B-22

QJava バイトコードの並列性抽出に関する研究
A study of parallelism extraction for QJava Byte-code

鳥居　衛†　　曽和　将容†
Mamoru Torii　Masahiro Sowa

一. はじめに

式評価の中間結果を LIFO (Last In First Out) メモリに貯めえるスタックマシンと、式評価の中間結果を FIFO (First In First Out) メモリに貯めえるキューマシンは２つの計算機アーキテクチャがあります。並列実行可能なキューマシンを“並列キューマシン”と呼ぶ。並列マシンの性能を引き出すには、データの依存関係の解析を行って、並列実行可能な命令列を出力する並列化コンパイラが必要である。本研究は、並列キューマシン用の命令コードを生成するコンパイラに関する研究である。

また可搬性のある言語として Java 言語が注目を集めている。Java 言語は安全性があり、Java バイトコードのファイルサイズはCやC++などの実行可能ファイルサイズより極めて小さい。しかし、JVM (Java 仮想マシン) を用いて実行するために実行速度が遅くなるという欠点がある。そこで、我々は Java の実行速度向上を図るために、キューマシン上の Java 並列度抽出について考案しています。

二. 概要

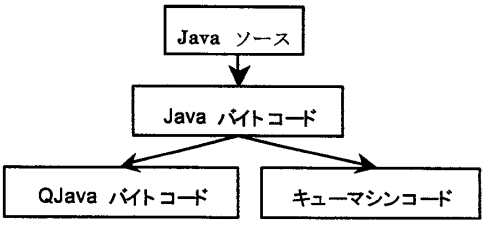

図１ QJava コードとキューマシンコードの関係図

本研究は、図１のように Java バイトコード (Java クラスファイル) から並列性を抽出して QJava バイトコードまたはキューマシンコードを生成する研究である。また、Java ソースから Java バイトコードを生成するのは Sun Javac を使用する。

生成した QJava バイトコードは同研究室の QVM (QJava バーチャルマシン) で実行される。キューマシンコードは同研究室のキューマシン命令セットに従って、キュープロセス上で実行される。

三. 構文木の生成

スタック演算モデルは構文木の深さ優先で命令を抽出し、演算する。それに対して、キュー演算モデルは構文木の幅優先で命令を抽出し、演算する。元々Java バイトコードはスタックマシン向けのコードだから、Java バイトコードから構文木を生成すれば簡単にキュー命令を生成することができる。

構文木を生成する手順：

１、命令の pop 数と push 数を計算

２、構文の生成単位の決定

スタックにデータを入れ始めるから、中のデータがなくなるまで、一つの構文木として生成する

３、命令の降順で構文木を生成

４、push 待ちスタックの作成

push 待ちスタック － push 命令を待つ pop 命令が保存されるスタックである。

５、push 命令を遭遇するたびに、構文木を生成

６、ノード位置の決定

（１）push 数=1 の場合、子ノードは、左から右へ順番に配置

（２）push 数>1 の場合、親ノードは、左から右へ順番に配置

７、同じノードの結合

生成した構文木につき、同じノードが存在した

† 電気通信大学大学院情報システム学研究科
情報ネットワーク学専攻
The Graduate School of Information Systems,
University of Electro-Communications

場合結合する。

今回提案した構文木の生成方法は、すべてのJavaバイトコードを対処となる。

四. 最適化

スタック演算モデルは構文木の移動を行っても命令コードの実行順序が変化することがない。キュー計算モデルでは、構文木の移動を行うと命令コードの実行順序が変化する。

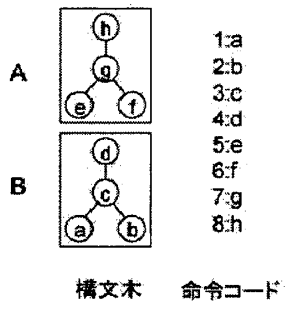

図２ 移動前の構文木

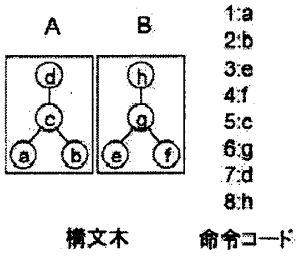

図３ 移動後の構文木

構文木を移動することによって生じる実行命令順序の変化が図２と図３で説明する。図３のように命令 **a,b** の直後に他の命令があり、命令 **a,b** の実行終了から命令 **c** の実行開始までに時間的余裕が生じる場合は、命令 **c** の実行時間を、他の命令の実行時間で隠蔽することが可能になる。

隣接する構文木でノード間の依存関係を調査し、構文木がどれだけ移動できるのを決める。

四. おわりに

本研究において、**Java** バイトコードから **Qjava** バイトコードまたはキューマシンコードに変換することを試み、その手法を提案した。

これにより **Java** プログラムは **QVM** またはキューマシン上でより速く実行することができる。

今後は構文木から更にキューモデルに適する最適化方法を見つけ出し、より並列度の高いコードを生成する予定である。

参考文献

１、前田敦司、中西正和、“新しい計算モデルキューマシンとその並列関数型言語への応用”情報処理学会論文誌

２、中田育男 “コンパイラの構成と最適化”朝倉書店(1999)

３、MICHAEL WOLFE “HIGH PERFORMANCE COMPILERS FOR PARALLEL COMPUTING” Addison-Wesley Publishing Company (1996)

４、川田宗太郎、曽和将容、“並列キューマシンの設計とシミュレーションによる性能評価”平成１１年卒業論文

５、田代浩一、曽和将容、“QJava コンパイラに関する研究”平成１４年修士論文

６、ティム、リンドホルム, “Java 仮想マシン仕様第２版”, ピアソン・エデュケーション 2001