4 大学間連携グリッド基盤の運用

小林 泰三　九州大学情報基盤研究開発センター
天野 浩文　九州大学情報基盤研究開発センター
青柳 睦　九州大学情報基盤研究開発センター
合田 憲人　国立情報学研究所

グリッド管理運用から見えてくるもの

本稿は，現在進行中である大学間連携グリッド基盤の構築と運用の準備作業を通して判明した問題とその対処方針を，主に管理者の視点で解説するものである．しかし，大部分の読者は大学の基盤センターにあるような大規模な計算機の管理者ではなく，ましてやさらに大規模な計算機グリッド（以下，グリッド）の管理や運用とは直接関係を持たない立場である場合がほとんどであろう．そのような読者にとって本稿はどのような意味を持つのであろうか．この問に一言で答えるとすると，それは「計算環境の大規模化と分散化に付随した諸問題」に共通するものがある，と言えそうである．グリッドを管理運用する視点から解説することにより，計算環境の大規模化と分散化に連なる問題の整理ができれば，本稿も解説記事として意義のあるものになるのではないかと思う．

計算環境の大規模化と分散化であらわれる問題の全体像やイメージを掴むために，これからの計算環境の見通しとグリッドの管理運用との関係を，大まかに把握するところからはじめよう．昨今のCPUのメニーコア化やマルチコア化の流れから分かるように1CPUコアの性能向上には限界が見え始めており，この状況で計算速度の向上を実現するには並列計算がそのほとんど唯一の解決手段になっている．つまり，より大きな計算パワーを望めば望むほど多くの計算ノードを利用せざるを得なくなり，場合によっては異なるCPUアーキテクチャやポリシーで運用されている並列環境を同時に使う必要も出てくるであろう．日々の研究をスパコンなど基盤センターの計算機を利用して行っている多くのユーザにとってはこれらの問題はすでに現実のものとして迫っており，このような研究者にとって今後ますます重要になってくるものの1つは，大規模な並列計算やGPGPUなどのアクセラレータの効率的な利用など，大規模かつヘテロな環境への対応だと言われている．このような状況で誰しもが思うのは「単純にMPIのランクを何十万何百万にスケールアップすれば済む問題ではない」ことであり，「アクセラレータ付き環境のようなヘテロな状況へ対応するためにはプログラミングをどうすべきか」といったことであろう．少なくとも確実に言えることは「現在の並列計算のスタイルを単純に拡張しただけでは問題は解決しそうにない」ことである．一方で，グリッドの管理と運用で求められるのは，なんといってもまずは研究者が必要とする大規模な並列計算環境を構築して提供することである．並列計算環境の構築と提供であればグリッド以前から各基盤センターでサービスが提供されているが，グリッドがそれらと大きく異なるのは，これまで各基盤センター（サイト）単位で並列計算環境が閉じていたのがグリッドに参加する基盤センターグループへと並列度の階層が1段上がるところである．この階層が1段増えた大規模な分散並列環境は複数のサイトが集まって提供されるのであるから，当然のことながら複数の管理者で管理されることになる．一方でA大学の基盤センター管理者がB大学の基盤センターの管理者を兼ねることはほとんどないので「グリッドを構成するすべてのノードにわたって管理者権限を持つ管理者が実質上いない」ことになる．これをグリッドに資源提供しているある基盤センターの管理者（グリッドの管理運用にかかわる管理者のほとんどが置かれる立場）の側から眺め直すと「グリッドとして提供されるサービスは，自分が直接管理できない他サイトの状況に左右される」ことになる．つまり，ここでも「現在のサイト内で実現している並列計算環境の管理運用方法を単純に拡張しただけでは問題は解決しそうにない」ことが言えるのである．

ここまでの議論ではあえて「問題」の具体的な内容にまでは踏み込まなかったが，「問題」の背景はある程度見えてきたのではないだろうか．それを簡単にここでまとめ

てから，個々の問題に視点を移して議論を進めよう．

複雑な並列計算環境を駆使して大規模計算に挑まざるを得なくなっている研究者が抱える問題と，グリッド管理者が抱える問題である直接管理できない他サイトとの協調が必須であることとの間には「現在の方法を単純に拡張しただけでは問題は解決しそうにない」という共通した特徴がある．ここでその「現在の方法」とは何かを端的に表現するならば「並列化に関して徹頭徹尾面倒を見る」と言えるのではなかろうか．言うまでもないことであるが MPI や RPC でプログラミングするには，利用しようとするノードのキャッシュ構造などから全体のネットワークのトポロジや速度に至るまで，プログラムの動作の細部にまでわたって神経を行き届かせないと効率の良い計算はできないし，管理者はといえば，基本的に管理者権限を持っているノードの状態に全責任を持つからである．そしてこの「現在の方法」の延長では問題が解決しそうにないのであれば，1 から 10 まで全部の面倒を見るのはやめて対象を適当な階層に区切って対応するのは自然なことであろう．先に述べたように本稿の目的は，計算環境の大規模化と分散化に連なる問題の整理とその解決方法を議論することであるが，言い換えれば，問題の適切な階層化の方法を議論することと同じである．本稿では管理運用の観点から大学間連携グリッドの解説を行うが，解説を行う対象である大学間連携グリッドの構築そのものがまだテスト中であるので，どうしても問題提議が中心になることをあらかじめご承知おきいただきたい．しかしながら，それが広く一般の情報処理にかかわる研究者が抱えている問題を考察するためのヒントになれば幸甚である．

なお，グリッドの生い立ちや技術的な仕組みや実現されるさまざまな機能などについては，本特集の他の記事に詳しく紹介されているのでそちらをご参照いただきたい．また，現在実際に稼働しているグリッド環境としては米国の TeraGrid と欧州の EGEE が有名であるが，話題をグリッドの管理と運用に限れば本稿で以下に解説する内容とほぼ同じなのでこれらの海外のグリッドの事情をあらわには取り上げないことにする．

CSI Grid：大学間連携グリッド基盤

議論の本題に入る前に本稿の解説の舞台となる大学間連携グリッド基盤（以下 CSI Grid）について述べる．CSI Grid は，国立情報学研究所（NII）が推進している最先端学術情報基盤（CSI：Cyber Science Infrastructure）[1] 整備の 1 つとして実施されているものであり，2008 年 5 月に Version 1.0 が公開された NAREGI[2] をグリッドミドルウェアとして採用している．NAREGI の詳しい構造については別の記事があるのでそちらを参照していただくことにして，本稿を読み進むにあたっては NAREGI がグリッドを管理する役割を担う部分（管理ノード群）と，実際にユーザが計算に使う部分（演算ノード群）の大まかに 2 つの部分からなることを理解していていただければよく，詳細が必要な場面があればそのつど説明を加えることにする．

CSI Grid の立ち上げが決まった 2008 年夏の時点では，NAREGI そのものが大規模なグリッドとしての運用実績に乏しく，また，運用する側の大学センター側にも経験やノウハウの蓄積がなく，実運用へ向けての準備はまさしく手探り状態であった．これまでも，センターの共同利用などの例が挙げられるように，大学センター間での主に事務手続き上の連携には経験があるものの，計算機を実際に提供し合って 1 つのシステムを構築するのは初めてである．したがって CSI Grid の構築や運用に付随する問題を整理したり当事者が実際に膝を突き合わせて議論して問題解決をする場を提供するために「グリッド配備・運用タスクフォース」が設置されており，技術的な問題から運用上必要になってくる事務手続きまで幅広く議論と策定が行われている．そこで，このタスクフォースの紹介を通して CSI Grid の問題点を洗い出していくことにしよう．

■ グリッド配備・運用タスクフォース

技術面や事務処理業務面にかかわらず，複数の基盤センターが互いに協力して 1 つのグリッドを構築して運用しようとするときに一般的に問題になることは，各基盤センターの事情や運用ポリシーと利用するグリッドミドルが提供できる機能との間での妥協点を，どのように落とし込むかにあるといってよいであろう．たとえば，グリッドでは自基盤センター内部のサーバを学外にある別のサーバに直接ネットワークを通じて接続する必要があるが，そのためにはファイアウォールのポリシーを変更しなければならない基盤センターが出てくる可能性が高い．別の例としては，利用するグリッドミドルが対応していない環境しか事実上提供できない基盤センターがある場合もある．したがって，タスクフォースに求められる第 1 の仕事は，このようなそれぞれの基盤センターが抱えている事情とグリッドミドルが提供できる機能とその範囲を吟味して，グリッドとしてのサービスの全体像を決めることと，それぞれの基盤センターの管理者や事務職員がしなければならない作業を見積もっていくことになる．そして，この最初のタスクフォースの仕事である作業の見積もりを行うところから，すでに本質的な問題と向き合うことになる．それは，グリッドのように複数のセンターが歩調を揃えて初めて全体としてのサービ

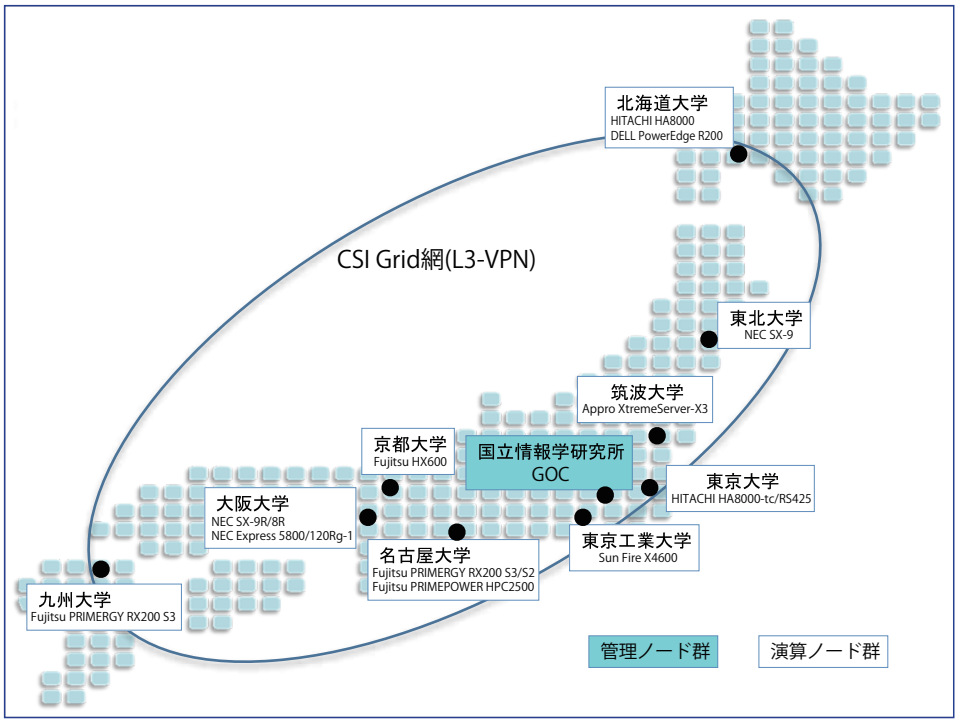

図 -1　CSI グリッド参加大学および機関

スが成り立つようなシステムを，構築して運用する経験を我々は持っていないことである．つまり，各センターの管理者たちにとっては，自分たちがユーザに提供しているサービスが，管理者権限を持たない別の基盤センターの計算機群の状態に依存している状況に対応した経験がないということであり，そこで発生し得る問題の類別も対応策もほとんど未整備で暗中模索状態である，ということである．

さて，このような状況で少しでも暗中模索度を下げるために，タスクフォースでは問題を技術面と業務制度的な側面に分けて議論することにしている．技術的な側面については，ノード構成やグリッドの機能要素を各基盤センターの都合に照らし合わせて最大公約数的なまとまりにした「ミニマム構成」と呼ばれるグリッドの構成を定義した．基盤センターでのグリッドに関する運用業務についても同様に，証明書発行を含む対ユーザ窓口業務や課金の枠組みに関して「グリッドパック」を策定した．「ミニマム構成」と「グリッドパック」はいずれも各基盤センターの都合の最大公約数的な位置づけであるので，この 2 つのミニマルセットの要件を満たしてさえいれば各基盤センターが独自に構成や業務の拡張をすることは可能である．逆に言えば，そのような拡張に支障がないようにミニマルセットは注意して構成されている．これらミニマルセットを明確にするメリットは，タスクフォース内で問題を整理する枠組みを与えるという直接的なもののほかに，各基盤センターがユーザに対して最低限保証するグリッドサービスを明確にして無用な混乱を避けることと，将来にわたる拡張性の議論の見通しを良くすることにある．なお，本稿では問題を技術面と業務制度的な側面に分けて扱っていることを思い描いていただければ十分であるので，「ミニマム構成」と「グリッドパック」の議論は割愛する．

次に，設置されたタスクフォースの構成員と活動を大まかに紹介する．7 大学の基盤センター長をメンバとするセンター長会議と，グリッドコンピューティング研究会での議論を受けて，2008 年 8 月に設置されたのが「グリッド配備・運用タスクフォース」である．参加大学および機関は，国立情報学研究所（NII）と**図 -1** に示す 9 大学である．各機関を結ぶネットワークは CSI Grid 網と呼ばれ SINET3 網上の L3-VPN[3] を利用している．各機関のグリッド内での役割分担は，NII が管理ノード群を担当し，その他の各大学センターは演算ノード群を提供する形をとっている．なお，NII は Grid Operation Center（GOC）と名付けられたグリッド運用の総合代理店的な役割も担っているほか，グリッドの電子認証に利用する電子証明書の発行も行う．

次に，タスクフォースの具体的な活動を簡単に紹介する．**図 -2** にあるように，2008 年の 8 月に発足してから直ちに基本方針を議論して作成し，ほぼ同時進行でグリッドの構築作業に入った．図は，上段が仕様策定に関することを，下段に実作業に当たる事柄を記してある．タスクフォースでは，議論の軸を技術面と事務処理業務面の 2 つにおいて進めている．双方ともそれぞれに柱があり，技術面では「ミニマム構成」を，事務処理業務面で

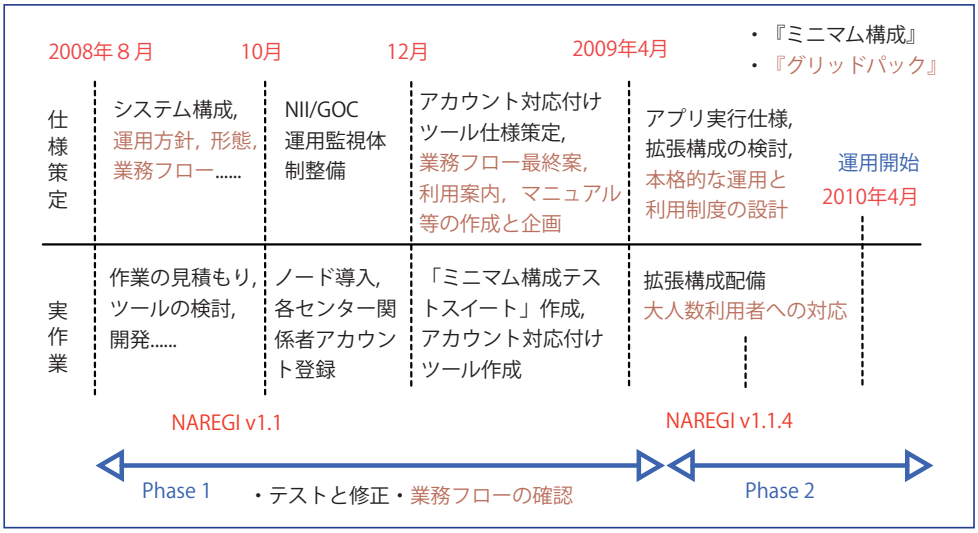

図 -2　グリッド配備・運用タスクフォースの活動

は「グリッドパック」を軸にして，それぞれ詳細な仕様が議論され策定されてきている．図中では，「ミニマム構成」に関連するものが黒字で，「グリッドパック」に関連するものが茶色の字で示してある．1 年目はグリッド構築の最初の段階で Phase 1 とされ，環境構築のほかには問題の洗い出しとテストと修正に活動の重点が置かれていた．本稿執筆時点では実運用に向けて詳細を詰めていく Phase 2 の段階に入っている．

グリッド構築と運用の諸問題

それでは本題に入って，CSI Grid の構築と運用で判明した諸問題を見ていこう．CSI Grid は NII と 9 大学の基盤センターがその構成員であり，それぞれの基盤センターから計算機資源を提供し合って構築されている．そこでまず基盤センター側で問題となるのは，CSI Grid に提供する計算機資源の詳細を決めなければならないことである．最も簡単かつ単純な方法は，CSI Grid 専用に適当な台数のノードを用意することであり，実際にそうしている基盤センターもある．これは最初の一歩としてのテスト段階ではリーズナブルであるが，理想的な状況，あるいは将来あるべきグリッドの姿としては，基盤センターが現在サービスしている環境をそのままグリッドとしても提供できるのが望ましい．そこでタスクフォースでは「既存の基盤センターの計算機資源や環境・サービスとグリッドとを共存させる方法」について議論が続けられている．グリッドに限らずどんなものでも新しいことを始めるときには，この「既存システムとの共存」が必ず問題になる．そこでよくとられる手法が，過去の資産との互換性を確保しながら中身をそっくり新しいものに置き換えてしまうか，ラッパーを作って極力既存のシステムを温存する方法であろう．中身をそっくり置き換えるのは，規模が小さいものであれば可能であるが，CSI Grid のような 10 機関がかかわる大規模な問題には現実的な方法ではない．したがって，タスクフォースでは，技術的なものも業務制度的なものもできるだけ既存の基盤センターのシステムを利用する方針を採っている．

■ グリッドの構築

では，グリッドを複数の基盤センターにまたがって展開する具体的な方法を見ていこう．まず最初にグリッドの完結した形態として最も基本的なものを取り上げる．**図 -3** は独立したグリッドを 1 つのサイト内で構築した場合の典型的な 2 つの例である．A 基盤センターは最もシンプルな構成である．グリッドにユーザがアクセスするための「ポータル」．グリッド内の必要な情報を収集して提供する「情報サービス」．ローカルスケジューラに渡すジョブを割り振る「メタスケジューラ」．ユーザのグリッド上での所属や権限を管理する「ユーザ管理」．大まかに以上の 4 つが一組で必須の管理ノード群でありピンク色で示してある．ユーザの認証に利用する電子証明書も自前で管理する場合には「認証局」も用意する．実際にユーザが計算を実行するノードは「演算ノード」であり黄緑色で示してある．実際の演算ノードは PC クラスタから SX9 のようなスパコンまでさまざまである．ユーザがグリッドでサブミットしたジョブは，最終的にはメタスケジューラが選定した「ローカルスケジューラ」のキューに送られる．ただし NAREGI の仕様では，「ローカルスケジューラ」は基盤センターがそれぞれ採用しているバッチキューシステムのローカルスケジューラをラッピングする形の GridVM scheduler（GVMS）と呼ばれるグリッドミドルウェアを必要としている．つまり，図中で水色で示したグリッドの「ローカルスケジューラ」が既存の基盤センター環境とグリッドとを共存させる鍵の 1 つを握っ

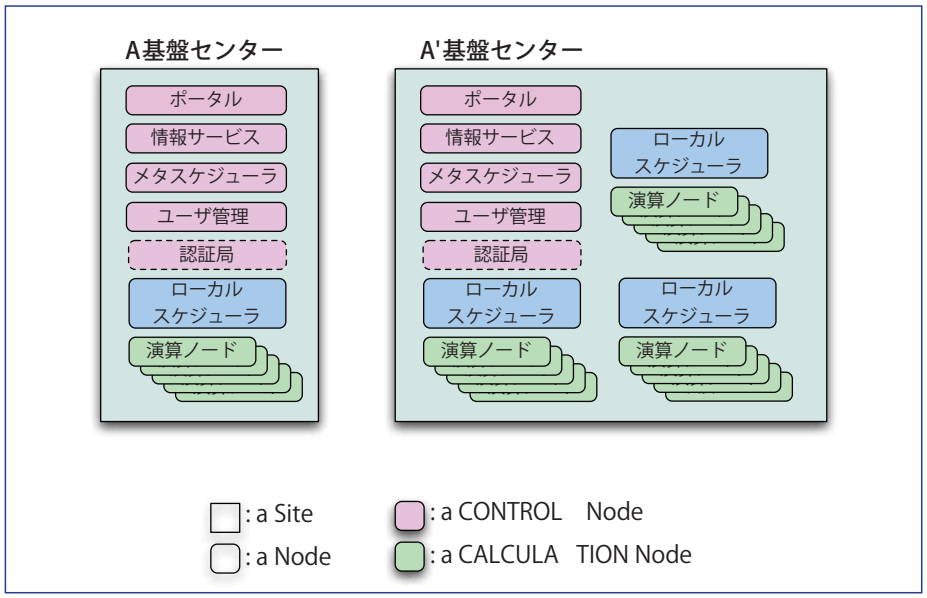

図 -3　グリッドの構成 1．1 つのサイトで 1 つのグリッドを構築する場合の例．

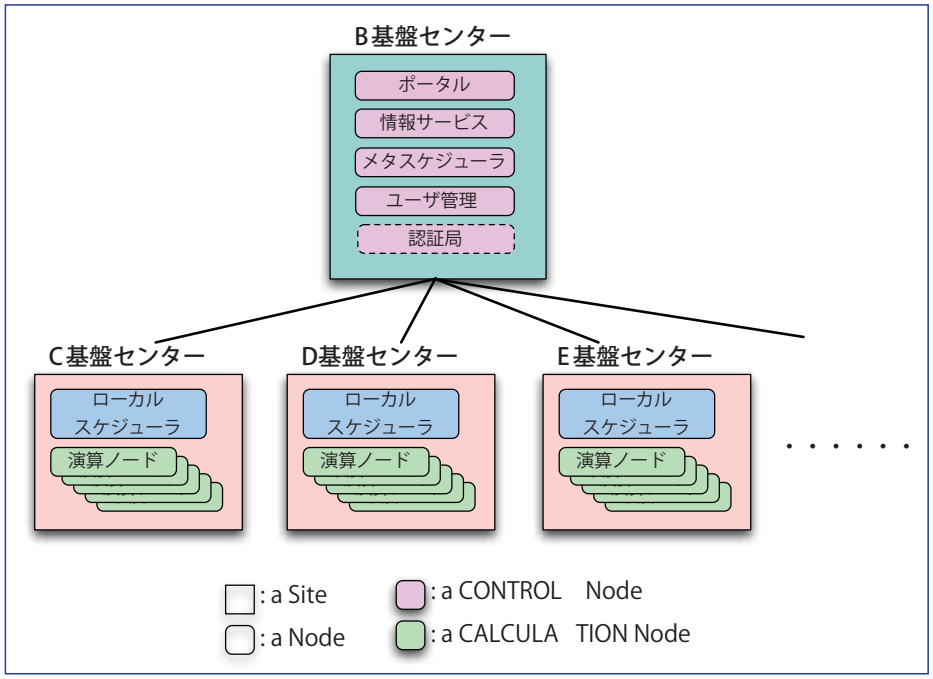

図 -4　グリッドの構成 2．複数のサイトで 1 つのグリッドを構築する場合の例．管理ノード群を受け持つサイトが 1 つあり，他の複数のサイトは演算ノード群を提供している．タスクフォースの「ミニマム構成」はこの構成を採用している．

ている．右の A' 基盤センターは，1 つのグリッド内に複数の演算ノード群を従えた場合の構成例である．1 つの管理ノード群に複数の演算ノード群を接続したもので，ユーザが複数の環境をシームレスに利用するグリッドの特徴を備えた最も単純な例である．

次に，1 つのグリッドを複数のサイトで構築する最も基本的な例を**図 -4** に示す．管理ノード群と演算ノード群をサイトで分けた最も単純な構成である．タスクフォースの「ミニマム構成」はこの構成を採用しており，図中の B 基盤センターに位置するのが NII である．9 大学の基盤センターは図中の C 基盤センター等の演算ノード群サイトとしての条件を満たせばよいことになっている．このようにサイト間でグリッドの構成を分けた場合には，各ノードの設定情報のやり取りとノード間のネットワーク接続がサイト間にまたがる問題が発生する．

最後に CSI Grid の現状に近い構成例を**図 -5** に示す．1 つの管理ノード群に複数の演算ノード群を接続できる一方で，図中の F 基盤センターのように，1 つの演算ノード群を複数の管理ノード群に接続させることもできる．

グリッドの構成に関する問題をまとめてみる．

- 「ローカルスケジューラ」ノードでの GVMS
- サイト間をまたがる各ノードの設定情報のやり取り

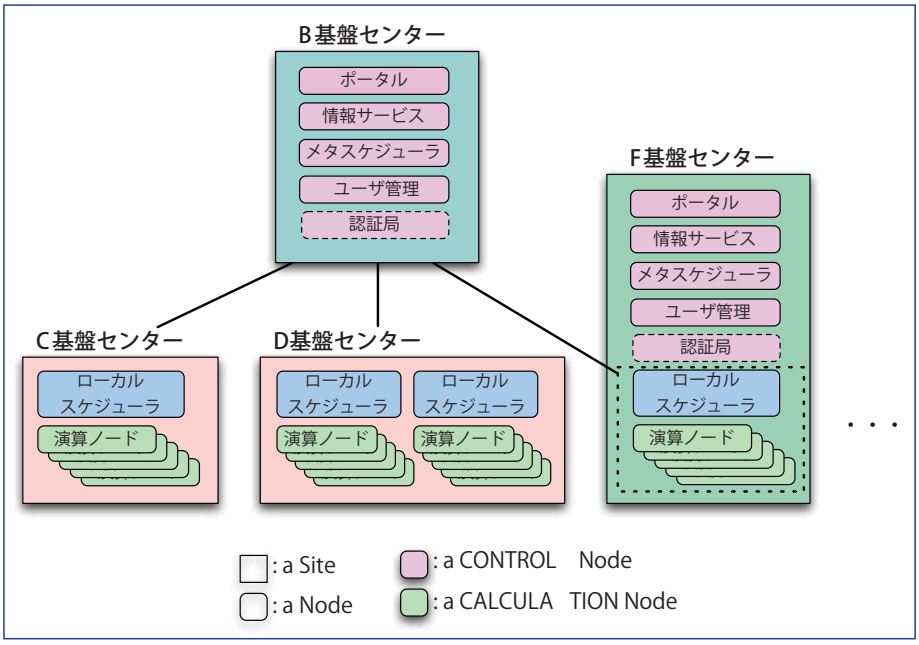

図 -5 グリッドの構成 3．複数のサイトで複数のグリッドを構築する場合の例．この例では F 基盤センターが独自に構築したグリッドの演算ノード部を基盤センター間のグリッドにも同時に提供している．CSI Grid では名古屋大学など F 基盤センターの構成を採用している基盤センターもある．

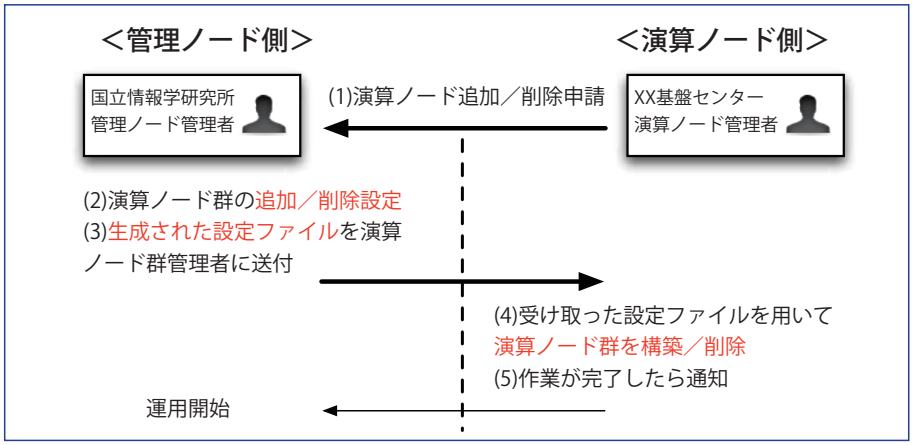

図 -6 グリッドの構築や構成変更の際に必要になるサイト管理者間の手順．赤字の部分がツールにより自動化され人為的ミスを劇的に減らすことに成功した．また，相互のやり取りも基本的に一往復であり，最小限で済むようになった．

• サイト間をまたがるノード間のネットワーク接続

「ローカルスケジューラ」の問題は，GVMS が対応しているローカルスケジューラの種類が限られていることと，基盤センターのバッチキューシステムの運用ポリシーと GVMS の動作が干渉する可能性があることである．いずれにしても現在の NAREGI を利用するには，問題が発生した基盤センターでは，GVMS を基盤センターのバッチキューシステムに合わせて改変するか，基盤センターの環境を GVMS の要求に合わせる必要がある．タスクフォースではそれぞれの具体的な対応方法を調査してその成果を共有する一方で，実際の対応はそれぞれの基盤センターの裁量に任せている．

サイト間の設定情報のやり取りは，NAREGI のインストーラを改良して対処した．**図 -6** にグリッドの構築や構成変更の際に必要になるサイト管理者間の作業手順を示す．特徴は，サイト管理者間のやり取りが一往復で完了することである．これは，設定に必要十分な情報を申請フォーマットとして整理したことと，多数のノードに対する設定作業を自動化したことによって実現した．特に設定作業は非常に煩雑であるので自動化は重要である．タスクフォースにて改良する以前の NAREGI インストーラは，図 -3 に示した構成，すなわち 1 サイトグリッドの構成しかサポートしていなかった．しかし，最初からグリッド環境の階層性を強く指向した設計になっており，管理ノード群と演算ノード群の設定ファイルを分けて管理する設計であったので改良は比較的容易であり，管理ノード群のみ，あるいは演算ノード群のみを構築するオプションの追加で対応できた．グリッド全体の整合性の

確保は既存のインストーラに実装されていた機能がそのまま利用でき，また，サイト間でやり取りをする設定ファイルもほぼそのままで利用可能であった．

さて，3つ目の項目であるサイト間のネットワークは，SINET3のL3-VPNを用いることによりファイアウォールの問題を回避した．この解決方法は問題の先送りに違いなく，本来であればコモディティなインターネット経由でも安全性と堅牢性を確保したグリッドを構築できるようにするべきであるが，そのためにはグリッドミドルが利用するポートの整理等を含めて，グリッドミドルウェアの設計から議論しなおさなければならず，グリッドの配備と運用を本分とするタスクフォースの域を超えているので，いったん保留の扱いにしている．

以上が，グリッドを構築する際に考慮しなければならない問題とその解決方法である．後半では，グリッドの運用をする上での諸問題を整理しよう．

■グリッドの運用

グリッドを実際にユーザにサービスとして提供するには，先に述べたグリッドの構築作業のほかに，耐障害性，ユーザ管理，ユーザの利用環境を整備する必要がある．それぞれの問題点を順に見ていこう．

耐障害性

グリッドのような大規模なシステムでは，Mean Time Between Failure(s)（MTBF）から想定される現場での実際の故障が大きな問題になる．CSI Gridのように日本国内で閉じたグリッド環境であったとしても，10基盤センターが各々1000ノード規模を提供したとすると，グリッド全体の総ノード数は1万台のオーダになる．仮にMTBFが24万時間のサーバ☆1をノードとして採用しても，グリッド全体で1万ノードあったとすれば1日に1台の率で故障することになる．要するに，グリッドの運用では頻繁に障害が起こることを想定しなければならないのである．そうでなくても計画停電などの各サイトの都合で一時的にグリッドから離脱するのはよくあることである．すなわち，グリッドが実運用される状況では，常にグリッド内のどこかの状態が常に変わり続けるのである．

このようにグリッドでは耐障害性への対策が重要であるが，現状では問題の把握と整理を含めて研究を進めているところである．したがって，本項でこれから述べることは仮説の域にとどまることをご承知おきいただきたい．さて，現在実用化されている耐障害性を実現したものとしては，ストレージのRAIDシステムが挙げられよう．RAIDの特徴は，広く知られているように構成の冗長性とハードウェアの存在としてのカプセル化である．冗長性は耐障害性を実現するための理論的な枠組みであり，カプセル化はRAIDを既存のシステムのストレージとして利用可能にするための実装的な枠組みであると言えよう．RAIDは既存のシステムのストレージとしてそのまま置き換えが可能であることが広く普及した要因であろう．一方で，グリッドは現在普及している計算機資源を統合して有効活用するのが目的であるから，RAIDのように耐障害性を実現するためのハードウェアを追加して解決を図るのは，将来はともかくいますぐに要求するのは非現実的である．したがって，現在の研究はソフトウェアでの解決を目指して進められている．ここでも重要であるのはリーズナブルなカプセル化の仕方を探すことであり，言葉を換えれば階層化の線引きを決めて各階層ごとに耐障害性を与えることである．

ユーザ管理

グリッドでのユーザ管理は，技術的な面よりも事務処理業務に解決すべき問題が多い．技術的な問題は，サイトローカルのユーザアカウントとグリッドのアカウントとの対応づけ程度であり，グリッドのアカウント管理はグリッドのミドルウェアですでにそのほとんどが実装済みであるからである．そこでここではグリッドを運用する上での事務処理業務面を議論する．

グリッドの運用に必要な事務処理業務は，利用者の申請書式から利用開始までの業務フローや課金など多岐にわたる．そこで包括的な仕組みが議論され策定されたのが「グリッドパック」である．

ここでも基本方針は，既存の各センター業務をできる限り尊重することである．すなわちグリッド関連業務は既存業務のラッパーとして策定されている．基盤センター間にまたがる業務は基本的にGOCの管轄下に置くことになり，GOCの業務としては認証局運用などが割り振られている．

事務手続きの面で考慮すべきことは，申請等の手続き面でのユーザの利便性を実現し，かつ，センター側の業務が煩雑にならず，これまでの業務のラッパーとして実現することである．これは，**図-7**に記したように，利用者とGOCとセンターの3者に役割を分割した後に，さらにセンターを第1センターと第2センターの2つに類別することにより解決している．第1センターとは，ユーザ側の視点から定義されるものであり，ユーザが「グリッドパック」の申請書を提出するセンターがそのユーザにとっての第1センターになる（図-7 (1)）．第1センターは申請者の身分確認や支払能力確認を行い，アカウント発行が可能と受理されると申請書にグリッド利

☆1 Dellの PowerEdge 6450 serverの MTBFは45753 hoursと公表されている．

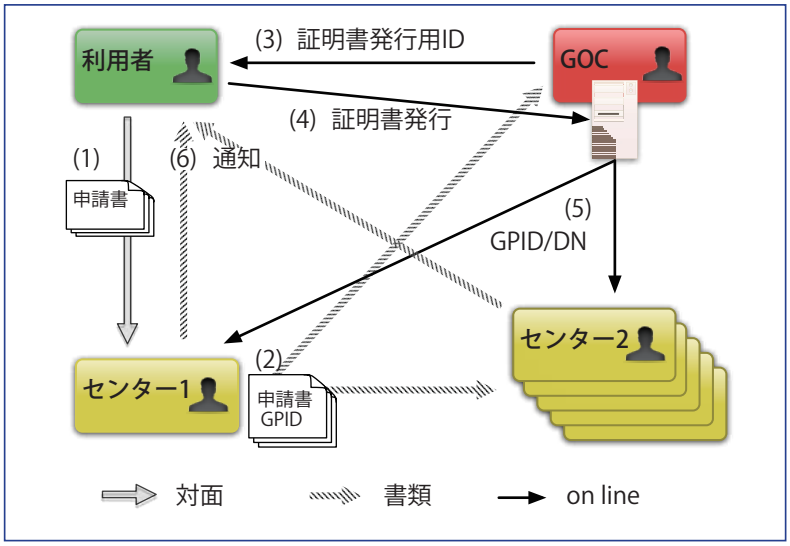

図 -7　グリッド利用申請の業務手順．ユーザは窓口となる第 1 センターに利用申請を出せばグリッドとして利用する他センターへの利用申請も同時に行われる．途中に電子証明書を発行するために GOC との 1 往復のやり取りが発生するが，基本的にユーザが書類を提出するのは第 1 センター 1 カ所のみになっている．

用者の通し番号である GPID を振って，第 2 センターと GOC に送付する（図 -7 (2)）．GOC からはグリッドを利用する際に必要な電子証明書を発行するためのライセンス ID を申請者に送付する（図 -7 (3)）．ライセンス ID を取得した申請者は GOC のサーバにアクセスして電子証明書を発行する（図 -7 (4)）．それと同時に証明書が所定の場所に格納されて subversion のリポジトリ経由で各センターに登録情報が伝わり（図 -7 (5)），各センターでのローカルアカウントと GPID との関連づけが完了すると申請者に利用許可の通知が届く（図 -7 (6)）．

この一連の「グリッドパック」利用申請処理業務の中で，事務的な問題は，身分確認や支払能力の確認方法と必要な書類がセンターごとに異なることである．「グリッドパック」では，第 1 センターが受理した申請者は，原則として第 2 センターもアカウントを発行することになっており，書類上必要な項目の摺り合わせが行われている．また，年度切り替え等でユーザの申請が一時期に集中するような場合に利用申請処理業務にボトルネックの発生が懸念されている．これは，ユーザに電子証明書発行のためのライセンス ID を送付する際に，ライセンス ID を暗号化したファイルとは別のルートで復号化キーを届ける必要があり，現在はそれを GOC の担当者が電話連絡で行っているために，短時日に大量の処理をこなせないところにある．問題点は明確で，復号化キーの送付方法である．解決案はいくつか上がっているが，本稿執筆時点では解決方法はまだ確定していない．

ユーザ利用環境

ユーザがグリッドでジョブを投入したときの問題は，ファイルシステムとコンパイル環境にある．ファイルシステムの問題とは，ファイルステージングやスクラッチファイルの扱い方が基盤センターごとにまちまちであるところにある．ファイルステージングとは，大量の IO を NFS 等の共有ファイルシステムに対して行うとシステム全体の IO 性能が著しく低下する問題を避けるために，ジョブを実行する前後に必要なデータをノードローカルのディスク上に転送することを言う．その方法が基盤センターごとに異なるのである．たとえば，ユーザのホームとジョブの実行ディレクトリを別にしている基盤センターでは，適切なファイルステージングをしないとジョブを実行できないが，NAREGI をそのまま展開したグリッド環境では，グリッド経由でくるジョブが常にその基盤センターが要求する作法に従っている保証はない．そこで，この問題には，グリッド側をユーザのホームとジョブ実行ディレクトリを環境変数で指定できるように修正して解決を行う方向で議論が進んでいる．

また，ユーザ自身が開発したプログラムをグリッドで実行しようとすると，まずはソースをコンパイルするためのジョブをグリッドに投げることになる．すると，投入されたコンパイルジョブは，どこかの基盤センターの演算ノードで実行されることになるが，演算ノードにコンパイル環境を提供していない基盤センターでは当然このジョブは実行できないことになる．これがコンパイル環境の問題である．こちらは現在解決策を模索しているところである．

大規模広域分散システムの階層構造

さて，これまでさまざまな観点からグリッドの構築と運用に関する諸問題を見てきたわけだが，これだけ多岐

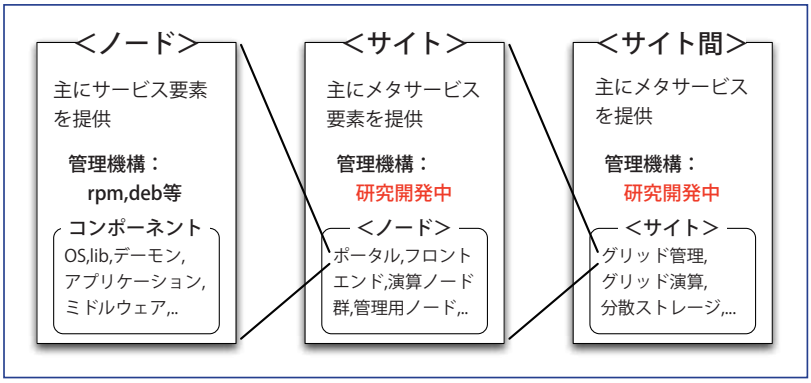

図 -8　大規模広域分散システムの階層構造

にわたる問題を綜合的に扱うためには対象を階層に分けるのが自然である．そこで最後に，グリッドの階層化について議論することにしよう．筆者らは，グリッドなどの大規模広域分散システムを階層化するための指導原理的なポイントを，その整合性の確保に置いている[4]．

図 -8 に階層化の一例を示す．「ノード」の階層は 1 台の計算機を扱い全階層中の最小単位である．「ノード」はそれ自身で計算機として独立した存在になることができるので，整合性確保の観点から最も下位の階層に位置づけるのがふさわしい．「ノード」環境の整合性を確保する仕組みは Linux においては rpm や deb 等のパッケージ管理機構としてすでに広く一般に利用されている．逆に，rpm の役割を整理してみれば，整合性確保の意味とその重要性を理解できるであろう．たとえば，rpm を用いずに kernel も gcc もすべてソースからコンパイルして Linux サーバを構築する手間を想像してみればよい．軽く数カ月の時間を要することであろう．構築作業，すなわちインストールだけでも非現実的な複雑さになるのは明らかであり，さらには，ノード全体の整合性を保ちながら何千という互いに依存し合ったライブラリの中から手作業で特定のライブラリのみを更新したり削除したりするのは夢のまた夢でしかない．このように，少し考えてみただけでもシステムの整合性を司る rpm などの管理機構の重要性は明らかであるが，不思議なことに学術研究の対象としては今までほとんど認識されていない．ともあれグリッドではすでに存在する rpm などの「ノード」環境の整合性を保証する管理機構を利用する．すなわち，グリッドのミドルウェアや現在研究開発中のツール類はすべて rpm パッケージとして配布するのを基本としている．こうして整合性が保証された「ノード」ではグリッドのミドルウェアが稼働して，グリッドを支えるサービスの要素，たとえば「ポータル」や「情報サービス」などを提供する．

「ノード」の 1 つ上の階層は「サイト」である．「サイト」は 1 つの基盤センターの様子を想像していただければよい．管理者が直接管理者権限で関与できる範囲であるのがこの階層の特徴である．「サイト」環境には，「ノード」環境での rpm のような，広く一般化された管理機構は今のところ存在せず，それぞれの基盤センターで個別に用意された管理ツール類が管理者によって叩かれて日々の管理業務が行われている．これらの現在利用されている管理ツール類は，それぞれの基盤センターの業務目的に特化されているために，グリッドの一員として求められる「サイト」としての整合性の自動確保が困難な場合が多い．たとえば，「演算ノード群」としてのメタサービス要素の提供をすることになったサイトに求められるのは，「演算ノード群」になるために必要なノード構成やノード間のサービス要素依存関係をそれぞれのサイトで解決することであり，基盤センターでの日々の管理業務とはかなり隔たったものになるからである．この階層の管理機構は現在研究開発中である．

「サイト」の上の階層は最上位の「サイト間」である．この階層には，たとえば，日本の CSI Grid と米国の TeraGrid とを結ぶような「サイト間」同士の関係も含むものとする．すなわち，大規模広域分散システムの最上位階層であり，ユーザが直接触れることになるメタサービス☆2 を提供しているのが特徴である．この階層の管理主体は，タスクフォースのようなサイト管理者の合議機関である．求められる管理機構は，メタサービスを提供するために必要なメタサービス要素とそれらの依存関係を自動解決して，各参加サイトに担当するメタサービス要素を適切に振り分けることである．この階層の管理機構も現在研究開発中である．

階層構造の 2 つの軸

最後に，まとめに代えて現在研究開発中である管理機構の基本的な概念を紹介して，上述の諸問題を解決する

☆2　1 つ 1 つのミドルウェアのサービスから見てのメタサービス．CSI Grid の場合は「グリッド」サービスそのもののこと．

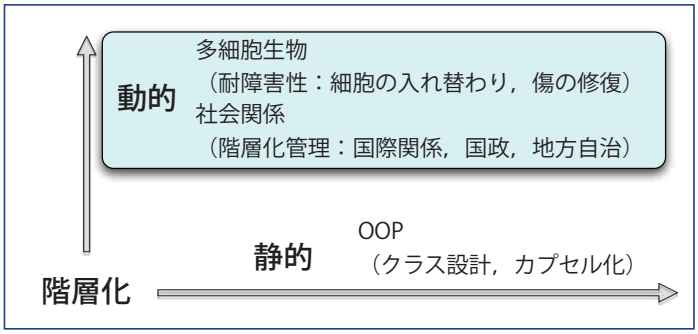

図 -9　階層構造の 2 つの軸

ための方向性を示唆しておく．

一般的に問題となる対象を階層に分けるときの難しさは，階層そのものを規定する軸の設定にあるのではないだろうか．特に，我々が直面している大規模広域分散システムを階層化するには，**図 -9** に示したように，静的な軸と動的な軸の双方を考慮して階層化する必要があると考えている．静的なものの代表としてここではオブジェクト指向プログラミングを挙げているが，ここで言いたいことは，OOP がこれまで我々が階層化という概念を展開する典型例であることである．対象が定常的であったり閉じた系であればこれで十分解決できてきたわけだが，耐障害性が問題になるような対象自体が動的に変化していくような状況では階層化概念の展開先も動的な方向へ拡張する必要性を強く感じている．その動的なものの例として，多細胞生物と社会関係をここでは挙げてみた．広く知られているように，我々の体を構成している細胞や原子分子は日々入れ替わっている．にもかかわらず我々は何十年も我々自身であり続けている．そこには耐障害性の手本があるはずであるし，多細胞生物の神経系の働きも動的な階層構造の良い見本である．一方で，社会における階層関係も，常にその階層の枠組み自体が議論され変更されているが，社会の営み自体は脈々と続いている．また，社会の階層構造は図 -8 の階層構造とよく似ている．たとえば「ノード」の階層は地方自治にあたり，「サイト」は国政，「サイト間」は国際関係に対応させることができる．この対応関係で言えば，個々のコンポーネントとしてのソフトウェアは建物に相当するであろう．そうするとソフトウェア開発者は大工さんやゼネコンになぞらえることができる．さて，造った建物が機能するには街としてのインフラが必要で，その街を設計管理するのは地方自治体の仕事であり，「ノード」では Linux のディストリビューション開発がそれに相当する．こうして見ると，グリッドの配備と運用へ向けての作業では，国の政府を作っているようなものだと言えるであろうし，住民票を取りに霞ヶ関へ行ったり，区役所が外交を担当することがあり得ないように，「サイト間」の管理で「ノード」内の情報を扱ってはいけないことも理解できる．ここで我々の普段の仕事を振り返ってみるとどうであろうか．大工さんが外交問題に取り組んだりしていないだろうか．

さて，大規模広域分散システムとしてのグリッドの管理と運用での対象の階層化を見てきたが，階層化と耐障害性が必要である点では大規模な並列計算をしなければならない研究者と本質的に同じ問題を抱えていると言えよう．本稿で扱う内容がいまだ研究開発途上であり，歯切れの悪い解説にならざるを得ないのが申し訳ないが，本稿が広く一般に大規模な分散環境に取り組んでいる研究者のヒントにでもなれば幸いである．

参考文献

1）http://csi.nii.ac.jp/
2）http://www.naregi.org/
3）http://www.sinet.ad.jp/service/network/l2/vpn/
4）Kobayashi, T. et. al. : A New Concept for Constructing HPC Environment — Packaging the NAREGI Grid Middleware by apt-rpm — , Proc. HPCAsia07, pp.249-250 (2007).

（平成 21 年 11 月 11 日受付）

小林 泰三
tkoba@cc.kyushu-u.ac.jp
2002 年立命館大学大学院理工学研究科総合理工学専攻博士後期課程単位取得退学，博士（理学），専攻は非線形物理学で金属ナノクラスターの MD で学位取得，2006 年より九州大学情報基盤研究開発センター学術研究員，NAREGI の apt-rpm packaging の設計と実装やインストーラーの設計を行う．

天野 浩文（正会員）
amano@cc.kyushu-u.ac.jp
1991 年九州大学大学院工学研究科情報工学専攻博士後期課程修了．工学博士．同大工学部助手，大型計算機センター助教授，情報基盤センター助教授を経て，現在，情報基盤研究開発センター准教授．電子情報通信学会会員．

青柳 睦（正会員）
aoyagi@cc.kyushu-u.ac.jp
専門分野は分子科学計算，近年は各種の連成解析および HPC アプリケーションの性能評価技術，グリッド計算基盤等の情報科学寄りの研究開発にも従事している．

合田憲人（正会員）
aida@nii.ac.jp
国立情報学研究所リサーチグリッド研究開発センター特任教授，東京工業大学大学院総合理工学研究科物理情報システム専攻連携教授．並列・分散計算，e- サイエンスに興味を持つ．博士（工学）．