2016 6/1 9:15 -30 B-5-2「第21回 計算工学講演会」

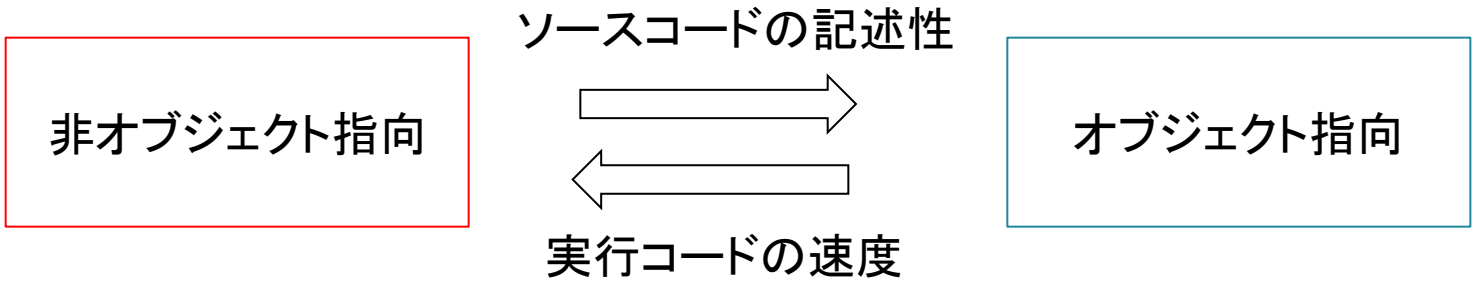
C++数式テンプレート変換に基づく有限体積法のための領域特化言語

伊藤 正勝, 宮島 敬明, 藤田 直行

宇宙航空研究開発機構　数値解析技術研究ユニット

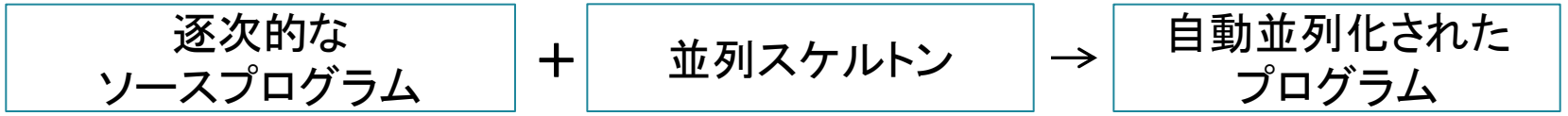
1. 序: なぜ、領域特化言語？　並列スケルトン？

- 計算科学：C++のジレンマ



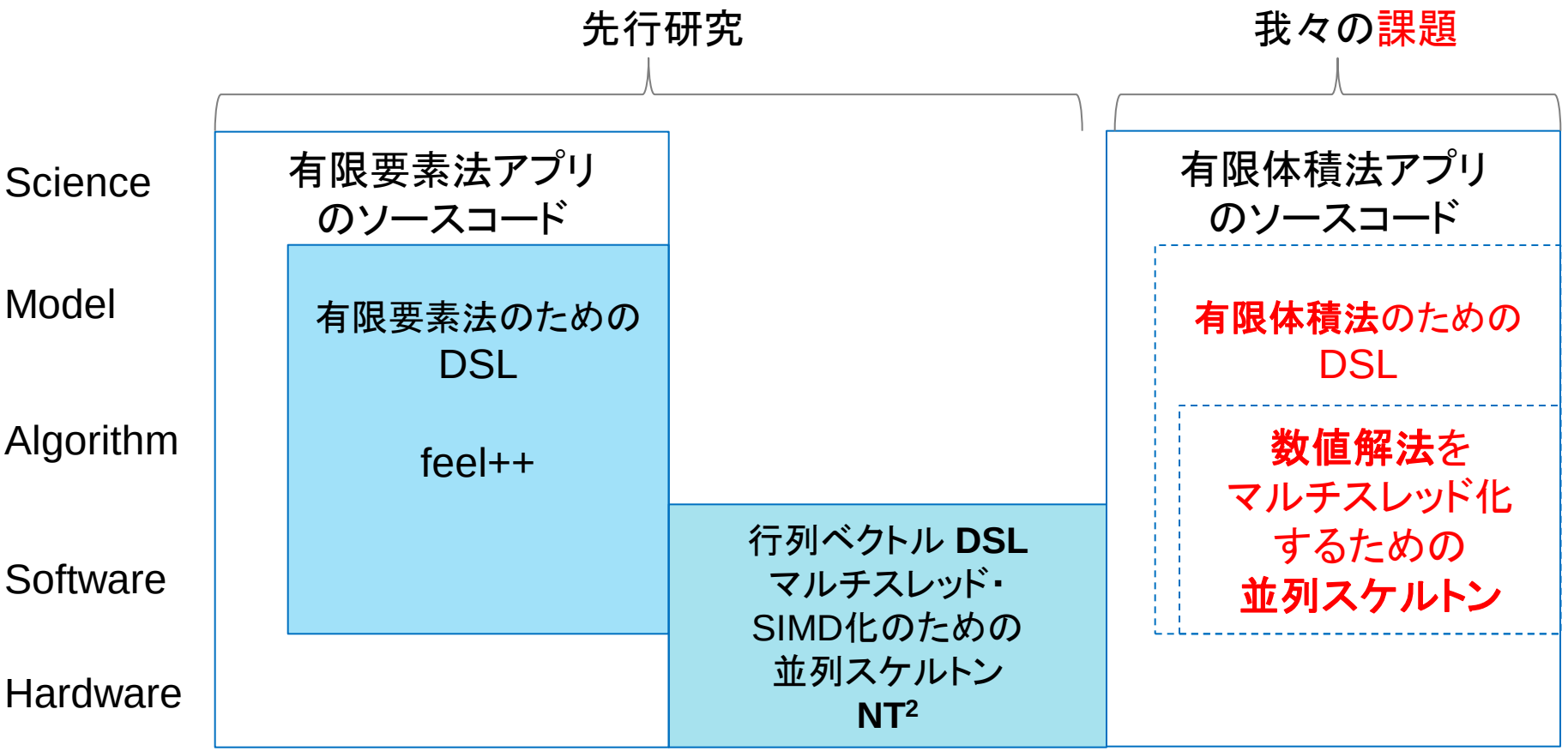
- 計算機科学：C++の記述性と速度の両立
  - 領域特化言語(DSL)のC++への埋め込み

  - スケルトン並列プログラミング



1.1 先行研究と課題

計算科学ソフトウェアの階層構造



2. 方法：数式テンプレートに基づく領域特化言語

- C++テンプレートライブラリ
  - 領域特化言語(DSL)とC++のソースコードを混在させられる。
- 埋め込み型の領域特化言語
  - 実装法 – 数式テンプレート変換のためのメタプログラミング
    - 煩雑、コンパイルエラーが異常に長い。

DSLをC++へ埋め込むためのDSL

- Boost.Protoライブラリ

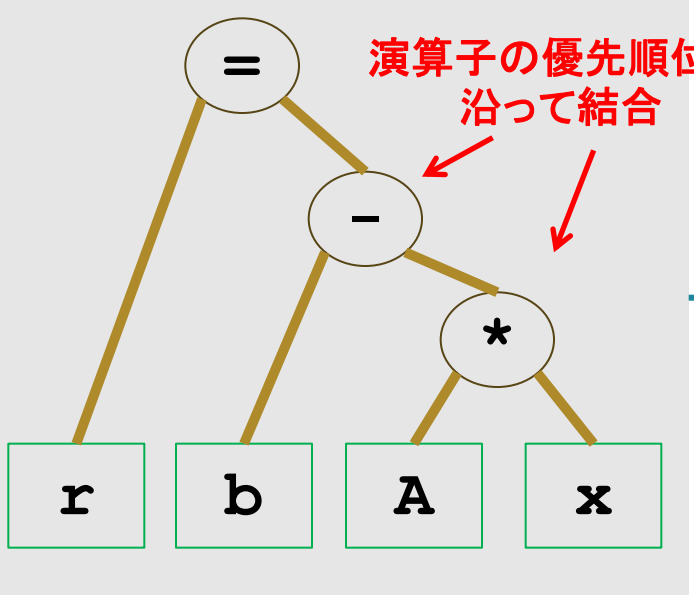
アプリケーションプログラムの階層構造

2.1 行列ベクトル数式からC++プリミティブへ

DSLによるソースコード
行列、ベクトルの数式

```
r = b - A * x;
```

数式テンプレート



数式テンプレートを中間コード
として、変換を繰り返す。

C++の基本式（プリミティブ）
に帰着させる。

並列化されたC++コード

```
#paragma omp parallel shared( r)
for (int i = 0; i < sz; i++) {
   r.data[i] = 0.0;
   for (int j = 0; j < sz; j++)
     r.data[i] +=
         b.data[i] - A.data[i][j] * x[j];
}
```

実行ファイル

2.2 有限体積法モデルからC++の基本式へ

数式テンプレートを介したソースコード変換

オブジェクト
DSLによるソースコード

グリッド

演算子

$$k\frac{d}{dx}\cdot\frac{d}{dx}-s\hat{I}d$$

→ 行列

定数項

$$-s\,T_\infty$$

→ ベクトル

C++の基本式
プリミティブ

組み込み型の
変数

配列

有限体積法における数式変換

$$\int_{\Delta x_i}\left(k\frac{d^2}{dx^2}T(x)-s\,T(x)\right)dx=-\int_{\Delta x_i}s\,T_\infty dx$$

→

$$\frac{k}{\Delta x}T_{i-1}-\frac{2k}{\Delta x}T_i+\frac{k}{\Delta x}T_{i+1}+s\Delta x\,T_i=-s\,T_\infty\Delta x$$

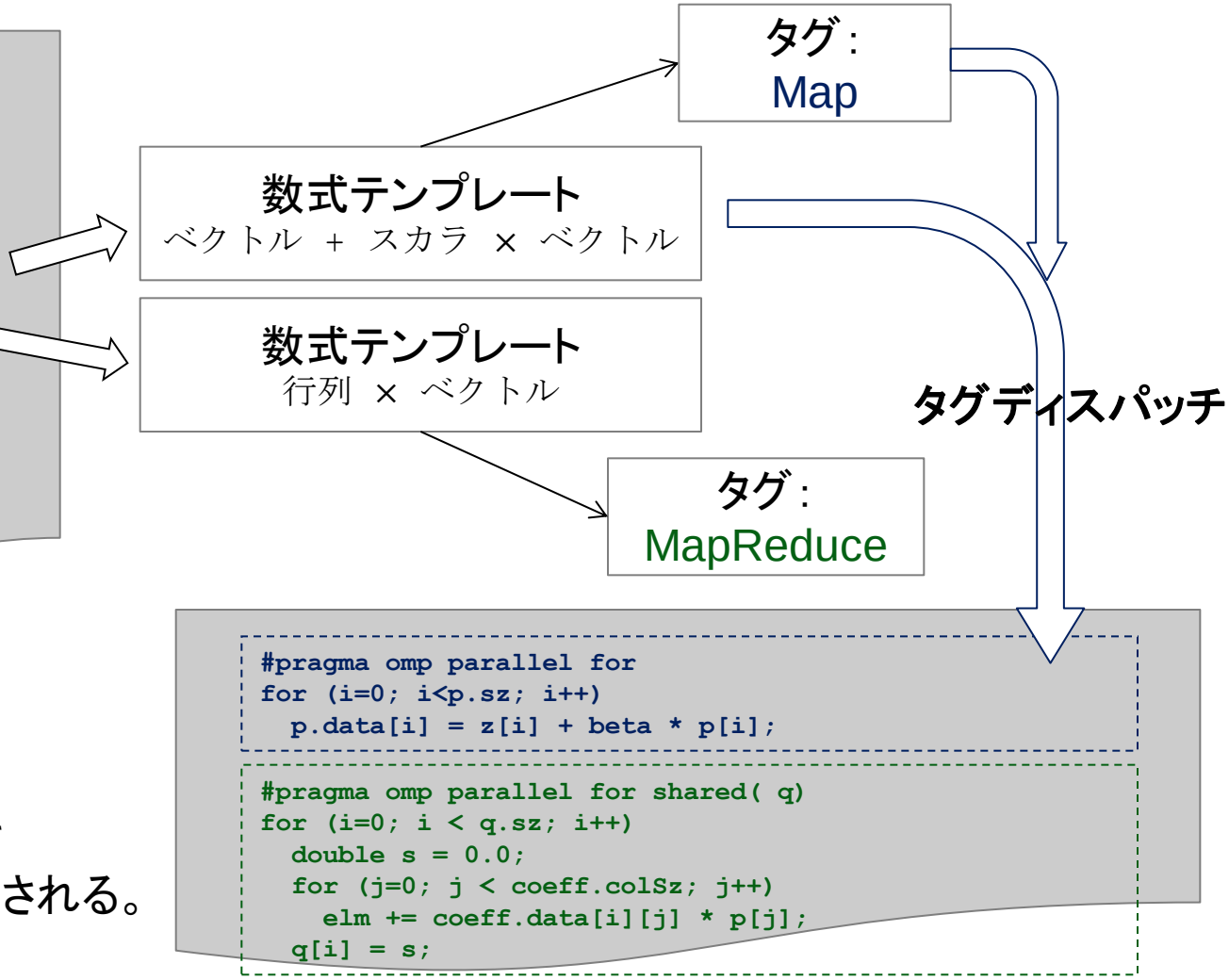
2.4 C++基本式への変換プロセスにおける並列スケルトンの組み立て

反復法の領域特化言語(DSL)による
ソースコード

```
// ...

p = z + beta * p;
q = coeff * p;

// ...
```



並列スケルトンの
組み合わせとして
C++コードが生成される。

3.適用例：領域特化言語によるプログラミング

一次元熱伝導ソルバのソースコード(*)

```
int main(int argc, char *argv[]) {

    int NumCtrlVol = atoi( argv[1] );
    FVM::Grid1D< FVM::CentDiffSchemeTag > grid( NumCtrlVol, CylinderLength);

    grid.addDirichletBoundary(-1, 0, HotTemperature);
    grid.addNeumannBoundary( NumCtrlVol, NumCtrlVol - 1, 0.0);
    const FVM::IdentityOperatorType IdOpr = FVM::IdentityOperatorType();
    const FVM::DifferentialOperatorType const DiffOpr = FVM::DifferentialOperatorType();
    auto opr = proto::deep_copy( ThermalConductivity * Area * DiffOpr * DiffOpr
                                 – ConvectiveHeatTransCoeff * Circumference * IdOpr );

    Matrix coeffMat( NumCtrlVol, NumCtrlVol);
    coeffMat = grid.discretizeOperator( opr );
    Vector rhsVec( NumCtrlVol);
    rhsVec = grid.discretizeFunction( - ConvectiveHeatTransCoeff *
                                      Circumference * AmbientTemperature );
    FVM::BoundaryCorrector bCorrector( grid, opr); bCorrector.applyTo( coeffMat);
    bCorrector.applyTo( rhsVec);

    SLA::DiagonalPreconditioner precond( coeffMat);
    SLA::ConjugateGradient< DLA::Matrix, SLA::DiagonalPreconditioner >
                                                    cg( coeffMat, precond);
    const Vector tempGuess( NumCtrlVol, (100.0 + 20.0) / 2.0);
    Vector temperature( NumCtrlVol);
    temperature = cg.solve(rhsVec, tempGuess, convergenceCriterion);

    printCalculatedAndExactTemperatureDistributions< Vector >( temperature);

     return 0;
}
```

グリッド

熱伝導問題の記述

$$T(0) = T_B, \quad \frac{dT(L)}{dx} = 0$$

$$k\frac{d^2}{dx^2}T(x) - s\,(T(x) - T_\infty) = 0$$

離散化
境界条件による補正

前処理付き
共役勾配法

(*) https://github.com/masa-ito/ProtoToPoisson/blob/master/src/test/airCooledCylinder.cpp

3.1 領域特化言語の使用法

- 領域特化言語の実体：テンプレートライブラリ

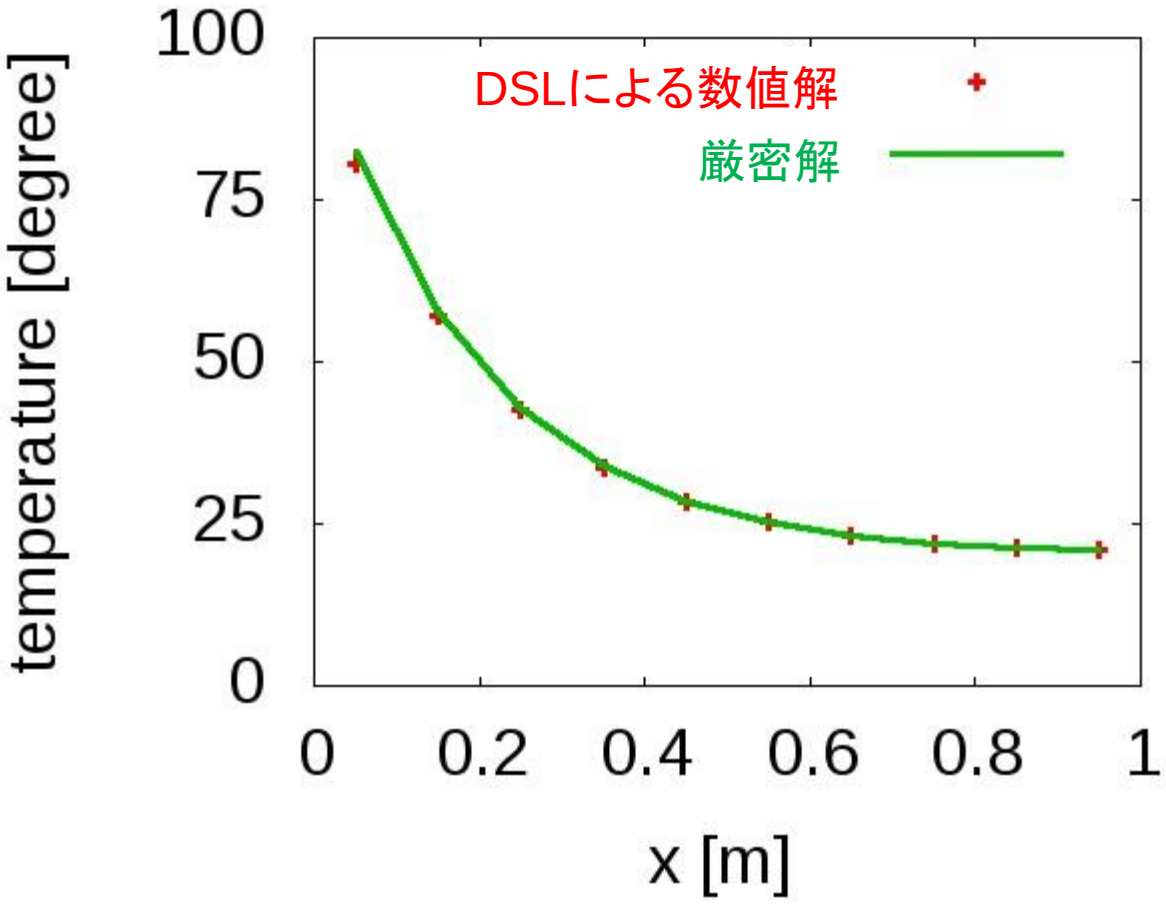
```
#include <FiniteVolumeMethod/FiniteVolumeMethod.hpp>
```

- 自動マルチスレッド化
  - ソースファイルの先頭で、OpenMP用のタグをインクルード

```
#ifdef _OPENMP
#include <ParallelizationTypeTag/OpenMP.hpp>
#endif
```

  - コンパイルオプションでOpenMPを指定すると
    - 逐次的なソースコードが、OpenMPでマルチスレッド化されたコードに変換される。

4. 結果：領域特化言語の動作検証



適用例(第3節)の熱伝導問題

L = 1(m)

T_B = 100(℃)

T_∞ =20(℃)

k = 1.0 x 103(熱伝導係数)

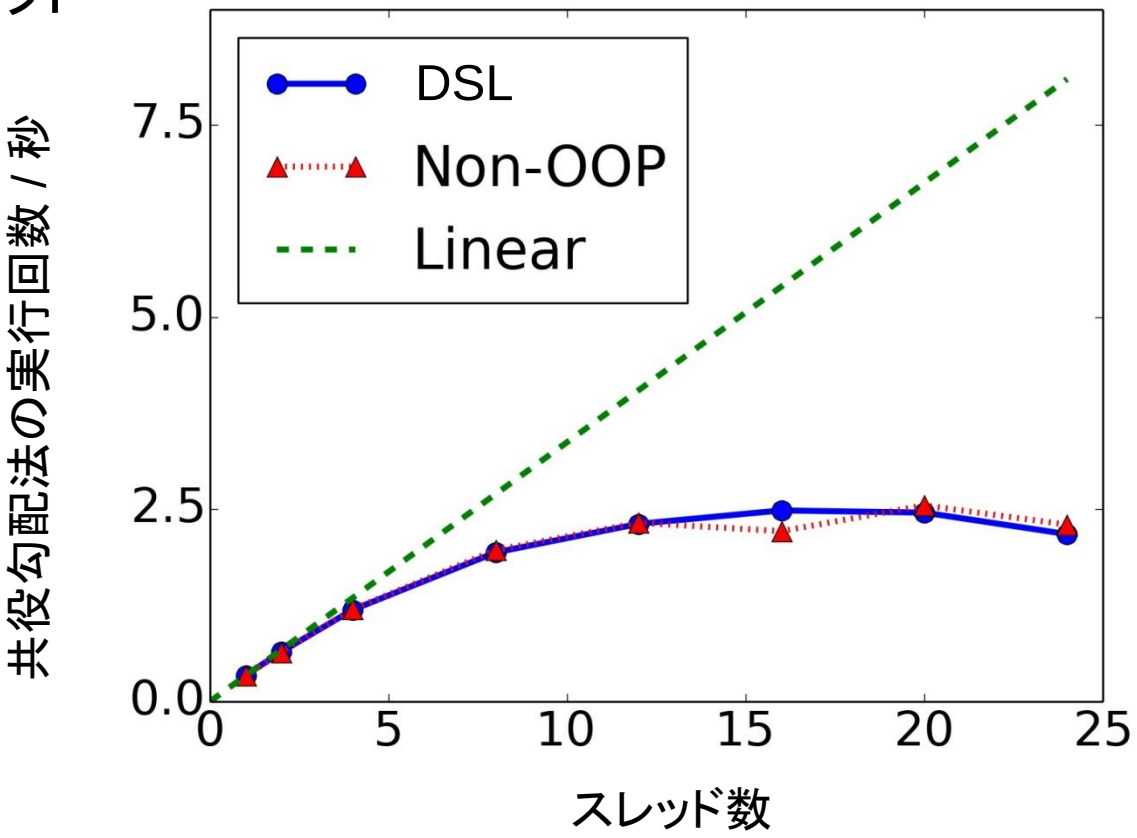
N = 25 (/m^2)

グリッド点 10個

- 非オブジェクト指向型プログラミング(C++)での数値解と一致
- 厳密解との最大誤差 2%

4.1 領域特化言語 vs 非オブジェクト指向

OpenMP化されたプログラムの実行速度をスレッド数に対してプロット



DSL

- 領域特化言語によるプログラミング
- コンパイラによる並列化

Non-OOP

- C++によるオブジェクトを使わないプログラミング
- 手動並列化

4.1 領域特化言語 vs 非オブジェクト指向（続）

- 並列化性能の計測条件
  - 点ヤコビ前処理付き共役勾配法の経過時間
  - グリッド点 1500個、計測80回の平均
- ハードウェア
  - 2プロセッサ x 12コア　Intel Xeon E5-2697 v2(30M Cache, 2.70 GHz)
  - メモリ 128Gb
- コンパイラオプション GNU C++ 4.8.5

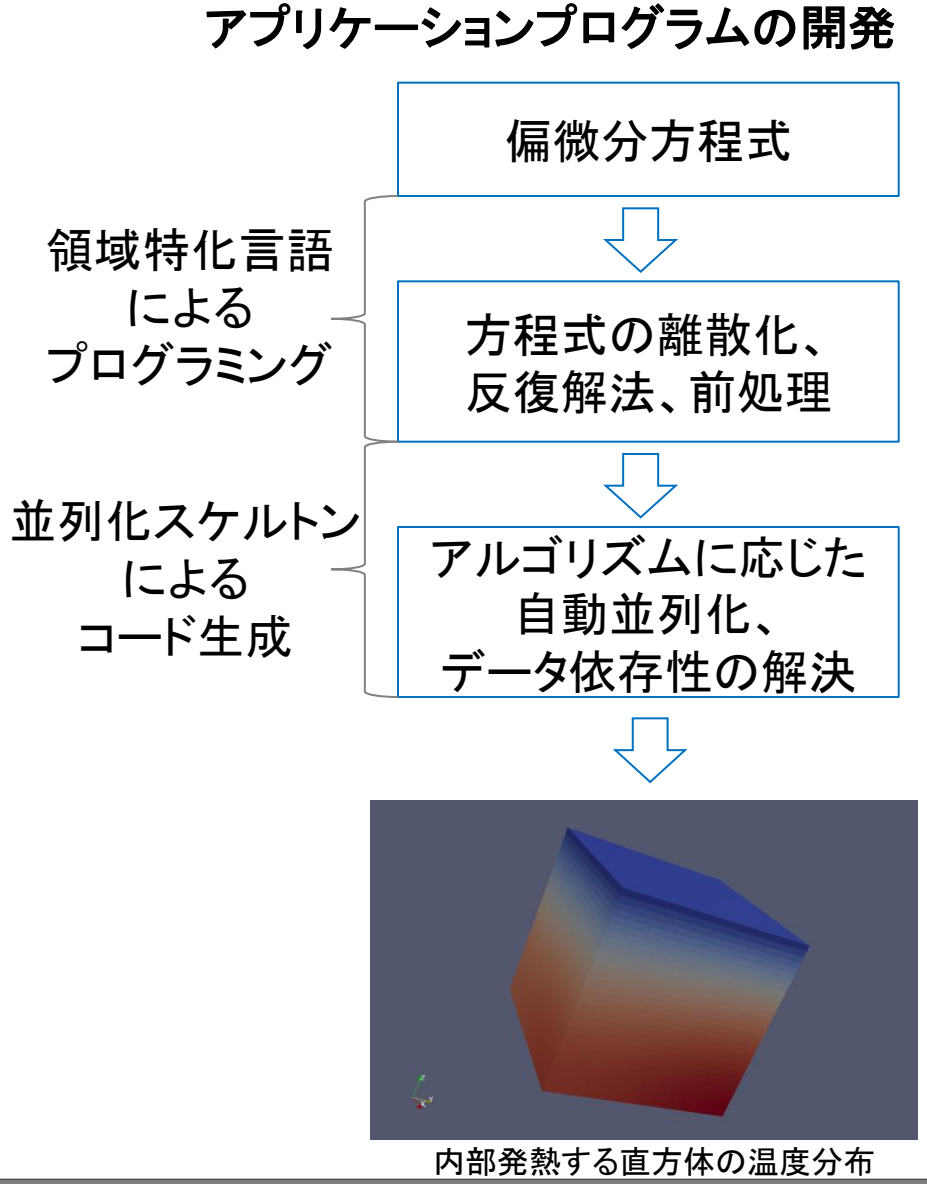
```
-std=c++11 -O3 –Winline --param max-inline-recursive-depth=32 ¥
--param max-inline-insns-single=2000
```

5. まとめ

- 有限体積法のための領域特化言語
  - C++テンプレートライブラリとして実装。
  - 一次元熱伝導方程式ソルバに適用可能。
- 並列スケルトン
  - 逐次型のソースコードからOpenMPで並列化されたプログラムが生成される。
- 記述性と実行速度の両立
  - 領域特化言語、並列スケルトンのどちらも、実行速度の低下は見られなかった。

5.1 今後の課題

- 領域特化言語の拡張
  - 有限体積法の離散化モデル
  - 数値解法
- 領域特化言語と並列スケルトンの相関
  - データ依存性の解決
  - オーダリング付きの並列化コードの生成



内部発熱する直方体の温度分布

宇宙航空研究開発機構
Japan Aerospace Exploration Agency

質問をお願いいたします。