社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－HPC－107（1）
2006／7／31

OPAL Operation Provider を用いた WSRF における 生体高分子シミュレーションシステム

市川 昊平 † **伊達 進** † **Sriram Krishnan**‡ **Wilfred Li**‡ **下條 真司** §

グリッドに代表される高性能計算技術の発展により、コンピュータシミュレーション技術が飛躍的に発達している。生命科学の分野では、今日まで開発されてきた、多種多様なシミュレーションプログラムを組み合わせ、統合的に利用することにより、これまでとは異なる視点で生体システムの解明へアプローチする試みがすでに始まっている。本論文では、既存のプログラムを Wrapping 手法により Web サービス化する OPAL を用いた、生体高分子シミュレーションシステムについて述べる。その際、OPAL が Web サービス化するアプリケーションをさらには、グリッドサービスとして利用可能にする OPAL Operation Provider についても言及する。これにより、OPAL 利用に基づく生体高分子シミュレーションの有用性について検証する。

A WSRF Simulation System for Biomolecule by OPAL Operation Provider

Kohei Ichikawa† **Susumu Date**† **Sriram Krishnan**‡ **Wilfred Li**‡ **Shinji Shimojo**§

The development of High-performance computing technology has led to the advancement in computer simulation technology. In the arena of life science, experts have been attempting the combinational and integrated use of such computer simulation programs developed until today with a new perspective to the reveal of biological system. In this paper, we describe a simulation system for understanding biological system, characterized by the use of OPAL. The OPAL allows us to easily make the existent simulation programs a web service in a wrapping manner. Also, OPAL Operation Provider that allows the web service realized with OPAL to utilize Grid function is detailed. After that, we consider the usefulness of the system taking advantage of OPAL.

1　研究の背景と目的

近年の計算技術の発達は、さまざまな研究開発分野でのコンピュータシミュレーションの役割をより重要なものにしつつあり、今日では多様なシミュレーションプログラムが開発され利用可能となっている。生命科学の分野においても、生体の現象を観測する視点の違いにより、生体高分子から、細胞、組織の振る舞いのシミュレーションまで微視的な視点から巨視的な視点に基づくシミュレーション手法が存在する。しかし、これら個々に発達してきたシミュレーションプログラムはいずれも生体の現象を一視点に基づいて捉えたものに過ぎず、生体の統合的な理解のためにはこれらシミュレーション手法の組み合わせが必要かつ有効であると考えられている。個々のシミュレーションプログラムが成熟してきたことを背景として、今日ではこれら複数のシミュレーションを組み合わせることによる生体の統合的な理解への期待が高まっている [1]。

一方、情報科学の分野では計算機の高性能化と、それらを結ぶネットワーク技術の発達により、グリッドコンピューティングに代表される分散コンピューティングに関する研究が近年盛んに行われている [2]。グリッドコンピューティングの研究では、地理的に分散した計算機、ストレージ、アプリケーションをネットワークを通じて共有利用する環境の構築及びそのための技術開発が行われている。現在では、このグリッド環境上において、大規模なアプリケーションを動作させたり、分散するアプリケーションを統合する事によって、より洗練されたアプリケーションの実現に対する期待が高まっている。

生命科学の分野も例外ではなく、生体シミュレーションをグリッド環境上で統合しようとする動きがある。個々の組織で長年にわたって研究・開発されてきた生体シミュレーションを地理的に分散した状態で、ネットワークを通じて共有して組み合わせ、生体の現象をより統合的に解析することを目的とする [1]。

グリッド環境上で生体シミュレーションを統合するためには、各組織が研究・開発を行ってきた既存のシミュレーションプログラムを連携させるための手法の確立が必要となる。各組織で開発されているシミュレーションプログラムはグリッド環境での動作を考慮して開発されてきたものではなく、グリッド環境上で動作させるためには、分散通信を可能とする処理の追加やプログラムの修正等を行わなければならない。しかし、現状ではそのような手法は確立されておらず、各組織、各プログラムごとに様々な試行錯誤を行っているのが現状である。

本研究では既存のシミュレーションプログラムをグリッド環境上で連携させるための手法の確立を目的とする。現在、われわれは生体高分子の運動をシミュレートするために、量子力学（QM: Quantum Mechanics）に基づく電子状態シミュレーションと古典力学に基づく分子動力学（MM: Molecular Dynamics Mechanics）シミュレーションを組み合わせた QM/MM 連成シミュレーションシステムの構築を行っている。本論文では、既存のアプリケーションのグリッド化に必要な要件をわれわれの事例に基づき明確化し、シミュレーションプログラム

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
‡ San Diego Supercomputer Center
§ 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University

の連携手法の提案を行う。次節でより詳しく述べるが、QM/MM 連成シミュレーションはプログラムの異質性、希少な専用ハードウェア装置の分散環境共有、プログラムの組み合わせの複雑さなどの理由からグリッド化に際して様々な問題があり、QM/MM 連成シミュレーションを通してグリッド化の一般的な手法を探ることは有意義であるといえる。また、提案手法に従って実装した生体高分子シミュレーションシステムに関して、今後の課題や応用について検討する。

以下、2 節ではわれわれが現在取り組んでいる生体高分子シミュレーションの特徴とグリッド化における課題を述べる。3 節では現在までに行われてきたグリッド化手法に関するアプローチについて述べ、既存手法の問題点を指摘すると共に、本研究の提案手法について述べる。4 節では開発した OPAL Operation Provider について述べ、5 節では OPAL Operation Provider を用いたシミュレーションプログラムの実装について解説し、6 節では提案手法の有用性に関して考察を行う。そして、7 節で本論文のまとめを行い、今後の課題及び応用について考察する。

2 QM/MM 連成シミュレーションの特徴とグリッド化の課題

生体高分子シミュレーションは生体内の蛋白質などの高分子の時系列的な運動をシミュレートし、その化学的な性質を解明することを目的としている。分子シミュレーションでは分子間に働く電気的性質による分子間力を基に 0.1〜数 fsec といったタイムステップで計算し、これを数千〜数万ステップ繰り返すことによって時系列的な分子の運動を得る。

前節でも簡単に述べたが、われわれは生体高分子の運動をシミュレートするために、2 種類のシミュレーション手法を用いる。一つは古典力学に基づき経験的に求められた分子間力を用いて、個々の分子の運動を表すニュートン方程式を解く分子動力学（MM）シミュレーションである。MM シミュレーションは電子が関わる化学反応等の再現性は低いが、巨大な系の動的な運動をシミュレートするのには向いている。もう一方の手法は量子力学（QM）に基づき経験的な値を使わず、シュレディンガー方程式を近似的に解くことによって、分子間力を求める電子状態シミュレーションである。QM シミュレーションは電子が関わる化学反応を良く再現でき、精度良く分子の運動をシミュレートすることができる。これらを組み合わせ、精度が求められる部分には QM を、その他の領域に関しては MM を適用することによって、巨大な系のシミュレーションを現実的な時間内で行うことが可能となる。

これら QM と MM のシミュレーションの統合にあたっては、以下のような課題がある。

- シミュレーションプログラムの異質性
  個々のシミュレーションプログラムはこれまで異なる組織で長年に渡って開発されてきたものであり、全くの異質のものである。そのため、グリッド化の課程においてプログラムインタフェースを揃える等の処置が必要である。また、現在も個々の組織で活発に研究・開発が続けられており、単なる一過性の処置ではなく、容易なグリッド化の手法を確立する必要があると考えられる。
- 希少なハードウェア装置の共有利用
  古典力学をベースにした分子動力学分野は成熟しており、現在では分子動力学計算の中心的な計算である個々の粒子間の相互作用計算を行う MDGRAPE[3] という専用ハードウェアが開発されている。この MDGRAPE は分子動力学シミュレーションに関しては汎用 CPU の数十倍の性能を達成するが、維持・管理している組織は少なく、ネットワークを介した共用利用が望まれている。
- プログラムの組み合わせの複雑さ
  QM と MM シミュレーションはそのシミュレーションの計算途中で 1 タイムステップごとに原子の座標やそれぞれの原子に加わる力のベクトル等のデータを交換するが、QM と MM の組み合わせ方法は一通りだけではない。QM シミュレーションはアルゴリズムの違いにより複数通りの手法が存在し、それらを切り替えて使用したり、それぞれの結果を合成して用いたり、その組み合わせは方法は複雑である。このようなアプリケーションに固有の要件にも対応可能なグリッド化手法を考察する必要がある。

これらの課題は QM/MM 連成シミュレーションに限らず、既存のアプリケーションをグリッド環境上で統合利用する際に課題となりうる項目であると考えられる。そのため、QM/MM 連成シミュレーションのグリッド化を通してグリッド環境における連携手法を探ることは、一般的なアプリケーションのグリッド化手法の確立にも応用可能と考えられる。

3 シミュレーションプログラムのグリッド化のアプローチ

本節では、前節で取りあげた課題を解決する手段として現在一般的に行われているアプリケーションのグリッド化の手法を紹介すると共に、その手法の問題点を挙げる。そして、その問題を解決する手段として本研究が提案するアプローチに関して説明する。

3.1 グリッド化の一般的なアプローチ

分散するアプリケーションを統合利用する方法として現在一般的に行われている手法は、アプリケーションを構成する各プログラムをコンポーネント化し、そして何らかのコンポーネントフレームワーク上でそれらを統合する手法である。コンポーネントとは公開されたインタフェースの定義とその実装からなり、コンポーネントの利用者は実装の詳細までは知る必要はなく、公開されているインタフェースを通じてそのコンポーネントの機能を利用することが可能となる。代表的なものとして、Open Grid Service Architecture（OGSA）[4] や Common

Component Architecture（CCA）[5] があり、サービス指向アーキテクチャやコンポーネント指向アーキテクチャとして知られている。このコンポーネント指向アーキテクチャに従って、プログラムをコンポーネント化することによって、前節で挙げたような課題は以下のように解決されると考えられる。(1) コンポーネントは個々のプログラムの異質性を隠蔽し、(2) プログラム同士が密接に依存し合うことを避けることによってプログラム同士の関係を疎結合に保ち、コンポーネント同士の組み合わせを容易に変更することが可能となる。(3) また、分散コンポーネントフレームワークを利用することによって、リモートのリソースを透過的に利用することが可能となる。グリッド技術の標準化に取り組む Global Grid Forum（GGF）で議論されている OGSA は Web サービスの技術をコンポーネントフレームワークの基礎技術として利用し、様々なアプリケーションをサービスとして実装することによってグリッド化を図っている。

コンポーネント化によるアプリケーションのグリッド化が一般的になると、アプリケーションのグリッド化の問題は如何にしてアプリケーションをコンポーネント化するかというコンポーネント化手法の確立の問題に置き換えられる。

3.2 既存アプリケーションのコンポーネント化手法

既存のアプリケーションのコンポーネント化手法には (1) 個々のアプリケーション自体を書き換えて分散通信等の処理を加えてコンポーネント化する方法と (2) アプリケーションには変更を加えず、コンポーネント内でアプリケーションそのものを実行し、入出力・実行制御等を行う Wrapping 手法がある。前者は、コンポーネントの設計者によって自由に分散通信が設計できるが、開発工数が大きく、アプリケーションの実装を熟知していないと実装は不可能である。また、アプリケーションの更新が活発な場合は、何度も修正をして追従していく必要があり、現実的ではない。後者は既存アプリケーションに対する変更が無く、アプリケーションの実装への依存が低いが、アプリケーションの個々の特殊な要件には対応が難しい。ただ、コンポーネント化手法の確立という意味では後者の方が現実的であり、現在いくつかの Wrapping コンポーネントの実装が存在する。

以下に既存の Wrapping コンポーネントの例をいくつか挙げる。

- OPAL
  Web サービスによって構築された Wrapping サービス [6]
- Application Manager
  CCA の一実装である XCAT で用いられている既存アプリケーションの Wrapping コンポーネント [7]
- GAP Service
  In-VIGO システムフレームワークで用いられている Wrapping コンポーネント [8]

これらの Wrapping コンポーネントに共通することは、既存アプリケーションの入出力、実行方法等に関するデータ（メタデータ）を指定することによって、アプリケーションを Wrapping するコンポーネントを生成することである。いずれも入力データを用意して、アプリケーションを起動し、出力データを取得するという構成である。

3.3 既存手法の問題点

前節でも少し述べたが、Wrapping によるコンポーネント化手法は以下の点で問題があると考えられる。

- 単純なインタフェース
  アプリケーションを起動し、その終了を待って、出力を得るという単純なユースケースしか想定していない。グリッドを利用するアプリケーションの中には長時間の計算の途中で他のコンポーネントとインタラクションを取りたいアプリケーションが存在するが、そのようなユースケースは想定していない。
- 生成されたコンポーネントの拡張が不可能
  アプリケーションのメタデータを与えて生成されたコンポーネントは、それ自体で閉じた実装になっており、他の機能を追加できる余地は用意されていない。
- グリッドの機能を十分に活かせない
  Wrapping コンポーネントの想定するユースケース内でしかグリッドの機能を利用できないため、グリッドのセキュリティ機能、非同期通信機能等を自由に利用することはできない。

QM/MM 連成シミュレーションでは一つのシミュレーションの間にタイムステップごとにそれぞれのプログラム間でデータを交換したり、交換のための同期を取ったりする必要があるが、これらの要件は既存の Wrapping コンポーネントではサポートできないのが現状である。

3.4 提案手法

本研究では WSRF を実装する Globus Toolkit4（以下、GT4）の基本的な実装モデルである Operation Provider として Wrapping コンポーネントを構築し、それを用いて既存アプリケーションを WSRF 上のサービスとしてコンポーネント化する手法を提案する。

Operation Provider とは、任意のサービスに Operation を提供するサービスである。この場合の Operation とは WSDL 内に定義する Web サービスが持つ Operation を示す。Operation Provider を利用する設定を行うと、サービスの実装に変更を加えることなく、Operation Provider が提供する Operation をサービスに追加することができる。Operation Provider を利用する場合、サービスの WSDL 内で Operation Provider のインタフェースを継承する設定を記述し、WSDD 内で Operation Provider の実装を指定するが、Operation Provider のモデルは正にオブジェクト指向における継承のようなモデルである。実際には GT4 の実装では、WSDL 内で継承設定されたインタフェースを全て読みとり、全ての Operation を一つの WSDL にまとめる作業（flatten）を行い、エンド

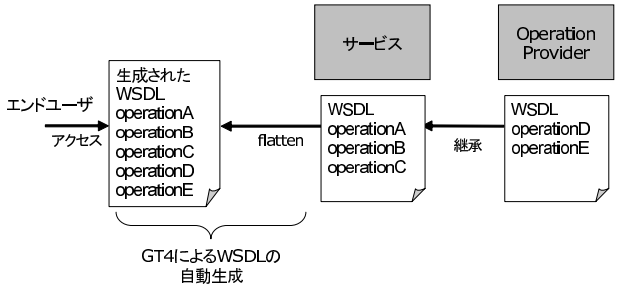

図 1　Operation Provider の概要

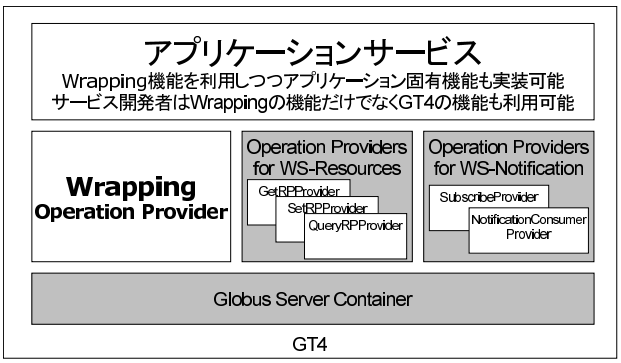

図 2　Wrapping Operation Provider の位置づけ

ユーザには一つの大きな WSDL が見えるようにしている。そして、エンドユーザが Operation を呼び出すと、それに対応した Operation Provider の実装に処理を割り振ることによって Operation Provider の機能を実現している。図 1 に概要を示す。

Operation Provider として Wrapping コンポーネントを実装することによって、Wrapping コンポーネントはそれ自体で閉じた実装にはならず、利用するサービス側で自由に機能を追加することが可能となる。アプリケーションのサービス開発者の立場から見ると、アプリケーションの Wrapping 処理に関しては Operation Provider に任せ、自身はアプリケーションに特化した処理の記述に集中することができる。また、GT4 はサービスの内部状態（WS-Resources）へのアクセスや非同期通信（WS-Notification）等も Operation Provider として実装しており、サービス開発者は同様の手順でそれらの機能を自身が開発するサービスに取り入れることが可能となる。したがって、Operation Provider として Wrapping コンポーネントを実装することによって先に挙げた既存の Wrapping コンポーネントの問題点は解決されると考えられる。図 2 に Operation Provider としての Wrapping コンポーネントの位置づけを示す。

4　OPAL Operation Provider

本研究では前述した既存の Wrapping コンポーネントの一つである OPAL を Operation Provider（以下、OPAL OP）として実装した。Operation Provider はその実装自体も Web サービスであり、Web サービスとして構築されている OPAL は Operation Provider に容易に移植が可能であったためである。

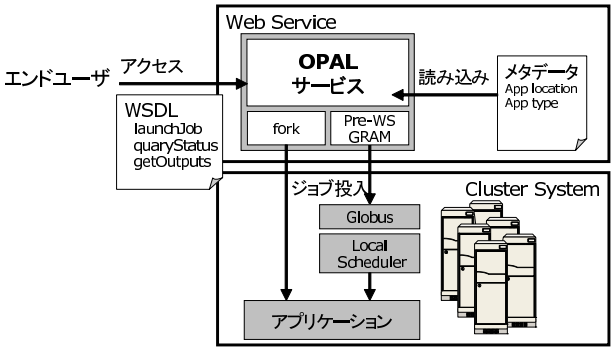

図 3　OPAL の概要

4.1　OPAL

OPAL は San Diego Supercomputer Center（SDSC）のプロジェクトである National Biomedical Computation Resource（NBCR）で開発された既存アプリケーションの Wrapping Web サービスである。アプリケーションのバイナリの場所、アプリケーションの種類（MPI か、MPI でないか）、プロセスの生成方法（Globus を使うか、使わないか）等を記述したアプリケーションメタデータを記述することによって、OPAL サービスがその情報を読みとりアプリケーションを適切に起動し、ジョブの状態等を管理する。

OPAL サービスは主に以下の機能を有する。図 3 に概要を示す。

- 入力ファイルのステージング
  入力ファイルは SOAP メッセージに base64 エンコードして埋め込むことによって、アプリケーションの作業ディレクトリにステージングされる。
- アプリケーションの実行
  ジョブ起動要求の SOAP メッセージにアプリケーションのコマンドライン引数を埋め込むことができ、アプリケーションは Fork によってプロセスが生成されるか、もしくは Globus Gatekeeper を通して Local Scheduler にジョブとして投入される。
- 出力ファイルの取得
  出力ファイルは Tomcat を通して http で取得できるほか、SOAP メッセージに base64 エンコードして取得可能である。

また、オプショナルな機能として、GSI 認証、サービスの状態の SQL データベースへの保存等の機能がある。非常にシンプルな構成であるが、コマンドラインベースのアプリケーションを Wrap するという目的は十分に果たしており、Operation Provider として利用するには Wrapping 以外の余計な機能が付いていないため、適している。

4.2 OPAL Operation Provider の実装

Web サービスならば、ほぼそのままの実装を Operation Provider に移植できるため、OPAL Operation Provider の構築は容易であった。OPAL を Operation Provider として利用するための変更点は以下の 2 点であった。

- Web サービスと GT4 が使用する WSDL のわずかな仕様の違いのための修正
- Tomcat と AXIS 上での動作を前提としている OPAL の実装を Globus のコンテナでも実行可能とするための修正

上述の修正を行えば、OPAL を Operation Provider として利用できる。ただし、OPAL OP は OPAL とは違って単体では利用できず、アプリケーション開発者自身が作成するサービス内で OPAL OP のインタフェースを継承する設定を行い、WSDD にて OPAL OP の実装を指定する必要がある。そのため、OPAL の使用手順は (1)OPAL の設定（アプリケーションメタデータ等）、(2) ビルド・デプロイ、(3) 実行であったのに対し、OPAL OP は (1)OPAL の設定、(2) サービスを新規作成し、OPAL OP を継承、(3) ビルド・デプロイ、(4) 実行となり、一つ作業項目が増えることになる。そこで、われわれは OPAL のビルドに利用している ant のビルドスクリプトに修正を加え、OPAL OP の利用するサービスの雛形を自動生成するようにした。これにより、OPAL OP の利用方法は元々の OPAL とほぼ同様に、(1) 設定、(2) ビルド・デプロイ（ビルドの中でサービスの雛形自動生成）、(3) 実行という手順で済むようになった。OPAL の機能以外にアプリケーションに固有の要件を満たす実装を加えたい場合は、自動生成されたサービスに実装を加え、ビルドし直せばよい。

この OPAL OP によって既存アプリケーションを容易に Wrapping し、さらに個々のアプリケーションに固有の要件にも対応可能なコンポーネント化の手法が確立されたと言える。

5 QM/MM 連成シミュレーションの実装

2 節で述べたように、われわれの QM/MM 連成シミュレーションはアプリケーションの実行中に 1 ステップごとにデータをプログラム間で転送し合う必要がある。そのため、単純な Wrapping コンポーネントの利用だけでは問題は解決できず、OPAL OP のような拡張可能なコンポーネント化の仕組みが必要である。

QM/MM 連成シミュレーションの実装は 3 段階に分けて進めた。また、図 4 に OPAL OP によって実装された QM/MM 連成シミュレーションの概要を示す。

5.1 Adapter プログラムの構築

まず、最初に行ったのは、QM、MM それぞれのプログラムの複雑性を隠蔽し、2 つのプログラムが同期を取れるようにするために Adapter プログラムを構築した。QM/MM 連成シミュレーションを構成する QM、MM プログラムのそれぞれ自身も複数のプログラムから構成さ

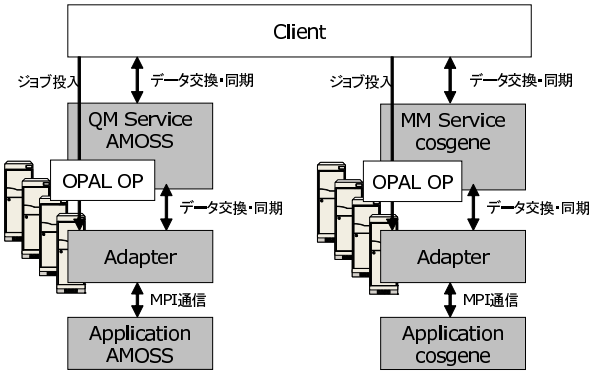

図 4 OPAL OP による QM/MM 連成シミュレーションの概要

れており、MPI-2 の動的プロセス生成機能（spawn）を用いて、実行時に動的に並列計算環境を構築するプログラムである。このような動的な環境をローカルスケジューラから扱うことは困難である。そこで、Adapter はこの動的な MPI プログラムを隠蔽し、シミュレーションプログラムを単純な MPI プログラムとしてみせる役割を果たす。また、QM、MM 間でデータ交換のための同期を取るため、この Adapter はファイルロック方式による同期を行っている。つまり、各プログラムで計算が始まる時に、ファイルを生成し、終わった時に削除する。そして、ファイルの有無に応じて、現在のステップがまだ計算中か、もう終了したかを判断し、データ転送を適切なタイミングで行う。

5.2 OPAL OP によるコンポーネント化

次に OPAL OP によってプログラムを Wrapping した。この課程は単純に OPAL の設定ファイル（アプリケーションのメタデータ）に実行バイナリとして、前述の Adapter を指定し、MPI ジョブの実行方法の記述を行うのみである。後は ant ツールによりサービスが生成され、ビルドされる。

5.3 データ入出力と同期処理の追加

最後に生成されたサービスに対し、1 ステップごとにデータを交換するためのデータ入出力のインタフェースと、Adapter プログラムが処理するロックファイルの有無を確認し、同期を取る処理を追加することによって、それぞれのプログラムの連携を実現させた。

6 OPAL Operation Provider の有用性の考察

OPAL OP の有用性は大きく分けて以下の 2 つあると考えられる。

6.1 拡張可能な Wrapping サービスの提供

OPAL OP はアプリケーション開発者によってアプリケーション固有の要件を処理するための拡張を可能としている点が有用であると考えられる。元の OPAL 等の Wrapping ツールはそれ自体で閉じた完成されたコン

ポーネントであり、アプリケーション開発者によって拡張できる余地が無かった。しかし、OPAL OP は OPAL の機能を GT4 の Operation Provider として構築し、任意のサービスからライブラリのように利用可能なツールとして Wrapping 機能を実現することで、アプリケーション開発者に対しサービスの拡張の機会を与えている。OPAL OP によってアプリケーション開発者はジョブの投入やその制御といった処理は OPAL OP に任せて、アプリケーション固有の要件の処理に集中することが可能となる。グリッド上で動作するアプリケーションの実行は長時間に及ぶ場合が多く、その実行中に細かにプログラムの実行を制御したい要望は強くあり、OPAL OP が提供する拡張可能性はそのようなアプリケーションに対し有用だと考える。

6.2 他の Operation Provider との併用利用が可能

OPAL OP は GT4 が定義する Operation Provider として実装したことによって、アプリケーション開発者は OPAL OP だけでなく GT4 の Operation Provider も同様に利用可能となる。GT4 は状態保持機能や非同期通信等も Operation Provider として実装しており、それらを利用することによってより洗練されたアプリケーションを記述することも可能となる。現在の QM/MM 連成シミュレーションは 1 ステップごとの同期に古くから利用されているファイルロック形式を利用しているが、これを Web サービスの状態保持機能と非同期通信に置き換えることによって、サービス同士で自律的に同期を取るようなプログラムも書くことが可能である。

7 まとめと今後の課題

本研究ではわれわれの事例である QM/MM 連成シミュレーションを通して、分散する既存アプリケーションの連携、統合方法の課題を洗い出し、課題を解決する手段として Operation Provider として既存アプリケーションの Wrapping ツールを構築するアプローチを提案した。そして、Wrapping サービスの一つである OPAL を Operation Provider として構築し直し、OPAL OP を実装した。また、OPAL OP を用いて QM/MM 連成シミュレーションを構築し、その有用性について考察を行った。

今後の課題としては、OPAL OP が現在内部で独自に管理しているジョブのステータス（ジョブの実行中、終了等）を Web サービスの状態保持機能を利用して管理し、サービス間の非同期通信を用いてその情報を取得可能にする拡張が考えられる。これによって、アプリケーション開発者がジョブのステータスを逐次ペンディングし処理待ちをする必要は無くなり、より効率的なアプリケーションを記述できるようになると期待される。そのためには Operation Provider が Web サービスの状態保持機能を利用する必要があるが、GT4 の Operation Provider の仕様上の問題のために状態を保持する Resource の実装を Operation Provider 側で提供することは困難であり、解決しなければならない問題として認識している。

謝辞

AMOSS サービスの構築において、ご助言頂いた NEC の高田俊和博士、中田一人氏、佐久間俊広博士に感謝する。また、cosgene サービスの構築において、ご助言頂いた大阪大学蛋白質研究所の中村春木教授、米澤康滋助教授、日立製作所の何希倫博士、黒澤隆氏に感謝する。本研究は文部科学省科学技術振興費主要 5 分野の研究開発委託事業の IT プログラム「スーパーコンピュータネットワークの構築」の一環として実施された研究成果の一部である。

参考文献

[1] Haruki Nakamura, Susumu Date, Hideo Matsuda, and Shinji Shimojo. A challenge towards next-generation research infrastructure for advanced life science. *New Generation Computing*, Vol. 22, No. 2, February 2004.

[2] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal Supercomputer Applications*, Vol. 15, No. 3, pp. 200–222, 2001.

[3] Yuichi Yatsuyanagi, Toshikazu Ebisuzaki, Yasuhito Kiwamoto, Tadatsugu Hatori, and Tomokazu Kato. Simulations of diocotron instability using a special-purpose computer, MDGRAPE-2. *Phys. Plasmas*, Vol. 10, pp. 3188–3195, 2003.

[4] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. In *Open Grid Service Infrastructure WG*. Global Grid Forum, June 2002.

[5] David E. Bernholdt et al. A component architecture for high-performance scientific computing. *International Journal of High Performance Computing Applications*, Vol. 20, No. 2, pp. 163–202, 2006.

[6] Sriram Krishnan, Brent Stearn, Karan Bhatia, Kim K. Baldridge, Wilfred Li, and Peter Arzberger. Opal: Simple web services wrappers for scientific applications. In *San Diego Supercomputer Center Technical Report TR-2006-5*, Feb 2006.

[7] Dennis Gannon, Sriram Krishnan, Liang Fang, Gopi Kandaswamy, Yogesh Simmhan, and Aleksander Slominski. On building parallel and grid applications: Component technology and distributed services. In *2nd International Workshop on Challenges of Large Applications in Distributed Environments (CLADE 2004)*, June 2004.

[8] Vivekananthan Sanjeepan, Andrea M. Matsunaga, Liping Zhu, Herman Lam, and Jose A. B. Fortes. A service-oriented, scalable approach to grid-enabling of legacy scientific applications. In *The 2005 IEEE International Conference on Web Services (ICWS 2005)*, pp. 553–560, July 2005.