VII. 高性能計算システム研究部門

1. メンバー

教授　　朴　泰祐，高橋　大介，建部　修見

准教授　　塙　敏博（客員，東京大学）

助教　　多田野　寛人，小林　諒平，藤田　典久

学生　　大学院生　21 名，学類生　7 名

学内共同研究員

安永　守利，和田　耕一，桜井　鉄也，山口　佳樹，今倉　暁

（以上，システム情報系）

学外共同研究員

小柳　義夫（RIST），石川　裕（理化学研究所），

松岡　聡（理化学研究所），中島　浩（京都大学），

天野　英晴（慶應義塾大学），後藤　仁志（豊橋技術科学大学），

関口　智嗣（産業技術総合研究所），中尾　昌広（理化学研究所），

佐野　健太郎（理化学研究所），川島　英之（慶應義塾大学），

田中　昌宏（慶應義塾大学），平賀　弘平（慶應義塾大学）

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。

今年度の研究としては以下のテーマが挙げられる。

- 多重複合型演算加速スーパーコンピュータ Cygnus のシステム及びランタイムソフトウェア，及びアプリケーションソフトウェアの開発
- アプリケーション分野との共同研究による演算加速アプリケーション開発
- 「富岳」における数値計算ライブラリ開発及び性能評価
- 大規模並列分散ファイルシステムの性能及び拡張性の向上
- 複数右辺ベクトルを持つ行列計算の高速化と精度向上

3. 研究成果

【1】OpenACC による GPU+FPGA 混合プログラミングフレームワーク（朴）

OpenACC は近年注目されている GPU を中心とした演算加速装置のプログラミングを，汎用 CPU における OpenMP のように，逐次プログラムをベースに演算加速集中部分に directive（指示文）を挿入することでコンパイラが演算加速デバイス用のカーネルコードを生成するようにし，incremental にプログラムを高速化可能な言語フレームワークである。現状では，商用 OpenACC コンパイラは主として NVIDIA 社製 GPU 向けで，FPGA を対象とするものは存在しない。このため，GPU 用と FPGA 用，それぞれ異なったバックエンドコンパイラを用いてプログラミング環境を開発する。

これまで GPU 向けの OpenACC コンパイラとして NVIDIA 社傘下の PGI 社によるコンパイラが広く用いられており，我々はこれを GPU 用の OpenACC バックエンドコンパイラとして用いる。対して，FPGA 向けの OpenACC 環境としては米国 ORNL (Oak Ridge National Laboratory)の FTG (Future Technology Group)が開発を進めている OpenARC コンパイラに着目する。OpenARC は様々な演算加速デバイス向けの OpenACC コンパイラであり，Intel 社製 FPGA 向けのコンパイル機能がテスト版として実装されている。OpenARC が行う FPGA 向け OpenACC コンパイル機能は, OpenACC 記述を最終的に OpenCL に変換する。この OpenCL コードは Intel 社の FPGA 向け SDK に付属する OpenCL コンパイラで最終的に FPGA の回路合成が行われる。従って，1 つのユーザプログラムから GPU で演算加速する部分と FPGA でこれを行う部分を分離し，それぞれを GPU 向け及び FPGA 向けのコンパイラで処理し，最後に部分オブジェクトを合成すれば OpenACC 統一プログラミング環境が完成する。

OpenARC コンパイラについては, FPGA 向け最適化は ORNL 内の開発プロジェクトで行われているため，我々は当該チームとの国際共同研究を進め，この開発版コンパイラを使用することができるようにした。ORNL にとっても，OpenARC コンパイラの先進的利用ケースを得ることができるため，この共同研究は順調に進められている。ただし，1 つの OpenACC コードを両デバイス向けに分離することは容易ではなく，これをユーザが手で行うことは大きな負担となる。そこで, 今年度研究ではこれを自動的に行う言語処理系 MHOAT (Multi-Hybrid OpenACC Translator)を開発した。図 1 は MHOAT の処理の概念図である。

MHOAT によって FOGA と GPU という 2 種類のデバイス用に分離された OpenACC コードは，それぞれのバックエンドコンパイラ（GPU: PGI Compiler, FPGA: OpenARC）によってオブジェクトが生成され, 最終的に gcc によるリンクを経て, 1 つのホストプログラムから FPGA と GPU の各デバイス用カーネルプログラムを起動可能なオブジェクトが完成する。OpenARC が処理した FPGA 向け OpenCL コードを Intel 環境でコンパイルし，NVIDIA 社 GPU のオブジェクトと融合する技術については，昨年度の研究で開発した，OpenCL+CUDA コンパイラを実現する環境を利用している。

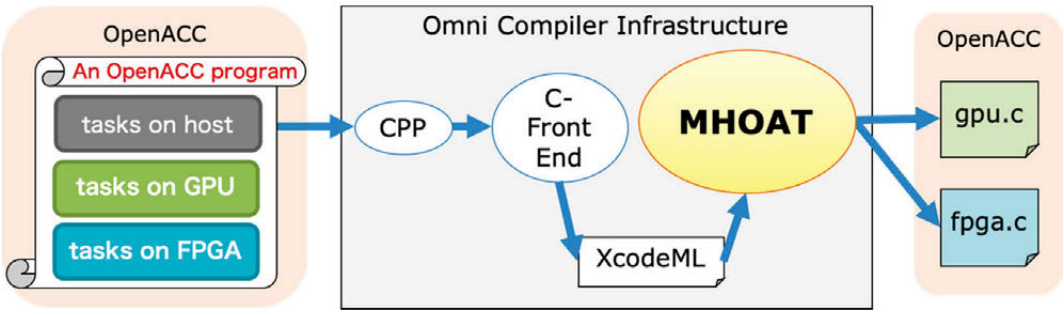


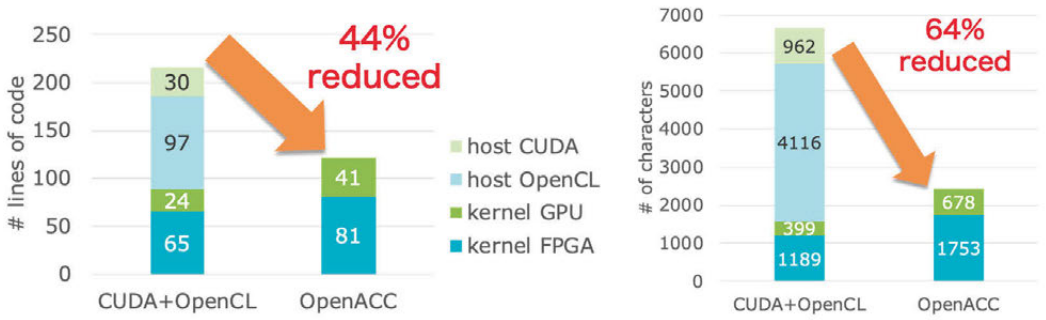
図 1 MHOAT の処理フロー

機能及び基本性能評価として，以下のような複合演算加速プログラムを OpenACC で記述した。

- GPU 上で行列・行列乗算を行う。
- それで生成された行列とベクトルの乗算を行う。
- それで生成されたベクトルを FPGA に転送し，これを固定ベクトルとする CG 法（共役勾配法）による反復行列解法を FPGA 上で行う。
- 最終結果を CPU に戻す。

なお，今年度の MHOAT 実装では，FPGA と GPU の間で DMA 転送を行う機能の組み込みまでは行えず，GPU と FPGA のデータ転送は CPU を介して間接的に行っている。

図 2 に従来の CUDA+OpenCL で両デバイスのプログラムを書いた場合と，MHOAT を用いて OpenACC のみで記述した場合の行数（a)と文字数(b)の比較を示す。



(a) 行数　　(b) 文字数

図 2 CUDA+OpenCL による従来の記述と MHOAT を用いた OpenACC のみによる記述の行数(a)及び文字数(b)の差

図から明らかなように，MHOAT を用いた OpenACC のみによる統一的記述では，特にホストコード（CPU 上で動作するプログラム）の記述量が非常に少なくなり，ユーザのプログラミング労力を減らすことができている。また，複数言語を用いるプログラミングは，その習

得にも時間を要し，その複雑さゆえバグの混入の可能性も高い。この評価では MHOAT が多重複合演算加速環境における FPGA+GPU プログラミングを従来よりはるかに容易にすることが可能ということが示せた。

続いて性能評価である。上記実験で用いたコードから生成されたプログラムを，計算科学研究センターが運用する多重複合演算加速クラスタ Cygnus 上で評価した。Cygnus の GPU+FPGA 混載ノードには Intel Xeon Gold CPUx2 基，Intel Stratix10 FPGAx2 基，NVIDIA Tesla V100x4 基が搭載されているが，本実験では CPU，FPGA，GPU をそれぞれ 1 基ずつ用いて評価した。結果を図 3 に示す。

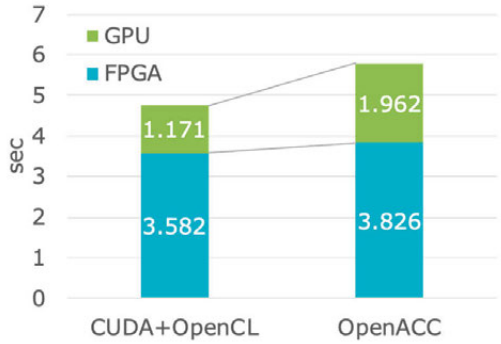


図 3 評価プログラムの実行時間：CUDA+OpenCL と MHOAT による OpenACC のみの比較

ここに示す通り，総演算時間では MHOAT 実行のほうが性能が低下した。GPU 部では約 1.7 倍，FPGA 部では約 1.07 倍，それぞれ実行時間が長くなってしまった。特に GPU については，OpenACC コンパイラの成熟度から考え，OpenACC コードの指示文記述に何らかの最適化が必要と考えられる。これは今後の課題であるが，先述の記述の容易さを総合的に考えれば，MHOAT の OpenACC のみによる環境の方がプログラムの生産性の点で優れていると考えられる。

これらの研究成果については情報処理学会第 169 回 HPC 研究会，同第 172 回 HPC 研究会にて発表済みである。

【2】気象アプリケーション City-LES の GPU 向け高速化（朴）

GPU と FPGA の融合による演算加速のアプリケーションの一つとして，本センター地球環境研究部門で開発中の都市気象モデルシミュレーション City-LES の GPU・FPGA 融合化を進めている。今年度は特に，これまで CUDA で開発してきたコードを全面的に OpenACC 化し，今後の OpenACC による統一的プログラミングへの準備を進めた。昨年度まで，CUDA による City-LES 実装を行ったが，CPU 上での実行がわずかに残り，全体の実行時間の約 85%が GPU 側に移植されたにも関わらず，CPU 部分が残っているため CPU と GPU の間のデータ転

送コストが全く無視できず，結果として全実行時間の約 50%がデータ転送時間であるということになっていた。

今年度は実計算ループにおける CPU 実行部分の全てを GPU に行わせることにしたが，それらの関数を個別に CUDA 化するためのコストは決して小さくない。また，全てのデータを GPU 側で管理するため，メモリ制御を容易に記述できる必要がある。そこで，実装を全面的に見直し，全てを OpenACC に書き直すことにした。OpenACC による記述では，CPU 上のホストコードと GPU 上のカーネルコードを個別に指定するのではなく，OpenACC の指示文で演算加速カーネル化する場所を指定するため，データの所在を容易に記述できるだけでなく，全データを GPU 上に確保し，例え並列性が低くても敢えてその部分を GPU で実行するようにプログラムすることが容易にできた。この，例え部分的な GPU 実行が CPU 実行より遅くても，データ転送時間を消去することができれば全体的に高い性能が得られるという，発想の転換が大きな性能向上につながった。

本センターにおける多重複合型演算加速スーパーコンピュータ Cygnus の最大 32 ノード（128 GPU）を用いて評価を行った。まず，単体ノード上の 2 台の GPU (NVIDIA Tesla V100) と 2 台の CPU (Intel Xeon Gold)を用い，CPU のみの実行，データ移動を伴う昨年度までの GPU 実装，OpenACC によってデータ移動をなくした GPU での実装を比較した。結果を図 4 に示す。ここに示された通り，データ移動を含んだ GPU コードでは CPU の 2 倍以上の性能が得られているものの，その実行時間の約 50%は GPU と CPU の間のデータ移動時間である。これが OpenACC 化によって全データを GPU に閉じ込めた結果，データ移動のある場合の約 2.8 倍の性能向上が得られ，最終的に CPU のみによる実行に比べ約 6.2 倍の性能向上が得られた。

さらに，最大 32 ノード，128 GPU を用いた性能評価結果を図 5 に示す。ここでは 1 ノード実行の場合を 1 とし，weak scaling，つまりノード当たりの問題サイズを一定にしたままノード数を増やして性能向上を評価した。GPU の結果は全て OpenACC のみによるものである。結果として，32 ノードの場合でも 1 ノード実行に比べ並列処理効率が 86%と高く保たれており，いずれのノード数でも CPU のみに比べ約 9 倍の性能が得られた。

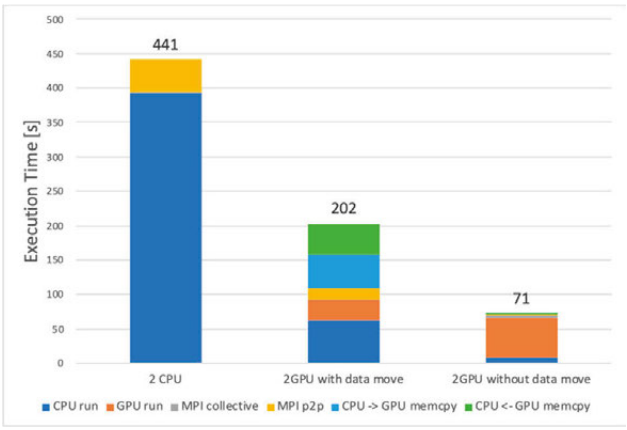


図 4　Cygnus の単体ノードにおける GPU・CPU 間データ通信削減の効果

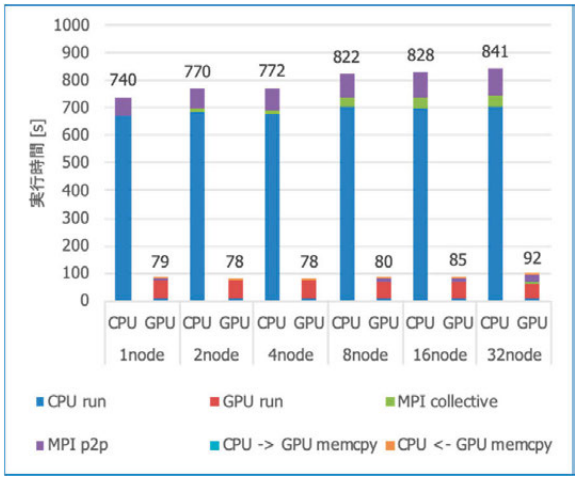


図 5　Cygnus 32 ノードを用いた GPU 実装の weak scaling 性能向上

また，今回のフル OpenACC 実装によって，MHOAT 等の基盤を用いた GPU・FPGA 統一プログラミングを適用することが可能となった。今年度の研究では FPGA による OpenACC 化まで踏み込めなかったが，City-LES では単純なステンシル計算だけでなく，地表面の性質やビル形状，さらに日光の照射をレイトレーシング処理するなど，非常に詳細なシミュレーションとなっており，GPU の持つ SIMD 演算性能だけでは効率が非常に低下する可能性が高い。ここに FPGA を投入していくのが次年度以降の課題である。

これらの研究成果については情報処理学会第 170 回 HPC 研究会，同第 173 回 HPC 研究会にて発表済みである。

【3】Arm SVE 命令を用いた FFT の実装および理研シミュレータによる性能評価（高橋）

「富岳」のプロセッサである A64FX に向けた FFT の実装を Arm SVE（Scalable Vector Extension）命令を用いて行い，理研シミュレータにより性能評価を行った。

Arm SVE 実装の方針としては，Fortran で書かれた FFTE ライブラリのカーネルを，SPIRAL コードジェネレータで生成した SVE intrinsic 版の FFT カーネルで置き換えることにより高速化を図る。高い基数（例えば 16）の FFT カーネルを手でコーディングするのは多大な労力を必要とするが，SPIRAL では基数を入力するだけで FFT カーネルを自動で生成できる。Arm SVE 命令を用いて FFT カーネルを生成するにあたり，Vector Length Agnostic（VLA）プログラミングモデルを用いた。

SPIRAL は離散フーリエ変換（DFT）などのデジタル信号処理に対して C 言語や Fortran の最適化したコードを自動生成するシステムである。Carnegie Mellon University の Franz Franchetti のグループで開発されている。ソースコードは http://www.spiral.net/から入手することが可能である。SPIRAL において変換の種類とサイズを指定すると，サイズに応じたア

ルゴリズムが生成される。このアルゴリズムは，より小さい演算の木（rule tree）に分解され，ここでいくつかの生成可能な木からより性能の良いものを選択する。

FFT カーネルとして，今回は out-of-place の FFT アルゴリズムである Stockham FFT アルゴリズムを用いた。高い基数の FFT カーネルを積極的に用いることにより，メモリアクセス回数を減らすことができる。その一方で，高い基数の FFT カーネルは多くのテンポラリ変数を必要とするため，浮動小数点レジスタが不足する。FFTE を富士通 Fortran コンパイラでコンパイルした際に基数 2，3 および 4 の FFT カーネルではソフトウェアパイプライニングが適用されたが，基数 5 以上の FFT カーネルでは浮動小数点レジスタ数の不足によりソフトウェアパイプライニングが適用されなかった。FFTE では基数 2，3，4，5 および 8 の FFT カーネルを，SPIRAL では基数 2，3，4，5，6，8，10，12 および 16 の FFT カーネルを用いた。SPIRAL が生成した基数 2 の FFT カーネルを図 6 に示す。

```
void dft2_sve_jk(float64_t *Y, float64_t *X,
                 float64_t *TW1, int l1, int m1) {
  svfloat64x2_t s49, s52, svex2_3, svex2_4;
  int a104, a105;
  float64_t a106, a107;
  svbool_t pg1;
  svfloat64_t s50, s51, s53, s54, s55, s56;
  for (int j1 = 0; j1 < l1; j1++) {
    int k1 = 0;
    pg1 = svwhilelt_b64(k1, m1);
    do {
      a104 = k1 + j1 * m1;
      s49 = svld2_f64(pg1, X + 2 * a104);
      s50 = s49.v0;
      s51 = s49.v1;
      s52 = svld2_f64(pg1,
            X + 2 * (a104 + l1 * m1));
      s53 = s52.v0;
      s54 = s52.v1;
      s55 = svsub_f64_x(pg1, s50, s53);
      s56 = svsub_f64_x(pg1, s51, s54);
      a105 = ((2)*( j1));
      a106 = TW1[a105];
      a107 = TW1[a105 + 1];
      svex2_3.v0 = svadd_f64_x(pg1, s50, s53);
      svex2_3.v1 = svadd_f64_x(pg1, s51, s54);
      svst2_f64(pg1, Y + 2 * (k1 + 2 * j * m1),
                svex2_3);
      svex2_4.v0 = svnmls_n_f64_x(pg1,
        svmul_n_f64_x(pg1, s56, a107),s55 ,a106);
      svex2_4.v1 = svmla_n_f64_x(pg1,
        svmul_n_f64_x(pg1, s56, a106), s55, a107);
      svst2_f64(pg1, Y + 2 * (k1 + (2 * j1 * m1) + m1),
                svex2_4);
      k1 += svcntd();
      pg1 = svwhilelt_b64(k1, m1);
    } while(svptest_any(svptrue_b64(), pg1));
  }
```

図 6　SPIRAL が生成した基数 2 の FFT カーネル

性能評価にあたっては，FFTE 6.0 (http://www.ffte.jp/) の一次元複素数 FFT ルーチン zfft1d を用いて，理研シミュレータにより A64FX の性能を測定した。$n = 2\sim65536$点の順方向 FFT を 10 回実行し，その平均の経過時間を測定した。$n = 65536$の場合のワーキングセットサイズは 3MiB であり，2 次キャッシュに収まる。n点 FFT の演算回数は$5n\log_2 n$として MFlops 値を算出した。A64FX のそれぞれ 1 スレッドを用いた。コンパイラは Fujitsu C/C++/Fortran compiler 4.0.0 20190701 および Arm C/C++/Fortran compiler 19.1 を用い，コンパイラオプションとして，”-Kfast,restp=all”（Fujitsu），”-Ofast -march=armv8-a+sve”（Arm）を用いた。Arm Fortran コンパイラで生成したバイナリが理研シミュレータで実行できなかったため，FFT カーネル

のみを Arm コンパイラでコンパイルして，それ以外のルーチンを富士通コンパイラでコンパイルしてバイナリを生成した。

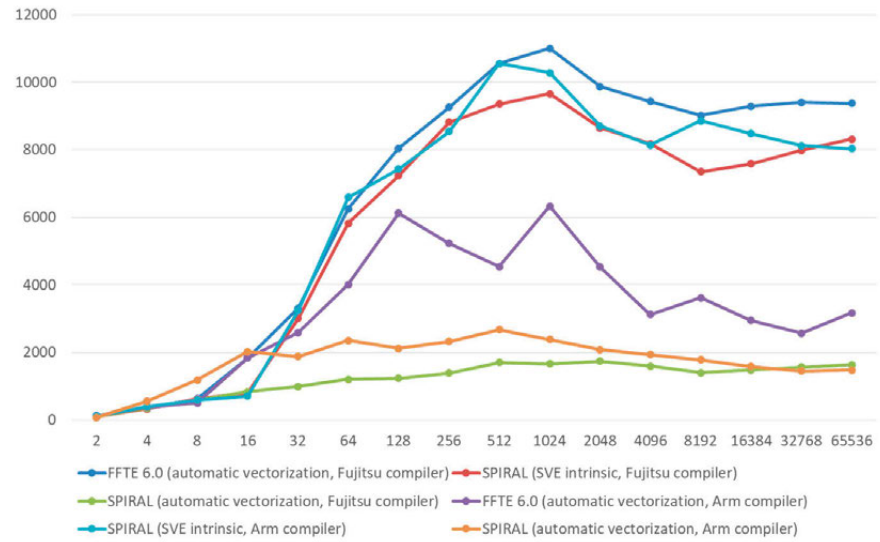


図 7　power-of-two FFT の性能（理研シミュレータ）

理研シミュレータを用いて power-of-two FFT の性能を測定した結果を図 7 に示す。グラフの横軸はデータサイズであり，縦軸は演算性能（MFlops）である。富士通 C コンパイラを用いた場合でも，Arm C コンパイラを用いた場合でも，SPIRAL（SVE intrinsic）は SPIRAL（automatic vectorization）よりも高速である。富士通 C コンパイラを用いた場合，$n = 256$において SPIRAL（SVE intrinsic）は SPIRAL（automatic vectorization）よりも約 6.33 倍高速である。さらに Arm C コンパイラを用いた場合，$n = 32768$において SPIRAL（SVE intrinsic）は SPIRAL（automatic vectorization）よりも約 5.62 倍高速である。これらの結果から，SPIRAL により intrinsic を用いた明示的なベクトル化は効果的であることが分かる。

富士通 C/Fortran コンパイラを用いた場合，FFTE（automatic vectorization）は SPIRAL（SVE intrinsic）よりも高速である。一方で Arm コンパイラを用いた場合，SPIRAL（SVE intrinsic）は$n = 16$を除いて FFTE（automatic vectorization）よりも高速である。特に$n = 32768$において SPIRAL（SVE intrinsic）は FFTE（automatic vectorization）よりも約 3.16 倍高速である。富士通 Fortran コンパイラでコンパイルした FFTE は Arm Fortran コンパイラでコンパイルした場合よりも高速である。一方で Arm C コンパイラでコンパイルした SPIRAL（SVE intrinsic）は $n =$ 8，16，256，4096，65536 を除いて富士通 C コンパイラでコンパイルした場合よりも高速である。したがって，SPIRAL（SVE intrinsic）に対しては富士通 C コンパイラよりも Arm C コンパイラの最適化が有効に働いたと考えられる。

SPIRAL コードジェネレータで生成した FFT カーネルの性能は，Fortran で記述した FFTE の FFT カーネルとほぼ同じ性能になった。富士通コンパイラと Arm コンパイラでは挙動がかなり異なることも分かった。具体的には，Fortran では富士通コンパイラが，C では Arm コンパイラが高速である。今後は複数スレッドによる実行を行う予定である。

【4】AVX-512IFMA を用いた多倍長整数乗算の高速化（高橋）

乗算は最も基本的な四則演算の一つである。また 64bit を超えた精度の数を多倍長数といい，多倍長整数乗算はソフトウェアで解決する問題である。この演算は公開鍵暗号システムや数式処理システムに応用される。乗算においてキャリーの発生は課題の一つであり，これに対応するためオペランドの 1 ワードの bit 数を減らす表現（reduced-radix 表現）手法を用いた。また近年では Intel から Cannon Lake プロセッサが出荷されたことにより AVX-512 Integer Fused Multiply-Add（IFMA）命令が利用可能になった。さらに高速な乗算アルゴリズムの一つに Karatsuba 法がある。

関連研究として，reduced-radix 表現と AVX-512 命令を用いた多倍長整数乗算の高速化を行った研究がある。[Keliris and M. Maniatakos 2014]では AVX-512 Foundation（F）を使用し Intel Software Development Emulator（SDE）で命令数を評価した結果，GNU Multiple Precision Arithmetic Library（GMP）に対し 4096bit どうしの乗算で約 1.45 倍性能向上した。[Gueron and Krasnov 2016]では AVX-512 IFMA を用いて実装し命令数のみの性能評価を行った結果，4096bit どうしの乗算で AVX-512IFMA により，GMP に対し約 8 倍の性能向上が得られている。また[Edamatsu and Takahashi 2018]では AVX-512F を使用して Xeon Phi プロセッサにおける実行時間の評価を行った結果，4096bit どうしの乗算で GMP に対し約 2.3 倍の性能向上が得られている。

Algorithm 1 BasecaseMultiply

Input: $A = \sum_{i=0}^{m-1} a_i \beta^i, B = \sum_{j=0}^{n-1} b_j \beta^j$

Output: $C = AB := \sum_{k=0}^{m+n-1} c_k \beta^k$

1: $C \leftarrow A \cdot b_0$
2: **for** $j \leftarrow 1$ **to** $n-1$ **do**
3: $\quad C \leftarrow C + \beta^j (A \cdot b_j)$
4: **end for**
5: return C.

本研究では，多倍長整数を符号なし 64bit 整数型配列で表現する。64bit どうしの加算は最大で 65bit になり，キャリーが発生する可能性があるため余分な処理が追加される。そこで，乗算前に 1 ワードあたりの bit 数を 52bit に減らした表現（reduced-radix 表現）に変換する。52bit である理由は，AVX-512IFMA は浮動小数点数演算の仮数部を利用した整数の積和演算を行うため，オペランドは 64bit のうち 52bit のみが使用されるためである。これにより 52bit

どうしの加算となり，53bit 目以降をキャリー蓄積の空間として利用する。蓄積したキャリーは乗算の後でまとめて処理する。

Algorithm 2 KaratsubaMultiply

Input: $A = \sum_{i=0}^{n-1} a_i \beta^i, B = \sum_{j=0}^{n-1} b_j \beta^j$
Output: $C = AB := \sum_{k=0}^{2n-1} c_k \beta^k$

1: **if** $n \leq n_0$ **then**
2: return **BasecaseMultiply**(A, B)
3: **end if**
4: $k \leftarrow \lceil n/2 \rceil$
5: $(A_0, B_0) := (A, B) \bmod \beta^k$
6: $(A_1, B_1) := \lfloor (A, B) \ / \ \beta^k \rfloor$
7: $(A_2, B_2) := (A_0 - A_1, B_0 - B_1)$
8: $C_0 \leftarrow$ **KaratsubaMultiply**(A_0, B_0)
9: $C_1 \leftarrow$ **KaratsubaMultiply**(A_1, B_1)
10: $C_2 \leftarrow$ **KaratsubaMultiply**(A_2, B_2)
11: return $C := C_0 + (C_0 + C_1 - C_2)\beta^k + C_1\beta^{2k}$.

本研究では Karatsuba 法（Algorithm 2）による実装を行った。Algorithm 1 は Karatsuba 法でも用いられる筆算法である。ワード数nの乗算の計算量について，筆算法と Karatsuba 法はそれぞれ$O(n^2)$と$O(n^{\log_2 3}) \approx O(n^{1.585})$である。本研究では Algorithm 2 の 7，11 行目には AVX-512F，2 行目の筆算法には AVX-512IFMA 命令を用いる。

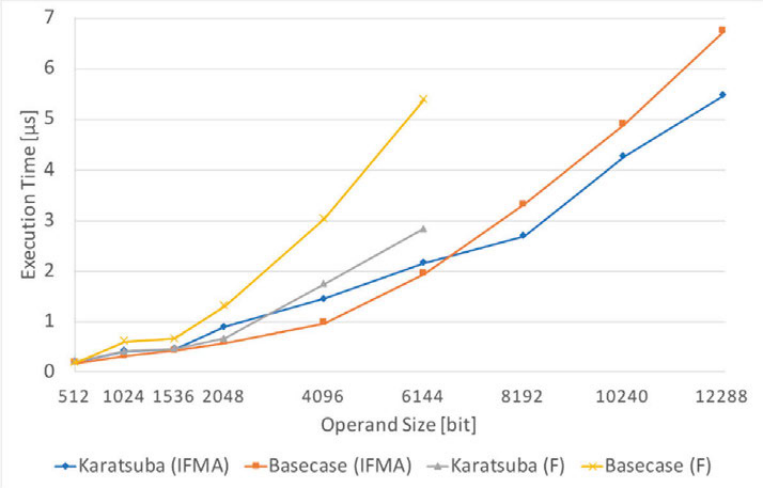


図 8　筆算法と Karatsuba 法との実行時間の比較

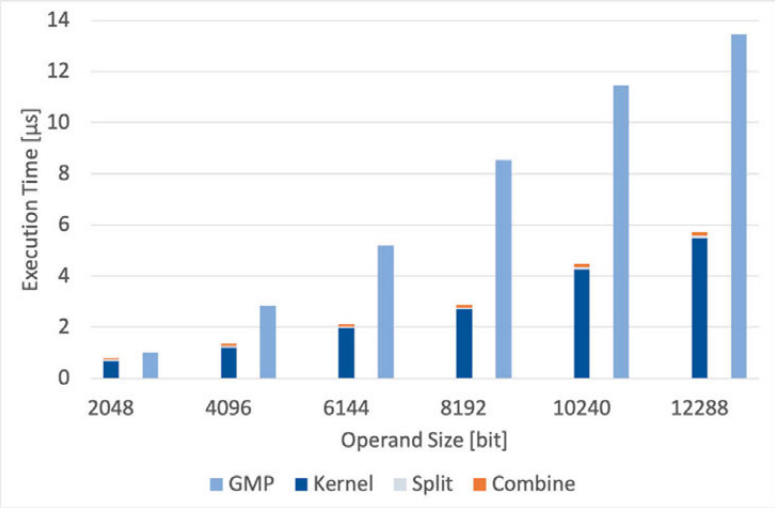


図 9　GMP と本研究の実装の実行時間

図 8 は筆算法と Karatsuba 法に AVX-512F と AVX-512IFMA を適用した 4 種類の多倍長整数乗算の実行時間を示している。オペランドが 512bit の場合はいずれもほぼ同等の実行時間であるが, 6144bit までは AVX-512IFM による筆算法が, それ以降のサイズでは AVX-512IFMA による Karatsuba 法が最も速い結果が得られたため，AVX-512IFMA 命令は AVX-512F 命令よりも高速に乗算を処理することが示された。また図 9 は本研究の実装と GMP との実行時間の比較を示している。図 の結果から，7168bit を境に筆算法から Karatsuba 法へと切り替える実装を行った。グラフからすべてのサイズで GMP よりも高速に乗算を処理することが示された。本研究は GMP に対し最大で約 2.97 倍の性能向上が得られている。

【5】ノードローカルバーストバッファの研究（建部）

スーパコンピュータにおける，演算性能とストレージ性能のギャップを解消するため，計算ノードに不揮発性メモリ，NVMe SSD などの高速なストレージをもつ階層的なストレージの研究が急務となっている。ノードローカルストレージは，計算ノードのメモリと同様に，計算ノードが割り当てられている間しか利用することができない。そのため，本研究では，ジョブの起動時に，即座に計算ノードのローカルストレージを用いた並列ファイルシステムを構築して，バーストバッファとして利用することを検討している。バーストバッファシステムは，当初，バースト的なデータ書込みに対応するための一時的なバッファとして研究開発がすすめられたが, 近年はキャッシュ的利用, 一時データ領域としての利用のほか, In transit データ解析のための利用など様々な用途に広がってきている。そのため，一般的なファイルシステムのインターフェースを持つことは，より広範囲の利用が可能となり大きなメリットとなる。ノードローカルバーストバッファの研究開発にあたり，これまで研究開発を進めてきた Gfarm ファイルシステムをベースとして行った。Gfarm ファイルシステムは，もともと

計算ノードのローカルストレージを用いた並列ファイルシステムであり，ノードローカルストレージの局所性を利用する特徴を持っている。この Gfarm ファイルシステムを，高速な不揮発性メモリ，NVMe SSD に対して用いるためには，遠隔ファイルアクセスの高速化が求められる。その問題を解決するために，これまでの通信プロトコルに対し上位互換となるように RDMA を用いた通信プロトコルの設計を行った。また，メタデータ性能を高速化するため，メタデータの冗長性と性能のトレードオフについて評価を行った。

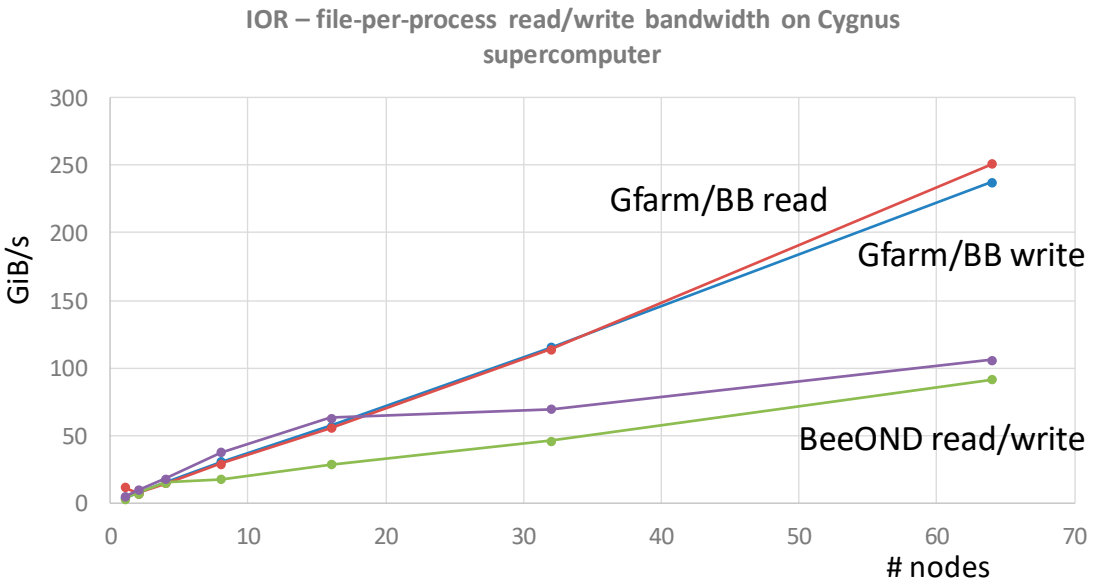


図 10 並列ファイルアクセス性能

図 10 に Cygnus スーパコンピュータにおける並列ファイルアクセス性能を示す。Gfarm/BB は本研究で実装したバーストバッファシステムである。ノード数を増加させると性能が向上していることが分かる。BeeOND は，同様のシステムであるが，ノード当たりの性能が低いことが分かる。Gfarm/BB はノードローカルストレージの局所性を考慮して書込みを行っているのに対し，BeeOND は複数のストレージにストライピングして書き込んでいるため，この性能差となっている。本成果は，論文誌 Journal of Computer Science and Technology において公表した。

【6】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）

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。これまで，大規模な観測データを用いた深層学習を進めるための準備として，2006 年から 2018 年までの全国合成レーダ GPV と，JAMSTEC から提供された可降

水量のデータを用い，機械学習のためのデータの準備を行った。これにより，可降水量の急な変化による降雨などの学習を目指している。大規模データにより学習を行う場合，大規模入力データの読込性能がボトルネックとなる。並列ファイルシステムの数多くの小規模データに対し，多数の計算ノードが一斉にランダムにアクセスするためである。従来は，その性能低下を避けるため，あらかじめノードローカルストレージにデータをステージングしておき，そのデータを用いて学習を行う方法がとられている。しかしながら，大規模入力データの場合はこのステージング時間が問題となる。例えば，学習時間が 75 秒で完了するものの，ステージング時間は 10 分ほどかかるという報告もある。本研究では，このステージング時間を短縮，隠蔽するため，並列ファイルシステムのデータをノードローカル SSD にプリフェッチする方法を提案した。これまでは，読込データはランダムに読込まれるためプリフェッチは難しいと考えられていたが，分散機械学習フレームワークにプリフェッチ機構を組込むことによりこの問題の解決を図った。筑波大学の Cygnus スーパコンピュータにおいて評価を行い，プリフェッチを効果的に行うことにより，ノードローカル SSD にあらかじめステージングしておいたときとほぼ変わらない性能で大規模データの深層学習を行うことが可能であることを示した。

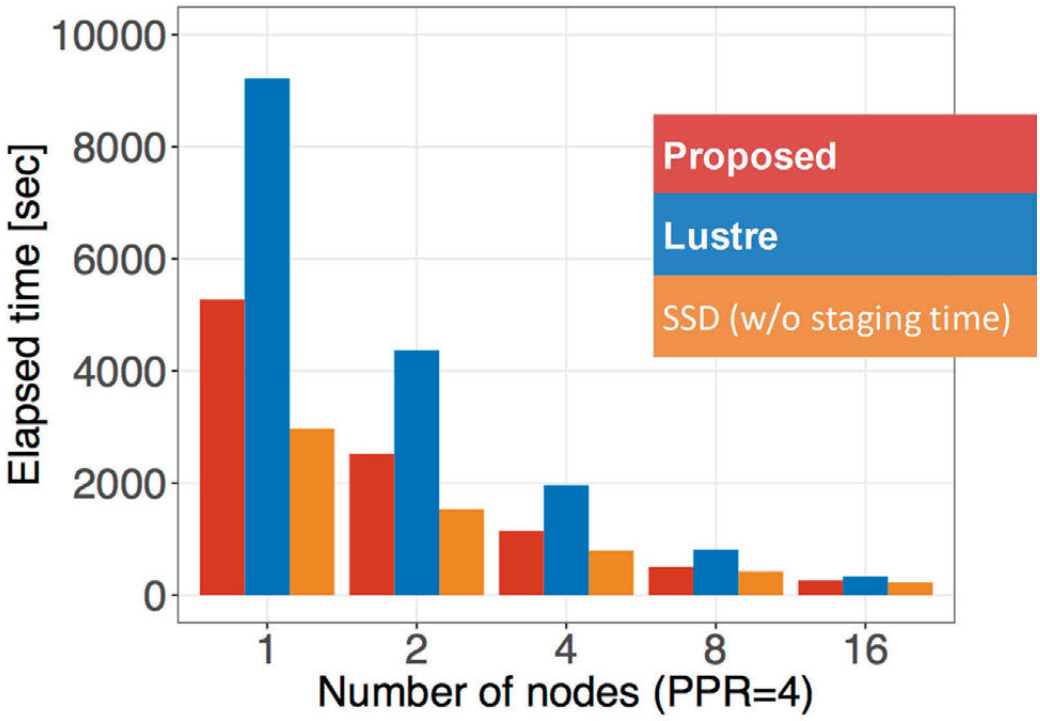


図 11 Cygnus スーパコンピュータによるプリフェッチの性能

図 11 に Cygnus スーパコンピュータにおける評価結果を示す。SSD は，あらかじめステージングを行ったときの処理時間で，この処理時間はプリフェッチ処理が完全に隠蔽されたと

きの最高値を示している。本プリフェッチを用いることにより，並列ファイルファイルシステムから読込むときに比べ，1 ノードで 1.74 倍，16 ノードで 1.26 倍高速に処理することが可能となった。これにより，大規模データの深層学習において読込データのボトルネックを解消することができると考えられる。本成果は国際会議 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT 2019)において発表した。

【7】大規模メタゲノムデータ解析の研究（建部）

メタゲノム解析は環境サンプルなどから抽出したゲノムを網羅的に解析するものであり，特定のゲノムの培養が難しいケースに対しても解析が可能となる。シーケンサの処理能力の向上により，クエリデータ，参照データベースの双方のデータ量が増えている。これにより，メモリ量，演算速度の問題により，1 ノードでの処理ができなくなってきた。この問題を解決するために，これまでクエリデータを分割して分散解析する方法が提案されている。しかしながら，参照データベースのデータ量の増加により，参照データベースについても分割することが必要となっている。本研究では，クエリデータと参照データベースの両方を分割し，分散データ解析する手法を提案する。両データを分割することにより，分散データ解析した結果の集計処理が複雑となるが，本研究では Pwrake ワークフローシステムを用いることにより解決した。また，Gfarm/BB により，ノードローカルストレージを効率的に用いることで大規模処理時においても，I/O 性能がオーバヘッドとならないこと，演算性能がスケールすることを確かめた。

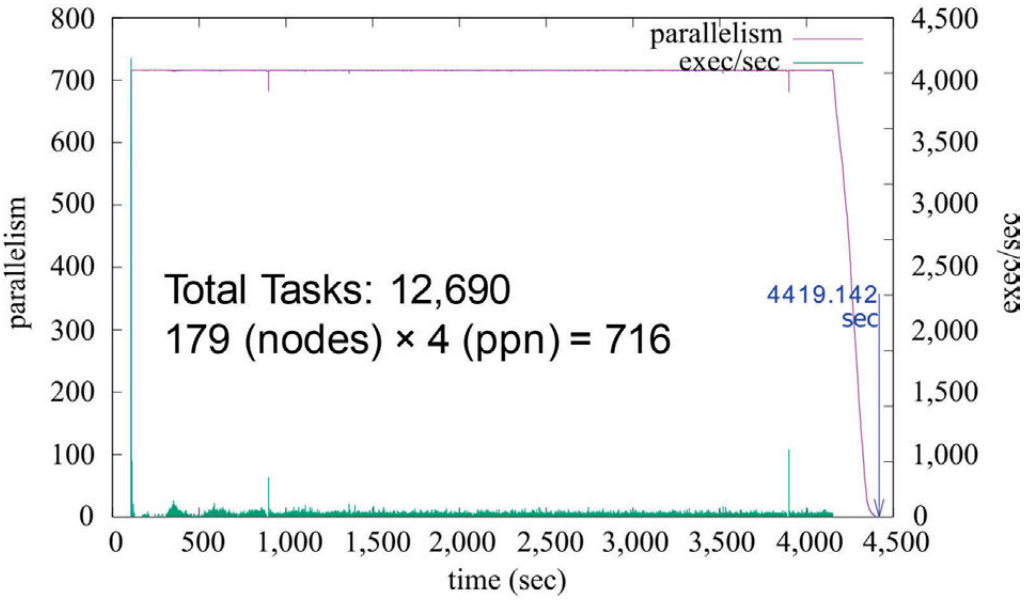


図 12 TSUBAME3.0 による大規模実行の模様

図 12 に東京工業大学のスーパコンピュータ TSUBAME3.0 の 180 ノードを用いた大規模実行の模様を示す。1 ノードは Gfarm/BB のメタデータサーバ，Pwrake ワークフローシステムのマスターサーバとして用い，残りの 179 ノードで演算を行っている。各ノードに GPU が 4 枚あるため，各ノードでは 4 プロセスを実行している。この実行の模様は最も時間を費やすアライメントプロセスを示しているが，実行開始から利用可能な計算ノードをフルに使っていることが分かる。62 GB のデータベースと 71 GB のクエリデータのメタゲノム解析を 180 ノード用いて前処理, 後処理を合わせて 2 時間弱で実行することが可能であった。本規模は我々が知る限りで最大のメタゲノム解析である。本成果は国際会議 Third IEEE International Workshop on Benchmarking, Performance Tuning and Optimization for Big Data Applications (BPOD 2019)において発表した。

【8】In transit 気象データ解析の研究（建部）

筑波大学計算科学研究センターの気象グループは City-LES 気象 LES モデルシミュレーションコードを開発している。City-LES は超高解像度の LES シミュレーションを行うため，さまざまなデータ解析が可能であるが，この研究では動的モード分解を行うことにより，乱流構造の解析を行う。動的モード分解は，時系列データのモード分解を行うものであり，時系列データを高速に格納し，データ解析時に高速に読み込むことが必要となる。そのために，本研究では Oakforest-PACS スーパコンピュータのバーストバッファシステム IME の利用を検討した。IME は，演算性能とストレージ性能の性能ギャップを埋めるため，並列ファイルシステムと演算ノードの間のストレージ階層として導入されたものである。

図 13 に City-LES の書込み性能を示す。本グラフは計算ノード数を増やしたときの書込み性能を示している。下の二本の線は並列ファイルシステムに書込みを行った場合である。1 ストライプの場合と 128 ストライプの場合が示されているが，128 ストライプの場合でも 10 GB/s ほどで頭打ちとなっている。一方，IME を用いた場合，256 ノードを用いた場合でも性能向上し，60 GB/s ほどの性能を達成している。読込みについては，複数の時系列データを読込むことが必要となるが，多次元配列のデータ分散が書込みプロセスと読込プロセスで異なることが問題となる。それぞれの読込プロセスが必要なデータを直接読み込むと，細切れとなり読込性能は劣化してしまう。これを解決するために，2 フェーズで読込を行い，読込は最も性能が良くなるデータ分散で読込を行い，その後ノード間で必要なデータ転送を行うようにした。これにより読込性能を改善することができた。本手法は，従来は 1 ファイルについて行われていたが，本研究ではそれを複数ファイルに応用した。本成果は国際会議 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT 2019)において発表した。

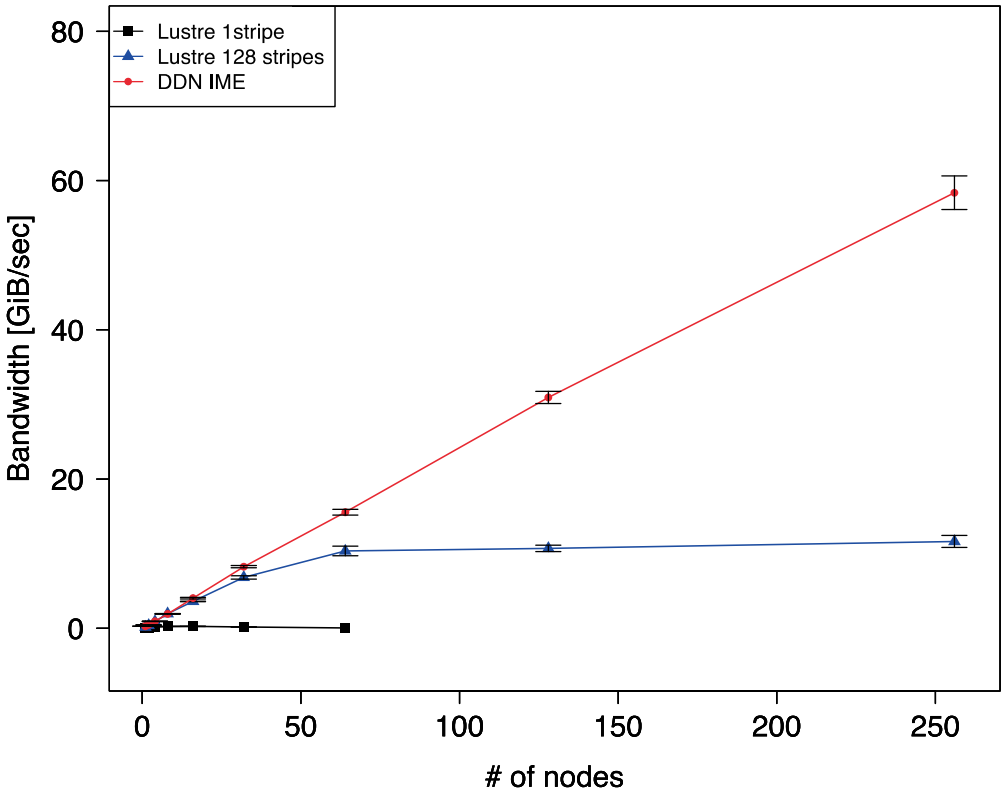


図 13 City-LES の書込み性能

【9】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，フェイルオーバ機能の高度化，暗号化ファイルシステム対応を行った。

Gfarm ファイルシステムは，メタデータ，ファイルデータが冗長化されており，単一障害についてはフェイルオーバにより隠蔽することが可能である。メタデータについては，マスターメタデータサーバと複数のスレーブメタデータサーバが動作しており，マスターメタデータサーバに障害が発生した時，各メタデータサーバのコミットログを確認し，最も新しいログをコミットしているサーバがマスターメタデータサーバに昇格する。この時，障害が発生したメタデータサーバのコミットログが参照できないと最新かどうかの確認ができないままフェイルオーバを行うこととなる。この場合，最悪ケースではコミット済みのログが失われてしまうこととなる。このケースについて，確認が取れるまでは読込オンリーファイルシステムとして動作するように改修を行った。これにより，コミットログの確認が取れるまでの間も安全にフェイルオーバが行えるようになった。

暗号化ファイルシステムの対応については，EncFS と組み合わせることにより Gfarm ファイルシステムに暗号化されたデータの格納を行う。このとき，EncFS はマルチスレッドでアクセスするが，Gfarm ライブラリはマルチスレッドセーフではないため逐次的にしか動作することができない。そのため，マルチスレッドセーフとなるように改修を行った。これにより，マルチスレッドプログラムからのファイルアクセスが高速となった。

これらの成果はまだリリースされていないが，まもなくリリースを予定している。

【10】ブロッククリロフ部分空間反復法の近似解精度向上に関する研究（多田野）

正則な n 次行列 A とL本の右辺ベクトルをもつ連立一次方程式：

$$AX = B \tag{1}$$

は，素粒子物理学分野における物理量計算や，疎行列に対する固有値解法の内部問題等で現れる。同方程式の求解速度，近似解精度はアプリケーションに対して大きな影響を及ぼすため，両面に優れた数値解法の開発が必要となる。連立一次方程式 (1) に対する効率的な数値解法として，ブロッククリロフ部分空間反復法がある。同法は複数の右辺ベクトルをもつ連立一次方程式を同時に解くことができ，クリロフ部分空間反復法と比較して少ない反復回数で解が得られることがある。その結果として，より高速な求解が可能となる。しかしながら，近似解精度の面では，右辺ベクトル数 L が多いと精度劣化が発生し，高精度の近似解が得られないことがある。我々はこれまで，ブロッククリロフ部分空間反復法の「積型」と呼ばれる解法群に注目し，Block BiCGGR 法や Block GWBiCGSTAB 法などの高精度近似解の生成が可能な解法を構築してきた。令和元年度は，残差の収束性に優れた GPBiCG 法を複数右辺ベクトル版に拡張した Block GPBiCG 法を構築するとともに，生成される近似解の精度改善に関する改良を行った。

連立一次方程式 (1) の第 $(k+1)$ 番目の近似解を X_{k+1} とすると対応する残差行列 R_{k+1} は

$$\boldsymbol{R_{k+1} = B - AX_{k+1}} \tag{2}$$

となる。X_{k+1} が真の解であるとき，R_{k+1} はゼロ行列となる。Block GPBiCG 法では，X_{k+1} と R_{k+1} は以下の漸化式で求められる。

$$X_{k+1} = X_k + P_k\alpha_k + Z_k, \tag{3}$$

$$R_{k+1} = R_k - (AP_k)\alpha_k - \eta_k Y_k - \zeta_k (AT_k). \tag{4}$$

ここで，P_k, Z_k, Y_k, T_k は $n \times L$ 補助行列，α_k は L 次行列，ζ_k, η_k はスカラーパラメータである。Block GPBiCG 法においては, パラメータ ζ_k, η_k は残差行列のフロベニウスノルム $\|R_{k+1}\|_{\mathrm{F}}$ が最小になるように決められる。また，$\eta_k = 0$ と固定して $\|R_{k+1}\|_{\mathrm{F}}$ が最小になるように ζ_k を決定すると，Block BiCGSTAB 法に帰着する。関係式 (2) と漸化式 (3), (4) より，$AZ_k = \eta_k Y_k + \zeta_k (AT_k)$ の関係が導かれる。

ここで，記号$\langle\cdot\rangle$は数値的に計算された誤差を含んだ行列を表すとする。真の残差行列$B - A\langle X_{k+1}\rangle$と漸化式で計算された残差行列$\langle R_{k+1}\rangle$の差を表す誤差行列を$E_{k+1}$とし，

$$E_{k+1} = B - A\langle X_{k+1}\rangle - \langle R_{k+1}\rangle \tag{5}$$

とする。丸め誤差のない正確な演算では，E_{k+1}はゼロ行列となる。式 (5) より，不等式：

$$\|E_{k+1}\| - \|\langle R_{k+1}\rangle\| \leq \|B - A\langle X_{k+1}\rangle\| \leq \|E_{k+1}\| + \|\langle R_{k+1}\rangle\|$$

が得られ，$\|\langle R_{k+1}\rangle\|$が十分小さくなったとしても$\|B - A\langle X_{k+1}\rangle\| \approx \|E_{k+1}\|$となり，真の残差行列は$E_{k+1}$の影響を受ける。ここでは，注目する誤差を「$n \times L$行列と$L$次行列の積で発生する丸め誤差」に限定し，それ以外の演算で発生する誤差は無視できると仮定する。このとき，誤差行列E_{k+1}は以下で表される。

$$E_{k+1} = \sum_{i=0}^{n} \left(E_i^{(Y)} + E_i^{(P)} \right),$$
$$E_i^{(Y)} \equiv \langle \eta_i Y_i \rangle + \langle \zeta_i (AT_i) \rangle - A\langle Z_i \rangle,$$
$$E_i^{(P)} \equiv \langle (AP_i)\alpha_i \rangle - A\langle P_i \alpha_i \rangle.$$

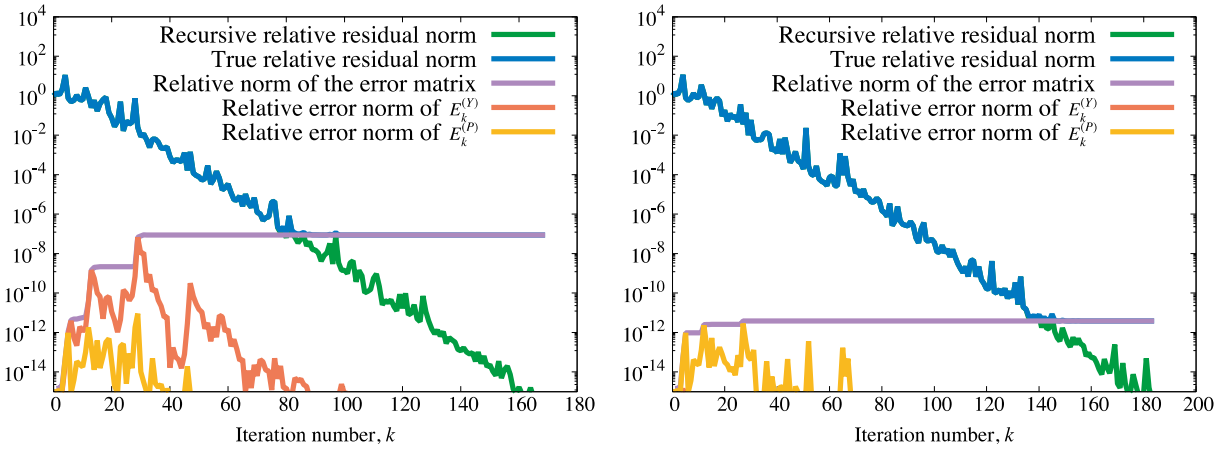
したがって，行列$E_i^{(Y)}$と$E_i^{(P)}$の影響を抑えることができれば誤差行列E_{k+1}の影響を抑えることができ，結果として近似解の精度改善につながる。

数値計算を行っている以上，丸め誤差は必ず発生し，これを完全になくすことは困難を極める。そこで本研究では，現在注目している「$n \times L$行列とL次行列の積で発生する丸め誤差」の発生は許容するが，行列$E_i^{(Y)}$，$E_i^{(P)}$に及ぼす影響を極力小さくするように，漸化式の再構築を行った。行列$E_i^{(Y)}$，$E_i^{(P)}$の影響を低減するアプローチをそれぞれ，提案手法 1，提案手法 2 と呼ぶ。これらの手法を組み込んだ方法を Modified Block GPBiCG 法と名付けた。ここでは漸化式の再構築に関する詳細は割愛するが，現在用いている誤差に関する仮定の下では，提案手法を用いることで行列$E_i^{(Y)}$と$E_i^{(P)}$はゼロ行列となる。また，ブロッククリロフ部分空間反復法では，右辺ベクトル数が増加すると残差行列の収束性が悪くなるため，これを緩和するために Modified Block GPBiCG 法に対して残差行列の正規直交化を組み込んだ手法も併せて開発した。この手法を Modified Block GPBiCGrQ 法と名付けた。また，パラメータη_kを 0 に固定した Modified Block BiCGSTAB 法，及び Modified Block BiCGSTABrQ 法も併せて開発した。

数値実験によって，提案手法の有効性を確認する。テスト問題として，最適化問題において現れる行列 majorbasis を係数行列にもつ連立一次方程式を用いた。同方程式のサイズnは 160,000，非零要素数は 1,750,416 である。右辺項Bは乱数で設定し，ベクトル数Lは 30 とした。実験環境は，CPU : Intel Xeon E5-2620v3 2.4GHz (6 cores)×2，メモリ : 64GiB DDR4 2133MHz，コンパイラ：Intel Fortran 19.0.0，コンパイルオプション：-qopenmp -axCORE-AVX2 であり，OpenMP を用いて 12 スレッド並列で計算した。

図 14 に，Block GPBiCG 法と Block BiCGSTAB 法における，漸化式で計算された相対残差ノルム（$\|R_k\|_F / \|B\|_F$），真の相対残差ノルム（$\|B - AX_k\|_F / \|B\|_F$），及び誤差行列の相対ノルムの（$\|E_k\|_F / \|B\|_F$，$\left\|E_k^{(Y)}\right\|_F / \|B\|_F$，$\left\|E_k^{(P)}\right\|_F / \|B\|_F$）変化を示す。図 14(a) に示すように，Block GPBiCG 法では行列 $E_k^{(Y)}$，$E_k^{(P)}$ の相対ノルムがともに大きい値になっており，その影響で誤差行列 E_k の相対ノルムも大きくなっている。漸化式の相対残差ノルムは減少しているが，真の相対残差ノルムは途中で停滞している。一方，Block BiCGSTAB 法においては，行列 $E_k^{(Y)}$の影響は小さいが，$E_k^{(P)}$ の影響により誤差行列 E_k の相対ノルムが大きくなり，その影響で相対残差ノルムが反復途中で停滞している。

図 15 に，提案手法を適用した場合の Block GPBiCG 法，Block BiCGSTAB 法における漸化式で計算された相対残差ノルム，真の相対残差ノルム，及び誤差行列の相対ノルムの変化を示す。図 15 (a) に示すように，提案手法 1 を適用することで行列 $E_k^{(Y)}$ の影響を抑えることができる。同様に，図 15 (b) に示すように提案手法 2 を適用することにより，行列 $E_k^{(P)}$の影響を低減できる。しかしながら，誤差行列 E_k に影響を及ぼす要因が残っているため，いずれの場合も真の相対残差ノルムは反復途中で停滞している。一方，図 15 (c) に示すように Block GPBiCG 法に対して提案手法 1 と 2 の両方を適用することで，真の相対残差ノルムを十分小さくすることができた。また，Block BiCGSTAB 法では行列 $E_k^{(P)}$が強く影響していたが，図 15 (d) に示すように，提案手法 2 を適用することで誤差行列 E_k の影響を低減することができた。

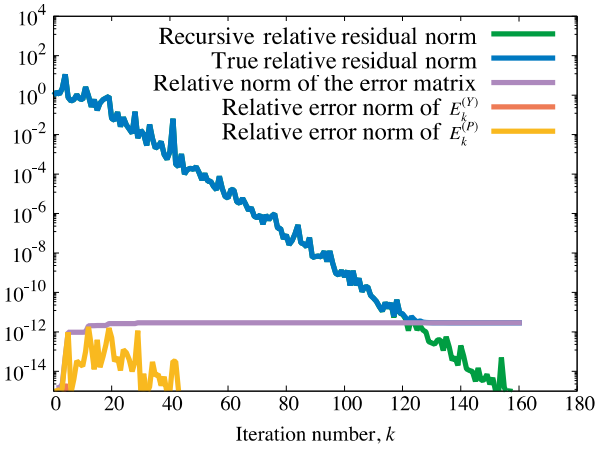


(a) Block GPBiCG 法　　(b) Block BiCGSTAB 法

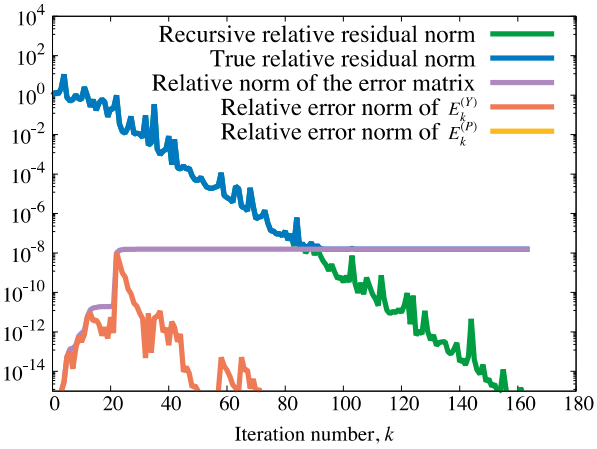
図 14　漸化式で計算された相対残差ノルム，真の相対残差ノルム，及び誤差行列の相対ノルムの変化

次に，右辺ベクトル数 L を変化させたときの真の相対残差ノルム，反復回数，及び右辺ベクトル 1 本あたりの求解時間の変化を示す。図 16 (a) に示すように，提案手法を適用しない

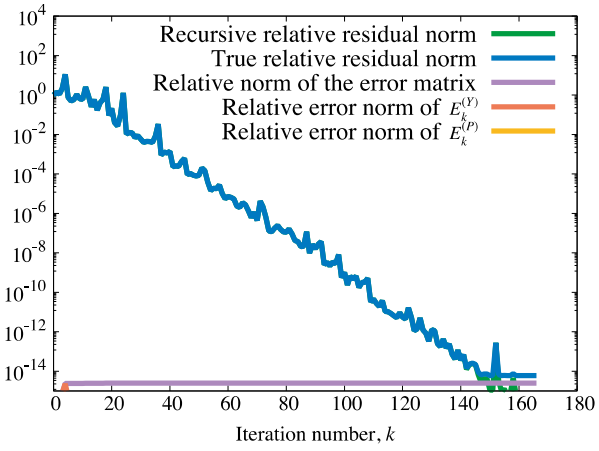
場合は右辺ベクトル数の増加にしたがって真の相対残差ノルムが増加する傾向にあるが，提案手法を適用することで増加を抑えることができた。また，図 16 (b), (c) に示すように，右辺ベクトル数の増加に伴って，求解に要した反復回数，及び右辺ベクトル 1 本あたりの求解時間は減少傾向にあった。



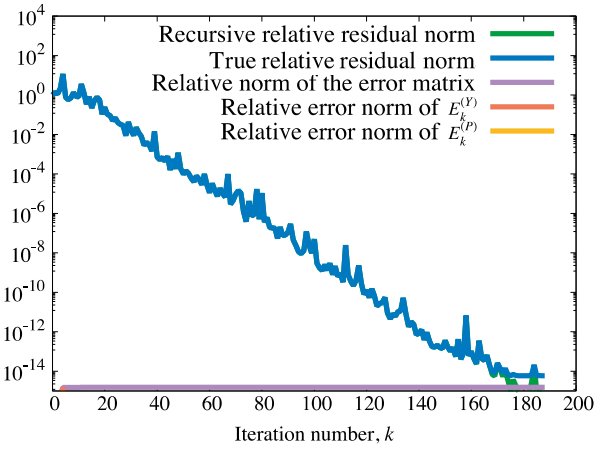
(a) Block GPBiCG 法＋提案手法 1



(b) Block GPBiCG 法＋提案手法 2

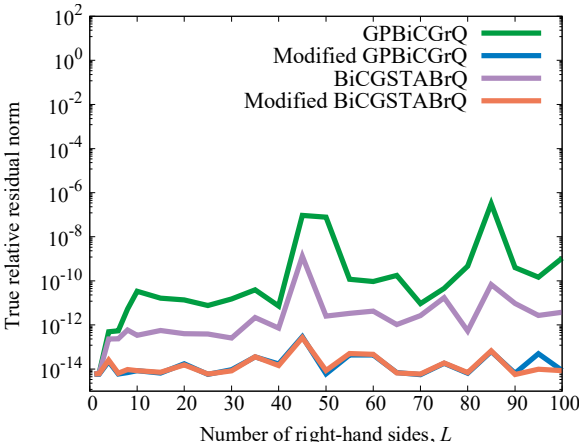


(c) Block GPBiCG 法＋提案手法 1,2

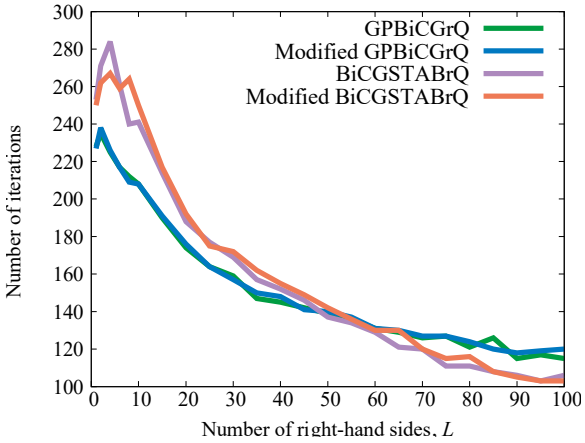


(d) Block BiCGSTAB 法＋提案手法 2

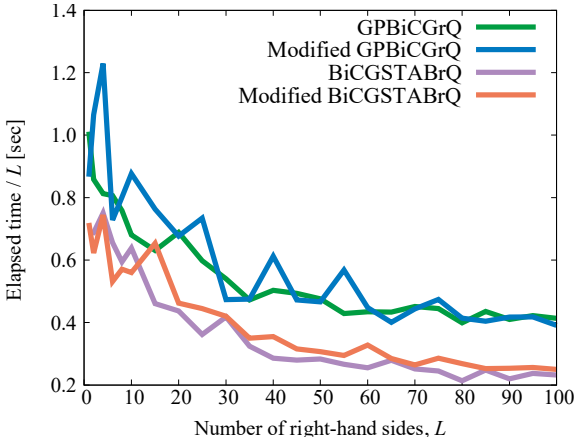
図 15　提案手法を適用した場合の漸化式で計算された相対残差ノルム, 真の相対残差ノルム, 及び誤差行列の相対ノルムの変化



(a) 真の相対残差ノルムの変化



(b) 反復回数の変化



(c) 右辺ベクトル 1 本あたりの求解時間

図 16　右辺ベクトル数 L の変化が真の相対残差ノルム，反復回数，右辺ベクトル 1 本あたりの求解時間に与える影響

【11】GPU・FPGA 連携による宇宙輻射輸送コードの演算加速 (小林，藤田，朴)

GPU・FPGA 複合演算加速が必要とされる理由は，複数の物理モデルや複数の同時発生する物理現象を含むシミュレーションであるマルチフィジックスアプリケーションに対して有効と睨んでいるためである。マルチフィジックスでは，シミュレーション内に様々な特性の演算が出現するので，GPU だけでは演算加速が困難な場合がある。そのため，GPU だけでは対応しきれない特性の演算の加速に FPGA を利用することで，アプリケーション全体の性能向上を狙う。

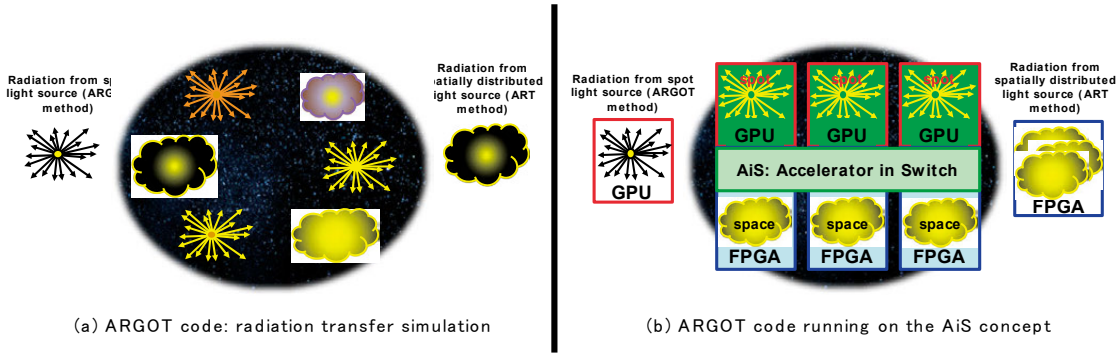


図 17 (a) 宇宙輻射輸送コード ARGOT の概観．(b) GPU・FPGA 連携による ARGOT コードの演算加速

本研究では，マルチフィジックスの例である，宇宙輻射輸送シミュレーションコード Accelerated Radiative transfer on Grids using Oct-Tree (ARGOT) を対象にする。ARGOT は本センターで開発されている宇宙輻射輸送を解くプログラムであり，図 17 (a) に示すように，点光源と空間に分散した光源の 2 種類の輻射輸送問題を含む。なお，点光源からの輻射輸送の計算を ARGOT コードにおける ARGOT 法，空間に広がる光源からの輻射輸送の計算を ARGOT コードにおける ART 法呼ぶ。ここで重要なのは，ART 法は ARGOT プログラムの中で 90%以上の計算時間を占めるアルゴリズムであり，本研究では ART 法の演算をこれまでに開発してきた FPGA カーネルを用いて加速させる。また，ARGOT 法の演算には既に ARGOT プログラムに実装されている GPU カーネルを用いることで，図 17 (b) に示すように主要演算部分を GPU と FPGA に適材適所的に機能分散して ARGOT コードを最適化する。

図 18 に CPU 実装，GPU 実装，FPGA 実装の性能比較を示す．ARGOT(CPU) / ART(CPU) は ARGOT 法と ART 法がどちらも CPU 実装，ARGOT(GPU) / ART(GPU) はどちらも GPU 実装，ARGOT(GPU) / ART(FPGA) は ARGOT 法が GPU 実装，ART 法が FPGA 実装であることを表す。また，w/ DMA，wo/ DMA はデバイス間 DMA を用いているか，用いていないか，をそれぞれ表す。CPU 実装は C 言語ベースであり，OpenMP を用いたスレッド並列処理が適用されている。本研究では，単一の Xeon CPU (14 コア) を利用しており，1 コア 1 スレッドのマッピングでプログラムを実行した．GPU 実装は，CPU 実装をベースとしており，コードは CUDA で記述されている。

ARGOT 法と ART 法をどちらも CPU で実行した場合，すなわち ARGOT(CPU) / ART(CPU) を 1 としたときの ARGOT コードの実行速度向上比を図 16 に示す。ARGOT(GPU) / ART(GPU) に注目すると，メッシュサイズ 16^3 と 32^3 における速度向上は，それぞれ 1.18 倍，2.19 倍である。これは，P100 GPU にとってこれらのメッシュサイズが小さすぎるため，3,584 CUDA Core に対して十分な演算の並列度が得られないことに起因している。そのため，メッシュサイズ

が大きくなるにつれて，ARGOT(GPU) / ART(GPU)の性能は向上し，メッシュサイズ 128^3 においては，ARGOT(CPU) / ART(CPU) の 18.4 倍の性能となる。

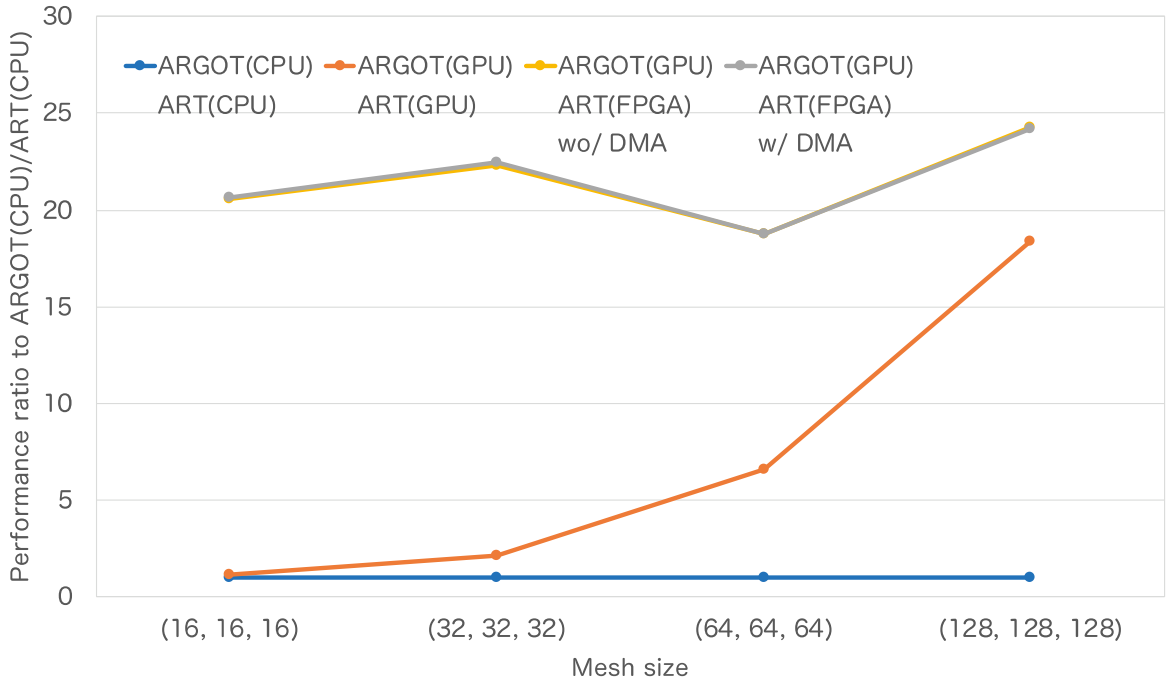


図 18 ARGOT 法と ART 法をどちらも CPU で実行した場合を 1 としたときの ARGOT コードの実行速度向上比

これに対し，ARGOT(GPU) / ART(FPGA) は，デバイス間 DMA の有無によらず，どの問題サイズであっても常に GPU 実装より優れていることが分かる。DMA の有無によって性能差がほとんど生じないのは，ARGOT コード中の GPU--FPGA 間通信が全体の処理の 1%程度であるためである。先行研究で報告されている通り，FPGA 版 ART は空間並列だけでなくパイプラインによって時間方向にも並列計算を行う。そのため，問題サイズが小さい場合であっても性能を発揮でき，その場合，GPU を大幅に凌駕する性能を持つことを示した。

【12】OpenCL 対応 FPGA 間通信機能による GPU・FPGA 複合型演算加速 (小林，藤田，朴)

GPU や FPGA といった異なる種類の計算デバイスがノード内に混在するような複雑なプラットフォーム上では，各デバイスで実行される演算をどのようにプログラミングし，全デバイスを協調動作させるかが重要な課題となる。これまでに我々は，GPU デバイスのグローバルメモリと FPGA デバイスの外部メモリ間で CPU を介さずにデータ転送を実現する機能を，PCIe DMA 転送用の IP (Intellectual Property) コアを用いて FPGA 上に実装し，その機能を FPGA ベンダーの提供する OpenCL ツールチェインの仕組みと Verilog HDL とを活用することによ

って制御する手法を提案している。その GPU-FPGA 間 DMA 転送技術に加え，我々は OpenCL から制御可能な FPGA 間通信技術も開発しており，本研究ではこれらの技術を融合し，複数ノード上における GPU-FPGA 間連携を実現した。

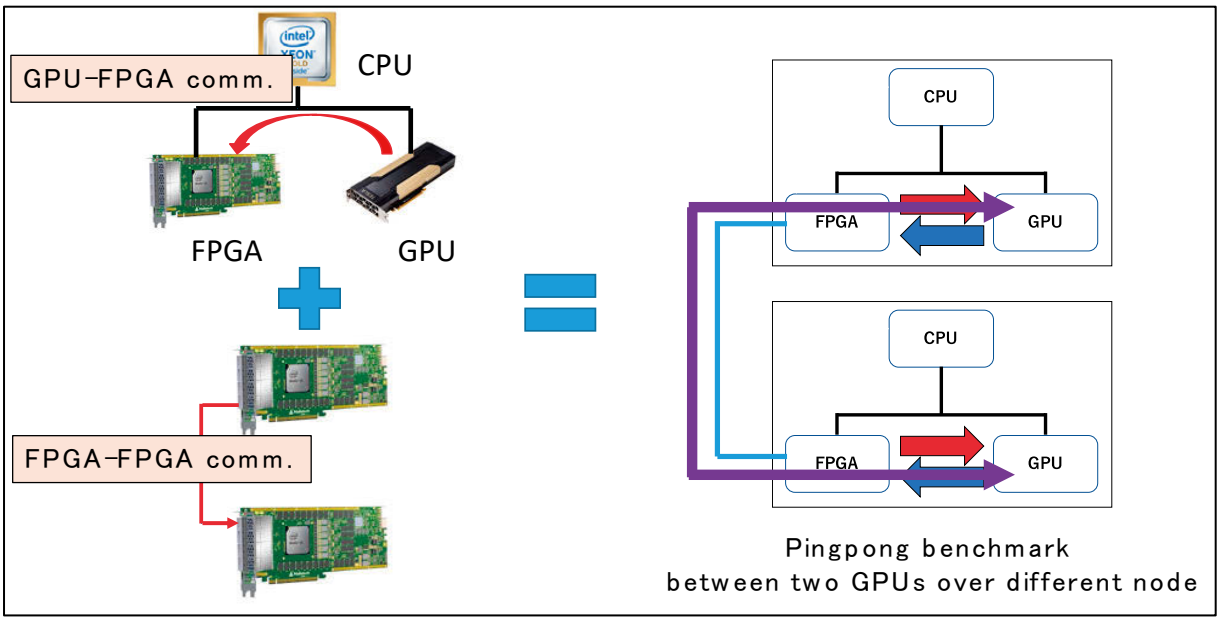


図 19 OpenCL 対応 GPU-FPGA 間通信技術と FPGA-FPGA 間通信技術との融合

複数ノード上における GPU-FPGA 間連携が正しく実現できているかを検証するために，図 19 に示すような pingpong ベンチマークを行った。この pingpong ベンチマークでは，初期データはノード 0 の GPU メモリに cudaMemcpy によって書き込まれ，それは FPGA を介することによって，ノード 1 の GPU に送信される．その後，ノード 1 の GPU で受信したデータは，FPGA を経由してノード 0 の GPU に返信される。これを実現するために，ノード 0 の FPGA では，初期データを送信するカーネル Kernel ping とノード 1 から送信されたデータを受信する Kernel pong が起動しており，ノード 1 の FPGA では，ノード 0 からのデータを受信して送り返すカーネル Kernel reply が起動している。なお，起動するカーネルは，MPI の rank に応じて選択している。

図 20 に 4 バイト通信時におけるノードを跨いだ GPU 同士の pingpong ベンチマークにおける通信レイテンシを示す。レイテンシの実測値は 5.04 μ sec であり，理論値は 3.02 μ sec である。なお，理論値は先行研究の結果をベースとしている。このため，理論性能と比べて 2.02 μ sec 余分にかかっていることが分かる。しかしこれは，現在の実装が最適化されていないことに大きく依存している。

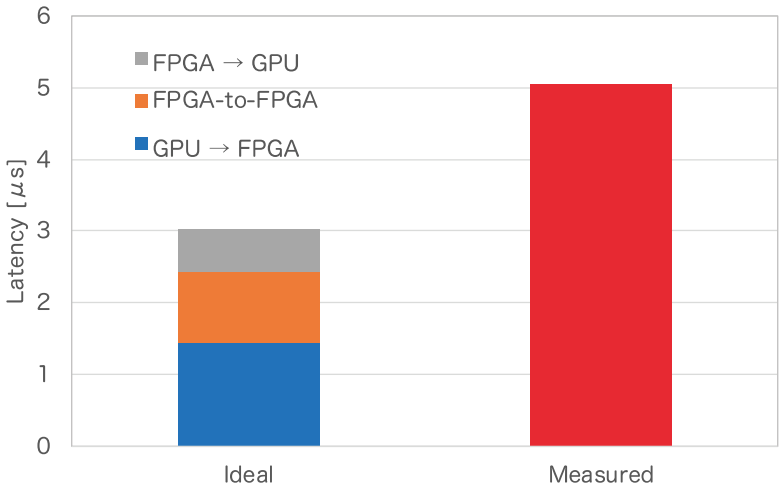


図 20 ノードを跨いだ GPU 同士の pingpong ベンチマークにおける通信レイテンシ

ここで重要なことは，我々が提案した手法により，GPU と FPGA が異なるノード上で協調して動作することが可能になったことである。これは，FPGA についての深い知識や多大な実装労力を必要とせずに，GPU・FPGA 中心のアプリケーションコードを実装できることを意味する。今回の評価結果をもとに，異なるノード上で複数の GPU と FPGA 上で動作する HPC アプリケーションを実装し，その性能を評価することが今後の課題である．

【13】OpenCL による FPGA 上の演算と通信を融合した並列処理システムの開発（藤田，小林，朴）

近年，Field Programmable Gate Array (FPGA)が高性能計算の分野で注目されているが，絶対的な演算性能は他の演算加速装置と比べて弱く，適用できる計算が少ないという問題がある。我々は FPGA が持つ強力な通信機構に注目しており，他の演算加速装置はこの特性を有しない FPGA に固有のものである。従来の FPGA 開発においては，ハードウェア記述言語（HDL）を用いて低レベルな記述をしなければならず，開発コストが高いという問題があった。この問題は，ソフトウェアで用いられる言語（C, C++, OpenCL 等）を用いてハードウェアを記述できる高位合成（HLS）技術の発展により解決されつつある．科学技術計算を記述することは HLS を用いて実現できるが，通信機構を扱うことは容易ではない。そこで，我々は，OpenCL から通信機構を扱えるフレームワーク Communication Integrated Reconfigurable CompUting System (CIRCUS)の研究開発を行っている。

本センターでは 1 ノードあたり 2 FPGA ボードを搭載するスーパーコンピュータ Cygnus を運用しており，本研究では CIRCUS 通信フレームワークを Cygnus 上に実装し性能評価を行った。評価結果を下図に示す。この評価は Cygnus に搭載されている FPGA ボードを最大 8 枚用いたものであり，1-hop が隣接する FPGA 間通信，2-hops が隣接の隣接 FPGA 間通

信を意味し，最大で 7-hops までの評価となる。最大スループットは 90.2Gbps，最小レイテンシは 500ns を達成した。また，CIRCUS はルーターを内蔵しており隣接していない FPGA 間で通信が行える。その際の 1-hop あたりの追加レイテンシは約 250ns という結果が得られた。今後は，CIRCUS 通信を実際の高性能計算アプリへ追加し，複数の FPGA を用いた並列計算を実施していきたいと考えている。

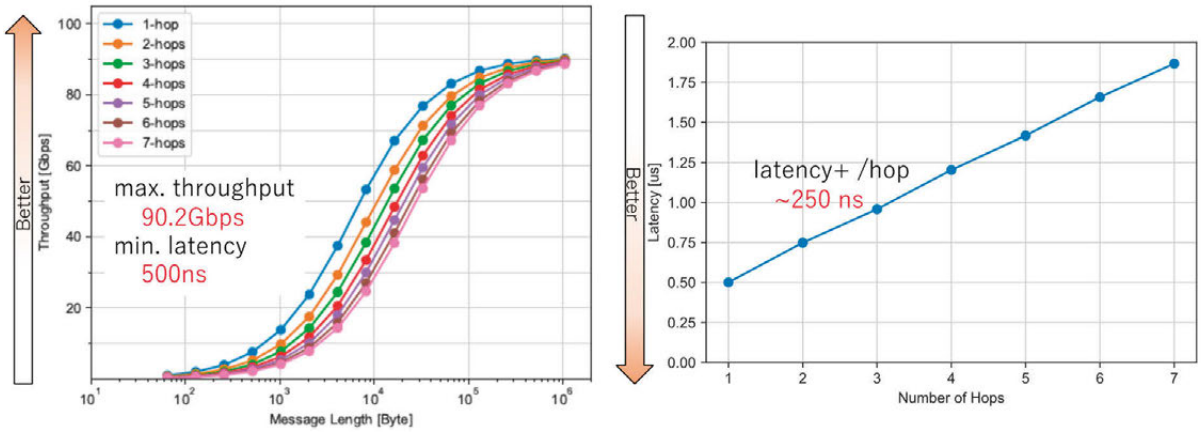


図 21　CIRCUS 通信のスループット（左），通信レイテンシ（右）の評価結果.

4. 教育

学生の指導状況（学生氏名，学位の種類，論文名）

1. 辻大亮，修士（工学），都市気象コード City-LES の GPU クラスタによる高速化，筑波大学大学院システム情報工学研究科修士業論文，令和 2 年 3 月（指導 : 朴泰祐）
2. 綱島隆太，修士（工学），GPU・FPGA 協調プログラミング環境に関する研究，筑波大学大学院システム情報工学研究科修士業論文，令和 2 年 3 月（指導 : 朴泰祐）
3. 中道安祐未，修士（工学），GPU・FPGA 複合演算加速による宇宙物理コードの高速化，筑波大学大学院システム情報工学研究科修士業論文，令和 2 年 3 月（指導 : 朴泰祐）
4. 枝松拓弥，修士（工学），AVX-512 命令を用いた多倍長整数乗算の高速化，筑波大学大学院システム情報工学研究科修士業論文，令和 2 年 3 月（指導 : 高橋大介）
5. 辻祥太，修士（工学），メニーコアプロセッサにおける行列転置の実装と評価，筑波大学大学院システム情報工学研究科修士業論文，令和 2 年 3 月（指導 : 高橋大介）
6. 堀江悠樹，修士（工学），分散合意手法 Raft を用いたトランザクション処理の実装と評価，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員 : 建部修見）

7. 河合祐輔，修士（工学），高次元データの本質的な次元推定の高速化と性能評価，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
8. 北澤昂大，修士（工学），クラウドストレージによる階層型ストレージシステムの研究，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
9. 町田健太，修士（工学），大規模データに対するメタゲノム解析システムの研究，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
10. 澤田一樹，修士（工学），分散バーストバッファの I/O 性能に関する研究，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
11. 芹沢和洋，修士（工学），大規模機械学習の高速化に関する研究，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
12. 田村駿弥，修士（工学），空間索引木 R-tree の並列挿入手法に関する研究，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
13. 田辺敬之，修士（工学），トランザクション処理における並行性制御法の評価，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
14. 佐藤拓人，修士（工学），気象学大規模乱流構造解析ワークフローに関する研究，筑波大学大学院システム情報工学研究科修士論文，令和 2 年 3 月（指導教員：建部修見）
15. 西野裕貴，学士（工学），光・電子融合第一原理アプリケーションの GPU による高速化，筑波大学情報学群情報科学類卒業論文，令和 2 年 3 月（指導：朴泰祐）
16. 渡邉孔英，学士（工学），演算加速装置による生命科学向け分子動力学法の高速化，筑波大学情報学群情報科学類卒業論文，令和 2 年 3 月（指導：朴泰祐）
17. 柏野隆太，学士（工学），大規模高性能計算のための FPGA 間通信技術に関する研究，筑波大学情報学群情報科学類卒業論文，令和 2 年 3 月（指導：小林諒平）
18. NAM WOON，学士（工学），Smoothing filter を用いた Screen Space Ambient Occulusion への適用，筑波大学情報学群情報科学類卒業論文，令和 2 年 3 月（指導：高橋大介）
19. 杉﨑行優，学士（工学），Fast Computation of the Exact Number of Magic Series with an Improved Montgomery Multiplication Algorithm（改良版モンゴメリ乗算アルゴリズムを用いた魔法列の正確かつ高速な数え上げ），筑波大学情報学群情報科学類卒業論文，令和 2 年 3 月（指導：高橋大介）

20. 倉本健，学士（工学），分散深層学習におけるデータ読込性能の評価，筑波大学情報学群情報科学類卒業論文，令和 2 年 3 月（指導教員：建部修見）
21. 石川翔大，学士（工学），ブロッククリロフ部分空間反復法による鞍点型連立一次方程式の求解高速化，筑波大学情報学群情報科学類卒業研究論文，令和 2 年 3 月（指導：多田野寛人）

5. 受賞，外部資金，知的財産権等

受賞

1. SCA2020, Taisuke Boku, Asia HPC Leadership Award, Feb. 26th, 2020.

外部資金

1. 文部科学省高性能汎用計算機高度利用事業費補助金　朴泰祐（代表）, H29～33 年度, 32,580 千円（R01 年度），「次世代演算通信融合型スーパーコンピュータの開発」
2. 科学研究費補助金基盤研究 (B)（一般），朴泰祐（代表），H30～32 年度，5,500 千円（R01 年度），「再構成可能システムと GPU による複合型高性能計算プラットフォーム」
3. 理化学研究所受託研究，朴泰祐（代表）, H27～H32 年度, 6,600 千円（R01 年度），「ポスト京の並列プログラミング環境およびネットワークに関する研究」
4. 科学研究費補助金基盤研究 (C)，高橋大介（代表），H31～R3 年度，1,040 千円（H31 年度），「エクサスケールシステムにおける高速フーリエ変換のアルゴリズムに関する研究」
5. 科学研究費補助金　基盤研究(B)（一般），建部修見（代表），H29～H31 年度，7,930 千円，「極端気象予測を拓くビッグデータ機械学習基盤の研究」
6. NEDO，建部修見（分担），H30～H32 年度，9,531 千円，「実社会の事象をリアルタイム処理可能な次世代データ処理基盤技術の研究開発」
7. 共同研究（富士通研究所），建部修見（代表），H31 年度，3,630 千円，「大量データ管理に向けたストレージシステム」
8. 科学研究費補助金若手研究，小林諒平（代表），H31～H32 年度，2,470 千円（R01 年度），「FPGA を用いた超高速ハードウェアソーティング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. 藤田典久, 小林諒平, 山口佳樹, 朴泰祐, 吉川耕司, 安部牧人, 梅村雅之, 宇宙輻射輸送コードにおける OpenCL による FPGA 演算加速最適化, 情報処理学会論文誌トランザクション コンピューティングシステム (ACS), 12 巻, 3 号, pp.64-75, 2019.
2. Daisuke Takahashi and Franz Franchetti, “FFTE on SVE: SPIRAL-Generated Kernels”, Proc. International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2020), pp. 114-122 (2020).
3. Daisuke Takahashi, “On the computation and verification of π using BBP-type formulas”, The Ramanujan Journal, Vol. 51, No. 1, pp. 177-186 (2020).
4. Daisuke Takahashi, “Implementation of Parallel 3-D Real FFT with 2-D Decomposition on Intel Xeon Phi Clusters”, Proc. 13th International Conference on Parallel Processing and Applied Mathematics (PPAM 2019), Part I, Lecture Notes in Computer Science, Vol. 12043, pp. 151-161, Springer (2020).
5. Takuya Edamatsu and Daisuke Takahashi, “Accelerating Large Integer Multiplication Using Intel AVX-512IFMA”, Proc. 19th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2019), Part I, Lecture Notes in Computer Science, Vol. 11944, pp. 60-74, Springer (2020).
6. Samar Aseeri, Benson K. Muite, and Daisuke Takahashi, “Reproducibility in Benchmarking Parallel Fast Fourier Transform based Applications”, Companion of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE'19), pp. 5-8 (2019). (vision paper)
7. Yasuhiro Nakamura, Hideyuki Kawashima, Osamu Tatebe, "Integration of TicToc Concurrency Control Protocol with Parallel Write Ahead Logging Protocol", International Journal of Networking and Computing, Vol.9, No.2, pp.339-353, doi: 10.15803/ijnc.9.2_339, 2019
8. Harunobu Daikoku, Hideyuki Kawashima, and Osamu Tatebe, "Skew-Aware Collective Communication for MapReduce Shuffling", IEICE Transactions on Information and Systems, Vol.E102-D, No.12, pp.2389-2399, doi: 10.1587/transinf.2019PAP0019, 2019
9. Takuto Sato, Osamu Tatebe, Hiroyuki Kusaka, "In-situ data analysis system for high resolution meteorological large eddy simulation model", Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, pp.155-158, doi: 10.1145/3365109.3368769, 2019

10. Kazuhiro Serizawa, Osamu Tatebe, "Accelerating Machine Learning I/O by overlapping data staging and mini-batch generations", Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, pp.31-34, doi: 10.1145/3365109.3368768, 2019
11. Kenta Machida, Osamu Tatebe, "GHOSTZ PW/GF: Distributed Parallel Homology Search System for Large-scale Metagenomic Analysis", Proceedings of the Third IEEE International Workshop on Benchmarking, Performance Tuning and Optimization for Big Data Applications (BPOD 2019), pp.3492-3700, doi: 10.1109/BigData47090.2019.9006499, 2019
12. Osamu Tatebe, Shukuko Moriwake, Yoshihiro Oyama, "Gfarm/BB - Gfarm file system for node-local burst buffer", Journal of Computer Science and Technology, Vol.35, Issue 1, pp.61-71, doi: 10.1007/s11390-020-9803-z, 2020
13. Kohei Hiraga, Osamu Tatebe, Hideyuki Kawashima, "Scalable Distributed Metadata Server Based on Nonblocking Transactions", Journal of Universal Computer Science, Vol.26, Issue 1, pp.89-106, 2020
14. Hiroto Tadano, Development of the Block BiCGGR2 method for linear systems with multiple right-hand sides, Japan Journal of Industrial and Applied Mathematics, Vol. 36, No. 2, pp. 563—577, 2019.
15. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Ayumi Nakamichi, Taisuke Boku: OpenCL-enabled GPU-FPGA Accelerated Computing with Inter-FPGA Communication, (short paper), IXPUG Workshop at HPC Asia 2020, pp.17-20, January 2020
16. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Ayumi Nakamichi and Taisuke Boku: GPU-FPGA Heterogeneous Computing with OpenCL-enabled Direct Memory Access, 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)/pp.489-498, 2019-07
17. Norihisa Fujita, Ryohei Kobayashi, Yoshiki Yamaguchi, Taisuke Boku, Parallel Processing on FPGA Combining Computation and Communication in OpenCL Programming, Proceedings of 2019 IEEE International Parallel and Distributed Processing Symposium Workshops, pp.479-488, 2019.

B) 査読無し論文

1. 辻大亮, 多田野寛人, 朴泰祐, 池田亮作, 佐藤拓人, 日下博幸, "都市気象コード City-LES の並列 GPU 実装の最適化と性能評価", 情報処理学会研究報告ハイパフォーマンスコンピューティング(HPC), 2019-HPC-170(5), pp.1-9, 2019 年 7 月.

2. 辻大亮, 朴泰祐, 池田亮作, 佐藤拓人, 多田野寛人, 日下博幸, "都市気象コード City-LES の OpenACC による並列 GPU 実装とデータ転送最適化", 情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC）, 2020-HPC-173(22), pp.1-8, 2020 年 3 月.
3. 綱島 隆太, 小林 諒平, 藤田 典久, 中道 安祐未, 朴 泰祐, " GPU-FPGA 協調プログラミングを実現するコンパイラの開発", 情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC）, 2019-HPC-172(11), pp.1-10, 2019 年 12 月.
4. 渡部裕, 李珍泌, 佐野健太郎, 朴泰祐, 佐藤三久, "ストリーム計算による最適化を併用する FPGA 向け OpenMP コンパイラの試作", 情報処理学会研究報告ハイパフォーマンスコンピューティング(HPC), 2019-HPC-170(3), pp.1-8, 2019 年 7 月.
5. 綱島隆太, 小林諒平, 藤田典久, 中道安祐未, 朴泰祐, "GPU-FPGA 協調計算を記述するためのプログラミング環境に関する研究", 情報処理学会研究報告ハイパフォーマンスコンピューティング(HPC), 2019-HPC-169(10), pp.1-9, 2019 年 5 月.
6. 渡部裕, 李珍泌, 朴泰祐, 佐藤三久, "演算加速機構を対象とするタスク並列ランタイムにおけるデータ移動最適化", 情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC）, 2020-HPC-173(11), pp.1-9, 2020 年 3 月.
7. 枝松拓弥，高橋大介，“AVX-512IFMA を用いた多倍長整数乗算の高速化”，日本応用数理学会 2019 年度年会講演予稿集, 2 pages, 2019.
8. 高橋大介，“Xeon Phi クラスタにおける二次元分割を用いた並列三次元実数 FFT の実現と評価”，日本応用数理学会 2019 年度年会講演予稿集, 2 pages, 2019.
9. 芹沢 和洋, 建部 修見，大規模機械学習訓練における I/O 性能の高速化，研究報告ハイパフォーマンスコンピューティング（HPC），2019-HPC-170 (9), 12 pages, 2019 年 7 月
10. 北澤 昂大, 建部 修見, Gfarm とクラウドストレージによる階層型ストレージシステムの研究，研究報告ハイパフォーマンスコンピューティング（HPC），2019-HPC-170(25), 7 pages, 2019 年 7 月
11. 町田 健太, 建部 修見, 大規模ゲノムデータに対す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），2019-HPC-170 (31), 9 pages, 2019 年 7 月
12. 高橋 宗史，建部 修見，分散オブジェクトストレージ Ceph のための Spark ストレージアダプタの設計，研究報告ハイパフォーマンスコンピューティング（HPC），2019-HPC-171 (1), 8 pages, 2019 年 9 月

13. 畑中 智之，建部 修見，ユーザ権限における Docker オーケストレーションの構築と撤去，研究報告ハイパフォーマンスコンピューティング（HPC），2019-HPC-171 (3), 6 pages, 2019 年 9 月
14. 町田 健太, 建部 修見, 分散並列相同性検索システム GHOSTZ PW/GF の大規模環境における評価, 研究報告ハイパフォーマンスコンピューティング（HPC）, 2019-HPC-172(2), 9 pages, 2019 年 12 月
15. 澤田 一樹, 建部 修見, 分散バーストバッファが持つ I/O 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）, 2019-HPC-172(20), 6 pages, 2019 年 12 月
16. 小林 諒平, 藤田 典久, 中道 安祐未, 山口 佳樹, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村, 雅之, GPU・FPGA 複合演算加速による宇宙輻射輸送コード ARGOT 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） 2020-HPC-173(8) 1 - 11 2020 年 3 月
17. 小林 諒平, 藤田 典久, 中道 安祐未, 山口 佳樹, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村, 雅之, OpenCL 対応 GPU・FPGA デバイス間連携機構による宇宙輻射輸送コードの演算加速, 研究報告ハイパフォーマンスコンピューティング（HPC） 2019-HPC-172(8) 1 - 9 2019 年 12 月
18. 小林 諒平, 藤田 典久, 山口 佳樹, 中道 安祐未, 朴 泰祐, OpenCL 対応 FPGA 間通信機能による GPU・FPGA 複合型演算加速, 研究報告ハイパフォーマンスコンピューティング（HPC） 2019-HPC-170(5) 1 - 9 2019 年 7 月
19. 中道 安祐未, 藤田 典久, 小林 諒平, 朴 泰祐, 吉川 耕司, 梅村 雅之, GPU・FPGA 複合演算加速による輻射流体シミュレーションコード ARGOT の実装, 研究報告ハイパフォーマンスコンピューティング（HPC） 2019-HPC-170(22) 1 - 5 2019 年 7 月
20. 藤田 典久, 小林 諒平, 山口 佳樹, 上野 知洋, 佐野 健太郎, 朴 泰祐, スーパーコンピュータ Cygnus 上における FPGA 間パイプライン通信の性能評価, 研究報告ハイパフォーマンスコンピューティング (HPC), 2020-HPC-173, pp.1-11, 2020.
21. 藤田 典久, 小林 諒平, 山口 佳樹, 朴 泰祐, 再構成可能なハードウェアを用いた演算と通信を融合する手法の提案と性能評価, 研究報告ハイパフォーマンスコンピューティング (HPC), 2020-HPC-171, pp.1-9, 2019.

(2) 国際会議発表

A) 招待講演

1. Taisuke Boku, “Japanese Supercomputer Development and Hybrid Accelerated Supercomputing”, HPC-AI Advisory Council, Perth, Australia, Aug. 27th, 2019.

2. Taisuke Boku, “Next Generation Accelerated Supercomputing: Cygnus System at University of Tsukuba”, AHeDD2019/IPAB2019, Kawasaki, Nov. 29th, 2019.
3. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Taisuke Boku, Kohji Yoshikawa, Masayuki Umemura: Cygnus: GPU + FPGA accelerated supercomputing platform, 1st International Workshop on Reconfigurable High Performance Computing (ReHPC'2019), Sep. 13th, 2019.

B) 一般講演

1. Daisuke Takahashi and Franz Franchetti, “FFTE on SVE: SPIRAL-Generated Kernels”, Proc. International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2020), ACROS Fukuoka, Fukuoka, Japan, January 16, 2020.
2. Takuya Edamatsu and Daisuke Takahashi, “Accelerating Large Integer Multiplication Using Intel AVX-512IFMA”, 19th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2019), Deakin Downtown, Melbourne, Australia, December 9, 2019.
3. Daisuke Takahashi, “Implementation of Parallel 3-D Real FFT with 2-D Decomposition on Intel Xeon Phi Clusters”, 13th International Conference on Parallel Processing and Applied Mathematics (PPAM 2019), Bialystok University of Technology, Bialystok, Poland, September 11, 2019.
4. Samar Aseeri, Benson K. Muite, and Daisuke Takahashi, “Reproducibility in Benchmarking Parallel Fast Fourier Transform based Applications”, 2019 ACM/SPEC International Conference on Performance Engineering (ICPE'19), Indian Institute of Technology, Bombay, Mumbai, India, April 9, 2019.
5. Osamu Tatebe, “Bridging a gap between computing and storage”, France-Japan-Germany trilateral workshop: Convergence of HPC and Data Science for Future Extreme Scale Intelligent Applications, Tokyo, November 7, 2019
6. Hiroki Ohtsuji, Erika Hayashi, Naoto Fukumoto, Eiji Yoshida, Takuya Okamoto, Takeru Kuramoto, Osamu Tatebe, “Mitigating the Impact of Tail Latency of Storage Systems on Scalable Deep Learning”, 4th International Parallel Data Systems Workshop (PDSW 2019), Work-in-Progress session, Denver, November 18, 2019
7. Kazuhiro Serizawa, Osamu Tatebe, "Accelerating Machine Learning I/O by overlapping data staging and mini-batch generations", 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, New Zealand, December 3, 2019

8. Takuto Sato, Osamu Tatebe, Hiroyuki Kusaka, "In-situ data analysis system for high resolution meteorological large eddy simulation model", 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, Auckland, New Zealand, December 5, 2019
9. Kenta Machida, Osamu Tatebe, "GHOSTZ PW/GF: Distributed Parallel Homology Search System for Large-scale Metagenomic Analysis", Third IEEE International Workshop on Benchmarking, Performance Tuning and Optimization for Big Data Applications (BPOD 2019), Los Angeles, December 11, 2019
10. Hiroto Tadano, Performance evaluation of the Block GWBiCGSTAB method with dynamic grouping strategy, International Conference on Simulation Technology (JSST2019), Miyazaki, Japan, Nov. 2019.
11. Ayumu Saitoh, Yuji Hirabuki, Hiroto Tadano, Teruou Takayama, Atsushi Kamitani, Volumetric 3D reconstruction based on tomographic image data: acceleration by Global ICCG method, International Conference on Simulation Technology (JSST2019), Miyazaki, Japan, Nov. 2019.
12. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Ayumi Nakamichi, Taisuke Boku: OpenCL-enabled GPU-FPGA Accelerated Computing with Inter-FPGA Communication, IXPUG Workshop at HPC Asia 2020 2020 年 1 月 17 日
13. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Ayumi Nakamichi and Taisuke Boku: GPU-FPGA Heterogeneous Computing with OpenCL-enabled Direct Memory Access, The Ninth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES), May 20th, 2019
14. Norihisa Fujita, Ryohei Kobayashi, Yoshiki Yamaguchi, Taisuke Boku, Parallel Processing on FPGA Combining Computation and Communication in OpenCL Programming, Proceedings of 2019 IEEE International Parallel and Distributed Processing Symposium Workshops, The Ninth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES), May 20th, 2019.

(3) 国内学会・研究会発表

A) 招待講演

1. 朴泰祐, FPGA を中心とした新しい演算加速クラスタソリューション, Intel FPGA Technology Day, Tokyo, 2019 年 11 月 26 日

B) その他の発表

1. 中道 安祐未, 藤田 典久, 小林 諒平, 朴 泰祐, 吉川 耕司, 梅村 雅之, GPU・FPGA 複合演算加速による輻射流体シミュレーションコード ARGOT の実装, 研究報告ハイパフォーマンスコンピューティング（HPC） 2019-HPC-170(22) 1 - 5 2019 年 7 月
2. 辻大亮, 多田野寛人, 朴泰祐, 池田亮作, 佐藤拓人, 日下博幸, "都市気象コード City-LES の並列 GPU 実装の最適化と性能評価", 情報処理学会研究報告ハイパフォーマンスコンピューティング(HPC), 2019-HPC-170(5), pp.1-9, 2019 年 7 月.
3. 辻大亮, 朴泰祐, 池田亮作, 佐藤拓人, 多田野寛人, 日下博幸, "都市気象コード City-LES の OpenACC による並列 GPU 実装とデータ転送最適化", 情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC）, 2020-HPC-173(22), pp.1-8, 2020 年 3 月.
4. 綱島 隆太, 小林 諒平, 藤田 典久, 中道 安祐未, 朴 泰祐, " GPU-FPGA 協調プログラミングを実現するコンパイラの開発", 情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC）, 2019-HPC-172(11), pp.1-10, 2019 年 12 月.
5. 渡部裕, 李珍泌, 佐野健太郎, 朴泰祐, 佐藤三久, "ストリーム計算による最適化を併用する FPGA 向け OpenMP コンパイラの試作", 情報処理学会研究報告ハイパフォーマンスコンピューティング(HPC), 2019-HPC-170(3), pp.1-8, 2019 年 7 月.
6. 綱島隆太, 小林諒平, 藤田典久, 中道安祐未, 朴泰祐, "GPU-FPGA 協調計算を記述するためのプログラミング環境に関する研究", 情報処理学会研究報告ハイパフォーマンスコンピューティング(HPC), 2019-HPC-169(10), pp.1-9, 2019 年 5 月.
7. 渡部裕, 李珍泌, 朴泰祐, 佐藤三久, "演算加速機構を対象とするタスク並列ランタイムにおけるデータ移動最適化", 情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC）, 2020-HPC-173(11), pp.1-9, 2020 年 3 月.
8. 枝松拓弥，高橋大介，“AVX-512IFMA を用いた多倍長整数乗算の高速化”，日本応用数理学会 2019 年度年会，東京，2019 年 9 月 5 日.
9. 高橋大介，“Xeon Phi クラスタにおける二次元分割を用いた並列三次元実数 FFT の実現と評価”，日本応用数理学会 2019 年度年会，東京，2019 年 9 月 3 日.
10. 芹沢 和洋, 建部 修見，大規模機械学習訓練における I/O 性能の高速化，情報処理学会 HPC 研究会，北海道，2019 年 7 月 24 日
11. 北澤 昂大, 建部 修見，Gfarm とクラウドストレージによる階層型ストレージシステムの研究，情報処理学会 HPC 研究会，北海道，2019 年 7 月 25 日
12. 町田 健太, 建部 修見，大規模ゲノムデータに対する分散並列相同性検索システムの提案，情報処理学会 HPC 研究会，北海道，2019 年 7 月 26 日
13. 高橋 宗史，建部 修見，分散オブジェクトストレージ Ceph のための Spark ストレージアダプタの設計，情報処理学会 HPC 研究会，東京，2019 年 9 月 20 日

14. 畑中 智之，建部 修見，ユーザ権限における Docker オーケストレーションの構築と撤去，情報処理学会 HPC 研究会，東京，2019 年 9 月 20 日
15. 建部修見, OFP のファイルシステムと高速ファイルキャッシュシステム, 第 3 回 OFP 利活用報告会「OFP 徹底利活用入門」，千葉，2019 年 10 月 11 日
16. 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，東京，2019 年 10 月 25 日
17. 町田 健太, 建部 修見, 分散並列相同性検索システム GHOSTZ PW/GF の大規模環境における評価, 第 172 回 HPC 研究発表会, 沖縄，2019 年 12 月 18 日
18. 澤田 一樹, 建部 修見, 分散バーストバッファが持つ I/O の性能評価, 第 172 回 HPC 研究発表会, 沖縄，2019 年 12 月 19 日
19. 建部修見，Gfarm ファイルシステムの最新機能，Gfarn ワークショップ 2020，宮崎，2020 年 2 月 7 日
20. 多田野 寛人, Block GWBiCGSTAB 法における漸化式動的グループ化の性能評価, 2019 年並列／分散／協調処理に関する『北見』サマー・ワークショップ（SWoPP2019），北見市民会館, 2019 年 7 月.
21. 倉本 亮世, 多田野 寛人, ブロック積型反復解法の近似解精度劣化の原因解析と高精度化, 日本応用数理学会 2019 年度年会, 東京大学駒場 I キャンパス, 2019 年 9 月.
22. 多田野 寛人, 漸化式動的グループ化による Block GWBiCGSTAB 法の収束性・近似解精度改善, 日本応用数理学会 2019 年度年会, 東京大学駒場 I キャンパス, 2019 年 9 月.
23. 多田野 寛人, Saddle Point Problem に関する一考察と高速解法への展望, 【非線形問題の解法と可視化に関する研究会】2019 年度第 1 回研究会, 自然科学研究機構核融合科学研究所, 2019 年 9 月.
24. 小林 諒平, 藤田 典久, 中道 安祐未, 山口 佳樹, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村, 雅之, GPU･FPGA 複合演算加速による宇宙輻射輸送コード ARGOT 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） 2020-HPC-173(8) 1 - 11 2020 年 3 月
25. 小林 諒平, 藤田 典久, 中道 安祐未, 山口 佳樹, 朴 泰祐, 吉川 耕司, 安部 牧人, 梅村, 雅之, OpenCL 対応 GPU・FPGA デバイス間連携機構による宇宙輻射輸送コードの演算加速, 研究報告ハイパフォーマンスコンピューティング（HPC） 2019-HPC-172(8) 1 - 9 2019 年 12 月
26. 小林 諒平, 藤田 典久, 山口 佳樹, 中道 安祐未, 朴 泰祐, OpenCL 対応 FPGA 間通信機能による GPU・FPGA 複合型演算加速, 研究報告ハイパフォーマンスコンピューティング（HPC） 2019-HPC-170(5) 1 - 9 2019 年 7 月

27. 藤田 典久, 小林 諒平, 山口 佳樹, 上野 知洋, 佐野 健太郎, 朴 泰祐, スーパーコンピュータ Cygnus 上における FPGA 間パイプライン通信の性能評価, 研究報告ハイパフォーマンスコンピューティング (HPC), 2020-HPC-173, pp.1-11, 2020.
28. 藤田 典久, 小林 諒平, 山口 佳樹, 朴 泰祐, 再構成可能なハードウェアを用いた演算と通信を融合する手法の提案と性能評価, 研究報告ハイパフォーマンスコンピューティング (HPC), 2020-HPC-171, pp.1-9, 2019.

(4) 著書，解説記事等

1. Daisuke Takahashi, “Fast Fourier Transform Algorithms for Parallel Computers”, Springer (2019).
2. Daisuke Takahashi, “Fast Fourier Transform in Large-Scale Systems”, Masaaki Geshi (Ed.), The Art of High Performance Computing for Computational Science, Vol. 1, Springer, pp. 137-168 (2019).

7. 異分野間連携・産学官連携・国際連携・国際活動等

異分野間連携（センター内外）

- 量子物性研究部門と光電子相互作用における第一原理計算のアプリケーション開発と「富岳」を始めとする各種スーパーコンピュータへの実装に関する共同研究を実施
- 地球環境研究部門と City-LES コードの GPU 化，及び大規模データ解析に関する共同研究を実施
- 素粒子物理研究部門と Japan Lattice Data Grid（JLDG）の構築、運用に関して連携
- 計算情報学研究部門（データ基盤分野）とデータベース処理の FPGA 化に関する共同研究を実施

産学官連携

国際連携・国際活動

- 米国アルゴンヌ国立研究所と FPGA の並列処理応用と通信技術，及びストレージシステムに関する共同研究を実施
- 米国オークリッジ国立研究所と FPGA 向け OpenACC 複合コンパイラ，及びバーストバッファシステムに関する共同研究を実施
- フランス CNRS 及びドイツ・アーヘン工科大学とエラーフリーMPI を用いた PGAS 言語実装に関する共同研究を実施（SPPEXA）

8. シンポジウム，研究会，スクール等の開催実績

- 国際会議 ICPP2019 (48th International Conference on Parallel Processing), Kyoto, Aug. 5th-8th, 2019
- XcalableMP ワークショップ，東京, 2019 年 11 月 5 日
- ISC2019 (2019 International Supercomputing Conference, Frankfurt, 16th-20th Jun. 2019) における Workshop "Using FPGAs to Accelerating HPC and Data Analytics on Intel-Based Systems"
- SC19 (2019 International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, 17th-22nd Nov. 2019) における BoF "Reconfigurable/FPGA Clusters for High Performance Computing"
- SC19 (2019 International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, 17th-22nd Nov. 2019) における Panel Discussion "Reconfigurable Computing in HPC: Success Stories Today and Future?"
- Gfarm シンポジウム 2019，東京，2019 年 10 月 25 日
- Gfarm ワークショップ 2020，宮崎，2020 年 2 月 7 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：筑波大学情報環境委員会委員
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：理化学研究所客員主管研究員
4. 朴泰祐：HPCI 連携サービス委員会委員長
5. 朴泰祐：HPCI コンソーシアム理事
6. 朴泰祐：PC クラスタコンソーシアム理事
7. 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
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 共用ストレージ運用部会副部会長
14. 建部修見：HPCI 連携サービス運営・作業部会委員
15. 建部修見：理化学研究所客員主管研究員

16. 建部修見：情報通信研究機構協力研究員
17. 建部修見：HPCI 利用研究課題審査委員会レビューアー
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

10. 社会貢献・国際貢献

1. Taisuke Boku: Steering Committee Chair, International Conference on High Performance Computing in Asia-Pacific Region (HPCAsia)
2. Taisuke Boku: General Chair, 48th International Conference on Parallel Processing (ICPP2020)
3. Taisuke Boku: Steering Committee Member, Intel eXtreme Performance Users Group (IXPUG)
4. Taisuke Boku: Scientific Committee Member, Supercomputing Frontier Europe 2020
5. Taisuke Boku: Organizing Chair, IXPUG Workshop HPCAsia 2020
6. Taisuke Boku: Committee Member, SC19 Panel Committee
7. Taisuke Boku: Organizing Committee Member, IXPUG Workshop ISC 2019
8. Taisuke Boku: Program Committee Member, SC19
9. Daisuke Takahashi: The 14th International Workshop on Automatic Performance Tuning (iWAPT 2019) Program Committee Member
10. Daisuke Takahashi: The International Conference on Computational Science (ICCS 2019) Program Committee Member
11. Daisuke Takahashi: ISC High Performance 2019 (ISC 2019) Research Poster Committee Member
12. Daisuke Takahashi: Benchmarking in the Data Center in Conjunction with International Conference on High Performance Computing in Asia Pacific Region (HPC Asia 2019) Organizing Committee Member
13. Daisuke Takahashi: IEEE 13th International Symposium on Embedded Multicore SoCs (MCSoC-19) Program Committee Member
14. Daisuke Takahashi: The 19th International Conference on Computational Science and Its Applications (ICCSA 2019) Publicity Committee Member
15. Daisuke Takahashi: Special Session on Auto-Tuning for Multicore and GPU (ATMG) in Conjunction with IEEE 13th International Symposium on Embedded Multicore SoCs (MCSoC-19) Program Committee Member
16. Daisuke Takahashi: The 4th International Workshop on GPU Computing and AI (GCA'19) in Conjunction with 7th International Symposium on Computing and Networking (CANDAR'19) Program Committee Member

17. Daisuke Takahashi: The 7th International Workshop on Computer Systems and Architectures (CSA'19) in Conjunction with 7th International Symposium on Computing and Networking (CANDAR'19) Program Committee Member
18. Daisuke Takahashi: The 17th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2019) Program Committee Member
19. 高橋大介:電子情報通信学会／情報処理学会 第 18 回情報科学技術フォーラム（FIT2019）担当委員
20. 高橋大介：情報処理学会 2019 年度論文賞選定ワーキンググループ（ジャーナル）委員
21. 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
22. 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
23. Osamu Tatebe: IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC19)
24. Osamu Tatebe: Program Committee, IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2019)
25. Osamu Tatebe: Program Committee, IEEE International Conference on Cluster Computing (Cluster 2019)
26. Osamu Tatebe: Program Committee, International Supercomputing Conference 2019
27. Osamu Tatebe: Program Committee, 4th International Parallel Data Systems Workshop (PDSW 2019)
28. 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委員長
29. Hiroto Tadano: The 38th JSST Annual International Conference on Simulation Technology (JSST2019): Publication Co-Chair
30. Hiroto Tadano: 48th International Conference on Parallel Processing (ICPP2019): Deputy Chair
31. 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 運営委員
32. 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 運営委員
33. 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 編集委員
34. 多田野寛人：日本応用数理学会 2019 年度研究部会連合発表会 実行委員
35. Ryohei Kobayashi: Program Committee, International Conference on Field-Programmable Technology (FPT'19)
36. Ryohei Kobayashi: Program Committee, 7th International Workshop on Computer Systems and Architectures (CSA'19) held in conjunction with CANDAR'19
37. Ryohei Kobayashi: Publicity Chair, The 48th International Conference on Parallel Processing (ICPP 2019)

38. Ryohei Kobayashi: Program Committee, IEEE Symposium on Low-Power and High-Speed Chips and Systems (COOL Chips 22)
39. 小林諒平: xSIG 2019 プログラム委員
40. 小林諒平: SWoPP2019 実行委員 (コンピュータシステム研究会担当幹事)
41. 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
42. 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

1. 藤田 典久：2020/02/24-2020/03/07, 米国 Argonne National Laboratory へのマンスリーサバティカルを実施.