ハイパフォーマンス
コンピューテイング 87−26
（２００１．７．２７）

# 分散計算システム WDC 用基本ライブラリの構築

田代友成† 大野和彦† 中島浩†

我々は分散計算システム WDC (Widely Distributed Computatin) を提案した．その特徴としては，(1) 随時参加型，(2) 問題適用範囲の汎用性，(3) 能力に応じた計算量配分が挙げられる．

本報告ではこの WDC を，より大規模分散でき，より多くの人が参加でき，より使いやすいシステムとすることを目的とした改良項目について議論する．具体的にはクライアントとサーバを仲介するプロクシシステム，Web ランキングをサポートするためのデータベース化，ユーザが求める計算をより簡単に記述できるような基本クラスの提案を行う．

# Basic library for distributed computation system WDC

Tomoshige Tashiro,† Kazuhiko Ohno† and Hiroshi Nakashima†

We have proposed the distributed computation system WDC, that has the following features; (1)free join style; (2)widely applicable problem domain; and (3)dynamic computation amount adjustment.

In this paper, we discuss about the improvement of WDC aiming to make it more scalable with respect to its spatial distribution and number of participants, and also more usable.

More specifically, we propose a proxy system that mediate between the server and clients, a database system to support Web ranking, and basic classes to simplify user programs.

## 1. はじめに

インターネットに接続された多数のコンピュータを利用し，大規模計算を行う研究が最近注目されている．

その一つに，まずインターネット上にて計算参加者を募り，その参加者が利用していない時間の計算能力を大規模計算に提供する方法がある[1)-4)]．

そのシステムとして，我々は WDC (Widely Distributed Computation) を提案した[5)]．その特徴として，(1) 随時参加型，(2) 問題適用範囲の汎用性，(3) 能力に応じた計算量配分が挙げられる．

そして，WDC はこの特徴を生かしつつ，よりよいシステムとして改良を続けている．WDC の改良目標としては，

- より大規模分散可能
- より多くの人が参加できる
- 機能ををより使いやすく

等が挙げられる．

この目標をもとに本論文では，サーバ，ネットワークの負荷を分散するプロクシシステム，計算ステータスの公表を助けるデータベース化，ローカルサイトでの反復した実行に使用できるクライアントコンテナ，そして，クラス作成を容易にするベースクラス等を提案する．

なお，本論文において**ユーザ**とは，WDC を用いて特定の計算をさせようとしている者のことを指す．

## 2. WDC システムの概要

WDC は，(1) 随時参加型，(2) 問題適用範囲の汎用性，(3) 能力に応じた計算量配分に特徴のある，クライアントサーバ型の分散計算システムである．

特定の計算を行おうとするユーザは，以下の手順で分散計算を行う．

(1) ユーザの想定する問題が WDC の用意する計算モデルに合致した計算か確かめる．

(2) WDC の用意するモデルを用いて，その計算部分 (**domain**，**calc**，**result** の 3 クラス) のみを記述する．

(3) ユーザの記述したクラスと WDC システムをコンパイルし，クライアントサーバプログラムを得る．

(4) ユーザのサイトにサーバを立ち上げ，計算参加者を募り，多くの計算参加者にクライアントを

† 豊橋技術科学大学
Toyohashi University of Technology

ダウンロード，実行してもらう．

(5) 立ち上げられたそれぞれのクライアントは，サーバと通信し計算を行う．

### 2.1 計算モデル

本システムでは以下の手順に分割可能な大規模計算を対象としている．

(1) 定義域を個々の部分定義域に分割する．

(2) それぞれの部分定義域に対し，独立して部分計算を行う．

(3) それぞれの部分計算より，各々の部分解を得る．

(4) 求められた全ての部分解を統合し，最終的な解を得る．

計算モデルを図 1 に示す．

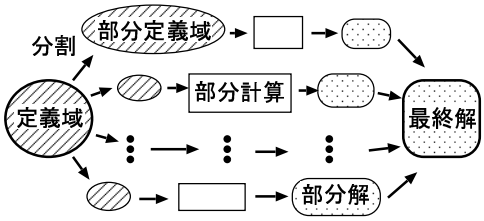


**図 1** 計算モデル

つまり，

- 定義域が部分定義域に分割可能
- 部分定義域が各々独立して計算可能

の 2 つの条件を満たしていれば，ユーザは WDC により分散計算を行うことができる．

### 2.2 実行モデル

実行モデルを図 2 に示す．本システムは，クライアント/サーバ型システムである．図 1 のモデルは実行時には以下のように動作し，分散計算を実現する．

#### 2.2.1 サ ー バ

サーバ起動時にまず，(1) 定義域が初期状態にセットされる．これが計算開始時の定義域となる．

次に，(2) クライアントからの接続を待ち，(3) 定義域から部分定義域を分割し，(4) その部分定義域をクライアントに送信する．

その後サーバは，(5) クライアントからの接続を待ち，接続される毎に，(6) 部分解をクライアントから受信し，(7) 部分解を最終解に統合し，(8) 定義域から部分定義域を分割し，(9) その部分定義域をクライアントに送信する．

この (5) から (9) までの繰返しを，全体の計算が終了するまで行う．

#### 2.2.2 クライアント

計算参加者によって起動されたクライアントはそれぞれ，次の動作をサーバから計算終了を通知されるまで繰り返す．

(1) サーバと通信し部分定義域を得て，(2) その部分定義域から部分計算を行い部分解を得，(3) 得られた部分解をサーバに送信する．

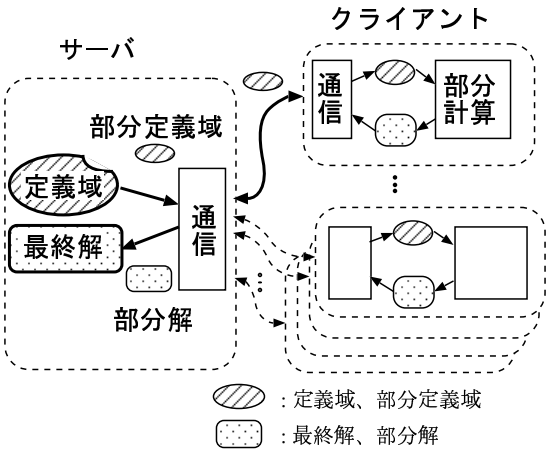


**図 2** 実行モデル

## 3. プ ロ ク シ

大規模分散計算を行おうとした場合，負荷を集中させない機構が必要となる．計算能力の比較的低いクライアントに大量に参加してもらい計算を行うこのシステムでは，クライアントの要求が集中するとボトルネックとなってしまう．

つまり，本システムではスケーラビリティが重要であり，サーバの負荷を分散させ，軽減させる機構が必要である．そこで，サーバとクライアントの中間に位置し，サーバからはクライアント，クライアントからはサーバと見なすことのできる，負荷分散機構プロクシを提案する．図 3 にプロクシの概念を示す．

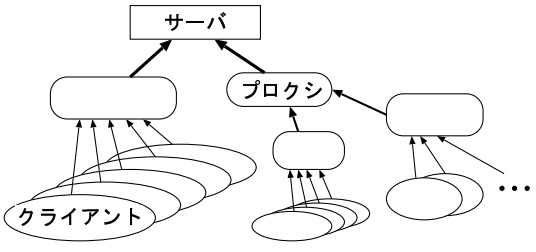


**図 3** プロクシシステム

### 3.1 効 果

プロクシシステムを導入することによって得られる

効果には，主に次の 2 つが挙げられる．

- 複数のクライアントがサーバに対して行う要求をプロクシからの要求として 1 つにまとめることで，サーバの処理するクライアント数を減らす．
- 複数のクライアントがサーバへと通信するとき，サーバ，クライアント間ネットワークの間にプロクシを置くことで，通信をまとめ，通信量を減らす．

また，これらの効果を持つプロクシを，階層的に接続することを可能とすることにより，より大規模分散計算を実現できるシステムとなる．

### 3.2 目　　標

プロクシシステムが円滑に動作するために必要な，具体的な目標を説明する．

#### 3.2.1 偏りの少ない配分

まず，多重に階層化したプロクシの中で動作するためには，全体的にすべてのプロクシで偏りのない計算量配分を行う必要がある．

あるプロクシだけに定義域が偏って多く存在している場合を考える．すると，終了するときにそのプロクシだけ多くの計算量を持ち，そのプロクシの担当するクライアントだけ計算終了までに時間を要する．

つまり，それぞれのプロクシには，それぞれのプロクシが担当するクライアントの計算能力の合計に見合った計算量を，バランスして持たせる必要がある．

#### 3.2.2 様々な立ち上がりに追従

コンピュータの性能の向上は著しく，数年前の計算機と現在の計算機では性能が大きく異なる．そのため計算に参加してくるクライアントの能力は，性能の高いものから性能の低いものまで様々である．

性能の高いクライアントの場合は急速に計算量を増やしていき，立ち上がりも速いが，性能の低いクライアントの立ち上がりは遅く，すぐに落ち着いてしまう．この，異なる性質を持つクライアントに，それぞれ異なる，最適な計算量を与えなければならない．

そして，そのクライアント自体，数十台まとめて同時に参加してくる可能性もあり，そのような場合にもプロクシは対応できなければならない．

#### 3.2.3 定義域量を特定サイズと前提を置かない

定義域の量とその計算時間の粒度は，ユーザの設定する計算問題により，大きく変化する．

つまり，最小単位の定義域の計算であっても，非常に時間がかかる場合と，ある程度大きな単位の定義域の計算であってもあまり時間がかからない両方の場合が想定されるということである．

システムとしては，このように定義域量についてあらかじめ前提を置くことは許されず，その計算量は実行時に，クライアントの計算能力と定義域の必要とする計算能力の関係のみから求める必要がある．

#### 3.2.4 タイムアウト多重化問題の回避

本システムはサーバ以外の信頼性は絶対ではなく，サーバやプロクシなどの上位層は，プロクシやクライアントなどの下位層がネットワークの断絶などにより突然使用できなくなることを考慮しておく必要がある．そのため，上位層は下位層のトラブルをタイムアウトにより検知する．

しかし，プロクシシステムを導入することによりサーバとクライアント間が多重の階層を持つことになると，タイムアウトの多重化問題が発生する．プロクシがクライアントのタイムアウトを検知したことをサーバが直接知ることはできず，階層の数だけ待つ必要が出てしまう．その階層の数は不定であり，サーバは一定時間内にクライアントがタイムアウトしたことを知ることができなくなる．

そこで，このような問題が起こらないように適切な対処が必要となる．

### 3.3 プロクシシステムの設計

3.2 節にて示した目標を達成するようなプロクシシステムを設計する．

基本は，クライアントからの要求にサーバと同様に応えることである．プロクシにおいても，クライアントについてのスケジューリングはサーバと同じものとしている．

そしてそのクライアントに渡す定義域は，サーバ，または上位プロクシから取得する．その上で以下 2 点の機能を付けることにより，前節の目標を達成する．

- 定義域の複数化
- 割り当て定義域の分割と返却

#### 3.3.1 定義域の複数化

クライアントは，計算中一つの定義域を持ち計算を行う．しかし，プロクシの場合においては，定義域を一つに限定すると以下の不都合が起こる．

まず，プロクシはサーバへ一つの定義域を要求し，その定義域をクライアントのために分割する．その後，多くのクライアントから分割した部分解が返却される．このとき，サーバから割り当てられた部分定義域に対応する部分解をサーバへ返すためには，すべての部分解をまとめ一つの部分解とした後で返却しなければならない．

このため，プロクシはすべての部分解が集まるまでサーバへ部分解を返却できない．もしこの時クライアントにタイムアウトが起きていると，タイムアウトま

で待つ必要がある．

そこで，プロクシに対しては定義域を複数持たせることによりこの問題を解決する．もし，プロクシの持つ 2 以上の定義域のうち，先に分割を始めすべて分割し終えた定義域に対する解がすべて集まっていない状態の時でも，接続してきたクライアントには次の定義域から分割を行う．また，複数のクライアントから急激に要求がやってきた場合には，プロクシの持つすべての定義域がなくなってしまうが，その場合においてもサーバへと追加で定義域を要求することによりクライアントの要求に応える．

#### 3.3.2 割り当て定義域の分割と返却

サーバが分割する定義域が無くなり，そのすべての解が返却されたときに計算は終了する．しかし，解はすべて順調に返却されるとは限らない．

もし，最後の部分定義域を持つあるクライアントがタイムアウトしている場合，計算終了はそのタイムアウトの検知と再割り当てのために余分な時間がかかってしまう．

これを解決するためには，割り当てられた定義域の分割とサーバへの返却を行う機構があればよい．この機構によって，サーバが分割する定義域がなくなった場合について以下の手順で終了時間を短縮できる．

プロクシが持つ計算中の定義域について，サーバは分割返却要求を出す．それを受けたプロクシは，計算済の定義域と，未計算の定義域に分割し，サーバへ返却する．次に，分割した定義域の計算済のものについてはその部分解を最終解へと統合する．計算中であった定義域については，複数のクライアント，またはプロクシに再分割/再配分することによって重複して計算させ，最も速く解を返してきたものを採用し，計算を終了する．

### 3.4 クライアントに対するプロクシの自動配分

計算参加者は，明示的にサーバのかわりにプロクシを指定してクライアントプログラムを実行することもできる．また，特に指定がなければ，起動後の要求はサーバに対して行うように設定されている．

計算参加者に対して，クライアントプログラム起動時にプロクシの指定を求める場合，計算参加者は以下の点について疑問を持つだろう．

- なぜ直接サーバではなく，プロクシに接続しなければいけないか
- プロクシとしてはどのようなホストがあるのか
- 負荷分散のためにプロクシに接続するとした場合，どのプロクシに接続すれば適切なのか

以上の点をそれぞれの計算参加者に考えてもらうことは，計算参加者に対する負担となってしまう．

そこで，計算参加者に接続するプロクシを判断してもらうのではなく，システム側でクライアントに対するプロクシの自動配分を行うことができれば，計算参加者に負担はかからない．

この，プロクシの自動配分機能として以下の機構を持たせる．まず，(1) サーバは，サーバのサイトにあるプロクシ，または，サーバ管理者が信頼できると考えるサイト上で動作するプロクシを信頼する．次に，(2) サーバは，クライアントから来た接続要求に対して，信頼できる適切なプロクシを紹介する．

このとき，適切なプロクシを選択する方法としては，

- ランダムにプロクシを紹介する
- 信頼できるプロクシからの負荷や担当クライアント数情報を元に，最も負荷の低いプロクシを紹介する
- ネットワークの中でクライアントに最も近いプロクシを紹介する

等が考えられる．

## 4. データベース化

大規模分散計算では，計算参加者にいかにモチベーションを維持して長く計算に参加してもらうかという課題がある．

分散計算の実例である RC5 や SETI@home などのシステムでは，Web ランキングシステムを用いることで計算参加者の競争心を煽り，モチベーションを維持している．

このように，ランキングシステムは重要であり，本システムでもランキングシステムやその他，計算状況を逐一表示していく機構が必要である．上記のような，計算状況を外部へ報告する機構を構築するためには，WDC システムとしては詳細な計算状況を逐一データベースへ送るようにすればよい．

Web ランキング等，表示側のシステムはこのデータを使用し，様々な形態での情報提供を行えるようになる．

## 5. WDC-HTTP ブリッジ

現在，インターネットでは多くのサイトがファイアウォールを採用している．そのようなサイトでは，外部への接続手段は電子メール，WWW，SecureShell 等に限られてしまい，他の独自プロトコルでの通信は難しい場合がある．

本システムのような独自のプロトコルを使用するシステムでは，ファイアウォール上を通過できるプロト

コルを限定されてしまうと外部への通信が不可能であり，ファイアウォール内にある計算機の資源を提供してもらうことが不可能となる．

しかし，WDC システムでは接続要求はすべてクライアントから出され，サーバから接続要求を行うことは無い．つまり，ファイアウォール内の計算機資源を提供しようと考えた場合，ファイアウォール内からファイアウォール外へと接続することはあっても，ファイアウォール外からファイアウォール内へと接続を要求することはないということになる．

もし，ここで他のフリーソフトウェア同様，サイト管理者の検査後，許可を得た上で，ファイアウォール内からファイアウォール外へと HTTP により通信することができれば，ファイアウォール内の計算機も分散計算へと参加できる．

そこで，WDC-HTTP ブリッジ機構を設ける．WDC-HTTP ブリッジは，ファイアウォールを越えて通信が行われるとき HTTP へ変換することで，ファイアウォール内外の WDC 間の通信を行う．WDC-HTTP ブリッジの概念を図 4 に示す．

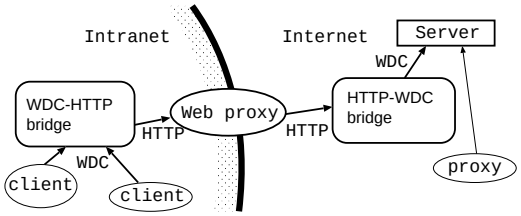


**図 4** WDC-HTTP ブリッジ

まず，ファイアウォール内に WDC プロトコルを HTTP に変換する WDC-HTTP ブリッジを用意する．ファイアウォール内のクライアントはすべてその WDC-HTTP ブリッジに対し要求を行う．要求を出された WDC-HTTP ブリッジはその要求を HTTP に変換し，Web プロクシを通して，ファイアウォール外に存在する HTTP-WDC ブリッジへと通信する．

HTTP-WDC ブリッジは，HTTP で通信された内容を WDC プロトコルに変換する．変換後は普通のプロクシ同様サーバへ要求を出す．

HTTP-WDC ブリッジは HTTP で要求を受け付ける Web サーバとして動作し，HTTP での要求をすべて処理する．

## 6. クライアントコンテナ

WDC システムは，広域分散計算用途のみではなく，ローカルサイト内の多くの計算機を用いた分散計算にも使用できる．また，ユーザは WDC プログラムを書いている場合のデバッグ手法として，まずローカルサイト内で比較的小さな計算を行い検証する場合もあると考えられる．

このような場合には，比較的小規模に繰り返し WDC での計算が行われることになる．WDC システムでは，各分散計算毎にクライアントをダウンロードして実行する必要があり，すべての計算機に対して毎回この操作を行うと手間が大きい．

そこで，計算毎のクライアントの配布を自動化する機構として，クライアントコンテナを提案する．

ユーザは，一度クライアントコンテナを全計算機上で起動しておく．すると，そのクライアントコンテナが指定のクライアントを自動的にダウンロードし，実行する．クライアントの実行はサーバが発行する停止命令により停止することもでき，また計算終了/停止後，次の命令によりクライアントの再ダウンロード/再実行を行う．

このようにすることで各計算毎のクライアントのダウンロードが自動化され，ユーザの手間が減ることになる．

クライアントコンテナの概念図を図 5 に示す．

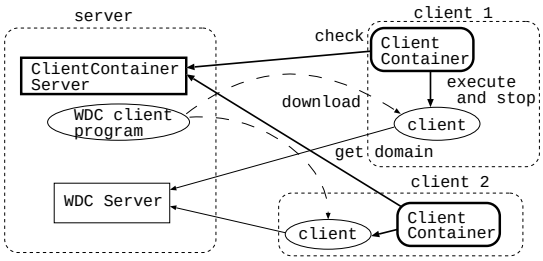


**図 5** クライアントコンテナ

## 7. ベースクラス

WDC システムでは，ユーザはベースクラスを継承して記述することにより簡単なプログラミングで分散計算を実現できる．このベースクラスをより使いやすいものにすることによって，より簡単に使用できるシステムとなる．

そのベースクラスの構成としては，まず，扱うデータの性質から分類を行う基本クラスと，その基本クラスから派生し，問題別に分類された応用クラスとからなる．

その基本クラスの設計について説明する．

### 7.1 定 義 域

基本クラスは，定義域，計算，解の 3 つに分かれ，

それぞれにおいて性質の異なる問題を想定することができる．

基本的に定義域で考えられるデータの性質としては，あらかじめ決められた範囲内やパラメータを調査する方法，そして，ファイル等外部入力から得られるデータを分割配布する方法が考えられる．

#### 7.1.1 linear クラス

本システムの対象とする問題の中で，特にパラメトリックコンピューティングと呼ばれる分野が最も使われる分野の一つである．すべてのパラメータについてシミュレーションを行い，その結果を調査する．

その場合に最も簡易に実現できるのが整数へとマップできる問題である．定義域からはユーザが設定した最大値までの整数を部分定義域として配布する．ユーザは calc クラス内でその整数に対応した計算を行う．

#### 7.1.2 n 次元クラス

問題の性質が n 次元であらわすことができる問題である場合に使用できる．このクラスは 2 次元，3 次元空間についての問題を自然に表すことができる．その他，様々なパラメータを持つ問題の場合，それぞれのパラメータについて整数で表すことができるならば，各次元の最大値を各パラメータの最大値とすることにより，すべてのパラメータの組み合わせについての計算を行うこともできる．

#### 7.1.3 ファイルクラス

計算内容が入力ファイルによって指定される計算の場合に用いることができる．

要求された部分定義域の大きさに対応する分のデータを入力ファイルから読み出し，それをクライアントやプロクシへ渡す．

ログファイルの解析，実験データの解析等に用いることができる．

### 7.2 解

解は問題の性質よって変化するが，その性質毎にあったベースクラスを用意する．

#### 7.2.1 notice クラス

まず，ある特定の解を探すという場合について notice クラスにおいてサポートする．notice クラスにおいて解が見つかった場合，即座にサーバへと伝えられ全体の計算が終了する．

#### 7.2.2 all クラス

すべての計算を行い，そのすべての解を出力する必要のある場合には all クラスを使用する．

all クラスでは，計算された部分定義域に対応するすべての部分解を集める．

#### 7.2.3 filter クラス

出力としてある一定の条件を越えた解など，フィルタにより出力/非出力を決定する問題には filter クラスを使用する．

filter クラスでは，一定の値以上の結果や最大値，最小値等，ユーザの記述するフィルタによって出力/非出力を決定し，フィルタによって残された解についてのみ出力を行う．

#### 7.2.4 ソートクラス

WDC システムでは解が返却されてくる順序は保証されていない．その方式で問題のない計算の場合は上記のクラスを使用できるが，定義域の並びと同じ順序で出力する必要のあるクラスについてはこのソートクラスを使用する．

## 8. おわりに

本論文では，WDC システムについてよりよいシステムへと改良する以下の提案を行った．

- より大規模に計算するためのプロクシシステム
- モチベーションを高めるためのデータベース化
- より多くの人に参加してもらうための HTTP-WDC ブリッジ
- ローカルでの使用環境をよりよくするクライアントコンテナ
- そして，ユーザが簡単にプログラムを書くための基本クラス

今後，この提案について実装，評価を行うことでその有用性を確認したい．

## 参考文献


1) David McNett: “distributed.net Project RC5”, http://www.distributed.net/.
2) David P. Anderson: “SETI@home”, http://setiathome.ssl.berkeley.edu/
3) Hiromitsu Takagi, Satoshi Matsuoka, Hidemoto Nakada, Satoshi Sekiguchi, Mitsuhisa Satoh, Umpei Nagashima: “Ninflet: a Migratable Parallel Objects Framework using Java”, *ACM 1998 Workshop on Java for High-Performance Network Computing*, pp.151-159, 1998.
4) Luis F. G. Sarmenta: “Bayanihan: Web-Based Volunteer Computing Using Java”, *Proc. of the 2nd International Conference on World-Wide Computing and its Applications (WWCA'98)*, 1998.
5) 田代友成 大野和彦 中島浩: “分散計算システム WDC の設計と実装”, 並列処理シンポジウム *JSPP2001 pp. 271-278*, 2001.