グリッドコンピューティングの標準的実装技術と検証モデルによる評価

Introduction and Evaluation of Standard Technology for Grid Computing Based on a Verification Model

内 山 忠 夫

要 約 ネットワーク接続された多数のコンピュータを活用し，大規模な処理能力を獲得するための技術として，主にサイエンス分野で使用されてきたグリッドコンピューティング（以下グリッドと呼ぶ）は，ユーティリティコンピューティングなど，次世代企業 IT インフラの要素技術として活用が期待されることから，ビジネス分野においても注目されている．

グリッドとは「広域ネットワーク上の計算，データ，実験装置，センサー，人間などの資源を仮想化・統合し，必要に応じて仮想計算機（Virtual Computer）や仮想組織（Virtual Organization）を動的に形成するためのインフラ」であり，技術的な視点から見ると，ネットワーク上に分散している異機種のコンピュータ・リソースを仮想化してユーザに提供する仕組みと言える．したがって，グリッドの実現には，リソースを仮想化し，更に仮想化したリソースをユーザに提供するための機能が必要である．このような機能を実装し，グリッド環境を構築するために使用されるのが，グリッドミドルウェアである．

本稿では，実績ある代表的なグリッドミドルウェアを用いて実際に構築した実験的グリッド・システムの紹介を通じ，グリッド環境の構築手法の一例を示す．また，同システムの簡単な検証結果を踏まえながら，現在の標準的なグリッド実装技術の特長や問題点について考察する．

Abstract “ Grid computing ”, which has been mainly used for academic fields, is now coming into focus as a core technology for new enterprise systems, such as the utility computing.

The grid computing can be defined as an infrastructure which is used to form a “ virtual computer ” or a “ virtual organization ” by consolidating all kinds of resources on a world wide network, such as computing power, data, experimental devices, sensors, man powers, and so forth. From the technical point of view, grid computing is a mechanism for presenting “ hypothetical resources ” by virtualization of local resources from decentralized heterogeneous computers on networks. Grid middleware is now the standard utility that provides basic functionalities required for building such the grid environment.

This paper introduces an example of method of building grid environment by using some typical grid middleware based on the experience of developing some experimental grid system models, and discusses benefits and problems of current standard technologies for grid computing with referencing the results of brief verification tests on these grid models.

1. はじめに

グリッドコンピューティング（以下グリッドと呼ぶ）は，ネットワーク接続された多数のコンピュータを活用し，大規模な処理能力を必要とするサイエンス分野において，1990 年代から研究が始まった．現在では，主に HPC（High Performance Computing：高速技術計算）に

おける分散コンピューティング技術として，研究機関や民間企業の研究・開発部門で活用されている．また，ユーティリティコンピューティング[*1]など，次世代企業 IT インフラの要素技術としての活用が期待されることから，ビジネス分野においても注目され，IBM 社など多数の有力ベンダが積極的な取り組みを見せ始めている．

このように注目を集める一方で，グリッドは，いまだに研究者やベンダ主体の技術という認識が強く，その解釈も立場によって様々であることから，一般の利用者にとって実態をつかみにくいのが現状である．

本稿では，標準的なグリッドの解釈に基づき，グリッドの実現に必要な機能とは何かを明らかする．また，これらの機能を実装する代表的なグリッドミドルウェアと，同ミドルウェアを用いて実際に構築した実験的なグリッド・システムを紹介し，グリッド環境の構築手法の一例を示す．さらに，同システムの簡単な検証結果を踏まえながら，現在の標準的なグリッド実装技術の特長や問題点について考察する．

2. グリッド概要

2.1 グリッドとは

グリッドという言葉は，電気を伝える高圧送電線網（パワーグリッド）に由来している．コンセントに差し込めばいつでもどこでも必要なだけ電力が得られるように，簡単なインタフェースに接続するだけで，ネットワークを通して，安全に・安定して・安易に様々な情報サービスを享受できるようにするための基盤技術を目指したものである．

グリッド協議会[*2]によれば，グリッドとは「広域ネットワーク上の計算，データ，実験装置，センサー，人間などの資源を仮想化・統合し，必要に応じて仮想計算機（Virtual Computer）や仮想組織（Virtual Organization）を動的に形成するためのインフラ」である（図 1）．

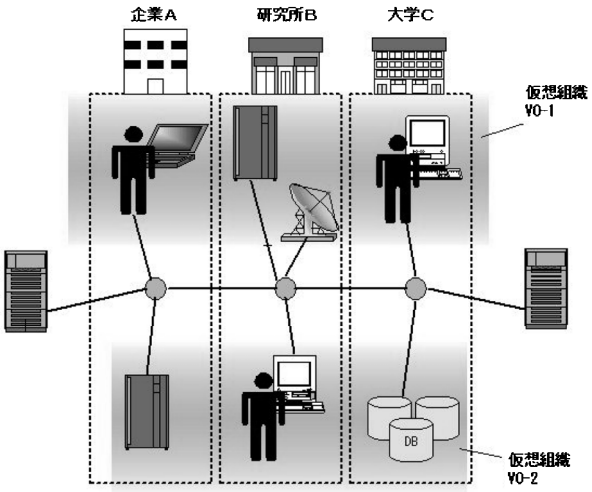

図 1　グリッドによる仮想組織の形成

2.2 グリッドの種類

図 2 は，グリッドの分類の一例である．広い意味では，遊休 PC を活用しようとするいわゆる分散計算型までを含めてグリッドと考えられるが，狭義のグリッドは，柔軟な情報サービス

の提供を目的としたものを指す.狭義のグリッドは主な目的に応じて以下の三つに分類される.

- データグリッド：超大規模データ処理のためのグリッド
- コンピューティンググリッド：高速計算サービスのためのグリッド
- ビジネスグリッド：高信頼ウェブサービスのためのグリッド（グリッドとWebサービスを統合することで，資源の共有に加え，柔軟で自己管理機能を備えた環境を提供する仕組み）

しかし，これらの境界は曖昧で日々変化している．通常は実現しようとするサービスによって，複数のグリッド技術のコンポーネントを組み合わせ構成されたものが一つのシステムとして実行される.

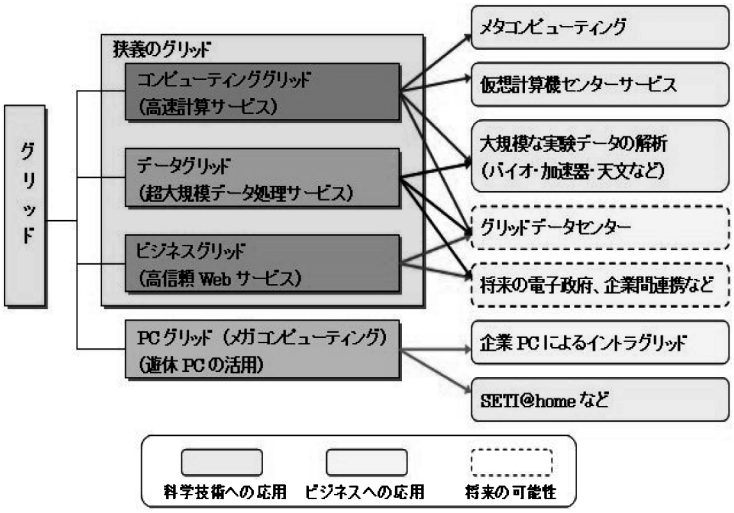

図2　グリッドの種類と応用例

3.　グリッドの仕組み

サイエンス分野で実績を積み上げてきたグリッドは,ビジネス分野への適用が始まっている.特にユーティリティコンピューティングやオートノミックコンピューティング[*3]の基盤として，企業ITインフラを統合する役割が期待されている．しかし，様々なベンダが提供する異機種のリソースを統合するためには，グリッドを構築するための技術がオープンかつスタンダードであることが求められており，グリッド技術の標準化の取り組みが進んでいる.

本章では，グリッドを構築するために必要な標準的な機能とその実装アーキテクチャ，およびこれに基づいて開発されたグリッドミドルウェアの技術について概説する.

3.1　グリッドアーキテクチャ

グリッドを技術的な視点から見ると，ネットワーク上に分散している異機種のコンピュータ・リソースを仮想化してユーザに提供する仕組みである．グリッド研究の第一人者である米アルゴンヌ国立研究所兼シカゴ大学のIan　Foster博士は，このリソースの仮想化から，仮想化されたリソースをユーザに提供するまでに必要な機能を，図3のような五つの階層に分けて説明している.

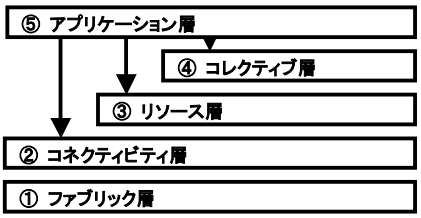

図3　グリッドアーキテクチャの階層モデル

以下は，各階層の概要である．

① ファブリック層

コンピュータのリソースそのものとそのインタフェースを表す．サーバやストレージなどの物理的なものやファイルシステムなどの論理的なものが含まれる．

② コネクティビティ層

ファブリック層にアクセスするためのプロトコルを提供し，組織をまたがった分散環境で各リソースの相互アクセスを可能にする．安全に通信するための暗号化，シングルサインオンによる認証などのセキュリティ機能などが含まれる．

③ リソース層

各リソースに対して，プロセスの実行，データ転送，リソース情報の問い合わせなどをできるようにする．この層でリソースは仮想化されてユーザに提供される．

④ コレクティブ層

仮想化されたリソースの利用を手助けする様々なサービスが含まれる．例えばリソースの情報を収集・蓄積して利用者の要求に最適なリソースを検索できるようにしたり，複数リソースに対してジョブを割り当てたりする機能などである．

⑤ アプリケーション層

直接ユーザが操作・実行するアプリケーションや具体的な処理を行うプログラム（遺伝子解析，気象予測，金融工学，設計シミュレーションなど）が含まれる．

このようにグリッドは様々なレベルで様々な機能を必要としており，これまでの分散処理や並列処理の研究開発で培われてきた技術を集大成することにより実現されている．

3.2　グリッドミドルウェア“Globus Toolkit”の概要

Globus Toolkit は，前節で述べたグリッドの実現に必要な機能を実装したオープンソースのミドルウェアの一つであり，ファブリック層とアプリケーション層をつなぐ役割を果たす．2005年1月現在，Globus Alliance[*4] より，安定版としてバージョン2（以下 GT 2 と略す），およびバージョン3（以下 GT 3 と略す）が正式リリースされており，前者はサイエンス分野で数多くの採用例がある．また，グリッドの国際的標準化団体である GGF（Global Grid Forum）[*5] でも，標準化仕様の参照実装として取り上げられたことから，グリッドミドルウェアの標準的存在となっている．本節では，Globus Toolkit の基本的なコンセプトを，最も普及している GT 2 を中心に紹介する．

3.2.1　Globus Toolkit 2

GT 2 の主要な機能は，図4に示すように四つのコンポーネント（①～④）から成り立って

いる．これら各コンポーネントの機能概要と名称は，次の通りである．

① セキュリティ―GSI（Grid Security Infrastructure）
グリッド・システム全体のセキュリティを担当する．シングルサインオン を実現した PKI（Public Key Infrastructure：公開鍵暗号化基盤）[*6] に基づくセキュリティ機能を提供する．［コネクティビティ層に相当］

② リソースとジョブの管理―GRAM（Grid Resource Allocation Manager）
セキュリティ機能を用いてリモートの計算資源上で実行ファイルを安全に実行する．［リソース層に相当］

③ リソース情報の収集―MDS（Monitoring and Discovery Service）
グリッド環境に関するリソース情報（計算資源のアーキテクチャ，OS，ロードアベレージ，ストレージ資源の残量，ネットワーク資源の現在のスループットなど）をモニタリングし，これを蓄積して検索可能とする．［コレクティブ層に相当］

④ データの管理―GridFTP
セキュリティ機能を用いて安全にデータを転送する．［リソース層に相当］

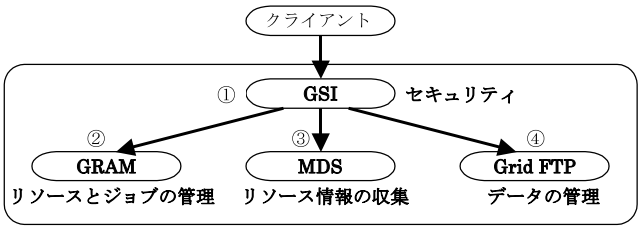

図 4　GT 2 の機能構成

3.2.2　OGSA と GT 3

GT 2 は，サイエンス分野を中心に広く採用され，その意味では成功を収めたが，基盤となる汎用プロトコルがないという問題点があった．例えば，GRAM は HTTP をベースにしているが，MDS は LDAP を用いている．このため，新しくモジュールを追加する際には，プロトコルレベルから構築しなければならない．また，複数のプロトコルが利用されているため，ファイアウォールを超えることができず，特に企業で導入しようとする際に問題となる．そこで登場したのが，2002 年に GGF で発表された OGSA（Open Grid Services Architecture）という新しいアーキテクチャである．OGSA は，Web サービスをベースとして設計したグリッドサービスと呼ばれる一種のコンポーネントを用いてグリッドを構築するアーキテクチャであり，OGSA で Globus Toolkit を再構築したものが GT 3 である．

図 5 に示す通り，GT 2 では，各モジュールがそれぞれ別のプロトコル上に実装されていたのに対し，GT 3 では，HTTP と SOAP を用いたグリッドサービスの上に，全てのモジュールが実装されている．

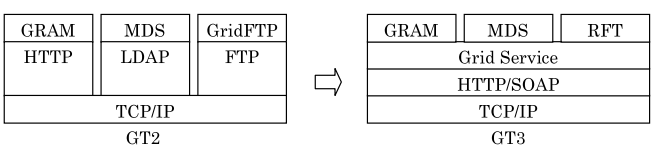

図 5　GT 2 と GT 3 のモジュール構造

4. 検証モデルに見るグリッド環境の構築事例と評価

前章では，グリッドに必要な機能と，これを実装するための標準的グリッドミドルウェアである Globus　Toolkit の概要を説明した．ここでは，グリッド環境の構築事例として，実際に Globus Toolkit 3 をベースに構築した実験的グリッド・システムのモデルを紹介し，その検証結果を踏まえて，現状の標準的グリッド実装技術の特長や問題点について考察する．また，いまひとつの構築事例として，Globus Toolkit の代わりに実績ある商用グリッドミドルウェアを用いて同様のシステムを構築した結果についても紹介する．

4.1 グリッド活用検証モデル―カメラ付き携帯電話メール翻訳システム

カメラ付き携帯電話で撮影した文書画像（英文，ハングル文等）をメールで受け付け，グリッド環境で翻訳するシステムである．（以降，携帯メール翻訳システムと呼ぶ．）全国から頻繁に投入されることを想定した多数の CPU 負荷の高い処理（画像処理，翻訳処理）をグリッド技術で分散処理し，処理 1 件当たりのターンアラウンドタイムを確保することを狙いとしている．

このようなシステム・モデルを選択した理由は次の通りである．

1） グリッドの活用例が多い CPU 負荷分散の検証に適したモデルである．
2） 利用者が急増する携帯電話のサービスと結びつけることにより，一般使用者を意識したビジネス・モデルとしての可能性を模索できる．
3） 画像変換や音声処理など，その他の CPU 負荷の高い処理のサービスにも応用できるモデルである．

図 6 に Globus Toolkit をベースとした同システムの概要を示す．

4.1.1 Globus Toolkit によるシステム構成

Globus Toolkit を基盤とした環境を，グリッドとして実際に機能させるためには，提供される API を用いて Globus　Toolkit の機能を利用し，目的に合わせて協調動作させるためのアプリケーションを作り込む必要がある．

図 7 に，機能モジュールの面から見た携帯メール翻訳システムの論理的システム構成を示す．★が記された部分は，全て Globus　Toolkit によって提供されるため，以下の四つのモジュールについて作り込みを行った．

1） 受信メール 1 件につき，一つのプログラムが起動される．同プログラムは，PHP（Hypertext　Processor）[*7] で作成されたフリーのアプリケーションを利用して開発したもので，受信メールから添付データ（撮影された文書画像）を切り出して共用ファイルに保存し，2）のジョブ振り分けプログラムを起動する．この際，リモート・システムに実行させるジョブ内容（この場合は 4）のデータ処理）を RSL（Resource Specification Language）と呼ばれる XML[*8] 形式のジョブ記述言語で指定する．
2） ジョブ振り分け［ポータルノード上で起動］
　リモート・ジョブ投入インタフェースとして Globus　Toolkit が標準で提供するコマンドライン・プログラム“ managed-job-globusrun ”（Java 実装）を解析し，変更を加えることによって作成した．処理概要は次の通りである．
　① 3）の最適ホスト検索プログラムを呼び出し，ジョブの投入ホストを決定する．

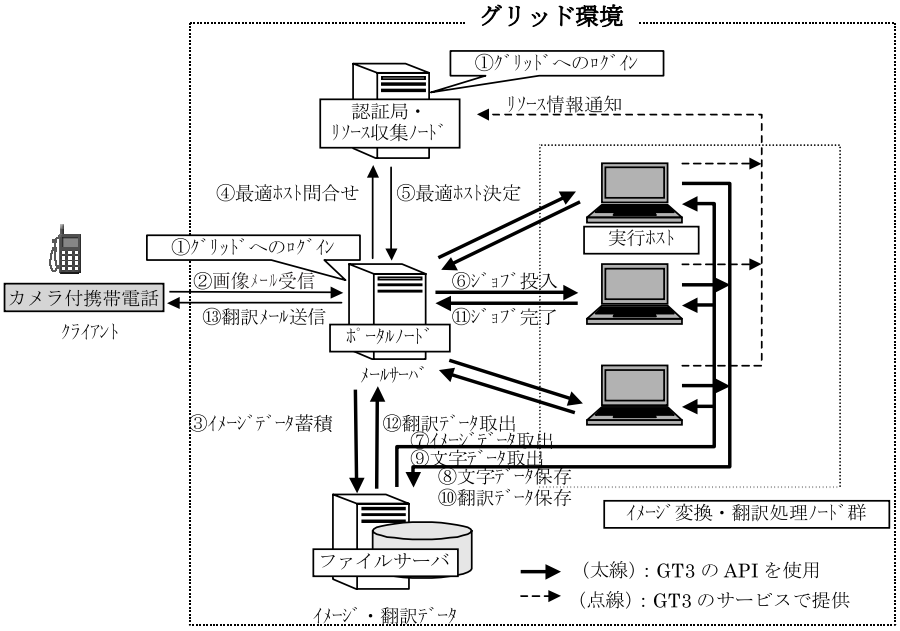

図 6　携帯メール翻訳システム（Globus Toolkit 版）概要図

② Globus Toolkit のリモート・ジョブ投入処理モジュール（GlobusRun）を呼び出し，①で決定したホストに RSL で指定されたジョブを投入する．ジョブ完了，またはタイムアウトで制御が戻る．

③ 結果を読み込み，リクエスタにメールを返信する．エラーの場合はエラー内容のメールを送信する．

3） 最適ホスト検索［リソース収集ノード上で起動］

Globus Toolkit のリソース情報検索サービスを利用して収集した各実行ホストの CPU 情報を元に，実行最適ホストを決定するプログラムを Apache AXIS[*9] ベースの Web サービスとして作成した．処理概要は次の通りである．

① 全実行ノードのリソース情報を取得する Globus Toolkit 標準のコマンドライン・プログラム（grid-info-search）を呼び出す．

② コマンド実行後の標準出力から全実行ノードの CPU 速度［MHz］と CPU アイドル率［%］を取り出す．（grid-info-search に検索のキーワードを指定することにより，特定のリソース情報のみを取り出すことができる．）

③ CPU 速度［MHz］×CPU アイドル率［%］の値をもとにリソースの余裕度の高い上位 3 ホストを取り出す．

リソースの余裕度＝CPU 速度［MHz］×CPU アイドル率［%］

④ 上位 3 ホストから乱数関数により最適ホストを決定し，ホスト名を返す．

4） データ処理［各実行ノード上で起動］

共用ファイルサーバに保存されたイメージ・データを読み込み，OCR ソフトと翻訳ソフトを用いて翻訳した結果を，テキスト・ファイルとして共用ファイルサーバに格納するコマンドライン・プログラムである．共用ファイルサーバと実行ノード間のデータ転送には，Globus Toolkit のファイル転送機能（Grid FTP）を使用する．

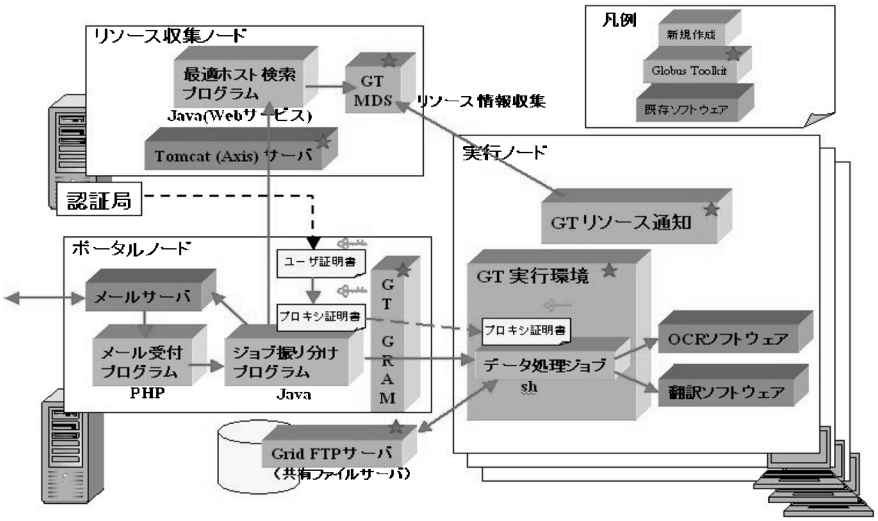

図 7　Globus Toolkit による携帯メール翻訳システム構成図

表 1 は，同システムの構築に使用したソフトウェアおよび主なハードウェア構成の一覧である．一部の有償ソフトウェア（*,**参照）を除き，殆ど無償のソフトウェアで構築することができた．OS も，全てオープンソースの無償ソフトウェアである Linux（Red Hat 9）とした．

表 1　携帯メール翻訳システムの構築に使用したソフトウェア/ハードウェア構成一覧

ホスト	ソフトウェア	CPU	メモリ
ポータルノード	GT3，メールサーバ（postfix）	Xeon 3.0GHz×2	6GB
リソース収集ノード	GT3，Tomcat，AXIS	Xeon 2.4GHz	2GB
共有ファイルサーバ	GT3	Xeon 2.4GHz	2GB
実行ノード 1	GT3，OCR ソフト（GOCR），翻訳ソフト（インターネット 翻訳の王様）*	Xeon 2.4GHz	2GB
実行ノード 2	GT3，OCR ソフト（GOCR），翻訳ソフト（インターネット 翻訳の王様），VMWare**	Pentium4 3.0GHz	512MB
実行ノード 3	実行ノード 2 と同じ	Pentium4 3.0GHz	512MB
実行ノード 4	実行ノード 2 と同じ	Pentium4 3.0GHz	512MB
実行ノード 5	実行ノード 2 と同じ	Pentium4 3.0GHz	512MB

＊　英文翻訳ソフト（Linux 対応版）

＊＊　Windows 上で仮想的に UNIX 互換 OS を稼動させるためのソフト．実行ノード 2〜5 については，既存の Windows 環境を保持する必要があったため，VMWare 上で Linux を稼動させた．

4.1.2　商用グリッドミドルウェア LSF によるシステム構成

CPU 負荷分散向きの代表的かつ実績ある商用グリッドミドルウェアの一つとして，Platform Computing 社[*10] の LSF（Load Sharing Facility）があげられる．図 8 は，Globus Toolkit の代わりに LSF を使用して携帯メール翻訳システムを構築したときの構成図である．リソース情報の収集，最適実行ホストの選択，およびリモート・ジョブ投入などの機能は，全て LSF が提供するため，先に作成したメール受け付けプログラムのジョブ投入先を LSF に変えるだけで，グリッド環境を構築することができた．

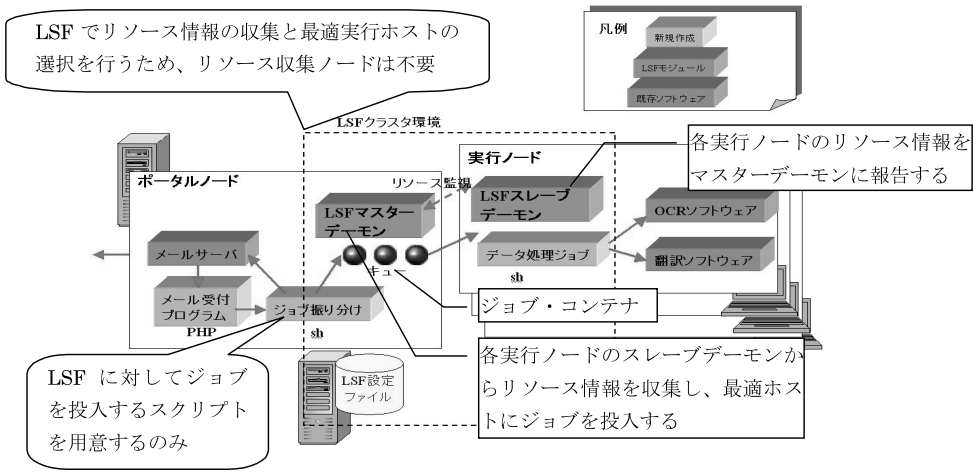

図 8　LSF による携帯メール翻訳システム構成図

4.1.3　システムの評価

携帯メール翻訳システムの評価に際しては，実験的モデルであることを踏まえ，主要コンポーネントの基本動作とその連携の正当性，および最低限の効率確保にチェック・ポイントを絞り，簡単な検証テストを実施した．表 2 に検証テストの内容とその結果を示す．

表 2　携帯メール翻訳システム　検証テスト項目および結果

No.	テスト項目	期待される結果	実行結果	
			GlobusToolkit 版	LSF 版
1	システムに負荷をかけない状況でメールを 1 件処理する。 （注)添付イメージのサイズは平均 1.0MB 程度とする。	メール受付から翻訳結果の返信まで全ての処理が正常に行われる。また、実行ノードとしてどのホストが選択されても同じ結果が得られる。	OK	OK
		平均ターンアラウンドタイム（メール送信から結果受信まで）30 秒以内。	NG 平均 40 秒以上	OK 平均約 20 秒
2	実行ノードの CPU 負荷を変化させながらメールを 1 件処理する。	ジョブの実行ホストとして CPU 負荷が少なく、CPU 速度の高いホストから優先的に選択される。	OK	OK
3	大量のメール要求（1 件／秒）を想定した擬似的なラッシュ環境のもとで、メールを 1 件処理する。	全実行ノードの平均 CPU 使用率が 80%以上に達した状況で、ほぼ1と同等のターンアランドタイムが確保される。	NG (注)1の NG を受け、実行不可と判断。	OK

上記のテストの結果，Globus Toolkit，および LSF を用いて，携帯メール翻訳という目的に特化したグリッド環境を構築し，基本的な負荷分散のしくみを実装できることが分かった．しかしながら，前者においては十分な効率を確保できないなど，実用化に向けての問題点があることも判明した．次節では，検証テストの結果を踏まえながら，これらのグリッドミドルウェアを用いてグリッド環境を構築した際の利点や問題点について考察する．

4.2　グリッドミドルウェアによるグリッド環境構築の利点と問題点

4.2.1　グリッド環境の利点

グリッドの仕組みを使わずに携帯メール翻訳システム相当の負荷分散環境を構築することを

想定した場合，例として以下のような処理を行うことが考えられる．

① クライアントから，rsh や ssh などのリモートシェル[*11] を使って，全ての実行ノードのリソース状況（CPU 負荷など）を調べる．
② 処理するノードが決まったら，telnet などで対象ノードにリモートログインする．
③ FTP を使用して，処理に必要なデータをファイルサーバからジョブ実行ノードに格納し，処理を実行する．
④ 処理が終了したら，処理後のデータをもう一度 FTP でファイルサーバに返す．
⑤ リモートログインしているジョブ実行ノードからログアウトする．

この場合，①のリモートシェルや②のリモートログイン，および③，④の FTP などの使用において，リモートホストに対する複数の認証を行う必要が発生する．また，実行ノード数が多いときは，全てのノードのリソース状況を調べるのに時間がかかり，始めに調べたノードの状況が変わってしまう結果，適切な実行ノードを選択できない恐れもある．

これに対し，グリッドミドルウェアを用いてグリッド環境を構築した場合は，シングルサインオンをサポートする認証機能，および実行ノードのリソース情報収集機能により，いずれの問題も解決できる．また，HTTP/SOAP ベースのグリッドサービスを使用することにより，ファイアウォール越しのジョブ投入を行うこともできる（Globus Toolkit 3 を使用した場合のみ．LSF の場合，上位の姉妹製品を使用する必要がある）．

従来の技術を用いても，手間をかけてアプリケーションを作りこむことによって，リモートホストにジョブを実行させるための負荷分散の仕組みを作ることは可能かもしれないが，それら手間のかかる開発工程を減らすとともに，より強力な仕組みの構築を可能とするのが，グリッドミドルウェアを使用する最大の利点と言うことができる．

4.2.2 グリッド環境の問題点

前述の検証テストの結果，Globus Toolkit 3 を使用した携帯メール翻訳システムにおいては，どんなに軽いジョブを実行しても 1 件あたりの処理に 40 秒以上を要した．内部処理をトレースするため，各処理ステップでタイムスタンプを採取したところ，図 9 に示す通り，以下のポイントで大きく時間がかかっていることが判明した．

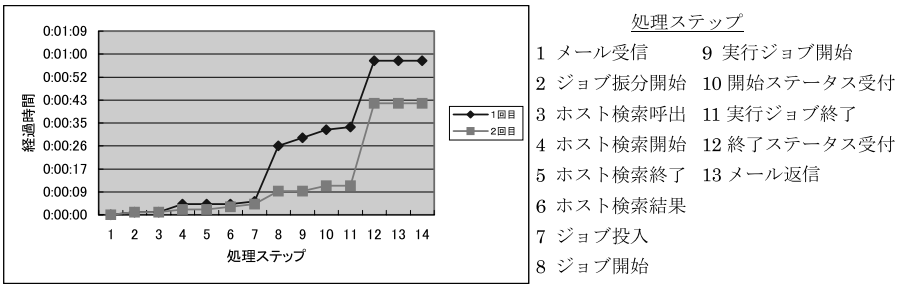

図 9　Globus Toolkit による携帯メール翻訳の処理ステップと実行時間

① リモートホストへのジョブ投入（ステップ 7）からジョブ開始（ステップ 8）まで
1 回目：21 秒/2 回目：5 秒
② ジョブ実行終了（ステップ 11）から終了ステータスの受付（ステップ 12）まで

1回目：24秒/2回目：30秒

いずれのポイントも，Globus Toolkit内部の制御処理であり，ここでリモートホスト上での実質的なジョブ実行時間（2秒〜4秒）を上回る時間を要していることから，効率に関するGlobus Toolkitの本質的な問題と言える．この点が改善されない限り，実用化に際しての大きな障害となり得る．

なお，上記①で1回目の実行(グリッド・システムが初期化されてから初めてのジョブ実行)の方が，2回目以降の実行よりも時間がかかっているのは，1回目の実行時にリモートジョブ処理のグリッドサービスや実行環境の生成が行われるため,その負荷によるものと考えられる．

一方，商用のLSFを使用したシステムでは，今回設定した全てのチェック・ポイントをクリアし，秒当たり1件ほどのペースではあるが，ラッシュ・テストの実行を可能とした．それでも，LSFへジョブを投入してからリモートホストで実行が開始されるまで，最低でも4〜5秒を要することが確認されており，即応性の追及には限界があると考えられる．このようなグリッドミドルウェアによって構築されたグリッド環境は，実行時間が比較的長いジョブのバッチ的処理には向いているが,ジョブのスケジューリングに際してタイムラグの発生を伴う限り，秒当たり100件ないし1000件のオーダーの処理効率が要求されるようなトランザクション的処理には，あまり向いていないと言える．

さらに，勘定系の業務など，一般的なビジネス分野におけるトランザクション処理を考えた場合，共有データへの排他的なアクセス（参照，更新など）の仕組みが必須であり，通常はデータベース・システムにより実装されている．しかしながら，今回取り上げたグリッドミドルウェアによるグリッド環境では，ファイルレベルのデータ共有が前提であり，データベースとの連携の便が，インフラのレベルで特に考慮されているわけではない．よって，データアクセスの面から見ても，トランザクション・システムとの親和性は高くない．

以上より，今回検証したようなグリッド環境を，トランザクション・システムへの適用にも耐えられるよう改善する為には，タイムラグのない迅速なジョブ（トランザクション）のスケジューリングと,データベース・レベルの共有データアクセスのサポートが必要と考えられる．

5. おわりに

現段階で主流と言われるグリッドミドルウェアは，ビジネス領域の中でも製造業に代表される技術計算分野への適用が主体であり，その分野においてはTCO削減等の良好な導入効果が報告されている．代表的グリッドミドルウェアとして本稿で紹介したGlobus Toolkitや Platform LSFも，その例に漏れない代わりに，トランザクション処理などが主体のその他一般的なビジネス分野には適用しにくいことが明らかとなった．Globus Toolkitの場合は，新しい標準アーキテクチャOGSAを採用することにより，サイエンスとビジネスの両者に適する標準的グリッド基盤の提供を指向しているが，少なくとも現状のGT 3レベルの実装では，効率面の問題が大きいなど，まだ実用に耐え得ない状況にあるのは前述の通りである．

このように，真に適切なグリッド製品の登場が見られない現状を反映してか，一般的なビジネス分野に対するグリッドの浸透には，しばらく時間が掛ると言われている．しかしながら，経済産業省が主催する「ビジネスグリッドコンピューティングプロジェクト」[*12]において，新しいタイプのグリッドミドルウェア(データセンタの運用を対象に，多数のサーバを束ねて障害や災害に高い耐性を持つシステムを構築するなど)の開発と製品化が進められるなど，活性

化の兆しは増しつつある．今後，GGF が推進する標準化仕様が進化・浸透し，ビジネスの分野においても次世代の Globus Toolkit がスタンダードの地位を獲得するのか，あるいは新しい別のアプローチがこれに取って変わるのか，慎重に見極めるべき時期が近づいていると考える．

＊1　処理能力や記憶容量などの計算機資源を，電気・ガスなどの公共サービスのように必要なときに必要なだけ購入して利用する方式．
＊2　日本国内におけるグリッド技術の普及と国際的標準化への貢献を目的とした産官学連携の団体（http：//www.jpgrid.org/）．
＊3　人手を介さず，コンピュータ自身に自律的にシステムの運用管理を行わせるための技術．
＊4　グリッドコンピューティングに関する調査・研究やグリッドミドルウェア Globus Toolkit の開発で世界的に有名な学術団体．
＊5　グリッド技術の標準化団体．研究者のコミュニティより形成され，年 3 回の国際会議を開催中．
＊6　デジタル証明書を安全に発行，管理，運用するためのシステム基盤．証明書，認証局（証明書の発行機関），リポジトリ（証明書の収容サーバ）から構成され，通信相手が間違いなく本人であること（真正性），さらに通信内容が改ざんされていないこと（安全性）を証明する仕組みを提供する．
＊7　文書やデータの意味や構造を記述するためのメタ言語で，インターネット標準として勧告された．
＊8　オープンソースの汎用スクリプト言語．特に Web 開発に適しており，HTML に埋め込み可能．
＊9　Java 言語で書かれた SOAP 実装の一つ．オープンソースの Web サービス・エンジンとして有名．
＊10　1992 年設立以来 DRM（分散コンピューティング）で 10 年以上の実績を持ち，業界最大手．
＊11　ネットワーク上のあるマシンから，別のマシンのシェル（コマンドインタープリタ）を使用するためのプログラム．
＊12　ビジネス分野を対象とした高信頼・高安全なプラットフォームであるグリッドミドルウェアの開発と提供を目的とした経済産業省主催の国家プロジェクト．民間より NEC，日立，富士通の三社が参加．

参考文献
［1］ 日本アイ・ビー・エム　システムズ・エンジニアリング株式会社，「グリッド・コンピューティングとは何か　Globus Toolkit ではじめるグリッドの基礎」，ソフトバンクパブリッシング，2004 年 4 月
［2］独立行政法人　産業技術総合研究所グリッド研究センター編，「グリッド　−情報社会の未来を紡ぐ−」，丸善，2004 年 4 月
［3］ 三井情報システム協議会（MISCO），グリッドコンピューティング化の方向性と課題，三井業際研究所，2004 年 4 月
［4］ 日本アイ・ビー・エム　グリッド・ビジネス事業部　関　孝則，JAVA PRESS Vol.30 特別企画　第 1 章「グリッドコンピューティング最新事情」，技術評論者，2004 年 6 月 15 日
［5］ 産業総合研究所　グリッド研究センター　中田秀基，JAVA PRESS Vol.30 特別企画　第 2 章「Globus と OGSA の基礎知識」，技術評論者，2004 年 6 月 15 日
［6］ グリッド協議会，「グリッドについて」，http：//www.jpgrid.org/
［7］ The Globus Alliance, http：//www.globus.org/about/
［8］ The Globus Toolkit, http：//www−unix.globus.org/toolkit

執筆者紹介　内　山　忠　夫（Tadao Uchiyama）
1900 年日本ユニシス(株)入社．IX/CS シリーズ用 OS およびセキュリティ管理プロセッサの受入・保守に従事．2004 年ユニアデックス(株)に転籍．プロダクト事業部内のグリッドコンピューティング研究プロジェクトに参画．現在，同事業グループソフトウェアプロダクト統括部サーバソフトウェア二部所属．