javarへのループ自動並列化機能の実装

神原　公仁　　岩井　啓輔　　黒川　恭一

防衛大学校情報工学科

１．はじめに

Java は，近年並列処理分野においてもさまざまな研究が行われており，プラットホーム非依存性から PC クラスタやワークステーションクラスタなどの分散計算機環境においても盛んに行われている．それらの中には，通常の Java アプリ－ケーションの透過的な並列実行環境を提供するものもある[1]．分散計算機環境で Java の並列処理を行う方法の１つとして，マルチスレッドを利用するものがあるが，その前提として対象となるアプリケーションやプログラムがマルチスレッドプログラムである場合が多い[2,3]．そのため，それらの並列処理環境を使用するためには，処理したいプログラムをユーザ自身でマルチスレッドプログラムに書き換えなければならず、現状ではツールもあまり整備されていない．

本研究では，このような並列処理環境を利用するために，シーケンシャルなユーザプログラムから適切なマルチスレッドプログラムを自動生成するコンパイラの開発を目的としている．その開発方法としては，ユーザの注釈記述（annotation）によりマルチスレッドプログラムを生成する javar[4]を改良し，自動並列化機能を付加してゆく方針である．このうち本稿では，javar へのループ自動並列化機能の付加について，実装の現状を報告する．

２．javar の改良

javar は，HP-Java Project[4]のひとつとして開発されたものであり，シーケンシャルな Java プログラムをマルチスレッドプログラムに変換するマルチスレッドプリプロセッサである．そのプロトタイプは Web 上で入手でき[5]，研究目的であれば自由に使用できる．図１に示す通り，javar により生成されたマルチスレッドプログラムは，標準の Java コンパイラによりクラスファイルを生成し，どのプラットホームでも標準の JavaVM で実行可能である．

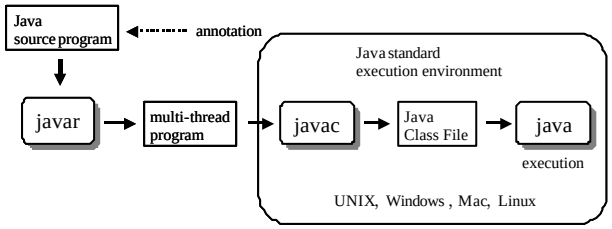

図１　javar

javar がプログラムのマルチスレッド化を行うためには，対象となるループや多分岐再帰メソッドに，ユーザが適切な注釈記述を行う必要があるが，これはマルチスレッド化したい部分に対して，ユーザによるデータ依存解析が行われていることを前提としている．例えばループの場合では，DOALL 型または DOACROSS 型であるのか，そして依存関係があればその配列の添字部分のパラメータについての注釈記述を，ユーザが適切に行わなければならない．

これに対して今回の実装では，対象となるループについて，ユーザが行わなければならなかった配列要素のデータ依存解析と適切な注釈記述をコンパイラが自動的に行うように javar を改良し，適切なパラメータを javar に与えてマルチスレッドプログラムへ変換させる機構を加えた．

３．データ依存解析手法

ループ内の配列要素に対するデータ依存解析を行うには，静的，あるいは実行結果に基づいた動的なデータ依存解析方法などが提案されている．その中で静的なデータ依存解析手法のひとつとして GCD（Greatest Common Devisors）テスト[6]がある．このテストでは，ループ内で使用される配列要素のデータ依存関係を解析できる．単純な Single Loop を用いて，以下に GCD テストの具体例を示す．

```
do I = 0,20
   A[6*I+3] = .....          : S
   ..... .. = A[3*I-2]       : T
end do
```

このようなループ内のステートメント S，T 間にデータ依存があるかを調べるには、配列 A において、以下の方程式(1)が解を持つかを調べることと同じことになる．

$$6*I_1 + 3 = 3*I_2 - 2$$
$$6*I_1 - 3*I_2 = -5 \qquad (1)$$
$$(0 \le I_1 \le 20,\ 0 \le I_2 \le 20)$$

一般的な表現では(2)式で表すことができる．ただし a，b，c は整数である．

$$a*I_1 + b*I_2 = c \qquad (2)$$

この方程式が解を持つ条件は，a とｂの最大公約数 d（=gcd(a，b)）が，c で割り切れる場合のみである．この例では，係数である 6 と 3 の最大公約数 3 が，-5 を割り切れないので解を持たない．つまり２つのステートメントの間にはデータ依存関係がないことがわかる．そのためこのループは，DOALL 型で実行できる．しかしこのような GCD テストでは，変数 I の変域が全く考慮されていないという問題点がある．

もうひとつのテストとして Banerjee テストがある．Banerjee テストは，中間値の定理に基づいて(2)式が変数 I の変域で解を持つかどうかを判定する．中間値の定理では，閉区間[X_1, X_2]において関数 f(x)が連続だとすると，f(X_1)と f(X_2)の間の任意の値 f(Y)=c に対して Y∈[X_1, X_2]となる Y が少なくとも一つは存在するとしている．つまり変数 I の変域での(2)式の左辺の最大値と最小値をそれぞれ計算し，右辺の値 c がその範囲にあれば，(2)式は解を持つことになる．GCD テストで例示したシングルループの場合，

$$\min = 6*0 - 3*20 = -60$$
$$\max = 6*20 - 3*0 = 120$$

となり，-60≦-5≦120 であるから，この例では解を持つことになるが，整数解とならないので，データ依存関係はない．このように，上述した２つのテストは，一般的にデータ依存関係を解析するために相補的に用いられている．そこで今回の実装では，この２つのテストを実装することにした．

４．ループ自動並列化機能の実装と性能評価

４．１　本コンパイラの構成

javar では，YACC 互換構文解析器生成ツールである bison によって，プログラムの構文解析を行い，構文木を生成する．今回の実装では，対象となるループに対して GCD テスト及び Banerjee テストを行い，ループにデータ依存がないならば適切なパラメータを生成し，その結果を受けて javar によってマルチスレッドプログラムを生成するという，ループ自動並列化機能を実装することとした．

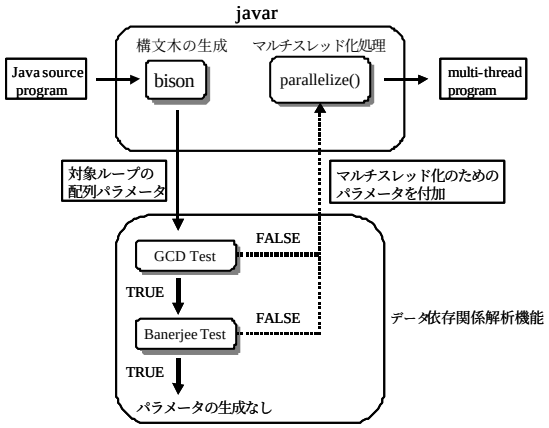

図２　ループ自動並列化機能

このループ自動並列化機能では，図２に示すように bison によって生成された構文木から，対象となるループブロック中に使用される配列要素のパラメータを抽出し，配列要素テーブルを作成する．この配列要素テーブルには，配列変数名，配列添字式が格納されている．そしてループ中のステートメントの左辺に使用される配列要素が，他のステートメントの右辺で計算処理に使用される場合に，GCD テストを適用する．GCD テストにより，データ依存関係がないもの（FALSE）は DOALL 型としてマルチスレッド化できる．しかし，データ依存関係があるもの（TRUE）は，次の段階として Banerjee テストを行い，変数 I の変域に基づいたテストを行う．Banerjee テストにより，データ依存関係がないと判定されれば，これも DOALL 型としてマルチスレッド化できる．この２つのテストにより，FALSE と判定されたループに対して適切なパラメータを生成し，マルチスレッド化処理を行う．

このようにして実装したループ自動並列化機能では，単純なシングルループとステートメント内の配列添字式が一次式のものに限定されているが，その配列要素のデータ依存関係を解析できる．

４．２　実装と性能評価

開発は，IBM/RS6000 44P MODEL 270， AIX V4.3.3，JDK 1.1.8 という環境で行った．

この改良 javar のループ自動並列化機能の性能評価としてベンチマークプログラムである Java 版 LINPACK[7]を用いて実験を行った．JAVA 版 LINPACK では，単純なシングルループが１４，多重ループが３，多次元配列のあるループが３ある．今回の実験では，その内単純なシングルループのみを対象とし，配列添字式で繰返し変数 I だけが参照配列に影響するとした．この場合，これらのループはすべてデータ依存関係がないものと判定できる．この実験において，改良 javar が，単純ループすべてについてデータ依存関係の解析を行ったところ，すべての単純ループが並列化可能であるとの判定を出した．このことから，改良 javar のデータ依存関係解析機能が，正しく動作していることがわかる．

５　おわりに

javar にループ自動並列化機能を付加するためのデータ依存解析手法として，GCD テストと Banerjee テストを適用した．この実装自体が研究の足がかりであるため，現段階では，自動並列化可能なループは単純なシングルループに限定している．多重ループに適用するためには，配列添字中のそれぞれの変域変数に対して２つのテストを適用することで対応できると考えている．また多次元配列についても各次元ごとに２つのテストを適用することで，データ依存関係の解析は可能である．さらにループ内の繰返し変数以外の変数を配列添字式に含む場合についてのデータ依存関係の解析や，データ依存関係のある DOACROSS 型の場合においてフロー依存関係を解析したり，適切なパラメータを生成したりすることには，まだ対応できていないため，今後の課題である．

参考文献

[1] Y.Aridor, M.Factor and A.Teperman: “ cJVM: a Single System Image of a JVM on a Cluster“, 1999 International Conference on Parallel Processing, pp.4-11.
[2] P.Launay, and J.L.Pazat : “ A Framework for Parallel Programming in Java ”, IRISA, France, Technical Report 1154. December 1997.
[3] M.Philippsen and M.Zenger : “JavaParty: Transparent Remote Objects in Java”, Concurrency: Practice and Experience, 9, 11, pp. 1125-1242, 1997.
[4] A.Bik and D.Gannon : “Automatically Exploiting Implicit Parallelism in Java”, Concurrency: Practice and Experience, 9, 6, pp. 579-619, June 1997.
[5] http://www.extreme.indiana.edu/~ajcbik/JAVAR/
[6] 笠原 博徳 :“並列処理技術”, コロナ社, pp. 118-121, 1991
[7] http://www.netlib.org/benchmark/linpackjava/