ハイパフォーマンス
コンピューティング 82−25
（2000. 8. 4）

分散計算システム WDC の設計と実装

田代友成† 大野和彦† 中島 浩†

TCP/IP 接続された多種多様で多数のコンピュータを対象として、クライアントプログラムをダウンロード、実行するだけで計算に参加できる分散計算システムが最近注目されている。しかし、既存の多くのシステムは、一つの問題に特化した設計であり、応用性・汎用性が高くない。

そこで本研究の分散計算システム WDC (Widely Distributed Computation) ではより汎用性に重点をおいたシステム設計を行った。計算を行うユーザは、問題特有の計算クラスを定義し、システムクラスとリンクすることにより、希望の計算を行うクライアントサーバ型のプログラムを得られる。本研究では WDC のプロトタイプを作成し、同性能 PC 16 台の LAN 環境と、性能の異なる PC 5 台の LAN/WAN 環境による評価を行った。その結果、ホモジニアス環境ではほぼリニアの台数効果が得られることと、ヘテロジニアス環境では計算能力の違いによって適切な負荷調整がなされていることが確認できた。

Design and implementation of the Distributed Computation System WDC

Tomoshige Tashiro,† Kazuhiko Ohno† and Hiroshi Nakashima†

Recently, free-join style distributed computation is so attractive that various systems based on this paradigm have been developed. These systems, however, are too specialized for their own applications to use as general purpose frameworks.

Thus, we propose a widely applicable computation system WDC (Widely Distributed Computation). Users of WDC are only required to program three classes for the essential computation part of their problems. These classes are linked with system classes for distributed computation to obtain a client/server type program.

A prototype of WDC has been implemented and evaluated with homogeneous LAN connected 16 PCs and heterogeneous LAN/WAN connected 5 PCs. The result with three small application shows that WDC works efficiently with respect to speed-up in homogeneous environment and adaptive load distribution in heterogeneous environment.

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。例えば実際に、暗号解析[1]、π の計算[2]、地球外生命体探査[3] などの分散計算システムが稼働している。

しかし、これらのシステムでは、それぞれの問題に特化した設計がなされ、応用性、汎用性が高くない。

そこで本研究では、広範囲の問題を対象とする随時参加型の分散計算システムを提案し、その実装、評価を行う。

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

2. 分散計算システム WDC の特徴

本研究では分散計算システム WDC (Widely Distributed Computation) を提案する。WDC は、TCP/IP で接続された多数の計算機を利用して分散計算を行うシステムである。このシステムは、以下の 3 つの方針に基づいて設計している。

2.1 随時参加型

計算への参加希望者は、誰でも計算に参加できる。全体として見た場合、計算に参加者が多く参加することにより、より高い計算性能を得ることができる。

また、インターネット上にあるコンピュータの利用状況は千差万別である。そして、それぞれの参加者毎に、計算機使用状況や、計算参加を決める時点などの違いがある。これら様々な状況にある参加者が提供する計算能力を、余すこと無く使うためには、

- 自由な時間に計算に参加できること
- 自由な時間に計算への参加を中止できること

- 計算が計算参加者のコンピュータ使用を妨げるなど参加者に不満を持たせないこと

が必要になる。そこでWDCでは計算の開始時点での参加要求など、計算機ユーザへの時間的な要求をせず、いつの時間でも参加でき、いつの時間でも参加を中止でき、しかも、個々の計算機ユーザに計算に参加していることをなるべく意識させない設計にしている。これにより、計算機ユーザは気軽に計算に参加できる。

2.2 問題適用範囲の汎用性

これまでのシステムでは、分散計算システム毎に、通信、計算、参加計算機の管理、計算状況の把握など、それぞれについてシステム全体を設計する必要があった。しかし、これらのシステムの計算に関わらない部分は共通化できると我々は考えた。そこでWDCでは特定の一つの問題を対象とせず、より一般的な計算モデルを用意した。その計算モデルに合致する計算であれば、具体的な計算の内容を記述するだけで分散計算を行うことができる。通信や、参加計算機の管理など計算以外の部分はWDCが用意する。これにより、分散計算をさせたいと思う者は、WDCを使用することで、計算に関係のない部分の記述に煩わされること無く、簡単に高度な分散計算を実現できる。

2.3 能力に応じた計算量配分

性能の高い計算機には計算量を多く、性能の低い計算機には計算量を少なく分配するといった、計算機の性能に応じた計算量配分が、計算を円滑に進める上で重要な点となる。しかし、クライアントの実効性能は実際に実行してみなければ分からない。WDCでは、この実行時における自動負荷調整機能を実装し、計算参加者が提供する計算能力を余すところなく利用する。

3. 設　　計

本章では、WDCの設計を説明する。

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。

3.1 適用対象

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。

例えば、$y = f(x), \{0 < x \leq X_N\}$ の計算を行う場合、定義域 $\{0 < x \leq X_N\}$ を、部分定義域 $\{0 < x_0 \leq X_0\}, \{X_0 < x_1 \leq X_1\}, \ldots, \{X_{N-1} < x_n \leq X_N\}$ に分割し、それぞれの部分定義域に対して部分計算 $y_n = f(x_n)\{X_{n-1} < x_n \leq X_n\}$ を行う。

計算モデルを図1に示す。

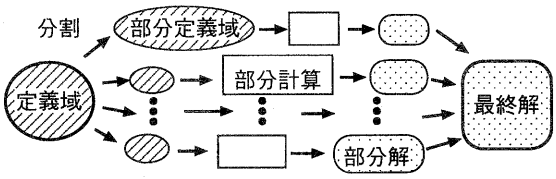

図1 計算モデル

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。

3.2 ユーザから見た実行モデル

ユーザから見た場合の実行モデルを図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。

3.2.1 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、(3.3節で説明する) 終了条件が来るまで行う。

3.2.2 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、次の動作を計算が終了するまで繰り返す。

(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。

この繰返しにより、定義域から徐々に部分解を得てゆく。こうして、計算が進行する。

3.3 計算終了条件

計算を終了させる条件には、以下の2つのうちいず

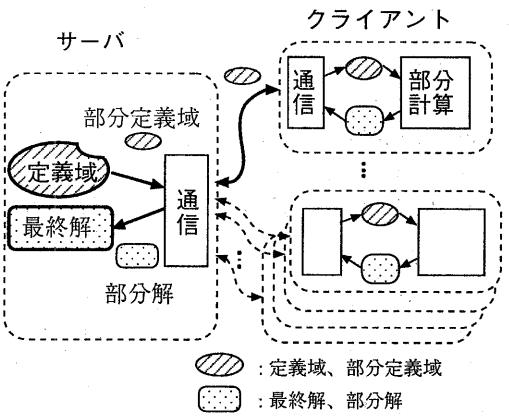

図 2 ユーザから見た実行モデル

れかを使用することができる。

- 定義域が全て分割され、さらに分割した部分定義域が各々計算され、その全ての部分解がサーバに返却された場合に終了する。これは、定義域全体から解を求めるタイプの問題に使用できる。
- クライアントから返却された、ある部分解が、計算自体を停止する条件を持っている場合、または、ある部分解を最終解に統合した時点で計算を停止してよい条件が揃った場合に終了する。これは、ある条件の部分解が見つかり次第停止してよい問題等に使用できる。

このどちらかの条件が満たされたとき計算は終了し、ユーザに結果が伝えられる。

例えば、$y = f(x), \{0 < x \leq X_N\}$ の計算を行い y の最大値を求める問題の場合、定義域全域において $y = f(x)$ を計算するタイプの問題になり、総当たりの暗号化により暗号を解読する問題の場合、1 つの答が見つかり次第計算を終了するタイプの問題となる。

3.4 実　　装

WDC の実装には C++ を用いた。前節での、サーバ、クライアント、通信、定義域、解などはすべてクラスとして実装される。

また、通信には TCP/IP を用い実装している。

3.5 WDC でのプログラミング

ユーザは、3.2 節で述べた動作の中で、個々の計算問題に依存した部分をプログラミングすればよい。つまり、ベースクラスから派生させた問題依存部の C++ クラスを書くだけで、分散計算サーバ/クライアントプログラムを作成することができる。

ユーザが記述するクラスは、以下の 3 つである。

- Domain (定義域)
- Calc (計算)
- Result (解)

それぞれのクラスについて説明する。

3.5.1 Domain クラス (定義域)

Domain クラスには、InitialState(), Divide(), IsEmpty(), Send(), Recv(), GetValue() の 6 つのメソッドを実装する。

- InitialState() 初期状態への設定
  サーバで計算を開始するときに、定義域を初期状態に設定する。
- Divide() 分割
  定義域オブジェクトから部分定義域を分割する。基本的に、サーバ上にある定義域が分割の対象になるが、部分定義域がさらに分割されることもある。そこで、任意の大きさの定義域を、任意の大きさに分割できるように定義する。
- IsEmpty() 空判断
  終了判定を行うために、分割後の定義域が空かどうかを判断する。
- Send(), Recv() 送信、受信
  サーバがクライアントと通信しているときに、定義域自体の送受信を行う。送受信では既に通信可能な状態の通信路がファイルストリームポインタの形で提供されている。つまり、Domain オブジェクトの移送に必要な定義域のメンバを、標準入出力関数を使って送受信するだけでよい。
- GetValue() Calc クラスとのインターフェース
  詳細は Calc クラスの節で述べるが、計算を行うときに、Calc オブジェクトが呼び出すメソッド。それぞれの計算の形態により自由にメソッドを定義できる。

3.5.2 Calc クラス (計算)

Calc クラスには、定義域オブジェクトから GetValue() を通して要素を取り出し、計算を行い、SetValue() を通して解オブジェクトに値をセットする機能を実装する。

適用する計算問題それぞれについて、定義域から計算するための要素を取り出す操作と、計算した値を解として設定する操作は各々異なる。そこで、WDC では特にこの授受形式を定めず、Domain、Result クラスに問題固有の GetValue(), SetValue() を定義し、Calc クラスでは計算時にそのメソッドを呼び出す。

3.5.3 Result クラス (解)

Result クラスには、Unify(), Send(), Recv(), SetValue(), OutputResult() の 5 つのメソッドを実装する。

- Unify() 解の統合
  解の統合とは、サーバに最終解を求めるために用意された解クラスと、計算されて返却された解クラスの 2 つの解を統合するために使われる。例えば、計算値の最大値を求める問題の場合、2 つの解クラスの値のうち大きい方の値を保存する。
- Send(), Recv() 送信、受信
  サーバがクライアントと通信しているときに、解クラス自体の送受信を行う。定義域の送受信同様、既に通信可能状態の通信路がファイルストリームポインタの形で提供されている。つまり、メソッド中はそれを使用し、解クラスのメンバを送受信するだけでよい。
- SetValue() Calc クラスとのインターフェース
  計算を行っているときに、Calc オブジェクトが呼び出すメソッド。それぞれの計算の形態により自由にメソッドを定義できる。
- OutputResult() 最終解の出力
  終了時等にサーバから呼ばれ、求められた最終解を、ユーザの望む形式で出力する。例えば、計算結果をファイルに出力する。

3.6 計算量の調整

WDC では、さまざまな性能のコンピュータに適切な計算量を配分するため、計算量の調整を行っている。一定時間の間に行うことができた計算量によって、適切な計算量を推測する。

3.6.1 理想計算時間

このシステムでは理想計算時間を設定し、その時間内に 1 回通信する事を理想としてスケジューリングを行う。

この値を増加させると、サーバ、クライアント間の通信量が減り、大量のクライアントと通信しても耐え得るようになる。また、この値を減少させると、サーバと頻繁に通信することになり、計算終了時に、より迅速に全てのクライアントの計算を終了させる事が可能となる。この関係はトレードオフであり、ユーザはそれぞれの要求に合わせて値を設定できる。

理想計算時間を短くするメリットとして、以下のような点が挙げられる。

- クライアントの信頼性が悪い場合、頻繁にサーバと通信させることでクライアントが異常終了した場合でも損失がより小さくなる☆。
- 全てのクライアントが短い間にサーバと通信を行うので、計算が終了したことを全てのクライアントに、より早く伝えることができる。
- 計算が終了するときの各クライアント毎の終了のばらつきが少なくなる。

理想計算時間を長くするメリットとしては、以下のような点が挙げられる。

- 通信量が減り、大量のクライアントが投入されてもネットワークトラフィックに負担がかかりにくくなる。

各クライアントの計算量は、理想計算時間と、実際に計算に必要とした時間の比により推定され、与える部分定義域のサイズを変化させることで調整される。

3.6.2 計算量調整の実装

計算量の調整について、本実装では次のような式を用いて計算している。

理想計算時間を t_i、実際に計算に必要とした時間を t_r、部分定義域のサイズを s、次の部分定義域の推定サイズを s_a とすると、計算式は次のようになる。

$$s_a = \begin{cases} 2s & t_r \leq t_i/2 \text{ のとき} \\ s(1 + \dfrac{t_i - t_r}{2t_r}) & t_i/2 < t_r < t_i \text{のとき} \\ s\dfrac{t_i}{t_r} & t_r \geq t_i \text{のとき} \end{cases}$$

これは、部分定義域サイズを、

- $t_r \leq t_i/2$ の場合は 2 倍
- $t_i/2 < t_r < t_i$ の場合は、理想計算時間と次の計算時間との差が、今回の計算時間との差の 1/2
- $t_r \geq t_i$ の場合は、理想計算時間

となるように設定しているものである。

4. 評　　価

本研究では、WDC システムを用い、実際に評価用プログラムを動作させ、システム性能の評価を行った。

4.1 評価用プログラム

性能評価には、3 つのプログラムを用いた。それぞれのプログラムについて説明する。

- 暗号解析
  一方向ハッシュ関数 MD5 により生成されたハッシュ値を用い、そのハッシュ値から元の平文を求める。解析は、文字数が少ない順、辞書順に、全ての平文をハッシュ値へと変換し、比較する方法 (全探索) により行う。
  この評価では、求める平文を "H␣␣" とし、"␣" から "H␣␣" までの、377953 回のハッシュ化を行い、元の平文を求め、計測した。
- マンデルブロ集合
  マンデルブロ集合は、繰り返し数を変化させると

☆ クライアントが異常終了した場合、サーバはタイムアウトしたことを検知し、そのクライアントに割り当てていた部分定義域を別のクライアントへ割り当て直す。

生成される画像が変化する。そこで、その繰り返し数を変化させ、マンデルブロ集合の画像群を生成する。生成した画像はサーバに送信する。

この評価では、繰り返し回数を 512 回までとし、512 枚の画像を生成して計測した。

- モルフォロジー演算

  モルフォロジー演算[4] とは、集合演算の一種であり、ノイズのある画像からクオリティの高い復元画像を得る時などに用いられる。計算時間は他の 2 次元画像処理より計算時間が長くなる傾向がある。この評価では、640 × 512 サイズの濃淡サンプル画像に対し、パラメータを変化させて tophat 変換を施した画像 312 枚を生成する。ただし、生成した画像はサーバに収集しない。

いずれのプログラムも、定義域を分割でき、部分定義域毎に独立して計算できる問題である。

4.2 台数効果

同じ性能の計算機がクライアントとして参加すれば、参加した計算機の数だけ全体の計算は高速になるのが理想である。そこで、参加クライアント数毎の速度向上率を測定した。

性能評価には、PC (PentiumII 450MHz、128MB) 16 台を 100Mbps Switching Hub で接続した環境を用いている。

N 台 ($1 \leq N \leq 16$) の計算機を用いて、各々クライアントを同時に実行し計算時間を測定、クライアントが 1 台時の計算時間との比較をした。結果を図 3 に示す。

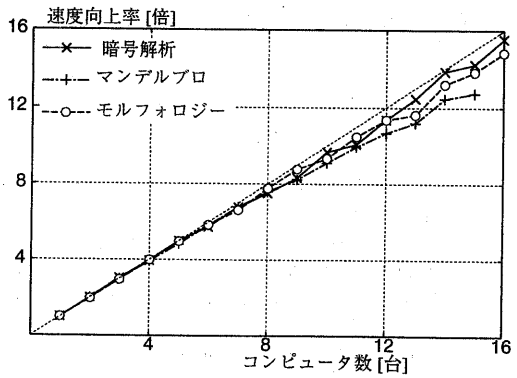

図 3 台数効果

4.3 計算量の調整

次に、計算量の調整機能の効果を評価した。

3.6 節で示したように、本システムはクライアントの能力を動的に判断し、その能力に見合った大きさの部分定義域を分配する。この、定義域の配分調整機能が機能し、クライアントの能力に応じた計算量配分がなされているかを調べる。

- LAN 内にある性能の異なった計算機と、WAN 上にある性能の異なった計算機で評価を行う。計算機とその接続形態については表 1 に示す。
- 表 1 に示した計算機上で同時にクライアントを実行し、計算を行い、各クライアントの処理量を測定する。
- 各クライアントについて、処理量 / 性能 を求める。この値が 1.0 に近いほど、適切な計算量が配分されたことになる。

この測定の結果を図 4 に示す。

表 1 評価用計算機

host	machine	distance from server
A	PentiumII 450MHz	LAN(100Mbps Ethernet)
B	Athlon 750MHz	LAN(10Mbps Ethernet)
C	PentiumII 266MHz	LAN(10Mbps Ethernet)
D	PentiumII 300MHz	WAN (TUT から木更津高専へ)
E	PentiumII 400MHz	WAN (TUT から熊本電波高専へ)

* TUT : 豊橋技術科学大学

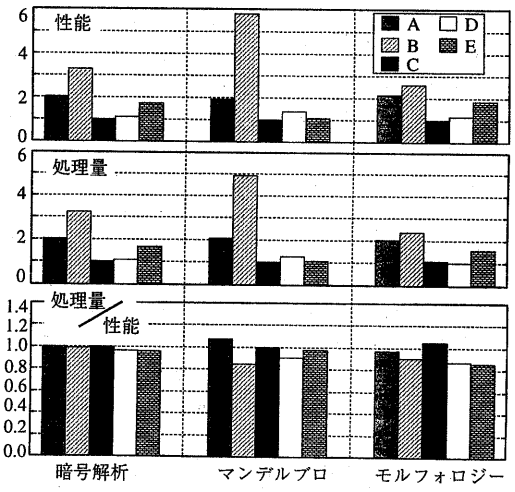

図 4 計算量の調整

4.4 考　　察

4.4.1 台数効果

性能評価の結果図から分かるように、ほぼ理想的な台数効果が得られ、ほぼ理想に近い計算量配分が行われていることが分かる。

この結果より、計算に参加する計算機が多くなれば

多くなるほど、全体の計算能力はその参加計算機数に比例して向上することが分かる。

4.4.2 計算量の調整

図4から分かるように、すべての結果について、処理量 / 性能が0.8以上、1.1以下に収まっている。つまり、各プログラムについて、各計算機の性能に応じた計算量が配分されたことが分かる。

この結果より、各々計算機の能力に応じた計算量を適切に配分していることが分かる。

5. 今後の方向性

今後、このWDCシステムをより実用的に、より有用にしていくために仕様面、実装面の両面を改良する。

5.1 問題の適用対象の拡大

現在、WDCは、あらかじめ与えられた定義域を元に計算を行っており、計算結果により定義域が変化する計算は対象としていない。しかし、この制限を緩和することができれば、問題の適用対象を拡大することができる。そこで、今後、定義域が計算の結果により動的に変化する計算についてもサポートできる計算モデルへとWDCを改良する。

例えば、(1) 解を求めてツリーを探索する問題における枝刈りを行う、(2) 評価関数により解の存在する確率が最も高い領域を動的に把握し、優先的に計算する、などの問題のサポートなどが考えられる。

具体的には、解クラスから、定義域クラスへの結果のフィードバック機構を設けることで、(1)、(2)とも解決すると考えている。

5.2 定義域、解のベースクラスの充実

今回の評価に用いた問題では、Domain、Calc、Resultクラスを基本ベースクラスからの継承で記述した。しかし、ほとんどの問題はカテゴライズでき、どういう種類なのかを特定することができる問題である。

そこで、問題のタイプ毎にベースクラスを用意することによって、ユーザの記述するコード量を減らすことができる。今後、このタイプ別のベースクラスの充実を図る。

5.3 異機種混在での動作のサポート

現在のプロトタイプ版では実現していないが、WDCはクライアントが様々なコンピュータ上で動作するシステムである。Windows上でクライアントが動作すれば、多くの計算参加者を見込む事ができるなど、今後、この様々なOS用のクライアントが混在して動作できる実装を行う。

5.4 プロクシ

このシステムでのプロクシサーバとは、サーバからは性能の高いクライアントとなり、クライアントからはサーバとなる、サーバとクライアントの中間的な存在のプログラムである。

このプロクシ機能を実装することにより、サーバの負担を複数のプロクシに分散でき、ネットワーク負荷を軽くすることができ、より大規模な分散計算を行うことができる。

6. おわりに

本論文では、汎用性に重点を置いた分散計算システムWDCを提案し、その特徴、有用性、設計について述べた。

また、WDCを同性能PC 16台のLAN環境と、性能の異なるPC 5台のLAN/WAN環境によって評価した。その結果、この随時参加型であるWDCのモデルが実際に機能し、ホモジニアス環境ではほぼリニアの台数効果が得られることと、ヘテロジニアス環境では計算能力の違いによって適切な負荷調整がなされ、計算を円滑に進めることができることを示した。

今後、先に示した方向について研究をすすめていくことで、より有用なシステムとしていきたい。

謝辞 遠隔の実験、検証環境を快く貸していただいた、熊本電波高専 田辺 正実教授、木更津高専 丸山 正男助手に感謝します。

参考文献

1) distributed.net Project RC5, http://www.distributed.net/.
2) Colin Percival: PiHex A distributed effort to calculate Pi. , http://www.cecm.sfu.ca/projects/pihex/
3) SETI@home, http://setiathome.ssl.berkeley.edu/
4) 小畑 秀文: モルフォロジー, コロナ社, 1996.
5) Ian Foster and Carl Kesselman: Globus, A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications, 1997*
6) Satoshi Sekiguchi, Mitsuhisa Sato, Hidemoto Nakada, and Umpei Nagashima: – Ninf –, Network base information library for globally high performance computing, *Parallel Object-Oriented Methods and Applications (POOMA)*, 1996.
7) Henri Casanova and Jack Dongarra : NetSolve, A Network Server for Solving Computational Science Problems, *Proceedings of Super Computing '96*, 1996,