卒業論文

# グリッドコンピューティングの調査と検討


氏　　名　　：木ノ下　稔
学籍番号　　：2210990079-0
指導教員　　：山崎 勝弘 教授
提 出 日　　：2003 年 2 月 21 日

立命館大学　理工学部　情報学科

内容梗概

本論文ではグリッドコンピューティングにおける動向とその開発プロジェクトの調査を米国、英国、日本に分けて行った。また、多くのグリッド開発プロジェクトで利用され業界標準となりつつある GulobusProject のグリッドミドルウェア GlobusToolKit を調査した。そして Globus の主要の４つ機能であるセキュリティ(GSI)、資源管理(GRAM)、情報サービス(MDS)、データ管理(DMS)の詳細を得た。そしてこの Globus によるグリッド環境を構築しシステムテストを行った。

ヘテロジニアスなグリッド環境においての従来の負荷分散では理想的な並列効果が得られない。そのため、ヘテロジアスに適応した負荷分散をする必要がある。そこでヘテロジニアスなグリッド環境における負荷分散の評価するため評価コストモデルを提案し検討を行った。評価コストモデルから算出した値をもとに、ヘテロジニアスな環境においての負荷均衡について評価と考察を行った。そして最適な負荷分散を行うことにより４倍から 16.9 倍へと並列効果をあげることができた。この結果により、最適な負荷分散をすることによりグリッド環境においても並列効果を得ることが可能であると確認した。

目次

表目次



**図目次**

# 1　はじめに

近年、広い地域に配置された計算資源を用いて分散/並列計算を行うグリッドコンピューティングに関する研究がさかんに行われるようになってきた。グリッドコンピューティングは広域ネットワーク上に分散された計算資源を仮想的な高性能計算機(メタコンピュータ)とみたてて分散/並列計算を行うシステムである。グリッドコンピューティングシステムにおいては、ユーザ認証、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 プロジェクトによる Globus ToolKit はこのようなグリッドコンピューティングに必要とされる基本的なサービスを提供する。Globus はグリッドコンピューティング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を提供する低レベルなツールキットであり。Globus が提供するツールを用いて上位レベルにグリッドコンピューティングを構築することが可能となる。例えば、通信、ユーザ認証には Globus を用いて MPICH を実相した MPICH-G などが存在する。Globus はグリッドコンピューティングシステムに必要とされる基本的サービスを提供しており。グリッドコンピューティングシステムの構築ミドルウェアとして事実上標準になりつつある。[3] [12]

本研究ではこのようにさかんに研究されているグリッドコンピューティングについて、そのアーキテクチャや現在どのようなプロジェクトが進められておりどのような研究が行われているのかを調査を行う。そして、その有用性について研究室においてどうのような実験やその技術を利用すればよいかを検討することを目的としている。

グリッドコンピューティングを調査する上で注目したのが Globus Project が開発している GlobusToolKit である。Globus Project は米国アルゴンヌ国立研究所と南カルフォルニア大学との共同開発でプロジェクトであり GlobusToolKit はグリッドシステムを容易に構築するためのツールキットである。GlobusToolKit は世界中のグリッドプロジェクトでほとんど使用されて、標準として広く普及しているグリッドミドルウェアである。また、今回このミドルウェアを使ってグリッド環境を構築をした。

第 2 章でグリッドコンピューティングについて調査し、米国、英国、日本における主要グリッド開発プロジェクトについて述べる。第 3 章では今回使用した Globus が提供するシステムと Tool 群のなかで重要なシステムウェアについて述べ。4 章で実際に Globus を計算機にインストールし設定とシステムテストし、グリッドを利用してみての使用感を延べ。第 5 章でグリッド環境下での並列手法について検討する。

# 2　グリッドコンピューティングとは

## 2.1　　グリッドコンピューティングとは

グリッドとは

「グリッド」とは電力の送電線網(電力網)を指す言葉である。これは我々が電力を使用する時に電源ゲーブルをコンセントに差し込んだり、スイッチを ON にするだけで、発電機のことを考える必要がないように、グリッド技術を利用するときにも、コンピュータの構成を気にする必要がないことを表している。

さらには、電力網が現在の社会的・産業的基盤を形成しているのと同じように、コンピュータ資源ネットワークが近い将来の産業基盤になるだろう。なってほしいという願いがこめられている。当初、グリッドはスーパーコンピュータの利用技術から発展してきたため、グリッドコンピューティングと呼ばれていた。しかし、グリッドが発展しグリッドの実現分野が Computing(計算)のみならず。大規模データや特殊実験装置の共有にまでおよぶようになったため。現在では広い意味でグリッドと呼ばれている。また、グリッドという言葉は米国のプロジェクトの名称であり、情報ネットワークを電力網のアナロジとしてとらえた場合に InfomationGrid(iGrid)と呼ぶこともある。

(1)　グリッドの目的

グリッドの目的は、広域に分散した計算資源や情報資源をネットワークを介してまとまって管理し、ユーザが必要なときに必要なだけ利用できる環境を提供することである。これによってネットワークに接続された複数のスーパーコンピュータを一つの巨大なコンピュータのように使用したり、遊休状態になっている多数のパソコンを利用してスーパーコンピュータと同様の計算の量を実現することができる。このときユーザは無制限のリソースをもつ 1 台の超大型コンピュータを使用しているように見えるが、物理的には複数台のコンピュータを使用している。

(2)　分散システムにおいてのグリッド

分散コンピューティング・アーキテクチャは大別すると以下の 3 種類に分けることができる。

- マルチプロセッサ
- クラスタ
- グリッドコンピューティング

この 3 つのアーキテクチャはそれぞれことなった特性を持ち、得意とする分野も異なる。

マルチプロッセサ・アーキテクチャは 1 台のコンピュータに複数個のプロセッサを搭載し、各プロセッサに処理を分散する。このアーキテクチャには「複数のプロセッサでメモリを共有するタイプ」「メモリを共有しない複数のプロセッサをクロスバー・スイッチなどで接続するタイプ」そしてこの 2 つを組み合わせた「ハイブリッド・タイプ」の 3 種類が含まれる。いずれのタイプもプロセッサ間の通信速度は非常にはやい。そのため各プロセッサで行う計算が相互依存するような並列処理アプリケーションの実行に適している。

クラスタアーキテクチャは同一構成の複数台のマシンをネットワーク接続して、各マシンに処理を分散する。もちろん、複数のマルチプロセッサシステムを使って、クラスタを構築することも可能である。ただし各マシン間の通信性能により絶対的なスループットの方が求められる分野に向いている。

グリッドコンピューティング・アーキテクチャはネットワーク上のシングル・プロセッサシステ、マルチプロセッサシステム、クラスタ・システムの集合体を仮想的な単一コンピュータ(グリッド)として扱う。グリッドを構成する各システムの相互通信速度はクラスタ・システムのパワーよりも低速になる。さらに各システムの処理能力にばらつきがあるので、効率よく処理分散させるには、複雑な並列処理管理機構が必要とされる。その一方で精度の高い並列処理で非常に高いスループットを実現させることができる。また膨大なコンピュータ資源を統一的に扱うことができるため、様々な応用が考えられる、アーキテクチャである。

(3)　グリッドの構成単位

グリッド・システムを構成単位という支店で分類すると、定義され明確に分けられているわけではないが以下の 3 つに分けられる。

- クラスタ・グリッド
- キャンパス・グリッド
- グローバル・グリッド

クラスタ・グリッドは 1 つの LAN に所属するマシン構成するシンプルなグリッドである。部門ごと、プロジェクトごとなどの比較的小規模な構成になることが多い(大規模なクラスタ・グリッドを構築することもできる)。リソース管理やユーザ認証などの仕組みが比較的単純ですむためネットワークの品質も確保しやすい。そのため高スループット、ハイ・コスト・パフォーマンスのどちらの要求にも対応できる。

キャンパス・グリッドは複数のクラスタ・グリッドを重ねて稼働させリソースの共有を実現する構成である。そして、企業や研究所といった組織ごとに構成されることが多い。単一のクラスタ・グリッドに対し、柔軟性および拡張の面で優れ、部門間での協調作業が行えるようにある点でも有用である。

グローバル・グリッドは、キャンパス・グリッドの集合体である。各キャンパス・グリッドはインターネットなどの外部ネットワーキングをかいして相互接続される。キャンパス・グリッドかんで合意したリソース管理ポリシーやプロトコルに従って、世界中に分散されて共有リソースを利用可能にする構成である。

## 2.2 グリッドの階層構造

グリッドの考え方は前項でもあげたように多岐にわたっている。地理的に分散したサーバ、ストレージといった計算資源・データ資源のみならず実験装置、観測装置、研究者をも含めた広義のリソースを対象とし、高速ネットワークによりリモートユーザも物理的な距離や計算機の構成の際などを意識させずに信頼度を高く使うことができるようにとインフラと計算機環境を提供することにある。そのためグリッドは必然的に階層構造を取ることになるが、階層間のインターフェイスを標準化・共通化することによって、拡張性、システムの構成の容易性が格段に向上することになる。現在のグリッドの考え方で、その構成はおおざっぱにいうと

- アプリケーション層
- 問題解決環境/プログラミングツール層
- 主要サービス/グリッドミドルウェア層
- 物理層(ネットワーク、コンピュータ(OS も含めて)、ストレージ)

の 4 層に分類される(図 1)

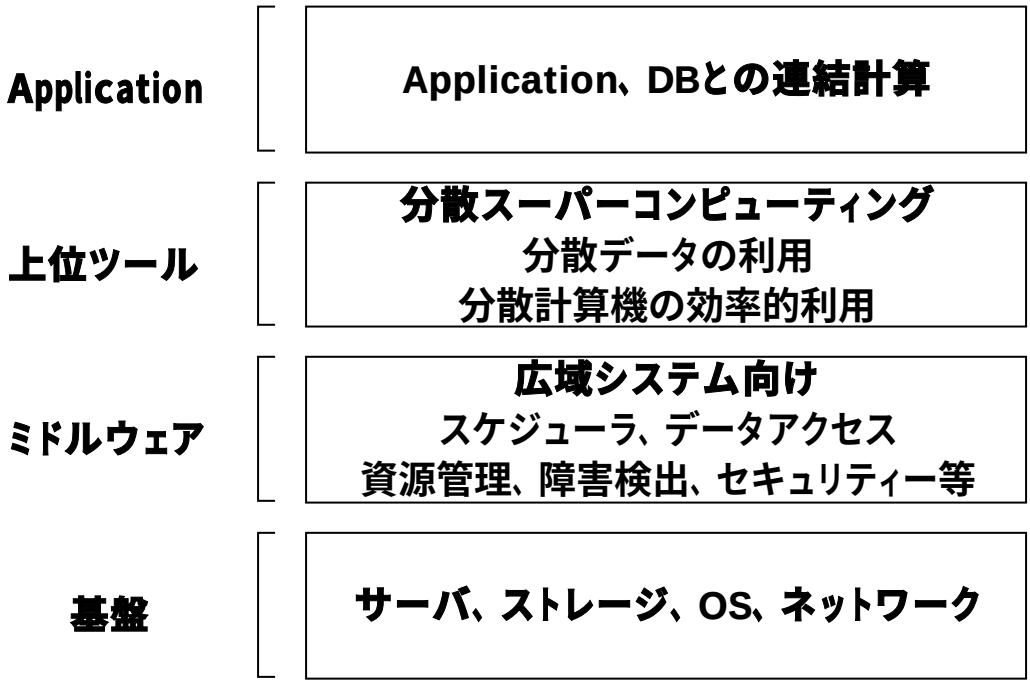


**図 1 グリッドの階層構造**

## 2.3 グリッド開発プロジェクト

米国、欧州、アジア諸国、日本のグリッド技術への取り組みを行っているプロジェクトを表1～3に示す。[11]

(1) 米国のグリッドプロジェクト

米国におけるグリッド技術は、過去のグリッド技術開発と実証システムの開発の蓄積がある。これはグリッド技術がスーパーコンピュータをネットワークを介して利用する米国の研究から始まったからである。その上に、さらなるグリッド技術開発、大規模な実証システム、グリッド応用開発が積極的に進められている。例えば NSF の TeraGridProject では、TeraByte スケールの大容量データ処理の研究が行われており、大規模実証システムとして全米の 4 つの研究センターが参加し 13.6TeraFlops の高性能コンピュータと 576TeraByte の巨大ストレージが 40Gbps の高速ネットワークに接続される計画である。もう一つの注目すべきプロジェクトとして、NIH の Biomedical Infomatics Research Network (BIRN)がある。このプロジェクトでは、カルフォルニア大サンディエゴ校を中心とする全米 10 の医療研究期間によって、アルツハイマー病などの研究のため、脳の様子を可視化し、データを共有するグリッドが開発中である。これには医療の現場を含め多くの研究者が参加している。

**表 1 米国主要グリッドプロジェクト**

| プロジェクト | 機関 | 期間 | 目的 |
|---|---|---|---|
| PACI | NCSA | 1998~ | Gloubs を普及させ、分散した研究所支援のための計算グリッド構築 |
| Information Power Grid | NASA | 1999~ | 異機種分散コンピューティングシステムにより Multi-disciplinary Simulation を実行 |
| Access Grid | ANL,LBNI,LANL,etc | 1999~ | グリッド上に遠隔会議システムなどコラボレーション用システム開発・運用する |
| Grid Physics Network | ANL,U of Florida,etc | 2000~ | 高エネルギー物理や天文学におけるデータの共有化 |
| Particle Physicss Data Grid(PPDG) | ANL, LBNL,Caltech,SDSC | 2000~ | 素粒子物理データの共有化 |
| Tera Grid | NCSA, SDSC,ANL | 2000~ | OnePetabyteGateway の構築 |
| International Virtual-Data Grid Lab | U of Florida Uof Chicago, etc | 2001~ | 米国、欧州、日本等のデータグリッドを統合し技術開発、普及をはかる |
| Network For Earthquake Eng Simulation Grid(NEES) | U of Southern California, ANL, NCSA,etc | 2001~ | 地震シミュレーション |
| National Virtual Pbservatory(NVO) | Johns Hopkins U, etc | 2001 | 仮想宇宙観測研究所 |
| Grid | U of Chicago, etc | 2001~ | グリッド基本ソフトウェアの統合運用サポート |
| Fusion Grid | ANL,LBNL,etc | 2001~ | 核融合研究のグリッド |
| Earth Systems Grid | ANL,LINL,NCAR,etc | 2001 | 気象予測のグリッド |

(2) 英国、欧州でのグリッドプロジェクト

英国では 2000 年度から e-Science プロジェクトを開始している。このプロジェクトのねらいは科学研究を IT により推進加速させることにある。具体的にはグリッド技術をベースにして、高価なコンピュータや高価な実験装置の共同利用、世界的に分散した大規模データの共同利用を実現する。プロジェクトを精力的に進めている。グリッド基盤としては、英国 9 カ所に国立 e-Science センターを設立し、高性能コンピュータを高速ネットワークにより接続したグリッドを構築している。このグリッド上につくられる応用は、高エネルギー物理、ゲノムバイオ、タンパク質構造解析、医療・保険、環境、機構、宇宙、化学など多岐にわたっている。E-Science のリーダである。John Taylor(Director General Reacarch Councils UK OST)は「グリッドは英国にとって国際的な科学研究に参加するための基盤である」と言っている。

欧州には、スーパーコンピュータのメーカはないが、スーパーコンピュータ応用の蓄積亜 h 大きくグリッド応用への取り組みはやい、EU の代表的なプロジェクトとしては大規模デ

ータを取り扱う EU Data Grid がある。CREN が中心となり米国、日本とも連携を取って進められている。EU のプロジェクトは EU 加盟国の複数国の共同研究が条件であるが、グリッドは共同研究の基礎を構築するものなので、EU プロジェクトによく適合している。一方、英国以外の欧州各国でも、ドイツの Unicore、イタリアの INFN Grid(Italian National Institute for Research in Nuclear Physics)、オランダの Grid-based Virtual Laboratory Amsterdam(VLAM)などのプロジェクトが進められている。

**表 2 英国主要グリッドプロジェクト**

| プロジェクト | 機関 | 機関 | 目的 |
|---|---|---|---|
| e-Science | | 2001～ | 化学研究促進のためのグリッド開発の全体プログラム |
| Grid Particle Physics | Universities of Birmingham,Bristol,Cambridge, Edingburg,Glasgow,Lancaster,Liverpool,etc | | EU DataGrid、USGriPhyN と PPGrid と共同で素粒子物理学研究 |
| Astro Grid | Universities of Edimburgh,Leicester,Cambridge,Queens Brelfast,etc | 2001～ | EU AVO と US NVO と共同で宇宙観測研究 |
| Grid in 6th Framework Program | | 2003 | FP6 全体は IT、バイオ、ナノ、食料、宇宙、環境、エネルギー知的社会が対象 |
| EU Data Grid | CERN, Uof Heidelberg,IBM UK,etc | 2001 | ペタバイトのデータ処理をリアルタイムで実行するネットワーク |
| Euro Grid | | 2001 | グリッド応用開発：分散生物モデル、気象予測 CAE エミュレーション、グリッドミドルウェア |
| GRIP | | 2002 | UNICORE と GLOUBS |

(3) 日本でのグリッドプロジェクト

日本では 2001 年から開始された ITBL(IT-Based Laboratory)プロジェクトでは、日本原子力研究所と理化学研究所を中心に仮想研究室の構築を目指し、スーパーコンピュータを通信回線で接続し共同研究の環境を作っている。グリッドを全面にだしたプロジェクトとしては 2001 年度補正予算にて、産総研グリッド研究センター設立、東工大キャンパスグリッド構築などが動き出した 2002 年には阪大を中心としたバイオグリッドプロジェクトが開始され、また、化学研究費「ゲノム情報科学」研究の中で、東大、北陸先端大、徳島大、理化学研究所などによるバイオインフォマティクス用グリッド構築が開始されている。2003 年には世界高水準のグリッド構築を目指したナショナル・リサーチグリッド・イニシアティブや、

グリッドによる柔軟なサービスの提供するソフトウェアを開発するビジネスグリッドプロジェクトなど本格的なグリッド構築が計画されている。なお 2002 年から開始されたスーパ SINET プロジェクトによって、大学や国立研究所の計算センターが高速ネットワークにせつぞくされ、これらはグリッド構築のための重要な基盤となっている。

アジア地区では韓国、中国をはじめ、台湾シンガポールなどでグリッド研究が進められている。米国、オーストラリアを含めたアジア太平洋地区のグリッド活動を推進する母体としてアジア太平洋グリッド APGrid とアジア太平洋グリッド応用会議(PRAGMA : Pacific-Rim Application &グリッド-Middlware Assembly)があり、日本がリード役となってこの地区のグリッド活動を推進してきている。また Asia-Pacific Advanced Network(APAN)の活動によって、グリッドのペースとなる高速ネットワークが構築されてきている。このような活動の実績から、世界の標準化団体グローバル グリッド フォーラム GGF)内でアジアが世界 3 局のひとつとして発言権を確保できている。アジア各国とも積極的にプロジェクトを立ち上げてきており。キャッチアップは速い。

**表 3 日本主要グリッドプロジェクト**

| プロジェクト | 機関 | 期間 | 目的 |
|---|---|---|---|
| IT-Based Laboratory | 原研、理研、他 | 2000～ | スーパ SINET を介してスーパーコンピュータを接続した仮想研究環境の構築 |
| 科研費特定領域研究 | 阪大、東工大、国立天文台、他 | 2001～ | 医療グリッド阪大下条研、P2P データグリッド東工大松岡阪、国立天文台、 |
| 東工大キャンパスグリッド | 東工大 | 2001～ | 学内キャンパスグリッドの構築 |
| 阪大バイオグリッドセンター | 阪大 | 2002～ | グリッドインフラ構築 |
| E サイエンス実現プロジェクト「スーパーコンピュータネットワーク構築」 | 阪大、他 | 2002～ | 大規模データグリッドとコンピューティングの連結技術の開発 |
| ネットワークコンピューティング技術開発「グリッドクラスタフェデレーション」 | 産総研、東工大、筑波大、東大 | 2003～ | クラスタ技術、グリッド技術、10 ペタバイト級の大容量データ処理の実現 |
| ナショナルリサーチグリッド・イニシアティブ | 計画中 | 2003～ | 世界最高水準のグリッド環境をこうちくし、バイオ、ナノ分野等と情報通信分野との連携の基で行う融合領域研究を推進 |

# 3　グリッドミドルウェア

## 3.1　　グリッドミドルウェアについて

グリッドミドルウェアとはグリッド環境を構築､又はシステムを提供するソフトウェア群である。ソフトウェアとしては次にあげるようなソフトウェアがる

- GlobusToolKit
  -Globus Project(USA)
  -http://www.globus.org/
- Legion
  -Legion Project: University of Virginia(USA)
  -http://legion.virginia.edu/
- STA
  -ITBL(JAPAN)
  -http://www.itbl.jp/
- UNICORE
  -UNICORE Project(EU)
  -http://www.unicore.org/

## 3.2　　Globus プロジェクトによる Globus ToolKit

(1)　Globus TookKit とは

GlobusToolKit(以下 Globus)は米アルゴンヌ研究所と南カルフォルニア大学情報科学研究所を中心に構成されている The Globus Project が開発しているグリッド環境を構築ためのミドルウェアである。The Globus Project によって開発されている。Globus はグリッドコンピューティング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を提供する低レベルなツールキットであり、Globus が提供するツールを用いて上位レベルにグリッド コンピューティングシステムを構築することができる

(2)　Globus の目的

The Globus Project の目的は、共同作業を行う組織間でのコンピュータ、ストレージデータ､アプリケーションの共有を可能にすることによって､コンピュータの新しい利用方法をうみ出すことにある。Globus はオープンソースのプロトコル、サービス及びツールセットにより構成されるツールキットであり､透過的で階層的分散マルチベンダーグリッドコンピューティング環境を実現できる。Globus ディレクトリサービスはグリッド上でのリソースを特定、それらのリソースの管理、選択確保を容易とし、リソースの構成、機能、ステー

タスへの統一された quely 方法を提供する。また、グローバル・グリッド・フォーラム(GGF)におけるグリッド・コンピューティングのプロトコルと API の標準化を行っている。

(3) Globus のアーキテクチャ

Globus のアーキテクチャを図 2に示す。Globus のアーキテクチャは上から

- Application,
- High-level Services and Tools
- Core Services,
- Local Services

から構成される。また Core Services についての詳細は表 4である

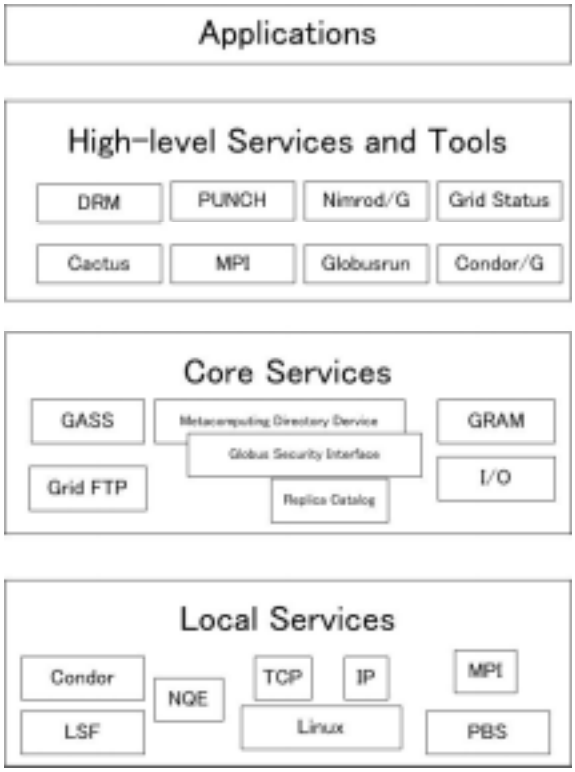


**図** 2 Globus **のアーキテクチャ**

表 4 Globus の Core services

| サービス | 名前 | 概要 |
|---|---|---|
| 資源管理 | GRAM | リソースの割り当ておよびプロセス生成 |
| 通信 | Nexus | Unicast/Multicast 通信サービス |
| 情報 | MDS | システム構造および状態に関する情報へのアクセス |
| セキュリティー | GSI | authentication などのセキュリティーサービス |
| 状態管理 | HBM | システム状況サービス |
| 遠隔データアクセス | GASS | データへのリモートアクセスサービス |
| 実行ファイル管理 | GEM | 実行ファイルの構築、キャッシングおよび配置 |

## 3.3 Globus の機能

Globus ToolKit はグリッド環境において計算機のための 3 つの要素を提供する。まず、ジョブのリモート実行を可能にするリソース管理である GRAM(Globus Resource Allocation Manager)グリッドを構成する各ノードが提供するリソースの割り当て機能も含む。第二は情報サービス MDS(Monitoring and Discovery Service)でグリッド構成に関する動的・静的な情報を提供する。そして最後はデータ管理サービス DMS(Data Management Service)ありグリッド環境にあるデータへのリモートアクセス管理を行う。さらにこれらの要素をつなぎあわせるのがセキュリティ基盤 GSI(Grid Security Infrastructure)である。次項以降では Globus の上で実行される際に使われる機能とその状態遷移を順におって説明する。[4]

### 3.3.1 セキュリティ(GSI)

GSI は公開鍵暗号方式を採用指定すべてのユーザとサービスが公開額及び秘密鍵を持つ GSI には以下のような用件が求められている。

- グリッドのリソース間でのセキュアな通信
- 組織間の境界をこえたセキュリティのサポート。グリッドのためには集中管理的なセキュリティーシステムは利用できない
- グリッドのユーザに対するシングル・サイン・オンのサポート-マルチリソースやサイトを含めた計算のための証明書の委譲も含まれる。

(1) 証明書(Certification)

GSI 認証の中核となるコンセプトは、証明書である。グリッド上のすべてのユーザやサービスが証明書によって識別される。その証明書には、ユーザやサービスを認証し識別するために、つぎのような重要な情報が含まれている。これらの情報は X.509 証明書フォーマットでエンコードされる。

・ 誰であるかのサブジェクト名
・ その公開鍵
・ 証明書に署名している認証局(CA)の ID
・ 認証局からの証明書

(2) 相互認証(Mutual Authentication)

グリッド上の相互のパーティが証明書をもっていて、両方のパーティがお互いに相手を証明書を署名した認証局を信用しているのであれば、二つのパーティはお互いに信用することができる。このことを相互認証と呼ぶ。GSI は SSL(Secure Sockets Layer)通信によって相互認証を行う。

(3) 暗号通信(Confidential Communication)

デフォルトでは、GSI はパーティ間での秘密通信は実施しない。ひとたび相互認容が実行されたなら GSI は切りはなされるため、暗号化や複合かのためのオーバーヘッドなしで通信が可能となる。

(4) セキュアな秘密鍵(Private Key)

Globus ToolKit により提供される GSI ソフトウェアのコアは、利用者の秘密鍵がローカルコンピュータのストレージのファイルに格納されていることを前提としている。そのコンピュータ
のストレージのファイルに格納されていることを前提としてる。そのコンピュータの他の利用者が秘密鍵を盗まれないようにするために、その鍵はパスワードで暗号化されている。GSI の利用するには、利用者はパスワードを入力し、プライベートキーを含むファイルを復号化しなければならない

(5) 委譲(Delegation)とシングル・サイン・オン(Single Sign On)

GSI では標準的な X.509 代理証明書を使うことにより委譲が可能である。これによって利用者に呈して何度もパスワードを入力する手間を省いている。もしグリッドでの計算において、複数のグリッドのリソースを利用する場合、あるいは利用者にかわってサービスを要求するエージェントが存在する場合に、プロキシ(Proxy)を設定することにより利用者のパスワードを再入力する必要はなくなる。このような GSI 機能をベースにすることにより、組織を超えた環境において以下に述べるグリッドの残りの 3 つのサービスを安全に提供することが可能となる。

### 3.3.2 資源管理(GRAM)

グリッド上で複数のリソースを活用して計算を行う場合に、全体のリソースを割り当てることは極めて重要である。Globus ToolKit2.0 にはこのためのツールとして GRAM が用意されている。

(1) ジョブの実行の仕組み

GRAM は遠隔でのジョブ実行、モニタリング、ジョブの実行の終了のための API を提供している。GRAM を使ったジョブ実行の流れを図 3に示す。

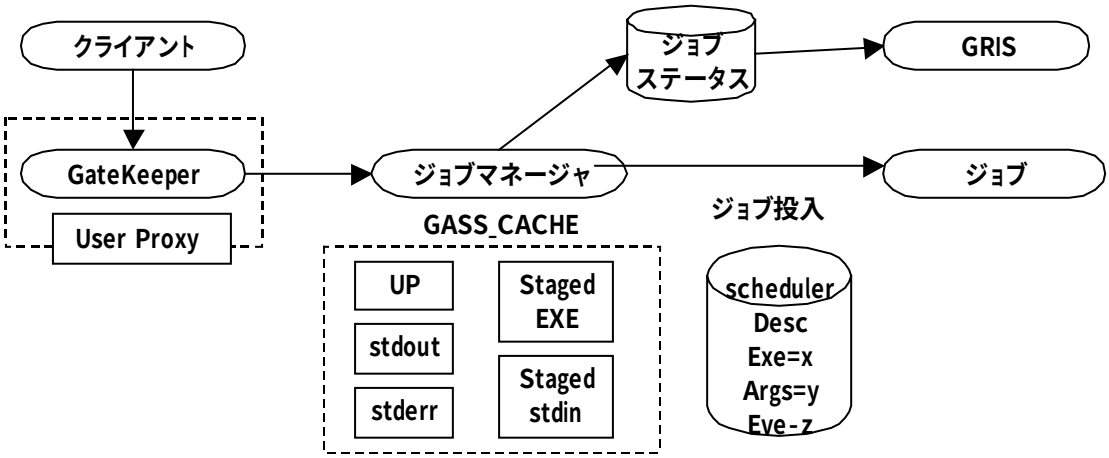


**図 3 ジョブの実行の仕組み** [6]

クライアントにおいてジョブが投入されると、ジョブの実行の要求がリモートコンピュータの GateKeeper に送られる。GateKeeper は要求内容を解釈し、それのジョブに対するジョブマネージャ(Job Manager)を生成する。ジョブマネージャがリソースに対してジョブを投入する。同時にジョブマネージャは遠隔プログラムを起動し、ユーザにジョブの状態の変化を通知できるようにモニタする。アプリケーションが終了したときにジョブマネージャも終了する。このジョブの実行処理のなかで GRAM は次の役割を果たす

- ジョブの要求の概要を記述する言語 RSL(Resource Specification Language)の解釈と処理この RSL によりリソースの選択、ジョブプロセスの生成、そしてジョブの制御が記述される。
- 遠隔監視とすでに生成されたジョブの管理
- ローカル増井の状態を監視するための情報を持つ MDS(後述 2.3.3)を更新

(2) RSL

RSL はリソースを記述するための共通交換言語である。GRAM の様々なコンポーネントがシステム内の他のコンポーネントと協力してマネージメント機能を実行するための RSL ストリングを操作する。RSL には複雑なリソースを記述するためにスケルトンシンタックスが用意されており様々なソース管理コンポーネントが､共通に属性値というペアの使用を導入している。RSL によってユーザはリソース要求やパラメータを指定することができる。ジョブをリモートノードで実行するためには GateKeeper がそのリモートノードで動作していなければならない｡アプリケーションはリモートのマシンにおいてすでにコンパイルされている必要がある。ジョブ要求のなかで stdin や stdout をしていすることができる。

ジョブ要求は最初にリモートノードの GateKeeper により処理される。その結果、新しいジョブに対してジョブマネージャが生成される｡ジョブマネージャはユーザとの様々な通信とジョブの実行を取り扱う

(3) GateKeeper

ジョブを遠隔から実行するために､リモートホスト上でルート権限で走っているプロセスである。GateKeeper はジョブ要求が投入される前にリモートコンピュータの上に存在しなければならない､GateKeeper はクライアントノードからアロケーションリクエストを受けとると以下の処理を行う。

- クライアントノードとの間で相互認証を実施
- リモートのリクエスター(requestor)をローカルユーザにマッピングする。
- ローカルホスト上で、ローカルユーザに対してジョブマネージャを起動
- 新しく生成されたジョブマネージャに対してリソースの割り当てを依頼

各種アプリケーションはリモートマシン上でコンパイルされていてもよい

(4) ジョブマネージャ

ジョブマネージャは GateKeeper に投入された要求を実行するために GateKeeper によって生成される。ローカルシステムの上でジョブが起動し､それ以降のクライアントノードとすべての通信を処理する。

(5) リソースモニタ(Resource Monitor)

ローカル･ジョブスケジューリングシステムとリソースをモニタする。ジョブマネージャに対して推定遅延時間を通知する。

### 3.3.3 情報サービス(MDS2.1)

組織的､地理的に分散した複数のシステムから構成される一つの一貫したシステムの上で動作する｡アプリケーションやサービスにとってそのシステムに関する情報のためのインフラストラクチャは必須のものである｡ここでの情報サービスに対してはつぎのような要求がある。

- システムコンポーネントに対する静的・動的な情報へのアクセス
- ヘテロジニアスでかつ動的な環境においてリソース連携の構築と改造に対する基礎データの提供
- 情報への統一された柔軟なアクセス
- スケーラブルで動的なデータへの効率的なアクセス
- マルチ情報リソースへのアクセス
- 非集中管理

この情報インフラの一部として Globus においては MDS(Monitoring and Discovery Service)がグリッドのためにディレクトリサービスを提供する。MDS はグリッドとそのすべてのコンポーネント状態についての静的かつ動的な情報を管理するために拡張フレームワークを使う｡MSDによってどのリソースが利用可能なのかグリッドの状態はどうなのか、現在のシステム状態および構成をベースにしてアプリケーションを最適化するためにはどうしたらよいのかといった情報を提供する。このためのインターフェイスとして LDAP(Lightweight Directory Access Protocol)を適応している。
MDS は情報提供コンポーネントとして GRIS(Grid Resource Information Service)と GRIS で取得した情報を収集するためのコンポーネントとして GIIS(Grid Index Information Service)を提供する。GRIS は以下の情報を取得する。

- IP Adress
- OS の名前とバージョン
- CPU に関する情報:タイプ、CPU 数、バージョン、速度、キャッシュ
- メモリ：物理・仮想メモリのサイズ、空き等
- ネットワークインフラ情報：マシン名とアドレス
- ファイルシステム概要：サイズ、空き容量

GRIS と GISS の関係のアーキテクチャを図 4に示す。

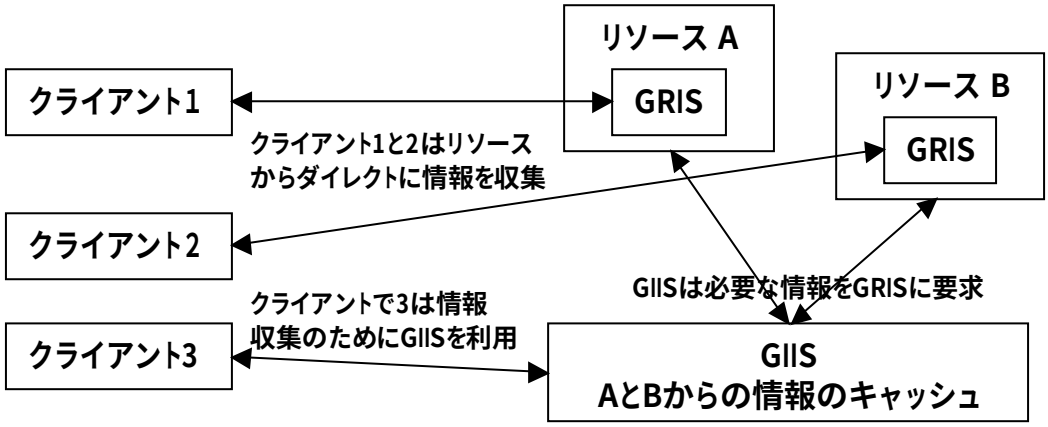


**図 4 グリッド情報サービス**

GRIS は特定のリソースに関する情報を収集し、複数の情報プロバイダに提供可能である。問い合わせには LDAP プロトコルを用いる。GIIS は複数の GRIS サーバで収集した情報を効率的に提供することを支援する。

### 3.3.4 データ管理(DMS)

Globus ではデータ・グリッド環境を構築するためには新しいコンポーネントが含まれている。データ管理サービスは大きく 2 つのコンポーネントで構成される。

・ データ転送とアクセス。
- GASS ：シンプル他マルチプロトコルファイル転送ツール
- GridFTP ：WAN に呈しても高性能で安全なデータ転送プロトコル

・ データレプリケーションと管理
- レプリカカタログサービスとレプリカ管理サービスを提供
- GASS はデータファイルや実行ファイルのステージングや I/O リダイレクションを行う
- GRAM と強調して動作する。GSI 対応の HTTP をデータ転送プロトコルとして使用。必
- 要に応じてコピーしたデータをキャッシュする。

# 4　Globus のインストールとテスト

## 4.1　　Globus のインストール

ここでは Globus をインストールするための手順について説明する。なおここでは Globus ToolKit2.2 をもとにして説明している。Globus2.2 ではインストールの際に GPT(Grid Packaging Toolkit)というインストーラが用意された。Globus では GPT パッケージの集まりを bundle と呼びソースとバイナリの２つが容易されている。これらのパッケージは前章で述べた。リソース管理、情報サービス、データ管理サービスの３つに分かれており、それぞれには Client、Server、SDK と 3 つのタイプに分かれ配布され、計 9 種類のパッケージがある。またバンドルをインストールした後は次の 3 種類の証明書(CA)を取得しなければならない

・ User 証明書：Globus を使うユーザは必ずこの証明書の発行が必要
・ HOST 証明書：ホストを Globus のサーバとして使うために必要
・ LDAP 証明書：Globus をつかったファイル転送(GridFTP 等)を使用するときに必要

では実際のインストール手順を説明する。

(1)　システム構成と環境変数のセット

・ システム構成

Globus の動作が確認されているのは Globus Project によると以下の環境である。

- □ Linux Kernel 2.x Intel x86
- □ Linux Kernel 2.4 Intel IA-64(Itanium)
- □ IRIX 6.5 MIPS
- □ Solaris7 UltraSPARC
- □ Solaris8 UltraSPARC
- □ AIX 5.1
- □ HP Tru64 5.1

今回、実験として使用した PC と OS 等の計算機リソースは表 5である。

**表 5 計算機リソース**

| | |
|---|---|
| ホスト名 | Chicken.hpc.cs.ritsumei.ac.jp |
| OS | Redhat7.3 (kernel-2.4.18-18.7.x) |
| CPU | Intel(R) Pentium(R) 4 CPU 2.53GHz 2nd chash 512KB |
| Memory | 1.5GB (DirectRDRAM) |
| NIC | Intel Ethernet Pro 100 |
| 備考 | /home は nfs でマウント |

・ Globus に必要なソフトウェア

Globus は C と Perl によって書かれている。そのため Globus のインストールには Perl5.0005 以上が必要となる。また、明示的には書かれていないが、gcc、make、といったコンパイラ関係の開発環境のものも入れておかなければならない。

・ Globus ユーザの作成

Globus では gloubs というユーザを作成し、インストールはすべてそのユーザで行うことを推奨している。もちろん root や個人ユーザでもインストールは可能だがこれはセキュリティーの問題となるためセキュリティー機構が高い Globus では避けるようにと Globus のドキュメントには書かれている。

・ 環境変数

環境変数 GLOBUS_LOCATION という環境変数が必要となる。GLOBUS_LOCATION で指定した場所に Globus を構築・インストールすることができる。また、GPT_LOCATION も設定する。これは Globus と Globus ツールとを分けてインストールするために設定する。今回は両方を/usr/local/globus に設定をした。また.bashrc に以下のように環境変数を追加する。

```
export GLOBUS_LOCATION=/usr/local/globus
export GPT_LOCATION=/usr/local/globus
```

・ インストールディレクトリ

最後に Globus をインストールするディレクトリを root になって作成する。このディレクトリは上記の環境変数で設定した GLOBUS_LOCATION のディレクトリを指定する。またこのままでは globus ユーザでの書き込みができないためディレクトリのオーナを変えてやる必要があるなお以下から#で始まるのはスーパーユーザ、$は globus ユーザ、kinosita$は kinosita というユーザでの実行をあらわす。

```
$ su -
# mkdir /usr/local/globus
# chown globus /usr/local/globus
```

以上で globus をインストールするための環境が整った。続いて GPT のインストールを行う。

(2)　GPT のインストール

Globus のホームページから Gpt-2.2.2.src.tar.gz をダウンロードしてくる。次に以下のコマンドを実行する。Gpt のインストール先は$GPT_LOCATION で設定した場所にインストールされる。また gpt は perl を内部で呼び出しているために PATH を設定してやる必要があるが通常 Linux では Perl は/usr/bin/perl にインストールためデフォルトで動くがそれ以外の場所に perl がインストールされている場合は./build_gpt –with-perl=”perl PATH”としてやる

```
$ tar zxvf gpt-2.2.2-src.tar.gz
$ cd gpt-2.2.2
$ ./build_gpt
```

bundle のインストール

bundle を Globus のホームページからダウンロードする。Bundle は Management、Information Services、Data Management の３つに分けてありそれぞれには Client、Server、SDK と３つに分けて配布されている。これらはバイナリとソースの２種類づつある。今回はソースからのインストールを行った。また bundle のそれぞれの flavors は表 6 のようになっている。インストールコマンドは以下とおりである。

```
・バイナリから
$GPT_LOCATION/sbin/gpt-install [option] [bundle-name] [flavor(s)]
・ソースから
$GPT_LOCATION/sbin/gpt-build [option] [bundle-name] [flavor(s)]
```

**表** 6 bundle **と** Flavor

| **Bundle** | **Flavors** |
|---|---|
| Data Management Client | Gcc32dbg |
| Data Management Server | Gcc32dbg |
| Data Management SDK | Gcc32dbg |
| Information Services Client | Gcc32dbgpthr |
| Information Services Server | Gcc32dbgpthr |
| Information Services SDK | Gcc32dbgpthr |
| Resource Management Client | Gcc32dbg |
| Resource Management Server | Gcc32dbg |
| Resource Management SDK | Gcc32dbg |
| Replica | Gcc32dbgpthr |
| GSI | Gcc32dbg |

実際に入力したコマンドは次の通りである。

```
$GPT_LOCATION/sbin/gpt-build
globus-data-management-client-2.2.3-src_bundle.tar.gz ¥
gcc32dbg
$GPT_LOCATION/sbin/gpt-build
globus-data-management-server-2.2.3-src_bundle.tar.gz
gcc32dbg
$GPT_LOCATION/sbin/gpt-build
globus-data-management-sdk-2.2.3-src_bundle.tar.gz gcc32dbg
$GPT_LOCATION/sbin/gpt-build
globus-information-services-client-2.2.3-src_bundle.tar.gz
gcc32dbgpthr
$GPT_LOCATION/sbin/gpt-build
globus-information-services-server-2.2.3-src_bundle.tar.gz
gcc32dbgpthr
$GPT_LOCATION/sbin/gpt-build
globus-information-services-sdk-2.2.3-src_bundle.tar.gz
gcc32dbgpthr
```

(3) Configure the installationeiga

Bundle のインストール完了後$GLOBUS_LOCATION/sbin/gpt-postinstall を実行して ToolKit 専用にカスタマイズする。

```
$GLOBUS_LOCATION/sbin/gpt-postinstall
```

最後に setup-gsi をする

```
$GLOBUS_LOATION/setup/globus/setup-gsi -nonroot
```

以上で Globus のインストールは終了である。

## 4.2 CA の取得と設定

Globus を使うために CA 等の取得する必要がある。Globus を使用するにあたり環境変数等を整えるために以下のコマンドをログイン時に入力する。またこれらの入力の手間を省くために.bashrc に書いておく

```
[csh 系] source $GLOBUS_LOCATION/etc/globus-user-env.csh
[bash 系] . $GLOBUS_LOCATION/etc/globus-user-env.sh
```

Globus を使用するには次の３つの certificate が必要となる

- User certificate
- Host certificate
- LDAP certificate

GRAM や GridFTP を使う際には Host certificate が必要となり MSD を使う際には LDAP certificate が必要となる。Certificate は Globus を再インストールする際にも使い回すことができるため保管しておく必要がある。誤って消してしまった場合には CA の再発行が必要となり認証局に対して再申請しなければならないまた Globus は自分でつくった CA を用いることも可能である。

1. User certificate

User certificate はそのユーザが正規のユーザーであることを示すためのものなので root といった特殊なアカウントではなく個別のユーザで行う必要がある。以下は User certificate を取得する時の応答である。

```
[kinosita@chicken ~]$ grid-cert-request
Enter your name, e.g., John Smith: Minoru kinosita
A certificate request and private key is being created.
You will be asked to enter a PEM pass phrase.
This pass phrase is akin to your account password,
and is used to protect your key file.
If you forget your pass phrase, you will need to
obtain a new certificate.

Using configuration from /usr/local/globus/etc/globus-user-ssl.conf
Generating a 1024 bit RSA private key
.....++++++
```

上記のコマンドによりホームディレクトリに.globus というフォルダが作られる。その中にある usercert_request.pem を ca@globus.org に送くる。このファイルは特殊なファイルであるためメールソフトなどで添付するのではなく次のコマンドを入力し sendmail といった MTA が動いているホスト上で送る。このときメールを送るホストは certificate を作ったドメインないであることに注意する。

```
kinosita@pegasus~]$ cat /home/kinosita/.globus/usercert_request.pem
¥ | mail ca@globus.org
```

Globus では 2 営業日後にメールが来ると書いてあったが、たいがい次の日にはメールは来ている。これをそのまま内容を変えずに~/.globus/usercert.pem に保存する。このとき元々あるファイルを上書きするかと注意されるがこれは pem 作成時に自動的に生成されるものであり容量は０バイトなので消してしまってかまわない。

2. Host certificate

GateKeeper や GridFTP を動かしたいときには Host certificate が必要となる。また Globus は DNS に登録してある名前でマシン名を解決するため DHCP で動いているマシンでは Globus は動かないので注意する。次のコマンドを入力し Host certificate を取得する

```
$grid-cert-request –service host –host chicken.hpc.cs.ritsumei.ac.jp
```

これも上記の User certificate と同様の手順にて $GLOBUS_LOCATION/etc/hostcert_reques.pem のファイルを mail にて ca@globus.org に送る。２営業日後に送られてくるメールを$GLOBUS_LOCATION/etc/hostcert.pem に保存する。このときこのファイルのパーミッションを 600 に変更する。

3. LDAP certificate

LDAP サービスを動かしたいときは root になって次のコマンドを実行する。

```
# grid-cert-request -service ldap -host chicken.hpc.cs.ritsumei.ac.jp
```

そして User,Host certificate と同様に ca@globus.org に $GLOBUS_LOCATION/etc/ldap/ldapcert_request.pem のファイルをメール送ります。作成時は root であるがメールを送るときには root ユーザで送らないように注意する。２営業

日後に送られてくるメールを$GLOBUS_LOCATION/etc/ldap/ldapcer.pem に保存するこれも同様にパーミッションを 600 にする。以下はそのときの応答である。

```
[root@chicken ~]# grid-cert-request -service ldap -host
chicken.hpc.cs.ritsumei.ac.jp
Using configuration from /usr/local/globus/etc/globus-host-ssl.cont
Generating a 1024 bit RSA private key
....++++++
.....................................................++++++
writing new private key to '/usr/local/globus/etc/ldap/ldapkey.pem
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished
Name or a DN.
```

## 4.3 システムテスト

User certificate 等を取得後 Globus が正常に動くか GRAM、MDS、DMS をテストする。DMS については GridFTP をテストして確かめた。また Globus 上でクイックソートのプログラムを動かして実行速度を計測した。

1. GRAM

Gatekeeper、GlobusProxy を次の手順で動作させる。

```
>grid-proxy-init –debug –derify
>globus-personal-gatekeeper –start
>glbous-job-run [contact]※ command
>globus-personal-gatekeeper –killall
>grid-proxy-destroy
※[contact]には gatekeeper 起動時に出力される
```

2. MDS

MDS を次のコマンドで起動する。

```
>$GLOBUS_LOCATION/sbin/globus-mds start
```

このコマンドにより GRIS のための OpenLDAP2.0 が起動する。

次にコマンドで GRIS と GIIS の動作がテストできる。

```
>$GLOBUS_LOCATION/bin/grid-info-search –anonymous -L
```

3. DMS(GridFTP)

まず DMS を起動させる。

```
>grid-proxy-init –debug –verify
>$GLOBUS_LOCATION/sbin/in.ftpd –s –p 5678※ &
※-s は in.ftpd をデーモンで動かし –p にてポート番号を指定
```

また、.gridmap というファイルをホームディレクトリに作りその中に次の１行を書き込む

```
"/O=Grid/O=Globus/OU=your domain/CN=your name" your-account
```

GridFTP の基本コマンドは次の通りである。

```
>$ globus-url-copy sourceURL destURL
```

4. telnet と Globus でのクイックソートプログラム実行

表５の計算機リソースを使い telnet と Globus を使ってクイックソートのプログラムを実行してその実行時間について測定した。コマンドは以下の通りである。なおクイックソートのデータ数はランダムな１～32000 までの数字が１０万個である。入力したコマンドを次の通りである。

```
Telnet > time ./quick_sort > /dev/null
Globus>time globus-job-run ¥
  "chicken:41504:/O=Grid/O=Globus/OU=hpc.cs. ¥
```

**表 7 測定結果**

| 方法 | Telnet | Globus |
|---|---|---|
| Real | 0.052s | 31.125s |
| User | 0.049s | 0.170s |
| Sys | 0.002s | 0.037s |

(単位は ms)

この実行結果をみると telnet をつかった場合と Globus を使った場合とでは telnet を使った方が速度においては優秀な結果である。また他の unix 系の一般的なコマンド ls や cd もこの測定結果に近い時間がかかった。このように Globus 環境ではディレクトリの観覧や移動といったことだけならばいらだちを感じてしまうかもしれない。しかし大規模計算つまり、一つの計算に何日もかかるようなプログラムにおいてはこのような小さい数字は無視できると考えられる。

# 5　グリッドコンピューティングの利用方法の検討

## 5.1　　グリッド環境によるヘテロジニアス化

本来グリッドについての明確な定義は RFC 等で決まっているわけではない。Gird の第一人者であり Globus の開発メンバーである Ian Foster 氏は The Anatomy of theGrid [1] で「Grid とは多々の組織に支配されている計算機リソースを統括するインフラストラクチャ」と述べている。そこで、本研究でグリッドとは「異なる管理下における、異なるネットワーク(場所)での、異なる計算機資源(リソース)」と定義することとする。世の中にある計算機リソース(ここではアーキテクチャ)は次の３つに分類できるといえる。

・　汎用コンピュータ(PC、SPARC 等)
・　汎用クラスタ(PC クラスタ)
・　スーパーコンピュータ(クラスタ構成も含む)

現在、本研究室にあるのは上記の２つである汎用コンピュータと汎用クラスタである。今後はクラスタが普及していくに従ってハードウェアの段階的なアップグレードによるノード性能の不均質(ヘテロジニアス)環境となってくる。また、他研究室との共同研究において汎用クラスタを複数使用するなどヘテロなクラスタ環境は増加する。一般的にこのようなヘテロジニアスな計算機環境においてクラスタ構成を組むと性能の遅い計算機やクラスタに足を引っ張られて速度向上が理論値のように出ない。そこでこのような計算機リソースの拡大におけるヘテロジニアス化についてグリッドを用いて効果的な負荷分散とタスクスケジューリングの解決方法を検討した。また、本研究ではこのような研究室単位における小規模なグリッド構成を小規模グリッドと定義する。今回、このようにモデルを小規模グリッド構成としたのは、小規模グリッド環境における手法を検討することにより問題解決方法を単純にすることをねらいとしている。

## 5.2　　小規模グリッド環境の並列処理

ヘテロジニアスな小規模グリッド環境において負荷分散について次のようにして評価と考察を行う。

(1)　通常の並列処理手法を用いて負荷分散の評価を行う
(2)　ヘテロジニアス環境下での最適な負荷分散の方法と評価を行う
(3)　Globus を用いた実装方法について検討を行う

(1) ヘテロジニアス環境における負荷分散の評価

一般的な並列処理手法としては次のような手法がある。

- 分割統治法
- プロセッサファーム
- プロセスネットワーク
- 繰り返し変換

これらはいずれも同一性能のマシンを複数台用いたときに、いかに均等に負荷分散を行うということでスケーラビリティーが得られる手法である。小規模グリッド環境においてはこのような同一性能マシンが複数台あるということはきわめて希であると考える。一般的にヘテロジニアスな環境において負荷均衡がとれないのは従来の並列処理、並列アルゴリズムの考え方が同一スペック同 CPU という条件下でのアルゴリズムであるからである。それに従おうとするとヘテロジニアスな環境においては逆に負荷均衡がとれないという結果に陥る。

従ってこれらの手法をヘテロジニアス環境に適応させる必要がある。そこで小規模グリッド環境下でヘテロジニアスな計算機、クラスタ構成においての並列化手法を数値計算モデルを用いて検討した。モデル条件は次の通りとする。

- 単純に計算処理能力と通信能力だけに注目して負荷分散をするコスト評価モデルである
- システム構成は図 5である
- 評価コストは式(5.1)により算出される値である
- バリア同期考慮のため評価コストが一番高いのが全体の評価コストとする
- クラスタ内部の通信コストはクライアントからの通信コストに含む
- クライアントのジョブの計算量は１０００とする

$$\text{評価コスト} = \frac{\text{計算量} \times \text{通信コスト}}{\text{計算処理能力}} \qquad (5.1)$$

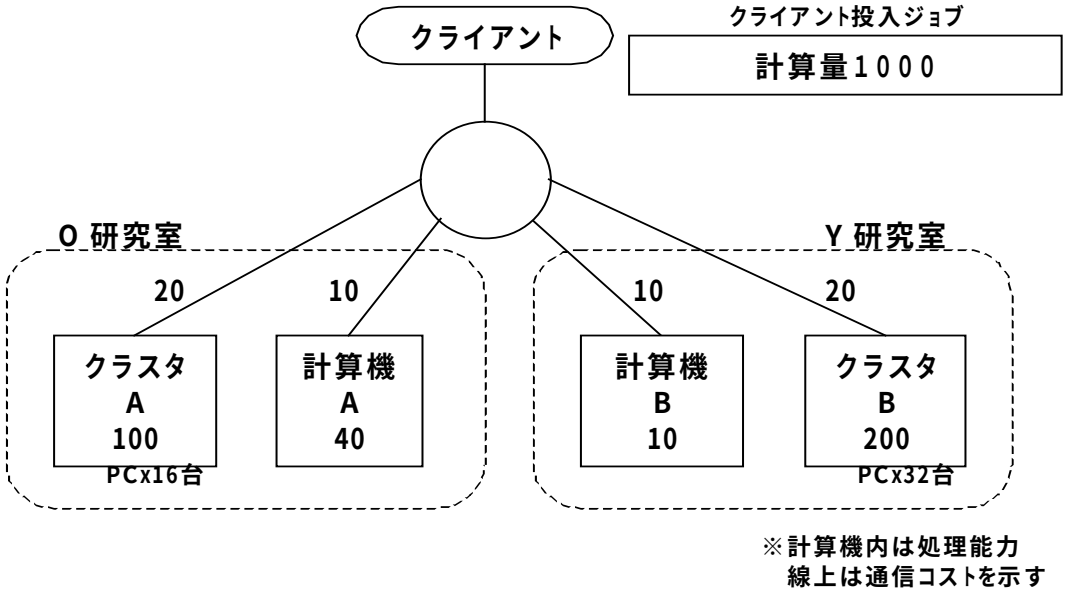


**図 5 ヘテロジニアスなグリッド環境の計算モデルシステム構成**

図 5のようなヘテロジニアス環境下での評価コストを考えた時それぞれの計算機リソース単体での評価コストを表 8に示す。

**表 8 計算機リソース単体での評価コスト**

| 計算機リソース | クラスタ　A | 計算機　A | 計算機　B | クラスタ　B |
|---|---|---|---|---|
| 評価コスト | 200 | 250 | 1000 | 100 |

次に４つの計算機リソースを同時に使用した場合を示す。従来の並列アルゴリズムでは、計算量は計算機リソースの台数に応じて均等に振り分けられる。そのためそれぞれの計算機リソースには全体 1000 に対して１台 250 の計算量のジョブが投入されることになる。その場合４つの計算機リソースの評価コストは表 9のようになる。

**表 9 ４つの計算機リソースを同時に利用した場合の評価コスト**

| 計算機リソース | クラスタ　A | 計算機　A | 計算機　B | クラスタ　B |
|---|---|---|---|---|
| 評価コスト | 50 | 62.5 | 250 | 25 |

モデル条件で最も評価コストの高い値をその評価コストとする。すると計算機 B の評価コストが最もわるく他の計算機リソースは計算機 B の処理が終わるまでバリア同期待ちとなり、結局、評価コスト 250 が最終的な評価コストである。これは計算機 B １台でやった

時に比べて４倍しか並列効果が得られなかったということを意味する。本来、並列処理は計算の力の増加分だけ並列効果が得られるはずである。図 5のモデルの場合、総計算処理能力はで 350 である。そのため並列効果は 35 倍(評価コストとしては 1/35)にならなければならない。このように単純な負荷分散では十分な並列効果が得られないことがわかる。

(2) ヘテロジニアス環境での最適な負荷分散

前節で単純な負荷均衡では十分な並列効果が得られないことを確認した。そこでヘテロジニアスな環境において最適な負荷分散をとるように並列手法を適応させる。小規模グリッド環境においては通信の回数を押さえるかがスケーラビリティーの向上のポイントであうｒ。したがって本研究では計算能力の高い計算機リソースに多くの処理を与えるという手法を提案する。式①より得られる評価コストが４つの計算機リソースに均一になればよい。また、クラスタ A に割り当てる計算量を$X_1$、同様に計算機 A、計算機 B、クラスタ B の計算量をそれぞれ$X_2$、$X_3$ 、$X_4$として式①に代入すると

$$\frac{X_1 \times 20}{100} = \frac{X_2 \times 10}{40} = \frac{X_3 \times 10}{10} = \frac{X_4 \times 20}{200} = \alpha \qquad (5.2)$$

が得られる。この式について解くとそれぞれ

$$40X_1 = 50X_2 = 200X_3 = 20X_4 \qquad (5.3)$$

となる。このとき、各$X$の係数に注目するとこの値は計算能力あたりの計算量１に対しての負荷量となる。また評価コストモデルは高いほどわるいという一般の考えとは逆の値をとるため、この比に対して逆比をとってやると$\frac{1}{5}:\frac{1}{4}:\frac{1}{1}:\frac{1}{10}$となる。この分母比を取り出してやると$5:4:1:10$となり、それぞれの計算機リソースに対しての計算量比が算出される。この分母比を式(5.1)に代入すると

$$\frac{5 \times 20}{100} = \frac{4 \times 10}{40} = \frac{1 \times 10}{10} = \frac{10 \times 20}{200} \qquad (5.4)$$

となる。式(5.4)の比は$1:1:1:1$となり均等な負荷分散ができていることが分かる。

次に実際に値を代入して評価コストを求める。(表 10)

**表 10 ヘテロジニアスに環境での最適負荷分散の評価コスト**

| 計算機リソース | クラスタ　A | 計算機　A | 計算機　B | クラスタ　B |
|---|---|---|---|---|
| 評価コスト | 59 | 59 | 59 | 59 |

表 10よりヘテロジニアス向けに最適化した負荷分散は最終評価個コストは59となった。これは計算機 B が１台でしたときの評価コストに対して 16.9 倍という結果である。これは理想的な並列効果とまではいかないが､ヘテロジニアス環境においても適当な負荷分散をすることにより単体で処理するよりもよい結果を得ることが可能であると言える。

(3) 小規模グリッド環境でのヘテロジニアスな負荷分散処理の実装の検討

評価コストモデルによる結果から小規模グリッド環境における並列化手法として計算処理量を処理能力に応じて分散させることが有効であることが言える｡これはこの負荷分散手法ではプログラムが始まる前に静的に振り分けられることができる利点である。しかし、これはあくまで数値計算による評価コストモデルである｡実際の小規模グリッド環境下においてこれらを実装させる方法を検討する｡このヘテロジニアスな環境でこの並列処理を実装させるのに重要な要素を次にあげる。

1. 各計算機リソースにおける処理能力の把握
2. 計算機リソースのジョブの監視
3. データを各計算機リソースに割り当てる方法

これら３つがヘテロジニアスな環境における最適な並列手法において重要なのではないかと考える｡これらのうち上記２つのリソースの処理能力把握とジョブ監視については３章で述べた
GRAM と MDS で実装することが可能であると分かる。最後に３についてどのようにリソースにデータを割り当てるかについては GRAM の機能に RSL というリソースを記述するための共通交換言語がある。RSL は複雑なリソースを記述するために使うものである。簡単な例を示す。

```
(* this is acomment *)
& (executable = a.out(*< that is an unquoted literal *))
        (directory = /home/nobody)
  (arguments = “arg1” “arg2”)
```

この命令文は、「ディレクトリ”/home/nobody”の中の”a.out”という実行ファイルを引数”arg1”,”arg2”で一度実行せよ」という意味である。

このように Globus ではリソースの引数までの細かな制御がが可能である。従って本研究で提案したヘテロジニアスな環境における最適な並列処理手法は有効であると考えられる。

# 6　おわりに

本論文においてグリッドコンピューティングについての動向を述べるとともにその開発プロジェクトの調査を行った。グリッドミドルウェアの標準である Globus の機能を詳細に調べることにより、グリッドが提供するサービスやシステムを理解できた。また Globus を使ってグリッド環境の構築とテストを行ない、その使用感についても経験することができた。そして、ヘテロジニアスな小規模グリッド環境における最適な負荷分散についての考察と検討をした。

今後の課題として引き続き Globus の機能を調べる必要がある。今回は構築だけのため調査が CoreService 層だけに終わってしまったが、グリッドの本来の効果を得るためには High-level Services 層を利用しなければならない。また負荷分散についても評価計算モデルを用いて行ったが、評価だけでなく実際にはどのように負荷が変わるのかを実験する必要がある。そして負荷分散だけではなくジョブスケジューリングといった要素についての最適解を考察することで、より負荷均衡がはかれると思われる。評価計算モデルについては他の並列処理の要素を加えることにより、さらに実測値に近い値が得られると考えられる。

また、2003 年１月に Globus ToolKit のバージョン 3.0 がリリースされた。バージョン 3.0 から OGSA(Open Grid Service Architecture)が組み込まれた。これは Web サービスを提供するものであり。よりユーザへの透過性と親和性が提供されることになる。今後はこのような Web サービスとの融合性についても検討が必要である

# 謝辞

本研究の機械を与えて下さり、数々の助言を頂きました山崎勝弘教授、小柳滋教授に心より感謝します。また本研究にあたり、いろいろな面で貴重なご意見を、ご指導を頂きました本研究室院生の方々心より感謝いたします。

# 参考文献


[1] Ian Foster,Carl Kesselman,Stevn Tuecke : The anatomy of the grid Enabling scalable virtual organaizations, International Journal of Supercomputer Applications, vol.15 No.3 pp.200-222 ,2001.

[2] Ian Foster and Carl Kesselman : The GRID Blueprint for a New Computing Infrastructure, Morgan Kaufmann,1999.

[3] 三浦謙一 : グリッドコンピューティングの動向について,京都大学学術情報メディアセンター全国共同全国利用版広報 Vol1 No3 167-172, 2002.

[4] 門岡良昌,三浦謙一 : Globus Toolkit2.0 について,京都大学学術情報メディアセンター全国共同利用版広報, Vol.1 No.4 pp.219-224, 2002.

[5] 岸本光弘,三浦謙一 : グリッドコンピューティングの動向(その３),京都大学学術情報メディアセンター全国共同利用広報 Vol2 No3 pp280-287, 2003.

[6] 田中良夫 : Globus Toolkit, JSPP2002 チャートリアル資料 pp.33-76, JSPP2002, 2002.

[7] 金沢大学 田子研究室 : http://superdry.s.kanazawa-u.ac.jp/, 2002.

[8] Globus : http://www.globus.org/, 2002.

[9] GGF : http://www.gridforum.org/, 2002.

[10] グリッド協議会 : http://jpgrid.org/, 2002.

[11] 三好崇之:グリッドの構築, **2002 年度情報ネットワーク分野研究会資料**,2002.

[12] 田中良夫,平野基孝,佐藤三久ほか : Globus を用いたグローバルコンピューティグ環境の構築とその評価,インターネットコンファレンス'99 論文集, pp.97-106,1999.