Vol.2009-ARC-184 No.10
2009/8/4

メニーコア向けタスクスケジューリングシステムの検討

三好健文[†1,†2] 笹田耕一[†3] 植原昂[†1]
佐野伸太郎[†4] 森洋介[†1] 吉瀬謙二[†1]

本論文では，並列化タスクとして分割したプログラムを，メニーコアアーキテクチャに割り当てる場合のコア割り当て問題に着目する．まず，予備評価によりコア割り当てがプログラムの実行時間に影響することを示す．さらに，コア割り当てを最適化するために，通信レイテンシと通信データ総量による評価関数により QAP 問題として定式化し，メタヒューリティクスである GRASP 法により解を求める．提案手法をいくつかのプログラムに適用し，メニーコアアーキテクチャのシミュレータで実行サイクル数を計測し，その効果を評価する．

A Study of Task Scheduling System for Many-core Architecture

Takefumi MIYSOHI ,[†1,†2] Koichi SASADA ,[†3]
Koh UEHARA ,[†1] Shintaro SANO ,[†4]
Yousuke MORI [†1] and Kenji KISE [†1]

An optimization method for task assignment onto many core architecture is considered. At first, it is shown that program execution time depends on the manner of task assignment onto many core architecture by a pre-experiment. And next, assignment problem is formulated as QAP problem by using evaluation function of communication latency and data transfer volume. The formulated problem is solved by GRASP which is a kind of meta-heuristics. The proposed method is evaluated by using several experiments.

1. はじめに

大規模マルチコアやメニーコアといった複数のコアを持つプロセッサアーキテクチャは，低消費電力，省スペースで高い計算性能を得ることができるため，近年注目を集めている．特に，IBM/Sony Cell Broadband Engine[1)] や，Cyclops-64(160 コア)[2)] などのコア間のメモリ転送をソフトウェアで明示的に管理するアーキテクチャは，プログラムを効率良く実行できるとして着目されている．これは，キャッシュのコヒーレンシを考慮した大規模なキャッシュコントローラのためのリソースが削減できることや，高い計算性能を引き出すためにプログラマがメモリ使用方法を明示的に指定できることなどによる．例えば文献 3) では，Cyclops に LU 分解を最適に実装することで高い計算性能を引き出している．しかし，その一方で，ユーザがメモリ制御やスケジューリング，コアへの割り当てを考慮しなければならなず，プログラミングを行う上で大きな負担となる．

特に，メニーコアアーキテクチャでは，コア間の通信レイテンシが小さい一方で，その構造によるコア間毎の通信レイテンシの差が大きい．そのため，性能を引き出すためには，プログラムを適切に並列化するだけではなく，並列化したタスクを実行するコアを適切に選択し，通信が必要なタスク間のレイテンシを短くすることが重要であると考えられる．タスクのコア割り当ては，SMP アーキテクチャやキャッシュを持つ共有アドレス空間のプロセッサ，あるいはコア間の通信レイテンシの差異が小さいアーキテクチャの場合には，ほとんど考慮する必要がない．また，通信レイテンシが大きいクラスタ計算機などにおいては，各ノードに割り当てられるプログラムの粒度が大きく，効率良く計算を実行するためには通信レイテンシを計算時間にどう埋め込むかが支配的である．しかし，コア数が 100 個を越えるメニーコアアーキテクチャにおいては，通信レイテンシが比較的小さいことから，その性能を引き出すためには，プログラムを小さな粒度で分割し中・細粒度並列性を有効に活用することが重要である．このとき，通信レイテンシを計算の中に埋め込むだけではなく，

†1 東京工業大学大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†2 独立行政法人 科学技術振興機構
Japan Science and Technology Agency
†3 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
†4 東京工業大学工学部
School of Engineering, Tokyo Institute of Technology

通信が必要なタスク同士をレイテンシの小さなコア同士に割り当てることが必要になる．

本論文では，メニーコアアーキテクチャにおけるタスクのコア割り当ての問題に焦点を当てる．まず第 2 章で，タスクのコア割り当ての違いにより，プログラムの実行時間に差が生じることを予備評価により示す．次に第 3 章で，コア割り当てを最適化するために，データ通信量に着目した手法を提案する．提案手法では，仮に決定したコアへの割り当てによりデータ通信量を測定し，このデータ通信量と通信レイテンシからなる値を評価関数とする QAP 問題として定式化する．メタヒューリスティクスである GRASP を用いてこれを解くことでコアの割り当てを得る．一般に，プログラムを並列化して効率良く実行するためには，タスクの計算資源への割り当てだけではなく，並列化手法や命令スケジューリングを考える必要があるが，今回は簡単のために，プログラムは，すでに並列化タスクに分割されているものとする．第 4 章で，いくつかのプログラムに対し提案手法を適用した結果を示す．

2. タスク割り当て問題

プログラムを並列化したタスクのコアへの割り当てが，どの程度実行時間に影響を与えるか予備評価に示す．ここでは，メニーコアアーキテクチャM-Core を仮定して，タスクの最適なコア割り当てが自明であるプログラムの実行時間を調べる．

2.1 対象アーキテクチャ

本論文では，メモリ管理を明示的に行う必要があるメニーコアアーキテクチャを対象としたタスク割り当て手法を考えるために，M-Core アーキテクチャ(M-Core)[4] を対象として評価を行う．M-Core のモデルを図 **1** に示す． M-Core は，ノードと呼ぶメッシュ状に配置された計算ユニットをもつ．図 1 は，8×8 のノードをもつ例である．

各ノードはコアとルータで構成される (図 **2**)．コアは，演算処理ユニットである PE（プロセッシングエレメント)，ノードメモリ（各ノードが持つ小規模メモリ)，およびDMAC(Direct Memory Access Controller) で構成される． ノード間でデータを共有する場合には，DMA 転送を用いて明示的にデータを通信する．DMA 転送は DMA_PUT と DMA_GET に大別される．前者は自ノードのノードメモリにあるデータを，他ノードのノードメモリ又はメインメモリに転送する．後者は他ノードのノードメモリ又はメインメモリにあるデータを，自ノードのノードメモリに転送する．DMA 転送によるコア間の通信レイテンシは，(1)DMAC を介してルータにアクセスし，(2)X-Y 次元順にルータを辿り (3) ルータから DMAC を介しメモリに書かれる，までのサイクル数となる．

M-Core 上での動作を，ソフトウェアシミュレータである SimMc を用いて評価する．

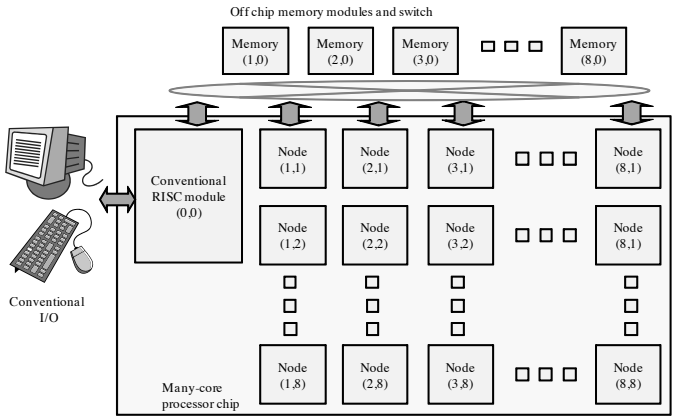

図 **1** M-Core アーキテクチャ
Fig. 1 M-Core Architecture

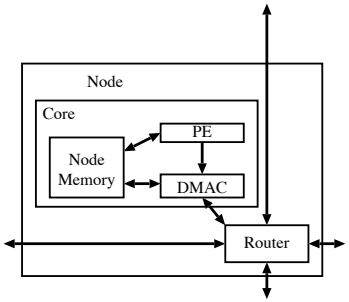

図 **2** M-Core 内のノード
Fig. 2 Node of M-Core

2.2 コア割り当ての性能への影響評価

例として，図 **3** のように，すべてのコアが同時にに近傍の 4 個のコアと相互にデータ通信を行うプログラムを考える．コア数を $4 \times 4 = 16$ 個とし，各々100 回データを送受信する時に，図 3 のように適切にタスクをコアに割り当てた場合と，コア間のレイテンシが長くなるよう人為的に配置した場合の実行サイクル数を測定し比較した．

測定結果を図 **4** に示す． 実線は，コア間の通信距離が長くなるように配置した場合の実

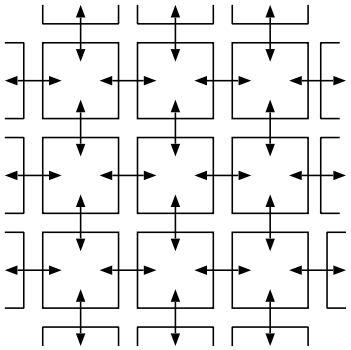

図 3　すべてのコアが同時に近傍の 4 個のコアと通信するプログラム
Fig. 3　A program that all cores communicate with own neighbor cores simultaneously

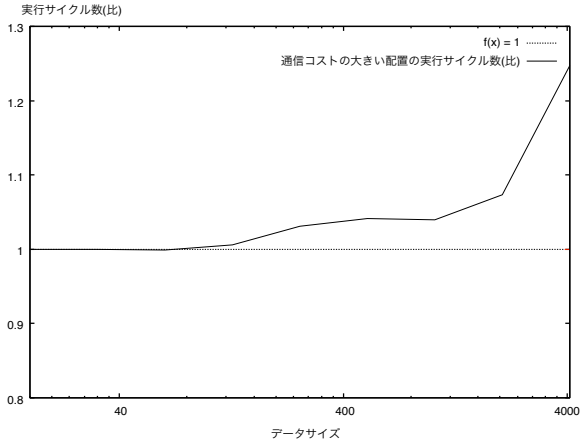

図 4　タスクのコア割り当てによる実行時間の評価
Fig. 4　Evaluation of program execution time with task assignment

行サイクル数と適切にタスクをコアに割り当てた場合のサイクル数の比である．比が 1 より大きいことは，コア間の通信距離が長くなるように配置した場合の方が最適に配置した場合より実行サイクル数が大きいことを意味する．この実験により，データサイズが計算命令数に対し大きくなると，計算で通信レイテンシを隠蔽することができないために，実行時間にタスクのコアへの割り当てが大きく影響することがわかる．

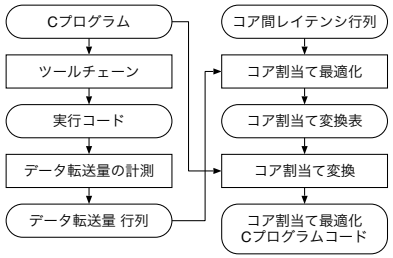

図 5　タスクのコア割り当て手法の概要
Fig. 5　Overview of a method to assign tasks onto cores

3. タスクのコア割り当て最適化手法の提案

タスクのコアへの割り当てを決定するためにデータ転送量とコア間の通信レイテンシによる評価関数を用いる． 図 5 にタスクのコア割り当て手法の概要を示す．提案手法は，C 言語のプログラムを入力として受けとる．ここで与えられるプログラムは，各コアで実行されるタスクに分割されているとする．与えられたコードをターゲットアーキテクチャ向けツールチェーンにより実行コードに変換し，一度実行することおでコア間のデータ通信量を測定する．M-Core を対象とする場合には，SimMc により実行する．プログラムの実行により各コア間に発生したデータ転送量からデータ転送行列 D を生成する．データ転送行列の要素 D_{ij} は，プログラムを実行したときに，コア (i, j) 間に発生したデータ転送の総量である．

アーキテクチャにおける固有値であるコア間通信のレイテンシ行列 (L) と D を用いることで，タスクのコア割り当ての変換表を生成する．この変換表は，元々あるコアに割り当てられていたタスクを表を引いて得られる別のコアに割り当てることを示す．最後に変換表に基づいてソースコードを変更することで，コアへの割当てを最適化した C プログラムコードを得ることができる．

3.1 データ転送量とレイテンシによる評価関数

コア (i, j) 間のデータ転送 $D(i, j)$ におけるデータ通信 $d_{ij}[k] \in D(i, j)$ にかかる通信コストは，コア (i, j) 間のレイテンシを $L(i, j)$，データ $d_{ij}[k]$ が送信されてから受信側で必要になるまでの時間を $t(d_{ij}[k])$ とするとき，$L(i, j) - t(d_{ij}[k])$ で得られる．ただし，$L(i, j) - t(d_{ij}[k])$ が負の場合にはコストは 0 とする．コア (i, j) 間の総データ通信のコストは，

$$\sum_k (L(i,j) - t(d_{ij}[k])) \quad (1)$$

であり，従って，プロセッサ全体での通信コスト C は

$$C = \sum_i \sum_j \sum_k (L(i,j) - t(d_{ij}[k])) \quad (2)$$

で与えられる．ここで，通信路におけるデータの衝突は考慮していない．簡単のため，コア i から送信されたデータはコア j ですぐに必要とされるとすると，$t(d_{ij}[k]) = 0$ となりコア (i,j) 間の総データ通信のコストは，

$$\sum_k (L(i,j) - t(d_{ij}[k])) = L(i,j) \times D_{i,j} \quad (3)$$

となる．

このとき，タスクのコア割り当てによる通信コストを最小にする問題は，この C を最小にするための (i,j) をみつける問題であり，QAP(Quandratic Assignment Problem) 問題となる．QAP 問題は NP 困難であるので，メタヒューリスティクスに基づく様々な発見的な解放が考えられている[5]．ここでは，実装が容易で良質な結果を得やすい GRASP 法[6] を用いてこの問題を解くことを考える．

GRASP に基づく解法アルゴリズムを図 **6** に示す． まず初期配置として，与えられたタスクをコアにランダムに割り当てる (random_allocate)．次に，この初期配置に対する局所最適解を貪欲的に探索する (local_optimization)．得られた局所最適解の評価値が今までの評価値より小さい場合には，解を更新し (update)，またカウンタを 0 にクリアする．評価値が今までの評価値より大きい場合には，解を更新せずカウンタの値をインクリメントする．解の更新が発生しなかった回数 count があらかじめ規定しておいた MAX_ITERATION になった時，十分良い解が得られたとして探索を終了する．

予備評価に用いた，4×4 のメニーコアアーキテクチャにおいて各コアが近傍の 4 つのコアと通信を行うプログラム (図 3) にコア割り当て決定手法を適用した時の振舞いを図 **7** に示す． 赤い点で示したものが各イテレーションにおいて初期配置から貪欲的に導出したタスクのコア割り当ての局所最適解の実行サイクル数であり，黒い実線が得られた局所最適解のうち最小となった評価関数の値である．局所最適解の値は初期配置によって一様にばらついている．実際に得られたタスク割り当てを図 **8** に示す．このプログラムにおける自明な最適解が得られていることが確認できる．

```
min = Integer.MAX_VALUE
count = 0
do{
    s = random_allocate;
    local_optimiztion(s);
    cost = cost_calculation(s);
    if(cost < min) {
        update(s);
        min = cost
        count = 0;
    } else {
        count++;
    }
}while(count < MAX_ITERATION);
```

図 **6** GRASP に基づく解法
Fig. 6 A solver based on GRASP

4. 実験と評価

提案手法の効果をシミュレーションを用いた実験により評価する．対象とするアーキテクチャは，節 2.1 でコア割当ての影響の評価の予備評価に用いた M-Core とする．M-Core のシミュレータ SimMc を用いてプログラムを実行した時の実行サイクル数を比較によって，評価を行う．

4.1 結　　果

次の 4 種類のプログラムを用いて，実行時間の短縮に提案手法によるコア割当て最適化が寄与するかどうか評価する．

4 近傍と通信するプログラム (2D) 2 次元インデックスに割付けられたタスクが，各タスクの近傍 4 つのタスクと相互に 512 バイトのデータ通信を行う図 3．コアとタスクのサイズは共に，4×4，4×8，8×8 の 3 種類．

立方体 6 近傍と通信するプログラム (3D) 3 次元インデックスに割付けられたタスクが，各タスクの近傍 6 つのタスクと相互に 512 バイトのデータ通信を行う．コアの個数は 64 個で，3 次元インデックスのサイズ $3 \times 3 \times 3$，$4 \times 4 \times 4$ の 2 種類，および $3 \times 3 \times 3$ でトーラスにデータ授受を行うプログラム．

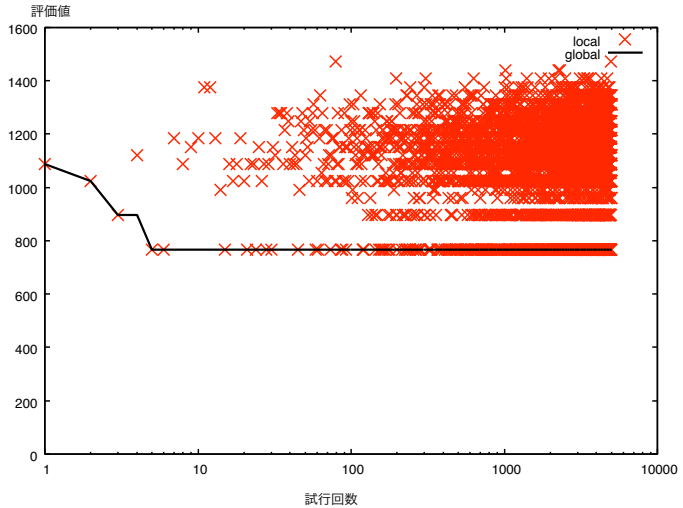

図 7　各イテレーションにおける局所最適解と最適解候補の振舞い
Fig. 7　Results of local and global optimization at each iteration step

一点集中アクセス (BARRIER)　すべてのタスク $\{T\}$ が特定のタスク T_0 に対して一度にデータを 送信し，タスク T_0 はすべてのデータを受信したあと $\{T\}$ に対してデータを送り返す．コアとタスクのサイズは共に，4×4，4×8，8×8 の 2 種類．

行列計算 (MM)　128×128 の行列同士の掛け算．コアの個数は，3×5，6×5，7×9 の 3 種類．

実験の結果を図 **9** に示す．“original” は，人間により自然にコア割り当てを行った場合，“proposed” は，提案手法によりコアの割り当てを変更した場合，そして，“proposed.rev” は提案手法の評価関数を負にして通信レイテンシが増加するようにコア割り当てを行った場合を示す．それぞれ “original” の場合の実行サイクル数を 1 として正規化した値で示している．

2D，3D プログラムではタスクのコアへの割り当てが実行時間に大きく影響することが，“proposed.rev” の実行時間から分かる．提案手法は人間が自然に記述した自明の配置と同様の配置を導出し，実行時間は “original” と “proposed” で等しい．3D では，人間が二次元メッシュ上に適切にタスクを配置することは難しく，そのため提案したタスクのコア割り当て手法が有効である．サイズ $4 \times 4 \times 4$ の 3D のプログラムに対しては，提案手法を適用

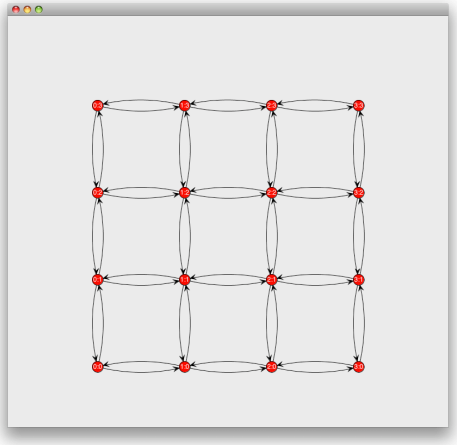

図 8　GRASP によるタスクのコア割り当て結果
Fig. 8　A results of task assignment onto core by GRASP

することで約 7%実行速度が向上した．

一方で，3D でトーラス通信を行うプログラムでは実行時間が長くなり，mm や barrier に対しては，予想した程の効果を得ることができなかった．mm や barrier は，レイテンシをわざと長くした場合の “propsed.rev” で実行時間が大きくなっていないことから，コア割り当て以外の原因が考えられる．以降で考察を行う．

4.2　通信経路の衝突

提案手法によって，実行速度時間が短縮するどころか長くなってしまったケースがある．たとえば $3 \times 3 \times 3$ の 3 次元配列データをトーラスで授受するサンプルプログラムでは，オリジナルのプログラムの実行サイクル数 348085 に対し，割当て変更後では 35401 と，約 1.7%実行サイクル数が増大した．

図 10 はこのプログラムのオリジナルと配置変更後のコア割り当てをダンプしたものである．ここで，配置変更後の方が，全体の通信レイテンシが短く最適化されていることがわかる．一方で，経路を共有するタスク間通信が多数存在しており，この経路の競合が実行時間の増大の原因となっていると考えられる．

4.3　コア-DMA 間のバンド幅による通信速度の律速

また，実行速度が予想ほど向上しなかった原因として，コアと DMAC 間のバンド幅に

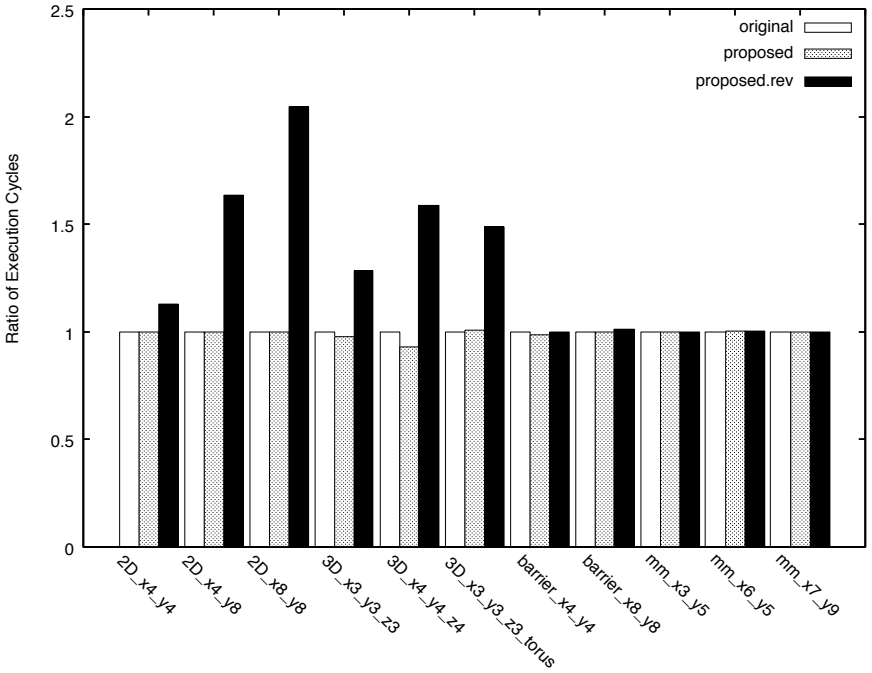

図 9 実行時間の比較
Fig. 9 Comparison of execution cycles

よって通信速度が律速していることが考えられる．barrier および mm では，すべてのコアから同時に一様な通信が発生する．そのため，ノード内の 4 入力ルータに同時にデータが到着し，コア-DMAC 間でデータがつまってしまう．

これを解決するためには，ネットワーク内のバンド幅よりネットワークインターフェイスとコア間のバンド幅が速くなければならない．事実，Cell B.E. では，ネットワークインターフェイスとコア間のバンド幅がネットワークのバンド幅の 2 倍になっている[7]．対象としたメニーコアアーキテクチャの場合，理想的には，コア-DMA 間のバンド幅がノード-ノード間のバンド幅の 4 倍あればよいことになる．

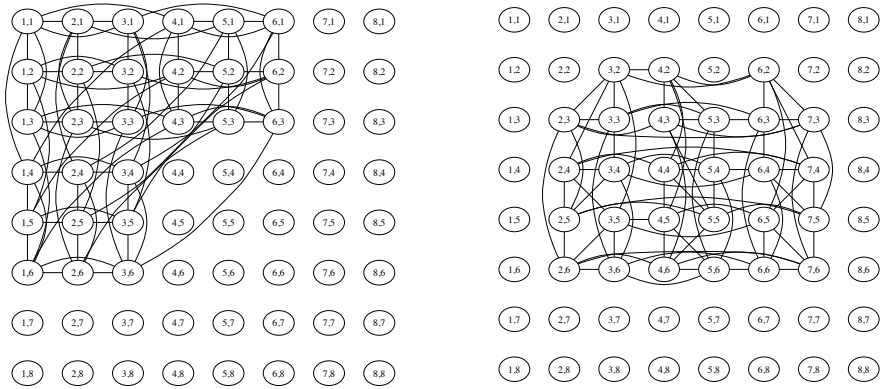

図 10 3 × 3 × 3 でトーラスにデータ授受を行うプログラムの提案手法適用前 (左) と適用後 (右) のコア割り当て結果
Fig. 10 The result of task assignment before applying proposed method (left hand side) and after applying proposed method (right hand side) of the 3 × 3 × 3 torus

5. 関連研究

マルチ/メニーコアアーキテクチャに対して与えられえたタスクをコアにどのように割り当てるかに関する研究は多くない．メッシュ接続型マルチ/メニーコアアーキテクチャ向けのタスクのコア割りあてに関して，連続あるいは不連続の領域に効率良くタスクを割り当てる手法の研究がある．たとえば，文献 8) や 9) は，タスクのコア割リ当てを高速に効率良く実行する手法について述べている．また，メニーコアチップ内の温度に着目したタスのコア割り割て手法[10] がある．

また，マルチ/メニーコアアーキテクチャのためのスケジューリング手法は，数多く研究されている．たとえば，NUMA 向けスケジューリング[11] では，遠隔のプロセッサとデータを共有するためのアクセスレイテンシを考慮したスケジューリング手法を提案している．この手法では，各プロセッサのロカールメモリと共有資源アクセスの通信レイテンシと，共有通信リソースへの競合を考慮している．しかし各コア間のデータ通信レイテンシの差異および，プログラムを分割して得たタスクをどのコアに与えるかについては言及されていない．文献 12) では，マルチプロセッサにおけるスケジューリング問題を QAP 問題として定式化し，GRASP 法を用いて，その解を求めている．ただし，この論文では，対象アーキテク

チャを同一処理能力をもつ完全結合されたプロセッサと仮定して評価関数を定義している．

6. ま と め

本論文では，並列化タスクとして分割したプログラムを，メニーコアアーキテクチャ上で実行する場合のコア割り当て問題に着目し，コア割り当てと実行性能に関する予備評価を行った．予備評価の結果，コア割り当てが実行時間に影響を与えることが分かった．通信レイテンシと通信データ総量による評価関数を定義し，コアへの割り当て問題をこの評価関数による QAP 問題として定式化し，メタヒューリスティクスである GRASP 法により解を求めた．メニーコアアーキテクチャである M-Core を対象として，いくつかのサンプルプログラムに対し提案手法を適用し，その効果を評価した．提案手法により，タスクのコア割り当ての決定が困難なプログラムに対し，有効な割り当てを発見し，実行速度が向上した．しかし，その一方で，提案手法により性能が向上しないケースもあり，それらの原因について考察を行った．

今後の課題として，提案手法では性能向上し得ないケースに対して有効なコア割り当てを求めることが考えられる．提案手法は，必要とする計算時間にも問題がある．貪欲法に基づく局所最適解を求める手法には多くの計算時間が必要となる．オンラインでのコアの最適化を考える時には，より軽量な方法が必要である．また，この論文ではコア割り当てにより実行時間に差が生じることを示し，これを解くために，通信コストに関する様々なパラメタを省略した．今後，必要なパラメタの洗い出しと，それらによるコア割り当てによる実行時間の理論値の導出を検討している．

謝　　辞

本研究の一部は，科学技術振興機構・戦略的創造研究推進事業 (CREST) 「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による．

参 考 文 献

1) IBM Systems and Technology Group. Cell Broadband Engine Programming Handbook Version 1.1, 4 2007.
2) S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A.Singh, T.Jacob, S.Jain, S.Venkataraman, Y.Hoskote, and N.Borkar. An 80-tile 1.28tflops network-on-chip in 65nm cmos. pp. 98–589, Feb. 2007.
3) IoannisE. Venetis and GuangR. Gao. Mapping the lu decomposition on a many-core architecture: challenges and solutions. In *CF '09: Proceedings of the 6th ACM conference on Computing frontiers*, pp. 71–80, New York, NY, USA, 2009. ACM.
4) 植原昂, 佐藤真平, 高前田伸也, 渡邉伸平, 吉瀬謙二. メニーコアプロセッサの HW/SW 研究開発を加速する実用的な基盤環境. pp. 199–207, May 2009.
5) HassanMishmast Nehi and Shahin Gelareh. A Survey of Meta-Heuristic Solution Methods for the Quadratic Assignment Problem. *APPLIED MATHEMATICAL SCIENCES*, No. 45–48, pp. 2293–2312, 2007.
6) ThomasA. FEO and MAURICIOG.C. RESENDE. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, pp. 109–134, 1995.
7) M.Kistler, M.Perrone, and F.Petrini. Cell multiprocessor communication network: Built for speed. *Micro, IEEE*, Vol.26, No.3, pp. 10–23, May-June 2006.
8) ByungS. Yoo and ChitaR. Das. A fast and efficient processor allocation scheme for mesh-connected multicomputers. *IEEE Trans. Comput.*, Vol.51, No.1, pp. 46–60, 2002.
9) S.Bani-Mohammad, M.Ould-Khaoua, and I.Ababneh. An efficient non-contiguous processor allocation strategy for 2d mesh connected multicomputers. *Inf. Sci.*, Vol. 177, No.14, pp. 2867–2883, 2007.
10) Xiongfei Liao, Wu Jigang, and Thambipillai Srikanthan. A temperature-aware virtual submesh allocation scheme for noc-based manycore chips. In *SPAA '08: Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pp. 182–184, New York, NY, USA, 2008. ACM.
11) Guan-Joe Lai and Cheng Chen. A new scheduling strategy for numa multiprocessor systems. In *ICPADS '96: Proceedings of the 1996 International Conference on Parallel and Distributed Systems*, p. 222, Washington, DC, USA, 1996. IEEE Computer Society.
12) 田中貴文, 藤田聡. GRASP 法に基づくマルチプロセッサスケジューリング問題のためのメタヒューリスティック解法の提案と評価. 電子情報通信学会論文誌 D, Vol.83, No.9, pp. 919–926, 2000.