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，額田 彰， 塙 敏博（客員教授，東京大学）
助教	多田野 寛人，小林 諒平，藤田 典久
研究員	平賀 弘平
学生	大学院生　14 名、学類生　7 名、研究生　2 名
学内共同研究員	
	安永 守利，櫻井 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（国立情報学研究所）， 松岡 聡（理化学研究所），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）

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，ストレージシステム，並列入出力，分散システムソフトウェアなど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を主に行った。

- GPU 及び FPGA の協調計算に関する研究
- FPGA 向け OpenACC コンパイラの最適化に関する研究
- メニーコアプロセッサにおける数値計算ライブラリの開発及び性能評価
- ノードローカルストレージを用いたシステムソフトウェアの研究開発
- Gfarm ファイルシステムの研究開発
- GPU におけるデバイス間接続ネットワークの有効活用に関する研究
- 鞍点型連立一次方程式に対する階層並列型数値解法の並列実装と性能評価，及び 3 次元画像再構成問題への応用
- 統一的なプログラミングモデルによる GPU-FPGA 連携の実現のための実践的検討
- FPGA 向け実アプリケーションの開発に資するライブラリの開発

3. 研究成果

[1] OpenACC による GPU・FPGA 協調プログラミング（朴，小林，藤田）

我々の研究チームで近年注力しているテーマが GPU と FPGA というマルチ演算加速デバイスを用いた協調計算であり，このコンセプトを CHARM (Cooperative Heterogeneous Acceleration with Reconfigurable Multidevices)と呼んでいる。計算科学研究センターで運用中の Cygnus スーパーコンピュータはその一部が CHARM コンセプトの実証実験に充てられているように，センターとして重要な研究テーマと捉え，アプリケーション分野との共同研究も活発に行なっている。

研究の一環として，OpenACC による GPU・FPGA 複合プログラミング環境の研究を行っているが，本テーマは理化学研究所計算科学研究センター（R-CCS）及び米国 Oak Ridge National Laboratory（ORNL）との共同研究により進めている。

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。CHARM コンセプトに基づく OpenACC によるプログラミングを行えるような処理系を開発し，アプリケーションユーザでも容易にプログラム開発が行え，実アプリケーションの高速化を実現することが本研究の目的である。

CHARM コンセプトに基づくプログラミング環境を CAMP (Cooperative Accelerating by Multidevice Programming)と呼ぶ。従来の CAMP の一つの実装は，GPU については CUDA で，FPGA については OpenCL で個別にプログラミングし，それらを連結可能とすることで最終バイナリコードを実現したり，これを Intel oneAPI 環境で実装するような研究を行ってきたが，これをよりスマートな単一プラットフォーム，すなわち OpenACC だけで記述可能にする。このため，OpenACC の directive を拡張し，演算をオフロードする際のターゲットデバイス（GPU または FPGA）を指定可能とし，メタコンパイラによってコードを分解した上で指定されたデバイスのためのバックエンドコンパイラに与え，最終的な部分コードをリンクして実行バイナリを生成するというアプローチを取る。このメタコンパイラを MHOAT (Multi-Hetero OpenACC Translater)と呼んでいる。

MHOAT から呼ばれる GPU 及び FPGA 向けの OpenACC バックエンドコンパイラには PGI GPU Compiler と OpenARC をそれぞれ用いる。OpenARC は ORNL が開発を進めている研究用コンパイラである。また，MHOAT のパーザ部分と変換されたソースコードの出力には我々と R-CCS で共同開発している Omni Compiler フレームワークを用いる。図に MHOAT の処理の概念図を示す。

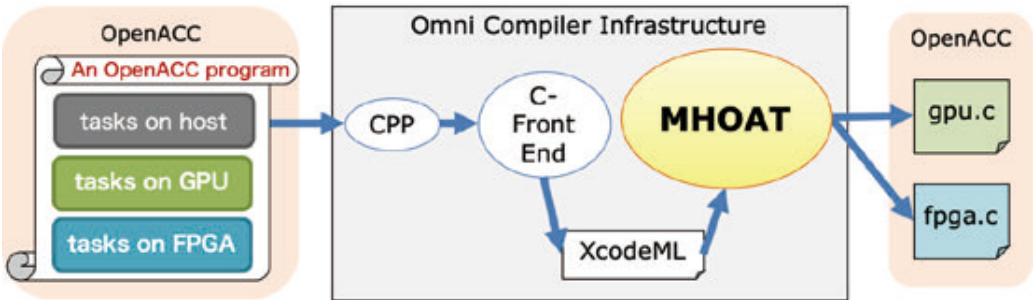


図 1. MHOAT の処理フロー

令和 4 年度の研究では，本センターの宇宙物理学研究部門・梅村教授のグループと共同で開発している宇宙初期天体シミュレーションコード ARGOT を MHOAT のターゲットコードとし，一部手作業によるコード最適化を行い，CHARM コンセプトによる実アプリケーションの GPU・FPGA 協調計算の有効性を示した。同コードの計算中核部分を成す 2 種類の演算である ARGOT 法と ART 法について，それぞれ GPU 及び FPGA で機能分散的に使用する。ARGOT 法の計算は基本的に 3 次元空間での重力ツリーコード計算に類似しており，GPU だけで十分高速化が可能である。一方，ART 法はメモリアクセスがランダムになり，ベクトル演算のベクトル長が短くなる等，GPU に不向きな計算であり，実際に小規模問題を対象とした場合，CPU と比較してほとんど加速できないという問題があったが，FPGA ではこれを十分高速化できることがわかっている。

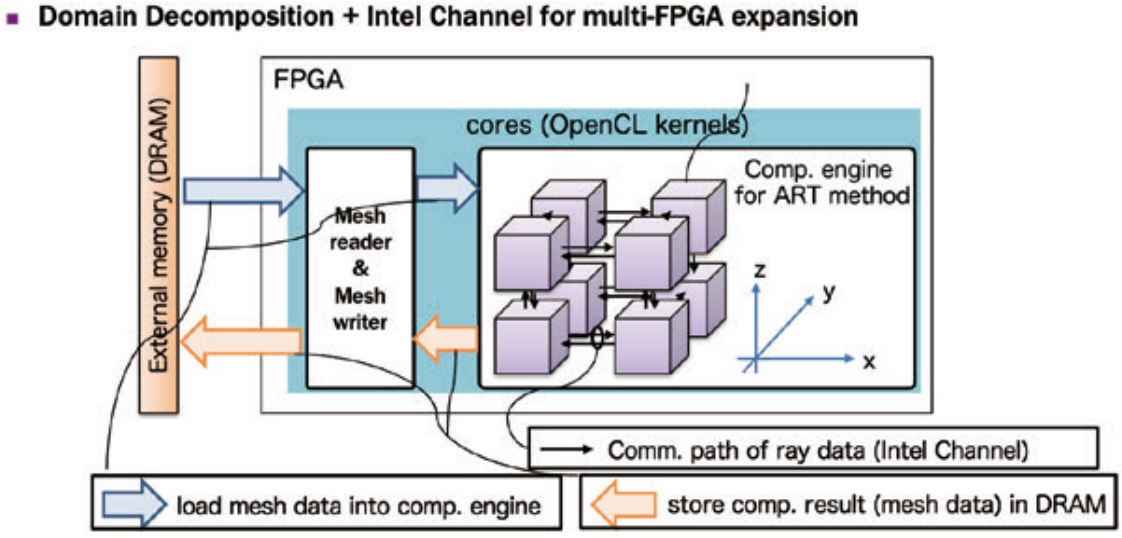


図 2. 複数カーネルによる ART 法の実装イメージ

MHOAT 向けに OpenACC 記述された ARGOT コードは，特に FPGA 向けに特別な最適化を施している。まず，FPGA 内の演算回路を最大限に活かすために，空間分割によって FPGA

内で並列に動作するセル構造を構築し，これを複数のカーネルに振り分ける（今回の実装では 8 カーネル）ことで，空間並列性を向上させる。これらのカーネル間は MPI 並列のような通信が必要となるが，Intel FPGA 向けの OpenCL 処理系に実装されている channel 機能を用いることでこれを効率的に実装している。OpenARC は Intel FPGA 向けに実装されており，この channel 機能を OpenACC の拡張機能として用いることができる。マルチカーネル化した ARGOT コードのイメージを図に示す。ただし，現在の OpenARC は channel の標準的なデフォルト設定のみをサポートしているため，この部分は MHOAT によるトランスレーションの後で部分的なパッチを当て，問題を回避している。また，FPGA 上の reduction 演算でどうしてもパイプラインピッチが延びてしまう部分があり，こちらについては一部の関数を Verilog HDL で記述している。MHOAT はこのように部分的に低レベル言語記述されたコードも取り扱うことができる。

これらの最適化の結果，MHOAT による ARGOT コード実装は，i) GPU のみによる CUDA 実装，ii) GPU のみによる OpenACC 実装，iii) GPU (CUDA) と FPGA (OpenCL)によるナイーブな実装，i) GPU と FPGA を OpenACC で統一記述した MHOAT 実装，の中で最速であった。特に，GPU 部分を OpenACC 実行したことで CUDA 実装よりも ARGOT 法の効率が良くなったことに加え，ART 法の OpenACC (MHOAT+OpenARC)による FPGA 実装が，従来の OpenCL による実装とほぼ同じ速度で実行できたことが高速化につながった。図 3 にこれらの速度比較を示す（問題サイズ: 32x32x32，GPU=NVIDIA V100, FPGA=Intel Stratix10）。

この結果により, MHOAT が OpenACC のみで実アプリケーションが記述でき, かつ CHARM コンセプトに基づく適材適所的な GPU と FPGA の割当により GPU だけの場合に比べ極めて高性能（最大で 9.9 倍）を実現できることがわかった。

これらの研究成果は国際会議 International Workshop on HPC by Heterogeneous Hardware (H3) in ISC2023 で発表予定である。

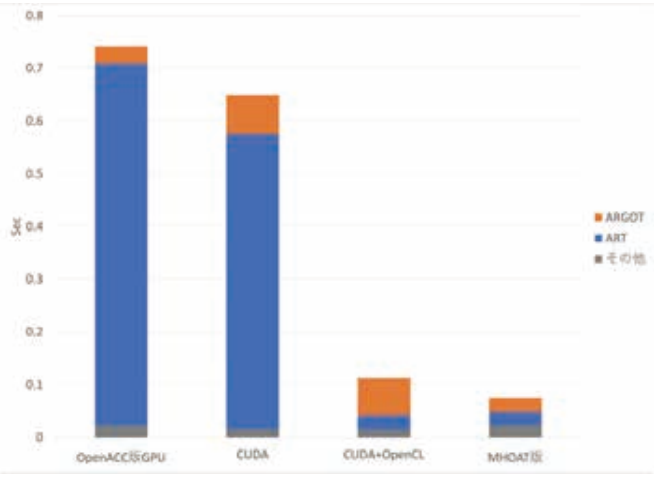


図 3. MHOAT による ARGOT コードの GPU+FPGA 実装の性能評価

[2] FPGA 向け OpenACC コンパイラのための性能最適化

GPU・FPGA により協調計算を進めるに当たり、MHOAT では米国 ORNL で開発中の研究用コンパイラ OpenARC を FPGA 向け OpenACC コンパイラとしてバックエンドに使用しているが、今後の最適化やノード間並列化、さらに藤田らが中心に開発している FPGA 間通信システム CIRCUS 等への対応を容易にするために、我々独自の FPGA をターゲットにした OpenACC コンパイラの開発を進めている。これは理化学研究所計算科学研究センターとの共同研究で、ベースとなるプラットフォームとして、これまで共同開発してきた Omni Compiler のツールキットを利用する。また、Omni Compiler には過去に OpenACC と GPU 向け OpenCL に変換する実装が行われているため、これを利用して FPGA 向け実装を進めている。

令和 4 年度の研究では主として科学技術計算に出てくる典型的なベクトル処理（行列積、ベクトル内積、リダクション等）を FPGA の高位合成によって記述・コンパイルする際の最適化について予備評価した。Omni Compiler によって OpenACC コードを OpenCL 化するので、その出力すべき OpenCL コードのイメージを Intel OpenCL SDK for FPGA でコンパイルし、実機測定して評価した。

FPGA は基本的にループ処理をパイプライン化するため、単一ループを処理しただけではパイプラインが 1 つ生成されるだけで、クロック周期によって性能が頭打ちになってしまう。そこで、複数のパイプラインの並列動作、すなわち空間並列性の増強によって性能を向上させる。単純なループであれば、unrolling を行うことで 1 ループ内に複数のデータ並列を記述すればその数だけパイプラインが生成できる。しかし、FPGA の外付け DRAM は GPU 搭載の HBM 等に比べバンド幅が低いため、無闇にパイプラインを増やしてもデータ転送がボトルネックになる。そこで、FPGA チップ内に回路として実装される SRAM（Intel 用語では Block RAM=BRAM）を addressable cache として用い、バンド幅を増強する手法を試した。しかし、Intel OpenCL コンパイラでは、BRAM のバンド幅を増強するため、一定以上の unroll をかけると BRAM が複製され、そこにデータが分配されるという最適化が自動的に行われることが判明した。このため、unroll 数が増えると性能は向上するが BRAM 容量が圧迫されるというトレードオフが存在する。典型的なベクトル加算処理において、ループの unroll 数と BRAM 容量、さらに回路合成時の動作周波数のパラメータサーチを行い、最適な組み合わせを検討した。

単純なループ unrolling だけでは空間並列性の限界があり、BRAM を圧迫するだけでなく、動作周波数も低下する傾向にあることがわかったので、さらにマルチカーネルによる空間並列性の向上も検討した。複数の演算カーネルを同一 FPGA 上に並べ、非同期に並行動作させることで空間並列性を向上させる。これは、MPI による並列プロセス実行に似ており、データの共有も FPGA カーネル間で行う。Intel OpenCL SDK for FPGA は channel によるカーネル

間通信が提供されるため、カーネル間通信にこれを用いることで分散されたデータの共有時のオーバヘッドが低減できる。

最終的にカーネル内のループ unrolling とマルチカーネルによる空間並列性の向上により、演算性能がどの程度高まり、BRAM 容量や動作周波数とのトレードオフがどうなるかについて総合的に評価するため、CG 法による疎行列連立一次方程式解法をベンチマークとして評価した。結果を図に示す。行列サイズは 3000x3000、非零要素率は 4%で、FPGA ボードは Cygnus に搭載されているのと同じ Intel Stratix10 搭載の BittWare 520N を用いた。

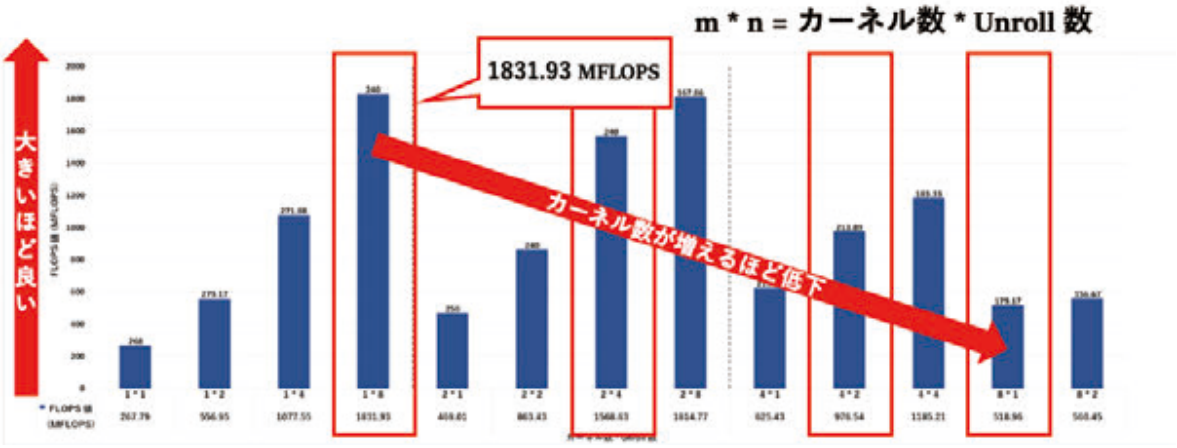


図 4. ループ unrolling とマルチカーネルの併用による CG 法ベンチマーク性能評価

この評価では、マルチカーネルによる空間並列性向上は相対的に小さく、ループ unrolling による効果はカーネル数が増えると減少するという結果になった。ループ unrolling はカーネル数が 1 の時に最も効果的で、unroll 数にほぼ比例して性能が向上した。丸カーネルに効果がなかった理由の一つは、カーネル数の増加に伴い動作周波数が大きく低下したことにある。一方、unroll 数を増やした時に BRAM 使用率が大きく増えてしまうことは確認された。そこで、トレードオフの関係の指標として、BRAM 使用率 1%当たりの性能を比較してみた。結果を図に示す。結果から、やはりマルチカーネルは BRAM 使用率を抑えるためには効果的であり、BRAM 使用率当たりの最高性能はカーネル数 2、unroll 数 4 の場合に得られた。今後の OpenACC からの OpenCL カーネル生成において、これらの知見を自動最適化に組み入れる検討を進める。また、CG 法のような典型的処理だけでなく、より複雑な演算における空間並列性向上手法についても検討を進める予定である。

これらの研究結果は情報処理学会 HPC 研究会 2023-HPC-188(22)において発表された。

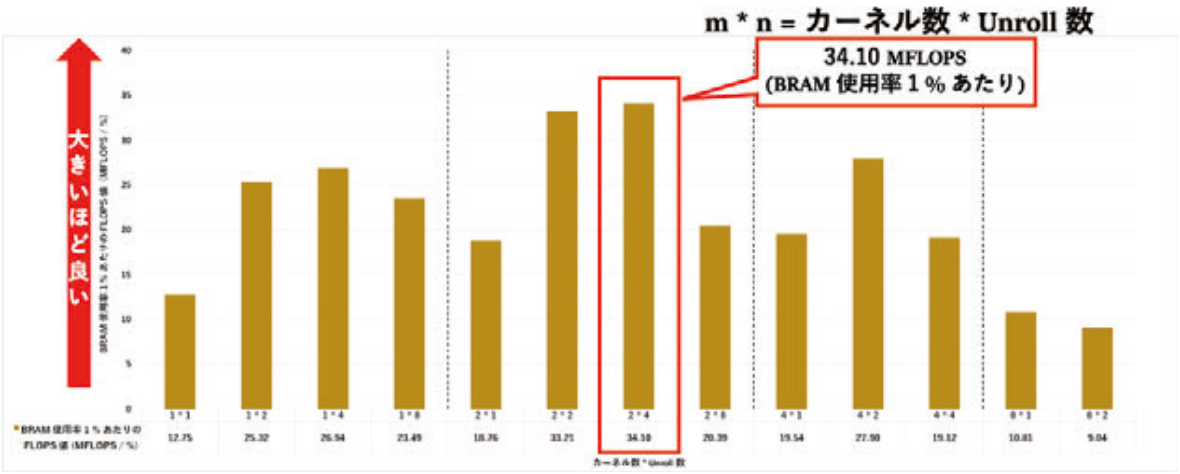


図 5. マルチカーネルとループ unroll の併用時の BRAM 使用率 1%当たりの性能

[3] 数論変換の並列化（高橋）

高速 Fourier 変換（fast Fourier transform、以下 FFT）は、現在科学技術計算において広く用いられているアルゴリズムである。FFT は複素数や実数を用いて計算されることが多いが、これらの変換は環や体でも計算できることが知られている。このような変換は、数論変換（number-theoretic transform、以下 NTT）と呼ばれる。NTT は準同型暗号、多項式乗算、多倍長精度乗算などに利用されている。本研究では、Arm Scalable Vector Extension（SVE）命令を用いて NTT カーネルをベクトル化すると共に、MPI と OpenMP を用いて NTT を並列化して性能評価を行った。

n点 NTT は$\mathbf{F}_p = \mathbf{Z}/p\mathbf{Z}$（$p$は素数）において以下のように表すことができる。

$$y(k) = \sum_{j=0}^{n-1} x(j)\omega_n^{jk} \bmod p, \qquad 0 \le k \le n-1$$

ここで、ω_nは 1 の原始n乗根である。

n点 NTT を直接計算すると$O(n^2)$の演算回数を必要とするが、FFT と同様のアルゴリズムを適用することで、演算回数を$O(n \log n)$に削減することが可能である。Out-of-place FFT アルゴリズムとして知られる Stockham FFT アルゴリズムを基数 2 の NTT に適用したアルゴリズムを図に示す。

Algorithm 1 Stockham radix-2 NTT algorithm

Input: $n = 2^q$, $X_0(j) = x(j)$, $0 \le j \le n-1$, and ω_n is the primitive n-th root of unity.

Output: $y(k) = X_q(k) = \sum_{j=0}^{n-1} x(j)\omega_n^{jk} \bmod p$, $0 \le k \le n-1$

1: $l \leftarrow n/2$
2: $m \leftarrow 1$
3: **for** t **from** 1 **to** q **do**
4: **for** j **from** 0 **to** $l-1$ **do**
5: **for** k **from** 0 **to** $m-1$ **do**
6: $c_0 \leftarrow X_{t-1}(k+jm)$
7: $c_1 \leftarrow X_{t-1}(k+jm+lm)$
8: $X_t(k+2jm) \leftarrow (c_0+c_1) \bmod p$
9: $X_t(k+2jm+m) \leftarrow \omega_n^{jm}(c_0-c_1) \bmod p$
10: **end for**
11: **end for**
12: $l \leftarrow l/2$
13: $m \leftarrow 2m$
14: **end for**

図 6. 基数 2 の Stockham NTT アルゴリズム

図のアルゴリズムの 8 行目と 9 行目には、剰余加算、剰余減算、剰余乗算が含まれている。$0 \le a, b < N$の場合の剰余加算$c = (a+b) \bmod N$は、加算$c = a+b$と、$c \ge N$の場合の条件付き減算$c-N$に置き換えることができる。剰余乗算は Montgomery 乗算や Shoup 乗算を用いることで時間の掛かる除算を実質的に行うことなく、乗算、加減算およびシフト演算のみで行えることが知られている。Shoup 乗算アルゴリズムを図に示す。図のアルゴリズムにおいて、βが 2 のべき乗であるとき、1 行目のABをβで割った切り捨て商は右シフトにより、2 行目の$(AB-qN)$をβで割った余りはビットマスクにより計算することができる。

Algorithm 2 Shoup's modular multiplication algorithm

Input: A, B, N such that $0 \le A, B < N$, $N < \beta/2$
precomputed $B' = \lfloor B\beta/N \rfloor$

Output: $C = AB \bmod N$

1: $q \leftarrow \lfloor AB'/\beta \rfloor$
2: $C \leftarrow (AB - qN) \bmod \beta$
3: **if** $C \ge N$ **then**
4: $C \leftarrow C - N$
5: **return** C.

図 7. Shoup 乗算アルゴリズム

n点 NTT において$n = n_1 \times n_2$と分解できる場合には、six-step FFT アルゴリズムと呼ばれる手法を NTT に適用することができる。Six-step NTT アルゴリズムは以下のようになる。

Step 1: $n_1 \times n_2$行列を$n_2 \times n_1$行列に転置

Step 2: n_1組のn_2点 multicolumn NTT

Step 3: ひねり係数（$\omega_n^{j_1 k_2}$）の$\mathrm{mod}\ p$での剰余乗算

Step 4: $n_2 \times n_1$行列を$n_1 \times n_2$行列に転置

Step 5: n_2組のn_1点 multicolumn NTT

Step 6: $n_1 \times n_2$行列を$n_2 \times n_1$行列に転置

ここで、Step 1、4 および 6 における行列の転置は MPI の全対全通信（MPI_Alltoall）を用いて実装することができる。また、複数の Shoup 乗算を Arm SVE 命令を用いて SIMD 化を行うとともに、Step 2、5 の multicolumn NTT に対して MPI と OpenMP を用いて並列化を行った。

性能評価として、63 ビットの modulus を持つ six-step NTT の実装の性能を Fujitsu PRIMEHPC FX1000 で測定した。各 MPI プロセスは 12 コア 12 スレッド、つまり 1 ノードあたり 4MPI プロセスを用いている。

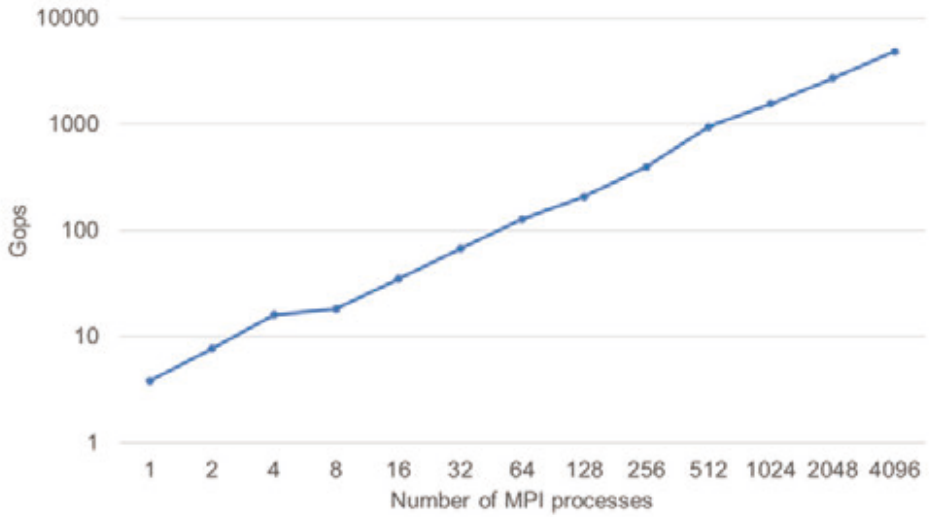


図 8. 並列 NTT の性能（Fujitsu PRIMEHPC FX1000、4 MPI /ノード、$N = 2^{26} \times$MPI プロセス数）

図は、並列 NTT の性能を示している。MPI プロセス数を 4 から 8 に増やしたときの性能向上が小さいのは、ノード内で 4 つの MPI プロセスが通信しているからである。4096 MPI プロセスで2^{38}点 NTT を計算した場合、実行時間の約 80%が全対全通信で占められている。Fujitsu PRIME HPC FX1000 は、6 次元トーラスネットワークである Tofu interconnect D を採用しているが、ノード数が増えると最大ホップ数も増え、全対全通信の通信バンド幅が小さくなる。性能評価の結果、提案する並列 NTT の実装は Fujitsu PRIMEHPC FX1000（1024 ノード、4096 MPI プロセス）において 4831 Gops の性能を示した。

[4] キャッシングファイルシステムの研究（建部）

これまでスーパコンピュータの計算ノードのローカルストレージシステムを活用したアドホック並列分散ファイルシステム CHFS の設計を行ってきた。CHFS では，計算ノードのローカルストレージを用いて高並列な分散キーバリューストアを構築し，その上に高並列なファイルシステムの設計を行った。設計にあたり，ノード数に対するスケーラビリティを阻害する要因となる集中データ構造や逐次処理を避けた。Cygnus を用いて性能評価を行ったところ, ノード数を増加させると I/O バンド幅およびメタデータ性能が向上することが分かった。また，BeeOND と GekkoFS との性能比較を行った。BeeOND ではノード数を増やしてもメタデータ性能はあまり向上しなかった。GekkoFS についてはノード数を増やすと性能は向上するものの，CHFS のほうがメタデータ性能は 5.5 倍高かった。

CHFS は並列ファイルシステムとは別のファイルシステムであるため，利用者が並列ファイルシステムとの間で必要なファイルのコピーを行う必要がある。この操作はステージングと呼ばれるが，しばしばファイルやディレクトリの指定誤りにより問題が起こる。この問題を解決するため，CHFS にキャッシングファイルシステムの機能を追加するための設計を行った。キャッシングファイルシステムは，並列ファイルシステムと同じ名前空間をもち，並列ファイルシステムとの間のファイルコピーはシステムが自動的に行う。しかしながら，これまでキャッシングファイルシステムではメタデータ性能が低くなる問題があった。メタデータは並列ファイルシステムで管理するため，純粋にオーバヘッドが増えるためである。この問題を解決するため，利用者にとって無理のない範囲で並列ファイルシステムとキャッシングファイルシステムの間の一貫性を緩めることを考える。ここで以下の仮定をおいた。

1. 入力ファイルは，ジョブ実行中は変更されない
2. ジョブによるファイル，ディレクトリ作成は必ず成功する
3. 既存ファイルを更新するときにはその前に読込む
4. 並列ファイルシステムへの更新はジョブ終了時までに行われる

これらの仮定はバッチキューイングシステムを利用する場合，暗黙に行われている仮定である。ただし，実行状況等を実時間で知りたい場合はキャッシングファイルシステムを用いず，並列ファイルシステムに直接書き込む必要がある。これらの仮定をおくと，並列ファイルシステムとの一貫性のチェックを大幅に削減することが可能となる。読込については，キャッシングファイルシステムになければ，並列ファイルシステムを参照する必要があるが，すでにキャッシングファイルシステムに存在すれば並列ファイルシステムを参照する必要はない。書込みについては，キャッシングファイルシステムに書込み，並列ファイルシステムを参照する必要はない。これらの仮定のもと，CHFS にキャッシングファイルシステムの機能を追加するため，メタデータにダーティフラグとキャッシュフラグを追加し，キャッシュ機能，フラッシュ機能の設計を行った。

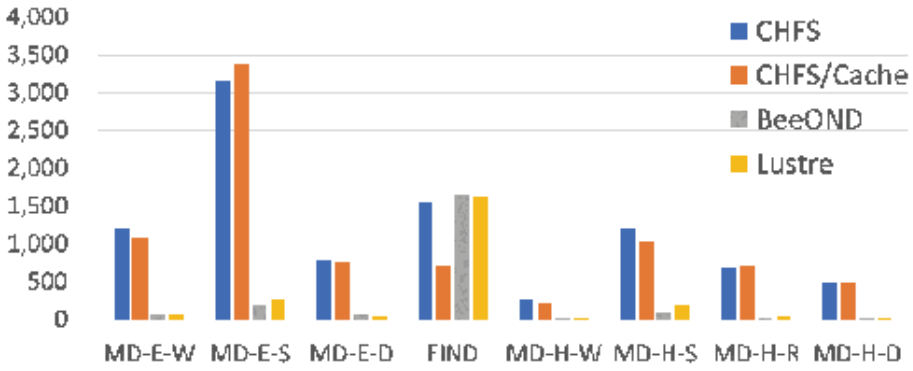


図 9. Cygnus 48 ノードにおけるメタデータ性能

図に Cygnus 48 ノードを用いた時のメタデータ性能の評価結果を示す。MD-E は各プロセスがそれぞれ異なるディレクトリに 0 バイトのファイルを作成，メタデータ取得，消去する操作を表し，グラフは 1 秒あたりの操作数を示す。MD-H は各プロセスが同一のディレクトリに 47.008 バイトのファイルを作成，メタデータ取得，参照，消去する操作である。FIND は作成したファイルのなかから特定のファイルを検索する操作である。評価対象は，キャッシュ機能を持たない CHFS，今回設計したキャッシュ機能を含む CHFS/Cache，キャッシュ機能を持たない BeeOND，並列ファイルシステム Lustre である。まず，CHFS と CHFS/Cache の比較では FIND を除きほとんどメタデータ性能が変わっていないことが分かった。これにより，当初の目的であるキャッシングファイルシステムにおけるメタデータ性能の低下が解決した。また，BeeOND との比較により，CHFS/Cache の性能が高いことが分かった。Lustre との比較により，CHFS/Cache を用いることにより Lustre の性能を大幅に改善可能であることが分かった。これらの成果は，国際ワークショップ ESSA において発表した。今後，さまざまなアプリケーションベンチマークによる評価により研究開発を進め，Pegasus 等のスーパコンピュータに導入する予定である。

[5] 分散ファイルシステムおよびグリッド・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。

HPCI 共用ストレージにおいて，ネットワークトラヒックが高くなるとネットワークの性能が落ち，接続が切断される状況が続いていた。本件については，ネットワークの調査を進めるとともに，Gfarm のパラメータの調整により対応を進めた。具体的には，ネットワークが切断される頻度が高く，3 PB/月のペースで書き込まれるデータの東西拠点間の複製作成が間に合わない状態であった。そのため，並列作成の並列度を 400 から 1,000 に上げる対応を行っ

た。並列度を上げた状態で，状況を観察したところ，ネットワークエラーがさらに増えることはなく，複製作成が間に合うようになった。今回は Gfarm のパラメータ調整により解決したが，引き続きネットワークの調査は必要となる。

また，クラウドストレージのオープンソースプラットフォームとして普及している Nextcloud に対し，外部ストレージとして Gfarm を利用可能とする開発を進めた。これにより Nextcloud の使いやすいインターフェースを用いて，Gfarm のアクセスが可能となる。開発において問題となるのはユーザ証明書をどのように取得するかであるが，ここについては myproxy サーバから証明書を取得することで解決を図った。また，証明書の有効期限が切れた場合は自動的に myproxy サーバから証明書を取得しなおす機能も追加し，myproxy サーバの証明書を更新し続けることにより，Nextcloud から切れ目のないアクセスを続けることが可能となった。本システムは https://github.com/oss-tsukuba/nextcloud-gfarm において公開するとともに，筑波大学のスーパコンピュータ Pegasus システムにも導入され，実運用されている。

GSI のサポート終了に伴う次期セキュリティ対応として，OAuth2 のトークンを用いた認証を行うための設計，開発を進めた。昨年度の開発においては，トークンの有効性検証に Keycloak サーバを利用していたが，その方式ではユーザ数，地理的分散に対しスケーラブルではないため Gfarm サーバにおいて有効性検証を行うよう設計，開発を行った。また，トークンの有効期限は短く，自動更新が必須となる。その仕組みとして jwt-agent の設計を進めた。jwt-agent はオープンソースで開発が進められている oidc-agent と同等の機能を持ち，トークンの更新を行うが，oidc-agent は更新のためリフレッシュトークンも保持するのに対し，jwt-agent ではそのトークンは保持しない。リフレッシュトークンは jwt-server が暗号化した形で保持し，セキュリティリスクを軽減することができる。jwt-agent は https://github.com/oss-tsukuba/jwt-agent において公開している。

[6] DO CONCURRENT による OpenSWPC の GPU 化（額田）

既存の CPU 向けアプリケーションを GPU で高速化するためには基本的に GPU 用にプログラムを修正する必要がある。この修正は単純作業ではなく、ある程度の GPU に関する基礎知識を持っていなければならないが、使用する言語によってその負担は大きく変動する。CUDA、OpenCL、hip のような言語を使用すれば GPU の細粒度並列実行など全ての機能を活用することができ、より広範な計算の高速化が可能となる一方で、プログラムを大幅に修正する必要がある。OpenACC や OpenMP target はよりハードルが低く、既存の CPU 用コードに指示詞（ディレクティブ）を挿入することで GPU 用コードを生成するため、一つのコードで CPU 用と GPU 用の両方に使用できコード管理も容易である。近年新たに追加された方法に DO CONCURRENT というものがある。これは Fortran 2008 で追加された標準並列化構文で、NVIDIA のコンパイラがこのループ部分について GPU コードを生成することができる。コン

パイラオプションの切り替えによって CPU 用と GPU 用の両方のバイナリを生成できることになる。さらに DO CONCURRENT は既存の DO ループに非常に近い構文であるため GPU 用のプログラムを作成するために必要な知識が非常に少ないという利点がある。そこで地震波の伝播シミュレーションコード OpenSWPC を対象に、アプリケーション全体を DO CONCURRENT を用いて GPU 化した。OpenSWPC は OpenMP+MPI のハイブリッド並列化で実装されており、既に多くの CPU ベースのシステムで動作実績がある。OpenMP で並列化されている多くの DO ループの内、GPU で実行させるループについて DO CONCURRENT に書き換える。OpenSWPC の場合は時間ステップに関するメインループの内側が対象となる。

OpenSWPC は３次元シミュレーションであり、各次元方向のループで構成される３重ループが多い。CPU 向けの OpenMP 並列化では一番外側のループのみ並列化を行っている。CPU のコア数はそれほど多くないためこれで十分であるが、GPU の場合はけた違いのコア数となるため３重ループをまとめて並列化する必要がある。

DO CONCURRENT は関数呼び出しについて pure function に限定するという Fortran 2008 の制約があり、さらに GPU 用のコード生成を行う場合には一切の関数呼び出しができないというコンパイラの実装による制限がある。このため関数をインライン展開する必要がある。OpenSWPC には２種類の関数呼び出しがあるが、一方は完全にインライン展開を行い、もう一方はマクロ展開で対応した。この辺りは今後のコンパイラの対応に期待したい。

```
DO CONCURRENT (I=1:N)
    X(I)=V(I)*V(I)
END DO
```

GPU プログラムを煩雑にしている要因の一つは CPU のメモリの他に GPU が専用の高速メモリを搭載していることであり、GPU 側の計算で必要なデータを適宜転送する必要がある。DO CONCURRENT では言語としてこの転送を記載する必要がなく、また指定することも不可能である。上記のような DO CONCURRENT ループではループ内で使用されている変数について必要な転送を行うコードがコンパイラによって生成される。X(1:N)、V(1:N)などの配列が使用されているが、これらが動的に確保された ALLOCATABLE 変数であれば GPU 用コードの場合に Unified Memory という CPU メモリと GPU メモリの間をオンデマンドにページ単位で移動する種類のメモリとして確保される。これは CPU からも GPU からもアクセス可能であり、GPU からのみアクセスを続けることによって GPU メモリ上にある状況を保つことができる。一方で ALLOCATABLE ではない固定長などの配列である場合には CPU メモリとして確保され、DO CONCURRENT ループに入るたびにデータの転送が行われ、この転送がボトルネックとなりやすいため ALLOCATABLE 変数に変更する。

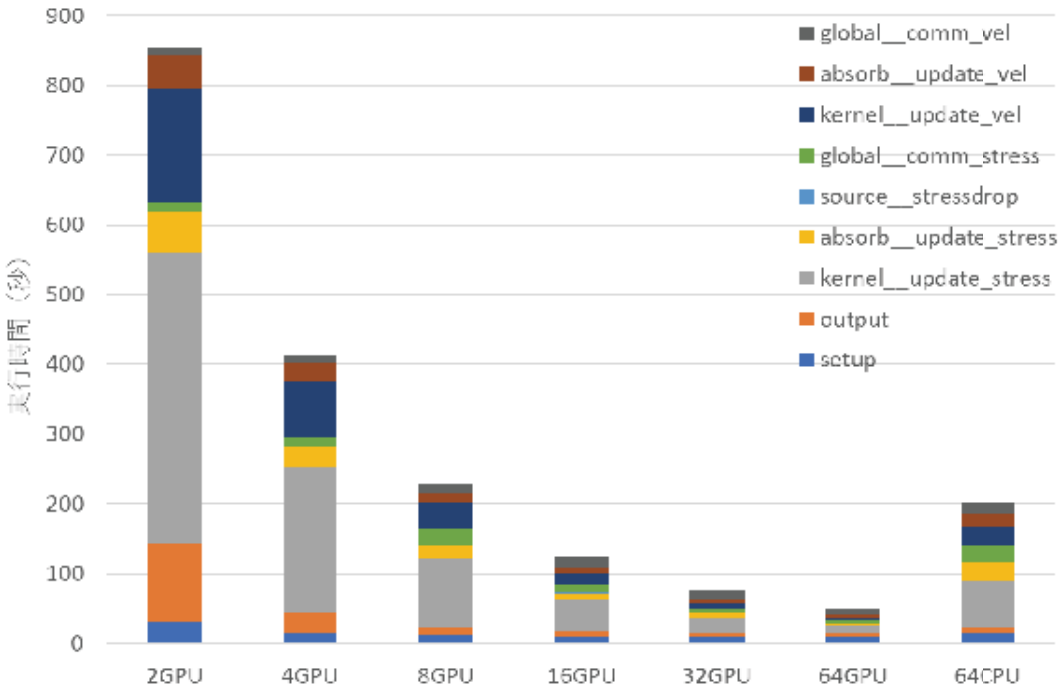
CPU･GPU 間転送を減らすために不可欠な Unified Memory であるが、一方で問題点もある。MPI でノード間データ転送を行う場合にその送受信バッファが Unified Memory の場合に通信性能が低下する。CPU と GPU の間をオンデマンドに移動してしまうメモリ領域であるためネットワークデバイスに転送操作をオフロードさせることができず、一度 CPU メモリ上のバッファを経由するという実装になる。特に GPU 間で直接通信する GPUDirect RDMA と比較すると通信性能の低下は著しい。

```
real,device,allocatable :: sbuf(:)

!$acc data deviceptr(sbuf)
do concurrent (k=kbeg:kend)
     sbuf(k) = vx(k)
end do
!$acc end data

call MPI_Isend(sbuf,…)
```

この問題を回避する方法は現状 DO CONCURRENT の機能だけでは不可能であるため、上記のように CUDA Fortran と OpenACC の機能と利用する。まず通信バッファについて CUDA Fortran の機能で device 属性を付加する。これによって allocate で確保した際に Device Memory だけが確保される。次のループでこの送信用バッファにデータを書き込んでいるが、ここで sbuf がデバイス側のアドレスになっていることを伝えるために OpenACC の data ディレクティブを使用している。MPI_Isend ではそのまま sbuf を渡しているが、CUDA-aware の MPI ライブラリは関数内で自動的にデバイス側のアドレスであることを判別して GPUDirect RDMA を使用する。



これまで述べた最適化を行った GPU コードを用いて性能評価を行った。データは関東エリアを範囲とする小田原地震のシミュレーションを使った Strong Scaling の評価となる。Pegasus の最大 64 ノードを使用し、各ノードは NVIDIA H100 Tensor Core GPU と Intel Xeon Platinum 8468 (Sapphire Rapids, 48 コア)を 1 基ずつ搭載する。kernel_*,absorb_*のメインのステンシル計算部分は GPU 数の増加に従って短くなっている。global_comm_*の通信部分に関してはノード数の増加に従って通信方向が増えるため単調ではないが 16GPU 以上では短くなっていく。比較用に CPU 用コードを 64 ノードで実行した場合を一番右に示しているが、8GPU に近い実行時間となっており、1GPU が約 7CPU に相当する演算性能を達成していると考えられる。

[7] 鞍点型連立一次方程式に対する階層並列型数値解法の GPU による高速化（多田野）

鞍点型と呼ばれる連立一次方程式：

$$\begin{bmatrix} A & B \\ C^{\mathrm{T}} & O \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}$$

は構造解析，非圧縮流体解析，偏微分方程式に対するメッシュレス離散化法などで現れ，求解高速化が必要とされている．ここで，A は正則な n 次疎行列，B，C は $n \times m$ 列フルランク行列，O は m 次零行列であり，$\boldsymbol{x}$，$\boldsymbol{f}$ は n 次元ベクトル，$\boldsymbol{y}$，$\boldsymbol{g}$ は m 次元ベクトルである．鞍点型連立一次方程式の問題サイズが大規模である場合は，クリロフ部分空間反復法などの反復法による求解を余儀なくされる．しかしながら，行列 B，C の列数 m が多い場合は，行列 A が良条件であったとしてもクリロフ部分空間反復法の収束性が悪化し，求解に多くの反復回数を要したり求解不能になることもある．我々はこの状況を打破するために，同方程式のブロ

ック構造を利用した数値解法（以下，提案法）を開発した．提案法における計算の主要部は，行列 A と複数の右辺ベクトルをもつ連立一次方程式の求解である．この連立一次方程式の複数右辺ベクトルは互いに依存関係がないため，少数の右辺ベクトルをもつ複数の連立一次方程式に分割することができる．分割された複数の連立一次方程式は同時求解が可能であり，各連立一次方程式も並列に求解可能であることから，提案法は階層型の並列性をもつ．本研究では，提案法の高速化を図るために GPU を用いた実装を行い，その性能評価を行った．

以下では提案法について簡単に述べる．鞍点型連立一次方程式の解ベクトル $\boldsymbol{x}$, $\boldsymbol{y}$ は以下のように記述される．

$$\begin{cases} \boldsymbol{x} = A^{-1}\boldsymbol{f} - A^{-1}B\boldsymbol{y}, \\ \boldsymbol{y} = \left(C^{\mathrm{T}}A^{-1}B\right)^{-1}\left(C^{\mathrm{T}}A^{-1}\boldsymbol{f} - \boldsymbol{g}\right). \end{cases}$$

ここで，行列 U とベクトル $\boldsymbol{v}$ をそれぞれ $U \equiv A^{-1}B$, $\boldsymbol{v} \equiv A^{-1}\boldsymbol{f}$ と定義し，さらに $\hat{X} \equiv [U\ \boldsymbol{v}]$, $\hat{B} \equiv [B\ \boldsymbol{f}]$ と定義することで，複数右辺ベクトルをもつ連立一次方程式 $A\hat{X} = \hat{B}$ が得られる．この連立一次方程式を解き解行列 $\hat{X}$ を求めることにより，鞍点型連立一次方程式の解ベクトルを計算することができる．この複数右辺連立一次方程式の係数行列は A であり，行列B, C の列数 m の増加は係数行列には影響を及ぼさない．そのため，複数右辺連立一次方程式は鞍点型連立一次方程式よりも収束性の観点から反復法での求解が容易である．なお，この複数右辺連立一次方程式の求解部分が提案法の主要計算部となる．この部分を高速化することにより，提案法の高速化も可能となる．

次に，提案法の並列化と GPU 化について述べる．複数右辺連立一次方程式 $A\hat{X} = \hat{B}$ の右辺行列 $\hat{B}$ を構成する列ベクトルは互いに依存関係がないため，分割が可能である．これらを分割して得られた複数の連立一次方程式は同時求解が可能であり，分割された各連立一次方程式も並列求解が可能である．よって，提案法は階層型の並列性をもつ．行列 $\hat{X}$ と $\hat{B}$ を

$$\hat{X} = \left[\hat{X}^{(0)}, \hat{X}^{(1)}, \dots, \hat{X}^{(P-1)}\right], \hat{B} = \left[\hat{B}^{(0)}, \hat{B}^{(1)}, \dots, \hat{B}^{(P-1)}\right]$$

と分割する．ここで，P は MPI プロセス数とし，$\hat{X}^{(j)}, \hat{B}^{(j)}$ $(j = 0, 1, \dots, P-1)$ は $n \times s$ 行列である．但し，$s \equiv (m+1)/P$である（簡単のため，$m+1$ は P で割り切れるとする）．このように分割することにより，P個の複数右辺連立一次方程式が得られる．

$$A\hat{X}^{(j)} = \hat{B}^{(j)}, j = 0,1, \dots, P-1.$$

実装の方針として，第 j 番目の連立一次方程式 $A\hat{X}^{(j)} = \hat{B}^{(j)}$ に対して，第 j 番目の MPI プロセスを割り当てる．また，各 MPI プロセスに対して GPU を 1 基割り当て，GPU で複数右辺連立一次方程式の求解を行う．なお，複数右辺連立一次方程式の求解には，ブロッククリロフ部分空間反復法を適用する．ブロッククリロフ部分空間反復法の特長として，複数本の解ベクトルを同時に計算できること，及び右辺ベクトル数の増加に伴い反復回数が減少する傾向があることが挙げられる．ブロッククリロフ部分空間反復法で必要となる計算カーネルは，

1) 疎行列・密行列積：$Y = AX$

2) ブロック AXPY 計算：$Y = X\alpha$

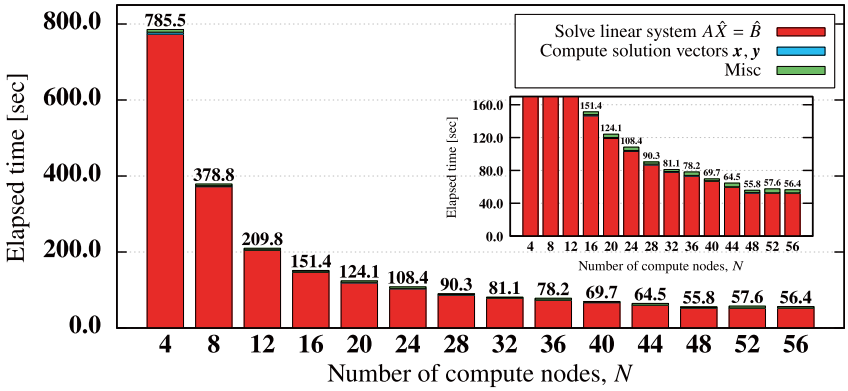
3) ブロック内積計算：$\beta = X^{\mathrm{T}}Y$

4) s 次係数行列とs 本の右辺ベクトルをもつ連立一次方程式の求解

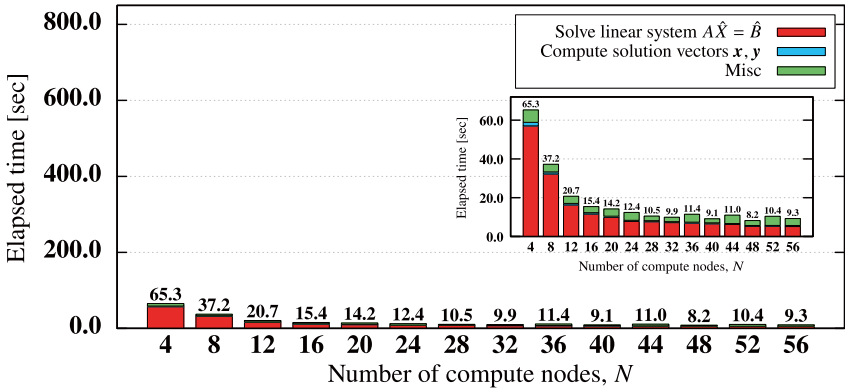
である．ここで，A は n 次疎行列，X, Y は $n \times s$ 密行列，α, β は s 次密行列である．GPU におけるこれらの計算の実行には，NVIDIA CUDA ライブラリを用いる．1) には cuSPARSE ライブラリの関数 `cusparseSpMM` を用いる．2), 3) の計算には cuBLAS ライブラリの関数 `cublasDgemm` を用いる．また，4) には cuSOLVER ライブラリの関数 `cusolverDnDgetrf`，及び `cusolverDnDgetrs` を用いる．GPU での連立一次方程式の求解完了後に，得られた解行列 $\hat{X}^{(j)}$ が host 側に転送される．その後，行列 C と $\hat{X}$ は行方向に分割され各 MPI プロセスに分散される．鞍点型連立一次方程式の解ベクトルの計算には，行列 $C^{\mathrm{T}}U$ とベクトル $C^{\mathrm{T}}\boldsymbol{v}$ が必要となるが，これらの部分和は各 GPU で計算され，`MPI_Reduce` により第 0MPI プロセスに縮約される．現時点の実装では，解ベクトル $\boldsymbol{y}$ を求めるための連立一次方程式の求解は第 0MPI プロセスの GPU でのみ行っている．同方程式の求解後，ベクトル $\boldsymbol{y}$ を全プロセスに送信し，解ベクトル $\boldsymbol{x}$ を全プロセスの GPU で並列に計算する．

数値実験を通して提案法の性能評価を行う．テスト行列として，行列 A には SuiteSparse Matrix Collection で公開されている行列 atmosmodl（行列サイズ n：1,489,752，非零要素数：10,319,760）を用い，行列 B, C の列数 m は 3,000 とした．実験環境として，筑波大学計算科学研究センターのスーパーコンピュータ「Cygnus」を用いた．各計算ノードには，CPU として Intel Xeon Gold 6126 (12 cores) が 2 基，GPU として NVIDIA Tesla V100 が 4 基搭載されている．コンパイラは nvfortran ver. 22.3.0，MPI ライブラリは OpenMPI ver. 4.1.2 を用いた．各計算ノードには GPU が 4 基搭載されているため，ノード内には MPI プロセスを 4 つ立ち上げ，各 MPI プロセスに GPU を 1 基割り当てた．また，各 MPI プロセス内は OpenMP を用いて並列化を行い，OpenMP のスレッド数は 6 とした．各連立一次方程式の求解には，ブロッククリロフ部分空間反復法の 1 つである Block GWBiCGSTABrQ 法を用いた．前処理法として近似逆行列前処理を用いた．前処理行列は全ての連立一次方程式で共通であるため，前処理行列は全計算ノードの CPU を用いて並列計算した．

図に計算ノード数 N の変化に対する鞍点型連立一次方程式の求解時間変化を示す．計算ノード内の MPI プロセス数は 4 であるため，全プロセス数は $P = 4N$ となる．図(a)に示すように，CPU のみで計算を行った場合は，複数右辺連立一次方程式 $A\hat{X} = \hat{B}$ の求解時間が全体の時間の大半を占めている．一方，図(b)に示すように，GPU を用いることで複数右辺連立一次方程式の求解時間を大幅に短縮することができ，それに伴い鞍点型連立一次方程式の求解も高速化することができた．特に $N = 4$ の場合は GPU を用いることにより，鞍点型連立一次方程式の求解時間が CPU 版と比較して約 12 倍高速になった．以上より，提案法は GPU を搭載した並列計算環境に対しても高い親和性があることが確認された．



(a) CPU のみで計算した場合（挿入図は 0〜170 秒の拡大図）



(b) GPU を用いた場合（挿入図は 0〜72 秒の拡大図）

図 10. 計算ノード数 N の変化に対する鞍点型連立一次方程式の求解時間の変化

[8] GPU・FPGA 複合型演算加速クラスタを用いた宇宙輻射輸送コード ARGOT の多ノード並列化 (小林，藤田，朴)

高い演算性能とメモリバンド幅を有する GPU (Graphics Processing Unit) に演算通信性能に優れている FPGA (Field Programmable Gate Array) を連携させ，双方を相補的に利用する GPU-FPGA 複合システムに関する研究を推進している．そのターゲットアプリケーションである宇宙輻射輸送コード ARGOT の演算加速は，GPU と FPGA の両方を搭載した 1 つのノードを使用し，1 つの CPU プロセスが GPU と FPGA でそれぞれ動作する CUDA と OpenCL カーネルを呼び出して制御することによって成されてきた．

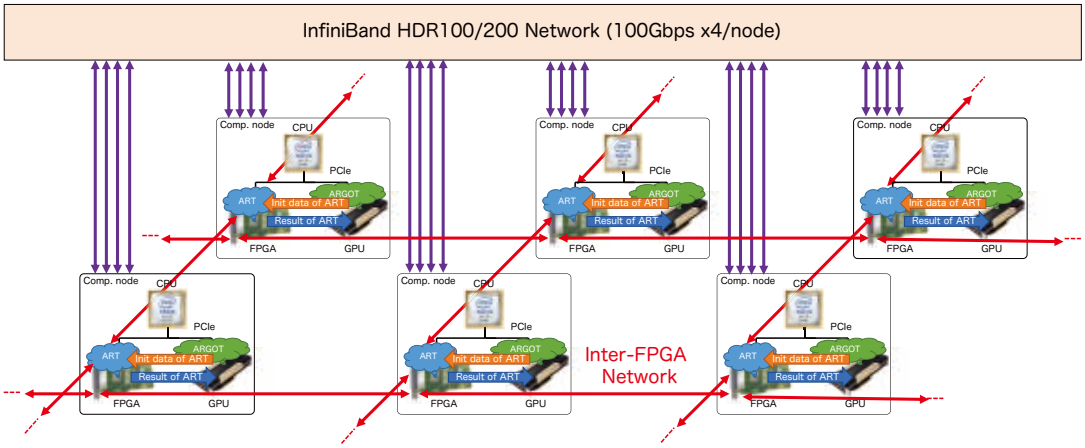


図 11. GPU・FPGA 連携 ARGOT コードの多ノード並列化の概観

本研究では，その実装を MPI および FPGA 間通信技術である CIRCUS (Communication Integrated Reconfigurable CompUting System) を用いて複数ノードで動作するように拡張した．図に GPU・FPGA 連携 ARGOT コードの多ノード並列化の概観を示す．すべての計算ノードに GPU と FPGA が搭載され，各ノードに割り当てられた MPI プロセスによって，両タイプのデバイスが制御される．スーパーコンピュータ Cygnus を用いて実装する場合，ARGOT コード内で GPU が不得手とする演算カーネルである ART 法は光リンクで接続された FPGA 間で並列実行され，残りの演算全ては MPI プログラミングモデルを用いて複数の GPU 上で並列化される．

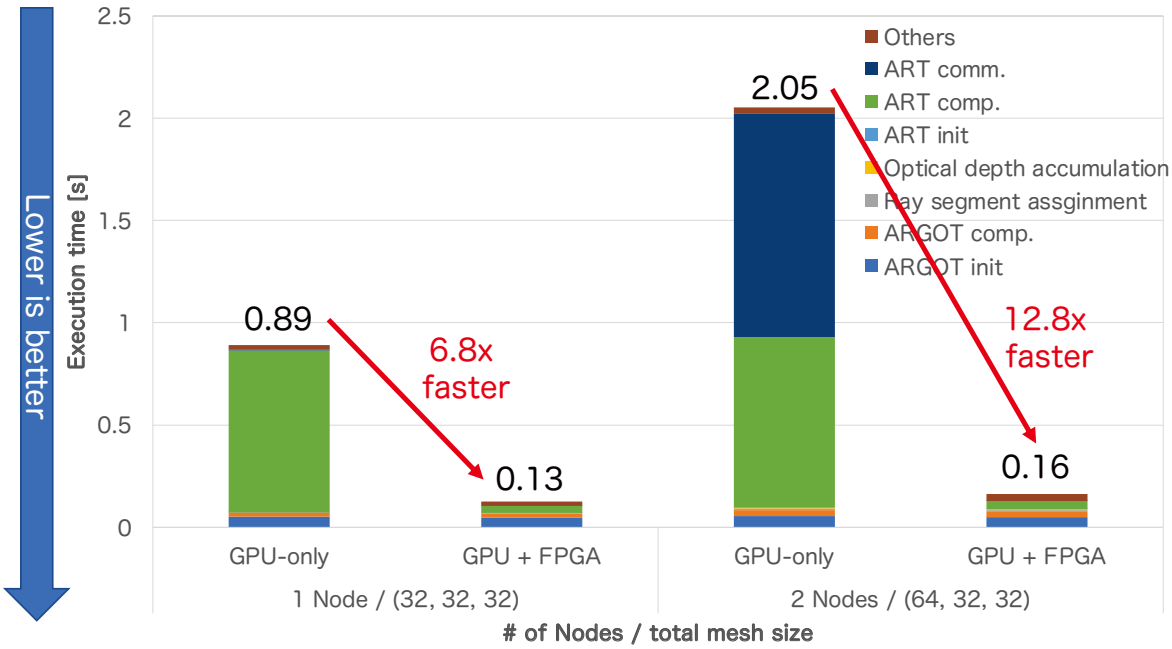


図 12. 弱スケーリングの条件における GPU 実装の ARGOT コードと GPU・FPGA 連携 ARGOT コードの性能比較

現在の FPGA 実装では，問題サイズが 32^3 までしか扱えないため，1 ノードあたり 32^3 の問題サイズを割り当て，弱スケーリングの条件で GPU 実装の ARGOT コードと GPU・FPGA 連携 ARGOT コードの性能を比較した．図に評価結果を示す．なお，今回の評価では，最大 2 ノードを使用し，2 つのノードにおける問題サイズの合計は 64×32×32 とした．1 ノードにおける ARGOT コードの性能にまず着目すると, GPU 実装では ART 法の実行時間が ARGOT コード全体において支配的であることが分かる. その ART 法を FPGA にオフロードすることによって，ARGOT コードの実行時間は 0.89 秒から 0.13 秒に短縮され，GPU のみの実行に比べて 6.8 倍の高速化を達成した．これは FPGA に実装された ART アクセラレータは，パイプライン並列および空間並列性を活用して，ART 法の実効性能を最大化しているためである．そして，ARGOT コードを 2 ノードで実行すると，GPU 実装の場合は並列 ART 法の MPI 通信に要する時間が ARGOT コードの総実行時間の半分を占めていることが分かる．一方，ART 法を 2 ノードのマルチ FPGA で実行した場合は，オーバーヘッドを殆ど生じさせずにその通信処理を実行でき, その結果 ARGOT コードの実行時間は 0.16 秒に短縮されている. これは，通信と計算を融合したパイプラインが CIRCUS によって構築されたことにより，すべての ART 演算と両ノードの FPGA における FPGA 間通信は完全に同時に実行され，パイプラインが充填されれば，64×32×32 の問題サイズを対象とした ART 法はほぼ 100%の効率で実行されるためである．つまり，2 ノード実行によって得られた 12.8 倍の性能向上は，ART 法の演算実効性能の最大化に，細粒度データ転送の効率化の効果が上乗せされたものである．ただし，2 つのノードの結果には最大 10^{-1} の相対誤差があり，その原因については現在調査中である．

[9] 並列 FPGA 環境における集団通信に関する研究（藤田、小林、朴）

筑波大学計算科学研究センターでは，直接接続された複数 FPGA 上で利用可能な通信フレームワーク CIRCUS (Communication Integrated Reconfigurable CompUting System) を開発している. しかし, CIRCUS は 1 対 1 の通信しかサポートしておらず, MPI (Message Passing Interface) がサポートしているような HPC アプリケーションで広く利用されている集団通信は実装されていない．CIRCUS を使った高性能計算 FPGA アプリケーションを実装することを考えると，このような集団通信の機能なしに実装することは難しく，CIRCUS の集団通信対応が強く求められる．本研究では，集団通信の一つである Allreduce の実装と性能評価を行う．Allreduce は反復法の収束判定や機械学習で用いられており，重要な集団通信の一つである．

CIRCUS における Allreduce 通信は, 木構造を用いるアルゴリズムを用いて実装する. 図(左)にあるように，Reduce カーネルと Broadcast カーネルを FPGA 上に実装する．Reduce カーネルは, 葉にあたる FPGA からデータを集約し, 根にある FPGA に送るカーネルである. Boardcast カーネルは Allreduce 計算結果を全ての FPGA に木構造に従って配るカーネルである. 図(右)

に性能評価の結果を示す．最大で 64.6Gbps の性能が得られた．最大で 100Gbps の通信が行えることを考えると，この性能は十分であるとは言い難い．現状，64KB までしか性能を計測出来ておらず，メッセージ長が短いことが性能低下の原因の一つであると考えられる．しかしながら，CIRCUS はフロー制御がなく，少量ある受信バッファがあふれるとデータが消失してしまう．そのため，メッセージ長を長くすると Allreduce 通信が安定しないという問題を抱えている．加えて，この問題を緩和するために追加している処理に由来するオーバーヘッドがあり，性能を制限していることが明らかとなった．フロー制御がないため，必ず受信を担当するカーネルが先に起動するように制御して全体を実行する必要があり，本来ならば不要な処理が含まれてしまう．

今後の課題として，フロー制御がないことを補うために，通信プロトコルの変更を検討している．現在の実装では，Intel 社が提供する SerialLite III プロトコルを用いて FPGA 間通信を行っているが，これにフロー制御機能がない．代替プロトコルとして，オープンソースの FPGA 間通信プロトコルである Kyokko を CIRCUS に組み込むことを検討しており，我々が対象としている FPGA ボードで動作するか，通信性能，フロー制御の機能について確認を進めている．

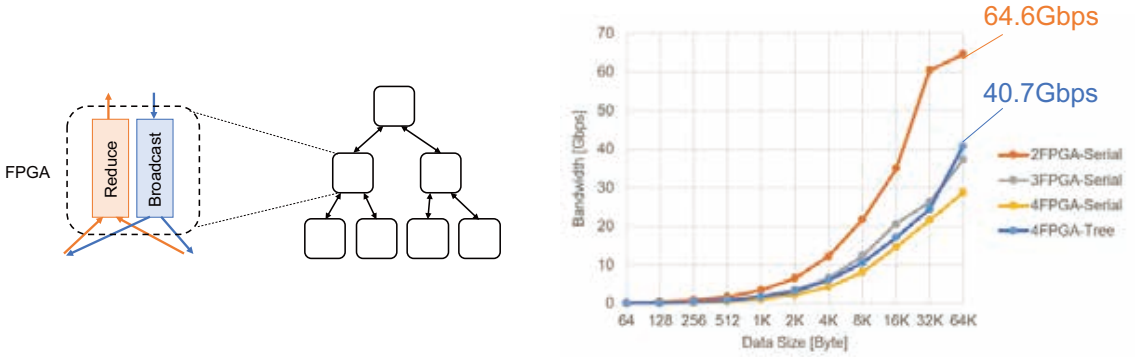


図 13. （左）Allreduce 実装の概要　（右）性能評価結果

[10] HBM2 搭載 FPGA における高性能計算向けメモリシステムに関する研究（藤田、小林、朴）

近年 FPGA ベンダーより広帯域なメモリである High Bandwidth Memory 2（HBM2）が搭載されている FPGA ボードがリリースされつつある．Intel FPGA においては，HBM2 を用いることで最大 512GB/s の外部メモリ帯域を利用できる．HBM2 は比較的低速のメモリを多数束ねたアーキテクチャであり，全てのメモリチャンネルに同時アクセスしなければ，その高い理論ピーク性能を得ることが出来ない．HBM2 は他の演算加速装置でも利用されており，特に高性能な Graphics Processing Unit（GPU）に搭載されている．GPU においては，高性能なキャッシュや内部メモリネットワークを土台として，多数のコアによる並列アクセスが行われ，

HBM2 の高い性能を引き出している．しかしながら，FPGA には HBM2 を扱うためのそのような（固定機能としての）専用メモリシステムが実装されていない．このようなメモリシステムを用いず，複雑なメモリアクセスを行うことが一般的である HPC アプリケーションを実装することは困難であり，HBM2 搭載 FPGA 向けかつ HPC アプリケーション向けメモリシステムが求められている．これらのシステムは，再構成可能な回路要素を用いて FPGA 内に実装しなければならない．

本研究では，この問題を解決するために，HBM2 搭載 FPGA 向けかつ HPC アプリケーション向けメモリシステムの提案と実装を行う．提案するシステムは図 14（左）にあるように 2 つの主要コンポーネントから構成される．メモリアクセスの柔軟性を担保し複数のメモリチャンネルを効率よく扱うためのクロスバと，FPGA に内蔵されているメモリ（Block RAM: BRAM）を用いるキャッシュ（Addressable Cache）から構成される．また，アプリケーションの開発には，ソフトウェアで用いられている C 言語や C++言語を用いて FPGA ハードウェアを開発できる高位合成（High-Level Synthesis: HLS）を前提として設計を行っている．

HBM2 とキャッシュ間データ転送は Direct Memory Access Controller (DMAC)を用いて行われる．そして，DMAC の制御は，RISC-V をベースに簡略化した独自プロセッサを用いて制御する．どこのデータをどこにいつ転送するかは，RISC-V 命令を用いたソフトウェアとして表される．このような設計方式を採用することにより，専用設計のハードウェアがもつ高い性能と，ソフトウェアが持つ高い柔軟性を両立できる．また，異なるアプリケーションに本システムを適用する際でも，メモリシステムは同じ設計を使い回すことができる．

図（右）に性能評価の結果を示す．最大で 102GB/s の性能が得られた．本実験では 8 チャネル分の実装を行っており，理論ピーク性能は 102.4GB/s であるため，期待通り動作していることがわかる．今後の課題としては，FPGA がもつ全チャンネル（32）に対する実装を行うことと，実際のアプリケーションで性能評価を行うことがあげられる．全チャンネル実装に対しては，クロスバがボトルネックとなり動作周波数が維持できないことが問題であることがわかっており，多段ネットワークなど複雑度が対数オーダーとなるネットワークを用いることでこの問題を解決できないか検討を進めている．

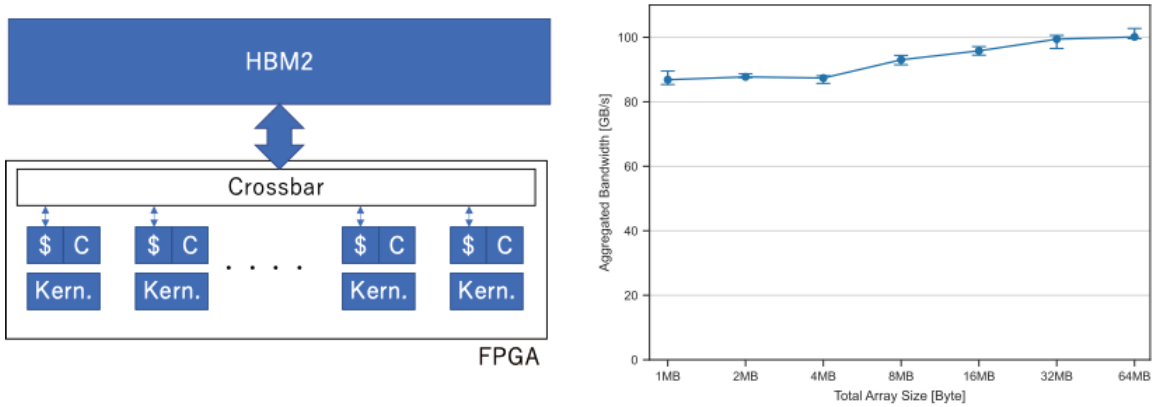


図 14. （左）提案システムの概要図。 （右）バンド幅性能評価結果

4. 教育

1. 枝松拓弥，博士（工学），A Study of High Performance Multiple-Precision Integer Multiplication and Division Using SIMD Instructions（SIMD 命令による高性能な多倍長整数の乗算と除算についての研究），筑波大学大学院理工情報生命学術院システム情報工学研究群博士論文，令和 5 年 3 月（指導：高橋大介）
2. 菊池航平，修士（工学），並列 FPGA 環境における集団通信に関する研究，筑波大学大学院理工情報生命学術院システム情報工学研究群修士論文，令和 5 年 3 月（指導：朴泰祐）
3. Nam Woon，修士（工学），四元数に基づくヒューマノイドアニメーションの深層学習の比較と評価，筑波大学大学院理工情報生命学術院システム情報工学研究群修士論文，令和 5 年 3 月（指導：高橋大介）
4. 大滝恒輝，修士（工学），銀河形成向け流体力学シミュレーションの GPU 実装と性能評価，筑波大学大学院理工情報生命学術院システム情報工学研究群修士論文，令和 5 年 3 月（指導：高橋大介）
5. 巨畠和樹，修士（工学），HPC における一時ストレージとその応用に関する研究，筑波大学大学院理工情報生命学術院システム情報工学研究群修士論文，令和 5 年 3 月（指導：建部修見）
6. 笠井大暉，修士（工学），ノードローカルストレージを利用する一時分散キャッシュファイルシステムの設計と実装，筑波大学大学院理工情報生命学術院システム情報工学研究群修士論文，令和 5 年 3 月（指導：建部修見）

7. 立木 佑弥，修士（工学），遊休 GPU を利用したホスト・デバイス間通信の高速化，筑波大学大学院理工情報生命学術院システム情報工学研究群修士論文，令和 5 年 3 月（指導：額田彰）
8. 菅沼夏樹，修士（工学），多数の右辺ベクトルを持つ連立一次方程式に対する反復解法の高速化に関する研究，筑波大学大学院理工情報生命学術院システム情報工学研究群修士論文，令和 5 年 3 月（指導：多田野寛人）
9. 市塚勇正，学士（情報工学），数独パズルにおける複数解の個数計算，筑波大学情報学群情報科学類卒業論文，令和 5 年 3 月（指導：高橋大介）
10. 鈴木貴裕，学士（情報工学），reduced-radix 表現の Toom-Cook 乗算への適用，筑波大学情報学群情報科学類卒業論文，令和 5 年 3 月（指導：高橋大介）
11. 長橋朋也，学士（情報科学），スライドパズルにおけるゼロ認識パターンデータベースの OpenMP を用いた並列構築，筑波大学情報学群情報科学類卒業論文，令和 5 年 3 月（指導：高橋大介）
12. 丸山泰史，学士（情報工学），差分プライバシー保証のための汎用プログラミング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，令和 5 年 3 月（指導：建部修見）
13. 伊能一翔，学士（情報工学），GPU による行列積計算を対象とする自動チューニングの研究，筑波大学情報学群情報科学類卒業論文，令和 5 年 3 月（指導：額田彰）
14. 関拓己，学士（情報科学），総和計算による GPU の性能解析，筑波大学情報学群情報科学類卒業論文，令和 5 年 3 月（指導：額田彰）
15. 齋藤颯人，学士（情報工学），Group-wise 更新付き Block Krylov 部分空間反復法における漸化式の可変的グループ化による性能改善，筑波大学情報学群情報科学類卒業論文，令和 5 年 3 月（指導：多田野寛人）
16. 北爪開人，学士（情報工学），高性能並列 FPGA 環境における通信高度化に関する研究，筑波大学情報学群情報科学類卒業論文，令和 5 年 3 月（指導：藤田典久，朴泰祐，小林諒平）

5. 受賞、外部資金、知的財産権等

外部資金

1. 科学研究費補助金基盤研究 (A)，朴泰祐（代表），R3～R6 年度，10,530 千円（R4 年度），「多重複合演算加速機構を用いた次世代スーパーコンピューティング」
2. 科学研究費補助金基盤研究 (C)，高橋大介（代表），R4～R6 年度，1,170 千円（R4 年度），「メニーコア超並列クラスタにおける多倍長演算に関する研究」

3. 文部科学省委託研究，高橋大介（分担），R4 年度，4,000 千円，次世代計算基盤に係る調査研究
4. 科学研究費基盤研究（A），建部修見（代表），R4～R8 年度，7,250 千円，次世代ストレージアーキテクチャの研究
5. 文部科学省委託研究，建部修見（分担），R4 年度，54,450 千円，HPCI の運営（HPCI 共用ストレージ用大規模分散ファイルシステムの機能整備等）
6. NEDO，建部修見（分担），H30～R4 年度，6,128 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. 科学研究費補助金基盤研究 (C)，多田野寛人（代表），R2～R4 年度，1,040 千円（R4 年度），「鞍点型連立一次方程式に対する階層並列型高速数値解法の開発」
8. 科学研究費補助金若手研究，小林諒平（代表），2022~2024 年度，3,250 千円（2022 年度），「GPU・FPGA 複合型グラフ構造データ分析基盤の創出」

6. 研究業績

(1) 研究論文

A) 査読付き論文

1. Riadh Ben Abdelhamid, Yoshiki Yamaguchi, Taisuke Boku, “A scalable many-core overlay architecture on an HBM2-enabled multi-die FPGA”, ACM Transactions on Reconfigurable Technology and Systems, Vol. 16, Issue 1, No:15, pp. 1-33, June 2022.
2. Taisuke Boku, Norihisa Fujita, Ryohei Kobayashi, Osamu Tatebe, “Cygnus - World First Multihybrid Accelerated Cluster with GPU and FPGA Coupling”, Proc. of Int. Workshop on Deployment and Use of Accelerators (DUAC2022), Aug. 2022.
3. Takuya Edamatsu and Daisuke Takahashi, “Fast Multiple-Precision Integer Division Using Intel AVX-512”, IEEE Transactions on Emerging Topics in Computing, Vol. 11, No. 1, pp. 224-236, 2023.
4. Daisuke Takahashi, “On the use of Montgomery multiplication in the computation of binary BBP-type formulas for mathematical constants”, The Ramanujan Journal, Vol. 59, No. 1, pp. 211-219, 2022.
5. Yukimasa Sugizaki and Daisuke Takahashi, “A Fast Algorithm for Computing the Number of Magic Series”, Annals of Combinatorics, Vol. 26, No. 2, pp. 511-532, 2022.
6. Takuya Edamatsu and Daisuke Takahashi, “Efficient Large Integer Multiplication with Arm SVE Instructions”, Proc. International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2023), pp. 9-17, 2023.

7. Daisuke Takahashi, “An Implementation of Parallel Number-Theoretic Transform Using Intel AVX-512 Instructions”, Proc. 24th International Workshop on Computer Algebra in Scientific Computing (CASC 2022), Lecture Notes in Computer Science, Vol. 13366, pp. 318-332, Springer, 2022.
8. Osamu Tatebe, Hiroki Ohtsuji, “Caching Support for CHFS Node-local Persistent Memory File System”, Proceedings of 3rd Workshop on Extreme-Scale Storage and Analysis (ESSA 2022), pp.1103-1110, 10.1109/IPDPSW55747.2022.00182, 2022
9. Sohei Koyama, Osamu Tatebe, “Scalable Data Parallel Distributed Training for Graph Neural Networks”, Proceedings of Workshop on AI for Datacenter Optimization (ADOPT'22), pp.699-707, 10.1109/IPDPSW55747.2022.00121, 2022
10. Yuya Tatsugi, Akira Nukada, “Accelerating Data Transfer between Host and Device using Idle GPU”, GPGPU '22: Proceedings of the 14th Workshop on General Purpose Processing Using GPU, Article No. 2, pp. 1-6, ACM, New York, Apr. 2022.
11. Adrián P. Diéguez, Margarita Amor, Ramón Doallo, Akira Nukada, Satoshi Matsuoka. “Efficient High-Precision Integer Multiplication on the GPU”, The International Journal of High Performance Computing Applications, SAGE Publications, vol. 36, no. 3, pp. 356-369, May 2022.
12. Shota Ishikawa, Hiroto Tadano, and Ayumu Saitoh, “Application and performance evaluation of a method using block structures for saddle point problems appearing in image reconstruction problems”, JSIAM Letters, Vol. 14, pp. 114-118, 2022.
13. Hiroki Nakano, Hiroto Tadano, Norikazu Todoroki, and Toru Sakai, “The Haldane Gap of the S=1 Heisenberg Antiferromagnetic Chain”, Journal of the Physical Society of Japan, Vol. 91, No. 7, pp. 074701-1-074701-5, 2022.
14. Hiroto Tadano, “Implementation of a hierarchical parallel solver for saddle point problems on a GPU cluster”, Proc. of The 41st JSST Annual International Conference on Simulation Technology (JSST2022), pp. 220-223, 2022.
15. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Taisuke Boku, Kohji Yoshikawa, Makito Abe, and Masayuki Umemura, “Accelerating Radiative Transfer Simulation on NVIDIA GPUs with OpenACC”, PDCAT 2022: Parallel and Distributed Computing, Applications and Technologies, Lecture Notes in Computer Science, vol 13798, pp.344-358, 2023-04.
16. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Taisuke Boku, Kohji Yoshikawa, Makito Abe, Masayuki Umemura, “GPU–FPGA-accelerated Radiative Transfer Simulation

with Inter-FPGA Communication”, In Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia '23), pp.117–125, 2023-02.

17. Yoshiyuki Morie, Yasutaka Wada, Ryohei Kobayashi, Ryuichi Sakamoto, “Data Transfer API and its Performance Model for Rank-Level Approximate Computing on HPC Systems”, International Journal of Networking and Computing Vol.13, No.1, pp. 48-61, 2023-01.
18. Kento Miura, Ryohei Kobayashi, Toshiyuki Amagasa, Hiroyuki Kitagawa, Norihisa Fujita, Taisuke Boku, “An FPGA-based Accelerator for Regular Path Queries over Edge-labeled Graphs”, 2022 IEEE International Conference on Big Data (Big Data), pp. 415-422, 2023-01.
19. Taisuke Boku, Norihisa Fujita, Ryohei Kobayashi, Osamu Tatebe, “Cygnus - World First Multihybrid Accelerated Cluster with GPU and FPGA Coupling”, In Workshop Proceedings of the 51st International Conference on Parallel Processing (ICPP Workshops '22), Article 8, pp.1–8, 2023-01.
20. Ryohei Kobayashi, Kento Miura, Norihisa Fujita, Taisuke Boku, Toshiyuki Amagasa, “An Open-source FPGA Library for Data Sorting”, IPSJ Journal of Information Processing 30, pp.766-777, 2022-10.
21. Yoshiyuki Morie, Yasutaka Wada, Ryohei Kobayashi, Ryuichi Sakamoto, “Performance Evaluation of Data Transfer API for Rank Level Approximate Computing on HPC Systems”, 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp.445-448 2022-08.
22. Yuka Sano, Ryohei Kobayashi, Norihisa Fujita, Taisuke Boku, “Performance Evaluation on GPU-FPGA Accelerated Computing Considering Interconnections between Accelerators”, The Proceedings of the 12th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART 2022), pp.10-16 2022-06.
23. Kohei Kikuchi, Norihisa Fujita, Ryohei Kobayashi, Taisuke Boku, "Implementation and Performance Evaluation of Collective Communications Using CIRCUS on Multiple FPGAs", In proceedings of the International Conference on High Performance Computing in Asia-Pacific 2023 (HPC Asia '23), pp. 15-23, Feb. 2023. doi: https://doi.org/10.1145/3581576.3581602.
24. Norihisa Fujita, Ryohei Kobayashi, Yoshiki Yamaguchi, Taisuke Boku, “Implementation and Performance Evaluation of Memory System using Addressable Cache for HPC Applications on HBM2 equipped FPGAs,” Proceedings of 20th International Workshop for Algorithms, Models, and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar 2022), pp. 121-132, 2023. doi: https://doi.org/10.1007/978-3-031-31209-0_9.

B) 査読無し論文

1. 山口博將，高橋大介，“ルジャンドル予想の数値的検証”，情報処理学会第 85 回全国大会講演論文集, 2 pages, 2023 年 3 月.
2. 高橋大介，“数学定数に対する 2 進 BBP 型公式の計算における Montgomery 乗算の使用について”，日本応用数理学会 2022 年度年会講演予稿集, 2 pages, 2022 年 9 月.
3. 高橋大介，“Intel AVX-512IFMA 命令を用いた並列数論変換の実現と評価”，日本応用数理学会 2022 年度年会講演予稿集, 2 pages, 2022 年 9 月.
4. 巨畠和樹, 建部修見, 不揮発性メモリを用いた分散オブジェクトストレージの設計, 研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2022-HPC-184，No. 3，pp. 1-10，2022 年 5 月.
5. 巨畠和樹，小山創平，平賀弘平，建部修見，HPC 環境を想定した探索的データ解析におけるノードローカルストレージの利用の検討，研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2022-HPC-185，No. 19，pp. 1-8，2022 年 7 月.
6. 建部修見, CHFS アドホック並列分散ファイルシステムのアクセス性能の評価, 研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2022-HPC-185，No. 31，pp. 1-6，2022 年 7 月.
7. 平賀弘平，建部修見，MPI-IO/CHFS:ノードローカル不揮発性メモリを活用するアドホック分散ファイルシステムのための MPI-IO 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. 2022-HPC-185，No. 33，pp. 1-9，2022 年 7 月.
8. 笠井大暉, 建部修見, 分散キャッシュファイルシステムの設計と実装, 研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2022-HPC-186，No. 6，pp. 1-6，2022 年 9 月.
9. 勢見達将, 額田彰.「DO CONCURRENT 構文による OpenSWPC の GPU 化」, 情報処理学会研究報告, Vol. 2023-HPC-188, No. 19, pp. 1-7, 2023 年 3 月.
10. 中野博生, 轟木義一, 多田野寛人, 坂井徹, “富岳を用いた量子スピン系の大規模並列数値対角化計算の現状”，日本物理学会 2022 年秋季大会講演概要集, 1 page, 2022 年 9 月.
11. 多田野寛人, “GPU クラスタにおける鞍点型連立一次方程式に対する階層並列型解法の実装と性能評価”，日本応用数理学会 2022 年度年会講演予稿集, 2 pages, 2022 年 9 月.
12. 菅沼夏樹, 多田野寛人, “複数右辺連立一次方程式に対するブロック・グローバル混合型反復法の構築と性能評価”, 日本応用数理学会 2022 年度年会講演予稿集, 2 pages, 2022 年 9 月.

13. 溝谷祐大, 小林諒平, 藤田典久, 朴泰祐, 天笠俊之, “FPGA 間通信フレームワーク CIRCUS を利用した複数 FPGA によるグラフ幅優先探索の提案”, 第 15 回データ工学と情報マネジメントに関するフォーラム (DEIM 2023), 2a-7-5, 2023-03.
14. 佐野由佳, 小林諒平, 藤田典久, 朴泰祐, 佐藤三久, “FPGA 高位合成における演算性能向上のための空間並列性記述に関する研究”, 研究報告ハイパフォーマンスコンピューティング（HPC）/ 2023-HPC-188(22)/pp.1-10, 2023-03.
15. 小林諒平, 藤田典久, 山口佳樹, 朴泰祐, 吉川耕司, 安部牧人, 梅村雅之,“GPU･FPGA 複合型演算加速クラスタを用いた宇宙輻射輸送コード ARGOT の多ノード並列化”, 研究報告ハイパフォーマンスコンピューティング（HPC）/ 2022-HPC-185(1)/pp.1-6, 2022-07.
16. 瀬口知洋, 中井榛希, 山口佳樹, 藤田典久, 小林諒平, 朴泰祐, “並列化に伴うデータ空間の分割とそれによるアクセスパターンの変化がもたらす HBM の振る舞い調査”, IEICE-CPSY2022-15, IEICE-122(133) pp.83-88 2022-07.
17. 佐野由佳, 小林諒平, 藤田典久, 朴泰祐, “ノードを跨いだ GPU・FPGA 複合型演算加速による宇宙物理シミュレーションの実装と評価”, 研究報告ハイパフォーマンスコンピューティング（HPC）/ 2022-HPC-184(6)/pp.1-7, 2022-05.
18. 菊池航平, 藤田典久, 小林諒平, 朴泰祐, "並列 FPGA 環境における通信システム CIRCUS を用いた集団通信の実装と性能評価", 研究報告ハイパフォーマンスコンピューティング（HPC）, 2022-HPC-187, 8 pages, 2022.
19. 佐野由佳, 小林諒平, 藤田典久, 朴泰祐, 佐藤三久, "FPGA 高位合成における演算性能向上のための空間並列性記述に関する研究", 研究報告ハイパフォーマンスコンピューティング（HPC）, 2023-HPC-188(22), 10 pages, 2023.

(2) 国際会議発表

A) 招待講演

1. Taisuke Boku, " How FPGA can compensate with High Performance Computing ?", Keynote, Int. Conf. FPT2022, Hong Kong, Dec. 2022.
2. Taisuke Boku, " Multi-Hetero Accelerated Computing – Challenge toward Extreme Heterogeneity", Int. Workshop ExHET2022 (in PPoPP2022), on-line, Apr. 2022.
3. Taisuke Boku, "How FPGA can contribute to HPC ?", VLSI-DAT2022 Symposium, Taipei, Apr. 2022.
4. Taisuke Boku, "HPC/BD/AI Supported by Big Memory Supercomputer", Int. Workshop EC2-2022, Warsaw, May 2022.

5. Taisuke Boku, "Cygnus-BD: the Big Memory Supercomputer for HPC, Big Data and AI", 2022 MVAPICH Users Group Workshop, Columbus, Aug. 2022.
6. Osamu Tatebe, “Persistent Memory Supercomputer Pegasus for Data-driven and AI-driven Science”, IXPUG Workshop at HPC Asia 2023, Feb. 2023.
7. Ryohei Kobayashi, “OpenACC-Enabled GPU-FPGA Accelerated Computing for Astrophysics Simulation”, OpenACC and Hackathons Asia-Pacific Summit 2022, 2022-08-23.

B) 一般講演

1. Taisuke Boku, “Cygnus - World First Multihybrid Accelerated Cluster with GPU and FPGA Coupling”, Proc. of Int. Workshop on Deployment and Use of Accelerators (DUAC2022), on-line, Aug. 2022.
2. Daisuke Takahashi, “Implementation of Parallel Number-Theoretic Transform on Manycore Clusters”, SIAM Conference on Computational Science and Engineering (CSE23), Amsterdam, The Netherlands, Feb. 2023.
3. Takuya Edamatsu and Daisuke Takahashi, “Efficient Large Integer Multiplication with Arm SVE Instructions”, International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2023), Singapore, Feb. 2023.
4. Daisuke Takahashi, “An Implementation of Parallel Number-Theoretic Transform Using Intel AVX-512 Instructions”, 24th International Workshop on Computer Algebra in Scientific Computing (CASC 2022), Gebze, Turkey, (Virtual Conference), Aug. 2022.
5. Osamu Tatebe, Hiroki Ohtsuji, “Caching Support for CHFS Node-local Persistent Memory File System”, 3rd Workshop on Extreme-Scale Storage and Analysis (ESSA 2022), Virtual event, Jun. 2022.
6. Sohei Koyama, Osamu Tatebe, “Scalable Data Parallel Distributed Training for Graph Neural Networks”, Workshop on AI for Datacenter Optimization (ADOPT'22), Virtual event, May 2022.
7. Tatsumasa Seimi, Akira Nukada, “GPU Acceleration of OpenSWPC using DO CONCURRENT”, GPU Technology Conference 2023 Spring, Poster, Online, Mar. 2023.
8. Hiroto Tadano, “Implementation of a hierarchical parallel solver for saddle point problems on a GPU cluster”, The 41st JSST Annual International Conference on Simulation Technology (JSST2022), Virtual Conference, Sep. 2022.
9. Ryohei Kobayashi , Norihisa Fujita, Yoshiki Yamaguchi, Taisuke Boku, Kohji Yoshikawa, Makito Abe, Masayuki Umemura, “GPU–FPGA-accelerated Radiative Transfer Simulation

with Inter-FPGA Communication”, HPC Asia '23: International Conference on High Performance Computing in Asia-Pacific Region, 2023-03.

10. Kento Miura, Ryohei Kobayashi, Toshiyuki Amagasa, Hiroyuki Kitagawa, Norihisa Fujita, Taisuke Boku, “An FPGA-based Accelerator for Regular Path Queries over Edge-labeled Graphs”, 2022 IEEE International Conference on Big Data (Big Data), 2022-12.
11. Ryohei Kobayashi, Norihisa Fujita, Yoshiki Yamaguchi, Taisuke Boku, Kohji Yoshikawa, Makito Abe, and Masayuki Umemura, “Accelerating Radiative Transfer Simulation on NVIDIA GPUs with OpenACC”, PDCAT 2022: Parallel and Distributed Computing, Applications and Technologies, Lecture Notes in Computer Science, 2022-12.
12. Yuka Sano, Ryohei Kobayashi, Norihisa Fujita, Taisuke Boku, “Performance Evaluation on GPU-FPGA Accelerated Computing Considering Interconnections between Accelerators”, HEART2022: International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies, 2022-06.
13. Yoshiyuki Morie, Yasutaka Wada, Ryohei Kobayashi, Ryuichi Sakamoto, “Performance Evaluation of Data Transfer API for Rank Level Approximate Computing on HPC Systems”, 24th Workshop on Advances in Parallel and Distributed Computational Models 2022-05.
14. Kohei Kikuchi, Norihisa Fujita, Ryohei Kobayashi, Taisuke Boku, "Implementation and Performance Evaluation of Collective Communications Using CIRCUS on Multiple FPGAs", the International Conference on High Performance Computing in Asia-Pacific 2023 (HPC Asia '23), Feb. 2023.
15. Norihisa Fujita, Ryohei Kobayashi, Yoshiki Yamaguchi, Taisuke Boku, “Implementation and Performance Evaluation of Memory System using Addressable Cache for HPC Applications on HBM2 equipped FPGAs,” 20th International Workshop for Algorithms, Models, and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar 2022), Aug. 2022.

(3) 国内学会・研究会発表

A) 招待講演

1. 朴泰祐, "NVIDIA H100 を搭載した最新スーパーコンピュータ Pegasus が拓く HPC&AI", NVIDIA GTC2022 Japan, on-line, 2023 年 3 月.
2. 朴泰祐, " GPU・FPGA 複合計算による次世代 HPC/AI 加速技術", PC クラスタワークショップ(FPGA), on-line, 2022 年 5 月.
3. 朴泰祐, " oneAPI による GPU+FPGA マルチヘテロ環境プログラミングとアプリケーション実行", PC クラスタワークショップ(OSS Programming), 2022 年 4 月.

B) その他の発表

1. 山口博將，高橋大介，“ルジャンドル予想の数値的検証”，情報処理学会第 85 回全国大会，東京，2023 年 3 月.
2. 高橋大介，“数学定数に対する 2 進 BBP 型公式の計算における Montgomery 乗算の使用について”，日本応用数理学会 2022 年度年会，オンライン開催，2022 年 9 月.
3. 高橋大介，“Intel AVX-512IFMA 命令を用いた並列数論変換の実現と評価”，日本応用数理学会 2022 年度年会，オンライン開催，2022 年 9 月.
4. 巨畠和樹, 建部修見, 不揮発性メモリを用いた分散オブジェクトストレージの設計, 情報処理学会 HPC 研究会，オンライン，2022 年 5 月.
5. 巨畠和樹，小山創平，平賀弘平，建部修見，HPC 環境を想定した探索的データ解析におけるノードローカルストレージの利用の検討, 情報処理学会 HPC 研究会, 下関, 2022 年 7 月.
6. 建部修見，CHFS アドホック並列分散ファイルシステムのアクセス性能の評価，情報処理学会 HPC 研究会，下関，2022 年 7 月.
7. 平賀弘平，建部修見，MPI-IO/CHFS:ノードローカル不揮発性メモリを活用するアドホック分散ファイルシステムのための MPI-IO の設計. 情報処理学会 HPC 研究会，下関，2022 年 7 月.
8. 笠井大暉, 建部修見, 分散キャッシュファイルシステムの設計と実装, 情報処理学会 HPC 研究会，神戸，2022 年 9 月.
9. 建部修見, Gfarm ファイルシステムの最新機能, Gfarm／実用アプリ・シンポジウム, 2022 年 10 月.
10. 建部修見，Pegasus ビッグメモリスーパコンピュータではじまるこれからのデータ科学・ビッグデータ AI，Gfarm／実用アプリ・シンポジウム，2022 年 10 月.
11. 建部修見，Pegasus ビッグメモリスーパコンピュータではじまるこれからのデータ科学・ビッグデータ AI，第 22 回 PC クラスタシンポジウム，2022 年 12 月.
12. 建部修見，Pegasus ビッグメモリスーパコンピュータではじまるこれからの HPC、ビッグデータ、超大規模 AI，JLUG 2022，2022 年 12 月.
13. 建部修見, Gfarm ファイルシステムの最新機能, Gfarm ワークショップ, 2023 年 2 月.
14. 中野博生, 轟木義一, 多田野寛人, 坂井徹, “富岳を用いた量子スピン系の大規模並列数値対角化計算の現状”，日本物理学会 2022 年秋季大会, 東京工業大学大岡山キャンパス, 2022 年 9 月.
15. 多田野寛人, “GPU クラスタにおける鞍点型連立一次方程式に対する階層並列型解法の実装と性能評価”，日本応用数理学会 2022 年度年会, オンライン, 2022 年 9 月.

16. 菅沼夏樹, 多田野寛人, “複数右辺連立一次方程式に対するブロック・グローバル混合型反復法の構築と性能評価”, 日本応用数理学会 2022 年度年会, オンライン, 2022 年 9 月.
17. 多田野寛人, “鞍点型連立一次方程式に対する階層並列型数値解法の GPU による高速化”, 【非線形問題の高性能解法と可視化技術に関する研究会】2022 年度第 1 回研究会, 日本大学津田沼キャンパス, 2023 年 3 月.
18. 齋藤颯人, 多田野寛人, “漸化式の可変的グループ化による Block GWBiCGSTAB 法の性能改善”, 【非線形問題の高性能解法と可視化技術に関する研究会】2022 年度第 1 回研究会, 日本大学津田沼キャンパス, 2023 年 3 月.
19. 溝谷祐大, 小林諒平, 藤田典久, 朴泰祐, 天笠俊之, “FPGA 間通信フレームワーク CIRCUS を利用した複数 FPGA によるグラフ幅優先探索の提案”, 第 15 回データ工学と情報マネジメントに関するフォーラム (DEIM 2023), 2a-7-5, 2023-03.
20. 佐野由佳, 小林諒平, 藤田典久, 朴泰祐, 佐藤三久, “FPGA 高位合成における演算性能向上のための空間並列性記述に関する研究”, 第 188 回ハイパフォーマンスコンピューティング研究発表会, 2023-03.
21. 小林諒平, 藤田典久, 山口佳樹, 朴泰祐, 吉川耕司, 安部牧人, 梅村雅之,“GPU･FPGA 複合型演算加速クラスタを用いた宇宙輻射輸送コード ARGOT の多ノード並列化”, 第 185 回ハイパフォーマンスコンピューティング研究発表会, 2022-07.
22. 瀬口知洋, 中井榛希, 山口佳樹, 藤田典久, 小林諒平, 朴泰祐, “並列化に伴うデータ空間の分割とそれによるアクセスパターンの変化がもたらす HBM の振る舞い調査”, SWoPP2022: 並列／分散／協調システムとディペンダブルコンピューティングおよび一般 2022-07.
23. 佐野由佳, 小林諒平, 藤田典久, 朴泰祐, “ノードを跨いだ GPU・FPGA 複合型演算加速による宇宙物理シミュレーションの実装と評価”, 第 184 回ハイパフォーマンスコンピューティング研究発表会, 2022-05.
24. 菊池航平, 藤田典久, 小林諒平, 朴泰祐, "並列 FPGA 環境における通信システム CIRCUS を用いた集団通信の実装と性能評価", 第 187 回ハイパフォーマンスコンピューティング研究発表会, 2022-12.

(4) 著書、解説記事等

なし

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）

- 「初期天体形成シミュレーションにおける GPU+FPGA 連携プログラミング及び実行に関する研究」計算科学研究センター・宇宙物理研究部門・梅村グループとの共同研究
- 素粒子物理研究部門と Japan Lattice Data Grid（JLDG）の構築，運用に関して連携を行っている。
- 「FPGA を活用したグラフ処理アプリケーションの高速化に関する研究」計算科学研究センター・計算情報学研究部門 (データ基盤分野)・天笠グループとの共同研究

国際連携・国際活動

- GPU・FPGA 協調プログラミングシステム MHOAT の開発において米国 Oak Ridge National Laboratory と共同研究
- 米国エネルギー省と文部科学省が締結しているシステムソフトウェアに関する共同研究契約により，主に米国アルゴンヌ国立研究所と共同研究を行っている。

8. シンポジウム、研究会、スクール等の開催実績

- Gfarm／実用アプリ・シンポジウム 2022，東京（ハイブリッド），2022 年 10 月 28 日
- Gfarm ワークショップ 2023，長崎（ハイブリッド），2023 年 2 月 10 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：理化学研究所客員主管研究員
3. 朴泰祐：PC クラスタコンソーシアム理事
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：「富岳」成果創出加速課題領域総括
6. 朴泰祐：「次世代計算基盤に関する調査研究」Program Director
7. 朴泰祐：HPCI 計画推進委員会委員
8. 朴泰祐：「富岳」課題推進委員会委員
9. 朴泰祐：HPCI 連携サービス委員会委員
10. 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
11. 高橋大介：理化学研究所客員主管研究員
12. 高橋大介：HPCI 利用研究課題審査委員会レビューアー

13. 高橋大介：HPCI 連携サービス運営・作業部会委員
14. 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
15. 建部修見：HPCI セキュリティインシデント即応委員会委員
16. 建部修見：HPCI 連携サービス運営・作業部会委員
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：情報通信研究機構協力研究員
19. 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
20. 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
21. 多田野寛人：日本シミュレーション学会　理事
22. 小林諒平：理化学研究所計算科学研究センター客員研究員
23. 小林諒平：計算科学研究センター計算機システム運用委員会委員
24. 小林諒平：計算科学研究センター次世代計算システム研究開発室室員
25. 藤田典久：理化学研究所計算科学研究センター客員研究員
26. 藤田典久：文部科学省研究振興局技術参与

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: Steering Committee Member, IEEE Cluster
3. Taisuke Boku: Steering Committee Member, International Conference on Parallel Processing
4. Taisuke Boku: 2022 ACM Gordon Bell Prize Committee Vice Chair
5. Taisuke Boku: Program Committee Member, 2022 International Conference on Parallel Processing
6. Daisuke Takahashi: The International Journal of High Performance Computing Applications Editor
7. Daisuke Takahashi: The 17th International Workshop on Automatic Performance Tuning (iWAPT 2022) Program Committee Member
8. Daisuke Takahashi: The 22nd International Conference on Computational Science and Its Applications (ICCSA 2022) Publicity Committee Member
9. Daisuke Takahashi: The International Conference on Computational Science (ICCS 2022) Program Committee Member
10. Daisuke Takahashi: The 36th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2022) Program Committee Member

11. Daisuke Takahashi: 9th International Workshop on Large-scale HPC Application Modernization (LHAM 2022) in Conjunction with 10th International Symposium on Computing and Networking (CANDAR'22) Program Committee Member
12. Daisuke Takahashi: The 20th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2022) Program Committee Member
13. Daisuke Takahashi: 24th IEEE International Conference on High Performance Computing and Communications (HPCC-2022) Program Committee Member
14. 高橋大介：情報処理学会論文誌査読委員
15. 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
16. Osamu Tatebe: Program Committee, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC22)
17. Osamu Tatebe: Program Committee, 36th IEEE International Parallel & Distributed Processing Symposium (IPDPS)
18. Osamu Tatebe: Program Committee, IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2022)
19. Osamu Tatebe, Program Committee, International Supercomputing Conference 2022
20. Osamu Tatebe: Workshop Chair, 3rd Workshop on Extreme-Scale Storage and Analysis (ESSA 2022)
21. Hiroto Tadano: Publication Co-Chair, The 41st JSST Annual International Conference on Simulation Technology (JSST 2022)
22. 多田野寛人：日本シミュレーション学会「非線形現象の高性能数値解析技術研究委員会」　委員
23. 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
24. Hiroto Tadano: Local Scientific Program Committee, 10th International Congress on Industrial and Applied Mathematics (ICIAM 2023)
25. 小林諒平：電子情報通信学会 コンピュータシステム研究専門委員会 幹事補佐
26. 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
27. 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
28. 小林諒平：電子情報通信学会 英文論文誌編集委員
29. 小林諒平：電子情報通信学会 ISS ソサイエティ誌編集委員
30. 小林諒平：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
31. 小林諒平：SWoPP 2022 組織委員長
32. 小林諒平：xSIG 2022 プログラム委員

33. Ryohei Kobayashi：HPCAsia2023 Proceedings Chair
34. Ryohei Kobayashi：HEART2022 Publication Chair
35. Ryohei Kobayashi：FPL2022 Publicity Co-chair
36. Ryohei Kobayashi：FPL2022 Program Committee Member
37. Ryohei Kobayashi：CANDAR 2022 Program Committee Member
38. Ryohei Kobayashi：CANDAR 2022 CSA workshop Program Committee Member
39. Ryohei Kobayashi：COOL Chips 25 Program Committee Vice Chair
40. 藤田典久：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
41. 藤田典久：電子情報通信学会論文誌コンピューティングシステム編集委員
42. Norihisa Fujita: Program Committee, The 23rd International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'22)
43. Norihisa Fujita: Program Committee, Auto-Tuning for Multicore and GPU 2022 (ATMG2022)