jh230058

メニーコア CPU,GPU の最適なリソース割り当てに関する研究

河合 直聡（名古屋大学 情報基盤センター）

本研究では、メニーコア CPU および GPU のリソースを、アプリケーションおよび使用するシステムの特性に併せて適切に割り当てるソフトウェアの開発を目的とする。アプリケーションの並列化では並列単位での負荷の均衡化が並列性能を得るために必須となるが、場合によってはこれが困難となる。また、システムによっては並列化時に全リソースを使用せず、一部だけを使うことで、単位時間あたりの消費電力だけでなく、計算時間も削減できる場合がる。さらに、使わないリソースをノード間通信やファイル IO に割り当てれば、システムの利用効率を最大化できる。本研究では、このようなアプリケーションやシステムの特性を加味した、最適なリソース割り当てを提供する基盤ソフトウェアを開発、アプリケーションのさらなる高性能化、低消費電力化を実現する。

1. 共同研究に関する情報

(1) 共同利用・共同研究を実施している拠点名東

東北大学 サイバーサイエンスセンター
東京大学 情報基盤センター
名古屋大学 情報基盤センター
京都大学 学術情報メディアセンター
九州大学　情報基盤研究開発センター
mdx

(2) 課題分野

大規模計算科学課題分野

(3) 共同研究分野(HPCI 資源利用課題のみ)

超大規模数値計算系応用分野

(4) 参加研究者の役割分担

・河合 直聡（課題代表者、DCB 研究、DCB・UT-Helper 統合）
・塙 敏博（課題副代表者、UT-Helper 研究）
・伊田 明弘（アプリケーション提供、評価）
・星野 哲也（DCB の GPU 対応）
・大島 聡（DCB の GPU 対応）
・三木 洋平（アプリケーション提供、評価）

2. 研究の目的と意義

本研究では、メニーコア CPU および GPU のリソースを、アプリケーションおよび使用するシステムの特性に併せて適切に割り当てるソフトウェア基盤の開発を目的とする。並列化されたアプリケーションの性能阻害要因は多数あり、並列単位での負荷の不均衡や、アプリケーションの特性によって不足するハードウェアリソースなどが挙げられる。現在の大規模クラスタで一般的な MPI+OpenMP による並列化では、プロセスレベルおよびスレッドレベルの両方で負荷の均一化が並列性能を得るために必須となる。しかし、アプリケーションによってはこの要件を満たすのが困難な場合がある。また、メモリバンド幅律速なアプリケーションでは、使用するコア数をある一定以上に増やしても、メモリの性能上限から、性能向上が得られず、却って

使用コア数の増加によって、動作クロックの低下や無駄な電力消費に繋がっている。このような背景から、アプリケーションのプロセス毎の負荷やシステムの特性を加味し、最適なリソース割り当てを提供する基盤ソフトウェアが必要となっている。本研究では動的なコア割付を実現する DCB ライブラリおよび最適なリソース割り当てを提供し、また余剰リソース上に Helper スレッドを生成、バックグランドでファイル IO やノード間通信を実行する UT-Helper を併用した基盤ソフトウェアを開発する。

この基盤ソフトウェアの開発により、既存のアプリケーションの性能を改善するだけでなく、消費電力も同時に削減し、システム利用効率の最大化を図る。

3. 当拠点の公募型共同研究として実施した意義

本研究でのソフトウェア開発にあたっては、様々なアーキテクチャ、システム、構成での検証が必要不可欠である。JHPCN 制度では、各大学センターの計算機が利用可能であり、本研究の遂行に適していた。また、本研究で得られた知見、開発したソフトウェアの公開は、今後のスーパーコンピュータの運用効率化に資するものである。

近年のスーパーコンピュータでは、世界的な電力事情の逼迫などを背景に、消費電力削減が重要視されている。一般的にアプリケーションの高速化は計算時間の短縮によって消費電力削減に寄与するが、それ以外の手法での消費電力削減も試行錯誤されている。本研究で提案する消費電力削減手法は簡易なアイデアに基づいており、また特殊な権限等も必要ないため、直ちに様々なシステムで実用化しやすく、HPCI で提供されている計算資源の全てを効率化する物である。

4. 前年度までに得られた研究成果の概要

新規課題のため、該当なし。

5. 今年度の研究成果の詳細

本年度は DCB の研究として新たなコア割付ポリシーを提案、評価した。DCB では、MPI+OpenMP で並列化されたアプリケーションを対象に、プロセス毎の負荷の不均衡をコアレベルで均一化するように、異なる数のコアを各プロセスに割り当てる手法である。図 1 の General Environment に示すように、プロセス間に負荷の不均衡(緑のバーの長さの差)がある場合、一般的な MPI＋OpenMP の環境では各プロセスに割り付けられるコア数は同じなため、コアレベルでも負荷の不均衡が発生する。そこで、DCB では各プロセスに割付コアを変更することで、コアレベルで負荷を均一化させる。DCB の研究開始時には計算時間短縮または消費電力削減のいずれか

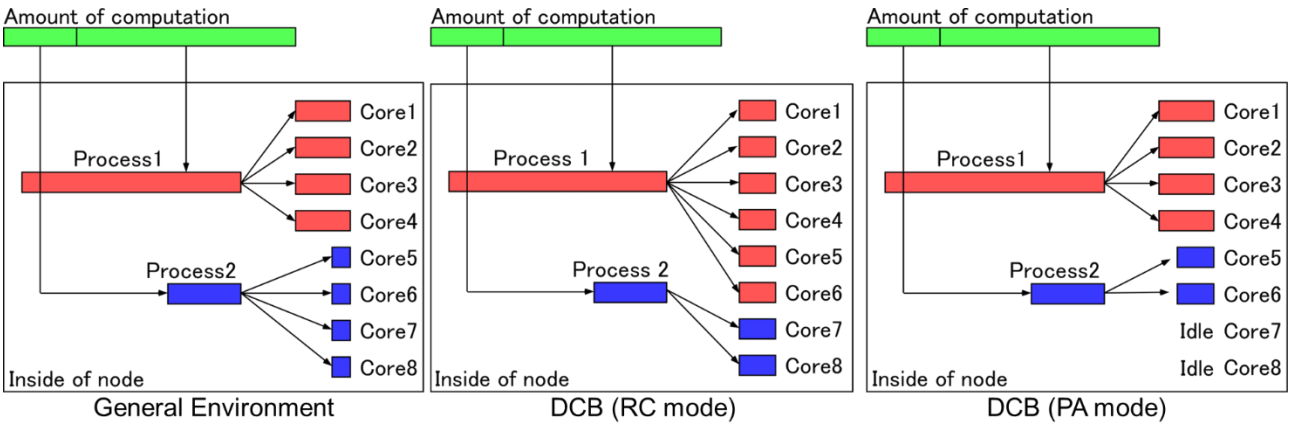


図 1 DCB の動作モード(RC、PA ポリシー)

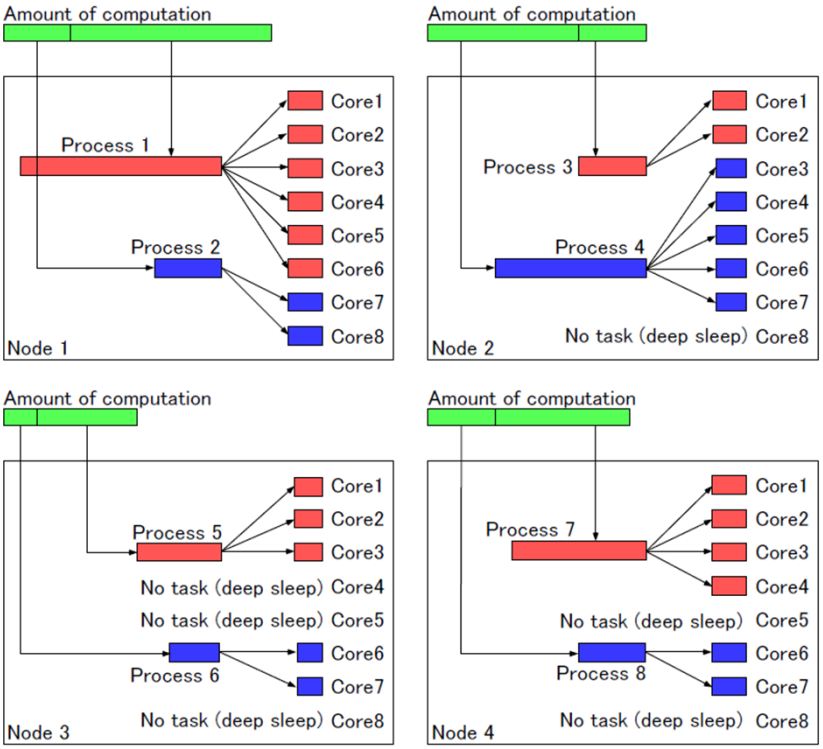


図 2 ハイブリッドポリシー

を目的とした 2 つのポリシーを提案した。計算時間の削減を目的としたポリシー(図 1 の RC ポリシー)では、全てのコアを使用し、負荷の大きいプロセスにはより多くのコアを割り付け、計算時間の削減を図る。消費電力削減のポリシー(図 1 の PA ポリシー)では、演算量が最も多いプロセスに合わせて、他のプロセスに割り当てるコア数を削減、アプリケーションの実行時間を長くせずに使用するコア数を削減して消費電力削減を図っている。本課題申請時に、これらのポリシーの効果は確認できており、期待通りの結果を得ている(研究業績 ((2).1)が、本年度ではさらにハイブリッドなポリシーを提案した。これまでの DCB の問題は、DCB の適用範囲がノード内に限定される点にあり、ノード間の負荷の不均衡は別の方法で削減する必要がある。実際、論文((2).1)では、ノード間の負荷の不均衡を削減するために、ノードへのプロセスの再配置を検討しており、最適なノードへのプロセス配置を導出するために、組み合わせ最適化問題への落とし込みおよび疑似量子アニーラの活用を提案した。結果、Fat Tree などシンプルなネットワークトポロジで構成されたシステムでは DCB を適用した状態から更に計算時間を半分する効果を確認している。ただし、プロセスの再配置には時間がかかり、また様々な条件下での組み合わせ最適化問題の、実用的な近似解の求解の困難性や、疑似量子アニーラから近似解を得るのに時間がかかるなど、実用的な利用にはまだ研究すべき点が多くある。本年度に提案したハイブリッドなポリシーは、より簡易にノード間の負荷の不均衡を取る方法であり、実用的である。ハイブリッドなポリシーでは RC ポリシーを適用した状態で、最も演算量が多いプロセスに合わせて、それ以外のプロセスに割り付けるコア数を削減するように PA ポリシーを適用する。これにより、図 2 に示すように、演算量の多いノードでは全てのコアを使用して計算時間を削減し、演算量の少ないノードでは使用するコア数を削減、単位時間辺りの消費電力削減に寄与する。また、ハイブリッドポリシーでの消費電力削減は使用コア数の削減による単位時間あたりの消費電力削減効果だけでなく、アプリケーションの実行時間短縮による消費電力削減効果もある。図 3 は DCB を使用した場合の効果を示している。評価環境は Wisteria/BDEC-01 Odyssey(WO) 、 Oakbridge-CX(OBCX) および Camphor3(C3)である。評価に使用したアプリケーションは Lattice H-matrix の行列ベクトル積である。Lattice H-matrix はオリジナルの H-Matrix の問題点あった通信時間削減を達成しているが、それと引き換えにプロセス間の負荷の不均衡が大きくなっているため、DCB でこれの削減を試みた。なお、評価にあたっては様々なパラメータ、入力モデルでの評価が必要だったため、Xcrypt を使用し

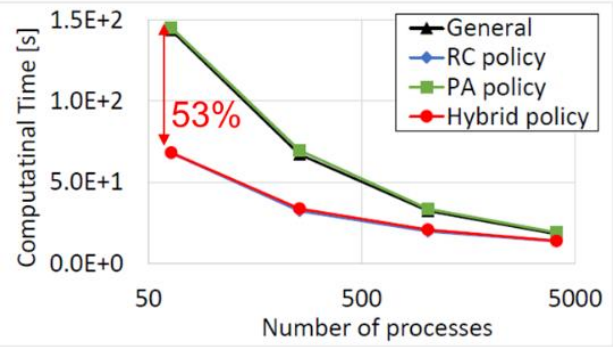
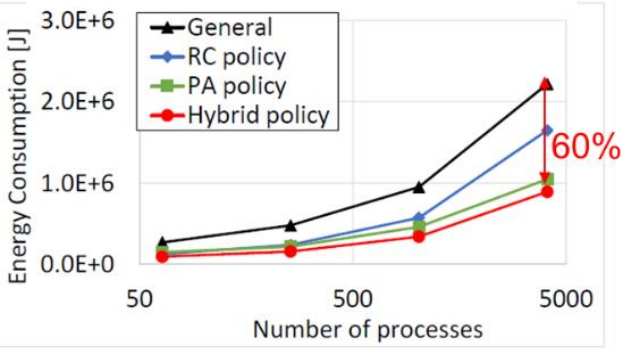


図 3.a 計算時間（Wisteria/BDEC-01 Odyssey）



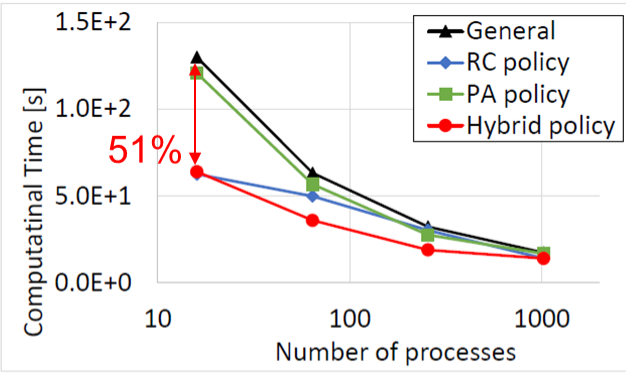


図 3.c 計算時間（Oakbridge-CX）

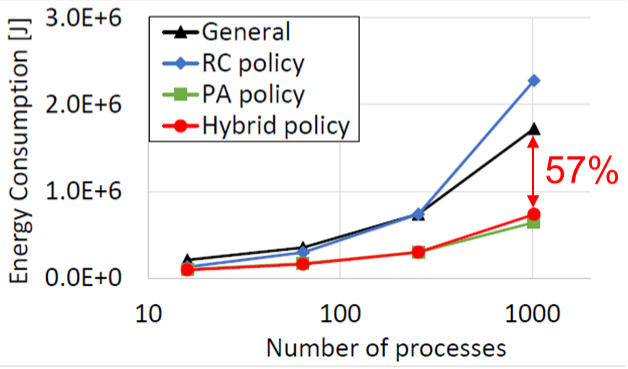


図 3.d 消費電力（Oakbridge-CX）

て全パラメータでのジョブ投入を自動化している。なお、ジョブ投入にあたっては、他ユーザーへの配慮のため、全ジョブを一気に投入するのではなく、Xcrypt の機能を使用して十数ジョブが投入されている状態を維持

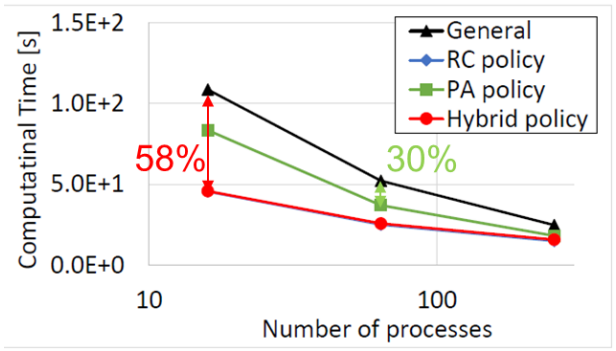


図 3.e 計算時間（Camphor 3）

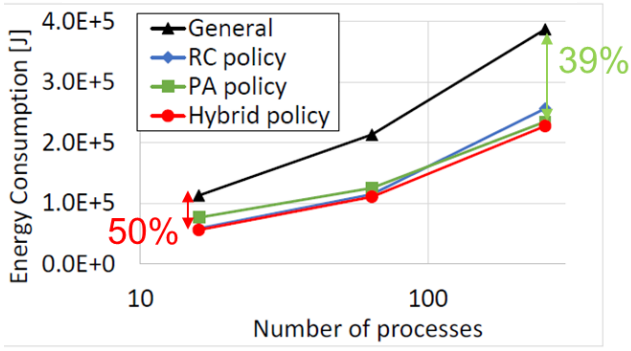


図 3.f 消費電力（Camphor 3）

するようにした。図 3 の a, b はそれぞれ WO での計算時間および消費電力を示しており、計算時間は Hybrid≒RC＜PA≒General の、消費電力は Hybrid≦PA≦RC＜General の傾向を示している（General は DCB を使用していない結果を表す）。これは想定通りの結果であり、計算時間、消費電力削減の効果から Hybrid ポリシーの有効性が確認できる。図 3 の c, d は OBCX での結果を示しており、計算時間は Hybrid≦RC＜PA＜General の、消費電力は PA≦Hybrid＜General＜RC の傾向を示している。また、e, f は C3 での結果を示しており、計算時間は Hybrid ≒ RC＜PA＜General の、消費電力は PA≦Hybrid≦RC＜General の傾向を示している。いずれの環境でも Hybrid ポリシーの有効性は確認できるが、同時に、OBCX および C3、特に C3 での PA ポリシーの計算時間短縮が目立つ。これは、使用するコア数の削減により、供給電力、冷却に余裕が発生し、Turbo Boost が有効に働いたためである。実際に、C3 で PA ポリシーを使用した場合の動作クロックを確認した結果、最大で 3.2GHz での動作を確認しており（基本的な動

作クロックは 1.9GHz)、結果として PA ポリシーの利用により計算時間が短縮できている。使用するコア数を削減してアプリケーションの実行時間を削減できている結果は、UT-Helper の研究でも同様に確認できており、今後の DCB のコア割付の自動チューニング手法の検討ではこの点も考慮していくとともに、使用しないコアへの Helper スレッドの割付を実現していく予定である。

本年度はさらに DCB の GPU 運用のための手法を検討した。CPU への DCB の適用と同様に、各プロセスが利用する GPU 数を変更し、GPU 単位での負荷を均衡化させる。DCB の特性上、割付を変更するリソース数(CPU ではコア数、GPU では GPU 数) が多い方が効果が大きくなるため、ここでは Multi instance GPU(MiG)を使用する。A100 以降の MiG では 1 つの GPU を最大で 7 つの GPU に分割可能であるため、1 ノード辺り 8 つの A100 を搭載したシステム(mdx を想定)では、最大で 56GPU のシステムとして扱うことができる。なお、MiG を使用した場合、一定の性能低下があるため、同 Nvidia の Multi Process Service(MPS)の利用も検討したが、同一 GPU を MPS 経由で複数プロセスから使用した場合の性能低下が大きく、この性能低下の原因を突き止められなかったため、現状は MiG の使用を前提とする。複数 GPU を使う場合、1 プロセス辺り 1GPU が一般的な実装となるが、DCB 適用のためには、簡易に単一プロセスから複数 GPU を使える環境を作る必要がある。そこで、MPI+OpenACC で並列化されたアプリケーションを前提として、各プロセスで使用する GPU 分のスレッドを OpenMP で生成、スレッド事に異なる GPU を acc_set_device_num で指定し、プロセス毎に異なる数の GPU を利用する方法を検討した。実際に、簡易な実装を行った結果、要件を満たす動作を確認できている。ただし、OpenMP で複数 GPU を利用する実装のため、スレッド間で頻繁にデータの共有が必要な実装となっている場合、GPU 間のデータ転送による性能低下が想定される。したがって、スレッド間の並列化も MPI を想定したような極力データ共有が発生しない実装が前提となる。現在は Lattice H-matrix を OpenACC で GPU 化しており、これが完了したのちに、上記を満たすよう OpenMP 並列化も加える予定である。

6. 進捗状況の自己評価と今後の展望

本研究は 3 か年で実施予定であり、昨年度(1 年目)に実施予定としていた内容は以下である。

1. DCB、UT-Helper の研究の遂行
2. DCB および UT-Helper の連携方法の模索
3. GPU 対応の基礎研究
4. ノード間負荷分散の研究

1 の DCB に関する研究では、5 で述べたように新たな知見が得られており、進捗は順調である。UT-Helper の研究に関しては新たな研究成果は見いだせていないが、2 の DCB との連携と合わせて来年度に実施予定である。

2 の DCB と UT-Helper の連携に関しては、以前として具体的な方法を見いだせておらず、遅延している。現在問題となっているのは Helper スレッドを生成する方法である。Helper スレッドは、ユーザーが記述したメインプログラムのバックグランドでの処理を想定しているため、ユーザーが記述した OpenMP で生成されるスレッドとは別に管理される必要がある。しかし、OpenMP の仕様ではスレッドの独立した管理が困難なことはわかっており、現在は pthread の使用を検討している。pthread は POSIX で規格化された API であり、スレッド生成時には tid が発行される。DCB でのコア割付は sched_set_affinity という API を使い、tid に対して使用するコアを指定しているため、pthread で Helper スレッドを生成、DCB で Helper スレッドを割り付けるコアの指定が可能ではないかと考えている。

3 の GPU 対応のための方針は、5 節で述べたように確認できており、来年度以降の GPU 対応の準備は予定通り完了している。

4 のノード間の負荷分散の研究に関しては、疑似量子アニーラの使用による実現を検討している。本研究内容は 3 か年かけて実施予定であり、予定通りの進捗である。5 節で述べたようにノード間の負荷の均衡化のためには、どのプロセスをどのノードに配置するかの組み合わせ最適化問題を解く必要がある。これまでは必要な制約条件にノード間の負荷の均一化という 1 つの最適化項を加えた形で疑似量子アニーラに解かせていたが、ネットワークの特性を考慮していないため、WO のような Tofu ネットワークでは、プロセス間通信の対象がネットワーク的に遠くなり、使用するノード数を増加させた場合に計算時間が逆に長くなる現象を確認している。よって、組み合わせ最適化問題にネットワークトポロジを想定した最適化項を組み込む方法を検討した。現在はネットワークトポロジを考慮した最適化項も含めた組み合わせ最適化問題の定式化は達成しているが、最適化項を複数にした問題を元に疑似量子アニーラから良好な近似解を得られていない。これは解くべき問題が大規模なために、部分問題に分割して疑似量子アニーラに解かせているが、分割数が増えると最適化項を満たしにくくなるためである。また、最適化項間の重み付けなど、ハイパーパラメータのチューニングが困難となるのも一因である。

本研究は 2024 年度にも採択されており、2024 年度に実施予定の研究内容は以下である。

1. DCB、UT-Helper の研究の遂行
2. DCB および UT-Helper の連携
3. GPU 対応の実施
4. ノード間負荷分散の研究

1 の DCB の研究遂行に関しては、現在は Lattice H-matrix の行列ベクトル積に適用し評価しているが、今年度は H-matrix の QR 分解プログラムや、宇宙物理での粒子法によるシミュレーションなど、異なるアプリケーションへの適用を検討する。これらのアプリケーションへの適用で課題となるのは、DCB でプロセスに割り当てるコア数を変更したときのオーバーヘッドである。現在適用している Lattice H-matrix の行列ベクトル積は行列を生成した時点で、行列ベクトル積の演算量が見切れ、1 回のコアの割付変更で済んでいる。しかし、今年度に適用を予定している QR 分解や粒子シミュレーションでは動的なコアの割付変更が必要となる。現在の DCB でのコア割付は、オーバーヘッドが比較的大きくなっており、これを最小化するよう DCB を改良する。また、UT-Helper の研究では Helper スレッドの生成方法が確率された後に、MPI 通信の隠蔽やバックグランドでのファイル I/O などに着手していく予定である。さらに、Helper スレッドの役割として重要なのは、DCB 利用時の最適なコア割付の提供であり、これを優先的に着手する。5 節でも触れたように、DCB での最適なコア割付はアプリケーションだけでなく、システムの特性に強く依存する。現在は各プロセスの演算量から一意に算出しているが、最適な割付は異なると考えられる。来年度以降の研究では、コア割付の自動チューニング機構を Helper スレッドに担わせ、最適なコア割付の自動的な実現を検討していく。現在想定している最適なコア割付手法は、各プロセスに割り付けられたコアの動作クロックを Helper スレッドで取得し、動作クロックが均一になるようにコアの割付を変更する自動チューニング手法などを検討していく。

3 の GPU 対応に関しては 5 節で述べたように、Lattice H-matrix の行列ベクトル積を対象とした実装を完了させ、評価するとともに、期待通りの効果が得られる MPI+OpenMP+OpenACC の実装方法を検討して

いく。Lattice H-matrix での評価をもとに様々な知見を獲得し、3 年目に機械学習など、GPU で広く利用されているアプリケーションへの適用を模索していく。

4 のノード間負荷分散の研究に関しては、今年度の研究を引き続き実施予定である。現在検討している手法は、ネットワークトポロジに基づく最適化項を加えずに組み合わせ最適化問題を疑似量子アニーラに解かせ、取得した複数の近似解から、ネットワークトポロジを考慮した解を選択する手法を検討する。

7. 研究業績

(1) 学術論文 （査読あり）

(2) 国際会議プロシーディングス （査読あり）

1. M. Kawai, A. Ida, T. Hanawa, and K. Nakajima “Dynamic Core Binding for Load Balancing of Applications Parallelized with MPI/OpenMP”. International Conference on Computational Science – ICCS 2023.
2. M. Kawai, A. Ida, T. Hanawa, and T. Hoshino, “Optimize Efficiency of Utilizing Systems by Dynamic Core Binding”, In Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region Workshops(HPCAsia '24 Workshops).

(3) 国際会議発表（査読なし）

(4) 国内会議発表（査読なし）

(5) 公開したライブラリなど

DCB ライブラリの公開
https://bitbucket.org/naosou/dcb/src/master/

(6) その他（特許, プレスリリース, 著書等）