2013 年 7 月 8 日 情報システム基盤学合同輪講発表資料

A Task Remapping Technique for Reliable Multi-core Embedded Systems

著者： Chanhee Lee，Hokeun Kim，Hae-woo Park，Sungchan Kim，Hyunok Oh，Soonhoi Ha
出典： *International Conference on Hardware/Software Codesign and System Synthesis，pp.307-316，2010.*
発表者： 高性能コンピューティング学講座 本多・近藤研究室　1353028 朴 傑

1　はじめに

プロセッサは半導体製造技術の向上で高速化・高集積化を達成してきた。近年では、トランジスタの寿命が短くなり、プロセッサコアの故障率の増大が問題点になっている。プロセッサコアが故障した場合、そのコアに割り当てたタスクを他のコアにマイグレーションすれば、耐故障性の向上が見込める。従来から、マイグレーションに必要なコストを抑えるようタスクマイグレーションを行う手法が提案されてきた [1]。しかし、ストリーミング・アプリケーションのような入力ストリームに対してタスクを周期的に実行するアプリケーションでは、マイグレーション後のスループットの低下を最小限に抑えることが重要である。

本研究では、ストリーミング・アプリケーションをターゲットとし、スループットの低下を最低限に抑えつつ、かつマイグレーションコストを抑える手法を提案する。

2　問題定義

本研究では図 1(a) に示すようなタスクセットを入力とする。3 コアを持つプロセッサの場合、正常状態ではタスクは 3 つのコアに割り当てるが、その中のどれか 1 つが故障した場合、故障したコアに割り当てたタスクを他のコアにマイグレーションする (図 1(b))。

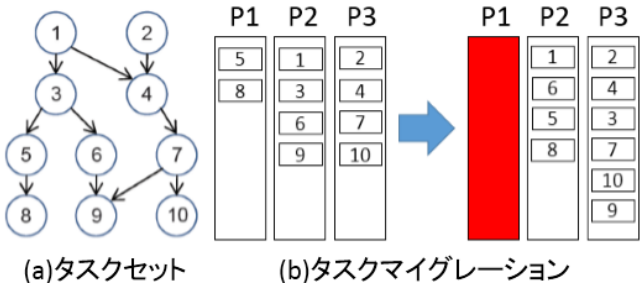

図 1: タスクセットとタスクマイグレーションの概念図

本研究で想定するシステムは processor-pool (pp) をベースとしたマルチコアプロセッサである。数多くあるコアをいくつかの pp に分け、pp 内はいくつかのコアがバスを経由して pp 内の専有メモリに繋がっている。同一 pp 内のコアは全て均質である。本研究では pp 内のマイグレーションを想定する。

3　提案手法

提案手法は (1) タスクスケジューリング/マッピング (2) コア故障時のタスクリマッピング (3) エンコーディングの 3 つのステップに分けられる。

(1) ではコンパイル時に正常状態と故障が起きた状態それぞれについて、スループット性能を最大にする静的スケジューリング/マッピングを行う。(2) では (1) で求めた結果を用い、各コアの故障パターンに対してタスクマイグレーションコストを最小限に抑えるリマッピング結果を求める。(3) ではリマッピング結果を効率的にメモリに保存するために split encoding 法を用いて、少ないデータ量でタスク割当てパターンを保存する。その後 (1) ステップに戻り更にコアに故障が発生した場合についてリマッピング結果を求める。

以下に 3 つのステップについて詳細に記述する

3.1　タスクスケジューリング/マッピング

提案手法のステップ (1) には QEA (Quantum-inspired Evolutionary Algorithm)[2] アルゴリズムを用いる。このアルゴリズムはタスクセットに対して、複数の初期解から確率的にいくつかの有効なスケジューリング解を求め、その中でグループ毎に最良解を選び、それらの解をそれぞれ個体とし、それより良い解を同様に求めていく。進化的アルゴリズムの一種である。

本ステップではいずれかのコアが 1 個故障した場合のスループットを最大にするマッピングも求める。ここで故障後のマッピング結果は故障前のマッピング結果を考慮せず、仮想プロセッサ (VP) に割当てるものとし残りのコア数でスループットを最大にするマッピングを求める。図 2 に一例を示している。4 個のコアのうち、1 個のコアが故障した場合、残り 3 コアでタスクを実行するスケジューリングを行う。なお、どのコアが故障したとしても、ここでは同じスケジューリング結果になる。

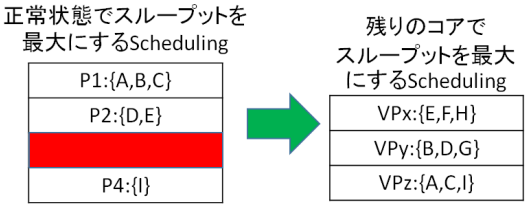

図 2: 故障前後のスケジューリング/マッピング

3.2 マイグレーションコストを考慮したリマッピング

ステップ (2) では、ステップ (1) で求めたスケジューリング結果を用いてコアの故障パターンに対して、マイグレーションコストを最小にするリマッピング結果を求める。

まずコア故障後のスケジューリングを、仮想コアに実際のコアを当てはめた場合のマイグレーションコストを計算する。

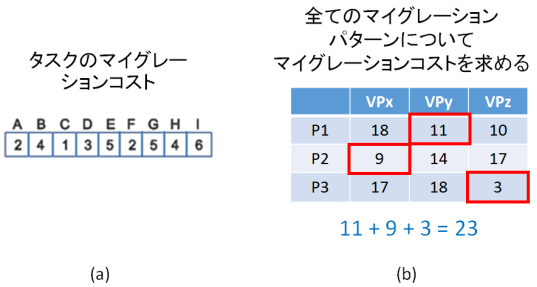

図 3: マイグレーションコストとリマッピングの決定

図 2 の結果を例として説明する、また各タスクのマイグレーションコストはタスク実行時間の 10 %と仮定し、その値が図 3(a) のようになるとする。この場合、例えば、P1 に対し故障後の仮想プロセッサ VPx を割当てるためには、まず元々P1 に割り当てたタスク {A,B,C} をコアから追い出し、新たにタスク {E,F,H} をコアに割り当てる必要がある。この際発生するマイグレーションコストはタスク {A,B,C} を追い出すために発生するコスト 7 と {E,F,H} を新たに割当てることで発生するコスト 11 の和で、18 になる。図 3 の (b) は全パターンのタスクのマイグレーションコストを示している。

このように全ての割当てパターンについてコストを計算し、その中から全体のマイグレーションコストが最も少ない組合せを選択することで、スループットが高く、かつマイグレーションコストの小さなリマッピングを行うことができる。

3.3 エンコーディング

実行時にスケジューリングを行うのは現実的ではないため、全ての故障パターンについて求めたリマッピング結果を予めメモリに保存する必要がある。その際に各故障パターンに対して、タスクのマッピング情報を符号化して保存する。一般的な符号化手法の Unified encoding では、データ量が多くなってしまうという問題点がある。本研究では故障発生時のスループットを最大にするスケジューリング結果と、非故障コアと仮想コアとのマッピング情報を分けて符号化し、メモリに保存することで、データ量を最低限に抑える。

4 評価実験

実験では従来手法 [1] と提案手法において、単一コアの故障をターゲットとし、スループットやマイグレーションコストについて比較を行った。

全タスクの実行時間が同じであると設定した場合の結果を図 4 に示す。提案手法は従来手法と比べスループットでは最大 24 %程度高くなり、マイグレーションコストは従来手法に比べ最大 2 倍になる結果となった。

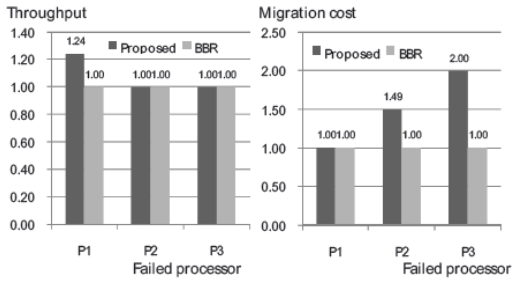

図 4: タスク実行時間が同じであると設定した場合の結果

また、タスクの実行時間が全てランダムであると仮定した場合、図 5 で示しているように提案手法のは従来手法と比べスループットが少なくとも 15 %程度の高くなり、提案手法の有効であることがわかる。一方、マイグレーションコスト面では従来手法と比べ多くのオーバーヘッドが生じるが、このオーバヘッドは許容できる範囲であり、タスクのデッドライン制約を満たしていることもわかった。

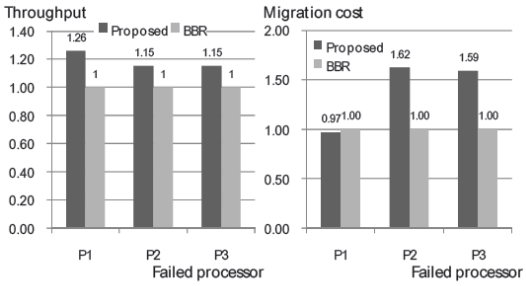

図 5: タスク実行時間をランダムに設定した場合の結果

5 おわりに

本研究では、スループットの低下を最低限に抑えつつ、マイグレーションコストを抑える手法を提案した。従来手法との比較実験では、スループット面では優れており、マイグレーションコストは許容できる範囲に抑えられたため、提案手法の有効性を確認できた。

参考文献

[1] Chengmo Yang and Alex Orailoglu, “Predictable Execution Adaptivity through Embedding Dynamic Reconfigurability into Static MPSoC Schedules,” International Conference on Hardware/Software Codesign and System Synthesis，2007.

[2] Hoeseok Yang and Sonnhoi Ha, “Pipelined Data Parallel Task Mapping/Scheduling Technique for MPSoC,” Proceedings of the Conference on Design, Automation and Test in Europe ， 2009.