5 ユーティリティサービスを実現するGridASP

伊藤　智 >> 産業技術総合研究所グリッド研究センター：satoshi.itoh@aist.go.jp

GridASPとは

グリッド技術が社会にもたらす最も大きな変革の一つは，情報サービスのユーティリティ化である．我々は，アプリケーションの実行をユーティリティサービスとして提供するビジネスモデル GridASP（Grid Application Service Provider）[1] を提案するとともに，GridASP を実現するためのソフトウェア GridASP Toolkit を開発している．

我々が提案するビジネスモデル GridASP は，**図 -1** に示すように，独立した 3 種類の事業者が連携することでアプリケーションの実行サービスを提供する．GridASP は，エンドユーザがサービスにアクセスするポータルを運営するポータル事業者，アプリケーションを用意しライセンスを管理するアプリケーション提供者，および，計算資源を運営しアプリケーションの実行リクエストをジョブとして受けて処理を実施するリソース提供者から構成される．

これら 3 種類の事業者が連携する GridASP モデルでは，自分の得意とする技能だけを持って事業を開始することが可能であり，かつ，専門性を活かすことによって事業の経営効率を高めることが可能である．たとえば，クラスタの運用が得意であれば，リソース提供者としてクラスタを大規模に運用することで 1 台あたりの運用コストを低く抑えることができる．リソース提供者は，複数のポータル事業者と契約することができるので，一方のポータルからアプリケーション実行の要求が少なくても，他のポータルからアプリケーション実行の要求を受けることでクラスタの資源を有効に運用することができる．逆に，ポータル事業者は，複数のリソース提供者と契約することができるので，一方のリソース提供者のクラスタで処理が実施できなくても，他のリソース提供者に処理を依頼することで，エンドユーザからのリクエストを受け付けることができる．また，エンドユーザに対するセキュリティとして，「誰が」と「何を」の両方を知る事業者をなくすことが可能である．ポータル事業者はエンドユーザが誰かを認証する必要があるが，リソース提供者は知る必要がない．リソース提供者のシステムでは，計算に必要なデータがなければならないが，ポータル事業者は見る必要がない（詳細は後述）．

GridASP では，プロバイダには投資を最小限に抑えつつ資源を効率よく運用することを可能とし，エンドユーザにはユーザ情報を秘匿しつつアプリケーションの実行サービスを享受することが可能となる．

従来技術や関連技術との比較

ここでは，前章までに述べられていたビジネスグリッドミドルウェアとの比較と，従来のユーティリティモデルとの比較について述べる．

GridASP が対象とするビジネスにおける業務は，大量のコンピューティング能力を必要とするような業務，たとえば，製造業における構造解析，流体解析，製薬業におけるドラッグスクリーニング，金融業におけるリスク

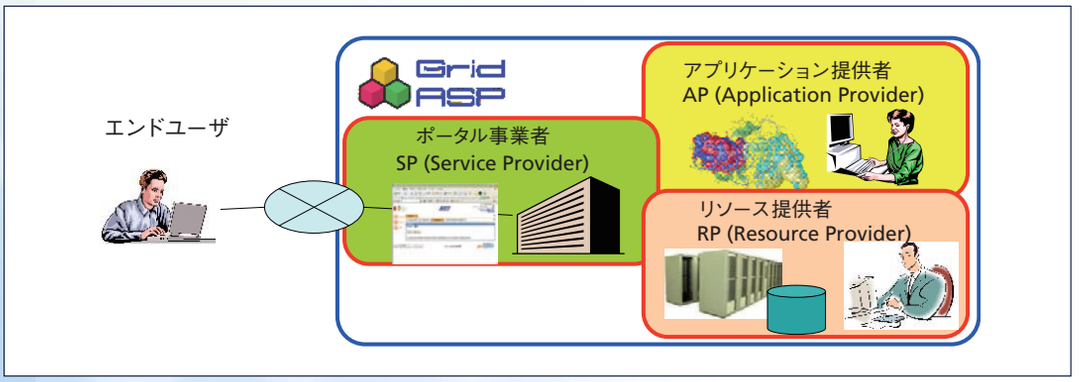

図 -1　GridASP の概念図

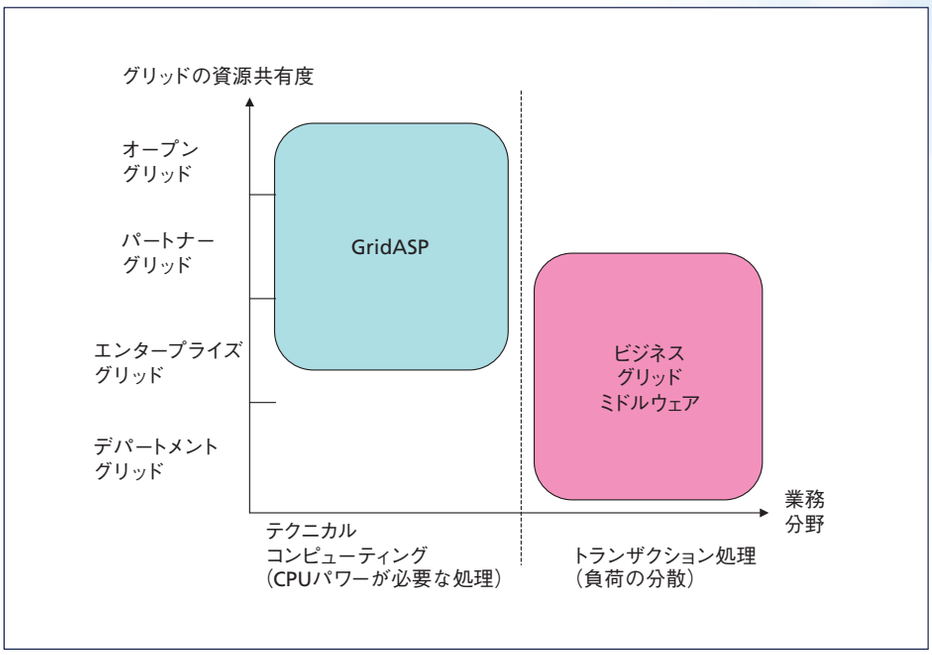

図 -2　ビジネスグリッドミドルウェアと GridASP の対象分野の違い

管理分析，などである．一方，ビジネスグリッドミドルウェアは，Web3 階層などで構成されるトランザクション処理を中心とした業務アプリケーションを対象としている．GridASP とビジネスグリッドミドルウェアとはグリッド技術を適用する応用分野が大きく異なり，互いに補完する立場にある．

また，グリッドにおける資源共有の度合いに応じた分類をしてみると，GridASP とビジネスグリッドミドルウェアの狙いの違いがより明確になる．1 カ所に集中し部門内のリソースを統合した「デパートメントグリッド」，企業内の複数の部門にまたがって最適化される「エンタープライズグリッド」，提携企業やグループ企業など複数の企業に分散したデータベースや計算資源を仮想化している「パートナーグリッド」，そして，強いガバナンスで連携する組織同士でリソースを共有する段階を越え，動的な組織間の連携や不特定多数の利用者にリソースの提供を行う「オープングリッド」と分類することができる．

このようにグリッドの資源共有度に応じた分類を考えてみると，**図 -2** に示すように，ビジネスグリッドミドルウェアは「エンタープライズグリッド」のシステムが主たる対象であり，ディザスタリカバリシステムなどいくつかのシナリオは「パートナーグリッド」の分類に入っている．一方，テクニカルコンピューティングの分野ではグリッドの適用はすでに「エンタープライズグリッド」として実用化が進んでいる．GridASP は，資源共有度を「エンタープライズグリッド」にとどまらず，「パートナーグリッド」や「オープングリッド」にまで拡大することを目指すものである．

ユーティリティとしてコンピューティング能力を提供するビジネスはこれまでも，RCS（Remote Computing Service）や ASP（Application Service Provider）というかたちで実施されてきた．しかし，20 年ほど前に行われていた RCS では，IT インフラの整備が十分でなかったことと，利用するための費用が高かったことから，大きな普及にはいたらなかった．その頃ネットワークを介して利用するためには高額な専用線を引かねばならず，ユーザがそのコストの支出を避けるためには，コンピュータシステムが設置された場所に赴いて端末操作を行う必要があった．また，ストレージが高価であったためシステムに多くのデータを置くことができず，ユーザが毎回データを持ち帰らねばならなかった．短期的であっても，アプリケーションのライセンスを買わねばならないため，利用者が高額の負担をしなければならなかった．さらに，コマンドラインによるバッチ処理が一般的で，使いやすい GUI などは整備されていなかった．このように，遠隔地のコンピュータを利用することは大変に不便な状況であり，大きな普及にはいたらなかった．現在では，高速なネットワークやストレージが安価に利用できるようになったほか，アプリケーションのライセンスを 1 年より短い単位で販売するベンダも登場してきた．また，ポータルなどの使いやすい GUI も登場し，20 年ほど前に立ちはだかっていた大きな障害はなくなっている．

その後登場した ASP は，アプリケーションの実行を，ユーティリティとして外部のリソースで実行するモデルである．エンドユーザは Web ブラウザを使ってネットワーク越しに ASP のポータルにアクセスし，実行した

いアプリケーションと入力データを指定して，処理を依頼する．ASP からの計算処理結果は，Web ブラウザからダウンロードして受け取る．利用した分の料金を支払えば，必要な時に必要なだけサービスを利用することができる．

ただし，ASP として広く普及・活用されているアプリケーションの分野は，グループウェア，メールシステムなど長期的かつ定常的に利用されるシステムがほとんどであり，テクニカルアプリケーションの分野についてはわずかである．従来の ASP モデルでテクニカルアプリケーションに対するユーティリティを実現する場合，次のような問題点があるため大規模な普及にはいたっていないと考えている．

プロバイダは，ポータル，コンピュータ，アプリケーション，すべての資源と環境を揃えなければサービス提供が始められず，ビジネス開始への障壁が高い．プロバイダが提供するアプリケーションに対してエンドユーザからのリクエストが少ないと，計算資源は遊んでしまう．逆に，計算資源が固定されるので処理可能なジョブ数は制限されている．そのため，エンドユーザからのリクエストが多いと，受けられない依頼が生じたり，実行を待たせたりすることになる．また，多くの製造業が解析データを外部の組織に知られることを避けているが，従来の ASP モデルでは，誰がどのような計算をしているかプロバイダにすべて見えてしまう．

前章で述べた GridASP のモデルは，これら従来の RCS や ASP が持つ問題点を解決することができる．

GridASPを実現するソフトウェアGridASP Toolkit

GridASP のビジネスモデルは，ポータル事業者，アプリケーション提供者，リソース提供者という 3 種類の事業者が連携して，エンドユーザにサービスを提供する．GridASP では，エンドユーザとポータル事業者の間に利用者と提供者という大きな界面が存在する．

エンドユーザに対するインタフェースとして，ユーザ端末で実行するアプリケーションプログラムから呼び出し可能な API（Application Programming Interface）を設ける方法や，Web サービスなどの標準的なインタフェースを利用することも可能である．しかし，これらの場合，ユーザ端末に利用システムを別途開発して導入しなければならないため，エンドユーザにとって GridASP 利用の障壁を高めることになる．そこで，我々は，一般的なブラウザがあればアクセス可能な Web ポータルを，エンドユーザへのインタフェースとして選択した．また，アプリケーション提供者やリソース提供者がシステムに対して操作を必要とする場合も，同じ Web ポータルへアクセスすることでオペレーションが行えるようにした．Web ポータルの場合は，エンドユーザが使い慣れた GUI から直接利用するなどの柔軟性は得られないが，GridASP のビジネスモデルを実現し，検証するための最初のステップとして採用した．

GridASP のシステムには，エンドユーザのリクエストを受けてジョブを実行する機能，複数のアプリケーションのワークフローを実行させる機能，エンドユーザの情報を隠蔽するための機能，アプリケーションをクラスタ上に配備するための機能などが必要である．我々は，このシステムを実現するためのソフトウェアの実装として GridASP Toolkit を開発した．**図 -3** に GridASP Toolkit の全体アーキテクチャを示す．上に述べた機能は，実装上の細かい機能コンポーネントをいくつか組み合わせて実現している．図中では，それらの対応を点線枠で示している．Tomcat および JetSpeed をポータルの中核とし，それらと連携するかたちで機能を実装した．以下，それら機能の要件とその実装方法の概要を述べる．

❖ ジョブ実行機能

エンドユーザは，ブラウザを利用してポータルにアクセスし，アプリケーションの実行をリクエストすることができる．エンドユーザからのリクエストに応じて，クラスタ上で実行される処理単位をジョブと呼ぶ．エンドユーザは，ポータルに登録済みのアプリケーションから，実行したいアプリケーションを選択し，入力データを指定することでジョブの実行に必要な情報を規定する．入力データは手元のユーザ端末にあるファイルをポータルにアップロードし，ジョブの実行終了後，結果をファイルとしてユーザ端末にダウンロードできる．

ポータルはジョブの内容を Globus[2] で使用している RSL（Resource Specification Language）を用いて記述し，WS-GRAM（Web Service Grid Resource Allocation and Management）によって，リソース提供者のクラスタに投入する．リソース提供者のクラスタでは，複数の計算ノードがいわゆるクラスタ管理ツールによって管理・制御されており，WS-GRAM との連携モジュールが RSL で記述されたジョブの情報を各クラスタ管理ツールの仕様に変更して投入する．ここで，クラスタ管理ツールとは，プラットフォームコンピューティング（株）の LSF，アルテアエンジニアリング（株）の PBS，サン・マイクロシステムズ（株）の SGE（Sun Grid Engine）などである．

また，ポータルは，エンドユーザが計算を実行するクラスタを指定しなくても，アプリケーションが実行可能なリソースの中から CPU 負荷の少ないクラスタを選択するブローカの機能を提供する．

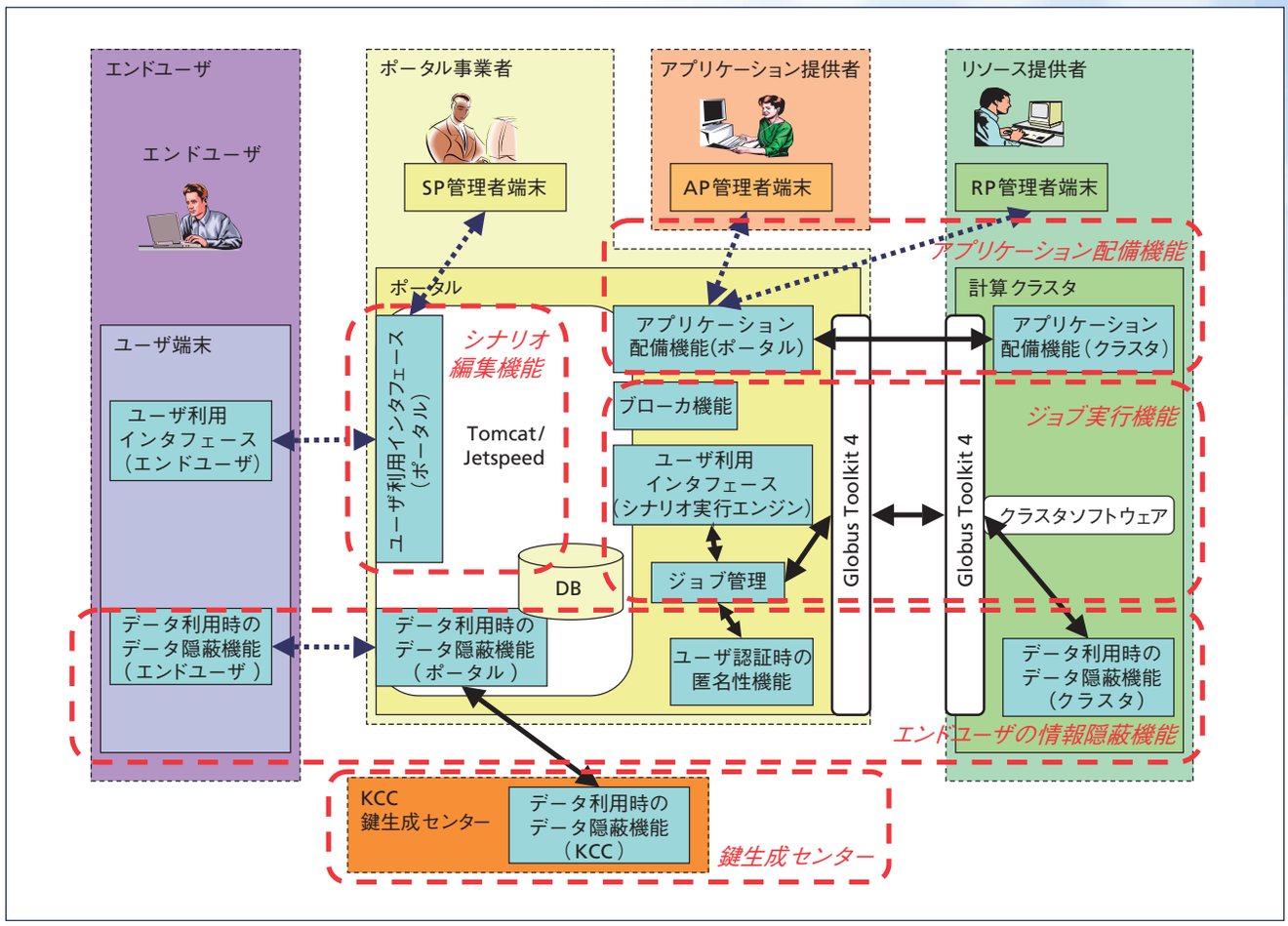

図 -3　GridASP Toolkit 全体アーキテクチャ

❖ シナリオ編集機能

企業における解析業務の多くが，単一のアプリケーションを実行するだけではなく，複数のアプリケーションを連続実行や，並列実行させるニーズを持っている．たとえば，自動車の設計では，メッシュ生成，構造解析，流体解析，衝突解析，結果の可視化などを必要に応じて実施したり，熱流体解析と構造解析など，複数のアプリケーションを連成実行したりする．また，設計パラメータの最適な解を得るために，パラメータの値を複数変更して同一計算を行わせる場合もあり，並列実行によって効率を上げることができる．

そこで，GridASP Toolkit では，アプリケーションのワークフローを定義できるシナリオ編集機能を提供している．シナリオ編集機能は，図 -3 に示すユーザ利用インタフェースの中に実装している．登録済みのアプリケーションを選択し，連続的な実行や並列的な実行を GUI で指定してシナリオを作成できる．作成したシナリオはテンプレートとして保存でき，パラメータを変えて繰り返し実行することが可能である．エンドユーザの投入したシナリオの順序に従ってアプリケーションの実行を制御する機構を，シナリオエンジンとしてポータル上で動かしている．

❖ エンドユーザの情報隠蔽機能

解析業務の内容は企業にとって機密情報となる場合が多い．「誰が」と「何を」の両方を知る事業者をなくす GridASP のモデルを，GridASP Toolkit では以下のように実現している（**図 -4** 参照）．

ポータル事業者はエンドユーザが誰であるかを認証し，ジョブの処理内容を受け取る．リソース提供者には誰の依頼であるかを伏せるため，匿名 ID を用いてジョブの実行を依頼する．リソース提供者は，ポータル事業者からのジョブであることを認証して実行する．リソース提供者は，「誰が」リクエストしたのかを知ることがない．図 -3 では，ポータル上のユーザ認証時の匿名性機能で実現している．

一方，ポータル事業者に対して「何を」計算しているのかを隠蔽するため，エンドユーザとリソース提供者の間で共通鍵を利用する．エンドユーザがファイルをアップロードする際に共通鍵によって暗号化され，ポータルを介してクラスタに運ばれた後，アプリケーションを実行する前に復号化される．アプリケーションの実行結果

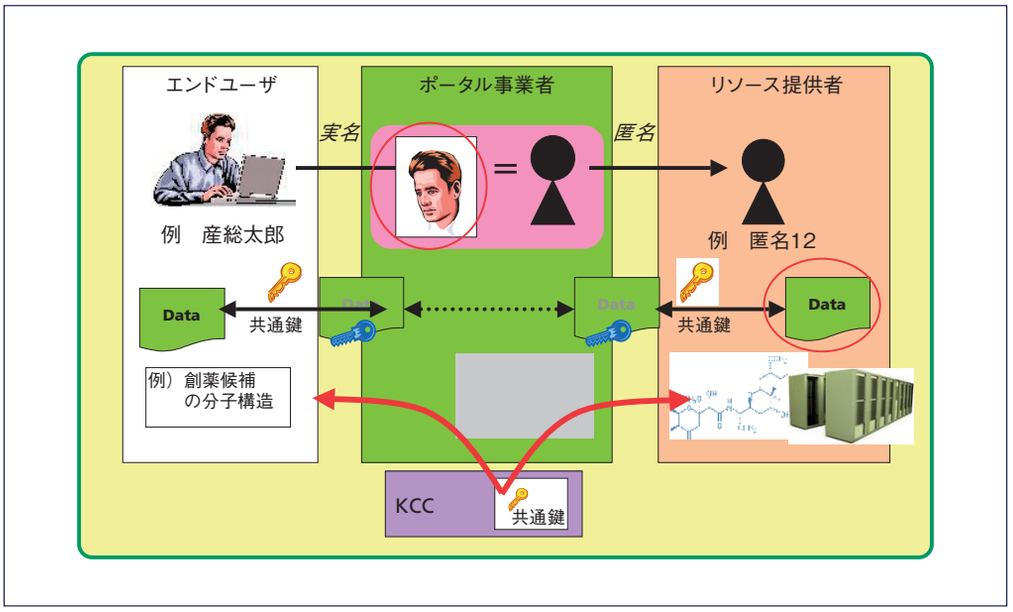

図 -4　エンドユーザの情報隠蔽

についても同様であり，ポータル事業者は暗号化されたファイルを見ることができない．図 -3 では，データ利用時のデータ隠蔽機能として実装しており，各プレーヤのシステム上に分散して動作する．

また，リソース提供者のクラスタでアプリケーションが終了し，出力結果が暗号化されてポータル事業者に渡された後，クラスタ上のユーザデータは自動的にすべて削除し，ユーザデータが露出する時間を最小限にしている．

❖ 鍵生成センター（KCC）

暗号化のための共通鍵はポータルが知っていてはいけないが，どのエンドユーザのジョブをどのリソース提供者のクラスタで実行したかを知っているのはポータルだけである．そこで，共通鍵を生成し，エンドユーザとクラスタにだけ配信するための仕組みとして，鍵生成センター（KCC : Key Creation Center）を開発した．

KCC は，ポータルとは独立のサーバである．ポータルは，鍵を配信したい相手（エンドユーザかクラスタ）の公開鍵と一緒に鍵の生成を KCC にリクエストする．KCC は共通鍵を生成し，配信先相手の公開鍵と KCC の公開鍵それぞれで暗号化した共通鍵をポータルに返す．暗号化された共通鍵は，対応する秘密鍵を持っている本人でなければ復号化することはできず，ポータルは共通鍵を復元することはできない．また，KCC の公開鍵で暗号化された共通鍵は，ポータルで管理し，同じ共通鍵を別の相手に配信する必要が出た際に，KCC に再度渡される．

KCC は，共通鍵の保管はせず，生成だけを行う．したがって，1 つの KCC が複数のポータル事業者に対して運用を行うことも可能である．

❖ アプリケーション配備機能

アプリケーションのことをよく知っているアプリケーション提供者と，計算機へのインストール権限を持っているリソース提供者は異なっている．また，どの計算機でどのアプリケーションを動かすかが変更になる可能性もある．そこで，GridASP Toolkit ではオンデマンドでアプリケーションが配備できる機能を実装している．

アプリケーション提供者は，アプリケーションをパッケージとしてポータルに登録する．このパッケージには，アプリケーション本体のほか，アプリケーションをインストール（アンインストール）するためのスクリプトや，実行時に必要な環境設定を行うスクリプトなどが含まれている．リソース提供者は，アプリケーションの配備が必要となった時点で，ポータルに登録されているアプリケーションパッケージをダウンロードしてインストールスクリプトを実行し，アプリケーションの配備を完了する．この工程を経て，アプリケーションはエンドユーザから利用可能となる．アプリケーション配備機能は，ポータルとクラスタ上に配置され，連携して動作する．

ビジネスグリッドミドルウェアとの連携

ビジネスグリッドミドルウェアは，Web アプリケーションを用いた業務システムに関して，負荷変動やシステム障害などに対して安定したシステム運用を実現する基盤を提供している．GridASP Toolkit がエンドユーザに提供するインタフェースはポータル事業者が運用する Web アプリケーションとしてのポータルである．そ

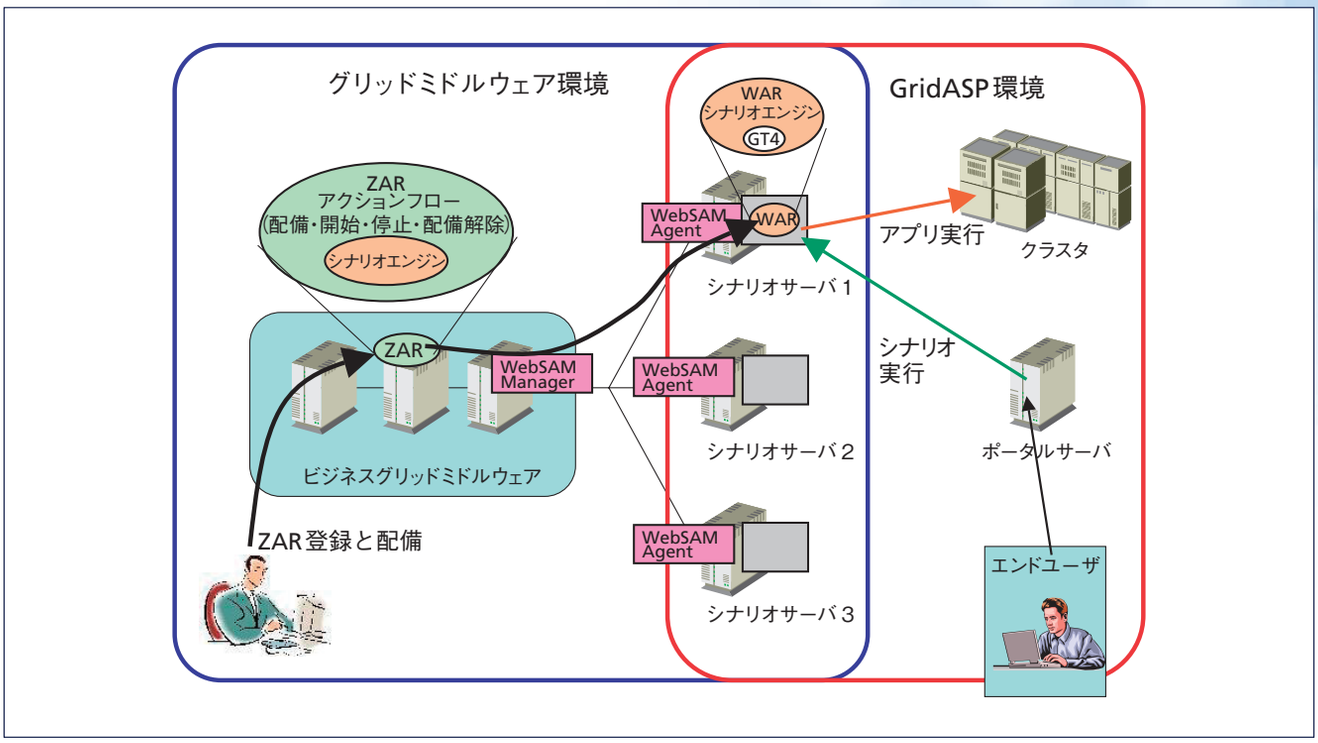

図 -5　ビジネスグリッドミドルウェアと GridASP の連携におけるシステム構成

こで，ポータル業務の一部をビジネスグリッドミドルウェアで管理されたシステム上で運用することによって，GridASP のサービスを安定して稼働することができる．

エンドユーザによるアプリケーション実行のリクエストは，先に述べたように，複数のジョブの実行順序がシナリオとして記述され，ポータル内に置かれたシナリオエンジンがこのシナリオに従ったジョブの実行を制御している．シナリオエンジンは，エンドユーザのリクエストごとにシナリオの開始から終了までの間，ポータル上のプロセスとして稼働する．したがって，エンドユーザの数およびリクエストされたジョブの数が増えると，シナリオエンジンの負荷が増大し，ポータルのレスポンスが低下する可能性がある．そこで，制御すべきシナリオが増えた場合のシナリオエンジンにおける負荷分散処理を，ビジネスグリッドミドルウェアの自律制御機能を活用して実現した．大量のジョブが投入された場合，ジョブ自身を処理するため CPU への負荷は増大するが，CPU への負荷はリソース提供者のクラスタ上で動作するクラスタ管理ツールによって負荷分散が実現されている．

図 -5 に示すように，GridASP のポータルの一部であるシナリオエンジンを，グリッドミドルウェアの基盤上で動かすシステム構成を考える．まず，シナリオエンジンをポータルから独立したシナリオサーバ上で稼働させ，分散実行を可能とした．続いて，シナリオエンジンをビジネスグリッドミドルウェアが扱えるように，J2EE（Java 2 Enterprise Edition）で開発した．ビジネスグリッドミドルウェアで取り扱えるように，ZAR（Zero administration ARchive）と呼ばれるファイルに，アプリケーションとしてのシナリオエンジンの運用情報を記述し，ビジネスグリッドミドルウェアに登録した．シナリオエンジンのソフトウェア自身は Java 仕様に準拠した WAR (Web ARchive) ファイルの形式として ZAR ファイルの中に含まれており，ビジネスグリッドミドルウェアによってシナリオサーバに配備・起動される．シナリオエンジンを稼働させるシナリオサーバには，NEC の運用管理ツール WebSAM が導入され，サーバの負荷を監視してビジネスグリッドミドルウェアに連絡する．シナリオサーバの負荷が増大した時にプールしたリソースからサーバを追加するような運用ポリシを ZAR ファイルに設定しておくことにより，負荷増大時に自動的にサーバが追加割り当てされ，ポータルシステムを安定して稼働させることができた．ビジネスグリッドミドルウェアの詳細については「2. ビジネスグリッド技術解説」を参照されたい．

GridASP 実証実験

企業におけるテクニカルコンピューティングの分野にユーティリティビジネスを導入する方法の一つとして GridASP を提案し，このビジネスモデルを実現するソフトウェア GridASP Toolkit を開発した．このソフトウェ

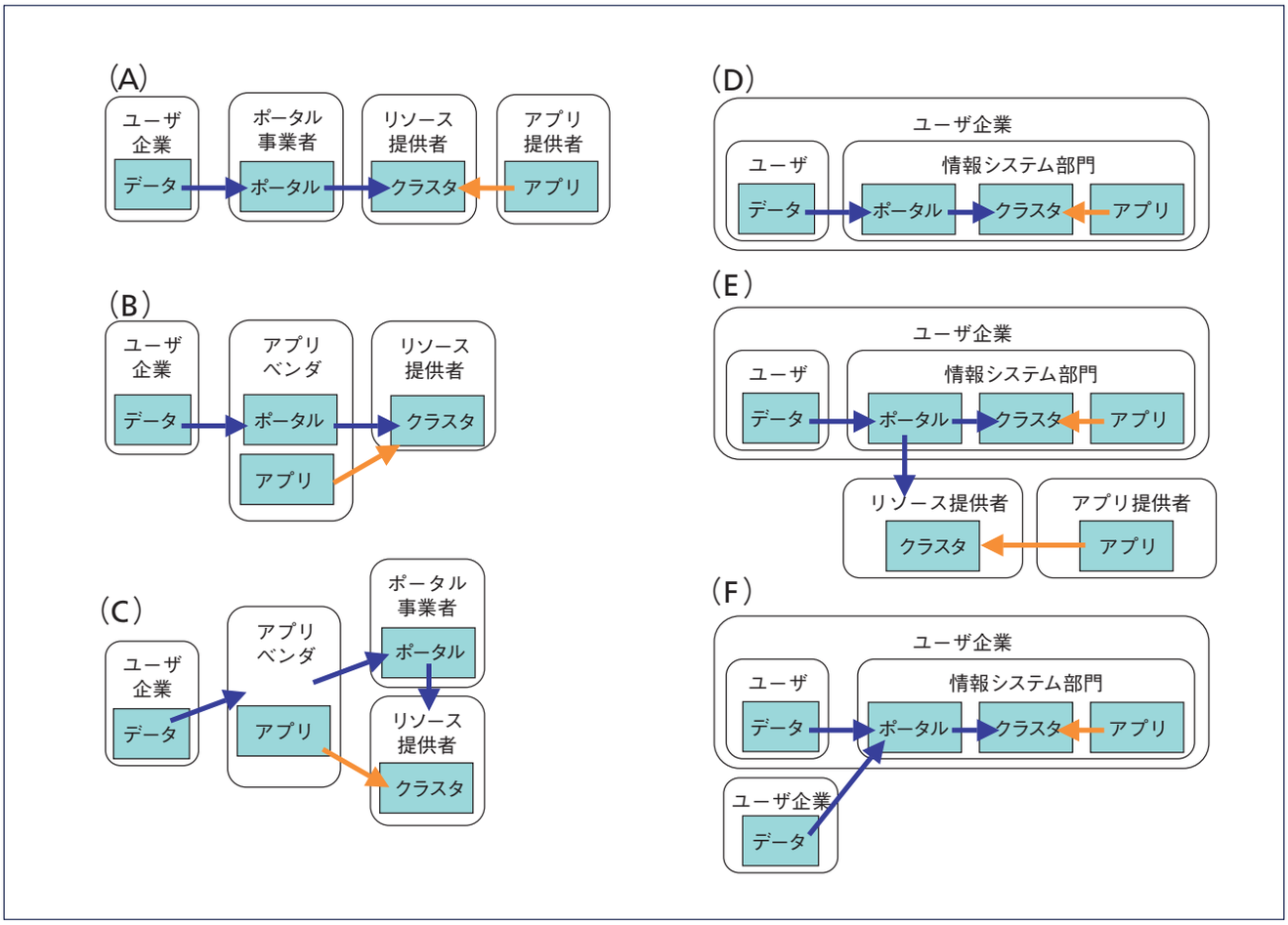

図 -6　GridASP Toolkit を活用した企業の連携形態（青矢印がジョブの流れ，橙矢印がアプリケーションの配備の流れを示す）

アは，**図 -6** に示すように，企業におけるさまざまな形態で活用することができる．以下に GridASP Toolkit を活用するいくつかの形態を列挙する．ただし，形態によっては，GridASP のモデルが持つすべての特徴を活かせるわけではない．

(A) 企業によるユーティリティサービスの提供とユーザ企業によるサービス利用
固定資産を持ちたくないユーザ企業が，アプリケーションの実行サービスを，外部の企業からユーティリティとして利用する．

(B) アプリベンダによるサービスポータル
アプリケーションプログラムを開発したアプリベンダが，ライセンスを売るだけでなく，ポータルを立ち上げてそのアプリケーションの実行サービスを提供する．

(C) アプリベンダが行う受託解析サービスのリソースを外部企業から利用
アプリベンダがユーザ企業から委託を受け，他の企業が提供するリソース上で解析を実施する．

(D) 社内コンピュータの統合
社内の情報システム部門が所有するコンピュータを統合するポータルを運用し，社内の利用者へ実行環境を提供する．

(E) 社内のコンピュータと社外のコンピュータの相補的利用
社内の情報システム部門が運用する同一環境ポータルから，社内のコンピュータだけでなく，他企業が運用するコンピュータも利用できる．

(F) 社内の利用で構築したポータルとリソースを，外部企業にも提供
社内に所有するコンピュータを，自社内で利用するだけでなく，余剰分を外部の企業（関連会社など）にも利用させる．

開発したソフトウェアが有効に機能し，上記のような形態に適合することを検証するため，企業と共同でいくつかの実証実験を実施した．

(A) として実施した実験の 1 つは，エンドユーザとしての三共（株）（以後三共），ポータル事業者としてのインテック・ウェブ・アンド・ゲノム・インフォマティクス（株）（以後 W&G）と共同で実施したものである．産総研はリソースプロバイダとして，つくばに所有する AIST スーパークラスタの一部（8 ノード 16CPU）を計

算リソースとして提供した．この実験では，アプリケーションとして，製薬企業で必要な化学分野の計算を行うソフトウェアを採用し，このアプリケーションを利用できるポータルを W&G が社内の富山研究所に構築した．三共は，平成 17 年 10 月末から翌年 1 月までの約 3 カ月間，約 200 件のジョブを実行した．実験期間中に年末年始など休日が多くあったものの，平均して 50%の CPU 稼働率であった．GridASP Toolkit で構築したポータルを利用して，ジョブの実行および結果の取得が問題なく実行できただけでなく，AIST スーパークラスタは一度も停止・再起動することなく安定して運用できた．これは，大規模なクラスタを安定して稼働できるようになってきたことのほかに，ポータルによるインタフェースがクラスタへの直接コマンド入力に比べてエンドユーザの操作ミスを招きにくいことが大きな理由として挙げられる．

（A）の形態のもう 1 つの取り組みとして，データセンタ事業者が提供するリソースを，ユーザ企業などに利用していただく実験も行っている．民間のデータセンタでは，外部ネットワークに接続しているサーバを限定し，ほとんどのサーバをプライベートアドレスの空間に閉じ込めている場合がある．今回，ニイウス（株），および NEC フィールディング（株）に協力いただき，それぞれのデータセンタに設置されたクラスタへ GridASP Toolkit を導入しシステムの構築・運用を行った．その結果，民間では一般的なこのようなネットワーク環境の下でも GridASP 向けにクラスタを構築・運用することが可能であることを確かめた．

（C）の形態を検証するため，自動車の設計などで重要な構造解析系のアプリケーションを取り上げ，実験を進めている．（株）日本総合研究所から構造解析ソフトウェア LS-DYNA を，フルーエント・アジアパシフィック（株）から熱流体解析ソフトウェア Fluent を提供いただくとともに，受託解析での利用を視野に入れて，両者がポータルを介したジョブ投入の実行と評価を開始した．この形態の検証は今後の課題である．

（D）の形態を検証するため，鹿島建設（株）（以降鹿島）と 4 月から共同研究を開始した．鹿島では，構造物設計のための構造解析や流体解析を行う計算機環境として，日立製作所の SR11000 と HA8000 クラスタ，および NEC の SX-8 を所有し，グリッド環境を構築している．鹿島との取り組みでは，これらのグリッド環境に GridASP Toolkit を導入し，同一のポータルから各計算機へのジョブ投入の実現を目指している．6 月までにポータルの構築を完了し，HA8000 を対象としたジョブの投入を実現した．今後その他の計算機に対する統合を目指すとともに，将来的には（E）の形態として，外部リソースへの透過的な接続も検討する．

なお，（B）と（F）の形態については，現時点ではまだ取り組むことができていないが，今後企業の協力を得て進めたい．

これらの実証実験を通して，GridASP Toolkit がさまざまな形態で利用可能であることが検証できた．しかし，GridASP を着実な事業として進めるためには，エンドユーザとプロバイダの間の契約の仕方，課金の設定方法，GridASP に適した商用アプリケーションのライセンスなど，社会的な課題が多く残されている．企業と連携した実験をさらに進めることで，これらの課題についても解決を目指す．

参考文献

1）http://www.gridasp.org/
2）http://www.globus.org/

（平成 18 年 7 月 28 日受付）