論文の内容の要旨

論文題目　**Designing NEKO: A Speculative Multithreading Chip Multiprocessor**
（スレッド投機実行チップマルチプロセッサ NEKO の設計と評価）

氏名　**バルリ ニコ デムス**

チップマルチプロセッサ（CMP）は次世代マイクロプロセッサアーキテクチャとして注目され実用化されつつある。CMP では複数のスレッドを並列実行することができ、スレッドレベル並列性の多いサーバ用途においては、非常に高いスループットを発揮できる。しかし、デスクトップ用途においては、同時に稼働されているスレッドの数が普段は少ないため、CMP の資源の利用率があまり良くない。さらに、単一スレッドだけを実行する場合、同量の資源を使ったスーパースカラプロセッサよりも一般的に性能が低い。このような問題で、CMP はデスクトップマイクロプロセッサとしては受け入れられがたい。

上記の問題を解決するためにスレッド投機実行という手法が提案された。スレッド投機実行では、単一スレッドプログラムを複数のスレッドに分割し、投機的に並列実行する。数値計算プログラムによく使われている従来の並列化手法と違い、スレッド投機実行ではスレッドの間で制御依存やデータ依存が許されている。実行結果の正しさを保証するために、ハードウェア及びソフトウェアのサポートが追加される。

本研究は、効率的なスレッド投機実行を行う CMP を設計することを目的とし、NEKO アーキテクチャを提案する。NEKO アーキテクチャはまずコンパイラが単一スレッドプログラムに対してスレッド分割を行う。実行時にはプログラムオーダーに従いスレッドを動的に予測し CMP のプロセッシングユニット（PU）に割り当てる。割り当てられたスレッドはそれぞれ各 PU 上で並列に実行することで、従来の実行方式に比べてより高い性能を目指す。

既存のスレッド投機実行アーキテクチャに比べて、NEKO アーキテクチャには以下の性能向上手法が提案され組み込まれている。

１．Dual-length パスベーススレッド予測手法

この手法はスレッド予測精度を向上させる手法である。我々が行った解析によれば無限サイズの予測テーブル（エイリアスなし）を仮定しパスベース予測でスレッドを予測する場

合、充分長いパス情報を利用すれば、高いヒット率で予測できることがわかった。しかし、予測テーブルサイズを有限にするとパス情報が長くなるほどエイリアスが起りやすくなる。プログラムによってヒット率が悪化するケースが出てきた。そこで、我々は Dual-length パスベース予測手法を提案する。この手法ではパスの長さの長いものと短いもの、２つの予測器を組み合わせハイブリッド構成にする。この構成では、スレッドの数が少なくエイリアスが起りにくいプログラムはパスの長い予測器の高い予測性を活用することができる。また、スレッドの数が多くエイリアスが起りやすいプログラムの場合には、パスの長さが短いものでヒット率悪化を抑制する。さらにこの構成では、テーブルの更新アルゴリズムを工夫することによって、片方のテーブルで予測できるものは他方のテーブルを干渉しないようにすることができ、全体のエイリアスの影響を抑制することができる。

２．スレッドマージとスレッド予測の早期検証手法

手法１がスレッド予測精度を向上させるのに対し、ここで提案する２つの手法は予測がミスした時に発生するペナルティを削減する。一般的な手法ではスレッド予測ミスが発生した場合、そのスレッドの実行を破棄し、新しいスレッドを同じ PU に割り当て実行を再開する。スレッドマージでは、ミスになったスレッドと同じ PU ではなく、その前のスレッドのある PU に新しいスレッドの実行をマージする。これは、新しいスレッドの命令の一部はすでにフェッチされ（場合によっては実行もされ)、この PU のパイプラインにすでに入っているということを利用する。新しいスレッドの実行をこの PU 上にマージすることによって PU のパイプラインに命令をフェッチするオーバヘッドを削減することができる。もう一つの手法、早期検証では予測がヒットかミスかをできるだけ早く検証できるようにコンパイラが支援を行う手法である。コンパイル時には各後続スレッドの制御等価位置を解析し、その位置に後続スレッドのアドレスが実行時にわかるように特別な命令を入れる。プログラムの制御フローがこの命令に至ったら、PU は後続スレッドのアドレスを確定し、CMP のスレッド管理ユニットに通知する。スレッド管理ユニットは予測の検証を行い、予測がミスした場合より早期にスレッドを破棄し新しいスレッドの実行を再開できる。

３．低レイテンシレジスタ通信機構

NEKO アーキテクチャはスレッド間のレジスタ依存を許している。従って、あるスレッドが生成した値を、それに依存する後続スレッドが使用する場合、正しい値が伝わるようにしなければならない。我々はスレッド間のレジスタ依存解析を行い、依存関係の大部分はプログラムオーダで隣接している二つのスレッドに存在していることがわかった。また、通信バンド幅よりも通信レイテンシの方が性能に大きく影響することがわかった。これらの解析結果を踏まえて、NEKO アーキテクチャにはリングトポロジの通信データパスを採用し

た。また、生成したレジスタの値を実際通信するかどうかがまだ確定していない時点でもできるだけ次の PU の近くに送れるようにしている。通信プロトコルには Producer-initiated 型を採用し、通信手続きによるオーバヘッドを最低限にする。また、提案するデータパスはレジスタファイルやリネームマップに追加するポート数が少ない。このように我々は低レイテンシレジスタ通信機構を実現し、高い性能を目指している。

４．更新型・リードブロードキャスト コヒーレンス プロトコル

スレッド投機実行アーキテクチャではスレッドレベルメモリ投機を行うものが一般的である。スレッドレベルメモリ投機では、投機スレッドは前のスレッドの全てのメモリアクセスが解決できなくても、メモリロードを積極的に行う。その後、前のスレッドが同じところにメモリストアをした場合はバイオレーションが発生し、バイオレーションが生じたスレッドを破棄し再実行させる。このような機能を実現するキャッシュ機構は関連研究ですでにいくつか提案されているが、これらは全て無効化型コヒーレンスプロトコルを使っている。しかし、我々の解析では無効化型プロトコルでスレッド投機実行を行うと高い共有ミス(sharing miss)が発生することがわかった。また、スレッド投機実行ではスレッドの実行が分散され、キャッシュの空間的・時間的局所性が低くなり、ミス率が著しく上がることもわかった。これらの問題を解決するために更新型・リードブロードキャスト コヒーレンス プロトコル を提案する。このプロトコルはスレッドがストアしキャッシュを更新した場合、その更新を同じラインのコピーを持つキャッシュにも通信し更新させる。これによって無効化型プロトコルに多発する共有ミスを抑制する。また、リードブロードキャスト機能を使って、あるキャッシュにミスが発生し新しいラインを持ってくる場合、他のキャッシュもそのラインをスヌープし自分のところにも入れる。これによってバンド幅を無駄にすることなくプリフェッチに近い機能を実現し、局所性の低下によるキャッシュミス率の増加も押さえる。

本論文は NEKO アーキテクチャの設計についての解析及び提案手法を述べている。まず既存の問題点を洗いだし定量的に解析をする。その結果に基づいて提案を行い、その提案手法を NEKO アーキテクチャに組み込む。さらに、詳細なシミュレーションを行い提案した手法及び NEKO アーキテクチャ全体を評価する。評価結果から我々が提案した手法がそれぞれの問題に対し効果的であること、また NEKO アーキテクチャを用いることで従来の実行方式より高い性能が得られることを確認し、NEKO アーキテクチャの有効性を示す。