2004－HPC－99　(23)
2004／7／31

P2P 分散システム XtremWeb 上での Grid RPC システム OmniRPC の設計

中島佳宏[†1]　佐藤三久[†2]　朴泰祐[†2]
高橋大介[†2]　Samir Djilali[†3]　Franck Cappello [†3,†4]

OmniRPC はグリッド環境での並列プログラミングのための Grid RPC システムである．本稿では，OmniRPC の対象とする計算資源を，遠隔地にあるサーバやクラスタだけではなくオフィスや自宅にある個人用 PC まで広げるために，Peer to Peer 環境の分散計算システム上で遠隔手続き呼び出しを行うシステムについて提案する．このシステムでは，OmniRPC クライアントと P2P 分散システムの間にブリッジを行う Agent にて遠隔呼び出しをジョブに変換し，P2P システムに投入する．また，リモート実行プログラムとの入出力にファイルを用いて処理を行う方式に変更する．これにより，P2P システムによる大規模な計算資源と耐故障性を利用することができる．本稿では，その P2P システムとして XtremWeb を用いて，P2P 環境対応の OmniRPC を構築したシステム OmniRPC/XW の設計について報告する．

Design of Grid RPC System OmniRPC on XtremWeb P2P Grid

Yoshihiro Nakajima,[†1] Mitsuhisa Sato,[†1] Taisuke Boku,[†1]
Daisuke Takahashi,[†1] Samir Djilali[†3] and Franck Cappello [†3,†4]

OmniRPC is a Grid RPC system for parallel programming in a grid environment. In this paper, we describe a design of remote procedure call system for Peer to Peer distributed computing environment in order to exploit not only remote servers and clusters but also PCs in office or home. In proposed system, OmniRPC agent works as bridge between P2P distributed computing system and OmniRPC client to convert remote procedure calls into remote jobs submitted into P2P system. Input and output between remote program is implemented via files in submitted jobs. This design allows us to exploit a large amount of computational resources and fault tolerance feature provided by P2P system. We presents the design of OmniRPC/XW, which is one of the P2P-enabled versions of OmniRPC for P2P Grid System, XtremWeb.

1. はじめに

インターネットをはじめとする広域ネットワークの進歩により，広域ネットワーク上の計算機資源やデータの共有，並列分散コンピューティングの支援を可能にするグリッド技術が注目されている．

本稿では，サーバやクラスタ計算機などの常設された計算機以外に個人用 PC 等の計算機も活用すべく，Peer to Peer (P2P) 分散システム XtremWeb 上に Grid RPC システム OmniRPC による並列プログラミング環境を構築するための設計について述べる．

我々は，これまでグリッド環境上における複数の計算資源を遠隔手続き呼び出し (Remote Procedure Call: RPC) を用いて，簡単に並列分散プログラミングを行うためのミドルウエア OmniRPC[9] の開発を行ってきた．Grid 向け RPC は，グリッドにおける標準的なプログラミングモデルの一つとして有望であり，特にパラメータサーチアプリケーションやタスク並列アプリケーションに対する有効なプログラミングモデルとして注目されている．

近年，コモディティPC の普及とネットワークの技術革新により，個人がネットワークに接続された計算機を複数保有するようになった．しかし現在，オフィスや家庭にある PC において，その計算能力の高性能さに関わらず，使用されるアプリケーションはオフィス文章や表計算など比較的計算量を必要としないものが多く，PC の計算能力は有効に活用されていない．

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
†3 Laboratoire de Recherche en Informatique, Université Paris Sud, France
†4 INRIA, France

一方で， 企業などで，製薬のための分子構造設計や電子回路設計のシミュレーションなど，莫大な量の計算を迅速に処理するニーズが非常に高くなっている．個人 PC 等を含めれば潜在的には数万台に達する計算能力を集結し，かつ，一つの目的のために計算させることが可能になれば，ユーザは非常に低いコスト巨大な計算能力が手に入り，大規模計算を必要とする研究開発を促進する強力なツールが実現する．

そこで，サーバもしくは遠隔地にあるクラスタをグリッドでの計算のプラットフォームとして設計した OmniRPC を，個人用 PC を含む Volatile な計算資源を利用する P2P 環境に対応させる．提案するシステムの目的は，OmniRPC により，P2P 環境に対して有効な並列プログラミング環境を与えることである．また提案するシステムは，P2P 環境をサポートするジョブ実行ミドルウエアをバックエンドのシステムとして活用し，その上に OmniRPC を実装することによって実現する．本稿ではその計算実行のミドルウエアとして XtremWeb を使用し，その上に OmniRPC を構築する．

2 章で OmniRPC の概要を述べ，4 章で P2P 環境向けのジョブバッチシステム XtremWeb の概要について述べる．5 章で OmniRPC の XtremWeb 上での設計について述べ，6 章で関連研究を述べ，終章でまとめと今後の課題をあげる．

2. OmniRPC システム

OmniRPC は，クラスタ環境から広域ネットワークで構成されたグリッド環境までシームレスな並列プログラミングを可能にする Grid RPC システムである．また，OmniRPC は，マスター/ワーカ型の並列プログラミングをプログラミングモデルとしてサポートしている．OmniRPC が想定しているグリッド環境は，複数の計算機クラスタがインターネットに接続され，それらのクラスタを相互利用する環境である．さらに OmniRPC は，クラスタのマスターノードだけがグローバル IP を持ち，スレーブノードはプライベートアドレスを用いて構成されている，現在のクラスタ環境に多くみられるような構成も計算機資源として利用可能である．

OmniRPC の API は，基本的に Ninf[5] の API を踏襲しており，さらにワーカープログラム側の計算データの状態を保持する persistency をサポートしている．この persistency をサポートした API を利用することにより，ユーザは効率的なプログラミングが可能となる．また，ユーザは，非同期呼び出しの API を用いることにより並列プログラミングを行うことができる．

OmniRPC では，一つのリモート実行プログラムは複数の関数を含むことができる．このため，ユーザは，インタフェースを記述する IDL 記述において，複数のインターフェースをまとめてモジュールとして定義する．また，グリッド環境において典型的な並列アプリケーションであるパラメータ検索などのアプリケーションを効率的にサポートするために，OmniRPC では，リモート実行モジュールの起動時に実行される初期化手続きを定義することによって，実行モジュールを起動時に自動で初期化する機能を有している．我々は，この機能を自動初期化実行モジュール機能と呼ぶ．

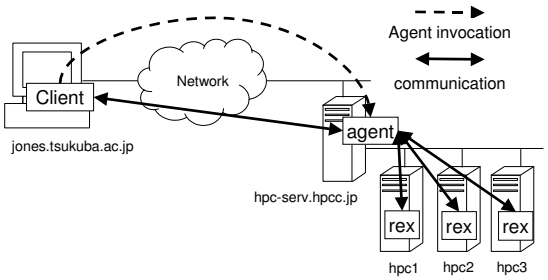

図 1 OmniRPC システムの概要

OmniRPC は，エージェントを使用してクライアントプログラムと複数のリモート実行プログラムとの通信を多重化し一つのコネクションで行うことができる．この通信の多重化を利用することにより，ユーザは，プライベートアドレスで構築されたクラスタや 1000 台規模のリモートホストを利用することができる．

Grid 環境として，OmniRPC は，Globus Toolkit[2] の他に，SSH（Secure Shell）による認証も使用できる．さらに，SSH が用いられている環境においては，ポートが制限されていたり，使用するネットワークの途中に firewall が存在することが多い．そのような環境のために，エージェントとクライアントプログラムの通信は SSH の port forwarding 機能を用いて中継させるため，ユーザは，firewall のある環境においても，リモート計算機資源を利用することができる．

OmniRPC の動作イメージを図 1 に示す．この図では，通信の多重化が使用され，クラスタの計算ノードに向けて RPC が行われている．OmniRPC においては，クライアントプログラムが起動されたときに，Agent は OmniRPC のホストファイルで指定されたホスト上において起動される．次に，クライアントプログラムの RPC 呼び出しに対して，Agent が計算ノードにあるリモート実行プログラムを起動する．

3. P2P 環境での OmniRPC システムの設計

3.1 P2P 分散計算システム

本研究が対象とする P2P 環境は，XtremWeb や BOINC[1] 等の P2P 分散計算システムである．これらのシステムでは，計算資源となる計算機においてワーカプログラム (デーモン) を立ち上げ，これらのワーカープログラムが 1 つまたは複数のサーバに接続することによって，システムを構成する．それぞれのワーカプログラムは，http などの通常良く用いられるプロトコルを用いて，ワーカ側から接続するため，ワー

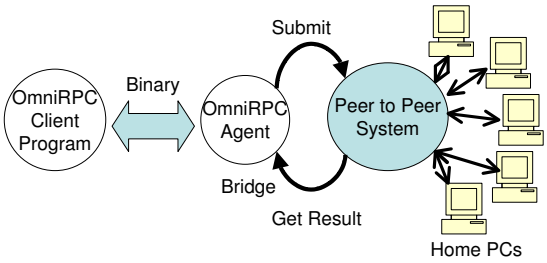

図 2 P2P 環境での OmniRPC システムの概略

カーが DCHP で割当られる動的なアドレスを持ち，firewall の内部でもよいことになる．

これらのシステムでの基本的な単位はジョブであり，ユーザによりサブミットされたジョブがシステムに参加している計算資源に適宜割り当てられ実行される．個人用 PC を含む計算資源上で実行されているワーカが，それぞれのワーカープログラムがサーバにジョブを取りに行く形態，いわゆる Pull モデルがとられる．

P2P 環境において，計算資源は Volatile であること，すなわち，計算に参加する計算機は故障はもちろん，計算資源のオーナーの都合による中断，停止が行われることを前提となる．このような場合には，システムがノードの停止を検知し，再スケジューリングを行い，他の計算資源に割り当てる機能を持つ．

3.2 P2P 分散計算システム上での OmniRPC システム

これまでの OmniRPC は基本的に，クライアントがサーバにある計算機にワーカを起動、接続し，データをサーバに転送・計算を行うシステムであり，Push モデルのシステムであるといえる．計算中には、クライアントとワーカはコネクションを維持し，このコネクションを使って計算の依頼，結果の転送を行っている．

RPC を前節で述べた P2P 分散計算システム上で行うためには，コネクションにより行っていた計算の依頼，結果の転送を，ファイルを介して行う．すなわちドキュメントベースの通信により行うことが考えられる．あるリモート関数の遠隔呼出を行う場合には，その入力をファイルに格納し，リモートの関数の実行プログラムとともにジョブとしてサブミットする．実行プログラムがワーカにより実行され，その結果がファイルに出力され，それを結果として返すことにより，呼出が完了する。

OmniRPC システムでは Agent を用いることにより，遠隔の実行モジュールの起動，レジストリの管理，ワーカとの通信の中継を行わせることができる．これを用いて，現状の OmniRPC のシステムを大幅に変更することなく P2P 分散計算システムに対応するために，Agent に P2P システムとのブリッジを行う機能を付けることにより実現することにした．システムの概略図を**図 2** に示す．Agent が、リモート関数の呼出の引数を入力ファイルとしリモート実行プログラムとともにジョブとして P2P システムにジョブをサブミットし，終了後，出力ファイルを結果として返す．

もう一つの方法として，クライアントとワーカの間の通信を，サーバを介して P2P 環境の仮想的なコネクションを提供することによって，これまで OmniRPC で用いたコネクションベースの遠隔呼出を行う方法が考えられる．しかし、この場合，ワーカ側の障害に対する処理をクライアント側が行わなくてはならず，システムが複雑となる可能性がある．遠隔呼出を１つのジョブとして，切り離すことにより，ジョブの実行保証を P2P システムの耐故障機能に任せることができ，システムが簡単となる．

3.3 P2P 分散計算システム利用の利点・欠点

これまで，遠隔のサーバやクラスタ計算機を活用するために作成された OmniRPC を P2P 環境に対応させることは，アプリケーション作成者にとって常設されたサーバやクラスタ計算機だけでなく個人用の PC を活用することができ，大規模な計算資源を利用が可能になる．

ただし，P2P システムを計算資源として用いる場合，従来の OmniRPC システムで用いていたコネクションを用いるのに比べ，断続的な通信かつ，ドキュメントベースで処理を行うため，通信性能，ジョブ起動のオーバーヘッドによって性能が劣ることは避けられないと思われる．しかし，これはアプリケーションの通信と計算の比率に依存する問題であり，粗粒度の並列性のあるアプリケーションであれば十分な性能が得られる．このようなアプリケーションに対し，従来のグリッド環境に加え，P2P 分散システムによる巨大な計算資源が利用できることは十分な意義があると考える．

4. XtremWeb

XtremWeb は，インターネット上での大規模分散処理を目的としたミドルウエアである[3]．XtremWeb は，BOINC や Condor[8] などの分散システムプラットフォームと同様に，インターネットに接続された遠隔計算資源 (PC, Workstation, Server) や LAN 内部にある計算資源プールを活用する．

XtremWeb は現在のグローバルコンピューティングの以下２つの問題点を解決するべく設計されている．

- グローバルコンピューティングアプリケーションのための汎用的なツール・サービスが無い．
- 各プロジェクトごとに全く別のアプリケーションとボランティアコンピュータを使用している．

そのため XtremWeb では以下の目標としたフレームワークを提供する ．

- 膨大な量のノードを活用するべく高いスケーラビリティ
- ヘテロジーニアスな計算機環境を考慮
- コンピュータリソースのオーナーはリソース制限などのポリシーを定義可能
- Fault tolerance に対応

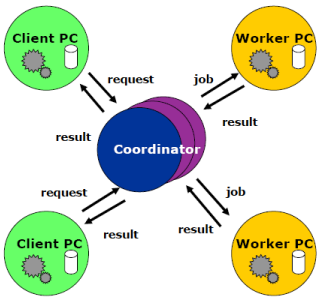

図 3 XtremWeb の概要

- 悪意のある操作や改竄などを防ぐ (Software Sandbox)
- 使うのが簡単

XtremWeb の概要を**図 3** に示す．XtremWeb は，Worker，Coodinator, Client の 3 つから構成される．

4.1 Worker

Worker は，XtremWeb における計算実行のためにボランティア計算機の計算能力の提供，サーバーによって提供されたアプリケーションの実行と，主に 2 つの役割を果たす．計算機資源を提供しているユーザを重視し，XtremWeb はユーザ自身が計算機資源の提供ポリシーの設定や悪意のあるプログラムからの防御，セキュリティーを考慮したアーキテクチャ構成となっている．Worker プログラムが動作している計算機資源の仮想化について，XtremWeb では，計算資源提供ポリシーの設定を使用できる計算資源の最大値を設定することで行う．ここで，Worker プログラムが Server に問い合わせにいくため, NAT や Firewall があるような環境においても本システムは動作可能になっている．

4.2 Server (Coodinator)

XtremWeb のサーバはアプリケーション (アーキテクチャごとにコンパイルされた実行ファイル) やクライアントから投入されたタスクのホスティングを行う．また，計算を行う Worker にそれらのタスクやアプリケーションを提供する．XtremWeb はパラメータサーチアプリケーションに重点をおいており，アプリケーションの Worker への配布は，異なるプラットフォームごとにコンパイルされたバイナリを用意し，アプリケーションの名前，概要，出力結果について定義も行う．また，Worker のアップグレードはシステムが自動で行うようになっている．

アプリケーションへの参照やパラメータを設定することでクライアントからタスクを XtremWeb に投入する．ここで，パラメータは複数のファイルから構成されてもよい．タスクがデータベースに保存されたら，Server はユニークな *task identifier* を作成する．それ以後の操作に関して，この *task identifier* を用いて行う．

4.2.1 Scheduling

XtremWeb のスケジューラーは Dispatcher と Schedulaer の 2 つに分けることができる．

Dispatcher はタスクプールの中からタスクの群を選びそれらをスケジューラへ配布する．Dispatcher はもっとも重要なタスクを提供するアプリケーションを決める．この優先度はアプリケーションに対するタスク実行の割合を設定することで決まる．次に，Dispatcher は最後に投入されたそのアプリケーションに対するタスクを選択する．

Scheduler はタスクの完了の管理を行う．XtremWeb では FIFO によってタスクがスケジューリングされている．Worker が仕事を要求したときに，ランタイム環境やあらかじめコンパイルされたアプリケーション実行ファイルに合わせて，スケジューラはタスクを選択する． またスケジューラは Worker プログラムからのハートビートを活用して，タスクが実行中であるかどうかを検出し，もしタスクがシステム異常などによって中止されてしまった場合には再スケジューリングを行う．

Dispatcher とスケジュールポリシーは動的に変更可能である．サーバーのスループットはタスクサーバーのクラスタを利用して向上させることができる．またロードバランシングに関しては外部のツールを用いて実現することができる．

4.3 Client

Client は，実行アプリケーションの Server への登録，タスクの Server への登録・その登録したタスクへの *task identifier* の取得，タスクの実行状況の情報取得．依頼したタスクの結果取得を行うことができる．

4.4 XtremWeb の動作

XtremWeb の動作は以下の手順に従う．Client でネイティブコードの実行プログラムをあらかじめ Server に登録する．次に計算を行うデータセットを作成し，それを Client から Server にタスクを登録する．登録されたタスクに対して Server は管理を行い，Worker からの Job の割り当て要求に応じ，実行ファイルとデータセットを送る．

Worker プログラムは起動時に Server に，Worker のホスト情報などを登録する，CPU が使用されていないときに Worker は Server にタスクの割り当て要求を行い，実行ファイルと計算を行うデータを受け取る．このデータに対して計算を実行し，結果を Server に送り直す．

Client は，Server に投入したタスクの状態を問い合わせ，依頼した計算が終わっている場合には，出力結果を Server よりダウンロードする．

5. OmniRPC/XW

本章では，XtremWeb 上で OmniRPC を実現する OmniRPC/XW の設計について述べる．

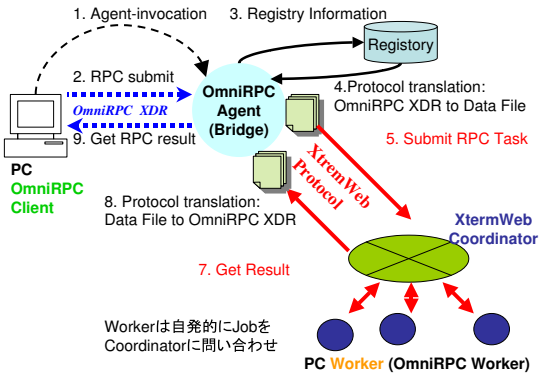

図 4 提案システムの概要

5.1 OmniRPC/XW の設計目標

遠隔手続き呼び出しで記述されたグリッド向けアプリケーションを，プログラムの変更無く，実行環境を Cluster だけではなく Volatile な個人 PC に対応させるために，バックエンドに XtermWeb を使用しそのシステム上に OmniRPC を構築する．

ここで，本システムでは以下の目標を達成する．

- P2P 環境においての並列プログラミング環境の提供
- P2P 環境による耐故障性を備えた Grid RPC システムを構築
- ソースプログラムの変更なしにクラスタ環境から Volatile な個人用 PC 環境への対応
- 実行プログラムの Deployment の自動化

以上のことを考慮し，提案するシステムでは OmniRPC に以下の拡張を行う．

- OmniRPC の Agent に XtremWeb の Server と OmniRPC のクライアントプログラム間のプロトコルブリッジを行う機能
- OmniRPC のワーカープログラムでデータ入出力をファイルから行う機能
- XtremWeb へのレジストリ情報の登録とリモート関数のインタフェース情報の登録

図 4 に本システムの概要を示す．以後，OmniRPC からの RPC に対して作成した XtremWeb のタスクを RPC タスクと呼ぶ．XtremWeb システム自身は，システム異常等により中止されたタスクを，再スケジューリングし，他の計算機で処理を行う耐故障性を備えている．そこでこの機能を利用することで，Agent によって XtremWeb に投入した RPC タスクの処理は必ず終了する．よって，Agent は単に RPC タスクが終了するまでポーリングする事で，システム全体としては耐故障性を保証することが可能になる．

5.2 OmniRPC/XW Bridge Agent

OmniRPC/XW Brige /Agent は，OmniRPC クライアントと XtremWeb のサーバ間における，コネクションベースとドキュメントベースの処理形態の差異の吸収，データのプロトコル変換，RPC タスクの XtremWeb への登録・監視・結果の取得の役割を担う．Agent の OmniRPC クライアントからの RPC に対応する動作フローは以下のようになる．

(1) OmniRPC のクライアントからリモート関数を呼び出す際の入力データに対応する XDR[7] 形式のデータを受信．
(2) 受信したデータをファイルに，XDR のデータをそのまま書き出し，リモートで実行するアプリケーションのためのファイルを作成し，それを ZIP 形式で圧縮する．
(3) RPC に対応するリモート関数を持つ実行モジュールを検索し，そのモジュールを動作させるスクリプトを作成．
(4) 作成したスクリプトとデータを含む ZIP アーカイブを XtremWeb のサーバに登録．
(5) 登録した RPC タスクに対応する *task identifier* を取得．
(6) RPC タスクが終了するまでポーリングする．終了した場合には，XtremWeb のサーバから *task identifier* を用いて処理したデータを含むファイルを圧縮した ZIP アーカイブを取得．
(7) 出力ファイルからデータ読みとり，OmniRPC のプロトコルに変換．
(8) OmniRPC のクライアントへ RPC の出力データを送信する．

5.3 ファイル入出力へのワーカプログラムの拡張

従来の OmniRPC のワーカープログラムにおいて，ネットワークのコネクションからデータ入出力を行い計算を行っていたところを，ファイルからも行えるようにした．XtremWeb の Worker での処理は，アプリケーションの実行プログラムに対するコマンドオプションを与えることにより行う．そのため，リモート関数を含む実行モジュールをコマンドオプションで，以下のように指定する．

```
% worker_module -xw -xw-in <InputDataFile> \
                   -xw-out <OutputDataFile>
```

XtremWeb 環境下で使用するオプションを指定，次に入力データを含むファイル名，出力データを書き出すファイル名を指定することにした．

5.4 XtremWeb 対応の Registory 情報

XtremWeb の Server へ OmniRPC のワーカープログラムを登録するには，従来の OmniRPC のワーカープログラムの Registory 情報を登録するプログラム *omrpc-register* を拡張したものを用いて行うようにした．また，RPC で呼ばれるリモート関数と XtremWeb の実行モジュールの対応をとるために，OmniRPC で使用しているリモート関数，モジュール名とそのモジュールへのパスが記述されているスタブファイルに，XtremWeb に登録されているバイナリパスとリモート関数のインターフェース情報を持つファイルのパスパスの項目を新しく増やした．

実行時のリモート関数の情報取得については，Agent がスタブファイルに指定されたパスにあるインタフェー

ス情報が記述されたファイルを読み込むことによって行う．

5.5 XtremWeb 対応のホストファイルの設定

XtermWeb をバックエンドで使用する際には，設定ファイルを以下のように設定する．

```
<?xml version="1.0" ?>
<OmniRpcConfig>
  <Host name="hpc-serv.hpcc.jp" >
    <Agent invoker="ssh" mxio="on" />
    <JobScheduler type="xw" maxjob="10" />
    <XWConfig path="/home/user/config" />
  </Host>
</OmniRpcConfig>
```

JobScheduler の type に xw を指定し，RPC タスクを XtremWeb の Server にキューイングする数を設定する．またホストファイルに XtremWeb を使用する際に必要なユーザ情報や Server 情報の設定ファイルのパスを *XWConfig* に指定する．

5.6 問 題 点

現在の XtremWeb の実装では，Data Persistency を用いたプログラムを作成することはできない．そのため OmniRPC/XW では，OmniRPC で開発してきた Data Persistency を用いたプログラムを実行することはできない．この点については，XtremWeb の開発チームと，XtremWeb にデータをキャッシュすることができるように Data Persistency の機能について検討を行っていく予定である．

6. 関 連 研 究

遊休 PC を対象としたグリッドコンピューティングを可能にさせる研究としては，SETI@home[6], Folding@home[4] が挙げられる．これらのプロジェクトでは，アプリケーションに特化してシステムを設計しており，ほかのアプリケーションを作成する際において，プログラミングモデルがなく，ほかのアプリケーションに適応するのが難しい．我々の研究は，これに対して汎用的なプログラミング環境を提供することを目的とする．

Condor は計算機の空き時間を利用する事を目的としたジョブスケジューリングシステムである．ジョブの割り当てはサーバが担当するため，ユーザはどの計算機を使うのかを意識することなくジョブを投入することができる．また，チェックポイントとマイグレーション機能をサポートしているため計算途中のジョブの停止，ほかの計算機への移動を行い対故障性に優れる．この Condor の上に耐故障性を重視した RPC システムを実装した Ninf-C[10] がある．我々のシステムと Ninf-C は同じようなアーキテクチャをとっている．

7. お わ り に

本稿では，P2P 環境において Volatile な計算機を Grid RPC プログラミングモデルで利用可能にするため，XtremWeb 上での OmniRPC の設計について述べた．現在，提案システムを実装中である．今後の課題としては，実装を終了させ，1000 台規模の計算機を用いた提案システムの基礎的な性能評価や実アプリケーションによる性能評価が挙げられる．その評価でのポイントは，スケーラビリティ，Agent を入れたことによる性能への影響，故障するノードが存在する場合の性能である．また，本稿では，P2P 環境向けのミドルウエアとして XtremWeb を使用したが，今後は BOINC や JXTA などの環境にも対応する事も考えている．

謝辞 本研究の一部は，科学研究費補助金特定領域研究 (2) 課題番号 14019011「計算物理学分野の Grid アプリケーションと並列プログラミングシステムの研究」，JST-ACT 「GRID テクノロジを用いた創薬プラットフォームの構築」，日仏共同研究 FJ Grid による．

参 考 文 献

1) Berkeley Open Infrastructure for Network Computing: http://boinc.berkeley.edu/.
2) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *The International Journal of Supercomputer Applications and High Performance Computing*, Vol. 11, No. 2, pp. 115–128 (1997).
3) Germain, C., Neri, V., Fedak, G. and Cappello, F.: XtremWeb: Building an Experimental Platform for Global Computing, *GRID*, pp. 91–101 (2000).
4) Larson, S. M., Snow, C. D., Shirts, M. and Pande, V.S.: Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology., *Computational Genomics* (2002).
5) Ninf Project: http://ninf.apgrid.org/.
6) SETI@home project: http://setiathome.ssl.berkeley.edu/.
7) Sun Microsystems, Inc: RFC 1014 - XDR: External Data Representation standard (1987).
8) Thain, D., Tannenbaum, T. and Livny, M.: Condor and the Grid, *Grid Computing: Making the Global Infrastructure a Reality* (Berman, F., Fox, G. and Hey, T.(eds.)), John Wiley & Sons Inc. (2002).
9) 佐藤 三久, 朴 泰祐, 高橋 大介: OmniRPC:グリッド環境での並列プログラミングのための Grid RPC システム, 情報処理学会論文誌コンピューティングシステム, Vol. Vol. 44, No. SIG11 (ACS 3), pp. 34–45 (2003).
10) 中田秀基, 田中良夫, 松岡聡, 関口智嗣: ”耐故障性を重視した RPC システム Ninf-C の設計と実装”, 先進的計算基盤システムシンポジウム (SACSIS2004) (2004).