第 2 回 Grid ゼミ

ゼミ担当者　：　田中　裕也，昌山　智，坂田　大輔
指導院生　：　下坂　久司，斉藤　宏樹，岩橋　崇史，中山　靖一
開催日　：　2003 年 6 月 4 日

ゼミ内容: 第 1 回では，Grid の概要を学んだ．第 2 回では実際に Grid がどのような構成になっていて，どのような技術で実現されているかについて学ぶ．主な Grid を実現するための技術としてミドルウェアである NetSolve，Ninf，Condor やツールである Globus について学ぶ．

1　はじめに

Grid は地理的な距離や計算機の構成の差異をユーザに意識させずに，計算資源を提供する．Grid は，スイッチを入れるだけで電気が使えるのと同じように，利用者が世界各地のコンピュータの計算リソースをいつでもどこでも利用できることを目的としている．

複数のコンピュータを利用して，計算処理を高速化する点では，Grid は従来の並列処理の延長にある技術である．また，分散された資源をまとめて使おうという考えからも注目されてきた．以下に Grid と従来の並列処理の違いを示す．

- 多数の組織から，様々な種類のリソースが提供されて構築されるヘテロな環境．
- 動的に変化する環境．
- 外乱の多いネットワークでつながれている．
- 手にできる計算リソースの規模が格段に大きい．

本ゼミでは，ネットワーク利用におけるセキュリティの問題，様々な計算リソース間のアーキテクチャの問題などを，Grid ではどのように克服しているのかを説明する．

2　Grid のアーキテクチャ

Fig. 1 のようにコンピュータは，ハードウェア上で OS が動き，その OS の上でアプリケーションが動くといった階層構造を持っている．

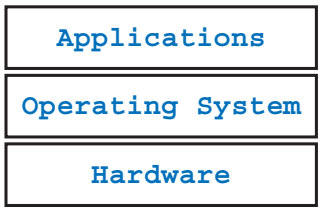

Fig. 1　コンピュータの階層モデル

Fig. 2　OSI 参照モデル

Fig. 2 のように OSI 参照モデルも階層構造を持つ．

Fig. 3 のように Grid もこれら 2 つと同様に下位層により上位層が構築される階層構造を持っている．

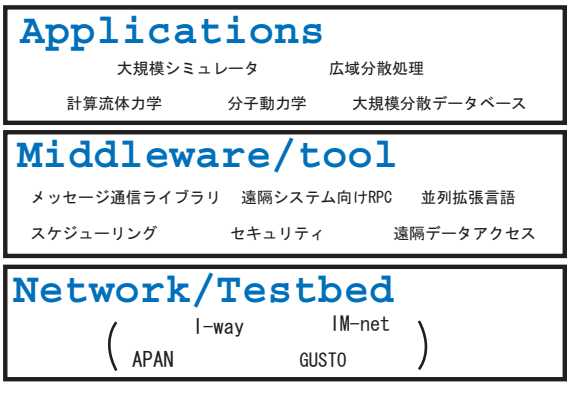

Fig. 3　Grid の階層モデル

Grid の階層化モデルは大きく分けて Applications 層，Grid 環境で動くアプリケーションの作成や実際に動かす際の手助けをするシステムを扱う Middleware/tool 層，物理的環境を扱う Network/Testbed 層の 3 層で構成される．特に Middleware/tool 層はネットワーク利用におけるセキュリティの問題や様々な計算リソース間のアーキテクチャの問題を克服するための重要な層である．

以下に Middleware/tool 層に位置付けされるミドルウェア，ツールを示す．

- Globus
  ミドルウェアの互換性を統一するためのツールキット.

- NetSolve
  ネットワークを介して遠隔地にある科学技術計算ライブラリを利用できるシステム.

- Ninf
  NetSolve と類似しており，遠隔地にあるハードウェア，ソフトウェア，データベース等を利用できるシステム.

- Condor
  分散した資源を使って高スループットな計算を行うシステム

これらの Middleware/tool について，説明する.

3 Middleware/tool

3.1 Globus

Grid においては，ユーザの認証，通信，遠隔計算機上でのプロセス生成などの様々な要素技術が必要となる．Globus はこのような Grid に必要とされる基本的なサービスを提供する．Globus は Grid のための資源管理機構，ユーザ認証システム，通信ライブラリなどのサービスを提供する．Globus は，これらのサービスにより，低レベルの互換性を保つことができ，上位のミドルウェアの構築を容易にする.

3.1.1 Globus Toolkit

Globus Toolkit は Globus プロジェクトによって作られたツールキットであるユーザ認証，通信，資源管理・監視機構などの，Grid において必要となるさまざまな要素技術をライブラリおよび API という形で提供する巨大なパッケージである．Globus Toolkit の利用手順を Fig. 4 に示す.

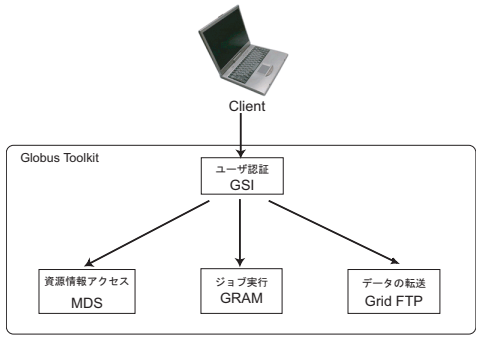

Fig. 4　Globus Toolkit の利用手順

ここでは，ユーザ認証を行う GSI，ユーザがジョブを走らせることができるサービスである GRAM，利用できる計算資源の情報を取得できるサービスである MDS について説明する．Globus Toolkit のサービスを Table 1 に示す.

- GSI
  GSI(Globus Security Infrastructure) は Grid におけるセキュリティサービスを提供する．GSI の認証メカニズムは，PKI(Public Key Infrastructure) と呼ばれる公開鍵暗号方式を用いたセキュリティ技術に基づいている．PKI とは，ユーザや計算資源などの認証単位が秘密鍵と公開鍵の対を持っており，公開鍵を流通させることによって，集中型の公開鍵の認証サーバを持たなくても任意の二者間での認証ができるセキュリティ技術である．GSI の利点は，ユーザは計算資源を使用する際に，1 度だけ認証を行い，そのときにプロセスがユーザに代わって資源を利用できるように証明書を発行してくれるというところにある.

- GRAM
  GRAM(Globus Resorce Allcation Management) は計算機管理のためのサービスを提供する．このサービスを受けることで，ヘテロ性を考慮することなく，プログラムをリモートリソース上で実行することが可能となる．Fig. 5 に GRAM の仕組みを示す.

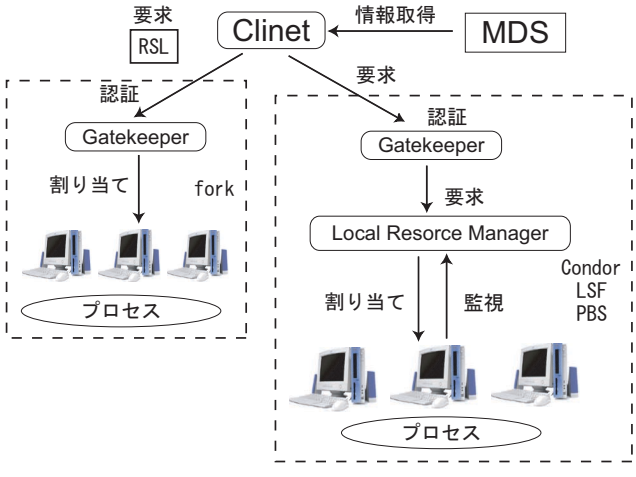

Fig. 5　GRAM

GRAM は fork，LSF や Condor など，プロセスの生成・管理の方法に応じた型（manager type）を持つ．計算サーバは 1 つ以上の GRAM を提供する．そして，各 GRAM ごとにクライアントからの計算要求を受け付けるサーバプロセスである gatekeeper をデーモンとして稼働させる.

- MDS
  MDS(Globus Meta-computing Directory Service

Table 1　Globus Toolkit のサービス

Service	Name	Description
Resource Managemanet	GRAM	リソースの割り当て，プロセス生成
Communication	Nexus	Unicast/Multicast 通信サービス
Information	MDS	システムの構造，状態に関する情報サービス
Security	GSI	authentication などのセキュリティサービス
Helath and status	HBM	システムの状況サービス
Remote data access	GASS	データへのリモートアクセスサービス
Executable management	GEM	実行ファイルの構築，キャッシングおよび配置

: MDS) は情報サービスへのアクセスを提供する．MDS の構成を Fig. 6 に示す．

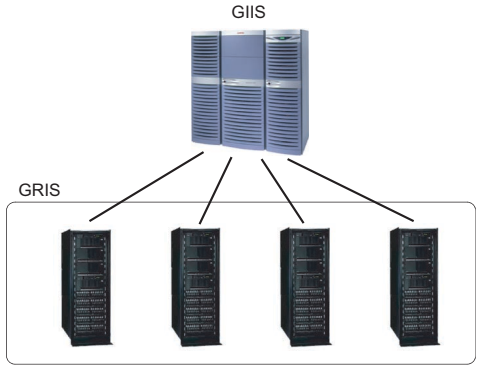

Fig. 6　MDS の構成

MDS の利点は，どのサイトで提供されているサービスも同一の方法で得ることができ，動的データへの効率的なアクセスや複数の情報源から同時アクセスなどを行うことが可能である．また，単一のサーバで情報を保持しているのではなく，サーバが分散している非集中型管理である．MDS は GIIS と GRIS で構成されている．

- GIIS(Grid Index Information Service)
  複数のサーバから情報を収集する．Grid 上でサーチエンジンの役目を担い,「CPU800MHz 以上，メモリー 120M バイト以上のサーバ」などのように検索を可能にする．
- GRIS(Grid Resource Information Service)
  個々のリソース上で実行されるサーバであり，負荷状態やプロセス情報などのリソース固有の情報を提供する．

3.2　NetSolve(Network Enabled Server)

NetSolve とは，テネシー大学 Innovative Computing Laboratory の Jack Dongarra 等によって開発された Grid RPC(remote procedure call) System である．Grid RPC System は，クライアント・サーバ型の計算システムであり，サーバに存在する計算ルーチンをクライアント側のプログラムから，容易に実行できるようにすることを目的としている．この仕組みにより，NetSolve は遠隔地にある計算資源にアクセスすることができ，遠隔地のコンピュータにある最適化されたライブラリを使用することができる．NetSolve の最新バージョンは，2002 年 6 月 5 日に出された 1.4.1 となっている．

3.2.1　NetSolve の利点

NetSolve を用いずに LAPACK(Linear Algebra Package)[1]などの数値ライブラリを利用する場合，プログラムを実行させる方法として主に以下の 2 つが挙げられる．

1. LAPACK のソースファイルあるいはオブジェクトファイルを手元のコンピュータにインストールして利用する．
2. 計算センタなどのスーパーコンピュータに login し，既にインストールされているライブラリを利用する．

これらの方法で，遠隔地の計算資源を利用することはユーザにとって手間がかかる．前者の方法ではライブラリのバージョンが上がるたび，あるいはソフトウェアのバグが見つかるたびに，ライブラリを新しくする必要がある．後者の方法では逐一計算センタのスーパーコンピュータに login し，実行する必要がある．

NetSolve を使うことでユーザは最新のソフトウェアのインストールの手間や，遠隔地のコンピュータへの login の手間が省けるようになる．

3.2.2　NetSolve の構成

Fig. 7 に NetSolve の構成を示す．NetSolve は，NetSolve クライアント，NetSolve エージェント，NetSolve リソースで構成される．NetSolve クライアントは，手元のコンピュータで動き，NetSolve エージェントを通じて実際の計算は NetSolve リソースで行われる．NetSolve リソースとしては，ワークステーション，スーパーコン

[1]LAPACK とは netlib で公開されているライブラリの 1 つで，線形代数の計算を行うルーチンがまとめられている．LAPACK を使うことで，連立一次方程式，最小二乗法などの高速ルーチンが手軽に利用できる．

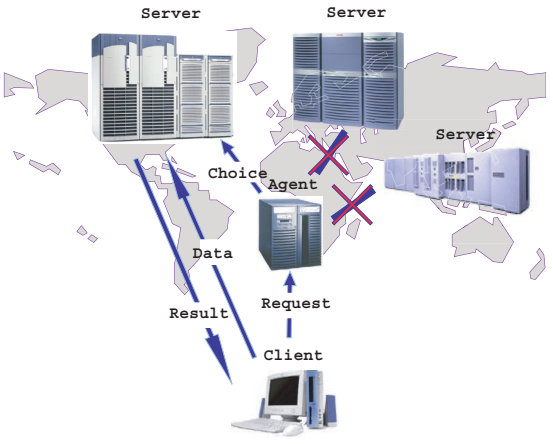

Fig. 7　NetSolve の構成

ピュータ，計算機クラスタなど，様々な計算資源があり，リソースの種類に関係なく同一のインターフェースで利用することができる．NetSolve エージェントはその時その時のネットワークの状況や計算機の負荷を見ながら最適なリソースを選ぶ．以下に実際の手順を示す．

1. request
   クライアントが実行の要求をエージェントに対して行う．

2. choice
   エージェントは要求を受け取り，適切な計算リソースを選ぶ．

3. data
   クライアントは選ばれた計算リソースにデータを送る．

4. result
   計算リソースは，クライアントからデータを受け取り，数値ライブラリを実行し，結果をクライアントに返す．

以上より，実際にライブラリを実行する NetSolve リソースはネットワークに接続され，起動していればよいため，世界中のコンピュータで実行することができる．

3.3　Ninf

3.3.1　Ninf とは

Ninf(Network based Information library for High Performance Computing) は，1994 年から電子技術総合研究所で設計・開発されているシステムである．Ninf は，広域ネットワーク上の数値計算ライブラリや数値情報に必要なデータベースを通じて，主に科学技術計算分野の情報ならびに計算資源を提供・共有するものである．

3.3.2　Ninf システム

Ninf はサーバ・クライアント型の計算システムである．Ninf サーバはクライアントからの要求を受信し，要求された実行プログラム (Remote Library Exectutable) を実行する．Ninf サーバは，数値計算ライブラリは実行可能な形で提供している．クライアントは Ninf の RPC (Remote Procedure Call[2]) を用いて提供されている計算資源を利用することができる．ソースプログラムとして提供される情報に比べてインストールなどの作業も必要なく，必要な時に最新のものを利用することができる．ゆえに，計算資源の共有だけでなく，共有されるライブラリを選択することによって高品質な計算を得ることができる．

Fig. 8 で示すように，クライアントと Ninf サーバの仲介を行なうものに Ninf メタサーバがある．Ninf メタサーバは複数の Ninf サーバの情報を管理し，Ninf サーバが提供するサービスの内容や状況に応じて適当な Ninf サーバを選択する機能を提供する．これによって，ユーザは実際に計算が行われる計算機を意識することなく計算資源を利用できる．

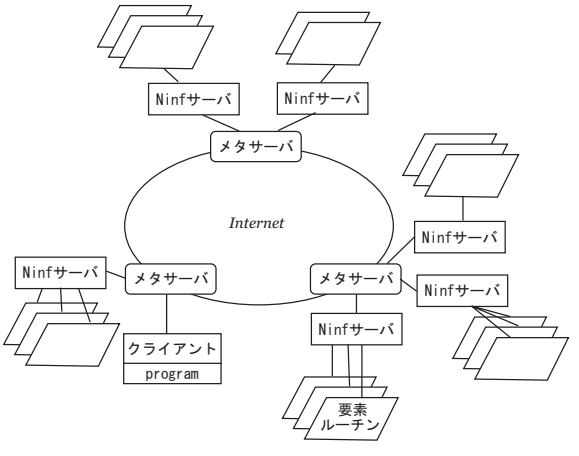

Fig. 8　Ninf のメタサーバ

クライアントと Ninf サーバの通信について Fig. 9 に示す．クライアントが Ninf の数値ライブラリを利用する流れを以下に示す．

1. クライアントは Ninf サーバと通信してインターフェイス（引数情報）を取得する．

2. クライアントは，その情報に基づいて引数を Ninf サーバへ送信する．Ninf サーバは fork などでプロセスを立ち上げて，実行プログラムを実行する

3. クライアントとサーバ間に通信路を確立する．(コールバック[3])

[2] リモートホスト上のアプリケーション機能を実行するためのプロトコル
[3] 通信において、いったん通信先を呼び出し、相手側からの発信を求めること

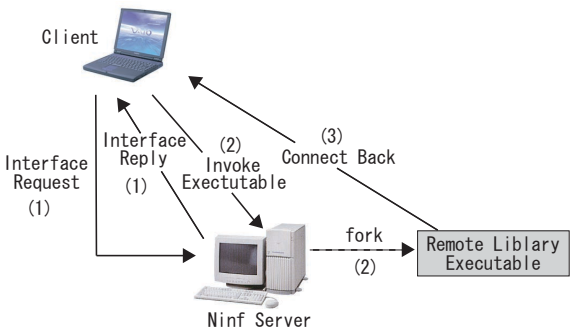

Fig. 9　Ninf

3.3.3　Ninf-G

Fig. 10 で示すように，Ninf-G は Ninf を Globus 上に実装したシステムである．Ninf-G は，Ninf サーバとクライアント間のやり取りを Globus のサービスである GRAM，GRIS，Globus IO に置き換えたものである．GRAM は，クライアントから実行プログラムのインタフェースを受信して，fork などでプロセスを立ち上げて実行する．GRIS は，実行プログラムのインタフェース情報を提供する．Globus IO は，セキュリティで保護された通信路を提供する．Ninf-G は、Globus 上に Ninf を実装することで Ninf よりセキュリティ面が強化された．また，サーバ側は Globus の知識を学習するだけで Ninf サーバを構築可能となる．Ninf-G のクライアントとサーバの通信について以下に示す．

1. クライアントはインタフェース情報を GRIS から引き出す．
2. クライアントはその引数情報を Ninf サーバへ送信する．Ninf サーバは実行プログラムを GRAM によって起動する．
3. クライアントとサーバ間に Globus IO を用いた通信路の確立．

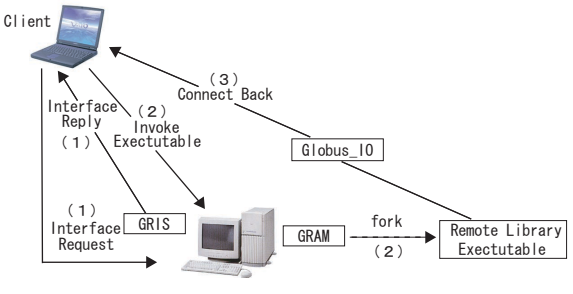

Fig. 10　Ninf-G

3.4　Condor

Condor は，計算機の空き時間を利用することを目的とした，分散リソース管理のミドルウェアである．ユーザは計算資源として割り当てられた計算機のうち，アイドル状態（入力待ちで何も行っていない状態）のものを計算資源として用いることができる．計算（ジョブ）の割り当ては，Condor Central Manager が担当するため，ユーザは実際にどの計算機を使うのかを意識することなく，ジョブを投入できる．また，MPI や PVM などといった並列計算用ライブラリもソースプログラムを変更することなく利用できる．

3.4.1　Condor の動作

Fig. 11 に Condor の構成を示す．Condor の動作を以下に示す．

1. Central Manager がユーザとリソースの ClassAd 情報を集める．
2. ClassAd 情報をもとに，スケジューリング情報をユーザとリソースへ送る．
3. リソースがユーザへ一時的なコントロール権限を与える．
4. ユーザがジョブをリソースへ送り，実行させる．

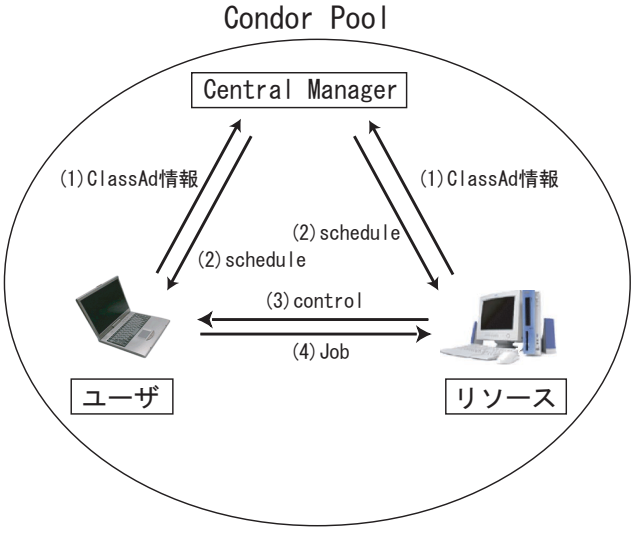

Fig. 11　Condor

3.4.2　Condor の特徴

- 空いてるリソースの獲得
  ユーザがジョブを Condor に投入すると，Condor は空いているリソースを見つけ，そのリソースでジョブを実行する．そのリソースが何らかの理由で利用できなくなった（所有者が作業を始めた or 計算機の負荷が重くなった）場合，Condor はそのジョブを違うリソースに移動させ，ジョブを再開させる．もし，適当なリソースが見つからない場合，ジョブは適当なリソースが見つかるまでサスペンド（一時停止）される．

- PC の需給調停
  Condor では，ClassAds(Classified Advertisements) という仕組みを用いて，リソース提供者とリソース利用者の需給調停を行っている．リソース提供者は，どれくらいの資源を，どんな状態のときに提供するか，詳細に指定することができる．また，リソース利用者も，どのようなリソースを，どのような優先順位で必要としているかを指定することができる．これにより，提供者は自分の都合の良いときだけ PC を提供し，利用者は欲しいリソースを利用することができる．

- Condor のジョブ処理
  Condor を使用するためには，利用するすべての計算機上で，Condor をインストールする必要がある．Condor では，Condor がインストールされ，ユーザが投入したジョブを処理できるマシンの集まりを “Condor pool” と呼ぶ．そして，Condor pool のマシンの中で pool 全体を管理するマシンがある．このマシンを “Central manager” と呼ぶ．

  Condor はユーザが投入したジョブを処理するために，Central manager は Condor pool の中から現在使われていない計算機を探し出す．そして，その中からユーザの必要とするメモリや OS 等の条件を満たすものを選んで，ジョブを実行する．

3.4.3 Condor-G

Condor-G は Condor と Globus の技術的な結合を行ったミドルウェアである．Condor-G では 2 つの特徴を持つ．

- Globus Toolkit によってサポートされた，マルチドメイン環境におけるセキュリティとリソースへのアクセス．

- Condor によってサポートされた，単一ドメイン環境におけるジョブ管理とリソースの利用である．

Condor-G は Globus Toolkit によるドメイン間（inter-domain）のリソース管理のプロトコルと，Condor によるドメイン内（intra-domain）のリソースとジョブの管理を併せ持つことによって，複数ドメインのリソースをまるで単一ドメインのように利用することが可能になっている．

3.4.4 Condor と Condor-G の違い

Condor pool 内の機能は 2 つに分けられる．1 つはジョブ管理，もう 1 つはリソース管理である．Condor と Condor-G の違いはジョブ管理機能にある．

離れたリソースでジョブを実行する場合，Condor-G では Condor によるプロトコルの代わりに Globus Toolkit を使う．Condor-G は離れたリソースへのアクセスと，ジョブの管理を行うことができる．

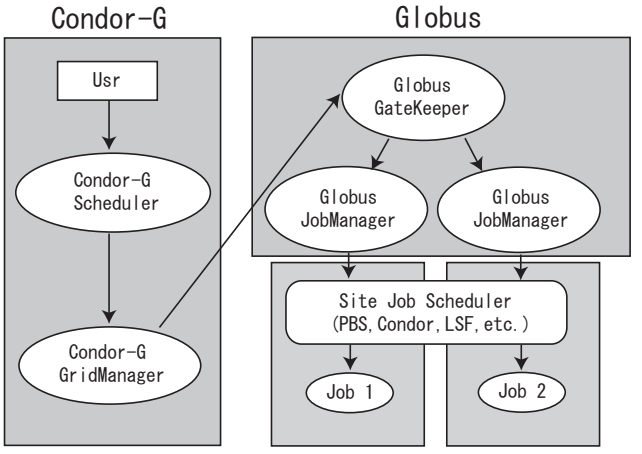

Fig. 12　Condor-G におけるジョブ管理

3.4.5 Condor-G におけるジョブ管理

Condor-G で，離れたリソースにジョブを渡す場合，どうするかについて説明する．ジョブの流れを Fig. 12 に示す．

まずユーザは Condor-G のスケジューラにジョブを投入する．スケジューラでジョブのスケジューリングを行った後，Grid Manager へジョブを送る．Grid Manager はグリッドに関する管理をおこなっており，Globus Gatekeeper へジョブを送る．

Globus Gatekeeper に送られたジョブは Globus JobManager を通じて，適したリソースへ送られる．リソース内でのジョブのスケジューリングは，それぞれのリソースで使われているミドルウェア (PBS,Condor,LSF など) が行う．

4 Network/Testbed

Grid のソフトウェアを開発すると，その研究成果を実際の環境で試すことが必要となる．そのとき用いられるのが Testbed である．Testbed とは，Grid のソフトウェアをテストするために使用する，ネットワーク，サーバ，クライアントといった Grid の物理的環境のことである．Testbed は，アプリケーションの作成，ユーザの拡大などの点においても非常に重要である．

Globus プロジェクトでは GUSTO(Globus Ubiquitous Supercomputing Testbed Organization) という Grid の Testbed を構築している．これは，現在 23ヶ国，125 サイト (大学，研究所など) で構成され，Grid の Testbed としては最大のものである．

NetSolve のインターネットを用いた環境における Testbed として，Tennessee 大学，NCSA(National Center for Supercomputing Applications)，デンマークコン

ピューティングセンタなどで実際に NetSolve リソースが起動し，いつでもどこでも実験が行える環境が整っている．

5 OGSA

OGSA(Open Grid Services Architecture) とは Globus プロジェクトと IBM が共同して Web サービスの融合と推進の標準化のために提案した Grid をビジネスで利用するための仕様・標準規格のことである．2002 年 2 月にトロントで開催された GGF4 で OGSA は提案された．これまで科学技術計算向けに利用されてきた Grid であるが，ビジネス向けにも Grid を利用しようとする試みである．2003 年 7 月頃にリリースされる Globus Toolkit v3.0 より開発される予定である．OGSA は Fig. 13 に示すような Web サービスと Grid 技術の間に属する位置付けになっている．また，Fig. 14 のように Web サービスの技術と Grid の技術の両方をサポートする仕様となっている． OGSA は GGF で今後も検討・改良される技術である．

5.1 OGSA の構成

以下に示す Web サービスに重要とされる要素を OGSA は Grid の標準機能として集約している．

- XML … 通信内容の記述
- SOAP … 通信プロトコル
- UDDI … ディレクトリサービス
- WSDL … サービス記述言語

Fig. 15 より Web サービスの流れを示す．

1. Web サービスの提供者は WSDL という言語でサービス内容などを記述し，UDDI ディレクトリに登録する．
2. Web サービスの利用者は UDDI ディレクトリから利用したい Web サービスを検索し，結果を受け取る．
3. SOAP という通信プロトコルを使用し，XML 文書をやり取りする．

5.2 OGSA の効能

Web サービスと Grid 技術の統合によってこれまで科学技術用の計算にしか使われなかった Grid 技術をビジネスの面においても，つまり我々一般人や企業でも使えることになる．

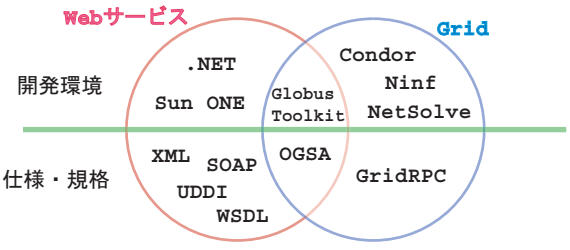

Fig. 13 OGSA の位置付け

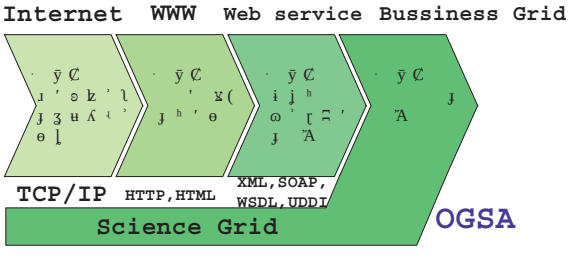

Fig. 14 Grid 技術と Web サービスの融合

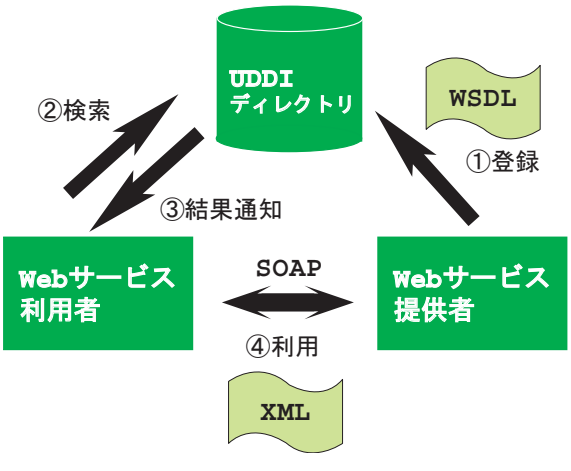

Fig. 15 Web サービス

参考文献

1) [Globus]
http://mikilab.doshisha.ac.jp/dia/seminar/2002/pdf/globus.pdf

2) [NetSolve]
http://ninf.apgrid.org/papers/hpc0110nakada/hpc0110.pdf
http://mikilab.doshisha.ac.jp/dia/seminar/2002/pdf/netsolve.pdf

3) [Ninf]
http://ninf.apgrid.org/welcome.shtml

4) [Condor]
http://taka-lab.ise.osaka-sandai.ac.jp/~hiratam/kenkyu/report/condor/condor_ins/index.html
http://www.cs.wisc.edu/condor/condorg

5) [OGSA]
http://www.jpgrid.org/papers/ogsaeaw.pdf
http://magazine.fujitsu.com/vol53-5/paper16.pdf