

GFD-I.045

オープン・グリッド・サービス

コモンマネージメントモデル (CMM) WG

<http://forge.gridforum.org/projects/cmm-wg/>

編集：

Frederico Buchholz Maciel, Hitachi, Ltd.

2005 年 3 月 1 日

OGSA におけるリソース管理

本文書の位置づけ

本文書では、OGSA (オープン・グリッド・サービス・アーキテクチャ) におけるリソース管理について、グリッド関係者への情報を提供するものである。ここでは標準や技術的推奨事項などは定めない。配布は自由である。

著作権情報

Copyright © Global Grid Forum (2003-2005). All Rights Reserved.

概要

グリッドは、他の計算環境と同様にして、ジョブ、セキュリティ、ストレージ、ネットワークなどの管理を含む、ある程度のシステム管理を必要とする。グリッドでは、リソースが異機種混在型で分散していることが多く、複数の管理ドメインを横断していることから、グリッドの管理は本質的に複雑な作業となる。

本文書では、グリッドとくに OGSA に固有の管理の問題点を議論する。最初に用語を定義し、グリッドに関係する管理要件について説明する。つぎに、グリッド管理に関与する個々のインタフェース、サービス、アクティビティなどを議論する。グリッド管理には、グリッド内部の管理やグリッドインフラストラクチャそのものの管理も含まれる。最後に、OGSA のマネージャビリティの状況に関して、包括的にギャップ解析を行い、整備中の分散型管理標準では与えられないグリッド固有の管理機能を主に特定する。このギャップ解析は、今後の作業の基礎となるものと位置づけている。

目次

1. はじめに

1.1 関連作業

- 2. 定義
- 3. OGSA における管理
 - 3.1 要件
 - 3.2 レベル
- 4. リソースモデル
- 5. OGSA 機能の解析
 - 5.1 基本マネージャビリティ (インフラストラクチャサービス)
 - 5.2 一般マネージャビリティインタフェース
 - 5.3 特定マネージャビリティインタフェース

- 6. 結論
 - 6.1 ギャップのまとめ
 - 6.2 今後の作業
- 7. セキュリティに関して

著者情報

用語

知的財産権について

著作権情報

参考文献

1. はじめに

いかなる計算環境であっても、ある程度のシステム管理は必要である。たとえば、システム状態の監視とメンテナンス、ソフトウェアのアップデート、ユーザアカウントのメンテナンス、ストレージやネットワークの管理、ジョブのスケジューリング、セキュリティ管理など、さまざまな管理作業がある。その複雑さは、管理を必要とするリソースの数や種類が増えるにしたがって増大し、リソースが分散している場合にはさらに複雑になる。

グリッドコンピューティングモデルでは、複数の管理ドメインを横断する形で分散した異機種混合型のリソースが使われており、従来のあらゆる IT 管理の問題に加え、新たな問題も生じている。そしてコンポーネントリソースの管理だけでなく、グリッド自体の管理にも問題がある。たとえば、グリッド環境では、共有リソースはつねにアクセス可能でなければならず、主要なインフラストラクチャサービスも利用可能でなければならない。さらには仮想組織が維持されている必要がある。また、メンバードメインのどこかで発生するかもしれないフォルトを検知し、報告し、扱うことができなければならない。研究所や企業ではますますグリッド技術の採用が進んでいるため、グリッド環境と従来の IT 環境との区別がつきにくくなっており、上述の問題はさらに広がると考えられる。

有効なシステム管理は、リソースが管理可能でその管理にツールを使用できる場合に限って可能となる。今日、システム管理者は、システムベンダー、サードパーティ・サブラ

イヤ、オープンソース・コミュニティが提供するさまざまな管理ツールを選ぶことができる。しかしこれらのツールは、独立に動作し、限られた数のリソースを管理する専用インタフェースやプロトコルを使うことが多い。そのためこれらのツールでは、効率的で優れた統合管理システムを構築することが難しい。これは、マネージャビリティ標準の開発を通して取り組まれてきた問題である。マネージャビリティ標準は、優良な管理ツールが優良なリソースを統一的な方法で管理できるようにし、ツール同士が相互運用できるようにするものである。その場合システム管理者は、管理ツールの素性に関係なく、そのツールが統合管理環境で協調して機能するかどうかという観点から管理ツールやサプライヤを選ぶことができるようになる。

グローバル・グリッド・フォーラム (GGF) のオープン・グリッド・サービス・アーキテクチャ (OGSA) ワーキンググループ (WG) (<https://forge.gridforum.org/projects/ogsa-wg> 参照) は、ウェブサービスのインフラストラクチャに基づく次世代グリッドの実装のための標準アーキテクチャを開発している。ウェブサービスはまた、整備中の分散管理標準の基礎となるものであり、他の目的でも企業内でますます使われるようになってきている。この共通基盤によりグリッド・コミュニティは一般的な IT のために開発が行われている分散型管理を利用することが可能となるが、グリッド特有の管理要件を考え、抜け落ちていた部分 (「ギャップ」) を特定し、そのギャップを埋めるために必要であればさらなるグリッド管理標準を開発することが重要である。

本文書では、グリッドに固有の管理問題をウェブサービスや計算環境の管理問題とは切り離して詳しく議論し、ギャップを特定するところから始める。最初に用語を定義し、グリッドに関係する管理要件について説明する。つぎに、グリッド管理に関与する個々のインタフェース、サービス、アクティビティなどを議論する。グリッド管理には、グリッド内部の管理やグリッドインフラストラクチャそのものの管理も含まれる。最後に、OGSA のマネージャビリティの状況に関して、包括的にギャップ解析を行い、整備中の分散型管理標準では与えられないグリッド固有の管理機能を主に特定する。このギャップ解析は、今後の作業の基礎となるものと位置づけている。

1.1 関連作業

この作業の基礎となるものは、OGSA-WG が作成した OGSA 文書[1]とその関連用語集[2]である。

本文書ではまた、OASIS(構造化情報標準促進協会)のウェブサービス分散管理 (WSDM) 技術委員会 (TC) (OASIS と WSDM については、それぞれ <http://www.oasis-open.org> と <http://www.oasis-open.org/committees/wsdm/> を参照) で行われている作業に基礎を置くことを目的としている。以下は、WSDM の目的説明である。

ウェブサービスの管理を定義すること。これには、ウェブサービスアーキテクチャと分

散りソースの管理技術が含まれる。またこの TC では、管理可能なリソースとしてウェブサービスのモデルを開発する。

WSDM TC は、ウェブサービスの管理 (MOWS) [3]とウェブサービスを使った管理 (MUWS) [4,5]を扱った別々の文書を作成している。その文書で定められたインタフェースは、IT の世界におけるマネージャビリティの中心的な標準になるものと期待されており、グリッド管理の基礎を成すであろう。

本グループおよび他のグループにより開発された文書は熟考されたものであるが、新しいバージョンを検討していく必要があると思われる。

そのほか、以下のような関連作業がある。

- ・ e-サイエンスギャップ解析[6,7]、GGF データエリアギャップ解析[8]など、他のギャップ解析がある。これらの解析では、グリッドに関する管理について言及されているが、グリッドのマネージャビリティの側面についてはとくに解析を行っていないようである。

- ・ グリッド監視アーキテクチャ (GMA) [9,10]は、グリッドの監視アーキテクチャやその本質的なやりとりについて、主要な部分を記述するものである。監視は管理の一部であるため、われわれの仕事は GMA とある程度重なるが、対立するものではない。われわれの作業には GMA の多くの要素が含まれている。ただしその要素は、そのままの形ではなく、リファクタリングされていたり、別の用語を用いて記述されていたりする場合がある。

2. 定義

(グリッドあるいはその他の)管理は、エンティティを監視、制御し、その環境において維持するプロセスである。そして、内外の状況変化に適切に対応するプロセスである。

マネージャは、管理のアクションを起こすものである。マネージャは、人間が動かす管理コンソールであるか、あるいは対象を自動的に監視、制御できるソフトウェアエンティティのどちらかである。

マネージャビリティは、エンティティを管理する上で有用な情報を定義するものである。エンティティには、マネージャとそのエンティティのやりとりを可能にする計装を通じて管理をサポートするという側面があるが、マネージャビリティにはそうしたエンティティの側面も含まれている。マネージャビリティは、エンティティそのものか、あるいは別の手段が提供するものである。

マネージャビリティインタフェースは標準化インタフェースセットであり、これによりマネージャが共通の管理アクションを起こすためにエンティティとやりとりをすることができる。通常管理アクションには、エンティティの起動、停止、パフォーマンスデータの収集などがある。

管理可能なエンティティとは、マネージャビリティインタフェースを提供し、それによ

り（その名が示すように）管理することが可能なエンティティを指す。管理可能なエンティティは、次のようなものである。

- ・物理エンティティ（ノード、ネットワークスイッチ、ディスクなど）あるいは論理エンティティ（プロセス、ファイルシステム、プリントジョブ、サービスなど）
- ・個別のエンティティ（シングルホストなど）あるいは集団エンティティ（クラスタなど）
- ・一時的なエンティティ（プリントジョブなど）あるいは永続的エンティティ（ホストなど）

リソースモデルは、管理可能なエンティティの抽象表現であり、これによりエンティティのスキーマ（概念的階層構造および相互関連性）と特徴（属性や管理オペレーションなど）が定義される。

管理可能なリソース（あるいは単純に、リソース）という言葉は、管理可能なエンティティと同じ意味を持つ。この言葉は、ソフトウェアライセンス、帯域幅、ルーティングテーブルなどのエンティティを含んでいる。これらは、一般には有用なものであるマネージャビリティインタフェースをエクスポートしないが、他の手段によって管理されることが可能なものである。¹

リソース管理は、リソースに対して行われるさまざまな形態の管理を表す一般用語である。そこには、下記のような通常の分散リソース管理（DRM）アクティビティや、ITシステム管理アクティビティなども含まれるが、これに限られるわけではない。

- ・予約、フローカリング、スケジューリング
- ・インストレーション、デプロイメント、プロビジョニング
- ・メータリング
- ・アグリゲーション（サービスグループ、WSDM コレクションなど）
- ・VO 管理
- ・セキュリティ管理
- ・監視（パフォーマンス、アベイラビリティなどの監視）
- ・制御（開始、停止など）
- ・問題の特定やフォルト管理

リソース管理にはさまざまな管理作業が含まれるが、管理作業が使用するメカニズム（検出など）は含まれない。

リソース管理は多くの管理分野におけるさまざまなアクティビティを含んでいるため、1つだけのアクティビティを指すためにこの言葉を使うのはあいまいであり、避けるべきである。

リソースマネージャは、1つあるいは複数のリソース管理機能を実装するマネージャである。

3. OGSA における管理

3.1 要件

OGSA グリッドにおけるマネージャビリティは、WSDM MUWS の仕様[4,5]に基礎を置く。すなわち、リソースが管理可能であるためには、MUWS が定めるマネージャビリティに関する最低限の機能をリソースが提供しなければならない。現在の MUWS バージョン 0.5 では、アイデンティティ、ステート、メトリクスに関する要件が規定されている。MUWS の次期リリースでは、通知、検出、コンフィグレーション、コレクションも含まれると予想される。これらはいずれも管理に重要なものであり、OGSA サービスの範囲内で必要に応じてサポートすべきものである。

次のリストは OGSA におけるおもな管理要件を列挙したものである。これらの要件は、グリッドのような、中央集権的な制御概念を持たないスケールの大きな分散環境においてとくに重要になる。

¹グリッド環境では、リソースという言葉は、管理可能なエンティティのうちプールされたもの（ホスト、ソフトウェアライセンス、IP アドレスなど）やキャパシティを提供するもの（ディスク、ネットワーク、メモリなど）のみに使用されることが多い。このような種類のリソースのために、プールおよび（あるいは）キャパシティの一部をアロケートしたり、使用したりすることができる。この定義では、プロセス、プリントジョブ、レジストリサービス、VO（仮想組織）はリソースではない。なお、これは管理可能なエンティティとしてのリソースの定義の一部であることに注意すべきである。

・スケーラビリティ：管理アーキテクチャは、潜在的に数千もの数が存在するリソースに対しスケールする必要がある。このスケーラビリティを実現するには、管理を階層的に、そして（あるいは）ピアツーピア的に（フェデレートして、あるいは協調して）行う必要がある。したがって OGSA は、このような形の管理を許可する必要がある。階層的管理は、マネージャビリティインタフェースを通じて実装することができる。マネージャビリティインタフェースは、リソースをグループ化し、集団管理することを可能にするものである（たとえば WSDM コレクションインタフェースを実装するグリッド監視アーキテクチャ（GMA）[9,10]のアグリゲータやインターメディアリなど）。階層的管理技術には以下のものが含まれる。

- マネージャが 1 つのリクエストで複数のリソースに同じアクションを施すことができるようなプロキシを提供

- リソースデータ（たとえば平均負荷、平均予約率など）をアグリゲートするメトリクスを計算

- イベントのフィルタリングやアグリゲーション

- ステート（予約中、作動中、不具合、アイドル、サチュレート状態など）のためにリソースをポーリング、そしてリクエストに応じて結果を提供。さらにステートが変化した場合にイベントを送信（通知のプルおよびプッシュとしても知られる）。

- ・相互運用性：管理アーキテクチャは、たとえば異なる製品間の境界を超える形で、ソフトウェアやハードウェアそしてサービスの境界を広げることができなければならない。そのため、標準化された幅広い相互運用性が「ストープパイプ」を避ける上で重要となる。これには 2 種類の相互運用性が必要である。

- 異なるレベル間の相互運用性（たとえばリソースとマネージャ間など）

- 同じレベル内での相互運用性（ブローカにアクセスするスケジューラなど）

どちらの相互運用性においても、インタフェースが標準的な方法で定義されていることが必要である。これは、グリッド固有の管理標準と一般的な IT 管理標準の両方に適用される事柄である。

- ・セキュリティ：管理には 2 つのセキュリティの側面がある。

- 安全性の管理：セキュリティインフラストラクチャは管理可能である必要がある。そこには、認証、認可、アクセス制御、VO、アクセスポリシーの管理も含まれる。

- 安全な管理：管理作業にセキュリティメカニズムを使用する。管理は、それ自身のインテグリティを保証し、リソースの所有者や VO のアクセス制御ポリシーを遵守することができなければならない。

- ・信頼性：管理アーキテクチャは、いかなる不具合も許してはならない。これを可能にするには、複数の管理可能なリソースをマネージャが管理できるようにしなければならず、また管理可能なリソース側も、複数のマネージャからの管理を受けられるようにしなければならない。

- ・ポリシー：認証スキーム、トランスポートプロトコルの選択、QoS メトリクス、プライバシーポリシーなどの要件や機能をサポートするにはポリシーアサーションを導入する必要があるが、管理アーキテクチャはこのポリシーアサーションを施行することができなければならない。

・パフォーマンス監視：パフォーマンス監視機能は、「グリッド監視アーキテクチャ」で概説された下記の要件を満たす必要がある。

- パフォーマンスデータの意味が失われないよう低レイテンシを保つ
- 高データレートを処理する
- 測定コストを最小限に抑える

・ピアツーピア管理要件：大規模なピアツーピアシステムから構成されるグリッドシステムは、以下のような一般要件を満たす必要がある。これらの要件はマネージャビリティにも当てはまる[11]。

- 検出：通常の分散型システムで検出メカニズムを使用する場合、ピアツーピアシステムのメンバーシップはたいてい高度に動的であるため、効果的で効率的な検出メカニズムに強く依存することになる。

- セキュリティ：コミュニティベースのトラストメカニズム、複製、ユーザアイデンティティのベリフィケーションなどいくつか特定の要件が与えられる。ユーザのプライバシーや匿名性もそうしたシステムの特徴である。

- ロケーション認識：これは、相対的、絶対的、あるいはコンテキスト上のプロキシミティを利用するアプリケーション機能である。ロケーションベースのサービスやシステムレベルの最適化を提供する際にこれが重要となる。

- グループサポート：ピアツーピアシステムでは、一時的に数が増すような動的グループの生成や管理が可能である。ピアツーピア管理は、動的ユーザグループの生成や管理ができなければならない。

3.2 レベル

OGSA グリッドでは、リソースに関して 3 種類の管理がある。

- ・リソースそのものの管理（たとえばホストのリポートやスイッチ上のパーティション設定）
- ・グリッドリソースの管理（たとえばリソースの予約、監視、制御）
- ・リソースで構成された OGSA インフラストラクチャの管理（たとえばレジストリサービスの監視）

これらの管理を実現するため、いくつかのインタフェースがある。こうしたインタフェースは、表 1 の中央の列および図 1 の右側に示したように、3 つのレベルに分類される。

表 1 OGSA における管理とインタフェースの関係

管理の種別	インタフェースのレベル	インタフェース
リソースそのものの管理	リソースレベル	WBEM、SNMP 等
	インフラストラクチャレベル	WSRF、WSDM 等
グリッド上でのリソース管理	OGSA 機能レベル	機能インタフェース
OGSA インフラストラクチャの管理		特定のマネージャビリティインタフェース

各レベルとインタフェースの詳細を以下に示す。なおその際、マネージャビリティインタフェースに焦点を当てて解説しているが、実装部分（たとえばインタフェースを実装するサービスに関して）については触れていない。また、1つのサービスが（おそらくは互いに機能面で関連性がない）複数のインタフェースを実装することができることや、サービスとサービスが示す機能とは切り離されているものであること（たとえばリソースに対するマネージャビリティのプロバイダはこのリソースとは別物であること）に注意が必要である。したがって、サービスに関する記述は正確さに欠けることから、代わりにインタフェースに基づいた記述をここで選んだのである。

図 1 に示された OGSA の機能はすべてのレベルをカバーしており、この OGSA 機能を実装するのに必要なリソースにおける機能もそこには含まれている。インタフェースは図中の小さい丸で表されている。

- Domain-specific capabilities ドメイン固有の機能
- OGSA capabilities OGSA 機能
- Execution mgmt services 実行管理サービス
- Data services データサービス
- Security services セキュリティサービス
- Infrastructure services インフラストラクチャサービス
- Resources リソース
- OGSA functions level OGSA 機能レベル
- Infrastructure level インフラストラクチャレベル
- Resource level リソースレベル

図 1 OGSA における管理のレベル

リソースレベルでは、リソースは固有のマネージャビリティインタフェースを通じて直

接管理される。分散したリソースに対しては通常、SNMP（簡易ネットワーク管理プロトコル）、CIM/WBEM²、JMX（Java 管理エクステンションズ）、独自のインタフェースなどのマネージャビリティインタフェースが使用される。リソースレベルでの管理には、監視（イベント通知などの技術を使ってリソースの状態を取得すること）、セットアップと制御（リソースの状態を設定すること）、および検出などが含まれる。

インフラストラクチャレベルではリソースの基本管理作業が行われ、OGSA 環境におけるマネージャビリティと管理の両方に対する基盤が形成されている。複数のサプライヤが導入する大量の数と種類のリソースを、限られた数のリソースマネージャと共に統合するには、この基本管理作業の標準化が必要である。インフラストラクチャレベルでは以下のものが提供される。

- ・基本マネージャビリティモデル：これはリソースをサービスとして表し、検出やアクセス等の標準的なウェブサービス手段で OGSA のリソースを操作できるようにするものである。このモデルにより、たとえば検出、停止、イントロスペクション、監視など、最低限のリソースの管理を行うことができるようになる。

基本マネージャビリティモデルとリソースモデルを 1 つずつ使用すれば、アイデンティティ、関連性（依存関係やコンポーネントの定義）、作業状況、統計などの一貫性が管理の全階層においてとれることになり、これにより相互運用性が向上する。たとえばあるグリッドノードを予約し、そこにアプリケーションをデプロイしてアプリケーションの使用をメータリングする場合、相互運用性が成立するならば、予約が使用したアイデンティティやデプロイメントとリソース使用のサービスは同一のエンティティに対して共通のものであり、同じエンティティを参照するはずである。

基本マネージャビリティモデル自体はリソースモデルではない、という点は重要である。リソースそのもののリソースモデルは、基本マネージャビリティモデルを通じてアクセスするものである。このことを図 1 に示した。図 1 の矢印は、リソースレベルのインタフェースと、このリソースに対応するインフラストラクチャレベルのインタフェースとをつなぐものであり、前者のインタフェースを後者のインタフェースとして表している。

²WBEM（ウェブベース・エンタープライズ管理）は、エンタープライズレベルでの計算環境管理を統合するために開発された一連の技術を指す。WBEM は、リソースモデルのセマンティクスを定義する CIM（共通情報モデル）と、リソースモデルにアクセスするためのコード化とプロトコルで構成される。

- ・たとえば以下のような、OGSA の機能に共通する基本機能
 - 多くのリソースに共通する機能（開始や停止など）のためのインタフェース
 - ステートやトランジションを含むリソースの状態グラフ表現、およびステートを変える

オペレーション

- リソース間の関連性およびその種別（「含んでいる」、「使用している」など）を記述、検出する方法

- 通知

・OGSA の機能を実装するすべてのサービスに共通の一般的なマネージャビリティインタフェース。これは、イントロスペクション、監視、サービスインスタンスの生成と廃棄といった機能を持っている。

OGSA 機能レベルには、次のような 2 種類の管理インタフェースがある。この 2 つは図 1 で各機能の上部に円で表示されている。

・機能インタフェース：いくつかの共通する OGSA の機能（ジョブ管理等）は、リソース管理の 1 つの形である。OGSA 機能を提供するサービスは、機能インタフェースを通じてその機能をエクスポートする。

・マネージャビリティインタフェース：それぞれの機能には特定のマネージャビリティインタフェースがあり、そのインタフェースを通じて機能が管理される（たとえばレジストリの監視やジョブマネージャの監視など）。このインタフェースは、一般的なマネージャビリティインタフェースを拡張し、その機能の管理に専用のマネージャビリティインタフェースをつけ加えることができる。

あるジョブマネージャサービスに対する上述のインタフェースの簡単な例が図 2 に示されている。

機能インタフェースとマネージャビリティインタフェースは、（とくにリソースマネージャの場合において）明確に区別しない場合も多い。両者の重複がある程度存在する場合、これらのインタフェースが異なる役割とアクセス許可を持った異なるユーザによって起動されるため、何らかの区別をつけておくことが望ましい。たとえば図 2 では、機能インタフェースは起動中のアプリケーションのマネージャ（またはユーザ）（商用データセンター使用例[3]の「グリッド管理者」）が使用しているが、マネージャビリティインタフェースはシステムマネージャ（[3]の「IT ビジネスアクティビティマネージャ」）が使用している。

機能インタフェースとマネージャビリティインタフェースを論理的に区別する 1 つの方法は、WSDM MUWS 仕様[4,5]に概説されているように、メトリクス、作業状況、関連性、アイデンティティなどのための管理「機能」を作ることである。この区別方法では、マネージャビリティインタフェースが機能インタフェースになることも可能であり、アクセス制御ポリシーをインタフェースやカテゴリのレベルで役割や権限に基づいて作ることもで

きる。

マネージャビリティは後から付け足されることも多く、そのため機能インタフェースが存在していてもマネージャビリティインタフェースが存在しないこともよくある。

Submit job	ジョブのサブミット
Cancel job	ジョブのキャンセル
Check job status	ジョブのステータスチェック
etc.	その他
Functional interface	機能インタフェース
Job Manager Service	ジョブマネージャサービス
Manageability interface	マネージャビリティインタフェース
Start/stop service	開始 / 停止サービス
Statistics	統計
Security	セキュリティ
etc.	その他

図 2 機能インタフェースとマネージャビリティインタフェースの例

インフラストラクチャレベルのインタフェースと OGSA 機能レベルのインタフェースはサービスが提供するものであるが、それぞれ異なる性質をもつ。インフラストラクチャレベルでは、サービスはリソースのマネージャビリティ（おもにリソースモデルのセマンティクス）に対するラッパーである。このラッパーによりマネージャビリティにアクセスする手段が与えられる。OGSA 機能レベルでは、サービスはリソースの特徴よりも高いレベルで機能を提供する。あるいは、リソースモデルにアクセスしないようなインタフェースを提供する。

検出は、リソースレベル、インフラストラクチャレベル、OGSA 機能レベルの間の違いについて具体的な例を示してくれる。リソースレベルでの検出には、ネットワークに接続されたデバイスを見つけるためにネットワークをスキャンすることも含まれる。インフラストラクチャレベルでの検出には、サービスの機能を検出するためにそのサービスのメタデータ（リソースのプロパティなど）をイントロスペクトすることも含まれる。OGSA 機能レベルでの検出には、使用可能なリソースへのリファレンスを持つ 1 つあるいは複数のレジストリにアクセスすることも含まれることがある。

このようなレベル分けは、レベル間に明確なインターフェースを定義することでレベル間の相互運用性を向上させている。（たとえばサービスに直接データを送るリソースの専用ア

アダプタを使って)これらのレベルを回避するサービス(OGSA 機能を実装するサービス)をビルドすることができるものの、それは相互運用性の観点からは望ましいことではない。なぜなら、たとえばそのことによりサービスとアダプタが合うようなリソースの種類が限られるからである。

4. リソースモデル

リソースモデルは、リソースのプロパティ、オペレーション、イベントおよびリソース間の関連性を定義することでリソースを記述するものである。リソースはこのモデルが与えた記述に従って管理(監視やアロケーション等)されるため、リソースモデルはリソース管理にとってあらゆる面で重要である。リソースモデルは、機能インタフェースとマネージャビリティインタフェースの両方に使用される。

リソースモデルには次の2種類がある。

- ・ IT システム管理：システムの監視と制御に主に使われる。
- ・ リソース記述：ブローカリング、メータリング、プロビジョニングに主に使われる。

リソースモデルの例として以下のものがあげられる。

・ SNMP MIB (管理情報ベース)：これは主にネットワーク管理をカバーするものであるが、ホスト管理など他の分野でも使用される。

・ JMX JSR77：これは、J2EE (Java 2 エンタプライズエディション) プラットフォーム [13]のマネージャビリティ関連のリソースモデルである。

・ CIM：これは以下の部分に対するモデル(スキーマ)を含む³。

コア：高レベルなアブストラクション(論理のおよび物理的エレメント、コレクション)
物理的：見たり触ったりできるもの(たとえば物理的パッケージ、物理的ラック、物理的ロケーション)

システム：コンピュータシステム、オペレーティングシステム、ファイルシステム、プロセス、ジョブ、診断サービスなど

デバイス：ハードウェアの論理的機能(バッテリー、プリンタ、ファン、ネットワークポート、ストレージ量など)

ネットワーク：サービス、エンドポイント/インタフェース、トポロジーなど

ポリシー：「もし~ならば」のルール、ルールのグループ分けと適用性

ユーザとセキュリティ：アイデンティティと権限の管理、ホワイトページとイエローページのデータ、RBAC(役割に基づいたアクセス制御)など

アプリケーションとメトリクス：ソフトウェアとソフトウェアサービスの、デプロイメントとランタイムの管理

データベース：プロパティとデータベースが行うサービス（データベースコンポーネント、バックアップストレージ、ステータス、統計などを扱う）

イベント：通知とサブスクリプション

相互運用性：ウェブベース・エンタープライズ管理（WBEM）インフラストラクチャの管理

サポート：ヘルプデスクの知識の交換、問題の対処

セキュリティ保護と管理：侵入検知、ファイアウォール、アンチウイルス、その他のセキュリティメカニズムの通知と管理

ブロックストレージとファイルストレージ

アプリケーションサーバ：J2EE 環境を管理するため、JSR77 の CIM マッピングを更新（ステートとトランジションをモデル化する）ビヘビオアとステート、およびユーティリティコンピューティング（プロビジョニング、会計とメータリング、予約処理などのためのユーティリティ計算サービスと関連データの管理）の分野における新たな作業

- ・ WSDM MOWS ウェブサービスモデル
- ・ 予約、ブローカリング、スケジューリングのためのリソース記述

UNICORE リソーススキーマ

Globus RSL（リソース仕様言語）と GLUE（グリッドラボラトリ統一環境）スキーマ（<http://www.hcib.org/glue/glue.htm> 参照）

JSDL（ジョブサブミッション記述言語、GGF の JSDL-WG により定義される）

- ・ 会計とメータリングのためのリソース記述

使用記録（GGF の UR-WG により定義される）

- ・ インストレーション、デプロイメント、プロビジョニングのためのリソース記述

³JSIM（ジョブサブミッション情報モデル、GGF の CGS-WG により定義される）が複数の分野でスキーマに追加された。

- コンフィグレーション記述言語（CDL、GGF の CDDML-WG により定義される）
- データセンタマークアップ言語（DCML、<http://www.dcml.org> 参照）

いくつかのリソース記述は、それ自身モデルと指定とされたわけではないが、暗黙のモデルを含んでいる。そのモデルでは、たとえば、どのエンティティが存在しているか、そ

の属性は何か、などの点を定義している。

前述したように、グリッド技術は研究所や企業などでますます採用が進んでいるため、グリッド環境と従来の IT 環境との区別があいまいになっている。これはリソースモデルについても言えることである。従来の IT 環境で使用されていたモデルが、グリッド環境でも多く使われるようになってきており、その逆も成り立つ。そのよい例が CIM である。CIM は現在、OGSA で使うことができるリソースモデルとして検討されているところである。CIM はその幅と拡張性から、セキュリティ、実行管理、自己管理などの複数の OGSA の機能の管理作業で活用することができる。

リソースモデルでは、セマンティクスとレンダリングの区別をつけることが重要である。セマンティクスはリソースモデルの概念（モデルのエンティティ、そのエンティティのプロパティや関連性）を含んでいる。レンダリングは、与えられた言語でセマンティクスを表現すること、そして（あるいは）ネットワーク上でのモデルの送信方法やアクセス方法に関する仕様を表す。モデルのレンダリングは、そのセマンティクスを伝えるようにすることができる。たとえば CIM モデルはリソースのセマンティクスを含んでいるが、その XML 表現や HTTP マッピングは CIM のレンダリングである。セマンティクスが複数のレンダリングを持つことがある。たとえば他の CIM レンダリングもこれまでに提案されている。

リソースモデルのセマンティクスにはその意味が含まれており、そのためそのレンダリングよりも相互運用性を達成する上でより重要である。1 つのモデルに対して 2 つのレンダリング間でトランスレートすることは難しい問題ではないが、2 つの異なるモデルに対するセマンティクス間のトランスレーションは複雑になる場合が多い。たとえば、モデルによってファンは物理的エンティティであったり論理的エンティティであったりする。ファンをシャーシ、冷却装置、エンクロージャサービス、物理的パッケージングなどに分類することもできるが、あるいは具体的な数値は異なるものの同じようなプロパティ（ステータスなど）を持つものとみなすこともできる。セマンティクス間の自動トランスレーションは、そのセマンティクス同士がマッチしない限り行うことができない。このマッチングの例の 1 つが、GRIP プロジェクト[14]（<http://www.grid-interoperability.org> も参照）の一端として行なわれている Globus と UNICORE のリソース間のマッピングである。また CIM は、自身のセマンティクスを他のリソースモデルのセマンティクスにマップするメカニズムを持っている[15]。

理想的には 1 つのリソースモデルを使うことが望ましい。これは、モデル間を仲介することと比較すると、1 つのリソースモデルを使ったほうが相互運用性を達成しやすいからである。しかしこれは理想論である。一般的な IT やグリッドのこれまでの発展は、リソースモデルを 1 つにまとめる方向には進んでこなかった。そのため与えられたグリッドで複数のリソースモデルを同時に使用することを想定しなければならない。したがって、モデル同士の整合性をとるために（GLUE スキーマで行なわれているように）モデル間で調整を

行うこと、そして異なるモデルのセマンティクスにマッチするメカニズムが必要である。これはとくに OGSA にとって重要なことである。OGSA では、グリッドの機能が相互運用しなければならない複数の機能の組み合わせで構成されているからである。その機能のそれぞれが複数のセマンティクスおよび（または）レンダリングをおそらく使用しているのである。

既存のモデルを再利用して新しいリソースモデルを作成するのが望ましい。これにより、より高い相互運用性を達成できるだけでなく、手間も少なくなるからである。たとえば、この新しいリソースモデルは別のリソースモデルの一部としても上位集合としても作成することができる。あるいは複数のリソース記述を 1 つのリソースモデルのレンダリングとして作成することもできる（その際、このモデルあるいはその一部を表現するそれぞれのリソース記述言語を用いるが、モデル独自の構文たとえばモデル独自の XML スキーマを使う。）

リソースモデル間の調整が必要となる分野が 2 つある。

- ・リソース記述間の調整（予約、メータリング、プロビジョニング等の OGSA サービス間の相互運用性を容易にするため）
- ・標準管理モデルとリソース記述の間の調整（リソースとリソースマネージャ間の相互運用性を容易にするため）

リソースモデルに対する作業で望ましいもう 1 つの方向性が、リソース管理のためのメカニズムについてモデル中立性をとることである。モデル中立性により、モデル自体が統一されていなくても、複数のリソースモデルを使用するメカニズムの統一が可能になるのである。WSRF と WSDM はこうしたメカニズムの一例である。

5. OGSA 機能の解析

ギャップ解析には、それぞれの OGSA 機能に対するマネージャビリティインタフェースの各レベルにおいて見失った機能を見つけ出すという目的がある。そのためギャップ解析は、図 3 に示したように、管理レベルを横の列、OGSA 機能を縦の列とする表の空欄を埋めるものとして概念的には考えることができる。表の欄が空白（あるいは不十分）の場合、そこがギャップを表す。ただし機能インタフェースの解析は OGSA-WG の仕事の 1 つであるため、ギャップ解析はマネージャビリティ（基本マネージャビリティ、一般的マネージャビリティ、特定マネージャビリティのインタフェース）のみをカバーする。必要な場合は機能ごとにモデルを調べる。

機能 レベル	実行管理サービ ス	データサービス	・・・	セキュリティサ ービス
-----------	--------------	---------	-----	----------------

特定マネージャ ビリティインタ フェース	(5.3.1 節)	(5.3.2 節)	...	(5.3.7 節)
一般マネージャ ビリティインタ フェース	(5.2 節)			
基本マネージャ ビリティ	(5.1 節)			
モデル	(5.3.1 節)	(5.3.2 節)	...	(5.3.7 節)

図3 ギャップ解析 (概念図)

ギャップ解析では、管理の候補となるグリッドの要素の一覧を作成するため、マネージャビリティインタフェースを提供する必要がある。このリストは、可能であるはずの管理アクションの種類や必要な共通マネージャビリティインタフェースセットを特定する際に使用するよう作られている。インタフェースによってはすでに定義されているが、これから特定する必要があるインタフェースもある。このリストは OGSA 文書[1]から部分的に引用したものである。

OGSA の機能は、その背景にある多くの仕様と同様に、現在もまだ定義が進んでいる最中であるが、分野によってはマネージャビリティの解析を完了させるための作業が十分に進んでいない。仕様の作成が完了したケースもあるが、それを採用するかどうかは明らかではない。そのような場合、ギャップ解析により今後解析が必要になる項目がはっきりすることになる。

5.1 基本マネージャビリティ (インフラストラクチャサービス)

5.1.1 解析

OGSA ベースのグリッドでは、すべての管理可能なリソースはウェブサービスであるか、あるいはウェブサービスが表現するものであるか、のいずれかである。リソースを表現する際に基本となるものが、仕様に関する WS-リソースフレームワーク (WSRF) ファミリーにより定義されたインタフェースとビヘビアである (<http://www.oasis-open.org/committees/wsrf> 参照)。さらに OGSA では、イベント通知のための WS-通知 (WSN) のインタフェースとビヘビアが使用できることを想定している (<http://www.oasis-open.org/committees/wsn> 参照)。WSRF と WSN は、管理に便利なインタフェースの基本セットを指定している。たとえば、以下のような特性がリソースにより実装された場合、適切な権限を持ったマネージャがこれを利用することができる。

- ・WS-リソースライフタイムは、リソースの停止時刻をクエリしたりそれを変更したりするためにマネージャが使用できるオペレーションを指定する。これによりリソースがただちに停止することも多い。

- ・WS-リソースプロパティは、リソースのプロパティのリスト(リソースプロパティドキュメント)を発行する手段や、マネージャがそのドキュメントを取得したりプロパティの値のクエリや修正を行うための手段を提供する。

- ・WS-リソースプロパティはまた、リソースのプロパティ値が変化した場合に、マネージャが通知をリクエストしたり受け取ったりすることができるような機能を定義する。一般にこの機能は、WSN 通知メッセージを使って提供される。

WSRF と WSN の仕様および WSDM MUWS は、基本マネージャビリティインタフェースに対する中心的機能を提供する。以下では、そのそれぞれの仕様をリソース管理に適用した場合の重要な特徴を示す。

- ・WSRF 仕様は、現在のところ、WS-リソースプロパティ、WS-基本フォルト、WS-リソースライフタイム、WS-サービスグループを含んでいる。これらの仕様はいずれも、次のような重要な管理上の特徴を有する。

リソース表現：WSRF は、ウェブサービスとステートフルリソースとを関連付けるために、インプライドリソースパターンを定める。

リソースプロパティ値：いずれのリソースプロパティも、適切な権限をもつマネージャにより検出、クエリ、修正することができる。

監視機能：マネージャは、リソースプロパティの値に対するいかなる変更についても、その非同期通知を予約しておくことができる。WSRF は通知サポートに関しては WSN に依存する。

リソースライフタイム管理：指定された時間に自動的に破棄するようリソースをスケジュールすることができる。スケジュールされた破棄時刻は変更することが可能であり、ただちに破棄することもリクエストできる。

サービスアグリゲーション：サービスのコレクションは、サービスグループにより表現される。サービスグループは、特定の管理ドメインや他の組織に属するサービスや、特定の管理オペレーションを必要とするサービスをグループ分けする際に有用である。マネージャは、サービスグループのメンバーシップへの変更に関し、通知をリクエストしたり受け取ったりすることができる。ただしサービスグループは、そのグループの全メンバーに対するバルクオペレーションをはっきりと提供することはないことに注意すべきである。

フォルト管理：WS-基本フォルトは、ウェブサービスのフォルトメッセージを支持する共通の方法を提供することにより、フォルトの判断と管理をサポートする。

・WSN は、トピックベースのイベントサブスクリプションを通じた、通知ソースあるいはブローカからのリソース監視をサポートしている。通知は、たとえばリソースが生成された、破棄された、あるいはリソースのプロパティの値が変わった、などの知らせをもたらす。WSN 仕様には、WS-基本通知、WS-ブローカード通知、WS-トピックスが含まれる。

・WSDM MUWS は、ウェブサービスを使って管理基盤を与えるものである。以下の機能は、現在検討されているこうしたウェブサービスの一部である。

アイデンティティ：リソースのユニークな識別子。

ステート：リソースの状態グラフに対する基本モデル。標準状態、トランジション、ステートを変えるオペレーションなどがある。この状態グラフは拡張可能である。

メトリクス：リソースから集めた値（統計など）および関連するメタデータ（メタデータはカウンタあるいはゲージ、収集の周期などメトリックの種別を決めることができる）そしてリセットオペレーション。

コンフィグレーション：リソースのビヘビアに影響を持つプロパティの表現と変更。

通知とイベント：イベントのための共通レンダリング。既存の複数のイベント形式に共通する基本情報を含む。

リソース間の関連性：関連リソースおよび関連性の種別（「含まれている」、「使用している」など）。

コレクション：リソースのグループ分け。これによりマネージャが、各リソースに別々に作業を行う代わりに、グループのすべてあるいは一部に対して作業をすることができる。

検出：リソースおよびそのマネージャビリティを見つけ出す手段。

相互に関連付けできる名称：リソースに関する追加情報。これにより、2つのマネージャが、持っているリソースに同じものがあるかどうかを判断することができる。

・WSDM MOWS は、ウェブサービスの管理に対し、MUWS 仕様に基づいて作成されている。以下の MOWS 機能は、MUWS の機能に加えて現在検討されているものである。

アイデンティフィケーション：管理されているウェブサービスのエンドポイント

リクエスト処理ステート：ウェブサービスにサブMITされたそれぞれのリクエストの処理状況

5.1.2 ギャップと残された問題

以下のギャップが特定されている。

・マネージャビリティ機能性、そしておそらくはリソースモデルを、インフラストラクチ

レベルにおけるサービスに定義する必要がある。

こうした一般的なファクトリサービスを特定することは重要である。結果として、これらのサービスは他の主要なインフラストラクチャサービスと同じように管理することができるからである。マネージャビリティインタフェースは、ファクトリがどのサービスを生成するかをクエリする上で必要になる。また、ステータスやパフォーマンスを監視する上でも必要である。

WS-合意は、インフラストラクチャの一部であり、予約やデータアクセスなどのアクティビティに使用される。これらのアクティビティは、いずれも専用のインタフェースを持ち、専用の管理を必要とする。その正確なオペレーションとパフォーマンスはグリッドにとって重要であり、監視する必要がある。

・ WSDM TC は現在、MOWS のウェブサービスモデルを作成し、MUWS へのマッピングを定義している。ただし TC の網領には MUWS から他のモデル (CIM、SNMP、グリッド関連モデルなど) へのマッピングの定義は含まれていない。これらのマッピングは、定義が必要である (WSDM に関する詳しい記載では、リソースモデルやそれぞれのモデルのマッピングを想定している)。DMTF (分散管理タスクフォース) の WS-CIM ワーキンググループは現在、CIM からウェブサービスへのマッピングについて検討を行っている。

・ グリッド関連モデル間のマッピングや IT 標準 (CIM や SNMP など) 間のマッピング方法については研究されているが、著者の知る限り、一般的な IT 標準モデルを用いたグリッド関連モデルのマッピングについては現在のところ研究されていない。

以下に残された問題を列記する。

・ WSDM のステートモデルがジョブ制御やプロビジョニングなどの OGSA の機能に適しているかどうかを調べる必要がある。

・ オリジナルの CMM プランには、「オペレーショナルポートタイプのように、複数のリソースあるいはドメイン固有のリソースマネージャから取り出したカノニカルなサービス (開始 / 停止 / 一時停止 / レジューム / Quiesce)」が含まれている。この特定のインタフェース (開始 / 停止) は、WSDM のカノニカルステート・オペレーションで実現することができる。他のカノニカル・インタフェースの必要性については検討を要する (これらは WSDM で現在予定されている機能に含まれていない)。

・ WSDM はモデルに依存しないため、最低レベルの相互運用性を可能にするために使用する一組のリソースモデルを選ぶ必要がある。1 つのリソースモデルに関してアグリーメントが期待できない場合は、おそらく一組のプロファイルを定義しなければならないことにな

る。扱わなければならないリソース（たとえばライセンス）がたくさんあり、この定義は複雑な作業となる。

- ・1つのリソースには、複数のリソース管理アクティビティのための複数の状態グラフ（デプロイメント状態、作業状態など）が必要となる。WSDMはこの機能を有しているが、この機能が必要な場合はベリファイしなければならない。

5.2 一般マネージャビリティインタフェース

5.2.1 解析

OGSA のサービスはいずれも、少なくとも停止、イントロスペクション、監視などの最低限の管理を行うためのインタフェースを提供する。OASIS WSDM TC では、OGSA のサービスに適用できるウェブサービスのための標準マネージャビリティインタフェース（MOWS）を定義することになっている。

5.2.2 ギャップと残された問題

以下のギャップがこれまでに特定されている。

- ・セキュリティ機能が普及しているため、すべてのサービスに共通のセキュリティ管理のアクティビティが必要である。そうしたアクティビティの例としては、異なる役割（エンドユーザやマネージャなど）のためのサービスへのアクセス許可の管理、使用するプロトコルバインディングの管理などがあげられる。

- ・いかなるサービスでも、サービス拒否攻撃などのセキュリティ上の問題を抱えることがある。理想的には、マネージャビリティが、そうした問題が発生した際に問題点を知らせてくれる手段を提供するべきである。そのようなマネージャビリティインタフェースを、すべてのサービスに対して定義する必要がある。そこには WSDM におけるリソースのマネージャビリティもおそらく含まれる。

以下に残された問題を列記する。

- ・OGSA グリッドに固有の一般的なインタフェースが MOWS 以外にあるかどうか、検討する必要がある。

5.3 特定マネージャビリティインタフェース

この節では、特定マネージャビリティインタフェースの解析、および特定の機能に固有のモデルがあればそのモデルに対する解析を行う。以下の節では OGSA 機能レベルでの機能、すなわちインフラストラクチャサービスを除くすべての OGSA 機能を解析する。それぞれの機能について、なぜそのサービスを管理することが重要なのかという点を説明し、そのマネージャビリティの解析を行い、ギャップと残された問題点を指摘する。

5.3.1 実行管理サービス

5.3.1.1 解析

OGSA の実行管理サービス (EMS) 機能は、リソースの選択、予約、コンフィグレーションを扱い、リソース上でのプログラムの実行を制御するものである。これはグリッドの中心的な機能であり、効果的な管理が重要となる。マネージャビリティの観点から EMS を解析した結果を以下に示す。

・ジョブとサービスコンテナはリソースである。これらにはその機能とプロパティを定義するリソースモデルがある。このリソースモデルには WSDM 準拠のインタフェースを通じてアクセスすることができる。ジョブドキュメントは、ジョブのマネージャビリティ情報 (すなわち属性) に対応する。

サービスコンテナ (たとえばノード) の場合、リソースそのものには、リソース消費の監視や実行停止などの基本的な作業のためのマネージャビリティインタフェースがある。しかしリソースは、必要とされる機能すべてに対してインタフェースを直接提供することはない。(たとえばリソースがデプロイメントに対するインタフェースを提供することはない場合が多い。) したがってリソースのマネージャビリティプロバイダが不足しているインタフェースを提供しなければならない。サービスコンテナのマネージャビリティは、それが直接提供されたものであっても間接的に提供されたものであっても、実際のリソースに「近い」ものである可能性が高いことに注意するべきである。たとえば多くの場合、そのマネージャビリティはリソース自体に実装され、そして (あるいは) 作動するのである。

ジョブもリソースであるが、サービスコンテナとは大きく異なる。EMS では、ジョブはリソースがコミットされ実行が開始される前にジョブを生成すると定めている。そのためジョブの生成時には、どのサービスコンテナがそれを実行するかはわからない。したがって、ジョブのマネージャビリティをジョブの実行リソースに近いプロバイダを通じてのみ行うように要請するのは不可能な話であり、ジョブ実行の動的な性格上、ジョブの場所をその終了時までプロバイダが追跡することは困難である。この問題への対処法の 1 つは、ジョブマネージャがマネージャビリティプロバイダを持ち、ジョブへのリファレンスをコンテナヘアサインするようその後のマイグレーション等で更新することである。

・ジョブマネージャ、実行計画サービス、候補セットジェネレータ、情報サービス、デプロイメントサービス、コンフィグレーションサービス、予約サービスは、いずれも OGSA 機能レベルでのサービスである。

リソース (ジョブやサービスコンテナ) には、OGSA 機能レベルのサービス機能を有効にする機能を持ったインタフェースがある。ただしこのリソースが自らこれらの機能を実装することはない。たとえば、

リソースはマイグレーションを有効にする機能を持っているが、マイグレーションは機能レベルでは実際には実装されない。

ジョブマネージャは、ジョブの再スケジューリングを行ったり一組のジョブにバッチオペレーションを行ったりするインタフェースを提供するが、これはジョブそのものの機能ではない。

5.3.1.2 ギャップと残された問題

ギャップは見つかっていない。

以下に残された問題を列記する。

- ・ジョブとサービスコンテナのリソースモデルを定義する必要がある。既存のリソースモデル（CIM、GLUE、UNICORE リソーススキーマ、など）を解析し、再利用することが必要である。

- ・ジョブマネージャがクライアントに対して（ジョブに対するインタフェースを拡張するなどして）ジョブの同じインタフェースをエクスポートすべきかどうか、検討が必要である。その場合、ジョブの制御は一度で済むことになる。

- ・ジョブに対するマネージャビリティプロバイダは、実際のジョブに「近い」とは限らない。ジョブに近いことが保証されているインタフェース（実際のコンテナにあるマネージャビリティプロバイダにより実装されたもの）も定義する必要があるかどうか、より解析を進めて決める必要がある。その場合、ジョブマネージャとコンテナとの間の相互運用性が得られることになる。

- ・パーシステントステートハンドリングサービスがリソースであるか、それとも OGSA 機能レベルでのサービスにあたるのかが明確でない。

- ・マネージャビリティインタフェースに関して

ジョブ管理のためのマネージャビリティインタフェースがある（JSIM）。このインタフェースの機能が実行管理サービスのニーズを十分満たしているかどうか、より深い解析が必要である。

たとえば、ジョブマネージャは個々のリソースキューの状態を監視し、制御できなければならない。例として、負荷のバランスをとるためにキュー間でジョブを移動すること、優先順位を変えること、計画したダウンタイムに合わせることなどができなければならない。ジョブマネージャはまた、OGSA で定義されたように、ワークフローややりとりを行わない多数のジョブを管理する場合がある。これには、ジョブインスタンスの特定や表現など、特定のマネージャビリティ機能が要求される。

実行計画サービスの制御や監視、あるいは候補セットジェネレータなどのための、他の

マネージャビリティインタフェースが必要になる。

5.3.2 データサービス

5.3.2.1 解析

データサービス機能には、データアクセス、表現、変換、管理などに対する機能を提供するサービスが含まれる。多くのグリッドでは、このようなサービスがいろいろな種類で多数存在しており、すべてのグリッドではないがほとんどのグリッドにとってこれらのサービスは基本的なものである。これらのサービスは重要なインフラストラクチャサービスであり、そのアベイラビリティとパフォーマンスは監視や管理が必要である。

OGSA 機能レベルでは、データサービスはおもにデータ管理、すなわちデータのプロビジョニング、アロケーション、キャッシング、複製、仮想化などを扱っている。データサービスは、関係するさまざまなデバイス（リソース）において、最終的には機能インタフェースおよびマネージャビリティインタフェースを通じて実装される。データそのものをリソースとしてモデル化することは可能であるが、データサービスに関する現在の提言ではこのことに重きを置いていない。

5.3.2.2 ギャップと残された問題

現在、データサービスのアーキテクチャが OGSA-WG 内の設計チームにより作成されており、機能インタフェースとマネージャビリティインタフェースの両方が他の GGF WG によって定義づけが行われているところである。

現段階では、ギャップや残された問題点は明らかではないが、データサービスのアーキテクチャに関して進展が見られれば、マネージャビリティのより詳細な解析を行うことが可能になるはずである。とくにマネージャビリティは忘れてはならない。他の機能と同様に、データサービスのインタフェースは（キャッシングや複製の効率を監視するなどの）インフラストラクチャ管理を可能にするものでなければならない。

5.3.3 コンテキストサービス

5.3.3.1 解析

コンテキストサービスは、現在のところ VO 管理とポリシー管理から構成されている。現行のサービス記述では、ギャップ解析を行うことはできないが、マネージャビリティの解析は行うことができる。

VO は複雑なリソースであり、重要な管理上の問題を提供するシステムでもある。マネージャは VO のレジストリを検出、管理し、VO を作成、破棄し、個々の VO に割り当てられたリソースとユーザを管理することができる必要がある。VO がグリッドという概念の根幹を成すとすれば、その管理は重要である。

VO を操作するインタフェースは、VO の生成や破棄などのマネージャビリティ機能を提

供する。またこのインタフェースは、ユーザ、グループ、サービスなどのエンティティを、VO、VO 内のユーザの役割の操作、VO に対するアグリーメントやポリシーの付与に結び付ける役目も果たす。このインタフェースには、ある種のモデルが必要になることがある。なお、CIM スキーマのようなセキュリティやユーザ管理に関係する既存のモデルを再利用することも可能である。

ポリシー・サブシステムは、十分に定義されていれば、レポジトリを含む複数の関連サービスから構成される場合が多い。ポリシー・サブシステムは、ほとんどのグリッドで重要なインフラストラクチャコンポーネントであるため、それを監視し特定のエレメントを制御する機能が必要不可欠である。

2 種類のマネージャビリティインタフェースが必要になる。1 つはリソース上でポリシーを管理（追加、削除、変更など）するためのもの、もう 1 つはポリシーそのものの管理を行うポリシーレポジトリに対するものである。

5.3.3.2 ギャップと残された問題

上述のように、ギャップ解析は現段階で行うことができない。

5.3.4 情報サービス

5.3.4.1 解析

OGSA 情報サービスは、リソース管理の監視に関する部分に焦点を当てたものである。そこには監視情報の送信や保管といった関連アクティビティも含まれる。情報サービスには、リソース管理の 2 つの重要なコンポーネントである検出サービスとロギングサービスが含まれている。イベントの管理もここに分類される。

検出サービスは、すべてのグリッドでデプロイされる可能性が高い。3.2 節で触れたように、検出は多くのレベルで行われ、OGSA 機能レベルにおける機能はおもにレジストリから構成されている。サービスは、サービスとして表現されたリソースを含めて、1 つあるいは複数のレジストリに登録される必要がある。そうすることにより、サービスを検出することができ、サービスのインタフェースや機能に対してクエリできるのである。おもなレジストリサービスは、グリッド内のすべてのリソースを検出、マッピング、そして管理するための出発点となるものである。レジストリサービスが使用可能であり、正しく機能することが重要である。そのためマネージャは、その動作やパフォーマンスを監視し、必要に応じてインスタンスやコピーを生成、破棄できる必要がある。

ロギングサービスは重要なインフラストラクチャサービスであり、それに依拠して管理する必要がある。ロギングサービスのパフォーマンスを監視するだけでなく、ストレージスペースの限界や、容量が低いあるいは不十分であると判断する条件、定期的な消去、アクセス制御など、多くの側面を扱う必要がある。1 つのグリッドの中にある別々の管理ドメインは、保存などに関して異なるポリシーを持っていることがある。このことがより複雑な

管理オペレーションの1つとなるケースも多い。ロギングサービスは、現在 GGF 内で立ち上げられつつある新しいワーキンググループが定義することになる。

情報サービスの中心となるものの1つが、コンシューマとプロデューサのインタフェースである。このインタフェースは、グリッド内の監視情報を発行、取得するための統一された方法を提供するものである。WSDM は、コンシューマとプロデューサのインタフェースの基本的要件、たとえばモデル中立性や拡張性など、を満たしている。しかしコンシューマとプロデューサのインタフェースでは、プロデューサからコンシューマへデータを送るプッシュモデルや、監視情報を取得するためのクエリや保持期間の設定などを含めた監視情報の永続的保管、情報のアグリゲーション、メトリクス（リソース間平均、時間平均などの統計関数）の計算など、より豊富な機能を想定している。この機能の少なくとも一部分だけでも、WSDM インタフェースを拡張することで実装することができなければならない。そうすることにより、WSDM が、コンシューマとプロデューサのインタフェースの基礎となるかもしれない。

情報サービスでは、情報伝達の基本としてサービスのメッセージングやキューイングを暗に想定している。これらのサービスは重要なインフラストラクチャサービスとなる可能性が高い。メッセージボリュームやストレージスペース（該当する場合に限る）を取り扱うため、パフォーマンスの監視や、使用できるインスタンスやコピーの数の管理などが管理要件に含まれる。

5.3.4.2 ギャップと残された問題

以下のギャップが見出されている。

- ・レジストリ、メッセージングサービス、キューイングサービスのためのマネージャビリティインタフェースが必要となる。また、おそらくはそのマネージャビリティを表現する単純なモデルが必要となる。
- ・専用の通知サービスやイベントサービスが定義されている場合は、これらのサービスを重要なインフラストラクチャサービスとして管理する必要がある。

残された問題点を以下に列記する。

- ・プロデューサとコンシューマのインタフェースがどのように WSDM インタフェースに関係しているかが今のところ明確ではない。OGSA での監視におけるそれぞれのインタフェースの役割も同様に明らかではない。この問題は、プロデューサとコンシューマのインタフェースの仕様を定義する際に解決する必要がある。
- ・保持機関や消去ログなど、管理作業のためのロギングにマネージャビリティインタフェースが必要である。ロギングインタフェースに関する現在の提言では、これらの機能の一部が取り入れられているものの、それで十分かどうか検討する必要がある。同様のことがプロデューサとコンシューマのインタフェースにも当てはまる。

- ・グリッド環境に固有のイベントが存在するかどうか、調べる必要がある。

5.3.5 リソース管理サービス

5.3.5.1 解析

リソース管理サービスは、その名前に関わらず、OGSA のリソース管理のための機能の一部を提供するに過ぎない。リソース管理サービスは、ほとんどの種類のリソースに適用できるかどうか分類上の基準となる。すなわち、予約、プロビジョニング、メータリングはこのカテゴリに入る（たとえば、帯域幅はリソースであり予約ができる、データはリソースでありデプロイできる、ライセンスの使用はリソースでありメータリングできる）。この点で、実行やデータをおもに扱う実行管理サービスやデータサービスとは対照的である。リソース管理サービスは、OGSA 機能レベルにある。

CDDLML ワーキンググループでは、サービスのコンフィグレーション方法、サービスのグリッド上へのデプロイする方法、そのデプロイメントライフサイクルを管理（インスタンスの作成、イニシエーション、開始、停止、リスタート等）する方法、などの問題が扱われる。マネージャは、CDDLML が定義するインタフェースを使って、グリッド内のアプリケーションやその他の種類のサービスをコンフィギュア、デプロイ、再デプロイ（おそらく別のコンフィグレーションでリロケート）、停止するための機能を必要とする。インスタレーションとプロビジョニングは別の問題である。

メータリングサービスは、事実上、インフラストラクチャサービスである。リソース使用を記録、課金する場合にメータリングサービスを継続的に使用する必要があるため、マネージャは他の重要なサービスと同様にそのオペレーションを監視、制御しなければならない。会計、課金、支払いの各サービスは OGSA に含まれないが、メータリングサービスの機能上でビルドされる。

5.3.5.2 ギャップと残された問題

以下のギャップが見出された。

- ・プロビジョニングは、ホストやサービスからライセンス、帯域幅、データにいたるまで、あらゆる種類のリソースを対象とする必要がある。CDDLML-WG の作業は、現在サービスに重点を置いているが、他の種類のリソースをカバーするよう拡張すべきである。

残された問題点を以下に列記する。

- ・メータリング、情報サービス、WSDM の関連性について分析する必要がある。これにはリソースモデルも含まれるが、メータリングと情報サービスとではモデルが異なることがある。

5.3.6 自己管理サービス

5.3.6.1 解析

自己管理サービスは、ポリシーに沿い、さらに（あるいは）サービスレベルアグリーメント（SLA）に準拠する形で、IT システムをコンフィギュアし、修復し、最適化するものである。このサービスとこの機能を与えるメカニズムの定義はまだ始まったばかりであり、ギャップ解析ができる段階ではない。

自己管理サービスは、中央集中化されたモノリシックなサービスではなく、複数のレベルに分かれた一組のサービスになると思われる。たとえば、負荷が増加している中、データベース上で SLA に準拠するためには、マネージャはストレージを最適化することがある。最適化が十分でなければ、マネージャはより高いレベルでこのデータベースに 1 つあるいは複数のノードを追加するかもしれない。それでも十分でなければ、マネージャはさらに高いレベルで複数のサイト間でアクセスを分散させることができる。この階層構造的な性質により、マネージャはステータスを報告したりコマンドや SLA パラメタを受け取ったりするためのインタフェースが必要となる。

5.3.6.2 ギャップと残された問題

上述のように、ギャップ解析は現段階で行われていない。

5.3.7 セキュリティサービス

5.3.7.1 解析

OGSA を構成するセキュリティサービス（およびそのインタフェース）は、現在 OGSA-WG が定義を行っているところである。

認証や認可などのサービスは管理が必要であり、専用のマネージャビリティインタフェースが求められる。

5.3.7.2 ギャップと残された問題

現在のところモデルに関する議論が行われていないため、これが残された問題の 1 つであるといえる。しかし、CIM などの既存のモデルにおけるスキーマ（たとえばユーザとセキュリティのスキーマ、セキュリティ保護と管理のスキーマ）（およびその背景にあるマネージャビリティに関する知識）は、セキュリティのためのマネージャビリティにとって有用であり、利用することができるものである。とくに CIM には、アイデンティティや基本原則、クレデンシャル、役割、権限、認証と認可の規則などを記述するための基本モデルとオントロジーがある。

6. 結論

本文書では、グリッド、とくに OGSA に固有の管理問題を議論した。はじめに用語を定義し、グリッドに関連する管理の要件を解説した。つぎにグリッド管理に係る個々の

インタフェース、サービス、アクティビティなどについて議論した。その際、グリッド内での管理と、グリッドインフラストラクチャ自体の管理の両方を扱った。最後に OGSA におけるマネージャビリティの状態に関し、包括的なギャップ解析を行った。これは今後の作業のための基礎になるものと考えている。

6.1 ギャップのまとめ

ギャップ解析からは、以下のようにおもに 2 つのパターンが浮上している。

第 1 に、現在 OGSA におけるマネージャビリティの機能性は十分ではない。この機能性は、より柔軟性に富み自己管理できるシステムや管理の手間がかからないシステム、そしてグリッド技術により改善されると見られる属性を提供する上で重要なものであることから、その定義が必要になる。

第 2 に、OGSA は EMS のジョブやコンテナあるいは VO などの新しいエンティティを定義しており、これらのエンティティはリソースモデルを必要とする。リソースモデルは、たとえば既存のモデルを再利用するなどして、4 節のガイドラインに基づいて定義すべきである。

6.2 今後の作業

多くの OGSA 機能、その互いの関連性、OGSA が基礎を置く標準は、いずれも進化を続けているものである。今回の作業では、それらの現況をギャップ解析の基礎として使用した。そのため OGSA や関連する標準の改良や進化をこれからも追いかける必要がある。現状における今回の作業が、この改良や進化への手引きとなることを望む。

これまでに見つかっているギャップに関する解析、たとえばインタフェースやモデルの定義、既存の機能が十分であるかどうかの解析などは、各分野の専門グループが必要な知識を持っていることから、それぞれのグループが独自にあるいは共同で対処することにより適切に行うことができる。またその作業は、各機能の定義の重要な部分として行われるべきである。モデルを定義する上で必要な専門知識の一部は、他のグループ、たとえば CGS-WG や DMTF などから取得する必要があると考えられる。

7. セキュリティに関して

3.1 節でふれたように、セキュリティは管理の主要要件に含まれている。セキュリティは本文書で扱った多くの管理機能の 1 つである。

著者情報

編集：

編集者情報は原文のまま

寄稿：

寄稿者名は原文のまま

Hiro Kishimoto、 Bryan Murray、 Jay Unger、 Ravi Subramaniam、 Lawrence Flon、 Jeffrey Frey、 Bill Horn、 および OGSA-WG のメンバーの方々には、本文書の内容改善に役立つ議論をしていただき、ここに感謝の意を表する。

用語

第 2 節「定義」で、OGSA の管理用語について簡単な解説が与えられている。関連語句の詳しい定義については、OGSA 用語集[2]を参照のこと。略語については本文中で定義されている。

知的財産権について

本文書に記載された技術の実装や使用に関連すると考えられるいかなる知的財産権およびその他の権利についても、GGFは、その有効性や範囲に関して特定の立場をとるものではない。また、こうした権利下にあるいかなるライセンスについても、それが使用できるあるいは使用できない範囲について、特定の立場をとるものではない。さらに、これらのいかなる権利についても特定する努力をGGFが行うものではない。出版やライセンス保証のために用意した権利請求書、および、開発者や本仕様書の使用者が所有権を使用する上での一般的なライセンスや許可を取得する作業の結果については、GGF事務局から入手可能である。

関係者の方々には、本文書で薦められていることを実践する上で必要なあらゆる著作権、特許、特許利用、その他の所有権に対して注意を向けるよう GGF は希望する。情報は GGF 委員長までお寄せいただければ幸いである。

著作権情報

Copyright © Global Grid Forum (2003-2005). All Rights Reserved.

本文書およびその翻訳物は、複製し、他者に提供することができる。本文書に関してコメントや別の説明を与えている派生著作物、あるいは本文書の普及を助ける派生著作物についても、そのままあるいは部分的に、制約なしに作成、複製、発行、頒布が可能である。ただしそのような複製物や派生著作物については、上記の著作権情報と本段落に書かれていることを記載しなければならない。ただし、GGFや他の組織に対する著作権情報や参考文献を削除するなどといった本文書自体の変更は、いかなる形であっても禁止する。なお、グリッド提言 (Grid Recommendations) を開発する目的で変更が必要ならば、その限りで

はない。この場合、GGFドキュメントプロセスで決められた著作権に対する手続きを遵守する必要がある。また、英語以外の言語に翻訳する際に改変が必要な場合も、その限りではない。

上に与えた制限付き許諾は永続的なものであり、GGFあるいはその後続組織または委嘱組織が無効にすることはない。

本文書とそこに含まれる情報は保証されたものではない。グローバル・グリッド・フォーラムは、ここにおける情報の使用がいかなる権利をも侵害していないこと、市販性、特定目的との適合性に関するいかなる黙示保証をも侵害していないことを含む（ただし必ずしもこれらに限定されない）明示あるいは暗示の保証を拒否するものである。

参考文献

以下省略