VII. 高性能計算システム研究部門

1. メンバー

教授	朴 泰祐，建部 修見，高橋 大介，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
主任研究員	前田 宗則
研究員	平賀 弘平
学生	大学院生 16 名，学類生 4 名，研究生 1 名
学内共同研究員	安永 守利，櫻井 鉄也，山口 佳樹，今倉 暁（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），後藤 仁志（豊橋技術科学大学）， 佐野 健太郎（理化学研究所），中尾 昌広（理化学研究所）， 天野 英晴（慶応義塾大学），川島 英之（慶応義塾大学）， 田中 昌宏（慶応義塾大学）

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，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，GPU・FPGA 利用技術，並列数値計算の高速化などの研究を行っている。今年度は以下の研究を行った。

- 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
- 超高速ストレージシステム・Gfarm ファイルシステムの研究開発
- 複数の整数除算の高速化
- 容易な GPU 化手法の検証と性能評価
- 高精度近似解を生成するブロッククリロフ部分空間反復法に関する研究
- グラフニューラルネットワークの学習精度と実行性能評価
- 並列 FPGA 環境における FPGA 間通信に関する研究
- 複数の演算加速装置を統一的に扱えるプログラミング環境に関する研究

3. 研究成果

[1] 多重複合型演算加速システムのプログラミングシステム（朴，小林，藤田）

以前より継続している 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コンセプト CHARM (Cooperative Heterogeneous Acceleration with Reconfigurable Multidevices) の下，演算加速向けの単一プログラミング言語 OpenACC のみによるプログラミングを可能とする言語処理系 MHOAT (Multi-Hetero OpenACC Translator) を開発した。本研究は理化学研究所計

算科学研究センター（R-CCS）及び米国 Oak Ridge National Laboratory（ORNL）との共同研究によるものである。

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。MHOAT は一種の source-to-source コンパイラであり，プログラム内の独自拡張 directive により，演算加速デバイスオフロード部分をどのデバイスに割り当てるかをユーザが指定できる。指定に基づき，各デバイスに行わせるべきプログラム断片を生成し，各デバイスに対応するバックエンドコンパイラを呼び出して処理させ，最終的なバイナリ断片をリンクして単一のバイナリ及び FPGA にローディングする bit stream を生成する。バックエンドコンパイラには，GPU 部分は NVIDIA 社の nvhpc，FPGA 部分は ORNL の研究用コンパイラ OpenARC を用いる。

令和 5 年度の成果として，実アプリケーションである宇宙物理初期天体シミュレーションを行う ARGOT プログラムを MHOAT 向けの OpenACC コードとして記述し，これを MHOAT でコンパイル，最終的に多重複合型演算加速スーパーコンピュータ Cygnus の単一ノードで GPU と FPGA を併用して実行することに成功した。

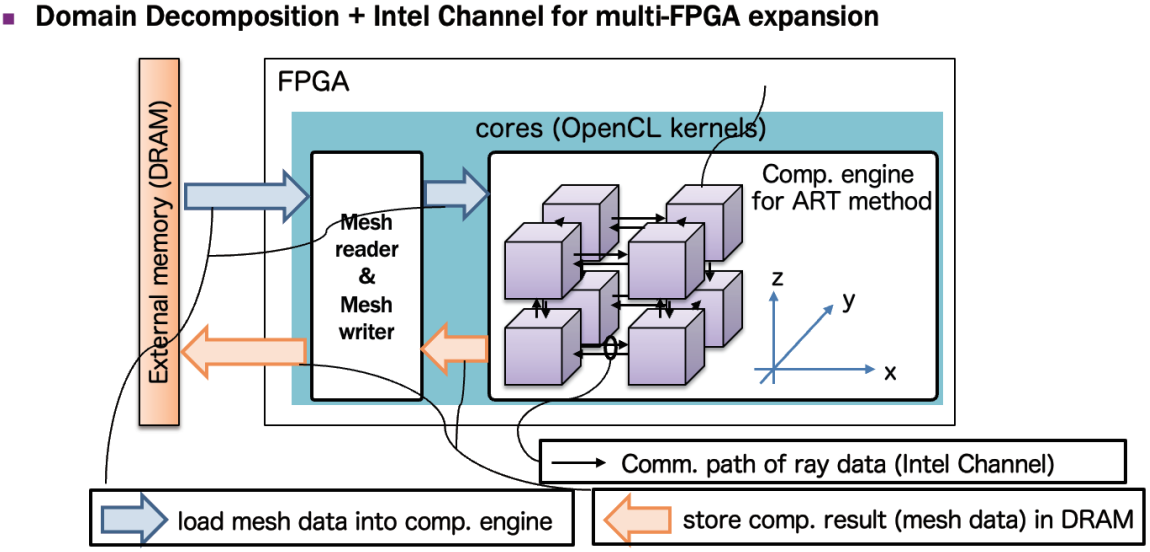


図 1 複数カーネルによる ART 法の実装イメージ

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。

ART 法を，OpenACC を用いて FPGA に実装する場合，性能を上げるために FPGA 上で複数のカーネルを同時実行し，domain decomposition 手法によって空間分割した部分計算を各カーネルに実行させる。この様子を図 1 に示す。図では OpenCL kernels となっているが，これは MHOAT が FPGA 向けの OpenACC コードを処理する際，OpenARC コンパイラを用いて OpenACC コードを OpenCL に変換し，最終的に Intel FPGA SDK for OpenCL ツールを用いて FPGA のバイナリを生成するためである。従来研究ではこの ART 法計算を OpenCL で直接記述していたが，MHOAT によりユーザは OpenACC というより高いレベルで記述することが可能になった。また，このような domain decomposition された問題空間のデータ交換が必要になるが，ここには Intel Channel という FPGA kernel 間通信機能が用いられるが，MHOAT ではこれを使うような独自拡張も備えられている。

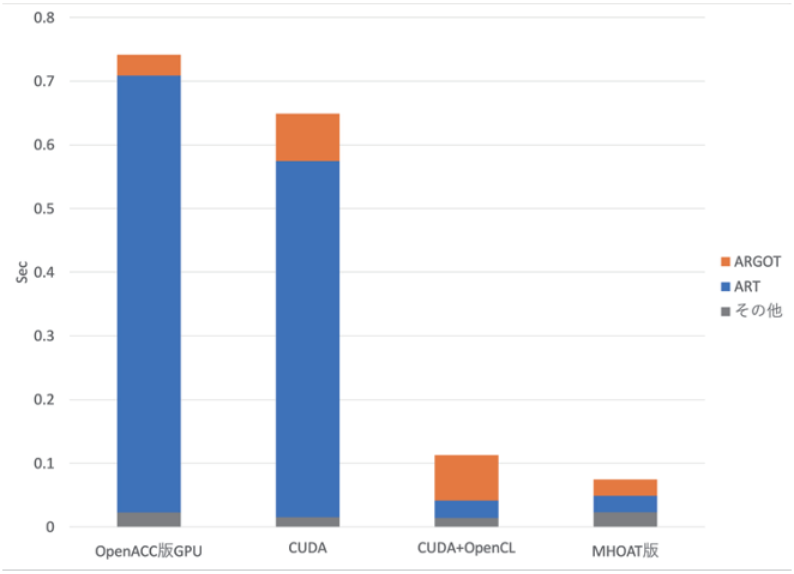


図 2 MHOAT による ARGOT コードの GPU+FPGA 実装の性能評価

図 2 に MHOAT による性能向上の様子を示す。評価したのは ARGOT コード全体の 1 タイムステップ実行時間で，全コードの OpenACC 版を GPU のみで実行した場合，同じく CUDA 版を GPU のみで実行した場合，CHARM コンセプトに基づく CUDA (GPU) + OpenCL (FPGA) を手作業で別にコーディングして実行した場合，OpenACC コードを MHOAT で処理し GPU+FPGA 実行した場合である。グラフより，MHOAT による実装が最も高速となっている。原理的には CUDA+OpenCL と同等になるはずであるが，ARGOT 法部分の GPU 実行性能において OpenACC 版の方が CUDA 版より高速であることからこのような結果となった。以上より，MHOAT による CHARM コンセプトのプログラム記述の有効性が，実アプリケーションである ARGOT への適用で証明された。

これらの研究成果は国際会議 International Workshop on HPC by Heterogeneous Hardware (H3) in ISC2023 及び情報処理学会 ACS 論文誌で発表済みである。

[2] FPGA 向け高性能 OpenACC コンパイラの開発（朴，小林，藤田）

前節で述べた CHARM コンセプトにおける GPU+FPGA 協調計算において，FPGA の処理は絶対性能において GPU よりも性能が劣ることは否めない。しかし，GPU が得意とする大規模な単純並列（データ並列）が適用できない場合は，ノード間通信が多発するケースでは FPGA の導入が効果的である。とはいえ，FPGA 自体の性能を HPC コードで十分引き出すことは必要であり，我々は理化学研究所計算科学研究センター（R-CCS）と共同でこの研究を進めている。この中で，高レベルな演算加速器向けプログラミングを可能とする OpenACC をターゲットに，FPGA の特性を活かすためのコーディングの工夫といくつかの機能拡張を独自 directive として実装した Omni OpenACC for FPGA compiler を開発している。同コンパイラは OpenACC コードを OpenCL に変換し，Intel FPGA SDK for OpenCL ツールによって最終的に FPGA 用の bit stream ファイルを生成し，同時にこれをコントロールするホスト CPU 用の C コードを生成する。

令和 5 年度の研究では，FPGA の持つ演算機能を最大限に利用するための空間並列性をいかに向上させるかに焦点を当て，ベクトル処理における loop unrolling と，domain decomposition と PGAS (Partitioned Global Address Space) の考え方に基づく kernel decomposition に関する研究を行った。空間並列性を増やすことは，1 クロック内での演算回数を増やすことになり，性能をダイレクトに向上させる。

Loop unrolling は一般的に用いられるベクトル処理高速化であり，コンパイラが生成する OpenCL には loop unrolling を指示する機能が備わっている。これを OpenACC レベルで利用するため，同様の directive を OpenACC 上で記述可能とした。また，reduction 演算が必要な場合に対し，中間変数を自動的に複製し，最後に reduction 演算を実行することで，FPGA の動作を円滑に行い，変数へのアクセスが性能ボトルネックとならないような最適化も行った。これにより，reduction 演算を伴う loop unrolling であっても FPGA 回路の Initiation Interval (II) が大きくならないようにでき，クロック速度に応じた性能が達成できる。しかし，loop unrolling を深くすると FPGA 上の Statick RAM である Block RAM (BRAM) の使用量が増大することがわかっており，これはある程度演算並列性が高まると BRAM へのアクセスがボトルネックとならないように Intel OpenCL コンパイラが自動的に BRAM

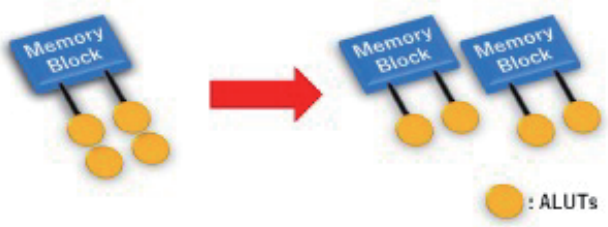


図 3 BRAM アクセスバンド幅向上のための
イメージコピー

の配列を複製し，バンド幅を向上させるためである（図 3 参照）。BRAM サイズには上限があり，DRAM のように FPGA 外で容量を増やすことができない。

そこで，kernel decomposition という独自手法を開発した。これは PGAS モデルの並列処理で domain decomposition を仮想配列上で行えることにヒントを得たものである。従来研究である Omni XcalableACC や UPS 等の PGAS 言語のスキームを OpenACC 内に取り入れ，単純なベクトルまたは行列計算を domain decomposition による複数カーネルに分散させ，これらを同時実行して空間並列性を向上させる手法である。必要なメモリを kernel に分散させ，その中で適度な loop unrolling を行うことにより，BRAM 容量を抑えつつ空間並列性を向上させることが可能となる。ただし，kernel decomposition を完全に自動で行うことは難しいため，Omni OpenACC for FPGA のプロトタイプコンパイラでは，XcalableACC で取り入れた分散配列に対する directive を取り入れ，kernel を一種の分散メモリノードと捉え，PGAS モデル向けの kernel 間通信や BRAM 間同期機能，配列参照インデックスの補助関数等を導入した。図 4 に kernel decomposition のイメージを示す。

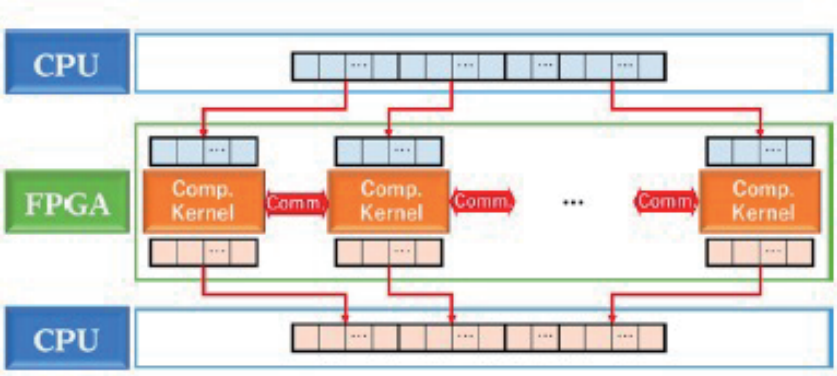


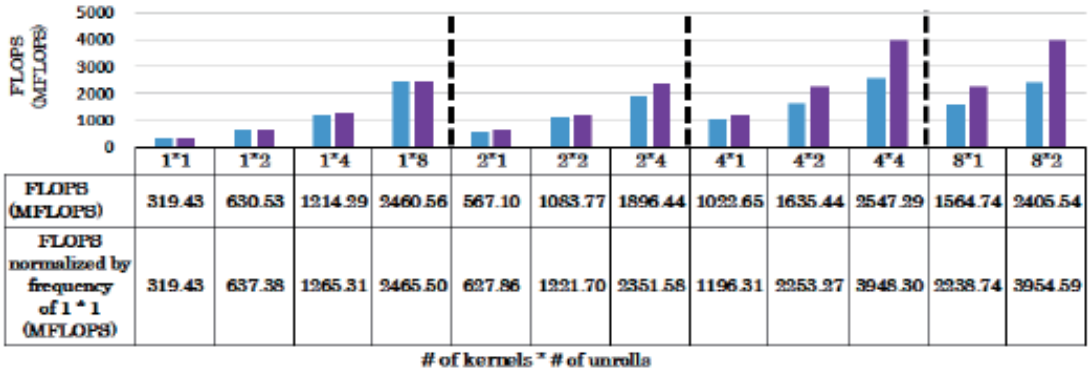
図 4 FPGA 内での kernel decomposition 手法による複数 kernel 同時実行

Omni OpenACC for FPGA プロトタイプコンパイラを用い，疎行列計算を CG 法で行うベンチマークコードをコンパイルし，Intel Straxi10 FPGA 上で評価した。結果を図 5 に示す。FLOPS は実際の性能を表し，図中では水色で示している。しかし，このように各種手法で FPGA 内の空間並列性を増すと，回路が複雑になり動作周波数が低下する。FPGA の動作周波数は回路合成を行う際にコンパイラにより決定されるため，これを手動によって直接高めることはできない。そこで,「仮に周波数が一定であったとしてどの程度空間並列性が向上し」性能が向上するか」を見るため，最も動作周波数が高い 1x1 の場合を基準に，性能を正規化してみたこれが normalized FLOPS で図中では紫色のバーで示している。結果として，4x4 または 8x2 という，空間並列性が 16 倍になったケースが normalized FLOPS ではほぼ同等の性能になるというリーズナブルな結果となった。しかし，動作周波数の低下により，実際の性能は空間並列性に線形には向上しないこともわかった。

一方，BRAM 使用量に着目すると，想定したように loop unrolling が深くなると BRAM 使用量が増えることが CG 法でも確認された。結果を図 6 に示す。例えば, 空間並列性が 8 の場合,

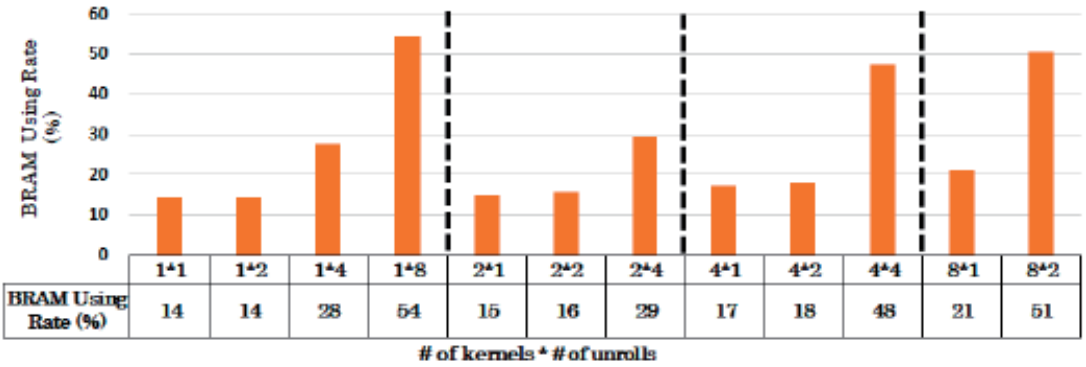
両手法の組み合わせとして 1x8，2x4，4x2 等が考えられる。図 5 ではこれらの 3 パターンの性能は少なくとも normalized FLOPS ではほぼ同等である。しかし，図 6 を見ると，BRAM 使用量は loop unrolling が浅いほど減っており，kernel decomposition によってメモリ使用量を削減しつつ性能を保つことが可能であることがわかった。さらに, OpenACC の directive という，ユーザにとって操作しやすい記述を変えるだけでこのような組み合わせを容易に試すことができ，性能チューニングとメモリ使用量限度というトレードオフを制御する機能が提供できたことも重要な成果である。

これらの結果は IEEE Cluster 2023 のポスター発表，及び HPC Asia 2024 のポスター発表でそれぞれ公開され，前者は最優秀学生ポスター賞にノミネートされた。また，現在これらの成果を国際会議向けに投稿中である。



	1*1	1*2	1*4	1*8	2*1	2*2	2*4	4*1	4*2	4*4	8*1	8*2
FLOPS (MFLOPS)	319.43	630.53	1214.29	2460.56	567.10	1083.77	1896.44	1022.65	1635.44	2547.29	1564.74	2405.54
FLOPS normalized by frequency of 1*1 (MFLOPS)	319.43	637.38	1265.31	2465.50	627.86	1221.70	2351.58	1196.31	2253.27	3948.30	2238.74	3954.59

図 5 Loop unrolling と kernel decomposition を組み合わせた CG 法の演算性能（m x n は kernel decomposition の kernel 分割数が m，loop unrolling の深さが n であることを表す）



	1*1	1*2	1*4	1*8	2*1	2*2	2*4	4*1	4*2	4*4	8*1	8*2
BRAM Using Rate (%)	14	14	28	54	15	16	29	17	18	48	21	51

図 6 CG 法における BRAM 使用量（m x n については性能評価の図と同じ）

[3] GPU+FPGA 複合計算向け oneAPI プログラミング（朴，小林，藤田）

Intel oneAPI は CPU を含むあらゆるプロセッサ向けに単一のコードでのプログラミングを可能とするフレームワークで，DPC++ 言語を基本とする。これは我々が進めている CHARM コンセプトに基づく GPU+FPGA 多重複合演算加速を実装する一つの理想的な環境と

考えられるが, 通常の oneAPI コンパイラ, ランタイム, ツールキットだけでは, 同一コードを, 複数のデバイスのうち「どれか 1 つ」に対してコンパイル・実行できるが，これらを混在させることは難しい。第三者より提供されている外部モジュールを導入し，コード断片のオフロード記述を修正することで対応できる。

しかし，oneAPI は誰でも Intel から正式な商用版を入手可能で，CHARM での使用法やモジュール拡張方法を公開することで適用可能であることから，その条件について整理し，実際に簡単なベクトル処理のようなトイプログラムを示すことでその適用性を評価した。令和 5 年度研究では，ベクトルに対する簡単な演算処理を行い，これを GPU 及び FPGA を併用するベンチマークコードとして記述し, Cygnus の単体ノード上で実行可能であることを示した。NVIDIA V100 GPU 及び Intel Stratix10 FPGA を用いた場合，カーネル部分の実行時間については妥当であったが，FPGA の起動部分に非常に時間がかかり，この原因究明と性能改善が今後の課題である。

本研究については HPC Asia 2024 内のワークショップ Intel eXtreme Performance Users Group (IXPUG) にて発表済みである。

[4] キャッシングファイルシステムの研究（建部）

これまでスーパー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，および並列キャッシングファイルシステムに拡張した CHFS/Cache の研究開発を行ってきた。CHFS では, 計算ノードの不揮発性メモリ, 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。これまでの評価により，高い性能と高いスケーラビリティを有することが示されている。CHFS/Cache では，この性能とスケーラビリティを損なうことなく，キャッシングファイルシステムの機能を持たせるための設計を行った。既存システムでは，並列ファイルシステムとの間で強い一貫性を維持するため，特に小さいファイルの処理に関して大きなオーバヘッドが生じていた。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の間の一貫性の緩和について考察し，設計を行った。これまでの評価により，小さいファイルの書込みにおいて性能が低下しないこと，キャッシュされたファイルについては性能低下なく読込が可能であることを示した。

令和 5 年度についてはこの並列キャッシングファイルシステムの高度化を行った。並列キャッシングファイルシステムでは, 書込んだデータを並列ファイルシステムに書き戻す（フラッシュする）必要があるが，この処理を行っている間，既存システムでは並列キャッシングファイルシステムのアクセス性能が低下する問題があった。この問題の原因は, フラッシュ処理と並列キャッシングファイルシステムのアクセスにおいて計算ノード上のストレージア

クセスの競合と，ネットワークアクセスの競合が起こるためである。この問題を解決するため，I/O-aware フラッシング方式の提案を行った。I/O-aware フラッシング方式では，各キャッシングファイルシステムのサーバにおいてキャッシングファイルシステムのアクセス状況のモニタリングを行う。アクセスがなくなったら一定時間待ち，まだアクセスがないようであればフラッシュする。フラッシュ中に，アクセスが再開したらフラッシュを中断する。この方式は，各サーバがそれぞれフラッシュの判断を行うため，中央サーバが不要で，サーバ数が増えてもオーバヘッドが増えることはない。また，典型的なアプリケーションにおいては，計算フェーズと I/O フェーズが交互に現れることから，この方式により I/O を行わない計算フェーズにフラッシュ処理が行われることが期待される。この方式のための実装方式を示し，研究開発を進めている CHFS/Cache に実装した。

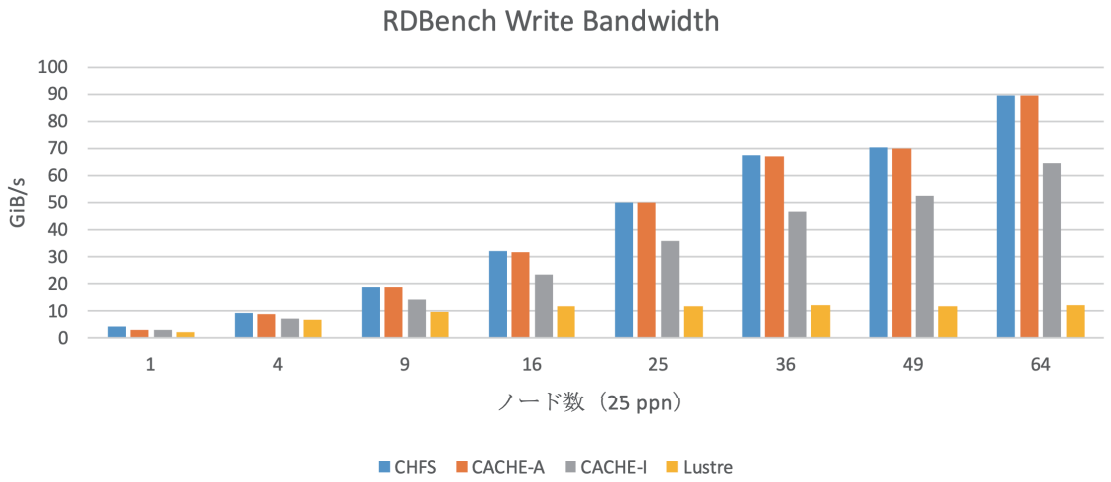


図 7 アプリケーションにおける書込み性能の評価

図 7 にアプリケーションベンチマークを用いた書込み性能を示す。実験は筑波大学のスーパーコンピュータ Pegasus を用い，各ノード 25 プロセスのウィークスケーリングの評価を実施した。Lustre は，直接並列ファイルシステムに書込んだ時の性能であるが，16 ノード以降は性能が向上していない。CHFS はフラッシュを行わない場合，CACHE-A は提案手法の I/O-aware フラッシング方式を用いた場合である。これらはノード数を増やすと性能が向上し，また CHFS と CACHE-A はほぼ同様の性能となっている。これにより I/O-aware フラッシング方式ではフラッシュにおける競合が回避できていることがわかる。一方，よく利用されるフラッシュ方式である書込んだ直後にフラッシュする方式の CACHE-I では競合により 30% ほどの性能低下となっている。これらの成果は国際ワークショップ REX-IO において発表した。

[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 共用ストレージの第 2 期の運用が 5 年目となり，HDD 障害が増えてきた。HDD の単体障害であれば，RAID により自動的に復旧可能であるが，同一 HDD グループにおいて 3 台以上の HDD に障害は発生するとデータロストが発生してしまう。HPCI 共用ストレージの場合，西拠点と東拠点に少なくとも一つずつ複製を持っているため，仮にどちらかの複製がロストしたとしても Gfarm が自動的に復旧することが可能である。そのため, 運用 5 年目となって障害頻度がかなり上がってもデータロストは発生しなかった。ただし注意すべき点として，サイレントデータ障害が複数発生したことがあげられる。サイレントデータ障害は HDD 等でエラーを発生することなく，データが壊れている障害である。Gfarm では，すべてのデータについてダイジェスト（HPCI 共用ストレージでは MD5 ダイジェスト）を保持しており，書込み後一定時間たつと読み込んでそのダイジェストと比較を行う。この処理は書込みベリファイと呼ばれるが，今回はこの処理において複数のサイレントデータ障害が見つかった。いずれのケースにおいても正しい複製が存在しており，データロストには至らなかった。サイレントデータ障害が発生した場合においては，ストレージシステムにおける不具合が疑われるため，その調査を行った。結果，その直接の原因は解明できなかったが，予防保守などで対応を図った。早期のストレージシステムのハードウェアの更新が望まれる。

また，ネットワーク障害によるファイル作成時の複製作成失敗に伴う，再複製作成までの時間短縮を図った。HPCI 共用ストレージでは，西拠点と東拠点に少なくとも一つずつファイル複製を作成している。一方で，西拠点と東拠点の間のネットワークにおいてたまにファイル転送速度が大きく落ち，接続エラーになってしまうケースがある。その場合，そのファイルの複製を再び作成する必要があるが, その再作成の始動に時間がかかってしまうことがあった。本来この問題はネットワーク側で対処するべきものと考えるが，ネットワーク側の原因追及は難しくここ数年調査を続けているがまだ特定できていない。そのため，この問題を解決すべく Gfarm 側の機能強化を図った。当初は，並列複製作成コマンドに対し，機能拡張することを考えていたが，その方法より Gfarm のメタデータサーバにおいて複製作成に失敗したファイルを優先して再び複製作成する対応の方が，機能面，実装面において優れていることが判明したため，その機能整備を行った。これまでは複製作成に失敗したファイルはメタデータサーバの複製チェックプロセスにおいて複製を再作成していたが，とくに複製作成に失敗したファイルを優先的に作成しないため，結果として複製の再作成に時間がかかっていた。本機能は Gfarm バージョン 2.8.1 においてリリースされ，すでに HPCI 共用ストレージにおいて利用されている。

さらに，GSI のサポート終了に伴う次期セキュリティ基盤対応として，OAuth2 のトークンを用いた認証基盤の高度化を行った。具体的には，昨年度までの実装では，TLS クライアント証明書認証，また OAuth2 のトークンを用いた認証において，サーバ側，クライアント側のどちらか，あるいはその双方で認証に失敗すると接続が切断してしまい，その次の認証方

式を試さない問題があった。この問題を解決するために，それぞれのエラーケースにおける処理を適切にするよう機能整備を行った。成果は Gfarm バージョン 2.8.3 でリリースし，すでに HPCI において配備が完了している。また，昨年度までに OAuth2 のアクセストークンの有効期限を更新するためのサーバとして jwt-server を開発してきたが，このサーバが冗長構成をとることができず，障害が発生するとアクセストークンの更新ができなくなってしまう問題があった。そのため，jwt-server の冗長構成を可能とするよう開発を行った。成果は https://github.com/oss-tsukuba/jwt-server および https://github.com/oss-tsukuba/jwt-agent において公開しており，すでに HPCI において配備が完了している。

[6] 複数の整数除算の高速化（高橋）

整数除算は広く使われている算術演算であり，多くのアプリケーションに含まれている。一般的に，除算は加算，減算，および乗算よりも遅いことが知られている。このため，加算，減算，および乗算のみで除算を行うアルゴリズムが提案されている。特に，不変な整数による除算のアルゴリズムや，整数による除算のアルゴリズム，そして逆数を用いた除算が提案されている。さらに，倍精度浮動小数点演算を用いた 32 ビットおよび 64 ビットの整数除算も提案されている。

本研究では，被除数が不変であり，複数の除数が単調増加または単調減少する場合の複数の整数除算を考える。このような複数の整数除算は，整数を複数の法の Montgomery 表現に変換する際に現れる。

符号なし整数の除算は，商$q = \lfloor a/b \rfloor$と剰余$r = a - bq$ $(0 \le r < b)$で定義される。ここで被除数は a であり除数は b である。被除数が不変であり，複数の除数が単調増加または単調減少する場合の複数の整数除算において，被除数と除数がある条件を満たす場合，最初に除算によって商を 1 つだけ計算すれば，残りの商は以前に計算された商を最大 1 回補正することによって得られることを示す。

定理 1. 整数 $a \ge 0, b > 0$ および $c \ge 0$ を考える。もし $ca \le b(b+c)$ であれば

$$\left\lfloor \frac{a}{b} \right\rfloor < \left\lfloor \frac{a}{b+c} \right\rfloor + 2 \qquad (1)$$

証明 任意の実数 x について，$x - 1 < \lfloor x \rfloor \le x$が成り立つ。よって

$$\frac{a}{b} - 1 < \left\lfloor \frac{a}{b} \right\rfloor \le \frac{a}{b}, \qquad (2)$$

$$\frac{a}{b+c} + 1 < \left\lfloor \frac{a}{b+c} \right\rfloor + 2 \le \frac{a}{b+c} + 2 \qquad (3)$$

不等式 (2) と (3) より，不等式 (1) が成立するための条件 $a/b \le a/(b+c)+1$ は $ca \le b(b+c)$ となる。（証明終わり）

定理 1 より，a, b, および c が $ca \leq b(b+c)$ の条件を満たす場合，商$\lfloor a/b \rfloor$は商$\lfloor a/(b+c) \rfloor$ に対して最大 1 回の補正を加えることで得られる。したがって，定理 1 は除数が単調減少する際に適用できる。除数が単調増加する場合は，除数の順序を逆にすることで同様にして商を得ることができる。

Algorithm 1 Multiple integer divisions with an invariant dividend and monotonically increasing divisors

Input: $a \geq 0,\ b_0, b_1, \ldots, b_{n-1} > 0,\ b_i \leq b_{i+1}$ for $0 \leq i \leq n-2,\ ca \leq b_0(b_0+c)$, where $c = \max(b_{i+1} - b_i)$ for $0 \leq i \leq n-2$

Output: $q_i = \lfloor a/b_i \rfloor$ for $0 \leq i \leq n-1$, $r_i = a \bmod b_i$ for $0 \leq i \leq n-1$

1: $q_{n-1} \leftarrow \lfloor a/b_{n-1} \rfloor$
2: $r_{n-1} \leftarrow a - b_{n-1}q_{n-1}$
3: **for** i **from** $n-2$ **downto** 0 **do**
4: $\quad q_i \leftarrow q_{i+1}$
5: $\quad r_i \leftarrow a - b_i q_i$
6: $\quad$ **if** $r_i \geq b_i$ **then**
7: $\quad\quad q_i \leftarrow q_i + 1$
8: $\quad\quad r_i \leftarrow r_i - b_i$.

図 8 不変な被除数と単調増加する除数に対する複数の整数除算

不変な被除数と単調増加する除数に対する複数の整数除算のための提案アルゴリズムを図 8 の Algorithm 1 に示す。被除数 a，最小の除数 b_0，隣接する除数間の最大差 c について定理 1 の条件が満たされることがあらかじめ分かっていれば，Algorithm 1 を用いて複数の整数除算を効率的に計算することができる。

Algorithm 1 の 1 行目と 2 行目は，被除数 a と除数 b_{n-1} に対して，それぞれ商 q_{n-1} と剰余 r_{n-1} を計算する。4 行目では，以前に計算された商 q_{i+1} を近似商として用いる。5 行目では，剰余 r_i が被除数 a，除数 b_i，および商 q_i から求められる。剰余 r_i が除数 b_i 以上である場合，7 行目と 8 行目で商 q_i と剰余 r_i を補正する。

Algorithm 1 と同様にして，被除数が不変で単調減少する除数を持つ複数の整数除算に対するアルゴリズムを実現することができる。整数除算では，商や剰余だけを計算したい場合がある。このような場合，Algorithm 1 を修正して，必要な結果のみを出力することができる。

さらに，Algorithm 1 のベクトル化について考える。Algorithm 1 において，以前に計算された商を補正することによって次の商が得られるが，この処理はデータ依存性のためにベクトル化することができない。しかし，以前に計算された複数の商を同時に補正することで，このデータ依存性を解決し，処理をベクトル化することが可能になる。そのために，ベクトル長の倍数における除数の既に計算された商を近似商として用いる。したがって，Algorithm 1 における c の値は，ベクトル長の間隔における除数間の差の最大値となる。不変な被除数と単調増加する除数を持つ複数の 64 ビット符号なし整数除算をベクトル化した例を図 9 に示す。

```
void vdivinc64(uint64_t *q, uint64_t *r, uint64_t a, uint64_t *b, int n)
/*  Compute q[0:n] = floor(a / b[0:n]) and r[0:n] = a mod b[0:n].
    Requires a >= 0, b[0:n] > 0, b[i] <= b[i + 1] for 0 <= i <= n - 2
    and c * a <= b[0] * (b[0] + c),
    where c = max(b[i + VLEN] - b[i]) for 0 <= i <= n - VLEN - 1. */
{
  int i, j;

  for (i = n - 1; i >= n - VLEN; i--) {
    q[i] = a / b[i];
    r[i] = a - b[i] * q[i];
  }

  for (j = n - VLEN - 1; j >= 0; j -= VLEN) {
    for (i = j; i >= j - VLEN + 1; i--) {
      q[i] = q[i + VLEN];
      r[i] = a - b[i] * q[i];
      if (r[i] >= b[i]) {
        q[i]++;
        r[i] -= b[i];
      }
    }
  }
}
```

図 9 不変な被除数と単調増加する除数を持つ複数の 64 ビット符号なし整数除算をベクトル化した例

性能評価として，以下の 4 つの実装を用いて，複数の 64 ビット符号なし整数除算の実行時間を比較した。

1. Intel 64 アーキテクチャの 64 ビット符号なし整数除算命令 div を用いた実装
2. Intel 64 アーキテクチャ命令を用いた提案アルゴリズムの実装
3. Intel Short Vector Math Library（SVML）の packed 64 ビット符号なし整数除算を計算する組み込み関数 _mm512_div_epu64() を用いた実装
4. Intel Advanced Vector Extensions 512（AVX-512）命令を用いた提案アルゴリズムの実装

評価には以下の設定を用いた。64 ビット符号なし整数除算のバッチサイズ n は 128 から 1024 の間で変化させた。これらの場合，除数，商，剰余は L1 キャッシュに収まる。被除数 a は $[0,2^{64}\text{-}1]$ の範囲の乱数に設定される。

64 ビット符号なし整数除算の各バッチは 100 万回実行した。1 秒あたりの 64 ビット符号なし整数除算の回数（Mops）は，平均経過時間に基づいて計算した。

実行環境として，Intel Xeon Platinum 8368 プロセッサを用いた。今回の評価ではベクトル化に焦点を当てるため，シングルコア，シングルスレッドで性能を評価した。Intel

C Compiler（version 19.1.3.304）を使用した。コンパイラのオプションは icc -O3 -xCORE-AVX512 -qopt-zmm-usage=high を用いた。

表 1 不変な被除数と単調増加する除数に対する複数の 64 ビット符号なし整数除算の性能（Mops）

Batch size n	Intel 64	Proposed (Intel 64)	Intel SVML	Proposed (Intel AVX-512)
128	336.622	507.979	600.133	1110.869
256	338.892	503.765	604.713	1069.585
512	339.121	505.398	603.699	1068.396
1024	338.809	505.617	604.131	1065.383

表 1 は，不変な被除数と単調増加する除数に対する複数の 64 ビット符号なし整数除算の性能を示している。バッチサイズ n が 1024 の場合，提案アルゴリズムは，Intel Xeon Platinum 8368 プロセッサにおける Intel 64 アーキテクチャの 64 ビット符号なし整数除算命令および Intel SVML よりも，それぞれ約 1.49 倍および約 1.76 倍高速である。Intel Xeon Platinum 8368 プロセッサの Ice Lake マイクロアーキテクチャは，Skylake マイクロアーキテクチャなどの以前の Intel マイクロアーキテクチャと比較して，Intel 64 アーキテクチャ整数除算命令に必要なサイクル数が大幅に少ないため，Intel Xeon Platinum 8368 プロセッサでは，提案アルゴリズムと Intel 64 アーキテクチャの 64 ビット符号なし整数除算命令との性能差が小さくなっている。

[7] 容易な GPU 化手法の検証と性能評価（額田）

コストパフォーマンスや電力効率に優れる GPU を導入するスーパーコンピュータは増加し続け，現在では主流となっている。一方で各アプリケーションを GPU に対応するように改変する必要があり，このアプリの数だけある作業は膨大な人的コストを要する。NVIDIA 社製 GPU の場合では CUDA で実装されたコードが GPU の全ての機能を利用でき最高の性能を発揮できるのに対して，GPU で実行する部分を別のカーネル関数として抜き出す必要がある等プログラムに多くの修正を必要とする。より平易な方法として CPU 用のコードにディレクティブを挿入することによってコンパイラが GPU 用のコードを生成する OpenACC 等が導入され，GPU 化対象の計算部分がループ構文などで実装されている場合に非常に容易に GPU 化が可能である。昨年度に地震波の伝播シミュレーションコード OpenSWPC を do concurrent 構文での GPU 化を実施したが，Fortran 2008 の標準並列化構文である do concurrent で実装されたマルチスレッド実行用 CPU コードから一切の改変なしで GPU 用コードを生成することができる。特に allocatable 属性の配列には CPU と GPU の両方からアクセスできる managed

memory が割り当てられることにより自動的に CPU と GPU 間の転送が必要最低限の回数行われ，また明示的に転送を指定する必要がないことでコーディングの間違いも生じにくい。一方で managed memory も万全ではなく，特に CPU と GPU 間をページ単位で移動する性質から MPI などのネットワーク通信用のバッファメモリとしては適しておらず性能低下の原因となる。このため通信バッファだけは device memory を利用するために CUDA Fortran と OpenACC の機能を用いて対応する必要があった。

今年度は do concurrent による手法の他に OpenACC，OpenMP(target)，及び CUDA Fortran の各手法による実装を行い，それらのコーディングコスト及び性能評価を行った。いずれも MPI 通信のバッファには device memory を利用する。

まず OpenACC では device memory を利用するため managed memory は利用せずに実装する。このため CPU と GPU 間のデータ転送を手動で制御する必要がある。OpenSWPC ではメインループに入る箇所で GPU が参照する配列を全て GPU に転送するもので, 定期的にスナップショットを保存する処理以外ではデータを device memory に置き続けることが可能であり, またそうすることが性能向上のために必須である。このため managed memory を利用する場合と比べると CPU と GPU 間のデータ移動を全て把握する必要がある点でコード分析のコストが増加する。アプリケーションを実際に何度も実行することが許容される状況であれば実行された転送が行われなくなるようにディレクティブを追加していくことでも対応可能である。

OpenMP では GPU 等への対応に target ディレクティブを利用する。GPU を利用する時にも CPU 側での処理にもマルチスレッド並列を利用するような場合にも全て OpenMP で実装することができる。また NVIDIA 社製の GPU だけでなく AMD 社，Intel 社の GPU や他のアクセラレータデバイスへの対応できることを考えると今後期待が高まっている手法である。構文やディレクティブの名前等に差はあるが OpenACC で実装されている主要な機能は全て OpenMP target に実装されているため OpenACC とほぼ同等の移植コストになる。

do concurrent については既に述べた通り CPU 用のコードを do concurrent 構文で実装してあればそのまま利用でき，do ループやその OpenMP 並列化されたものから do concurrent 構文への修正も作業量は多いかもしれないが単純作業である。また MPI 化されたアプリケーションでは通信部分だけ device memory か host memory を利用するように対応しなければ通信性能が低下する。これには非常に僅かではあるが CUDA Fortran と OpenACC の知識を必要とする。

最後に CUDA Fortran による実装であるが，ここでは CUDA C/C++ のようにカーネルを関数として分離する方法ではなく，ディレクティブによって GPU カーネルを使用する方法を提案する。さらにコンパイル時に allocatable 属性の配列に対して managed memory を利用するオプションを追加することによって他の手法と同等の移植コストで実現可能である。データ転送を解析する必要がなく，通信部分について device memory を指定するだけで完了するため CUDA Fortran の機能だけを使用している。通信部分について CPU 用でも利用できるコード

は以下のように実装する。コンパイラで CUDA Fortran が有効化されている場合のみ **!@cuf** から始まる行が有効になり，**!@cuf kernel** ディレクティブで GPU カーネルが生成される。

```
real,allocatable :: sbuf(:)
!@cuf attributes(device) :: sbuf
!$cuf kernel do
do concurrent (k=kbeg:kend)
    sbuf(k) = vx(k)
end do

call MPI_Isend(sbuf,…)
```

OpenACC (ACC)，OpenMP (OMP)，do concurrent (STD)，CUDA Fortran (CUF) のそれぞれで実装したコードの性能を Pegasus で比較したものを図 10 に示す。4 GPU を必要とする小田原地震のデータを使用し，データサイズ固定で最大 64 GPU を使用する Strong scaling の評価である。OpenACC と OpenMP の実行時間はそれぞれ GPU 数によって変動はあるが，総じて大きな差はないと言える。do concurrent についてもそれに準ずる性能を実現しており，移植コストの低さを考慮すると十分に有力な候補となりうる。最後の CUDA Fortran だけ他より著しく性能が低くなっている。GPU カーネルの実行時間が伸びていて，他の手法より生成された GPU カーネルの使用レジスタ数が多くなっていることがこの性能低下の一因と考えられる。検証のため比較的簡単なコードを使用した場合にはレジスタ数及び性能に大きな差異は見られず，OpenSWPC のような複雑なカーネルを利用する場合にのみこの問題が露呈するようである。CUDA Fortran は移植コストの低さから有望な手法ではあるが，少なくとも現状では

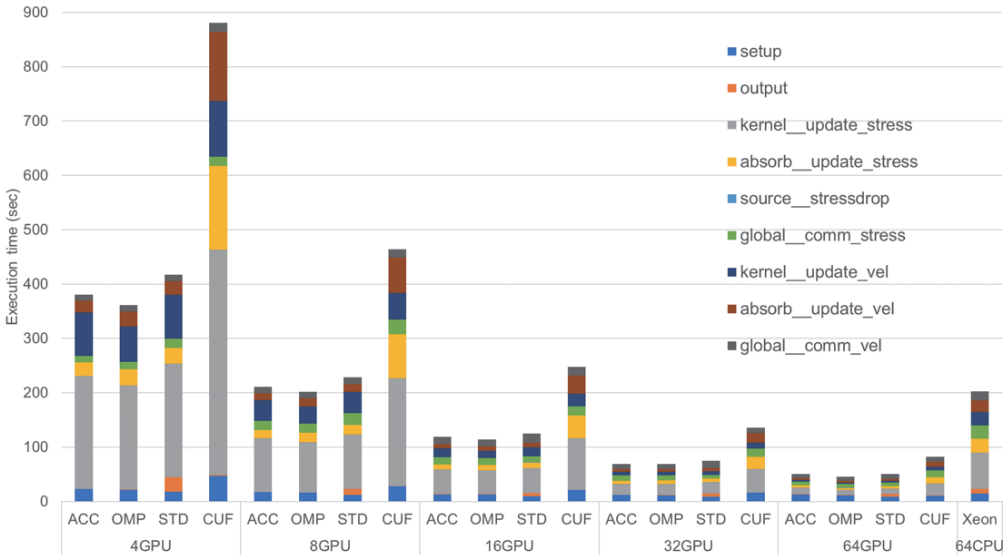


図 10 Pegasus 上での OpenSWPC の実行時間比較

GPU カーネルの性能に疑念があり今後のコンパイラの改善を期待する。図 10 の一番右は 64 ノードの Sapphire Rapids CPU のみを用いた場合である。およそ 8 GPU での性能に相当し，H100 GPU1 基あたり 8 CPU 分の性能を実現していることが分かる。

[8] 高精度近似解を生成するブロッククリロフ部分空間反復法に関する研究（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：

$$AX = B$$

は素粒子物理学分野の物理量計算，疎行列に対する固有値解法の内部問題，及び近年我々が研究を行っている鞍点型連立一次方程式に対する階層並列型数値解法の内部問題などにおいて現れる。同方程式の求解は計算時間の大きな割合を占め，得られた近似解の精度はアプリケーションの精度にも影響を及ぼすため，高速・高精度数値解法が必要とされている。同方程式に対する数値解法として，ブロッククリロフ部分空間反復法がある。同法は，右辺ベクトル数 L が増加すると求解に要する反復回数が減少する傾向がある一方で，得られる近似解精度が劣化する傾向がある。我々はこれまでの研究において，高精度近似解を生成できるブロッククリロフ部分空間反復法の開発を行ってきた。精度改善の手法として，1) 漸化式の再構築による誤差の抑制，2) Group-wise 更新と呼ばれる手法の適用による誤差の抑制，の 2 つに大別できる。近年我々は，2) の手法をブロッククリロフ部分空間反復法の一解法である Block BiCGSTAB 法に適用した Block GWBiCGSTAB 法を提案した。同法では s 回の反復における漸化式更新量をグループ化して精度劣化を引き起こす計算を回避することにより，高精度近似解を生成している。しかしながら，パラメータ s の与え方は解法の収束性，近似解精度に大きな影響を及ぼすとともに，係数行列 A によって最適な値が異なる。令和 5 年度は，収束性が優れた解法として知られている Block GPBiCG 法に対して Group-wise 更新手法を適用するとともに，グループ化周期を固定せず，解法内の情報を用いてグループ化周期を可変的に制御する手法を構築した。

複数右辺連立一次方程式 $AX = B$ の第 $(k+1)$ 番目の近似解を X_{k+1} とすると，残差行列 R_{k+1} は $R_{k+1} = B - AX_{k+1}$ と表される。X_{k+1} を用いて計算される残差行列は真の残差と呼ばれる。Block GPBiCG 法では，近似解 X_{k+1} と残差行列 R_{k+1} の更新量をそれぞれ ΔX_k, ΔR_k とすると，以下の漸化式で近似解と残差行列を計算できる。

$$X_{k+1} = X_k + \Delta X_k, \qquad \Delta X_k = P_k\,\alpha_k + Z_k,$$
$$R_{k+1} = R_k + \Delta R_k, \qquad \Delta R_k = -(AP_k)\alpha_k - \eta_k Y_k - \zeta_k\,(AT_k).$$

ここで，P_k, Z_k, Y_k, T_k は $n \times L$ 行列，α_k は $L \times L$ 行列，ζ_k, η_k はスカラーを表す。漸化式更新量 ΔX_k と ΔR_k の間には $-A$ 倍の関係：$\Delta R_k = -A\Delta X_k$ が成り立つが，有限精度の計算では誤差の影響によりこの関係式が崩れ，漸化式の残差行列と真の残差行列の間にギャップが生じる。

本研究では，Group-wise 更新手法を Block GPBiCG 法に適用することで誤差の影響の低減を図る。この手法を Block GWGPBiCG 法と名付けた。同法は外部反復と内部反復で構成さ

れ，内部反復において近似解の更新量がグループ化される。$\pi(m)$ を m 番目の外部反復開始時の反復番号，j を内部反復の反復番号とすると，反復番号 k は $k = \pi(m) + j$ と表される。近似解 $X_{\pi(m)+j+1}$ の漸化式は以下のように書き直される。

$$\begin{aligned} X_{\pi(m)+j+1} &= X_{\pi(m)+j} + P_{\pi(m)+j}\alpha_{\pi(m)+j} + Z_{\pi(m)+j} \\ &= X_0 + \sum_{i=\pi(0)}^{\pi(1)-1}(P_i\alpha_i + Z_i) + \cdots + \sum_{i=\pi(m-1)}^{\pi(m)-1}(P_i\alpha_i + Z_i) + \sum_{i=\pi(m)}^{\pi(m)+j}(P_i\alpha_i + Z_i). \end{aligned}$$

一方，残差行列 $R_{\pi(m)+j+1}$ の漸化式は以下のように書き直される。

$$R_{\pi(m)+j+1} = R_0 - A\left[\sum_{i=\pi(0)}^{\pi(1)-1}(P_i\alpha_i + Z_i)\right] - \cdots - A\left[\sum_{i=\pi(m-1)}^{\pi(m)-1}(P_i\alpha_i + Z_i)\right] - A\left[\sum_{i=\pi(m)}^{\pi(m)+j}(P_i\alpha_i + Z_i)\right].$$

上記のように本手法においては，グループ化された更新量に対して行列 A を乗じることにより，漸化式の残差行列と真の残差行列の間のギャップ低減を図っている。グループ化された更新量を表す $n \times L$ 補助行列：

$$V_{j+1}^{(m)} \equiv \sum_{i=\pi(m)}^{\pi(m)+j}(P_i\alpha_i + Z_i) = V_j^{(m)} + P_{\pi(m)+j}\alpha_{\pi(m)+j} + Z_{\pi(m)+j}$$

を導入すると，近似解 $X_{\pi(m)+j+1}$ と残差行列 $R_{\pi(m)+j+1}$ の漸化式は以下のように表される。

$$\begin{aligned} X_{\pi(m)+j+1} &= X_{\pi(m)} + V_{j+1}^{(m)}, \\ R_{\pi(m)+j+1} &= R_{\pi(m)} - AV_{j+1}^{(m)}. \end{aligned}$$

右辺ベクトル数 L が大きい場合は，数値的に不安定な状態に陥りやすく，残差の収束性が悪化する。この不安定性は残差行列に対して正規直交化を施すことで改善することが知られており，正規直交化の適用は実用上必要不可欠となっている。残差行列 R_k を $R_k = \acute{R}_k\xi_k$ と thin QR 分解する。ここで，$\acute{R}_k$ は $n \times L$ 列直交行列，ξ_k は L 次上三角行列である。この残差行列の関係式を全ての漸化式に適用して変形することで，数値的に安定なアルゴリズムが得られる。この手法を Block GWGPBiCGrQ 法と呼ぶ。同法の計算過程では，以下の thin QR 分解が実行される。

$$\acute{R}_{\pi(m)+j+1}\tau_{j+1}^{(m)} = \mathrm{qr}\left(\acute{R}_{\pi(m)} - A\acute{V}_{j+1}^{(m)}\right).$$

ここで，$n \times L$ 行列 $\acute{V}_{j+1}^{(m)}$ と L 次行列 $\tau_{j+1}^{(m)}$ はそれぞれ $\acute{V}_{j+1}^{(m)} \equiv V_{j+1}^{(m)}\xi_{\pi(m)}^{-1}$, $\tau_{j+1}^{(m)} \equiv \xi_{\pi(m)+j+1}\xi_{\pi(m)}^{-1}$ で定義される。また，$\mathrm{qr}(\cdot)$ は行列の thin QR 分解を表す。

Block GWGPBiCGrQ 法の内部反復の停止条件を設定する必要がある。単純な停止条件として，内部反復の最大反復回数を s 回に固定することが考えられる。これを固定グループ化と

呼ぶ。s はユーザが設定するパラメータであるが，この設定は解法の収束性や得られる近似解精度に大きな影響を及ぼす。具体的には，s を小さく設定すると収束性は良いものの得られる近似解精度が若干悪くなり，s を大きく設定すると近似解精度は向上するが収束性が悪化する傾向がある。本研究ではこの収束性の悪化について調査を行い，収束性が悪化するときは thin QR 分解で生じる行列 $\tau_j^{(m)}$ の条件数が増加する傾向にあることがわかった。したがって，行列 $\tau_j^{(m)}$ の条件数を内部反復の停止条件に利用することで，収束性の悪化を防ぐことができると考えられる。またこれまでの研究において，残差ノルムが大きい状態で内部反復が終了すると，近似解精度が悪化することがわかっている。本研究では，収束性と近似解精度の両方を改善するために，以下の内部反復の停止条件を提案した。

$$\textbf{If } \mathrm{cond}\left(\tau_{j+1}^{(m)}\right) \geq \theta \text{ and } \frac{\left\|R_{\pi(m)+j+1}\right\|_{\mathrm{F}}}{\left\|R_{\pi(m)}\right\|_{\mathrm{F}}} \leq 1, \textbf{then exit}$$

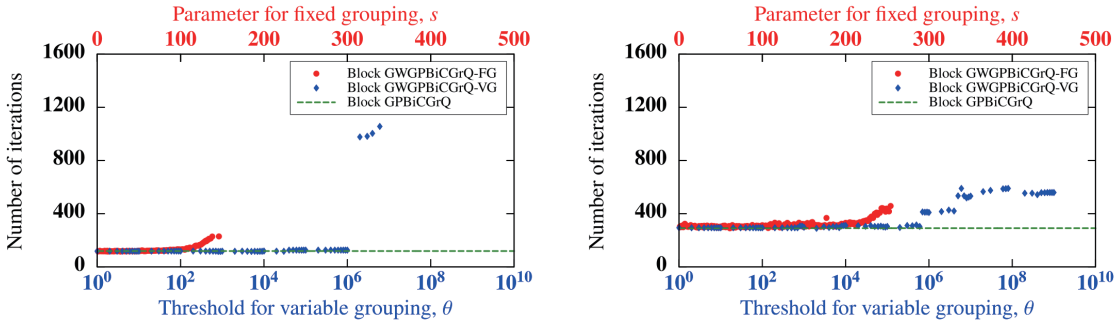
ここで，$\theta \geq 1$ は閾値であり，cond(·) は行列の条件数を表す。2 つ目の条件は，残差ノルムが反復開始時の残差ノルム以下になった場合のみ内部反復を停止することを保証している。上記停止条件を用いた場合は内部反復の反復回数が一定ではなくなるため，このグループ化手法を可変的グループ化と呼ぶ。Block GWGPBiCGrQ 法に対して固定グループ化，可変的グループ化を適用した手法をそれぞれ，Block GWGPBiCGrQ-FG 法，Block GWGPBiCGrQ-VG 法と名付けた。

数値実験によって提案手法の性能を評価する。係数行列 A として，SuiteSparse Matrix Collection で公開されている行列 torso3（サイズ n：259,156，非零要素数：4,429,042）と FEM_3D_thermal2（サイズ n：147,900，非零要素数：3,489,300）を用いた。右辺ベクトル数 L は 50 とし, 右辺項 B は乱数で与えた。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」の 1 ノードを用い，OpenMP を用いて 24 スレッド並列で計算した。Block GWGPBiCGrQ-FG 法のパラメータ s は s = 1, 2, …, 500 とし，Block GWGPBiCGrQ-VG 法のパラメータ θ は以下で設定した。

$$\theta = a \times 10^b \quad (a = 1.0, 2.0, \ldots, 9.0;\ b = 0, 1, \ldots, 9).$$

反復法の収束判定条件は相対残差ノルム $\| R_k \|_{\mathrm{F}} / \| B \|_{\mathrm{F}}$ が 10^{-15} 以下になった場合とした。なお，以下に示す図 11 〜図 13 では，収束判定条件が満たされた場合ケースのみをプロットした。

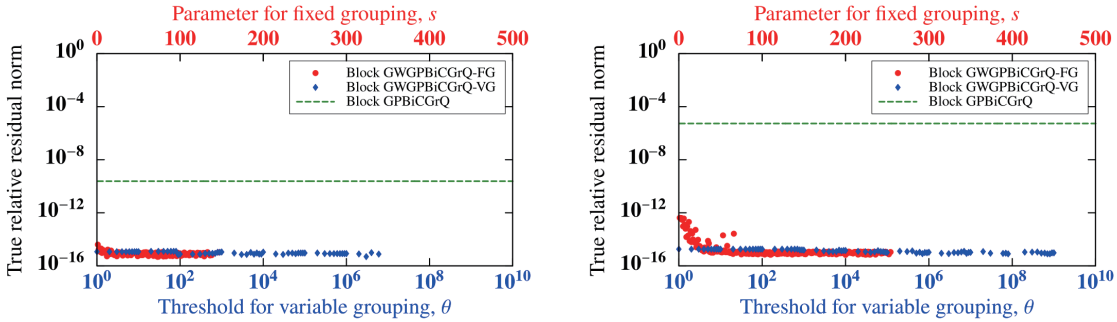
図 11 に Block GPBiCGrQ 法と Block GWGPBiCGrQ 型解法の反復回数を示す。Block GWGPBiCGrQ 型解法のパラメータが小さい場合は，Block GPBiCGrQ 法とほぼ同じ反復回数であった。しかしながら，パラメータを大きく反復回数が増加する傾向にあった。Block GWGPBiCGrQ-FG 法では，係数行列が異なると求解可能なパラメータ s の範囲が異なっている。一方 Block GWGPBiCGrQ-VG 法では，両係数行列においてパラメータ θ が 10^{-6} よりも小さい場合は反復回数がほぼ一定であった。したがって，可変的グループ化は固定グループ化よりも係数行列に対するパラメータ依存性が低いことがわかった。



(a) torso3. (b) FEM_3D_thermal2.

図 11 Block GPBiCGrQ 法と Block GWGPBiCGrQ 型解法の反復回数

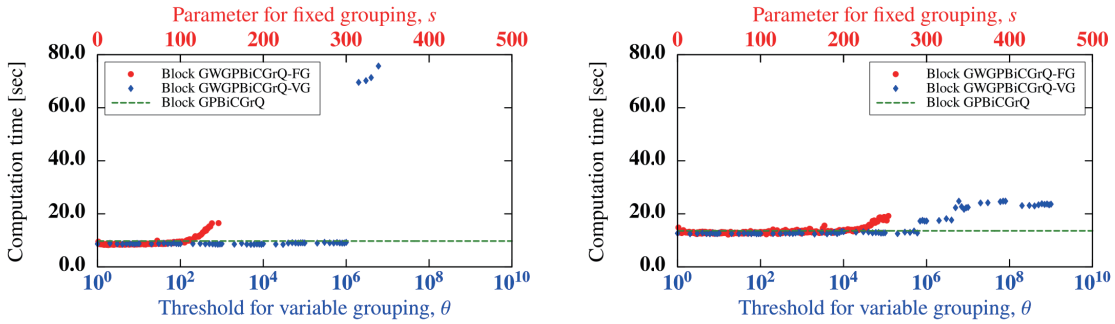
図 12 に Block GPBiCGrQ 法と Block GWGPBiCGrQ 型解法の真の相対残差を示す。真の相対残差は $\| B - AX_k \|_F / \| B \|_F$ で与えられ，近似解 X_k の精度を評価する指標の 1 つである。同図に示すように，Block GWGPBiCGrQ 型解法は Block GPBiCGrQ 法よりも高精度の近似解を生成できていることがわかる。Block GWGPBiCGrQ-FG 法では，s が 1 に近づくにつれて真の相対残差が増加する傾向にある。これに対して，Block GWGPBiCGrQ-VG 法では収束判定条件が満たされた θ において，真の相対残差がほぼ一定であった。



(a) torso3. (b) FEM_3D_thermal2.

図 12 Block GPBiCGrQ 法と Block GWGPBiCGrQ 型解法の真の相対残差

図 13 に Block GPBiCGrQ 法と Block GWGPBiCGrQ 型解法の計算時間を示す。Block GWGPBiCGrQ 型解法はパラメータの値が大きくなると反復回数が増加するため，それに伴い計算時間も長くなっている。パラメータの値が小さい場合は，Block GWGPBiCGrQ 型解法の計算時間は Block GPBiCGrQ 法の計算時間とほぼ同じであった。Block GWGPBiCGrQ-VG 法では，内部反復の停止条件に行列 $\tau_j^{(m)}$ の条件数の計算が必要となる。この計算量は $O(L^2)$ であるため，右辺ベクトル数 L が少ない場合は計算時間への影響は少ない。このとき，同法の反復回数が Block GPBiCGrQ 法とほぼ同じである場合は，計算時間もほぼ同じになる。



(a) torso3.　　(b) FEM_3D_thermal2.

図 13 Block GPBiCGrQ 法と Block GWGPBiCGrQ 型解法の計算時間

[9] NVIDIA H100 GPU におけるグラフニューラルネットワークの学習精度と実行性能評価（小林，藤田，朴）

グラフ構造は，SNS，交通網，化合物の構造式，購買履歴などのありとあらゆるデータを表現できる非常に高い汎用性を持ち，この構造で表現されたデータを分析することによって，SNS 上のスパムボット検知，混雑予測，新薬の毒性予測，商品推薦といった様々な応用が可能になる。グラフ構造データを分析する手法として，近年の深層学習の発展に伴い，世界中の企業や研究機関が注目するグラフニューラルネットワーク (GNN) という技術が存在する。GNN は汎用性が高く，ノード分類，リンク予測，グラフ生成など，グラフに関するさまざまな問題の解決に使用することができる。この汎用性により，GNN はコンピュータビジョン，自然言語処理，推薦システムなど，様々な領域で強力なツールとなっている。その一方で，近年におけるデータの大規模化や機械学習アプリケーションの多様化から GNN の学習精度の向上および学習時間の短縮を実現する手法の確立が望まれている。

GNN はグラフ構造データを入力として取るが，画像認識で用いられるニューラルネットワークと同様に大量の行列積和演算を内包しているため，GNN のための演算加速装置としては GPU が用いられるのが一般的である。しかし，特定のグラフ構造やメモリアクセス要件の処理には限界があることが知られている。一方，GNN を加速させるハードウェアとして FPGA に注目した研究開発も注目を集めているが，動作周波数の低さが実行性能に少なからず影響を及ぼす可能性について報告されている。従って，我々は GPU と FPGA を適材適所的に活用することによって，GNN の学習精度の向上と学習時間の短縮を同時に実現することを目指す。その予備評価として NVIDIA 社が現在提供する最新型 GPU である NVIDIA H100 GPU を用いて実施した。

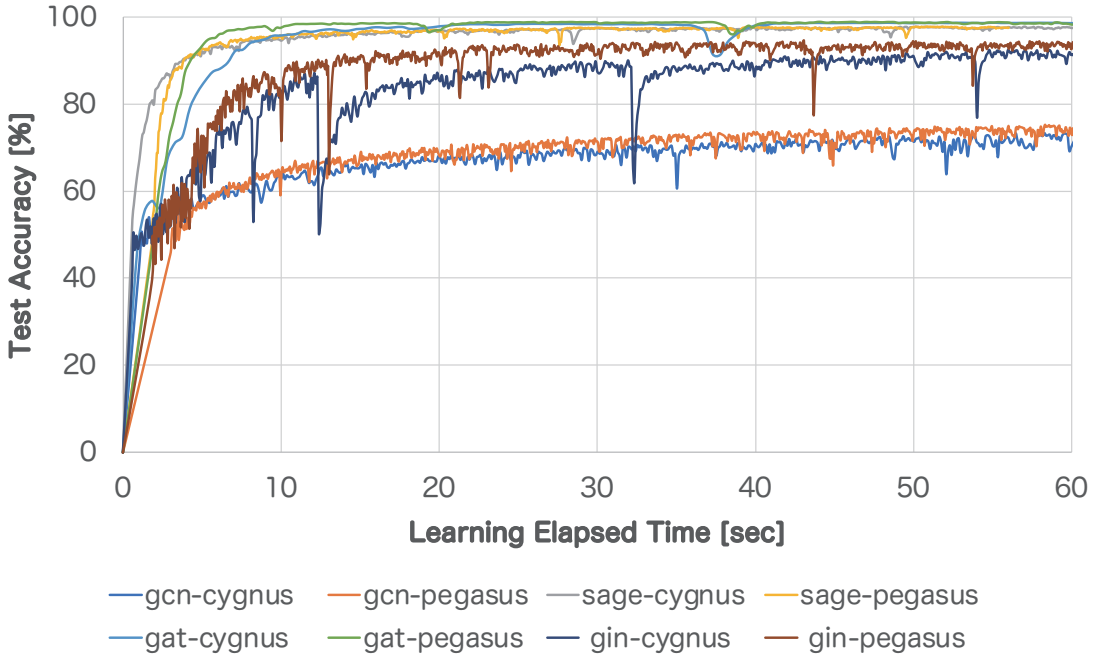


図 14　PPI データセットを学習する GNN モデルと実行プラットフォームに応じた学習時間とモデル制度の推移

NVIDIA V100 GPU (Cygnus)，H100 (Pegasus) をそれぞれ 1 台使用し，PPI データセットを学習する GNN モデルと実行プラットフォームに応じた学習時間とモデル精度の推移を評価した。図 14 にその結果を示す。性能評価では，エポック数の上限 1,000 と設定し，学習開始から 60 秒経過するまでのモデル精度をプロットした。なお，時間計測には PyTorch の torch.cuda.Event を使用した。

GCN モデルは最終的に 73 ~ 74% で推移している一方，GraphSAGE と GAT モデルは 95% 以上の精度を達成した。GCN はノードの全近傍を平均してノードの特徴を計算するため，全ての近傍ノードを同等に扱う。すなわち，中心のノードに近いノードも遠いノードも同様扱うが，中心のノードにより近いノードの情報は遠いノードの情報よりも重要である可能性が高く，全ての近傍を等しく扱うことは中心のノードの特徴計算において必ずしも最善の方法ではない可能性がある。さらに，本実験では GNN の層数は 3 であるため，遠いノードの情報を取り扱うのが難しいという問題もある。一方，GraphSAGE は各ノードの近傍を一定の数だけサンプリングして集約する挙動であるため，これにより各タンパク質の特徴がその近傍のタンパク質の特徴に基づいて更新され，全体のネットワーク構造を効果的に捉えることに成功していると考えられる。

GAT モデルでも高いモデル精度となったのは，Attention 機構が影響している。PPI データセットにおけるネットワークは生物学的な複雑性を持っており，各タンパク質（ノード）間の相互作用（エッジ）は一律ではなく，その強度や影響は大きく異なる場合がある。具体的には，あるタンパク質が他のタンパク質と強く相互作用している場合，そのタンパク質は

ネットワーク内で重要な役割を果たしている可能性が高く，GAT はこのような「強い」相互作用をより大きな Attention 係数で捉え，結果としてそのタンパク質の特性をより正確に表現している。これにより GAT は PPI ネットワークの複雑な特性を正確に捉え，最終的には GraphSAGE より高いモデル精度を達成しているが，エッジ数に比例した Attention 係数の計算オーバーヘッドの影響で学習時間の応じたモデル精度の向上は GraphSAGE よりも緩やかとなる。

GIN モデルの精度も学習時間が進むにつれて向上するが，GraphSAGE や GAT よりも精度の増減が多く見られ，全体的に不安定な推移を示した。GIN モデルは近傍ノードの特徴を集約する際に，グラフの構造を強く考慮するが，その際ノード間のエッジを考慮し，異なるエッジの重要度を自動的に学習する。これにより，ネットワーク全体のトポロジーをより正確に捉えることができるが，PPI データセットのような複雑なネットワークでは GIN モデルのこの能力が逆効果になる可能性がある。PPI データセットは複数のタンパク質間の相互作用を表すネットワークであり，一部の相互作用は非常に特異的（つまり，他の多くの相互作用とは異なる特性を持つ）である可能性がある。GIN モデルがこれらの特異的な相互作用を重要と判断し，それらに過度に重みを付けるとモデルはそれらの特異的な特性に過度に適合してしまい他の一般的な相互作用に対する予測精度が低下する恐れがある。また，これは GIN モデルが訓練データの特異的な特性に過度に適合してしまい，新しいデータ（テストデータ）に対する予測性能が低下, すなわち過学習してしまう傾向にもつながる。これらの影響によって，GIN モデルの精度が一部で下がってしまっている可能性がある。

学習時間については，NVIDIA H100 GPU 上で動作させた GNN モデルは，NVIDIA Tesla V100 GPU で動作させた場合と比較し，1.6 ~ 1.7 倍高速に学習を実行することが分かった。しかし，H100 GPU の演算性能は少なくとも V100 の 3 倍以上であるため，この実効性能は演算性能ではなくメモリバンド幅によって律速されている可能性がある。まだ PPI データセットのみでの実験であり，これはあくまで仮説の段階であるので，今後は異なるデータセットを使用して同様の実験を実施し，測定された性能の変動が一貫しているかどうかを確認していく。また，NVIDIA から提供されているプロファイラである Nsight Systems を用いて，GPU の実行時間・メモリ使用量・キャッシュ利用状況などの詳細な情報を収集して，上記の仮説を検証していくと共に，各 GNN モデルが GPU の性能をどの程度まで引き出せるのかを定量的に明らかにしていく。

[10] 並列 FPGA 環境における FPGA 間通信に関する研究（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。しかしながら，CIRCUS はフロー制御が実装されておらず，アプリケーションで通信バッファの残量を管理しなければバッファが溢れ，通信が消失するという制限がある。この制限の影

響により，これまでの研究で集団通信のような複雑な通信を CIRCUS 上に実装することが困難であることが明らかとなっていた。本年度は，この問題を解決するために CIRCUS の通信プロトコルの改善に取り組んだ。

現在の CIRCUS 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ため，CIRCUS にフロー制御が実装できていない。この問題を解決するために，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。

Kyokko はいくつかの FPGA ボードで動作確認がとられているが，Cygnus で用いている BittWare 520N FPGA ボードでの動作は対応出来ていない。また，100 Gbps 通信の環境での動作確認も十分ではない。本年度は，Kyokko の 520N ボードへの移植および 100 Gbps 環境で正常に動作するかの検証を行った。

Kyokko は FPGA ボードに依存しないプロトコル部と，FPGA ボードに依存する I/O 部が分離された設計がなされている。我々は 520N ボードに対応した I/O 部を実装し，Kyokko が 520N ボードで動作するように拡張を行った。また，100 Gbps 通信に対応するために，より高速なクロックを用いて Kyokko が動作できることを確認した。

1 枚の 520N ボードにある 1 番ポートと 2 番ポートをケーブルで接続し，ループバック構成を構成し，Kyokko を 520N ボードと組み合わせたときの性能を測定する。表 2 に測定結果を示す。物理層の転送速度である 100Gbps に対して，Kyokko 上で 99.9% のバンド幅を達成した。わずかに性能が低下しているのは，通信維持のために必要な制御用パケットが一定間隔で自動的に送信されるためである。表 3 にレイテンシ（通信遅延）の測定結果を示す。180ns という非常に低い通信遅延を達成した。通信機構が FPGA の回路に直結しているため，このような広帯域で低遅延な通信が実現できており，本結果は FPGA の通信性能が他の演算加速装置を凌駕していることを示している。高性能計算で用いられている他の演算加速装置（Graphics Processing Unit 等）は，演算に特化しており，通信機構は外部に追加する必要があり，一体化していないことによるオーバーヘッドが無視できない。

表 2 通信バンド幅の測定結果

バンド幅	99.985Gbps
効率	99.985%

表 3 レイテンシの測定結果

平均遅延（クロック）	71.0
平均遅延（ns）	181.8

[11] 複数の演算加速装置を統一的に扱えるプログラミング環境に関する研究 (藤田, 小林, 朴)

科学技術計算の分野ではスーパーコンピュータを用いて大規模な計算が行われているが，性能の向上に伴って消費電力も増大していることが問題となっており，単に性能が高いだけでなく，電力あたりの性能（Performance per watt）が高いことも重要視されている。電力問題を解決するために，演算加速装置（アクセラレータ）を搭載する計算機が広く用いられている。科学技術計算の分野では Graphics Processing Unit (GPU) が演算加速装置として広く用いられている。

一方で，演算加速装置の多様性が増しているという問題も存在する。現在，高性能計算で演算加速装置として用いられる GPU を開発している企業は，主に 3 社ある (AMD 社･Intel 社･NVIDIA 社)。一般的に，ユーザは複数のスーパーコンピュータや，手元の開発環境など様々な環境を横断して利用しているため，アプリケーションはどのような環境でも，効率よく動作することが求められている（性能可搬性)。しかしながら，従来の開発環境は単一の演算加速装置を対象としているため，複数の計算機間で性能を維持しつつアプリケーションを共有することが困難である。また，1 台の計算機に複数の演算加速装置が搭載されている場合もあり，性能可搬性を実現するには，複数の演算加速装置を扱えることも重要である。

本研究の目的は，複数の演算加速装置を統一的に扱えるプログラミング環境を実現することである。本目的を達成するためには，プログラミング言語面だけでなく，様々な演算加速装置を扱うためのソフトウェアの整備が求められる。我々が研究開発しているプログラミング言語「CHARM-SYCL」と，米国 Oak Ridge National Laboratory (ORNL) が研究開発している複数の演算加速装置環境 (Multi Heterogeneous) を扱えるランタイムフレームワーク「IRIS」を融合し，目的達成を目指す。

図 15 と図 16 はベクトル加算するベンチマークの評価結果である。図 15 は，CHARM-SYCL & IRIS で実装したものと，CUDA ネイティブで実装した場合を比べる性能評価である。CUDA で実装した物は Baseline と考えられるため，CHARM-SYCL&IRIS 実装と比べることでオーバーヘッドを測定できる。CHARM-SYCL と IRIS を使うことでランタイムのオーバーヘッドが発生するものの，最大で 6% の性能低下であり十分高速であることがわかる。図 16 は同じベンチマークを NVIDIA GPU と AMD GPU を 1 台ずつ搭載した計算機で実行したものである。NVIDIA のみ，AMD のみ，NVIDIA&AMD の 3 パターンの実行を実現できている。2 台用いた場合に性能が単純に和として増えていないが，これは，性能差を考慮せずに 2 台の GPU に同じだけの計算を割り振っているためであり妥当な結果といえる。これらの結果より，単一のアプリケーションコードで同種複数・異種複数どちらのケースも対応でき，プログラムの可搬性が実現できていることを示した。

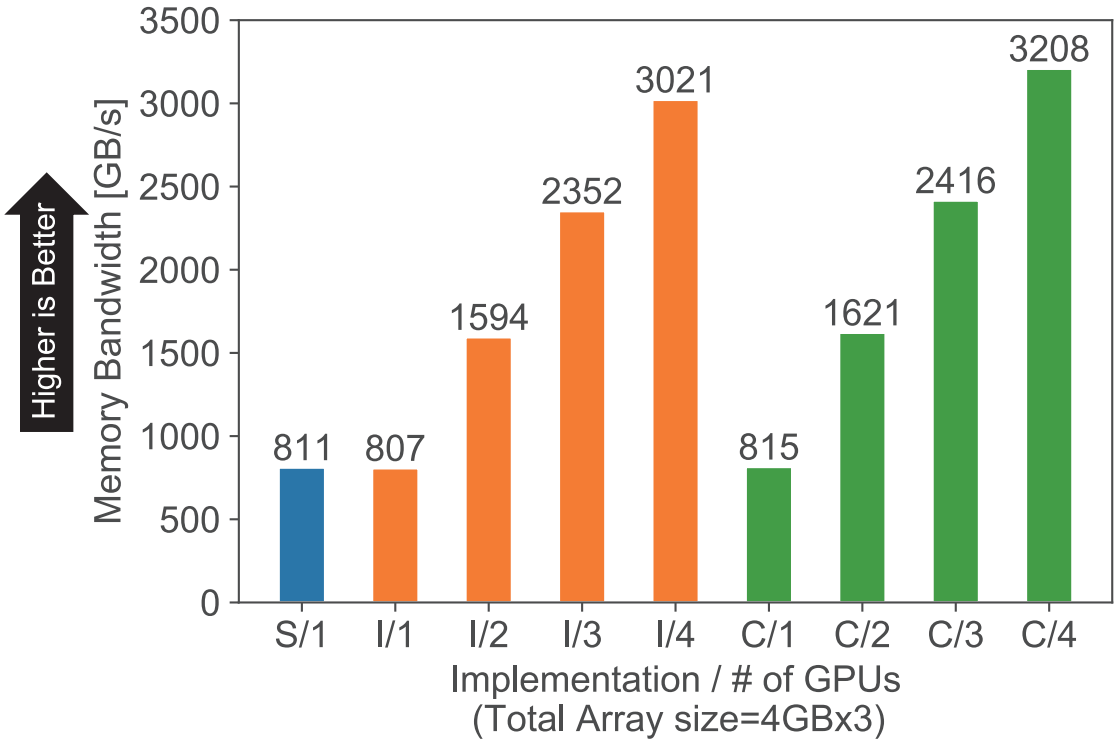


図 15 NVIDIA GPU を 1 ～ 4 台用いる場合の性能評価結果。S は CHARM-SYCL & Native Backend，I は CHARM-SYCL&IRIS Backend，C は CUDA 実装を表す。

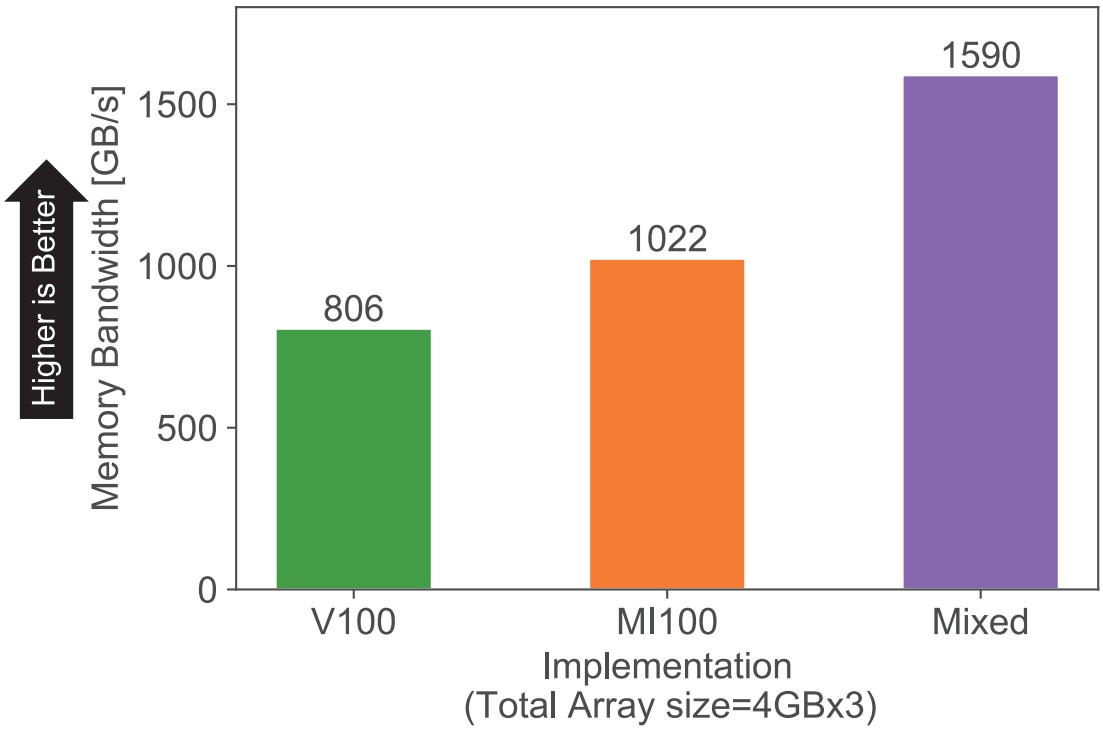


図 16 NVIDIA GPU と AMD GPU を組み合わせる場合の性能評価結果。V100 は NVIDIA V100 GPU を 1 台，MI100 は AMD MI100 GPU を 1 台，Mixed は V100 と MI100 を 1 台ずつ用いる場合を表す。

4. 教育

学生の指導状況

1. 佐野 由佳，修士（工学），高性能 FPGA 向け高位合成プログラミングにおける空間並列性向上手法，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 6 年 3 月（指導：朴泰祐）
2. 小山 創平，修士（工学），アドホックファイルシステムの設計と応用，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 6 年 3 月（指導：建部修見）
3. 木下 嵩裕，修士（工学），高レベル I/O ライブラリにおけるコンテキストによる高速化，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 6 年 3 月（指導：建部修見）
4. 山口 博將，修士（工学），GPU を用いたルジャンドル予想の数値的検証，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 6 年 3 月（指導：高橋大介）
5. 勢見 達将，修士（工学），DO CONCURRENT 構文によるシミュレーションコードの GPU 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 6 年 3 月（指導：額田彰）
6. 中野 将生，学士（情報工学），非同期通信ライブラリ async-ucx の性能評価，筑波大学情報学群情報科学類卒業論文，令和 6 年 3 月（指導：建部修見）
7. 川上 昌汰, 学士(情報科学), GPU における 8 倍精度浮動小数点数の高速フーリエ変換，筑波大学情報学群情報科学類卒業論文，令和 6 年 3 月（指導：高橋大介）
8. 田中 雅俊，学士（情報工学），高精度行列積計算によるブロッククリロフ部分空間反復法の近似解精度向上, 筑波大学情報学群情報科学類卒業論文, 令和 6 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. Daisuke Takahashi, 2023 Class of IEEE Computer Society Distinguished Contributors, 2024 年 2 月 18 日
2. Hiroto Tadano, JSST2023 Outstanding Presentation Award, Development and performance evaluation of the Block GPBiCGrQ method with variable grouping strategy, 2023 年 12 月 26 日

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，「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（A），建部修見（代表），R4 ～ R8 年度，3,650 千円，「次世代ストレージアーキテクチャの研究」
3. 文部科学省委託研究，建部修見（分担），R5 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）」
4. NEDO，建部修見（分担），R5 ～ R7 年度，32,498 千円，「超分散コンピューティング環境における高性能ストレージ基盤技術」
5. 特別共同研究，建部修見（代表），R4 ～ R7 年度，31,667 千円，「スケーラブルなデータストア及び高速データ処理に関する共同研究事業」
6. 科学研究費補助金基盤研究 (C)，高橋大介（代表），R4 ～ R6 年度，1,430 千円，「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」
7. 文部科学省委託研究，高橋大介，建部修見（分担），R5 年度，6,000 千円，「次世代計算基盤に係る調査研究」
8. 科学研究費補助金基盤研究 (C)，多田野寛人（代表），R5 ～ R7 年度，1,820 千円，「鞍点型連立一次方程式に対する高速・高精度階層並列型解法の開発」
9. JST 先端国際共同研究推進事業「次世代のための ASPIRE」，小林諒平（分担），R5~R8 年度，780 千円，「次世代の高効率計算基盤を実現する適応型データ圧縮ハードウェアの探求」
10. 科学研究費補助金基盤研究 (C)，小林諒平（分担），R5~R7 年度，80 千円，「多要素協調型 Approximate Computing 実現に向けた HPC アプリケーション解析手法」
11. 科学研究費補助金若手研究，小林諒平（代表），R4~R6 年度，520 千円，「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. 綱島隆太，小林諒平，藤田典久，朴泰祐，Seyong Lee, Jeffrey S. Vetter, 村井均，中尾昌広，辻美和子，佐藤三久，“OpenACC 単一記述による GPU+FPGA 複合デバイス処理システム”，情報処理学会論文誌コンピューティングシステム (ACS), Vol. 16, No. 2, pp. 1-15, 2023 年 11 月
2. Wentao Liang, Norihisa Fujita, Ryohei Kobayashi, Taisuke Boku, “Using Intel oneAPI for Multi-hybrid Acceleration Programming with GPU and FPGA Coupling”, Proceedings of Int. Workshop on Intel Extreme Performance Group (IXPUG) 2024, in HPC Asia 2024, pp.69-76, Jan. 2024
3. Taisuke Boku, Ryuta Tsunashima, Ryohei Kobayashi, Norihisa Fujita, Seyong Lee,

Jeffrey S. Vetter, Hitoshi Murai, Masahiro Nakao, Miwako Tsuji, Mitsuhisa Sato, “ OpenACC Unified Programming Environment for Multi-hybrid Acceleration with GPU and FPGA” , Proc. of Workshop on HPC on Heterogeneous Hardware (H3), in ISC2023, Lecture Notes in Computer Science, Vol. 13999, pp. 662–674, 2023

4. Osamu Tatebe, Kohei Hiraga, Hiroki Ohtsuji, “I/O-Aware Flushing for HPC Caching Filesystem” , Proceedings of 3rd Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads (REX-IO), pp. 11-17, 2023
5. Sohei Koyama, Kohei Hiraga, Osamu Tatebe, “Accelerating I/O in Distributed Data Processing Systems with Apache Arrow CHFS” , Proceedings of 3rd Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads (REX-IO), pp. 1-4, 2023
6. Daisuke Takahashi, “Multiple Integer Divisions with an Invariant Dividend and Monotonically Increasing or Decreasing Divisors” , Proc. 23rd International Conference on Computational Science and Its Applications (ICCSA 2023), Part II, Lecture Notes in Computer Science, Vol. 13957, pp. 393-401, Springer, 2023
7. Akira Nukada, Taichiro Suzuki, Satoshi Matsuoka, “Efficient checkpoint/Restart of CUDA applications” , Parallel Computing, Vol. 116, No. 103018, 9 pages, Elsevier, Jul. 2023
8. Hiroto Tadano, “Implementation and performance evaluation of a hierarchical parallel solver for saddle point problems on a GPU cluster” , Journal of Advanced Simulation in Science and Engineering, Vol. 10, No. 1, pp. 116-131, 2023
9. 中野博生 , 轟木義一 , 多田野寛人 , “富岳における高プロセス並列ジョブのメモリ消費について” , 日本シミュレーション学会論文誌 , Vol. 15, No. 2, pp. 56-63, 2023
10. Hiroto Tadano, “Development and performance evaluation of the Block GPBiCGrQ method with variable grouping strategy” , Proc. of the 42nd JSST Annual International Conference on Simulation Technology (JSST2023), pp. 55-58, 2023
11. Hiroki Nakano, Hiroto Tadano, and Norikazu Todoroki, “Parallel calculations of the extremely large number of MPI processes in Fugaku” , Proc. of the 42nd JSST Annual International Conference on Simulation Technology (JSST2023), pp. 21-24, 2023
12. Hiroto Tadano, “Development and performance evaluation of the Block-product type iterative methods with the variable grouping strategy” , Journal of Advanced Simulation in Science and Engineering, Vol. 11, No. 1, pp. 32-53, 2024
13. Norihisa Fujita, Beau Johnston, Ryohei Kobayashi, Keita Teranishi, Seyong Lee, Taisuke Boku, Jeffrey S. Vetter, “CHARM-SYCL: New Unified Programming

Environment for Multiple Accelerator Types” , SC-W ‘23: Proceedings of the SC ‘23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis pp.1651-1661, 2023

B）査読無し論文

1. 小山 創平 , 平賀 弘平 , 建部 修見 , Apache Arrow CHFS によるビッグデータ処理の I/O 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）, Vol. 2023-HPC-190, No. 5, pp. 1-11, 2023 年 8 月
2. 建部 修見 , 平賀 弘平 , 前田 宗則 , 藤田 典久 , 小林 諒平 , 額田 彰 , Pegasus ビッグメモリスーパコンピューターの性能評価 , 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-190, No. 7, pp. 1-12, 2023 年 8 月
3. 小山 創平 , 平賀 弘平 , 建部 修見 , FINCHFS: アドホック並列ファイルシステムの設計 , 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-192, No. 7, pp. 1-8, 2023 年 12 月
4. 杉原 航平 , 建部 修見 , スパースセグメントを活用した局所性志向バーストバッファにおけるフラッシュ手法の検討 , 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-192, No. 14, pp. 1-9, 2023 年 12 月
5. 丸山 泰史 , 建部 修見 , NVMe SSD 環境における CHFS の設計最適化の検討 , 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-192, No. 30, pp. 1-4, 2023 年 12 月
6. 中野 将生 , 建部 修見 , Rust の UCX ラッパー async-ucx の性能評価 , 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-192, No. 32, pp. 1-9, 2023 年 12 月
7. 平賀 弘平 , 建部 修見 , PMEMBB: 不揮発性メモリと MPI 片側通信を用いた MPI-IO バーストバッファの設計 , 研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2024-HPC-193, No. 21, pp. 1-14, 2024 年 3 月
8. 木下 嵩裕 , 建部 修見 , CA VOL: HDF5 におけるコンテキストによる I/O 最適化 , 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2024-HPC-193, No. 22, pp. 1-8, 2024 年 3 月
9. 溝谷 祐大 , 小林 諒平 , 藤田 典久 , 朴 泰祐 , 天笠 俊之 , “ラベルの出現頻度に着目した FPGA を用いた正規パス問合せの提案” , 第 16 回データ工学と情報マネジメントに関するフォーラム (DEIM2024), pp. 1-8, 2024 年 2 月
10. Wentao Liang, Norihisa Fujita, Ryohei Kobayashi, Taisuke Boku, “Using Intel oneAPI for multi-hybrid acceleration programming with GPU and FPGA coupling” , 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-192, No. 16, pp.

1-7, 2023 年 12 月

11. 和田 康孝, 森江 善之, 小林 諒平, 坂本 龍一, “細粒度な Approximate Computing 適用に向けた演算精度変更による影響の評価”, 研究報告ハイパフォーマンスコンピューティング（HPC）2023-HPC-191(13) pp.1-7, 2023 年 9 月
12. 森江 善之, 和田 康孝, 小林 諒平, 坂本 龍一, “Cast と通信の並列実行のための予備実験”, 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-191, No. 14, pp. 1-6, 2023 年 9 月
13. 藤田 典久, 小林 諒平, Beau Johnston, Narasinga Rao Miniskar, Seyong Lee, Keita Teranishi, Jeffrer S. Vetter, 朴 泰祐, “SYCL に基づく複数の演算加速装置を統一的に扱えるプログラミング手法の提案”, 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-190, No. 1, pp. 1-13, 2023 年 8 月
14. 小林 諒平, 藤田 典久, 朴 泰祐, 天笠 俊之, “NVIDIA H100 GPU におけるグラフニューラルネットワークの学習精度と実行性能評価”, 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-190, No. 17, pp. 1-8, 2023 年 8 月
15. 古川 和輝, 山口 佳樹, 横野 智也, 吉川 耕司, 藤田 典久, 小林 諒平, 安倍 牧人, 朴 泰祐, 梅村 雅之, “輻射輸送シミュレーションのための FPGA と GPU によるスクラッチパッドメモリの効率と有効性の分析”, IEICE-RECONF2023-6 123(71), pp. 29-34, 2023 年 6 月
16. 北爪 開人, 藤田 典久, 小林 諒平, 朴 泰祐, “HPC 利用に向けた FPGA 間シリアル通信コントローラ Kyokko の Intel FPGA への実装”, 研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2023-HPC-189, No. 4, pp. 1-9, 2023 年 5 月

（2）国際会議発表

A）基調講演

1. Taisuke Boku, “Next-Gen Accleration with Multi-Hybrid Devices - Is GPU Enough?”, 2nd Workshop on Communication, I/O, and Storage at scale on next-generation platforms, ISC2023, Hamburg, May 2023

B）招待講演

1. Taisuke Boku, “New Challenge for HPC and AI by Big Memory (Data) Supercomputer Pegasus”, HPC-AI Advisory Council Japan, virtual, Apr. 19, 2023
2. Taisuke Boku, “Challenge on Extreme-Hetero Application Programming”, 2023 ACM/IEEE 8th International Workshop on Extreme Scale Programming Models and Middleware in SC23, Denver, Nov. 13, 2023
3. Taisuke Boku, “Programming on Extreme-Hetero HPC Systems”, Mini Symposium

on Vision of Today's and Future Programming Model, SIAM-PP24, Baltimore, Mar. 5, 2024

4. Osamu Tatebe, All Flash PFS of Oakforest-PACS II and HPCI Nation-Wide Storage System, DDN SCA24 Data, HPC and AI Summit, Sydney, Feb. 19, 2024
5. Ryohei Kobayashi, "Accelerating astrophysics simulation with GPUs and FPGAs" , ADAC (Accelerated Data Analytics and Computing Institute) ~ Applications Working Group Monthly Seminar ~, online, Jun. 2023

C) 一般講演

1. Wentao Liang, Norihisa Fujita, Ryohei Kobayashi, Taisuke Boku, "Using Intel oneAPI for Multi-hybrid Acceleration Programming with GPU and FPGA Coupling" , Int. Workshop on Intel Extreme Performance Group (IXPUG) 2024, in HPC Asia 2024, Nagoya, Jan. 2024
2. Taisuke Boku, Ryuta Tsunashima, Ryohei Kobayashi, Nrohisa Fujita, Seyong Lee, Jeffrey S. Vetter, Hitoshi Murai, Masahiro Nakao, Miwako Tsuji, Mitsuhisa Sato, " OpenACC Unified Programming Environment for Multi-hybrid Acceleration with GPU and FPGA" , Proc. of Workshop on HPC on Heterogeneous Hardware (H3), in ISC2023, Hamburg, May 25, 2023
3. Osamu Tatebe, Kohei Hiraga, Hiroki Ohtsuji, "I/O-Aware Flushing for HPC Caching Filesystem" , 3rd Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads (REX-IO), Santa Fe, NM, Oct. 31, 2023
4. Sohei Koyama, Kohei Hiraga, Osamu Tatebe, "Accelerating I/O in Distributed Data Processing Systems with Apache Arrow CHFS" , 3rd Workshop on Re-envisioning Extreme-Scale I/O for Emerging Hybrid HPC Workloads (REX-IO), Santa Fe, NM, Oct. 31, 2023
5. Kohei Sugihara, Osamu Tatebe, Accelerate Stage-out in Single Shared Files from Node-local Burst-buffers, 8th International Parallel Data Systems Workshop Work-in-Progress Presentation, Denver, CO, Nov. 12, 2023
6. Daisuke Takahashi, "Implementation of Parallel Number-Theoretic Transform on GPU Clusters" , SIAM Conference on Parallel Processing for Scientific Computing (PP24), Baltimore, MD, USA, Mar. 7, 2024
7. Daisuke Takahashi, "Multiple Integer Divisions with an Invariant Dividend" , 10th International Congress on Industrial and Applied Mathematics (ICIAM 2023), Tokyo, Japan, Aug. 21, 2023
8. Hiroto Tadano, "Development and performance evaluation of the Block GPBiCGrQ

method with variable grouping strategy" , The 42nd JSST Annual International Conference on Simulation Technology (JSST2023), Niigata, Japan, Aug. 2023

9. Hiroki Nakano, Hiroto Tadano, Norikazu Todoroki, "Parallel calculations of the extremely large number of MPI processes in Fugaku" , The 42nd JSST Annual International Conference on Simulation Technology (JSST2023), Niigata, Japan, Aug. 2023
10. Norihisa Fujita, Beau Johnston, Ryohei Kobayashi, Keita Teranishi, Seyong Lee, Taisuke Boku, Jeffrey S. Vetter, "CHARM-SYCL: New Unified Programming Environment for Multiple Accelerator Types" , RSDHA: 3rd Workshop on Redefining Scalability for Diversely Heterogeneous Architectures in conjection with SC' 23, Denver, CO, Nov. 2023

D) ポスター発表

1. Sohei Koyama, (Advisor) Kohei Hiraga, Osamu Tatebe, Fast Checkpointing of Large Language Models with TensorStore CHFS, The International Conference for High Performance Computing, Networking, Storage, and Analysis, ACM Student Research Competition Posters, Denver, CO, Nov. 14-16, 2023
2. Sohei Koyama, Kohei Hiraga, Osamu Tatebe, FINCHFS: Design of Ad-hoc File System for High-Performance Computing Workload, 22nd USENIX Conference on File and Storage Technologies (FAST), Work-in-Progress Report and Poster, Santa Clara, CA, Feb. 27-28, 2024
3. Yuka Sano, Taisuke Boku, Mitsuhisa Sato, Miwako Tsuji, Norihisa Fujita, Ryohei Kobayashi, "Performance improvement by enhancing spatial parallelism on FPGA for HPC applications" , 2023 IEEE International Conference on Cluster Computing (CLUSTER), Poster, Santa Fe, NM, Oct. 2023
4. Yuka Sano, Taisuke Boku, Norihisa Fujita, Ryohei Kobayashi, Mitsuhisa Sato, Miwako Tsuji, "Enhancing spatial parallelism on loop structure for FPGA" , International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2024), Poster, Nagoya, Jan. 2024
5. Tatsumasa Seimi, Akira Nukada. "Performance Evaluation of OpenSWPC using Various GPU Programming Methods" , International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2024), Poster, Nagoya, Jan. 2024
6. Norihisa Fujita, Beau Johnston, Ryohei Kobayashi, Keita Teranishi, Seyong Lee, Taisuke Boku, Jeffrey S. Vetter, Unified Programming Environment for Multiple

Accelerator Types with Portability, The 6th R-CCS International Symposium, Kobe, Jan. 2024

（3）国内学会・研究会発表

A）招待講演

1. 小林 諒平 , “グラフニューラルネットワークにおける HPC 最前線” , 液体・ガラスへのデータ駆動アプローチ〜グラフニューラルネットワークとその周辺〜 , 神戸 , 2023 年 11 月

B）その他の発表

1. 小山 創平 , 平賀 弘平 , 建部 修見 , Apache Arrow CHFS によるビッグデータ処理の I/O 高速化 , 第 190 回情報処理学会ハイパフォーマンスコンピューティング研究発表会 , 函館 , 2023 年 8 月
2. 建部 修見 , 平賀 弘平 , 前田 宗則 , 藤田 典久 , 小林 諒平 , 額田 彰 , Pegasus ビッグメモリスーパコンピューターの性能評価 , 第 190 回情報処理学会ハイパフォーマンスコンピューティング研究発表会 , 函館 , 2023 年 8 月
3. 建部 修見 , Gfarm ファイルシステムの最新機能 , Gfarm シンポジウム , 秋葉原 , 2023 年 9 月
4. 建部 修見 , 次期フラグシップマシンのストレージシステム調査研究の中間報告 , Gfarm シンポジウム , 秋葉原 , 2023 年 9 月
5. 小山 創平 , 平賀 弘平 , 建部 修見 , FINCHFS: アドホック並列ファイルシステムの設計 , 第 192 回情報処理学会ハイパフォーマンスコンピューティング研究発表会 , 沖縄 , 2023 年 12 月
6. 杉原 航平 , 建部 修見 , スパースセグメントを活用した局所性志向バーストバッファにおけるフラッシュ手法の検討 , 第 192 回情報処理学会ハイパフォーマンスコンピューティング研究発表会 , 沖縄 , 2023 年 12 月
7. 丸山 泰史 , 建部 修見 , NVMe SSD 環境における CHFS の設計最適化の検討 , 第 192 回情報処理学会ハイパフォーマンスコンピューティング研究発表会 , 沖縄 , 2023 年 12 月
8. 中野 将生 , 建部 修見 , Rust の UCX ラッパー async-ucx の性能評価 , 第 192 回情報処理学会ハイパフォーマンスコンピューティング研究発表会 , 沖縄 , 2023 年 12 月
9. 平賀 弘平 , 建部 修見 , PMEMBB: 不揮発性メモリと MPI 片側通信を用いた MPI-IO バーストバッファの設計 , 第 193 回情報処理学会ハイパフォーマンスコンピューティング研究発表会 , 札幌 , 2024 年 3 月

10. 木下 嵩裕, 建部 修見, CA VOL: HDF5 におけるコンテキストによる I/O 最適化, 第 193 回情報処理学会ハイパフォーマンスコンピューティング研究発表会, 札幌, 2024 年 3 月
11. 中野 博生, 轟木 義一, 多田野 寛人, 坂井 徹, “富岳における大規模並列化による量子スピン系シミュレーション”, 日本物理学会 第 78 回年次大会, 2023 年 9 月
12. 多田野 寛人, “鞍点型連立一次方程式の階層並列型解法に対する漸化式動的グループ化反復法の適用と性能評価”, 日本応用数理学会 第 20 回研究部会連合発表会, 2024 年 3 月
13. 溝谷 祐大, 小林 諒平, 藤田 典久, 朴 泰祐, 天笠 俊之, “ラベルの出現頻度に着目した FPGA を用いた正規パス問合せの提案”, 第 16 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 2024), 2024 年 2 月
14. Wentao Liang, Norihisa Fujita, Ryohei Kobayashi, Taisuke Boku, “Using Intel oneAPI for multi-hybrid acceleration programming with GPU and FPGA coupling”, 第 247 回システム・アーキテクチャ・第 192 回ハイパフォーマンスコンピューティング合同研究発表会, 2023 年 12 月
15. 和田 康孝, 森江 善之, 小林 諒平, 坂本 龍一, “細粒度な Approximate Computing 適用に向けた演算精度変更による影響の評価”, 第 191 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 年 9 月
16. 森江 善之, 和田 康孝, 小林 諒平, 坂本 龍一, “Cast と通信の並列実行のための予備実験”, 第 191 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 年 9 月
17. 藤田 典久, 小林 諒平, Beau Johnston, Narasinga Rao Miniskar, Seyong Lee, Keita Teranishi, Jeffrer S. Vetter, 朴泰祐, “SYCL に基づく複数の演算加速装置を統一的に扱えるプログラミング手法の提案”, 第 190 回ハイパフォーマンスコンピューティング研究発表会（SWoPP2023）, 2023 年 8 月
18. 小林 諒平, 藤田 典久, 朴 泰祐, 天笠 俊之, “NVIDIA H100 GPU におけるグラフニューラルネットワークの学習精度と実行性能評価”, 第 190 回ハイパフォーマンスコンピューティング研究発表会（SWoPP2023）, 2023 年 8 月
19. 古川 和輝, 山口 佳樹, 横野 智也, 吉川 耕司, 藤田 典久, 小林 諒平, 安倍 牧人, 朴泰祐, 梅村 雅之, “輻射輸送シミュレーションのための FPGA と GPU によるスクラッチパッドメモリの効率と有効性の分析”, リコンフィギャラブルシステム研究会, 2023 年 6 月
20. 北爪 開人, 藤田 典久, 小林 諒平, 朴 泰祐, “HPC 利用に向けた FPGA 間シリアル通信コントローラ Kyokko の Intel FPGA への実装”, 第 189 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 年 5 月

（4）著書，解説記事等

1. 寒川光，高橋大介，有理算術演算―高精度数値計算のためのアルゴリズムとプログラミング―，森北出版，2023 年 9 月

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）

– 宇宙物理研究部門との共同研究における GPU+FPGA 多重演算加速環境の初期天体シミュレーションコードの性能向上への適用
– 素粒子物理研究部門と共同で Japan Lattice Data Grid（JLDG）の構築，運用
– 計算情報学研究部門（データ基盤分野）との共同研究における 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

産学官連携

– 学際ハブ拠点形成事業「AI 時代における計算科学の社会実装を実現する学際ハブ拠点形成」における生命科学シミュレーションコード Amber の高スループット実行に関する研究（アヘッド・バイオ・コンピューティング社との共同研究）

国際連携・国際活動

– Oak Ridge National Laboratory との GPU+FPGA 複合演算加速プログラミング環境に関する共同研究
– 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，米国アルゴンヌ国立研究所とのストレージに関する共同研究

8. シンポジウム，研究会，スクール等の開催実績

– Gfarm シンポジウム 2023，東京（ハイブリッド），2023 年 9 月 8 日
– Gfarm ワークショップ 2024，沖縄，2024 年 1 月 29 日～ 30 日

9. 管理・運営

朴泰祐：情報環境委員会委員
朴泰祐：計算科学研究センターセンター長
朴泰祐：計算科学研究センター次世代計算機システム研究開発室室長
朴泰祐：計算科学研究センター高性能計算機システム研究部門主任
朴泰祐：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長
朴泰祐：計算科学研究センター HPCI 推進室室員
朴泰祐：計算科学研究センター情報セキュリティ委員会委員長

建部修見：情報環境企画室室員
建部修見：ネットワーク管理委員会委員
建部修見：情報セキュリティ技術小委員会委員
建部修見：基幹ネットワーク小委員会委員
建部修見：計算科学研究センター副センター長
建部修見：計算科学研究センター高性能計算システム運用開発室室長
建部修見：計算科学研究センター情報環境委員会部局技術責任者
建部修見：計算科学研究センター計算機システム運用委員会委員長
建部修見：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
建部修見：計算科学研究センタービッグデータ・AI 連携推進室室員
建部修見：計算科学研究センター情報セキュリティ委員会委員
高橋大介：学術情報メディアセンター運営委員会委員
高橋大介：計算科学研究センター計算科学振興室長
高橋大介：計算科学研究センター先端計算科学推進室室員
高橋大介：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
高橋大介：計算科学研究センター学際計算科学連携室室員
高橋大介：計算科学研究センター情報セキュリティ委員会委員
額田彰：計算科学研究センター先端計算科学推進室室員
額田彰：計算科学研究センター高性能計算システム運用開発室室員
多田野寛人：計算科学研究センター情報セキュリティ委員会委員
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
小林諒平：計算科学研究センター先端計算科学推進室室員
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
小林諒平：計算科学研究センター高性能計算システム運用開発室室員
藤田典久：計算科学研究センター高性能計算システム運用開発室室員

10. 社会貢献・国際貢献

朴泰祐：理化学研究所客員主管研究員
朴泰祐：PC クラスタコンソーシアム理事
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
朴泰祐：「富岳」成果創出加速課題領域総括
朴泰祐：「次世代計算基盤に関する調査研究」Program Director
朴泰祐：HPCI 計画推進委員会委員
朴泰祐：「富岳」課題推進委員会委員
Taisuke Boku: Steering Committee Chair, International Conference on High

Performance Computing in Asia-Pacific Region (HPCAsia)
Taisuke Boku: Steering Committee Member, IEEE Cluster
Taisuke Boku: Steering Committee Member, International Conference on Parallel Processing
Taisuke Boku: 2023 ACM Gordon Bell Prize Committee Chair
建部修見：HPCI セキュリティインシデント即応委員会委員
建部修見：HPCI 連携サービス運営・作業部会委員
建部修見：HPCI 利用研究課題審査委員会レビューアー
建部修見：理化学研究所客員研究員
建部修見：情報通信研究機構協力研究員
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
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
Osamu Tatebe: Poster Co-chair, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC23)
Osamu Tatebe: Publicity Co-chair, IEEE International Conference on Cloud Computing (CLOUD 2023)
Osamu Tatebe: Program Committee, 10th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT 2023)
Osamu Tatebe: Program Committee, IEEE International Conference on Cluster Computing (CLUSTER 2023)
Osamu Tatebe: Program Committee, 52nd International Conference on Parallel Processing (ICPP 2023)
Osamu Tatebe: Program Committee, ACM International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2023)
Osamu Tatebe: Program Committee, 8th International Parallel Data Systems Workshop (PDSW 2023)
Osamu Tatebe: Program Committee, 13th International Workshop on Advances in High-Performance Computational Earth Sciences: Applications & Frameworks (IHPCES 2023)
高橋大介：理化学研究所客員主管研究員
高橋大介：HPCI 利用研究課題審査委員会レビューアー
高橋大介：HPCI 連携サービス運営・作業部会委員
Daisuke Takahashi: Editor, The International Journal of High Performance Computing Applications
Daisuke Takahashi: Program Committee, The 18th International Workshop on Automatic Performance Tuning (iWAPT 2023)

Daisuke Takahashi: Program Committee, The International Conference on Computational Science (ICCS 2023)
Daisuke Takahashi: Publicity Committee, The 23rd International Conference on Computational Science and Its Applications (ICCSA 2023)
Daisuke Takahashi: Program Committee, 52nd International Conference on Parallel Processing (ICPP 2023)
Daisuke Takahashi: The SIAM Activity Group on Supercomputing Career Prize (SIAG/SC Career Prize) and the SIAM Activity Group on Supercomputing Early Career Prize (SIAG/SC Early Career Prize) Joint Selection Committee Member
高橋大介：情報処理学会論文誌査読委員
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
多田野寛人：日本シミュレーション学会理事
Hiroto Tadano: Committe, The 42nd JSST Annual International Conference on Simulation Technology (JSST2023)
Hiroto Tadano: Local Scientific Program Committee, 10th International Congress on Industrial and Applied Mathematics (ICIAM2023)
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
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
小林諒平：理化学研究所客員研究員
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
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
小林諒平：電子情報通信学会英文論文誌 D 編集委員
小林諒平：電子情報通信学会論文誌 Low-Power and High-Speed Chips 特集号編集幹事
小林諒平：電子情報通信学会論文誌 Forefront Computing 特集号編集幹事
小林諒平：電子情報通信学会論文誌 Forefront Computing 特集号編集委員
小林諒平：電子情報通信学会 ISS ソサイエティ誌編集幹事
小林諒平：電子情報通信学会 ISS ソサイエティ誌編集委員
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
小林諒平：アダプティブコンピューティング研究推進体　広報イベント WG 副グループ長
小林諒平：SWoPP 2023 組織委員
小林諒平：SWoPP 2023 実行委員（コンピュータシステム研究会）
小林諒平：xSIG 2023 プログラム委員

Ryohei Kobayashi：Proceedings Chair, FPT’ 23
Ryohei Kobayashi：Program Committee, FPT’ 23
Ryohei Kobayashi：Review Committee, Open Accelerated Computing Summit 2023
Ryohei Kobayashi：Poster Chair, IEEE RTCSA/NVMSA 2023
Ryohei Kobayashi：Program Committee, HEART2023
Ryohei Kobayashi：Program Committee, FPL2023
Ryohei Kobayashi：Program Committee, CANDAR 2023
Ryohei Kobayashi：Program Committee, CANDAR 2023 CSA workshop
Ryohei Kobayashi：Program Vice Chair, COOL Chips 26
藤田典久：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
藤田典久：情報処理学会論文誌コンピューティングシステム（ACS）編集委員
藤田典久：Publicity Chair, HPC Asia 2024
Norihisa Fujita: Program Vice Chair, Performance Optimization and Auto-Tuning of Software on Multicore/Manycore Systems (POAT2023)

11. その他

該当なし