2005－HPC－103（13）
2005／8／3

複数グリッドミドルウエア上で動作する Grid RPC システム OmniRPC の設計と実装

中島佳宏[†,††]　佐藤三久[†]　相田祥昭[†]
朴泰祐[†]　高橋大介[†]　Franck Cappello [†††]

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，従来ワーカとの直接通信でおこなっていた OmniRPC の遠隔呼び出しをドキュメントベースすることにより，遠隔のジョブとして実行できる機構を設計した．これにより，OmniRPC の対象とする計算資源を，複数のグリッドミドルウエアで管理されている計算機に拡大し，これまでバッチシステムとして利用されてきた形態を，遠隔手続き呼び出しのプログラミングモデルを用いて利用できる．それぞれのグリッドミドルウエアにおいて OmniRPC から利用するために必要な汎用なインターフェースについて検討，設計を行い，XtremWeb と CyberGRIP の 2 つのグリッドミドルウエアに提案するシステムを対象に実装し，予備的な性能評価を行った．提案システムにより，複数のグリッドミドルウエアが提供する計算機上で，遠隔手続き呼び出しによるプログラミングモデルを用いて統一的に利用できるフレームワークを提供する．

Design and Implementation of a Grid RPC System on Multiple Grid Middlewares

YOSHIHIRO NAKAJIMA,[†,††] MITSUHISA SATO,[†] YOSHIAKI AIDA,[†]
TAISUKE BOKU,[†] DAISUKE TAKAHASHI[†] and FRANCK CAPPELLO [†††]

OmniRPC is a Grid RPC system for parallel programming in a grid environment. In this paper, we propose an extension of OmniRPC to make user of computing resources managed by several grid middleware by converting each RPC call to a remote job with document-base communication. By this system, we can exploit not only remote servers and clusters but also computing resources which are provided with grid middlewares on different sites. We designed a general interfaces to construct OmniRPC system on each different grid middlewares and applied the proposing system to grid middlewares: XtremWeb and CyberGRIP. We present implementations for the two grid middlewares and report a preliminary performance of those implementations by using an application. Our model can provide a general framework of parallel programming model by remote procedure calls bridging between a large scale computing resource pool.

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．

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) 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[1] の開発を行ってきた．Grid RPC は，グリッド環境における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して有効である．

近年，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．このために，大量の計算ジョブを実行する XtremWeb や Condor[2]，CyberGRIP[3] などジョブ実行を行うグリッドミドルウエアが開発されている．しかし，これらのミドルウエアは，依然としてファイルを中心としたジョブバッチシステムであり，ジョブのメインの計算を行うプログラムと，そのジョブの結果のデータの解析を行うプログラムそれぞれ作成しなくてはならない．さらに，これらのミドルウエアを利用するためのプログラミングモデルはまだ確立されていない．

そこで本稿では，OmniRPC を，個人用 PC を含む Volatile な計算資源を利用する P2P 環境やエンタープ

† 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
†† 日本学術振興会特別研究員
Research Fellow of the Japan Society for the Promotion of Science
††† INRIA, France

ライズ向けに大量の処理を行うためのグリッドミドルウエアに対応させ，従来コネクションベースで行っていた RPC を，ドキュメントベースにすることによって，リモートジョブとして実行できるようにする．提案するシステムは，ジョブを実行するグリッド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．

Djilali らは，XtremWeb 上で遠隔手続き呼び出しを用いることでバッチシステム上にプログラミングモデルを構築した[4]．しかし，クライアント側は RPC スタイルでプログラミングが可能であるが，リモート側は，ファイルからデータを入力し計算しその結果をファイルに出力するプログラムを別に作成する必要がある．このため，ユーザには統一的なプログラミングモデルは提供されていない．

中田らは，Condor の上に耐故障性を重視した RPC システム Ninf-C を実装した[5]．Ninf-C では，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，他の計算機へのジョブの移動を行い対故障性を保証する．しかし，Condor 以外のミドルウエア上では動作しない．我々のシステムと Ninf-C は類似のアーキテクチャをとっている．

本稿では，グリッドミドルウエア上で OmniRPC を構築するために必要なインターフェースの設計について述べる．その設計に基づき，実際に 2 つのグリッドミドルウエアに適応させ，それらの実装について述べ，予備的な性能評価を行う．

2 章で OmniRPC の概要を述べ，3 章で複数グリッドミドルウエア上で動作する Grid RPC システムの設計について述べる．4 章で 2 つのグリッドミドルウエアへの適応事例をのせ，5 章で簡単な性能評価と議論を行い，終章でまとめと今後の課題を述べる．

2. OmniRPC システム

OmniRPC は，クラスタから広域ネットワークで構成されたグリッドに至る様々な計算機環境において，シームレスなマスター/ワーカ型の並列プログラミングを可能にする Grid RPC システムである．

OmniRPC で想定しているグリッド環境は，インターネット上で複数の計算機クラスタが接続され，それらのクラスタを相互利用するような環境である．また OmniRPC は，現在のクラスタ環境に多く見られる，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た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[6] 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，効率的なプログラミングが可能となる．また，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．

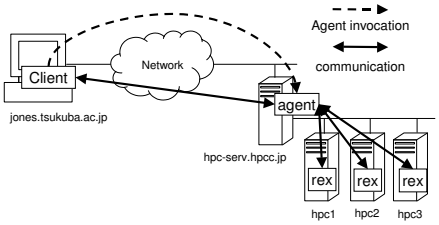

図 1 OmniRPC システムの概要

OmniRPC は，エージェントを使用してリモート実行プログラムのプロセス管理を行う．また，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間の通信を多重化して一つのコネクションで行う．

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を行い，遠隔のクラスタの計算ノードに向けて RPC が行われている．

3. 複数グリッドミドルウエア上で動作する OmniRPC システムの設計

3.1 対象とするグリッドミドルウエア

本研究が対象とするグリッドミドルウエアは，XtremWeb[7] や Condor[2], CyberGRIP,Grid MP[8] などの，サーバーに対してクライアントからジョブ実行を依頼し，システムがジョブを遠隔計算機に割り当てジョブを実行させ，そして，システムがジョブ実行の結果を遠隔計算機から回収するようなジョブバッチシステムである．

これらのジョブバッチシステムでの基本的な単位はジョブであり，ユーザにより投入されたジョブは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．ジョブを構成する入出力データはファイルを介してアクセスされる．

また，これらのグリッドミドルウエアでは，ハードウエアやソフトウエア，またはネットワークの異常により中断されたジョブを再キューイングし，新しい計算機に再スケジューリングする機能を有する．

3.2 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，実行環境を各組織にあるグリッドミドルウエア上の計算機資源に対応させる．

このために，本システムでは以下を目標とする．

- グリッドミドルウエア上における RPC による並列プログラミングモデルの提供
- 耐故障性を備えた Grid RPC システムの構築
- ソースプログラムの変更を行わないクラスタ環境から複数グリッドミドルウエア環境への対応
- 新しいグリッドミドルウエアに対応可能な汎用的なインターフェースの提供
- 実行プログラムの Deployment の自動化

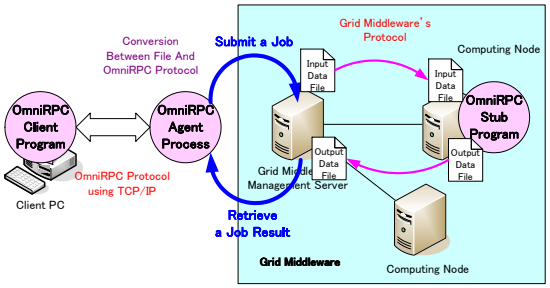

図 2　グリッドミドルウエアに対する OmniRPC の実行モデル

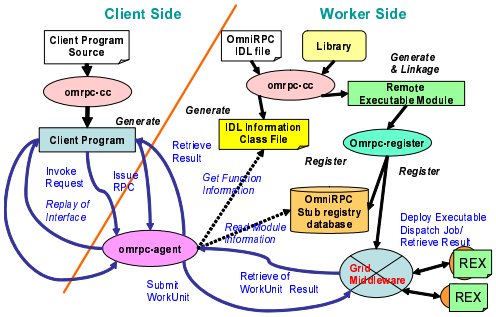

図 3　プログラム作成のフロー

3.3　グリッドミドルウエア上で動作する Grid RPC システムの設計

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でワーカを起動・接続し，データをサーバに転送・計算を行う Push モデルのシステムである．また，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．

グリッドミドルウエアに対する OmniRPC の実行モデルを**図 2** に示す．

遠隔手続き呼び出しを複数グリッドミドルウエア上で行うためには，コネクションにより行っていた計算の依頼と結果の転送を，ファイルを介して，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び出しを行う場合は，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投入する．実行プログラムがワーカにより実行されて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，呼び出しが完了する。

OmniRPC システムでは，エージェントプロセス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．現状の OmniRPC のシステムを大幅に変更することなく複数のグリッドミドルウエアに対応するために，このエージェントプロセスにグリッドミドルウエアのサーバへのブリッジを行う機能を付加し実現することにした．

またグリッドミドルウエア自身は，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，エージェントによってサーバに投入したジョブの実行は必ず終了する．よって，Agent は単にジョブ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．

ただし，提案するシステムでは，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，それぞれのグリッドミドルウエアによる通信性能・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とって，従来のグリッド環境に加え，グリッドミドルウエア上での大量の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．

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 が各グリッドミドルウエアの Server と OmniRPC のクライアントプログラム間のプロトコル変換を行う機能
- 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
- リモート関数のインタフェース情報の管理
- 新しいグリッドミドルウエアに容易に対応可能なインタフェース

図 3 に提案するシステム上での，プログラミング作成時のフローを示す．ユーザは基本的に OmniRPC と同じようにクライアントプログラムとそのリモートでの IDL (Interface Description Language) を記述する．そして OmniRPC が提供するツールを用いてコンパイルやリモートプログラムの登録などを行う．

3.4　複数グリッドミドルウエアへ対応可能なインターフェースの設計

前節で述べた OmniRPC の実行モデルにおいては，エージェントはクライアントプログラムとグリッドミドルウエアのサーバ間でのプロトコル変換を行う．そこで，グリッドミドルウエア操作とジョブの実行マネージメントをする汎用的な BatchSystem クラスと，ジョブ実行要求と計算機資源要求，ジョブの入出力データを管理する WorkUnit クラスの 2 つを作成し，これをエージェントにおいて利用することによって複数のグリッドミドルウエアに対応できるようにする．

BatchSystem クラスでは，３つのキュー（投入する WorkUnit の管理，実行中の WorkUnit の管理，結果の WorkUnit の管理) と，それぞれにグリッドミドルウエアに WorkUnit を投入するスレッド，結果を取得するスレッドを持つ．基本的に，ジョブを投入する場合には，このクラスに対して行い，結果の取得もこ

のクラスを通して行うようにする．但し，グリッドミドルウエアごとに，システム依存のジョブの投入・状態取得・結果の取得・削除の操作についてはインターフェースを実装したクラスを提供することにする．この BatchSystem から利用される操作インターフェースを BatchSystemInterface と呼ぶ．

WorkUnit クラスは，使用するアプリケーション名，RPC の入力・出力データを保持するファイル名，ワーカープログラムに転送するファイル名などを保持する．ジョブ実行要求や計算機使用要求とデータのファイルヒエラルキーはシステム固有であり，これらの要求を生成する操作と，RPC のデータを書き込むストリームを取得する操作，RPC の結果のデータへの入力ストリームを取得する操作が必要である．

各グリッドミドルウエアに対応させるには，この WorkUnit クラスを継承し，システム固有のジョブ要求や計算機使用要求とデータのファイルヒエラルキーの要素を保持させる．

以上，各グリッドミドルウエアに BatchSystemInterface と WorkUnit クラスを継承したクラスを提供することによりシステムを実現する．

3.4.1 グリッドミドルウエアの操作インターフェース

前節で述べた各グリッドミドルエアごとに異なる操作を実装する際に必要な BatchSystemInterface について述べる．実装する操作は，システムを使うためのセットアップと終了処理，ジョブ要求の投入・状態取得・結果の取得・削除とスタブプログラムに対応する WorkUnit の作成である．

具体的には以下にあげる 6 つのメソッドをシステムごとに実装する．

Initialize()グリッドミドルウエアを使用するためにアカウントやセッションを設定する操作．

Finalize()グリッドミドルウエアの使用を終了する操作

createWorkUnit(StubInfo s)指定したスタブプログラムをそれぞれのグリッドミドルウエアに投入するために，それぞれシステム固有のジョブを管理する WorkUnit を継承したクラスのインスタンスを生成する操作．

unregister(WorkUnit u)グリッドミドルウエアのサーバに登録された WorkUnit を削除する操作．

retrieveCompleteWorkUnit(WorkUnit u)終了したジョブの出力ファイルやログファイルをサーバーより取得する操作．

getWorkUnitStatus(WorkUnit u)指定された WorkUnit に対応するジョブの状態を取得する操作．

3.4.2 グリッドミドルウエア上のジョブの記述

prepareToSubmit()グリッドミドルウエア上にジョブを投入する前に，抽象化された WorkUnit をグリッドミドルウエア独自のジョブ要求フォーマットへ変更する操作．

getOutputStream()ジョブの実行時に必要な OmniRPC クライアントから送られてくるデータを書き込む，入力データファイルへの OutputStream を取得する．

getInputStream()ジョブの実行で生成された，OmniRPC クライアントへ送る出力データファイルからの InputStream を取得する．

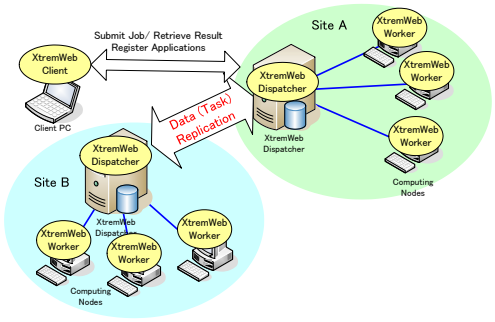

図 4 XtremWeb の概要

4. 複数グリッドミドルウエアへの適用

グリッドミドルウエアの例として XtremWeb と CyberGRIP を提案するシステムに適応させる．それぞれ適応させたシステムを OmniRPC/XW, OmniRPC/CG と呼ぶ．

4.1 使用したグリッドミドルウエア

XtremWeb および CyberGRIP におけるジョブ実行のフローを**図 6** に示す．また，各ミドルウエアの差異を**表 1** に示す．

4.1.1 XtremWeb

XtremWeb は，インターネットに接続された計算機やイントラネット内部にある計算資源プールを活用し，大規模分散処理を目的とした中央集権的な管理・ジョブスケジューリング・結果収集を行うグローバルコンピューティングミドルウエアである[7]．

図 4 に XtremWeb の概要を示す．XtremWeb は，*Client*, *Worker* と *Coordinator (Dispatcher)* の 3 つから構成される．*Client* は，実行アプリケーションの *Coordinator* への登録，タスクの *Coordinator* 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．*Coordinator (Dispatcher)* は，アプリケーションや *Client* から投入されたタスクのホスティングを行い，また，計算を行う *Worker* にタスクやアプリケーションを提供する．*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を提供する．

4.1.2 CyberGRIP

CyberGRIP は富士通研究所が開発した，既存のバッチシステムを仮想的に一つのバッチシステムとして利用できるようなフレームワークを提供し，大量の計算ジョブを処理するグリッドミドルウエアである[3]．

CyberGRIP の概要を**図 5** に示す．CyberGRIP は，OJC (Organic Job Controller), GRM (Grid Resource

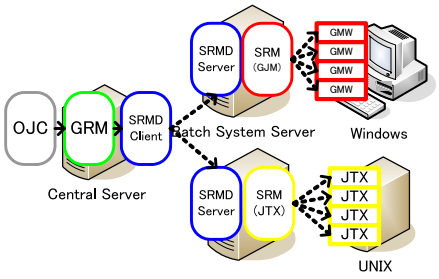

図 5 CyberGRIP の概要

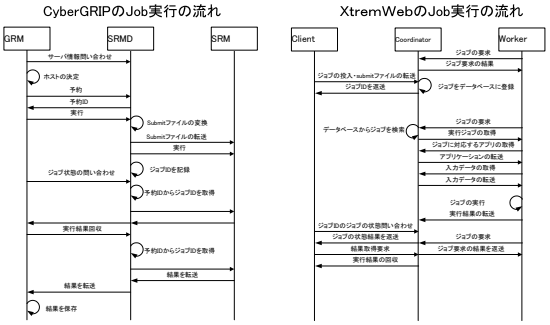

図 6 XtremWeb と CyberGRIP におけるジョブ実行のフロー

Manager), SRMD (Site Resource Manager Dispatcher), SRM (Site Resource Manager) から構成されている．SRM は CyberGRIP での既存のバッチシステムの総称であり，そのバッチシステムとして Condor や CyberGRIP が提供するバッチシステムである JTX が利用できる．SRMD は，SRM を仮想化し，複数のバッチシステムを統合する役割を果たす．ここで，アカウントの差異やファイル転送などを行う．GRM は SRMD で仮想化された複数のバッチシステムを統合しユーザのジョブを実行させる．またユーザージョブのキューイングや計算リソースとジョブのマッチングを行ったり，ジョブの実行状態の監視を行う．OJC は，大量のジョブを CyberGRIP 上で投入管理する機能を提供する．

4.2 実　　装

4.2.1 OmniRPC/XW

サーバーの操作に関して，XtremWeb のアカウントでセッションを生成し，このセッションが必要であり，そのため Initialize においてユーザ認証およびセッションの生成を行うようにする．

また，XtremWeb では，アプリケーションやデータファイルは Coordinator に保存されている．また，実行データを含むファイルヒエラルキーが Zip 形式で圧縮されたファイルが返される．このため，そのファイルを解凍し処理したデータを受け取る操作を retrieveCompleteWorkUnit() で行うようにした．

また，リモートで実行されるアプリケーションに関して，バイナリ名とアプリケーション名を別々に管理しているため，そのマッチングが必要である．そして，クライアント側でジョブの識別子を作成する必要がある．この操作を createWorkUnit() で行うようにした．

WorkUnit クラスを継承したクラスにおいて，prepareToSubmit() で，ジョブ投入に必要なユニークなジョブ ID 生成と 実行プログラムに与えるコマンド引数を指定する．さらに，XtremWeb では，入出力データファイルのファイルヒエラルキーを Zip アーカイブ形式で扱う，そのため，InputStream や OutputStream も Zip ファイルを介した操作で行うようにした．

4.2.2 OmniRPC/CG の実装

サーバを操作するインターフェースに関して，今回はライブラリに問題があったため，シェルコマンドをラッピングして行うことにした．

ミドルウエアの操作に関してはほとんどシステムが面倒をみてくれるため，ジョブ投入とジョブの状態取得だけを実装している．しかし，CyberGRIP では，ジョブ要求の操作において，ジョブ実行要求と計算機の使用要求の両方が必要となる．さらに，アプリケーションのコマンドオプションは ジョブ要求に記述できないため，シェルスクリプトを作成しその中でアプリケーションとコマンド引数を記述することにより実現した．そして，シェルスクリプトをジョブ要求での実行ファイルとして指定する．入出力ファイルへのストリーム操作に関しては，通常のファイル操作で行うことが可能であるため特別な操作は行っていない．

5. 予 備 評 価

実装した OmniRPC/XW, OmniRPC/CG 上でアプリケーションを動作させシステムの性能評価を行う．性能評価に利用した計算機はローカルなクラスタ環境（CPU: Dual Xeon 2.4GHz，Memory：1GB, NetWork: 1Gb Ethernet, OS: Linux kernel 2.6.9）を利用した．クライアントプログラムはマスターノードで実行させ，それぞれ XtremWeb と CyberGRIP のサーバプロセス数は 1 とし，各ノードでの同時実行ジョブ数は 1 とする．

ベンチマークプログラムとして最尤法で分子系統樹推定を行うプログラム PAML[9] を OmniRPC で並列化したものを用いる．使用したデータの並列度は 50 である．1RPC にかかる計算時間は約 50 秒，データ転送量はクライアントプログラムからワーカープログラムに約 20KB, ワーカープログラムからクライアントに約 50KB である．

比較のために従来の OmniRPC ライブラリを用いたものを使用した．

図 7 に計算機の使用台数とそのプログラムの実行時間について示す．ただし，OmniRPC/XW は 16 台を使ったときのみ結果が得られたので報告する．

残念ながら，OmniRPC/XW においては良好な性能をえることができなかった．これはジョブスケジューラで適切にスケジューリングが行われていない可能性があり，原因については調査中である．

表 1 各グリッドミドルウエアの項目における差異

項目	XtremWeb	CyberGRIP
アカウンティング	あり	なし
利用のためのセッション管理	あり	なし
ジョブ要求の指定	Java API を使用	CyberGRIP の GRM スクリプト (ファイル)
ジョブ要求の内容	コマンド引数 入力データがある Zip ファイルの指定	使用するディレクトリの指定，転送ファイル名 ログファイル
実行プログラムの指定	サーバーに登録されたアプリケーション名	実行バイナリのパス
サーバの使用要求	特になし	OS, アーキテクチャの指定
ジョブの ID	クライアントが生成	ジョブ投入時にサーバが返答
ジョブの投入・取得時のデータ形式	Zip 形式のアーカイブ	特になし
サーバからの結果の回収	クライアントが明示的に行う	システムが転送
実行形態	ワーカがサーバーに問い合わせ	サーバからワーカに依頼
サーバーへのアクセス方法	Java API	Shell コマンド

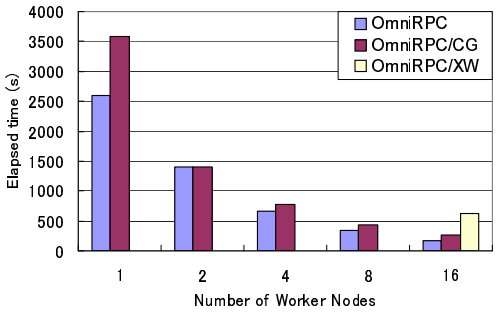

図 7 実行結果

OmniRPC/CG 性能に関して 16 台を使用した際に，OmniRPC の 1 ノードを使用したときと比較して約 9.8 倍の高速化ができた．OmniRPC Agent でのファイルの転送や，ジョブ要求の作成などのオーバーヘッドがあるのにも関わらず，性能向上が得られた．今回使用したベンチマークテストでは 1RPC で約 50 秒であったが，提案システムでは 5 分以上のジョブ実行時間を想定している．しかし，CyberGRIP では台数が増えるに従い，GRM サーバに対してジョブの投入・結果の取得・状況取得がに関わる操作の時間が非常に長いことがあった．これについても原因を調査している．

6. おわりに

遠隔手続き呼び出しのプログラミングモデルを用いてグリッドミドルウエア上の計算機を利用可能な汎用なインターフェースについて検討し，設計を行った．その設計に基づき，XtremWeb と CyberGRIP の 2 つのグリッドミドルウエアに提案するシステムを適応させた．提案システムにより，複数のグリッドミドルウエアが提供する計算機上で，遠隔手続き呼び出しによるプログラミングモデルを用いて統一的に利用できるフレームワークを提供した．

今後の課題としては実アプリケーションによる本システムの性能評価があげられる．また，このフレームワークではワーカープログラムで側の耐故障性を保証できるが，クライアントプログラムは側については未対応である．そこで，クライアントプログラムも含めた耐故障性を保証できるフレームワークの検討を行う予定である．

謝辞 富士通研究所の CyberGrip 開発チームには，ソフトウエアの提供していただき，また本システムの実装についての御助言をいただきました．本研究の一部は，文部科学省科学研究費補助金 課題番号 172002「大容量分散コンピューティングのための大規模スケーラブル P2P グリッド基盤の研究」, 課題番号 177324, および，日仏共同研究プログラム (SAKURA) による．

参 考 文 献

1) 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.
2) Michael J. Litzkow, Miron Livny, and Matt W. Mutka. Condor - a hunter of idle workstations. In *ICDCS*, pp. 104–111, 1988.
3) 小橋 博道, 清水 智弘, 今村 信貴, 上田 晴康, 野口 弘, 山下 智規, 門岡 良昌, 宮澤 君夫. グリッドミドルウェア CyberGRIP による組織を横断した計算機利用. 第 7 回問題解決環境ワークショップ & 第 2 回グリッドセミナー論文集, pp. 63–68, 10 2004.
4) Samir Djilali. P2p-rpc: Programming scientific applications on peer-to-peer systems with remote procedure call. In *CCGRID*, pp. 406–413. IEEE Computer Society, 2003.
5) 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), May 2004.
6) Ninf Project. http://ninf.apgrid.org/.
7) Gilles Fedak, Cécile Germain, Vincent Néri, and Franck Cappello. Xtremweb: A generic global computing system. In *CCGRID*, pp. 582–587. IEEE Computer Society, 2001.
8) United Devices. Grid mp. http://www.ud.com.
9) Ziheng Yang. Paml: a program package for phylogenetic analysis by maximum likelihood. *Computer Applications in BioScience*, Vol.13, pp. 555–556, 1997.