計算機アーキテクチャ　147－12
ハイパフォーマンス　89－12
コンピューティング
（2002．3．7）

Javaによる並列計算ソフトウェアの開発と評価

坂口聡 †　小畑正貴 ††

本稿ではJavaによって作成されたPCクラスタ・ソフトウェアの開発とその簡単な評価について述べる。本PCクラスタ・ソフトウェアはJava仮想マシンで実行可能である。したがって、本PCクラスタ・ソフトウェアはJava仮想マシンが利用できるすべてのコンピュータで利用することが可能で、他のPCクラスタと比較すると導入が安全で容易である。本PCクラスタの新ヴァージョンは携帯電話をサポートしている。現在、本クラスタの携帯電話拡張機能はNTT DoCoMoの503iシリーズのみで動作する。本クラスタの旧ヴァージョンはMPICHと比較すると0.2〜0.7倍の性能だったが、新ヴァージョンでは0.3〜1.2倍の性能を発揮した。

Implementation and Evaluation of PC cluster software written by Java

Akira SAKAGUCHI † and Masaki KOHATA ††

In this paper, we present an implementation and a simple evaluation of the PC cluster software written in Java alone. This PC cluster software needs the Java Virtual Machine. So that, this PC cluster software is possible to run under all computers and an installation is safer and easyer than another PC clusters. This PC cluster of new version supports cellular phones. Now, this PC cluster supports only NTT DoCoMo 503i series of cellular phones. The performance of the PC cluster with this software of the old version shows 0.2 to 0.7 times as much as the MPICH. The new version shows 0.3 to 1.2 times.

1. はじめに

新しくPCクラスタを導入する場合、一般的には同じ構成の計算機を複数台用意してPCクラスタを構築する。同じ構成の計算機を複数台利用することによって、ある程度PCクラスタの性能を予測することができる。学習用や研究用として、このような構成のPCクラスタ専用機を導入する場合は問題ない。しかし、並列計算を常に行う環境でなければ、新たに導入したPCクラスタ専用の計算機を有効利用できない状態となってしまう。常にPCクラスタを必要としない環境下、同じ構成にこだわる必要がない環境下で、既に導入されている計算機をPCクラスタとして利用したい場合も多いと思われる。

しかし、導入されている計算機のOSやその設定がPCクラスタ向きではなかったり、導入する予定のPCクラスタが対応していなかったりする場合、既存の計算機に新たにOSをインストールし、その設定する必要がある等、導入は困難となる。複数のOSを一台の計算機に共存させることは可能だが、OSの導入時に既存のデータを誤って破壊してしまう危険性もある。アーキテクチャの異なる計算機同士をPCクラスタとして利用する場合、その実現はいっそう手間がかかるものとなる。新規に購入してきた複数台の計算機を利用する場合でも、OSのインストール以外に専門知識の必要な設定を行う必要がある等、安易にPCクラスタの導入が行えるわけではない。

本論文では安全かつ手軽に既存の計算機を利用した異機種混合のPCクラスタを構築するためのソフトウェアの実装と、そのソフトウェアを利用し並列計算を行った簡単な評価について述べる。新規に複数台の計算機を導入する場合でも手軽に利用できる実装を目指した。

本PCクラスタ・ソフトウェア(以下、本ソフトウェア)の作成と実行にはJavaを利用している。既に同様の研究[1)2)]がなされているが、導入や設定に専門的な知識が必要、システムとして大げさ、少なからずアーキテクチャに依存する等、安全で手軽な導入とは言い難い。本ソフトウェアは関連ファイルをコピーして、Java仮想マシンをインストールするだけで稼働させ

† 岡山理科大学工学部
Okayama University of Science, Faculty of Engineering
†† 倉敷芸術科学大学工学部
Kurashiki University of Science and the Arts

ることができる。それ以外に必要なファイルはユーザプログラムとユーザプログラムから利用するデータ、使用する計算機のリストだけである。将来的には計算機リストも必要ない構成となる予定である。

他の Java を利用した PC クラスタがそうであるように、本ソフトウェアでも使用する計算機上に並列計算のタスクを実行するためのサーバを実行しておくことにより、アーキテクチャに依存することなく並列計算を行うことが可能となっている。同様の研究において実現されている計算機が並列計算の途中で停止した場合からの復帰機能のような様々な高度な機能は、本クラスタにおいてはまったく用意されていない。本クラスタにおいて用意されていない様々な機能を使用したい場合、ユーザプログラムでその都度実装するか、本ソフトウェアに手を加えることで実現は可能である。

更に本ソフトウェアは通常の PC クラスタとしての機能以外に、携帯情報端末、特に携帯電話での並列計算機能も未完成ながら搭載した。今後、高機能化・高速化し、高度な演算が可能な携帯情報端末の増加が予想されるが、携帯電話拡張機能はその空き時間を有効に利用するための機能である。将来的には携帯電話以外の携帯情報端末もサポートする。

本ソフトウェアでは PC クラスタが注目されている昨今において、多くの PC クラスタのように演算速度の向上や通信速度の向上や高機能化だけを目指すのではなく、他の PC クラスタと比べて導入がきわめて容易である PC クラスタの開発を第一の主眼としている。

近年、低価格で高性能な計算機が手にはいるようになり、PC クラスタ専用機でなくてもある程度高速な演算を行えるようになっている。本ソフトウェアを用いることにより、必要なときに、既存の低価格で高性能な計算機を容易に PC クラスタとして転用することによって、さらに高速な演算を行うことができる環境を一時的に構築することができるメリットは大きいと思われる。

2. Java による PC クラスタソフトウェア

本クラスタは分散メモリ型の PC クラスタを構成する。本ソフトウェアはいくつかの Java のクラスファイルで構成されている。計算機間の通信には HORB、携帯電話間または計算機-携帯電話間の通信には iHORB を利用している。

本クラスタは 2001 年 03 月までに作成した旧ヴァージョン (Version 0.1 系列および 0.2 系列) と 2002 年 02 月上旬までに作成した新ヴァージョン (Version 0.3 系列) が存在する。新ヴァージョンと旧ヴァージョンとの主な違いは利用している通信ライブラリ、送受信バッファの実装方法と携帯電話のサポート、SMP サポートの有無、MPI ライクに利用できる命令の有無である。通信ライブラリは旧ヴァージョンでは RMI を、新ヴァージョンでは HORB を利用している。携帯電話との通信には iHORB を利用している。送受信したいデータは通信バッファに格納し、そのデータに対して送受信が行われる。ローカルホストに対するデータの送受信も、旧ヴァージョンではローカルホスト相手の通信で行っていたが、新ヴァージョンでは通信バッファをサーバ側ではなくユーザプログラムの継承元である CalculatorThread クラス内に持っている。そのため、ローカルホストからのデータ参照を高速に行えるようになった。通信手順のより詳細な比較は各通信手順の章で説明する。MPI ライクな命令は旧ヴァージョンでは実装されていたが、新ヴァージョンでは実装されていない。MPI ライクな命令の廃止は MPI を利用したユーザプログラムと本クラスタを利用したユーザプログラムでは多くの部分が違っており、移植性を考慮する必要があまり感じられず、新ヴァージョン作成の際に旧ヴァージョンから引き継がなかったためである。

2.1 全体構成

ユーザがユーザプログラムを作成し、そのコンパイルを行い、計算の開始を指示し、その結果を得る計算機をマスター。並列計算に用いられるだけの計算機をスレイヴとする。携帯電話もスレイヴである。

マスターにはプログラムファイルとして CalculatorLoader アプリケーション、CalculatorArchiver アプリケーション、CalculatorServer クラス、CalculatorThread クラス、CalculatorThreadInterface インターフェース、CalculatorServerCPE クラス、CPES クラス、設定ファイルとして計算機リスト、ユーザプログラム、携帯電話用のスクリプトファイルを配置する。HORB と iHORB、Java の開発キットとランタイム環境 (Java 仮想マシン含む) も必要である。

スレイヴには最低限 CalculatorServer クラス、CalculatorThread クラス、CalculatorThreadInterface クラスと HORB と iHORB、Java の開発キットとランタイム環境が必要である。

携帯電話は iAppli である CPEC のみでよい。

2.2 利用手順

本ソフトウェアで構成した本クラスタを利用する手順は以下の通りである。

(1) CalculatorThread クラスを継承してユーザプログラムを作成し、任意のディレクトリへ配置

する。ユーザプログラムは CalculatorThread クラスを継承した任意のクラスとして作成され、実際の計算を記述するには CalculatorThread クラスの run メソッドを上書きする。

(2) ユーザプログラムから利用するデータファイルがあるならユーザプログラムの存在するディレクトリ以下の任意の場所へ作成する。

(3) CalculatorArchiver クラスを使ってユーザプログラムとデータファイルをディレクトリごと圧縮する。圧縮ファイルは ZIP 形式なので他ツールを使ってこのファイルを用意してもかまわない。

(4) 各計算機上で HORB サーバを実行する。その際、HORB サーバは本クラスタのサーバである CalculatorServer クラスを含む各クラスにアクセスできるようにしておく必要がある。そのためには classpath 環境変数か java コマンドのオプションを適切に指定する。

(5) CalculatorLoader クラスで圧縮ファイルを各計算機へ転送する。使用する計算機のリストや台数、携帯電話使用のオプションはここで指定する。計算機リストは計算機の IP アドレス、またはホスト名を列挙したプレーンなテキストファイルである。SMP を利用する場合はその計算機のエントリーを CPU の個数だけ記述する。

(6) 各計算機上で実行されている HORB サーバは CalculatorLoader クラスからのデータ転送を受け、CalculatorServer クラスをロードする。

(7) CalculatorServer クラスはユーザプログラムのアーカイブを展開して、ユーザプログラムを実行し、並列計算を開始する。その際、CalculatorServer クラスはアーカイブを展開するディレクトリにアクセスできる必要がある。

2.3 携帯電話のサポート

本クラスタで現在対応しているのは NTT DoCoMo の 503i シリーズで動作する iAppli だけであり、携帯電話拡張機能の携帯電話側は iAppli として作成されている。本クラスタと携帯電話間の通信ライブラリとして使用している iHORB が、現在のところ iAppli にしか対応していないことが原因である。携帯電話拡張機能利用の際は CalculatorServer クラスが実行されている計算機が最低 1 台必要で、その計算機上で iHORB サーバが実行される。iHORB サーバは CalculatorServer クラスから携帯電話拡張機能のサーバアプリケーションである CPES クラスを実行する。携帯電話側ではクライアントアプリケーションである iAppli の CPEC アプリケーションを実行する。

2.3.1 携帯電話での制限とユーザプログラム

iAppli はクラスファイルの動的ローディングをサポートしていない。よって、携帯電話では計算機用に用意された Java で記述され、コンパイルされたユーザプログラムのクラスファイルをネットワーク経由でロードして実行することができない。そこで、携帯電話での並列計算機能に限り、独自のインタープリタ型のスクリプト言語 (以下、本スクリプト言語) の実行エンジンを CPEC アプリケーションは内蔵している。これによってユーザプログラムを携帯電話に転送し、実行させることが可能となる。現在、本スクリプト言語では符号付き正数の四則演算とデータ送受信の機能を実現している。

本スクリプト言語は Java 言語とまったく互換性のないものである。したがって、携帯電話と通常の計算機の併用には 2 つの別なユーザプログラムを作成する必要がある。Java のクラスファイルを実行するインタープリタを搭載することは、プログラムサイズの最大を 10KB に限定している現在の iAppli では不可能と思われる。本スクリプト言語のインタープリタは、現在、kat1.frontier.ous.ac.jp で動作する CalculatorServerCPE クラスとの通信しか行えず、実行可能な本スクリプトのソースファイル名も test.cpe にしか対応していない。加えて、文法解釈の実装が不十分なため実用的なユーザプログラムを作成できる段階ではない。

2.3.2 スクリプト言語

本スクリプト言語には表 1 のような命令群を不完全ながら実装している。変数データはあらかじめ用意された数である 100 個程度しか使用できない。これは携帯電話に搭載されたメモリが少ないため本スクリプト言語から自由にメモリを扱うとメモリ不足となる可能性があるためと、本スクリプト言語が作成中であるためである。各変数へのアクセスはデータの番号を指定することによって行われる。

データ番号 16 に数値 128 を代入したい場合は「mov r16 128」と記述する。r はデータの番号を表している。逆にデータ番号 16 にデータ番号 17 を代入したい場合は「mov r16 r17」と記述する。浮動小数点の演算は iAppli では利用できない。本スクリプト言語でも現在のところサポートしていないため、浮動小数点を扱いたい場合、整数部と小数部を別のデータに分けて演算する必要がある。浮動小数点演算用のライブラリを用意することによって、浮動小数点に関する問題は解決できるものと思われる。

inc	データに 1 加える。
dec	データから 1 減ずる。
add	データに指定数を加える。
sub	データから指定数減ずる。
cmp	データを比較する。
jmp	指定されたラベルへ処理を移す。
send	データをサーバに転送する。
recv	データをサーバから取得する。
mov	データに数値を代入する。

表 1 主な命令一覧

2.3.3 携帯電話クラスタの利用案

現時点において、この携帯電話拡張機能を利用したクラスタが有用な速度と機能を発揮することは、携帯電話の性能が低いためと、実装が不十分なために不可能である。しかし、今後携帯電話を含む情報端末は高速化し、より一般化するに違いない。深夜時間帯に大量の携帯端末を利用して、並列計算を行うことも可能だと思われる。携帯電話でのクラスタの利用案として RSA Security 社の RC5-64 の解読を行っている distributed.net[3] や SETI@home[4] のような常時接続を想定していない利用形態があげられる。

RC5-64 と SETI@home は、どちらも計算結果はユーザの任意の時間にサーバへ送信することができる。携帯電話間では高速な通信を頻繁に行うことができなかったり、パケット料金がかかったり、接続が不安定であったりと言う点を考慮すると通信を積極的に行うことはできない。そのため、必要なデータをメモリの許す限りサーバ側から一気に受信し、必要な計算を行い、一気にサーバ側へ送信するという手順を繰り返す計算が本クラスタにおける携帯電話の利用案として適切だと思われる。

3. 評価および考察

3.1 MPICH と本クラスタの導入手順の比較

本ソフトウェアの導入が手軽で安全であることを説明するため、MPICH と本ソフトウェアの導入手順を比較する。まず、MPICH を利用する場合を考える。MPICH は代表的な MPI ライブラリとして知られているライブラリである。Windows や UNIX、Linux、FreeBSD 等の OS 用に移植されている。加えて多くのプログラム言語から利用するためのインターフェースを搭載している。以下は新規に計算機を複数台購入し、MPICH を利用した PC クラスタを構築する際に必要だと思われる作業の一覧である。この処理全部が必ずしも導入するすべての計算機に対して行われるわけではない。

- 計算機に OS をインストールする。
- IP アドレスやホスト名の設定等、計算機の基本設定を行う。
- コンパイラをはじめとする開発環境をインストールする。
- 各種サーバアプリケーションをメイクし、インストールする。
- NFS サーバの設定を行い、他マシンからデータを共有できるようにする。
- リモートシェルの設定を行い、他マシンでタスクを実行できるようにする。
- MPICH をメイクし、インストールする。
- ユーザプログラムを作成し、並列計算を行う。

最近の Linux ディストリビューションや FreeBSD 等をインストールする場合、コンパイラをはじめとする開発環境や各種サーバはインストール完了段階で利用できるようになっている。IP アドレスやホスト名等の設定についても、それら設定を簡単に行うためのツールを利用できることも多い。しかし、実際のインストール作業では NFS やリモートシェルの設定について等、MPI ライブラリに対する知識や計算機の基本設定とは別に UNIX のサーバ設定に関する知識が必要となる。加えて、ライブラリのメイクや各種サーバのメイク時にコンパイラのヴァージョンに依存するような様々なエラーに遭遇することもある。それらエラーの訂正は簡単に行えるものではない。

新規に購入した計算機を利用しない場合で、既存の計算機に導入されている OS が並列計算ライブラリの要求する OS ではない場合、1 台の計算機に複数 OS を導入するためにハードディスクのパーティションの切り直しを要求されるかも知れない。その際、操作ミスや諸事情によってハードディスク内のデータを失う可能性もある。とある Linux ディストリビューションでインストールには成功するが、他の特定不可能な原因で PC クラスタとして動作させることができないこともあった。既存の並列計算ライブラリはライブラリの一部として導入されない NFS やリモートシェル等のサーバや計算機に導入された OS の多くの設定に依存するため、導入時の負担は大きい。

一方、本クラスタを導入する場合、新規に購入した計算機を利用する場合でも以下のような作業だけで導入できる。

- 計算機に OS をインストールする。
- IP アドレスやホスト名の設定等、計算機の基本設定を行う。
- Java の開発キット (Java 実行環境込み) をインストールする。

- 本ソフトウェア関連ファイル (本ソフトウェアと HORB、更に携帯電話を利用するなら iHORB) をコピーする。
- 計算機のリストを作成する。
- ユーザプログラムを作成し、並列計算を行う。

本ソフトウェアの導入では、MPICH を利用した PC クラスタの構築と比べて必要な作業の種類が少ないだけでなく、 必要な専門的な知識も比較的少ない。更に Java 実行環境が提供されているなら、既存の計算機をハードディスクのパーティション変更をはじめとする大きな設定変更なしに PC クラスタのノードとして流用することもできる。手軽に既存の計算機を PC クラスタに参加させ、より高速な演算を行うこともできる。その際、計算機に加えられる変更は大きな負担となる内容ではない。MPICH で構築する PC クラスタと比べて本クラスタの構築はずっと手軽である。

3.2 ネットワークの通信速度

図 1 は通信速度についての測定を行った結果である。1KB のデータを 10^6 回転送するのに要した時間および 10KB のデータを 10^5 回転送するのに要した時間を測定した。両者とも送信している総データ量は変わらない。リモートメソッドの応答速度は HORB が RMI より高速であるとされているが、データ通信の速度に関しては大きな違いは見られなかった。ThisCluster は本ソフトウェアでの通信速度を示している。このベンチマークは送信するデータのタイプによっては、必ずしも図のような結果になるとは限らない。HORB や RMI の通信について、小さなデータを大量に送るより、データサイズが大きくても送信回数が少ない方が高速であることが確認された。

グラフ上で本クラスタと HORB で大きな差がみられる。HORB ではベンチマーク用のプログラムを作成してベンチマークを行っているのに対し、本クラスタでは本クラスタのユーザプログラムとしてベンチマークプログラムを作成したため、HORB 単体に比べて通信以外の処理に時間をとられているためである。

3.3 台 形 公 式

台形公式の並列プログラムによる本クラスタの評価を行った。図 2 は MPICH を利用した C 言語で作成された並列プログラムでの実行結果、本クラスタの旧ヴァージョン、本クラスタの新ヴァージョンで作成された並列プログラムでの実行時間である。Java 仮想マシンは旧ヴァージョンでは Sun の J2SDK,SE 1.3。新ヴァージョンは IBM の J2SDK 1.3.1 を使用している。

新ヴァージョンではローカルの送受信バッファへの

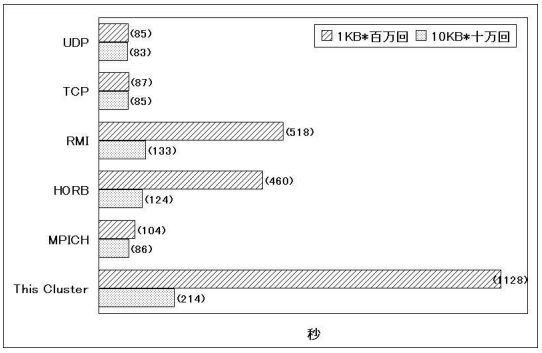

図 1 ネットワークのデータ転送速度

参照速度が高速になったが、未だに 1 回の送受信に時間がかかるためと初期化時のユーザプログラムの送信に時間がかかるためという二つの大きな理由から、2 台での計算が一番高速な数値となっている。大きなデータを 1 回送信するのに要する時間より、小さなデータを複数回転送するのに要する時間の方が大きい。加えて、ユーザプログラムの送信は一台ずつ完了を待つため、2 台での計算が最高速度を出したものと思われる。

しかし、本クラスタの新ヴァージョンは 3 台以上の台数でも MPICH よりも低速ながら、旧ヴァージョンの本クラスタより高速な演算を行うことができるようになっている。これは通信バッファによる影響だと考えられる。

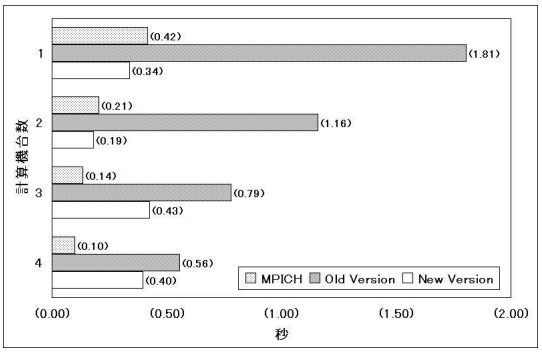

図 2 台形公式の実行時間

本クラスタにおいて利用台数が増加するとともに台数効果が著しく低下する。これは各計算機へのユーザプログラムの送信に時間がかかるためである。この問題を解決するために、当初、マルチスレッドで初期化を行うことにより回避していた。しかし、開発のある時点からマルチスレッドの同期の問題からエラーが発生し、それを回避するためにマルチスレッドを一時的に廃止した。

4. おわりに

本研究では Java による PC クラスタ・ソフトウェアを開発し、その簡単な評価を行った。本ソフトウェアの旧ヴァージョンから新ヴァージョンでは一応の高速化を実現した。簡単な並列計算で本クラスタの旧ヴァージョンは MPICH と比較すると 0.2〜0.7 倍の性能だったが、新ヴァージョンでは 0.3〜1.2 倍の性能を発揮した。さらに新ヴァージョンでは携帯電話の参加という新たな試みを行った。導入についても専門知識をほとんど必要とせず、手軽に導入できるソフトウェアとなっている。しかし、MPICH や他の PC クラスタと比較すると速度や機能面ではるかに劣っている。この点は今後の開発で補う必要がある。

本ソフトウェアにおいて、今後、考えられるの改良点をあげる。

- 計算機リストを必要としない初期化手順
- 通信ライブラリの自前実装
- 携帯電話拡張機能の MIDP 対応
- 同期命令の再実装
- 初期化のマルチスレッド化
- 補助機能強化

計算機リストを必要としない初期化手順により、計算機リストの作成の手間を省き、より簡単な運用を目指す。実装方法としては、任意の計算機 1 台で動作する本クラスタのサーバをマスターサーバに指定して、各計算機で本クラスタのサーバを起動することで計算機リストの作成を自動化する。各計算機で本サーバの終了時にマスターサーバ上のリストから登録を削除するようにすれば常に利用可能なサーバの一覧を保持することが可能となる。

携帯電話拡張機能を MIDP に対応させることによって、iHORB が利用できなくなる。加えて、HORB を通信ライブラリとして利用している場合、どうしても通信のパフォーマンスは改善できない。通信ライブラリを自前で実装すれば両方の欠点を補うことができる。

同期命令のテストが不十分であり、同期に失敗することが多いため実装の見直しも必要である。現在同期命令は不完全な実装であるバリア同期しか実装していない。そのためユーザプログラムはバリア同期とフラグの操作を手動で行っての同期しか行えない。

更に必要ならば、旧ヴァージョンでは一部利用可能であった MPI ライブラリで用いられる命令に似せた命令を再び実装することにより、既存の MPI ライブラリを利用したユーザプログラムの移植性をあげると同時により高度な計算を行えるようになる。

初期化のマルチスレッド化については前述したとおりである。補助機能としては、計算機と携帯電話で同じように本スクリプト言語を実行できる機能や中断された計算の復帰、IO の一元化などが考えられる。しかし、改良や機能強化とともに複雑化しないよう注意する必要がある。

参考文献

1) 日下部明, 廣安知之, 三木光範:"Java による MPI の実装と評価",2000 年記念並列処理シンポジウム JSPP2000 論文集,pp.269-276(2000)
2) 高木浩光, 松岡聡, 中田秀基, 佐藤三久, 関口智嗣, 長嶋雲兵,Ninflet – Java による World-Wide High Performance Computing 環境, インターネットカンファレンス'97, 1997.
3) distributed.net,"stats.distributed.net", http://stats.distributed.net/
4) SETI@home,"SETI@home: Search for Extraterrestrial Intelligence at Home", http://setiathome.ssl.berkeley.edu/