6H-4

メニーコアプロセッサのためのタスク配置手法RMAPの評価

佐野 伸太郎† 吉瀬 謙二†
† 東京工業大学 大学院情報理工学研究科

1 はじめに

プロセッサに搭載されるコア数が増加したメニーコアアーキテクチャでは，Network-on-chip(NoC) によるコアの接続が予想される．NoC による接続は従来のバスやクロスバと違い，コア間の通信レイテンシが一定ではない．このため，どのコアにどのタスクを配置するのかという問題 (タスク配置問題) が重要になる．

本稿では，メニーコアプロセッサの性能向上を目指すタスク配置手法を提案，評価する．

2 タスク配置手法の提案

2 次元メッシュを対象としたタスク配置手法を提案する．提案する配置手法は図 1 のような 2 段階のプロセスを採用する．図 1(a) は並列化されたアプリケーションのタスクを表す．丸がタスクであり，矢印が通信の様子を表す．このような並列化されたアプリケーションはタスクの配置方法により，性能が異なる．提案手法ではタスクグラフに最適化手法 MOPT を適用し，密な配置を求める (図 1(b))．その後，実際に使用するプロセッサのコア数を考慮し，配置手法 RMAP[1] によって配置を行う (図 1(c))．

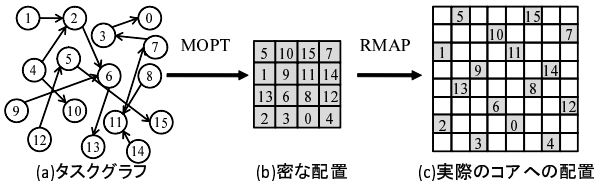

図 1: 提案するタスク配置手法の流れ

2.1 MOPT：配置最適化アルゴリズム

使用するアルゴリズムの全体像を図 2 に示す．灰色の枠で囲ったものをタスクリストと呼び，結合したタスクをタスク群と呼ぶ．このアルゴリズムはペアを作成するステップを繰り返すことで，最適な配置を求める．ステップ開始時にタスク群は単独のタスクである．ステップを n 回繰り返すことによりタスクは 2n 個のタスク群となる．ステップの内容は以下の 1),2),3) によって構成される．

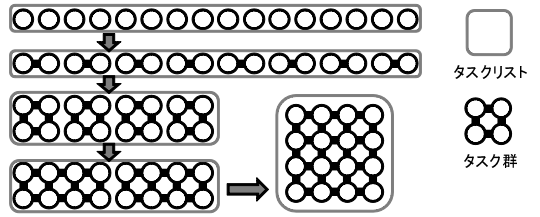

図 2: 配置最適化アルゴリズム MOPT の例

1) 現在のタスクリストからタスク群を一つ選択する．選択したタスク群と最も通信量の多いタスク群をペアとする．

2) 上で選択した 2 つのタスク群を結合する．結合するときに向き，裏表を変更し最も通信コストの少ない結合方法を選ぶ．通信コストは以下で定義されるものである．

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} Hop(Pos(i), Pos(j)) \times Comm(i,j)$$

ここで i, j はタスクを表し，n は結合するタスク群の総タスク数，Hop はホップ数，Pos はタスクを配置した位置，$Comm$ はタスク間の通信量である．

3) すべてのタスク群がペアになるまで，1),2) を繰り返す．

この最適化手法を **MOPT**(Merge Optimization) と名付ける．

2.2 RMAP：n ルーク問題を応用した配置手法

通信衝突削減のためのタスク配置手法を提案する．まず，衝突数削減のために，衝突がない理想的な配置を考える．衝突のない配置は 2 つのタスクが同じ行と列に存在しない配置によって得ることができる．このような配置は n ルーク問題を解くことによって発見できる．n ルーク問題は，配置問題の n クイーンをルークに置き換えたものである．n ルーク問題を解くためには，タスク数 N_{task} とすると N_{task}^2 個のコアが必要となる．このままでは，64 コアに 8 タスクしか配置することができない．

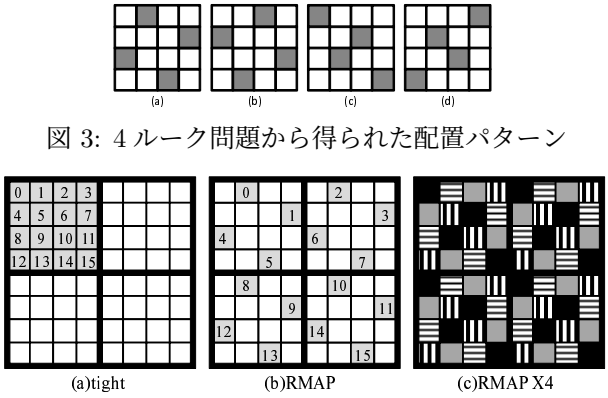

図 4: 64 ノードに RMAP を適応した配置

次に，衝突数を減らしながらより多くのタスクを配置する方法を考える．これは，与えられたコア数よりも小さなコア数の n ルーク問題の解を考えることによって解決できる．本稿では，4×4 コアの 4 ルーク問題の解を考える．4 ルーク問題の解は 24 個あるが，そのうちの 4 つを図 3 に示す．灰色で示したノードがタスクを配置する部分である．先に示したように，この配置では通信衝突が起きない．図 4(a) の 16 タスクを 64 コアに配置する場合，図 3(a) を 4 つ並べたパターンにおける灰色の部分にタスクを配置する．タスク配置の結果は図 4(b) である．このように，4-ルーク問題の解をタイル状に並べる手法を **RMAP**(Rook Mapping) と名付ける．

提案した RMAP を使用すると，タスク数 N_{task} に対して $4N_{task}$ 個のコアが必要となり，$3N_{task}$ 個のコアが余る．この余剰ノードを活用するために，複数の配置パターンを重ね合わせる手法を提案する．図 3 で示した 4 ルーク問題によって得られた 4 つの解 (a) から (d) はそれぞれ別々のコアを使用している．そのため，4 つの解

Evaluation of RMAP: A method of task mapping for many-core processor
Shintaro SANO†, and Kenji KISE†
†Tokyo Institute of Technology

を重ね合わせることができる．4つのパターンを重ね合わせると図4(c)のようになる．このように，4ルーク問題の解を重ね合わせる手法を **RMAP X4** と名付ける．

3 評価

3.1 評価方法

評価にはM-Coreアーキテクチャ[2]のソフトウェアシミュレータであるSimMcを用いる．ルータバンド幅は32bitである．プロセッサコアは2命令発行である．ルーティングはXY次元順ルーティングである．

ベンチマークとしてNAS Parallel Benchmarksからcg, ft, is, mgを用いる．問題クラスはWである．

評価に使用した配置方法を図5に示す．図5中のsequentialは各ベンチマークをMPIランク順に2次元メッシュに配置したものである．図5では16タスクを64コアに配置しているが，実際の評価には64タスクを256コアに配置したものを使用した．性能の評価には，**図5'seq+tight'** を基準とした性能向上率を用いる．

MOPTについて評価する． 図6'MOPT+tight'にMOPTによって得られた配置の性能向上率を示す．cgベンチマークにおいて最大8.5%の性能向上を示す．ただし，ftベンチマークにおいては0.8%低下している．これはftベンチマークに含まれる全体全通信が，sequentialに配置したときに最適化されているためであると考えられる．MOPTは効果のあるものを見きわめる必要がある．

RMAPについて評価する． 図6'seq+RMAP'からRMAPの使用によって，平均4.6%の性能向上を示すことが分かる．また，個々のベンチマークすべてで性能向上を示している．このことから，RMAPが効果的であることが分かる．

MOPTとRMAPを組み合わせた場合を考える．MOPTで効果が見えるcgベンチマークを見ると，'MOPT+tight'で8.5%，' seq+RMAP'で9.2%，'MOPT+RMAP'で11.9%の性能向上を示す．2つの手法を組み合わせることで，さらなる性能向上を狙えることが分かる．

RMAP X4について評価する． RMAP X4を評価するに当たって，比較はseq+tightを4つ敷き詰めたものと比較した．また，各ベンチマークを繰り返し実行する．これは常にすべてのベンチマークが動いている状況を作るためである．各ベンチマークの性能は3回以上の実行の平均によって算出した．

このように評価したとき，図6'seq+RMAP X4'で平均3.5%の性能向上を示す．複数のRMAPを組み合わせることは効果的であることが分かる．しかし，これは'seq+RMAP'の4.6%と比べると低い値である．原因として複数のアプリケーションを同時に走らせることで，ネットワークの混雑度が増加することが挙げられる．tightな配置であれば，4つのアプリケーションを同時に実行しても，他のアプリケーションによる通信の影響を受けない．これは対象アーキテクチャがXY次元順ルーティングを採用しているためである．

MOPTとRMAP X4を組み合わせた場合を考える．MOPTで効果が見えるcgベンチマークを見ると，'seq+RMAP X4'で8.0%，' MOPT+RMAP X4'で10.6%の性能向上を示す．MOPTとRMAP X4を組み合わせることで，さらなる性能向上を狙えることが分かる．

MOPTによる配置最適化の効果をより詳しく見るために，cgベンチマークのみにMOPTを適用し，その性能向上率を示す．図6'cgOnlyOpt+RMAP X4'にその性能を示す．'cgOnlyOpt+RMAP X4'の性能向上率の平均は4.5%であった．これは'seq+RMAP X4'の3.5%と比べて性能向上率が高い．また，cg以外のベンチマークも'seq+RMAP X4'に比べ性能向上を示している．これは，cgを最適化したことにより，通信の混雑度が低下し，全体の性能が向上したことが分かる．

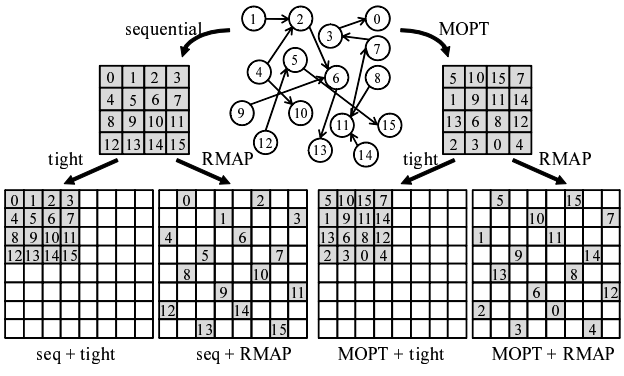

図 5: 評価に使用した配置方法

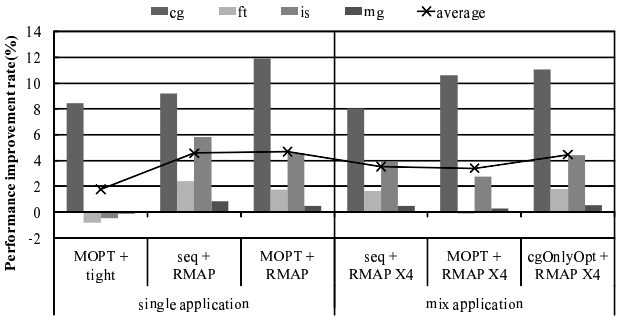

図 6: NAS Parallel Benchmarks による実験結果

4 関連研究

本稿で採用した配置最適化アルゴリズムは文献[3]と類似してるが，ペアの選択方法が異なる．

5 おわりに

本稿ではタスク配置手法としてMOPTとRMAPを提案した．提案手法をNAS Parallel Benchmarksによって評価した．256コアに64並列のアプリケーションを4つ同時に動かしたとき，RMAP X4とMOPTを部分的に使用した場合，単純な配置と比較し，平均4.4%の性能向上を示した．

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] Shintaro Sano et al. Pattern-based Systematic Task Mapping for Many-core Processors. *UPDAS 2010*, pp. 173 –178, November 2010.

[2] 植原他. 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, October 2010.

[3] Wein-Tsung et al. A New Binomial Mapping and Optimization Algorithm for Reduced-Complexity Mesh-Based On-Chip Network. *NOCS 2007*, pp. 317 –322, May 2007.