■6 群（コンピュータ －基礎理論とハードウェア）－ 5 編（コンピュータアーキテクチャ（II）先進的）

8 章 グリッドコンピューティング

（執筆者：合田憲人）［2010 年 5 月 受領］

■概要■

グリッドは，ネットワークに接続された計算機やストレージ，データベース，実験装置等の様々な資源を連携して利用するためのインフラ技術である．このうち，グリッド技術を用いてネットワーク上の資源を連携させ，これを計算に利用する方法はグリッドコンピューティング（やグリッド計算）と呼ばれる．本章では，グリッドコンピューティングの概要とそのプログラミング手法やミドルウェア技術に関して説明する．

【本章の構成】

8-1 節では，グリッドを構成するための仕組みである仮想組織やグリッドの利用方法について，グリッドコンピューティング以外の技術も含めて説明する．グリッドの資源を活用した計算を行うためには，他の分散コンピューティングシステムと同様にプログラミングが必要となる．8-2 節では，グリッド上の資源を利用して分散計算を実現するためのプログラミング技術について説明する．ネットワーク上の複数の資源からなるグリッドを実現するためには，これらの複数の資源を連携させるためのソフトウェアであるグリッドミドルウェアが必要となる．8-3 節では，グリッドミドルウェアの概要とその要素技術について説明する．

■6 群 － 5 編 － 8 章

8-1 グリッドの構成方法

（執筆者：田中良夫）［2009 年 4 月 受領］

8-1-1 仮想組織

グリッド[1)-3)]は，地理的，組織的に分散された計算機，データベース，実験装置など様々な資源を統合して利用することを可能とする．そしてそれは，「それらの資源が分散されていることを意識せずに簡単に使いたい」というユーザの要求と，「自組織のセキュリティポリシーに基づき，特定のユーザにのみアクセスを許可したい」という資源所有者の要求を満たしつつ実現される必要がある．このように，それぞれが独自のセキュリティポリシーに基づいて提供される複数の資源を束ねて簡単に利用する環境をユーザに提供するために，グリッドは「仮想組織（Virtual Organization：VO）」[4), 5)]と呼ばれる概念を基本に構成される．

グリッドにおいては，計算処理，データベース検索や実験装置の操作など，提供される資源を利用するためのサービスが標準的なプロトコルやアプリケーションインタフェースを通して提供される．仮想組織はポリシーを共有できるプロジェクトやコミュニティに対応するものと考えればよく，物理的に分散する複数のサービス群を目に見えない仕切りで囲み，仮想的な一つの組織に存在するサービス群としてユーザに提供する．資源を所有し，サービスを提供する大学や研究所などの現実社会の組織と区別するために，仮想組織という表現が利用されている．

図 8･1 の仮想組織の例を示す．A 大学と C 大学が所有するデータベースを検索して得られた結果に基づいて，B 大学が提供するシミュレーションを実行する場合，A 大学，B 大学，C 大学により構成される仮想組織 X が動的に形成され，ユーザは，それらのデータベースやシミュレーションプログラムを，それらがあたかも単一の組織にあるかのように利用することができる．また，C 大学の所有するデータベースを検索して得られた結果を，D 研究所が提供するプログラムを用いて処理し，その結果を入力として同じく D 研究所が提供するシミュレーションを実行する場合にも，同様に仮想組織 Y が形成され，ユーザに利用環境を提供する．

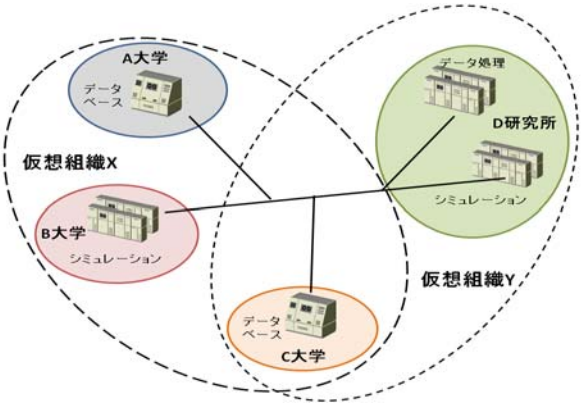

図 8・1　仮想組織の概要

仮想組織の形成はセキュリティにより実現される．具体的には，提供されるサービスの利用をユーザに許可するかしないかを設定することにより，先に述べた「目に見えない仕切り」を作る．サービスの利用を許可されるユーザは仕切りの中に存在してサービスを利用することができ，サービスの利用を許可されないユーザは仕切りの外に置かれてサービスを利用することができない．グリッドのセキュリティは，認証・認可に基づくアクセス制御を実現するほかに，ユーザが複数のサービスにアクセスする場合でも，仮想的に一度ログインすれば（パスワードを入力すれば），サービスにアクセスするたびにパスワードを入力する必要のないシングルサインオンの機能を実現している．これにより，複数の組織が特定のユーザに対して自組織の提供するサービスの利用を許可すれば，そのユーザは仮想的にログインすれば複数の組織により提供されるサービス群にアクセスすることができ，ユーザにとっては，それらのサービス群があたかも一つの組織（仮想組織）によって提供されているように見える．

各組織がユーザに対してサービスの利用を許可する方法はいくつかある．例えば，各組織が個別のユーザごとにセキュリティの設定を行う方法がある．この方法は実際に広く利用されている方法ではあるが，ユーザ数に比例した管理コストが発生するという問題点がある．そこで，仮想組織の形成とユーザの登録を別作業として行い，各組織は仮想組織への参加の可否のみを判断し，ユーザの登録は仮想組織の管理者に任せるというモデルが提案・実現されている [6]．このモデルには，サービスを提供するサービス提供者とユーザに加え，VO 管理者の役割が存在する．ユーザは基本的には一つ以上の VO に所属し，実際にサービスを利用する．VO 管理者は VO の構築，VO に所属するユーザの管理，ユーザ向けポータルの構築などを行う．そして，VO 管理者は VO に所属するユーザの登録や削除，グループの作成やユーザのグループへの登録や特定の権限や役割の付与など，VO におけるユーザ管理に責任をもつ．また，VO 管理者は，利用を希望するサービスがあればサービス提供者と交渉する．交渉の結果利用が許可されれば，その VO に所属するユーザがサービスを利用できるようになる．利用可能なサービス群のみを集めてポータルなどを通じてユーザに提供することにより，ユーザは必要なサービスのみを，それらがあたかも一つの組織の中に存在するように利用することができる．

サービス提供者は VO へのサービスの提供を許可する場合はその VO に対するアクセスを許可するべくシステムの設定を変更する．サービス提供者のポリシーに応じて VO 単位，VO の中のグループ単位，あるいはユーザ単位など様々なレベルでのアクセス制御が設定可能となっている．VO やその中のグループの情報を用いれば，個別のユーザに対するアクセス制御の設定は必要ない．このように，このモデルでは個別のユーザの管理は仮想組織の管理者（プロジェクトのリーダなど）が行い，各組織は「ある仮想組織に所属するすべてのユーザにアクセスを許可する」あるいは「ある仮想組織に所属し，特定の権限をもつユーザにアクセスを許可する」といった設定が可能であり，ユーザ数に対してスケーラブルかつ柔軟なセキュリティを実現しつつ仮想組織を形成することができる．

8-1-2 グリッドの分類

グリッドは主たる用途や実現方法によっていくつかに分類される．しかし，その定義は曖昧かつ流動的であり，明確に分類することは難しい．また，例えば後で述べるように大規模計算を主目的とする計算グリッドと，大規模データ検索・処理を主目的とするデータグリッ

ドなどがあるが，グリッドの本質は計算もデータ検索・処理もサービスとして抽象化し，それらを統合して利用することにあり，計算と大規模データ検索・処理を区別（分類）することに矛盾がある．しかし，グリッドの用途や実現方法としてどんなものがあるのかを概観することはグリッドの理解への一助でもあり，ここでは歴史的に確立してきたグリッドの分類について述べる．

(1) 計算グリッド

計算グリッドは，グリッド上に分散配置された計算機群を用いて高性能な計算サービスを提供する．例えば，複数のスーパーコンピュータを束ねて仮想的な 1 台の大規模スーパーコンピュータとして利用することにより，単体のスーパーコンピュータの性能では困難な大規模計算を実現するメタコンピューティングや，グリッド上の複数のクラスタ計算機により提供される大量のプロセッサ資源を利用して大量のタスク処理（小規模計算）を実行するハイスループットコンピューティングなどは計算グリッドに分類される．応用例としては，大規模シミュレーションや大規模最適化問題などがあり，量子・分子シミュレーションによるナノテク [7] や創薬，気象シミュレーション [8]，半正定計画問題の解法 [9] やサプライチェーンマネジメントなど多岐にわたる．

(2) データグリッド

データグリッドは，グリッド上に分散配置されたストレージ群を用いて大規模分散データベースやグリッドファイルシステムを構築・提供する．例えば，複数の組織が所有するデータベースを仮想的な一つのデータベースとして統合・提供するデータベース連携技術 [10), 11)] により，ユーザは複数のデータベースに対して適切な認証を受けたうえで一括した検索をかけることができる．また，グリッドファイルシステム [12), 13)] は実際には複数の組織に分散配置されるファイル群を単一のファイルシステムとしてユーザに提供する技術であり，ユーザはファイルの物理的配置を意識せずにそれらのファイルにアクセスすることができる．応用例としては，複数のデータベースを参照する地球環境科学分野，生命情報科学分野，天文分野や，実験装置が生成する大量のデータを解析・処理する高エネルギー分野などがあげられる．

(3) センサグリッド

センサグリッドは，雨量・湿度・気温など様々な情報を刻々と計測するセンサ群により提供されるデータの提供・処理を実現する．ネットワークを介して収集されるセンサデータを統合し，高度な観測サービスや観測されたデータの解析サービスを提供する．応用例としては，自然災害対策，環境モニタリングや農業分野などがあげられる．

(4) アクセスグリッド

アクセスグリッド [14)] は，人と人とがコミュニケーションを円滑に行うために映像，音声，会議資料などの電子データを複数拠点間で共有するための技術である．現在は H.323 を用いたテレビ会議システムや P2P 技術を用いた簡易会議システムなど，複数拠点を接続する会議システムが広く利用されるようになってきているが，アクセスグリッドはマルチキャストを

用いて大量の拠点を接続することを可能とするなど，高性能あるいはスケーラブルといった特徴をもつ．

(5) ビジネスグリッド

ビジネスグリッドは，企業内あるいは企業間で所有する計算機やストレージを共有することにより，信頼性の高いトランザクション処理やバッチ処理を低コストで実現する．先に述べた計算グリッドやデータグリッドなどが，主として科学技術分野を対象に，高性能あるいは大規模化といった観点から研究開発が進められてきたのに対し，ビジネスグリッドは主として企業業務での利用を想定し，高信頼やコスト削減といった観点から開発が進められている．

(6) PC グリッド

PC グリッドは，ネットワーク上の遊休 PC を集めて計算やその他の処理を実現する環境を提供する．今まで述べたグリッドは，その用途に応じて分類されたものであるが，PC グリッドは用途による分類ではなく，グリッドの実現方法の一種である．SETI@HOME [15] や Folding@home [16] などのボランティアコンピューティングや，企業内の遊休 PC の業務への活用や外部企業への貸し出しといったビジネス分野での利用など，その応用分野は多々ある．

■参考文献

1) I. Foster and C. Kesselman(Ed.), “The Grid: Blueprint for a New Computing infrastructure, 2nd edition,” Morgan Kaufmann Publishers, 2004.
2) 合田憲人, 関口智嗣(編著), “グリッド技術入門,” コロナ社, 2008.
3) 産業技術総合研究所グリッド研究センター編, “グリッド ー情報社会の未来を紡ぐー,” 丸善, 2004.
4) I. Foster, C. Kesselman, J. Nick, and S. Tuecke, “The Physiology of the Grid,” Grid Computing: Making the Global Infrastructure a Reality, Wiley, pp.217-249, 2003.
5) 田中良夫, 山本直孝, 関口智嗣, “地球観測グリッドにおけるセキュリティ基盤の設計と実装,” 情報処理学会論文誌コンピューティングシステム, vol.1, no.2, pp.169-179, 2008.
6) R. Alfieri, R. Cecchini, V. Ciaschini, L. dell’Agnello, Á. Frohner, K. Lőrentey, and F. Spataro, “From gridmap-file to VOMS: managing authorization in a Grid environment,” Future Generation Computer Systems, vol.21, no.4, pp.549-558, 2005.
7) H. Takemiya, Y. Tanaka, S. Sekiguchi, S. Ogata, R. K. Kalia, A. Nakano, and P. Vashishta, “Grid applications--Sustainable adaptive grid supercomputing: multiscale simulation of semiconductor processing across the pacific,” In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, ACM Press, pp.1-11, 2006.
8) H. Takemiya, Y. Tanaka, K. Shudo, and S. Sekiguchi, “Constructing Grid Applications using Standard Grid Middleware,” Journal of Grid Computing, vol.1, no.2, pp.117-131, Kluwer Academic Publishers, 2003.
9) A. Takeda, K. Fujisawa, Y. Fukaya, and M. Kojima, “Parallel Implementation of Successive Convex Relaxation Methods for Quadratic Optimization Problems,” Journal of Global Optimization, vol.24, no.2, pp.237-260, Springer Netherlands, 2002.
10) M. D. Stefano, “Distributed Data Management for Grid Computing,” Willey, 2005.
11) S. Lynden, S. Mirza, and I. Kojima, “Service-based Data Integration using OGSA-DQP and OGSA-WebDB,” In Proceedings of the Ninth IEEE International Conference on Grid Computing, 2008.
12) S. Maad, B. Coghlana, G. Quigleya, J. Ryana, E. Kennya, and D. O’Callaghan, “Towards a complete grid filesystem functionality,” Future Generation Computer Systems, vol.23, Issue 1., pp.123-131, 2007.
13) 建部修見, 森田洋平, 松岡 聡, 関 智嗣, 曽田哲之, “ペタバイトスケールデータインテンシブコン

ピューティングのための Grid Datafarm アーキテクチャ," 情報処理学会論文誌ハイパフォーマンスコンピューティングシステム, 情報処理学会, Vol.43, No.SIG 6 (HPS 5), pp.184-195, 2002.
14) Access Grid. http://www.accessgrid.org/.
15) SETI@HOME, http://setiathome.berkeley.edu/.
16) Folding@home, http://folding.stanford.edu/.

■6 群 － 5 編 － 8 章

8-2 グリッドでのプログラミング

（執筆者：中田秀基）［2009 年 3 月 受領］

8-2-1 グリッドの特殊性

グリッドは一般に複数のクラスタから構成される，本質的に不均質な分散・並列環境である．グリッド環境の特殊性を以下に列挙する．

- **計算ノードの不均質性**：グリッドを構成する各計算ノードは，一般に複数の管理組織に属し，運用開始時期も異なるため，均質性が保証されない．演算速度だけでなく，アーキテクチャそのものが異なる可能性もある．
- **システムの不安定性**：グリッドは，複数の管理組織に属する．膨大な数のノードで構成される．このため，すべてのノードが完全に動作することを保証することは非常に難しい．
- **ネットワークの不均質性**：グリッドは一般に複数のクラスタから構成される．この場合，同一クラスタ内のネットワーク速度とクラスタ間のネットワーク速度は大きく異なる．クラスタ間ネットワークは一般に低速である．スループットは十分大きくすることができるが，レイテンシを小さくすることは原理的に不可能である．
- **ネットワーク接続の非対称性**：一般にグリッド上のノードは，外部から接続可能であるとは限らない．そもそもサイト外との通信が不可能な場合もあるし，内部から外部への接続のみが可能な場合もある．

このような環境でプログラミングするためには，通常のクラスタ環境とは異なるプログラミングスタイル，もしくは特殊な実行系が必要となる．

8-2-2 マスタワーカ

グリッドのような特殊な環境でもうまく機能するプログラミング手法として古くから知られるマスタワーカ法がある．この手法では，ジョブを互いに独立した多数のサブジョブに分割してマスタが管理する．ワーカはマスタからサブジョブを受け取って処理し，結果をマスタに返却する．結果は新たなサブジョブである場合も多い．マスタが管理するサブジョブがすべてなくなるまでこの動作を繰り返す．

すぐにわかるように，この手法が利用可能であるためには，アプリケーションの対象となるジョブが，独立したサブジョブに分割できなければならない．これは大きな制約ではあるが，様々な大規模な実問題がこの手法の枠組みで処理できることが知られている．

マスタワーカ法の利点の一つは動的負荷分散がほぼ自動的に行われるため，ノードやネットワークの不均質性に対して寛容であることである．低性能のワーカは，一つのサブジョブの処理に時間がかかるため，処理するサブジョブの数は少なくなり，高性能のワーカはその逆となる．また，サブジョブの計算コストが不均質でも，サブジョブの数がノードの数に対して十分でありさえすれば，特に問題になることはない．更に，サブジョブの取得を先行させることによって，ネットワークレイテンシの増大に寛容な実装を行うことができる．

また，障害にも寛容である．マスタワーカ法ではワーカは状態をもたないため，アプリケーションの状態は本質的にマスタのみに存在するといえる．このため，マスタのみを対象に注意深くチェックポイントをとることによって，アプリケーション全体に耐故障性を与え

ることができる．

マスタワーカ法は問題解決の手法あり，特定の計算機構を前提とするものではない．例えば，MPI でマスタワーカ法を実装することも可能である．しかし，よりマスタワーカ法を実装しやすい計算機構も存在する．以下でいくつか紹介する．

(1) GridRPC

GridRPC はグリッド上で RPC（Remote Procedure Call）を提供する API 規格で，現在 GGF の GridRPC WG で規格化が行われている．GridRPC の計算機構は，一つのクライアントとサーバ（群）から構成される．GridRPC は，サーバ上に存在する計算関数を，クライアント上のプログラムから容易に呼び出す機能を提供する．この際，クライアントは複数のサーバの関数を同時に並行して呼び出すことができる．これによって非常に簡単にある種の並列プログラムを実現することができる．また，クライアントにマスタプログラムを，サーバ群にワーカプログラムを実装することによって，マスタワーカ法のプログラムを容易に実現することができる．GridRPC の実装としては産総研で開発された Ninf-G [1]，テネシー大学の NetSolve [2]，フランス ENS-Lyon の DIET [3]，筑波大学の OmniRPC [4] がある．

(2) マスタワーカ専用システム

マスタワーカに特化したシステムもいくつか存在する．このようなシステムは汎用性を失う代償として，マスタワーカシステムを容易に構築できるよう設計されている．このようなシステムの例としては Condor MW や Platform Computing Inc.の Symphony があげられる．

Condor MW（Master Worker）はウィスコンシン大学で開発されたスケジューリングシステムである Condor の追加コンポーネントの一つで，Condor 上でマスタワーカ型の計算を行うシステムである．

MW は C++で記述されており，ワーカ及び転送されるデータのテンプレートとなるクラスをユーザに提供する．ユーザはこのテンプレートクラスを拡張し，アプリケーションに即したクラスを実装するだけでよい．マスタワーカ間の通信路としては一般的なソケットによる通信を含めていくつかの方法が提供されている．

8-2-3 グリッド向け MPI 実装

MPI はメッセージパッシングに基づく，並列分散環境での標準的なプログラミングミドルウェアである．MPI は基本的にすべてのノードが落ちないことを前提としており，必ずしもグリッドに適しているわけではない．しかし，多くの既存アプリケーションが MPI で記述されているため，現実的には，グリッド環境に適した MPI 実装は重要である．

(1) ネットワーク不均質性への対応

MAGPIE [3] は，ネットワークが不均質な環境で効率的な MPI 実装を行おうという試みである．MAGPIE ではブロードキャストやバリアなどの集団通信に焦点をあて，低速な広域通信路上を伝搬するメッセージの量を削減することで，グリッド上での高速な MPI プログラムの実行を可能にしている．

MAGPIE のアルゴリズムは以下の 2 点を念頭に設計されている．

・1 回の集団通信アルゴリズムで使用される送信者・受信者間のパスに低速な広域通信路が最大で一度だけ含まれるようにする．
・一つのデータが広域通信路を 2 度通ることがないようにする．

これらによって，広域通信路の長大なレイテンシによって受ける影響を最小化する．典型的な集団通信の一つであるブロードキャストを例として詳しく見てみよう．

通常の MPICH ではブロードキャストは 2 進木で行われる．8 ノードの場合を考える．ノード 0 がブロードキャストを行う場合，まずデータをノード 4 に転送する．次に，ノード 0 はノード 2 に，ノード 4 は 6 に，というように転送を行う．すべてのノード間での通信速度が均一であり，ノード間でのデータ通信時間を t_1 とすると，ノード数 N に対して t_1*Log(2)N の時間でブロードキャストが終了する．

この方法は，ノード間の通信速度が均一である場合には最速であるが，ノード間で通信速度が異なる場合には最速ではない. 最悪のケースとして, ノード 0 のみが他のサイトにあり，ノード 0 から他のノードへの通信が，他のノード間の通信よりも低速である場合を考えてみよう．ノード 0 からのサイトをまたがる通信が 3 回（Log(2)8）も行われるため，そのたびに低速な通信路の影響を受ける．

MAGPIE では, このような場合, ノード 0 からの通信を一度だけにするように最適化する．ノード 0 からはノード 1 にだけ通信するようにし，残りのノードに対しては，ノード 1 から 2 進木でブロードキャストを行うようにすれば，より高速にブロードキャストを完了することができる．

(2) ネットワーク接続の非対称性への対応

ネットワーク接続の非対称性に対応した MPI 実装の研究も広く行われている．最も容易な方法としては，サイト内外の双方からアクセスできるノードを用意し，そのノードを中継点として用いることで接続性を確保する方法がある．この方法は GridMPI [4] で用いられている．

また，QCG-OMPI [5] や MC-MPI [6] では，ネットワークの特性に応じて，様々な手法を使い分けることで接続性を確保している．

■参考文献

1) Yoshio Tanaka, Hidemoto Nakada, Satoshi Sekiguchi, Toyotaro Suzumura, and Satoshi Matsuoka. Ninfg, “A reference implementation of rpc-based programming middleware for grid computing,” Journal of Grid Computing, 1(1), pp.41-51, 2003.
2) Netsolve. http://www.cs.utk.edu/netsolve/.
3) Thilo Kielmann, Rutger F. H. Hofman, Henri E. Bal, Aske Plaat, and Raoul A. F. Bhoedjang. MagPIe, “MPI’s collective communication operations for clustered wide area systems,” ACM SIGPLAN Notices, 34(8), pp.131-140, 1999.
4) Motohiko Matsuda, Tomohiro Kudoh, Yuetsu Kodama, Ryousei Takano, and Yutaka Ishikawa, “Efficient mpi collective operations for clusters in long-and-fast networks,” In Proc. of Cluster 2006, 2006.
5) Cammile Coti, Thomas Herault, Sylvain Peyronnet, Ala Rezmerita, and Franck Cappello, “Grid services for mpi,” In Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid’08), May 2008.
6) Kenjiro Taura, Hideo Saito, and Takashi Chikayama, “Collective operations for wide-area message passing systems using adaptive spanning trees,” International Journal of High Performance Computing and Networking (IJHPCN), 5(3), pp.179-188, 2008.

■6 群 － 5 編 － 8 章

8-3 グリッドミドルウェア

（執筆者：合田憲人）［2009 年 3 月 受領］

グリッドミドルウェアは，ネットワーク上の計算機やストレージなどの資源を連携させてグリッドを構成するために必要となる基本ソフトウェアである．グリッド上で複数の資源を連携させて利用するためには，計算（ジョブ）を遠隔の計算機上で実行する技術だけでなく，セキュリティやジョブのスケジューリング（ブローカリング）など，ネットワーク上に分散した複数の資源を適切に利用するための技術が必要となる．また，ネットワーク上に分散したファイルやデータベースへのアクセスを実現することも重要である．更に，グリッド上で実行されるアプリケーションプログラムの開発者や利用者を支援するための技術も必要である．グリッドミドルウェアは，これらの要素技術を組み合わせて構成され，ユーザの目的に応じて必要なサービスを提供する．**図 8･2** は，グリッドミドルウェアを構成する要素技術を階層的に表現した例である．

本節では，グリッドミドルウェアを構成する要素技術について解説する．各要素技術の詳細については，文献 1) を参照されたい．

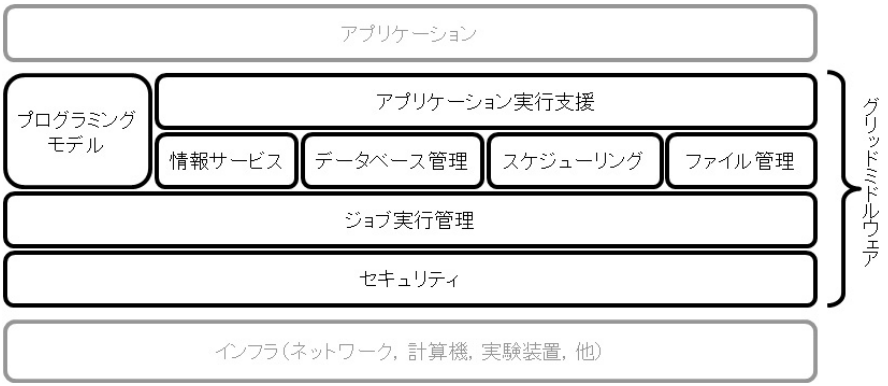

図 8･2　グリッドミドルウェアを構成する要素技術

8-3-1 セキュリティ

グリッド上のサービスをユーザが安全に利用するために重要となる技術がセキュリティである．グリッド上でのセキュリティ処理は，主としてユーザの認証や認可，ホストの認証，通信の暗号化から構成される．これらの処理については，インターネット上で様々な基礎技術が開発されており，グリッド上でもこれらの基礎技術が用いられている．しかし，グリッドのように複数の組織にまたがって処理を進める場合には，従来技術のみでは不十分であり，新たな技術も必要となる．

例えばユーザ認証の場合，グリッド上では資源が複数の異なる組織に分散しているため，従来のユーザ認証技術を用いて複数の資源を利用するためには，利用する資源ごとに認証手続きを行う必要があり，ユーザの負担が大きい．これに対して，シングルサインオンと呼ば

れる技術を用いることにより，1 回のユーザ認証で，ユーザの利用が許可されている範囲内（例えば VO）で，ユーザがすべての資源上のサービスを利用することができる．具体的にはユーザは，ポータルやコマンドライン上で一度だけユーザ認証の操作を行えばよく，その後，各資源上でのユーザ認証はユーザの介在なしにグリッドミドルウェアにより自動的に行われる．

8-3-2 情報サービス

グリッド上でユーザのジョブが安定して実行されるためには，グリッドミドルウェアがグリッド上の資源の状態を適切に把握し，かつこれらの情報がユーザに適切に提供される必要がある．情報サービスは，グリッド上の資源情報を管理する一種のデータベースであり，グリッド上の資源の情報の収集，収集した情報の集約，ユーザやソフトウェアからの問い合わせに対する情報提供などのサービスを行う．

情報サービスは，計算機やネットワークなどの情報をモニタリングするソフトウェアを介して情報を収集し，グリッド上のユーザやソフトウェアに提供する．ここで収集される情報は，計算機のハードウェア仕様やソフトウェア設定などの静的な情報だけでなく，CPU 負荷やハードディスク残量などの動的な情報も含まれている．

8-3-3 ジョブ実行管理

グリッド上のジョブは，グリッド上の資源を用いて安全にかつ安定して実行されなければならない．ユーザのジョブを安全に実行するためには，前述のセキュリティ技術を用いたユーザ認証のほかに，ジョブ（ユーザ）の資源に対するアクセス制御を行う必要がある．また，ユーザのジョブを安定して実行するためには，グリッド上の様々な資源に対してジョブの実行を依頼してかつ実行状態を監視するジョブ管理，ジョブの実行に必要な入力ファイルや出力ファイルの転送機能も必要である．このような処理をまとめてジョブ実行管理と呼ぶ．

ジョブ実行管理は，計算機などの資源上で窓口としての役割をもつソフトウェア（ここでは計算資源管理システムと呼ぶ）を介して実現される．計算資源管理システムは，ユーザからのジョブ実行依頼時にユーザのアクセス権限を調べることにより，ジョブの実行を受け付けるか否かを判断し，必要に応じてファイル転送を行うとととともに，受け付けたジョブを計算機のバッチスケジューラに投入するなど，各計算機の運用方針に従って実行する．

8-3-4 スケジューリング

上記であげたセキュリティ，情報サービス，ジョブ実行管理技術を用いることにより，グリッド上でジョブを実行することができるが，別の技術を組み合わせて利用することにより，より高性能なジョブの実行ができる．その一つがグリッド上のスケジューリング技術である．

グリッド上の資源は，ハードウェアの仕様やインストールされているソフトウェアの仕様が多岐にわたるだけでなく，複数組織に分散されているため，その運用方針も組織ごとに異なる．更にこれら静的な特徴に加えて，例えば計算機上の CPU やメモリ残量，ネットワークのスループットやレイテンシなど，動的に変動する特徴も多い．

グリッド上のスケジューリングの役割は，このような様々な資源の中から，ユーザのジョブを実行するための最適な資源を選択することにある．具体的には，スケジューリングのた

めのソフトウェアが，情報サービスから得られる資源情報とユーザから提供されるジョブの情報を比較し，ジョブを実行する資源を選択する．また，互いに通信する複数のジョブを同時に異なる資源に割り当てるコアロケーションなど，更に進んだサービスを提供することもグリッド上のスケジューリングの重要な役割である．

8-3-5 ファイル管理

グリッド上では，ジョブの利用するファイルが複数の資源に分散して保存されている場合も多い．ファイル管理は，グリッド上に分散したファイルに対して，ユーザがファイルの保存場所やアクセス方法を意識することなくアクセスするサービスを提供する．具体的には，分散して保存されるファイルを仮想的なディレクトリ構造やメタデータにより管理することにより，ユーザは仮想的なパス名を指定するだけでファイルにアクセスできるほか，ファイルの検索を行うことも可能になる．また，ファイルの高速転送やファイルの複製を行うことにより，ファイルへのアクセス性能を向上させることもファイル管理の提供する重要なサービスである．

8-3-6 データベース管理

グリッド上で実行されるアプリケーションには，ネットワークに接続されたデータベース上のデータを利用するものも多い．現在，様々なデータベースがネットワーク経由で利用可能であるが，実装方法によりアクセス方法が異なる．しかし，ユーザが異なるデータベースへアクセスする度に個々のデータベースの場所や利用方法の違いを意識することは難しい．

この問題を解決するため，グリッド上のデータベース管理は，異なる方式で実装された複数のデータベースに対して統一的なアクセス方法を提供するサービスや，複数のデータベースに対する効率のよい検索を行うサービスを提供する．具体的には，異種のデータベースを Web サービス技術により連携させるサービスや，グリッド上のジョブから依頼された検索結果をグリッド上の別の資源に転送して利用できるようにするサービスがあげられる．

8-3-7 プログラミングモデル

グリッド上に分散した資源を用いて実行される高性能なプログラムを開発するためには，従来技術では実現されていない複雑な処理が必要になる．しかし，アプリケーション開発者の視点では，開発者が従来から利用しているモデルを用いたプログラミングを行えることが望ましい．そのため，グリッド上でも，従来のモデルに基づくプログラミングモデル，具体的にはマスタワーカモデルやノード間直接通信モデルが主に用いられている．

マスタワーカモデルは，計算機を 1 台のマスタと複数のワーカに分け，マスタが計算を複数のワーカに割り当てることにより並列計算が進められるモデルである．ノード間直接通信モデルは，計算機間でメッセージを交換しながら処理を進めるモデルであり，並列プログラミングモデルとしてよく知られている MPI[2] がその代表例である．

8-3-8 アプリケーション実行支援

グリッド上でユーザが煩雑な作業を行うことなしに簡単にアプリケーションを実行するためには，アプリケーションの実行や監視，アプリケーションを構成する複数のジョブの

連携実行などの処理をユーザが容易に実行するためのアプリケーション実行支援技術が重要となる．

グリッドポータルは，グリッド上で提供されているアプリケーションの探索，実行，また実行中のアプリケーションの監視や制御といった作業をユーザが GUI を通して容易に実行するための技術である．ユーザは，Web ブラウザを用いることによりポータルサイトにアクセスし，これらの作業をブラウザ操作のみで行うことが可能である．

ワークフローは，複数のジョブを連携して実行する場合に，ソフトウェア間の実行順序や入出力の関係に従って適切にジョブを実行するための技術である．ユーザは，アプリケーション間の関係を示すワークフローを作成し，実行を依頼する．ユーザが記述したワークフローは，ワークフローエンジンと呼ばれるソフトウェアに解釈され，ジョブ間の依存関係に基づいてジョブが起動される．

■参考文献

1) 合田憲人, 関口智嗣(編著), “グリッド技術入門,” コロナ社, 2008.
2) Message Passing Interface Forum, http://www.mpi-forum.org/