VII.高性能計算システム研究部門

1. メンバー

教授	朴　泰祐
教授	高橋　大介
教授	建部　修見
准教授	川島　英之
助教	多田野　寛人
助教	小林　諒平
研究員	田中　昌宏
研究員	Mohamed Amin Jabri
研究員	藤田　典久
学生	大学院生　13 名、学類生　12 名

2. 概要

本研究部門では、高性能計算システムアーキテクチャ、並列プログラミング環境、GPU 利用技術、FPGA 利用技術、並列数値処理の高速化研究、分散システムソフトウェア、エクストリームビッグデータの基盤技術等の研究を行っている。

3. 研究成果

【1】　OpenCL から制御可能な FPGA 間高速イーサネット通信（朴、小林）

半導体微細加工技術の発展により、FPGA の演算および通信性能は飛躍的に向上し、高性能計算の分野への応用が期待されるようになった。FPGA の性能向上に加えて近年では、各 FPGA ベンダーが OpenCL ベースの開発環境を提供しており、設計者は従来よりも容易に開発を行うことが可能である。FPGA におけるこれらの技術革新は、設計者が一般的な開発環境を用いて、低レイテンシの通信を実行する回路に加えて、on-the-fly にオフロードされる CPU や GPU が不得手とする処理を実行する回路を FPGA 上に実装することによりアプリケーション全体の性能を向上させるという、図 1 に示すコンセプトの実現可能性を提示する。このコンセプトでは FPGA は単なる演算処理のオフローディングだけでなく、それ自体が持つ高速通信機能を積極的に用い、演算と通信を融合する回路を実装することで、これまでにない柔軟で電力効率の高い HPC 向けプラットフォームを構築する。このコンセプトを AiS（Accelerator in Switch）と名付ける。

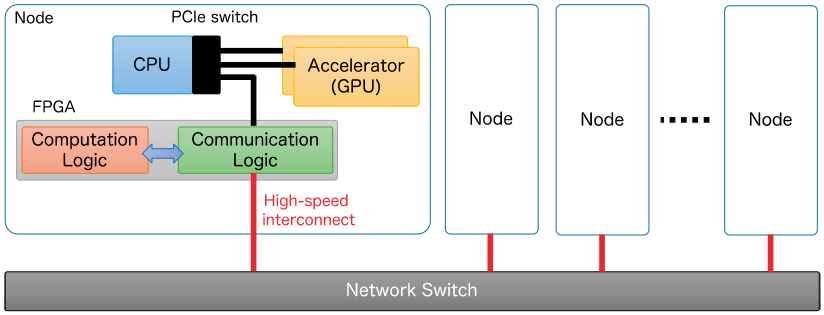


図 1　本研究が実現を目指すコンセプト AiS。低レイテンシの通信で支えられた並列処理が高速なインターコネクトで接続された複数の FPGA 上で実行される。

本研究では、そのコンセプトを実現するための OpenCL から制御可能な FPGA 間高速イーサネット通信手法について提案する。FPGA ベンダーが提供している OpenCL ツールチェインを活用し、FPGA の持つ高速通信リンクをユーザレベルから制御している。そして、ノード数は数十、数百とスケールさせることを前提としているため、通信の接続にはネットワークスイッチを利用する。このため、提案手法はイーサネット通信プロトコルをサポートしている。図 2 (a) にそのハードウェア実装と図 2 (b) に Ping-pong 通信用の OpenCL コードの一部を示す。

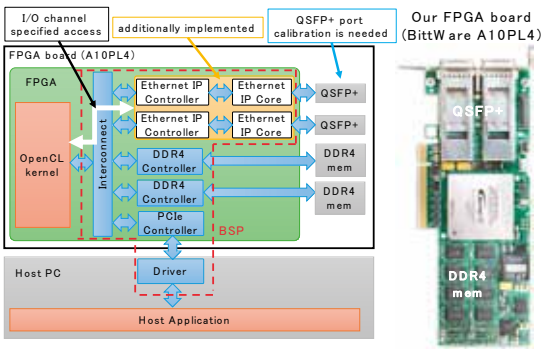
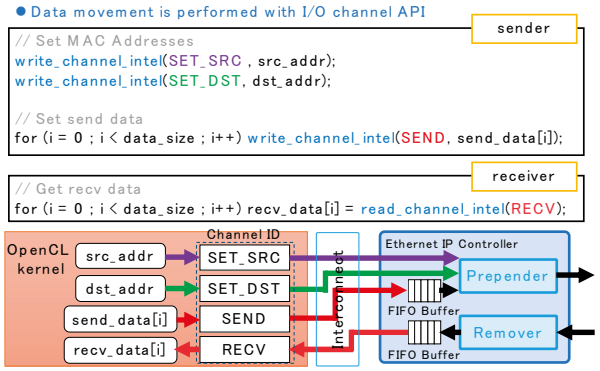


図 2　(a) ハードウェア実装の概略図



(b) Ping-pong 通信用の OpenCL コードの一部

FPGA で OpenCL プログラミングをするためには、Board Support Package（BSP）と呼ばれる、FPGA と FPGA ボードに搭載されている外部メモリなどのペリフェラルとの接続情報やホストの計算機との PCIe 通信のためのドライバなどを含む設定ファイルが必要であり、図 2 (a) の赤色の破線で囲まれた領域が BSP で提供される機能に相当する。本研究では、この BSP にイーサネット通信用のコントローラ（図 2 (a) の黄色のブロック）を組み入れることによって OpenCL からのイーサネット通信を実現している。そして、図 2 (b) に示すように、FPGA ベンダーが提供する OpenCL API（Intel FPGA SDK for OpenCL では I/O Channel API）を用いて、FPGA 間通信を実行する OpenCL カーネルコードを記述する。

AiS コンセプトに基づく一連の実証実験を行うため、計算科学研究センターに PPX（Pre-PACS-X）クラスタを導入した。PACS-X は第 10 世代の PACS システム開発計画であり、AiS コンセプトに基づく GPU と FPGA のハイブリッド演算加速装置を計算ノードに搭載する計画である。PPX は PACS-X の本格構築のために、予備実験や評価を行うためのミニクラスタである。H28 年度に 6 ノード、さらに H29 年度に 7 ノードを追加した計 13 ノードのシステムである。なお、第 1 期の 6 ノードには FPGA として Intel/Altera Arria10 を、第 2 期の 7 ノードには Xilinx Kintex をそれぞれ搭載しており、標準的に用いられる高性能 FPGA を比較実験できるように構成している。それ以外の要素は全ノードについて共通で、Intel Broadwell アーキテクチャの Xeon CPU を 2 ソケット、NVIDIA 社の Pascal アーキテクチャの Tesla P100 を 2 基ずつ、Mellanox InfiniBand EDR を 1 チャネル、各ノードに搭載している。また、FPGA 間通信実験のために、Aria10 あるいは Kintex FPGA ボードに搭載される 2 本の QSFP+ (40Gbps)のリンクを束ねる 100Gbps Ethernet スイッチも備える。提案手法の有効性を評価するために、通信レイテンシ・バンド幅の観点における FPGA 間通信性能の比較実験（図 3）を行った。

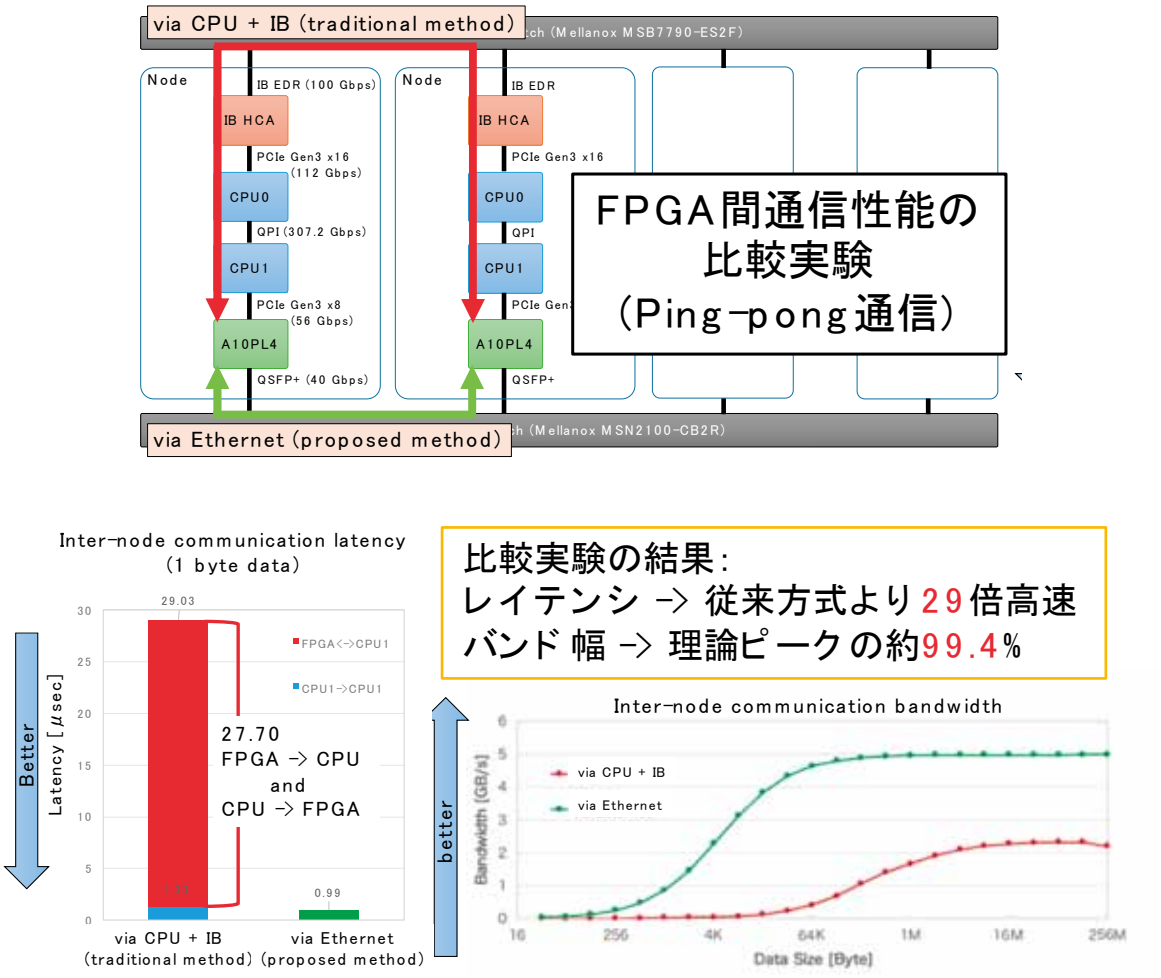


図 3　FPGA 間通信性能の比較実験

実験の結果、通信レイテンシは、CPU と InfiniBand を利用した従来方式より 29 倍優れており、InfiniBand における通信レイテンシとほぼ同程度であることが分かった。また、QSFP+の理論性能の 99.4%の通信バンド幅を達成しており、低レイテンシ通信によって通信バンド幅の立ち上がりが従来方式より優れていることが分かる。以上より、低レイテンシ・高バンド幅のイーサネット通信がユーザレベルから制御可能であることが定量的に確認された。

【2】 GPU・FPGA 連携システムの開発（朴、小林）

AiS コンセプトに基づく FPGA を用いた複合演算加速システムとして、FPGA と GPU を混載する演算ノードの実現が望まれる。これは、かつて HA-PACS/TCA の開発において実現した、PCIe 上に接続された GPU と FPGA を PCIe のパケット通信プロトコルを用いて通信させる技術と基本的に同じである。GPU と FPGA は、それぞれのメモリ（GPU の場合は global memory、FPGA の場合は DDR memory）を PCIe アドレス空間にマッピングすることで、PCIe バス上の DMA 機構を用いて双方のメモリ間でデータのコピーを行う。例えば、GPU のメモリ内容を FPGA にコピーする場合、CPU を介してソフトウェアで通信を行うと、GPU から CPU へのメモリコピーの後でその内容を CPU から FPGA にさらにコピーするという作業をプログラムで行わなければならず、GPU や FPGA の高速化に対する深刻な性能ボトルネックとなり得る。

H29 年度の研究において、FPGA 内の PCIe DMA ハードウェアを実装することで、GPU 上の global memory（PPX では NVIDIA P100 の HBM memory）と FPGA 上の DDR memory 間でデータ転送を実現する回路を Verilog HDL で構築し、その通信性能を評価した。この仕組みの様子を図 4 に示す。

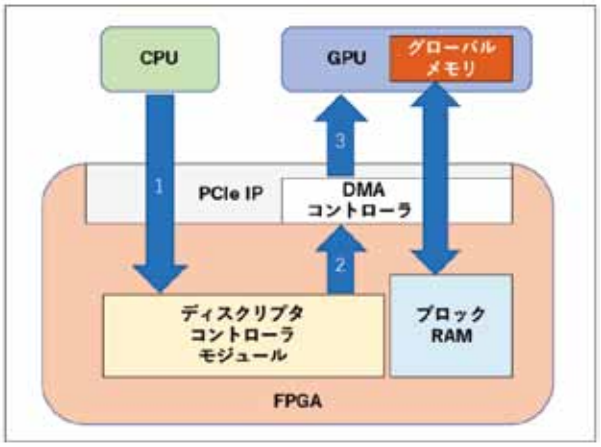


図 4 PPX ノードにおける GPU と FPGA 間の DMA 通信機構

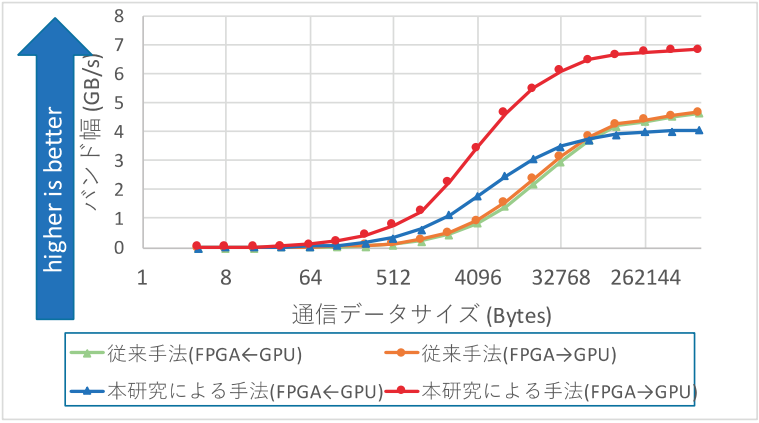


図 5　CPU 経由（従来手法）と FPGA-DMA による GPU と FPGA 間のデータコピー性能の差

PPX 上での DMA 転送の性能を図 5 に示す。ここでは CPU を介した従来手法と、本研究で開発した FPGA の DMA 転送を用いた GPU･FPGA 間直接転送の通信バンド幅の比較を示す。図から明らかなように、FPGA から GPU への通信では常に従来手法より性能が高く、最大で 2.3 倍の性能差、またデータ長が長い場合（256KB 以上）で約 1.4 倍の性能差が確認された。一方､GPU から FPGA への通信では 64KB データサイズまでは本研究の手法が勝っているが、これを超えると従来手法のほうが性能が良い結果となった。この理由は現在解析中であるが、特にデータサイズが小さくなる細粒度並列処理を行う場合、提案手法のほうが有利であることは確認された。

【3】　FPGA 上の高位合成による HPC アプリケーションの実装（朴、小林）

H29 年度で終了する JST-CREST における TCA（Tightly Coupled Accelerators）コンセプトを発展させ、FPGA により積極的にアプリケーションの一部をオフロードすると共に、引き続きネットワークインタフェース機能や GPU との連携機能を盛り込んだ新しいコンセプト AiS（Accelerator in Switch）の基盤研究として、FPGA 上での高レベル言語 OpenCL によるアプリケーション記述と、低レベル機能の Verilog HDL 記述を並行して行い、両者を融合した新しい FPGA プログラミングのフレームワークを構築する研究を推進した。本研究は計算科学研究センターにおける次期ハイブリッド演算加速クラスタの構築を目指す PACS-X 計画に引き継がれる重要な研究である。

H29 年度研究では、宇宙物理研究部門との共同研究の下、宇宙初期天体形成シミュレーションにおいて、重力と並ぶ重要な物理現象である輻射輸送現象の処理を FPGA オフローディングする研究を集中的に行った。ここで用いられるのは最も重い演算である ART 法の処理である。ART 法はその性質から、GPU による大規模並列処理に不向きであり、FPGA による演算加速が期待される。我々は各空間セルにおける多数の光線トラッキングをパイプライン処

理するアルゴリズムと、発生する処理をプールしておいて順次ワーカーに動的割付を行うアルゴリズムの 2 種類を FPGA 上に実装した。処理の都合上、前者は高レベル合成言語である OpenCL で、後者は低レベルハードウェア記述言語である Verilog HDL で実装した。また、処理系の制約により前者は Intel/Altera の Arria10 FPGA に、後者は Xilinx Kintex FPGA に実装した。両ハードウェアの論理回路規模、通信性能、テクノロジはほぼ互角であり、性能差は実装手法によるものであると考えて良い。

まず、OpenCL による実装と最先端 GPU である NVIDIA Tesla P100 との性能を比較する。結果を図 6 に示す。横軸は問題サイズ、縦軸は単位時間当たりの処理メッシュ数を性能指標として示す。図からわかるように、CPU 実装はノード当たり最大 28 コア用いた場合でも FPGA と GPU には遠く及ばない。また問題サイズが小さい場合、FPGA の性能は GPU の 10 倍以上である。しかし、問題サイズが大きくなると、演算の大粒度化により GPU の並列処理効率が向上するのに対し、FPGA では論理回路内蔵の BRAM メモリにデータが乗り切らず、DDR メモリを用いることで性能が低下する。今後、BRAM をうまくキャッシュ的に利用しつつ、性能を向上させる工夫が必要である。

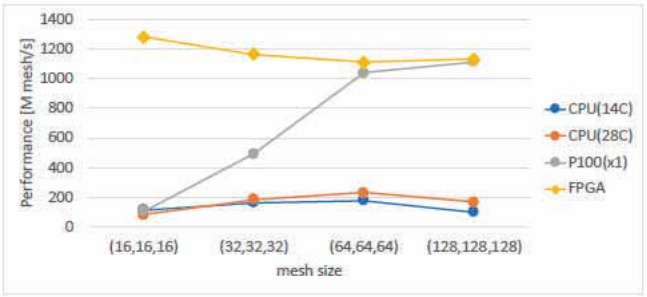


図 6　ART 法の FPGA 実装における性能と CPU 及び GPU による実装の性能比較

次に、OpenCL 実装と Verilog HDL 実装の性能比較を示す。図 7 は ART 法を Xilinx Kintex 上に Verilog HDL 実装した場合と Intel/Altera Arria10 FPGA に OpenCL 実装した場合の性能を、CPU における 1 コアでの地区実装性能を 1 として示したものである。参考のため、16 コアによるマルチスレッド実装の性能も示してある。ここからわかるように、両 FPGA 実装は CPU の逐次性能に比べ 100 倍～120 倍程度の性能向上、また CPU のマルチスレッド実装に比べても 10 倍以上の性能を達成している。しかし、両 FPGA 実装を比べると OpenCL 実装の方が 15%程度高速である。この差は主に、両実装を最終的に論理合成した結果としての動作周波数の差が影響している。Verilog HDL 実装が最大で 200.0 MHz であったのに対し、OpenCL 実装は 236.11 MHz まで伸びた。この結果は直感に反するものであり、現在 Verilog HDL 実装につ

いて性能チューニングを行っている。いずれにしても、FPGA へのオフローディングがこの問題に対して非常に有効に働くことが確認された。

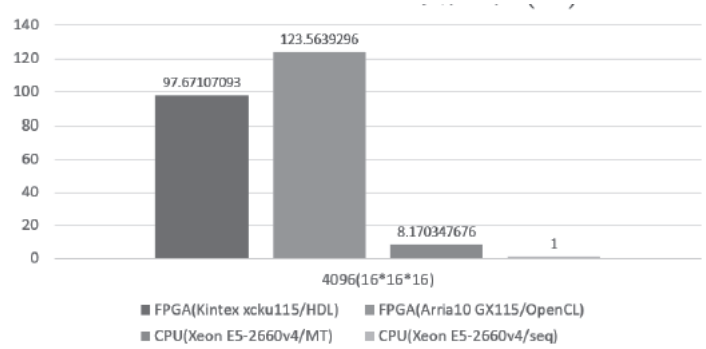


図 7　ART 法の OpenCL 実装と Verilog HDL 実装による比較（CPU 逐次実装を 1 とする）

【4】　メニーコアプロセッサ向けアプリケーション性能向上　（朴）

昨年度に継続し、CCS の矢花グループとの共同研究の下、同グループで開発中の物性第一原理計算コード ARTED（Ab initio Real Time Electron Dynamics simulator）のメニーコアプロセッサ向け性能最適化を行った。H29 年度は本格可動を開始した、JCAHPC における世界最大の KNL クラスタである OFP（Oakforest-PACS）の全ノードを利用した同コードの性能評価を実施し、極めて良好な weak scaling 及び strong scaling 性能を得ることができた。

OFP 上の計算ノードは Intel Xeon Phi（コード名 KNL: Knights Landing）が実装されており、従来のアクセラレータとしてのコプロセッサである Xeon Phi（コード名 KNC: Knights Corner）ではなく、self-bootable な CPU として動作する。このため、ノード間通信やメモリアクセスにおいて、KNC を用いていた COMA よりシンプルで効率的な処理が行えるだけでなく、KNC の約 3 倍の理論ピーク性能が実現され、大幅な性能向上が実現できる。しかし、基本的な性能チューニングは KNC とほとんど変わらないため、COMA 上で開発された KNC 向けの ARTED をほぼそのまま、OpenMP と MPI の呼び出し方を変更するだけで移植できる。図 8 に ARTED の最重要カーネル部分である、3 次元 25 点実空間ステンシル計算の主要部分コードを示す。ここでは non-temporal store の活用、512bit SIMD 命令の演算数と配置にフィットさせたデータ配列宣言、整数剰余演算を省くためのテーブル置き換え等の最適化を行っている。これらの結果、ステンシル計算カーネルでは KNL の理論ピーク性能の 25%の実効性能が達成された。

```
real(8),   intent(in)  :: B(0:NLz-1,0:NLy-1,0:NLx-1)
complex(8),intent(in)  :: E(0:NLz-1,0:NLy-1,0:NLx-1)
complex(8),intent(out) :: F(0:NLz-1,0:NLy-1,0:NLx-1)

#define IDX(dt) iz,iy,modx(ix+(dt)+NLx)
#define IDY(dt) iz,mody(iy+(dt)+NLy),ix
#define IDZ(dt) modz(iz+(dt)+NLz),iy,ix

do ix=0,NLx-1
do iy=0,NLy-1
!dir$ vector nontemporal(F)
do iz=0,NLz-1
  v=0; w=0
  ! z-computation
  v=v+Cz(1)*(E(IDZ(1))+E(IDZ(-1))) ...
  w=w+Dz(1)*(E(IDZ(1))-E(IDZ(-1))) ...
  ! y-computation
  ! x-computation
  F(iz,iy,ix) = B(iz,iy,ix)*E(iz,iy,ix) &
  &           + A          *E(iz,iy,ix) &
  &           - 0.5d0*v - zI*w
end do
end do
end do
```

図 8　Xeon Phi 向けに最適化された 3 次元 25 点ステンシルコードのカーネル部分

また、KNL においては高バンド幅メモリである MC-DRAM（16GB/node）と低バンド幅メモリである DDR4（96GB/node）をうまく利用することが必要であるが、ARTED コードは元来、ノード上の全データを一括してスキャンするのではなく、3 次元ステンシル計算に必要な部分コピーを計算バッファにコピーしてから処理するようになっている。このため、この計算バッファを MC-DRAM 上に確保することにより、ほとんど苦労せず flat mode における MC-DRAM の有効利用が可能である。他の多くのコードが flat mode への移植困難のため、効率の落ちる cache mode を用いているのに対し、性能を最大化することが可能となった。計算のワーキングセットが MC-DRAM の容量に収まる場合と、これを超える場合について、演算時間はリーズナブルであり FLOPS 値が落ちていない様子を図 9 に示す。

OFP 上ではその大規模並列性を活かし、波数空間での並列性を全計算ノードに展開した。ここで、各波数空間における 3 次元空間は比較的小さいため、これを複数ノードで並列する domain decomposition 法は用いない。これにより、ノード間通信コストを最小限にし、高い並列処理効率が望める。図 10 に、128 ノードまでの COMA 及び OFP における strong scaling 性能を示す。

もう一点、特筆すべき事項として、OFP の全ノード利用実験を通じて発見された重要な現象がある。KNL では一般的な Xeon CPU と同様、turbo boost mode が備わっており、プロセッサの電力消費目標を超えない範囲で瞬間的に動作周波数を向上させることができる。しかし、この boost 効果は演算コア単位で行われ、基本的に非同期である。また、多数の KNL プロセッサを用いる OFP のようなシステムでは各ノードの turbo boost も非同期に発生する。結果として、同一コードを同一データ量で処理しているにもかかわらず、その性能はノード毎にバラつくという現象が発生する。このことは、OFP を部分的に利用している状態ではあまり目立たないが、本研究のような全系利用では如実に現れることとなった。図 11 に、全系利用で

の ARTED 計算において、性能が最高のノードと最低のノードでの各部分処理の内訳を示す。ARTED では 1 つのハミルトニアン計算の中で数回の全系による MPI_Allreduce 処理が発生し、ここで全ノードの同期が取られるため、結果として最低速度のノードが律速となり、システム性能が相対的に低下することになる。現時点でこの現象に対する有効な手法は見つかっておらず、プロセッサを提供する Intel 社との協議においても重要な懸案となっている。

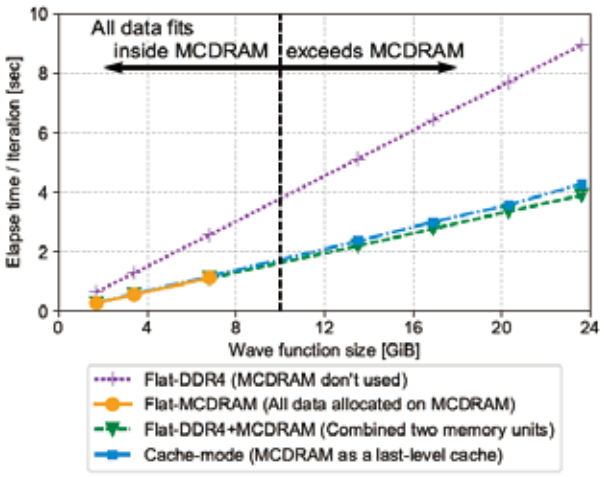


図 9　OFP の 1 ノード上で MC-DRAM 容量を超えるデータを扱う場合の処理時間（MC-DRAM 容量内のデータ処理に比べ線形に増加するのみで FLOPS 値は低下していない）

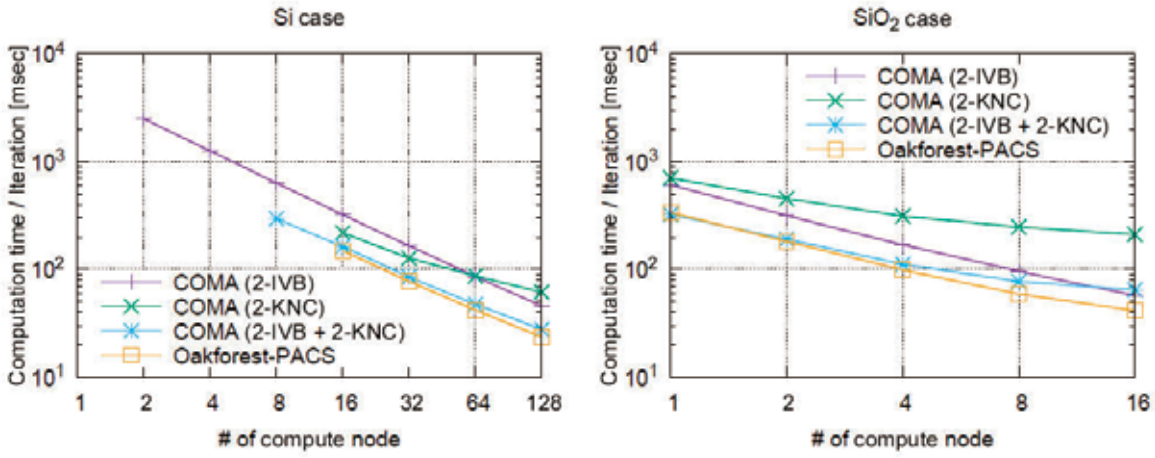


図 10　128 ノード及び 16 ノード（物質はそれぞれ Si 及び SiO_2）使用時の COMA（KNC）と OFP（KNL）の strong scaling 性能

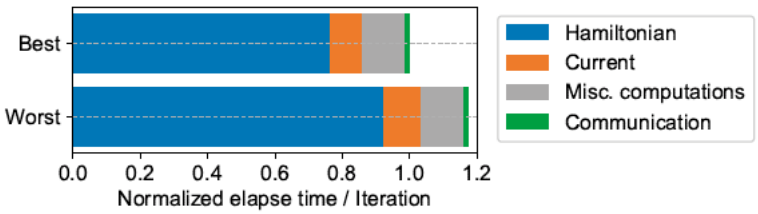


図 11　全系計算における turbo boost 状況に差により発生するノード間性能のバラつき

【5】　XcalableMP における動的タスク並列記述（朴）

XcalableMP（以下、XMP）は筑波大学計算科学研究センター（以下、CCS）で研究開発が開始され、その後理化学研究所計算科学研究機構（以下、AICS）に研究母体が移り、その後は AICS と CCS で共同研究が進められている、超並列記述向けディレクティブベース並列記述言語である。ユーザは並列実行プロセス配列のテンプレートに合わせ、各配列の分割と並列ループへの割り付けについて、元となる逐次コードに対して専用のディレクティブを追加することで比較的容易に超並列化を実現でき、さらにコード内で並列通信を意識することにより演算と通信の効率的実行についての性能チューニングが行える。

前年度より XMP におけるタスク並列処理における記述性及び性能の向上について研究を継続している。今年度は同処理系の実装について、OpenMP+MPI による実装と、ANL（Argonne National Laboratory）で開発されている Argobots を用いた実装を行い、さらに通信最適化として Send/Recv を用いた場合と Put/Get を用いた場合の比較評価を行った。

従来の OpenMP におけるタスク並列処理とフロー記述に MPI による通信同期を組み合わせることで、共有メモリと分散メモリを組み合わせたタスク処理の記述は理論的に可能である。しかし、その記述は複雑であり、単純なデータ並列に基づく OpenMP+MPI 記述以上に多くのバグを含む可能性を高め、プログラムの生産性も低下する。他方、単純なデータ並列では十分な並列性が生み出せず、複雑な分岐等を含むコードの大規模並列記述にはタスク記述は重要な要素である。XMP におけるタスク処理は、こういった背景の下、より簡潔な記述により適正なタスク並列処理を効率的に実現することを目標としている。これにより、ポスト京コンピュータで想定されるメニーコアアーキテクチャにおいて様々な手法でノード内並列性を向上させることにも大きく貢献すると考えられる。

#pragma xmp tasklet *[clause[, clause] ...] [on* { *node-ref* | *template-ref* } *]*
(structured-block)

#pragma xmp taskletwait *[on* { *node-ref* | *template-ref* } *]*

#pragma xmp tasklets
(structured-block)

where *clause* is :
{in | out | inout} (*variable[, variable] ...]*)

図 12　XMP におけるタスク記述追加文法

図 12 に XMP におけるタスク並列記述の基本文法を示す。tasklet 指示文はスレッドレベルのタスクの定義を行い、on 節に示された条件に従って全てのあるいは特定のノード上でタスクを生成する。in、out、inout の 3 種類の節によりタスクフローの制御を行う。tasklets 指示文は tasklet 指示文で指定されたタスクを並列に実行する構文である。taskletwait 指示文はタスクによる特定の条件の成立を待ち同期を取る構文である。XMP におけるタスク並列処理の例を以下に示す。

```
int A[100];  /* Distributed array */
int B[25];   /* Local array */
#pragma xmp nodes P(4)
#pragma xmp template T(0:99)
#pragma xmp distribute T(block) onto P
#pragma xmp align A[i] with T(i)
/* … */
#pragma xmp tasklet out(A[0:25], P(4)) on P(1)
taskA();

#pragma xmp tasklet in(A[0:25], P(1), B) on P(4)
taskB();

#pragma xmp taskletwait
```

図 13　XMP tasklet 指示文によるプログラミング例

昨年度の研究では、XMP におけるタスク構文の基本的なコンパイラ開発を行ったが、今年度はノード間通信の最適化を中心に、OpenMP+MPI 及び Argobots による実装と最適化を行った。さらに、XMP におけるグローバルビューモデルでもタスク並列を記述するために、gmove 及び reflect 指示文の機能拡張を行った。図 14 にこれらの機能拡張構文を示す。また、図 15 に tasklet reflect 構文を用いた記述例を示す。

#pragma xmp tasklet gmove *[clause[, clause] ...] [on { node-ref | template-ref }]*
(an assignment statement)

#pragma xmp tasklet reflect *(array-name[, array-name] ...)*
*[*chunksize *(reflect-chunksize[, reflect-chunksize] ...)]*

where *clause* is :
{in | out | inout} *(variable[, variable] ...])*

図 14　グローバルビューモデルにおける gmove 及び reflect 構文のタスク処理対応

```
int A[4],
#pragma xmp nodes P(2)
#pragma xmp template T(0:3)
#pragma xmp distribute T(block) onto P
#pragma xmp align A[i] with T(i)
#pragma xmp shadow A[1]

#pragma xmp tasklets
{
#pragma xmp loop (i) on T(i)
  for (int i = 0; i < 4; i++) {
#pragma xmp tasklet in(A[i]) on T(i)
    A[i] = i; /* taskA */
  }
#pragma xmp tasklet reflect(A)
  /* taskB */
}
```

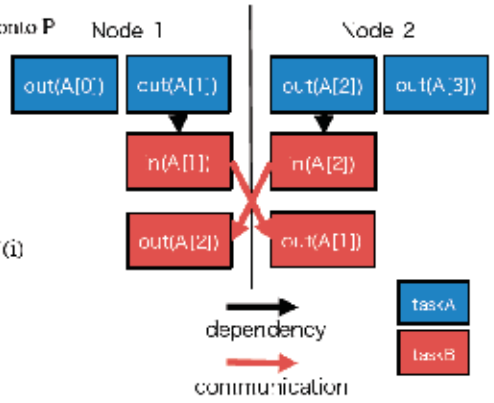


図 15　tasklet reflect 構文のコード例

XMP におけるタスク構文を OpenMP+MPI に変換する場合、各ノード内で多数のスレッドがタスク実行を行っている状況を想定し、MPI_Isend、MPI_Irecv、MPI_Test を用いたノンブロック非同期通信によって処理を行うが、gmove 構文については全てのノードが処理への関与を判断できるため、MPI_Send、MPI_Recv による記述に置き換えが可能である。これらにノード内タスク処理を OpenMP タスク記述で追加することで実装する。一方、XMP のローカルビューモデルでは coarray アクセスを実現するために Send/Recv 系ではなく Put/Get 系での実装を行う。

OpenMP に代わるもう一つのスレッドレベルの並列タスク実装方法として、細粒度軽量スレッドライブラリである Argobots を用いる実装及び最適を行った。Argobots ではスレッドは Working Unit（WU）という単位で記述されるユーザレベルスレッドであり、これらは直接コアにマッピングされるのではなく、各コアが実行する Execution Stream（ES）がそのキューに積まれた WU をスケジューリング実行する。また、全てのコアが単一の ES を共有することも可能であり、XMP ではこのモデルを用いる。

以上の実装について、JCAHPC が運用する Oakforest-PACS（OFP）及び筑波大学計算科学研究センターが運用する COMA 上で性能評価を行った。ここでは OFP の結果を示す。OFP は各ノードに Intel Xeon Phi（KNL: Knights Landing）プロセッサを用い、ノード間通信ネットワークに Intel OmniPath Architecture（OPA）を用いた超並列クラスタである。ベンチマークとしてブロックコレスキー分解とラプラスソルバを用いたが、ここでは前者の評価を示す。

まず MPI に Send/Recv 系を用い、OpenMP と Argobots でタスク並列実装を行った場合の評価を示す。ベースラインとなる評価は通常の並列ループによる実装で、これに OpenMP 及び Argobots を用いた場合のベースラインと最適化の結果、そして XMP グローバルビューによる結果を図 16 に示す。

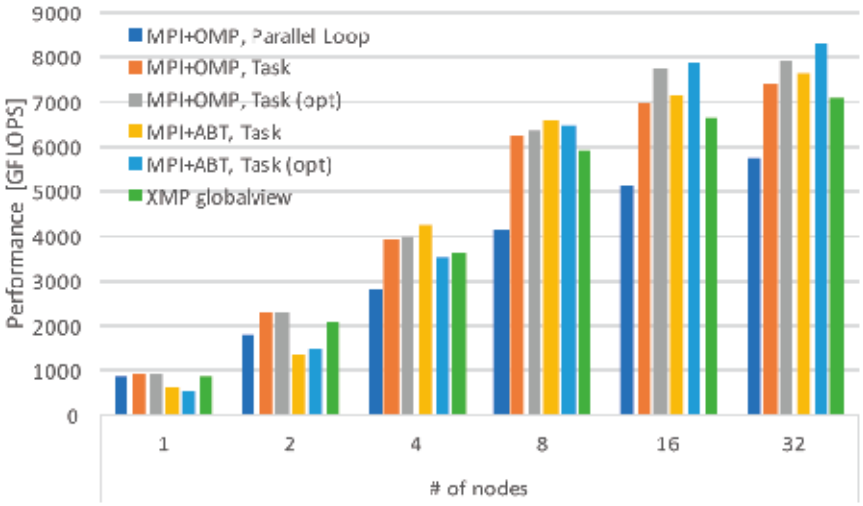


図 16　Send/Recv 系によるブロックコレスキー分解の実装と性能評価

MPI と組み合わせた OpenMP（OMP）または Argobots（ABT）実装及びそれぞれの最適化実装を比較すると、ノード数が少ない場合は OpenMP が高速だがノード数の増加、すなわちタスク粒度が細かくなるにつれ Argobots の性能が高くなるのがわかる。ベースラインとの比較で Argobots は最大で 34%の性能向上が見られた。Argobots 実装ではスレッドの yielding が頻繁に起こっていることが確認され、通信と演算のオーバーラップが効率的に行われ、タスク処理の効率が上がっていることがわかった。XMP のグローバルビューを用いたタスク並列（XMP globalview）においては MPI+OpenMP に比べ 5%ほど性能が低下している。これは、XMP globalview の場合、ブロックコレスキー分解の処理の一部において、演算結果がローカルに残っている場合でも通信によってこれを求めることで通信オーバヘッドが生じていることが確認されている。

続いて Put 系による実装の評価を示す。同じく OFP による性能評価を、同様にベースラインとなる並列ループ実装、OpenMP 及び Argobots によるタスク並列とその最適化結果について図 17 に示す。それぞれの実装の意味は図 16 と同じで、MPI 通信に Put 系を用いている。

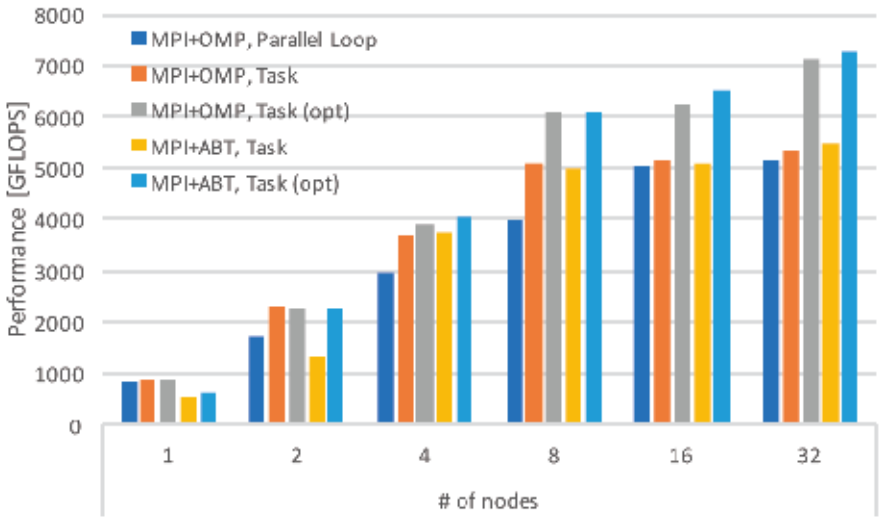


図 17　Put 系によるブロックコレスキー分解の実装と性能評価

この結果によると、OpenMP 及び Argobots とも、タスク処理の通信最適化が効いており、性能が大きく向上しているが、OpenMP と Argobots の差はあまり大きくない。32 ノード並列の場合、ベースラインからは 32～33%の性能向上が達成できており、通信最適化の効果が大きいことがわかる。しかし、絶対性能としては、32 ノードにおける最高性能である MPI+Argobots において、Send/Recv 系による実装が Put 系実装をやや上回ることが確認されたが、その差は大きくない。

以上のように、本年度の研究では XMP におけるタスク並列処理を Argobots によって実装することで性能が大きく改善することがわかったが、タスク並列による XMP グローバルビュー記述をそのまま実行した場合は MPI+Argobots の場合に比べ、細部での通信時間隠蔽等の最適化が十分行えず、引き続き改良が必要であることがわかった。

【6】　大規模並列システム性能予測ツール SCAMP（朴）

ポスト京システムを含む次世代超並列計算システムにおける weak scaling 性能の予測は、strong scaling に比べ比較的容易と考えられがちだが、実際には問題規模の適切な設定、通信時間の増加、キャッシュヒット率等のデータアクセス特性の変化があり、必ずしも容易ではない。また、通信パターンの変化についてもシミュレーションを行う必要があり、単純な机上評価では大きな誤差を生む可能性がある。そこで、同一プログラム・同一問題サイズの並列処理プロファイルを利用して、より大規模なシステムでの性能予測を行う手法とこれに基づくツールとして開発しているのが SCAMP（Scalable MPI Profiler）である。図 18 にこのコンセプトを示す。

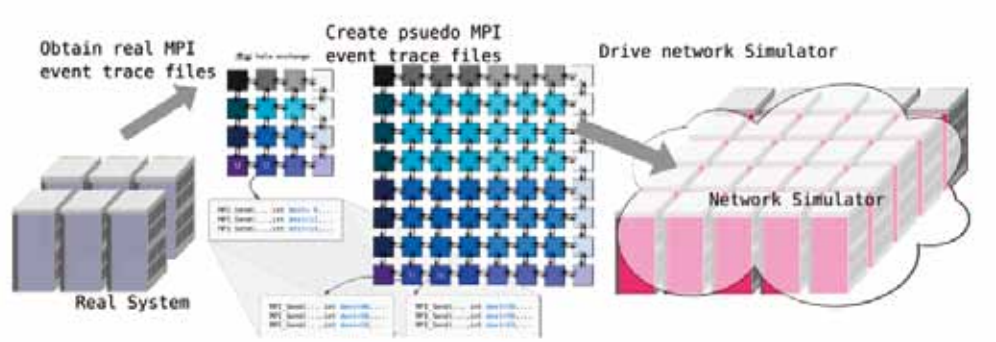


図 18　SCAMP における MPI プロファイルの「水増し」による大規模シミュレーション

本年度の研究では、昨年度までに概念設計を行った、現行実機での MPI プロファイルに対する「水増し」処理による仮想プロファイル作成作業において、LLVM-IR に基づく自動化作業を中心に研究を行った。SCAMP の基本的発想は、小規模システム（現行システム）における実 MPI プロファイルを、よりノード数の大きなシステムを想定した通信シミュレーションの対象となる仮想プロファイルに「水増し」することである。多くの典型的データ並列処理では、MPI 通信における通信相手やメッセージサイズが、総 MPI ランク数及び自 MPI プロセスのランク番号からの線形式で単純に表現可能な場合が多く、この点に着目するとこれらの並列 MPI コードでは「水増し」プロファイルにおける通信相手やメッセージ長を自動的に生成できると推測される。この仮定の下で LLVM によるコード解析から大規模システムのための仮想プロファイルを自動生成し、ネットワークシミュレータによって評価可能にする。図 19 に、MPI の実際の引数となる相手ランク番号から、バックトレースによって MPI の総ランク数と自プロセスランク番号に行き着く解析の様子を示す。

SCAMP ではこれらの水増しされたプロファイルを元に、Sandia National Laboratory によって開発された SST/Macro シミュレータを用いたネットワークシミュレーションを行い、次の世代の大規模システムにおける性能予測を行う。京コンピュータとポスト京コンピュータではほぼ形状の等しい ToFu ネットワーク及びその改良版を用いるが、このトポロジは SST/Macro で標準的に用意されており、評価が可能である。

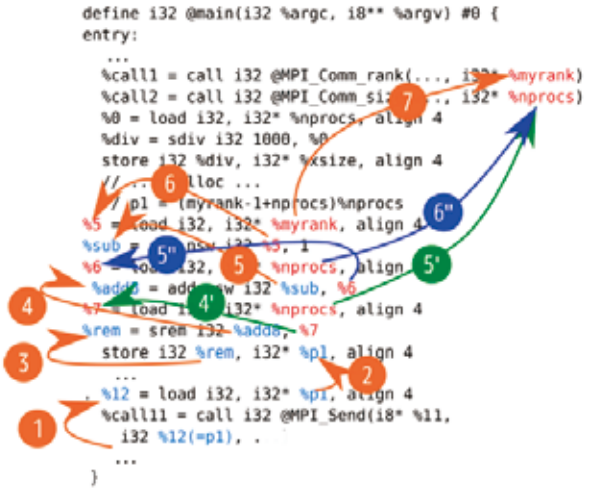


図 19　MPI 引数からのバックトレースによる MPI 実行パラメータの特定

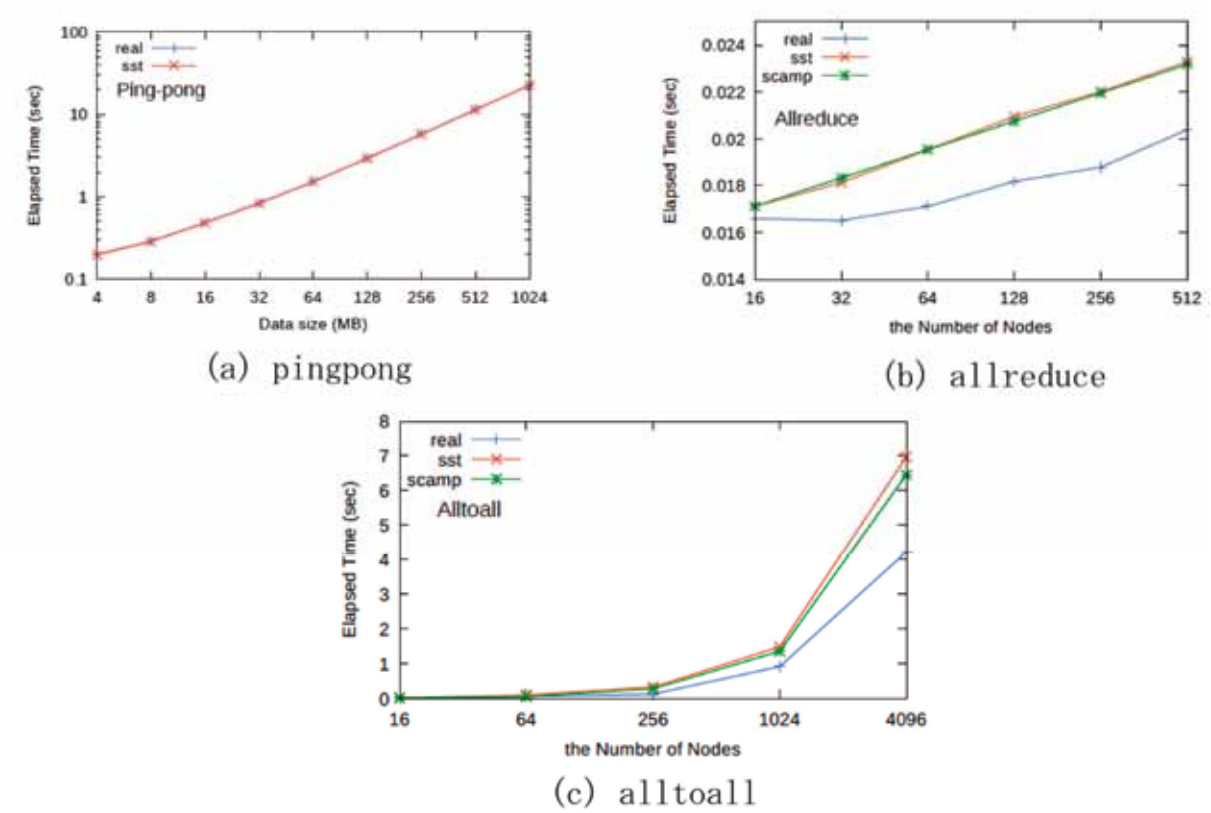


図 20　各種通信パターンにおける実機 (real)、SST/Macro (sst)、SCAMP (scamp)での予測実行時間

図 20 に pingpong、allreduce、alltoall という 3 種類の典型的通信を対象に、実機、SST/Macro、SCAMP による通信性能予測結果を示す。pingpong は 2 ノードによる一対一通信のため

SST/Macro の評価のみであるが、他の 2 種類は SCAMP による水増しプロファイラを用いた結果を示している。SST/Macro は手動による MPI パラメータの書き換え、SCAMP は自動化による書き換えである。allreduce 及び alltoall の両者で、実機での時間が SST/Macro 及び SCAMP によるシミュレーション時間を下回り、高速になっている。この誤差については現在解析中である。また、それと平行してステンシル計算や NPB-FT 等のベンチマークにおける評価を実施中である。

本年度の研究により、LLVM-IR により、SCAMP で想定している MPI ランク等のパラメータからの大規模システム用仮想プロファイルの自動生成が原理的に可能であることが確認された。今後、SST/Macro でのシミュレーション精度の向上と、より精度の高い strong scaling 予測のためにノード内処理時間の予測を組み合わせていく予定である。

【7】 Xeon Phi クラスタ上の並列 FFT における自動チューニング（高橋）

科学技術計算において広く用いられている高速 Fourier 変換（fast Fourier transform、以下 FFT）の性能を改善するために、自動チューニングに関する研究を行った。平成 28 年度は並列 FFT において演算と通信をオーバーラップさせる際の通信隠蔽のパラメータを自動チューニングする手法について検討を行っていた。平成 29 年度はさらに全対全通信方式についても自動チューニングを行い、通信隠蔽と組み合わせることで性能向上を図った。

今回実現した並列一次元 FFT は six-step FFT と呼ばれるアルゴリズムに基づいている。分散メモリ型並列計算機において six-step FFT を実現する際には、入力と出力をブロック分割した場合、全対全通信が 3 回行われることから、計算時間の大部分が全対全通信によって占められることになる。演算と通信をオーバーラップする手法としては、MPI の非同期通信を用いる方法が広く用いられているが、OpenMP を用いた通信用スレッドを導入する手法が Idomura らによって提案されている。この手法を応用することで、演算と通信を分割しパイプライン方式でオーバーラップさせることが可能である。

分散メモリ型並列計算機において並列一次元 FFT を自動チューニングする際には、全体に関わる性能パラメータとして主に以下の 4 つが存在する。

(1) 全対全通信方式
(2) 通信メッセージサイズの分割数
(3) 基底
(4) ブロックサイズ

これらの性能パラメータを探索することで、並列一次元 FFT の性能をさらに向上させることが可能である。なお、(1)～(2)は MPI プロセス間通信に関するパラメータであり、(3)～(4)は MPI プロセス内の性能に関するパラメータである。今回は(1)～(2)に対して自動チューニングを適用した。

これまでに、MPI の集合通信を自動チューニングする研究が行われている。また、InfiniBand で接続されたマルチコアクラスタにおいて、全対全通信をノード内とノード間の 2 段階に分けて行うことで、性能を向上させる手法が Kumar らによって提案されている。この手法をP個の MPI プロセスが$P = P_x \times P_y$と分解できる一般的な場合に拡張した、「2 段階全対全通信アルゴリズム」は以下のようになる。

ここで、NはすべてのMPIプロセスにまたがる配列の要素数の合計であるとする。

(1) 各 MPI プロセスにおいて、配列の添字の順序を$(N/P^2, P_x, P_y)$から$(N/P^2, P_y, P_x)$に入れ替えるようにコピーする。次に、P_x個の MPI プロセス間における全対全通信をP_y組行う。

(2) 各MPIプロセスにおいて、配列の添字の順序を$(N/P^2, P_y, P_x)$から$(N/P^2, P_x, P_y)$に入れ替えるようにコピーする。次に、P_y個のMPIプロセス間における全対全通信をP_x組行う。

この2段階全対全通信アルゴリズムでは、P_x個またはP_y個のMPIプロセス間で全対全通信が2回行われるため、P個のMPIプロセス間で全対全通信を行う場合に比べて、トータルの通信量は2倍となる。ところが、全対全通信のスタートアップ時間はMPIプロセス数Pに比例するため、Nが比較的小さく、かつMPIプロセス数Pが大きい場合には、単純な全対全通信アルゴリズムに比べて2段階全対全通信アルゴリズムが有利になる場合がある。

そこで、$P = P_x \times P_y$となるような、すべてのP_xとP_yの組み合わせについて探索を行うことによって、最適なP_xとP_yの組み合わせを調べることができる。なお、MPI プロセス数Pが2のべきになる場合には、すべてのP_xとP_yの組み合わせを試行したとしても、探索空間は$\log_2 P$となる。全対全通信の自動チューニング手法を図21に示す。

```
min_time = DBL_MAX;
for (i = 0; i <= log2(P); i++) {
  Px = 2^i;
  Py = P/Px;
  MPI_Barrier(MPI_COMM_WORLD);
  start = MPI_Wtime();
  for (count = 0; count < ITER_NUM; count++) {
    if (Px == 1 || Py == 1)
      MPI_Alltoall(sendbuf,。.., recvbuf, ...);
    else
      Two-Step-Alltoall(sendbuf, ..., recvbuf, ..., Px, Py, ...);
  }
  MPI_Barrier(MPI_COMM_WORLD);
```

```
      end = MPI_Wtime();
      if (end - start < min_time) {
        min_time = end - start;
        Qx = Px;
        Qy = Py;
      }
    }
    Px = Qx;
  Py = Qy;
```

図 21　全対全通信の自動チューニング

性能評価にあたっては、並列 FFT ライブラリである FFTE 6.2alpha と、自動チューニング手法を FFTE 6.2alpha に適用したもの、そして FFTW 3.3.7 との性能比較を行った。$N = 2^m$のmを変化させて順方向 FFT を連続 10 回実行し、その平均の経過時間を測定した。なお、FFT の計算は倍精度複素数で行っている。

Xeon Phi クラスタとして、最先端共同 HPC 基盤施設（JCAHPC）に設置されている Oakforest-PACS（8208 ノード）の 1024 ノードを用いた。FFTE に対しては、コンパイラは Intel Fortran compiler 18.0.1.163 を用い、コンパイルオプションは“mpiifort -O3 -xMIC-AVX512 -qopenmp”を用いた。FFTW に対しては、コンパイラは Intel C compiler 18.0.1.163 を用い、コンパイルオプションは“mpiicc -O3 -xMIC-AVX512 -qopenmp”を用いた。MPI ライブラリは Intel MPI 2018.1.163 を用いた。各ノードあたりのスレッド数は 64、MPI プロセス数は 1 に設定し、環境変数“KMP AFFINITY=compact”を設定して flat/quadrant モードで MCDRAM のみを用いて実行した。$N = 2^m$点 FFT の GFlops 値は$5N \log_2 N$より算出している。図 22 に並列一次元 FFT の性能を示す。図 22 から、通信隠蔽および全対全通信の自動チューニングの効果により FFTE 6.2alpha（no overlap）、FFTE 6.2alpha（NDIV=4）や FFTW 3.3.7 よりも FFTE 6.2alpha with AT の性能が高くなっていることが分かる。

図 23 に全対全通信の性能を示す。通信メッセージサイズが 16B～128KB の範囲で、2 段階全対全通信アルゴリズムが選択されており、自動チューニングを行った全対全通信が MPI_Alltoall よりも通信バンド幅が高くなっていることが分かる。

並列一次元 FFT において、通信隠蔽と全対全通信の自動チューニングを組み合わせることで、性能をさらに向上させることができることを示した。

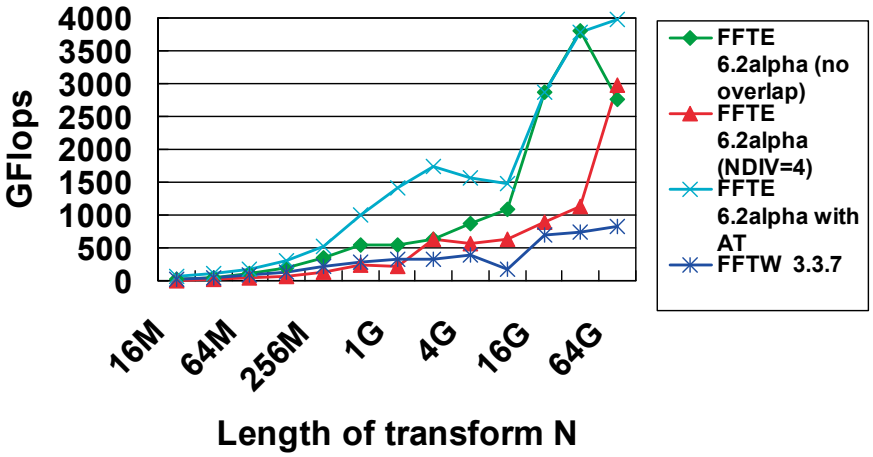


図 22　並列一次元 FFT の性能（Oakforest-PACS、1024 ノード）

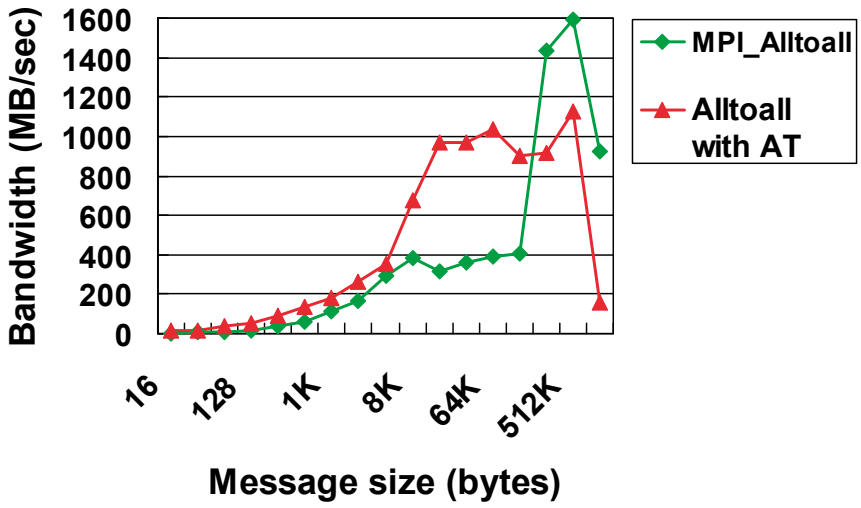


図 23　全対全通信の性能（Oakforest-PACS、1024 ノード）

【8】　Xeon Phi プロセッサにおける並列一次元実数 FFT の実現と評価（高橋）

FFT において入力データが実数である場合には、実数 FFT を用いると計算量および記憶領域を半分に減らすことができることが知られている。共有メモリ型並列計算機における並列実数 FFT として、FFTW や Intel MKL などが提案されている。本研究では、Xeon Phi プロセッサにおいて並列一次元実数 FFT を実現し性能評価を行った。

FFT は、離散 Fourier 変換（discrete Fourier transform、以下 DFT）を高速に計算するアルゴリズムとして知られている。

n点のデータに対する DFT は次式で定義される。

$$X_k = \sum_{j=0}^{n-1} x_j \omega_n^{jk}, \qquad 0 \le k \le n-1 \quad (1)$$

ここで、$\omega_n^{jk} = e^{-2\pi i/n}, i = \sqrt{-1}$である。

DFT の入力データが実数の場合、$n/2$点の複素数 DFT を用いてn点の実数 DFT を計算できることが知られている。具体的には、

$$x_j = r_{2j} + i r_{2j+1}, \qquad 0 \le j \le n/2-1 \quad (2)$$

とおくと、以下のような実数 DFT が得られる。

$$R_k = X_k - \frac{1}{2}(X_k - \bar{X}_{n/2-k})(1 + i\omega_n^k) \quad (3)$$

$$\bar{R}_{n/2-k} = \bar{X}_{n/2-k} + \frac{1}{2}\left(X_k - \bar{X}_{n/2-k}\right)\left(1 + i\omega_n^k\right), \qquad 1 \le k \le n/4-1 \quad (4)$$

$R_0 = \mathrm{Re}(X_0) + \mathrm{Im}(X_0)$

$R_{n/2} = \mathrm{Re}(X_0) - \mathrm{Im}(X_0),\ R_{n/4} = X_{n/4}$

six-step FFT アルゴリズムは並列処理に向いていることが知られているが、行列の転置が 3 回必要になる。図 24 に示すように行列の転置においてキャッシュブロッキングを行った場合、最内側のループのみを並列化すると、Xeon Phi 7250 における 68 コアの並列性を活用できない場合が存在する。具体的には、$n = 2^{18}$点複素数 FFT の場合に N1=N2=512、ブロッキングサイズを NB=8 とすると、最外側ループの反復回数は 64 回となるため、68 コアの並列性を活かすことができなくなる。

そこで、外側から 2 番目のループまでを OpenMP 3.0 の COLLAPSE(2)指示行により並列化することで、並列性を$64^2 = 4096$に増やすことが可能になる。n点の実数データに対して$n/2$点の six-step FFT を計算することでn点の実数 FFT を実現した。

```
SUBROUTINE TRANSPOSE(X,Y,N1,N2)
      PARAMETER (NB=8)
      COMPLEX*16 X(N1,N2),Y(N2,N1)
!$OMP PARALLEL DO COLLAPSE(2) PRIVATE(I,J,JJ)
      DO II=1,N1,NB
        DO JJ=1,N2,NB
          DO I=II,MIN(II+NB-1,N1)
            DO J=JJ,MIN(JJ+NB-1,N2)
```

```
            Y(J,I)=X(I,J)
          END DO
        END DO
      END DO
    END DO
  END DO
```

図 24　キャッシュブロッキングを行った行列の転置

In-cache FFT においては、基数 8 および 16 の FFT カーネルを組み合わせて使用した。基数 16 の FFT カーネルよりもメモリアクセス回数の多い、基数 8 の FFT カーネルを最大 3 ステップに抑えることにより、さらにメモリアクセスを削減することが可能である。Xeon Phi プロセッサの各コアは 32 個の zmm レジスタ（512 ビット）を持っているため、基数 16 の FFT カーネルにおいてもテンポラリ変数の多くをレジスタ上に保持することができる。

性能評価にあたっては、Xeon Phi における並列一次元実数 FFT を実現した FFT ライブラリである FFTE（version 6.2alpha）と、FFTW（version 3.3.6-pl1）、そして Intel MKL（version 2017 Update 1）との性能比較を行った。FFTW および Intel MKL は AVX-512 命令をサポートしている。$n = 2^m$のmを変化させて順方向 FFT を連続 10 回実行し、その平均の経過時間を測定した。なお、FFT の計算は倍精度実数で行っている。

評価環境として、Intel Xeon Phi 7250（MCDRAM 16GB+DDR4-2400 96GB、1.4GHz、68 core）を用いた。コンパイラは Intel Fortran compiler 17.0.1.132 を用い、コンパイルオプションは“ifort -O3 –xMIC-AVX512 -qopenmp”を用いた。Xeon Phi プロセッサあたりのスレッド数は 272 に設定し、環境変数“KMP AFFINITY=granularity=fine, balanced”を設定して flat/quadrant モードで MCDRAM のみを用いて実行した。

図 25 および図 26 に FFTE と FFTW、そして MKL の性能を示す。ここで、実行時間の単位は秒であり、$n = 2^m$点実数 FFT の計算量を$2.5n \log_2 n$として GFlops 値を算出している。図 25 および図 26 から、$n = 2^{22}$、$2^{24} \leq n \leq 2^{25}$および$n = 2^{29}$において FFTE の方が MKL よりも高い性能を発揮していることが分かる。

six-step FFT アルゴリズムに出現する行列の転置において外側ループの並列性を高くすることで、高い性能を得ることができた。

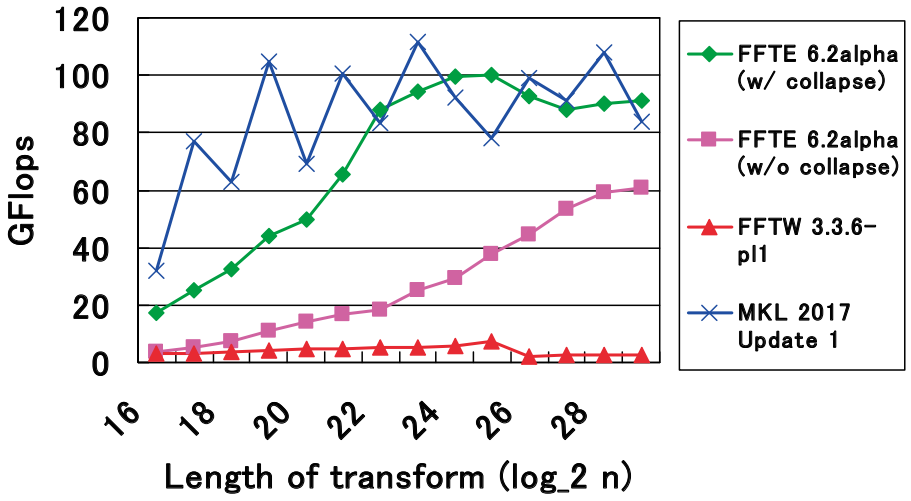


図 25　並列一次元実数 FFT の性能（Xeon Phi 7250、272 スレッド）

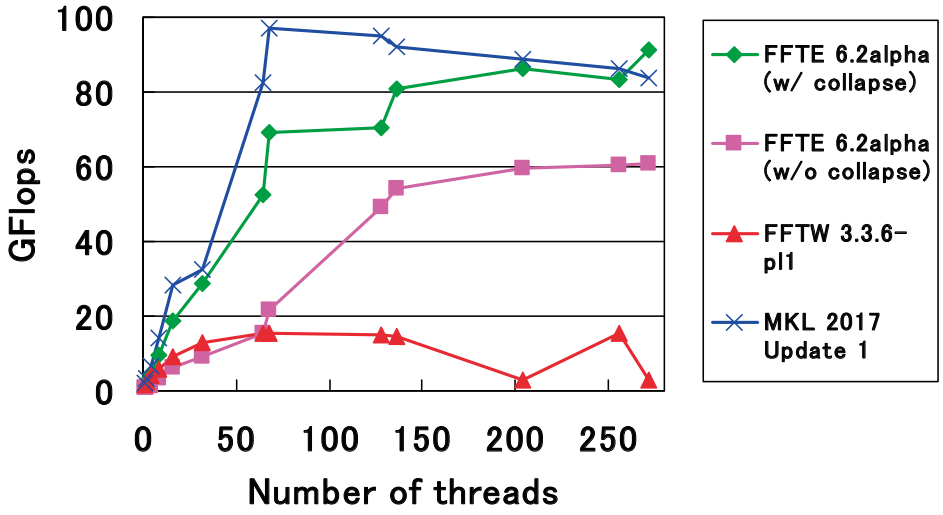


図 26 並列一次元実数 FFT の性能（Xeon Phi 7250、$n = 2^{29}$）

【9】　数学定数の特定の桁を計算する BBP 型公式の高速計算法（高橋）

πのような数学定数のn桁目の数字だけを計算することは、最初のn桁をすべて計算するよりも簡単ではないと広く信じられていた。ところが、1995 年に発見された BBP（Bailey-Borwein-Plouffe）型公式により、いくつかの超越数のn桁目の数字だけをさまざまな基数で計算できることが示された。BBP 型公式は数百桁以上の多倍長精度演算が不要（128 ビット程

度の精度で十分）であり、容易に実装できる。またメモリをほとんど必要としないという特長がある。

BBP 公式は以下の式で表される。

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right) \quad (1)$$

16 進$n+1$桁目から始まるπの数桁を計算することを考える。これは$\{16^n\pi\}$（ここで$\{x\} \equiv x - \lfloor x \rfloor$とする）を求めることと等価になる。式(1)から以下の式が得られる。

$$\{16^n\pi\} = \left\{\{16^n S(8,1,2)\} - \{16^n S(2,1,-1)\} - \{16^n S(8,5,0)\} - \{16^n S(4,3,-1)\}\right\} \quad (2)$$

ここで、

$$S(m,j,l) = \sum_{k=0}^{\infty} \frac{2^l}{2^{4k}(mk+j)} \quad (3)$$

である。また以下の式が得られる。

$$\{16^n S(m,j,l)\} = \left\{\left\{\sum_{k=0}^{n} \frac{2^{4(n-k)+l}}{mk+j}\right\} + \sum_{k=n+1}^{\infty} \frac{2^{4(n-k)+l}}{mk+j}\right\}$$

$$= \left\{\left\{\sum_{k=0}^{n} \frac{2^{4(n-k)+l} \bmod (mk+j)}{mk+j}\right\} + \sum_{k=n+1}^{\infty} \frac{2^{4(n-k)+l}}{mk+j}\right\} \quad (4)$$

式(4)の 1 番目の総和における分子において$\bmod (mk+j)$となっているのは、小数部分だけを計算すればよいためである。また、BBP 型公式のビット演算量は$O(n \log n\, M(\log n))$であることが知られている。ここで$M(d)$はdビットの乗算の演算量である。

Algorithm 1 右向きバイナリ法

Input: a, e, N positive integers

Output: $x = a^e \bmod N$

1: let $(e_l e_{l-1} \cdots e_1 e_0)$ be the binary representation of e, with $e_l = 1$
2: $x \leftarrow a$
3: **for** i **from** $l-1$ **downto** 0 **do**
4: $\quad x \leftarrow x^2 \bmod N$
5: $\quad$ **if** $e_i = 1$ **then**
6: $\quad\quad x \leftarrow ax \bmod N$
7: **return** x.

BBP 型公式における主要な計算は、式(4)のべき剰余$2^{4(n-k)+l} \bmod (mk+j)$である。べき剰余を計算する右向きバイナリ法を Algorithm 1 に示す。式(4)のべき剰余$2^{4(n-k)+l} \bmod (mk+j)$は正確に計算する必要がある。IEEE 754 の 128 ビット浮動小数点演算を用いた場合、16 進桁数nの上限は$(8n+5)^2 < 2^{113}$より$n = \lfloor \sqrt{2} \cdot 2^{53} \rfloor - 1 \approx 1.27 \times 10^{16}$となる。一方、64 ビット×64 ビット→128 ビットの符号なし整数演算を用いた場合、16 進桁数の上限は$(8n+5)^2 < 2^{128}$より$n = 2^{61} - 1 \approx 2.31 \times 10^{18}$となる。本研究では、IEEE 754 の 128 ビット浮動小数点演算よりも一般的に高速である 64 ビット×64 ビット→128 ビットの符号なし整数演算をべき剰余の計算に用いた。

Algorithm 1 における乗算剰余$x^2 \bmod N$および$ax \bmod N$に対しては、Montgomery 乗算を用いることで時間の掛かる除算を実質的に行うことなく、乗算、加減算およびシフト演算のみで乗算剰余を計算することができる。

また、式(4)の各項の値の範囲は$[0, 1)$となるため、除算と総和の計算は固定小数点演算で行うことができる。式(4)の各項における除算を 128 ビットの精度で行う場合、192 ビットを 64 ビットで割る符号なし整数除算$\lfloor (2^{128} \cdot x)/N \rfloor$を行う必要がある。もし、剰余$(2^{128} \cdot x) \bmod N$の値が事前に分かっていれば、この除算は Jebelean によって提案された exact division アルゴリズムを用いることで高速に行うことができる。Algorithm 2 に exact division アルゴリズムに基づく 192 ビットを 64 ビットで割る符号なし整数除算を示す。4 行目の **umulh** 関数は 64 ビット×64 ビット→128 ビットの符号なし整数乗算の上位 64 ビットを返す。

Algorithm 2 Exact division アルゴリズムに基づく 192 ビットを 64 ビットで割る符号なし整数除算

Input: x, N, r, μ such that $0 \le x < N,\ 0 < N < 2^{64},\ 2 \nmid N,\ r = (2^{128} \cdot x) \bmod N,$
$\mu = N^{-1} \bmod 2^{64}$

Output: $q = \lfloor (2^{128} \cdot x)/N \rfloor$

1: **if** $r = 0$ **then**
2: **return** 0
3: $q_0 \leftarrow (-r \cdot \mu) \bmod 2^{64}$
4: $q_1 \leftarrow [\{(2^{64} - 1) - \mathbf{umulh}(N, q_0)\} \cdot \mu] \bmod 2^{64}$
5: $q \leftarrow q_1 \cdot 2^{64} + q_0$
6: **return** q.

なお、式(4)の各項は独立であるため、複数のべき剰余と複数の整数除算に対して SIMD 化および並列化を行うことができる。式(4)において総和を計算する部分では、OpenMP の reduction 指示節を用いて並列化を行った。

性能評価にあたっては、提案手法に基づくプログラムと、Bailey によるプログラム（piqpr8.f、piqpr16.f）を用いて、BBP 公式によるπの 16 進 1000 万桁目の計算時間を比較した。提案手法では 128 ビット符号なし整数演算を用いており、SIMD 化を行った場合には 16 進桁数nの上限は$n \approx 5.76 \times 10^{17}$、SIMD 化を行わない場合には 16 進桁数$n$の上限は$n \approx 2.31 \times 10^{18}$である。Bailey によるプログラム piqpr8.f では 64 ビット浮動小数点演算を用いており、16 進桁数nの上限は$n \approx 1.19 \times 10^{7}$である。piqpr16.f では IEEE 754 の 128 ビット浮動小数点演算を用いており、16 進桁数nの上限は$n \approx 1.27 \times 10^{16}$である。

評価環境として、Intel Xeon E5-2670 v3（Haswell-EP 2.3GHz、12 コア）および Intel Xeon Phi 7250（Knights Landing 1.4GHz、68 コア）を用いた。提案手法に基づくプログラムにおいては、コンパイラは Intel C compiler 17.0.1.132 を用い、コンパイルオプションは“icc -O3 -xHOST -qopenmp”を用いた。Bailey によるプログラムにおいては、コンパイラは Intel Fortran compiler 17.0.1.132、コンパイルオプションは“ifort -O3 -xHOST -free”を用い、SIMD 化および並列化は行われていない。Xeon Phi 7250 における環境変数は“KMP AFFINITY=granularity=fine, balanced”を設定して flat/quadrant モードで MCDRAM のみを用いて実行した。

表 1　BBP 公式によるπの 16 進 1000 万桁目の計算時間（秒）

	Xeon E5-2670 v3 1 コア (1 スレッド)	Xeon E5-2670 v3 12 コア (24 スレッド)	Xeon Phi 7250 1 コア (1 スレッド)	Xeon Phi 7250 68 コア (272 スレッド)
提案手法（SIMD 化あり）	4.099	0.305	4.328	0.192
提案手法（SIMD 化なし）	6.847	0.423	23.568	0.375
piqpr8.f	25.781	N/A	96.053	N/A
piqpr16.f	440.833	N/A	1402.138	N/A

BBP 公式によるπの 16 進 1000 万桁目の計算時間を表 1 に示す。表 1 より Bailey によるプログラムよりも提案手法に基づくプログラムが高速であることが分かる。その主な理由としては、

・Bailey によるプログラムでは、べき剰余における剰余演算および級数の各項における除算を、64 ビット（piqpr8.f）または 128 ビット（piqpr16.f）浮動小数点演算で行っている。

・提案手法に基づくプログラムでは、べき剰余における剰余演算を Montgomery 乗算で行うと共に、級数の各項における除算を exact division アルゴリズムに基づく 128 ビット符号なし整数除算で行っている。

が挙げられる。

また、提案手法において、SIMD 化による性能向上は Xeon E5-2670 v3 の 1 コア（1 スレッド）で約 1.67 倍、Xeon Phi 7250 の 1 コア（1 スレッド）で約 5.45 倍となった。SIMD 化なしの場合は 64 ビット×64 ビット→128 ビットの符号なし整数乗算命令（x86_64 の mul 命令）が使えるのに対して、SIMD 化ありの場合は 32 ビット×32 ビット→64 ビットの符号なし整数乗算命令（AVX2 および AVX-512 の vpmuludq 命令）しか使えないのが主な理由である。

BBP 型公式における主要な計算である、べき剰余の計算を Montgomery 乗算で行うと共に、級数の各項における除算を exact division アルゴリズムに基づく 128 ビット符号なし整数除算で行うことで高速化した。また、複数のべき剰余と複数の整数除算に対して SIMD 化および並列化を行うことができることを示した。

【10】 エクストリームビッグデータの基盤技術（建部，川島）

エクストリームビッグデータ（EBD）アプリケーションの実行に求められる、数万～数十万プロセスからの並列アクセスを想定した IOPS、プロセス数に比例した読込、書込アクセスバンド幅性能を目標として、分散オブジェクトストアの研究を行っている。本年度は、共同研究者である東京工業大学の秋山グループと、メタゲノムアプリケーションについてストレージシステム、実行時システムのコデザインを進めた。メタゲノムアプリケーションでは、シーケンシングの対象となるクエリデータサイズが大きくなることが想定されており、大規模な解析のためにはデータベースファイルだけではなく、クエリファイルも分割して処理することが求められている。一方で、スパコンの並列ファイルシステムをアクセスしながらのデータ解析は、ファイルアクセス性能がボトルネックとなり効率が悪いことが分かっている。この問題を解決するために、計算ノードのローカルストレージを利用するデータ解析を検討した。ローカルストレージを用いることにより、計算ノード数に応じたファイルアクセス性能を達成することができるため、ファイルアクセス性能のボトルネックを解消できると考えられる。メタゲノムアプリケーションでは、クエリファイル、データベースファイルのすべての組み合わせでシーケンシングを行う必要がある。クエリファイルを n 分割、データベースファイルを m 分割したとき、$n \times m$ のシーケンシングが必要である。また、それぞれのシーケンシング処理は処理時間が均等ではない。ローカルストレージのデータ局所性を利用し、また負荷分散を均等にすることを目標に、Gfarm ファイルシステムと Pwrake ワークフローシステムを用いて、メタゲノムアプリケーションのワークフローの設計を行った。

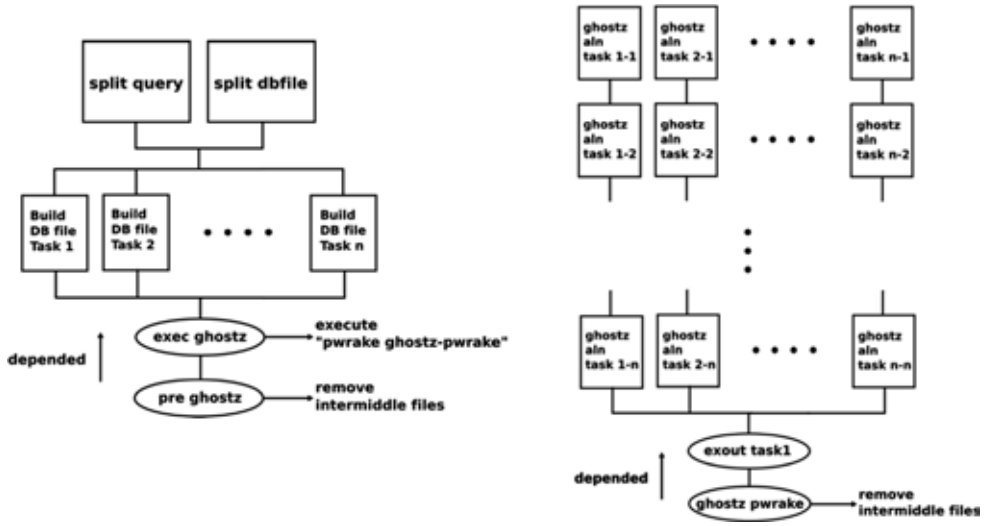


図 27　メタゲノムアプリケーションのワークフロー

図 27 に設計したワークフローを示す。左側はクエリファイルとデータベースファイルを分割して、データベースファイルをアプリケーションの入力形式に変換し、計算ノードのローカルストレージに格納するワークフローで、右側は分散して格納されたクエリファイルとデータベースファイルをシーケンシングするワークフローである。シーケンシングについてはタスクの依存関係はない。Pwrake ワークフローシステムを用いることにより、データの局所性を利用し、アイドルな計算ノードがないようにスケジューリングされる。シーケンシングでは入力ファイルが二つあるため、データの局所性の利用にはいくつかのパターンが考えられる。実験したところ、データベースファイルの局所性を優先した方がいいことが分かったため、Pwrake のワークフローにおいてデータベースファイルの局所性を優先させるようにした。

ワークフローのプロトタイプ実装を行い、7 ノードまで計算ノードを増やしていった時の実行時間を図 28 に示す。クエリファイルのサイズは 2 GB、データベースファイルのサイズは 5 GB である。

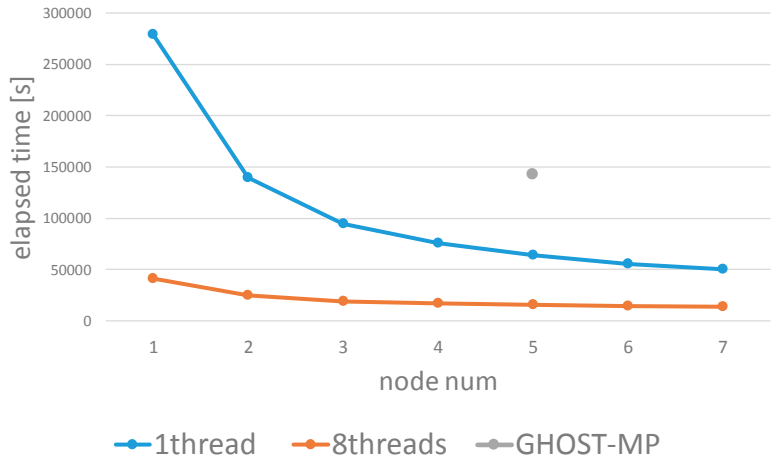


図 28　ノード数を変えた時のメタゲノムアプリケーションの実行時間

図 28 によるとノード数を増やすに従い、ほぼスケーラブルな性能を示している。GHOST-MP は MPI を用いた並列プログラムであり、8 スレッド利用した場合は 8.8 倍の高速化を達成した。今後、より詳細に性能を調べてさらなる高速化を行う予定である。

また、計算ノードのストレージは NVMe SSD が利用されることが多くなった。NVMe SSD は、SAS、SATA の SSD および HDD に比べ性能が向上している。計算ノードのローカル NVMe SSD を用いて、メタゲノムアプリケーションなどをより高速に実行するための分散ファイルシステムの研究開発を行った。このとき、アクセス性能の向上とメタデータ性能の向上が課題となる。アクセス性能については InfiniBand の RDMA を利用し、メタデータ性能については冗長性、永続性を犠牲にすることで性能向上を図った。計算ノードのローカル NVMe SSD は、計算ノードがジョブに割り当てられ、実行中しか利用することができない。そのため、ジョブ実行中の一時的な利用しかできないため、永続性、冗長性は大きな問題とはならない。永続性をなくしたところ、メタデータ性能は 13.6 倍となった。また、ファイルシステムの構築、撤去時間も減らすことができ、構築に 0.31 秒、撤去に 0.19 秒であった。今後、より大規模な構成において評価を続け、アプリケーションによる評価を行っていく予定である。

【11】　極端気象予測を拓くビッグデータ機械学習基盤の研究（建部、川島）

豪雨・突風・高温などの極端気象は人類に甚大な被害をもたらすが、その予測は極端気象に関する膨大な知識が必要である。本研究では、その知識を効率的に生成する機械学習基盤の構築を目的とする。平成 29 年度は、極端気象の分析手法について主に研究を進めた。極端気象として、まずフェーン現象をターゲットとし、過去の気象データを用い、典型的なフェーン日を正解として、機械学習を試みた。その結果、これまで経験則でフェーン日の条件と思われていたことが、ほぼフェーン日の判定条件となっていることが分かった。さらに、機

械学習ではフェーンらしさについて数値的に求めることも可能であり、定量的な評価の可能性も示した。今後は、得られた機械学習結果により、フェーン日の予測につなげていく。また、並列機械学習のため、R で記述されたエントロピー推定プログラムの MPI 並列化を行った。エントロピー推定には Simple Regression Entropy Estimator、Direct Regression Entropy Estimator、Entropy Estimator with Poisson noise structure Identity-link regression、Kernel Density Estimation の 4 種類を用いた。元データを全プロセスで保持する場合はほぼスケーラブルな性能向上を示したが、データを分散配置するとプロセス数を増やすと計算時間は減少するものの、通信時間が増大してしまい、計算時間と通信時間にはトレードオフがあることが分かった。これまでの逐次アルゴリズムは並列化効率が良くないため、今後、より効率的な並列実行が可能な並列アルゴリズムの研究を進める必要がある

【12】 分散ファイルシステム及びグリッド・クラウド技術に関する研究（建部）

文部科学省が進める革新的ハイパフォーマンスコンピューティングインフラ（HPCI）の HPCI 共用ストレージ、素粒子物理学データ共有システム JLDG のシステムソフトウェアとしても利用される Gfarm ファイルシステムの研究開発を行った。本年度は、データ移行支援、書込キャッシュストレージ支援、メタデータサーバのレスポンス改善を行った。データ移行支援では、運用中のストレージの移行を柔軟に行うために、さまざまな要求に応えるように機能拡張を行った。新規ストレージノードの導入にあたり、新規ストレージノードにはファイル複製のみ作成し、新規ファイルを作成したくないという要求があった。この要求を実現するため、仮想的に新規ストレージノードの負荷を上げ、新規ファイルが作成されないようにした上で、負荷が高くても複製は作成できるようなスイッチを準備した。また、ファイル移行は複製の場所指定により行うが、ファイル移行を一時的に停止したい要求があった。そのため、動的に複製の場所指定を有効・無効とするスイッチを動的に行えるようにした。また、ファイル移行の速度を調節するため、動的に最大並列複製作成数を調節できるようにした。書込キャッシュストレージ支援では、高速で信頼性の高いストレージを一時的な書込キャッシュとして利用するための機能拡張を行った。まず、書込キャッシュストレージに書込みを行うため、書込み先をキャッシュストレージに指定するか、それ以外のストレージの負荷を高くする。さらに、キャッシュストレージにファイルを保持する期間、キャッシュ容量を指定可能とした。メタデータサーバのレスポンス改善は、大量ファイル削除、大量ファイル複製作成、ファイル更新、複製場所指定時の複製チェック動作時などにおいて、メタデータサーバのレスポンスが悪くなり、たまに数秒〜10 数秒となることがあることに対する改善である。問題を追及した結果、mutex ロック期間が長くなると、ロックの公平性がなくなることが原因であった。そのため、ロックを要求した順番にロックの確保を行う ticket lock を実装し、解決を図った。Ticket lock に変更したところ、メタデータサーバのレスポンスの悪化は生

じなくなった。これらの成果は平成 30 年 3 月 22 日にリリースした Gfarm バージョン 2.7.10 に含まれている。

【13】 高性能な MapReduce Shuffle に関する研究（川島、建部）

分散処理フレームワーク MapReduce における All-to-All 集団通信フェーズである shuffle に関して、“skew”と呼ばれる MapReduce 特有の load imbalance 問題に対処可能な手法の検討を行った．初めに shuffle に適した Point-to-Point 通信モデルに関して MPI との処理モデル等の違いを通して考察し、モデルの実装例としてネットワーク帯域を最大限活用可能な socket ベースの実装を提示した．次に上記の Point-to-Point 通信モデルを用いた All-to-All 通信の構成法に関して、skew 耐性の観点から議論した．本研究では shuffle 対象のデータ本体（block）とそのメタ情報（meta-block）を単一の All-to-All 通信で交換する CSA （Coupled Shuffle Architecture）、block と meta-block をそれぞれ別の All-to-All 通信によって交換する DSA（Decoupled Shuffle Architecture）、DSA において各プロセスの消費メモリサイズを考慮して block の配置を決定する DSA w/ SMS （Skew-aware Meta-Shuffle）、の 3 手法の提案及び比較を行った．CSA では Pairwise 方式、DSA では Naive 方式の All-to-All 通信アルゴリズムをそれぞれ block の shuffle に採用した．上記 3 手法の評価のため、本研究では独自 in-memory MapReduce 処理系を C/C++で実装し、InfiniBand に代表される高性能インターコネクトの性能を最大限活用するため socket/verbs/ofi （libfabric） の 3 種類の通信方式に対応させた．評価実験では上記 3 手法を用いて skew 度合いを変えながら shuffle ワークロードを実行し、DSA w/ SMS 方式のみが極度の skew 状況下でも in-memory で shuffle を実行できることを確認した．また CSA、DSA の 2 手法の weak scaling 性能の比較を通して、skew 度合いに応じた両方式の挙動を詳細に調査した．その結果、低 skew 状況下ではネットワーク帯域をより有効に活用できる CSA 方式が優位となり、1024 プロセスでは DSA 比で 1.65 倍高速であった．また、高 skew 状況下では両手法共に straggler によって shuffle 全体の実行が律速されることが明らかになった．

【14】 高性能な並行性制御に関する研究（川島、建部）

トランザクション処理システムは並行性制御法とログ書き込み法から構成される．近年のいくつかの研究においては並行性制御法に着目し、その高性能化に取り組んだ例が存在する．トランザクション処理システムの要素技術として様々な選択肢が存在するため、課題の解決に当たって適した並行性制御法およびログ書き込み法を採用する必要がある．

メニーコア環境でトランザクション処理を高性能化するには、その並列化を高度化する必要がある．トランザクション処理ではデータオブジェクトへの更新アクセスにロックが必要であり、そのアクセス手法は悲観的手法と楽観的手法に分類される．悲観的手法はデータオ

ブジェクトへのアクセス毎にロックを獲得する一方、楽観的手法のロック獲得はコミット判定時のみに限られる．これにより近年の楽観的手法は悲観的手法に対して優位性を示している．楽観的手法にログ書き込み法を組み合わせるにはタイムスタンプ発行が必要になる．このタイムスタンプ発行に集中カウンタを用いると大幅な性能劣化が発現することが報告されている．本研究ではタイムスタンプの発行を非集中化して分散並列発行することで、優れた性能を示す楽観的手法 TicToc を対象に選定した．

ログ書き込み法にも様々な手法が存在する．NVRAM やフラッシュデバイスなどを前提とした passive group commit、P-WAL、SiloR がある．また、近年のトランザクションシステム研究では、NVRAM やフラッシュデバイスを前提とするものが多数ある．そのため、ログ書き込み法は NVRAM やフラッシュデバイスを対象とした手法を用いることが有用であると考えられる．これらの手法は並列ログ書き込み法という手法に分類される．並列ログ書き込み法とは、ログの永続化を複数のワーカースレッドが並列に行う方式であり、前述の passive group commit と P-WAL が該当する．そこでログ書き込み法には、並列ログ先行書き込み法である P-WAL を採用した．

TicToc と P-WAL を組み合わせた環境において、グループコミットにおける一括書き込みサイズ（トランザクション数）を変動させた場合の実験を行った．その結果、グループコミットにおける一括書き込みサイズが増える程、トランザクション処理性能が劣化していることが観察された．すなわち、近年のハードウェアならびに並行性制御法・ログ書き込み法の下では、グループコミットはトランザクション処理の敵となり得る知見を得た．

【15】 Block BiCGSTAB 型解法における近似解精度と数値的安定性の改善に関する研究（多田野）

複数本の右辺ベクトルをもつ連立一次方程式の数値解法である、ブロッククリロフ部分空間反復法に関する研究を実施した。ブロッククリロフ部分空間反復法を用いることで、クリロフ部分空間反復法を用いた場合よりも少ない反復回数で同方程式の近似解が得られることがある。しかしながら、右辺ベクトル数が多い場合は、数値的不安定性の影響で残差ノルムの発散・停滞、及び近似解精度劣化が発生することがある。

今年度は、Block BiCGSTAB 型解法における近似解精度と数値的安定性の改善について研究を行った。Block BiCGSTAB 型解法の残差行列は、Block BiCG 法の残差行列と、安定化多項式と呼ばれる行列多項式の積で表される。漸化式の構築の仕方によって、Block BiCGSTAB 法と Block BiCGGR 法を導出できる。Block BiCGSTAB 法は右辺ベクトル数が増加すると得られる近似解精度が悪化する欠点がある。一方、Block BiCGGR 法は Block BiCGSTAB 法で起こり得る近似解精度劣化を抑えるように漸化式を構築した解法である。また、右辺ベクトル数が多い場合は両解法とも数値的不安定性の影響で、残差が発散、または停滞することがある。

反復過程で現れる残差行列の正規直交化を行うことで、この数値的不安定性を緩和することができる。図 29 に、残差行列の正規直交化を行ったときの Block BiCGSTAB 型解法の反復回数と真の相対残差の変化を示す。真の相対残差は近似解精度を検証する指標の一つで、この値が小さければ高精度の近似解が得られていることを示す。テスト問題として、格子量子色力学計算（QCD）で現れる連立一次方程式を用いた。同方程式のサイズは 1,572,864, 非零要素数は 80,216,064 である。係数行列内に現れるパラメータ κ は 0.1358 とした。

図 29 (a)に示すように、右辺ベクトル数 L が少ない場合は、両解法とも L の増加に伴って反復回数が減少した。しかしながら、Block BiCGGR 法では多数の右辺ベクトルをもつ場合は残差行列の正規直交化を行うだけでは数値的不安定性を解消できず、反復回数が増大することが分かった。また、図 29 (b)に示す真の相対残差に着目すると、Block BiCGGR 法では高精度の近似解が得られているが、Block BiCGSTAB 法では L の増加に伴い真の相対残差が大きくなっており、高精度近似解が得られていない。

Block BiCGGR 法において右辺ベクトル数が多い場合の振る舞いを調査した結果、反復過程で現れる小規模連立一次方程式の係数行列の条件数が非常に大きくなっていることが分かった。この小規模係数行列は反復過程で現れる矩形の補助行列から生成され、この補助行列を構成する列ベクトル間の線形独立性が失われると、小規模係数行列の条件数が増大する。そこで本研究では、Block BiCGSTAB 型解法の二解法において、残差行列の他に補助行列に対しても正規直交化を施すアルゴリズムを構築した。数値実験結果を図 30 に示す。

図 30 (a) に示すように、残差行列と補助行列の両方の正規直交化を行うことで、Block BiCGGR 法は右辺ベクトル数を増加させても反復回数が減少するようになった。また、Block BiCGSTAB 法では、補助行列の正規直交化を行うことで、真の相対残差を小さくすることができた。Block BiCGSTAB 法における近似解の精度劣化は、矩形の補助行列と小規模正方行列の積の計算で発生する誤差が原因であった。補助行列の正規直交化を行うことで、行列積計算で発生する誤差の拡大を抑えられたことが、近似解精度改善の要因である。

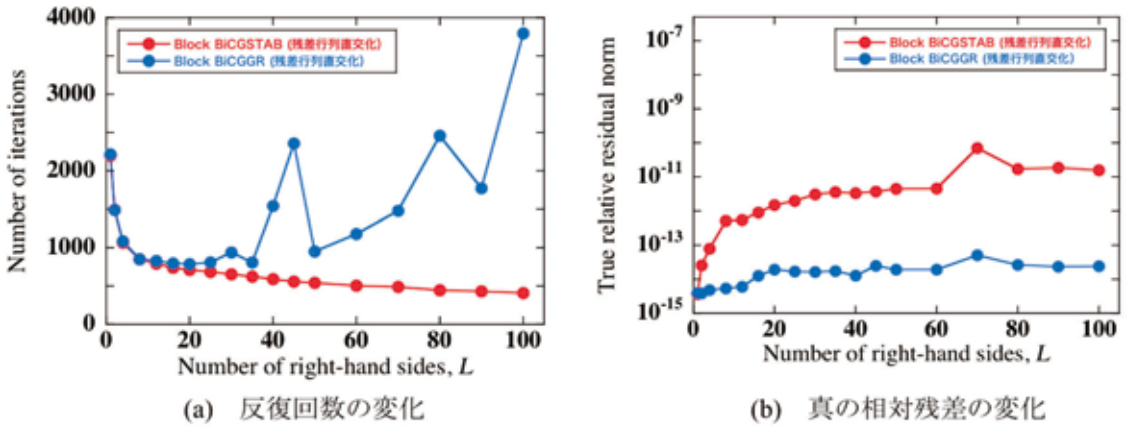


図 29 残差行列の正規直交化を組み込んだ Block BiCGSTAB 型解法の反復回数と真の相対残差の変化

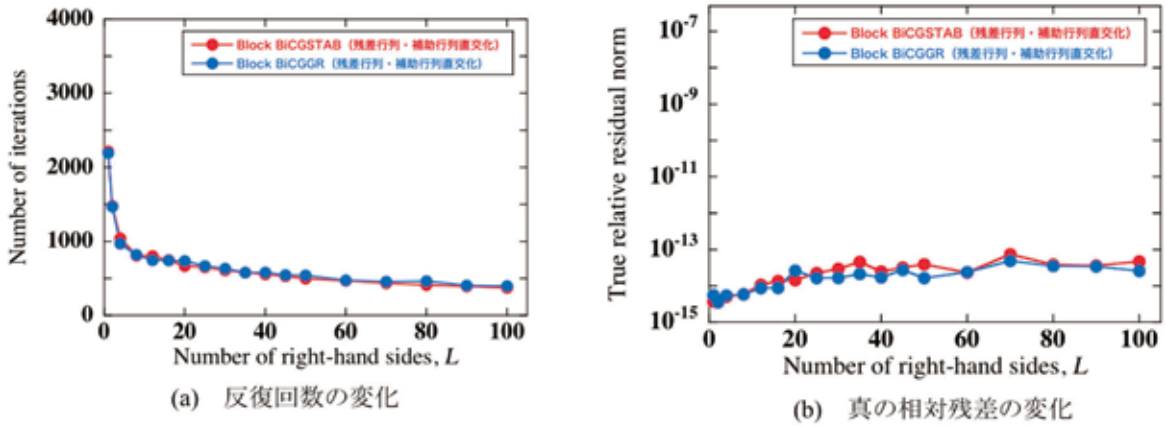


図 30 残差行列と補助行列の正規直交化を組み込んだ Block BiCGSTAB 型解法の反復回数と真の相対残差の変化

【16】 高収束かつ高精度近似解を生成するブロッククリロフ部分空間反復法の開発（多田野）

前項目で述べた Block BiCGSTAB 型解法では、近似解精度と数値的安定性を向上させるために、残差行列と補助行列の両方の正規直交化が必要であることが分かった。Block BiCGSTAB 型解法の安定化多項式のパラメータ数は 1 つであるが、本研究では 2 つのパラメータをもつ安定化多項式を導入し、高収束かつ高精度近似解が生成可能なブロッククリロフ部分空間反復法を構築した。この安定化多項式は、クリロフ部分空間反復法の一つである GPBiCG 法などでも用いられている。また、この提案法において、漸化式で求めた残差と真

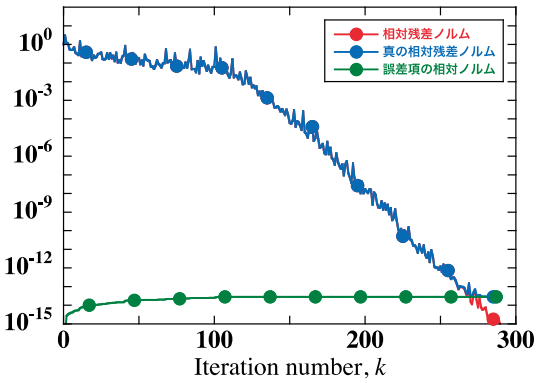


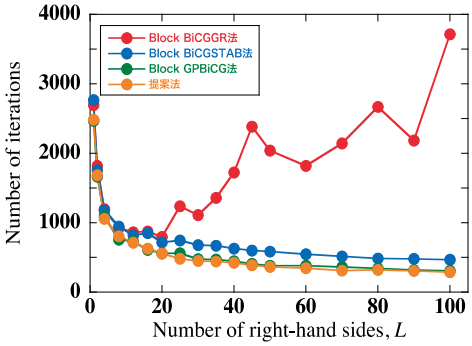
図 31　残差行列の正規直交化を組み込んだ提案法における相対残差ノルム、真の相対残差ノルム、及び誤差項の相対ノルムの履歴（右辺ベクトル数：100）

の残差の間に生じる誤差項の定式化を行った。提案法と同カテゴリに属する手法では、1 反復あたり 2 回の係数行列に関する積が必要であるが、提案法では高精度近似解を生成するために、1 反復あたり 3 回の係数行列に関する積が必要である。しかしながら、2 回分の積の計算は同時に実行することが可能であるため、計算時間の大幅な増加を抑えることができる。

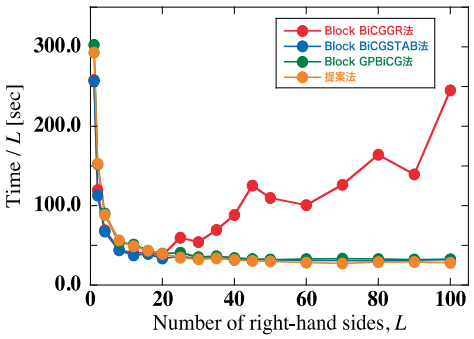
数値実験によって、提案手法の評価を行った。数値実験で用いたテスト問題は前項目と同じく格子 QCD 計算で得られる連立一次方程式で、係数行列のパラメータ κ は 0.1360 とした。実験環境は、CPU：Intel Xeon E5-2620 v3 2.4GHz×2、メモリ：64GByte、コンパイラ：gfortran ver. 5.4.0、コンパイルオプション：-O3 -fopenmp であり、OpenMP を用いて 12 スレッド並列で計算した。

図 31 に右辺ベクトル数が 100 本の場合の、残差行列の正規直交化を組み込んだ提案法の相対残差ノルム、真の相対残差ノルム、及び誤差項の相対ノルムの履歴を示す。相対残差ノルムは 290 回の反復で 10^{-15} を下回り、反復を停止した。このときの真の相対残差の値は、2.9×10^{-14} であり、相対残差ノルムとの開きはあるものの、高精度の近似解が得られた。また、このときの誤差項の相対ノルムの値も 2.9×10^{-14} であることから、本研究における誤差項の定式化で、誤差項が正しく表現できていることを確認した。

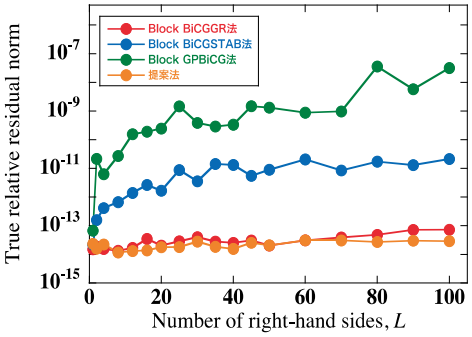
図 32 において、残差行列の正規直交化を組み込んだ Block BiCGGR 法、Block BiCGSTAB 法、Block GPBiCG 法、及び提案法について、反復回数、求解時間、真の相対残差の観点から比較を行う。図 32 (a)に示すように、Block BiCGGR 法を除いて、右辺ベクトル数の増加に伴い反復回数は減少する傾向が見られた。図 32 (b) に、右辺ベクトル数の変化に対する右辺ベクトル 1 本あたりの求解時間の変化を示す。提案法では 1 回反復あたり 3 回の係数行列に関する積が必要であるが、2 回分の積計算を同時に実行しているため、Block BiCGSTAB 法、Block GPBiCG 法と同程度の時間で求解することができた。図 32 (c) に、右辺ベクトル数の変化に対する真の相対残差の変化を示す。真の相対残差は提案法が最も小さい傾向にあり、



(a) 反復回数の変化



(b) 右辺ベクトル 1 本あたりの求解時



(c) 真の相対残差の変化

図 32 残差行列の正規直交化を組み込んだ 4 つの手法における反復回数、求解時間、真の相対残差の変化

高精度近似解を生成することができた。一方、Block BiCGSTAB 法と Block GPBiCG 法の真の相対残差は右辺ベクトル数の増加に伴い、大きくなる傾向にあり、高精度近似解が得られていない。

以上の結果より、提案法では残差行列の正規直交化を行うだけで、Block GPBiCG 法と同程度の収束性を示すことが分かった。また、提案法は Block BiCGGR 法と同程度以上の精度をもつ近似解を生成できることを確認した。しかしながら、提案法においても近似解精度に影響を及ぼす誤差項が存在しているため、この影響をさらに小さくすることは今後の課題となる。

【17】 差分圧縮を活用した FPGA ベースソーティングアクセラレータ（小林）

ソーティングはデータベースや画像処理、ゲノム解析など幅広い分野で利用される極めて重要な計算カーネルである。ソーティングの対象となるデータ量は、近年の目覚ましい IT 技術の進展に伴い爆発的に増大しており、その傾向は依然として続くことが予想される。そして、ここ数年の Internet of Things（IoT）時代の台頭により、我々の至る所にコンピュータが適用されるようになってきている。このため、将来においては、そのデータ量の増加に対応し、

かつ組み込みコンピュータから高性能サーバといった様々なプラットフォームにおいて高い実効性能を提供しうるソーティング技術の確立が必要である。本研究では、ソーティングネットワークとマージソートツリーを組み合わせた FPGA ベースのソーティングアクセラレータを提案する。提案するアクセラレータは PCIe bus を介して接続される。ソーティングの対象となるデータはホスト PC から送信され、アクセラレータでソートされた後、ホスト PC に返却される。図 33 に提案するアクセラレータの概要を示す。

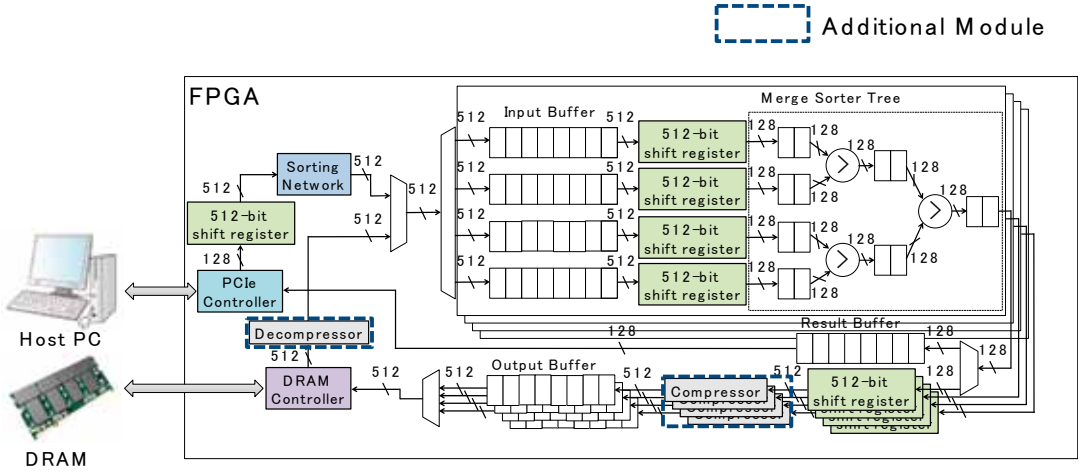


図 33　提案するソーティングアクセラレータの概要

このアクセラレータはマージソートツリーの葉の数などの構成パラメータを調節することで、ソーティング性能とハードウェアリソース使用量とを変化させた様々な構成にカスタマイズすることが可能である。そして、本研究では、アクセラレータの性能モデルも提案しているため、設計者はあらかじめソーティングアクセラレータの性能について試算でき、性能とハードウェアコストの観点から最適なソーティングアクセラレータを実装することができる。これが、本研究における第 1 の貢献である。

また、従来の FPGA ベースのソーティングアクセラレータではオフチップメモリ帯域によって、アクセラレータの動作効率が低減するという問題を抱えている。これに対処するために、本研究では、図 34 に示す差分圧縮アルゴリズムを活用したデータ圧縮手法を提案している。

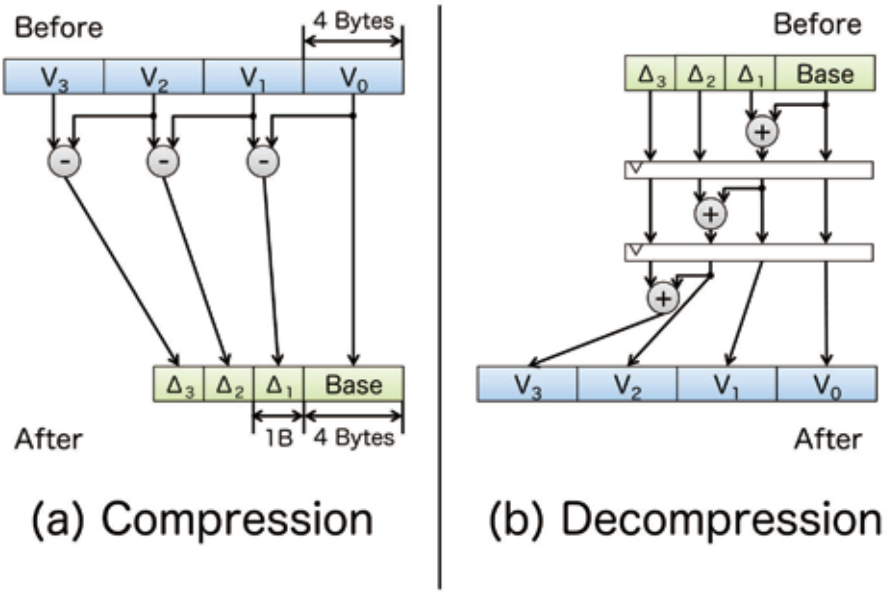


図 34　差分圧縮アルゴリズムを活用したデータ圧縮・展開手法

提案アクセラレータの動作が進行するにつれ、対象のデータセットは徐々にソートされていくため、要素間の差分が小さくなる。つまり、ソーティングが進行するにつれ、圧縮率は高くなり、その結果オフチップメモリ帯域を拡大するため、差分圧縮アルゴリズムとソーティングアクセラレータとの相性は非常に良い。提案した圧縮手法の効果を評価したところ、図 35 に示すようにハードウェアリソースをほぼ使用せずに動作効率を最大 54%改善し、アクセラレータの性能がメモリ帯域に制約される場合においては、提案するデータ圧縮手法が有効であることを明らかにしている。これが、本研究における第 2 の貢献である。

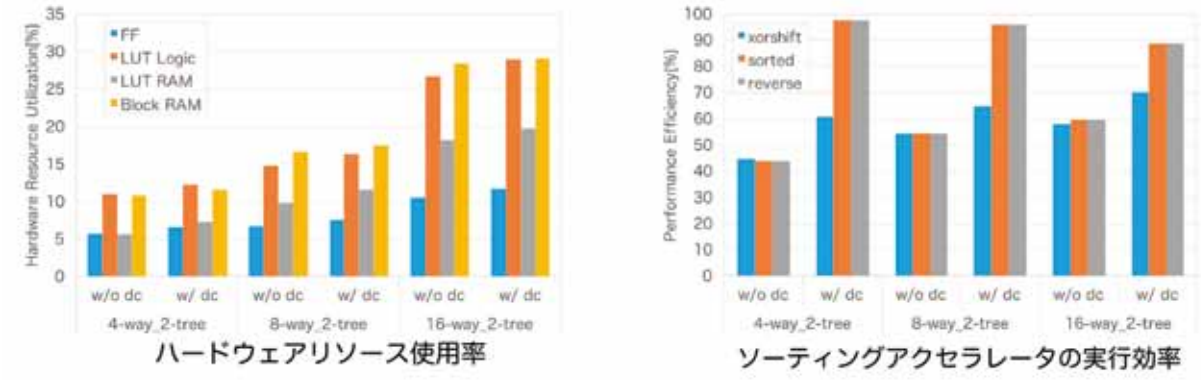


図 35　圧縮手法を適用した・適用しない場合におけるハードウェアリソース使用率とソーティングアクセラレータの動作効率

そして、全ての設計者が容易かつ自由に提案するアクセラレータを利用できるように、その RTL ソースコードをオープンソースにて公開している。高性能かつカスタマイズ可能であ

り、さらにメモリバンド幅について考慮しているオープンソースなソーティングアクセラレータはこれまでに存在していない。これが、本研究における第 3 の貢献である。

以上の貢献が認められ、本研究成果をまとめた論文が電子情報通信学会 情報・システムソサイエティ論文誌に採録された。

4. 教育

修士学位論文

1. 大畠佑真，修士（工学），FPGA における高速データ通信機能を用いた高性能並列処理に関する研究，筑波大学大学院システム情報工学研究科修士論文，平成 30 年 3 月（指導：朴泰祐）
2. 五味歩武，修士（工学），Julia プログラムのコード変換による性能チューニングフレームワーク，筑波大学大学院システム情報工学研究科修士論文，平成 30 年 3 月（指導：高橋大介）
3. 大黒晴之，修士（工学），高性能 MapReduce Shuffle 手法に関する研究，筑波大学大学院システム情報工学研究科修士論文，平成 30 年 3 月（指導：川島英之）
4. 村田直郁，修士（工学），高性能トランザクション処理システムに関する研究，筑波大学大学院システム情報工学研究科修士論文，平成 30 年 3 月（指導：川島英之）
5. 瀧沢亮太，修士（工学），分散計算による暗号化データ処理の高性能化に関する研究，筑波大学大学院システム情報工学研究科修士論文，平成 30 年 3 月（指導：川島英之）

卒業論文

1. 枝松拓弥，学士（工学），SIMD 命令を用いた多倍長整数乗算の高速化，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導：高橋大介）
2. 佐藤駿一，学士（工学），メニーコアプロセッサと GPU における疎行列ベクトル積の性能比較，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導：高橋大介）
3. 河合祐輔，学士（工学），pbdMPI を用いたエントロピー推定プログラムの並列化と性能評価，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導教員：建部修見）
4. 北澤昂大，学士（工学），クラウドストレージに関する性能評価と広域仮想ファイルシステムの設計，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導教員：建部修見）

5. 町田健太，学士（工学），Pwrake/Gfarm による分散並列相同性検索システムの提案，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導教員：建部修見）
6. 田村駿也，学士（工学），インメモリ B-link 木の設計と実装，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導：川島英之）
7. 堀江悠樹，学士（工学），耐ビザンチン障害性を持つ分散合意手法 PBFT の設計と実装，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導：川島英之）
8. 田辺敬之，学士（工学），多版法・楽観法・分散時刻印に基づく並行性制御法と並列ログ先行書込み法の結合，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導：川島英之）
9. 倉本亮世，学士（工学），複数右辺・複数シフトを持つ線形方程式に対する Shifted Block BiCGSTAB 法の近似解精度改善，筑波大学情報学群情報科学類卒業論文，平成 30 年 3 月（指導：多田野寛人）

集中講義

1. 計算科学のための高性能並列計算技術：大学院共通科目

5. 受賞、外部資金、知的財産権等

受賞

1. 大黒晴之, 川島英之, 建部修見, "大規模 MapReduce 実行環境向け Shuffle 通信の RDMA による高速化", The 1st. cross-disciplinary Workshop on Computing Systems, Infrastructures, and Programming (xSIG2017), Outstanding Presentation Award

外部資金

1. JST CREST 研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」, 研究課題名「ポストペタスケール時代に向けた演算加速機構・通信機構統合環境の研究開発」, H24～H29 年度, 23,710 千円(H29)（主たる研究者：朴泰祐）
2. 文部科学省高性能汎用計算機高度利用事業費補助金（次世代領域研究開発）課題名「次世代演算通信融合型スーパーコンピュータの開発」, H29～H33 年度, 13,000 千円 (H29)（研究代表者：朴泰祐）
3. 科学研究費補助金　基盤研究 (C)、高橋大介（代表）、H28～30 年度、1,430 千円（H28 年度）、「メニーコア超並列クラスタにおける有理数演算ライブラリに関する研究」

4. JST CREST 研究領域「ビッグデータ統合利活用のための次世代基盤技術の創出・体系化」，「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」，H25～H30 年度， 14,950 千円 (H29)（主たる共同研究者：建部修見）
5. 基盤研究(B)（一般），「極端気象予測を拓くビッグデータ機械学習基盤の研究」，H29 年度～H31 年度，5,590 千円 (H29)（研究代表者：建部修見）
6. JST CREST、川島英之（共同研究者）、H26～30 年度、16,860 千円（H29）、研究領域「科学的発見・社会的課題解決に向けた各分野のビッグデータ利活用推進のための次世代アプリケーション技術の創出・高度化」，「広域撮像探査観測のビッグデータ分析による統計計算宇宙物理学」
7. 科学研究費補助金　基盤研究 (C)、川島英之（代表）、H28～30 年度、910 千円（H29 年度）、「先進的デバイスの利活用による高性能データ基盤システムに関する研究」

知的財産権

（該当なし）

6. 研究業績

(1) 研究論文

A) 査読付き論文

1. Ryohei Kobayashi, Yuma Oobata, Norihisa Fujita, Yoshiki Yamaguchi, and Taisuke Boku, OpenCL-ready High Speed FPGA Network for Reconfigurable High Performance Computing, HPC Asia 2018 Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, pp.192-201, January 2018.
2. Masahiro Nakao, Hitoshi Murai, Hidetoshi Iwashita, Akihiro Tabuchi, Taisuke Boku and Mitsuhisa Sato: Implementing Lattice QCD Application with XcalableACC Language on Accelerated Cluster, Proc. of IEEE Cluster 2017, 9 pages, Hawaii, Sep. 8th, 2017.
3. Daisuke Takahashi, “An Implementation of Parallel 1-D Real FFT on Intel Xeon Phi Processors”, Proc. 17th International Conference on Computational Science and Its Applications (ICCSA 2017), Part I, Lecture Notes in Computer Science, Vol. 10404, pp. 401-410, Springer International Publishing, 2017.
4. Hiroyuki Takizawa, Daichi Sato, Shoichi Hirasawa and Daisuke Takahashi, “A Customizable Auto-Tuning Scenario with User-defined Code Transformations”, Proc. 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW

2017), The 12th International Workshop on Automatic Performance Tuning (iWAPT 2017), pp. 1372-1378, 2017.

5. 村田直郁, 川島英之, 建部修見, RDMA の適用による RAMP トランザクション処理の高速化, 情報処理学会論文誌データベース（TOD）, Vol. 10, No. 2, pp. 19-30, 2017
6. Fuyumasa Takatsu, Kohei Hiraga, Osamu Tatebe, “PPFS: A Scale-out Distributed File System for Post-petascale Systems”, Journal of Information Processing, Vol. 25, pp. 438-447, 2017 (DOI: 10.2197/ipsjjip.25.438)
7. Xieming Li, Osamu Tatebe, “Data-Aware Task Dispatching for Batch Queuing System”, IEEE Systems Journal, Vol. 11, Issue 2, pp. 889-897, 2017 (DOI: 10.1109/JSYST.2015.2471850)
8. Ryota Takizawa, Hideyuki Kawashima, Ryuya Mitsuhashi and Osamu Tatebe, “Performing External Join Operator on PostgreSQL with Data Transfer Approach”, Proceedings of HPC Asia 2018, 7 pages, 2018
9. Hiroto Tadano, Shusaku Saito, Akira Imakura, Accuracy Improvement of the Shifted Block BiCGGR Method for Linear Systems with Multiple Right-Hand Sides, Proc. International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing (EPASA2015), Lecuter Notes in Computational Science and Engineering, Vol. 117, pp. 171—185, 2017.
10. Shimpei Sato, Ryohei Kobayashi and Kenji Kise: ArchHDL: A Novel Hardware RTL Modeling and High-speed Simulation Environment, IEICE Transactions on Information and Systems, Vol.E101-E, No.2, pp.344-353 (February 2018).
11. Ryohei Kobayashi, and Kenji Kise: A High Performance FPGA-based Sorting Accelerator with a Data Compression Mechanism, IEICE Transactions on Information and Systems, Vol.E100-D, No.5, pp.1003-1015 (May 2017).

B) 査読無し論文

1. 廣川祐太, 朴泰祐, 植本光治, 佐藤駿丞, 矢花一浩: 電子動力学シミュレーション ARTED の KNL システム Oakforest-PACS での全系性能評価, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-160(20), 8 pages, 2017 年 7 月.
2. 松村和朗, 佐藤三久, 朴泰祐: フロー解析によるマルチ GPU 対応 OpenACC コンパイラ, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-160(22), 8 pages, 2017 年 7 月.

3. 辻美和子, 李珍泌, 朴泰祐, 佐藤三久: 疑似 MPI トレースプロファイルを用いた通信性能推定手法 SCAMP のための疑似トレースファイル作成手法の検討, 研究報告ハイパフォーマンスコンピューティング（HPC）,2017-HPC-161(14),1-10 (2017-09-12).
4. 中尾昌広, 村井均, 朴泰祐, 佐藤三久: Python と連携する PGAS 言語 XcalableMP のプログラミングモデル-Graph Order/degree 問題への適用-, 研究報告ハイパフォーマンスコンピューティング（HPC）,2017-HPC-162(18),1-10 (2017-12-11) .
5. 廣川祐太, 朴泰祐, 植本光治, 佐藤駿丞, 矢花一浩: 電子動力学シミュレーションコードのメニーコアプロセッサと GPU における性能比較, 研究報告ハイパフォーマンスコンピューティング（HPC）,2018-HPC-163(23),1-11 (2018-02-21) .
6. 高橋大介：数学定数の特定の桁を計算する BBP 型公式の高速計算法，日本応用数理学会 2017 年度年会講演予稿集, pp. 249-250, 2017.
7. 高橋大介：Xeon Phi プロセッサにおける並列一次元実数 FFT の実現と評価，日本応用数理学会 2017 年度年会講演予稿集, pp. 149-150, 2017.
8. 高橋大介：Knights Landing クラスタにおける並列 FFT の自動チューニング，2017 年ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2017 論文集, pp. 1-2, 2017.
9. 高橋大介：Xeon Phi クラスタ上の並列 FFT における通信隠蔽の自動チューニング，計算工学講演会論文集，Vol. 22，C-01-3, 2017.
10. 小林淳司, 建部修見，並列離散イベントシミュレータを用いた分散メタデータサーバのベンチマーク，研究報告ハイパフォーマンスコンピューティング（HPC），2017-HPC-160(26), 6 pages, 2017 年 7 月
11. 岩井厚樹, 建部修見, 同時複数タスク実行フレームワーク SMTEF を用いたメニータスク並列ベンチマークとその評価, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-160(33), 5 pages, 2017 年 7 月
12. 建部修見，Burst Buffer のための Gfarm ファイルシステム, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-161(2), 6 pages, 2017 年 9 月
13. 建部修見, Oakforest-PACS における IO-500 の評価, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-162(6), 5 pages, 2017 年 12 月
14. 田中昌宏, 建部修見, 川島英之, すばる HSC パイプラインの Pwrake/Gfarm による高速化手法の提案, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-162(9), 8 pages, 2017 年 12 月
15. 町田健太, 建部修見, Pwrake/Gfarm による分散並列相同性検索システムの提案, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-162(10), 9 pages, 2017 年 12 月

16. 北澤昂大, 建部修見, Oracle Storage Cloud の性能評価と Gfarm ファイルシステムへの組み込みの検討, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2017-HPC-162(11), 9 pages, 2017 年 12 月
17. 河合祐輔, 日野英逸, 建部修見, pbdMPI を用いたエントロピー推定プログラムの並列化と性能評価, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2018-HPC-163(13), 8 pages, 2018 年 3 月
18. 梶原 顕伍, 川島 英之, 建部 修見，Raft に基づく分散データベースにおけるデータ分割，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS）,2017-OS-141(20),pp. 1-6.
19. 中村 泰大, 川島 英之, 建部 修見，並列 WAL を適用した TicToc の評価，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS）,2017-OS-141(3),1-6 (2017-07-19)
20. 渡辺 敬之, 川島 英之, 建部 修見，並行実行木 Masstree における一括構築法の並列化，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS）,2017-OS-141(1),1-6 (2017-07-19)
21. 梶原 顕伍, 川島 英之, 建部 修見，Raft に基づく分散データベースの性能解析，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS）,2017-OS-140(15),1-9 (2017-05-09)
22. 横野 智也, 藤田 典久, 山口 佳樹, 大畠 佑真, 小林 諒平, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村 雅之: 宇宙輻射輸送計算における HDL 設計と OpenCL 設計の比較,情報処理学会研究報告 2018-HPC-163, No.24, pp.1 - 8, March 2018
23. 藤田 典久, 小林 諒平, 山口 佳樹, 大畠 佑真, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村 雅之: OpenCL を用いた FPGA による宇宙輻射輸送シミュレーションの演算加速, 情報処理学会研究報告 2017-HPC-161, No.12, pp.1 - 9 (September 2017)
24. 大畠 佑真, 小林 諒平, 藤田 典久, 山口 佳樹, 朴 泰祐: OpenCL と Verilog HDL の混合記述による FPGA 間 Ethernet 接続,情報処理学会研究報告 2017-HPC-160, No. 5, pp.1 - 9 (July 2017)

(2) 国際会議発表

A) 招待講演

1. Taisuke Boku, “Oakforest-PACS (OFP): Japan’s Fastest Sueprcomputer”, 1st French-Japan-German Workshop on HPC, Tokyo, Apr. 5th, 2017.

2. Taisuke Boku, “Oakforest-PACS (OFP): Japan’s #1 System driven by KNL and OPA”, IXPUG Spring Conference 2018, Cambridge, Apr. 11th, 2017.
3. Taisuke Boku, “Interconnection and I/O System on Oakforest-PACS: World Largest KNL+OPA Cluster”, Int. Workshop on Exascale Communication (ExaComm’2018) at ISC2018, Frankfurt, Jun. 22nd, 2017.
4. Taisuke Boku, “Toward Highly Productive Parallel Programming on Large Scale Accelerated Computing”, Int. Workshop on Parallel Programming Models and Systems Software for High-End Computing (P2S2) 2017, at ICPP2017, Bristol, Aug. 14th, 2017.
5. Taisuke Boku, “AiS: New Paradigm of Parallel FPGA Computing”, International HPC Forum, Guangzhou, Sep. 20th, 2017.
6. Taisuke Boku, “Oakforest-PACS (OFP): Japan’s Fastest Supercomputer and its Applications”, 40th ORAP Forum, Paris, Oct. 19th, 2017.

B) 一般講演

1. Taisuke Boku, “Computation/Communication Offloading to FPGA with GPU”, 3rd International Workshop on FPGA for HPC, Tokyo, Mar. 13th, 2018.
2. Taisuke Boku, “XcalableACC: Highly Productive Accelerated Programming Language for Extreme-Scale Computing”, SIAM Conference on Parallel Processing for Scientific Computing (PP18), Tokyo, Mar. 9th, 2018.
3. Taisuke Boku, “Reconfigurable Computation and Interconnection by FPGA for HPC”, SC17 BoF on FPGA for HPC, Denver, Nov. 14th, 2017.
4. Daisuke Takahashi, “Implementation of Parallel 1-D Real FFT on Intel Xeon Phi Processors”, 2018 Conference on Advanced Topics and Auto Tuning in High-Performance and Scientific Computing (2018 ATAT in HPSC), National Cheng Kung University, Tainan, Taiwan, March 27, 2018.
5. Ayumu Gomi and Daisuke Takahashi, “A Programming Framework for Performance Tuning in Julia”, SIAM Conference on Parallel Processing for Scientific Computing (PP18), Waseda University, Tokyo, Japan, March 7, 2018.
6. Daisuke Takahashi, “Implementation of Parallel FFTs on Cluster of Intel Xeon Phi Processors”, SIAM Conference on Parallel Processing for Scientific Computing (PP18), Waseda University, Tokyo, Japan, March 7, 2018.
7. Daisuke Takahashi, “An Implementation of Parallel 1-D Real FFT on Intel Xeon Phi Processors”, The 17th International Conference on Computational Science and Its Applications (ICCSA 2017), Trieste, Italy, July 4, 2017.

8. Hiroyuki Takizawa, Daichi Sato, Shoichi Hirasawa and Daisuke Takahashi, “A Customizable Auto-Tuning Scenario with User-defined Code Transformations”, The 12th International Workshop on Automatic Performance Tuning (iWAPT 2017), Orlando, Florida, USA, June 2, 2017.
9. Junji Kobayashi, Osamu Tatebe, “Simulation study of a distributed metadata server PPMDS”, 3rd Summer of CODES Workshop, Argonne, Jul. 11, 2017
10. Osamu Tatebe, “Simulation study of PPMDS metadata server using CODES”, Workshop on Japan-USA Collaboration for Extreme-scale System Software, Hawaii, Sep. 5, 2017
11. Osamu Tatebe, “System Software for Post-Petascale Data-Intensive Science”, JST/CREST International Symposium on Post Petascale System Software, Tokyo, Dec. 12, 2017
12. Osamu Tatebe, “Storage System of the Oakforest-PACS supercomputer”, CCS - LBNL Collaborative Workshop 2018, Tsukuba, Mar. 5, 2018
13. Osamu Tatebe, Junji Kobayashi, Atsuki Iwai, Misbah Mubarak, Rob Ross, “Codesign of Exascale Storage and Science Data Facilities”, 5th DOE/MEXT Workshop, Tokyo, Mar. 13, 2018
14. Hiroto Tadano, Stabilization of the Block BiCGGR method for linear systems with many right-hand sides, The 36th Annual International Conference on Simulation Technology (JSST2017), Tokyo, Japan, Oct. 26, 2017.
15. Hiroto Tadano, Development of a numerical stable Block Krylov subspace method for generating high accuracy solutions, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing (EPASA2018) Poster Session, Tsukuba, Japan, Mar. 5, 2018.
16. Hiroto Tadano, Convergence property and accuracy improvement of Block BiCGSTAB class solvers for linear systems with many right-hand sides, 18th SIAM Conference on Parallel Processing for Scientific Computing (PP18) Poster Session, Tokyo, Mar. 8, 2018.
17. Ryohei Kobayashi: Scalable Inter-FPGA Direct Communication for Parallel FPGA Applications, 18th SIAM Conference on Parallel Processing for Scientific Computing, March 2018
18. Ryohei Kobayashi, Yuma Oobata, Norihisa Fujita, Yoshiki Yamaguchi, and Taisuke Boku, OpenCL-ready High Speed FPGA Network for Reconfigurable High Performance Computing, HPC Asia 2018, January 2018

(3) 国内学会・研究会発表

A) 招待講演

1. 朴泰祐, “大規模・高性能計算システムの現状と今後”, IMPULSE コンソーシアム 2017, 東京, 2017 年 9 月 11 日.
2. 朴泰祐, “計算宇宙物理学と高性能計算技術のこれまでとこれから”, 天体形成論研究会, つくば, 2017 年 9 月 19 日.
3. 朴泰祐, “JCAHPC の新スーパーコンピュータ Oakforest-PACS ～ 1 年の運用を通してみた利用と成果～”, PC クラスタシンポジウム 2017, 東京, 2017 年 12 月 15 日.

B) その他の発表

1. 横野 智也, 藤田 典久, 山口 佳樹, 大畠 佑真, 小林 諒平, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村 雅之: 宇宙輻射輸送計算における HDL 設計と OpenCL 設計の比較, 第 163 回情報処理学会ハイパフォーマンスコンピューティング研究会, 松山, 2018 年 3 月.
2. 藤田 典久, 小林 諒平, 山口 佳樹, 大畠 佑真, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村 雅之: OpenCL を用いた FPGA による宇宙輻射輸送シミュレーションの演算加速, 第 161 回情報処理学会ハイパフォーマンスコンピューティング研究会, 熊本, 2017 年 9 月.
3. 大畠 佑真, 小林 諒平, 藤田 典久, 山口 佳樹, 朴 泰祐: OpenCL と Verilog HDL の混合記述による FPGA 間 Ethernet 接続, 第 160 回情報処理学会ハイパフォーマンスコンピューティング研究会, 秋田, 2017 年 7 月.
4. 大畠 佑真, 藤田 典久, 小林 諒平, 山口 佳樹, 朴 泰祐:高位合成による FPGA の高性能計算へ適用, 2017 年ハイパフォーマンスコンピューティングと計算科学シンポジウム（ポスター発表）, 神戸, 2017 年 6 月.
5. 辻美和子, 李珍泌, 朴泰祐, 佐藤三久: 疑似 MPI トレースプロファイルを用いた通信性能推定手法 SCAMP のための疑似トレースファイル作成手法の検討, 第 161 回情報処理学会ハイパフォーマンスコンピューティング研究会, 函館, 2017 年 9 月.
6. 廣川祐太, 朴泰祐, 植本光治, 佐藤駿丞, 矢花一浩: 電子動力学シミュレーションコードのメニーコアプロセッサと GPU における性能比較, 第 163 回情報処理学会ハイパフォーマンスコンピューティング研究会, 松山, 2018 年 2 月.
7. 中尾昌広, 村井均, 朴泰祐, 佐藤三久: Python と連携する PGAS 言語 XcalableMP のプログラミングモデル-Graph Order/degree 問題への適用-, 第 162 回情報処理学会ハイパフォーマンスコンピューティング研究会, 熊本, 2017 年 12 月.
8. 高橋大介, “数学定数の特定の桁を計算する BBP 型公式の高速計算法”, 日本応用数理学会 2017 年度年会, 東京, 2017 年 9 月 7 日.

9. 高橋大介，“Xeon Phi プロセッサにおける並列一次元実数 FFT の実現と評価”，日本応用数理学会 2017 年度年会，東京，2017 年 9 月 6 日．
10. 高橋大介，“Knights Landing クラスタにおける並列 FFT の自動チューニング”，2017 年ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2017，神戸，2017 年 6 月 5 日．
11. 高橋大介，“Xeon Phi クラスタ上の並列 FFT における通信隠蔽の自動チューニング”，第 22 回計算工学講演会，さいたま，2017 年 5 月 31 日．
12. 小林淳司, 建部修見，並列離散イベントシミュレータを用いた分散メタデータサーバのベンチマーク，第 160 回ハイパフォーマンスコンピューティング研究発表会, 秋田，2017 年 7 月 27 日
13. 岩井厚樹，建部修見，同時複数タスク実行フレームワーク SMTEF を用いたメニータスク並列ベンチマークとその評価，第 160 回ハイパフォーマンスコンピューティング研究発表会，秋田，2017 年 7 月 28 日
14. 建部修見，Burst Buffer のための Gfarm ファイルシステム，第 161 回ハイパフォーマンスコンピューティング研究発表会，函館，2017 年 9 月 19 日
15. 建部修見，Oakforest-PACS における IO-500 の評価，第 162 回ハイパフォーマンスコンピューティング研究発表会，熊本，2017 年 12 月 18 日
16. 田中昌宏，建部修見，川島英之，すばる HSC パイプラインの Pwrake/Gfarm による高速化手法の提案，第 162 回ハイパフォーマンスコンピューティング研究発表会，熊本，2017 年 12 月 18 日
17. 町田健太，建部修見，Pwrake/Gfarm による分散並列相同性検索システムの提案，第 162 回ハイパフォーマンスコンピューティング研究発表会，熊本，2017 年 12 月 18 日
18. 北澤昂大，建部修見，Oracle Storage Cloud の性能評価と Gfarm ファイルシステムへの組み込みの検討，第 162 回ハイパフォーマンスコンピューティング研究発表会，熊本，2017 年 12 月 18 日
19. 建部修見，Gfarm ファイルシステムの概要と最新機能，Gfarm シンポジウム，東京，2017 年 12 月 22 日
20. 河合祐輔，日野英逸，建部修見，pbdMPI を用いたエントロピー推定プログラムの並列化と性能評価，第 163 回ハイパフォーマンスコンピューティング研究発表会，松山，2018 年 3 月 1 日
21. 建部修見，Gfarm ファイルシステムの概要と最新機能，Gfarm ワークショップ，沖縄，2018 年 3 月 2 日

22. 多田野寛人, “多数の右辺ベクトルをもつ連立一次方程式に対する Block BiCGGR 法の数値的安定化について”, 2017 年度【非線形問題の解法に関する研究会】第 1 回非線形・可視化部門研究会, 自然科学研究機構核融合科学研究所, 2017 年 8 月 8 日.
23. 多田野寛人, “Block BiCGSTAB 型解法における数値的安定化と近似解の精度改善”, 2017 年度【非線形問題の解法に関する研究会】第 2 回非線形・可視化部門研究会, 自然科学研究機構核融合科学研究所, 2018 年 1 月 23 日.
24. 多田野寛人, “複数右辺ベクトルをもつ連立一次方程式に対する Block BiCGGR 法の近似解精度改善”, 日本応用数理学会 2018 年研究部会連合発表会, 大阪大学吹田キャンパス, 2018 年 3 月 16 日.
25. 廣川祐太, 朴泰祐, 植本光治, 佐藤駿丞, 矢花一浩: 電子動力学シミュレーション ARTED の KNL システム Oakforest-PACS での全系性能評価, 情報処理学会ハイパフォーマンスコンピューティング研究会, 秋田, 2017 年 7 月.
26. 松村和朗, 佐藤三久, 朴泰祐: フロー解析によるマルチ GPU 対応 OpenACC コンパイラ, 情報処理学会ハイパフォーマンスコンピューティング研究会, 秋田, 2017 年 7 月.
27. 中村泰大，並列 WAL を適用した TicToc の評価，情報処理学会システムソフトウェアとオペレーティングシステム研究会
28. 梶原 顕伍 , 川島 英之 , 建部 修見，Raft に基づく分散データベースにおけるデータ分割，情報処理学会システムソフトウェアとオペレーティング・システム研究会,2017-OS-141(20),pp. 1-6.
29. 中村 泰大 , 川島 英之 , 建部 修見，並列 WAL を適用した TicToc の評価，情報処理学会システムソフトウェアとオペレーティング・システム研究会,2017-OS-141(3),1-6 (2017-07-19)
30. 渡辺 敬之 , 川島 英之 , 建部 修見，並行実行木 Masstree における一括構築法の並列化，情報処理学会システムソフトウェアとオペレーティング・システム研究会,2017-OS-141(1),1-6 (2017-07-19)
31. 梶原 顕伍 , 川島 英之 , 建部 修見，Raft に基づく分散データベースの性能解析，情報処理学会システムソフトウェアとオペレーティング・システム研究会,2017-OS-140(15),1-9 (2017-05-09)

(4) 著書、解説記事等

（該当なし）

7. 異分野間連携・国際連携・国際活動等

国際連携

1. 朴泰祐、日独仏多国間共同研究 SPPEXA (German Priority Program on Software for Exascale Computing) 研究テーマ “MYX: MUST Correctness Checking for YML and XMP Programs”, 日本代表 PI.

8. シンポジウム、研究会、スクール等の開催実績

1. SPPEXA Workshop Japan 2017 主催, 東京, 2017 年 4 月 6 日
2. XcalableMP Workshop 2017 共催, 東京, 2017 年 10 月 31 日
3. 3rd International Workshop on FPGA for HPC 主催, 東京, 2018 年 3 月 12 日
4. Gfarm シンポジウム 2017，東京，2017 年 12 月 22 日
5. Gfarm ワークショップ 2018，沖縄，2018 年 3 月 2 日

9. 管理・運営

組織運営や支援業務の委員・役員の実績

1. 朴泰祐：計算科学研究センター計算機システム運用委員会委員長
2. 朴泰祐：筑波大学システム情報系人事委員会委員
3. 朴泰祐：筑波大学情報環境機構企画室会議委員
4. 朴泰祐：筑波大学ネットワーク管理委員会委員
5. 朴泰祐：HPCI 連携サービス委員会委員
6. 朴泰祐：学際大規模情報基盤共同利用・共同研究拠点（JHPCN）運営委員会委員
7. 朴泰祐：理化学研究所客員主管研究員
8. 高橋大介：筑波大学情報環境機構学術情報メディアセンター運営委員会委員
9. 高橋大介：理化学研究所客員主管研究員
10. 高橋大介：HPCI 利用研究課題審査委員会レビューアー
11. 高橋大介：HPCI 連携サービス運営・作業部会委員
12. 高橋大介：学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題審査委員
13. 建部修見：HPCI 連携サービス運営・作業部会委員
14. 建部修見：理化学研究所客員主管研究員
15. 建部修見：情報通信研究機構協力研究員
16. 建部修見：HPCI 利用研究課題審査委員会レビューアー
17. 建部修見：学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題審査委員
18. 建部修見：東京工業大学学術国際情報センター共同利用専門委員
19. 建部修見：特定非営利団体つくば OSS 技術支援センター理事長
20. 建部修見：SNIA 日本支部エクストリームストレージ研究会研究会長
21. 建部修見：情報処理学会論文誌コンピューティングシステム編集副委員長

22. 川島英之：産業技術研究所人工知能研究センター客員研究員

10. 社会貢献・国際貢献

1. Taisuke Boku: Steering Committee Chair, International Conference HPC Asia Series
2. Taisuke Boku: Organizing Committee Member, International Conference HPC Asia 2018
3. Taisuke Boku: Organizing Chair, 3rd International Workshop on FPGA for HPC
4. Taisuke Boku: Organizing Chair, SPPEXA Workshop Asia 2017
5. Taisuke Boku: Organizing Chair, IXPUG Workshop HPC Asia 2018
6. Taisuke Boku: Steering Committee Member, ISC2017
7. Taisuke Boku: Steering Committee Member, IXPUG
8. Taisuke Boku: Program Committee Member, ICPP2017
9. Taisuke Boku: Program Committee Member, IPDPS2017
10. Taisuke Boku: Program Committee Track Co-Chair, CCGrid2017
11. Taisuke Boku: Program Committee Member, NPC2017
12. Taisuke Boku: Program Committee Member, IXPUG Spring Conference 2018
13. Taisuke Boku: Program Committee Member, IWOMP2017
14. Daisuke Takahashi: Program Committee, The 15th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2017)
15. Daisuke Takahashi: Program Committee, International Workshop on Legacy HPC Application Migration (LHAM 2017) in Conjunction with 5th International Symposium on Computing and Networking (CANDAR'17)
16. Daisuke Takahashi: Program Committee, The 2nd International Workshop on GPU Computing and Applications (GCA'17) in Conjunction with 5th International Symposium on Computing and Networking (CANDAR'17)
17. Daisuke Takahashi: Program Committee, International Conference on High Performance Computing in Asia Pacific Region (HPC Asia 2018)
18. Daisuke Takahashi: Program Committee, Special Session on High Performance Computing for Application Conference on High Performance Computing & Simulation (HPCS 2017)
19. Daisuke Takahashi: Publicity Committee, The 17th International Conference on Computational Science and Its Applications (ICCSA 2017)
20. Daisuke Takahashi: Program Committee, The 11th International Conference on Frontier of Computer Science and Technology (FCST 2017)
21. Daisuke Takahashi: Program Committee, The International Conference on Computational Science (ICCS 2017)

22. Daisuke Takahashi: Program Committee, IEEE 11th International Symposium on Embedded Multicore SoCs (MCSoC-17)
23. Daisuke Takahashi: Program Committee, The 17th IEEE International Conference on Computer and Information technology (CIT 2017)
24. Daisuke Takahashi: Program Committee, The 12th International Workshop on Automatic Performance Tuning (iWAPT 2017)
25. 高橋大介：情報処理学会論文誌ジャーナル／JIP 編集委員会委員
26. 高橋大介：情報処理学会ハイパフォーマンスコンピューティング研究会運営委員
27. Osamu Tatebe: Program Chair, International Conference on High Performance Computing in Asia-Pacific Region
28. Osamu Tatebe: Program Track Chair, International Supercomputing Conference
29. Osamu Tatebe: Program Committee and Poster Vice Chair, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC17)
30. Osamu Tatebe: Program Committee, IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2017)
31. Osamu Tatebe: Program Committee, IEEE International Conference on Cluster Computing (Cluster 2017)
32. Osamu Tatebe: Program Committee, International Workshop on Advances in High-Performance Computational Earth Sciences: Applications & Frameworks (IHPCES 2017)
33. Hideyuki Kawashima: Program Track Co-Chair, International Conference on High Performance Computing in Asia-Pacific Region (HPCAsia'18)
34. Hideyuki Kawashima: Program Committee, IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-17)
35. Hideyuki Kawashima: Program Committee, 2017 IEEE/WIC/ACM International Conference on Web Intelligence (WI'17)
36. Hideyuki Kawashima: Program Committee, 2017 IEEE International Conference on BigData.
37. 川島英之：プログラム副委員長、xSIG (cross-disciplinary workshop on computing Systems, Infrastructures, and programminG)
38. 川島英之：情報処理学会データベースシステム研究会運営委員
39. 川島英之：情報処理学会システムソフトウェアとオペレーティングシステム研究会運営委員
40. 川島英之：電子情報通信学会知的環境とセンサネットワーク研究会運営委員
41. 川島英之：情報処理学会論文誌データベース編集委員
42. 川島英之：電子情報通信学会論文誌査読委員

43. Hiroto Tadano: Local Committee, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing (EPASA2018)
44. Hiroto Tadano: Publication Co-Chair, The 37th JSST Annual International Conference on Simulation Technology (JSST2018).
45. 多田野寛人：日本応用数理学会「行列・固有値問題の解法とその応用」研究部会幹事
46. 多田野寛人：日本応用数理学会 JSIAM Letters 編集委員
47. 多田野寛人：情報処理学会ハイパフォーマンスコンピューティング研究会運営委
48. 多田野寛人：情報処理学会論文誌コンピューティングシステム論文誌編集委員
49. Ryohei Kobayashi: Technical Program Committee, The Fifth International Symposium on Computing and Networking (CANDAR 2017), Track-2 (Architecture and Computer System)
50. 小林諒平：電子情報通信学会リコンフィギャラブルシステム研究会専門委員
51. 小林諒平：電子情報通信学会コンピュータシステム研究会専門委員

11. その他

海外長期滞在、フィールドワークなど

（該当なし）