2H-01

Java仮想機械上で動作する分散KL1言語処理系の実装

五十嵐宏† 上田和紀‡
† 早稲田大学大学院 理工学研究科 ‡ 早稲田大学 理工学部

1 はじめに

並行論理型言語は複雑なプロトコルをもつプロセス間通信を簡潔に記述でき、また静的解析技術により通信プロトコルの一貫性を保証することもできるので、分散処理記述に適していると考えられる。本研究では並行論理型言語KL1[2]を分散処理記述へ適用するための言語拡張を行ない、これを実現する処理系をJava環境上に実装した。

既存のKL1分散処理系は異機種混在(heterogeneous)環境および広域分散環境を考慮していないが、本研究ではそのような環境も考慮した処理系の作成を目指している。

2 分散処理記述のための拡張

従来のKL1分散処理系にもノードの概念はあったが、これらは密結合した並列計算機を対象としたもので、実行ノードの指定を整数IDで行うものであった。KL1/JではTCP/IPネットワークを前提に、IPアドレスとTCPポート番号を組み合わせたノードIDでノードの識別を行う。

2.1 ゴールの遠隔実行指定

プログラミング例を以下に示す。

```
remote_example(N) :- true |
  mklist(0,N,L),
  append(L,[d,123],X)
    @node("kl1://host3:3001"),
  print(X).
```

このプログラムは長さNのリストLを作り、リモートノードhost3:3001でリスト[d,123]と連結し、もとのノードの標準出力に表示する。

また、次々とノードを渡り歩くようなプロセスも記述できる。

```
agent([Next|Plan],State,Report) :- true |
    do_something(State,NState),
    current_node(Node),
    Report <= visited(Node).
    agent(Plan,NState,Report)@node(Next).
:- agent(["kl1://host1:3000",
          "kl1://host2:3005",
          "kl1://host3:3001"],
         [],Report), print(Report).
```

Implementation of Distributed KL1 on Java Virtual Machine
†Hiroshi IGARASHI, ‡Kazunori UEDA
†Graduate School of Science and Engineering, Waseda University
‡School of Science and Engineering, Waseda University

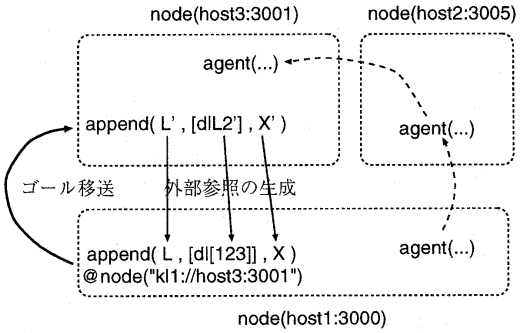

図 1: ゴールの遠隔実行

2.2 ストリームのネーミングサービス

前節の遠隔実行だけでは、異なる位置・時刻で別々に生成したプロセス群が通信を行うことができない。ストリームとして用いる変数を異なるプロセス間で共有する必要がある。

このため、ストリームのネーミングサービスを設け、名前をもとにしてプロセス間通信を始めるとっかかりとなるストリームを得られるようにする。1つのストリーム通信が確立すれば、このストリームを通して更に必要となるストリーム群を次々に張ることができる。

3 分散拡張のための技術

識別子の移送 記号アトムや述語記述子など識別子は、(1) 高速に比較できる (2) 効率よく記憶できる、という性質が要求されるので、通常処理系内部では整数値に直して扱う。本来の識別子 (記号アトムなら綴り、述語記述子なら述語名・引数個数) と整数値表現との対応を管理する表 (識別子表) を保持していなくてはならない。ノード間通信で生じる識別子の移送を実現するためにはこの識別子表をノード間で共有する必要がある。

論理変数の共有　論理変数を他ノードに移送するとき、もとのノードを指す外部参照を生成し、論理変数の共有を実現する。

4　Javaによる分散処理系実装

4.1　処理系の構成

本処理系 KL1/J は、次のような要素から構成されている。

ランタイムシステム・ライブラリ　KL1 言語の基本的機能や分散実行のための拡張を実装したものである。基本的機能を実現する実行核と分散拡張部分からなる。すべて Java 言語で記述されており、Java コンパイラで Java Byte Code にコンパイルし、Java Virtual Machine 上で動作させる。

コンパイラ　KLIC[3] の KL1 to C コンパイラをベースに作成した。KL1 プログラムをモジュール単位で、Java の `Module` クラス (のサブクラス) にコンパイルする。

ユーザプログラム　KL1 言語で記述する。最終的に Java Byte Code にまでコンパイルし、ランタイムシステムとともに動作させる。KL1 のモジュールがコンパイル単位であり、またノード間移送の単位でもある。

4.2　実行核の実装

実行核は、ランタイムシステムのうちノード内の言語機能を実現する部分である。基本的構造は既存処理系と同じである。KL1 プログラム中で扱える KL1 項は、すべて Java オブジェクトで表現している。

4.3　分散処理部分の実装

ノード間接続は 3 つの階層からなる。TCP/IP Layer は信頼できる双方向バイトストリームを提供する。Java 言語の Socket クラスを利用している。Object Serialization Layer は Java オブジェクトとバイト列との相互変換を行う。Java 言語の提供する直列化機能を拡張して実現している。Inter-node Communication Layer は言語のセマンティクスを実現するためのノード間メッセージを送受信する。

4.4　識別子の移送

Object Serialization Layer に識別子表を実現するための処理 (記号アトム ↔ 整数 ID の相互変換) を加えた。

4.5　論理変数の共有

Object Serialization Layer で論理変数の出入りを監視し、適切に論理変数と外部参照を置換するようにした。参照の解決は Inter-node Communication Layer で行う。また、論理変数と外部参照に共通のインタフェイスをもたせ、実行核から統一的に扱えるようにした。

4.6　柔軟なランタイムシステム

Java 言語のリフレクション機能を用いて、ランタイムシステムの一部を動的に交換できるようにした。分散環境の状況に応じたデータの積極転送・遅延転送の切替やコードモジュールのダウンロードポリシーの変更を行なうことができる。

5　評価

位置透過性については、ノード間における論理変数の共有を実現しており、ゴールをリモートノードで実行してもプログラムの意味は変わらないようにすることができた。

また論理変数の間接輸出が起らないようにし、動的なネットワーク環境においてゴールやそれに含まれる論理変数を仲介したノードが消滅しても、プログラムの実行が続けられるようにすることができた。

データやプログラムの表現は、Java 言語に基いて定義することにより、アーキテクチャ中立にすることができた。これにより処理系およびコンパイルされたユーザプログラムのポータビリティを向上し、処理系およびユーザプログラムのネットワークを介した移送を可能した。

5.1　今後の課題

静的解析情報を利用できるようコンパイラおよびランタイムシステムを拡張し、メッセージ指向スケジューリング [4] など、通信形態に応じて最適なノード間通信方法を選択させたい。

またコンパイル済みモジュールに静的解析情報をもたせ、ダウンロード時にモジュール間の整合性検査を行なうことも考えている。

参考文献

[1] 五十嵐 宏, 上田和紀. 分散 KL1 言語処理系の設計と実装, 並列処理シンポジウム *JSPP '99*, pp. 207, 1999

[2] Ueda, K. and Chikayama, T. Design of the Kernel Language for the Parallel Inference Machine. *Comput. J.*, Vol. 33, No. 6 (1990), pp. 494–500.

[3] Chikayama, T., Fujise, T. and Sekita, D., A Portable and Efficient Implementation of KL1. In *Proc. PLILP'94*, LNCS 844, Springer, 1994, pp. 25–39.

[4] Ueda, K. and Morita, M., Moded Flat GHC and Its Message-Oriented Implementation Technique. *New Gen. Comput.*, Vol. 13, No. 1 (1994), pp. 3–43.