2011 年 12 月 12 日 情報システム基盤学合同輪講発表資料

Utilizing Dynamically Coupled Cores to Form a Resilient Chip Multiprocessor

著者： Christopher LaFrieda, Engin .Ipek, José F. Martinez, Rajit Manohar
出典： *37th Int' Conf. on Dependable Systems and Networks (DSN '07) , pp.317 - 326, 2007.*
発表者： 本多・近藤研究室 1153009 齋藤翔太

1 はじめに

半導体プロセスの微細化によるトランジスタ数の増大により、性能向上と消費電力の削減が図れるプロセッサアーキテクチャとして複数コアを 1 チップに搭載するチップマルチプロセッサ (CMP) が主流となっている。しかし、トランジスタ数増大はチップあたりのソフトエラー率の爆発的増加を引き起こす。ソフトエラーはハードエラーと違って一時的なエラーであり、プロセッサの論理回路におけるソフトエラー率は 2011 年には 1,000FIT[1]にもなってしまうと言われている [1]。この他にも永続的なエラーであるハードエラーも生じる可能性があるので、CMP ではこのような状況下でも正常な動作を保ち続ける能力が必要とされている。

この論文では、ハード、ソフトエラーに対処するフォールトトレランス技術である Dynamic Core Coupling (DCC) を提案している。

2 Dual Modular Redundancy

ソフトエラーなどが発生した際にエラーを検出する手法として Dual Modular Redundancy (DMR) がある。DMR は事前に 2 つのコアを組み合わせコアのペアをつくり、スレッドを冗長に実行し、ある間隔 (チェックポイント間隔) ごとスレッドの状態を比較しエラーの検出を行う。

DMR は事前に 2 つのコア間で通信する専用の経路が形成されており、冗長に実行しているコアが片方でも壊れると、そのペアでは DMR 実行できなくなる。

なお、このように事前にコアを組合わせる DMR を形成した CMP を今後 SDMR とする。

3 提案手法

Dynamic Core Coupling (DCC) は実行時でもどの 2 つのコアでも DMR のためのコアのペアとして形成できる。DCC は SDMR と同様にハードエラー、ソフトエラーの両方を検出、またエラーから復帰することが可能である。しかし、SDMR と違って専用の通信経路は不要であり、ペアを組んだコアの片方が壊れても新たに他のコアとペアを組み、再び冗長に実行が可能である。

[1]1FIT は、デバイスを動作させて 10 億時間の期間で一つのエラーが起こることを表す。

DCC では冗長化したスレッド間の通信はすべて CMP の共有メモリ用のシステムバスやネットワークを介して行う。コア同士ペアを組んだときのレイテンシは遠くのコア同士でのやり取りは隣接するコア同士よりも大きくなってしまう可能性がある。また、通信が増える分システムバスのトラフィックの増加が深刻なパフォーマンスの低下に繋がってしまう。システムバスのトラフィックに関してはチェックポイント間隔を少なくとも 3,000 サイクル以上にすることでトラフィックの増加を 10 %以下にすることができることがわかっている。

DCC はチェックポイントでスレッドの同期、データ圧縮および比較、チェックポイントレジスタへチェックポイント時点でのレジスタ値の保存を行う。

DCC は逐次プログラム、並列プログラムのどちらの実行にも対応しているが以降では逐次プログラムの場合のみ解説する。

3.1 同期

チェックポイント間隔分のサイクルが経過するとチェックポイントリクエストがスケジュールされる。また、キャッシュのバッファオーバーフロー、割込み、I/O アクセス、コンテキストスイッチが発生すると、コア間でスレッドの同期およびデータの比較が行われる。そして、新たにチェックポイントが再スケジュールされる。

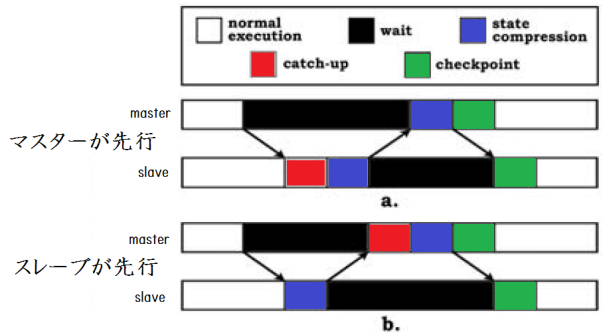

図 1: 同期時のコアの動作

ロード命令などで同時にマスターとスレーブが同じデータにアクセスする場合などではどちらかの命令がもう一方の命令が終わる終了するのを待つ必要がある。図 1 のマスターが命令を先行して実行している場合 (a) とスレーブが命令を先行して実行している場合 (b) について、デー

タの同期を説明する。

(a) ではまず、マスター側からスレーブ側に同期を要求する。スレーブ側はマスターよりも命令の実行が遅れているため、マスターが処理したところまでの命令を実行した後、スレッドの状態を圧縮しマスター側に送信する。マスター側は自分のスレッドの状態を圧縮し比較する。その後、チェックポイントに記録する。

(b) も、マスター側からスレーブ側に同期を要求する。スレーブは命令の実行が先行しているため、すぐにスレッドの状態を圧縮しマスターに送信する。マスターではスレーブ側が実行したところまで命令を実行し、状態を圧縮した後に比較する。その後、チェックポイントに記録する。

3.2 データの圧縮および比較

共有メモリのバンド幅には限界があり、コア間でのスレッドの状態のやりとりはバンド幅に大きな影響を与える可能性がある。そこで、バンド幅への影響を緩和するためにデータの圧縮を行う。

圧縮するデータはストアのデータとアドレス及び、整数レジスタファイルと浮動少数点レジスタファイルである。ストアのデータとアドレスは毎サイクル、レジスタファイルの値はチェックポイント時にそれぞれ CRC 値を計算する。チェックポイント時に行う状態の比較はこの CRC 値を比較することで行う。

3.3 リカバリ

DCC はエラーの検出だけでなく、エラーからの復帰も行うことができる。DCC は DMR 実行をしている時にエラーを検出すると、直前のチェックポイントまで状態を戻し再実行を行う。ソフトエラーであれば、これだけでエラーから復帰が可能である。

DCC では、2 度 DMR 実行でエラーが検出されると、OS にコアの追加を要求し TMR 実行を行う。TMR で実行すると、どのコアで実行した結果が違うのかを判別することができる。そこで、結果の違うコアの使用を停止し、残りの 2 つのコアで DMR 実行を再び開始する。

4 評価

評価には SESC シミュレータを用い SPECCPU2000 ベンチマークプログラムを実行して行った。まず、チェックポイント間隔のサイクル数と実行時間のオーバーヘッドの関係を調べ、次に SDMR と処理速度を比較した。

図 2 はチェックポイント間隔を 1,000、5,000、10,000 サイクルと変えたときの実行時間のオーバーヘッドである。全アプリのオーバーヘッドの平均値は、1,000 サイクルでは 20 %、5,000 サイクルでは 5 %、10,000 サイクルでは 3 %と間隔を長くするとオーバーヘッドが減少することがわかる。よって、実行時間のオーバーヘッドが一番小さい 10,000 サイクルをチェックポイント間隔とすることが望ましいと考えられる。

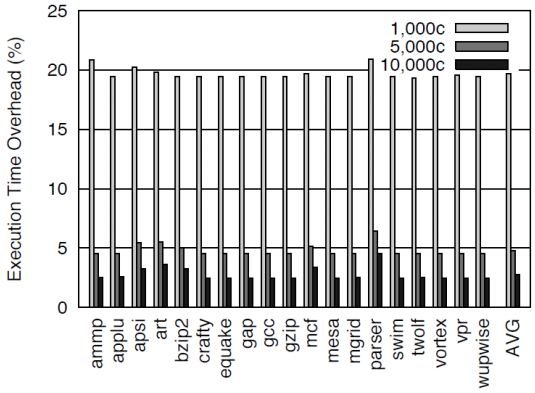

図 2: チェックポイント間隔の違いによる実行時間のオーバーヘッド

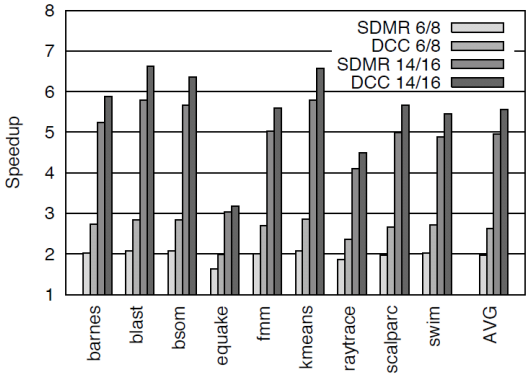

図 3: DCC と SDMR におけるスピードアップの平均

図 3 はオーバーヘッドのない SDMR と DCC を比較している。本評価は並列プログラムを用いて行っている。基準はフォールトトレランスなしの CMP でシングルスレッドを実行した際のパフォーマンスである。それぞれ、コア数が 8 あるいは 16 のときに 2 コアが使用できない状態での処理速度の比較である。SDMR では 2 コアの故障により DMR のペアが 2 つ組めない状態となり、残り 2 つの CPU は利用できない。

DCC はどちらの場合でも SDMR に比べ、DMR のペアが 1 つ多い状態で実行できるためパフォーマンスの向上がみられる。

5 まとめ

CMP において実行中でもコアを組合わせて仮想的に DMR を形成できる DCC を提案した。DCC は DMR を組んでいたコアが片方故障しても新たに他のコアと DMR を組めるため、SDMR と比べ故障時により多くのペアで冗長に実行できる。

参考文献

[1] Premkishore Shivakumar, Michael Kistler, Stephen W. Keckler, Doug Burger, and Lorenzo Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In Intl. Conf. on Dependable Systems and Networks, June 2002.