

ネットワーク分散環境における JAVA/C 言語連携に関する考察

寺元 貴幸* 高橋 原野**

Consideration of cooperation of a JAVA / C language in network distributed environment

Takayuki TERAMOTO, Genya TAKAHASHI

Recently a lot of high-performance personal computers became connected to Internet. Line speed becomes faster and faster by using ADSL technology. Distributed computing technologies (Grid and MPI .etc) attract attention in recent years. However, it is very difficult for us to use these technologies. We want to use these technologies more easily. We thought to use the Windows System which diffused most widely and constructed a distributed environment.

We investigated basic technology of distributed processing. As a result, we confirmed that JNI technology was very useful and convenient. JNI technology relates C language to JAVA language. We improved our distributed processing system using JNI. This paper describes a brief description of our distributed system and reports an adapting example of JNI.

Keywords Distributed Computing, Java language, C language, JNI technology, Windows System

1. はじめに

近年、パーソナルコンピュータの性能向上と低価格化に加え、ADSL や光ファイバーによる高速ネットワークが各家庭に普及するようになってきた。このためインターネットには高性能なパソコンが大量に接続されるようになって来ている。ネットワーク分散処理技術はそのような背景の下で発展してきた技術である。1990 年代までは大規模計算の主流はスーパーコンピュータであった。その後、パソコンを利用したクラスタ技術が注目を集めてきた。クラスタ技術はFig.1のように複数のパソコンをネットワークで接続して全体で 1 台のコンピュータのように動作させる方法である。また、さらにボランティアによる多数のパソコンを利用して大規模な計算を分担して行うタイプの分散処理もある。しかし、いわゆる一般ユーザにとってこれらの技術は以下の理由から容易に利用できるとは言えない。

- 一般ユーザがスーパーコンピュータや超高速ネットワークを利用するコンピュータ Grid¹⁻³⁾に参加することは現実的に困難である。

- クラスタの標準言語である MPI⁴⁻⁵⁾のプログラミングには並列計算に関する高い知識が必要で、効率的な並列化のためにはソフトウェア・ハードウェアに関する専門知識が不可欠になる。
- SETI@home⁶⁾のような大規模分散計算は、計算に参加することはできても、自分自身の必要とする計算（プログラム）を実行することはできない。

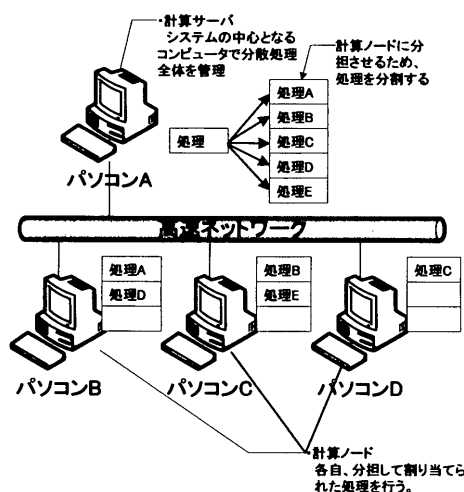


Fig.1 Network distributed processing technology

原稿受付 平成 16 年 8 月 31 日

情報工学科

** 情報工学科 5 年生

我々はこの状況を解決し、一般的なコンピュータ利用者が、比較的容易に利用することが可能なシステムの研究を行っている。我々が研究を行ってきた分散処理の一種である、多数パラメータ実行支援システムとオブジェクト分散型授業支援システムの研究の利便性を向上させるため、今回その基礎技術として Java 言語と C 言語の連携技術について報告したい。

1. コンピュータ環境と分散処理システム

2. プログラミング言語による制約

Linux OS(Operating System)⁷⁾や Free BSD⁸⁾などは Unix OS を基盤として開発が行われている。Unix OS は設計の段階からネットワークへの接続を考慮した OS であり、同様にネットワークでの利用を最重要事項に開発されたプログラミング言語 Java⁹⁾とは非常に相性が良い。この二つを利用すると比較的簡単に分散処理システムを構築することが可能である。しかし、パソコンの OS として事実上最も普及しているのは Microsoft 社の Windows システムである。Windows は C++言語を中心に開発されており、分散処理技術に関しては未成熟であり、しかもシステムに関して一般ユーザが入手できる情報が限られているという問題がある。Windows システムでも分散オブジェクト技術として DCOM や COM+があるが、OS 依存や脆弱性によるセキュリティの問題、そして開発環境が複雑でしかも高価であるといった問題が指摘されている。

以上のことから Windows システムを利用しつつ無料でしかも安定したネットワーク分散技術を利用できる環境が整備できれば、今後各種研究の基礎技術となりうると考える。

ここで OS と Java 言語の関係について考察してみる。Java 言語の主な開発コンセプトはプラットフォームから独立・非依存した実行処理形態を取ることである。これを実装するために OS ごとに用意された仮想マシン JVM(Java Virtual Machine)がシステムの違いを吸収し、その上でプログラムが実行される。

しかし実用的な分散システムを Java 言語だけで構築することは困難である。Java 言語だけで分散システムが構築できない最大の理由は、分散システムでは CPU の使用率やプログラムの自動起動などに密接に関係した機能が必要となるからである。C/C++を常に 100%使い切ってしまうような分散システムでは、ユーザの利便性を極端に損ねることになる。Fig. 3に示すように Java はコンパイル後に生成されるバイトコードを JVM が翻訳しシステムに実行させ、Java 言語自体には特定のシステムに依存

した API (Application Program Interface)を用意しないという設計思想に基づいている。システムの安全性を高めるために、このような低レベル部分にはアクセスできないことが、逆に実用的な分散処理システムでは妨げになってしまう。

そこで、Java 言語と C 言語を組み合わせるシステムを構築する必要があり、そのためには二つの言語の連携技術が必要になってくる。この連携技術について 3 章で述べる。

ただし、C 言語との連携技術は部分的にシステム依存が含まれることが避けられないために、OS の違いにより Windows でも 98-Me 系と NT-2000-XP 系により、違った仕組みとなる。今回は主に NT-2000-XP 系に関して報告を行う。

また分散処理システムはパソコンの電源が入り、OS が起動した後はいつでも利用できる準備が整っていることが望まれる。しかし Java 言語には自動起動のシステムが無く、しかもバッチ処理で起動した場合は各ユーザによって実行権限やアクセス権が異なってしまう。分散処理システムの起動を各パソコンの利用者自身が行わなければならないようなシステムでは確実性が保証されないし、ユーザが誤ってシステムを止めるなど不都合が多い。そこで、C 言語との連携技術に加え Windows 上で自動実行されるサービス化を Java 言語で行うことが必要となる。この部分に関しては 4 章で述べる。連携が行われた場合の動作の様子を Fig. 2 に示す。

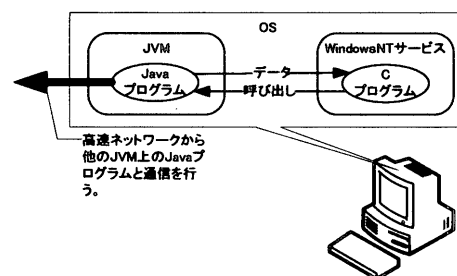


Fig. 2 Cooperation of a C and JAVA language

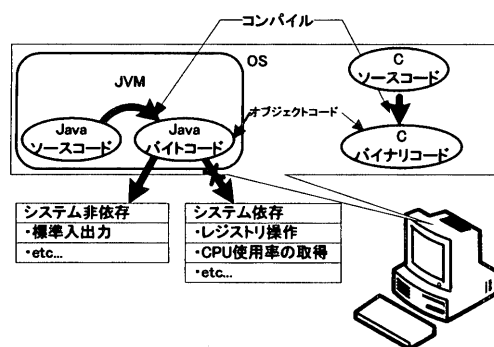


Fig. 3 Limit of Java by system dependence

2.2 分散処理システム

Java 言語は設計当初からネットワークでの利用をベースに設計されているため、ネットワーク関連のプログラムも容易に作成できる。また、シンプルなオブジェクト指向技術はより大規模な処理の実現やプログラムの管理を容易にするなどの利点がある。Java 言語のこの特徴を生かした国産の分散オブジェクト技術として HORB(Hirano Object Request Broker)¹⁰⁾がある。HORB は Java のクラスとして作成されており、このクラスを利用することで非常に容易に分散オブジェクトを構築できる。これは IDL (Interface Definition Language) の定義を必要とする CORBA テクノロジーに比べて格段にプログラミングが容易であり、しかも動作速度が速いという特徴がある。

我々はこの分散処理技術 HORB を利用して、高度な並列プログラミングの能力を持っていない者に対して並列計算を提供する多数パラメータ実行支援システムの構築¹¹⁾を行っている。具体的には Fig. 4に示すように一般ユーザの利便性を考慮して、パラメータの入力や計算の進捗状況の確認に Web ブラウザを使用し、計算に関する情報はデータベースで管理することで計算結果等を容易に表示・再利用できるようにした。システムの特徴をまとめると以下になる。

- 利用ユーザは計算したいパラメータの組み合わせを入力するだけで計算を実行できる。
- パラメータの入力は Web ブラウザで行う。
- 計算の進捗状況や計算結果も Web ブラウザで確認できる。
- 計算すべきプログラムの作成は、高度な並列計算の専門知識を必要としない。
- ユーザは計算に参加するコンピュータの台数や能力を意識する必要がない。

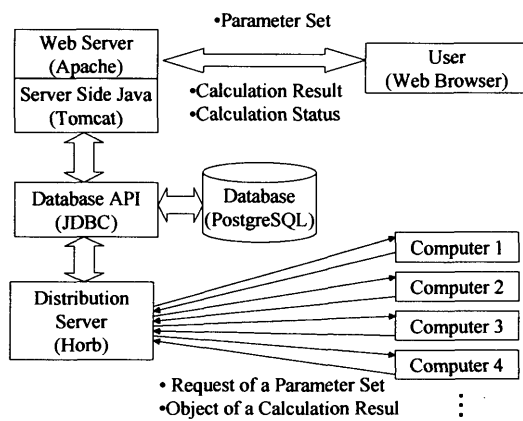


Fig. 4 Structure of the Support System

我々が開発した従来システムは、計算ノードが

Windows OS の場合に計算を開始できるようにするための準備が多少面倒であった。それは OS の起動後に特定ユーザでログインを行い、さらに HORB サーバと接続するためのクライアントプログラム(Java 言語)を起動するといった手順が必要であり、広く利用する場合の障害となっていた。今回、Java 言語と C 言語の連携により、これを完全に自動化することが可能となったので、その基盤技術を以下に述べる。

3. Java 言語と C 言語の連携

3.1 JNI (Java Native Interface)

2章で説明したとおり、実用的な分散処理システムを構築するには Java 言語と C 言語の連携を行う必要がある。連携にはいくつかの方法があるが、我々が開発を行っているネットワーク上で多くの計算を行うシステム、ならびに授業支援システム¹²⁾にとって必要な要素を検討した結果、以下のような項目が必要となることが分かった。

- CPU の利用率やメモリ使用率などの情報が取得できる。
- 大量のパラメータを相互に授受できる。
- 動作が十分に高速であり、Java の開発環境である SDK(Software Development Kit)のバージョンに大きく左右されない。

この要件で調査した結果、JNI (Java Native Interface) が有力であることが分かった。JNI は Java ネイティブメソッドを書いたり、Java 仮想マシンをネイティブアプリケーションに組み込んだりするために用意された API である。JNI は JDK1.2β 版から導入された技術で、比較的古くから利用できたが、Java2 以後はあまり注目されず関連資料や参考書がほとんど無いのが実情である。これはプラットフォーム非依存という Java 本来の大きな特徴を損なう可能性があり、しかも一時期 Microsoft と Java は非常に難しい関係にあったため、Java の開発を行っている Sun Microsystems が大きく宣伝しなかったためであろうと考えられる。しかし技術的にはかなり安定しており、Windows OS 上で分散システムを構築する場合は必要不可欠と考える。

JNI の主な目的は、各プラットフォーム上の Java 仮想マシンの実装間で、ネイティブメソッドライブラリをバイナリ互換とすることにある。JNI には Java から C のライブラリ関数を呼び出す手法と、逆に C から Java のバイトコードメソッドを呼び出す手法が存在する。JNI の実体は C のライブラリ群であり、いくつかの JNI 専用の型や関数群から構成されている。また、Java 言語仕様にも、JNI のため

の予約語などが含まれている。

一般に、Cプログラムのほうは、プラットフォームに依存した開発環境でコンパイルし、Cライブラリとリンクすることになるので、コンパイル後のコードを他のプラットフォームにコピーしても実行することは出来ない。このようにC言語部分にプラットフォームに依存したコードを利用してしまうと、Javaのプラットフォーム非依存性が失われてしまう。しかし、今回のようなシステムを構成する場合、JavaのAPIだけでは実現できない部分がある。依存性と利便性のトレードオフにより今回はJNIを利用することとした。また今回は計算ノードとしてWindows OSだけを対象としたので依存性の解決は今後の問題としたい。

3.2 Java のメソッドから C の関数を呼び出す

これはJavaのバイトコードから、C/C++のネイティブのコードを呼び出す手法である。これは既存のプログラムをDLL(Dynamic Link Library)として使用するため、Javaでは実現できない機能を比較的容易に実装できる。JavaからCの関数を呼ぶ手順は以下の通りである。

- (1) Cの関数を呼ぶJavaプログラムを書きコンパイルする。
- (2) そのバイトコードからC用のヘッダファイルを生成する。
- (3) (2)で生成したヘッダファイルを使用してCソースファイルを作成する。
- (4) Cソースファイルをコンパイルし、DLLを生成する。
- (5) Javaプログラムを実行し、結果を確認する。

この手順により、Fig. 5が示すようにDLLに書かれたCの関数を呼び出すことが可能になり、Javaでメソッドを呼び出すのと同様の操作でC言語の機能を呼び出すことが可能になる。この方法は、主にJava言語で行うのが困難なシステムよりの処理を追加したり、もしくは既存のCソースコードをラッピングしてJava言語で使用する場合に用いられる。

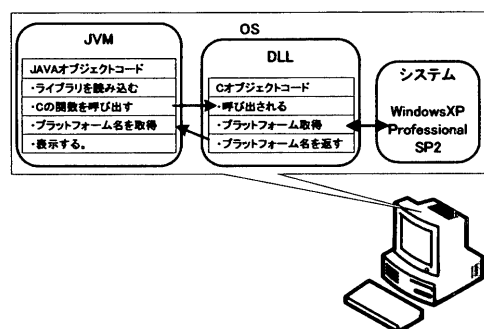


Fig. 5 Structure of a System Call by DLL

しかし、今回我々が研究している分散処理システムでは、この方法では処理を行うことが無理な部分が存在する。それは、本方法ではプログラムがJVM上で動作しているため、手動もしくはバッチプログラム等でJVMを起動しなければならないためである。この場合、ユーザがログインを行った後なら実行することができるが、ログイン前の待機状態のパソコンでは実行状態にすることはできないのである。この問題を解決するために次の方法を検討した。

3.3 C の関数から Java のメソッドを呼び出す

これはCのプログラムからJVMを起動し、そのJVM上でコンパイル済みのJavaバイトコードを走らせる方法である。この方法が可能になれば、OSが起動されていれば、たとえユーザがログインしていなくても分散処理システムを実行できるようになり、効率的な実行が可能となると同時に、ユーザによって不必要に停止される危険性が無くなるため、連続的な動作が可能となる。

CからJavaのメソッドを呼ぶ手順は以下の通りである。

- (1) Cから呼び出すJavaプログラムを書き、コンパイルする。
- (2) Cで起動用のソースコードを書く。
- (3) コンパイル時にJavaのインストールフォルダ以下のIncludeフォルダ、及びInclude/win32フォルダをオプションとして追加する。リンク時に、jvm.libをオプションとして追加する。
- (4) jvm.dllがあるフォルダに環境パスを通し、Cプログラムを実行して、結果を確認する。

以上の操作でC言語からJavaを呼び出すことが可能となる。しかし、Javaの起動を行うためのCのソースコードにはJVMを利用するための仕組みがいくつか必要となるので、次に、ソースコードに必要な事項を述べる。

- (1) jni.hをインクルードする。
- (2) JNI_GetDefaultJavaVMInitArgs()を用いてJVMのデフォルトオプションを取得する。
- (3) デフォルトオプションに任意のオプションを追加する。
- (4) JNI_CreateJavaVM()でJVMを作成する。
- (5) JNIEnv クラスの FindClass()で実行するクラスのクラスIDを取得する。
- (6) JNIEnv クラスの GetStaticMethodID()を用いて実行するメソッドIDを取得する。
- (7) JNIEnv クラスの CallStaticVoidMethod()でJavaのメソッドを呼ぶ。
- (8) JavaVM クラスの DestroyJavaVM()でJVMを開放する。

以上の手順により、C言語からJavaの機能を利用

用することで、4章で述べる分散処理システムのサービス化が可能になる。

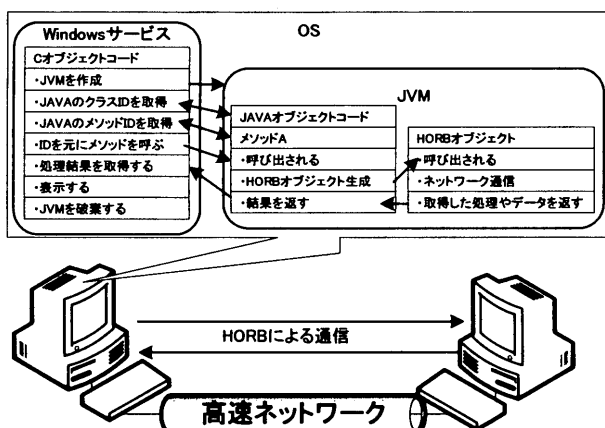


Fig. 6 Starting of JVM environment by C language

Fig. 6が示すようにCのプログラム中でJVMを起動し、その後JavaクラスライブラリのメソッドJVM上で走らせることが可能になり、今まで我々が作成した分散処理システムをそのまま動作させることが可能となった。

本方法は基本となるプログラムをCで書くことで、Windowsサービスでラッピングすることが可能のため、システム起動後のログイン前待機時にプログラムを実行状態にすることができ、このログイン前の状態は、パソコンの電源を投入しただけで実現できるのでWake on LANに対応したパソコンを利用すれば、ネットワーク経由で分散環境を整えることが可能であるほか、使用できるメモリやCPUパワーなどの計算機資源も無駄なく最大限利用できる。

また、ネットワーク部分にはHORBBを使うために、ネットワーク部分を完全にJavaで記述することができ、他のプラットフォームへの拡張が容易であることも重要な特徴である。

最近ではJVMの性能もよくなってきており、システムの一部をCで書き、JNIを通じてコールする方法は主流では無いかもしれないが、今回の方法は実用的なネットワーク分散技術として有効であることが確認できた。

4. Windows サービス

4.1 サービスとは

Windows上で動作するサービスは、ユーザインタフェースを持たず、OSシステム自体が動作を管理するプロセスである。一般的にはシステムの起動時

に起動され、終了時に終了するが、途中でシステム管理者が起動することもできる。各種共有サービスやデータベースサーバの機能などがこの機能を利用している。一般的に直接CPUやメモリを大量に消費せず、ユーザはその存在に気がつくことはほとんどない。また主に実行中の多くの時間をイベント待ちのまま過ごすという特徴がある。通常のアプリケーションはログインしたユーザが実行権を持つが、サービスの場合はシステムの権限を持ち、一般ユーザはサービスの一覧を表示する権限しか持たない。この動作はUnix OSで言う「デーモン」プロセスに相当する。

サービスは、SCM(Service Control Manager)によって制御されている。SCMが管理しているサービスの一覧は、Fig. 7のようになっており、Windowsの管理ツールから確認できる。この一覧からは特定のサービスを開始、停止、一時停止、再開などの制御を行えるようになっている。

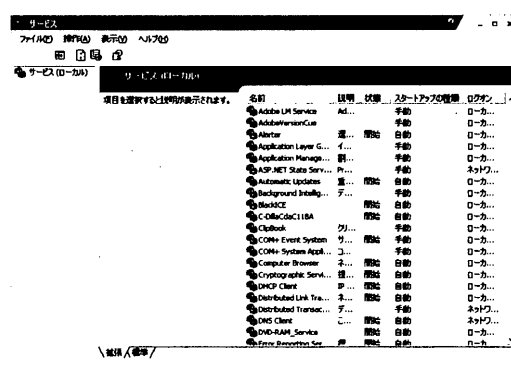


Fig. 7 List of Windows Service

4.2 サービスの構築

サービスは次の4つからなる。SCM, SCP(Service Control Program), サービスコンフィギュレーションプログラム、そしてサービス自体である。

SCMはサービスに対し、起動・終了・一時停止・処理続行などを指示する。SCPはサービスの起動・終了・一時停止・続行・その他の、サービスを制御するためのユーザインタフェースを提供する。サービスコンフィギュレーションプログラムは、サービスのインストール・削除・構成の変更を行う。

SCMはサービスプログラムを起動すると、サービスプログラムからStartServiceCtrlDispatcher関数が呼び出されるのを待つ。関数は新しいスレッドを作成し、そのスレッドでサービスのServiceMain関数を呼ぶ。呼んだServiceMain関数から戻った時点でそのスレッドを終了させる。

またサービスは制御ハンドラのHandler関数を持たなければならない。これはサービスがSCPか

ら、制御リクエストを受信したときに、サービス制御デスパッチャが呼び出す。したがってこの関数は制御デスパッチャのスレッドで実行される。

サービスの状態を変更させるには、サービスに制御リクエストを送ればよい。これは ControlService 関数を使用する。状態には次の4つがある。

停止
一時停止
続行
更新情報取得

4. 構築したサービス

最初にサービスとして構築したのは、多数パラメータ実行支援システムにおける計算サーバと計算クライアントの通信を行う部分である。この部分のサービス化により、ログイン作業なしに分散処理が可能であることを確認した。

また、別の用途として我々が研究を行っている授業支援システムにも応用してみた。Fig. 8に示すように、授業支援サーバがログイン情報の取得を要求すると、各学生機上のエージェントが OS にログイン情報を要求する。各端末で起動しているサービスはユーザがログインして稼動中なら使用中だとサーバに返し、ログイン前の待機中ならばサーバに使用不可だと返すものである。

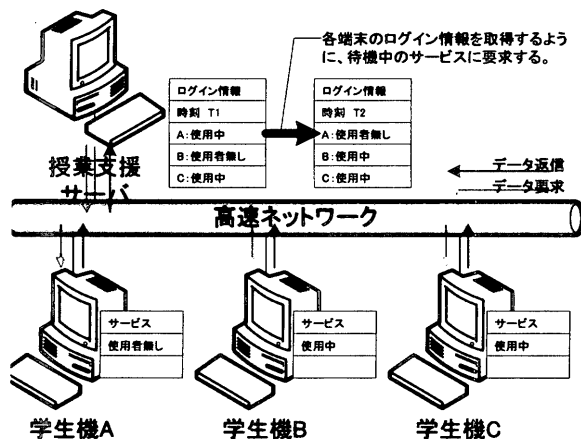


Fig. 8 Class support system

これが普通のアプリケーションとして起動した場合は、ログインしていないマシンの状態の取得が不可能となっていた。サービス化することで、システム起動と同時に稼動状態になるサービスだから可能なのである。

5. まとめ

今回は Java 言語と C 言語の連携技術を調査し、我々が研究を行っていた多数パラメータ実行支援システムおよびネットワーク分散型授業支援システムにおいてどのような方法が最適か検討を行った。その結果、JNI を利用して C 言語と Java 言語の連携を行うことで、従来のシステムで問題になっていた、初期起動の問題やユーザによる中断といった不具合が改善できた。これらの技術を利用することでさらにシステムの効率化や安定運用が期待できる。

今後の課題としては、プラットフォームの依存性の問題がある。利便性と引き替えに互換性を失ってしまったが、JNI による C 言語との連携は Windows システム以外でも利用できる。他のプラットフォームへの移植を進めると共に、支援システム全体の利便性をさらに改善していきたい。

参考文献

- 1) Global Grid Forum: <http://www.gridforum.org/>
- 2) Globus: <http://www.globus.org/>
- 3) UNICORE: <http://www.unicore.de/>
- 4) Message Passing Interface Forum, MPI: A Message Passing Interface Standard, University of Tennessee, Knoxville, Tennessee, (1994)
- 5) 青山幸也: 並列プログラミング虎の巻 MPI 版, 日本 IBM, (1997)
- 6) SETI@home: <http://setiathome.ssl.berkeley.edu/>
- 7) Linux: <http://www.linux.or.jp/>
- 8) FreeBSD: <http://www.freebsd.org/index.html>
- 9) Java: <http://www.java.com/>
- 10) HORB: <http://www.horb.org/horb-j/>
- 11) 寺元貴幸, 土居正行, 荒砂茜, 川部健: ネットワーク分散環境を利用した多数パラメータ実行支援システムの構築, 津山工業高等専門学校紀要, 45(2003) 53-58.
- 12) 寺元 貴幸, 岡田 正, 青山 龍一, 齊藤 智也, 中村恭志, 川田 重夫: データベースを活用した分散授業支援システムの構築, 日本計算工学会論文集, 3(2001)87-94.