機能分散型生体高分子シミュレーション実行環境

市川 昊平 † **伊達 進** † **中田 一人** ‡ **米澤 康滋** §
Rossen Apostolov§ **中村 春木** § **下條 真司** ‡‡

生体シミュレーションは微視的な視点から巨視的な視点まで空間軸と時間軸の様々なスケールにおけるシミュレーションの連携（マルチスケールシミュレーション）として認識されるつつあり、今日まで別々の学問として認識されてきたシミュレーション手法が連携することによる計算化学領域のパラダイムシフトに対する期待が高まっている。われわれはシミュレーションは時間軸を適当な間隔で区切り、各タイムステップにおける計算を繰り返すことによって時間的に連続なデータを得るものであると考察する。さらに、その考察に基づき、解析、監視・可視化、データ管理の一連の流れのモデルを提案する。本論文では、複数のシミュレーションプログラムを OGSA 上で連携させる機能分散型の QM/MM シミュレーション環境の構築手法について論じる。

A Distributed Function Environment for Simulation of Biomolecule

Kohei Ichikawa† Susumu Date† Kazuto Nakata‡ Yasushige Yonezawa§
Rossen Apostolov§ Haruki Nakamura§ Shinji Shimojo‡‡

In these days, biological simulation has been recognized as combinations with simulations at various scales from microscopic to macroscopic, raising our expectation of paradigm shift at computational chemistry. We recognize simulations as programs that calculate on individual time steps repeatedly and then retrieve time continuous results. Thus, a model is proposed for a flow of a time step composed of analysis, monitoring, visualization and data management. We constructed distributed function environment for simulation on OGSA.

1　研究の背景

近年、医療や創薬の分野において、生体組織の機能解明のために、生体内で起こる化学反応や物理現象をコンピュータ上でシミュレートし、その現象を分析する計算化学研究が盛んである。試験管内での実験や臨床実験に先だって予めコンピュータ上でシミュレーションを実施することにより、実験の対象となる物質や化学反応などの予測を可能とし、実験の効率化へとつながる。また、近年の量子力学シミュレーションの発達は、古典力学では説明不可能であった化学反応のコンピュータ上での再現を可能とし、バイオ情報工学の分野においても計算化学に対する期待は高くなる傾向にある。

生体の現象は階層的に様々な視点により観測され、それに応じて生体シミュレーションも階層的に様々な手法が存在する。生体は様々な組織から成り立っており、それら組織は細胞から構成され、細胞は蛋白質から構成されている。さらに微視的な視点に立つと、蛋白質は分子であり、分子は原子から構成される。そして、原子は核と電子から構成される。これら生体の階層構造に対応して、それぞれを理解する学問として、フィジオーム、セローム、プロテオームなどが存在し、それぞれにおいてシミュレーション手法が研究されている。

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
‡ NEC ソフト株式会社
NEC Soft, Ltd.
§ 大阪大学蛋白質研究所
Insititute for Protein Research, Osaka University
‡‡ 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University

近年では解析対象となる分子構造などのモデルの規模拡大及び複雑化に伴って、微視的な視点による手法と巨視的な視点による手法を組み合わせて全体を表現するというマルチスケールシミュレーションに関する研究開発が進められている。最も微視的な視点に立って生体全体を精密に分析することができれば、それが最良の解に成り得るが、計算量の問題から現実的な手法ではない。このことが現実では詳細な分析を行いたい部分に関しては微視的な視点で分析しつつ、それ以外の部分に対しては巨視的な視点における分析を組み合わせて全体を表現するマルチスケールシミュレーションが選択される理由となっている。このマルチスケールシミュレーションによってこれまで別々の学問として研究されていた手法が連携可能になり、今日までコンピュータシミュレーションで実施することが考えられなかった問題への応用可能性が見出され、計算化学領域におけるパラダイムシフトへの期待が急速に高まっている [1]。

しかし、異なるシミュレーションプログラムの連携は容易には実現できない現状がある。それは以下の 3 点の問題から説明できる。1）インタフェースの不一致、2）データ形式の相違、3）移植性の問題である。これらは異なる組織内で独自の視野に基づいて研究開発が推進されてきたことに起因する。独自開発のため、各プログラムはインタフェースも使用するデータ形式も全く異なっており、連携シミュレーションを実現するためには、インタフェース、データ形式の統一は必要不可欠となる。また、ソフトウェアの動作には専用ハードウェアや、そのソフトウェア専用にチューニングされたクラスタ環境が必要な場合があり、そのような場合はプログラム単体で別組織の環境に移植させることは費用的な面で困難であることは少なくない。これらの問題により生体シミュレーションを用いた研究は大きく阻害されているという現状がある。

本研究では、上述の問題を解決する手段として、地理的に分散するある種の機能を実現するシミュレーションプログラムをネットワークを通じて共有するという機能分散型の共有形態を考え、その連携手法の確立を目的とする。現在われわれは量子力学に基づく分子軌道法と古典力学に基づく分子動力学法を組み合わせた生体高分子シミュレーションの構築を行っている。本論文では、機能分散型シミュレーションに必要な要件をわれわれの事例に基づき明確化し、シミュレーション連携モデルを提案する。さらに、提案するモデルに従って生体高分子シミュレーションを実装し、今後の課題や応用に関して検討する。

以下、2 章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を通して、生体シミュレーションにおいて解決すべき課題を定義する。3 章では課題に対する本研究でのアプローチについて述べる。4 章では現在開発中のシミュレーションプログラムの実装について解説し、5 章では機能分散型の有用性の実証を目的とした、性能面での評価について述べる。そして、6 章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 分散生体シミュレーションにおける課題

2.1 QM/MM 連成シミュレーション

われわれは現在、生体高分子の生体内での働きをシミュレーションするために、量子力学（QM: Quantum Mechanics）に基づく分子軌道法と古典力学（MM: Molecular Mechanics）に基づく分子動力学法とを組み合わせたシミュレーションである QM/MM 連成シミュレーションに関して研究を進めている。生体高分子シミュレーションにおいては、分子運動をシミュレートする MM の力学パラメータのみではシミュレーションできない物理過程が存在するため、電子状態までをシミュレート可能な QM シミュレーションが必要である。計算対象となるモデルを全て QM シミュレーション可能であれば、それにこしたことはないが、QM シミュレーションは非常に計算コストが高く、わずか 10 個の原子の 5×10^{-14} 秒間のシミュレーションに 10 秒程度の時間を要し、現実的な時間でシミュレーションを行おうと考えると数百原子程度が限界である。そのため、計算対象となるモデル中の関心領域（例えば蛋白質の活性部位）のみを QM シミュレーションし、残りの部位は MM シミュレーションを実施する両者を組み合わせて全体を表現する手法を選択している。

QM シミュレーションには NEC によって開発された AMOSS [2] を利用し、MM シミュレーションには大阪大学蛋白質研究所、産業総合技術研究所、日立によって開発された presto-X [3] 内の cosgene を利用する。2 つのプログラムはこれまで独立に構築されてきたものである。それぞれのプログラムは MPI プログラムであり、今日までに様々な並列化手法による高速化が行われている。例えば、AMOSS は非常に高速な計算資源を要求し、intel アーキテクチャに特化した高速化が施されているのに対し、cosgene は理化学研究所が開発した分子動力学計算専用のハードウェアボード MDgrape2 を用いて高速化を図っている。両者共にプログラム単体で流通させることは困難であり、別のクタスタシステムなどに移植するよりは、各々を動作させることが可能な計算資源とあわせて共有する方がコスト面、作業工数の面でも有利であると考えられる。

2.2 QM/MM 連成シミュレーションにおける課題

計算化学の研究者がシミュレーションを行う際には試行錯誤の過程が必ず存在する [4]。生体シミュレーションに限らず、シミュレーションは現実世界の何らかの物理現象や化学反応をコンピュータ上に仮想的に再現することである。しかし、シミュレーションには、必ず物理的な近似が伴う。そのためシミュレーションでは正しく対象をシミュレートできているかどうかを計算中、常に検証できることが望ましい。正しくないシミュレーションと判断された場合、通常シミュレーションに関する近似の再構築やパラメーターの再設定を行う必要がある。QM/MM 連成シミュレーションにおいては、例えばシミュレーション中の温度やエネルギーの時系列変化や分子構造の非物理的変形は正しくシミュレーションが遂行されているかどうかの指標となる。

異なるシミュレーションプログラムの連携において、試行錯誤の過程の支援を考えると、以下のような問題が考えられる。

- インタフェース互換性の問題
- データ形式の互換性の問題
- シミュレーション状態把握の困難さの問題

まず、異なるアルゴリズムを採用するプログラムを連携、交換する場合、そのインタフェースの互換性の問題が存在する。ここで言う互換性の問題は、プログラムインタフェースの互換問題だけではなく、交換するデータ形式に関する互換問題も存在する。同じ解析問題に対して異なったアプローチを取るプログラム間においても、データの本質的な表現、意味は酷似していても、実際のデータ形式にはデータ構造や単位系などの違いが存在する。

また、プログラムが分散することによって、進行中のシミュレーションの状態を把握することが困難になり、シミュレーションの進行や失敗をいち早く判断することが出来ないという問題がある。シミュレーションは仮定した近似モデルのもとで数値計算を進めるものであり、計算自身は正しく進行したとしても、物理的、化学的に意味を持たない状態に遷移し、それ以上のシミュレーションの継続は無意味になる場合が存在する。そのような場合には、シミュレーションを中止し、モデル化のやり直し、精度変更等、パラメータを調整する必要がある。しかし、異なるプログラムの分散環境では、プログラムが地理的に様々に分散していることや、インタフェースが揃っていないことから、シミュレーションの失敗を直ちに判断することは困難であり、その結果、無駄なシミュレーションが非効率に進められる可能性がある。

3 アプローチ

2 章で述べた 3 つの問題は以下のように整理される。まず、インタフェース互換性の問題はそれぞれのプログラムが独自のアーキテクチャ上で構築されたことが原因であるので、共通のアーキテクチャに統合し直すことによって解決可能であると考える。

データ形式互換性の問題は、全てのデータを記述可能

な汎用データ形式は現実的かつ実際的でない。しかし、ある特定の領域に特化し、その領域で使用されるデータを網羅するデータ形式は可能であると考えられる。

異なるプログラム間におけるシミュレーション状態の把握の問題は、インタフェースの統合だけで解決できる問題ではない。シミュレーションプログラムとその状態を監視する仕組みとの関係を新たにモデル化し、プログラム間の関係をそのモデルに従って構築していく必要があると考える。以下、上記のそれぞれに関して本研究での具体的なアプローチに関して述べる。

3.1 インタフェースの統合

3.1.1 OGSA によるプログラムインタフェースの統合

グリッドコンピューティングの標準化団体（Global Grid Forum: GGF）においても、プログラムやリソースに対するアクセス手法を統一しようという動きがあり、OGSA（Open Grid Services Architecture）が提案されている。OGSA は Web サービスをベースにした Grid サービスを定義している。Grid サービスは基本的には XML を用いた RPC（Remote Procedure Call）モデルである。Grid サービスと Web サービスの大きな相違点の一つは、Web サービスがビジネスプロセスの記述に特化し、問題を単純化するために stateless のアプローチが選択されているのに対し、Grid サービスは比較的長時間のプログラム実行が必要な科学計算アプリケーションを考慮し、stateful な RPC を実現している点にある。

本研究では OGSA の XML を用いた stateful な RPC に注目し、プログラム間のインタフェース統合に採用し、各プログラム同士の連携に応用する。本研究で対象とする科学問題のシミュレーションは実際には時間軸を適当な間隔で区切り、各タイムステップにおける計算を繰り返すことによって時間的に連続なデータを得るものである。そのため、各タイムステップの前後の関係が重要な場合が多く、stateful な RPC モデルが不可欠になる。

3.1.2 bmsML によるデータ形式の統合

Web サービスをベースとした OGSA 上でプログラムインタフェースを構築する場合、交換されるデータは必然的に XML 形式になる。しかし、たとえ XML 形式であったとしても、そのスキーマが共通化されているか、もしくは変換可能であるなど互いのプログラム間で把握できるものでなければ、シームレスなデータ交換を行うことは不可能である。

われわれは以前から XML によるデータ形式の統一の必要性に着目し、生体高分子シミュレーション用のデータ形式として、bmsML（biomolecular simulation Markup Language）を提案してきた。bmsML はスカラー値や配列、行列式など、数値計算の基本的なデータ型を XML でどのように記述し、交換するかを定義するものである。本論文執筆時点では、表現できるデータ型はプリミティブであるが、データ交換用の統一データ形式としては十分な機能性を呈しているため、本研究では bmsML をデータ交換形式として採用した。bmsML に関してはまだまだ策定の最中であり、bmsML 自身の構造を記述する仕様や、その構造記述を用いたデータ交換方法などに関して議論が続けられている。

3.2 シミュレーションモデルの構築

連携するシミュレーションプログラムに状態を監視する仕組みをシームレスに組み込むためには連携させるシミュレーション環境に要求される機能を考察し、シミュレーションの一連の流れ（モデル）を定義する必要がある。つまり、シミュレーション中に互いがどのような制御メッセージやデータ交換方法をとるべきであるかを定義する必要がある。複数のシミュレーションプログラムに対応して、シミュレーション状態を把握するための監視プログラムも複数存在する。監視する対象はシミュレーション状態を可視化する画像であったり、シミュレーション中のある物理量（エネルギー値や収束率など）であったり様々な監視モジュールの組み合わせが考えられ、その組み合わせは非常に複雑になる。しかし、シミュレーションはタイムステップごとに刻まれ、1 ステップで生成されたデータをどのように処理し、監視、可視化プログラムに受け渡すかという部分は共通化可能である。そのため、本研究ではシミュレーションにおける計算モジュールと監視、可視化モジュールの関係を明確にし、シミュレーション連携に伴い、汎用的なモデルを構築する。

Web サービスや Grid サービス間の連携に関する研究は既に先行研究としてワークフローの研究が進んでおり、そこで議論されているサービス間の連携のモデルは本研究におけるシミュレーションの連携のモデルを構築する上で参考になる。具体的な例としては Web サービスの連携においては BPEL [5] の標準化が進みつつあり、Grid の分野においても Kepler [6] や Triana [7] など様々なワークフローに関する研究が行われている。

BPEL は Web サービスの本来の動機であるビジネスプロセスの処理に特化しており、サービス間でやり取りされるデータ、各サービスの呼び出しなどの制御は全て BPEL の実行エンジンで集中管理される。中央集中管理モデルの場合、プロセスの進行の制御を集中管理できるため、プロセス全体の状態の把握、制御が容易である。ただし、ビジネスプロセス上ではあまり大きなデータを扱わないため、中央集中管理型がボトルネックになることはないが、巨大なデータをやり取りする科学アプリケーションに対する適用は難しい。

それに対し、Grid 分野において研究されているワークフローはデータを処理していく過程であるデータフローに注目して複数サービスの連携を考えるものが多い。データフローに従って各サービス間でデータが流れていくため、データ交換におけるオーバーヘッドは小さい。ただし、制御メッセージが分散するため、並列に動作するサービス間での同期が難しいという問題がある。

本研究では、中央管理によるシミュレーション全体の同期の容易さとデータフローの非同期的なデータ処理の長所を選択し、計算開始の制御メッセージは中央コントローラによって集中管理し、計算開始における同期は必ず行う。しかし、計算終了の制御メッセージはその計算結果を必要とする別のサービス（監視サービスやデータ管理サービス）が複数あることを考え、非同期的にそれらのサービスへ直接配信するモデルを提案する。

提案するモデルを構成する要素はシミュレーションサービス、監視・可視化サービス、データ管理サービスからなる。図 1 はシミュレーションサービス間に流れる 1 ステップ分のメッセージのモデルを、QM/MM 連成シミュレーションと同じく、2 つのシミュレーションプログラムが連携している例をもって示している。第 1 のプログラムが終了すると、そのデータはコントローラを介

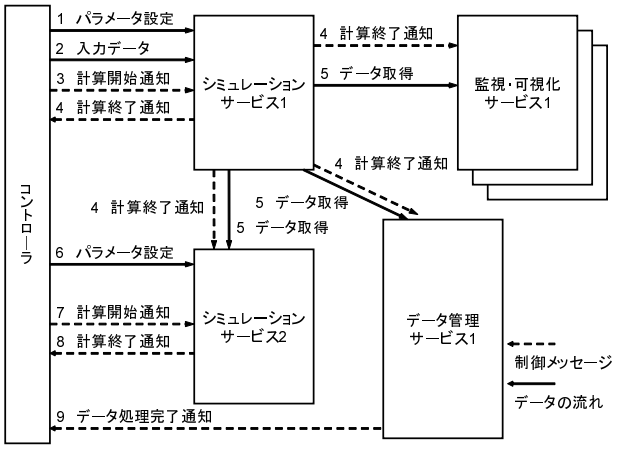

図 1: シミュレーションモデル

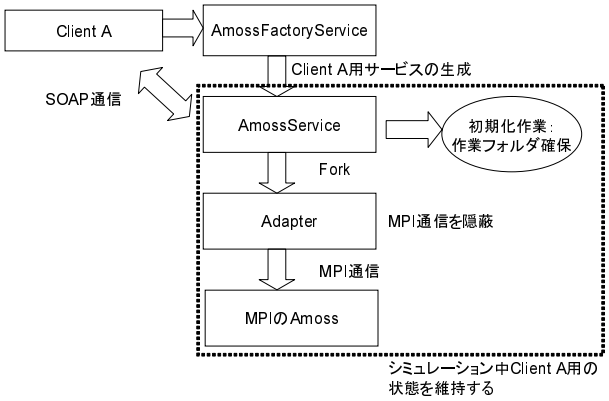

図 2: AMOSS サービスの実装概要

することなく第 2 のプログラムに渡され、実行される。第 1 のプログラムの次のステップの計算には第 2 のプログラムの結果が必要なため、待機状態にはいるが、第 1 のプログラムの結果を検証する監視・可視化サービスやデータ管理サービスは 2 つのサービスとは非同期的に活動を開始する。コントローラは第 2 のプログラムの終了と第 1 のデータの検証の終了の同期を取って、次のステップの制御を始める。このモデルでは、BPEL とは異なり完全な中央集中管理ではないため、データ転送などのオーバヘッドを減らし、かつ並列度を向上させることができるため、シミュレーション効率が向上すると考えられる。また、完全なデータ駆動型ではなく、計算開始制御は中央集中管理されるため、次のステップの開始を直前のステップ中に終わらなければならない処理（例えばデータ処理など）の終了に同期させることが容易である。

4 QM/MM 連成シミュレーションの実装

本論文執筆時点では、3 章で述べたモデルのうち、データ管理サービス以外のプロトタイプ実装は完了している。その解説を以下に行う。実装には OGSA の参照実装である Globus3.2 を用いている。

4.1 シミュレーションサービスの実装

図 2 は OGSA による AMOSS サービスの実装の概要である。Globus3.2 が提供する stateful なサービスを構築する機能である Factory パターンを用い、個々のユーザ専用のサービスを生成し、状態維持の機能を提供している。シミュレーションに使用している AMOSS や cosgene はその動作に伴って様々なファイルを読み書きするため、ユーザ専用のサービスが生成されると、サービスごとに作業用フォルダをサーバ上で確保している。このようにユーザ専用サービス生成、作業フォルダの割り当てを経て、シミュレーションの各ステップの状態を個々に維持できる環境を用意する。

図 2 中の Adapter は、MPI プログラムである AMOSS や cosgene を Java で実装されている Globus3.2 上で動作するサービスと結合する役割を持つ。Java から MPI プログラムを制御する方法には、以下の 3 通りの方法が考えられる。1 つ目は Java で書かれた MPI ライブラリ [8] を利用する方法であるが、AMOSS や cosgene で使用されている MPI ライブラリと実装上の互換性がないため利用できない。2 つ目は JNI（Java Native Interface）を用いて、AMOSS や cosgene で用いている MPI ライブラリを利用する方法がある [9]。しかし、Java のコードにネイティブ MPI ライブラリのコードがリンクされるため、MPI 通信の不具合でプログラムが停止する時、Java のプロセス、つまり OGSA のサーバプロセスをも巻き込んで停止するという問題が発生したため、この方法は避けた。3 つ目は AMOSS や cosgene を制御する MPI プログラムを新たに書き、そのプログラムを Java からは Fork し、標準入出力で制御メッセージのみをやり取りする方法である。この AMOSS や cosgene を制御するためのプログラムが図中の Adapter である。別プロセスとして MPI 制御部を切り離すことによって、サーバプロセスへは一切影響を与えることがない。

4.2 監視・可視化サービスの実装

図 3 は監視・可視化サービスの実装の概要である。監視・可視化サービスはポータルサーバ上に実装され、ユーザの要求に応じて非同期的にシミュレーションサービスへ問い合わせを行い、データの取得を行う。そして、自分の役割に応じて適切にデータを加工し、クライアントのブラウザからアクセス可能なように、Servlet や Applet を用いて情報を提供する。図 3 で表示している例は、シミュレーション中の分子構造の表示と、シミュレーションの系のステップごとの全エネルギー値をグラフ化したものである。共に cosgene サービスから PDB データや全エネルギーに関するデータを取り出し、可視化を行っている。分子構造表示には大阪大学蛋白質研究所が研究開発をしている jV（PDBj Viewer） [10] の Applet 機能を用いた。Script 機能により外部から制御できる点が既存の plug-in に比べて優れており、ポータルサイトから動的に変化する PDB データを参照する際に適している。全エネルギー値のグラフ化はエネルギー値の数値配列を cosgene サービスから取り出し、gnuplot によってグラフ画像を動的に生成することによって行っている。

5 実行時間に関する評価

本章では、QM/MM 連成シミュレーションにおいて、AMOSS サービスの計算時間、cosgene サービスの計算時間及びサービス間のデータ転送時間を測定することによって各所要時間の割合について評価を行い、機能分散型共有のアプローチの有用性に関して考察する。評価に

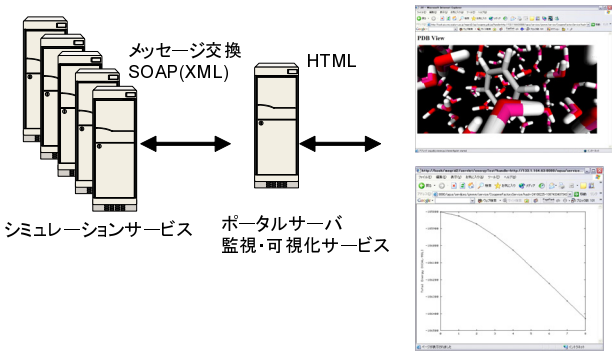

図 3: 監視・可視化サービスの実装概要

は水分子 225 個に囲まれたベンゼン分子 1 個のモデルを用いた。ベンゼン分子とその近くに存在する水分子 1 個を QM 領域とし、AMOSS サービスによって電子状態シミュレーションを行った。その他の水分子 224 個は MM 領域とし、cosgene サービスによって分子動力学シミュレーションを行った。評価には以下の環境を用いた。

- AMOSS サービス
  プロセッサ: Pentium Xeon 2.8GHz Dual
  メモリ: 2GB
  ノード数: 10 ノード
- cosgene サービス
  プロセッサ: Pentium4 2.8GHz
  メモリ: 2GB
  ノード数: 4 ノード
  MDgrape2 ボード搭載

測定結果を表 1 に示す。表 1 は 50 ステップ分を平均化して 1 ステップ分のデータにした計算時間と通信時間と、それぞれの時間比である。ただし、データ転送時間は AMOSS の結果を cosgene に転送する時間と、cosgene の結果を AMOSS に転送する時間の合計値である。この表を見ることにより、シミュレーションの 1 ステップ毎の実行時間の内訳を把握することができる。表より、電子状態までシミュレートする AMOSS の計算時間が最も長いことが分かる。cosgene は計算に汎用プロセッサの数十倍の演算能力がある分子動力学シミュレーション専用ボード MDgrape2 を利用しているため、計算時間は非常に短い。今回測定に用いたモデルでは 1 ステップ中に AMOSS から cosgene へ転送されるデータは 4 つあり、合計約 3KB である。また、cosgene から AMOSS へ転送されるデータも 4 つあり、合計約 75KB である。データ転送にかかる時間はこれらのデータを SOAP のメッセージに埋め込んで、転送するという前処理も含まれているが、ともにデータ量は小さく、転送にかかる時間もわずかである。

ネットワークを通じて、機能分散を行う環境を考える場合、データ転送に伴うオーバーヘッドの問題が懸念される。今回の QM/MM 連成シミュレーションの場合はデータ転送に 205.4 ミリ秒かかっているが、SOAP メッセージに埋め込まれる前のデータを SOAP と同じ http による転送時間を測定してみたところ 89.1 ミリ秒であった。このことから SOAP メッセージの構築にはかなりのコストがかかっていることが分かる。

表 1: QM/MM 連成シミュレーションの実行時間

	AMOSS	cosgene	データ転送	合計
所要時間 (msec)	9099.7	207.71	205.4	9512.8
時間比 (%)	95.7	2.18	2.16	100

計算対象となるモデルの規模を拡大した場合、1 ステップの計算により生成されるデータ量は線形に増加するのみであり、SOAP のメッセージ構築にかかる時間も同じく線形に増加すると考えられる。しかし、QM の計算量は電子軌道数にあてはめる基底関数の個数を n とすると $o(n^4)$ となり、MM の計算量は原子数を m とすると $o(m^2)$ 及び $o(m \log m)$ になることから、データ転送にかかる割合は相対的に減少する。そのため、モデルの規模が拡大するほどデータ転送による処理低下への影響は小さくなると考えられる。

以上のことから、QM 計算と MM 計算のネットワークを介した機能分散型の連携においては、データ転送におけるオーバヘッドは問題にはならず、機能分散型の連携は有効な手法であると考えられる。cosgene の計算時間に対し、データ転送にかかる時間が長いという問題はあるが、cosgene 計算は専用ボードによって高速化されたものであり、多少のオーバヘッドが存在したとしても、ネットワークを介して使用する価値はある。QM/MM 連成シミュレーションにおいては、データ転送によるオーバヘッドがもたらすデメリットよりも、各計算に特化してチューニングされた機器間をネットワークを介して接続することによって得られるメリットの方が大きいと考えられる。

6 まとめと今後の課題

本論文では生体シミュレーションにおけるマルチスケールシミュレーションに関して述べ、複数のシミュレーションプログラムを OGSA 上で連携させる機能分散型のシミュレーションモデルを提案した。われわれは QM/MM 連成シミュレーションの実装を通して、時間軸に沿って進められるシミュレーションの 1 ステップに関して注目し、解析、監視・可視化、データ管理の流れを定義した。具体的には並列度を高めつつシミュレーション全体の同期を取るために解析開始の制御メッセージは中央集中管理し、その結果を用いる監視・可視化サービス、データ管理サービスに対する解析終了通知は非同期的に分散配信するモデルを提案した。そして、機能分散型のシミュレーション連携モデルの有用性を確認するために、専用ハードウェアを利用して実装される cosgene サービスに関してパフォーマンス評価を行った。

本論文で述べた機能分散型シミュレーション連携環境により、異なるシミュレーションを連携させてシミュレーションを進めていく基盤技術は整った。今後はこのシミュレーション連携基盤を応用することにより、より高度な問題へと発展可能であると考える。例えば、以下が挙げられる。

- パラメータ調整の自動化
  現在のシステムはシミュレーションの流れをモデル化し、試行錯誤の過程をスムーズに連携させることによって、試行錯誤の過程を支援している。しかし、試行錯誤の本質的な問題であるパラメー

タ調整方法やアルゴリズムの調整方法に関しては何も支援していない。試行錯誤の過程の本質的な支援のためには、現在人間が行っているパラメータ調整など試行錯誤の過程を定量化し、シミュレーションがある条件で失敗する時はどのパラメータを調整するのかといった知見をデータベース化することによって、自動的にパラメータ調整やアルゴリズムの交換などを行う必要があると考える [4]。

- Drug Screening
  QM/MM 連成シミュレーションの 1 つの応用として、Drug Screening が挙げられる。蛋白質と薬の候補となる化合物の相互作用をシミュレーションすることによって、臨床実験などの前に候補化合物をより絞り込むことができる。一次 Screening としての docking と二次、三次 Screening としての QM/MM 連成シミュレーションの連携により、標的蛋白質に対する候補化合物を 100 万個程度の規模から数十個の規模まで絞り込むことも可能であると考える。

情報科学の側面から考えられる課題には以下のようなものが挙げられる。

- 並列化と最適化
  上述したような応用を考えると、パラメータを少し変えたシミュレーションを並列して実行したいといった要求が生じる。その場合、限られた資源の中で効率よくシミュレーションを並列化する必要がある。現在はシミュレーションの並列化効率に注目し、並列化を最適化するスケジューリングに関して検討している。
- 実験結果の管理、共有方法の確立
  これは本論文では未実装としていたデータ管理サービスのことである。実験結果の保存においては、シミュレーションの過程において生成されたデータ全てが必要なわけではない。シミュレーションの十分な検証が済んだ後ならば、毎ステップごとのデータはもはや必要なく、10 ステップごとのデータだけ残して管理するなどといった処理が必要になる。
- ワークフローによるシミュレーション同士の連携手法の確立
  上述したように QM/MM 連成シミュレーションにとどまらず、docking シミュレーションなどとの連携を考えると、各サービス間の連携はより複雑化すると考えられる。そのため、サービス間の連携を明確に記述する方法が求められる。つまり、本システムのシミュレーションモデルの構築時にも考慮したワークフローなどの記述による連携手法の確立を必要とされる。

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、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、ご助言頂いた日立製作所の何希倫博士、黒澤隆様に感謝する。jV に関してご助言頂いた大阪大学蛋白質研究所の木下賢吾助教授、情報数理研究所の佐藤寛之様に感謝する。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り、また、科学研究費補助金特定領域研究（C）「Grid 技術を適応した新しい研究手法とデータ管理技術の研究」（13224059）の助成を受けて行われた。

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] Toshihiro Sakuma, Hiroshi Kashiwagi, Toshikazu Takada, and Haruki Nakamura. Ab initio MO study of the chlorophyll dimer in the photosynthetic reaction center. i. a theoretical treatment of the electrostatic field created by the surrounding proteins. *Int. J. Quant. Chem.*, Vol. 61, pp. 137–151, 1997.

[3] Yoshifumi Fukunishi, Yoshiaki Mikami, and Haruki Nakamura. The filling potential method: A method for estimating the free energy surface for protein-ligand docking. *J. Phys. Chem. B.*, Vol. 107, pp. 13201–13210, 2003.

[4] Takashi Maeno, Susumu Date, Yoshiyuki Kido, Ichiro Hasegawa, and Shinji Shimojo. A system architecture assisting user trial-and-error process in in-silico drug design. In *7th International Conference on High Performance Computing and Grid in Asia Pacific Region (HPC Asia 2004 Biogrid Workshop)*, July 2004.

[5] Business process execution language for web services version 1.1. `http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/`, 2003.

[6] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. Kepler: An extensible system for design and execution of scientific workflows. In *16th International Conference on Scientific and Statistical Database Management (SSDBM'04)*, pp. 21–23.

[7] Ian Taylor, Matt Shields, Ian Wang, and Roger Philp. Distributed P2P computing within Triana: A galaxy visualization test case. In *International Parallel and Distributed Processing Symposium (IPDPS'03)*, April 2003.

[8] 日下部明, 廣安知之, 三木光範. Java による mpi の実装と評価. 情報処理学会 2000 年記念並列処理シンポジウム JSPP2000 論文集, pp. 269–275, 2000.

[9] Bryan Carpenter, Vladimir Getov, Glenn Judd, Tony Skjellum, and Geoffrey Fox. Mpj: Mpi-like message passing for java. *Concurrency: Practice and Experience*, Vol. 12, No. 11, September 2003.

[10] Kengo Kinoshita and Haruki Nakamura. eF-site and PDBjViewer: database and viewer for protein functional sites. *Bioinformatics*, Vol. 20, No. 8, pp. 1329–1330, February 2004.