 2012]
  - 全体通信回数$1/s$, 代わりに演算増
  - sが大きいと丸め誤差により不安定
    （=収束の悪化）
  - 通信回数と、演算量&安定性のトレードオフ

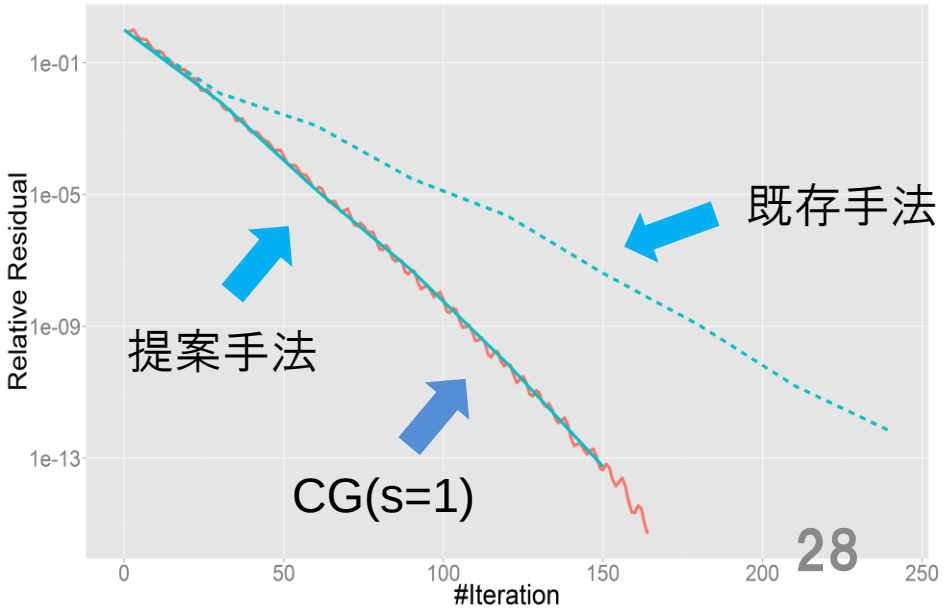
- **目的:** 安定性を改善してsを増やせるようにし,通信をさらに削減する
- **手段:** s -stepで作られる,s次元のクリロフ基底の性質を良好に保つ

通信削減アーノルディ(提案手法)

アーノルディ法が作る直交クリロフ基底=数値的には最良

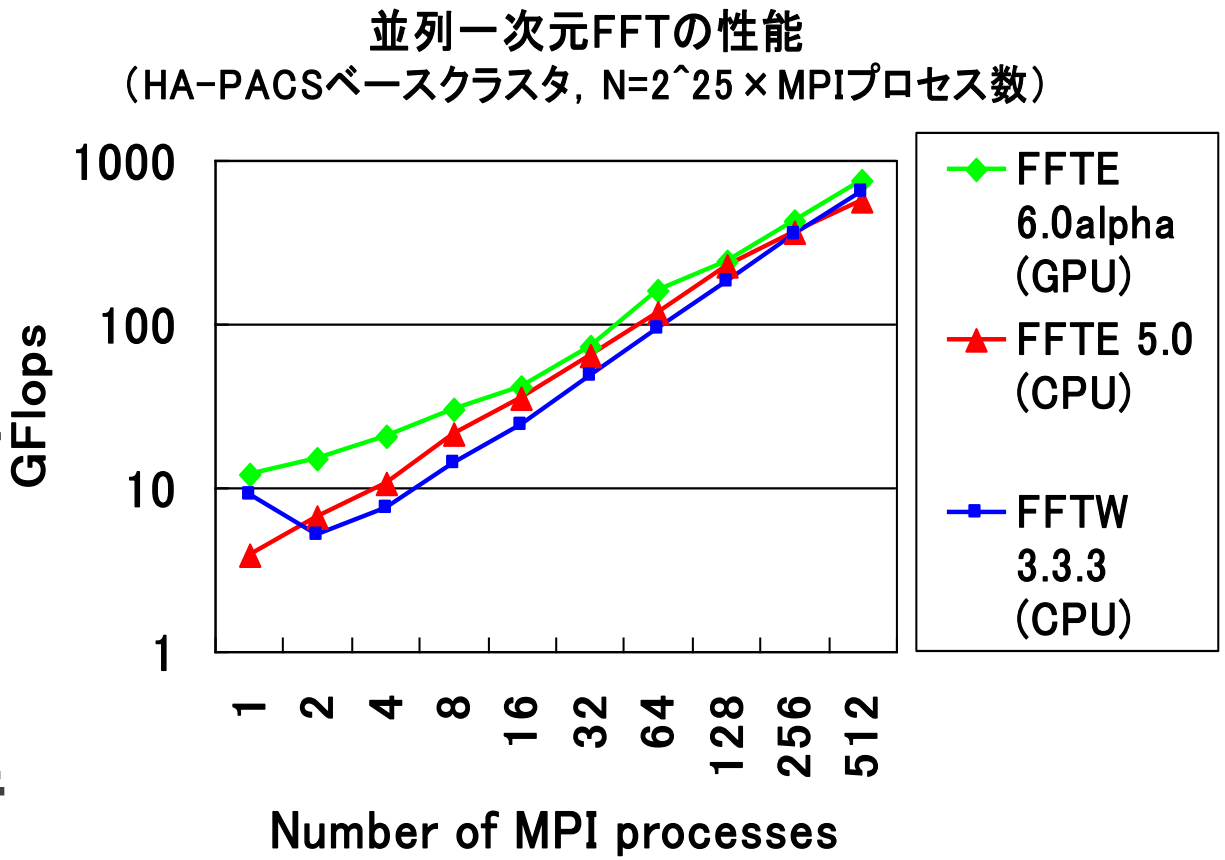
- **方針:アーノルディを通信削減化**
- **概要:**
  - Step1 各プロセッサが局所的に直交する基底を個別に生成
  - Step2 バタフライ通信で一つの基底へとマージ
- **性質:**
  - 全体通信回数1 ($1/s$倍)
  - 演算増$O(s^3 \log P)$
- **実験結果: 通常のCGと同程度**
  =最良の収束速度
- **今後の予定**
  - 大規模問題での実験
  - 提案手法の誤差解析

CG法へ適用時の収束履歴(mesh2e1 s=30)



GPUクラスタ向け1次元FFTライブラリの実装

- 複数のプラットホーム上における数値計算ライブラリを開発することで，アプリケーション開発者から見た場合，異なるシステムでも同一のインターフェースで性能可搬性を実現する．
- アーキテクチャが似ているプラットホーム間においては，パラメータの自動チューニングや実装の切り替えを行うことで性能可搬性を実現する．



[Takahashi+@CSE2013, to appear]

最近のその他の成果（抜粋）

- **滝沢グループ**
  - OpenCL＋MPIプログラミングの抽象化 (PLC2013)
  - OpenACCの最適化技法とその限界(ATMG2013)
  - アクセラレータのメモリ容量を超えるアプリへの対応
- **高橋グループ**
  - 代数的多重格子法（AMG）ライブラリの自動チューニング（IEICE trans. D）
  - GPUにおける3倍・4倍精度浮動小数点演算の実現と性能評価（IPSJ ACS）
  - GPUにおけるCRS形式疎行列ベクトル積の自動チューニング（CSE2012）
- **須田グループ**
  - GPU / Xeon phi による高性能化（SC12 poster 他）
  - 高性能・省電力のための自動チューニング（iWAPT 他）
  - ディレクティブベースの並列化の検討（SIAM CSE 他）
- **江川グループ**
  - HPCリファクタリングカタログ(LHAM2013, WSSP16, SENAC)
  - レガシーコードの性能可搬性調査(HPC研究会, ICFD2013)
  - レガシーコードのマイグレーションツール

まとめ

- 新世代システムにアプリを進化的に適応させるために・・・
  - HPCリファクタリング
    - システム依存性を切り離すためのガイドライン
    - コード編集を支援するツール群
  - コード変換フレームワーク Xevolver
    - ASTをXML形式でコンパイラの外部に公開
    - 構文情報に基づく解析や独自の変換規則の定義
  - 新世代システム向けに最適化された実装・アルゴリズム
    - クリロフ部分空間法の通信削減
    - GPUクラスタ向けの1次元FFT

今後の方針

各グループの成果の統合と実アプリを用いた評価