Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

# メッシュ/トーラス接続網に適したタスク配置手法

佐野　伸太郎[†1]　　吉瀬　謙二[†1]

近年，数万ノードを超えるメッシュ/トーラス接続網を採用した計算機が登場している．メッシュ/トーラス接続網のような直接網では，計算タスクが直接網のどこに位置するかで通信時間が異なる．特に通信のやりとりを行うタスクの位置は重要である．ここで，タスクを配置するノードを決定する問題をタスク配置問題と呼ぶ．タスク配置問題は組み合わせの数が多く，プログラマが手動で最適な配置を決定することは難しい．プログラムによって自動的にタスク配置を決定する場合においても，従来の提案手法は計算量が膨大になるという欠点があり，計算量を抑えた高速な決定手法が望まれる．本稿では，タスクのペアを作成し，高速にタスク配置を求めるアルゴリズムを提案する．提案手法は，ノード数 $N$ に対して，$O(N^2 \log N)$ の計算オーダである．OpenNSIM を使用し，NAS Parallel Benchmarks の通信トレースで評価したところ，最大 34%の通信時間の減少を確認した．

# Task Mapping Method for Mesh/Torus Connection Network

Shintaro Sano[†1] and Kenji Kise[†1]

## 1. はじめに

現在の高性能計算機では，複数台のコンピュータノードを結合するシステムが一般的である．例えば，BlueGene/L[1] では，最大 65,536 ノードを $64 \times 32 \times 32$ の 3 次元トーラスネットワークで結合する．今後，Exascale に向けてより多くのノードの結合が必要になると考えられる．

高性能計算機を構成する接続網として，直接網と間接網が存在する．直接網は間接網に比べ，構成の柔軟性とスケーラビリティに優れている．このため多数のノードから成るシステムで良く採用される．直接網の例として，BlueGene/L[1]，K Computer[2], Jaguar[3] などのシステムが挙げられる．これらのシステムのネットワークを見ると，BlueGene/L と Jaguar は 3 次元トーラス，K Computer は 6 次元メッシュ/トーラスを採用している．直接網の中でもメッシュ/トーラスネットワークの注目度が高いことが分かる．

直接網の問題として，ノード間の関係が均一ではないことが挙げられる．例えば，直接網はノードの位置によって他のノードとのホップ数が異なる．また，ネットワークのスループットも他のノードの通信状況に影響を受ける．このため，計算タスクをどのノードに割り当てるかという問題（タスク配置問題）が重要になる．間接網では，1 つのスイッチに多数のノードが接続されため，ノード間の関係は比較的均一である．

タスク配置問題は組み合わせの数が多く複雑な問題である．ノード数 $N$ のすべてのタスク配置を求める問題の計算オーダは $N!$ であり，全探索は現実的ではない．そのため，様々なヒューリスティクスな手法を使用したものがある．例として，Simulated annealing[4], Genetic algorithm[5]，Greedy Randomized Adaptive Search Procedures[6] が挙げられる．しかし，これらのヒューリスティクスな手法を使っても，1 万ノードを超えるシステムでの最適化は難しいと考えられる．例えば，文献 4) は，4096 ノードの 3D トーラスネットワークを対象とした実験で，8 コアの計算機 20 台で 16 時間をかけて最適化を行っている．

本稿では，1 万ノードを超えるシステムであっても，現実的な時間で終了するタスク配置アルゴリズム MOPT を提案する．提案手法 MOPT は，通信量の多い 2 つのタスクの結合を繰り返し行うことによって，最終的な配置を決定する．MOPT は，ノード数 $N$ に対して，$O(N^2 \log N)$ の計算オーダである．

以下，第 2 章では，提案手法 MOPT を説明する．第 3 章では，MOPT をネットワークシミュレータ OpenNSIM を利用し，評価する．そして，第 4 章でまとめと今後の課題について述べる．

## 2. タスク配置手法 MOPT の提案

メッシュ/トーラス接続網を対象としたタスク配置手法を提案する．提案するタスク配置手法の流れを**図 1** に示す．図 1(a) は並列化されたアプリケーションのタスクと通信の様子を表す．丸がタスクであり，矢印が通信方向を示す．数字はタスクの ID を表す．提案手法

†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

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

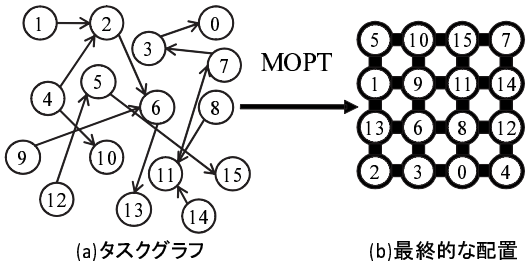


**図 1** 提案するタスク配置手法の流れ

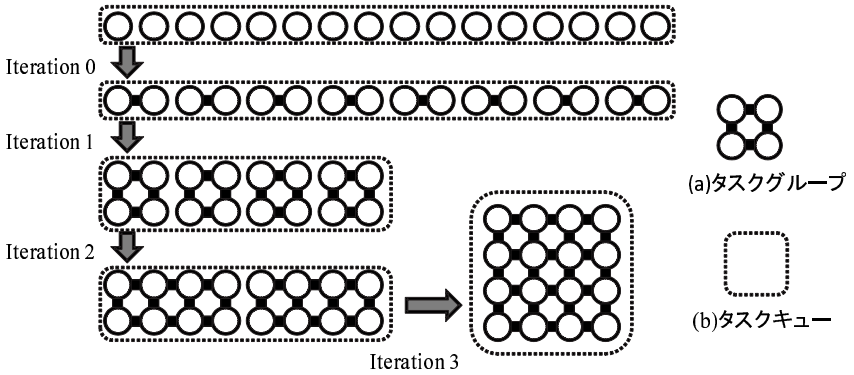


**図 2** MOPT アルゴリズムでの最適化の流れ

では，図 1(a) のようなタスクグラフに対して最適化を行い，対象ネットワークに適用可能な配置を作成する．2 次元メッシュに適用する例を図 1(b) に示す．

### 2.1 2 次元メッシュ/トーラスでの配置最適化アルゴリズム MOPT

提案する **MOPT**(Merge Optimization) アルゴリズムの全体像を**図 2** に示す．このアルゴリズムは繰り返しタスクを結合することによって最終的な配置を作成する．タスクの結合は評価関数が良い結合辺を選択する．評価関数は 2 つ存在する．1 つ目の評価関数は $CommCost$(式 2) であり，この評価関数を用いるアルゴリズムを **MOPT mincost** と呼ぶ．2 つ目の評価関数は $MaxLink$(式 3) であり，この評価関数を用いるアルゴリズムを **MOPT minlink** と呼ぶ．このアルゴリズムを適用するためには，ノード数が $2^n$($n$ は自然数) であることを想定している．

アルゴリズムを説明するための用語の定義を行う．まず，アルゴリズムの繰り返しを Iteration $i$ と表す．ここで，$i = \{0, 1, ... \log N - 1\}$ であり，$N$ は総タスク数とする．次に，図 2(a) の結合したタスクをタスクグループと定義する．単一のタスクもタスクグループである．図 2(b) に示す点線はタスクグループの集合であり，この集合をタスクキューと定義する．タスクキューは各 Iteration ごとに存在する．アルゴリズム開始時にタスクを ID 順にタスクキューに格納する．

各 Iteration では，以下に示すステップ 1 とステップ 2 によってタスクグループが結合する．

**ステップ 1**

現在の Iteration のタスクキューからタスクグループ $G_a$ を取り出す．取り出したタスクグループと最も通信量の多いタスクグループ $G_b$ をタスクキューの中から選択する．タスクグループ $G_p$ と $G_q$ の間の通信量 $CommGroup(G_p, G_q)$ は式 1 で定義する．

$$CommGroup(G_p, G_q) = \sum_{k=1}^{SizeG} \sum_{l=1}^{SizeG} Comm(T_{pk}, T_{ql}) \quad (1)$$

ただし

$SizeG$ : タスクグループのノードサイズ
Iteration $i$ を用いて表すと $SizeG = 2^i$ となる

$T_{pk}$ : タスクグループ $p$ の $k$ 番目のタスク

$Comm(T, T')$ : タスク $T$ とタスク $T'$ の通信量

と定義する．

選択した 2 つのタスクグループ $G_a$ と $G_b$ は現在のタスクキューから取り除かれる．

**ステップ 2**

上で選択した 2 つのタスクグループ $G_a$ と $G_b$ を結合し，新たなタスクグループを作成する．このとき，すべての結合パターンを試し，評価関数 $CommCost$(式 2) または $MaxLink$(式 3) が最小のものを選択する．結合パターンは 2.1.1 節に，評価関数は 2.1.2 節に記述する．新たなタスクグループは次の Iteration のタスクキューに格納する．

タスクキューにタスクグループが残っている場合は，ステップ 1 に戻る．タスクキューにタスクグループが残っていない場合で，最終 Iteration ならば，終了する．タスクキュー

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

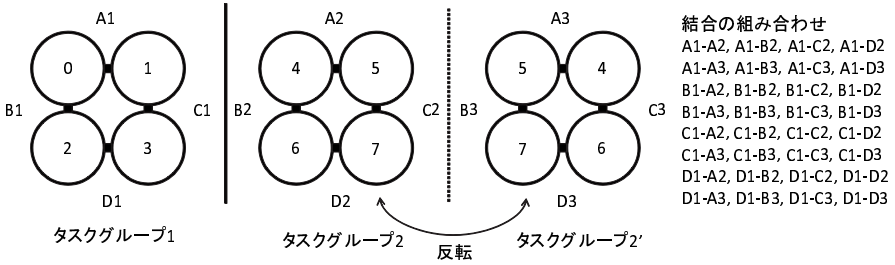


**図 3** 2 × 2 ノードの結合パターン

にタスクグループが残っていない場合で，最終 Iteration でないならば，Iteration $i$ を 1 つ進めて，ステップ 1 に戻る．

#### 2.1.1 結合パターン

2.1 節ステップ 2 における結合パターンについて記述する．タスクグループの結合は，辺の比によって 2 種類の組み合わせが存在する．偶数 Iteration の場合，1:1 の辺の比の結合を行う．奇数 Iteration の場合，1:2 の辺の比の結合を行う．

まず，辺の比が 1:1 であるタスクグループの結合について考える．ここで 2 × 2 ノードの 2 つのタスクグループを結合する例を考える．**図 3** に 2 つのタスクグループ，タスクグループ 1 とタスクグループ 2 を示す．丸はタスクを表し，丸中の数字はそれぞれのタスクの ID を示す．タスクグループ 2' はタスクグループ 2 を反転させたものである．それぞれのタスクグループの辺には，A1 から D3 までの ID を付ける．

この 2 つのタスクグループを結合する辺の組み合わせを考える．タスクグループ 1 と 2 を回転して結合する辺の組み合わせは,「A1-A2, A1-B2, A1-C2, A1-D2 B1-A2, B1-B2, B1-C2, B1-D2 C1-A2, C1-B2, C1-C2, C1-D2 D1-A2, D1-B2, D1-C2, D1-D2」の 16 種類となる．ただし，このままでは反転を含めたすべての組み合わせが含まれない．そこで，タスクグループ 2 を反転させて結合する．このときの組み合わせは,「A1-A3, A1-B3, A1-C3, A1-D3 B1-A3, B1-B3, B1-C3, B1-D3 C1-A3, C1-B3, C1-C3, C1-D3 D1-A3, D1-B3, D1-C3, D1-D3」の 16 種類である．タスクグループ 1 を反転したものと，タスクグループ 2 の結合は，タスクグループ 1 とタスクグループ 2' の結合と同じである．タスクグループ 1 を反転したものと，タスクグループ 2' の結合は，タスクグループ 1 とタスクグループ 2 の結合と同じである．このため，タスクグループ 1 と 2 の結合の組み合わせは 32 種類となる．辺の比が 1:1 であれば，辺の構成は変わらないため，結合の組み合わせは 32 種類である．

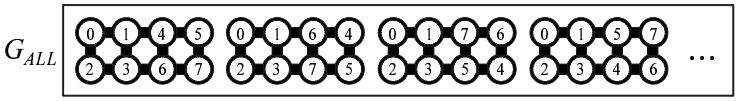


**図 4** 2 × 2 タスクグループ $G_a$ と $G_b$ の結合候補．すべての結合パターンを使用する．

同様の考えで，辺の比が 1:2 のタスクグループを結合する辺の組み合わせは 16 種類となる．

#### 2.1.2 評価関数

2.1 節ステップ 2 における評価関数について記述する．**図 4** に評価対象となる結合候補を示す．このような結合候補のタスクグループを $G_{ALL}$ と表記する．評価関数は 2 つ存在し，それぞれ $CommCost$ と $MaxLink$ である．

1 つ目の評価関数 $CommCost$ について説明する．これは通信コストに関する評価関数である．$CommCost$ を次式に示す．

$$CommCost(G_p) = \sum_{k=1}^{SizeGM} \sum_{l=1}^{SizeGM} Hop(Pos(T_{pk}), Pos(T_{pl})) \times Comm(T_{pk}, T_{pl}) \quad (2)$$

ただし，

$Gp \in G_{ALL}$

$SizeGM$ : 結合候補のタスクグループのノードサイズ

Iteration $i$ を用いて表すと $SizeGM = 2^{i+1}$ となる

$Pos(T)$ : タスク $T$ のノード位置

$Hop(N, N')$ : ノード位置 $N$ とノード位置 $N'$ のホップ数

と定義する．

2 つ目の評価関数 $MaxLink$ について説明する．これはリンク通信量に関する評価関数である．リンクとはノード間の信号線である．例えば 3 × 3 の 9 ノードの 2 次元メッシュを考えた場合，中心のノードは 8 本のリンクとつながっている．$LinkComm(L_x)$ をあるリンクの通信量と定義すると，評価関数 $MaxLink$ は式 3 で定義される．

$$MaxLink(G_p) = max\Big\{LinkComm(L_x) | L_x \in \text{タスクグループのすべてのリンク}\Big\} \quad (3)$$

### 2.2 MOPT の計算量

MOPT の計算量を考える．MOPT アルゴリズムのステップ 1 とステップ 2 の間に依存

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

関係がないため，ステップ 1 とステップ 2 を独立に考える．ノード数は $N$ とし，現在の Iteration を $i$ $(0, 1, ... \log N - 1)$ とする．

#### 2.2.1 ステップ 1 の計算量

ステップ 1 の計算量について考える．ステップ 1 は通信量の総和 $CommGroup$(式 1) を計算する．まず，この $CommGroup$ の計算量を求める．ここで，2 つのタスクの通信量はタスク ID で引くことが出来る 2 次元テーブルに格納されているとし，このコストは定数である．この 2 次元テーブルから通信量を引く計算コストを $c_1$ と置く．このとき，$CommGroup$ の計算コスト $C_{CG}$ は式 4 で表される．

$c_1$ : 通信量をテーブルから参照する計算量

$$C_{CG} = SizeG \times SizeG \times c_1 \tag{4}$$

次に，この $C_{CG}$ がある Iteration で何度呼ばれるかを考える．ここでステップ 1 の開始時のタスクキューのサイズを $SizeTQ$ とする．$SizeTQ = N/2^i$ である．タスクキューの先頭から一つ取り出し，残りの $SizeTQ - 1$ 個のタスクグループから通信量の多いグループを選択する．ステップ 1 が終了すると，タスクキューからはタスクグループが二つ取り除かれる．このため，ある Iteration で呼び出される $C_{CG}$ の回数は次式で表される．

$$\begin{aligned}
&C_{CG} \times (SizeTQ - 1) + C_{CG} \times (SizeTQ - 3) + ... \\
&= \sum_{k=1}^{SizeTQ/2} C_{CG} \times (SizeTQ - 2k + 1) \\
&= \sum_{k=1}^{N/2^{i+1}} \left(2^i \times 2^i \times c_1\right) \left(\frac{N}{2^i} - 2k + 1\right) \\
&= \left(2^i \times 2^i \times c_1\right) \left\{ \sum_{k=1}^{N/2^{i+1}} \left(\frac{N}{2^i} + 1\right) - \sum_{k=1}^{N/2^{i+1}} 2k \right\} \\
&= \left(2^i \times 2^i \times c_1\right) \left\{ \left(\frac{N}{2^i} + 1\right) \frac{N}{2^{i+1}} - \frac{N}{2^{i+1}} \left(\frac{N}{2^{i+1}} + 1\right) \right\} \\
&= \frac{N^2}{4} c_1
\end{aligned}$$

Iteration は $\log N$ 回繰り返されるため，最終的な計算量は次式で表される．

$$\frac{N^2 \log N}{4} c_1 \tag{5}$$

#### 2.2.2 ステップ 2 で CommCost を選択した場合の計算量

ステップ 2 で評価関数 $CommCost$ を選択したときの計算量について考える．結合パターンは 2.1.1 節で説明したように 32 種類または 16 種類である．ここでは 32 種類の結合パターンの計算量を考える．使用される評価関数 $CommCost$(式 2) は，通信量とホップ数のかけ算である．通信量の計算量は前節と同じく定数である．ホップ数の計算も定数であるため，通信量とホップ数の掛け合わせの計算量は定数であり，この計算量を $c_2$ とする．このとき $CommCost$ の計算量 $C_{CC}$ は式 6 で表される．

$c_2$ : 通信量をテーブルから参照し，ホップ数と掛け合わせる計算量

$$C_{CC} = 32 \times SizeGM \times SizeGM \times c_2 \tag{6}$$

ある Iteration でステップ 2 は $SizeTQ/2$ 回繰り返されるため，ある Iteration での計算量は次式で表される．

$$C_{CC} \times \frac{SizeTQ}{2}$$

Iteration は $\log N$ 回繰り返されるため，最終的な計算量は次式で表される．

$$\begin{aligned}
&\sum_{n=0}^{\log N - 1} \left(C_{CC} \times \frac{SizeTQ}{2}\right) \\
&= \sum_{i=0}^{\log(N)-1} \left(32 \times 2^{i+1} \times 2^{i+1} \times c_2 \times \frac{N}{2^{i+1}}\right) \\
&= 32 \times c_2 \times 2N \sum_{i=0}^{\log(N)-1} 2^i \\
&= 32 \times c_2 \times 2N(2^{\log(N)} - 1) \\
&= 32 \times c_2 \times 2N(N - 1)
\end{aligned} \tag{7}$$

#### 2.2.3 ステップ 2 で MaxLink を選択した場合の計算量

ステップ 2 で評価関数 $MaxLink$ を選択したときの計算量について考える．ここでも 32 種類の結合パターンの計算量を考える．リンク通信量の最大値を決定するためには，すべてのリンクの通信量が求まればよい．すべてのリンクの通信量を計算するために，各通信が通過するリンクに，各通信の通信量を足しあわせていく．各通信リンクに通信量を足す計算量を $c_3$ とする．すべての通信は $SizeGM \times SizeGM$ 個ある．それらの通信は平均 $i+1$ ホップであるので，$MaxLink$ は式 8 で表される．

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

$c_3$ : 各通信リンクに通信量を足す計算量

$$C_{ML} = 32 \times SizeGM \times SizeGM \times c_3 \times (i+1) \quad (8)$$

ある Iteration でステップ 2 は $SizeTQ/2$ 回繰り返されるため，ある Iteration での計算量は次式で表される．

$$C_{ML} \times \frac{SizeTQ}{2}$$

Iteration は $\log N$ 回繰り返されるため，最終的な計算量は次式で表される．

$$\begin{aligned}
&\sum_{i=0}^{\log N-1} C_{ML} \frac{SizeTQ}{2} \\
&= \sum_{i=0}^{\log N-1} \left(32 \times 2^{i+1} \times 2^{i+1} \times c_3 \times (i+1) \times \frac{N}{2^{i+1}}\right) \\
&= 32 \times c_3 \times 2N \sum_{i=0}^{\log(N)-1} (i+1)2^i \\
&= 32 \times c_3 \times 2N \left\{ \sum_{i=0}^{\log(N)-1} i2^i + \sum_{i=0}^{\log(N)-1} 2^i \right\} \\
&= 32 \times c_3 \times 2N \left\{2 + (\log N - 2)\, 2^{\log N} + 2^{\log N} - 1\right\} \\
&= 32 \times c_3 \times 2N \left(N \log N - N + 1\right) \quad (9)
\end{aligned}$$

#### 2.2.4 MOPT の計算オーダ

式 5, 7, 9 から，このアルゴリズム全体の計算オーダは $O(N^2 \log N)$ である．

### 2.3 3 次元メッシュ/トーラスでの MOPT

ここでは 2.1 節で説明した 2 次元メッシュ/トーラスでの MOPT アルゴリズムを 3 次元メッシュ/トーラスに拡張する方法について述べる．3 次元に拡張した MOPT アルゴリズムの全体像を**図 5** に示す．図 5 の立方体はタスクを表し，点線がタスクキューを表している．2 次元メッシュ/トーラスとの差は，Iteration $3n-1$ のときに，3 次元方向での結合が起きることなどである．

### 2.4 MOPT による最適化の流れ

MOPT を用いた最適化の流れを**図 6** に示す．MOPT による最適化にはタスクグラフが必要なため，まず，対象プログラムを対象環境で実行し，通信ログを取得する．次に，この通信ログからタスクグラフを作成し，MOPT による最適化を行う．MOPT の最適化は一般的な PC を使用する．最後に，MOPT によって得られた配置を使用し，対象環境で実行する．

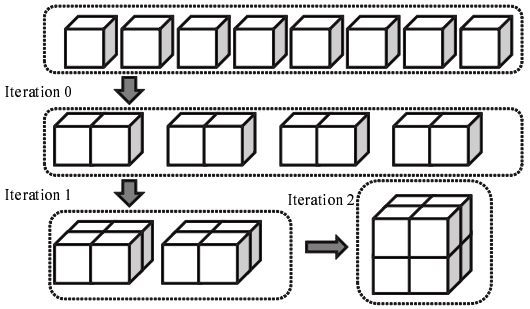


**図 5** 3 次元メッシュ/トーラスでの MOPT

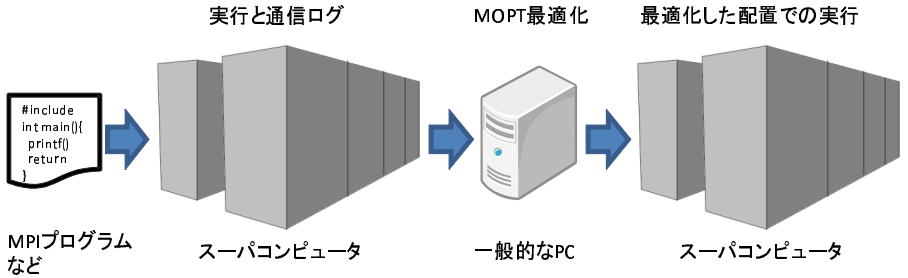


**図 6** MOPT による最適化の流れ

## 3. ネットワークシミュレータを用いた評価

MOPT アルゴリズムをネットワークシミュレータを用いて評価する．評価にあたっては，最適化に要した時間と，実際に得られた配置を使用したときの通信処理時間を測定する．通信時間の測定は OpenNSIM[7] を使用する．評価に用いるプログラムとして，bruck のアルゴリズムによる Allgather[8] と NAS Paralllel Benchmarks[9] から CG と SP ベンチマークの通信トレースを使用する．

### 3.1 評 価 環 境

通信時間の測定はインターコネクトシミュレータ OpenNSIM[7] を使用する．OpenNSIM

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

**表 1** 3 × 3 × 3 ノードに XYZ 次元順割り当てを適用した結果

| rank | z | y | x |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 0 | 2 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 0 | 1 | 2 |
| 6 | 0 | 2 | 0 |
| 7 | 0 | 2 | 1 |
| 8 | 0 | 2 | 2 |

| rank | z | y | x |
|---|---|---|---|
| 9 | 1 | 0 | 0 |
| 10 | 1 | 0 | 1 |
| 11 | 1 | 0 | 2 |
| 12 | 1 | 1 | 0 |
| 13 | 1 | 1 | 1 |
| 14 | 1 | 1 | 2 |
| 15 | 1 | 2 | 0 |
| 16 | 1 | 2 | 1 |
| 17 | 1 | 2 | 2 |

| rank | z | y | x |
|---|---|---|---|
| 18 | 2 | 0 | 0 |
| 19 | 2 | 0 | 1 |
| 20 | 2 | 0 | 2 |
| 21 | 2 | 1 | 0 |
| 22 | 2 | 1 | 1 |
| 23 | 2 | 1 | 2 |
| 24 | 2 | 2 | 0 |
| 25 | 2 | 2 | 1 |
| 26 | 2 | 2 | 2 |

は大規模インターコネクトの性能評価を目的としたシミュレータである．OpenNSIM では MGEN プログラムと呼ぶ MPI 相当の C プログラムが駆動し，その通信時間を測定することが出来る．OpenNSIM の設定は文献 4) と同じものを使用した．この設定は，ノード間バンド幅 5GB/秒，ハードウェアレイテンシ 120 ナノ秒である．

最適化にあたって MOPT アルゴリズムを C++を使用して実装した．この実装は並列化をしていない，シングルスレッドプログラムである．実装したプログラムの実行は Intel Core i7 870 を搭載した一般的なコンピュータを使用した．

MOPT アルゴリズムとの比較に，XYZ 次元順にランクを割り当てた配置を使用する．これは単純な配置法である．XYZ 次元順割り当てを 3 × 3 × 3 の 27 ノードに適用した場合，**表 1** となる．この配置法を **XYZ 次元順**と表記する．

また，評価関数の効果を検証するため，MOPT に単純な評価関数を追加する．この評価関数は，MOPT アルゴリズムのステップ 2 において，すべての結合パターンを試さず，一つを選択する．図 3 の場合，辺 C1 と B2 を結合する．この手法を **MOPT none** と名付ける．

### 3.2 bruck のアルゴリズムによる Allgather の最適化

bruck のアルゴリズムによる Allgather プログラムを用いて，MOPT を評価する．bruck のアルゴリズムは MPI 集団通信に一般的に使用されるアルゴリズムである．

評価のために，Open MPI[10] を参考に bruck のアルゴリズムを実装した．実装した MGEN プログラムを**図 7** に示す．このプログラムは各ノードが 12 回通信を行う．1 回目の通信量は 2048byte であり，通信回数が増えるたびに通信量は 2 倍になる．この bruck のアルゴリズムを 16 × 16 × 16 の 4096 ノードで実行し，実行時間を計測する．ネットワークトポロジーは 3 次元トーラスである．

```
#include <stdio.h>
#include "mgen.h"

#define DATA_SIZE 2048
#define MGEN_TAG0 0

int MGEN_Main( int argc, char** argv )
{
    int my_rank, comm_size;
    MGEN_Comm_rank(MGEN_COMM_WORLD, &my_rank);
    MGEN_Comm_size(MGEN_COMM_WORLD, &comm_size);

    int dist;
    for (dist = 1; dist < comm_size; dist <<= 1) {
        MGEN_Request req1, req2;
        MGEN_Status status;

        MGEN_Irecv(NULL, dist*DATA_SIZE, MGEN_BYTE,
                   (my_rank + comm_size - dist)%comm_size,
                   MGEN_TAG0, MGEN_COMM_WORLD, &req1);
        MGEN_Isend(NULL, dist*DATA_SIZE, MGEN_BYTE,
                   (my_rank + dist)%comm_size,
                   MGEN_TAG0, MGEN_COMM_WORLD, &req2);
        MGEN_Wait(&req1, &status);
        MGEN_Wait(&req2, &status);
    }
    return 0;
}
```

**図 7** 評価に用いた MGEN プログラム

**表 2** Allgaher の結果

| 配置手法 | MGEN 実行時間 | 通信コスト | 最大リンク通信量 | MOPT 最適化時間 |
|---|---|---|---|---|
| XYZ 次元順 | 20.43(ms) | 223.3(hope×GB) | 45.10(MB) | |
| MOPT none | 2.99(ms) | 52.4(hope×GB) | 8.68(MB) | 0.7(s) |
| MOPT mincost | 2.99(ms) | 51.1(hope×GB) | 8.64(MB) | 55(s) |
| MOPT minlink | 2.87(ms) | 51.3(hope×GB) | 6.90(MB) | 541(s) |

評価結果を**表 2** にまとめる．表 2 の MGEN 実行時間とは，OpenNSIM が出力した MGEN プログラムの実行時間である．MGEN 実行時間は計算時間を含まない，ネットワークの通信処理時間である．

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

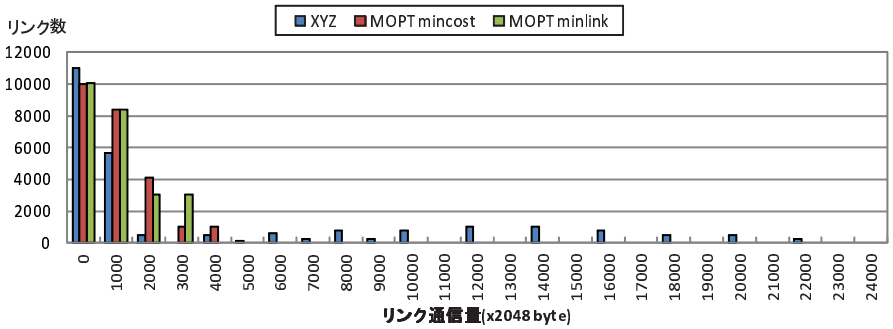


**図 8** 各リンクを流れたバイト数 (最小単位の 2048byte で割ったバイト数を表示している)

表 2 の MGEN 実行時間から，MOPT アルゴリズムによるタスク配置の結果，XYZ 次元順と比べ，通信時間の減少が確認できる．表 2 の通信コストから，XYZ 次元順に比べ，通信コストの減少が確認できる．また，通信コストを最小化した MOPT mincost が最も通信コストが低い．

表 2 の MGEN 通信時間を見ると，MOPT mincost と MOPT minlink は，MOPT none と大きな差がない．このことから，この Allgather プログラムでは，評価関数を用いた結合パターンの選択は大きな意味を持たず，通信量の多いペアを選択したことが有効であったことが分かる．

リンク通信量について考察するために，**図 8** に各リンクを流れたバイト数を示す．ただし，このバイト数は Allgather の最小通信量 2048byte で割った値をグラフの横軸としている．グラフの縦軸はあるバイト数流れたリンクがいくつあったかを示している．横軸は流れたバイト数である．このグラフから，XYZ 次元順タスク割り当に比べて，MOPT mincost, MOPT minlink で一つのリンクに通信が偏ることは少ないことが分かる．

最適化時間について考察する．表 2 から，MOPT mincost よりも MOPT minlink の方が最適化に時間がかかることが分かる．通信リンクは通信する経路を考えなければならないため，計算量が多い．どの手法も現実的な時間で終了していると言える．

### 3.3 NAS Parallel Benchmarks による評価

NAS Parallel Benchmarks[9] を使用し，MOPT アルゴリズムの評価を行う．評価には OpenNSIM を使用するが，OpenNSIM は一部の MPI 関数が実装されていないため，そのままでは NAS Parallel Benchmarks が実行できない．そのため，通信トレースを利用して実行を行う．通信トーレスは SimMc[11] を利用して取得する．SimMc の MPI ライブラリとして，集団通信を 1 対 1 通信で行うものを使用する．

**表 3** NAS Parallel Benchmarks CG の結果

| 配置手法 | MGEN 実行時間 | 通信コスト | 最大リンク通信量 | MOPT 最適化時間 |
|---|---|---|---|---|
| XYZ 次元順 | 2.75(ms) | 4.43(hope×GB) | 3.90(MB) | |
| MOPT none | 2.71(ms) | 3.64(hope×GB) | 4.84(MB) | 0.00(s) |
| MOPT mincost | 1.81(ms) | 3.36(hope×GB) | 2.44(MB) | 0.64(s) |
| MOPT minlink | 1.85(ms) | 3.38(hope×GB) | 2.93(MB) | 6.60(s) |

**表 4** NAS Parallel Benchmarks SP の結果

| 配置手法 | MGEN 実行時間 | 通信コスト | 最大リンク通信量 | MOPT 最適化時間 |
|---|---|---|---|---|
| XYZ 次元順 | 606 (us) | 1,504 (hope×MB) | 1,066 (KB) | |
| MOPT none | 519 (us) | 968 (hope×MB) | 1,117 (KB) | 0.01 (s) |
| MOPT mincost | 511 (us) | 726 (hope×MB) | 1,384 (KB) | 0.68 (s) |
| MOPT minlink | 475 (us) | 886 (hope×MB) | 968 (KB) | 6.86 (s) |

#### 3.3.1 NAS Parallel Benchmarks CG による評価

**表 3** に NAS Parallel Benchmarks CG の通信トレースを OpenNSIM で実行した結果を示す．ノード数は $8 \times 8 \times 8$ の 512 ノードで実行した．ネットワークトポロジーは 3D トーラスである．NAS Parallel Benchmarks CG のサイズは CLASS B を使用した．CLASS B ではプログラムの同じ実行パスを 75 回繰り返すが，そのうちの最初の 1 回を評価に使用する．

表 3 から MOPT mincost の MGEN 実行時間が最も短いことが分かる．この理由として，通信コストと最大リンク通信量が小さいことが挙げられる．MOPT mincost の最大リンク通信量が MOPT mincost よりも小さくなった理由は，MOPT mincost の通信コストの最小化によって，通信量全体が少なくなり，最終的な最大リンク通信量が小さくなったためと考えられる．

#### 3.3.2 NAS Parallel Benchmarks SP による評価

**表 4** に NAS Parallel Benchmarks SP の通信トレースを OpenNSIM で実行した結果を示す．OpenNSIM のノード数は $8 \times 8 \times 8$ の 512 ノードで実行した．ネットワークトポロジーは 3D トーラスである．ただし，SP は $n^2$ ノードでしか動作しないため，SP の実行は $22^2$

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

の 484 ノードで実行した．MOPT での最適化にあたっては，残りの 28 ノード（512 – 484）を通信をしないノードとして追加した．また，CG 同様，繰り返しの最初の 1 回を評価に使用する．

表 3 から MOPT minlink の MGEN 実行時間が最も短いことが分かる．このことから，通信しないノードを追加した MOPT アルゴリズムが適切に最適化を行うことが分かる．

## 4. 関連研究

高性能計算機のためのタスク配置アルゴリズムとして，RMATT[4] が提案されている．RMATT では最適化に Simulated annealing を使用する．ただし，探索空間の圧縮のために，タスクを複数のグループに分割する．MOPT のようにタスクのグループを作成する手法とは大きく異なっている．

Neowork on Chip のルータバッファ削減のための配置アルゴリズムとして，BMAP[12] が提案されている．この手法は MOPT と同様に，タスクのペアを作ることで最適化を行う．しかし，タスクのペアの作り方に大きな差がある．BMAP では通信が集中するようにペアを選択するが，MOPT では通信を分散させるようにペアを選択するため，大きな違いがある．

## 5. おわりに

本稿では，数万ノードを超えるメッシュ/トーラス接続網を採用した計算機向けに，高速なタスク配置手法 MOPT を提案した．この手法は通信量の多いタスクのペアを結合，タスクのグループを作ることによって最適化を行う．タスクの結合は様々なパターンが考えられるため，評価関数を用いてそれらのパターンから最適なものを選択した．通信コストを最小化するアルゴリズムとして MOPT mincost を提案し，最大リンク通信量を最小化するアルゴリズムとして MOPT minlink を提案した．これらの提案手法は，ノード数 $N$ に対して，$N^2 \log N$ の計算量である．

OpneNSIM を使用し，4096 ノードの Allgahter アルゴリズムの通信時間を評価したところ，通信時間は 14%に減少した．また，最適化にかかった時間は MOPT mincost で 55 秒と比較的短いものであった．

今後の課題として，MOPT アルゴリズムの 2 のべき乗ノード数以外への対応を考えている．

## 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 (JST CREST) の「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．OpenNSIM の開発と障害復旧にあたっていただいた，九州先端科学技術研究所の柴村英智様に感謝いたします．実験の設定ファイルを提供してくださった，富士通研究所の住元真司様に感謝いたします．

## 参考文献

1) The BlueGene/L Team: An Overview of the BlueGene/L Supercomputer, *ACM/IEEE Conference on Supercomputing*, p.60 (2002).
2) Yokokawa, M., Shoji, F., Uno, A., Kurokawa, M. and Watanabe, T.: The K computer: Japanese next-generation supercomputer development project, *International Symposium on Low Power Electronics and Design (ISLPED)*, pp.371 –372 (2011).
3) Bland, A.S., Kendall, R.A., Kothe, D.B., Rogers, J.H. and Shipman, G.M.: Jaguar: The world's most powerful computer, *Cray User Group*, pp.371 –372 (2009).
4) 今出広明，平本新哉，三浦健一，住元真司：大規模計算環境のためのランク配置最適化手法 RMATT，先進的計算基盤システムシンポジウム論文集， Vol.2011, pp.340–347 (2011).
5) Ascia, G., Catania, V. and Palesi, M.: A Multi-objective Genetic Approach to Mapping Problem on Network-on-Chip, *Journal of Universal Computer Science*, Vol.12, No.4, pp.370–394 (2006).
6) 三好健文，笹田耕一，植原　昴，佐野伸太郎，森　洋介，吉瀬謙二：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，情報処理学会研究報告. 計算機アーキテクチャ研究会報告 (ARC-184)， Vol.2009, No.10, pp.1–7 (2009).
7) 英智柴村，竜太郎薄田，智也平尾，真　吉田，隆行神戸，英樹三輪，郁夫三吉，弘士井上，和彰村上：クラウド環境による OpenNSIM インターコネクトシミュレーションサービス，情報処理学会研究報告. 計算機アーキテクチャ研究会報告， Vol.2010, No.15, pp.1–9 (2010).
8) Bruck, J., Ho, C.-T., Kipnis, S., Upfal, E. and Weathersby, D.: Efficient algorithms for all-to-all communications in multiport message-passing systems, *IEEE Transactions on Parallel and Distributed Systems*, Vol.8, No.11, pp.1143 –1156 (1997).
9) Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., Simon, H.D., Venkatakrishnan, V. and Weeratunga, S.K.: The NAS parallel benchmarks, *NASA Technical Report* (1991).

Vol.2011-ARC-197 No.15
Vol.2011-HPC-132 No.15
2011/11/28

10) Gabriel, E., Fagg, G., Bosilca, G., Angskun, T., Dongarra, J., Squyres, J., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R., Daniel, D., Graham, R. and Woodall, T.: Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Vol.3241, pp.353–377 (2004).
11) 植原　昂，佐藤真平，吉瀬謙二：メニーコアプロセッサの研究・教育を支援する実用的な基盤環境，電子情報通信学会論文誌. D, 情報・システム， Vol.93, No.10, pp.2042–2057 (2010).
12) Shen, W.-T., Chao, C.-H., Lien, Y.-K. and Wu, A.-Y.: A New Binomial Mapping and Optimization Algorithm for Reduced-Complexity Mesh-Based On-Chip Network, *First International Symposium on Networks-on-Chip(NOCS)*, pp.317 –322 (2007).