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　15 名、学類生　7 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、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】 TCA における CG 法計算に関する性能評価 [朴]

TCA (Tightly Coupled Accelrators)コンセプトに基づく FPGA (Field Programmable Gate Array)実装ハードウェアである PEACH2 の通信ドライバ及び制御方式について改良し、NAS Parallel Benchmarks の CG ベンチマークを GPU クラスた HA-PACS/TCA 上で 2 次元プロセスマッピング実装し、短メッセージ通信に強いという TCA の特性を生かし H27 年度の結果よりさらに性能を向上させた。

図 1 に NAS-PB CG の TCA 及び InfiniBand による実行性能の比較を、計算時間及び通信時間の内訳によって示す(Class=A)。16 ノード実行において、TCA による通信時間が InifiniBand を下回り、通信時間比較では 1.44 倍の性能向上が得られていることがわかる。2 次元分割法を有効に利用することにより、計算時間が短縮され、同時にメッセージ長も短くなるが、短メッセージに強いという TCA の長所が現れ、ストロングスケーリングにおいて InfiniBand を上回る性能が得られた。

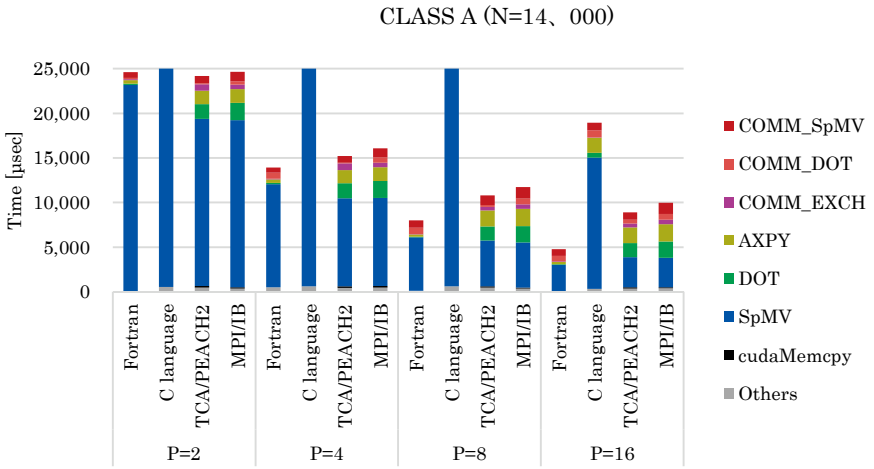


図 1 NASPB-CG の2次元マッピング実装における TCA/PEACH2 と InfiniBand による実装の性能評価

【２】 PGAS 言語向け通信ライブラリ GASNet の GPU 向け実装と TCA 実装［朴］

GASNet は米国 LBNL (Lawrence Berkeley National Laboratory)で開発された、PGAS (Partitioned Global Address Space)モデル実装用の低レベル通信ライブラリである。分散メモリ上で MPI よりも軽い通信を実現する。これまで GASNet は CPU における並列通信のみを対象にしてきたが、近年、これを GPU を持つクラスタノードに対応させる拡張が LBNL によって進められている。H27 年度に引き続き、LBNL の GASNet/GPU 開発チームと共同研究を行い、先方が InfiniBand を対象として進めているリファレンス実装と並行し、これを TCA 機構にも用い、両者の性能比較を行う研究を進め、国際会議共著論文及び国内会議共著論文の執筆を行った。TCA 機構の問題点の一つはプログラミング互換性であり、本来は TCA 専用の API を用いたアプリケーションあるいはライブラリの記述が必要であるが、本実装を用いることにより、GASNet/GPU の API で記述された処理系は原理的に TCA 上で実行できることになる。

TCA 上の GASNet/GPU（以下、GASNet/TCA）の設計・実装は、GPU からの通信要求を、CPU 上で実行されている通信デーモンで監視し、そこから発生する必要な通信を TCA の API を用いて発行することで、間接的に GPU からの TCA 要求を処理する。通信結果を返す必要がある場合は、同様の機構を用いて GPU に送る。これらは CUDA で提供されるピンダウンメモリ領域を使ったリングバッファを介して行われるが、通信データは TCA が提供する GPU 間リモート DMA を用いて送受信されるため、リングバッファが性能ボトルネックとなることはない。図 2 にリングバッファの基本構造を、図 3 に通信性能を最適化するために導入される4つの通信モードの様子を示す。現在の GASNet/GPU はリモート通信の対象（ローカル通信エリアではなく）となる segment を１つだけ許している。これらの4つのモードは、この制約を意識し、ユーザの要求に応じて最小限のバッファリングで通信が完了するように工夫されている。

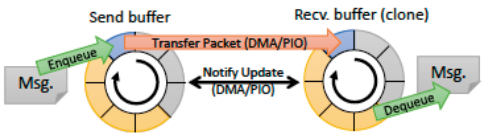


図 2 GASNet/TCA における GPU・CPU 間通信のためのパケット通信機構

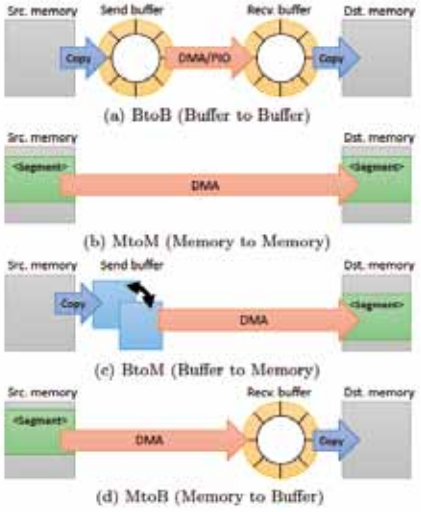


図 3 GASNet/TCA における 4 つの通信モード

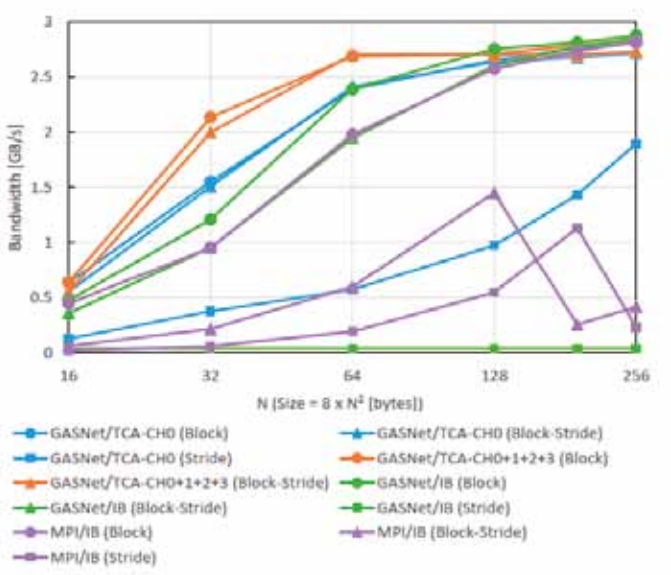


図 4 GASNet/GPU における 3 次元矩形領域の halo データ交換性能の比較

CCS における GPU クラスタ HA-PACS/TCA を用い、特に TCA が注目している多次元矩形領域における halo（隣接ノードとの接合面）データ交換について評価を行った結果を図 4 に示す。ここでは GASNet/GPU の API に基づき、3 次元矩形領域の halo データ交換を 1 次元方向（連続領域通信）、2 次元方向（ブロックストライド通信）、3 次元方向（ストライド通信）の 3 種類の方向で通信した場合の性能を、MPI/InifiniBand と TCA の両者について比較している。図からわかるように、TCA では連続領域通信とブロックストライド通信でほとんど性能が変わらないのに対し、ブロックストライド通信をパッキングによってのみ処理可能な MPI/InfiniBand では後者の場合に大きく性能が

低下していることがわかる。また、このモードでの通信に関しては、GASNet ではなく通常の MPI/InfiniBand を用いた場合よりも TCA がさらに良い性能を示している点も重要である。しかし、単純なストライド転送(3 次元 halo)については、TCA であってもパッキングを用いないと性能が低下するため、そのオーバヘッドの度合いは MPI/InfiniBand とほとんど同じである。ハードウェア性能の違いより、単純な連続通信あるいはパッキング通信では MPI/InfiniBand の方が性能が若干高いが、一般的な多次元 halo 通信において、TCA が常に高性能を示すことが確認された。

なお、本研究の国内発表である HPCS2016 における発表は、同会議の最優秀論文賞を受賞し、本研究の意義が広く認められると共に、本 CREST 研究をベースとした米国 LBNL との共同研究が共著論文による共同受賞という重要な成果を挙げた点で重要である。

【3】 GPU 直接起動型 MPI ライブラリ GMPI の開発［朴］

H27 年度に引き続き、GPU カーネルから直接 MPI 通信を起動するためのフレームワークである GMPI (GPU-ready MPI)の開発を行った。前年度の予備評価では、単純な pingpong 転送においては GMPI における GPU 間通信時間は、従来手法である GPU カーネルからの離脱と CPU での MPI 通信起動及び GPU カーネルの再呼び出しの合計時間を下回り、有効な通信手法であることが確認された。これにより、GPU 化する前の MPI プログラムを容易に並列 GPU 環境に移植することが可能となっていた。しかし、実アプリケーション例として Himeno Benchmark に適用した場合、本来は変化がないはずの GPU 内演算時間に影響が現れ、MPI 実装に比べ演算時間が延びるという現象が観測された。

H28 年度研究ではこの問題に関する改善を試みた。詳細な実行時間解析の結果、以下の現象が生じていることが間接的に認められた。

- MPI と分割カーネルによる実装では、任意個数のスレッドを起動可能で、それらの間ではカーネル関数の呼び出しを基準とした同期が取られるのに対し、GMPI ではカーネル関数から抜け出さずに通信を行うため、スレッド数は GPU 演算コア数を上回ることができない。
- このため問題全体の並列分割数の上限は GPU コア数となり、問題のデータ領域が大きい場合、GPU コアのレジスタ、ローカルメモリ、テキスチャメモリ等のハードウェアリソースの制約に対し問題がフィットせず、レジスタスピル等が頻発し、演算性能が低下する。
- さらにカーネル内でのコア間同期セマンティクスを維持するため、ソフトウェアオーバヘッドの大きい同期手法を実装しなければならず、そのコストは通信時間とカーネル呼び出し時間コストを上回ってしまう場合がある。

以上の問題点についての解決方法として、通信に関する同期を CPU 側で間接的にとり、コア間同期のコストを下げる手法を検証したが、スレッド数の縮小により GPU コア内のリソースを圧迫する問題については現在の CUDA ベースの枠組み内では十分な最適化が行えないことがわかった。

このため、本手法の適用範囲は制限され、比較的小規模な問題に対しては一定の並列性を持つ環境で性能を向上させることはできるが、より大きな問題に対してはコンパイラを含めた抜本的な最適化が必要であることがわかった。

【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。特に H28 年度後半においては、JCAHPC で導入した国内最大の Xeon Phi クラスタである Oakforest-PACS 向けに同コードを最適化し、これまで COMA 上の KNC (Knights Corner)で進めてきたメニーコアプロセッサ向けの ARTED コードの性能チューニングを、Oakforest-PACS の KNL (Knights Landing)向けに比較的スムーズに移植することに成功した。

まず、年度前半での KNC 向けのチューニングについて述べる。ARTED のカーネル部分は 3 次元の 25 点ステンシル計算であり、メモリインテンシブかつ CPU インテンシブな計算が支配的である。KNC には GDR5 規格の高速メモリが実装されており、これを有効活用することでメモリ性能依存な計算が高速化される。加えて、KNC の特徴である AVX512（512 ビット水平 SIMD ベクトル命令）を最大限に活かすためのデータ配置及び演算順序をコード上で最適化し、1 台の KNC の理論ピーク演算性能である 1TFLOPS の約 25～30%の実効性能を引き出すことに成功した。

```
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 5 KNC 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

図 5 に 3 次元 25 点ステンシルコードのカーネル部分の KNC 向け最適化結果を示す。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、KNC が苦手とする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。さらに、大規模並列化のために KNC の native mode を用いた実装と、CPU と KNC を併用する hybrid

mode による実装を行った。native mode 実行は、アクセラレータである KNC 上の Linux で MPI プロセスを直接実行することにより、KNC だけのクラスタを構築し、MPI 通信もその上で直接行う方法である。一方、hybrid mode 実行では、CPU 上の MPI プロセスと native mode 相当の KNC 上の MPI プロセスを一つの MPI 空間で実現し、クラスタ上の全リソースを用いた並列処理を実現する。我々は ARTED コードの hybrid mode 実行コードを実装し、COMA クラスタで性能評価した。

図 6 に COMA のノードを最大 128 台利用した場合の ARTED のストロングスケーリング性能を示す。CPU のみによる実行、KNC の native mode のみによる実行、symmetric mode で CPU と KNC に均等に負荷を割り付けた場合、そして symmetric mode で CPU と KNC の実効性能を考慮し最適負荷分散を行った場合の 4 通りの結果が示されている。全てのノード数において負荷バランスを考慮した symmetric mode 実行が最高性能を達成しているのがわかる。また、ノード数が比較的少ない場合では symmetric mode 実行の性能は CPU のみを用いた場合の 2 倍以上であり、KNC 部分では理論ピーク性能の約 25～30%の性能が達成された。

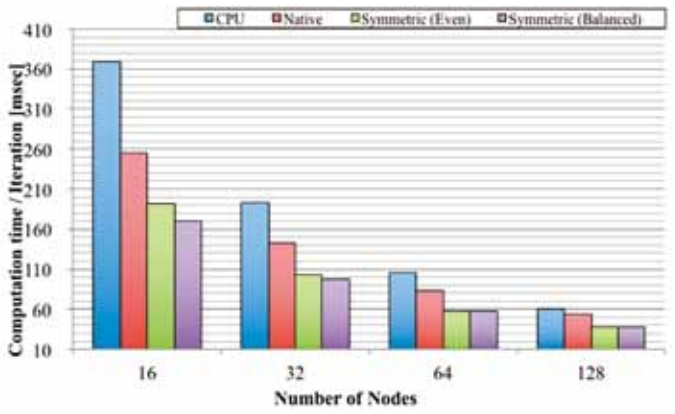


図 6 ARTED コードの各種実行モードにおけるストロングスケーリング評価

一方、これらの結果に基づき、同コードの MPI プロセス内を OpenACC 化する作業を、NVIDIA 社との共同研究により行った。コードは同社の最新 GPU である P100 (Pascal アーキテクチャ)上で実行され、Intel Xeon Phi (KNL: Knights Landing)プロセッサを上回る実効性能が達成された。しかし、P100 の理論ピーク性能は 5.3TFLOPS で、Oakforest-PACS の KNL の 3.0TFLOPS より 1.77 倍高性能であり、実効性能効率の点では KNL 実装の方が効率が高いことがわかった。

また、年度終盤において、JCAHPC が実施する大規模 HPC チャレンジプログラムの機会を用い、Oakforest-PACS の全系(8192 ノード、約 25PFLOPS)を用いた超大規模の ARTED 実行を行った。このトライアルでは、同システムを 3 日間専有し、通常実行では不可能な超大規模・長時間での実行を行った。ARTED コードの KNL 上の大規模並列性能はカーネルコード部分で約 25%、コード全体では約 17%であった。また、性能評価だけでなくサイエンスとしての光物性シミュレーション結果も得たが、これについては矢花グループの報告に詳しい。

【５】 PEZY-SC 向け OpenACC コンパイラの予備実装・評価［朴、佐藤（CCS フェロー）］

PEZY-SC プロセッサは ExaScaler 社のスーパーコンピュータシステムで採用されているアクセラレータである。このシステムは電力性能比が高いという特徴から注目を集めているが、PZCL という OpenCL の方言で記述するためプログラミングが煩雑であるという課題がある。そこで、PEZY-SC プロセッサのプログラミングを簡易にし、既存の OpenACC コードをより多くのプラットフォームで実行可能にするため、PEZY-SC 向けの OpenACC コンパイラを予備実装し評価した。

PEZY-SC には 1024 個のコアとキャッシュが階層的に構成されており、各階層において同期を取ることが可能である。各コアでは 8 スレッドが SMT で動作する。8 つのスレッドは実際には 4 つのペアで構成されており、通常はペアの片方だけしか実行されず、同期や明示的なスレッド切り替え命令によってもう片方が実行されるようになる。

コンパイラ実装には Omni OpenACC コンパイラという CUDA にコード変換を行う OpenACC コンパイラを拡張することで、OpenACC コードから PZCL にコード変換を行えるようにした。現在は予備実装段階で PEZY-SC 向けの最適化は実装されていない。

ベンチマークとして N-body と NAS Parallel Benchmarks CG (NPB-CG)の 2 つを用い、比較として PZCL コード 3 種類と OpenACC コードを用意した。評価環境には KEK の Suiren Blue の 1 ノードを使用した。性能を図 7 に示す。PZCL の Opt.1 はプロセッサ全体同期命令を用いて複数のカーネルをまとめることでカーネル起動コストを減らす最適化である。Opt.2 は明示的なスレッド切り替えによりメモリアクセスレイテンシの隠蔽やキャッシュの有効利用をする最適化である。

N-body では OpenACC は PZCL の 98%以上の性能が出ており、また PZCL での最適化の効果も小さかった。これは N-body は計算律速のプログラムであったからである。次に NPB-CG では OpenACC は PZCL (base)の 92%以上の性能を達成できた。性能低下の要因としてはコンパイラによるコード変換やリダクション変数に関わる余計な通信があげられる。また OpenACC は PZCL(Opt.1)と比較すると 69～99%、PZCL(Opt.1、2)と比較すると 62～88%の性能であった。Opt.1 は問題サイズが小さく実行時間が短い時に効果が大きく、また Opt.2 はどの問題サイズでも効果があった。コードの行数を比較すると、OpenACC は PZCL に比べて N-body は 48%、NPB-CG は 45%の行数で記述できており、OpenACC により簡易に記述ができたと言える。

OpenACC コンパイラにおいても PEZY-SC 向けに最適化したコード変換を行えるようにする必要がある。Opt.1 に関しては OpenACC kernels 指示文で指定された部分を 1 カーネルにまとめることは可能である。また Opt.2 に関しては対応する指示文を新たに導入することを検討している。また OpenACC は 1 ノード上のアクセラレータの処理しか記述できないため MPI などで通信を記述する必要がある。そこで XcalableACC を用いて PEZY-SC のクラスタにおいても、簡易な記述を可能にすることを計画している。

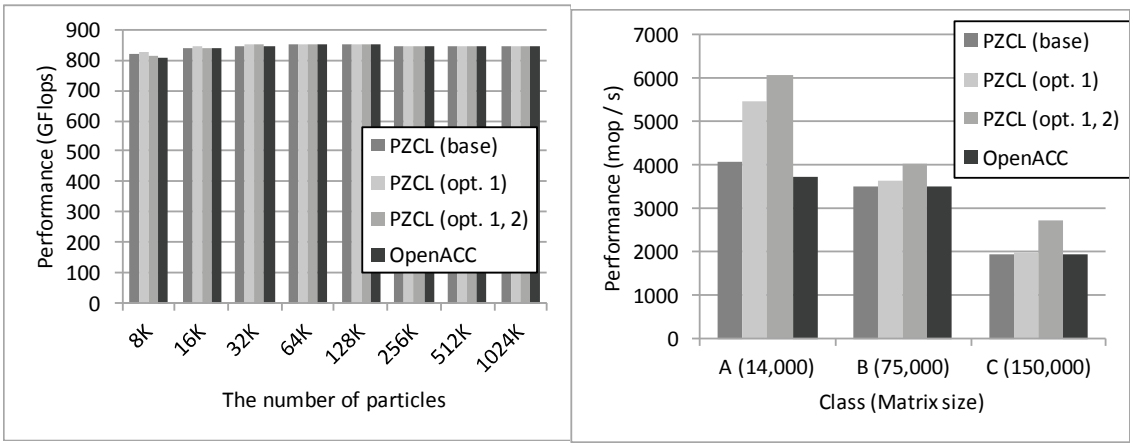


図 7: PZCL と OpenACC による N-body の性能（左）と NPB-CG の性能（右）

【６】 XcalableACC における片側通信の改善［朴、佐藤（CCS フェロー）］

XACC では XMP と同様に local-view モデルにおける coarray を用いた通信が可能である。NPB-CG における評価では global-view モデルよりも高い性能を達成し、MPI+OpenACC と同等の性能が得られたが、姫野ベンチマークでは global-view よりも性能が下がる問題があるためその改善を行った。

図 8 に姫野ベンチマークの性能を示す。評価環境には CCS の HA-PACS/TCA システムを用いた。XACC local-view (XACC-L)は 4×2 プロセス以上で性能が下がっており MPI+OpenACC で send/recv を使ったものと比べて 85%まで性能が低下していた。原因は coarray の通信がブロッキングになっており、複数の通信が同時に行われないためであった。現在の実装では coarray を単純に関数呼び出しに置き換えるだけのため、依存の有無にかかわらず通信がブロッキングとなっている。姫野ベンチマークでは 2 つの隣接ノードと同時に通信可能であるが、それが同時に通信できずに性能が低下した。

実験的に通信をノンブロッキングにしてみると (XACC-L(nb))、8×4 プロセスまで MPI+OpenACC の 98%以上と同等の性能まで改善した。また 8×8 プロセスにおいても、MPI+OpenACC で get を用いたものと同等の性能となっているため、この差は send/recv と get 通信自体の性能差によるものと言える。今後は通信をノンブロッキングにするためのコンパイラの改善およびユーザが指定するための指示文を提案し実装する。また現在は XACC/C のみで利用可能なので Fortran でも利用可能にする。

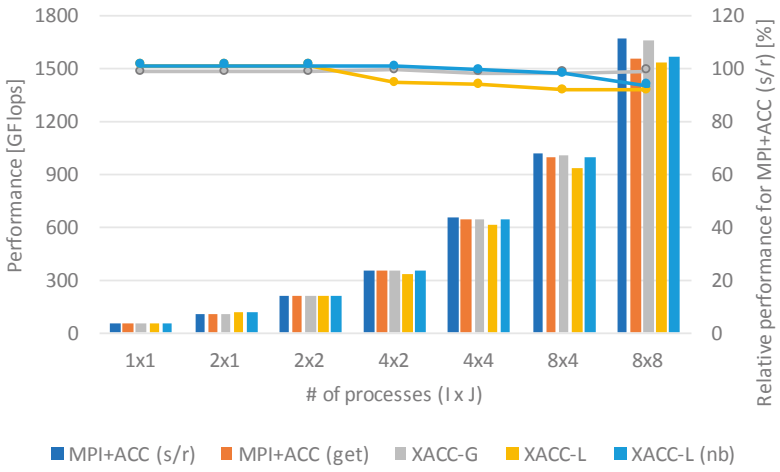


図 8: 姫野ベンチマークの性能

【7】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）

JST-CREST における TCA 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進めている。

H28 年度研究では、Xilinx 社及び Altera 社における最新鋭 FPGA である Arria10 及び UltraScale を用い、HPC アプリケーションを想定したカーネルループを OpenCL で記述し、その一方で FPGA 評価ボード上の特定のハードウェア機能を Verilog HDL で記述、最終的に後者をサブルーチン的に前者から呼び出すことができることを確認し、今後の研究の基礎となる技術を確立した。具体的には、FPGA ボード上の固有ハードウェアである LED 群を操作するルーチンを Verilog HDL で実装し、OpenCL 側から呼び出す実験を行った。また、OpenCL 側については標準的なメモリベンチマークである Stream Benchmark の各カーネルループを実装し、OpenCL だけでは dual channel memory の操作が行なえないことを確認した上、これを Verilog HDL で記述すればメモリバンド幅を有効活用できることを確認した。

これらはまだ基礎実験の範疇を出ないが、今後、OpenCL 高位合成と Verilog HDL による低レベル記述を組み合わせ、ユーザは OpenCL だけを記述してシステムの低レベルハードウェアを自由にアクセスすることができるようにし、これによって AiS のコンセプトを具体化するフレームワークを構築していく予定である。このコンセプトにより、CCS における PACS-X 計画を支える基盤技術が構築できると考えている。

【８】Xeon Phi クラスタ上の並列高速フーリエ変換（FFT）における通信隠蔽の自動チューニング（高橋）

高速 Fourier 変換（fast Fourier transform、以下 FFT）は、科学技術計算において今日広く用いられているアルゴリズムである。並列スーパーコンピュータの普及に伴い、並列 FFT アルゴリズムがさまざまな研究者によって提案されており、ライブラリとなっているものも多い。分散メモリ型並列計算機においてチューニングを行う際に、最適な性能パラメータはプロセッサのアーキテクチャ、ノード間を結合するネットワーク、そして問題サイズなどに依存するため、これらのパラメータをその都度手動でチューニングすることは困難になりつつある。そこで、自動チューニングを適用した FFT ライブラリとして FFTW や、SPIRAL などが提案されている。また、並列 FFT における演算と通信のオーバーラップ手法が提案されている。本研究では、並列一次元 FFT において通信隠蔽のパラメータを自動チューニングし Xeon Phi クラスタ上で性能評価を行った。

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にした場合、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)は各 MPI プロセス内の性能に関するパラメータである。今回は(2)~(4)に対して自動チューニングを適用した。

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場合、通信メッセージサイズの分割数（パイプラインの段数）を大きくすれば、オーバーラップの割合が高くなる。その一方で、通信メッセージサイズの分割数を大きくすれば、通信 1 回あたりの通信メッセージサイズが小さくなるため、通信バンド幅も小さくなる。また、演算と通信をオーバーラップさせる際には、通信によってメモリバンド幅が消費される。したがって通信メッセージサ

イズの分割数には最適な値が存在すると考えられる。そこで、通信メッセージサイズの分割数を NDIV としたとき、NDIV=1（オーバーラップなし）～16 の範囲で、通信すべき要素数が NDIV で割り切れるすべての場合について全探索を行う。

six-step FFT アルゴリズムでは、データ数Nを$N = N_1 \times N_2$と分解してN_1組のN_2点 FFT と、N_2組のN_1点 FFT をそれぞれ計算する。ここで、N_1とN_2を基底と呼ぶことにする。N_1とN_2の値は、$N = N_1 \times N_2$を満たしていれば任意に選ぶことができる。ただし、MPI プロセス数をPとした場合、N_1、 $N_2 \geq P$を満たす必要がある。通常は$N_1 \approx N_2 \approx \sqrt{N}$となるように$N_1$と$N_2$を選ぶことが多いが、性能が最も高くなるように$N_1$と$N_2$を選ぶことができる。なお、データ数$N$が 2 のべき乗になる場合には、すべての$N_1$と$N_2$の組み合わせを試行したとしても、探索空間は$\log_2(\sqrt{N}/P)$となる。

six-step FFT アルゴリズムにおいては行列の転置が必要になるが、この行列の転置はキャッシュブロッキングを行うことで効率よく実行できることが知られている。その際、最適なブロックサイズ NB は、問題サイズおよびキャッシュサイズ等に依存する。今回の実装では、ブロックサイズ NB を 2 のべき乗に限定して 4、8、16、32、64 と変化させている。

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 3.3.6-pl1 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 128 ノードを用いた。コンパイルオプションは FFTE に対しては“mpiifort -O3 -xMIC-AVX512 -qopenmp”を、FFTW に対しては“mpiicc -O3 -xMIC-AVX512 -qopenmp”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 に設定している。$N = 2^m$点 FFT の GFlops 値は$5N \log_2 N$より算出している。図 9 に並列一次元 FFT の性能を示す。図 9 から、$N \geq 2^{30}$の場合には通信隠蔽の効果により FFTE 6.2alpha (no overlap)や FFTW 3.3.6-pl1 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。また、FFTE 6.2alpha(NDIV=4)では通信メッセージサイズが常に 4 分割されており全対全通信性能が低くなっていることから、$N \leq 2^{29}$の場合には FFTE 6.2alpha (no overlap)よりも性能が低くなっていることが分かる。

図 10 に全対全通信の性能を示す。通信メッセージサイズが 512KB となる場合には、通信バンド幅が 2GB/s を超えていることが分かる。一方で、通信メッセージサイズが 256KB から 128KB に減少する際には通信バンド幅が約 1/7 になっている。このような場合には通信メッセージサイズを 2 分割すると通信時間が約 7 倍になってしまうことを意味している。表 1 に並列一次元 FFT における自動チューニングの結果を示す。FFTE 6。2alpha では$N_1 = N_2 = \sqrt{N}$または$N_1 = \sqrt{N/2}$、$N_2 = 2N_1$となるようにN_1とN_2が選ばれており、ブロックサイズとして

NB = 32 が用いられているが、これらのパラメータは必ずしも最適ではないことが自動チューニングの結果から分かる。特に、$N \leq 2^{25}$においては NDIV=1、つまり演算と通信をオーバーラップさせない場合が最も高速であることが分かる。

並列一次元 FFT において、通信隠蔽のパラメータについて自動チューニングを行うことで、性能をさらに向上させることができることを示した。

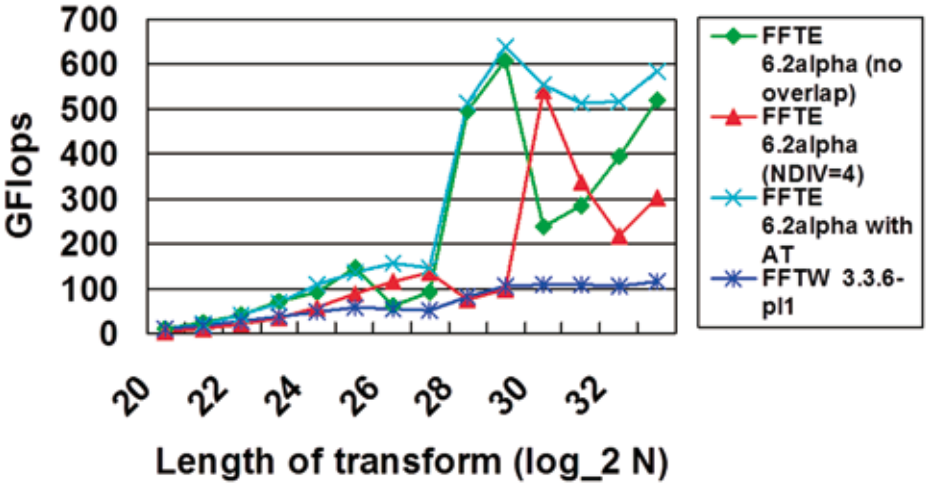


図 9: 並列一次元 FFT の性能（Oakforest-PACS、128 ノード）

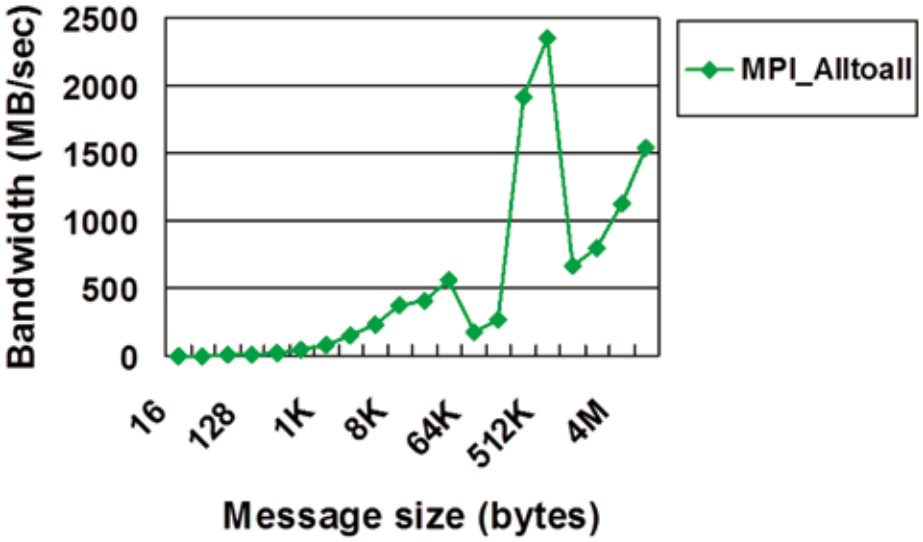


図 10: 全対全通信の性能（Oakforest-PACS、128 ノード）

表 1: 並列一次元 FFT における自動チューニングの結果（Oakforest-PACS、128 ノード）

N	FFTE 6。2alpha					FFTE 6。2alpha with AT				
	N_1	N_2	NB	NDIV	GFlops	N_1	N_2	NB	NDIV	GFlops
2^{24}	4096	4096	32	4	57.789	4096	4096	32	1	109.426
2^{25}	4096	8192	32	4	86.856	4096	8192	16	1	137.380
2^{26}	8192	8192	32	4	116.943	8192	8192	16	2	154.771
2^{27}	8192	16384	32	4	136.976	8192	16384	32	4	147.322
2^{28}	16384	16384	32	4	73.285	8192	32768	16	1	513.787
2^{29}	16384	32768	32	4	98.635	16384	32768	8	1	638.425
2^{30}	32768	32768	32	4	541.741	32768	32768	64	4	554.879
2^{31}	32768	65536	32	4	337.246	32768	65536	32	8	512.600
2^{32}	65536	65536	32	4	217.038	65536	65536	32	16	516.490
2^{33}	65536	131072	32	4	303.718	32768	262144	16	1	584.730

【９】Xeon Phi における多倍長精度浮動小数点演算の実現と評価（高橋）

多倍長精度浮動小数点演算を高速に行うために、これまでさまざまなライブラリが提案されている。また、GPU における多倍長精度浮動小数点演算の実装が行われている。

一方で、メニーコアプロセッサとして Xeon Phi の普及が進んでいる。Xeon Phi においてこれまでに多倍長精度整数演算を実装した例が知られているが、Xeon Phi における多倍長精度浮動小数点演算はまだ実装されていないのが現状である。そこで本研究では、Xeon Phi において多倍長精度浮動小数点演算を並列化し性能評価を行った。

多倍長精度浮動小数点数は、32 ビット整数の配列を用いて符号部、指数部、および仮数部で表現されている。1 番目の要素には符号s（1 または−1）が格納されており、2 番目の要素には指数部q（$-2^{31} \le q \le 2^{31}-1$）が格納されている。3 番目から$m+2$番目までの要素には仮数部$c$がビッグエンディアンで10進8桁ずつ格納されている。このような表現形式により、多倍長精度浮動小数点数の値は$s \times c \times 10^q$と表される。

また、n桁の多倍長精度浮動小数点数どうしの乗算は高速 Fourier 変換（FFT）を用いることで、$O(n \log n \log\log n)$の計算量で行えることが知られている。FFT を倍精度実数で計算した場合、FFT の精度の関係で倍精度実数配列の各要素には 10 進 4 桁までしか格納できないという制約がある。したがって、32 ビット整数には 10 進 9 桁まで格納可能であるが、10 進 4 桁との変換が容易である 10 進 8 桁を採用している。

n桁の多倍長精度浮動小数点数どうしの加減算は、明らかに$O(n)$の計算量で行えることが分かる。しかし、多倍長精度浮動小数点数どうしの加減算において、並列化を阻害する要因

はキャリーおよびボローの処理である。これらの処理を並列化するには、桁上げ先見（carry look-ahead）や桁上げ飛び越し（carry skip）方式などを用いることが考えられるが、今回の実装では、実現が容易である桁上げ飛び越し方式を並列化して、キャリーの伝搬を処理している。多倍長精度浮動小数点数どうしの減算においても、加算と同様にしてボローの処理が可能である。また、FFT を用いた多倍長精度浮動小数点数どうしの乗算における正規化の処理も、多倍長精度浮動小数点加減算と同様にして処理が可能である。

FFT を用いた多倍長精度浮動小数点数どうしの乗算の並列化においては、並列化された FFT を用いる必要があるが、並列 FFT ライブラリである FFTE を Xeon Phi 向けに最適化したものを用いた。多倍長精度浮動小数点加減乗算においては自動ベクトル化により Xeon Phi の SIMD 命令を用いるとともに、OpenMP による並列化も行っている。

性能評価にあたっては、Xeon Phi において実現した多倍長精度浮動小数点演算と、多倍長精度演算ライブラリである GMP 6.0.0a との性能比較を行った。加算（$\pi+\sqrt{2}$）、乗算（$\pi\times\sqrt{2}$）のそれぞれに対して 10 進数の桁数nを2^{10}～2^{28}まで変化させて連続 10 回実行し、その平均の経過時間を測定した。

評価環境として、Intel Xeon Phi 5110P（8GB、1.053 GHz、60 core）を用いた。実現した多倍長精度浮動小数点演算に対しては、コンパイラは Intel Fortran compiler 15.0.2.164 を用い、コンパイルオプションは“ifort -O3 -mmic -no-prec-div -openmp”を用いた。GMP に対しては、コンパイラは Intel C compiler 15.0.2.164 を用い、コンパイルオプションは“icc-O3 -mmic”を用いた。

GMP においてはアセンブラにより高度に最適化されたルーチンが提供されているが、Xeon Phi は x86 64 アーキテクチャの一部の命令をサポートしていないため、アセンブラによるルーチンを無効にしてビルドを行った。Xeon Phi あたりのスレッド数は 1 および 240 に設定し、ネイティブモードで実行を行った。なお、GMP は並列実行をサポートしていないため、1 スレッドにより実行を行った。

実現した多倍長精度浮動小数点演算と GMP の性能比較について、加算の性能を図 11 に、乗算の性能を図 12 にそれぞれ示す。図 11 から、加算においては 1 スレッドでは実現した多倍長精度浮動小数点演算よりも GNU MP よりも高速であるが、240 スレッド実行において$n \geq 2^{19}$では実現した多倍長精度浮動小数点演算が GNU MP よりも高速であることが分かる。

また、図 12 から、乗算においては 1 スレッドでは$n = 2^{10}$を除いて、実現した多倍長精度浮動小数点演算が GNU MP よりも高速であることが分かる。さらに、実現した多倍長精度浮動小数点演算において 1 スレッドよりも 240 スレッドが高速になるのは$n \geq 2^{15}$であることも分かる。

多倍長精度浮動小数点演算において、キャリーやボローの処理をベクトル化するとともに OpenMP による並列化を行うことで、桁数が大きな領域においては GMP に比べて高速に計算できることを示した。

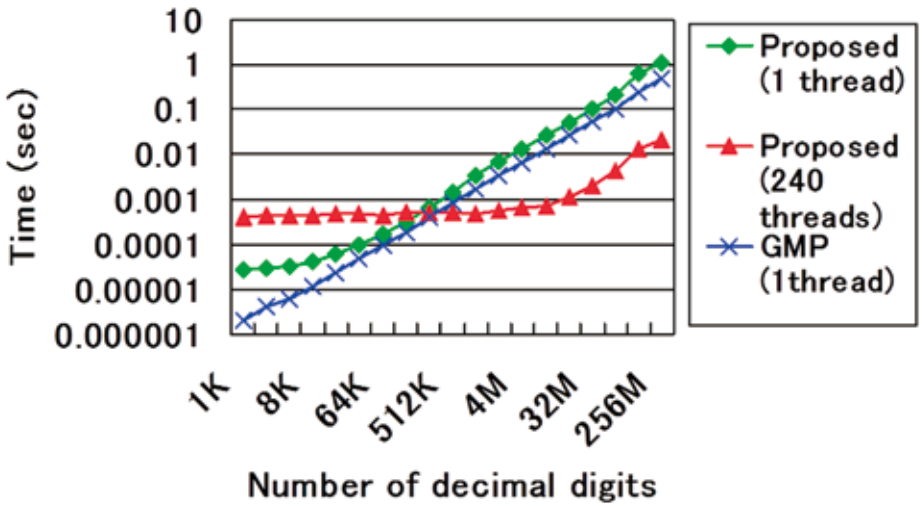


図 11: 多倍長精度浮動小数点加算の性能（Xeon Phi 5110P、240 スレッド）

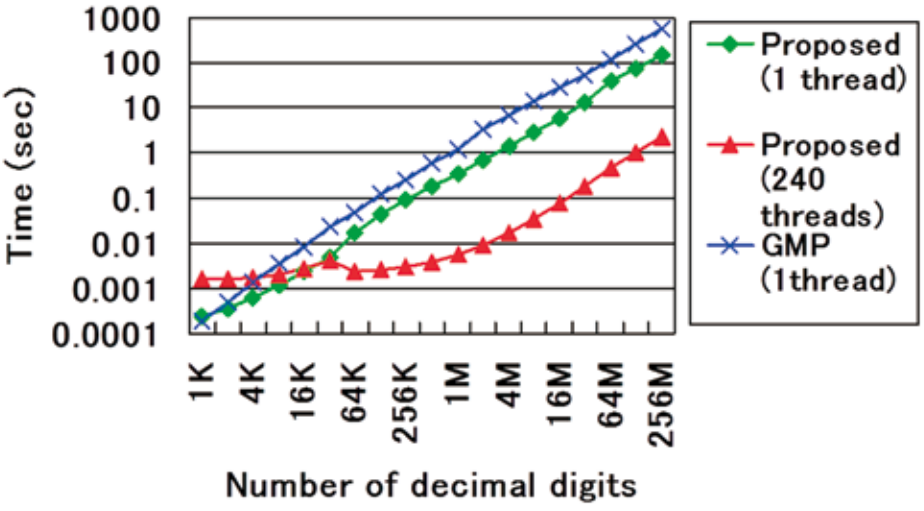


図 12: 多倍長精度浮動小数点乗算の性能（Xeon Phi 5110P、240 スレッド）

【１０】SIMD 命令を用いた整数除算の高速化（高橋）

整数除算は多くのアプリケーションで広く用いられている演算の一つである。一般的に除算は加減乗算に比べて遅いことが知られている。多くのプロセッサでは整数加減乗算の SIMD 命令がサポートされているが、整数除算の SIMD 命令をサポートしているプロセッサはほとんど存在しないのが現状である。また、逆数を用いて整数除算を求めるアルゴリズムが提案されているが、いずれも SIMD 化は考慮されていない。本研究では、SIMD 命令を用いて複数の被除数と除数に対する符号なし 64 ビット整数除算を高速化し性能評価を行った。

符号なし整数除算は被除数をa、除数をbとすると、商$q = \lfloor a/b \rfloor$および剰余$r = a - bq$ $(0 \leq r < b)$で定義される。提案手法では、IEEE 754 規格に準拠した浮動小数点演算を用いて符号なし 64 ビット整数除算を行う。除算は Newton-Raphson 法を用いると効率よく計算できることが知られている。Newton-Raphson 法によりa/bを計算するには、まず$1/b$を$f(x) = 1/x - b = 0$の解として以下の反復で計算する。

$$x_{k+1} = x_k + x_k(1 - bx_k) \qquad (1)$$

ここで、x_kと$(1 - bx_k)$の乗算はx_{k+1}の半分の精度で行うことができる。また、最後の反復ではa/bを

$$a/b \approx (ax_k) + x_k\{a - b(ax_k)\} \qquad (2)$$

として計算することで、aとx_kの乗算およびx_kと$\{a - b(ax_k)\}$の乗算はa/bの半分の精度で行うことができる。

Newton-Raphson 法は 2 次収束するので、符号なし 64 ビット整数どうしの除算を行う場合、式(1)により$1/b$を 32 ビット以上の精度で計算した後に、式(2)を計算すればよいことが分かる。提案手法では、まず単精度浮動小数点演算により初期値$x_0 \approx 1/b$を 24 ビットの精度で計算する。次に、倍精度浮動小数点演算により式(1)を 1 回反復すると$x_1 \approx 1/b$の精度は 48 ビットとなる。最後の反復では式(2)において倍精度浮動小数点演算によりaとx_kの乗算およびx_kと$\{a - b(ax_k)\}$の乗算を行い、それ以外は符号なし 64 ビット整数演算により計算することで、符号なし 64 ビット整数の商$q \approx \lfloor a/b \rfloor$が得られる。この商$q$は$\lfloor a/b \rfloor$よりも 1 だけ少ない場合があるため、符号なし 64 ビット整数演算により剰余$r = a - bq$を計算し、もし$r \geq b$である場合にはqに 1 を加える処理を行う。

提案手法では、IEEE 754 規格の最近接丸めではなく、$-\infty$への丸め（切り下げ）を用いる。この場合、初期値x_0の計算において符号なし 64 ビット整数の除数bを単精度浮動小数点数に変換する際にb以下の値に切り下げられるため、$x_0 > 1/b$となることがある。ところが、倍精度浮動小数点演算により式(1)を 1 回反復した後には$x_1 \leq 1/b$となるため、式(2)において$a - b(ax_k) \geq 0$となり、符号なし 64 ビット整数演算により式(2)を計算しても問題ないことが分かる。

性能評価にあたっては、提案手法に基づく符号なし 64 ビット整数除算と、Intel64 アーキテクチャの符号なし 64 ビット整数除算命令である div 命令、Intel SVML（Short Vector

Mathematical Library）に含まれている符号なし 64 ビット整数除算の組み込み関数との性能比較を行った。被除数は0～$2^{64}-1$の範囲の乱数とし、除数は1～$2^{64}-1$の範囲の乱数とした。256 要素の符号なし 64 ビット整数除算を 100 万回実行し、その平均の経過時間から 1 秒あたりの符号なし 64 ビット整数除算回数（Mops）を算出した。

評価環境として、Intel Xeon E5-2670 v3 および Intel Xeon Phi 5110P の 1 コア、1 スレッドを用いた。コンパイラは Intel C compiler 16.0.2.181 を用い、コンパイルオプションは Xeon E5-2670 v3 に対しては“icc -O3 -xHOST”を、Xeon Phi 5110P に対しては“icc -O3 -mmic”を用いた。上記のコンパイルオプションを用いた場合、提案手法に基づく符号なし 64 ビット整数除算では、Xeon E5-2670 v3 では逆数の近似値を 11 ビットの精度で計算する vrcpps 命令が、Xeon Phi 5110P では逆数の近似値を 23 ビットの精度で計算する vrcp23ps 命令が生成される。

提案手法に基づく符号なし 64 ビット整数除算と、Intel64 アーキテクチャの符号なし 64 ビット整数除算命令である div 命令、そして Intel SVML に含まれている符号なし 64 ビット整数除算の組み込み関数（Xeon E5-2670 v3 では_mm256_ div_epu64、Xeon Phi 5110P では_mm512_ div_ epu64）の性能を表 2 に示す。表 2 から、Xeon E5-2670 v3、Xeon Phi 5110P のいずれにおいても提案手法が div 命令や SVML よりも高速であることが分かる。

Newton-Raphson 法を用いるとともに、単精度および倍精度浮動小数点演算を用いて SIMD 化を行うことで Intel64 アーキテクチャの符号なし 64 ビット整数除算命令や Intel SVML に比べて高速に整数除算が行えることを示した。

表 2: 符号なし 64 ビット整数除算の性能（Mops）

	Xeon E5-2670 v3	Xeon Phi 5110P
提案手法	129.598	33.692
div 命令	103.234	10.450
SVML	126.022	29.624

【１１】データインテンシブサイエンスのためのシステムソフトウェア（建部、川島）

本研究では分散ファイルシステム、大規模データ処理実行基盤の研究を実施し、ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェアの設計を行い、性能評価を行った。以下、項目別に記述する。

・分散ファイルシステム

研究の狙いは、CPU コア数の増加に対し、アクセス性能がスケールアウトし、かつアクセス応答時間が長くならない分散ファイルシステムの研究開発を行うことである。本年度は、これまで研究開発を行ったメタデータサーバ、ローカルストレージをベースに分散ファイルシステムの設計を進め、評価を行った。まず、ローカルストレージを遠隔クライアントからアクセス可能にするためのサーバ設計を行った。このサーバ設計にあたり、クライアントが自ノードの場合は、サーバを経由せず直接ローカルストレージをアクセスする。これによりローカルアクセスの性能向上を図った。図 13 に 512 バイトブロックでアクセスした場合の I/O バンド幅を示す。このベンチマークではそれぞれのサーバノードに 16 クライアントを起動した。サーバを経由しないで直接ローカルストレージをアクセスすることにより 1.9 倍の性能向上を実現した。

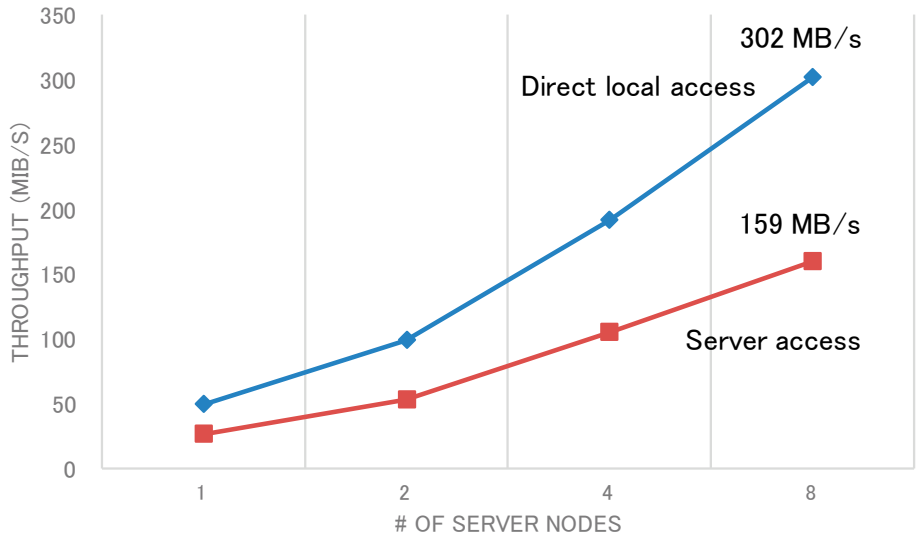


図 13: ストレージサーバ数を増加させたときの 512 バイトブロックアクセス性能

分散ファイルシステムの設計において、ファイルを作成する場合は、まずローカルストレージの利用を試み、利用できない場合はジャンプコンシステントハッシングによりリモートストレージを選択し、オブジェクトを作成し、そのエントリをメタデータサーバに登録する。ただし、この設計の場合、ファイルを作成するたびにストレージにオブジェクトを作成するためファイル作成遅延が増えてしまう。この問題を解決するために、バルクオブジェクト作成、オブジェクトプリフェッチングを提案し、ファイル作成のたびにオブジェクトを作成す

るのではなく、予め複数のオブジェクトを作成することによりファイル作成遅延を隠蔽する性能改善を行った。図 14 にメタデータサーバ数を変更したときのファイル生成性能を示す。このグラフは一秒間に生成したファイル数を示している。

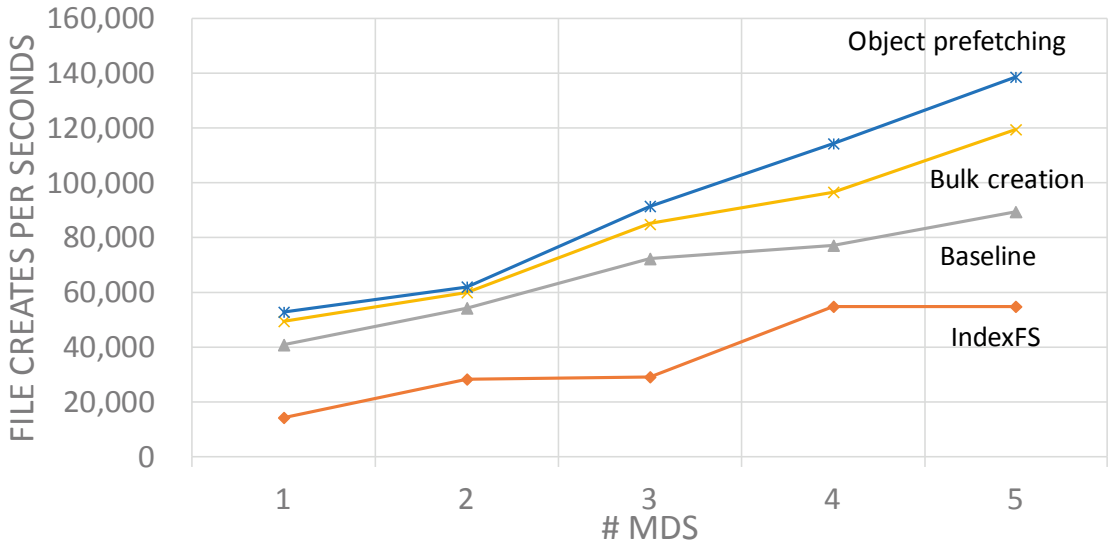


図 14: メタデータサーバ数を変更したときのファイル生成性能

IndexFS は米 CMU の研究者が SC2014 で発表したファイルシステムである。128 メタデータサーバで秒間 842,000 ファイルを作成したと報告され、現時点で最も高速と思われるファイルシステムである。評価では 128 ノードまで準備ができず 5 ノードまでであるが、IndexFS と性能を比較したところ、最適化をしていない baseline の性能でも、同じサーバ数において IndexFS の性能を上回っていた。さらに、バルクオブジェクト生成、オブジェクトプリフェッチングの最適化を行うことにより、5 ノードで毎秒 139,000 ファイルを生成する性能を示した。メタデータサーバ数を増やすことで性能が向上しており、性能向上がこのまま続くとすると 36 ノードで毎秒 1,000,000 ファイルの生成が可能となる。本成果は平成 28 年 12 月に IEEE データサイエンスとシステムに関する国際会議（DSS）で発表した。

・大規模データ処理実行基盤

本研究項目ではデータインテンシブサイエンスのアプリケーションを効率的に実行するため、MPI-IO、大規模ワークフロー実行、MapReduce 処理、バッチキューイングシステム、データベース管理システムなどの実行環境の研究開発を行う。本研究提案で研究開発する分散ファイルシステムは、全体としてのファイルアクセス性能はスケールアウトさせるように設計、実装を行っているが、ファイルアクセス性能が非均一である。そのため、効率的に利

用するためには、ファイルアクセスの局所性の利用とデータ移動を最小化するプロセススケジューリングが重要となる。

大規模ワークフロー実行の研究では、本年度は信頼性を向上させるため耐障害性の向上に関する研究を行った。これまでも、途中で中断したワークフローは、再実行することにより再開することが可能であったが、この再実行を自動化する試みである。障害については、ワークフロータスクの一時的な障害、ノード障害、タスクのエラーを考慮する。Pwrake ワークフローシステムのマスターの障害については、マスターノードは 1 ノードであるため、ワーカノードに比べ障害の可能性が極めて低く、手動で再実行することとする。タスクの一時的障害については、失敗したタスクを再実行する。他のノードで再実行しても異常終了する場合は、そのタスクにエラーがあると判断し、ワークフローを中断する。また、あるノードの通信の接続が切れる、あるいはそのノードで複数のタスクが異常終了した場合は、そのノードに障害が発生したと判断し、そのノードを切り離してワークフローを引き続き実行する。タスクを意図的に異常終了させた場合でも正しくワークフローを引き続き実行できることを確認した。また、ワークフローの入出力ファイルは Gfarm ファイルシステムに格納しているが、Gfarm ファイルシステムのサーバ障害についても評価を行った。Gfarm ファイルシステムのサーバ障害に対応するため、Gfarm ファイルシステム中のファイルに対し、ファイル複製を作成する設定にする必要がある。複製を作成した場合でもワークフローの実行時間が数%の増加でおさまること、また、Gfarm のサーバを意図的に異常終了させても正しくワークフローを引き続き実行できることを確認した。本成果は、平成 28 年 11 月に SC2016 併設のクラウド、グリッド、スーパコンピュータにおけるメニータスクコンピューティングに関する国際ワークショップ（MTAGS）で発表した。

バッチキューイングシステムの研究では、本年度はスケジューリング手法の改良を行った。これまでの手法は、CPU 負荷とアクセス局所性の指標を、パラメタ β を用い統合していたが、最適なパラメタ β の決定が難しかった。一方で、CPU 負荷については、動的な CPU 負荷を指標にするより、利用コア数に相当するスロット数で割り当てることで十分である。このことを利用し、提案手法の方針を見直した。具体的には、CPU 負荷については考慮せず、スロット数だけを考慮する。また、それぞれのジョブについて CPU インテンシブであるか、I/O インテンシブであるかを示す指標として RDR（Remote Degradation Ratio）を定義する。RDR はそれぞれのジョブに対し定義され、全てのアクセスがローカルストレージであった時のジョブの実行時間を L、全てのアクセスがリモートストレージであった時のジョブの実行時間を R としたとき、(R-L)/L で定義する。この指標は、リモートアクセスによるジョブの実行時間の延びの程度を示し、つまり I/O インテンシブな度合いを示している。改善したスケジューリング手法では、スケジューリングのスコアとして RDR にリモートにあるファイルサイズの割合をかけたものを用いる。スコアが大きい場合は、実行時間の延びが大きいこと

を示し、スコアが小さくなるようにスケジューリングを行う。この手法の評価を行い、これまでの手法より性能を向上させられることを示した。本成果は、平成 28 年 11 月に SC2016 併設のクラウドにおけるデータインテンシブコンピューティングに関する国際ワークショップ（DataCloud）で発表した。

【１２】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設計の最適化をすすめた。本年度は、フラッシュストレージを前提としてオブジェクトの生成、参照の性能を向上させるため設計してきたローカルオブジェクトストアについて、これまでの成果をまとめた論文が Journal of Information Processing に掲載された。さらに、数多くのアプリケーションからのアクセスを可能にするため、分散メタデータサーバ PPMDS を用いて、POSIX インターフェースをもつ分散オブジェクトストアの設計を行った。POSIX インターフェースはアプリケーションが標準的に用いているインターフェースであり、このインターフェースにより利用するアプリケーションの数を飛躍的に向上させることができる。分散メタデータサーバ PPMDS は POSIX インターフェースで必要な階層的な名前空間を分散サーバで管理する。またオブジェクトはジャンプコンシステントハッシングを用いて分散させる。

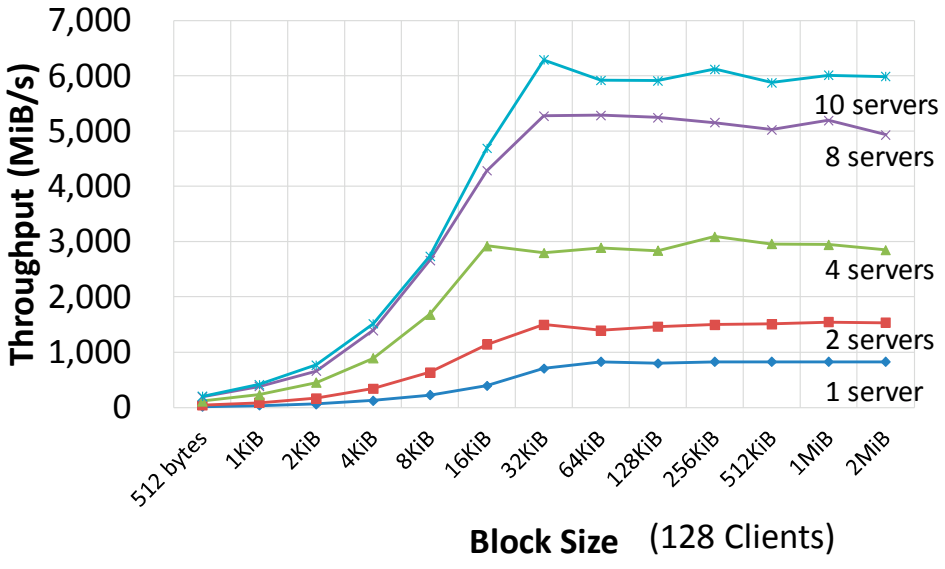


図 15: 分散オブジェクトストアのノード数を変えた時の書込性能

図 15 に HPC でよく用いられる IOR ベンチマークにより、128 クライアントプロセスがそれぞれ 128 MiB のオブジェクトを生成するときの書き込み性能を示す。分散オブジェクトス

トアの数を増やすにつれ、およそ 630 MiB/s の書き込みバンド幅が増加しており、スケーラブルな性能向上を示している。書き込み性能については、ジャンプコンシステントハッシングにより偏りのない書き込みを行っているため、サーバ数が増えた時にスケーラブルな性能向上が期待される。本成果は、データサイエンスシステムに関する IEEE 国際会議で発表した。

また同じプロジェクトの共同研究者である理化学研究所の三好グループとのコデザインとして、全メッシュ点についての近傍データの検索の高速化を図った。近傍データは移動するため、全メッシュ点について検索は一度しか行われない。そのため、インデックス生成と検索の両方を合わせたうえでの高速化が重要となる。このため、STR R (sort-tile-recursive-R) 木におけるインデックス生成の最適化として、データを一斉に用いてインデックス生成を行うバルクローディング手法の開発と、並列ラディックスソート、並列ノードパッキングによる高速化を行った。検索についてもマルチスレッドによる並列検索と並列擬陽性チェックによる最適化を行った。これにより従来のR木を用いた方法に比べ26.8倍の高速化を達成した。

【１３】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、ディレクトリクオータ管理機能の設計と実装、並列コピーのスケジューリングの改善、分散メタデータサーバの設計、RDMA（Remote Direct Memory Access）による高速アクセス機能の実装を行った。ディレクトリクオータ管理機能は、これまでのユーザ、グループによるファイル数、ファイルサイズなどのクオータ管理とは異なり、あるディレクトリ以下のファイルについてクオータ管理を行う機能である。このとき、ユーザ、グループは問わない。ディレクトリクオータ機能は UNIX では XFS で実装されている。XFS ではディレクトリクオータはプロジェクトクオータと呼ばれ、グループクオータ機能との排他利用である。そのため、利用用途が限られてしまう。Gfarm ファイルシステムでは、グループクオータ機能とともに用いることができ、かつオーバヘッドの少ないディレクトリクオータ機能の設計を行った。本機能は、HPCI 共用ストレージにおいてサブグループのクオータ管理を実施したいという要求を満たすために設計、実装したものである。並列コピーのスケジューリングの改善は、並列コピーを異なるクライアントで同時に実行したとき、書き込み先を衝突しないようにするものである。同時に実行する場合、書き込み先情報が同じであり、ファイル容量、CPU 負荷で選択すると同じ書き込み先が選ばれてしまう。これを回避するため、なるべく多くの書き込み先に対しラウンドロビンでスケジューリングする方式を設計し、実装した。分散メタデータサーバの設計は、ファイルシステムにおけるファイル数の増加、また多数クライアントからのア

クセス集中を回避するためのものである。RDMA による高速アクセス機能は、ファイルアクセスを高速化するため、データ転送に RDMA を用いるものである。設計のポイントは、これまでの RDMA ではないソケット通信のクライアントからのアクセスと RDMA によるアクセスの両方を可能にすることと、安定性を保持するためあまり大きな開発を必要としないで性能向上を図ることである。そのため、主要なデータアクセスプロトコルである PREAD と PWRITE プロトコルに対し RDMA 版を設計し、RDMA が利用できない環境の場合はこれまでのソケットベースの PREAD と PWRITE を用いることとした。性能評価の結果を図 16 に示す。

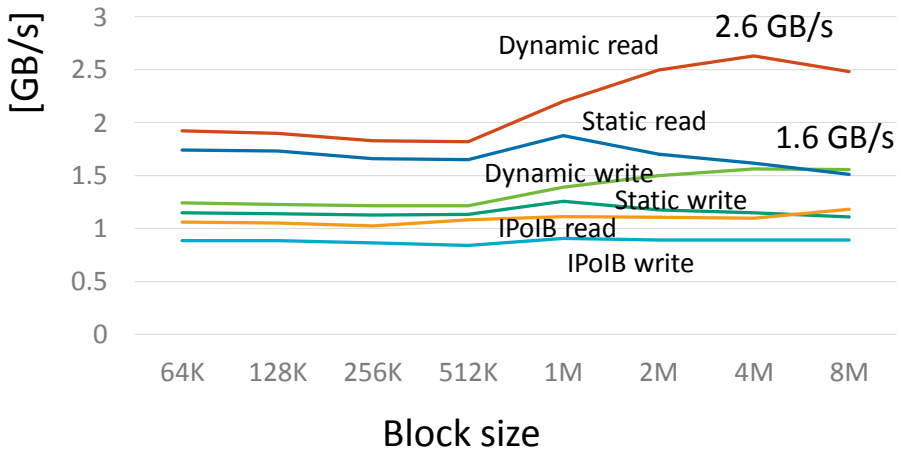


図 16: それぞれの通信方式におけるアクセスバンド幅

RDMA を用いたアクセスではメモリ登録を動的に行う Dynamic 方式の性能が高く、読込みで 2.6 GB/s、書込みで 1.6 GB/s の性能を達成した。これらの成果は平成 29 年 3 月 27 日にリリースした Gfarm バージョン 2.7.3 に含まれている。

【１４】高性能なログ先行書込みに関する研究（川島、建部）

フラッシュストレージをログ用のストレージデバイスとする時にふさわしい WAL プロトコルとして P-WAL を提案した。フラッシュストレージは複数のメモリチップに対して並列にアクセスすることで高い性能を発揮する。P-WAL はフラッシュストレージの特性を活用し、各ワーカーが専用の領域にログを書込む並列ログ書込み方式を用いる。この方式により従来の直列 WAL 方式で発生する、排他制御処理とストレージ I/O に伴う性能低下問題を解決する。P-WAL をトランザクションシステム上で実装し、性能評価を行った。その結果を

図 17 に示す。P-WAL は直列 WAL 方式に対してマイクロベンチマークで 10.0 倍、TPC-C ベンチマークにおいて 2.3 倍の性能向上を示した。本成果は情報処理学会論文誌で発表した。

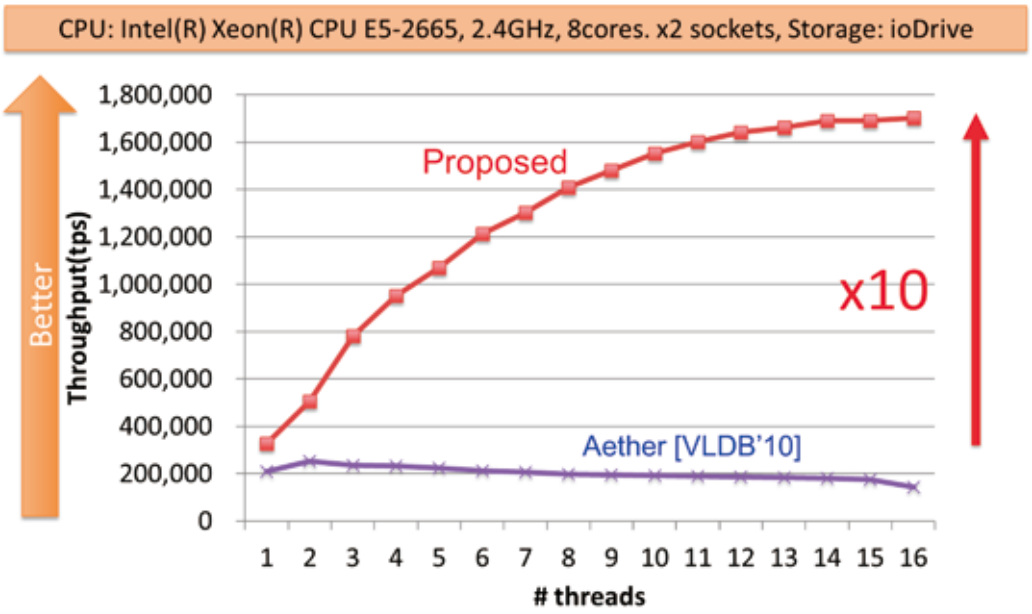


図 17: P-WAL の性能評価

【１５】高性能な分散トランザクション処理に関する研究（川島、建部）

分散データベース管理システムにおいて外部キー制約や二次索引、実体化ビューの管理を行うための高性能な処理方式として Read Atomic Multi-Partition (RAMP)トランザクションがある。RAMP トランザクションは隔離性を緩和することで高性能化が実現されているが、それを先進的デバイスによって高性能化する技法は未開拓である。そこで、本研究では高性能インターコネクトである InfiniBand を利用し、Remote Direct Memory Access (RDMA)の機能を用いて RAMP トランザクションを高速化する手法を提案した。まず、RDMA-Write による GET/PUT オペレーションの高速化手法として GET+/PUT+方式を提案した。続いて、RDMA-Read による更なる GET オペレーションの高速化手法として GET*方式を提案した。提案手法の評価のため、プロトタイプ In-Memory Key-Value Store を実装した。Yahoo! Cloud Serving Benchmark を用いた実験において、従来方式と比べて提案手法は最大 2.67 倍の高速化を達成した。これを図 18 に示す。研究成果は IEEE BigComp れを式に採択され、Best Paper Award on Big Data Processing (runners-up 同等)を受賞した。その後、情報処理学会論文誌に採録された。

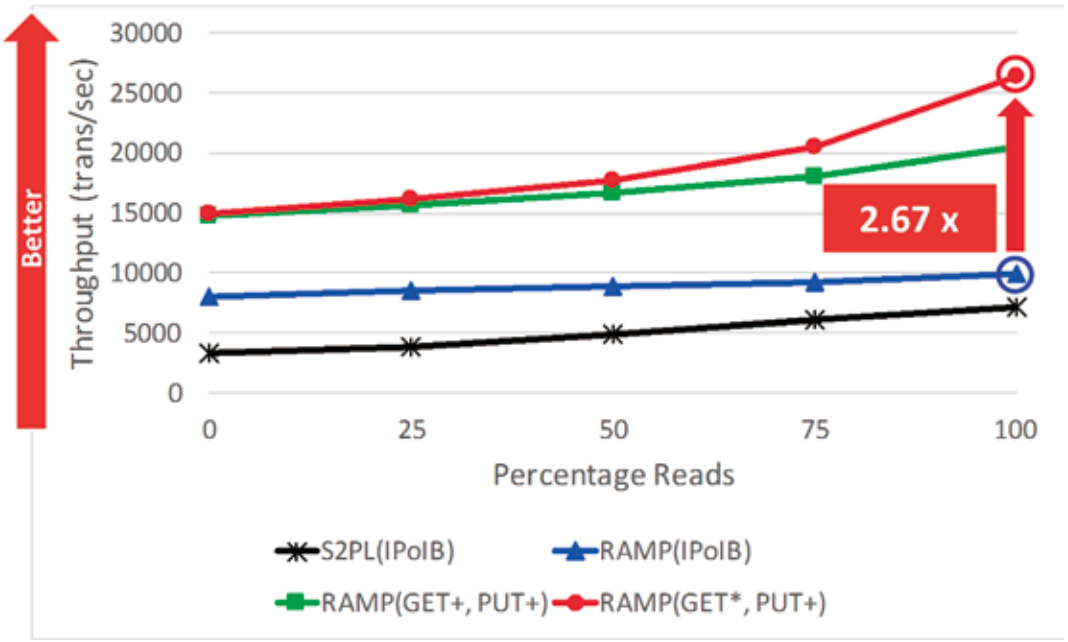


図 18: InfiniBand RDMA による In-Memory Key-Value Store 実装の評価

【１６】高性能な空間索引に関する研究（川島、建部）

本研究の対象は空間結合殻カウントである。空間結合殻カウントとは、空間結合を行った後にカウントを行う処理である。空間結合殻カウントは天文学においては、halo 周辺に存在する particle の数を数え上げる処理となる。この処理はシミュレーション結果に対して一度だけ実行される。空間結合殻カウントの効率化には、効率的な空間結合が必要である。そこで我々は CPU に最適化した sort-tile-recursive-R 木を提案した。提案手法は並列基数整列法、並列ノード梱包法、そして single instruction multiple data 命令を活用した。天文データを用いた実験において、提案手法は高性能 R 木に対して 26.8 倍の性能改善を示した。これを下図に示す。我々は更に主記憶に収まらない大規模データを扱うために、局所実体化法を提案した。この方式を効率化するために、我々は構築・検索・破壊パイプライン法を提案した。提案方式は従来技法に比べて 27.5 倍の高性能化を達成した。コードは GitHub で公開済である。また、成果は Workshop of Big Data Challenges、Research、and Technologies in the Earth and Planetary Sciences held as part of the IEEE Big Data Conference で報告した。

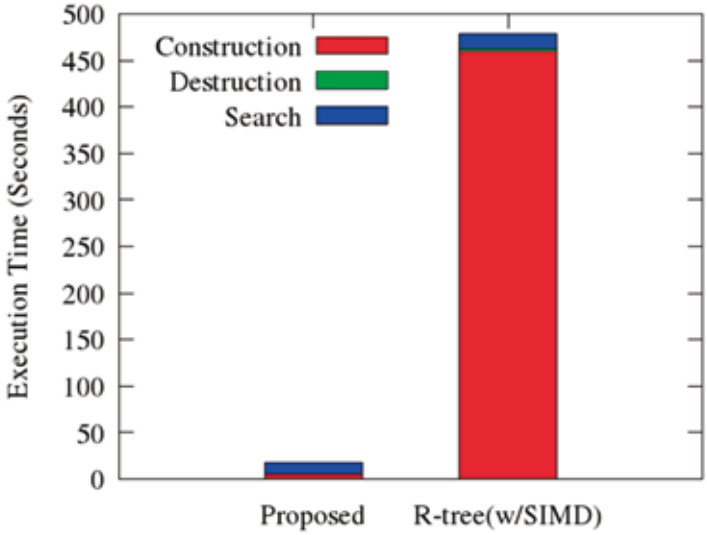


図 19: 高性能 R 木における性能向上

【１７】高性能な暗号化データ処理に関する研究（川島、建部）

暗号化データベースシステムでは暗号文上での関係演算子の使用が可能である。一方、このシステムの欠点はデータ処理コストの増加にある。本研究では加算演算子に着目し、暗号化データベースシステムにおける総和計算処理の並列化手法を提案した。本手法はタスク並列性を活用し、細粒度のデータオブジェクトを各スレッドに割り当てることで高速化を実現した。プロトタイプシステムを C++および OpenMP を利用して実装し、提案手法の評価を行った。提案手法をナイーブな手法と比較した結果、グルーピング演算を伴う総和計算問合せ処理において、提案手法は最大で 69.91 倍の高速化を達成した。この成果の一部は IEEE BigComp の成果で報告した。

【１８】複数シフト・複数右辺ベクトルをもつ連立一次方程式に対する高精度数値解法に関する研究（多田野）

複数右辺ベクトルをもつ連立一次方程式 $AX = B$（以下、シード方程式）と、複数シフトパラメータをもつ連立一次方程式 $(A + \sigma_\ell I)X^{(\sigma_\ell)} = B$（以下、シフト方程式）を同時に解く数値解法であるシフト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。シフトブロッククリロフ部分空間反復法では、シード方程式はブロッククリロフ部分空間反復法によって解かれる。ブロッククリロフ部分空間反復法は、各右辺に対してクリロフ部分空間反復法を適用するよりも少ない反復回数で近似解が得られる可能性がある解法である。また、シード方程式に対しては、生成されるブロッククリロフ部分空間がシード方程式と等しい性質を用いることにより、少ない計算量で近似解更新を行うことができる。しかしながら、シフト方

程式の近似解の精度は誤差の影響を受けやすく、反復の停止条件を満たした場合でも目的の精度に到達しないことがある。今年度は、シフトブロッククリロフ部分空間反復法の 1 つである、Shifted Block BiCGGR 法におけるシフト方程式の近似解の精度劣化原因について、数値的に解析を行った。

シフトブロッククリロフ部分空間反復法では、シード方程式の残差行列とシフト方程式の残差行列が同一のブロッククリロフ部分空間に属することを前提条件として、アルゴリズムが構築されている。本研究では、シード方程式の残差が属する部分空間と、シフト方程式の残差が属する部分空間を最大正準角を用いて比較を行い、近似解の精度劣化との関連性を調べた。最大正準角は 2 つの空間を比較する際に用いられる指標であり、最大正準角が 0°であれば 2 つの空間は等しいことを示す。

Shifted Block BiCGGR 法において、シフト方程式の近似解精度劣化と、2 つの残差が属する部分空間の差の関連性を調べる。シフト方程式の計算を単純に実装したものを従来版と呼び、縦長行列と小行列の積がなるべく現れないように改良したものを改良版と呼ぶ。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、シフトパラメータ σ は 1.0×10^{-4} とした。図 20 に、右辺ベクトル数を変化させたときのシード方程式、シフト方程式（従来版）、及びシフト方程式（改良版）の真の相対残差の変化を示す。なお、真の相対残差は近似解の精度の指標であり、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している。図 20 より、シード方程式の近似解は右辺ベクトル数によらず、高精度に計算できていることがわかる。しかしながら、シフト方程式の近似解は、右辺ベクトル数が増加すると真の相対残差の値が大きくなっており、高精度の近似解が得られていないことがわかる。右辺ベクトル数が 24 本の場合のシフト方程式の真の相対残差は、従来版では 2.6×10^{-4}、改良版では 6.9×10^{-8} であった。

図 21 に、右辺ベクトル数が 24 本の場合の 2 つの部分空間が成す最大正準角の変化と、真の相対残差の変化を示す。図 21 (a)に示すように、最大正準角は反復開始直後は 0° となっており、シード方程式とシフト方程式の残差が属する部分空間は等しい。しかしながら、従来版では 150 回目の反復付近で最大正準角が急激に大きくなり、230 回目の反復付近でほぼ 90° になった。一方、改良版では 300 回目の反復付近で最大正準角が急激に大きくなっており、400 回目の反復付近でほぼ 90° になった。また、図 21 (b)に示すように、真の相対残差は最大正準角が 90° に収束するまでは減少しており、最大正準角の増加が遅い改良版では真の相対残差はより小さな値に到達できている。この結果より、シード方程式とシフト方程式の残差が同じ部分空間に属するという条件が崩れると、近似解精度が悪化することがわかった。最大正準角の増加を抑える手法を構築できれば、今後更なる近似解精度の向上が可能と考えられる。

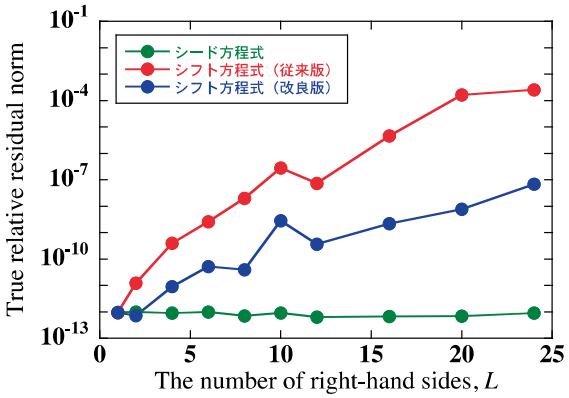
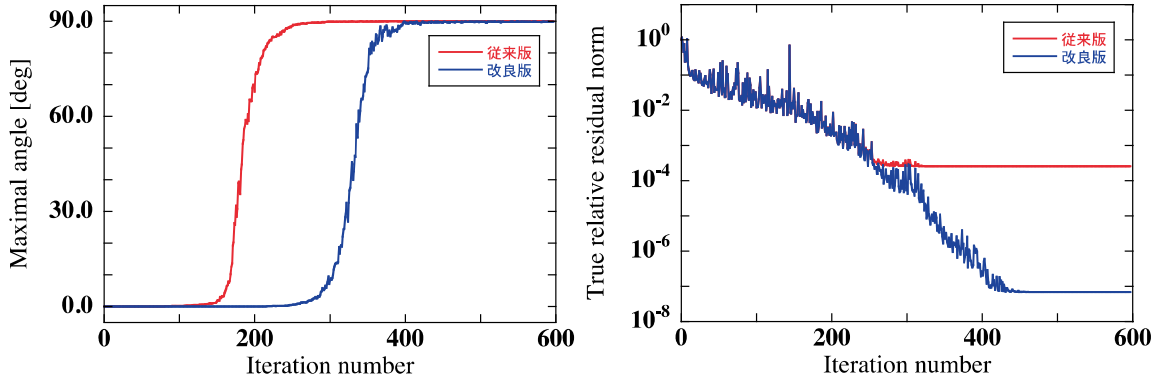


図 20: 右辺ベクトル数の変化に対する真の相対残差の変化



(a) 2 つの部分空間が成す最大正準角変化　　(b) 真の相対残差の変化

図 21: 右辺ベクトル数 L=24 の場合の 2 つの部分空間が成す最大正準角変化と真の相対残差の変化

【１９】Large Eddy Simulation で現れる 3 次元ポアソン方程式の求解高速化に関する研究（多田野）

地球環境研究部門の日下グループとの共同研究として、Large Eddy Simulation (LES) で現れる 3 次元ポアソン方程式の求解高速化についての研究を行った。気象分野における LES では、圧力や速度の時間発展計算に 3 次元ポアソン方程式が現れ、その求解に多くの計算時間を要していることから高速化が必要不可欠となっている。この 3 次元ポアソン方程式は有限差分法を用いて離散化され、最終的に大規模対称疎行列をもつ連立一次方程式に帰着される。

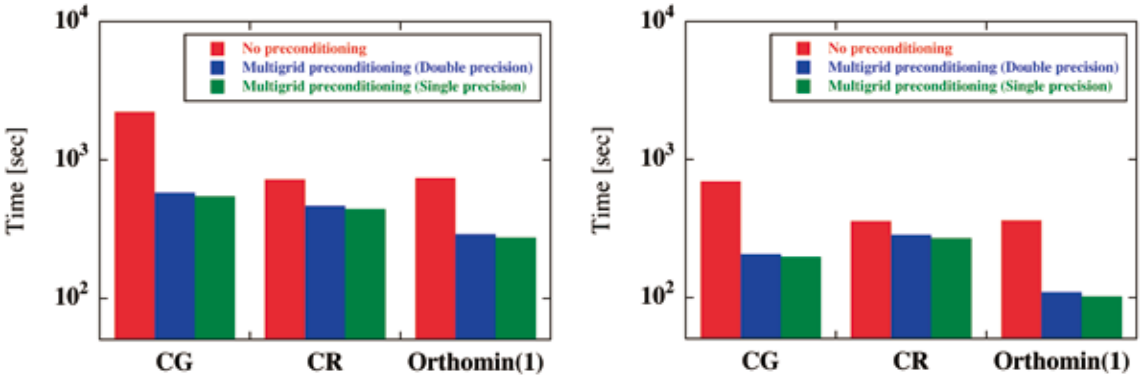
LES で現れる連立一次方程式に対してマルチグリッド法を前処理に用いた反復法を適用し、その性能を評価した。用いた反復法は、両側前処理を施した共役勾配法（CG 法）、共役残差法（CR 法）と、右側前処理を施した Orthomin(1)法である。また前処理として、単精度、及び倍精度のマルチグリッド法を用いた。連立一次方程式のサイズは 20,709,376、係数行列

の非零要素数は 144,441,344 であり、LES の 101 タイムステップで現れる 101 本の連立一次方程式を解く。初期解の与え方として、以下の 2 種類を採用した。

- Type I： 全タイムステップで初期解を **0** で与える
- Type II： 第 0 ステップの初期解は **0**、以降は前ステップで得られた近似解を与える

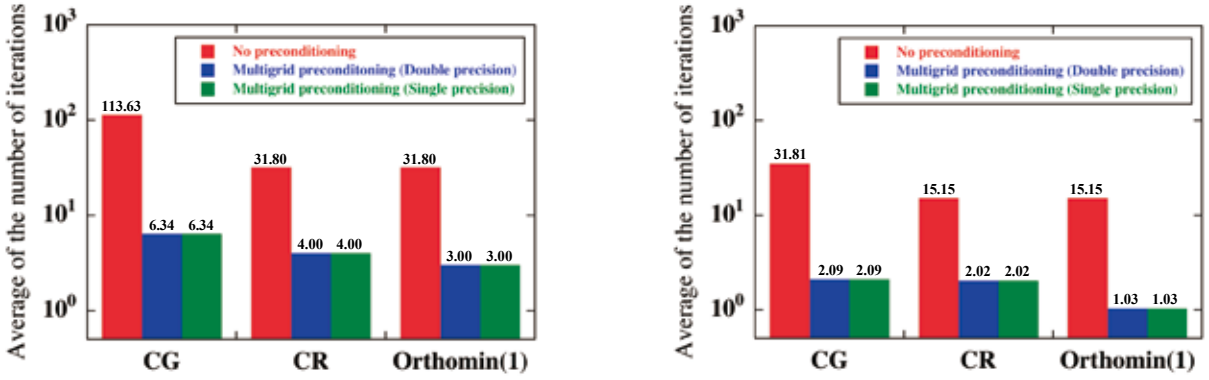
計算機環境として CPU：Intel Xeon E5-2620 2.4GHz（1 コアのみ利用）、メモリ：64GB、コンパイラ：gfortran ver. 5.3.1、コンパイルオプション：-O3 を用いた。

図 22 に 101 本の連立一次方程式を解くのに要した計算時間を示す。CR 法と Orthomin(1) 法は、CG 法よりも高速に求解可能であることがわかった。また、マルチグリッド前処理を用いることにより計算時間を更に短縮することができ、前処理部分のみを単精度で計算することで若干の高速化を図ることができた。図 22 (b) に示すように、前ステップで得られた解を初期解に採用することで高速化が可能であることがわかった。単精度マルチグリッド前処理付き Orthomin(1)法において、前ステップの近似解を初期解に用いた場合が最も高速で、101 本の連立一次方程式を 101.8 秒で解くことができた。また、図 23 に 101 本の連立一次方程式を解くのに要した平均反復回数を示す。単精度と倍精度のマルチグリッド前処理を用いた際の平均反復回数は等しいことから、前処理には単精度のマルチグリッド前処理で十分であることがわかった。



(a) Type I　　(b) Type II

図 22: 101 本の連立一次方程式を解くのに要した計算時間



(a) Type I　　(b) Type II

図 23: 101 本の連立一次方程式を解くのに要した平均反復回数

【２０】高速な Verilog HDL シミュレータの開発（小林）

ハードウェア記述言語による FPGA カスタムコンピューティングシステムの開発は未だデファクトスタンダードであり、RTL (Register Transfer Level)シミュレーションは設計したハードウェアの挙動が設計者の意図と一致しているかを保証する重要な工程である。一方、FPGA 上に大規模なハードウェアを実装するには長い RTL シミュレーション時間を必要とし、それは既存の RTL シミュレータでは現実的な時間内でハードウェアの挙動を検証することが困難であることを意味している。また、FPGA に実装可能な回路規模は、半導体プロセス技術の進歩の恩恵により年々増加している。そのため FPGA に実装されるハードウェアの規模は増大し、それに比例してハードウェアの検証に要する時間は増えることが予想されるので、今後はそのような問題を解決する高速な RTL シミュレーション環境が必要となる。そのため本研究では、既存の 2 つの技術、ArchHDL と Pyverilog を活用した新しい高速な Verilog HDL シミュレータを提案・実装した。図 24 にシミュレータの概要を示す。

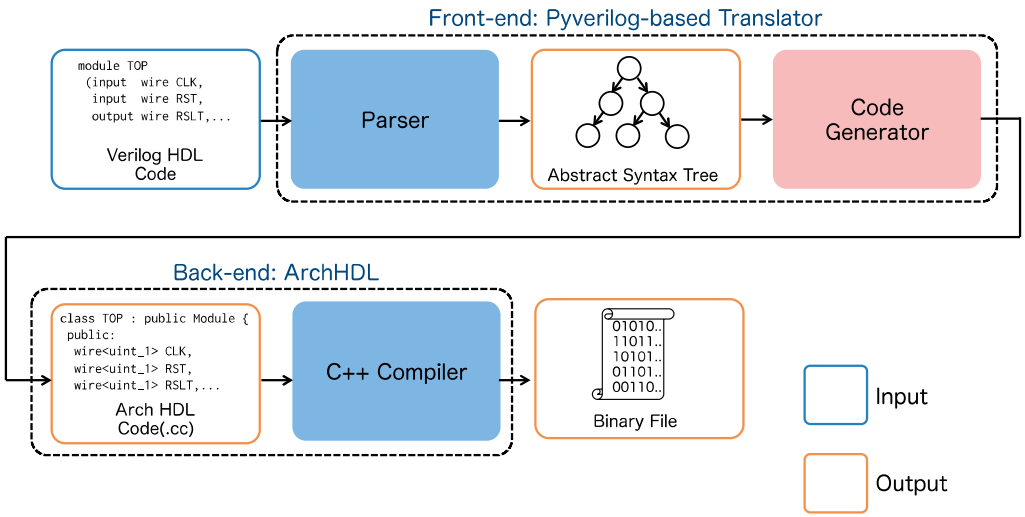


図 24: 高速な Verilog HDL シミュレータの概要

図 25: Verilog HDL コードと ArchHDL コードの対応関係

ArchHDL はシミュレーションエンジンとして利用される。ArchHDL は C++11 をベースとした RTL モデリングとシミュレーションのためのライブラリであり、RTL シミュレーションは OpenMP によって並列化が可能であるため、シミュレーション時間を短縮することが可能である。一方、Pyverilog は Verilog HDL のソースコードから ArchHDL のコードに変換するトランスレータの実装に活用される。図 25 に Verilog HDL コードと ArchHDL コードの対応関係を示す。図中の点線で囲まれた部分がそれぞれのコードが同じ機能を実現している。ここで留意してほしいのが、Verilog HDL コードでは FMUL、 FADD のようなサブモジュールを関数呼び出しのような形でインスタンス化できるが、ArchHDL コードではサブモジュールのポートが呼び出し元のどのワイヤ・レジスタに接続されているかを明記しなければならない。そのため、Verilog HDL コードから ArchHDL のコードに変換するためには、Verilog HDL のソースコードを解析し、どのようなハードウェア構造であるかを示す情報を取得する必要がある。そのためこのトランスレータの実装には、Verilog HDL のデザイン解析・コード生成のための Python ベースのオープンソースツールキット Pyverilog を利用している。Verilog HDL のソースコードをトランスレータによって ArchHDL コードに変換し、それを gcc や Intel Compiler などの標準的な C コンパイラでコンパイルすることによってシミュレーション用の実行バイナリが生成される。

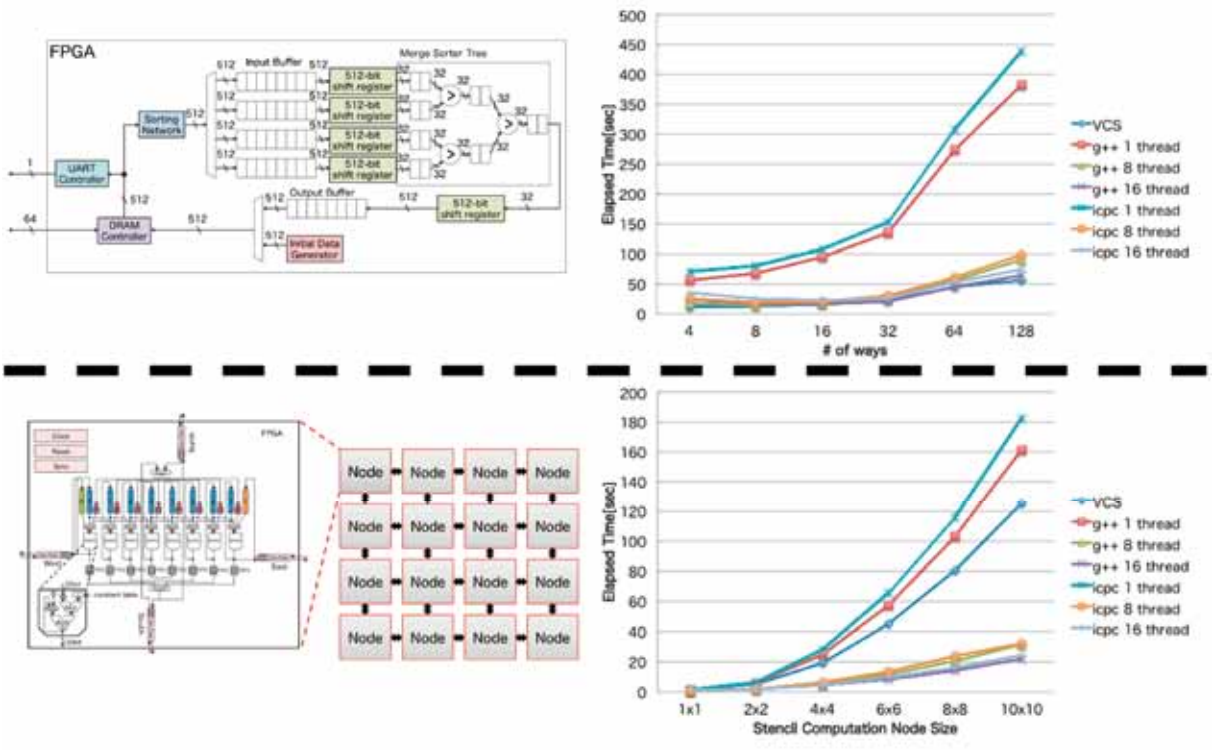


図 26: ソーティング用とステンシル計算用 FPGA アクセラレータの検証に要するシミュレーション時間（上がソーティング用、下がステンシル計算用)。ソーティングアクセラレータのシミュレーション時間における、横軸の way 数はソーティングアクセラレータ内の Merge Sorter Tree の葉の数を意味している。

図 26 に商用の高速なシミュレータである Synopsys VCS (Verilog Compiler Simulator)と開発した Verilog HDL シミュレータのシミュレーション時間を示す。シミュレーション用のハードウェアとして、ソーティングアクセラレータとステンシル計算用アクセラレータを用いた。ソーティングアクセラレータのシミュレーションにおいては、16 スレッド利用した場合のシミュレーション時間は VCS のシミュレーション時間とほとんど同じであった。これは、OpenMP によるシミュレーションの並列化の効果はあるものの、計算機の主記憶にアクセスする頻度が多くオーバーヘッドが顕在化しやすいシミュレーションであることに起因している。一方、ステンシル計算用アクセラレータのシミュレーションにおいては、g++でコンパイルし 16 スレッドを用いた結果、VCS と比較して 1.2 倍から最大 5.8 倍の高速化となった。これは、ソーティングアクセラレータとは異なり、ステンシル計算用アクセラレータは OpenMP によって並列シミュレーションが可能である部分が多い構成であることに起因している。つまりシミュレーションのオーバーヘッドが顕在化しにくいハードウェアであることを意味している。このようにハードウェアの種類によって、VCS に対するシミュレーション

速度比は異なるが、開発したシミュレータに適さないハードウェアであっても VCS と同等、それ以外なら大幅な性能差を達成していることが分かる。

現時点では、開発したシミュレータが対応できる Verilog HDL の構文は少ないため、シミュレーションできるハードウェアはある程度限定されている。そのため、今後は対応可能な構文を増加させてあらゆるハードウェアのシミュレーションを可能にし、開発したシミュレータの有用性をさらに評価していく。

4. 教育

学生の指導状況（学生氏名，学位の種類，論文名）

博士学位論文

1. 鷹津冬将，博士（工学），ポストペタスケールシステム向け高性能分散ファイル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博士論文，平成 29 年 3 月（指導教員：建部修見）
2. 李燮鳴，博士（工学），A Study on Data-Aware Scheduling for Post-Peta Scale Systems（ポストペタスケールシステムにおけるデータ配置を考慮したタスクスケジューリング），筑波大学大学院システム情報工学研究科博士論文，平成 29 年 3 月（指導教員：建部修見）

修士学位論文

1. 桑原悠太，修士(工学)，GPU クラスタにおける GPU セルフ MPI 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 29 年 3 月（指導：朴泰祐）
2. 佐藤賢太，修士（工学），密結合並列演算加速機構における汎用通信ライブラリの実装と評価，筑波大学大学院システム情報工学研究科修士論文，平成 29 年 3 月（指導：朴泰祐）
3. 馬浩宇，修士（工学），Implementation for High Performance Homology Search on Gfarm File System（Gfarm ファイルシステムにおける高性能相同性検索の実装），筑波大学大学院システム情報工学研究科修士論文，平成 29 年 3 月（指導：建部修見）
4. 堀尾健太郎、修士（工学）、暗号化データ処理の高性能化に関する研究、平成 29 年 3 月（指導：川島英之）
5. 神谷孝明、修士（工学）、並列ログ先行書き込み方式に関する研究、平成 29 年 3 月（指導：川島英之）

6. 三橋龍也、修士（工学）、大規模な空間結合集約の高性能化に関する研究、平成 29 年 3 月（指導：川島英之）

卒業論文

1. 松村和朗、学士（工学）、アクセラレータ用言語 OpenACC のマルチ GPU 実行環境への適用、筑波大学情報学群情報科学類卒業論文、平成 29 年 3 月（指導：朴泰祐）
2. 吉川直宏、学士（工学）、RNN による Git のコミットメセージ自動生成、筑波大学情報学群情報科学類卒業論文、平成 29 年 3 月（指導：高橋大介）
3. 岩井厚樹，学士（工学），並列ベンチマークのための同期複数タスク実行フレームワークの設計，筑波大学情報学群情報科学類卒業論文，平成 29 年 3 月（指導：建部修見）
4. 小林淳司，学士（工学），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性能評価，筑波大学情報学群情報科学類卒業論文，平成 29 年 3 月（指導：建部修見）
5. 梶原顕伍、学士（工学）、分散合意手法 Raft の高性能化、筑波大学情報学群情報科学類卒業論文、平成 29 年 3 月（指導：川島英之）
6. 渡邉敬之、学士（工学）、Masstree の効率的な構築に関する研究、筑波大学情報学群情報科学類卒業論文、平成 29 年 3 月（指導：川島英之）
7. 中村泰大、学士（工学）、タイムスタンプに基づく並行実行制御と並列ログ先行書き込みの結合、筑波大学情報学群情報科学類卒業論文、平成 29 年 3 月（指導：川島英之）

5. 受賞、外部資金、知的財産権等

【受賞】

1. 情報処理学会 HPCS2016 シンポジウム最優秀論文賞：佐藤賢太，藤田典久．塙敏博，松本和也，朴泰祐，Khaled Ibrahim，"密結合並列演算加速機構 TCA による GPU 対応 GASNet の実装と評価"
2. NVIDIA Best Paper Award: Hiroshi Maeda and Daisuke Takahashi, “Parallel Sparse Matrix-Vector Multiplication Using Accelerators”, 16th International Conference on Computational Science and Its Applications (ICCSA 2016), Beijing, China, July 6, 2016.

3. Best paper award on big data processing (runner-up), Naofumi Murata, Hideyuki Kawashima, Osamu Tatebe, Accelerating Read Atomic Multi-partition Transaction with Remote Direct Memory Access, IEEE International Conference on Big Data and Smart Computing, 2017.
4. 平成 28 年度山下記念研究賞（SIGOS）、川島英之、縮退表現に基づくシーケンスパタン集合の圧縮、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）、Vol. 2016-OS-136, No. 13, pp. 1-6.
5. 平成 28 年度電子情報通信学会 情報・システムソサイエティ（ISS）査読功労賞、川島英之
6. 平成 28 年度筑波大学システム情報系教育貢献賞（推薦）、川島英之
7. 筑波大学 BEST FACULTY MEMBER2016，朴泰祐，2017 年 2 月 20 日

【外部資金】

1. JST CREST、朴泰祐（研究代表者）、H23～H29 年度、24,883 千円（H28）、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 「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」
2. 理化学研究所共同研究、朴泰祐（研究代表者）、H27～H31、8,000 千円（H28）「ポスト京のプロセッサアーキテクチャ、電力制御技術、システムソフトウエアおよび数値計算ライブラリに関する研究」
3. JST CREST、高橋大介（共同研究者）、H23～28 年度、14,950 千円(H28)、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」，「数値計算ライブラリによる超並列複合システムの階層的抽象化に関する研究」
4. 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」
5. 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」，「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」，H23 年度～H28 年度，18,200 千円 (H28)（代表：建部修見）
6. 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 (H28)（主たる共同研究者：建部修見）

7. JST CREST、川島英之（共同研究者）、H26～30 年度、15,680 千円（H28）、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」
8. 科学研究費補助金　基盤研究（C)、川島英之（代表）、H28～30 年度、2,100 千円（H28 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」
9. 科学研究費補助金　若手研究（B)、多田野寛人（代表）、H27～28 年度、910 千円（H28 年度）、「複数右辺ベクトルをもつ連立一次方程式に対する高精度・高効率アルゴリズムの開発」

【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. 廣川 祐太, 朴 泰祐, 佐藤 駿丞, 矢花 一浩, "電子動力学シミュレーションのステンシル計算最適化とメニーコアプロセッサへの実装", 情報処理学会論文誌コンピューティングシステム（ACS）, Vol. 9, No.4, pp. 1-14, 2016.
2. 佐藤賢太, 藤田典久. 塙敏博, 松本和也, 朴泰祐, Khaled Ibrahim, "密結合並列演算加速機構 TCA による GPU 対応 GASNet の実装と評価", 2016 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（HPCS2016）論文集, 2016.
3. 廣川祐太, 朴泰祐, 佐藤駿丞, 矢花一浩, "電子動力学シミュレーションのステンシル計算に対するメニーコアプロセッサ向け最適化", 2016 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（HPCS2016）論文集, 2016.
4. Kenta Sato, Norihisa Fujita, Toshihiro Hanawa, Taisuke Boku, Khaled Z. Ibrahim, "GPU-ready GASNet Implementation on the TCA Proprietary Interconnect Architecture", Proc. of CSCI2016 (Int. Conf. on Computational Science and Computational Intelligence 2016), 6 pages, Las Vegas, Dec. 2016.
5. Akihiro Tabuchi, Yasuyuki Kimura, Sunao Torii, Video Matsufuru, Tadashi Ishikawa, Taisuke Boku, Mitsuhisa Sato, "Design and Preliminary Evaluation of Omni OpenACC Compiler for Massive MIMD Processor PEZY-SC", Proc. of

IWOMP2016 (International Workshop on OpenMP (LNCS 9903: OpenMP: Memory, Devices, and Tasks), pp.293-305, Nara, Oct. 2016.

6. Kazuya Matsumoto, Norihisa Fujita, Toshihiro Hanawa, ,Taisuke Boku, “Implementation and Evaluation of NAS Parallel CG Benchmark on GPU Cluster with Proprietary Interconnect TCA”, Proc. of VECPAR2016, 8 pages, Porto, Jul. 2016.
7. Yuta Hirokawa, Taisuke Boku, Shunsuke Sato, Kazuhiro Yabana, "Electron Dynamics Simulation with Time-Dependent Density Functional Theory on Large Scale Symmetric Mode Xeon Phi Cluster", Proc. of PDSEC2016 (in IPDPS2016), 8 pages, Chicago, 2016.
8. Daichi Mukunoki, Toshiyuki Imamura and Daisuke Takahashi, “Automatic Thread-Block Size Adjustment for Memory-Bound BLAS Kernels on GPUs”, Proc. 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-16), Special Session: Auto-Tuning for Multicore and GPU (ATMG), pp. 377-384, 2016. (DOI: 10.1109/MCSoC.2016.32)
9. Daisuke Takahashi, “Automatic Tuning of Computation-Communication Overlap for Parallel 1-D FFT”, Proc. 2016 IEEE 19th International Conference on Computational Science and Engineering (CSE 2016), pp. 253-256, 2016.
10. Daisuke Takahashi, “Implementation of Multiple-Precision Floating-Point Arithmetic on Intel Xeon Phi Coprocessors”, Proc. 16th International Conference on Computational Science and Its Applications (ICCSA 2016), Part II, Lecture Notes in Computer Science, Vol. 9787, pp. 60-70, 2016. (DOI: 10.1007/978-3-319-42108-7_5)
11. Hiroshi Maeda and Daisuke Takahashi, “Parallel Sparse Matrix-Vector Multiplication Using Accelerators”, Proc. 16th International Conference on Computational Science and Its Applications (ICCSA 2016), Part II, Lecture Notes in Computer Science, Vol. 9787, pp. 3-18, 2016. (DOI: 10.1007/978-3-319-42108-7_1)
12. Yoshihiro Oyama, Jun Murakami, Shun Ishiguro, Osamu Tatebe, “Implementation of a Deduplication Cache Mechanism using Content-Defined Chunking”, International Journal of High Performance Computing and Networking, Inderscience, Vol. 9, Issue 3, pp.190-205, 2016 (DOI: http://dx.doi.org/10.1504/IJHPCN.2016.076251)

13. Xieming Li and Osamu Tatebe, “Data-Aware Task Dispatching for Batch Queuing System”, IEEE Systems Journal (DOI: 10.1109/JSYST.2015.2471850) (to appear)
14. Yoshihiro Oyama, Shun Ishiguro, Jun Murakami, Shin Sasaki, Ryo Matsumiya, Osamu Tatebe, “Experimental Analysis of Operating System Jitter Caused by Page Reclaim”, The Journal of Supercomputing, Vol. 72, Issue 5, pp.1946-1972, May 2016 (DOI: 10.1007/s11227-016-1703-1)
15. Fuyumasa Takatsu, Kohei Hiraga, Osamu Tatebe, “Design of object storage using OpenNVM for high-performance distributed file system”, Journal of Information Processing, Vol. 24, No. 5, pp.824-833, 2016 (DOI: http://doi.org/10.2197/ipsjjip.24.824)
16. 渡邉英伸，黒澤隆，木村映善，水原隆道，村田健史，建部修見，広域分散ファイルシステムにおける UDT マルチストリームファイル転送ツール，電子情報通信学会論文誌 D, Vol.J99-D, No.5, pp.514-525, 2016
17. Xieming Li, Osamu Tatebe, “Improved Data-Aware Task Dispatching for Batch Queuing Systems”, Proceedings of the Seventh International Workshop on Data-Intensive Computing in the Clouds (DataCloud), pp.37-44, 2016
18. Masahiro Tanaka, Osamu Tatebe, “Fault Tolerance of Pwrake Workflow System Supported by Gfarm File System”, Proceedings of 9th Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS), pp.7-12, 2016
19. Fuyumasa Takatsu, Kohei Hiraga, Osamu Tatebe, “PPFS: a Scale-out Distributed File System for Post-Petascale Systems”, Proceedings of IEEE International Conference on Data Science Systems (DSS), pp.1477-1484, 2016
20. 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．採録決定．
21. Makoto Yabuta, Anh Viet Nguyen, Shinpei Kato, Masato Edahiro, Hideyuki Kawashima, Relational Joins on GPUs: A Closer Look. IEEE Transactions on Parallel and Distributed Systems, pp. 1-11, doi: doi:10.1109/TPDS.2017.2677451, 2017.
22. 神谷孝明，川島英之，星野喬，建部修見．並列ログ先行書き込み手法 P-WAL．情報処理学会論文誌データベース（TOD），Vol. 10, No. 1, pp. 24-39, 2017.
23. Li Jiang, Hideyuki Kawashima, Osamu Tatebe, Efficient Window Aggregate Method on Array Database System, Journal of Information Processing (JIP), Vol. 24, No.6, pp. 867-877, 2016.

24. 川島英之，データ基盤システムの動向，コンピュータソフトウェア，Vol. 33, No. 3 p. 3_44-3_49, 2016.
25. 齋藤 周作，多田野 寛人，今倉 暁，Shifted Block BiCGSTAB(l)法の構築とその高精度化，日本応用数理学会論文誌，Vol. 26, No. 3, pp. 318—352, 2016.
26. 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). (in print)
27. Ryohei Kobayashi, Tomohiro Misono, and Kenji Kise: A High-speed Verilog HDL Simulation Method using a Lightweight Translator, ACM SIGARCH Computer Architecture News - HEART '16 Volume 44 Issue 4, September 2016 Pages 26-31

B) 査読無し論文

1. 藤田典久, 大畠佑真, 小林諒平, 山口佳樹, 朴泰祐, "OpenCL と Verilog HDL の混合記述による FPGA プログラミング", 情報処理学会第 158 回 HPC 研究会報告 2017-HPC-158, 2016 年 3 月.
2. 田渕晶, 中尾昌広, 村井均, 朴泰祐, 佐藤三久, "アクセラレータクラスタ向け PGAS 言語 XcalableACC の片側通信機能の実装と評価", 情報処理学会第 158 回 HPC 研究会報告 2017-HPC-158, 2016 年 3 月.
3. 津金佳祐, 田渕晶大, 李珍泌, 村井均, 朴泰祐, 佐藤三久, "KNL メニーコア・プロセッサにおける PGAS 言語 XcalableMP アプリケーションの性能評価", 情報処理学会第 158 回 HPC 研究会報告 2017-HPC-158, 2016 年 3 月.
4. 廣川 祐太, 朴 泰祐, 佐藤 駿丞, 矢花 一浩, "電子動力学コード ARTED による Knights Landing プロセッサの性能評価", 情報処理学会第 157 回 HPC 研究会報告 2016-HPC-157, 2016 年 12 月.
5. 佐藤 賢太, 藤田 典久, 塙 敏博, 朴 泰祐, Ibrahim Khaled, "密結合並列演算加速機構 TCA における複数 DMAC の活用による GPU 対応 GASNet の性能改善", 情報処理学会第 156 回 HPC 研究会報告 2016-HPC-156, 2016 年 9 月.
6. 桑原 悠太, 塙 敏博, 朴 泰祐, "GPU クラスタにおける GPU セルフ MPI システム GMPI の予備性能評価", 情報処理学会第 155 回 HPC 研究会報告 2016-HPC-155 (SWoPP2016), 2016 年 8 月.

7. 田渕晶大, 木村耕行, 鳥居淳, 松古栄夫, 石川正, 朴泰祐, 佐藤三久, "EZY-SC 向け Omni OpenACC コンパイラの設計・試作", 情報処理学会第 154 回 HPC 研究会報告 2016-HPC-154, 2016 年 4 月.
8. 高橋大介, ”SIMD 命令を用いた整数除算の高速化”, 日本応用数理学会 2016 年度年会講演予稿集, 2016.
9. 高橋大介, “並列 FFT における通信隠蔽の自動チューニング”, 日本応用数理学会 2016 年度年会講演予稿集, 2016.
10. 五味歩武, 高橋大介, ”最適化手法を自動化する Xevolver フレームワーク用定義ファイルの実装”, 情報処理学会研究報告, Vol. 2016-HPC-155, No. 7, 2016.
11. 高橋大介, ”FFT における AT”, 2016 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 HPCS2016 論文集, pp. 47-48, 2016.
12. 高橋大介, ”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. 21, F-2-1, 2016.
13. 篠塚敬介, 高橋大介, ”中心-半径型区間演算に基づく矩形演算を用いた精度保証付き高速フーリエ変換”, 情報処理学会研究報告, Vol. 2016-HPC-154, No. 9, 2016.
14. 鷹津冬将, 平賀弘平, 建部修見, 高性能分散ファイルシステムのための分散メタデータサーバ PPMDS 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), Vol. 2016-HPC-155(1), 11 pages, 2016 年 8 月
15. 田中昌宏, 建部修見, ワークフローシステム Pwrake における耐障害機能, 研究報告ハイパフォーマンスコンピューティング(HPC), Vol. 2016-HPC-155(11), 7 pages, 2016 年 8 月
16. Mohamed Amin Jabri, Osamu Tatebe, “Performance assessment of highly concurrent sorted linked list with good spatial locality (Unrefereed Workshop Manuscript)”, Vol. 2016-HPC-155(35), 4 pages, Aug., 2016
17. 建部修見, Gfarm ファイルシステムの分散メタデータサーバ設計, 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2016-HPC-157(14), 7 pages, 2016 年 12 月
18. 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）, Vol. 2017-HPC-158(10), 9 pages, 2017 年 3 月
19. 建部修見, 佐々木慎, 髙橋一志, 大山恵弘, Gfarm ファイルシステムにおける RDMA アクセスの設計, 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2017-HPC-158(12), 6 pages, 2017 年 3 月

20. 岩井厚樹，建部修見，田中昌宏，並列ベンチマークのための同期複数タスク実行フレームワークの設計，研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2017-HPC-158(34), 8 pages, 2017 年 3 月
21. 中村泰大, 川島英之, 建部修見, 並行実行制御手法 TicToc と並列ログ先行書き込み手法 P-WAL の結合,情報処理学会第 139 回 OS 研究会,アクロス福岡, 2017 年 3 月 2 日.
22. 神谷孝明, 星野喬, 川島英之, 建部修見, トランザクション処理システムのリカバリ可能性の再考,情報処理学会第 139 回 OS 研究会,アクロス福岡, 2017 年 3 月 2 日.
23. 川島英之, 建部修見, 並列データベースシステムにおける演算子間データ配送方式の検討,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), Vol.2016-OS-138(2), No. 2016-08-01, pp. 1–5.
24. 村田直郁, 川島英之, 建部修見, RDMA を用いた RAMP トランザクションの高速化,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), Vol. 2016-OS-137(8), No. 2016-05-23, pp. 1–11.
25. 神谷孝明, 川島英之, 建部修見, 並列 WAL における共有カウンタの競合緩和化,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), Vol. 2016-OS-137(7),No. 2016-05-23, pp. 1–11.
26. 大黒晴之, 川島英之, 建部修見, In-memory MapReduce における最適な Shuffle 手法の検討,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), Vol.2016–OS–137(6), No. 2016–05–23, pp. 1–10.
27. 瀧沢亮太, 川島英之, 建部修見, データ移動方式による PostgreSQL への結合演算実装の試み,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), Vol.2016–OS–137(3), No. 2016–05–23, pp. 1–7.
28. 川島英之, 建部修見, 演算子間データ配送方式の検討,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), Vol. 2016-OS-137(1), No. 2016-05-23, pp.1-6.

(2) 国際会議発表

A) 招待講演

1. Taisuke Boku, "HPC Status in Japan", BDEC Workshop in ISC2016, Frankfurt Messe Jun 23rd, 2016.
2. Taisuke Boku, "Netxt Generation Interconnection for Accelerated Computing", ExaCOMM 2016 Workshop at ISC2016, Frankfurt Messe, Jun. 23rd, 2016.

3. Taisuke Boku, "FPGA Offloading on Computational Astrophysics and inter-FPGA communication", FPGA Workshop 2016, NCSA, Urbana Champaign, Oct. 13th, 2016.
4. Taisuke Boku, "Parallel Multi-Hetero System for Next Generation Computational Sciences", CODESIGN Workshop 2016 at HPC China 2016, Xian, Oct. 28th, 2016.
5. Taisuke Boku, "Codesigning for New Frontier of Computational Sciences", RECS2016 Workshop, Tokyo, Nov. 30th, 2016.
6. Taisuke Boku, "University of Tsukuba's Accelerated Computing", 3rd ADAC Symposium, Kashiwa, Jan. 25th, 2017.
7. Taisuke Boku, "Multi-Hybrid Platform for Next Generation Computational Science", AICS Symposium 2017, Kobe, Feb. 23rd, 2017.
8. Taisuke Boku, "Japan's Supercomputing Systems on Today and Tomorrow", HPC Saudi Arabia 2017, Jeddah, Mar. 14th, 2017.
9. Ryohei Kobayashi, "A survey of how to efficiently implement application-specific hardware on an FPGA", FPGA Workshop 2016, NCSA, Urbana Champaign, Oct. 12th, 2016.

B) 一般講演

1. Kenta Sato, Norihisa Fujita, Toshihiro Hanawa, Taisuke Boku, Khaled Z. Ibrahim, "GPU-ready GASNet Implementation on the TCA Proprietary Interconnect Architecture", Proc. of CSCI2016 (Int. Conf. on Computational Science and Computational Intelligence 2016), 6 pages, Las Vegas, Dec. 2016.
2. Akihiro Tabuchi, Yasuyuki Kimura, Sunao Torii, Video Matsufuru, Tadashi Ishikawa, Taisuke Boku, Mitsuhisa Sato, "Design and Preliminary Evaluation of Omni OpenACC Compiler for Massive MIMD Processor PEZY-SC", Proc. of IWOMP2016 (International Workshop on OpenMP (LNCS 9903: OpenMP: Memory, Devices, and Tasks), pp.293-305, Nara, Oct. 2016.
3. Kazuya Matsumoto, Norihisa Fujita, Toshihiro Hanawa, ,Taisuke Boku, “Implementation and Evaluation of NAS Parallel CG Benchmark on GPU Cluster with Proprietary Interconnect TCA”, Proc. of VECPAR2016, 8 pages, Porto, Jul. 2016.
4. Yuta Hirokawa, Taisuke Boku, Shunsuke Sato, Kazuhiro Yabana, "Electron Dynamics Simulation with Time-Dependent Density Functional Theory on Large

Scale Symmetric Mode Xeon Phi Cluster", Proc. of PDSEC2016 (in IPDPS2016), 8 pages, Chicago, 2016.

5. Daisuke Takahashi, “Automatic Tuning for Parallel FFTs on Cluster of Intel Xeon Phi processors”, 2017 Conference on Advanced Topics and Auto Tuning in High-Performance and Scientific Computing (2017 ATAT in HPSC), Taipei, Taiwan, March 10, 2017.
6. Daichi Mukunoki, Toshiyuki Imamura and Daisuke Takahashi, “Implementation Techniques for High Performance BLAS Kernels on Modern GPUs”, SIAM Conference on Computational Science and Engineering (CSE17), Atlanta, Georgia, USA, February 28, 2017.
7. Daisuke Takahashi, “Implementation of Parallel FFTs on Knights Landing Cluster”, SIAM Conference on Computational Science and Engineering (CSE17), Atlanta, Georgia, USA, February 28, 2017.
8. Daichi Mukunoki, Toshiyuki Imamura and Daisuke Takahashi, “Automatic Thread-Block Size Adjustment for Memory-Bound BLAS Kernels on GPUs”, 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-16), Special Session: Auto-Tuning for Multicore and GPU (ATMG), Lyon, France, September 22, 2016.
9. Daisuke Takahashi, “Automatic Tuning of Computation-Communication Overlap for Parallel 1-D FFT”, 2016 IEEE 19th International Conference on Computational Science and Engineering (CSE 2016), Paris, France, August 24, 2016.
10. Daisuke Takahashi, “Implementation of Multiple-Precision Floating-Point Arithmetic on Intel Xeon Phi Coprocessors”, 16th International Conference on Computational Science and Its Applications (ICCSA 2016), Beijing, China, July 4, 2016.
11. Hiroshi Maeda and Daisuke Takahashi, “Parallel Sparse Matrix-Vector Multiplication Using Accelerators”, 16th International Conference on Computational Science and Its Applications (ICCSA 2016), Beijing, China, July 4, 2016.
12. Daisuke Takahashi, “Automatic Tuning for Parallel FFTs on Intel Xeon Phi Clusters”, SIAM Conference on Parallel Processing for Scientific Computing (PP16), Paris, France, April 14, 2016.

13. Junji Kobayashi, Osamu Tatebe, “Simulation study of distributed metadata server”, 2nd Summer of CODES Workshop, Argonne, Jul. 12, 2016
14. Osamu Tatebe, “Data Integrity support for Silent Data Corruption in Gfarm File System”, Storage Developer Conference, Santa Clara, Sep. 19, 2016
15. Xieming Li, Osamu Tatebe, "Improved Data-Aware Task Dispatching for Batch Queuing Systems", the Seventh International Workshop on Data-Intensive Computing in the Clouds (DataCloud), Salt Lake City, Nov. 14, 2016
16. Masahiro Tanaka, Osamu Tatebe, "Fault Tolerance of Pwrake Workflow System Supported by Gfarm File System", Proceedings of 9th Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS), Salt Lake City, Nov. 14, 2016
17. Fuyumasa Takatsu, Kohei Hiraga, Osamu Tatebe, “PPFS: a Scale-out Distributed File System for Post-Petascale Systems”, Proceedings of IEEE International Conference on Data Science Systems (DSS), Sydney, Dec. 12, 2016
18. Naofumi Murata, Hideyuki Kawashima, Osamu Tatebe, Accelerating Read Atomic Multi-partition Transaction with Remote Direct Memory Access, IEEE International Conference on Big Data and Smart Computing, pp. 239-246, 2017. Best paper award on big data processing (runners-up).
19. Kentaro Horio, Hideyuki Kawashima, Osamu Tatebe, Efficient Parallel Summation on Encrypted Database System, IEEE International Conference on Big Data and Smart Computing, pp. 178-185, 2017. (acceptance ratio = 25%).
20. Ryuya Mitsuhashi, Hideyuki Kawashima, Takahiro Nishimichi, Osamu Tatebe, Three-Dimensional Spatial Join Count exploiting CPU Optimized STR R-Tree, Workshop of Big Data Challenges, Research, and Technologies in the Earth and Planetary Sciences held as part of the IEEE Big Data Conference, pp. 2938-2947, 2016.
21. Li Jiang, Hideyuki Kawashima, Osamu Tatebe, Fast Window Aggregate on Array Database by Recursive Incremental Computation, The IEEE 12th International Conference on eScience, pp. 101-110, 2016.
22. Harunobu Daikoku, Hideyuki Kawashima, Osamu Tatebe, On Exploring Efficient Shuffle Design for In-Memory MapReduce. BeyondMR workshop, Article 6, 2016..
23. Hiroto Tadano, Ryosaku Ikeda, Hiroyuki Kusaka, “Speeding up Large Eddy Simulation by Multigrid preconditioned Krylov subspace methods with mixed

precision", The 35th JSST Annual Conference International Conference on Simulation Technology (JSST2016), Kyoto, Japan, Oct. 27, 2016

24. Hiroto Tadano, "Numerical investigation of the cause of accuracy degradation of approximate solutions generated by Shifted Block Krylov subspace methods", The 1st Japan-Thailand Workshop on Numerical and Experimental Approaches to Nonlinear Problems, Bangkok, Thailand, Dec. 9, 2016.
25. Ryohei Kobayashi, Tomohiro Misono, and Kenji Kise: A High-speed Verilog HDL Simulation Method using a Lightweight Translator, International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART 2016), July 2016

(3) 国内学会・研究会発表

A) 招待講演

1. 朴泰祐, "Performance evaluation of electron dynamics simulation code ARTED on KNL cluster Oakforest-PACS", CDMSI シンポジウム, 柏, Dec. 6th, 2016.

B) その他の発表

1. 佐藤賢太, "密結合並列演算加速機構 TCA による GPU 対応 GASNet の実装と評価", 情報処理学会 HPCS2016, 仙台, 2016/6/7
2. 廣川祐太, "電子動力学シミュレーションのステンシル計算に対するメニーコアプロセッサ向け最適化",情報処理学会 HPCS2016, 仙台, 2016/6/6
3. 田渕晶大, "PEZY-SC 向け Omni OpenACC コンパイラの設計・試作",情報処理学会第 154 回 HPC 研究会, 横浜, 2016/4/25
4. 桑原悠太, "GPU クラスタにおける GPU セルフ MPI システム GMPI の予備性能評価",情報処理学会第 155 回 HPC 研究会, 松本, 2016/8/8
5. 佐藤賢太, "密結合並列演算加速機構 TCA における複数 DMAC の活用による GPU 対応 GASNet の性能改善", 情報処理学会第 156 回 HPC 研究会, 小樽, 2016/9/15
6. 廣川祐太, "電子動力学コード ARTED による Knights Landing プロセッサの性能評価", 情報処理学会第 157 回 HPC 研究会, 沖縄, 2016/12/21
7. 藤田典久, "OpenCL と Verilog HDL の混合記述による FPGA プログラミング",情報処理学会第 158 回 HPC 研究会, 熱海, 2017/3/9
8. 田渕晶, "アクセラレータクラスタ向け PGAS 言語 XcalableACC の片側通信機能の実装と評価", 情報処理学会第 158 回 HPC 研究会, 熱海, 2016 年 3 月.

9. 津金佳祐, “KNL メニーコア・プロセッサにおける PGAS 言語 XcalableMP アプリケーションの性能評価”, 情報処理学会第 158 回 HPC 研究会, 熱海, 2016 年 3 月.
10. 高橋大介, “SIMD 命令を用いた整数除算の高速化”, 日本応用数理学会 2016 年度年会, 北九州, 2016 年 9 月 12 日.
11. 高橋大介, “並列 FFT における通信隠蔽の自動チューニング”, 日本応用数理学会 2016 年度年会, 北九州, 2016 年 9 月 12 日.
12. 五味歩武, “最適化手法を自動化する Xevolver フレームワーク用定義ファイルの実装”, 情報処理学会第 155 回ハイパフォーマンスコンピューティング研究発表会, 松本, 2016 年 8 月 8 日.
13. 高橋大介, “FFT における AT”, 2016 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 HPCS2016, 仙台, 2016 年 6 月 6 日.
14. 高橋大介, “並列 FFT における通信隠蔽の自動チューニング”, 第 21 回計算工学講演会, 新潟, 2016 年 5 月 31 日.
15. 篠塚敬介, “中心-半径型区間演算に基づく矩形演算を用いた精度保証付き高速フーリエ変換”, 情報処理学会第 154 回ハイパフォーマンスコンピューティング研究発表会, 横浜, 2016 年 4 月 25 日.
16. 鷹津冬将, “高性能分散ファイルシステムのための分散メタデータサーバ PPMDS の評価”, 第 155 回ハイパフォーマンスコンピューティング研究発表会, 松本, 2016 年 8 月 8 日
17. 田中昌宏, “ワークフローシステム Pwrake における耐障害機能”, 第 155 回ハイパフォーマンスコンピューティング研究発表会, 松本, 2016 年 8 月 8 日
18. Mohamed Amin Jabri, “Performance assessment of highly concurrent sorted linked list with good spatial locality (Unrefereed Workshop Manuscript)”, 155th IPSJ SIGHPC Meeting, Matsumoto, Aug. 10, 2016
19. 建部修見, “Gfarm ファイルシステムの分散メタデータサーバ設計”, 第 157 回ハイパフォーマンスコンピューティング研究発表会, 那覇, 2016 年 12 月 22 日
20. 小林淳司, “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性能評価”, 第 158 回ハイパフォーマンスコンピューティング研究発表会, 熱海, 2017 年 3 月 8 日
21. 建部修見, “Gfarm ファイルシステムにおける RDMA アクセスの設計”, 第 158 回ハイパフォーマンスコンピューティング研究発表会, 熱海, 2017 年 3 月 8 日
22. 岩井厚樹, “並列ベンチマークのための同期複数タスク実行フレームワークの設計”, 第 158 回ハイパフォーマンスコンピューティング研究発表会, 熱海, 2017 年 3 月 10 日

23. 渡辺敬之, “並行実行木 Masstree の調査”,第 9 回データ工学と情報マネジメントに関するフォーラム, 2017 年 3 月 6 日.
24. 中村泰大, “並行実行制御手法 TicToc の調査”,第 9 回データ工学と情報マネジメントに関するフォーラム, 2017 年 3 月 6 日.
25. 梶原顕伍, “分散合意手法 Raft の調査”, 第 9 回データ工学と情報マネジメントに関するフォーラム, 2017 年 3 月 6 日.
26. 渡辺敬之, “並行実行木 Masstree の一括構築法”,第 9 回データ工学と情報マネジメントに関するフォーラム, 2017 年 3 月 6 日.
27. 中村泰大, “並行実行制御手法 TicToc と並列ログ先行書き込み手法 P-WAL の結合”, 情報処理学会第 139 回 OS 研究会,アクロス福岡, 2017 年 3 月 2 日.
28. 神谷孝明, “トランザクション処理システムのリカバリ可能性の再考”,情報処理学会第 139 回 OS 研究会,アクロス福岡, 2017 年 3 月 2 日.
29. 川島英之, “並列データベースシステムにおける演算子間データ配送方式の検討”,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), Vol.2016-OS-138(2), No. 2016-08-01, pp. 1–5.
30. 村田直郁, “RDMA を用いた RAMP トランザクションの高速化”,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), Vol. 2016-OS-137(8), No. 2016-05-23, pp. 1–11.
31. 神谷孝明, “並列 WAL における共有カウンタの競合緩和化”,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), Vol. 2016-OS-137(7),No. 2016-05-23, pp. 1–11.
32. 大黒晴之, “In-memory MapReduce における最適な Shuffle 手法の検討”,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), Vol.2016–OS–137(6), No. 2016–05–23, pp. 1–10.
33. 瀧沢亮太, “データ移動方式による PostgreSQL への結合演算実装の試み”,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), Vol.2016–OS–137(3), No. 2016–05–23, pp. 1–7.
34. 川島英之, “演算子間データ配送方式の検討”,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), Vol. 2016-OS-137(1), No. 2016-05-23, pp.1-6.
35. 多田野 寛人, “精度混合マルチグリッド前処理による Large Eddy Simulation の高速化”, 2016 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, 2016 年 8 月 1 日.

(4) 著書、解説記事等

1. 小林諒平: 緊急特集 本家 ARM の IoT ワールド入門 計算力時代到来...スパコン技術研究コーナ ソート専用コンピュータ最前線, CQ 出版社 Interface 2017 年 2 月号, pp.163-167, February 2017.
2. 小林諒平: IoT&スパコン！ラズパイ時代の自分用コンピュータ作り 第 6 章 ビッグデータ時代にますます重要！ハードウェア・データ処理に挑戦, CQ 出版社 Interface 2016 年 12 月号, pp.72-77, December 2016.
3. 小林諒平: IoT＆スパコン！ラズパイ時代の自分用コンピュータ作り 第 6 章 Appendix 2 基本演算の高速化が重要！ハードウェア並列ソート・アルゴリズム, CQ 出版社 Interface 2016 年 12 月号, pp.78-84, December 2016.

7. 異分野間連携・国際連携・国際活動等

1. 朴泰祐：日独仏・三ヶ国国際共同研究 SPPEXA（JST-CREST 内で実施）“MUST Correctness Checking for YML and XMP Programs”, 2015-2017,

8. シンポジウム、研究会、スクール等の開催実績

1. 第 1 回エクストリームストレージ研究会，東京，2016 年 5 月 23 日
2. 第 2 回エクストリームストレージ研究会，東京，2016 年 6 月 30 日
3. 第 3 回エクストリームストレージ研究会，東京，2016 年 8 月 3 日
4. 第 4 回エクストリームストレージ研究会，東京，2016 年 10 月 5 日
5. Gfarm ワークショップ 2016，神戸，2016 年 10 月 21 日
6. 第 5 回エクストリームストレージ研究会，東京，2016 年 11 月 4 日
7. Gfarm シンポジウム 2016，東京，2016 年 12 月 9 日
8. 第 6 回エクストリームストレージ研究会，東京，2017 年 1 月 12 日
9. 第 7 回エクストリームストレージ研究会，東京，2017 年 3 月 21 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：筑波大学システム情報系人事委員会委員

2. 朴泰祐：筑波大学情報環境企画室会議委員
3. 朴泰祐：HPCI 連携サービス委員会副委員長
4. 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
5. 朴泰祐：理化学研究所客員主管研究員
6. 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
7. 高橋大介：理化学研究所客員主管研究員
8. 高橋大介：HPCI 利用研究課題審査委員会レビューアー
9. 建部修見：HPCI 連携サービス運営・作業部会委員
10. 建部修見：理化学研究所客員主管研究員
11. 建部修見：情報通信研究機構協力研究員
12. 建部修見：HPCI 利用研究課題審査委員会レビューアー
13. 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
14. 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
15. 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
16. 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
17. 建部修見：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

10. 社会貢献・国際貢献

1. Taisuke Boku: Steering Committee Chair, HPC Asia Conferene Series
2. Taisuke Boku: Organizing Chair, HPC in Asia Session, ISC2016, Frankfurt
3. Taisuke Boku: Steering Committee, ISC2016
4. Taisuke Boku: Program Committee, Emergin Technology Session, SC2016
5. Taisuke Boku: Program Committee, ExaComm2016 International Workshop (ISC2016)
6. Taisuke Boku: Program Committee, Europar2016
7. Taisuke Boku: Program Committee, HiPC2016
8. Taisuke Boku: Program Committee, IWOMP2016
9. Daisuke Takahashi: Program Committee, 16th IEEE International Conference on Computer and Information Technology (CIT 2016)
10. Daisuke Takahashi: Program Committee, First International Workshop on GPU Computing and Applications (GCA'16) in Conjunction with 4th International Symposium on Computing and Networking (CANDAR'16)

11. Daisuke Takahashi: Program Committee, Special Session on Auto-Tuning for Multicore and GPU (ATMG) in Conjunction with IEEE 10th International Symposium on Embedded Multicore SoCs (MCSoC-16)
12. Daisuke Takahashi: Program Committee, 1st GPU Technology Workshop (GPUTech 2016) in Conjunction with the 16th International Conference on Computational Science and Its Applications (ICCSA 2016)
13. Daisuke Takahashi: Program Committee, 16th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2016)
14. Daisuke Takahashi: Program Committee, IEEE 10th International Symposium on Embedded Multicore SoCs (MCSoC-16)
15. Daisuke Takahashi: Program Committee, Eleventh International Workshop on Automatic Performance Tuning (iWAPT 2016)
16. Daisuke Takahashi: Program Committee, International Conference on Computational Science (ICCS 2016)
17. 高橋大介：情報処理学会論文誌コンピューティングシステム編集委員
18. 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
19. Osamu Tatebe: Program Committee, 25th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC) 2016
20. Osamu Tatebe: Program Committee, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) 2016
21. Osamu Tatebe: Program Committee, IEEE International Conference on Cluster Computing (Cluster) 2016
22. Osamu Tatebe: Program Committee, 8th IEEE International Conference on Cloud Computing Technology and Science (CloudCom) 2016
23. Osamu Tatebe: Program Committee, International Supercomputing Conference (ISC) 2016
24. Osamu Tatebe: Program Track Chair, Annual Meeting on Advanced Computing System and Infrastructure (ACSI) 2016
25. Osamu Tatebe: Program Committee, 6th International Workshop on Advances in High-Performance Computational Earth Sciences: Applications & Frameworks (IHPCES) 2016
26. Hideyuki Kawashima: Program Committee, IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-16)

27. Hideyuki Kawashima: Program Committee, 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI'16)
28. Hideyuki Kawashima: Program Committee, International Workshop on Smart Sensing Systems (IWSSS '16)
29. 川島英之：プログラム副委員長、xSIG (cross-disciplinary workshop on computing Systems, Infrastructures, and programminG)
30. 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
31. 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
32. 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
33. 川島英之：情報処理学会論文誌データベース編集委員
34. 川島英之：電子情報通信学会論文誌査読委員
35. 川島英之：電子情報通信学会英文論文誌小特集号編集委員
36. 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　主査
37. 多田野寛人：日本応用数理学会「若手の会」研究部会　運営委員
38. 多田野寛人：日本応用数理学会 JSIAM Letters 編集委員
39. 多田野寛人：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）プログラム委員

11. その他

海外長期滞在、フィールドワークなど