並列論理型言語の実行に適したマルチスレッド型プロセッサアーキテクチャに関する研究

細井 雅之 (910102)

北陸先端科学技術大学院大学 情報科学研究科

2002 年 2 月 15 日

キーワード: 並列,GHC,JTD, コンテキスト, スレッド, マルチスレッド.

概 要

本論文では並列論理型言語のためのアーキテクチャによるサポート及び並列性を内在するパラダイムに適した並列実行機構を提案する。関数型、論理型、オブジェクト指向型のいずれのパラダイムも逐次的なアーキテクチャで効率良くシミュレートされるが、それらのパラダイムに内在する並列実行可能性は無視されている。更に、各々のパラダイムに関する並列実行には多くの研究があるにもかかわらずそれらのパラダイムに共通に適用できる並列実行機構についてはほとんど関心が払われていない。また、関数型プログラムの実行に適したマルチスレッド型プロセッサ・アーキテクチャが提案されている。そこで本研究では、共通に適用可能な並列実行機構を提案するために、まず第一に、GHC と呼ばれている並列論理型言語に着目する。

1 緒言

本論文では並列論理型言語のためのアーキテクチャによるサポート及び並列性を内在するパラダイムに適した並列実行機構を提案する。かつてプログラミング言語のアーキテクチャによるサポートはアプリケーション固有のサポートの間接的な方法として伝統的に一般的であった。しかし高級言語計算機は少なくとも 20 年前から顧みられなくなった。現在は RISC を中心としたマイクロプロセッサ時代を迎えている。高級言語計算機はソフトウェアの課題をハードウェアで解決するという試みであった。しかし計算機のユーザ評価は高級言語計算機の達成度ではなく総合的な費用効率の良さである。ソフトウェアの課題としては主に次の 2 つがある。1 つは、ソフトウェア開発の方法を工学的に捉えようとするアプローチである。もう 1 つは、プログラミングのパラダイムを根本から見直し、これまでのノイマン型アーキテクチャに基づく命令型のパラダイムから他のパラダイムへ転換を図るものである。この中でも論理型プログラミングは並列処理あるいは並列計算機アーキテクチャの観点から関心が持たれた。しかしながらこれも総合的な費用効率の良さとは一致しない。費用効率の観点において、ハードウェアとしては、より簡単な構成、より少ない資源で実現されるマシンの方が性能で若干下がったとしても優位に立つ。従って高級言語計算機は成立しない。

一般にプログラム言語の処理は言語の表層的な構文の処理、演算実行順序制御、演算実行に分かれる。このうち構文の処理はコンパイラに任せるのが費用効率の点で優れている。一方ハードウェアによる高速化に最も効果があるのは演算実行部である。ハードウェアによる高速化は並列動作による効果である。高級言語処理に関して残された部分は、演算実行の制御である。これに関しては現在、命令パイプラインを有する逐次型アーキテクチャマシンを対象にした命令生成と命令スケジューリングに関心が払われている。関数型、論理型、オブジェクト指向型のいずれのパラダイムも逐次型アーキテクチャで経済的にシミュレート出来るが、これらのパラダイムに内在する並列実行可能性は無視されている。また、各々のパラダイムの並列実行については多くの研究があるが、それらに共通に適用できる並列実行機構についてはほとんど関心が払われていない。

2 論理型言語とその処理方式

本論文では、各々のパラダイムに共通に適用できる並列実行機構を検討するために、その前段階として、1つのパラダイムに着目し、そのパラダイムに内在する並列実行可能性を探求する。着目するパラダイムは論理型とし、特に並列論理型言語 GHC に焦点を充てる。並列論理型言語の効率良い実行を妨げる要因としては主に2つある。1つはユニフィケーションと呼ばれるパターンマッチングの繰返しである。ユニフィケーションを効率良く実行させるにはデータ型を出来る限り早い段階で決定する必要がある。この対策としては1命令で多方向への分岐が可能なタグ付き分岐命令 JTD が提案されている。もう1つは頻繁に生じる実行コンテキストの切替えである。並列論理型言語用マシンのこれまでの研究としては、並列マシンの性能の基本となる要素プロセッサ自体の高速化よりもむしろネットワークの構成やプロセッサ間通信の方式など、「並列」に重点をおいたものであった。一方で、並列論理型言語特有のコンテキスト・スイッチが予想以上に高頻度であるとともに、この処理の「重さ」が性能にかなりの悪影響を及ぼしていることが指摘されている。これはデータ依存関係による小さな粒度のプロセス（スレッド）間の同期を取るために発生する。従って、「軽い」コンテキスト・スイッチを実現する機構を備えた要素プロセッサのアーキテクチャの検討が必要である考えられる。

3 スレッドスケジューリング

我々は提案手法を証明するためにシミュレーションによる実験を行なう。シミュレーションプログラムは GHC をハンドコンパイルしたコードとする。従って、まず第一に、我々はマルチスレッドプログラム用のハンドコンパイルしたコードに対して、プログラム中のどのスレッドがある時点に実行されるかを決定する方法を確定しなければならない。そのために、この章では、マルチスレッドプログラミングが可能な典型的な言語として Java に着目し、Java 仮想マシンが、ある時点で実行されるべきスレッドをどのように決定す

るかという点を確認する。

4　アーキテクチャの設計

関数型プログラムの実行に適したマルチスレッド型アーキテクチャが提案されている。このプロセッサは各スレッドごとにハードウェア資源を有し、次に実行されるスレッドを取り出すスレッド選択ユニットを持つ。従って、データハザード、分岐ハザード、構造ハザードは避けられている。このプロセッサはキャッシュミスを扱う制御ユニットによりストールの生じないパイプラインとなっている。このパイプラインはスーパーパイプラインで構成されており、パイプラインステージと同数のスレッドを処理でき、このスレッドはプログラムカウンタ、レジスタファイル、及び各種レジスタを有する。このスレッドの切替え方針はサイクル毎となっている。この方針はクロック毎の切替えを認め、命令がロードされるかどうかとは無関係である。つまり、クロック毎という条件で異なるスレッドからのコンテキストを切替える。次々と続いていくスレッドは独立したものであり、パイプラインの実行に有効である。コンテキストの切替えはパイプラインの依存性を隠すことができ、コンテキストスイッチのコストを減らす。それゆえに、切替えによって性能の優位性が得られる。このプロセッサの主な特徴は 4 つある。第一に、このプロセッサはステージ数の数のスレッドで満たされるようにするための十分なスレッドがある限り、十分に利用される。そのため、スーパーパイプラインのステージ数を満たす十分なスレッドが必要なだけである。一度スーパーパイプラインのステージが満たされれば、このプロセッサはピーク時の性能で実行され続け、追加的にスレッドを加えても高速化されない。第二に、このプロセッサは単一のプロセッサ環境で多くのスレッドをサポートする。第三に、このアーキテクチャモデルはパイプライン、及びクロック毎の多くの命令を発行する専用のユニットで構成される。第四に、このプロセッサはパイプラインステージと同数のスレッドを発行することによりパイプラインハザードを隠蔽できる。一方で、このプロセッサは 17 段の命令パイプラインを有しているが、関数型プログラムを実行させることによる実際の効果を証明するには至らなかった。

そこで、いままで見てきたように、データ依存による細粒度のプロセス（スレッド）間の同期による GHC の重いコンテキストを克服し、さらに、並列実行可能性を最大限に生かすために、著者らは軽いコンテキストの切替えが可能なマルチスレッドプロセッサに着目し、並列論理型言語の実行に適したマルチスレッドパイプラインプロセッサを提案する。このアーキテクチャは並列論理型言語を効率良く実行させることが可能であり、並列性を内在する各々の言語に有益であり、並列実行可能性を最大限生かすことができる。この提案を証明するために我々は少なくとも 4 つのことを順に行なう。最初、単純な構成のマルチスレッドパイプラインプロセッサを設計する。次に、実際に並列論理型言語の実行をシミュレートする。さらに、並列論理型言語の並列実行可能性を最大限生かすことにより性能を向上させる。最後に並列性を内在する各々の言語に適用可能な共通の並列実

行機構を示す。我々はプロセッサを設計し、シミュレートするために、NTT の研究所で独自に開発されたハードウェア設計システムである高位論理合成システム PARTHENON を用いる。CPU は、小さな手順の連続という形で個々の命令を実行する。この手順の連続は、フェッチ-デコード- 実行:fetch-decode-execute サイクルと呼ばれることが多い。これは、すべてのコンピュータの処理の中核部分である。従って、最も基本的な命令サイクルは 3 つの段階:fetch,decode,execute からなる。これらの命令の段階は命令パイプラインによって実行される。より単純なプロセッサを設計し、シミュレーションを効率良くするために、我々は最も基本的なパイプライン構成として 3 段命令パイプラインを適用する。さらに、並列論理型言語のユニフィケーションを効率良く処理するためには上で見てきたように JTD 命令が必要である。従って、設計するプロセッサとしては、2 種類のプロセッサを設計し、提案したアーキテクチャが有効であることを示すために基礎的な評価を行なう。作製する 2 種類のうち、1 つは、JTD 命令を有する 3 段命令パイプラインで、もう 1 つは JTD 命令を有する 3 段命令パイプラインのマルチスレッドパイプラインとする。

5　動作シミュレーション

我々は提案したアーキテクチャが並列論理型言語の実行及び並列性を内在する各々の言語に有効であることを示すためにハンドコンパイルしたコードによるシミュレーションを 4 つ行なう。1 つは、タグ付き分岐命令のある場合とない場合のテストプログラムによるプロセッサの性能比較、もう 1 つはコードのスレッド数を増加させてスレッド数増加によるプロセッサの性能の飽和、その次は、同期処理の隠蔽効果を提示するための同期処理を伴うテストプログラムの実行、最後は、並列処理が可能で特定の言語に依存しないテストプログラムによる性能比較を行なう。この特定の言語に依存しないプログラムとしては行列演算を用いる。

6　考察

性能評価では以下のことが示されている。JTD 命令は約 70%のクロックサイクル数削減が可能であり、マルチスレッドアーキテクチャは並列論理型言語の実行、JTD 命令を用いた並列論理型言語の実行、及び並列性を内在するプログラムの実行に高性能を提供でき、CPI 値を約 1.0 にできる。その結果として、我々は 2 つのことを考察した。第一に、JTD 命令を有するマルチスレッドパイプラインプロセッサは並列論理型言語の並列実行可能性を最大限に利用することにより実行性能を大幅に向上させることが可能である。第二に、このプロセッサは並列性を内在する各々の言語に対して大いに有益な共通の並列実行機構を供給する。

7　結言

2つのことを結論とする。第一に、JTD 命令を有するマルチスレッドパイプラインプロセッサは並列論理型言語の並列実行可能性を最大限に利用することにより実行性能を大幅に向上させることが可能である。第二に、このプロセッサは並列性を内在する各々の言語に対して大いに有益な共通の並列実行機構を供給する。