論文の内容の要旨

論文題目　スレッド投機実行チップマルチプロセッサにおけるコンパイル手法の研究

氏名　　田代 大輔

マイクロプロセッサはその登場以来、めざましい進歩と性能向上を遂げ、単一チップ上で複数のスレッドを実行可能とするまでに至っている。マルチスレッド実行環境は並列化されたアプリケーションでは実行資源を有効に活用し高い性能を示すが、並列化されていないアプリケーションでは遊休実行資源を生じ、その性能を発揮できない。

この遊休実行資源を利用して並列化されていない単一スレッドプログラムの実行を高速化する技術としてスレッド投機実行がある。スレッド投機実行はプログラムの実行命令列を複数の「スレッド」に分割し、マルチスレッド実行環境でハードウェア、ソフトウェアによる様々な機構を用いて、分割したスレッドを投機的に並列実行することでスレッドレベル並列性を抽出し、速度向上を図る。

本論文では、SPEC95INT に代表される一般的な、非数値計算アプリケーションをターゲットとし、スレッド投機実行を行うチップマルチプロセッサである NEKO アーキテクチャにおいて、スレッド投機実行を行うためのコンパイル手法について検討、提案、評価を行う。

NEKO はスレッド制御投機、スレッドメモリ投機を行い、スレッド間レジスタ通信を行う機構を備えたチップマルチプロセッサである。NEKO でスレッド投機実行を行うには、コンパイラにより単一スレッドプログラムのスレッドへの分割を行い、スレッド間レジスタ依存を解析してスレッド間レジスタ通信を制御する専用命令を挿入し目的プログラムを生成する必要がある。

本論文では、まずスレッド間レジスタ通信の基本的な解析手順、と目的プログラムの生成手順について述べる。その上で、関連研究も含め、従来のスレッド投機実行で用いられてきたスレッドモデルが、スレッド分割において制約となり、意図しないスレッドへの分割が行われ、粒度の小さいスレッドが多数生成されスレッド投機実行の性能を低下させる要因となることを述べる。

粒度の小さなスレッドは、スレッドの開始、終了に伴うオーバヘッドが相対的に大きくなるため、実行効率が低下する。また、命令ウィンドウに十分な命令を供給することが出来ないため、スレッド投機実行の目的である等価的な命令ウィンドウの拡大を行えない。

この問題は、制御フローグラフのノードを、複数のスレッドで共有することを認めていないスレッドモデルにある。そこで、複数のスレッドでノードを共有することが可能な、「ノード共有スレッドモデル」を提案し、このスレッドモデルを実現するための要件を示す。

ノード共有スレッドモデルを可能とするためには、共有するノードで、複数のスレッドに対し同一の命令列を用いることができることが必要である。NEKO では、実行コードに挿入されるレジスタ通信に関する命令は複数のスレッドに対して同一とすることが出来るため、ノード共有

スレッドモデルを導入することが可能である。

その上で、ノード共有スレッドモデルを前提とし、従来のスレッド分割手法を改良することで、意図しないスレッドの分割を抑制し、粒度の小さいスレッドの生成を削減するスレッド分割手法を提案する。この分割手法は従来のスレッド分割手法と同様に、プログラムの制御構造を階層的に解析し、ループおよび関数をスレッドとしてスレッド分割を行う。その後、従来手法で生成される小さなスレッドを、ノード共有スレッドモデルによって可能となった、他のスレッドとの融合を行うことでスレッド粒度の拡大を図る。

更に、スレッド分割手法を改善するため、関数間解析を用いるスレッド分割を検討する。提案したスレッド分割は関数単位でスレッド分割を行うため、プログラム全体の構造を見据えた分割を行うことができない。そこでプログラム全体の関数の呼出グラフを作成し、リーフ関数から順に解析、スレッド分割を行い、その情報を順次呼び出し元の関数に伝搬させることで、関数単位では利用できなかった情報に基づくスレッド分割を可能とする。その一例として、粒度の小さい関数を親関数のスレッドに取り込むインライン化を提案する。

本論文で提案したスレッド分割手法は、スレッドの粒度に着目し、十分な粒度を確保することを考慮した手法であり、スレッド投機実行の性能を大きく左右するスレッド間のデータ依存を考慮していないという問題がある。一般的なアプリケーションでは、スレッド分割でデータ依存を考慮する余地はあまり大きくないと考え、異なる方法でスレッド間データ依存の問題を解決するアプローチとして、スレッドレベル値予測を行うことを検討し、値予測機構を提案する。

スレッドレベル値予測では、スレッドの開始時点で、先行スレッドから通信されるレジスタの値を予測し、そのレジスタを使用する命令を先行スレッドと同期を取らずに値投機実行を行う。これにより、スレッド間のデータ依存を解消し、速度向上を図る。スレッドレベル値予測では集中型の値予測器を用い、ストライド値予測を行う。ここで、値予測を行うレジスタを決定する際に、コンパイラの支援を行う。コンパイラはスレッド間のレジスタ依存を解析し、値予測による速度向上が期待できるレジスタを静的に決定する。これにより、予測器の予測テーブルサイズを削減し、ハードウェアコストの増大を抑えてスレッドレベル値予測を実現する。

提案したスレッド分割手法、およびスレッドレベル値予測について、SPEC95INT ベンチマークを用いてシミュレーションにより評価を行った。その結果、ノード共有スレッドモデルの導入により、平均スレッド粒度の拡大と、10 命令以下の粒度の小さいスレッドの大幅な削減が達成され、スレッド投機実行のオーバヘッドの削減、命令ウィンドウの等価的拡大により、スレッド投機実行の実行速度向上が可能であることが示された。

また、ノード共有スレッドモデルを前提としたスレッド分割手法として、提案手法と、ループおよび関数のみに着目したスレッド分割手法の比較を行った。また、提案した関数間解析による関数のインライン化の評価を行った。その結果、分割手法によらずスレッド粒度が拡大し、非ノード共有スレッドモデルによるスレッド分割より高い性能を示したが、関数にのみ着目したスレッド分割手法は、プログラム毎に性能のばらつきが大きく、ループに着目したスレッド分割手法では、性能の点で劣っていた。これは、関数、ループのみを対象としてスレッドレベル並列性を

抽出するのではなく、より幅広い領域でスレッドレベル並列性の抽出を図る分割手法が適していることが示された。

スレッドレベル値予測の評価では、一部のプログラムで速度向上が見られたが、一方で全く速度向上の得られないプログラムもあった。また、予測を行うレジスタを指定するコンパイラ支援は、一部のプログラムにおいて、全てのレジスタについて予測を行う場合と同等の性能向上を示した、コンパイラ支援を前提とすることで、値予測器の予測テーブルのサイズを大幅に削減できる。これにより、コンパイラ支援によりハードウェアコストを削減し、スレッドレベル値予測を現実的なハードウェアコストで実現し、速度向上を実現できる可能性を示した。