通信衝突削減のためのタスク配置最適化の評価

森江, 善之
九州大学大学院システム情報科学府 | 九州システム情報技術研究所

南里, 豪志
九州大学情報基盤センター

石畑, 宏明
東京工科大学コンピュータサイエンス学部

井上, 弘士
九州大学大学院システム情報科学研究院

他

出版情報：情報処理学会研究報告, 2008-HPC-114. 2008 (19), pp.79-84, 2008-03-05. 情報処理学会
バージョン：

通信衝突削減のためのタスク配置最適化の評価

森江善之[†1]　南里豪志[†2]　石畑宏明[†3]
井上弘士[†4]　村上和彰[†4]

本稿では，通信性能の悪化の主要因である通信の衝突を避けるためのタスク配置最適化の評価を行った．著者らはメッセージのタイミングを制御しつつ，衝突を回避するタスク配置最適化の研究をしている．その中でその最適化をツリートポロジに適用して，その手法が有効であることを示した．一方，メッシュやトーラスなどのネットワークトポロジにはホップ数を評価関数とする通信衝突削減のためのタスク配置最適化が研究なされており，有効性があるとされている．そこで，ネットワークトポロジが 3D メッシュの際に通信衝突回数を評価関数としたタスク配置最適化を適用した場合とホップ数を評価関数したタスク配置最適化を適用した場合にどのような違いがあるか調べる実験をし，考察を行った．

Evaluation of optimization of task allocation for reducing contentions

Yoshiyuki Morie,[†1] Takeshi Nanri,[†2] Hiroaki Ishihata,[†3]
Koji Inoue[†4] and Kazuaki Murakami [†4]

In this text, we evaluated the optimization of task allocation to avoid contentions that was the key factor of the communication performance degradation. We applied the optimization of task allocation controlling the timing of the message for avoiding contentions to the tree topology, and showed it was effectiveness. On the other hand, there were some optimizations of task allocation for reducing contentions. Those optimizations used the evaluation function which used the number of hops. Those optimizations against the mesh and torus topology were effective. We experimented and investigated what 's the difference between the optimization of task allocation which the evaluation function was the number of contentions and the number of hops when the network topology was 3D mesh. We considered about it.

1. はじめに

近年のスーパーコンピュータ性能ランキング TOP500[4] などにみられるように大規模並列計算機の計算ノード数の増加は著しい．計算ノード数が増加するとネットワークのためのハードウェアコストが莫大となる．このため，一般にメッシュやトーラス，ファットツリーなどのネットワークトポロジが用いられる．しかし，このようなネットワークトポロジでは，通信の衝突が頻発し，通信性能が悪化する．著者らはネットワークトポロジに起因する通信の衝突を回避し，通信性能を向上させる研究を行っている[1]．ツリートポロジによる実験は行ったが，3D メッシュといった大規模並列計算機で多く採用されているネットワークトポロジにおいての評価を行っていない．3D メッシュでは，他にもいくつか通信衝突削減のためのタスク配置最適化が提案されている．そこで，本稿では 3D メッシュにおける通信衝突削減のためのタスク配置最適化の効果について調べ，考察を行った．

本稿は，以下のように構成される．2 章で通信衝突削減のためのタスク配置最適化について述べる．3 章で実験・考察について述べ，最後の 4 章でまとめと今後の課題について述べる．

2. 通信衝突削減のためのタスク配置最適化

メッシュ，トーラスやツリーなどのネットワークトポロジでは，直接接続していない計算ノード同士が多

†1 九州大学大学院システム情報科学府/九州システム情報技術研究所
Graduate School of Information Science and Electrical Engineering, Kyushu University/Institute of Systems Information Technologies/KYUSHU
†2 九州大学情報基盤センター
Computing and Communications Center, Kyushu University
†3 東京工科大学コンピュータサイエンス学部 Tokyo University of Technology, School Computer Science
†4 九州大学大学院システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University

数存在する．これらの計算ノードが通信を行う際は，複数の他の計算ノードやスイッチを経由することになる．通信はネットワークの随所で行われるため，ある通信の経路が他の通信の経路と交わることがある．このことは通信の衝突を発生させ，通信時間を増加させる．

そこで，計算ノードに通信衝突が少なくなるようにタスクを割りつけることで通信時間の削減を行う．このような最適化のことをタスク配置最適化と呼ぶ．また，スイッチや計算ノードを経由する際に通るリンク数をホップ数と呼ぶ．

2.1 ホップ数を評価関数とするタスク配置最適化

T. Agarwal ら[2] や G. Bhanot ら[3] は，通信の衝突を削減するため，各通信のホップ数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タスク配置最適化手法を提案している．これらの手法では，複数のリンクを経由する通信が少なくなるので，結果的に通信が衝突する可能性が低くなる．これにより，通信衝突が削減され，通信性能の向上がなされる．

これらの手法の評価関数を式 1 に示す．

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

i, j はタスク，n はタスク総数である．$\pi(i)$ 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 への通信のホップ数を返す関数である．計算ノード間の通信量とホップ数の積をホップバイトと呼ぶ．1 バイトのデータを 1 ホップ分離れた計算ノード間で通信する際の通信コストを 1HB とする．この評価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．このため，この評価関数を最小とするタスク配置では，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．

2.2 通信衝突回数を評価関数とするタスク最適化

著者らは，各径路でどれほど通信衝突が発生するかを調べ，タスク配置を行う研究を行っている[1]．

ここでの通信衝突は同時に同一リンクを同一方向に通るメッセージが複数存在する場合に発生するものとする．このため，通信衝突を調べるには，どの通信が同時に実行されるかを知る必要がある．

問題を簡単化するため，仮定を置くこととする．まず，メッセージサイズはすべて等しいとする．通信は完全に衝突するか全く衝突しないかのどちらかとなる．また，同時に実行される通信の集合を与える．この集合を Concurrent Communication Set(CCS) とよぶ．同じ CCS に属す通信同士が同一リンクを同一方向に通過する場合は衝突が発生したとする．実際には同時に通信を実行するか分からないため，図 1 のように CCS ごとに同期をとり，同一 CCS に属する通信は同時に実行するようにする．

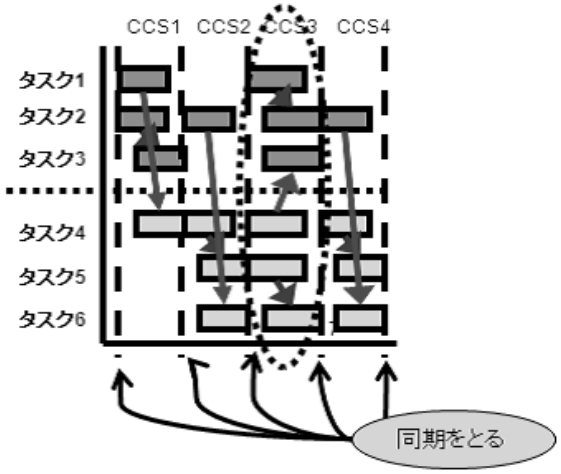

図 1 衝突を制御するための同期

評価関数では，各 CCS における各リンクで実行される通信の数の最大値の総和を求める．評価関数を式 2 に示す．以下のこの評価関数の値を通信衝突回数と呼ぶことする．

$$\sum_{t=0}^{N-1} \max_{i=0\cdots n-1} coll(t,\pi(i),\pi(tork(i))) \qquad (2)$$

t は CCS の順序，N は CCS の総数である．i はタスク，n はタスク総数である．$\pi(i)$ はタスク i が割り付けられている計算ノードを返す関数である．$tork(i)$ はタスク i の通信先のタスクを返す関数である．$coll(t,p,q)$ は t 番目の CCS における計算ノード p，q 間の各経路で要求される通信の数の中で最大の値を返す関数である．

各リンクでの通信衝突を調べるため，マルチパスとなるネットワークトポロジを持つ並列計算機に適用する場合は，ルーティングを考慮する必要がある．ルーティングは静的ルーティングを行える必要がある．これは動的ルーティングではメッセージがどのリンクを経由するか確定することができないからである．

3. 実　　験

3.1 実 験 概 要

大規模並列計算機で用いられるネットワークトポロジである 3D メッシュにおいて通信衝突削減のためのタスク配置最適化を適用した場合の性能評価実験を行う．今回は，通信衝突回数を評価関数に用いたタスク配置最適化とホップ数を評価関数に用いたタスク配置最適化を適用した場合のプログラムの実行性能の比較を行う．対象のプログラムには NAS Parallel Benchmark[5] の CG 法の通信パターンを抽出したプログラ

ムを用いた．

3.2 実験方法と環境

実験の手順としては CG 法の通信パターンに対してホップ数を評価関数としたタスク配置最適化と通信衝突回数を評価関数としたタスク配置最適化を行い，それぞれのタスク配置を適用し，実行する．本実験では，対象プログラムである CG 法の通信パターンを抽出したプログラムを 1000 回ずつ実行し，実行時間の平均値を求めた．また，ホップ数と実行時間の関係，通信衝突数と実行時間の関係を調べるため，上記の処理を 20 回行った．

まず，CG 法の通信パターンの CCS への分割方法を示す．全タスクを正方行列の順にならべ，各タスクを行番号 i と列番号 j に対応付ける．行番号が等しいタスク間で recursive doubling を行い，その後，行番号 i と列番号 j をもつタスクと行番号 j と列番号 i をもつタスク間で送受信を行う．recursive doubling とその後の送受信にはメッセージに依存関係があり同時に実行できない．そこで，まず recursive doubling とその直後の通信で分ける．次に同一行番号で行う recursive doubling では， k ステップ目において各タスクが 2^k 離れたタスク同士で送受信を行う．ステップ間の通信に依存関係があるため，同時に通信を行うことができない．そのため，ここで各 CCS に分ける．

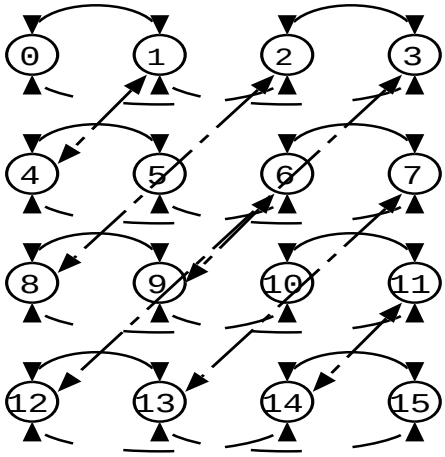

図 2 CG 法の通信パターン

次に，タスク配置の探索法であるが，今回はどちらの最適化においても Simulated Annealing を用いた．これは，ホップ数と通信衝突回数を評価関数として用いることでプログラムの実行時間にどのような影響が出るかを調べるため，探索方法を同一条件にするためである．また，計算ノード数が多くなると Simulated Annealing のような確率的な近似アルゴリズムを用いなければ，現実的な時間で探索を終了することができないためである．図 3 に疑似コードを示す．

最後に表 1 に実験環境を示す．

```
//T は温度    T0 は初期温度
T=T0
//初期タスク配置
pre_alloc=init_alloc
 //Tend は閾値
while(T>Tend){
 //alloc,pre_alloc はタスク配置
  alloc=pre_alloc
//E() は評価関数
  enagy=E(alloc)

  diff_enagy=pre_enagy−enagy
  Thold=exp(diff_enagy/T)
 //0〜1 の値の生成
  r=rand()/RAND_MAX

  if(Thold>r)then
    if(min_enagy>enagy) then
    min_enagy=enagy
        min_alloc=alloc
      endif
    pre_enagy=enagy
    alloc=pre_alloc
  endif
//温度の更新
  T=T∗0.9
enddo
```

図 3 SA の疑似コード

表 1 実験環境

計算ノード数	1024(2CPU/nodes)
CPU	PowerPC440 700GHz
RAM	1GB
OS	CNK (Compute Node Kernel)
コンパイラ	XLC++コンパイラ XLFortran
トポロジ	3D メッシュ・3D トーラス
通信帯域幅	1.4Gbps

本実験環境である BlueGene/L のルーティングでは，大きなメッセージの場合は動的ルーティングを用い，小さなメッセージの場合は静的ルーティングを用いる．その閾値は BGLMPI_RZV という環境変数で決められる．メッセージサイズがこの閾値を超えると動的ルーティングが用いられる．今回は環境変数をメッセージサイズより大きくし，動的ルーティングが使われないようにして実験を行った．動的ルーティングでは通信衝突回数を評価関数とするタスク配置最適化は適用できないためである．

Bluegene/L で用いられている静的ルーティングは

Dimension Order Routing (以下，DOR) であると考えられる．これは詳細なデータを入手することが出来なかったため，実験的に導いたものである．もし，DOR であったら，通信衝突を起こすような並列プログラムを実行してその時間を計測し実際に実行時間が増加するかを見た．

以下，DOR の説明を行う．簡単のため，2D-Mesh で説明を行う．まず，x 次元方向の計算ノードの転送を行う．宛先の計算ノードと x 座標が等しい計算ノードへ転送を行う．x 座標が等しい計算ノードに到着したら，今度は y 次元方向の計算ノードへ転送を行う．y 座標が等しい計算ノードまで転送を行う．このように送信タスクと受信タスクの割りつけられる計算ノードが決まれば，メッセージが通るリンクは一意に決まる．例として図 4 においてタスク 0,0 からタスク 2,2 へ送信が行われるとする．この場合，一旦タスク 2,0 を経由して 2,2 へと送られる．

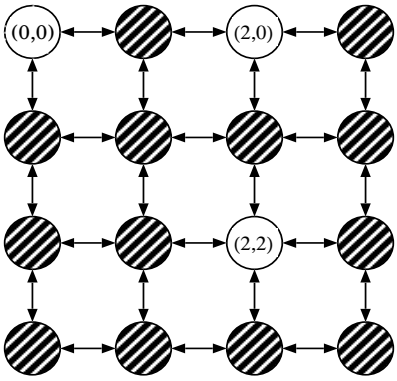

図 4 Dimension Order Routing (DOR)

3.3 実験結果

図 5 に実験結果を示す.

coll ave. は通信衝突回数を評価関数とするタスク配置最適化の実行時間の平均値である．同様に，hops ave. はホップ数を評価関数とするタスク配置最適化の実行時間の平均値である．coll min は通信衝突回数を評価関数とするタスク配置最適化の実行時間の最小値である．同様に hops min はホップ数を評価関数とするタスク配置最適化の実行時間の最小値である．

32，64，128 ノードのすべてでホップ数を評価関数とするタスク配置最適化の実行時間の平均値が短くなった．128 ノードでは通信衝突を評価関数とするタスク配置最適化の最小値がホップ数の評価関数とするタスク配置最適化の平均値を上回っている．

3.4 実験の考察

図 5 の実験結果からホップ数を評価関数とするタスク配置最適化の方がより高性能となっている．これは，通信衝突回数を評価関数とするタスク配置最適化が，無駄に同期待ちをしているためである考えられる．

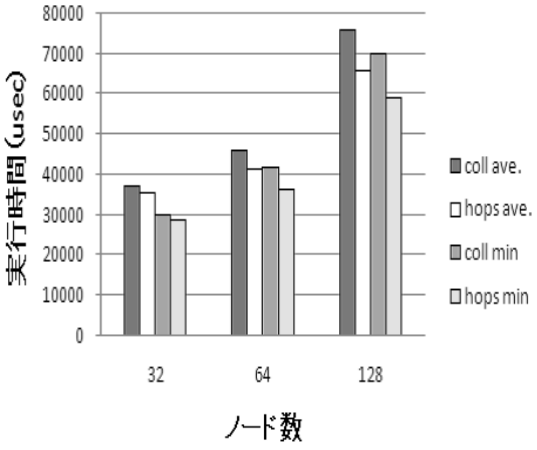

図 5 実験結果

ホップ数を評価関数とするタスク配置最適化ではどの通信が同時に実行されるか知る必要がないため，同期は行う必要がない．同期がなければ実行可能な通信があれば，すぐに通信を実行できるため，同期をしないことにより新たな衝突が起きない限りは性能が向上すると考えられる．また，ホップ数が少ないため，通信衝突の可能性が低くなっていると考えられる．これらのことからホップ数を評価関数とするタスク配置最適化の方が項性能になっていると考えられる．しかし，実際にホップ数が少なくなれば実行時間がすくなるくなるわけではない．

そこで，ホップ数と実行時間の相関関係を調べた．計算ノード数 32 の場合は，図 6 からわかるように，ホップ数が小さい場合には実行時間が短くなり，大きい場合には実行時間が長くなり相関関係は強いと言える．しかし，計算ノード数 64，128 では図 7，8 からは強い相関関係はみられない．これでは，ホップ数を最小に探索することが良いとは一概に言うことはできないと考えられる．

次に，評価関数である通信衝突回数と実行時間の相関関係を調べた．あまり通信衝突回数に差がない．図 9，10，11 から通信衝突回数が小さい場合には実行時間が短くなる傾向が見える．しかし，同じ通信回数の場合でも実行時間に差があり，評価関数の精度が高くないと考えられる．これは仮定において，通信衝突を発生したら，リンクにおける通信帯域を共有すると考えていることに問題があると考えられる．本実験環境の BlueGene/L では，スイッチおいて同時に通信が要求された場合は，片方の通信がブロックされ，もう一方の通信が先に送り出される．これではブロックされていた通信がのちにまた通信衝突を起こす可能性があり，そのようなことが起こった場合には評価関数の結果とは違う結果になると考えられる．

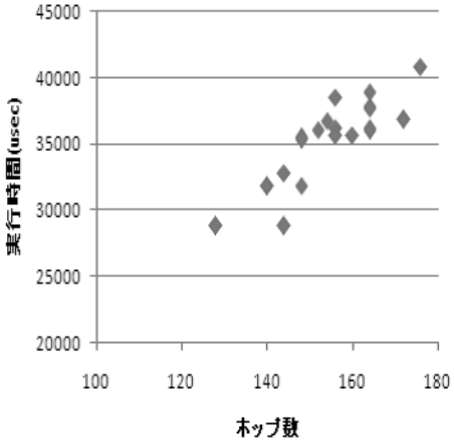

図 6 ホップ数と実行時間の関係 (32 ノード)

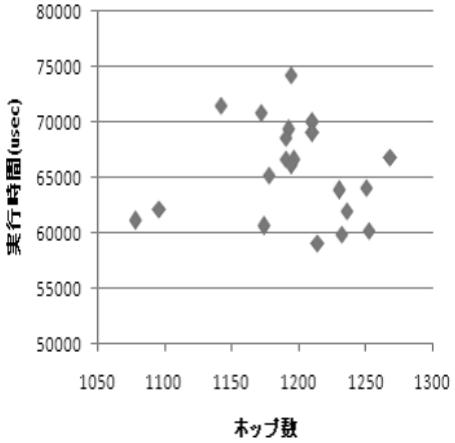

図 8 ホップ数と実行時間の関係 (128 ノード)

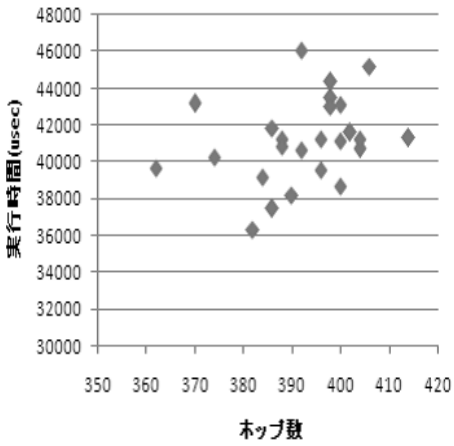

図 7 ホップ数と実行時間の関係 (64 ノード)

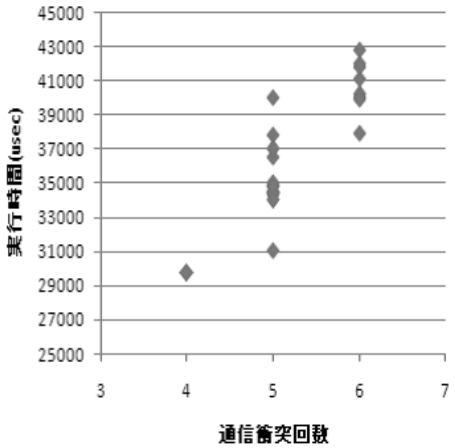

図 9 通信衝突回数と実行時間の関係 (32 ノード)

4. おわりに

本稿では，通信衝突削減のためのタスク配置最適化である 2 つのタスク配置最適化の評価を行った．ホップ数を評価関数とするタスク配置最適化と通信衝突回数を評価関数とするタスク配置最適化の評価を取り上げて評価した．大規模並列計算機で用いられる 3D メッシュをトポロジとした場合にはどちらがより有効であるかの調査を行った．ホップ数を評価関数とするタスク配置最適化の方がより高性能であったが，評価関数と実行時間の相関関係が弱い部分が見られた．

今後の課題は，今回の実験では計算ノード数が 128 個までの評価であったため，計算ノード数がさらに大きい場合でどのような結果になるか実験を行う必要があると考えられる．128 ノード程度であるとホップ数がそこまで大きくならないため，実験のような結果になった可能性もあるからである．また，他の通信パターンについても実験を行う必要があると考えられる．これらの実験を行うことで本実験において通信衝突回数を評価関数とするタスク配置最適化の性能向上が一様に低かった原因とその改善法を考えるつもりである．

謝辞 本研究は,「ペタスケール・システムインターコネクト技術の開発」プロジェクト (文部科学省 「次世代 IT 基盤構築のための研究開発」の研究開発領域「将来のスーパー コンピューティングのための要素技術の研究開発」（平成 17〜19 年度）の一つ), 理化学研究所「次世代生命体統合シミュレーションソフトウェアの研究開発」プロジェクトによるものである．

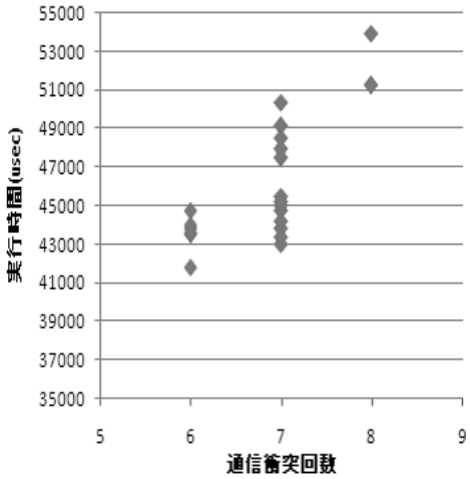

図 10 通信衝突回数と実行時間の関係 (64 ノード)

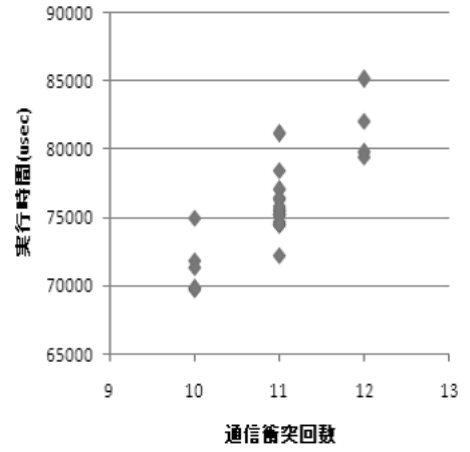

図 11 通信衝突回数と実行時間の関係 (128 ノード)

参 考 文 献

1) 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 (ACS19)，2007 年
2) T. Agarwal, A.Sharma, L. V. Kale, :Topology-aware task mapping for reducing communication contention on large parallel machines, Proceedings of IEEE International Parallel and Distributed Processing Symposium 2006, pp.1-10, in 2006.
3) Gyan Bhanot, Alan Gara, Philip Heidelberger, Eoin Lawless, James Sexton, Robert Walkup, :Optimizing task layout on the Blue Gene/L supercomputer. IBM J. Res. & Dev. 49, No. 2/3, pp.489-500, in 2005.
4) :TOP500 Supercomputer Sites, `http://www.top500.org/.`
5) :NAS Parallel Benchmark, `ttp://www.nas.nasa.gov/Resources/Software/npb.html`