FIBER

fiber-miniapp.github.io

Fork me on GitHub

Fiberミニアプリの性能評価

小村幸浩, 鈴木 惣一朗, 三上 和徳, 滝澤 真一朗, 松田 元彦, 丸山 直也

理化学研究所 計算科学研究機構

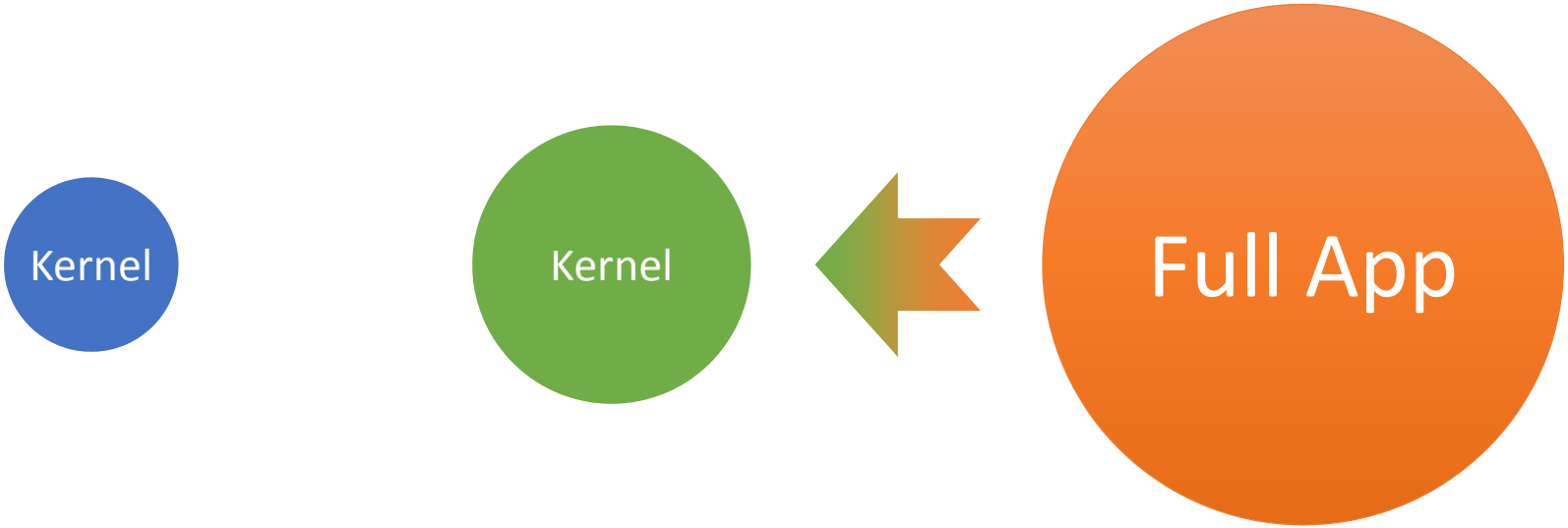
エクサスケールコンピューティングプロジェクト

アプリケーション開発チーム

FIBER

Fork me on GitHub

fiber-miniapp.github.io



- ループネスト1個程度
- 数十～百行程度
- 通信なし
- ファイルI/Oなし

✔ 単純

アプリケーション全体との乖離

- 元アプリを機能削減し簡略化
- 全体として元アプリと同等の構造を保持
- 通信、ファイルI/O有
- 数千～数万行程度
- ビルド、実行方法、結果確認等の整備
- オープンソース

「実際のアプリに即した評価」をもっと簡単に！

- 多数の開発者によって長期間にわたって開発
- 数万～数十万行
- 様々な並列化、通信、ファイルI/O

✔ 真のターゲット

入手可能とは限らない

プログラムが巨大

利用方法が複雑（入力準備、コンパイル、正当性等）

(1)先行ミニアプリプロジェクト

- Mantevo
  - 米国Sandia国立研究所
  - http://mantevo.org
  - PDE, MD など
- ExaCT
  - 米国DoE Codesign center
  - http://exactcodesign.org/
  - Combustion ミニアプリ
- LULESH
  - 米国Lawrence Livermore国立研究所
  - Hydrodynamics
  - Many variants: Serial, MPI, OpenMP, CUDA, OpenACC, Chapel, Charm++, Listz
- 現在はヨーロッパでも

(1) Fiberミニアプリ

FIBER
fiber-miniapp.github.io
Fork me on GitHub

日本発のミニアプリ集

- アプリFS[1]のロードマップ課題を解決するための
  アプリケーション（フルアプリ）から抽出したミニアプリ集
- 多くは京，Intelクラスタ等で動作
- フルアプリ開発者との協力のもと理研AICSにて整備・開発
- 東北大，筑波大，東大各チームに評価用ベンチマークとして提供

- フルアプリの提供を募集
  - 17本のアプリ提供あり
  - アプリFSにてミニアプリ化
- ミニアプリ版の提供
  - 8本の提供あり
  - アプリFSにて追加整備（ドキュメント等）

[1]「将来のHPCIシステムのあり方の調査研究・アプリケーション分野」

(1)本発表の目的

FIBER
fiber-miniapp.github.io
Fork me on GitHub

1. ミニアプリの作業方法の説明　(Marble Miniを例に)
2. 現在整備がほぼ終了しているアプリ　(1) Marble Mini (2)CCS QCD (3)FFVC Mini (4) NGS Analyzer Mini の概要説明と定量的な性能モデルの構築
3. 性能モデルから計算科学ロードマップに提示された要求性能の実現可能性について議論

- ミニアプリはGithubを利用して，ソースコードリポジトリとして公開
- 公開Webサイト

  http://fiber-miniapp.github.io/
- 現在公開中のアプリ

(1) CCS QCD　(2)NGS Analyzer Mini (3)FFVC Mini (4) Marble Mini (5) NICAM DC (6) mVMC Mini (6アプリ)

(2) ミニアプリ Marble Mini

(2)古典MD(分子動力学)アプリMARBLEの ミニアプリ化方針

FIBER
fiber-miniapp.github.io
Fork me on GitHub

- 方針
  - 生体高分子系のMDシミュレーション時のパフォーマンス特性を(なるべく)再現できること
  - MDアプリとしての最低限の機能は残す
  - 上記2条件の下で，なるべく小さなプログラムにする

- 機能を制限してプログラムを小型化
  - 水分子系(高分子なし)専用
  - NVE(ミクロカノニカル)アンサンブル専用
  - その他に削れる機能(エネルギー最小化，位置束縛，…)
- 特に，水分子系専用にすると
  - 削れるコード部分が多い
  - 非専門家でも入力データ作成が容易
  - 任意サイズのデータが作成可能 → 弱スケーリング測定が容易

→ ミニアプリ　Marble Mini を作成

(2) Marble Mini のプログラム構造とモデル化

FIBER
fiber-miniapp.github.io
Fork me on GitHub

〇プログラム構造

時間ステップループ{
  −速度の計算
  −位置の計算
  −力の計算
    −単距離成分の二体力計算
    −長距離成分のParticle Mesh Ewald(PME)法
  −速度の計算
}

赤字：ホットスポット

特徴
1. SPMDモデルで各プロセス内はセル対毎にOpenMP並列
2. 空間3次元にMPIプロセスをマッピング
3. 通信として （1) 空間3次元に対する袖領域の粒子に対する通信 (2) 3次元FFTのAlltoAll (3)全体通信

〇モデル化の方針

(i) 単距離成分の二体力計算部分の性能モデルを評価

(ii) 長距離成分のPME法部分の性能モデルを評価

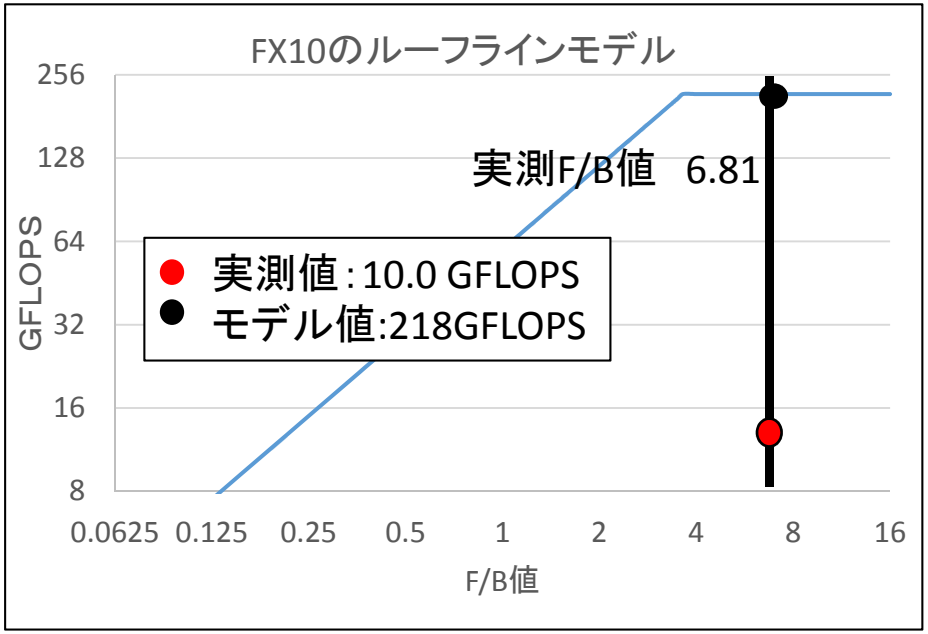
Marble Mini が提供している入力データを用いて，性能を評価する．
1ノード： 粒子数2688，単距離用セル (4, 4, 4)， PME用セル (32,32,32)

(2)単距離成分の二体力計算部分の性能モデル

FIBER
fiber-miniapp.github.io
Fork me on GitHub

通信はない，ルーフラインモデルで $T_{2体力}$ の性能モデルを構築

FX10 のシステム付属のプロファイラ機能を使用



FX10の実測値
10.0GFLOPS，　1.5 GB/s
→　実測F/B値　6.81

ルーフラインモデル

$$\mathrm{perf} = \min(FLOPS, 6.81 \times Bandwitdh)$$

$$T_{2体力} = M/perf$$

FLOPS : 1ノードの演算性能
Bandwitdh: 1ノードのメモリバンド幅
M : 演算量

性能低下の原因：粒子番号のランダムアクセスとなり，L1キャッシュミスが90%以上発生している．キャッシュからレジスタへのデータ移動が時間の主となり，性能が出ていないと予想される．
解消方法：粒子番号を整列し，小さなブロックに小分けし計算することで性能が向上すると予想される．

(2)長距離成分PME法の計算の流れ

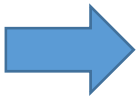
FIBER
fiber-miniapp.github.io
Fork me on GitHub

〇PME部分の計算の流れ

i. 粒子からメッシュへの電荷の外挿（通信なし）
ii. Forward-FFT （AlltoAll通信）
iii. 波数空間の電場の計算（通信なし）
iv. Backward-FFT （AlltoAll通信）
v. メッシュから粒子への電場から力の外挿（通信なし）

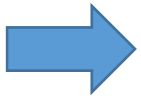
〇モデル化の方針

①i,iii, v は通信がないためルーフラインモデルを用いてモデル化



それぞれの実測F/B値 i)0.29 iii)0.40 v) 1.46
二体力計算と同様性能が低い

② iiのForward-FFTとivのBackward-FFTの計算ではAlltoAll通信があり，新たにモデル化



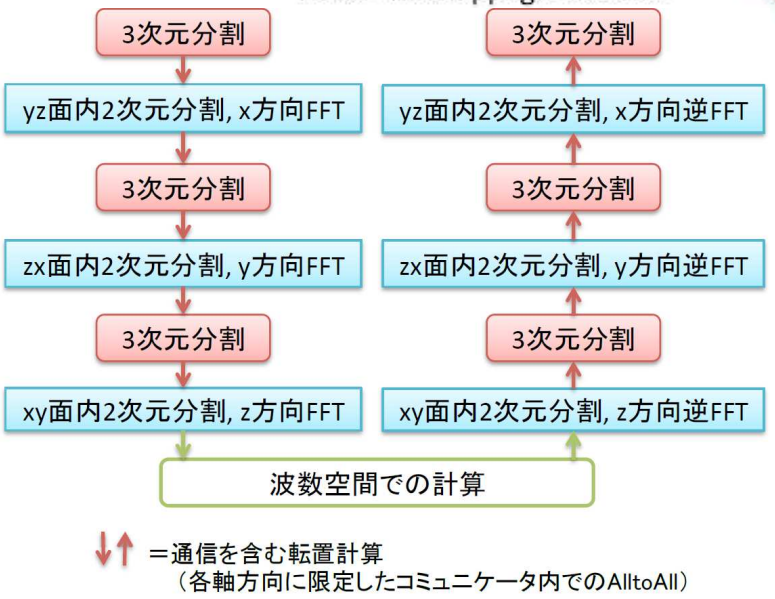
性能モデル
$T = T_{\text{FFT計算}} + T_{\text{AlltoAll通信}} + T_{\text{バッファ整理とコピー}}$

(2) Marble Mini の3次元FFTについて

FIBER

Fork me on GitHub

fiber-miniapp.github.io



○ Marble Mini の3次元FFT

Marble MiniがPME法で採用している3次元FFTの方法を右図に示す．1次元FFT計算部に対してはFFTWライブラリのルーチンを用いている．

○ 3次元FFT部分の計算部分，回数，データ量

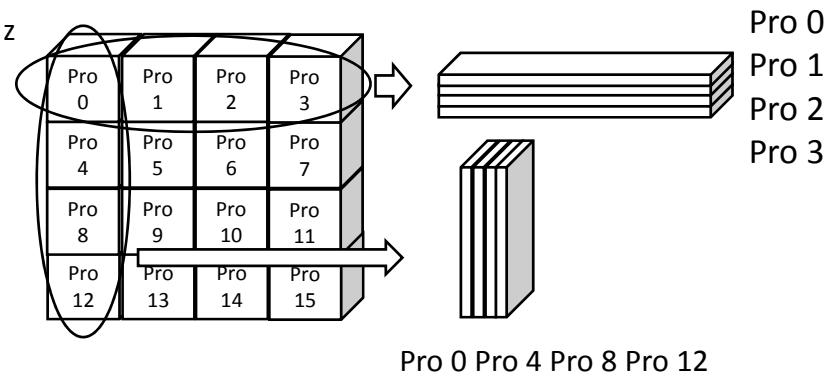
i. 1次元FFT計算：3回　$5N\log_2(P_i N_i)$

ii. 全体全通信：5回　$16N_i / P_i$ Byte (i=x,x,y,y,z)

iii. AlltoAll通信の整理と送受信バッファのコピー：計47N Byte

PME用セル 1ノード当たり(N_x, N_y, N_z) $N= N_xN_yN_z$

全プロセス数 (P_x, P_y, P_z)



(2) 3次元FFTの性能モデル

FIBER
fiber-miniapp.github.io
Fork me on GitHub

○ 3次元FFT部分(Forward-FFT)のモデル化

i. 1次元FFT 計算時間: $T_{FFT} = \sum_i \ 5Nlog_2(P_iN_i)/performanc \ \ i = (x, y, z)$

ii. 全体通信時間: $T_{AlltoAll} = \sum_i \ \frac{P_i^2}{4} l_{AlltoAll} + \frac{sN_i}{P_iB_{net}} \ \ i = (x, x, y, y, z)$

iii. AlltoAll通信の整理と送受信バッファのコピー: $T_{copy} = 47N$ / *Bandwitdh*

PME用セル 1ノード当たり(N_x ,N_y ,N_z) N= $N_xN_yN_z$
全プロセス数 (P_x , P_y , P_z)

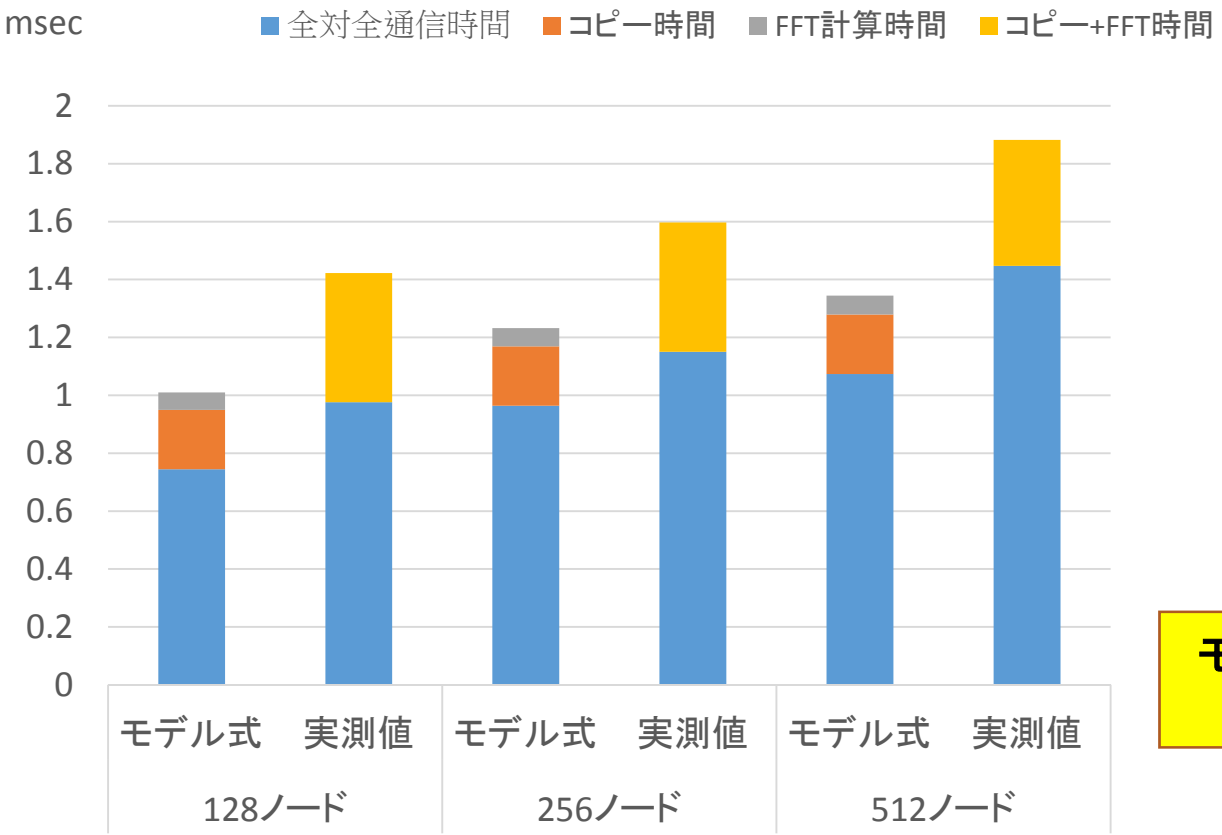
l : ネットワークレイテンシ
B_{net} : ネットワークバンド幅
$l_{AlltoAll}$:AlltoAllの1stepのレイテンシ
s : データ長
performance : 性能(F/B値 = 1)
Bandwitdh : メモリバンド幅

(2) 3次元FFTの性能モデルと実測値の比較

FIBER
fiber-miniapp.github.io
Fork me on GitHub

東京大学情報基盤センターのFX10を用いた比較

モデル式： $T = T_{FFT計算} + T_{AlltoAll通信} + T_{バッファ整理とコピー}$ msec



東大FX10の性能値
B_{net} : 4.76 GB/s
$l_{AlltoAll}$: 8usec
perfoemance : 218GFLOPS
Bandwitdh : 60GB/s

モデルと実測値に大きな差がないことがわかる

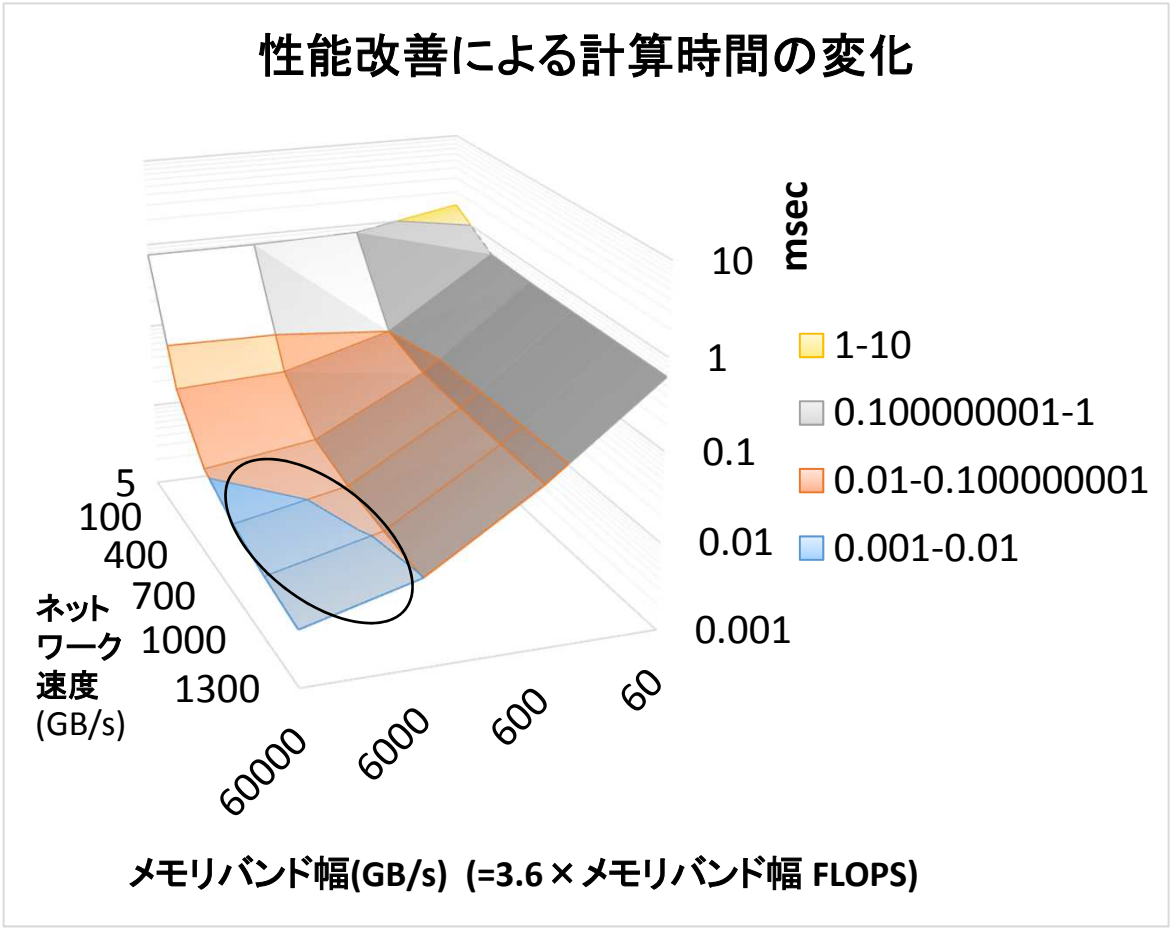
(2)要求性能の実現可能性のための条件

FIBER
fiber-miniapp.github.io
Fork me on GitHub

〇計算科学ロードマップに提示された要求性能

10万原子を100ノード程度で計算．多くのアンサンブルが望まれる．

次世代スパコンが 10万ノードあると仮定すると 5.0e-3 msec/step



仮定(1) 10万ノード数を想定
仮定(2) 超大規模系での計算性能の評価をする
仮定(3)性能はルーフラインモデルと同等．
仮定(4)二体力計算とFFTと逆FFTのみの部分のみを考慮
仮定(5) 1ノード1000粒子で1サンプルにつき128ノード使用

FLOPS，メモリバンド幅，ネットワーク速度，ネットワークレイテンシ全ての改善が必須

レイテンシを0.01 usec と固定する．
丸部分が要求性能値になる．

(3)ミニアプリ CCS QCD

(3)ミニアプリ CCS QCDの概要

FIBER
fiber-miniapp.github.io
Fork me on GitHub

〇ミニアプリCCS　QCDの概要

- 高エネルギー物理学で用いられる格子量子色力学プログラム
- 最も計算コストがかかるクォーク伝搬関数の計算部分を抜き出したもの
- クォーク伝搬関数を4次元(空間3次元+時空1次元)格子上の大規模疎行列の連立1次方程式を解く

$$\begin{pmatrix} D_{EE} & D_{EO} \\ D_{OE} & D_{OO} \end{pmatrix} \begin{pmatrix} x_E \\ x_O \end{pmatrix} = \begin{pmatrix} b_E \\ b_O \end{pmatrix} \Rightarrow \begin{pmatrix} D_{EE} & D_{EO} \\ D_{OE} & D_{OO} \end{pmatrix} K \begin{pmatrix} z_E \\ z_O \end{pmatrix} = \begin{pmatrix} b_E \\ b_O \end{pmatrix}, \quad \begin{pmatrix} x_E \\ x_O \end{pmatrix} = K \begin{pmatrix} z_E \\ z_O \end{pmatrix}$$

K:前処理行列

〇プログラム構造

- Clover 項計算

BiCGStab反復ループ {
  −時空奇数ドメインの行列計算
  −時空偶数ドメインの行列計算
  −前処理部分の計算
  −残差計算
}

特徴
1. SPMDモデルで負荷分散はとれている
2. 空間3次元にMPIプロセスをマッピング
3. 通信は空間3次元に対する袖通信と全体通信

FIBER
Fork me on GitHub
fiber-miniapp.github.io

(3)性能モデルの構築方針

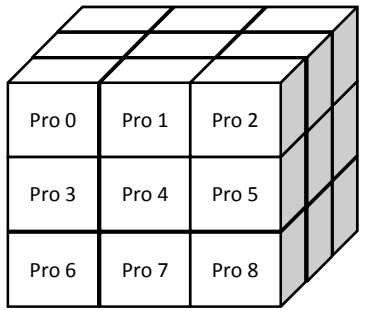
○モデル化の方針

BiCGStab反復ループ内のモデル構築

特徴
1. SPMDモデルで負荷分散はとれている
2. 空間3次元にMPIプロセスをマッピング
3. 通信は空間3次元に対する袖通信と全体通信

性能モデル
$T = T_{計算} + T_{袖通信コピー} + T_{袖通信} + T_{全体通信}$

計算科学ロードマップに提示された要求性能の実現可能性の議論のため
1ノードの大きさを $(N_x, N_y, N_z, N_T) = (8, 8, 8, 256)$ とし，性能モデルを構築する.

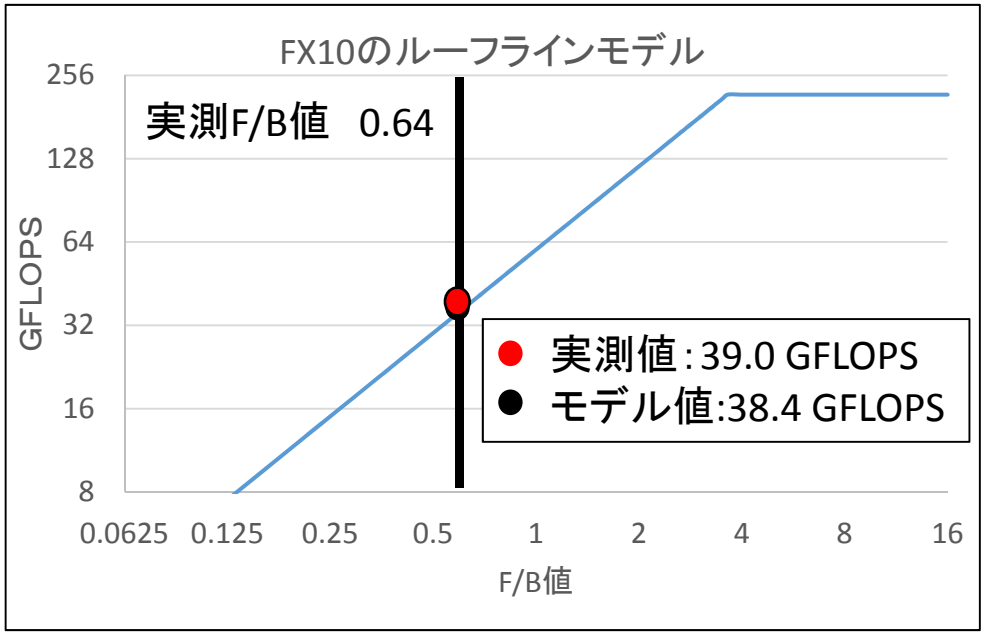


(3)単体性能のモデル化と性能比較

FIBER
fiber-miniapp.github.io
Fork me on GitHub

ルーフラインモデルを用いて，$T_{計算}$部分の性能モデルを構築

FX10 のシステム付属のプロファイラ機能を使用



FX10の実測値
39.0GFLOPS，　61.3 GB/s → 実測F/B値　0.64

ルーフラインモデル

$$\mathrm{perf} = \min(FLOPS, 0.64 \times Bandwitdh)$$

$$T_{計算} = M/perf$$

FLOPS : 1ノードの演算性能
Bandwitdh: 1ノードのメモリバンド幅
M : 演算量

〇性能比較

実測値39.0GFLOPS，性能モデル値38.4GFLOPS より，モデルより性能値が上回っている．

原因：モデルのピークとしたSTREAMのメモリバンド幅(60GB/s) より高い性能が出たため

(3)複数ノードに拡張した際のモデル化

FIBER
fiber-miniapp.github.io
Fork me on GitHub

$T_{袖通信コピー} + T_{袖通信} + T_{全体通信}$部分の性能モデルを構築

i. 複数ノードへの拡張による増加部分，回数，データ量
   A) 袖通信：12回　192 Ni byte
   B) 全体通信：Allreduce が 2stepで6回，16byteが3回，8byteが3回
   C) 送受信バッファのコピー：送受ともに12回　メモリアクセスが load 2 store 1 の 3×192 Ni byte

   1ノードのサイズ $(\mathrm{N_x}, \mathrm{N_y}, \mathrm{N_z}, \mathrm{N_T})$，$N_i = \frac{N_j N_k (1+N_T)}{2}$ $(i,j,k) = (x,y,z),(y,z,x),(z,x,y)$

ii. 複数ノード増加部分に対する各モデル
   A) 袖通信時間：$T_{袖通信} = 12(l + 192N_i/B_{net})$
   B) 全体通信時間：$T_{全体通信} = (l_{Allreduce} + s/B_{net}) \log_2(Proc)$
   C) 送受信バッファのコピー時間：$T_{袖通信コピー} = (12+12) \times 3l \times 192N_i/Bandwitdh$

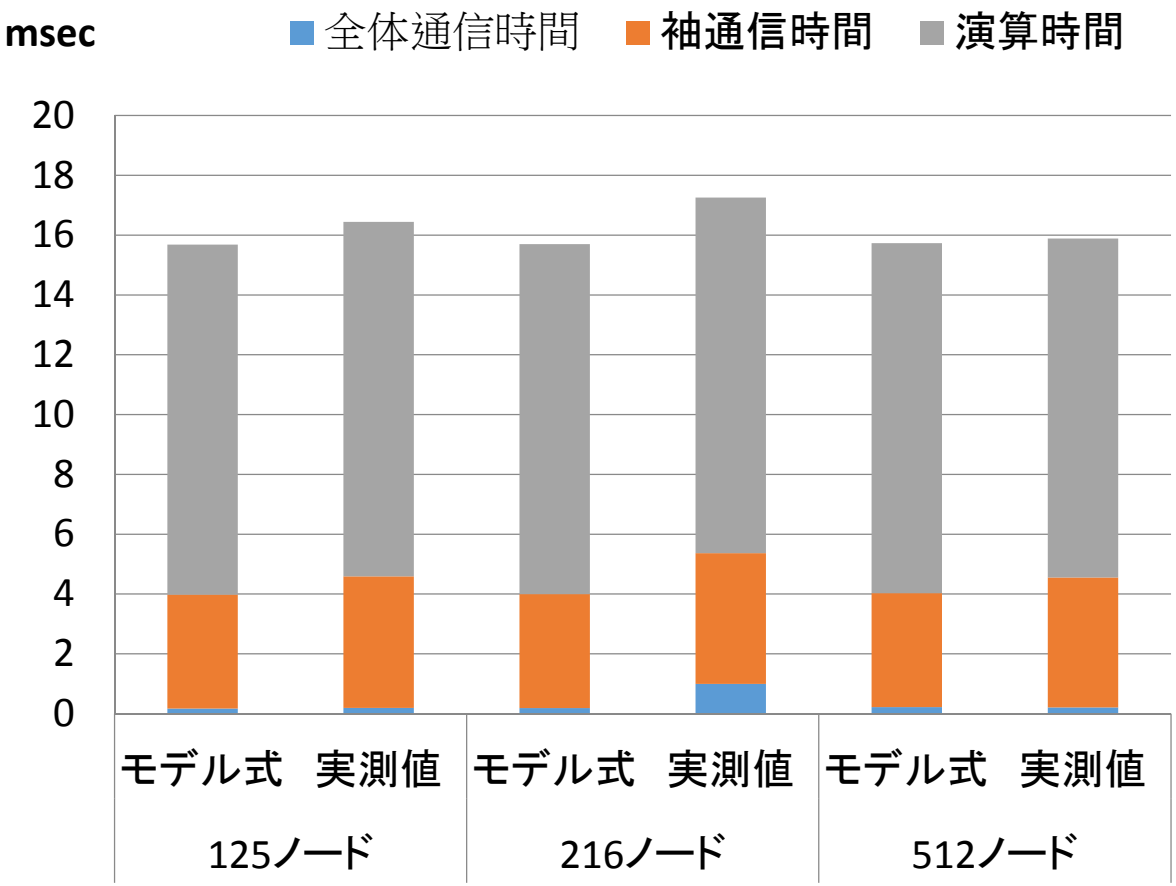
l：ネットワークレイテンシ
B_{net}：ネットワークバンド幅
$l_{allreduce}$：Allreduceの1stepのレイテンシ
s：データ長
$Bandwitdh$：メモリバンド幅

(3)性能モデルと実測値の比較

FIBER
fiber-miniapp.github.io
Fork me on GitHub

東京大学情報基盤センターのFX10を用いた比較

モデル式： $T = T_{計算} + T_{袖通信コピー} + T_{袖通信} + T_{全体通信}$
$= 9.8 + 1.9 + 4.0 + 2.4e\text{-}2 \log_2(Proc)$ msec



東大FX10の性能値
l : 0.92 usec
B_{net} : 4.76 GB/s
$l_{Allreduce}$: 8usec
s : 16byte
Bandwitdh ： 60GB/s

モデルと実測値に大きな差がないことがわかる

FIBER

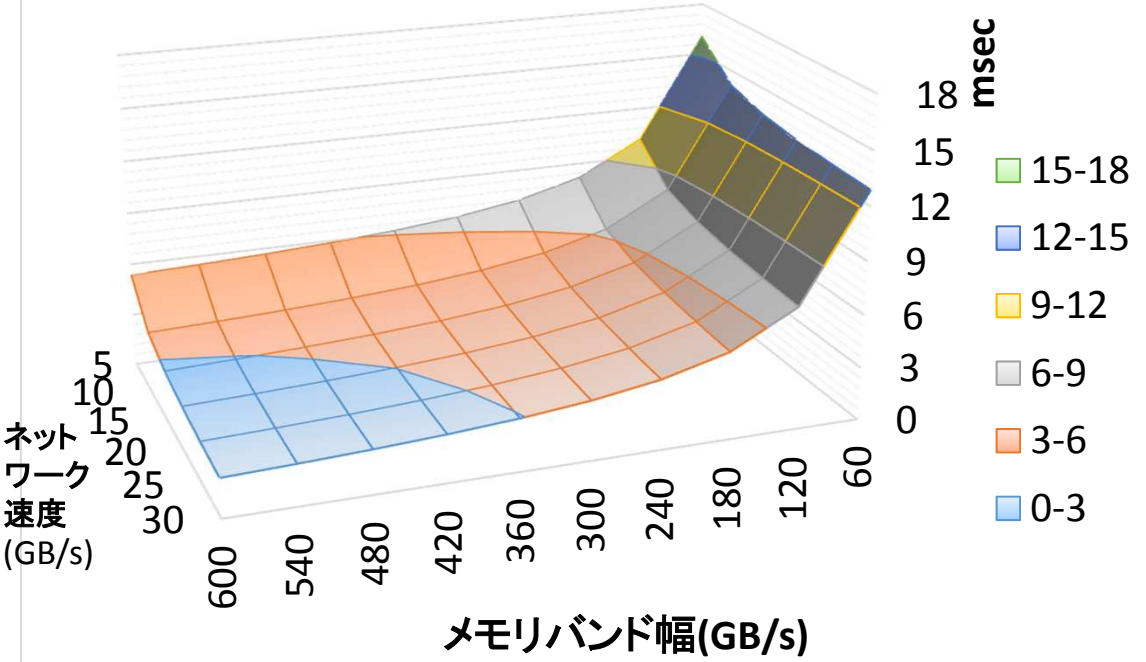
fiber-miniapp.github.io

Fork me on GitHub

(3)要求性能の実現可能性のための条件

○計算科学ロードマップに提示された要求性能の一つ

256^4 で格子で BICGStabの実行時間 3.1 msec / step 以下



仮定(1) 京と同じ規模のノード数を想定
仮定(2) 超大規模系での計算性能の評価をする
仮定(3) 1ノードのサイズを (8,8,8,256), 32768ノードでの評価

メモリバウンドのアプリのため,
メモリバンド幅の向上とネットワーク速度の向上が重要.

水色の領域が実現可能の領域である.

(4) ミニアプリ FFVC Mini

(4)ミニアプリ FFVC Miniの概要

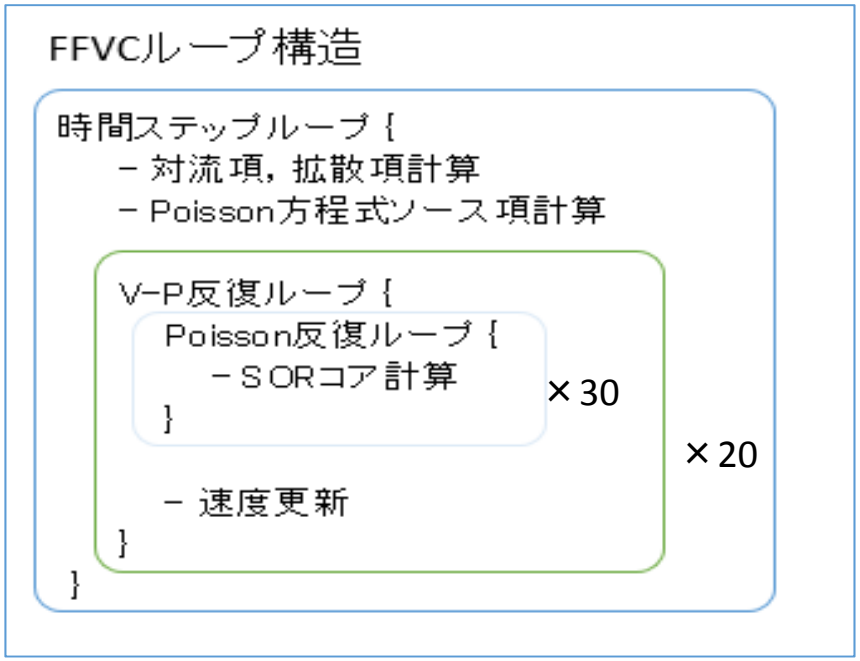
FIBER
fiber-miniapp.github.io
Fork me on GitHub

○ミニアプリFFVC Miniの概要

- 熱流体解析プログラム
- 3次元キャビティフローのベンチマークに特化
- 境界条件として外部境界のみ
- 使用アルゴリズムを固定:
  - 時間発展:Euler陽解法
  - 圧力Poisson方程式:ストライドアクセス型2色SOR反復法

○プログラム構造

特徴
1. SPMDモデルで負荷分散はとれている
2. 空間3次元にMPIプロセスをマッピング
3. 通信は空間3次元に対する袖通信と全体通信
4. V-P反復とPoisson反復の二重ループ
5. SORコア計算は 600回/時間ステップ呼ばれる



(4)性能モデルの構築方針

FIBER
fiber-miniapp.github.io
Fork me on GitHub

〇モデル化の方針

SORコア計算のモデル構築

特徴
1. SPMDモデルで負荷分散はとれている
2. 空間3次元にMPIプロセスをマッピング
3. 通信は空間3次元に対する袖通信と全体通信

性能モデル

$T = T_{計算} + T_{袖通信コピー} + T_{袖通信} + T_{全体通信}$

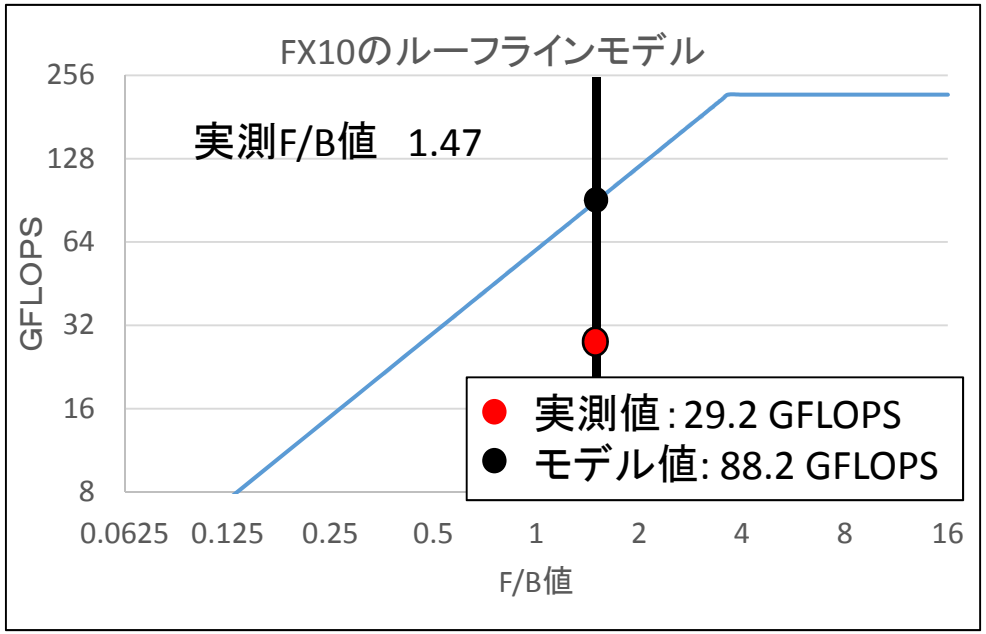
計算科学ロードマップに提示された要求性能の実現可能性の議論のため，1ノードの大きさを $(N_x, N_y, N_z)= (128, 128, 128)$とし，性能モデルを構築する.

(4)単体性能のモデル化と性能比較

FIBER
fiber-miniapp.github.io
Fork me on GitHub

ルーフラインモデルを用いて，$T_{計算}$部分の性能モデルを構築

FX10 のシステム付属のプロファイラ機能を使用



FX10の実測値
29.2GFLOPS，19.8 GB/s →実測F/B値　1.47

ルーフラインモデル

$\mathrm{perf} = \min(FLOPS, 1.47 \times Bandwitdh)$

$T_{計算} = M/perf$

FLOPS : 1ノードの演算性能
Bandwitdh: 1ノードのメモリバンド幅
M : 演算量

〇性能比較

実測値29.2GFLOPS，性能モデル値88.2GFLOPS より，モデルの3割程度の性能値

原因：整数演算があり，その部分が影響している可能性があげられる

(4)複数ノードに拡張した際のモデル化

FIBER
fiber-miniapp.github.io
Fork me on GitHub

$T_{袖通信コピー} + T_{袖通信} + T_{全体通信}$部分の性能モデルを構築

i. 複数ノードへの拡張による増加部分，回数，データ量
   - A) 袖通信：12回　4Ni byte
   - B) 全体通信：Allreduce が1回，8byte
   - C) 送受信バッファのコピー：送受ともに12回　バッファ変換のための演算が 44 N_i byte (シングルスレッド)

     1ノードのサイズ$(\mathrm{N_x}, \mathrm{N_y}, \mathrm{N_z})$，　$N_i = (N_j + 2)(N_k + 2)$　$(i,j,k) = (x,y,z),(y,z,x),(z,x,y)$

ii. 複数ノード増加部分に対する各モデル
   - A) 袖通信時間：$T_{袖通信} = 12(l + 4N_i/B_{net})$
   - B) 全体通信時間：$T_{全体通信} = (l_{Allreduce} + s/B_{net}) \log_2(Proc)$
   - C) 送受信バッファのコピー時間：$T_{袖通信コピー} = (12 + 12) \times 44N_i/perf\_1thread$

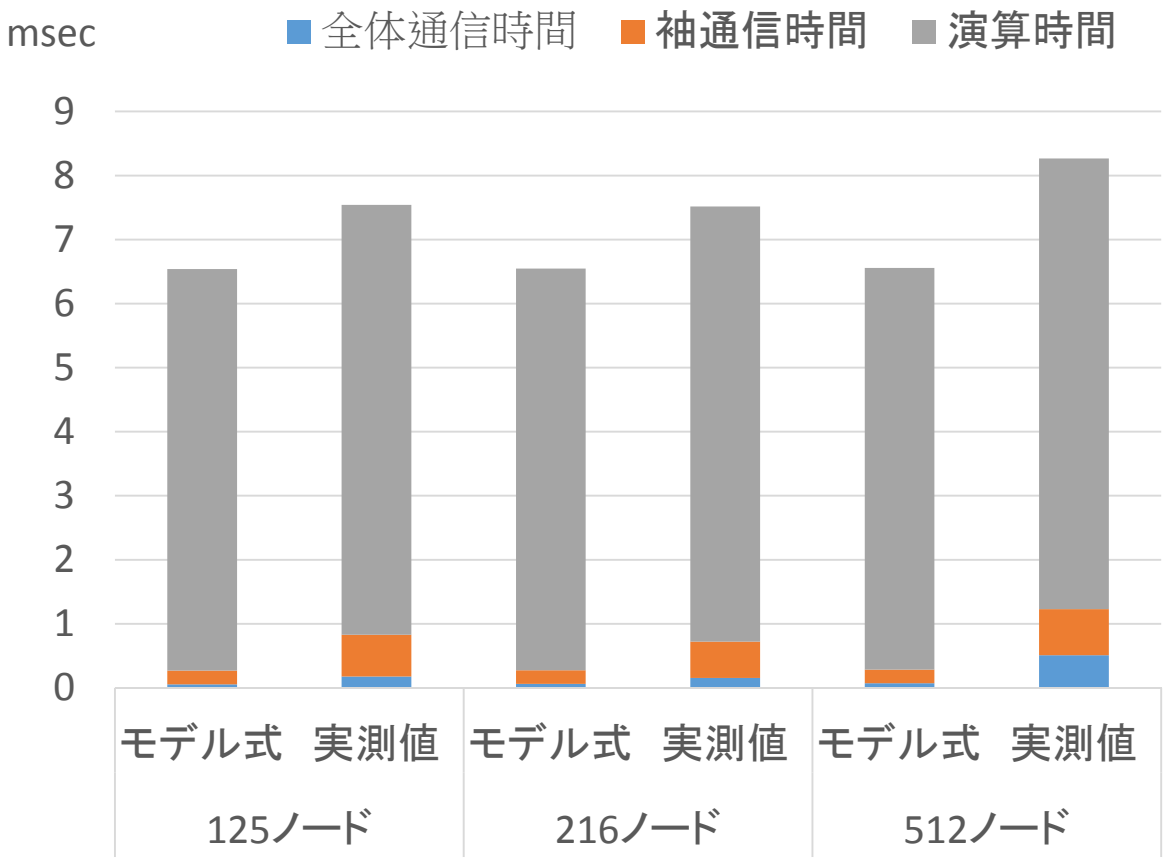
l : ネットワークレイテンシ
B_{net} : ネットワークバンド幅
$l_{allreduce}$:Allreduceの1stepのレイテンシ
s : データ長
perf_1thread :　シングルスレッドの性能(F/B値 3.6)

(4)性能モデルと実測値の比較

FIBER
fiber-miniapp.github.io
Fork me on GitHub

東京大学情報基盤センターのFX10を用いた比較

モデル式： $T = T_{計算} + T_{袖通信コピー} + T_{袖通信} + T_{全体通信}$
$= 4.9 + 1.3 + 0.2 + 8.0e\text{-}3 \log_2(Proc)$ msec



東大FX10の性能値
l : 0.92 usec
B_{net} : 4.76 GB/s
$l_{Allreduce}$: 8usec
s : 16byte
Bandwitdh : 60GB/s

袖通信部分に差がある.
原因:外部境界条件のため,外側の送受信バッファのコピーがないが,内側はあるため,その差が通信時間に上乗せされているためと予想できる.

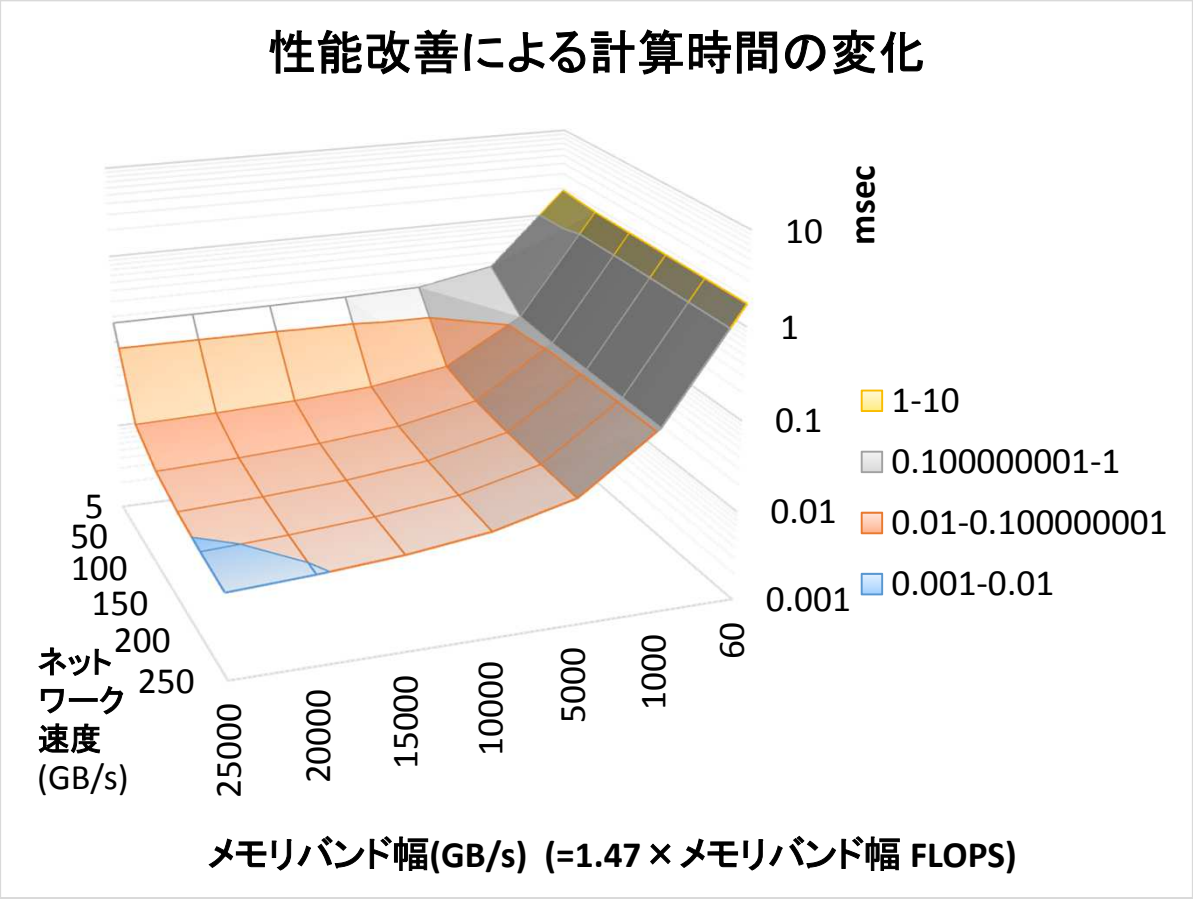
(4)要求性能の実現可能性のための条件

FIBER
fiber-miniapp.github.io
Fork me on GitHub

○計算科学ロードマップに提示された要求性能

4096^3 格子で実行時間が 0.0108 sec/step

→ SORコア計算 0.018 msec/ ループ



仮定(1) 京と同じ規模のノード数を想定
仮定(2) 超大規模系での計算性能の評価をする
仮定(3) 1ノードのサイズを(128,128,128), 32768ノードでの評価
仮定(4) $T_{計算}$ がルーフラインモデルと同性能仮定(5) $T_{袖通信コピー}$ がマルチスレッド化

FLOPS, メモリバンド幅, ネットワーク速度, ネットワークレイテンシ全ての改善が必須

レイテンシを0.05 usec と固定する.
水色の領域が実現可能の領域である.

FIBER

fiber-miniapp.github.io

Fork me on GitHub

(5) ミニアプリ
NGS Analyzer Mini

FIBER

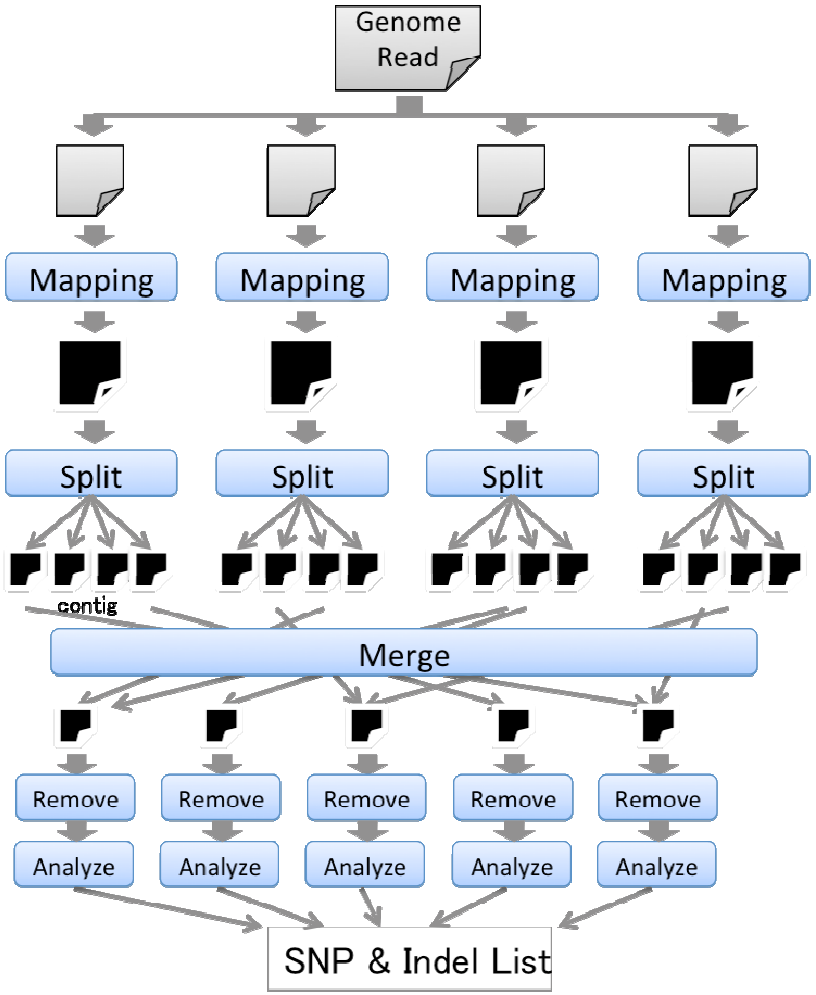
Fork me on GitHub

fiber-miniapp.github.io

(5) NGS Analyzer Mini

ヒトゲノム解析のワークフローを実行

ヒト個人間の遺伝的差異やがんゲノムの突然変異を同定する



Mapping	解析対象ゲノムを参照ゲノムにマッピング
Split	contig（連続する塩基配列の出現パターン）毎にMapping結果を分割
Merge	contig毎に結果をマージ
Remove	contig単位に重複を除去
Analyze	contig単位に変異を解析

- Merge以外のタスクは逐次プログラムでEPに実行
  - Merge前後で並列度が異なる
- Mregeでは共有ファイルシステムを介したファイルIOを行う
- ミニアプリでは以下を提供
  - ワークフロー全体を実行するMPIプログラム
  - モデル構築のために，Mapping (+split), Remove, Analyzeを個別実行するプログラム

(5) 2020年エクサにおける個人ゲノム解析の想定問題規模NGS Analyzer Mini

FIBER

Fork me on GitHub

fiber-miniapp.github.io

- 解析対象ゲノム数：200,000ゲノム
- ヒト一人あたりのゲノムデータ量：
  - 現在の100倍，およそ100TB
- 解析期間：3年，または5年
  - 1ゲノムあたり2520秒程度の時間での解析を計画

⇒ IOインテンシブな処理であり，データサイズと実行時間により問題規模が規定されているため，実行時間と，要求IO性能のモデル化を行う

(5) 京コンピュータ上での実行時間のモデル化 (1/3)

FIBER
fiber-miniapp.github.io
Fork me on GitHub

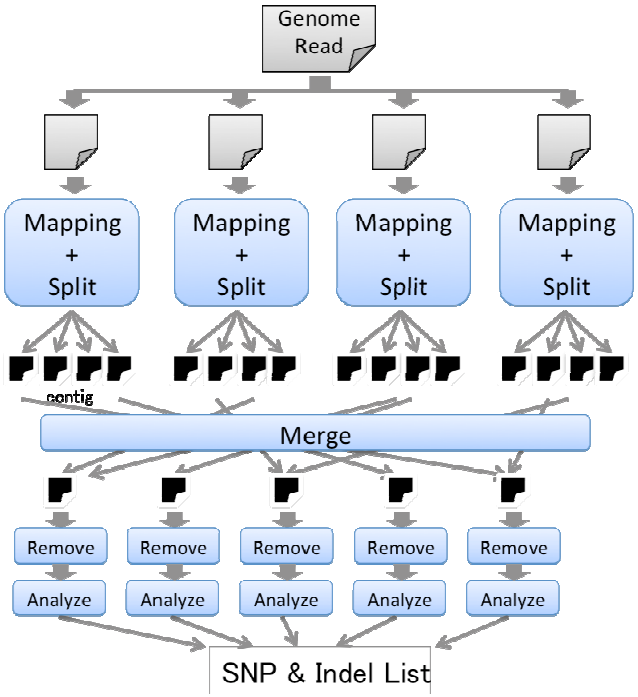
- NGS Analyzerワークフロー全体の実行時間

$$T_{Workflow} = T_{Mapping} + T_{Merge} + T_{Remove} + T_{Analyze}$$

  - 各ステップの実行時間はモデルと逐次実行プロファイル結果より算出
- Mapping, Remove, Analyzeの実行時間モデル
  - 処理は独立しているので，計算時間は実行ノード数に寄らず一定と仮定

$$T = \underbrace{T_{prof} \times (1-p)}_{\text{計算時間}} + \underbrace{\frac{N \times (Size_r + Size_w)}{IO\_Thput}}_{\text{IO時間}}$$

T_{prof}	逐次実行時の実行時間（プロファイルにて取得）
p	逐次実行時の，実行時間に占めるIOの割合（プロファイルにて取得）
N	実行ノード数
$Size_r$	Readサイズ
$Size_w$	Writeサイズ
IO_Thput	IO性能（次ページでモデル化）



FIBER
Fork me on GitHub
fiber-miniapp.github.io

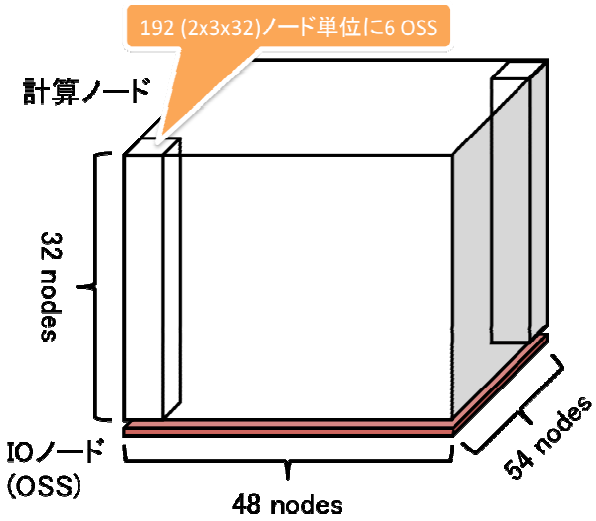
(5) 京コンピュータ上での実行時間のモデル化 (2/3)

- IO性能のモデル

$$IO\_Thput = \lceil \frac{N}{192} \rceil \times IO\_Thput_{192}$$

$$IO\_Thput_{192} = \frac{N \times (Size_r + Size_w)}{\frac{N \times Size_r}{R\_Thput} + \frac{N \times Size_w}{W\_Thput}}$$ ：1 IOグループの最大IO性能

京コンピュータのファイルシステム
- Lustreベースの並列FS，FEFSを使用
- 192ノード単位（IOグループ）に6 OSS存在
- IOグループあたり
  - Read : 2.5GB/s　(R_Thput)
  - Write : 2.0 GB/s　(W_Thput)
- 192ノード単位で，IO性能が増える



FIBER
fiber-miniapp.github.io
Fork me on GitHub

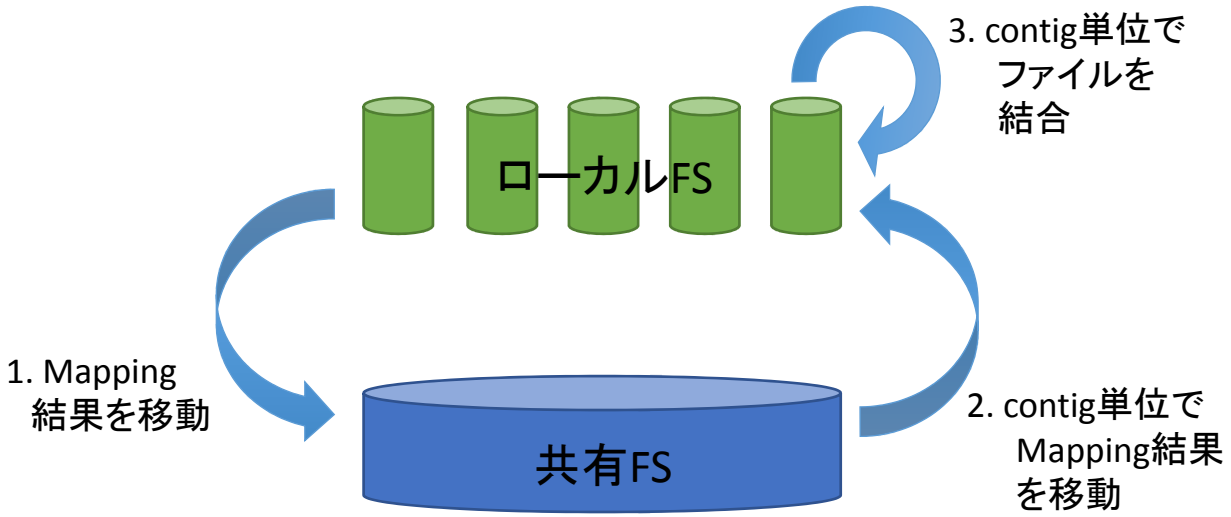
(5) 京コンピュータ上での実行時間のモデル化 (3/3)

- Mergeの実行時間のモデル

$$T_{Merge} = 3 \times \left(\frac{Size_{Total\_Mapout}}{\lceil \frac{N}{192} \rceil \times R\_Thput} + \frac{Size_{Total\_Mapout}}{\lceil \frac{N}{192} \rceil \times W\_Thput} \right)$$

$Size_{Total\_Mapout}$	Mappingの全プロセスからの出力の合計サイズ

- 3回のRead/Writeを行い，結果をマージ



FIBER
fiber-miniapp.github.io
Fork me on GitHub

(5)京コンピュータ上での実行時間モデル評価 – 設定

- 日本人男性ヒトゲノムデータの一部(72GB)を
  京コンピュータ546ノードで実行した実測時間と,
  モデルから算出した時間を比較

- パラメータ設定
  - 実行ノード数(N)
    - Mapping: 546
      - ゲノムファイル分割数
    - Remove, Analyze: 402
      - 塩基配列パターン数
  - Mappingの出力総量($Size_{Total\_Mapout}$)
    - 70.350 GB

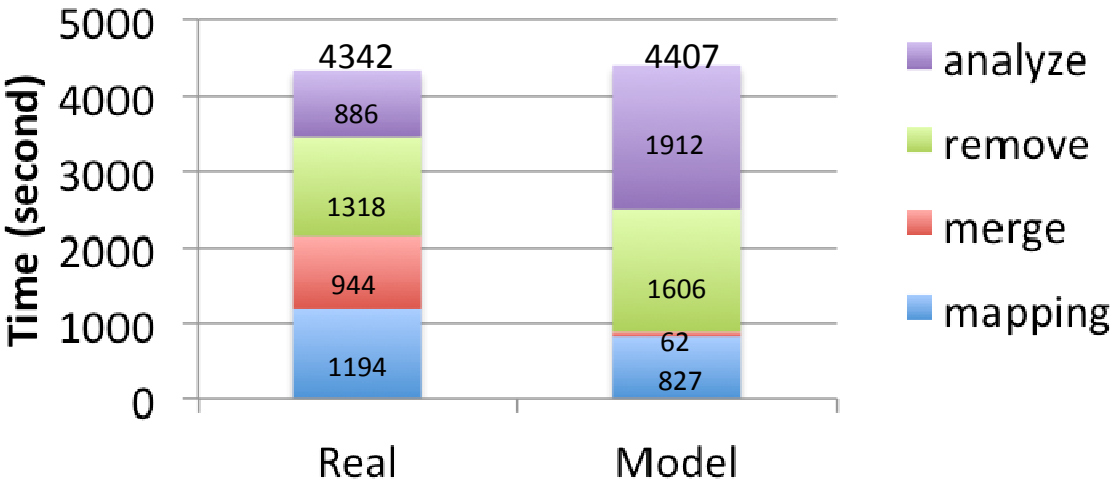
逐次実行プログラムの京でのプロファイル結果

	Mapping	Remove	Analyze
実行時間(sec) [T_{prof}]	218.29	1318.93	1127.43
IOの割合 [p]	0.04	0.01	0.03
Readサイズ (GB) [$Size_r$]	8.07	3.51	8.63
Writeサイズ (GB) [$Size_w$]	0.33	1.67	5.31

RemoveとAnalyzeではコンティグ毎に入力サイズが異なるが, ここでは最大サイズの入力を用いた

(5)京コンピュータ上での実行時間モデル評価 –結果

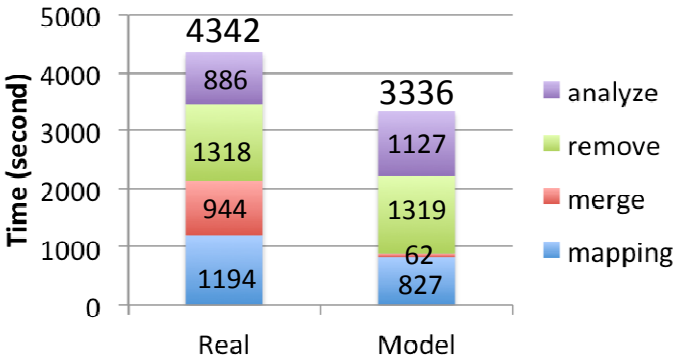
FIBER
fiber-miniapp.github.io
Fork me on GitHub



※プログラム修正したため，実測値は論文時点から計測し直している

- MergeとAnalyzeにて大きな誤差
- 誤差の原因
  - MDSへのアクセス負荷がモデル化できていないため，Mergeにて小さく見積もりすぎている可能性有り
  - Remove, Analyzeでは入力毎にサイズが大きく異なるため（167KB ～ 2.5GB），同時IOは少なく，IO競合の影響を大きく見積もりすぎている可能性有り

Remove, AnalyzeにてIO競合を考慮しないモデルでの結果



(5)エクサシステム要求性能の外挿モデル

FIBER
fiber-miniapp.github.io
Fork me on GitHub

- ワークフロー実行時の最大要求IO性能のモデル

$$IO\_Thput_{Workflow} = Max(IO\_Thput_{Mapping}, IO\_Thput_{Remove}, IO\_Thput_{Analyze})$$

  - ミニアプリにより現在提供されている，Mapping, Remove, Analyze の要求性能より求める（Merge単体のミニアプリ化は今後検討）
  - ミニアプリによる実測プロファイルと，各ステップのモデルより算出

- Mapping, Remove, Analyzeの要求IO性能モデル

$$IO\_Thput = \frac{Size}{T_{2020}} \times N$$

Size	1処理あたりの平均入出力サイズ
T_{2020}	2020年エクサにおける要求実行時間
N	並列数 （Mappingでは入力ゲノムの分割数， Remove, Analyzeでは塩基配列の出現パターンの数）

  - IOアーキテクチャは限定しない
  - Nプロセス分並列に処理を行う

(5)エクサシステム要求性能の評価 (1/2)

FIBER
fiber-miniapp.github.io
Fork me on GitHub

- 東京大学FX10にて取得した逐次プログラムのプロファイルをもとにモデルから算出
- 入力サイズ(Size):現在の100倍
  - Mapping: 100TB, Remove: 48GB, Analyze: 381GB
- 並列度(N)
  - Mapping: 833,334
    - 100TBの入力を25万塩基配列毎に分割
  - Remove, Analyze: 7,156
    - 定義されている最大パターン数
- 実行時間(T_{2020})
  - プロファイル結果を基に,その比率に従い均等分割
  - Mapping: 1,050.27 sec
  - Remove: 477.51 sec
  - Analyze: 994.19 sec

逐次実行プログラムのFX10でのプロファイル結果

	Mapping	Remove	Analyze
実行時間(sec) [T_{prof}]	242.19	110.11	228.81
IOの割合 [p]	0.04	0.03	0.10
Readサイズ (GB) [$Size_r$]	8.07	0.32	3.34
Writeサイズ (GB) [$Size_w$]	0.36	0.16	0.47

RemoveとAnalyzeではコンティグ毎に入力サイズが異なるが,ここでは平均サイズの入力を用いた

(5)エクサシステム要求性能の評価 (2/2)

FIBER
fiber-miniapp.github.io
Fork me on GitHub

- 要求IO性能 : 167.22 TB/s
  - Mapping: 167.22, Remove: 23.98, Analyze: 27.42 TB/s

- 2018年の目標性能 10TB/s (20MW消費電力時)より遥かに大きく，アルゴリズム・実装方法等の変更が必須
  - フルアプリ，ミニアプリとも既存のOSS群より構成されており，タスク間のインターフェースはファイル
    => 無駄なファイルIOが発生
  - IOを減らす実装・実行をすることで，大幅な改善が見込める
    - 例）Mappingにおいて，①参照ゲノムの読み込み回数を削減し，
      ②処理単位を大きく（25万配列 => 50万配列）することで，1/5の 31.69TB/s に要求性能を削減可能

(1)まとめと今後の課題

FIBER
fiber-miniapp.github.io
Fork me on GitHub

〇まとめ

1. Fiberミニアプリの概要について説明
2. Marble Miniを例にミニアプリの作業方法について説明
3. 現在整備がほぼ終了しているアプリ (1) Marble Mini (2) CCS QCD (3)FFVC Mini (4)NGS Analyzer Mini の4つのアプリケーションの概要説明と性能評価モデルの提案
4. 性能評価モデルから計算科学ロードマップに提示された要求性能の実現可能性について議論

〇今後の課題

1. 未整備のミニアプリ群の整備と性能評価モデルの構築
   (1) Modylas , (2) ALPS/looper, (3) RSGDX, (4) FFB, (5) CONQUEST
2. 性能評価モデルの精査
3. 各アプリのチューニング
4. コプロセッサに向けた，現ミニアプリのOpenACCなどを用いた拡張