Vol.2014-HPC-143 No.26
2014/3/4

レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究のアプリケーションの異機種環境での評価　～メニーコア環境を中心に～

片桐孝洋†　大島聡史†　中島研吾†　米村崇††　熊洞宏樹††
樋口清隆††　橋本昌人††　高山 恒一††　藤堂眞治†3，岩田 潤一†4
内田和之†4，佐藤正樹†5，羽角博康†5，黒木聖夫†6
安達斉†7，江口義之†7

本報告では，レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究におけるコデザインで用いるアプリケーションについて，コード最適化による性能チューニングを行った結果を報告する．また，富士通 PRIMEHPC FX10，日立 SR16000，Intel Ivy Bridge，および Intel Xeon Phi を用いた異機種環境で性能評価を行った結果を紹介する．

1. はじめに

本報告では，レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究（以降，単に調査研究とよぶ）におけるターゲットアプリケーションにおいて，概念設計マシンでの性能予測を妥当なものにするための性能最適化について述べる．

本報告の構成は以下のとおりである．2 章で調査研究の目的とターゲットアプリケーションを説明する．3 章では，富士通 PRIMEHPC FX10 を用いた性能チューニングの内容と効果を紹介する．4 章では，Intel Xeon Phi を用いた性能チューニングの内容と効果を紹介する．5 章では，FX10，Ivy Bridge，Xeon Phi と日立 SR16000 を用いた異機種環境で行った性能評価の結果を紹介する．

2. 調査研究の目的とターゲットアプリケーション

2.1 将来の HPCI システムのあり方の調査研究

本調査研究は，第 4 期科学技術基本計画（平成 23 年 8 月 19 日閣議決定）で掲げられた国家存立の基盤としての世界最高水準のハイパフォーマンス・コンピューティング技術の強化，及び科学技術基盤の充実強化に向けた重要な取り組みの一つとして，HPC 技術等の HPCI システムの高度化に必要な技術的知見を獲得することを目的とし，平成 24 年度，平成 25 年度に調査研究を実施するものである[1].

2.2 レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究

本調査研究は，東京大学を中心とし，九州大学，富士通，日立製作所，日本電気による調査研究である[1]．2018 年頃設置可能な並列システムを，汎用型プロセッサからのアプローチでフィージビリティ・スタディ（FS）を行う．アプリケーション，システムソフトウェア，アーキテクチャの co-design を行う．システムソフトウェアスタック共通化 (From PC cluster to high-end machines)を行う．

本報告はこのうち，アプリケーション性能予測に関する検討事項に相当する．

2.3 本調査研究における進め方

「今後の HPCI 技術開発に関する報告書」[3]，および，「計算科学ロードマップ白書」[4]を尊重し，京および FX10 におけるアプリケーション並列性能および I/O 性能，耐故障性および運用・保守の観点で課題を精査し，概念設計に反映する．進め方の概要を図 1 に示す．

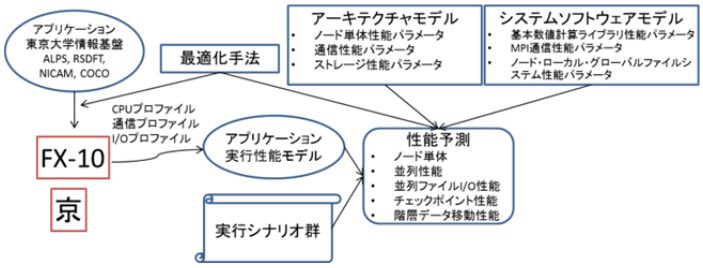


図 1　本調査研究における調査研究の進め方

本報告は，図 1 における性能予測のための手法と最適化方式の調査研究に関連する．

2.4 利用シナリオ

利用シナリオ（図 1 における「実行シナリオ群」）とは，各アプリケーションの実行において，特徴的なジョブの実行形態のことである．主に以下のアンサンブル型を想定し，入出力ファイルなどの I/O 性能を含め，システム全体の設計に反映させる．

- **アンサンブル型：**全系の 1/10～1/100 の資源を利用する 1 ジョブに対し，複数ジョブを同時実行して全資源を使い切る形態．この形態では，複数同時のファイル入力，および複数同時のファイル出力が起こる．

† 東京大学 情報基盤センター スーパーコンピューティング研究部門
†† 日立製作所 情報・通信システム社
†3 東京大学 物性研究所
†4 東京大学 大学院工学系研究科
†5 東京大学 大気海洋研究所
†6 海洋研究開発機構
†7 日立ソリューションズ東日本

2.5 性能予測手法

図 1 を進めるに当たり，ターゲットアプリケーションにおける概念設計中の計算機性能の実行時間を予測するため，以下の手法をとる．

1. **ホットスポット同定：**基本プロファイラ（主要な関数やループの実行時間が取得可能な性能プロファイラ）を用いて，複数のホットスポット（ループレベル）を同定する．その後，全体性能の予測をホットスポットのみで行う．
   - ホットスポットの部品化を行う．できるだけ，採用されている数理アルゴリズム（支配方程式，離散化方法）とホットスポットの対応がわかるようにする．
2. **ホットスポット分離：**計算部分，通信部分，I/O 部分のホットスポットを基のソースコードから分離する．
   - 計算部分：演算カーネルと呼ぶ．
   - 通信部分：通信カーネルと呼ぶ．
   - I/O 部分： I/O カーネルと呼ぶ．
3. **通信パターン確認：**プロファイラによる可視化ツールや対象コードを解析することで，通信パターンを確認する．
4. **詳細プロファイルと分析：**詳細プロファイラ（対象のループにおけるハードウェア上の性能情報が取得可能な性能プロファイラ）を用い，ホットスポットごとにハードウェア性能情報を取得して分析する．
   - 演算カーネルにおける，演算効率／命令発行量／キャッシュ利用効率，など．
   - 通信カーネルの，通信回数／量／通信待ち時間，など．
   - I/O カーネルの，データ読み書き，量／頻度，など．
5. **ベンチマーク化：**ホットスポットのみで動作するようにコードを再構成する．
   - マシン特化の書き方，および，汎用的な書き方，の２種を区別する．
   - 演算カーネル，通信カーネル，I/O カーネルの分類をする．
6. **詳細モデル化：**ハードウェア因子による実行時間の予測ができるようにする．

本調査研究では，詳細モデル化を行うに当たり，FX10 で提供される性能プロファイラを用いる．このことで，演算および通信カーネルを抽出できる．また，現存する計算機でのハードウェア因子について，プロファイラを通じて取得ができる．この情報をもとに，現在設計中の計算機での性能予測が可能となる．

2.6 ターゲットアプリケーションの特徴

前回までの報告で，以下の４アプリについての性能チューニングと詳細を報告した[4][5].

(1) **ALPS (Algorithms and Libraries for Physics Simulations):**新機能を持った強相関・磁性材料の物性予測・解明のシミュレーションである．虚時間経路積分にもとづく量子モンテカルロ法と厳密対角化を利用している．

(2) **RSDFT (Real-Space Density-Functional Theory):** Si ナノワイヤ等，次世代デバイスの根幹材料の量子力学的第一原理シミュレーションである．実空間差分法を利用している．

(3) **NICAM (Nonhydrostatic ICosahedral Atmospheric Model):**長期天気予報の実現，温暖化時の台風・豪雨等の予測のシミュレーションである．正 20 面体分割格子非静力学大気モデルを採用し，水平格子数 km で全球を覆い，積雲群の挙動までを直接シミュレーションする．

(4) **COCO (CCSR Ocean COmponent Model):**海況変動予測，水産環境予測のシミュレーションである．外洋から沿岸域までの海洋現象を高精度に再現し，気候変動下での海洋変動を詳細にシミュレーションする．

平成 25 年度から，理化学研究所のアプリ FS と連携し評価アプリケーションを拡張した．以下の表 1 に，評価対象のアプリケーションの定性的な特徴を記述する．

表 1　アプリケーションの特性

アプリ名	演算パターン	通信パターン
CCS-QCD	行列 - ベクトル積（連続アクセス）	隣接通信
Modylas	カットオフ付き相互計算，FMM	隣接通信，近傍通信
流体（非構造格子）	行列 - ベクトル積 (element-by-element)	隣接通信
NGS Analyzer	リード列の SA (Sequence Alignment) 座標計算、ファイル圧縮（zlib）	なし（ファイル I/O が中心）
NURON K+	擬一次元系の電位変化演算（疎行列連立微分方程式の求解）	1 対 1 通信（双方向、１方向）
Seism 3D	有限差分法演算	隣接通信
LETKF (+NICAM)	モード解析（固有値計算）	集団通信（Gather, Scatter）
GENESIS	カットオフ付き相互計算，FFT	隣接通信(x,y,z 方向，6 回/Step, isend, irecv)．FFT 通信（AlltoAll, Comm_split 版）
CONQUEST	ブロック疎行列ベクトル積	近傍通信（データ依存）

NT Chem	$O(n^3)$，行列 - 行列積	$O(n^3)$，1 対 1 通信を(全プロセス)/2
CA アルゴリズム評価用ベンチマーク	GeoFEM-Cube/CG version.2.00)，ss7p(s-step 7point stencil)，FDTD	ー（通信なし）

2.7 ターゲットアプリケーションの特徴

本稿では，性能チューニングの効果があったアプリケーションとカーネルをいくつか抽出して報告する．表 2 に，取り上げるアプリケーションの詳細を記述する．

表 2　本稿で取り扱う演算カーネル

アプリ名	カーネル名（プログラム上の位置）		実測 B/F	説明
NICAM	NICAM_02_mod_oprt (mod_oprt.oprt_divergence._PRL_1_)		1.08	力学過程に関する処理部分 5 種
	NICAM_04_mod_oprt3d (mod_oprt3d.oprt3d_divdamp._PRL_4_)		2.36	
	NICAM_05_mod_src (mod_src.src_flux_convergence._PRL_8_)		6.11	
	NICAM_07_mod_src (mod_src.src_flux_convergence._PRL_17_)		18.09	
	NICAM_08_mod_oprt (mod_oprt.oprt_divergence2_rev._PRL_18_)		6.56	
	NICAM_09_mod_mp_nsw6 (mod_mp_nsw6.mp_nsw6._PRL_5_)		0.28	物理過程に関する処理部分 1 種
COCO	COCO_flxomp2 (flxtrc._OMP_2_)		2.32	移流項に関する処理部分 3 種
	COCO_flxomp3 (flxtrc._OMP_3_)		3.37	
	COCO_flxomp5 (flxtrc._OMP_5_)		2.27	
CCS-QCD	clover		2.06	Clover 部，および，BiCGStab 部の 2 種
	BiCGStab	全体	1.70	
		行列積以外	5.29	
		行列積	1.20	

表 2 の実測 B/F とは，富士通のプロファイラから得られた，FX10 の 1 ノード当たりのメモリアクセス性能[GB/sec.]と FLOPS 値から計算したものである．実測の Byte per FLOPS 値 (B/F 値)といえる．以降で示す実測 B/F は，FX10 での実測値にもとづく値であり，FX10 以外の対象計算機での実測値ではない点に注意する．

2.8 本評価で利用する計算機

本稿で扱う計算機の詳細を以下にまとめる．

(1) FX10 スーパーコンピュータシステム（FX10）

- OS：専用 OS（XTCOS）
- 計算ノード数：4800
- 1 ノード理論性能：236.5GFLOPS（FX10 比：1.00）
- 総理論演算性能：1.13PFLOPS
- 1 ノード記憶容量：32GB，総主記憶容量：150TB
- インターコネクト：6 次元メッシュ/トーラス
- ノード間ネットワーク性能：5GB/s×双方向
- コンパイラ：富士通 Fortran コンパイラ．Version 1.2.1 P-id: T01641-02.

(2) HITACHI SR16000 モデル M1（SR16000）

- OS：AIX 7.1
- 計算ノード数：56
- 1 ノード理論性能：980.48GFLOPS（FX10 比：0.24）
- 総理論演算性能：54.906TFLOPS
- 1 ノード記憶容量：200 GB，総主記憶容量：11200GB
- ノード間ネットワーク性能：96GB/s（単方向）×双方向
- コンパイラ：日立 最適化 FORTRAN90 03-02-/A

(3)Xeon クラスタ(Ivy Bridge)

- OS：Red Hat Enterprise Linux Server release 6.2
- 計算ノード数：32（使用可能 14）
- CPU：Intel Xeon E5-2670 V2 @ 2.50GHz，2 ソケット×10 コア
- ハイパースレッディング：オン
- 1 ノード理論性能：400 GFLOPS（FX10 比：0.59）
- 1 ノード記憶容量：64 GB
- インターコネクト：InfiniBand
- コンパイラ：Intel Fortran version 13.0.0

(4) Intel Xeon Phi コプロセッサ（Xeon Phi）

- CPU：Xeon Phi 5110P（B1 stepping）1.053 GHz，60 core
- 記憶容量：8 GB
- 理論ピーク性能：1 TFLOPS（= 1.053 GHz x 16 FLOPS x 60 core）（FX10 比：0.23）
- Xeon クラスタの各ノードに 1 枚ずつ接続
- コンパイラ：Intel Fortran version 13.0.0

上記の FX10 比とは，(FX10 での理論 FLOPS 値)/(当該計算機での理論 FLOPS 値)である．

3. FX10 におけるチューニング内容と効果

3.1 概要

ここでは，東京大学情報基盤センターに設置された FX10 を用いて，CCS-QCD の演算カーネルのチューニングを行った結果（2014 年 1 月現在）について事例を紹介する．

3.2 CCS-QCD の演算カーネルチューニング

3.2.1 OMP プロセスの負荷均等化

●OMP プロセスの負荷均等化

CCS-QCD の問題サイズと実行形態の中には，32 x 32 x 32 の格子を 4 x 4 x 4 MPI 分割し実行するものがある．このとき，OMP でのスレッド並列では，ナイーブな実装の場合最外ループをスレッド並列化するため，最大のスレッド並列性が 32/4=8 となる．したがって FX10 のように，1 ノード当たり 16 コアを使い切るスレッド実行ができない．

そこで，各 MPI プロセスは Nx x Ny x Nz = 8 x 8 x 8 を計算するが，16OMP の並列性を使い切るため，最外ループと第 2 ループを融合してループ長 8x8=64 にしてから分割して計算する．図 2 に，修正を行ったコードを記載する．

図 3 は，図 2 のコード修正を行った効果[GFLOPS]を示している．コード修正を行うことで、BiCG 部については 1.11 倍，Clover 部については 2.53 倍の性能向上を得た．

3.3 CCS-QCD の通信チューニング

CCS-QCD の MPI プロセスについてのデータ分割と通信パターンは以下である．データ分割を図 4 に示す．

(1)データ分割

- 4 次元空間データを，xyz の空間 3 次元で分割し，MPI プロセスにマッピングする．

(2)通信パターン

- 隣接空間を担当する MPI プロセス間で，境界データを交換する．
- xyz 格子の端は，周期境界となるように，MPI プロセス間で境界データを交換する．
- 1 要素の MPI_allreduce がある．

図 4 から，CCS-QCD の主通信は，隣接空間を担当する MPI プロセス間で境界データを交換となる．この通信は，オリジナルのブロッキング通信に対して，ノンブロッキング通信に書き換え可能である．さらに，通信のオーバラッピングができる．さらに，通信に必要なコピーと通信のオーバラッピングができる．図 5 に，これらの通信実装の概要を示す．

表 3 に，図 5 の通信実装について評価を行った結果を載せる．なお，表 3 の Org. とはオリジナル実装，Ovlp. とはオーバーラップ実装，Copy + Ovlp. とはコピーとオーバーラップの併用実装の実行時間を意味している．また，通信・コピーオーバーラップ時の通信時間は，通信のみ行っている時間である．

表 3 よりコピーとオーバーラップの併用により，CLASS2 以上の問題で，オリジナルの実行に対し 10%～15%ほど通信時間の削減ができる．

```
!$OMP PARALLEL PRIVATE(ics,jcs)
!$OMP DO
  do jcs=1,CLSP/2
  do ics=1,CLSP/2
    zunit(ics,jcs)=(0.0d0,0.0d0)
  enddo
  enddo
!$OMP END DO
!$OMP DO
  do ics=1,CLSP/2
    zunit(ics,ics)=(1.0d0,0.0d0)
  enddo
!$OMP END DO
!$OMP END PARALLEL
```

(a) 変更前のコード

```
!$OMP PARALLEL PRIVATE(ics,jcs)
!$OMP DO COLLAPSE(2)
  do jcs=1,CLSP/2
  do ics=1,CLSP/2
    if (ics .eq. jcs) then
     zunit(ics,jcs)=(1.0d0,0.0d0)
    else
     zunit(ics,jcs)=(0.0d0,0.0d0)
    end if
  enddo
  enddo
!$OMP END DO
!$OMP END PARALLEL
```

(b) 変更後のコード

図 2 CCS-QCD における OMP 並列性の向上のためのコード修正

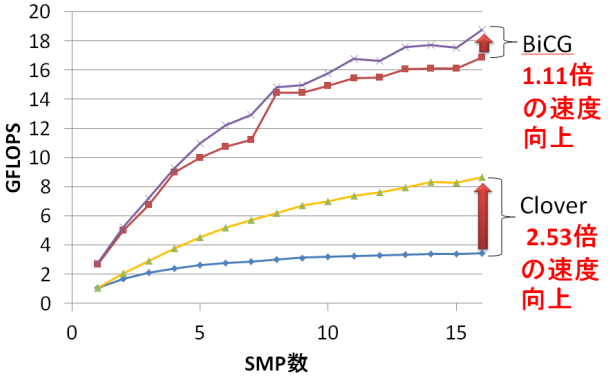


図 3 CCS-QCD におけるコード修正の効果(FX10)

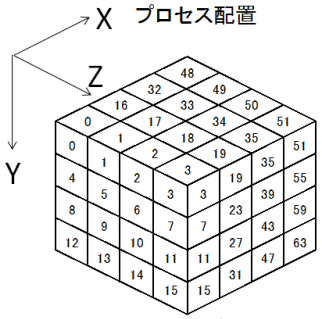


図 4　CCS-QCD のデータ分割と MPI プロセスマッピング

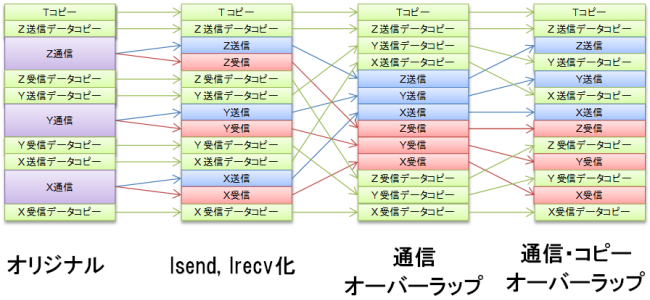


図 5　CCS-QCD の通信実装

表 3　通信チューニングの効果
（FX10，N ノード，16 スレッド／ノードでの実行．
ただし，N = X x Y x Z x T.）

問題クラス	問題規模 分割数 (X×Y×Z×T)	通信実装	演算 [msec/Iter]	通信 [msec/Iter]	合計 [msec/Iter]
CLASS1	8×8×8×32 (1×1×1×1)	Org.	1.1	0.0	1.1
		Ovlp.	1.1	0.0	1.1
		Copy+Ovlp	1.1	0.0	1.1
CLASS2	32×32×32×32 (4×4×4×1)	Org.	1.5	0.8	2.3
		Ovlp.	1.5	0.5	2.0
		Copy+Ovlp	1.5	0.5	2.0
CLASS3	64×64×64×32 (8×8×8×1)	Org.	1.5	0.8	2.3
		Ovlp.	1.5	0.5	2.0
		Copy+Ovlp	1.6	0.5	2.1
CLASS4 縮小	32×32×32×160 (4×4×4×1)	Org.	6.5	2.9	9.4
		Ovlp.	7.0	1.6	8.6
		Copy+Ovlp	7.0	1.6	8.5
CLASS5 縮小	32×32×32×256 (4×4×4×1)	Org.	10.3	4.5	14.7
		Ovlp.	10.9	2.5	13.4
		Copy+Ovlp	11.1	2.3	13.4

4. Xeon Phi におけるチューニング内容と効果

4.1 概要

ここでは，Xeon Phi コプロセッサでのチューニング内容について紹介する．コンパイラオプションのチューニングの概要を以下に示す．

- オプション　（説明）
- -opt-threads-per-core　（コアあたりのスレッド数の指定．）
- -opt-assume-safe-padding　（動的メモリがパディングされていることを仮定し処理を行う．）
- -no-vec　（ベクトル化を無効にする．）
- -O3　（最適化レベルに 3 を指定．デフォルトの最適化レベルは 2．）
- -opt-streaming-stores always　（最適化のためのストリーミング・ストアの生成を有効にする．）
- -opt-streaming-cache-evict=0　（ストリーミング・ロード／ストアが使用されたとき，キャッシュ退避命令を生成しないようにする．）

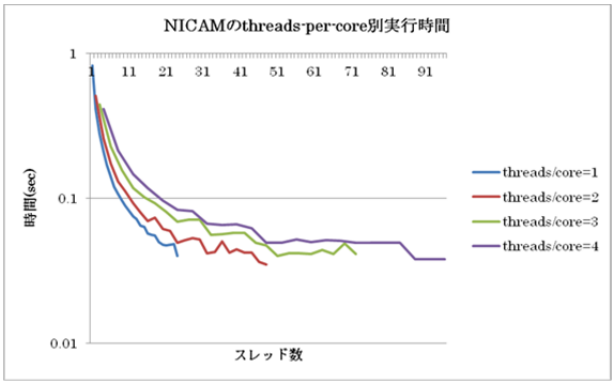
なお，-opt-threads-per-core による最適化では，{ 1 | 2 | 3 | 4 }について評価する．そのため実行スレッド数は，利用コア数×threads-per-core 数になる．

4.2 NICAM

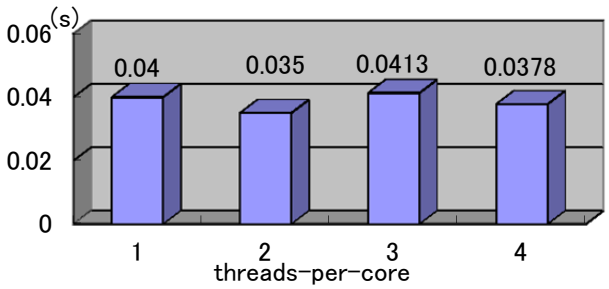
4.2.1 ．-opt-threads-per-core による最適化

図 6～図 11 に，NICAM の演算カーネルについて，-opt-threads-per-core を変化させた場合の実行結果について載せる．

●NICAM_02_mod_oprt (実測 B/F=1.08)



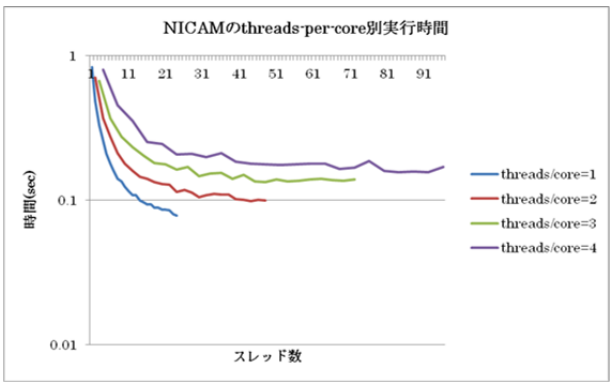
(a)　実行時間の変化



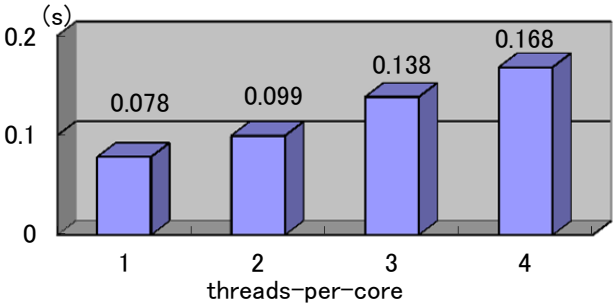
(b)　各 threads-per-core での最速の実行時間

図 6　NICAM_02_mod_oprt の性能（Xeon Phi）

●NICAM_04_mod_oprt3d (実測 B/F=2.36)



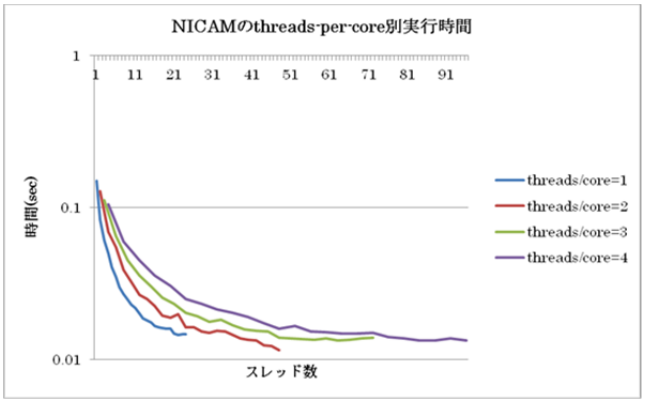
(a)実行時間の変化



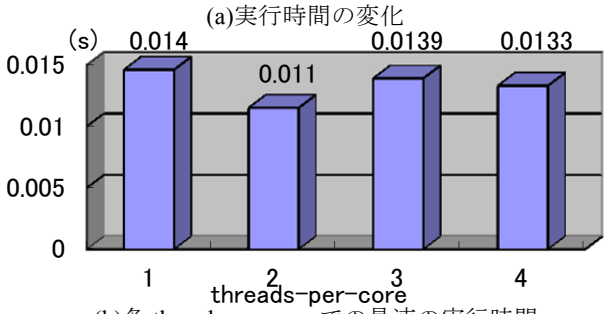
(b)各 threads-per-core での最速の実行時間

図 7　NICAM_04_mod_oprt3d の性能（Xeon Phi）

●NICAM_05_mod_src (実測 B/F=6.11)



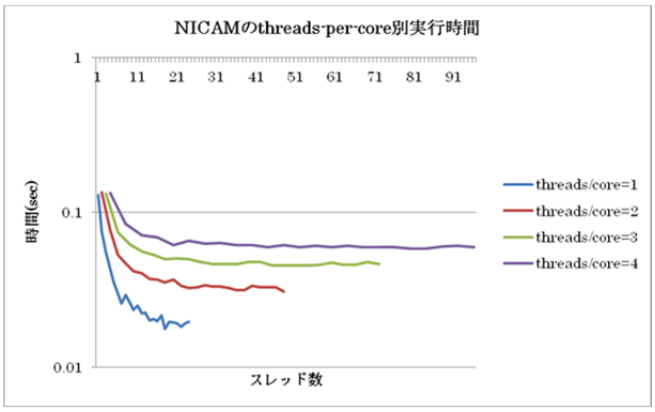
(a)実行時間の変化



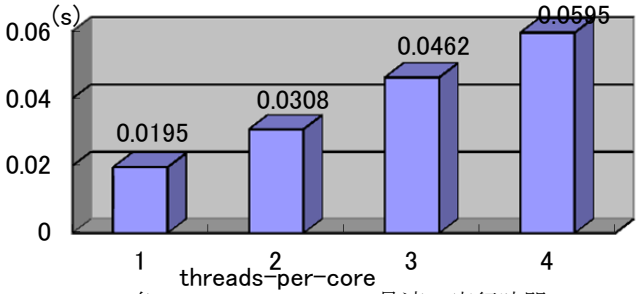
(b)各 threads-per-core での最速の実行時間

図 8　NICAM_05_mod_src の性能（Xeon Phi）

●NICAM_07_mod_src (実測 B/F=18.09)



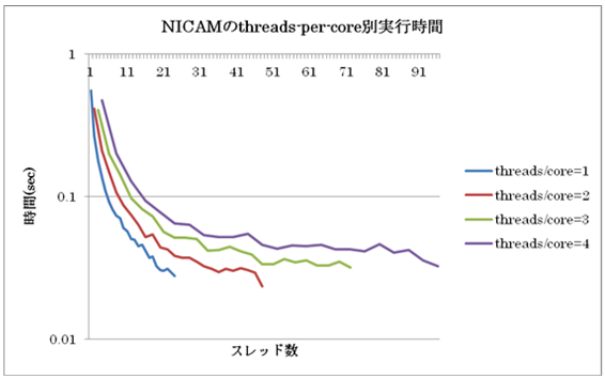
(a)実行時間の変化



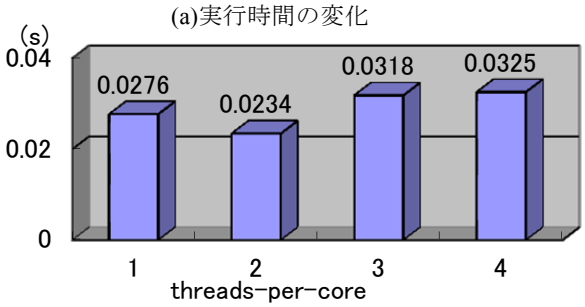
(b)各 threads-per-core での最速の実行時間

図 9　NICAM_07_mod_src の性能（Xeon Phi）

●NICAM_08_mod_oprt (実測 B/F=6.56)



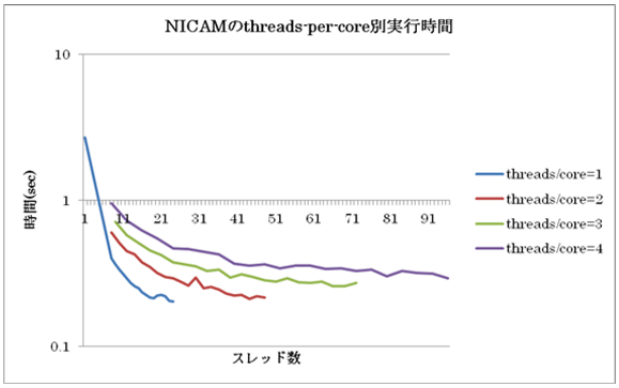
(a)実行時間の変化



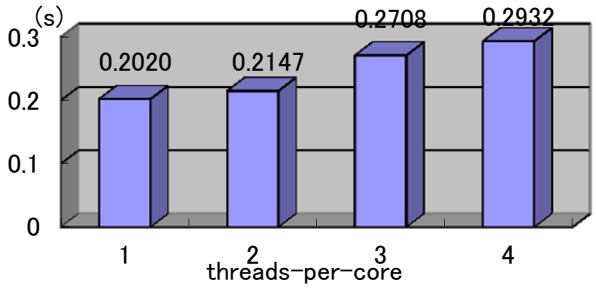
(b)各 threads-per-core での最速の実行時間

図 10　NICAM_08_mod_oprt の性能（Xeon Phi）

●NICAM_09_mod_mp_nsw6 (実測 B/F=0.28)



(a)実行時間の変化



(b)各 threads-per-core での最速の実行時間

図 11　NICAM_09_mod_mp_nsw6 の性能（Xeon Phi）

以上の図 6～図 11 から，いくつか例外があるが，実測 B/F が 6 以上の演算カーネルで，threads-per-core=2 が threads-per-core=1 よりも高速になる傾向が見受けられる．

なお，NICAM_07_mod_src は実測 B/F=18.09 と大きいが threads-per-core=1 が高速である．この理由の一つとして，この演算カーネルはコピーを主体としており，演算がほとんどない．したがって，例外的な処理といえる．

4.2.2 NICAM における最適化オプションの調査

-opt-threads-per-core 測定後の最適化オプションについて調査した．以下に詳細を載せる．

- NICAM_04_mod_oprt3d, NICAM_07_mod_src, NICAM_09_mod_mp_nsw6

```
-fpp -openmp -openmp_report2 -DUSE_TIMER
-DOPT_TUNING=0 -opt-threads-per-core=1
```

- NICAM_02_mod_oprt, NICAM_05_mod_src, NICAM_08_mod_oprt

```
-fpp -openmp -openmp_report2 -DUSE_TIMER
-DOPT_TUNING=0 -opt-threads-per-core=2
```

以上の最適化オプションを base options とし，以下に示す opt2～opt6 までの最適化オプションを変更し，主要計算部の実行時間を測定した．

- 検証した最適化オプション

```
*opt1: base options
*opt2: (base options) -opt-assume-safe-padding
*opt3: (base options) -no-vec
*opt4: (base options) -O3
*opt5: (base options) -opt-streaming-stores always
*opt6: (base options) -opt-streaming-cache-evict=0
```

base options での実行時間を 1 として正規化した時の，主要計算部の実行時間比率を図 12 に示す．

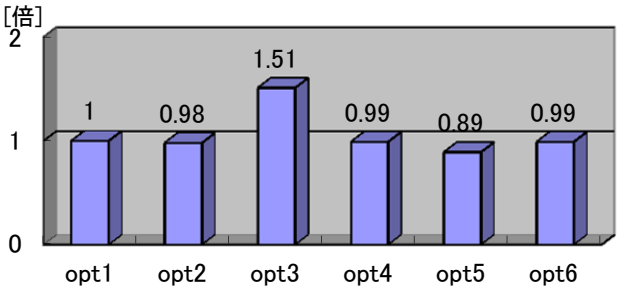


図 12　NICAM_05_mod_src の
コンパイラオプションの影響（Xeon Phi）

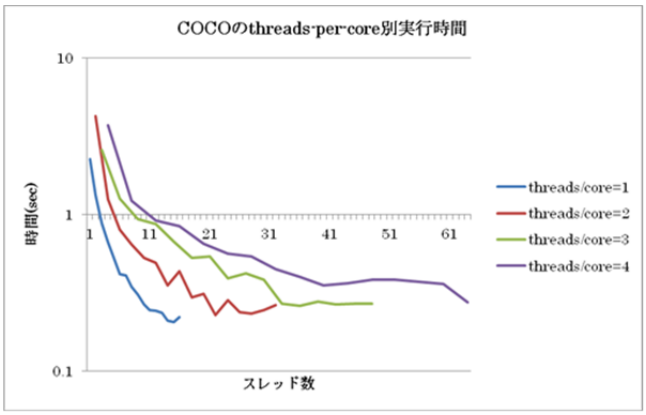
図 12 より，NICAM_05_mod_src では 11% ほど，-opt-streaming-stores always による高速化の効果が確認できる．なお，これ以外の NICAM の演算カーネルでは，-O3 が NICAM_02_mod_oprt，NICAM_04_mod_oprt3d，および NICAM_05_mod_src に対し 3%程度の高速化が観測された．

4.3 COCO の最適化

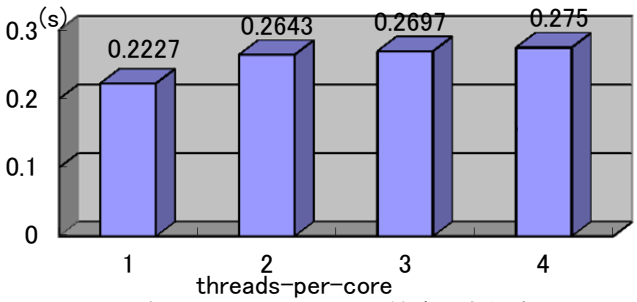
4.3.1 -opt-threads-per-core による最適化

図 13～図 15 に，NICAM の演算カーネルについて，-opt-threads-per-core を変化させた場合の実行結果について載せる．

●COCO_flxomp2 (実測 B/F=2.32)



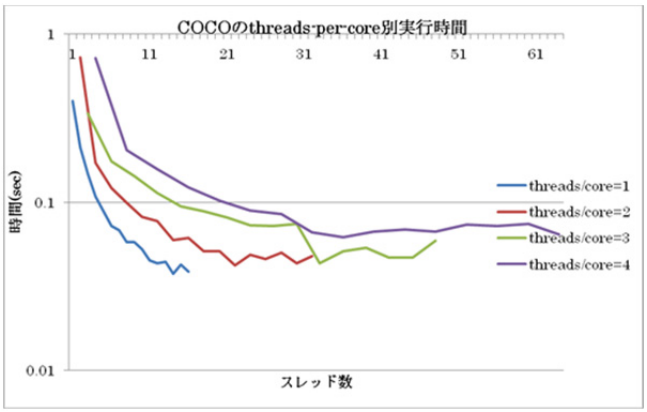
(a)実行時間の変化



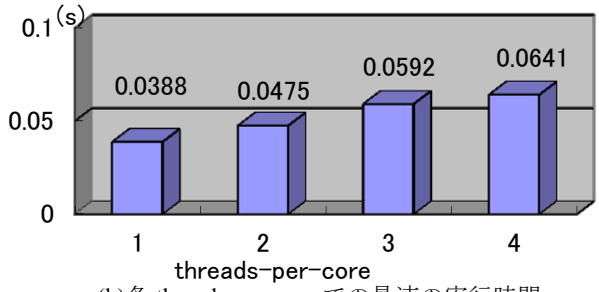
(b)各 threads-per-core での最速の実行時間

図 13　COCO_flxomp2 の性能（Xeon Phi）

●COCO_flxomp3 (実測 B/F=3.37)



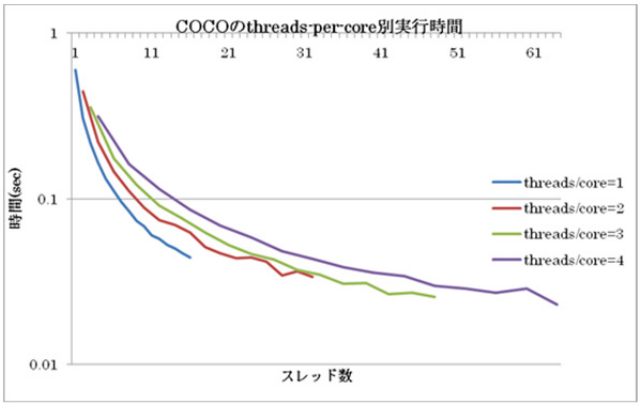
(a)実行時間の変化



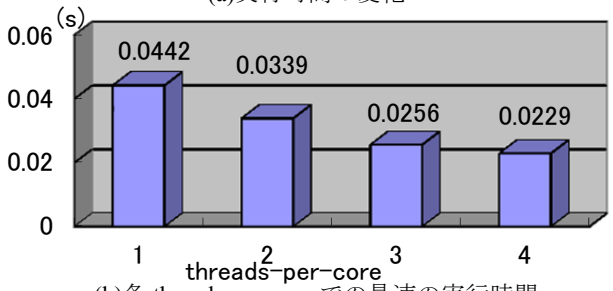
(b)各 threads-per-core での最速の実行時間

図 14　COCO_flxomp3 の性能（Xeon Phi）

●COCO_flxomp5 (実測 B/F=2.27)



(a)実行時間の変化



(b)各 threads-per-core での最速の実行時間

図 15　COCO_flxomp5 の性能（Xeon Phi）

図 13～図 15 から，COCO_flxomp5 (実測 B/F=2.27)のみ，threads-per-core=4 が高速となった．演算カーネル COCO_flxomp5 のみ，キャッシュブロック化がなされている．そのため，キャッシュブロック化の影響が考えられる．

4.3.2 COCO における最適化オプションの調査

-opt-threads-per-core 測定後の最適化オプションについて調査した．以下に詳細を載せる．

- COCO_flxomp2，COCO_flxomp3

```
-fpp -openmp -openmp_report2 -DUSE_TIMER
-DOPT_TUNING=0 -opt-threads-per-core=1
```

- COCO_flxomp5

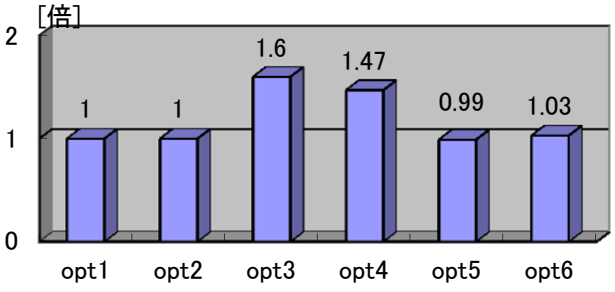
```
-fpp -openmp -openmp_report2 -DUSE_TIMER
-DOPT_TUNING=0 -opt-threads-per-core=4
```

以上の最適化オプションを base options とし，以下に示す opt2～opt6 までの最適化オプションを変更し，主要計算部の実行時間を測定した．

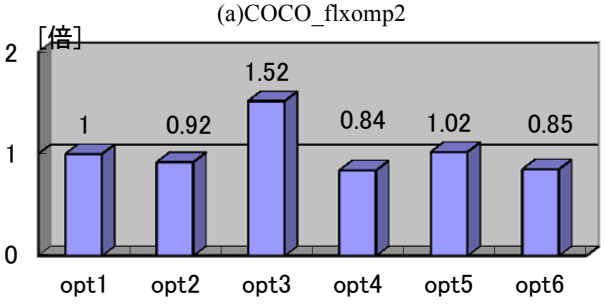
- 検証した最適化オプション

*opt1: base options
*opt2: (base options) -opt-assume-safe-padding
*opt3: (base options) -no-vec
*opt4: (base options) -O3
*opt5: (base options) -opt-streaming-stores always
*opt6: (base options) -opt-streaming-cache-evict=0

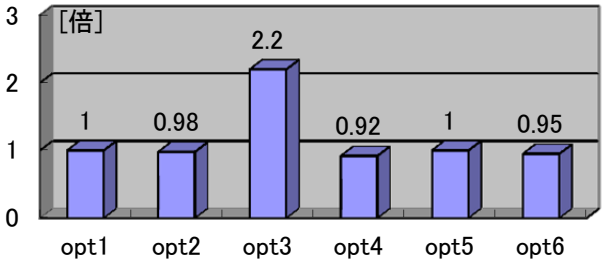
base options の実行時間を 1 として正規化した時の主要計算部の実行時間比率を図 16 に示す．



(a)COCO_flxomp2



(b) COCO_flxomp3



(c) COCO_flxomp5

図 16　CCS-QCD のコンパイラオプションの影響（Xeon Phi）

図 16 から，COCO_flxomp2 では，-opt-streaming-stores always について 1%ほど高速化の効果を確認できる．

COCO_flxomp3 では，-opt-assume-safe-padding，-O3，-opt-streaming-cache-evict=0 について，8%～16%ほど，高速化の効果が確認できる．

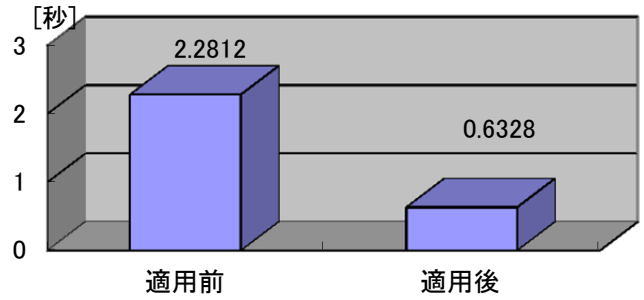
4.4 QCD の最適化

4.4.1 OMP ディレクティブ変更による最適化

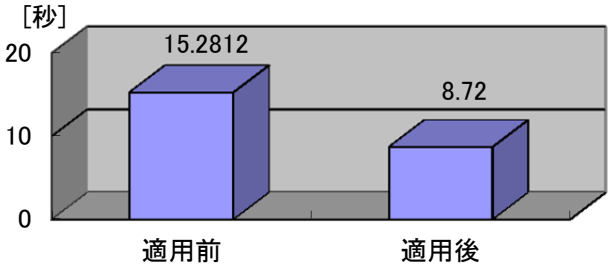
● 修正した OpenMP ディレクティブ

```
!$OMP PARALLEL DO COLLAPSE(3) PRIVATE…
  do ix=1,NX
  do iy=1,NY
  do iz=1,NZ
    :
```

上記修正を clover.h90，bicgstab_hmc.h90 に対して適用し，適用前後の実行時間の変化を測定する．図 17 にその結果を載せる．



(a) S15-clover



(b) S15-BiCGStab

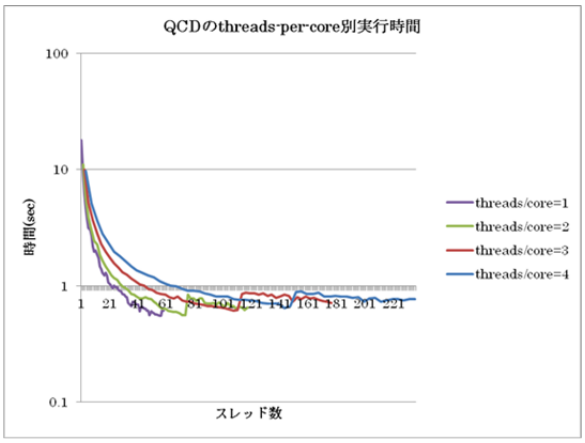
図 17 CCS-QCD の OMP ディレクティブ変更の効果 (Xeon Phi)

図 17 から，COLLAPSE ディレクティブ適用の結果，clover で約 7 割，BiCGStab で 4 割強，実行時間が減少した．

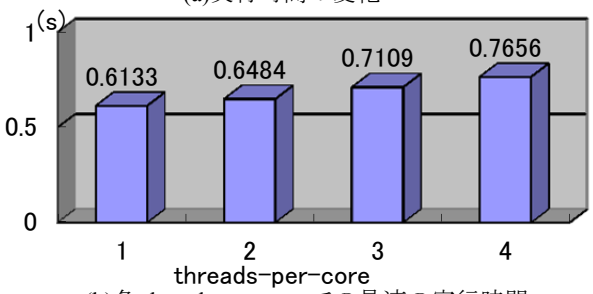
4.4.2 -opt-threads-per-core による最適化

図 18，図 19 に，CCS-QCD の演算カーネルについて，-opt-threads-per-core を変化させた場合の実行結果について載せる．

●S15 – clover (実測 B/F=2.05)



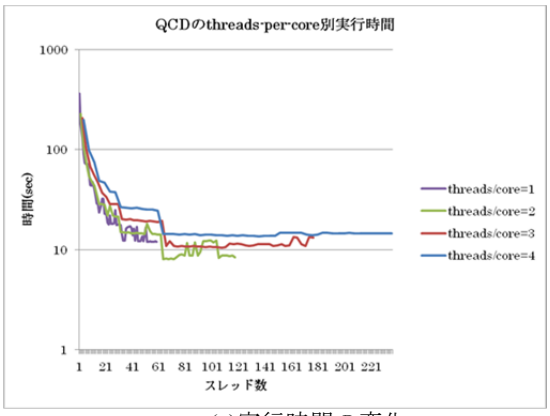
(a)実行時間の変化



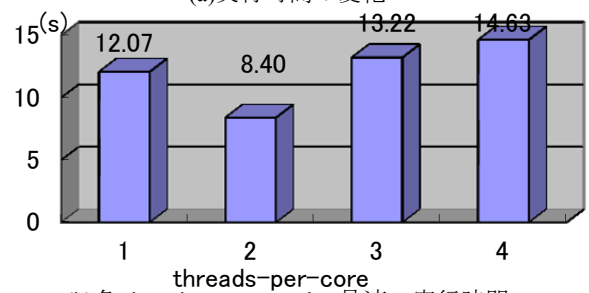
(b)各 threads-per-core での最速の実行時間

図 18 S15-clover の性能（Xeon Phi）

●S15 – BiCGStab (実測 B/F=1.70(全体), 5.29(行列積以外), 1.20(行列積))



(a)実行時間の変化



(b)各 threads-per-core での最速の実行時間

図 19 S15- BiCGStab の性能（Xeon Phi）

図 18, 図 19 から，BiCGStab (実測 B/F=1.70(全体), 5.29(行列積以外)，1.20(行列積))の時，threads-per-core=2 の効果が観測された．

4.5 Xeon Phi による最適化のまとめ

1 コア当たりのスレッド数を変化させる評価では，例外があるものの，実測 B/F が 5 以上のカーネルで，1 コア当たり 2 スレッド実行のほうが，1 コア当たり 1 スレッド実行よりも，高速である傾向がある．これは，高バンド幅を要求するカーネルではメモリアクセス時の待ち時間が起きいため，Hyper Threading によるメモリアクセス時間隠ぺいの効果が期待できるためと考えられる．詳しい解析は今後の課題である．

5. 異機種環境による評価

ここでは，異機種環境による性能評価について紹介する．本節での「チューニングあり」とは，FX10 でチューニングを行ったコードを異機種環境で動作させたものである．したがって，FX10 以外の環境でコード最適化したものではない点に注意する．なお Xeon Phi の実行ついては，4 章で調査した最速の実行形態で実行している．

5.1 NICAM の異機種評価

5.1.1 測定条件

機種ごとに以下の条件で性能測定を行った．

(1)**FX10**： 16smp
(2)**SR16000**： 32smp
(3)**Ivy Bridge**： 20smp
(4)**Xeon Phi**：

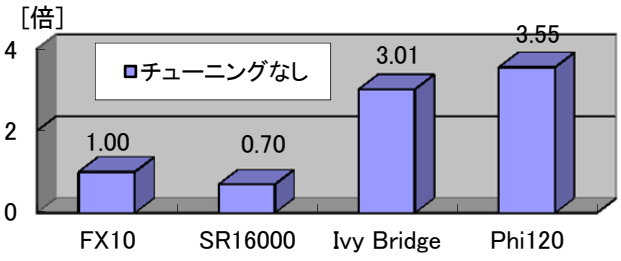
- **120smp** (NICAM_02_mod_oprt，NICAM_05_mod_src，NICAM_08_mod_oprt)
- **60smp** (NICAM_04_mod_oprt3d，NICAM_07_mod_src，NICAM_09_mod_mp_nsw6)

「チューニングあり」とは，FX10 でループ分割，ループ融合，および，アンローリングを実装してチューニングを行ったものである[4][5].

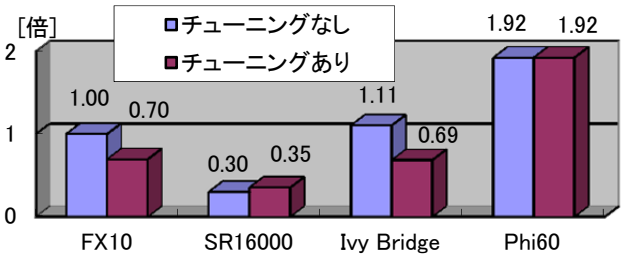
ここでは，FX10 上での「チューニングなし」の時間を 1.00 とし，それぞれの対象における実行時間の比を載せる．この実行時間の比が 1.00 未満のとき，FX10 での実行時間よりも高速であることを意味する．

5.1.2 NICAM の結果

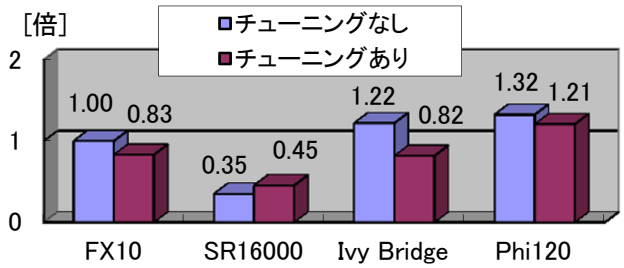
図 20 に，NICAM の主要カーネルについて，異機種環境で評価した結果を載せる．



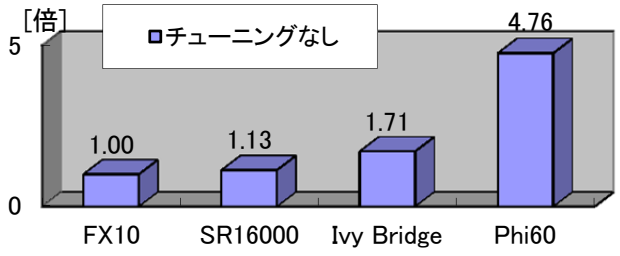
(a)NICAM_02_mod_oprt



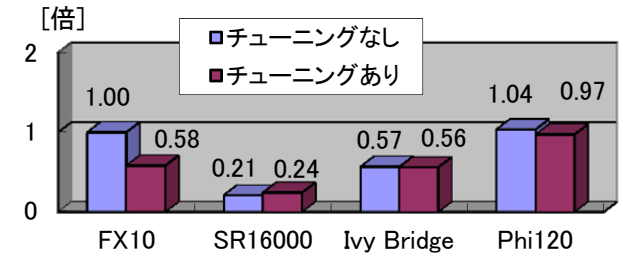
(b) NICAM_04_mod_oprt3d



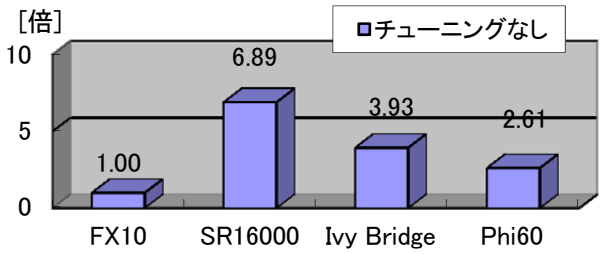
(c) NICAM_05_mod_src



(d) NICAM_07_mod_src



(e) NICAM_08_mod_oprt



(f) NICAM_09_mod_mp_nsw6

図 20　NICAM の異機種環境での評価結果

5.1.3 NICAM まとめ

図 20 から，チューニングについて FX10 で効果のある実装は異機種で有効とは限らない．また実行時間で，SR16000 が最も高速である場合が多い．ただし SR16000 と FX10 との理論 FLOPS 比（0.24）を考慮すると，SR16000 では必ずしも演算効率は高くない．一方，FX10 は 1 ノード当たりの理論 FLOPS 比からすると，効率が良いといえる．この理

由は，元のコードは FX10 で最適化されており，その他の CPU で必ずしも最適ではないからであろう．

5.2 COCO の異機種評価

5.2.1 測定条件

機種ごとに，以下の条件で測定した．

(1)**FX10**： 16smp

(2)**SR16000**： 32smp

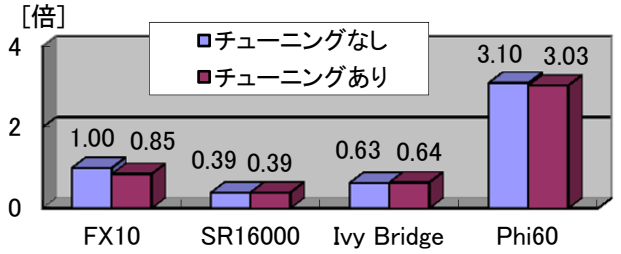
(3)**Ivy Bridge**： 20smp

(4)**Xeon Phi**

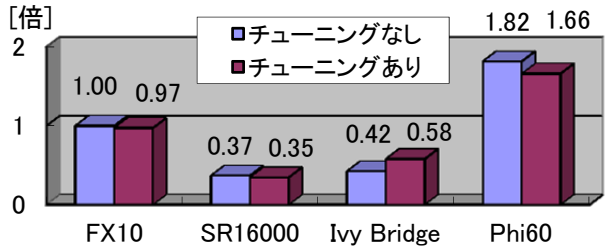
- **60smp** (COCO_flxomp2，COCO_flxomp3)
- **120smp**(COCO_flxomp5)

5.2.2 COCO の結果

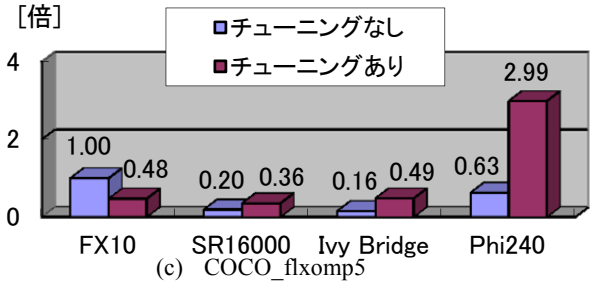
図 21 に，COCO の主要カーネルについて，異機種環境で評価した結果を載せる．



(a) COCO_flxomp2



(b) COCO_flxomp3

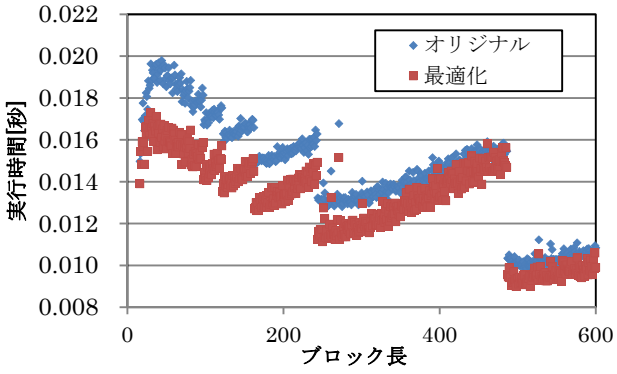


(c) COCO_flxomp5

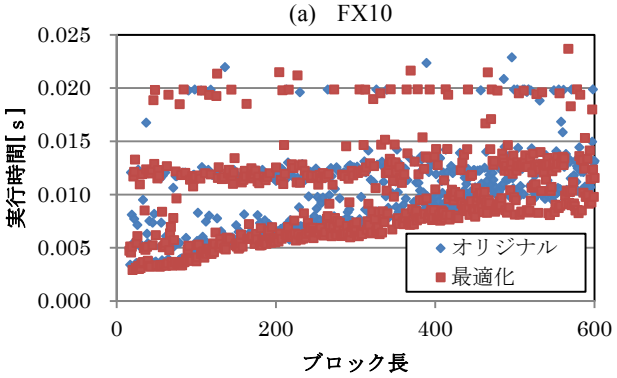
図 21 COCO の異機種環境での評価結果

5.2.3 COCO_flaxomp5 のブロック長の調整の効果

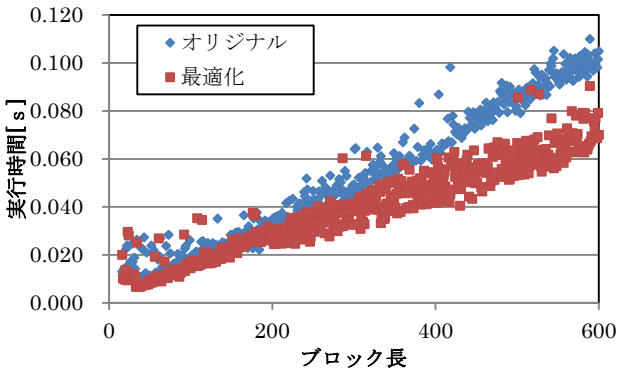
COCO_flaxomp5 はブロック化されているため，ブロック長が性能に影響を及ぼす．そこで，ブロック長を 1～600 まで変化させた場合の性能について，オリジナルコードとチューニング済みコード（最適化）の実行時間の変化を FX10，Ivy Bridge，および Xeon Phi で調査した．その結果を図 22 に示す．



(a) FX10



(c) Ivy Bridge



(b) Xeon Phi

図 22 COCO_flaxomp5 のブロック長調整の効果

図 22 より，FX10 ではブロック長が 500 周辺で高速であるが，Ivy Bridge と Xeon Phi はブロック長が 40 周辺で高速となる．表 4 に，最適なブロック長と実行時間を載せる．

表 4 最適なブロック長と実行時間

	項目	最適ブロック長	実行時間［秒］
1	FX10 オリジナル	511	0.0098
2	FX10 最適化	498	0.0089
3	Ivy Bridge オリジナル	17	0.0034
4	Ivy Bridge 最適化	20	0.0029
5	Xeon Phi オリジナル	36	0.0089
6	Xeon Phi 最適化	33	0.0067

5.2.4 COCO まとめ

図 21 から，Ivy Bridge は，COCO_flaxomp3, COCO_flaxomp5 で FX10 との理論 FLOPS 性能比(0.59)を考慮すると，効率が良いといえる．一方，Xeon Phi の実行時間は，理論 FLOPS 性能比に対して悪い．この理由は解析中であるが，COCO の演算カーネルは IF 文を含むため，最適化が Xeon Phi で不十分／困難であることが要因の 1 つとして考えられる．

5.3 CCS-QCD の異機種評価

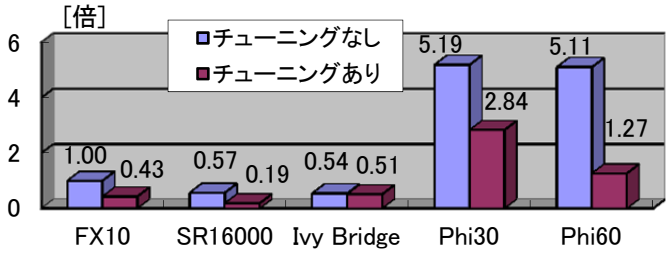
5.3.1 測定条件

問題サイズ N08P08C4 を用いて，機種ごとに以下の条件で行った．

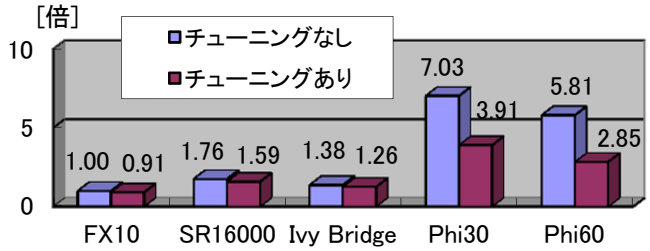
(1)**FX10**: 8mpi 16smp
(2)**SR16000**: 8mpi 8smp
(3)**Ivy Bridge**: 8mpi 8smp
(4)**Xeon Phi**：8mpi 30smp，8mpi 60smp

5.3.2 QCD の結果

図 23 に，CCS-QCD の主要カーネルについて，異機種環境で評価した結果を載せる．



(a) N08P08C4 – clover



(b) N08P08C4 – BiCGStab

図 23 CCS-QCD の異機種環境での評価結果

5.3.3 CCS-QCD まとめ

図 23 より，Xeon Phi の実行時間は，FX10 に対する理論 FLOPS 性能比を考慮すると演算効率が悪い. また, SR16000 および Ivy Bridge ともに，FX10 との理論 FLOPS 性能比に対して，演算効率が悪い．この理由は解析中であるが，各 CPU に対する最適実装について調査する必要がある．

5.4 異機種評価を通してのまとめ

全体を通して, FX10 における現在の実装は, 異機種 CPU に対して演算効率が良いといえる．現在の実装は，FX10 向きによく最適化されているといえる．一方，全般を通して Xeon Phi の演算効率が悪い．この原因分析は今後の課題である．本研究では主目的ではないが，Xeon Phi 向きの演算カーネルのチューニングを調査することは，コード最適化の観点から興味深い．

6. おわりに

本報告では，数値計算レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究での計算機設計に用いるアプリケーションにおいて，性能予測のための基礎データ取得時に必要な最適化，および異機種環境を通しての性能評価について説明した．

本評価を通して得た結論は，現状の FX10 での実行効率は十分であるといえる．FX10 向きに十分にチューニングされているといえる．一方で，Xeon Phi を中心とした異機種性能では，FX10 に対する理論 FLOPS 比の観点から，多くの場合，演算効率が悪い．この理由の一つは，本評価ではそれぞれの CPU 向きにチューニングをしておらず, FX10 に対してチューニングしたコードを移植しただけであることがあげられる．今後の性能解析が必要である．

また本評価の別の側面として，FX10 でチューニングされたコードを異機種環境に移植する場合，演算効率の差があまりないであろうことを示唆している．FX10 から異機種環境へのプログラムを移植する際の参考になる可能性がある．

最後に，OpenMP での並列化の際，対象となるループの並列性がスレッド数より少なくなる場合がある. この場合，ループ融合が必要である．ループ融合は手動で実装するほか，OpenMP のディレクティブで実装する方法がある．特に，Xeon Phi のようなメニーコア計算機での高スレッド実行では効果がある．

謝辞 本研究を行うに当たり，富士通 PRIMEHPC FX10 の性能プロファイラ情報など多数のご支援をいただいた富士通の諸氏に感謝いたします．

また，CCS-QCD Bench の利用に関し，広島大学　石川健一　准教授，筑波大学　藏増嘉伸　教授，宇川彰　教授，朴泰祐　教授に感謝いたします．

本研究は，文部科学省「将来の HPCI システムのあり方の調査研究」（平成 24 年度～平成 25 年度）の支援による．また本論文の結果の一部は，理化学研究所のスーパーコンピュータ「京」を利用するとともに，「京」以外の HPCI システム利用研究課題を遂行して得られたものです（課題番号:hp120128）．

参考文献

1) 文部科学省「将来の HPCI システムのあり方の調査研究」に係る実施機関の選定について」，平成 24 年 6 月 15 日．

http://www.mext.go.jp/b_menu/houdou/24/06/1322138.htm
2) HPCI 技術ロードマップ白書，2012 年 3 月.
http://open-supercomputer.org/wp-content/uploads/2012/03/hpci-roadmap.pdf
3) 計算科学ロードマップ白書，2012 年 3 月.
http://open-supercomputer.org/wp-content/uploads/2012/03/science-roadmap.pdf
4) 片桐ほか：レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究のためのアプリケーションと性能評価，情報処理学会研究報告 2012-HPC-137 (2012)
5) 片桐ほか：レイテンシコアの高度化・高効率化による将来の HPCI システムに関する調査研究のためのアプリケーション最適化と異機種計算機環境での性能評価，情報処理学会研究報告 2012-HPC-139 (2013)