VII. 高性能計算システム研究部門

1. メンバー

教授	朴　泰祐
教授	高橋　大介
教授	建部　修見
准教授	川島　英之
助教	多田野　寛人
研究員	梅田　宏明
研究員	田中　昌宏
研究員	Mohamed Amin Jabri
研究員	松本　和也
学生	大学院生　17 名、学類生　6 名

2. 概要

本研究部門では、高性能計算システムアーキテクチャ、並列プログラミング環境、GPU 利用技術、並列数値処理の高速化研究、広域分散環境におけるデータ共有を中心とするグリッド計算技術等の研究を行っている。

3. 研究成果

【1】 PEACH2 基本システムソフトウェアの開発と NASPB-CG における評価（朴）

PEACH2 の通信ドライバ及び制御方式について API 上でユーザがエラーを起こしにくいような変更とエラーハンドリングを行い、ユーザアプリケーションをより記述しやすくした。また、この仕様をもとに、NAS Parallel Benchmarks の CG ベンチマークを GPU クラスタ HA-PACS/TCA 上で 2 次元プロセスマッピング実装し、短メッセージ通信に強いという TCA の特性を生かし性能を向上させた。

図 1-1 にベンチマークの中核となる conj_grad 関数の実行時間を、PEACH2 による実装（TCA）と InfiniBand 上の MPI による実装（MPI）で比較した結果を示す。問題サイズ（CLASS）は S, W, A, B の順で大きくなる。最大 16 ノード（16 GPU）による評価では、CLASS=W 以上でスケーラビリティが見え始め、CLASS=B では概ねスケールしている。2 次元分割によって平均メッセージ長が短くなり、CLASS=W までは TCA の性能が上回る。一方、strong scalability が現れる CLASS=A 以上では、メッセージ長が長くなるため、TCA の優位性が減り、MPI との性能差はほとんどなくなる。さらに CLASS=B では InfiniBand のレイテンシを隠す十分なメッセージ長が得られるため、ノード数が増えると性能が逆転する。平成 26 年度に行った CG 法の

1 次元分割アルゴリズムに比べると TCA の優位性は高まったが、より大きな問題サイズにおいても性能を維持できるよう、さらに工夫が必要であることがわかった。

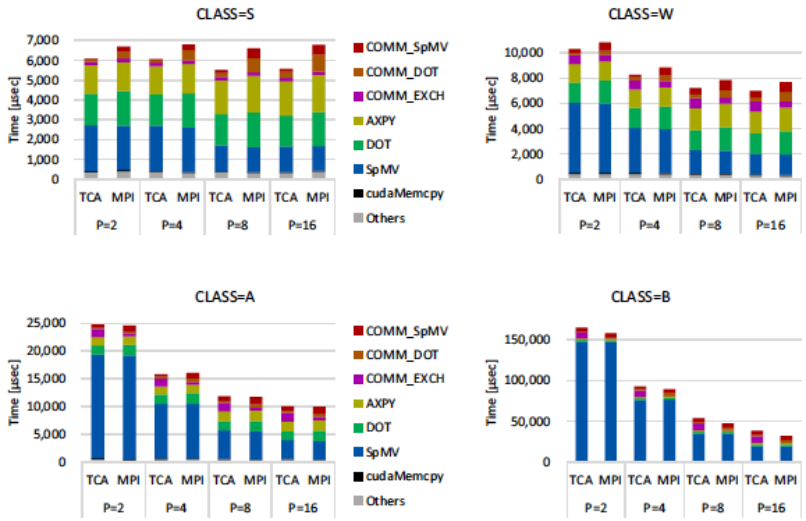


図 1-1 NASPB-CG の２次元マッピング実装における各クラス問題での TCA 実装と MPI/IB 実装の conj_grad 関数の実行時間比較

【2】 GPU 間直接通信 MPI ライブラリ GMPI の開発（朴）

従来の CUDA などの典型的 GPU 環境を並列処理に展開する場合、MPI 通信を GPU のカーネル実行と別途記述する必要があるが、MPI 通信を能動的に行えるのはホスト CPU のみであり、このため CPU 用の MPI プログラムを GPU 並列化するためには、通信に挟まれた計算部分の GPU カーネル化を通信が発生するたびに行わなければならない。この切り分けコストは並列 GPU アプリケーションコーディングにおいて大きな制約となり、プログラミングの複雑さとデバッグの難しさにより、プログラムの生産性を大きく低下させる要因となっている。本研究テーマでは、この問題を解決するために、GPU のカーネルコード上で直接 GMPI 関数を呼び出せる新しいフレームワークとして GMPI (GPU-self MPI)を提案・実装した。

当然ながら、GPU は能動的デバイスではないため、内部から自由に InfiniBand のような外部通信機構を呼び出すことはできない。このため、実際の MPI 通信はホスト CPU に依頼し、GPU 側とホスト CPU の間で通信依頼と結果のリターンを司る通信チャネルを作成する。この通信チャネルに通常の cudaMemcopy のような通常のメモリ参照関数を使うことはできない。なぜならば、これらの関数は GPU からホスト CPU に制御が戻っていることを仮定しているが、GMPI では GPU はホスト CPU と並行して走り続けることを想定するからである。近年の CUDA 環境では、ホスト CPU のアドレス空間に GPU のアドレス空間をピンダウンし、ここを双方で読み書きすることで、PCIe を介した仮想的な共有メモリ空間を作成することができる。そこ

で、ここに双方がアクセスする情報通信用のリングバッファを複数設け、互いにポーリングとアップデートを繰り返して必要な情報のやり取りを行う。図 2-1 にこの様子を示す。

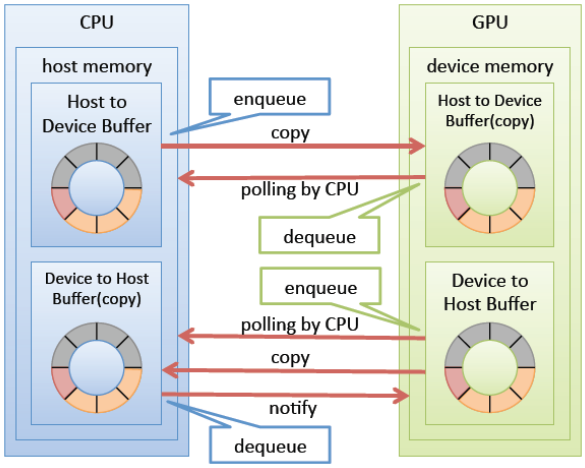


図 2-1 GMPI で用いるピンダウンメモリ領域を使った GPU・ホスト間のリングバッファ構造

ここで、ノード間 MPI 通信性能をできるだけ低下させないために、GMPI ではこのリングバッファを用いて全ての通信データをやり取りせず、GPU 対応 MPI が GDR 等の機能を使って最高の性能で実際の MPI 通信が実現されるように設計した。つまり、リングバッファを使ってやり取りされるのは MPI 関数の引数や結果の状態だけであり、通信対象となる実データは GPU 側メモリに置いたままで、実際の MPI 関数にその参照を委ねる。

GMPI においては、(a) MPI 関数を呼ぶたびにホスト CPU に制御を戻す必要がないためオーバヘッドが削減される、(b) プログラムの書き換えを最小限の手間で行える、という２つのメリットがある。(b)については自明であるが、(a)が正しいかどうかは、上述のリングバッファ機構を通じたデータ交換の性能に大きく依存する。そこで、従来方式の MPI-GDR/IB (InfiniBand 上の GDR を用いた MPI 通信)と GMPI の基本通信性能を、pingpong 通信を用いて評価した。結果を図 2-2 に示す。ここで、MVAPICH2-GDR は GDR を用いた MPI 本体の通信レイテンシであるが、実際の通信前には必ず GPU 側のメモリ読み書きが終了していることを保証するために、GPU との同期を取らなければならない。そこで、この同期コストも含めた通信レイテンシも測定した。これと GMPI による通信性能の比較を行った結果、図に示されるように、メッセージ長が 4KB 程度までの短い場合は 10～20%程度の性能差はあるが、これを超えるとほとんど性能低下はないことがわかった。よって、ある程度の長さのメッセージであれば、上記(b)のメリットが大きく活き、並列 GPU プログラミングの生産性が向上すると期待される。

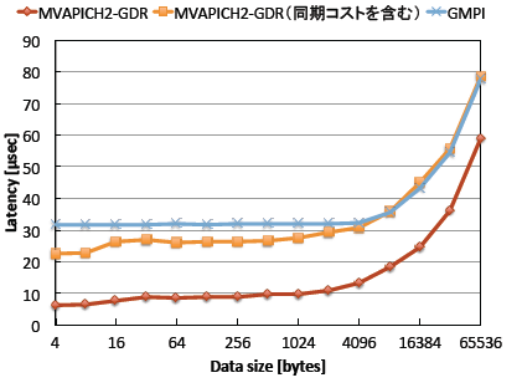


図 2-2 GDR 付き MPI の直接実行と GMPI を用いた場合の pingpong 性能評価

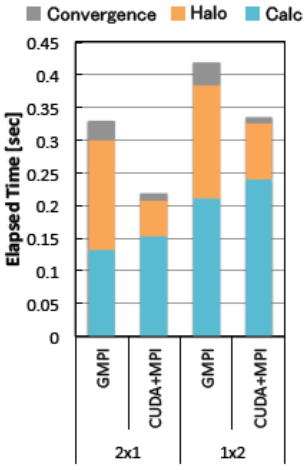


図 2-3 Himeno Benchmark (SMALL)の GMPI 性能評価

現在 GMPI はまだ実装中で、MPI の全関数に対応していないが、NAS Parallel Benchmarks 等の基本ベンチマークを評価するための関数セットは実装されている。これを用いて、Himeno Benchmark を 2 ノードで実行した場合の評価結果を図 2-3 に示す。ここでは 3 次元空間を 2 ノードに分割する際、データの連続方向分割と非連続方向分割（ブロックストライド転送を必要とする）の２通りの分割方法を比較した（問題サイズは SMALL）。結論として、どちらの分割方法でも GMPI の通信オーバヘッドは通常の CUDA+MPI に比べ大きいが、分割方法が連続方向であった場合の差は相対的に小さく、通信時間全体でみると 2 倍から 3 倍の時間がかかっているが、総計算時間では 40～20%程度の時間増加に収まっている。Himeno Benchmark は並列通信に比べ比較的計算量の小さい問題であるため、計算時間がより支配的なアプリケーションにおいては、さらに性能差が縮まり、GMPI によるプログラミングメリットが生きると考えられる。なお、Himeno Benchmark におけるオリジナルコード（CUDA+MPI）は 1317 行であるのに対し、GMPI 版では 1210 行と小さくなった。差はほぼ 100 行で 1 割弱であるが、

それ以上にプログラミングの見通しがずっと良くなり、CPU+MPI のプログラムからの移植性が極めて高くなる効果が期待される。今後は GMPI 関数の種類の充実化を行い、オーバヘッドの削減を目指す。その上で、より本格的な大規模コードの GMPI 化を行い、コード量比較と性能比較を進めていく予定である。

なお、本研究は情報処理学会 ACSI2016 において発表され、Outstanding Research Award を受賞した。

【3】 PGAS 言語向け通信ライブラリ GASNet の GPU 向け実装と TCA 実装（朴）

GASNet は米国 LBNL (Lawrence Berkeley National Laboratory)で開発された、PGAS (Partitioned Global Address Space)モデル実装用の低レベル通信ライブラリである。分散メモリ上で MPI よりも軽い通信を実現する。これまで GASNet は CPU における並列通信のみを対象にしてきたが、現在、これを GPU クラスタに対応させる拡張が LBNL によって進められている。そこで我々は、LBNL の GASNet/GPU 開発チームと共同研究を行い、先方が InfiniBand を対象として進めているリファレンス実装と並行し、これを TCA 機構にも用い、両者の性能比較を行う研究を進めている。本研究により、今後本格化されると思われる GPU を対象とした PGAS モデル言語の実装に TCA を容易に用いることができるようになると見込んでいる。

TCA 上の GASNet/GPU（以下、GASNet/TCA）の設計・実装は、前節で述べた GMPI に似ている。GPU 側から発生される GASNet 通信要求は、ホスト CPU と GPU で共有されるピンダウンされたメモリ空間に実装されたリングバッファを通じ、GPU 側の通信リクエストをホスト CPU が受け取って TCA 通信によって実施し、その結果をまた逆向きのリングバッファで返す。ただし、GMPI ではリクエストはそのまま MPI 通信に引き渡されていたのに対し、GASNet/TCA では InfiniBand を対象とした GASNet/GPU（以下、GASNet/IB）で想定される基本プロトコルを TCA の限られた API で実装するため、より一層の工夫が必要である。

図 3-1 に GASNet/TCA における GPU とホスト CPU の間のリクエストを受け渡すパケット通信機構の様子を示す。この機構を用い、図 3-2 に示す４つのデータ転送モードを実装した。ここで注意すべき点は、GASNet においては segment と呼ばれる連続アドレス空間を通信におけるオブジェクトと定義しているが、現在の GASNet/GPU プロトタイプではこの segment を１つだけ許していることである。並列 GPU 処理では当然、データは基本的に GPU 上に存在していると仮定するため、この 1 つだけの segment は GPU メモリ上に存在することになる。図 3-2 の各通信モードはこれを意識している。

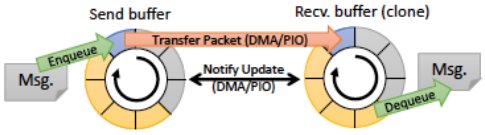


図 3-1　GASNet/TCA における GPU・CPU 間通信のためのパケット通信機構

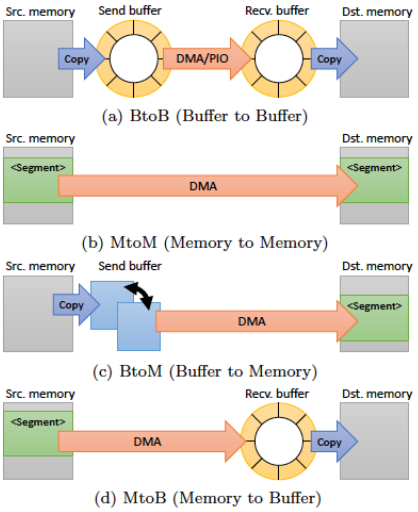


図 3-2 GASNet/TCA における 4 つの通信モード

LBNL で開発されている GASNet/IB は GPU を搭載した計算ノードに InfiniBand が搭載されていることを想定している。我々はこれを TCA の制約下で通信レイヤを置き換え、GASNet/GPU で定義される全ての core 通信 API と一部の extended API を実装した。なお、extended API は全て core API で実装することも可能で、今回実装していない extended API は、実装した core API で実現可能である。

GASNet はユーザが直接使用するのではなく、分散メモリ向け PGAS モデルに基づく UPC 等の言語実装を想定しているが、プログラミングによりユーザレベルで利用することは可能である。そこで、GASNet/IB と GASNet/TCA の性能比較を、低レベルな通信ライブラリ上で行った。性能評価は筑波大学における GPU クラスタ HA-PACS/TCA を用いて行った。

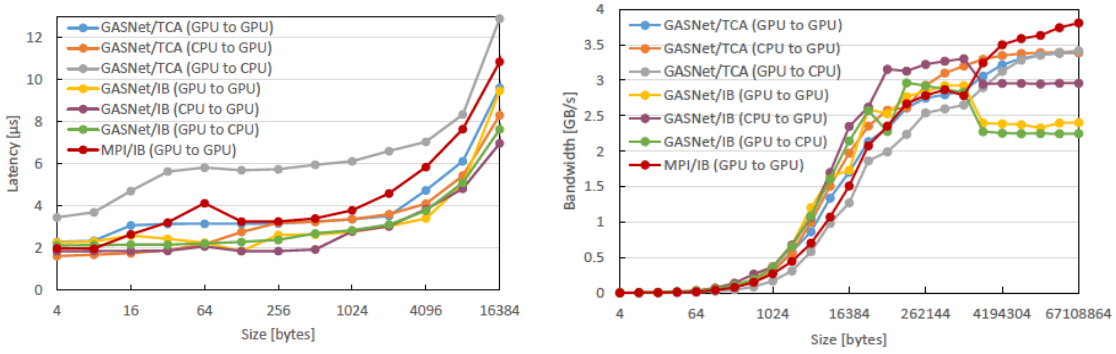


図 3-3　GASNet/TCA と GASNet/IB の各種メモリ間連続転送のレイテンシとバンド幅比較

図 3-3 は、GPU または CPU メモリからリモートノードの GPU または CPU メモリへのデータ転送の通信レイテンシ及びバンド幅の測定結果である。ここでは GASNet/TCA と GASNet/IB

の他に、参考データとして MPI/IB の結果も載せている。注目してもらいたいのは、”GASNet/TCA (GPU to GPU)” (青線) と”GASNet/IB (GPU to GPU)” (黄線) である。データ長が数十 Byte と非常に小さい区間では GASNet/TCA のレイテンシは GASNet/IB に比べ 3 割程度大きいが、数百 Byte 程度では互角または TCA の方が若干小さくなり差はなくなる。また、データ長が 256KB を超えた場合では両者は逆転し、バンド幅を見ると、GASNet/IB はこの近辺で性能が大きく低下しているのに対し、GASNet/TCA は伸び続け、3.3GB/s 程度で律速する（これは理論的なピークに近い）。GASNet/IB の性能が低下するのは、GPU メモリの管理方法が最適化されておらず、TCA 版では Page-Locked Host Memory を用いた PCIe 上のデータ転送最適化を行っているので差が生じていると考えられる。

次に、GASNet/GPU における extended API として、多次元矩型領域のドメイン分轄法で非常によく用いられる、多次元ステンシル計算等を想定したブロックストライド転送における性能評価を行った。TCA 機構では FPGA ハードウェア上でブロックストライド転送を単一メッセージとして扱うことができ、このようなアプリケーションを想定した通信においてその効果を発揮する。我々はこのブロックストライド転送の extended API を TCA 向けに最適化実装した。結果を図 3-4 に示す。

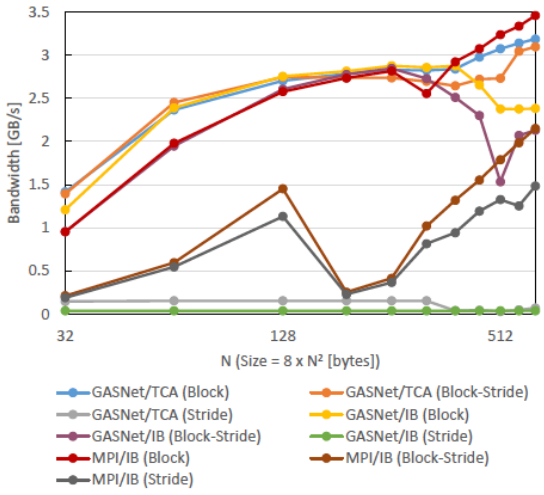


図 3-4 ブロックストライド転送を含む各種パターンの GASNet/TCA と GASNet/IB の比較 (GPU to GPU copy)

ここでは、“GASNet/TCA (Block-Stride)” (オレンジ線) と“GASNet/IB (Block-Stride)” (紫線) に注目して頂きたい。両者ともバンド幅は伸びているが、2 次元領域の要素数 (N) が 256 を超える辺りから GASNet/IB が失速するのに対し、GASNet/TCA での性能低下は小さい。これも、先に述べた通信最適化 (Page-Lock) が TCA 側で功を奏しているためだと思われる。なお、単純な Stride 転送では全く性能が出ていないが、これは GASNet/GPU ではデータの

pack/unpack を定義できないため（memory to memory copy を行うので）である。MPI/IB ではメッセージの送受信の際に pack/unpack を行っているため性能が高い。

本研究は CCS と LBNL の共同研究契約に基づいて行われ、特に基本部分の議論と HA-PACS を用いた MPI/IB の性能評価は、LBNL からの共同研究者が CCS を訪問滞在中に行われた。同共同研究契約の重要な成果であると位置付けられる。

【4】 電子動力学法コード ARTED の Xeon Phi (KNC)向け性能最適化（朴）

平成 26 年度より、量子物性研究部門の矢花グループとの共同研究として、同グループで京コンピュータ等の大規模システム向けに開発を進めている、量子動力学法に基づく電子シミュレーションコード ARTED (Ab-initio Real Time Electron Dynamics)を Intel Xeon Phi メニーコアプロセッサ向けに最適化を行っている。本研究は、現在本センターで運用中の大規模メニーコアクラスタである COMA 上のアクセラレータ、Intel Xeon Phi (KNC: Knights Corner)での高性能実行を実現することにより、同システムをより効率的に利用すると共に、平成 28 年度に導入予定の次期 Intel Xeon Phi (KNL: Knights Landing)プロセッサを用いた超並列クラスタ（Oakforest-PACS）への移植も視野に入れている。

ARTED は RSDFT (Real Space Density Functional Theory)を基本とし、これで計算された基底状態を実時間発展させるコードで、一般の RSDFT に比べ比較的小さな空間ドメインの問題に対し、大量の波数空間計算を多重に行う。このため、一般の大規模並列処理と異なり、空間ドメインを並列プロセスに分割するのではなく、波数空間の計算を並列処理する。各空間ドメインの格子はステンシル計算となるが、これに関しては並列化しないため、袖領域に関する通信（Halo 通信）は発生しない。代わりに波数空間の計算結果の Allreduce 処理が発生するが、この通信時間は 100 程度の並列プロセスではオーバヘッドとはならない。従って、本研究では空間ドメインの処理を Xeon Phi 上で多数のスレッドでどのように効率的に処理するかに傾注している。

現在の KNC アーキテクチャの Xeon Phi では、チップ当たり 61 個のコアがあり、各コア内で最大 4 つのハードウェアスレッドを利用できる。1 つのコアは OS 処理に向けられるため、最大 240 のスレッドを意識する必要がある。さらに、512bit の AVX 命令により、倍精度浮動小数点演算を最大 8 つ SIMD 処理可能である。よって、1 つの MPI プロセスを Xeon Phi チップに割り当ててスレッド並列処理を行う場合、極めて大量のスレッドレベル及び命令レベル並列処理を行わなければならない。また、60 のコアが最終レベルキャッシュを共有するため、データアクセスパターンを意識したメモリ配置に注意する必要がある。

以上の背景の下、オリジナルコードを図 4-1 のような形で最適化し、コンパイラによる自動並列化を適用した。ここで行った最適化は以下の通りである。

- 同一空間に対して異なる配列を割り当てていた多数の変数を多次元配列にまとめた。これにより、コンパイラによるベクトル展開が容易になる。
- 空間分割に対する周期境界条件付きの配列参照のインデックス計算が剰余演算を用いて行われていたのを、静的なインデックス配列に置き換えた。Xeon Phi では整数剰余計算が非常に遅いため、インデックス配列を参照した方が、メモリ参照回数が多くても結果的に高速になる。
- 大きなループ内で書き込みを行っている変数のうち、実際にはそのループ内で互いに参照されないものについては non-temporal store をディレクティブ指示し、キャッシュが汚れないようにした。
- 演算の順序を変更し、メモリアクセスが連続パターンとなるようにした。

これらに加え、自動ベクトル化の代わりに Xeon Phi のイントリンシック命令を手で書くことにより、手動ベクトル化を追加することで、さらに詳細な最適化を行った。この部分については、今後 KNL アーキテクチャ向けに再度変更する必要がある。

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

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

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

3次元配列に変換

インデックスを直接計算
（剰余テーブルを使用）

キャッシュを経由しない書き込みを指示
(for Intel Compiler)

メモリ上連続な領域から計算

図 4-1 Xeon Phi 向けに最適化されたカーネル部分

これらの最適化を全て適用した結果、オリジナルコードに比べ、Xeon Phi 上での実行性能が大幅に向上した。この結果を表 4-1 に示す。なお、最適化する前の同コード部分の性能は 30.06GFLOPS であったが、これが最終的に 224.5GFLOPS となり、約 7.5 倍の性能向上を達成した。

表 4-1 ARTED カーネルコード（MPI プロセス内）の最適化と性能向上

OpenMP 240 Thread	GFLOPS	ピーク性能比
自動ベクトル化実装	130.4	12.15 %
初期実装	113.7	10.58 %
複素数積最適化 (A)	114.9	10.70 %
非アラインアクセス最適化 (B)	179.4	16.70 %
インデックス計算ベクトル化 (C)	142.7	13.29 %
(B) + (C)	221.4	20.61 %
全最適化適用時 (A + B + C)	224.5	20.90 %

本研究では、この最適化コードをベースに、Xeon Phi だけでなくホスト CPU の Xeon でも波数空間分割したプロセスを実行し、COMA 上の 2 基の Xeon CPU と 2 枚の Xeon Phi カードという、ノード内の全リソースを利用した高性能計算を実行した。2 基の Xeon は NUMA アーキテクチャによる共有メモリ結合がされているが、メモリの局所性を有効活用するため、各 CPU に 1 つずつの MPI プロセスを配置し、スレッド並列は CPU 内で閉じるようにした。結果的に、ノード当たり 4 つの MPI プロセスを走らせることとした。ただし、Xeon と Xeon Phi では後者の方が性能が高いため、MPI プロセスに割り当てる波数を調整し、負荷分散を実施した（静的負荷分散）。この結果、SiO_2 を対象とした COMA 上の ARTED の実行で 128 ノードまでの strong scaling を実現した。ノード数を増加した場合のタイムステップ当たりの実行時間を図 4-2 に示す。

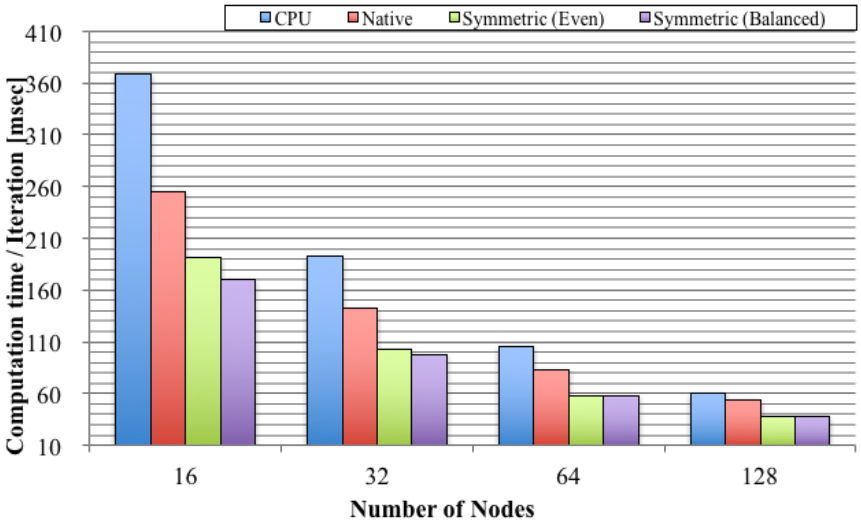


図 4-2 ARTED の COMA 上の実行時間(strong scaling)

図 4-2 では、最適化された CPU のみの実行、Xeon Phi を Native mode で実行した Xeon Phi のみの実行、CPU と Xeon Phi を両方使った Symmetric mode での実行、そして CPU と Xeon Phi の負荷分散を考慮して最適化した Symmetric mode での実行を示している。16 ノードの場合、負荷最適化された Symmetric mode 実行では、CPU のみの場合に比べ約 2.2 倍の性能向上となった。しかし、ノード数が増えるにつれ、それぞれのケースでの性能差が小さくなっている。

これは、strong scaling における通信時間ボトルネックが発生しているのではなく、Xeon Phi 上の演算処理量が小さくなり、ハードウェアが求める十分な処理量が確保できなくなり、効率が低下したことによるものである。

今後はより大きな物質と実空間サイズ・波数空間サイズを持つような問題を対象とし、Oakforest-PACS 導入時には KNL プロセッサでの大規模高性能処理が実現できるよう準備をしていく予定である。

【5】 TCA/Infiniband ハイブリッド通信（朴）

PEACH2 を効率的に利用する研究として、並列アクセラレータ記述言語 XcalableACC において、PEACH2 実装された TCA 通信網と従来の InfiniBand を並行して利用する研究を行った。今年度は、通信の pairwise などの最適化を行ったことで、袖領域交換（Halo）の実行時間が以前より短縮した。また、後述するハイブリッド通信による集団通信（Allreduce）を適用することで、集約（Reduction）の実行時間が短縮している。特に、Reduction の実行時間が InfiniBand（IB）通信に対して高速であるため、IB に対してハイブリッド通信は Himeno ベンチマークの問題サイズ Small（64×64×128）において最大 41%の性能向上を達成した（図 5-1）。

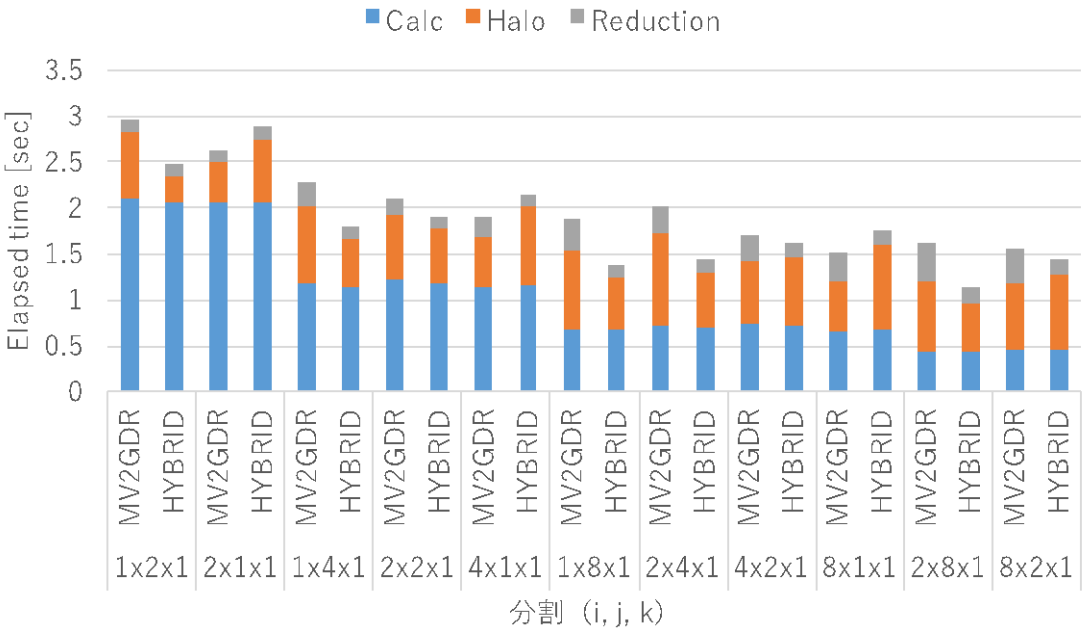


図 5-1 Himeno ベンチマーク（問題サイズ Small）

更に、集団通信（Allgather, Bcast, Allreduce）に対するハイブリッド通信の実装を行った。ハイブリッド通信による集団通信では、通信サイズに対して適切な通信パス（TCA/PEACH2 or IB）を選択することで、高い性能が得られている。それぞれ、IB に対して

ハイブリッド通信は最大 47%（図 5-2）、42%（図 5-3）、21%（表 5-1）の性能向上を達成している。

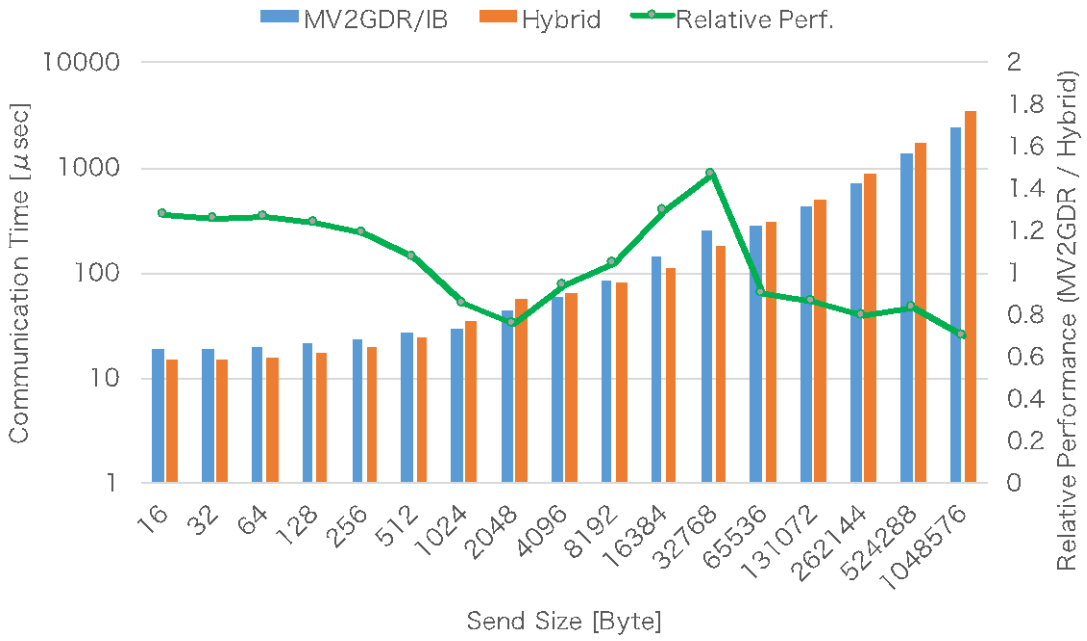


図 5-2 Allgather（8 ノード）

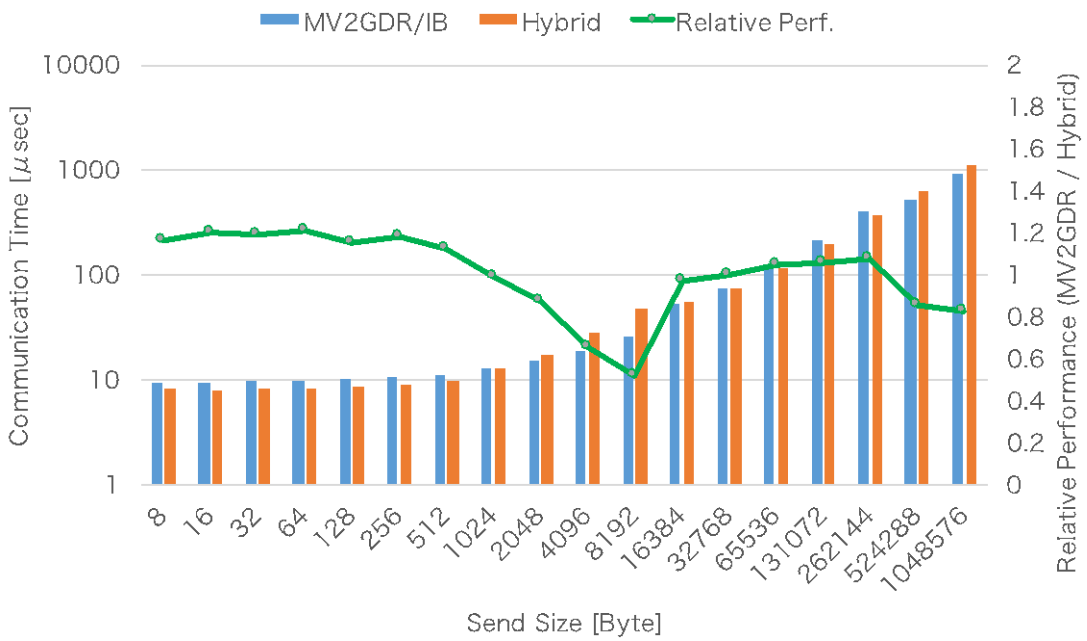


図 5-3 Bcast（8 ノード）

表 5-1 Allreduce の性能［μsec］

	MPI_Allreduce	Hybrid_Allreduce	Speedup (MPI/Hybrid)
8 ノード	18.322	13.390	1.36
16 ノード	22.835	16.007	1.42

また、ソースコードの行数を比較した結果、Hand-coding に対しておおよそ半分程度のコード行数で記述できることも確認した。**表** 5-2 に Himeno ベンチマークにおける SLOC の比較を示す。

表 5-2Himeno ベンチマーク：ソースコード行数

	MPI	TCA	OpenACC	XMP	Init	Calc	Sum
Serial					72	27	99
Hand-coding	141	16	18		110	29	345
XACC			12	23	88	29	152

【6】 XcalableMP における動的タスク並列記述（朴、佐藤[CCS フェロー]）

XcalableMP（XMP）2.0 から採用される予定である、分散メモリ環境向けの動的タスク並列機能の記述方法の提案と、その予備評価を行った。記述方法として tasklet 指示文を追加し、ブロックコレスキー分解のコードを用いて関連研究である StarPU と性能・生産性の比較を行った。

分散メモリ環境における動的なタスク並列機能を実現するためには、ノードを跨ぐタスク間の依存関係を記述する必要がある。しかし、ノード内における依存関係の記述は OpenMP 4.0 から登場した task 指示文の depend 節で表すことが可能だが、ノードを跨ぐタスク間の依存関係を表す機能は MPI、OpenMP 共に存在しない。そのため本研究では、ノード内で実行されるタスク内で MPI による通信を発生させ、その通信の完了によりタスクの実行を開始する方法をノード間での依存関係とした。図 6-1 にノードを跨ぐタスク間の依存関係の例を示す。taskA にて演算した配列 A[0:25]を taskB が用いる場合、taskA と taskB の間には実行順序に依存関係が存在する。この場合、ノード 1 では taskA の演算終了後に配列 A[0:25]を送信し、ノード 4 では taskB に到達した際に taskA から値の受信完了（配列 B は受信用のバッ

ファを表す）により実行を開始する。したがって、タスク内で必要な値の授受により分散メモリ環境における依存関係を指定することが可能である。

XMP において共有メモリ向けにプログラミングをする場合は、OpenMP と組合せて記述する必要がある。分散メモリ環境における動的タスク並列機能を実現する場合は、XMP の task 指示文により実行ノードを決定、XMP の task ブロック内にて OpenMP の task 指示文および depend 節によりノード内のタスクを生成し、OpenMP の task ブロック内にて演算や XMP の通信を記述する。通信には、XMP の分散配列を対象とした gmove 指示文や coarray が用いられる。これらの記述では、通信以外はほぼ MPI+OpenMP と同様の記述が求められるため、プログラムは煩雑になりやすい。そのため、XMP に新たな指示文として tasklet 指示文を追加し、上記の処理を単一の指示文で記述可能とすることで生産性の向上を目指す。

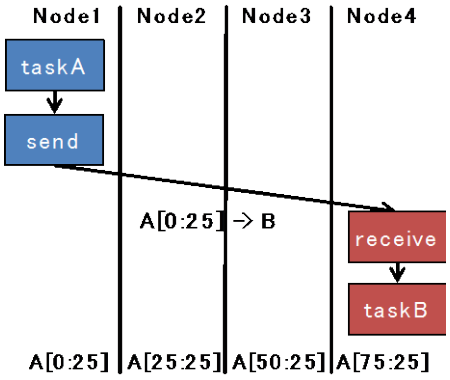


図 6-1 分散メモリ環境におけるタスク依存の例

図 6-2 に tasklet 指示文によるプログラミング例を示す。tasklet 指示文は、ユーザが OpenMP 同様の依存関係の記述に加え、ノードを跨いだ依存関係の場合の通信相手の記述、受信データを格納するバッファや実行ノードを指定する。依存記述は OpenMP task depend 節同様に in、out および inout とし、合わせて XMP の分散配列を記述する。通信相手は、XMP が提供するテンプレート、ノード集合で抽象的に記述し、さらにタグでより詳細な通信制御が可能である。図 2 の場合は、実行ノードは on 節により taskA はノード 1、taskB はノード 4 である。また、out 節に記述されたノード集合により taskA が実行終了後に MPI_Send により配列 A[0:25]をノード 4 へ送信し、in 節に記述されたノード集合、バッファによりノード 1 から MPI_Recv により受信した配列 A[0:25]を配列 B へ格納し、taskB の演算を開始する。

tasklet 指示文を用いたプログラムの評価には、筑波大学計算科学研究センターの COMA を用いた。1 ノードあたり 1MPI プロセスを割り当て、最大 16 ノード 16MPI プロセスとする。また、ノード内のスレッド数は 1 から 16 へと変動させる。対象としたプログラムはブロックコレスキー分解を行うコードである。tasklet 指示文と関連研究である StarPU で記述したコ

ードを実装し、性能・生産性の比較を行う。StarPU は、フランス国立情報学自動制御研究所で開発が進められている、ヘテロジニアスな環境におけるタスク並列機能を提供するランタイムライブラリである。また、本研究は tasklet 指示文の予備評価であるため、実際にコンパイラに機能を実装したのではなく、コンパイラによるコード変換を想定したコードを実装し、評価を行った。

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

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

#pragma xmp taskletwait
```

図 6-2　XMP tasklet 指示文によるプログラミング例

図 6-3 に行列サイズ 8192×8192、ブロックサイズ 128×128 のブロックコレスキー分解のコードの評価を示す。縦軸のラベルである XMP はコンパイラによるコード変換を想定した MPI+OpenMP コードである。結果として、スレッド数が 8、16 の場合において XMP コードの性能が良く、StarPU と比較して 16 プロセス、各プロセスにおけるスレッド数が 8 の場合に最大で 1.54 倍の性能を得た。しかし、スレッド数が 1、2、4 の場合の性能は低く、現在その原因を調査中である。

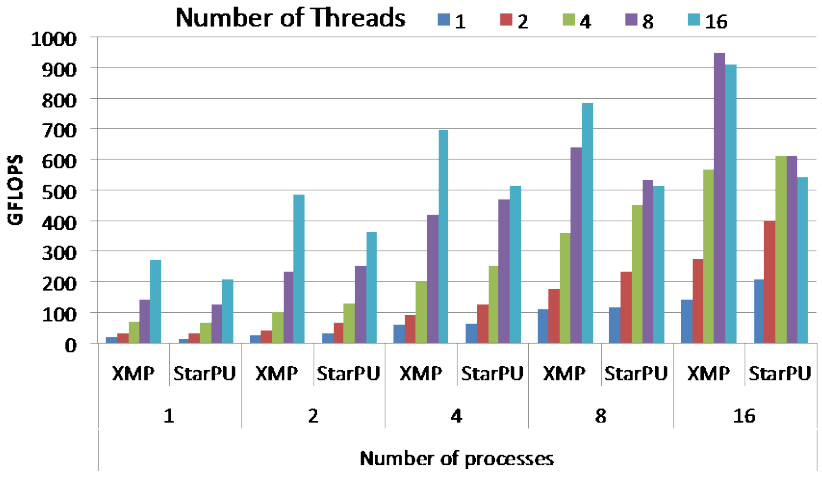


図 6-3 ブロックコレスキー分解の評価（行列サイズ 8192×8192）

生産性の評価には各実装のコード行数を用いた。XMP 実装は、グローバルビューモデルによる簡易なデータ分割に加え、OpenMP とほぼ同等の記述でタスク依存が記述可能な tasklet 指示文により、StarPU と比較して全体でコード行数を 85%に抑えられた。したがって、tasklet 指示文の生産性の高さを示すことができた。

今後の課題として、現在の実装は Send/Recv ベースな実装であるため、PGAS 言語と親和性の高い片側通信を用いた実装を行うことが挙げられる。また、指示文内のみではなく、ユーザがタスク内で片側通信や XMP の通信を記述可能とすることでより自由度が高く詳細なチューニングが可能な記述を考えている。

なお、本研究は CCS と理化学研究所計算科学研究機構とのポスト京システム開発共同研究の一環として行われた。

【7】 大規模並列システム性能予測ツール SCAMP（朴、佐藤[CCS フェロー]）

ポスト京システムを含む次世代超並列計算システムでは、ノード上のコア数の増強に加え、ノード数自体もさらに数倍～数十倍に増強されると見込まれる。理化学研究所・計算科学研究機構では以前より、プロセッサ性能の向上に対して既存のプロセッサのプロファイルを利用した性能向上予測を進めているが、本研究ではこれと直交して、同一プログラム・同一問題サイズの並列処理プロファイルを利用して、より大規模なシステムでの性能予測を行う手法とこれに基づくツールを開発する。同システムは SCAMP (Scalable MPI Profiler)と名付けられている。

SCAMP の概念は、MPI で記述されたプログラムの実システムでの計算及び通信プロファイルを元に、これがより大規模、すなわちより多くのノード数とプロセス数のシステムで実行された場合の「仮想 MPI プロファイル (Pseudo MPI Profile)」をシステマティックに作成し、これを並列システムシミュレータにかけることで、実行時間の予測を行うものである。シミュレータとしては、Sandia National Laboratory で開発された SST/macro を用いる。

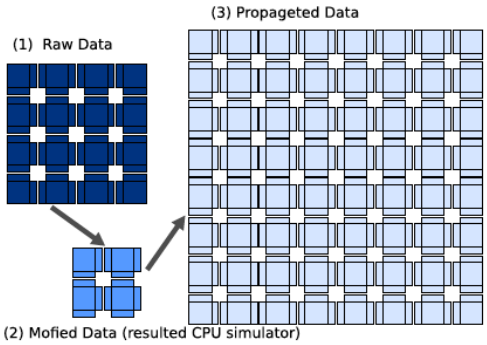


図 7-1 SCAMP で扱う MPI profile の概念

図 7-1 に SCAMP で扱う MPI profile の概念を示す。ここでは 2 次元ステンシル計算を取り上げる。Raw Data は実プログラムを実機で実行して得られるプロファイルで、これから CPU simulator で得られた「MPI プロセス数を増やした場合のプロセス当たりの実行時間が短縮された」プロファイルを生成したものが Modified Data、そして最後に MPI プロセス数を増加させたことで通信相手やメッセージサイズが変わった状況でのプロファイルが Propagated Data である。並列システムにおいてノード数（MPI プロセス数）が増えた場合、強スケーリングでは一般的に CPU 実行時間が短縮され、MPI 通信量の増加と通信オーバヘッドの増加が生じる。この状況を、CPU simulator による再評価と MPI プロセス数の増加を想定して「水増し」されたプロファイルによって表現するというのが SCAMP の手法である。この手法は当然、弱スケーリングにも適用できる。弱スケーリングの場合は CPU の実行時間が基本的に変化しないと想定し、MPI プロファイルの水増しのみを行うため、より簡便かつ正確なシミュレーションが実行できる。よって、平成 27 年度の研究では、弱スケーリングの場合のみを対象とした。

```
#define XSIZE 1000

np = MPI_Comm_size();
myid = MPI_Comm_rank();
xsize = XSIZE / np;
ux = malloc(sizeof(double), xsize + 2);
uux = malloc(sizeof(double), xsize + 2);
MPI_Send(&ux[1],1,MPI_DOUBLE,..,(myid-1+np)%np,.);
MPI_Send(&ux[xsize],1, MPI_DOUBLE,..,(myid+1)%np,.);
MPI_Recv(&ux[xsize+1],1,MPI_DOUBLE,..,(myid+1)%np,.);
MPI_Recv(&ux[0],1,MPI_DOUBLE,...,(myid-1+np)%np,.);
for(i=1; i<=xsize; i++)
    uux[i]=(ux[i-1]+ux[i+1])-2.0*u[i];
for(i=1; i<=xsize; i++)
    ux[i]=uux[i];
```

図 7-2 SCAMP で対象とするプログラムコードの典型例

SCAMP の手法は基本的にシンプルであり、特に SIMD 性の高いデータ並列型の処理を想定している。ターゲットコードの例を図 7-2 に示す。これは 2 次元ステンシルコードの典型例であり、MPI 通信の相手は全プロセス数（np）と自プロセスのランク（myid）で決定される。多くのデータ並列型プログラムでは、このように通信相手が自動的に決定される場合がほとんどで、これが MPI プロファイルの水増しのメカニズムである。より大規模なシステムを想定した、水増しされたプロファイルは図 7-3 のように通信相手ランクが変更される。

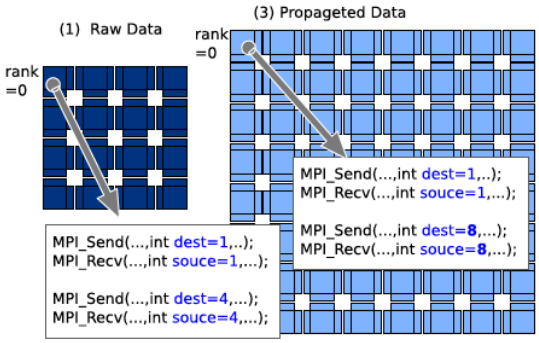


図 7-3 水増しされた MPI プロファイルのイメージ

QCD や Himeno Benchmark のような多次元ステンシル計算では、隣接プロセス間の point-to-point 通信と、全プロセス間での Allreduce 通信が必要となる。そこで、これらの通信パターンについて実際に京コンピュータ上で得たプロファイルと、SCAMP の手法で水増しされたプロファイルを SST/macro でシミュレーションした結果を図 7-4 に示す。

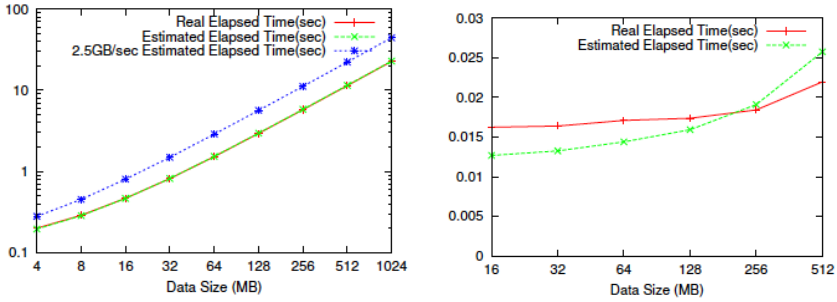


図 7-4 ping-pong 通信と Allreduce 通信の実機測定結果と水増しプロファイルのシミュレーション結果の比較

図 7-4 の左側の図は 2 プロセス間の pingpong 通信の実機評価と、京コンピュータの通信パラメータをモデル化して SST/macro の通信シミュレータで実行した場合の比較である。各種メッセージサイズに対してほぼ一定の差が出ており、これを補正するために実効通信バンド幅を補正すると、性能がほぼ一致することがわかる。また、右側の図ではデータサイズを変えた場合の Allreduce 時間を示しており、データサイズによって性能が逆転していることがわかる。これは、データサイズに依存した通信プロトコルの変化が SST/macro シミュレーション上では考慮されないことによると推測される。

これらの基本的通信性能評価を踏まえ、NPB-CG、NPB-FT、CCS-QCD 等のベンチマークを評価した。ここでは実アプリケーションに近いベンチマークとして CCS-QCD の結果を示す。図 7-5 は 2 つの問題サイズ 4x4x4 と 8x8x8 について、弱スケーリングを想定した場合、すなわち後者は前者の 8 倍 (=2x2x2) のプロセス数で実行し、CPU 時間は変わらないという想定の下で、ステンシル計算の隣接通信と Allreduce 通信を適用した。

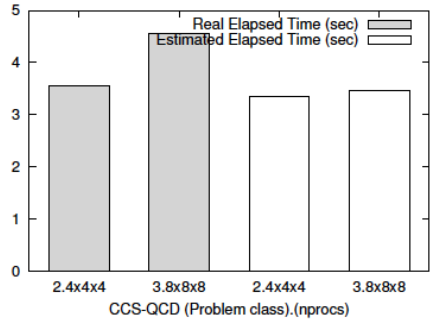


図 7-5 CCS-QCD ベンチマークの弱スケーリング評価結果

この結果より、比較的予測が簡単な弱スケーリングの場合でも、8 倍サイズのシステムの予測性能が実機性能より 20%強程度高く出ていることがわかる。この差は前述の Allreduce 時間の差や、実機でのジッター等のプロセス実行時間の微妙な差が影響していると思われるが、計算時間の予測はできていると結論できる。

現在、SCAMP のツール群を実装中であり、現時点では多くの作業が手作業によっている。平成 28 年度研究では、MPI プロファイルの水増しツール（現在は line editor を使って半自動的に生成）の構築と、ソースコードをコンパイラで解析することで、MPI 実行環境から得られるプロセス数等のパラメータからプロファイル水増しのためのパラメータを自動抽出する方法等を研究する予定である。

なお、本研究は CCS と理化学研究所計算科学研究機構とのポスト京システム開発共同研究の一環として行われた。

【8】 並列 FFT における通信隠蔽の自動チューニングに関する研究（高橋）

科学技術計算で広く用いられている高速フーリエ変換（FFT）の性能を改善するために、通信隠蔽の自動チューニングに関する研究を行った。分散メモリ型並列計算機においてチューニングを行う際に，最適な性能パラメータはプロセッサのアーキテクチャ、ノード間を結合するネットワーク、そして問題サイズなどに依存するため、これらのパラメータをその都度手動でチューニングすることは困難になりつつある。自動チューニングを適用した FFT ライブラリとして、FFTW や SPIRAL などが知られているが、通信隠蔽において自動チューニングを適用した FFT ライブラリはまだ提案されていないのが現状である。そこで、並列 FFT において通信隠蔽のパラメータを自動チューニングする手法について検討した。並列 FFT において演算と通信を分割しパイプライン方式でオーバーラップさせる場合、通信メッセージサイズの分割数（パイプラインの段数）を大きくすれば、オーバーラップの割合が高くなる。その一方で、通信メッセージサイズの分割数を大きくすれば、通信 1 回あたりの通信メッセージサイズが小さくなるため、通信バンド幅も小さくなる。また、演算と通信をオーバーラ

ップさせる際には、通信によってメモリバンド幅が消費される。したがって通信メッセージサイズの分割数には最適な値が存在すると考えられる。また、FFT を計算する際の基底や、行列の転置を行う際のブロックサイズについても自動チューニングを行うことが可能である。

図 8-1 に並列一次元 FFT の性能を示す。図 6-11 から、$N \leq 2^{29}$ の場合には、FFTE 6.1alpha（no overlap）と FFTE 6.1alpha with AT（自動チューニング）がほぼ同じ性能となっているが、$N \geq 2^{30}$ の場合には通信隠蔽の効果により FFTE 6.1alpha with AT の性能が高くなっていることが分かる。また、FFTE 6.1alpha（NDIV=4）では通信メッセージサイズが常に 4 分割されており全対全通信性能が低くなっていることから、$N \leq 2^{29}$ の場合には FFTE 6.1alpha（no overlap）よりも性能が低くなっていることが分かる。

分散メモリ型並列計算機における並列一次元 FFT において、通信メッセージサイズの分割数、基底の組み合わせ、そしてブロックサイズについて自動チューニングを行うことで、性能をさらに向上できることが分かった。

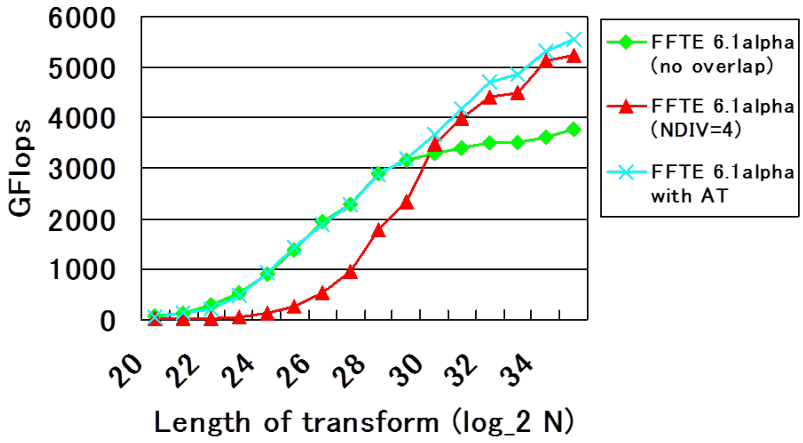


図 8-1 並列一次元 FFT の性能（Fujitsu PRIMEHPC FX100，512 ノード）

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

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

[分散ファイルシステム]

研究の狙いは、CPU コア数の増加に対し，アクセス性能がスケールアウトし，かつアクセス応答時間が長くならない分散ファイルシステムの研究開発を行うことである。本年度は、ファイルデータの冗長化について研究開発を行った。

分散ファイルシステムのアクセス性能をスケールアウトさせるために、計算ノードのローカルストレージを利用する場合、計算ノードの障害によりそのノードのローカルストレージに格納されたデータにアクセス不能となってしまう。この問題を解決するためこれまでファイル複製を他ノードのローカルストレージに保持する方法がとられたが、この方法ではストレージ領域が 2 倍以上必要となってしまう、データ転送量が 2 倍以上となってしまうという問題があった。この問題を解決するため、ノード間で冗長符号を用いることが考えられる。しかしながらこれまではノード間での冗長符号を書き込み時に行うことは冗長符号計算のオーバヘッドが大きく行われていなかった。そのため、書き込み時は冗長符号化のオーバヘッドがない複製を作成し、その後、冗長符号に変換する等の方法が用いられていた。この問題を根本的に解決するため、平成 26 年度にストレージに冗長符号計算とデータ転送をオフロードしたアクティブストレージを提案した。平成 27 年度は、本方式を RAID-5 に拡張した。図 9-1 にそのデータ転送パターンを示す。

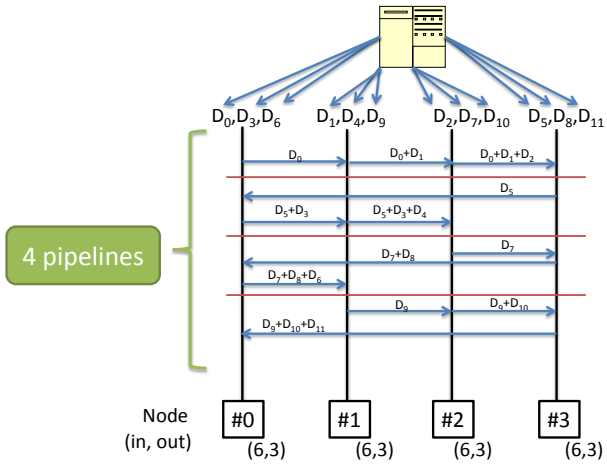


図 9-1　RAID-5 におけるアクティブストレージのデータ転送パターン

RAID-5 の場合は、3 ブロックずつに異なるストレージサーバにパリティブロックを持つため、図 9-1 のように 4 つの独立したパイプラインを構成することとなる。ストレージサーバ間のデータ転送を別ネットワークで行った 3D1P のアクティブストレージによる RAID-5 の性能を図 9-2 に示す。4 クライアント以上からのアクセスにおいて、3D の RAID-0 の性能を凌ぐ性能を示した。

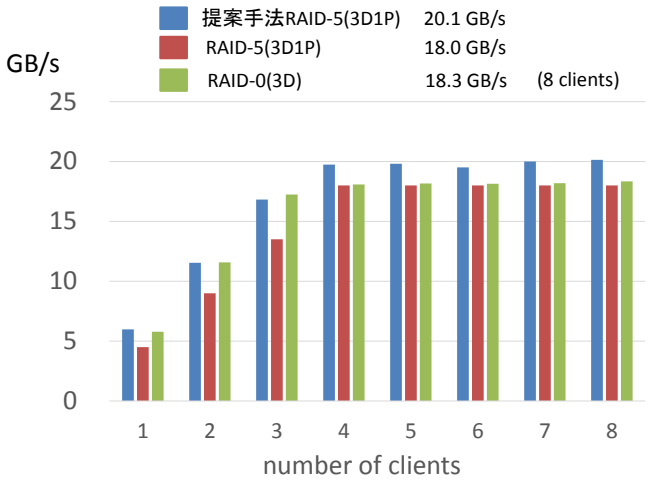


図 9-2 アクティブストレージ RAID-5 の性能評価

本成果は、平成 27 年 12 月に IEEE データサイエンスとデータインテンシブシステムに関する国際会議（DSDIS）で発表した。

[大規模データ処理実行基盤]

本研究項目ではデータインテンシブサイエンスのアプリケーションを効率的に実行するため，MPI-IO，大規模ワークフロー実行，MapReduce 処理，バッチキューイングシステム，データベース管理システムなどの実行環境の研究開発を行う。本研究提案で研究開発する分散ファイルシステムは，全体としてのファイルアクセス性能はスケールアウトさせるように設計、実装を行っているが，ファイルアクセス性能が非均一である。そのため，効率的に利用するためには，ファイルアクセスの局所性の利用とデータ移動を最小化するプロセススケジューリングが重要となる。さらに，データ移動と計算処理はオーバラップが可能であり，プロセススケジューリングとデータ移動のスケジューリングを最適化することにより，更に効率的な実行が可能となる。

バッチキューイングシステムの研究では、引き続きデータアクセス局所性を高めるためのスケジューリング手法の研究を進めた。昨年度は I/O インテンシブなジョブだけではなく CPU インテンシブなジョブについても扱うため、スケジューリング手法の拡張を行った。それぞれのジョブにおける入力ファイルからアクセス局所性の指標を求め、CPU の負荷と併せてパラメタ β によりノード割り当てを行う手法の設計を行った。今年度は最適な β の値を自動的に求めるための研究を行った。成果の一部は平成 27 年 5 月にグリーン、パーベイシブ、クラウドコンピューティングに関する国際会議（GPC）で発表し、IEEE Systems Journal への掲載が決定している。

【10】エクストリームビッグデータの基盤技術（建部、川島）

エクストリームビッグデータ（EBD）アプリケーションの実行に求められる、数万～数十万プロセスからの並列アクセスを想定した IOPS、プロセス数に比例した読込、書込アクセスバンド幅性能を目標として、分散オブジェクトストアの設計の最適化をすすめた。

平成 27 年度は、これまでに行った OpenNVM を用いたローカルオブジェクトストアの設計を改善したプロトタイプ実装と他の実装との性能比較を行った。提案している OpenNVM を用いたオブジェクトストアでは、オブジェクト生成性能が 16 スレッドで 747,000 ops/sec であった。性能評価した結果を図 10-1 に示す。directFS は OpenNVM を用いたファイルシステムである。従来のファイルシステムに比べ性能は高いものの、スレッド数を増やしても性能は変化しない。性能は 4 スレッドのときが最高で 74,000 ops/sec であり、我々の提案手法の方が 10 倍高速であった。NVMKV と RocksDB は Key Value ストアであり、今回設計したオブジェクトストアとはインタフェースが異なるものの、ストレージとして同様の機能を持つため比較を行った。スレッド数を増やすと多少は性能が向上するものの、最高でも NVMKV が 16 スレッドのときの 173,000 ops/sec であり、我々の提案手法の方が 4.3 倍高速であった。これらの成果を含め情報処理学会 ACS 論文誌に採録が決定している。

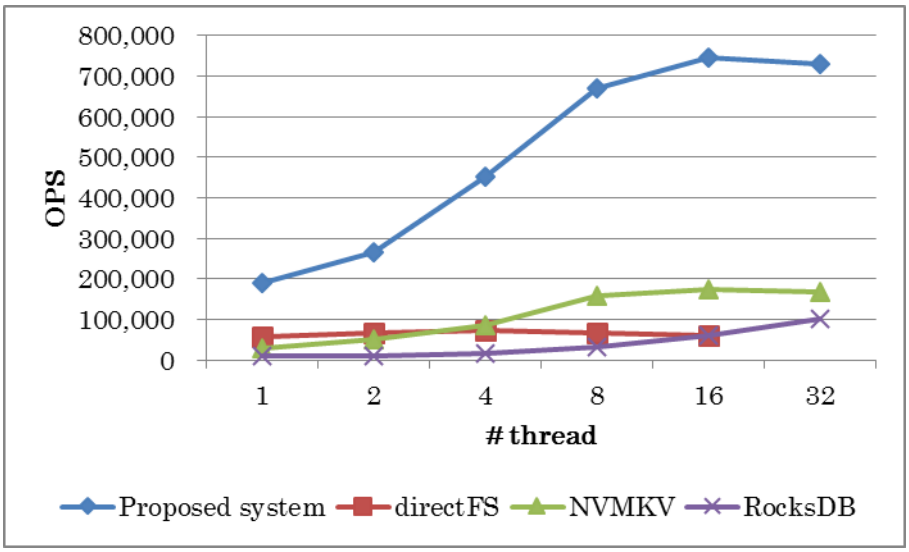


図 10-1 オブジェクト作成の性能評価

範囲検索可能な Key Value ストアの設計では、OpenNVM で提供される Key Value ストアに対して範囲検索を可能とするためにインメモリの B+木を設計している。これまでは並列に B+木をアクセスしたときに性能を向上させるため、ロックフリーな並列 B+木の設計を行ってきた。検索、挿入、削除に対し、B+木をロックすることなく並列に実行することが可能である。B+木の再平衡化は動的になされ、検索はその再平衡化により遅延しない。H27 年度は、

B+木の範囲検索を並列に行うため、空間局所性を持った並列連結リストの設計を行った。範囲検索を効率的に行うためには B+木のノードをつなぐ連結リストが必要であり、設計したロックフリーな並列 B+木のノードに対する並列連結リストの設計が必要となるためである。基本的にはロックフリーのアプローチをとり、compare and swap 命令を用いる。設計した並列連結リストを図 10-2 に示す。

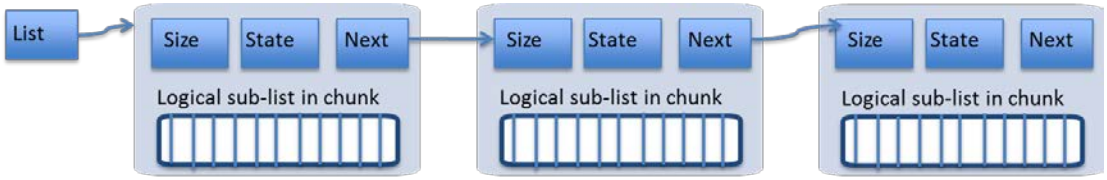


図 10-2 空間局所性を持った並列連結リスト

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

文部科学省が進める革新的ハイパフォーマンスコンピューティングインフラ（HPCI）の HPCI 共用ストレージ、素粒子物理学データ共有システム JLDG のシステムソフトウェアとしても利用される Gfarm ファイルシステムの研究開発を行った。本年度は自動的に余剰ファイル複製を削除する機能、HPSS（階層型ストレージシステム）への高速コピー機能、書込後の自動ベリファイ機能についての設計と実装を行った。Gfarm ファイルシステムは、ファイルサーバの障害などによりファイル複製数が減ると自動的に必要な数のファイル複製を作成する機能を持っている。ここで、障害を起こしたファイルサーバが復旧したときなど、ファイル複製数が必要数より増えてしまう。また、必要なファイル複製数を減らした場合も、必要数以上のファイル複製を持つこととなる。この余剰ファイル複製を自動的に削除する機能を設計した。ファイル複製削除にあたり、ファイルをアクセス中であることなどを考慮する必要がある。ファイルをアクセス中の場合は、ファイル削除を一時延期することとした。利用頻度が低いファイルは階層型ストレージへ移動するが、その移動を高速に行うため、並列にファイルコピーを行う高速ファイルコピー機能を設計、実装した。書込後の自動ベリファイ機能は、サイレントデータ破損への対策である。サイレントデータ破損を検出するためには、一度読み出してダイジェストを計算する必要があるが、一度も読込まれないファイルはその検査を行うことができない。そのため、これまでは手動で全ての書込みファイルに対し読込を行っていた。この作業を自動化するための機能である。ただし、書込後、すぐに読み出すとバッファキャッシュから読み出してしまうため、サイレントデータ破損の検査を行うことができない。そのため、デフォルトでは自動的に 6 時間後に読み出して検査を行う機構を設計、実装した。その他、不具合修正、ドキュメントの更新なども行った。この結果として、Gfarm バージョン 2.6.8 をリリースした。

【12】 天文データ処理に関する研究（建部、川島）

1. すばる望遠鏡データの高速処理化

すばる望遠鏡に搭載された超広視野主焦点カメラ Hyper Suprime-Cam (HSC)から得られる画像データを処理することで、研究者や一般人にとって有益な情報を有する画像を導出することができる。このデータ処理には長時間が必要である。一日分の入力データを処理するために 24 時間程度の処理時間が必要である。この処理時間を短くすることが研究の狙いである。

昨年度は解析パイプラインシステムを理解するために、IPMU 側との打合せを行い、現在のパイプラインシステムのプログラムならびに入力データを入手した。その後、プログラムの実行ならびに分析を行った。今年度は 2 つの研究を実施した。詳細を以下に述べる。

[Pwrake を用いた解析パイプラインの高速化]

本年度は解析パイプラインに Pwrake を適用した。Pwrake (Parallel Workflow extension for RAKE) とは、我々が開発しているワークフローシステムである。このシステムは Rake という Ruby 版ビルドツールをベースに、並列分散実行の機能を拡張したものである。特にデータインテンシブなワークフローを目的として、Gfarm ファイルシステムを利用し、ローカリティを高めるスケジューリングにより、高い I/O 性能を発揮する処理を可能にする。Pwrake と Gfarm を適用する様子を図 12-1 に示す。図の左側は現状を示し、ファイルアクセスがボトルネックになる様子が表されている。右側は Pwrake と Gfarm によりファイルアクセスが並列化されている様子が表されている。

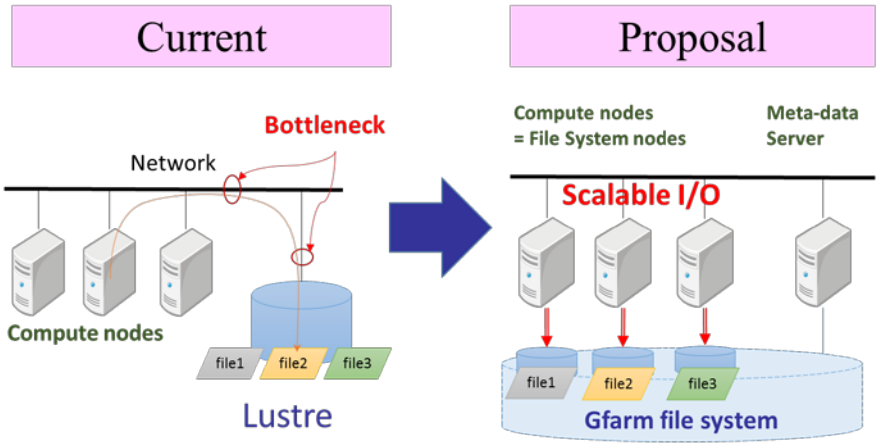


図 12-1 Pwrake と Gfarm の解析パイプラインへの適用

Pwrake を解析パイプラインに適用したところ，性能を改善することができた．CPU 利用率に関する性能向上に関する結果を図 12-2 に示す．図はいずれも横軸が実行時間，縦軸が使われているプロセスの数を示す．左図はオリジナルの解析パイプライン，右図は Pwrake を適

用したパイプラインの結果を示している．両図を比較すると右図では実行プロセス数が減らない一方，左図では実行プロセス数が時間の経過と共に増減を繰り返すことがわかる．また，右の場合は左の場合に比べて実行時間が短く，性能に優れることがわかる．

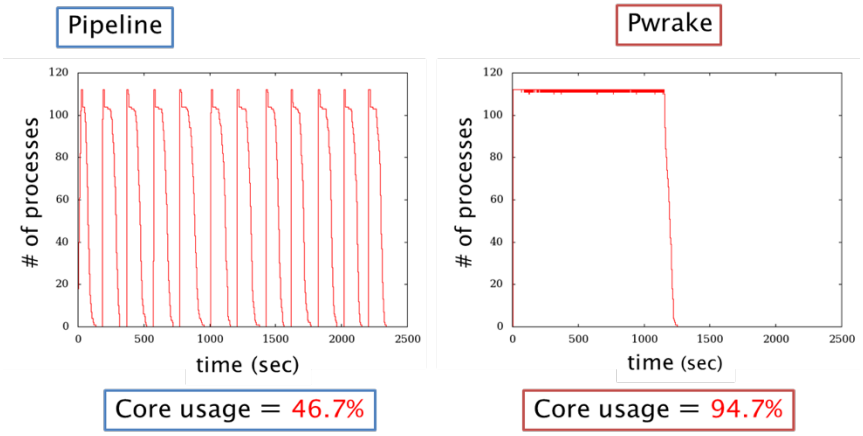


図 12-2　Pwrake 利用による起動プロセス数減少問題の解決

また，利用ノード数/CPU コア数を変動させた場合の実験結果を図 12-3 に示す．図の縦軸は実行時間，左側はオリジナルパイプラインの結果，右側は Pwrake を適用したパイプラインの結果を示している．図ではパイプラインを構成する各処理に要した時間も示されている．図 12-3 より次のことがわかる．まず，Pwrake は実行時間をおよそ 2/3 まで削減できている．次に，その性能改善は Frames に改善によるところが大きい．この理由は，Frames の並列化は容易だからである．

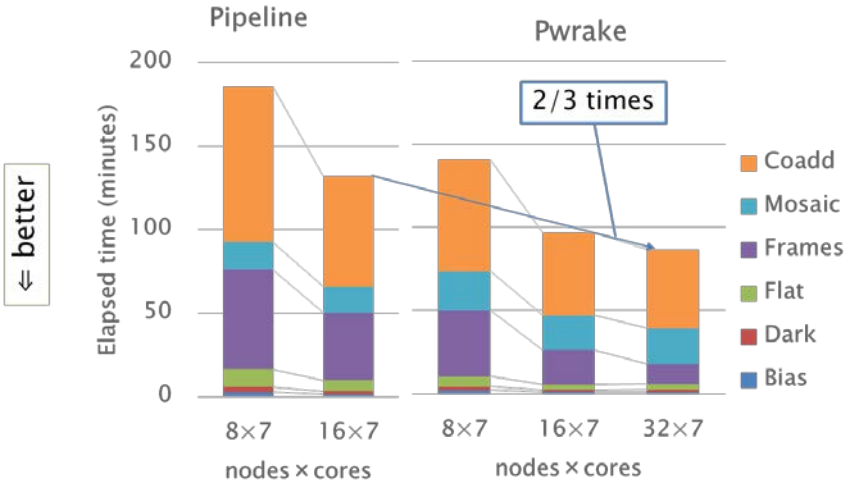


図 12-3　解析パイプラインの性能改善

[Gfarm 高性能化のための高速トランザクション処理]

Pwrake は Gfarm と連動して動作する。Gfarm は高い IO 並列度を有するが、メタデータサーバへのアクセス数が増加すると、そこがボトルネックとなり、データアクセスの並列度が向上しなくなる。この問題を解決するにはメタデータサーバで実行されるトランザクション処理を高速化する必要がある。そこで今年度は並列ログ先行書込技術 P-WAL を開発した。これを図 12-4 に示す。図の左側が従来 WAL であり、複数のログレコードを 1 つのログファイルへ一括して書込んでいる。図の右側が P-WAL であり、ログファイルへの書込みを並列的に実行している。

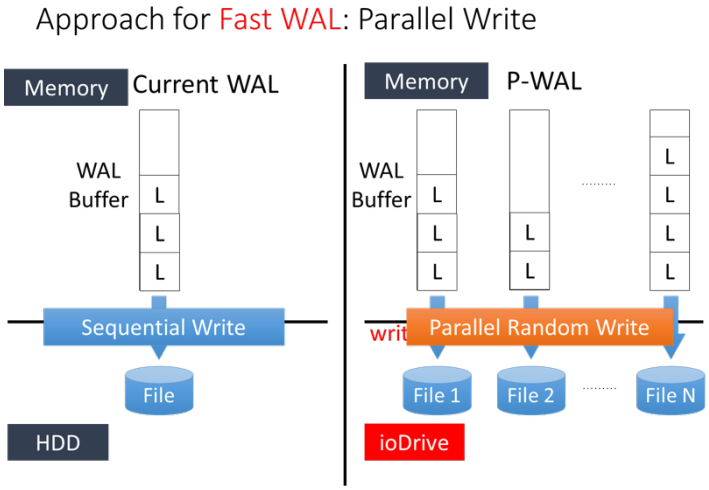


図 12-4　P-WAL：並列ログ先行書込

高速ストレージである ioDrive を用いた P-WAL の性能評価の結果を図 12-5 に示す。この図より P-WAL は従来技法よりも 2.43 倍程度の性能改善を示していることがわかる。

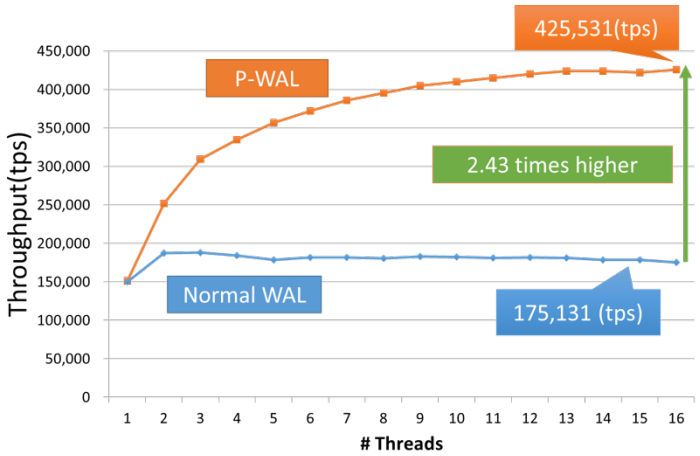


図 12-5　P-WAL の性能評価

2. 高速データベース構築

解析パイプラインから出力される結果は、天体カタログとして管理される。天体カタログの例としては SDSS が著名である。SDSS において、データはリレーショナルデータベースに登録され、 SQL での問合せが行われる。本研究の狙いはこれらの問合せを高速化することである。本件について 2 件の研究を行った。詳細を下記に述べる。

[リレーショナルデータベースにおける問合せ処理の高速化]

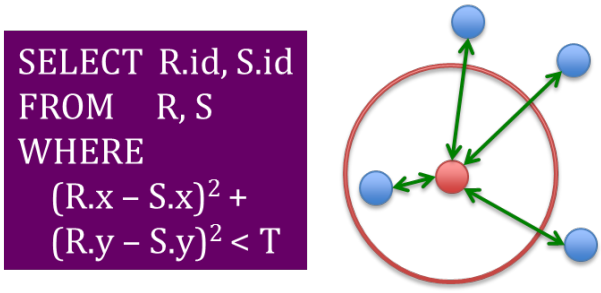


図 12-6　近傍星を探す問合せ

天体カタログでは様々な問合せ処理が行われる。初年度に IPMU と議論した結果、頻繁に使われるという近傍検索について取上げることにした。近傍検索とは、ある星の近い星を求めるという問合せに該当する。図 12-6 にこれを示す。図の左側は SQL ライクな問合せ表現、右側は直感的な表現であり、赤い問合せ点からユークリッド距離で一点範囲の星を探す状況を示している。この場合には 1 つのみが検索結果として返却される。空間索引と環結合を利用すると近傍探索処理を高速化できることが 26 年度でわかった。ただし検索結果数の変動に伴い性能向上がどのように変動するのかはわからなかった。そこで、27 年度では検索結果数によって空間索引の検索性能がどのように変わるのか測定した。その上で、検索結果数が大きな場合でも小さな場合でも検索性能が向上するように環結合と空間索引を組合せた実装とその評価を行った。

Q1: SELECT R.id, S.id FROM R, S WHERE (R.x - S.x)2 + (R.y - S.y)2 < r2

上記 Q1 に該当するワークロードの性能を 5 つの手法について評価した結果を図 12-7 に示す。横軸は選択率（0.01%は 10^8）、縦軸は実行時間を表す。データサイズは R, S ともに 100 万件である。CycloJoin については 10 台を分散処理用に用いた。R-tree ならびに R*-tree はシングルマシンで動作させているが、検索は SIMD とマルチコアを用いて並列実行している。CycloJoin と空間索引の組合わせ（CycloJoin-RTree, CycloJoin-R*Tree）が最速である。また、R*-tree は R-tree よりもかなり低速である。これは索引構築に要する時間が長いからである。また、このモジュールを PostgreSQL から呼び出すために、PostgreSQL の Nested Loops Join をフックする拡張モジュールを開発し、それを GitHub で公開した。

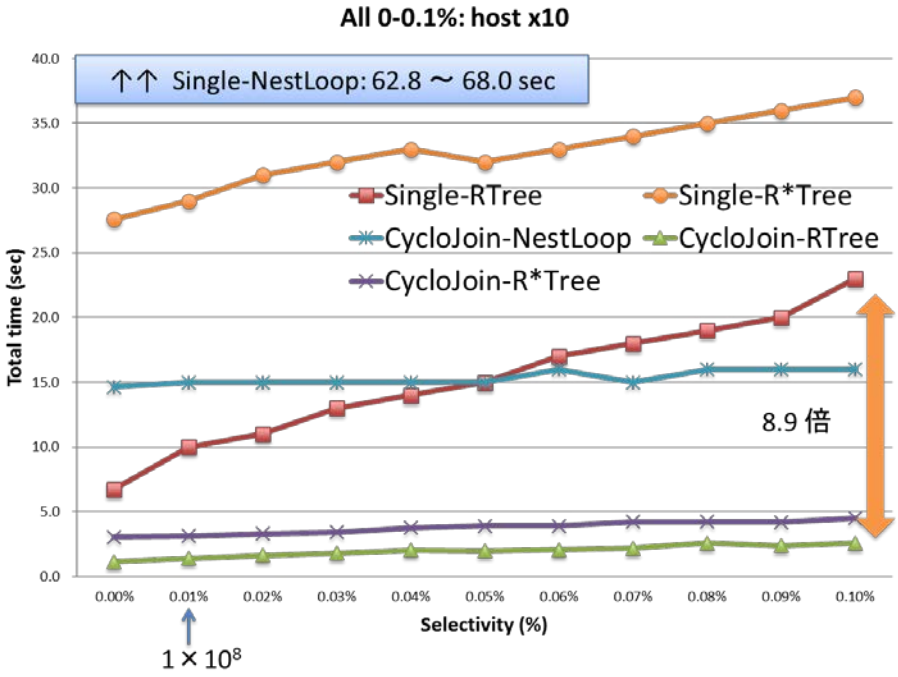


図 12-7　CycloJoin による近傍探索の高速化

[NoSQL データベースにおける問合せ処理の高速化]

NoSQL である配列データベースである集約演算の高性能化技法を検討した。再帰的インクリメンタル計算に基づく集約演算の高性能化技法を構築し、SciDB 上に実装を行い、その性能を確かめた。具体的には配列データベースにおけるウィンドウ集約演算（最大、最小、平均、総和、分散、標準偏差）を高性能化する手法を提案した。高性能化のためにインクリメンタル計算スキームを適用した。提案手法を代表的な配列データベースである SciDB に実装して実験的に評価した。その結果、提案手法は最大で従来手法に比べて 225 倍高速だった。実験に用いたコードは GitHub に公開した。

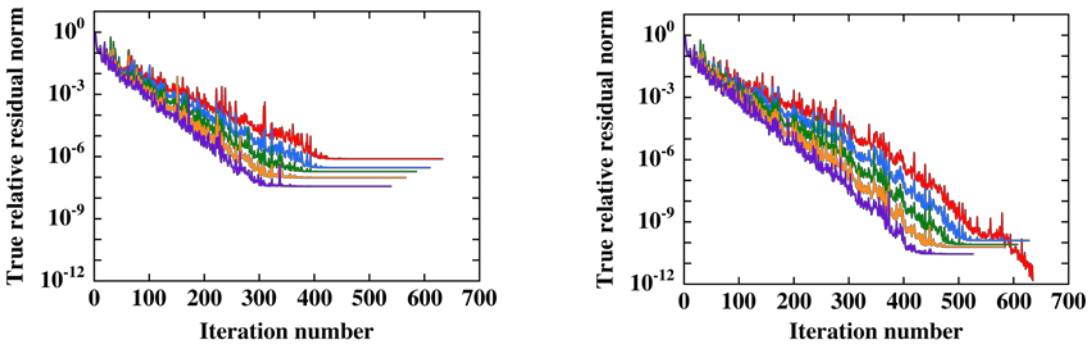
【13】　複数右辺ベクトル・複数シフトパラメータをもつ連立一次方程式の数値解法に関する研究（多田野）

複数右辺ベクトル，及び係数行列にシフトパラメータをもつ連立一次方程式（以下，シフトブロック連立一次方程式）の数値解法である，Shifted Block Krylov 部分空間反復法に関する研究を行った．1 本の右辺ベクトル，及びシフトパラメータをもつ連立一次方程式の数値解法として，Shifted Krylov 部分空間反復法がある．同法は，メインの連立一次方程式（以下，シード連立一次方程式）を解く過程で，他のシフトパラメータをもつ連立一次方程式（以下，シフト連立一次方程式）の近似解更新を少ない計算量で行うことができる．一方，複数右辺ベクトルをもつ連立一次方程式に対する数値解法である Block Krylov 部分空間反復

法は，各右辺に対して Krylov 部分空間反復法を適用するよりも，少ない反復回数で近似解が得られることがある．本研究では，Shifted Krylov 部分空間反復法と Block Krylov 部分空間反復法を組み合わせた，Shifted Block Krylov 部分空間反復法を構築した．

本研究では，我々がこれまで開発してきた Block Krylov 部分空間反復法である，Block BiCGGR 法と Block BiCGSTAB(l)法を基盤として，シフトブロック連立一次方程式に適用するための方法の拡張を行った．この拡張により，Shifted Krylov 部分空間反復法を各右辺ベクトルに対して適用する場合と比較して，シフトブロック連立一次方程式をより少ない反復回数で解くことが可能となった．しかしながら，最終的に得られる近似解の精度が劣化する問題点が発見された．近似解更新部分で現れる計算の大部分は，縦長行列と小行列の積であるため，この計算がなるべく現れないようにアルゴリズムを修正し，近似解の精度向上を図った．

図 13-1 に Shifted Block BiCGGR 法の単純な実装と改良法の真の相対残差履歴を示す．テスト問題は，格子量子色力学計算（QCD）で現れる連立一次方程式（行列サイズ：1,572,864，非零要素数：80,216,064）で，右辺ベクトル数を 12 とし，係数行列のシフトパラメータ σ は，0.0×10^{-3}，2.0×10^{-3}，4.0×10^{-3}，6.0×10^{-3}，8.0×10^{-3} の 5 つを用いた．真の相対残差は近似解の精度の指標であり，この値が小さい場合は高精度の近似解が得られていることを示している．単純な実装の場合は，真の相対残差が 10^{-7} 付近で停滞しており，高精度近似解が得られていない．一方，改良法を用いた場合は，単純な実装よりも全体的に真の相対残差が小さくなっており，高精度近似解が得られている．

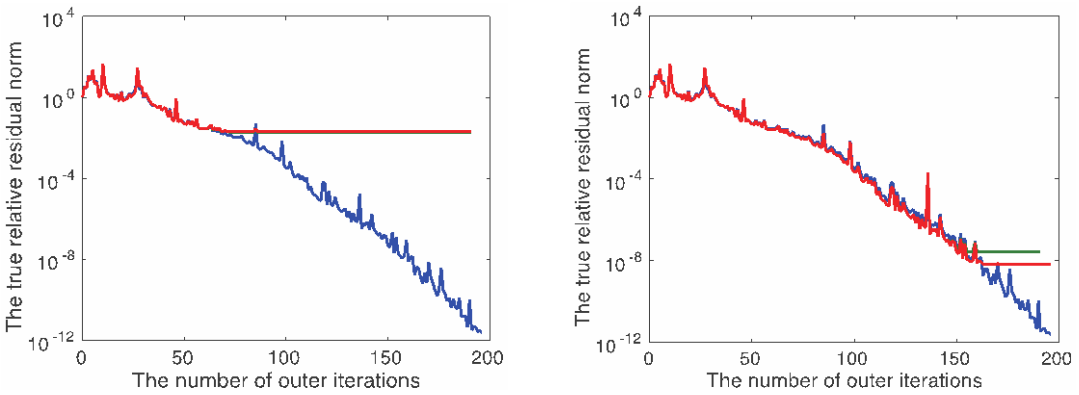


(a) 単純な実装　　(b) 改良法

図 13-1　シフトブロック連立一次方程式 $(A+\sigma I)X^{(\sigma)}=B$ に対する Shifted Block BiCGGR 法の真の相対残差履歴．■：$\sigma = 0.0\times10^{-3}$，■：$\sigma = 2.0\times10^{-3}$，■：$\sigma = 4.0\times10^{-3}$，■：$\sigma = 6.0\times10^{-3}$，■：$\sigma = 8.0\times10^{-3}$．

図 13-2 に Shifted Block BiCGSTAB(l)法の単純な実装と改良法の真の相対残差履歴を示す．テスト問題は，The University of Florida Sparse Matrix Collection の行列 Coupled（行

列サイズ：11,341，非零要素数：97,193）で，右辺ベクトル数は 16 とし，係数行列のシフトパラメータ σ は 1.0×10^{-3} とした．また，Shifted Block BiCGSTAB(l)法のパラメータ l は 2 とした．単純な実装，改良法の両方について，シード連立一次方程式の真の相対残差は十分に小さくなっている．単純な実装では，シフト連立一次方程式の真の相対残差は，倍精度計算，一部疑似 4 倍精度利用計算の両方とも 10^{-2} 付近で停滞しており，近似解の精度は極めて低い．一方，改良法においては倍精度で計算したシフト連立一次方程式の相対残差は 10^{-7} 付近まで減少している．さらに一部の計算に疑似 4 倍精度を用いることにより，真の相対残差は 10^{-8} 付近まで減少しており，高精度の近似解を生成することができた．



(a) 単純な実装　　(b) 改良法

図 13-2　シフトブロック連立一次方程式 $(A+\sigma I)X^{(\sigma)}=B$ に対する Shifted Block BiCGSTAB(l)法の真の相対残差履歴．ここで，■：シード連立一次方程式，■：シフト連立一次方程式（倍精度計算），■：シフト連立一次方程式（一部疑似 4 倍精度利用）．

4. 教育

博士学位論文

1. 小田嶋哲哉、博士（工学）、高並列言語による演算加速器及び相互結合網の効率的利用に関する研究、筑波大学大学院システム情報工学研究科博士論文、平成 28 年 3 月（指導：朴泰祐）
2. 藤田典久、博士（工学）、GPU クラスタにおけるアプリケーション高速化に関する研究、筑波大学大学院システム情報工学研究科博士論文、平成 28 年 3 月（指導：朴泰祐）
3. 大辻弘貴、博士（工学）、A Study on High-performance and Reliable Distributed Storage System（高性能かつ信頼性の高い分散ストレージシステムに関する研究）、筑波大学大学院システム情報工学研究科博士論文、平成 28 年 3 月（指導：建部修見）

修士学位論文

1. 廣川祐太、修士（工学）、メニーコア型プロセッサにおける計算科学アプリケーションの最適化に関する研究、筑波大学大学院システム情報工学研究科修士論文、平成 28 年 3 月（指導：朴泰祐）
2. 篠塚敬介、修士（工学）、融合積和演算命令を利用した中心-半径型区間演算に基づく精度保証付き高速フーリエ変換、筑波大学大学院システム情報工学研究科修士論文、平成 28 年 3 月（指導：高橋大介）
3. 前田広志、修士（工学）、アクセラレータを用いた疎行列ベクトル積の高速化、筑波大学大学院システム情報工学研究科修士論文、平成 28 年 3 月（指導：高橋大介）
4. 齋藤周作、修士（工学）、複数右辺・複数シフトを持つ線形方程式を高速に解く反復法の構築とその精度改善に関する研究、筑波大学大学院システム情報工学研究科修士論文、平成 28 年 3 月（指導：高橋大介）
5. 古江、修士（工学）、Implementation of Collective I/O on MPI-IO over Gfarm File System（分散ファイルシステム Gfarm 上での MPI-IO の Collective I/O の実装）、筑波大学大学院システム情報工学研究科修士論文、平成 28 年 3 月（指導：建部修見）
6. 蒋立、修士（工学）、Efficient Window Aggregate Processing for Multi-dimensional Scientific Big Data in Array Databases（アレイデータベースにおける多次元科学的ビッグデータ向けの効率的なウィンドウ集約処理）、筑波大学大学院システム情報工学研究科修士論文、平成 28 年 3 月（指導：建部修見）

卒業論文

1. 大畠佑真、学士（工学）、OpenCL に基づく FPGA プログラミングによる高性能計算に関する研究、筑波大学情報学群情報科学類卒業論文、平成 28 年 3 月（指導：朴泰祐）
2. 桐井祐樹、学士（工学）、並列離散イベントシミュレーションを用いた大規模分散システムの性能評価、筑波大学情報学群情報科学類卒業論文、平成 28 年 3 月（指導：建部修見）
3. 五味歩武、学士（工学）、自動変換によるプログラムの可読性を維持した最適化および移植の省力化、筑波大学情報学群情報科学類卒業論文、平成 28 年 3 月（指導：高橋大介）
4. 大黒晴之、学士（工学）、In-memory MapReduce の高速化、筑波大学情報学群情報科学類卒業論文、平成 28 年 3 月（指導：川島英之）
5. 村田直郁、学士（工学）、RDMA を用いた分散トランザクション処理の高速化、筑波大学情報学群情報科学類卒業論文、平成 28 年 3 月（指導：川島英之）

6. 瀧沢亮太、学士（工学）、リレーショナルデータベースシステムにおける並列ハッシュ結合の評価、筑波大学情報学群情報科学類卒業論文、平成 28 年 3 月（指導：川島英之）

集中講義

1. 「CCS HPC セミナー」（一般公開、全学共通科目）、2015 年 2 月 1 日～2 日
2. “HPC Seminar”（全学共通科目）及び “Japan Korea HPC Winter School”（韓国 KISTI との共催）2015 年 2 月 15 日～17 日

5. 受賞、外部資金、知的財産権等

受賞

1. Outstanding Research Awards: Yuta Kuwahara, Toshihiro Hanawa, Taisuke Boku, “A proposal of GMPI: GPU self MPI for GPU clusters”, 情報処理学会 ACSI2016 シンポジウム, 福岡, 2016 年 1 月 18 日.
2. The Best Technical Talk Award: Hiroki Ohtsuji, Osamu Tatebe, “Breaking the Trade-off between Performance and Reliability of Network Storage System”, PRAGMA29, Depok, Indonesia, Oct. 7, 2015
3. 最優秀学生発表賞: 神谷孝明, “P-WAL: 並列ログ先行書き込みの提案”、情報処理学会システムソフトウェアとオペレーティング・システム研究会、2015 年 11 月 25 日.
4. 平成 27 年度日本応用数理学会論文賞 JSIAM Letters 部門, J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, K. Kimura, “A numerical method for nonlinear eigenvalue problems using contour integrals”, JSIAM Letters, Vol. 1, pp. 52-55, 2009.

外部資金

（名称、氏名、代表・分担の別、採択年度、金額、課題名）

1. JST CREST、朴泰祐（代表）、H24 年度～H29 年度、29,000 千円 (H27)、研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」，「ポストペタスケール時代に向けた演算加速機構・通信機構統合環境の研究開発」
2. 理化学研究所共同研究、朴泰祐（代表）、H27 年度～H31 年度、8,000 千円(H27)「ポスト京の並列プログラミング環境およびネットワークに関する研究」
3. JST CREST、高橋大介（共同研究者）、H23～28 年度、15,600 千円(H27)、研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」,「数値計算ライブラリによる超並列複合システムの階層的抽象化に関する研究」

4. JST CREST、建部修見（代表）、H23 年度～H28 年度、31,709 千円 (H27)、研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」, 「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」
5. JST CREST、建部修見（共同研究者）、H25～30 年度、5,200 千円 (H27)、研究領域「ビッグデータ統合利活用のための次世代基盤技術の創出・体系化」, 「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」
6. KDDI 研究所共同研究、川島英之（代表）、H27 年度、1,000 千円、「データ仮想化システムにおける大規模データ処理方式に関する研究」
7. JST CREST、川島英之（共同研究者）、H26～30 年度, 5,500 千円（H27）、研究領域「科学的発見・社会的課題解決に向けた各分野のビッグデータ利活用推進のための次世代アプリケーション技術の創出・高度化」, 「広域撮像探査観測のビッグデータ分析による統計計算宇宙物理学」
8. 科学研究費補助金　基盤研究 (B)、川島英之（分担）、H24～26 年度、800 千円（H26 年度）、「高性能計算リソースの抽象化を実現するランタイムシステム」
9. 科学研究費補助金　若手研究 (B)、多田野寛人（代表）、H27～28 年度、1,690 千円（H27 年度）、「複数右辺ベクトルをもつ連立一次方程式に対する高精度・高効率アルゴリズムの開発」

知的財産権

なし

6. 研究業績

(1) 研究論文

A) 査読付き論文

1. 小田嶋哲哉, 朴泰祐, 塙敏博, 児玉祐悦, 村井均, 中尾昌広, 田渕晶大, 佐藤三久, “アクセラレータ向け並列言語 XcalableACC における TCA/InfiniBand ハイブリッド通信”, 情報処理学会論文誌コンピューティングシステム（ACS）, Vol. 8, No.4, pp. 61-77, 2015 年 11 月.
2. 松本和也, 塙敏博, 児玉祐悦, 藤井久史, 朴泰祐, “密結合並列演算加速機構TCAによる GPU 間直接通信における Collective 通信の実装と性能評価,” 情報処理学会論文誌コンピューティングシステム（ACS）, Vol. 8, No.4, pp. 36-49, 2015 年 11 月.
3. Tetsuya Odajima, Taisuke Boku, Toshihiro Hanawa, Hitoshi Murai, Masahiro Nakao, Akihiro Tabuchi, Mitsuhisa Sato, “Hybrid Communication with TCA

and InfiniBand on A Parallel Programming Language for Accelerators XcalableACC", Proc. of HUCAA2015 (in Cluster2015), Chicago, 2015.

4. Toshihiro Hanawa, Norihisa Fujita, Tetsuya Odajima, Kazuya Matsumoto, Taisuke Boku, "Evaluation of FFT for GPU Cluster Using Tightly Coupled Accelerators Architecture", Proc. of HUCAA2015 (in Cluster2015), Chicago, 2015.
5. Toshihiro Hanawa, Hisafumi Fujii, Norihisa Fujita, Tetsuya Odajima, Kazuya Matsumoto, Yuetsu Kodama, Taisuke Boku, "Improving Strong-Scaling on GPU Cluster Based on Tightly Coupled Accelerators Architecture", Proc. of IEEE Cluster2015 (short paper), Chicago, 2015.
6. Kazuya Matsumoto, Toshihiro Hanawa, Yuetsu Kodama, Hisafumi Fujii, Taisuke Boku, "Implementation of CG Method on GPU Cluster with Proprietary Interconnect TCA for GPU Direct Communication", Proc. of AsHES2015 in IPDPS2015, Hyderabad, 2015.
7. T. Amagasa, S. Aoki, Y. Aoki, T. Aoyama, T. Doi, K. Fukumura, N. Ishii, K. Ishikawa, H. Jitsumoto, H. Kamano, Y. Konno, H. Matsufuru, Y. Mikami, K. Miura, M. Sato, S. Takeda, O. Tatebe, H. Togawa, A. Ukawa, N. Ukita, Y. Watanabe, T. Yamazaki and T. Yoshie , "Sharing lattice QCD data over a widely distributed file system", Journal of Physics: Conference Series, 664, 8 pages, 2015 (DOI:10.1088/1742-6596/664/4/042058)
8. Hiroki Ohtsuji, Osamu Tatebe, "Network-based Data Processing Architecture for Reliable and High-performance Distributed Storage System", Euro-Par 2015: Parallel Processing Workshops, Lecture Notes in Computer Science, Vol. 9523, pp.16-26, 2015 (DOI: 10.1007/978-3-319-27308-2_2)
9. Shin Sasaki, Kazushi Takahashi, Yoshihiro Oyama, Osamu Tatebe, "RDMA-based Direct Transfer of File Data to Remote Page Cache," Proceedings of 2015 IEEE International Conference on Cluster Computing (Cluster), pages 214-225, 2015. (DOI: 10.1109/CLUSTER.2015.40)
10. Hiroki Ohtsuji, Osamu Tatebe, "Server-side Efficient Parity Generation for Cluster-wide RAID System", Proceedings of 7th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp.444-447, 2015 (DOI: 10.1109/CloudCom.2015.25) (short paper)

11. Hiroki Ohtsuji and Osamu Tatebe, “Active-Storage Mechanism for Cluster-wide RAID System”, Proceedings of IEEE International Conference on Data Science and Data Intensive Systems (DSDIS), pp.25-32, 2015
(DOI: 10.1109/DSDIS.2015.101)
12. Shin Sasaki, Ryo Matsumiya, Kazushi Takahashi, Yoshihiro Oyama, Osamu Tatebe, “RDMA-based Cooperative Caching for a Distributed File System,” Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp.344-353, 2015
(DOI: 10.1109/ICPADS.2015.51)
13. Yasin Oge, Masato Yoshimi, Takefumi Miyoshi, Hideyuki Kawashima, Hidetsugu Irie, Tsutomu Yoshinaga, “Design and Evaluation of a Configurable Query Processing Hardware for Data Streams”, IEICE Transactions on Information and Systems, Vol. E98-D, No. 12, pp.2207-2217, December 2015.
14. L. Su, A. Imakura, H. Tadano, T. Sakurai, “Improving the convergence behaviour of the BiCGSTAB method by applying D-norm minimization”, JSIAM Letters, Vol. 7, pp. 37-40, 2015.

B) 査読無し論文

なし

(2) 国際会議発表

A) 招待講演

1. Taisuke Boku, “Extreme-SIMD Accelerator toward Exascale Computing”, International HPC Forum China 2015, Tianjin, May 20, 2015.
2. Taisuke Boku, “HPC Status Report from Japan”, HPC in Asia Session at ISC2015, Frankfurt, Jul. 15, 2015.
3. Taisuke Boku, “Case Study on PBS Pro Operation on Large Scale Scientific GPU Cluster”, PBS Works 2015, Mountain View, Sep. 16, 2015.
4. Taisuke Boku, “Accelerator+Communication+Language: New Age of Extreme Parallel Computing”, CODESIGN Workshop 2015 (in HPC China), Wuxi, Nov 10, 2015.
5. Taisuke Boku, “Communication and Calculation Co-design for HPC on FPGA”, ANL-FPGA Workshop 2015, Argonne, Jan. 21, 2016.

6. Daisuke Takahashi, “Sparse Matrix-Vector Multiplication on GPUs”, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing (EPASA2015), Tsukuba International Congress Center, Tsukuba, Japan, September 14, 2015.

B) 一般講演

1. Tetsuya Odajima, Taisuke Boku, Toshihiro Hanawa, Hitoshi Murai, Masahiro Nakao, Akihiro Tabuchi, Mitsuhisa Sato, “Hybrid Communication with TCA and InfiniBand on A Parallel Programming Language for Accelerators XcalableACC”, Proc. of HUCAA2015 (in Cluster2015), Chicago, 2015.
2. Toshihiro Hanawa, Norihisa Fujita, Tetsuya Odajima, Kazuya Matsumoto, Taisuke Boku, “Evaluation of FFT for GPU Cluster Using Tightly Coupled Accelerators Architecture”, Proc. of HUCAA2015 (in Cluster2015), Chicago, 2015.
3. Toshihiro Hanawa, Hisafumi Fujii, Norihisa Fujita, Tetsuya Odajima, Kazuya Matsumoto, Yuetsu Kodama, Taisuke Boku, “Improving Strong-Scaling on GPU Cluster Based on Tightly Coupled Accelerators Architecture”, Proc. of IEEE Cluster2015, Chicago, 2015.
4. Kazuya Matsumoto, Toshihiro Hanawa, Yuetsu Kodama, Hisafumi Fujii, Taisuke Boku, “Implementation of CG Method on GPU Cluster with Proprietary Interconnect TCA for GPU Direct Communication”, Proc. of AsHES2015 in IPDPS2015, Hyderabad, 2015.
5. Hiroshi Maeda and Daisuke Takahashi, “Performance Evaluation of Sparse Matrix-Vector Multiplication Using GPU/MIC Cluster”, Proc. 2015 Third International Symposium on Computing and Networking (CANDAR'15), 3rd International Workshop on Computer Systems and Architectures (CSA'15), pp. 396-399 (2015).
6. Daisuke Takahashi, “Automatic Tuning for Parallel FFTs on Intel Xeon Phi Clusters”, 2016 Conference on Advanced Topics and Auto Tuning in High-Performance and Scientific Computing (2016 ATAT in HPSC), National Taiwan University, Taipei, Taiwan, Feb. 19, 2016.
7. Xieming Li and Osamu Tatebe, “Data-Aware Task Dispatching”, the 10th International Conference on Green, Pervasive and Cloud Computing (GPC 2015), Fiji, May 4, 2015

8. Fuyumasa Takatsu, Kohei Hiraga, and Osamu Tatebe, “Design of object storage using OpenNVM for high-performance distributed file system”, the 10th International Conference on Green, Pervasive and Cloud Computing (GPC 2015), Fiji, May 4, 2015
9. Yuki Kirii, Hiroki Ohtsuji, Kohei Hiraga, Osamu Tatebe, “Simulation of PPMDS: A Distributed Metadata Management System”, Summer of CODES Workshop, Argonne, USA, Jul. 13, 2015
10. Hiroki Ohtsuji, Osamu Tatebe, “Network-based Data Processing Architecture for Reliable and High-performance Distributed Storage System”, 4th Workshop on Big Data Management in Clouds (BigDataCloud), Vienna, Austria, Aug. 24, 2015
11. Shin Sasaki, Kazushi Takahashi, Yoshihiro Oyama, Osamu Tatebe, “RDMA-based Direct Transfer of File Data to Remote Page Cache,” 2015 IEEE International Conference on Cluster Computing (Cluster), Chicago, USA, Sep. 10, 2015
12. Hiroki Ohtsuji, Osamu Tatebe, “Breaking the Trade-off between Performance and Reliability of Network Storage System”, PRAGMA29, Depok, Indonesia, Oct. 7, 2015 (The Best Technical Talk Award)
13. Hiroki Ohtsuji, Osamu Tatebe, “Server-side Efficient Parity Generation for Cluster-wide RAID System”, 7th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Vancouver, Canada, Dec. 3, 2015
14. Hiroki Ohtsuji and Osamu Tatebe, “Active-Storage Mechanism for Cluster-wide RAID System”, IEEE International Conference on Data Science and Data Intensive Systems (DSDIS), Sydney, Australia, Dec. 11, 2015
15. Shin Sasaki, Ryo Matsumiya, Kazushi Takahashi, Yoshihiro Oyama, Osamu Tatebe, “RDMA-based Cooperative Caching for a Distributed File System,” 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS), Melbourne, Australia, Dec. 17 2015
16. H. Tadano, S. Saito, A. Imakura, “Accuracy improvement of the Shifted Block BiCGGR method for linear systems with multiple shifts and multiple right-hand sides”, Poster Session, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing (EPASA2015), Tsukuba, Japan, Sep., 2015.

17. H. Tadano, S. Saito, A. Imakura, “Development of the Shifted Block BiCGGR method and accuracy improvement of approximate solutions”, International Conference on Simulation Technology (JSST2015), Toyama, Japan, Oct., 2015.

(3) 国内学会・研究会発表

A) 招待講演

1. 朴泰祐, “JCAHPCにおける国内最大PCクラスタの導入と運用に向けて”, PCクラスタワークショップ, 仙台, 2016 年 2 月 19 日.

B) その他の発表

1. Yuta Kuwahara, Toshihiro Hanawa, Taisuke Boku, “A proposal of GMPI: GPU self MPI for GPU clusters”, ACSI2016, 2016. (ACSI2016 Outstanding Research Award)
2. 松本和也, 塙敏博, 児玉祐悦, 藤井久史, 朴泰祐, “密結合並列演算加速機構TCAを用いたGPU間直接通信によるCollective通信の実装と性能評価”, 2015年ハイパフォーマンスコンピューティングと計算科学シンポジウム（HPCS2015）論文集, 2015.
3. 津金佳祐, 朴泰祐, 村井均, 佐藤三久, William Tang, Bei Wang, “PGAS 言語 XcalableMP のハイブリッドビューによる核融合シミュレーションコードの実装と評価”, 2015 年ハイパフォーマンスコンピューティングと計算科学シンポジウム (HPCS2015) 論文集, 2015.
4. 廣川祐太, 朴泰祐, 佐藤駿丞, 矢花一浩, “電子動力学シミュレーションコードの Symmetric モードによる Xeon Phi クラスタ への実装と性能評価”, 情報処理学会第 153 回 HPC 研究会報告 2016-HPC-153, 2016 年 3 月.
5. 佐藤賢太, 藤田典久, 塙敏博, 松本和也, 朴泰祐, Ibrahim Khaled, “密結合並列演算加速機構 TCA による GPU 対応 GASnet の実装”, 情報処理学会第 153 回 HPC 研究会報告 2016-HPC-153, 2016 年 3 月.
6. 辻美和子, 李珍泌, 朴泰祐, 佐藤三久, “SCAMP: MPI 疑似通信プロファイル によるネットワーク性能推定手法の提案”, 情報処理学会第 153 回 HPC 研究会報告 2016-HPC-153, 2016 年 3 月.
7. 廣川祐太, 朴泰祐, 佐藤駿丞, 矢花一浩, “Xeon Phi クラスタにおける Symmetric 並列実行による電子動力学シミュレーションの性能評価”, 情報処理学会第 151 回 HPC 研究会報告 2015-HPC-151, 11 pages, 2015 年 10 月.
8. 桑原悠太, 塙敏博, 朴泰祐, “GMPI : GPU クラスタにおける GPU セルフ MPI の提案”, 情報処理学会第 151 回 HPC 研究会報告 2015-HPC-151, 8 pages, 2015 年 10 月.

9. 佐藤賢太, 藤田典久, 塙敏博, 朴泰祐, “密結合演算加速機構 TCA における Verbs 実装による MPI 環境の実現”, 情報処理学会第 150 回 HPC 研究会報告 (SWoPP2015) 2015-HPC-150, 8 pages, 2015 年 8 月.
10. 松本和也, 塙敏博, 藤田典久, 朴泰祐, “密結合並列演算加速機構 TCA による並列 GPU コードの性能予測モデル”, 情報処理学会第 150 回 HPC 研究会報告 (SWoPP2015) 2015-HPC-150, 6 pages, 2015 年 8 月.
11. 小田嶋哲哉, 朴泰祐, 塙敏博, 村井均, 中尾昌広, 田渕晶大, 佐藤三久, “アクセラレータ向け並列プログラミング言語 XcalableACC における TCA/InfiniBand?ハイブリッド通信”, 情報処理学会第 150 回 HPC 研究会報告 (SWoPP2015) 2015-HPC-150, 12 pages, 2015 年 8 月.
12. 飯島大貴, 廣川祐太, 塙敏博, 朴泰祐, “密結合演算加速機構 TCA アーキテクチャの Intel MIC プロセッサへの適用”, 2015-HPC-149, 2015.
13. 高橋大介, “Xeon Phi における並列 FFT の実現と評価”, 日本応用数理学会 2015 年度年会講演予稿集 (2015).
14. 高橋大介, “Xeon Phi における多倍長精度浮動小数点演算の実現と評価”, 日本応用数理学会 2015 年度年会講演予稿集 (2015).
15. 椋木大地, 今村俊幸, 高橋大介, “NVIDIA GPU におけるメモリ律速な BLAS カーネルのスレッド数自動選択手法”, 情報処理学会研究報告, Vol. 2015-HPC-150, No. 13 (2015).
16. 高橋大介, “Xeon Phi クラスタにおける並列 FFT の自動チューニング”, 計算工学講演会論文集, Vol. 20, E-2-2 (2015).
17. 椋木大地, 今村俊幸, 高橋大介, “NVIDIA GPU における GEMV カーネルの自動チューニング”, 計算工学講演会論文集, Vol. 20, E-2-1 (2015).
18. Mohamed Amin Jabri, Osamu Tatebe, “Design of concurrent B+Tree index for native KVS on non-volatile-memories”, 研究報告ハイパフォーマンスコンピューティング(HPC), 2015-HPC-150(25), 4 pages, Aug. 2015
19. 田中昌宏, 建部修見, 川島英之, 村田直郁, “すばる HSC データ処理の性能調査と Gfarm/Pwrake の適用”, 研究報告ハイパフォーマンスコンピューティング(HPC), 2015-HPC-150(36), 8 pages, 2015 年 8 月
20. 桐井祐樹, 建部修見, Ross Robert, “CODES/ROSS による分散ファイルシステムのための分散メタデータサーバ PPMDS の評価”, 研究報告ハイパフォーマンスコンピューティング(HPC), 2015-HPC-152(7), 9 pages, 2015 年 12 月
21. Osamu Tatebe, “System Software for Post Petascale Data-Intensive Science”, Sapporo Summer HPC Seminar 2015, Sapporo, July 23, 2015

22. Mohamed Amin Jabri, Osamu Tatebe, “Design of concurrent B+Tree index for native KVS on non-volatile-memories”, 情報処理学会ハイパフォーマンスコンピューティング（HPC）研究発表会, 別府, Aug. 5, 2015
23. 田中昌宏，建部修見，川島英之，村田直郁，“すばる HSC データ処理の性能調査と Gfarm/Pwrake の適用”，情報処理学会ハイパフォーマンスコンピューティング（HPC）研究発表会, 別府, 2015 年 8 月 6 日
24. 建部修見, “Gfarm ファイルシステムの概要と実演”, Gfarm ワークショップ, 神戸, 2015 年 9 月 4 日
25. 桐井祐樹，建部修見，Ross Robert，“CODES/ROSS による分散ファイルシステムのための分散メタデータサーバ PPMDS の評価”，情報処理学会ハイパフォーマンスコンピューティング（HPC）研究発表会，札幌，2015 年 12 月 16 日
26. Fuyumasa Takatsu, Kohei Hiraga and Osamu Tatebe, “Design of object storage using OpenNVM for high-performance distributed file system”, 2nd Annual Meeting on Advanced Computing System and Infrastructure (ACSI2016), Fukuoka, Jan. 18, 2016
27. Shin Sasaki, Kazushi Takahashi, Yoshihiro Oyama and Osamu Tatebe, “RDMA-based Direct Transfer of File Data to Remote Page Cache”, 2nd Annual Meeting on Advanced Computing System and Infrastructure (ACSI2016), Fukuoka, Jan. 20, 2016
28. 川島英之，建部修見，“縮退表現に基づくシーケンスパタン集合の圧縮”，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2016-OS-136, No. 13, pp. 1-6, 2016.
29. Li Jiang, Hideyuki Kawashima, Osamu Tatebe, “Recursive Incremental Computation for Efficient Window Aggregate over Array Database”, 情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2016-OS-136, No. 5, pp. 1-15, 2016.
30. Komei Kamiya, Hideyuki Kawashima, Osamu Tatebe, “Efficient Write Ahead Logging with Parallel Write”, Annual Meeting on Advanced Computing System and Infrastructure, 2016.
31. 堀尾健太郎，川島英之，建部修見，“暗号化データベースシステムにおける総和計算の並列化技法に関する評価”，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2016-OS-136, No. 6, pp. 1-9, 2016.

32. 神谷孝明，川島英之，建部修見，“並列ログ先行書き込みの評価”，情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC）, Vol. 2015-HPC-150, No. 37, pp. 1-6，2015.
33. 田中昌宏，建部修見，川島英之，村田直郁，“すばる HSC データ処理の性能調査と Gfarm/Pwrake の適用”，情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2015-HPC-150，No. 36，pp. 1-8，2015.
34. 堀尾健太郎，川島英之，建部修見，“暗号化データベースシステムにおける総和計算処理の並列化”，情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2015-HPC-150，No. 23，pp. 1-6，2015.
35. 三橋龍也，川島英之，建部修見，“環結合による類似検索の高性能化”，情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2015-HPC-150，No. 16，pp. 1-8，2015.
36. 川島英之，建部修見，“縮退表現に基づくシーケンス演算処理の効率化”，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2015-OS-134，No. 9，pp. 1-7，2015.
37. Li Jiang, Hideyuki Kawashima, Osamu Tatebe, “Design of Automatic Supernova Detection System”，情報処理学会研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2015-HPC-150，No. 19，pp. 1-8，2015.
38. 堀尾健太郎，川島英之，建部修見，“暗号化データベースシステムにおける効率的な集約処理の評価”，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2015-OS-133，No. 19，pp. 1-10，2015.
39. 神谷孝明，川島英之，建部修見，“P-WAL: 並列ログ先行書き込みの提案”，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2015-OS-133，No. 18，pp. 1-10，2015.
40. 齋藤周作，多田野寛人，今倉暁，“Shifted Block BiCGSTAB(l)法の構築とその精度について”，第 44 回数値解析シンポジウム（NAS2015），ふどうの丘，2015 年 6 月.
41. 多田野寛人，齋藤周作，今倉暁，“複数右辺ベクトル・複数シフトをもつ線形方程式に対する Shifted Block Krylov 部分空間法の近似解の精度改善”，【プラズマ壁相互作用における非線形現象の理論モデル構築と画像・動画解析手法開発に関する研究会】＆【プラズマ工学・電磁会解析とその数値解析手法およびビジュアライゼーションに関する研究会】合同研究会　第 1 回非線形・可視化部門研究会，自然科学研究機構　核融合科学研究所，2015 年 9 月.

(4) 著書、解説記事等

7. 異分野間連携・国際連携・国際活動等

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

8. シンポジウム、研究会、スクール等の開催実績

1. Gfarm ワークショップ 2015，神戸，2015 年 9 月 4 日
2. Gfarm シンポジウム 2015，東京，2015 年 12 月 14 日
3. International Workshop on Language, Network and System Software (LENS2015), Tokyo, Oct. 29-30, 2015.

9. 管理・運営

組織運営や支援業務の委員・役員の実績

1. 朴泰祐：計算科学研究センター副センター長
2. 朴泰祐：計算科学研究センター・計算機システム運用委員長
3. 朴泰祐：筑波大学情報環境機構企画室室員
4. 朴泰祐：筑波大学ネットワーク管理委員会委員
5. 朴泰祐：HPCI 連携サービス委員会委員
6. 朴泰祐：理化学研究所客員研究員
7. 朴泰祐：東京大学情報基盤センター・スーパーコンピュータ専門委員会委員
8. 建部修見：HPCI 連携サービス運営・作業部会委員
9. 建部修見：理化学研究所客員研究員
10. 建部修見：HPCI 利用研究課題審査委員会レビューアー
11. 建部修見：学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題審査委員
12. 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
13. 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
14. 建部修見：情報処理学会ハイパフォーマンスコンピューティング研究会運営委員
15. 建部修見：インターネットカンファレンス 2015 プログラム委員

10. 社会貢献・国際貢献

1. Taisuke Boku: Organizing Chair, HPC in Asia Session, ISC2015
2. Taisuke Boku: Program Committee, HiPC 2015
3. Taisuke Boku: Member, 2015 Gordon Bell Prize Selection Committee, ACM

4. Taisuke Boku: Member, DoE INCITE Proposal Review Committee
5. Taisuke Boku: Program Committee, International Workshop ExaComm2015
6. Taisuke Boku: Program Committee, SC15 Invited Talk Session
7. Taisuke Boku: Program Committee, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) 2015
8. 朴泰祐：情報処理学会 ACSI2015 シンポジウム・組織委員長
9. Osamu Tatebe: Program Committee, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) 2015
10. Osamu Tatebe: Program Committee, IEEE International Conference on Cluster Computing (Cluster) 2015
11. Osamu Tatebe: Program Committee, 7th IEEE International Conference on Cloud Computing Technology and Science (CloudCom) 2015
12. Osamu Tatebe: Program Committee, International Supercomputing Conference (ISC) 2015
13. Osamu Tatebe: Track Chair, 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2015)
14. Osamu Tatebe: Program Committee, Architecture, Languages, Compilation and Hardware support for Emerging ManYcore systems (ALCHEMY) Workshop 2015
15. Osamu Tatebe: Program Committee, 4th Workshop on Big Data Management in Clouds (BigDataCloud) 2015
16. Osamu Tatebe: Program Committee, 4th International Workshop on Advances in High-Performance Computational Earth Sciences: Applications & Frameworks (IHPCES) 2015
17. Hideyuki Kawashima: Program Committee, The IEEE International Conference on Multimedia Big Data (BigMM).
18. Hideyuki Kawashima: Program Committee, IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-15)
19. 川島英之：WebDB Forum プログラム委員
20. 川島英之：情報処理学会 データベースシステム研究会 運営委員
21. 川島英之：電子情報通信学会 論文誌査読委員
22. 川島英之：電子情報通信学会 知的環境とセンサネットワーク研究会 運営委員
23. 川島英之：電子情報通信学会英文論文誌小特集号 編集委員
24. 川島英之：Annual Meeting on Advanced Computing System and Infrastructure (ACSI) 2016 プログラム委員

25. International Workshop on Eigenproblems: Algorithms; Software and Applications, in Petascale Computing (EPASA2015) Local Committee
26. 多田野寛人: 日本応用数理学会「行列・固有値問題の解法とその応用」研究部会 主査
27. 多田野寛人: 日本応用数理学会「若手の会」研究部会 運営委員
28. 多田野寛人: 日本応用数理学会 JSIAM Letters 編集委員
29. 多田野寛人: 情報処理学会ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2016 プログラム委員
30. 多田野寛人 : Annual Meeting on Advanced Computing System and Infrastructure (ACSI) 2016 プログラム委員

11. その他

海外長期滞在、フィールドワークなど

なし