Vol.2012-ARC-202 No.35
Vol.2012-HPC-137 No.35
2012/12/14

通信衝突を削減するタスク配置最適化における通信タイミングの予測方式の影響

森江 善之[1,2,a)]　南里 豪志[1,2,b)]

概要：本稿では，今後より重要となってくると考えられる多次元メッシュ/トーラス向けの通信衝突を考慮したタスク配置最適化技術の提案を行う．また，提案技術で重要となる通信タイミングの予測方式の違いが各タスク配置最適化に与える影響について評価を行った．本評価では，同時に発行される通信の集合を用いた通信タイミングの予測方式と通信が一斉に発行されたと仮定する通信タイミングの予測方式とその中間案である同時に発行される通信の集合を拡張した通信タイミングの予測方式をそれぞれを用いたタスク配置最適化の性能評価実験を行った．CG 法をベンチマークプログラムとする実験においては，同時に発行される通信の集合を用いた通信タイミングの予測方式が最も高速となった．一方で，Recursive doubling をベンチマークプログラムとする実験においては，通信が一斉に発行されると仮定する予測方式が最も通信性能が向上した．その中間案はどちらのベンチマークプログラムでも性能が出なかった．また，これらの原因について通信パターンと通信タイミングの予測方式の関係から考察を行った．

1. はじめに

近年，先端科学技術計算などの要求から計算機にはより高い性能が求められている．そこで，計算機センターなどでは，多数の計算ノードを持つ大規模分散メモリ型並列計算機の導入が進められている．一方で，計算ノード数が大きくなってくると通信のオーバヘッドがより問題となってくる．しかし，現在，ハードウェアコストの問題などからネットワークをクロスバ網で構成することは困難となっている．そこで，比較的低いハードウェアコストとなるメッシュ/トーラスやファットツリーなどのネットワークトポロジを採用することが一般的となっている．しかし，これらのネットワークトポロジでは，計算ノード間のリンクを共有しているため，通信衝突が発生する．この通信衝突は通信性能を十分に悪化させる．

このため，メッシュ/トーラスやファットツリーのようなネットワークトポロジをもつ並列計算機においては，通信衝突を削減することが重要となってくる．筆者ら [3][4][5] は，ネットワークトポロジに起因する通信衝突がタスク配置に依存していることに注目して通信衝突を削減するタスク配置最適化の研究を行っている．この研究では，既存のタスク配置最適化と異なり，ある時間帯で実行される通信が通過するリンクを調べて，各時間帯における通信衝突を検出するというより詳細なタスク配置最適化技術の提案を行っている．

そこで，本稿では，今後より重要となってくると考えられる多次元メッシュ/トーラス向けの通信衝突を考慮したタスク配置最適化技術の提案を行う．また，提案技術で重要となる通信タイミングの予測方式の違いにより，タスク配置最適化の分類を行い，各通信タイミングの予測方式が通信性能に与える影響について性能評価実験を実施することにより評価を行う．また，この実験結果をもとに通信パターンと通信タイミングの予測方式の関係について考察を行う．

本稿の第 2 章では，通信性能の向上のためのタスク配置最適化技術の関連研究の紹介を行う．次に，第 3 章において詳細な通信衝突の予測が重要であることを示す例を紹介する．次に，第 4 章では，通信衝突を考慮したタスク配置最適化技術の提案を行い，第 5 章では，タスク配置最適化に用いられる通信タイミングの予測方式について述べる．第 6 章では，通信タイミングの予測方式の異なる各タスク配置最適化による性能評価実験について述べる．第 7 章にて，性能評価実験に関する考察を行う．最後に第 8 章でまとめと今後の課題について述べる．

[1] 九州大学情報基盤研究開発センター，福岡市東区箱崎 6-10-1
[2] 独立行政法人科学技術振興機構，CREST，東京都千代田区三番町 5
[a)] morie.yoshiyuki.404@m.kyushu-u.ac.jp
[b)] nanri@cc.kyushu-u.ac.jp

Vol.2012-ARC-202 No.35
Vol.2012-HPC-137 No.35
2012/12/14

2. 関連論文

本章では，通信性能の向上を目的としたタスク配置最適化の関連研究について述べる．

T. Agarwal ら [6] や G. Bhanot ら [7] は 3D メッシュや 3D トーラスを対象としたタスク配置最適化技術を提案している．通信衝突を削減するため，各通信のメッセージサイズとホップ数の積の総和が小さくなるようにタスクを計算ノードに割り付ける．このようなタスク配置最適化を Task Allocation by using HopByte (TAHB) と呼ぶこととする．TAHB の目的関数は全ての計算ノード間のホップバイトの総和となる．ホップバイトとは計算ノード間の通信量とホップ数の積のことである．このため，この目的関数を最小とするタスク配置では，プログラムを通して通信量の多いタスク同士をホップ数が少なくなるような計算ノード同士に割り付けることになる．このタスク配置最適化では，複数のリンクを経由する通信量が少なくなるので，通信衝突が発生する可能性が低くなり通信性能の向上がなされる．また，今出ら [9][10] は，大規模並列計算環境のためのランク配置最適化ツール Rank Map Automatic Tuning Tool (RMATT) の提案を行っている．RMATT は TAHB と同様の考え方からホップ数とノード間の転送量の全体最適化をアプリケーションの通信パターンを用いて行うものである．対象とするネットワークトポロジは 3D トーラスである．目的関数は，ホップバイトと通信パターンの律速点における通信量を積算した値を用いている．しかし，これらの研究では，通信衝突において重要となる通信タイミングについて考慮がなされていない．

著者ら [4][3] は，同時に同一リンクを同一方向へ通過するメッセージの個数を調べて，タスク配置最適化を行う研究を行っている．このタスク配置最適化では各リンクでの通信衝突の発生を場所だけでなく時刻も含めて予測することで，通信衝突を削減するタスク配置を決定することができる．性能評価実験ではネットワークがツリートポロジ，ファットツリーの並列計算機を用いて通信性能が向上することを示した．しかし，多次元メッシュ/トーラスのようなネットワークの直径が大きいネットワークトポロジにおける評価は行っていない．

3. 詳細に通信衝突を考慮することによる通信性能に対する効果

ここで，ホップ数とメッセージ数の積を調べるだけでは通信性能を向上することができない例を示す．ここでは，簡単のためにネットワークをツリートポロジとする 6 ノードの並列計算機へタスクを割り付ける場合を用いて説明する．次に示すような通信パターンを実行するプログラムがあるとする．まず，はじめのステップでタスク 1 とタスク 2 の間で通信を行い，次のステップでタスク 4 とタスク 5 の間で通信を行う．さらに，次のステップでタスク 1 とタスク 3，タスク 4 とタスク 6 の間で通信を行い，最後のステップでタスク 1 とタスク 6，タスク 2 とタスク 4，タスク 3 とタスク 5 の間で通信を行う．通信は各ステップになってはめじて実行されるものとする．

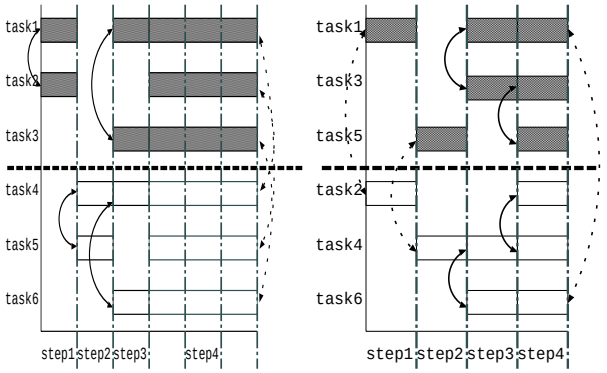

図 1　タスク配置 1 とタスク配置 2 の時の通信の様子．（左）タスク配置 1，（右）タスク配置 2

このような通信パターンを持つプログラムのタスクを対象の計算機に割り付ける．この並列計算機のネットワークは 2 段のツリーとなっており，下位のスイッチ A, B と上位のスイッチ C で構成されている．下位のスイッチ A, B にはそれぞれ 3 つの計算ノードを接続されており，スイッチ A，B をスイッチ C を用いて接続している．このような並列計算機のスイッチ A に接続されている計算ノードにタスク 1，2，3，スイッチ B に接続されている計算ノードにタスク 4，5，6 を割り付けるタスク配置をタスク配置 1 とする．このタスク配置の場合，図 **1** の左の図で示すように第 4 ステップにおいて 3 つの通信が同時にスイッチ C を通過するため，通信衝突が発生し，実行時間が増大する．

次に，通信のタイミングを考慮に入れ，通信衝突が起きないようにスイッチ A にタスク 1，3，5，スイッチ B にタスク 2，4，6 を割り付ける．このタスク配置をタスク配置 2 とする．このタスク配置では図 1 の右の図で示すように各ステップでスイッチ C を通過する通信が分散し，通信衝突を発生させない．このため，タスク配置 1 に比べ，タスク配置 2 ではより高速に通信を行うことができる．

タスク配置 1 とタスク配置 2 はどちらもスイッチを通過する通信の数が 3 であるので，メッセージサイズが同じであれば，TAHB の目的関数では同一コストであるとみなされる．このことから通信衝突を回避するタスク配置最適化では，より詳細な情報からタスク配置最適化を行う必要がある．

4. 通信衝突を考慮したタスク配置最適化技術

前章からわかるように通信衝突は，同時に同一リンクを同一方向に通過した場合にのみ発生する．このため，特に

Vol.2012-ARC-202 No.35
Vol.2012-HPC-137 No.35
2012/12/14

通信に順序関係がある場合などは，各通信により通信開始時刻が異なるため，通信衝突によるペナルティを見積もることはメッセージサイズやホップ数という情報だけでは困難となる．これらを考慮するためには，通信がどのリンクを通過するかと各通信のデータ転送がどのタイミングで行われるかというより詳細な情報を知る必要がある．通信がどのリンクを通るかは通信の送信元と宛先，ネットワークトポロジとルーティングアルゴリズムから得られる．また，各通信のデータ転送のタイミングは，同時にデータ転送が開始される通信の集合を与えることで得られる．これらの情報を与えて，通信衝突をより正確に調べ，通信時間を予測し，通信時間が削減されるタスク配置の求解を行う．このようなタスク配置最適化技術を Task Allocation with Consideration Concurrent Communication (TAC3) と呼ぶ．

4.1 Concurrent Communication Set

提案技術では，同時に発生する通信間での使用リンクの競合を予測する．そこで，同時に転送を開始する通信の集合 Concurrent Communication Set(CCS) を定義する．同時に通信のデータ転送が開始されるための必要条件は次の 2 つとなる．一つは，受信元・宛先がどちらも異なる通信であること，もう一つは，ある通信が完了して初めて実行される通信という順序関係ではないことである．CCS はこの 2 つの条件を満たす通信の集合であるとする．

この CCS は，カーネルコードを 1 度実行した通信ログから抽出するものとする．また，CCS は同じプログラムに同じ入力を与えられた場合は同じ結果となるようにする．CCS を生成する方法はいくつか考えられるが，本稿では，図 **2** に示す手順で CCS を生成する．この CCS の生成アルゴリズムを利用することで，プログラマの負担を減らすことが可能となる．

4.2 目的関数

本提案では文献 [1][2][6] と同様に，タスク間通信とネットワークトポロジをそれぞれグラフ $G_t = \{V_t, E_t\}$, $G_n = \{V_n, E_n\}$ で表し，グラフ間の節点同士の一対一写像 P の最適化としてタスク配置最適化問題を定義する．ただし，$G_t = \{V_t, E_t\}$ は，各通信がどのリンクで要求されるかと各通信のデータ転送がどの時間帯で実行されるかを考慮するため，E_t の定義に変更を加える．グラフは有向グラフとし，辺 $e_{ab} = (v_a, v_b) \in E_t$ は，計算ノード a から計算ノード b への独立した個々の通信を表す．また，その通信の ID である cid_{ab}，通信のデータ量である $cdata_{ab}$，CCS の ID である ccs_{ab} という値を持っているとする．

TAC3 の目的関数では，対象並列計算機上で実行される対象プログラムの通信時間の見積もりを行う．まず，各 CCS における各タスクの予想通信時間を求め，その最大

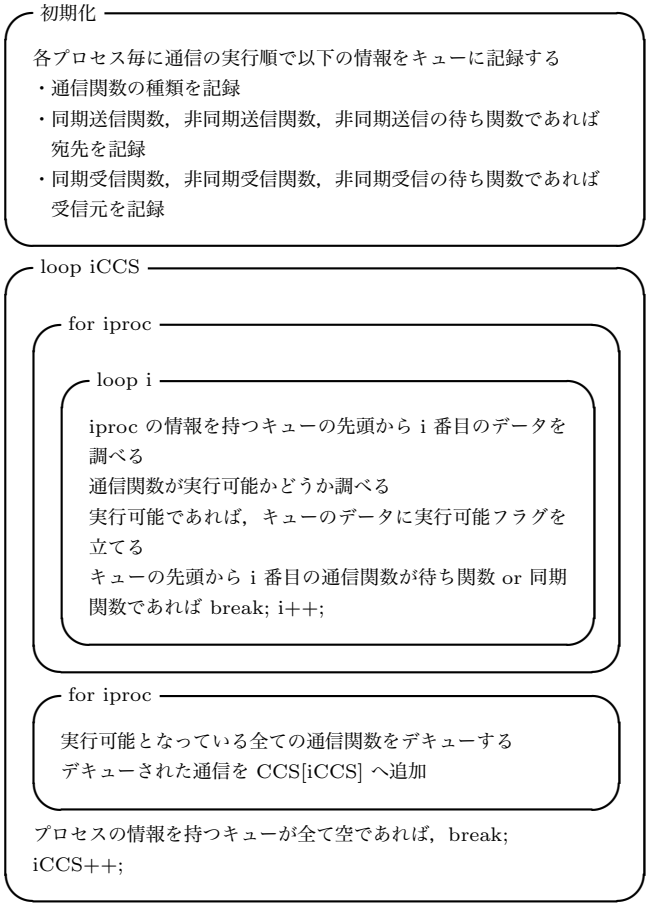

図 **2** CCS の生成アルゴリズム

値を得る．ここで得た各 CCS の予想通信時間の最大値を積算する．これにより，各 CCS における通信衝突による通信時間の増加を加味する．目的関数の計算式は式 (1) となる．

$$f(V_t, V_n, P) = \sum_{t=0}^{N-1} \max_{i=0\cdots n-1} M(t, i, tork(i)) \times coll(t, \pi(i), \pi(tork(i)))/B \quad (1)$$

t は CCS の ID，N は CCS の総数である．i はタスクの ID，n はタスクの総数である．$\pi(i)$ はタスク i が割り付けられている計算ノードの ID を返す関数である．$tork(i)$ はタスク i の宛先のタスクの ID を返す関数である．B は各リンクの通信帯域幅である．$M(t, i, j)$ は t 番目の CCS におけるタスク i からタスク j への通信のデータ量を返す関数である．また，$coll(t, p, q)$ は t 番目の CCS における計算ノード p，q 間の経路の各リンクで要求される通信数の最大値を返す関数である．この目的関数の値を最小とするタスク配置を求め，適用することで通信性能向上を図る．

本提案では，著者らの既存研究におけるツリー構造を対象としたタスク配置最適化と違い，ルーティングを考慮して目的関数の値を計算している．これは本提案で対象としている多次元メッシュ/トーラスにおいて計算ノード間の経路がマルチパスになっているからである．このため，タ

スク配置最適化のマッピング対象となる並列計算機のルーティングを比較的詳細に把握する必要がある．例えば，多次元メッシュ/トーラスでは，一般に次元オーダルーティング (DOR) が用いられる．この場合，アルゴリズムだけでなく，各軸をどの順序で選択するかなどの情報が必要となる．これは，同じ DOR でも軸を選択する順序が異なれば，通過するリンクが異なり，各リンクで要求される通信を決定することができなくなるからだ．また，他にもルーティングが静的である必要がある．本技術では，通信がどのリンクを経由するかを事前に確定する必要があるからである．

5. 通信衝突を考慮したタスク配置最適化における通信タイミングの予測方式

本稿では，通信タイミングの予測方式の比較を行うため，分類を行う．

5.1 CCS に従い通信タイミングを仮定する予測方式

TAC3 では，通信タイミングは，CCS ごとに従っていると仮定して，通信衝突の予測を行う．このとき，通信衝突による通信開始時刻のずれを考慮しない．このため，異なる CCS の所属する通信間では通信衝突が発生しないという仮定を置いている．しかし，実際には通信衝突が発生すれば，それ以降の通信の開始時刻にはずれが生じる可能性がある．これにより想定していない．異なる CCS に所属する通信間で通信衝突が発生する可能性がある．

この通信の予測方式を用いる場合は，プログラム中に同期を挿入し，各通信が CCS に従って発行されるようにすることが出来る．これにより，異なる CCS に所属する通信間で通信衝突が発生する可能性がなくなる．通信のずれを排すことができるため，CCS に従い，事前に正確な通信タイミングの予測が出来る．しかし，一方で，同期のコスト，同期待ちコストが発生する．この予測方式を用いたタスク配置最適化は，その通信性能の向上とこれらのコストとのトレードオフ関係を考慮する必要がある．

この予測方式を用い，CCS ごとに同期を行うタスク配置最適化を TAC3sync とする．これに対して，CCS ごとに同期を行わないタスク配置最適化は TAC3nosync と呼ぶこととする．

5.2 通信が一斉に開始すると仮定する予測方式

次に TAHB や RMATT について考える．TAHB, RMATT の通信タイミングの予測方式は，全ての通信が同時に発行されると仮定していると考えることが出来る．この場合は，全ての通信間にて通信衝突が発生することを考慮する必要があるため，実際には，多くの時間帯で通信衝突が発生しないタスク配置を排除してしまう可能性がある．一方，すべての通信間で通信衝突が発生すると仮定しているので，TAC3 で検討の必要性がある同期に関するコストは発生しない．

5.3 連続する複数の CCS に所属する通信が同時に発行されうると仮定する予測方式

前節までの２つ予測方式の中間案として，通信の実行開始時刻に幅があると考えて通信タイミングを予測する方式が考えられる．この予測方式では，図 **3** のように，あるCCS における通信の発行されるタイミングにおいて，その前後 n 個の CCS に所属する通信も発行されうると仮定する．これにより，通信衝突が発生しやすいと考えられる直近の異なる CCS に所属する通信との通信衝突の見積もりも可能となる．このような通信タイミングの予測方式に

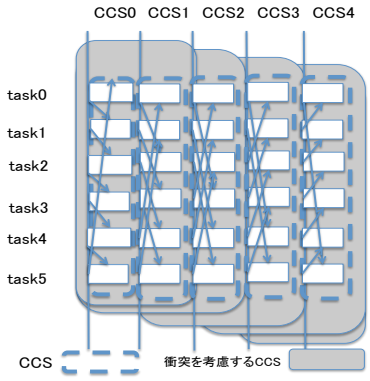

図 **3** 通信衝突を考慮する CCS の範囲

より通信衝突の見積もりを行うタスク配置最適化を TAC3 Wide Range (TAC3WR) と呼ぶ．

6. 性能評価実験

通信タイミングの予測方式の違いによる通信性能の比較を行う性能評価実験を実施する．

6.1 実験概要

FX10 において各通信タイミングの予測方式を用いたタスク配置最適化による CG 法及び Recursive doubling の通信パターンを実行するベンチマークプログラムを実行し，その実行時間を計測する．このとき，TAC3nosync, TAC3sync, TAC3WR は CCS の集合とメッセージサイズを入力とするタスク配置最適化を行い，タスク配置を出力させる．また，TAHB 及び RMATT では，通信の宛先，送信元，メッセージサイズを入力とするタスク配置最適化を行い，タスク配置を出力させる．

6.2 実験環境

実験環境として用いる FX10 の仕様を示す．CPU は Fujitsu SPARC64TM IXfx 1.848GHz (16 コア)，メモリは 32GB，計算ノード数は 768 ノード，OS は Linux ベース

独自 OS である．また，コンパイラ，MPI ライブラリは Fujitsu Technical Computing Suite v1.0 である．ネットワークは，Tofu interconnect[8] を用いており，このネットワークの各リンクの理論帯域は 5GB/s となる．この Tofu interconnect のネットワークトポロジは 6 次元メッシュ/トーラスで各軸のサイズは 4x2x8x2x3x2 となっており，ルーティングは拡張次元オーダルーティングである．今回の実験では故障計算ノードを用いないようにしたため，次元オーダルーティングと同様のルーティングを行う．本実験環境でのルーティングの決定順序は X 軸，Y 軸，Z 軸，a 軸，c 軸，b 軸となっている [11].

また，本実験では，ノード形状の違いによる性能を比較するため，各ノード数において複数の形状による実験を行った．今回使用した九州大学の FX10 では，一般ユーザは最大で 96 個の計算ノードを使用できるため，2x2x2x2x3x2 (形状 1), 4x2x1x2x3x2 (形状 2)，1x2x2x2x3x2 (形状 3), 4x1x1x2x3x2(形状 4) の 4 種類の形状で実験を行った．

また，FX10 は，複数の研究者によって共有されているため，計算機資源を有効に使用できるよう空いている計算ノード群に自動的にジョブがスケジュールされる．この時，論理 3 次元では，形状を指定することは可能であるが，6 次元で形状を指定することはできないため，計算ノードを割り当てられるまで 6 次元の形状が分からない．そこで，実行開始時に実タスクに対して仮想タスクを与えることで，各タスクの仕事内容を交換する仮想的なタスクの再配置を行うこととした．

6.3 ベンチマークプログラム

ベンチマークプログラムとして NAS Parallel Benchmarks[12] の CG 法（Conjugate Gradient method）と集団通信アルゴリズムである Recursive doubling の通信パターンを用いた．今回は通信性能のみを比べるため，CG 法はそのカーネルコードの通信関数に関連する部分を抽出した．この抽出したカーネルコードを 1000 回ループするプログラムを作成した．この時，メッセージサイズは自由に設定できるようにしている．同様に Recursive doubling の通信パターンを 1000 回実行するプログラムもベンチマークプログラムとして作成した．本実験では，通信衝突の影響を調査するのが目的であるため，通信衝突の影響が明確となるメッセージサイズとして 1MB を採用した．

6.4 比較タスク配置

本実験では，TAHB，RMATT，TAC3, TAC3WR により出力されたタスク配置による通信性能比較を行う．

まず，通信性能の基準とするため，デフォルトタスク配置を用意する．デフォルトタスク配置は FX10 で標準で与えられるタスク配置である．ここでは，6 次元メッシュ/トーラスを論理 3 次元トーラスに見せるためのタスク配置が行われている．

次に，TAHB，RMATT は，目的関数以外は TAC3 と同一の探索アルゴリズム，同一の条件下でタスク配置最適化を実施し，出力されたタスク配置である．このとき，用いた目的関数は，それぞれ以下のようになっている．まず，TAHB の目的関数は式 (2) で与えられる．

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} C(i,j)\cdot H(\pi(i),\pi(j)) \tag{2}$$

i, j はタスクの ID，n はタスクの総数である．$\pi(i)$ はタスク i が割り付けられている計算ノードの ID を返す関数である．$C(i,j)$ はタスク i からタスク j への通信量を返す関数である．また，$H(p,q)$ は計算ノード p から計算ノード q へのホップ数を返す関数である．次に，RMATT の目的関数は式 (3) で与えられる．

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} (hop(\pi(i),\pi(j))\cdot size(i,j)) \times max_{link} \tag{3}$$

i, j はタスクの ID，n はタスクの総数である．$\pi(i)$ はタスク i が割り付けられている計算ノードの ID を返す関数である．$size(i,j)$ はタスク i からタスク j への通信量を返す関数である．また，$hop(p,q)$ は計算ノード p から計算ノード q へのホップ数を返す関数である．$link$ はあるノード間に流れた転送量を示し，max_{link} は全ノード間における link の最大値である．

また，TAC3sync は，TAC3nosync とタスク配置は同じものを用いるが，TAC3sync ではそのプログラム内で各 CCS の通信間で同期を行っている．

また，TAC3WR は CCS の前後 1 個の CCS も同時に通信が発行されるうるとして通信タイミングを予測し，タスク配置最適化を行う．ある CCS とその CCS の前後 1 個の CCS をまとめて新しい CCS をすることで，TAC3 の目的関数である式 (1) を用いて通信コストを見積もることができる．

この時，TAHB，RMATT，TAC3，TAC3WR は，タスク求解アルゴリズムとしてヒューリスティックであるシミュレーティッドアニーリングを用いているため，それぞれのタスク配置最適化において 10 個ずつのタスク配置を生成し，プログラムを実行することとした．これらのタスク配置の探索を行う際，初期温度 $1.0e+1$，終了温度 $1.0e-8$，同一温度での繰り返し回数 2500，温度スケジュール係数 0.9 とした．このとき，10 個のタスク配置による各プログラムの実行時間の平均の比較を行った．

6.5 実験結果

まず，図 **4** に CG 法の各形状に置ける各タスク配置でのプログラムの実行時間を示す．CG 法では，TAC3nosync が全ての形状において最も通信性能が高かった．TAC3sync

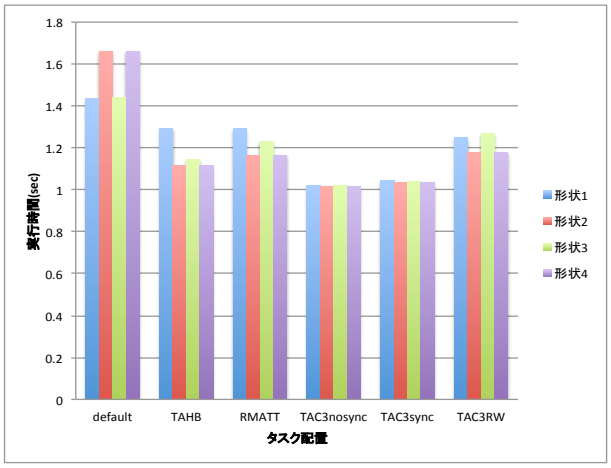

図 4　CG の各タスク配置における実行時間

は，TAC3nosync より 2%程度通信性能が悪化した．これは同期コストが加わったからである．また，TAC3RW に対しては最大 26%の性能向上を示した．これは，TAHB や RMATT と同等の値であった．

次に，図 **5**, 図 **6** にそれぞれ形状 3 と形状 4 における各タスク配置での Recursive douling のプログラムの実行時間と形状 1 と形状 2 における各タスク配置での Recursive douling のプログラムの実行時間を示している．Recusive Doubling では，CG 法と異なり，多くのランク，形状において TAHB が高速となった．形状 3, 形状 4 においては，TAC3sync が TAHB の次に通信性能がよく 2〜3%程度の性能の違いであった．また，TAC3nosync や TAC3WR は形状 3 でかつ 36 ランク以外は性能が悪かった．特に TAC3WR の 48 ランク，形状 4 においては TAHB に対して最大となる 22%の性能悪化を示した．

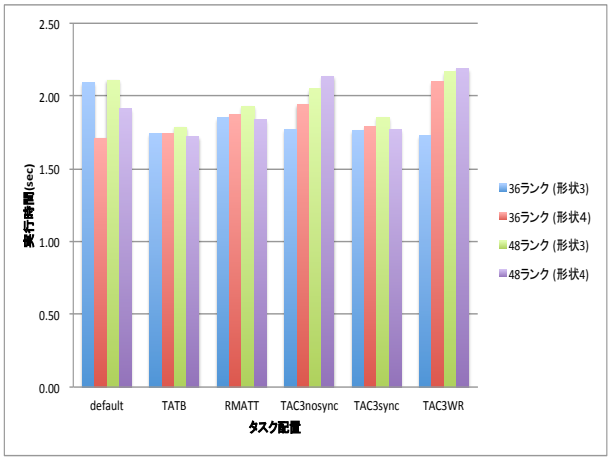

図 5　Recursive doubling の各タスク配置に置ける実行時間 (48 ノード)

形状 1，形状 2 においては，TAHB がランク数，形状によらず高性能となった．また，TAC3nosync，TAC3sync，TAC3WR では 72 ランクで形状 1 の際は TAHB に対して 5〜7% の性能向上を示しているが，84 ランク，96 ランクの際は，形状を問わず，性能が低く，最大で 16%程度悪化している．RMATT では，ランク数や形状における性能の違いは TAHB と同様の傾向となっているが，性能の絶対値は，TAHB より 5〜16% 程度悪化している．

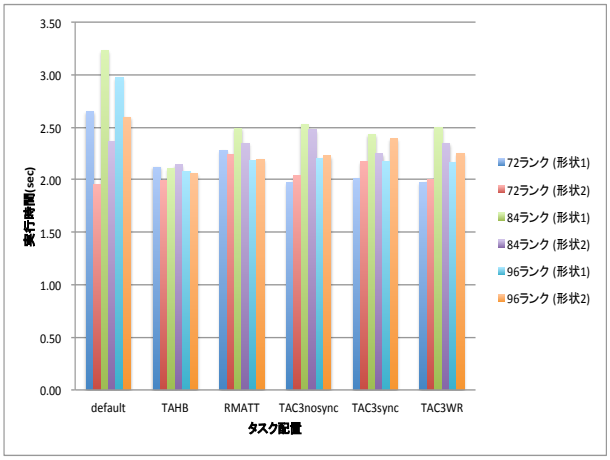

図 6　Recursive doubling の各タスク配置に置ける実行時間 (96 ノード)

7. 考察

CG 法のように各 CCS において各タスクの通信相手が異なる通信パターンの場合は，どの通信タイミングで計算ノード上のタスクと通信を行うかということを評価する CCS に従い通信タイミングを仮定する予測方式を用いた TAC3 が有効であったと考えられる．今回の実験では TAC3nosync がより性能が向上したが，これは，通信衝突が比較的少なかったため，そのずれが後の通信の開始時刻に影響を与えることがほとんどなかったと考えられる．今回は 48, 96 ノードという比較的少ないノード数で実験を行ったため，そのような結果になった考えられる．以降は，よりノード数の多い環境での実験が必要になると考えられる．

一方で，Recursive doubling では，通信が一斉に開始すると仮定する予測方式を用いる TAHB の方が通信性能の高いタスク配置を探索しやすいことが実験からわかった．これは，各通信タイミングごとに通信の傾向が一定である Recursive doubling に対しては，通信タイミングを考慮することなく通信パターン全体でのメッセージ量を考慮する TAHB の目的関数の方が親和性が高いからであると考えられる．一方で，実際に TAC3 では，本稿の実験において探索中に高原に陥り，通信衝突のないタスク配置を求解することはできなかった．72 ランクで TAC3 の性能が悪化しなかった理由は探索の初期タスク配置が通信衝突の少ないタスク配置であったからだと考えられる．

今回の実験において連続する複数の CCS に所属する通信が同時に発行されうると仮定する予測方式を用いる

TAC3WR では，3 つの CCS に所属する通信が同時に通信を発行すると仮定した．CG 法の実験では CCS が 4 つであるため，TAC3WR では，TAHB や RMATT と同様にほぼすべての通信が同時に実行されると予測して通信衝突を考慮した．このため，TAHB や RMATT とほぼ同等の通信性能となった考えられる．一方，Recusive doubling では，TAC3 と同様に，通信衝突が発生すると予測すると仮定した通信の集合ごとに通信衝突を見積もるため，TAC3 と同様に高い通信性能となるタスク配置を探索することが出来なかったと考えられる．

8. おわりに

本稿では，通信衝突を削減するタスク配置最適化の提案を行い，通信タイミングの予測方式がタスク配置最適化による通信性能に与える影響に着いての評価を行った．CG 法をベンチマークプログラムとする実験においては，TAC3nosync や TAC3sync が最も通信性能が良くなった．一方で，TAC3WR は TAHB や RMATT と同等の性能となったこと示した．また，Recursive Doubling をベンチマークプログラムとする実験においては TAHB が TAC3nosync，TAC3sync に対しても通信性能を向上させた．Recursive doubling でも TAC3WR は，TAHB に比べ，低い通信性能となったことを示した．また，これらの原因について考察を行った．

今後の課題としては以下が挙げられる．他のベンチマークプログラムでの実験，より大きいタスク数，ノード数での実験を行うことが考えられる．これは，それぞれの通信予測方式の適用範囲をより詳細に調査するためである．特に TAC3WR では，各通信パターンに対して考慮すべき適切な CCS の範囲をどのように決定するべきかを調査する必要がある．また，タスク配置最適化プログラムの探索アルゴリズムの高速化，並列化などが考えられる．現在は，プロセス，及び，スレッドの並列化を施していないため，探索時間の高速化が必要となる．また，CCS を自動で抽出するプログラムの実装などが挙げられる．これは，プログラマの設計生産性を向上することに寄与する．

謝辞　本研究は主に九州大学情報基盤研究開発センターの研究用計算機システムを利用しました．

参考文献

[1] S. H. Bokhari, “On the mapping problem,” IEEE Trans. Computers, 30(3), pp. 207-214, 1981.
[2] S. Y. Lee and J. K. Aggarwal, “A mapping strategy for parallel processing,” IEEE Trans. Computers, 36(4), pp. 433-442, 1987.
[3] Y. Morie, T. Nanri, and M. Kurokawa, “Task allocation method for avoiding contentions by the information of concurrent communication,” in Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Networks, pp. 62-69, Feb. 2011.
[4] 森江 善之, 末安 直樹, 松本 透, 南里 豪志, 石畑 宏明, 井上 弘士, 村上 和彰, “通信タイミングを考慮した衝突削減のための MPI ランク配置最適化技術,” 情報処理学会論文誌コンピューティングシステム, Vol.48, No.13, pp.192-202, Aug. 2007.
[5] Y. Morie, T. Nanri, “ask Allocation Optimization for Neighboring Communication on Fat-Tree,”in Proceedings of The Fifth International Symposium on Advances of High Performance Computing and Networking (AHPCN-2012), Jun. 2012 (in press).
[6] T. Agarwal, A.Sharma, and L. V. Kale, “Topology-aware task mapping for reducing communication contention on large parallel machines,” in Proceedings of IEEE International Parallel and Distributed Processing Symposium 2006, pp. 1-10, 2006.
[7] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. Sexton, and Robert Walkup, “Optimizing task layout on the Blue Gene/L supercomputer,” IBM J. Res. & Dev. 49, No. 2/3, pp. 489-500, 2005.
[8] Y. Ajima , Y. Takagi , T. Inoue , S. Hiramoto and T. Shimizu, “The tofu interconnect,”in Proceedings of the 19th IEEE Annual Symposium High Performanace Interconnects, pp.87–94, 2011.
[9] 今出 広明，平木 新哉，三浦 健一，住元 真司，“大規模並列計算環境のためのランク配置最適化手法 RMATT,”SACSIS2011, pp.340–347, Mar. 2011.
[10] 今出 広明，平木 新哉，三浦 健一，住元 真司，黒川 原佳，横川 三津夫，渡邊貞，“大規模計算向け通信時間最適化ツール RMATT における実行時間の高速化,”HPCS2012, pp.340–347, jan. 2012.
[11] Y. Ajima, T. Inoue, S. Hiramoto, T. Shimizu, “Tofu: Interconnect for the K computer,”, FUJITSU Sci. Tech. J., Vol. 48, No. 3, pp. 280—285, July 2012.
[12] “NAS Parallel Benchmarks,” 入手先 ⟨http://www.nas.nasa.gov/Resources/Software/npb.html⟩
[13] 長尾智治，“最適化アルゴリズム,” 昭晃堂，2000.