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

2004－HPC－100　(6)
2004／12／17

Grid Computing による科学技術計算のためのワークフローシステム

増田 慎吾[1]　蟻川 浩[1]　市川 本浩[1]
藤川 和利[2]　砂原 秀樹[2]
1 奈良先端科学技術大学院大学情報科学研究科
2 奈良先端科学技術大学院大学情報科学センター

概要

近年、計算機とネットワークの性能向上とともに、Grid と呼ばれる新たな分散処理技術が注目を浴びている。その中で、科学技術計算への応用研究が数多く行われてきた。科学技術計算では、一連の処理の中で途中の結果から計算における入力パラメータを調整し直し、再度実行するという試行錯誤が必要な場合が多い。そこで、本研究報告ではこの試行錯誤を Grid 上で効率的に行うためのワークフローシステムの提案を行う。また、現在の実装状況と実行例、今後の予定と課題について説明する。

The workflow system for scientific simulation with Grid Computing.

Shingo Masuda[1]　Hiroshi Arikawa[1]　Motohiro Ichikawa[1]
Kazutoshi Fujikawa[2]　Hideki Sunahara[2]
1 Graduate school of Information Science, Nara Institute of Science and Technology
2 Information Technology Center, Nara Institute of Science and Technology

Abstract

In recent year, with the improvement in a performance of the computer and the network, the new distributed processing technology, called 'Grid', is attracting attention. And Grid computing has been applied to scientific simulation in a lot of cases. In a partial of scientific simulation, we need to doing 'Cut and Try' for getting result which has good quality we want. In this paper, we describe Workflow system which is increasing efficiency of doing 'Cut and Try' and getting good quality result in Grid Environment. Moreover we explain progress of implementation, use case of this system, and a future subject.

1.　はじめに

近年、計算機とネットワークの性能向上とともに、組織を超えて様々な種類の計算機を共有し、通常では得ることの難しい多くの計算機資源を獲得するためのGrid と呼ばれる新たな分散処理技術が注目を浴びている。特に、多くの数値計算を行いCPU資源やメモリを大量に必要とする科学技術計算における応用研究が数多く行われてきた。その中で、ユーザがGrid上で実行したい一連の流れを自動化するために、ワークフローシステムの導入が行われている。ワークフローのコンセプトはビジネスの分野で広く利用されており、"Workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."[1]と定義されている。これにより、Gridを利用した科学技術計算を行う場合の利便性が促進される。

ここで、ワークフローの用語を定義すると、プログラムの実行やファイルの転送などのすべての処理をワークと呼ぶ。ワークフローは、ワークの実行順序以外に、データの流れも表現する。このデータの流れは、データフローとも呼ぶ。

一方、科学技術計算では、多くの試行錯誤が行われている。そして、科学技術計算を行う人の期待としてある、「必要となる精度をもつ結果を、より高速に得る。」ためには、自動処理によって試行錯誤が効率的に行える必要がある。

我々は本研究を、「たんぱく質立体構造予測」を実例に取り上げて進めている。そこで、本研究報告ではこの例を元に科学技術計算における試行錯誤を支援し、効率的に進めるためのワークフローシステムを提案する。

次節では、「たんぱく質立体構造予測」を例にした科学技術計算における試行錯誤と、計算機環境に求められる機能を説明する。次に3. ではGrid環境とワークフローシステムの関連研究について説明する。2. における要求から、ワークフローシステムに取り組むべき機能について述べる。そして、4. では、試行錯誤を効率的に行えるワークフローシス

テムの提案を行う。5. では、現在実装中のシステムの構成を説明し、6. で現状と今後の予定や課題を説明する。

2. 科学技術計算と試行錯誤

ここでは、「たんぱく質立体構造予測」を例に、科学技術計算の試行錯誤について説明する。

2.1 たんぱく質立体構造予測について

「たんぱく質立体構造予測」とは、アミノ酸配列情報からたんぱく質の立体的な“かたち”を予測することである。その方法は大別すると 2 種類に分けられ、1 種類目はすでに構造が既知のアミノ酸配列から類似する部分を検索する方法である。これは、数 10 分程度の時間で実行できるが、既知のアミノ酸配列と類似しないたんぱく質では答えを求めることができない。そこで、2 種類目として物理化学的な計算によって構造を予測する方法がある。この方法はまず、たんぱく質全体を細かなパーツに分解し、各パーツに対し乱数を使って色々な構造を当てはめる。そして、全体として物理的に安定しているかを計算し、より安定している方を選んでいく(以下、この当てはめから安定した構造を探索する過程を〈A〉と呼ぶ)。この〈A〉を乱数の初期値を変えることで数 100 通りから数 1000 通り行う。次に、この多量の結果の中には非常に似た構造のものが存在するので、類似するもので集合を構成する。この作業はクラスタリングと呼ばれる。実際の答えは、最も大きい集合の中に存在すると考え、その集合から最もエネルギー値が低いもの、つまり物理化学的に安定しているものを選び答えとする。この方法は、構造の組み換えとエネルギー値の計算を大量に実行し時間がかかるため、Grid環境により〈A〉を並列に実行することで高速化する。

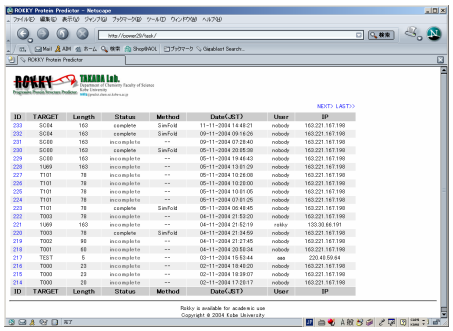

図 1 たんぱく質立体構造予測実行画面

2.2 試行錯誤の例

一方、実際の計算においては、単純にこれらの方法を実行するだけでは良い結果を得られない場合がある。ここで、良い結果とはたんぱく質の研究者から見て「たんぱく質らしい」結果で、実際の構造に近いと予想される結果を指す。そして、実際に計算を行う場合は、まず 1 種類目の方法で答えを見つける。この方法では、部分的にしか答えがわからない場合や全く答えがわからない場合がある。そこで見つからない部分に関して 2 種類目の方法を適用する。2種類目の方法を実行する場合には、対象を複数に分割して計算する場合がある。そして、分割方法によって得られる結果の良さは変わる。また、分割された部分によって、必要な〈A〉が数 100 通りでよい場合や、数 1000 通りの場合などがある。また〈A〉には、温度スケジューリングなどさまざまなパラメータがあり、これらが良い値かどうかは計算を実行してみないと判断できない(以降、たんぱく質の分割方法、〈A〉を何通り実行するか、温度スケジューリングなどをすべてパラメータと呼ぶ)。そこで、一度計算を行った結果からパラメータを調整し直し、もう一度実行するといった試行錯誤な操作を行っていく必要がある。

2.3 計算機環境に求められる機能

一方、「より高速に、実際の構造に近いと予想される結果を得る」ために、単に計算が高速に実行できる環境だけでなく、上記の試行錯誤を効率的に行うための機能が計算機環境に求められている。我々は Grid 環境で実行することを対象にこの問題を考え、前提として以下の特性があるものとする。第1に多種類の計算機が存在し、処理能力がそれぞれ異なる。第 2 に、計算機資源は複数の人によって共有され、利用できる計算機資源は時間によって異なる。このような特性を考慮し、試行錯誤がなされる計算をより速く完了するための「戦略」があり、それを実現することが必要であると考える。

「戦略」とは以下のような例が考えられる。まず、計算機資源を多く確保できない場合は、できるだけ失敗に関わる計算処理を少なくしたい。そこで、最後まで計算を行う前に、途中の結果からこのまま続けた方が良いかを判断し、必要ならばパラメータを修正し実行し直す操作が必要になる。また、パラメータを修正し計算をやり直す場合に、全体を実行し直すのではなく、必要な部分だけを実行し直すことで、無駄な処理を少なくする必要がある。一方で、計算機資源が多く確保できる場合には、複数のパラメータで処理を並行に行うことで、良い結果に短時間でたどり着けるようにしたい。このように「戦略」とは、プログラムの実行方法を条件に応じて変更することである。

そして、この「戦略」をユーザが容易に実行できる環境が必要となる。さらに、これらの操作すべてをユーザが一つ一つ手動で行った場合、ユーザの手間がかかる他、ユーザが結果を確認しないと失敗した計算に時間が消費され効率が悪い。そこで、可能な範囲で試行錯誤も自動的

に行える必要がある。

次節では、ワークフローシステムについて関連研究とその課題を説明する。

3. 関連研究

Grid 用ワークフローシステムは多くの Grid プロジェクトから開発されている。例えば、UniCore[3], MyGrid[4], などのプロジェクトによるGrid 用クライアントソフトは、ワークフローを定義し、一連のジョブの自動実行が可能である。また、Globus Toolkit[5]においては GridAnt[6] と呼ばれるワークフローツールのテスト版が用意されている。

ワークフローシステムはワークフローを入力するユーザインタフェースアプリケーションと、各ジョブの実行を行うサービスによって構成される。ユーザが入力したワークフロー情報は、XML 形式などの記述言語によって表現され、各 Grid サービス間でやり取りされる。ワークフローシステムによってどのようにプログラムを実行できるかは、ワークフロー情報を記述する言語にどのような表現ができるかによって決まる。ワークフローを記述するための言語はGrid技術の標準化団体 GGF(Global Grid Forum)[8]においても標準化の議論がなされているが、現状では各システムが独自に設計された言語を利用している。たとえば、MyGridにおけるワークフロー 記述言語は Scufl(Simple Conceptual Unified Flow Language)[7]が利用され、GridAntは JAVA によるソフトウェアのビルド自動化スクリプトである Apache Ant をベースにGrid用に拡張した言語である。以降では、既存のワークフロー記述言語がどのような表現が可能になっており、ワークフローシステムによってどのようにプログラムの実行が行えるかを説明する。

ワークフロー記述言語の基本は、

1. どのプログラムを実行するか
2. 入力ファイル、出力ファイルはどれか
3. どの順に実行するか
4. プログラム間のデータのやり取り

という情報を表現することである。これにより、科学技術計算における一連の作業を自動化する。(例えば、計算を行った後、可視化ツールで画像を作り、結果を画面に表示するといった流れを行う。)科学技術計算には、初期値を変えての多量の並列実行や繰り返しといった事が行われる。そこで Grid ワークフロー記述言語には、並列処理や繰り返し処理を表現する手段が用意されている。ユーザインタフェース上でも、複数のワークを定義する必要はなく、並列実行する数を入力するだけでよい。他に便利な機能としては、あるプログラムの出力を別のプログラムの入力とする場合のデータフォーマットの不一致を自動的に解決する機能がある。各プログラムが扱うデータに関しての情報を記述することが可能となっており、相互に変換する方法が保存されている。これは、ワークフロー記述言語とは別に、プログラム情報を記述する言語が用意されている。

一方、これら既存の Grid ワークフローシステムを用いた場合、試行錯誤を伴う科学技術計算を行う場合に必要な、途中の結果からプログラムの実行を変更することができないという問題がある。これは、既存のワークフローシステムがファイルなどのデータに関して、プログラム間のやり取り(データ形式やその方法)や保存(場所やデータ形式、ファイルの名前空間)の管理を行うが、内容の意味的な情報に応じたワークフローの動作変更を行うことができないからである。そして、ユーザが特定のプログラムに対し手動でパラメータを変更した場合でも、再実行する必要のあるプログラムだけを抜き出して自動実行する機能はない。

以上の点から次節では、これらの問題点を解決するための新たな機能を持つワークフローシステムとその構成について提案する。

4. 試行錯誤を支援するシステム

我々は、以下の機能がワークフローシステムに必要であると考え、新たなワークフローシステムとして提案する。

4.1 パラメータ変更と再実行

まず、パラメータの変更などが行われた時に、自動的にプログラムを再実行するための機能として、次の①～③を提案する。

①各ワークに関連するデータの保存

計算を行うプログラムは、問題のデータやパラメータをファイルなどから入力し、結果をファイルなどに書き出すことを行い、プログラムによっては既存のファイルの内容を変更する場合があることである。そのため、計算のやり直しには、プログラムの実行する前の状態にファイルを戻すことが必要となる。そのため、各プログラム(ワーク)の実行に関連するファイルをすべて保存しておく機能が必要になる。また、以前の計算結果を自動的に保存するようにすることで、様々なパラメータ値に対して比較を行えるようする。このように、すべてのワークに対して入力と出力をともに保存しておく。

②パラメータ変更時に、再実行すべきワークの決定

プログラムを再実行する前に、再実行の必要となるプログラムはどれかを決定する機能が必要となる。ここでは、いくつかのシナリオを例に詳細を説明する。

第 1 の例は、ワークフローが図 2 に示した形になっているものとする。これは、A～D のワークがあり、A を実行した後、B と C を並列に実行し、D を実行する。図1意味は、A が完了した場合、B と C どちらも実行可能であることを示す。B、C の順に実行しても、C、B の順に実行しても、また同時に実行しても構わないとする。そして、D は必ず B,C 両方が完了してから実行するものとする。ここで、今 D を実行しているとする。B、C に関する結果は確認できる状態である。このとき、B のプログラムに関係するパラメータを変更する。そして、再実行をユーザが指示すると、D の実行を中断し、B のプログラムを新しいパラメータで実行し直す。A と C は実行する必要はないので実行しないようにする。ここで、以前のパラメータによって出力された B の結果は、別の場所に移動し上書きされても消えないようにする。この操作はコピーではない。プログラムによっては、すでに結果用のファイルがあると正常に動作しない場合があるからである。B の再実行完了後には、D の再実行を行う。B の再実行と同様に、自動的に途中までの結果は別の場所に移動する。

第 2 の例は、ワークフローが図 3 に示した形になっているものとする。意味は図 2 と同様に見るが、注意すべき点は D と E のプログラムの違いである。D は、B と C のプログラムの両方が完了していないと実行できないが、E は C のプログラムが完了していれば良い。そして、今 D と E が実行中であるとし、B のプログラムのパラメータを変更したとする。この場合、D のプログラムの実行は中断し、E のプログラムは実行したままにしておく。そして、B と D の順に第 1 の例と同様に再実行処理を行う。このように、プログラムの再実行処理は、ワークフローを完全に停止させることなく実行する必要がある。

③再実行をどの順で実行するかを決定する機能

パラメータの変更が複数の場合には、プログラムの再実行順序は複数の方法が考えられ、どの順序で再実行するかを決定する必要がある。ここでは、上で利用した図 3 のワークフローを用いてその例を示す。

まず、BとCのプログラムには、それぞれ 1 個のパラメータを必要とする。そして、最初に B のプログラムに Bp1 というパラメータ値を与え、Cのプログラムに対して Cp1 というパラメータ値を与え実行する。次に、Fを実行中にBのパラメータを Bp2 に変更したとする。さらに、Cのパラメータを Cp2 に変更する。この場合、(B のパラメータ、C のパラメータ)=(Bp1,Cp1), (Bp2,Cp1), (Bp2,Cp1), (Bp2, Cp2)の場合が考えられる。すべてのパターンを実行する場合、D と F のプログラムでは 4 パターンで実行しなければいけない。ここで、Fを実行中にパラメータ変更をしたので、(Bp1, Cp1)の組み合わせに関しては、D と E のプログラムは処理が終わっている。そこで C のプログラムの Cp2 のパラメータ値を先に実行すると、(Bp1, Cp2)の組み合わせに関して D,E,その後 F のプログラムの再実行を行うことができる。B のパラメータ値 Bp2 の場合を先に実行した場合、(Bp2, Cp1)のパターンにおいて、Dの再実行、その後 F の再実行を行う。このため、実行できるプログラムの数、確認できる途中結果などが、プログラムの実行順によって変わる。この問題は、利用できる計算機資源の数にも依存するため、どのように処理すべきか一概には決定できない。また、ユーザや問題によってどのようなパターンを先に確認したいかも異なると考えられる。そこで、ユーザがどのような順で実行できるかを選択でき、かつどのくらいの時間で、どのパターンについて途中結果や最終結果が出てくるかの見積もりを表示し、ユーザの選択を支援する必要がある。時間の見積もりに関しては、すでに(Bp1, Cp1)に関する実行を行っていることから、その実行時間をもとに見積もることができると考えられる。

以上の例ついて手動で行う場合、ユーザがジョブの管理を意識する必要があるため、このようなプログラムの再実行処理の自動化は非常に有用であると考えられる。

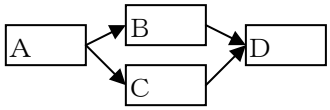

図 2 ワークフロー1

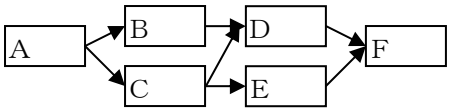

図 3 ワークフロー2

4.2 試行錯誤の自動化

4.1 ではユーザが手動でパラメータを修正する場合について説明した。次に、パラメータの修正も状況に応じて自動化することを考える。この自動化を行うには、途中の結果を取り込み、そこから現在の計算が良いかどうかを判断し、パラメータの修正と必要な計算の再実行を行う機能が必要である。そして、ユーザが容易に結果ファイルの情報からプログラムの実行を変更するルールを設定できる環境が必要である。そこで、以下の④、⑤の機能を提案する。

④パラメータに複数の候補を設定する機能

各プログラム(ワーク)のパラメータに、ユーザが候補と

適用順序を設定できるようにしておく。

⑤出力結果によるワークフローの動作変更機能

ユーザのプログラムからの出力情報を、ワークフローシステムに取り込み、パラメータを変えるためのインターフェースを用意する。ユーザは、途中の結果ファイルからパラメータがこれで良いかを判断するプログラムを用意する。ワークフローシステムには、そのプログラムの出力情報から、ある条件がそろえば別のパラメータを試すというワークフローを入力できるようにする。そして、2．3の「戦略」にある、利用できる計算機資源に応じた動作の変更を行う。つまり計算機資源が多く確保できるときは、あらかじめ複数のパラメータで実行し、確保できない時は優先順位の高いパラメータから処理を行うようにする。またパラメータの変更によって結果が悪くなれば、自動的に元に戻して処理を行う。このような処理の違いを、ひとつのワークフローで実行できるようにする。そしてワークフローには特に処理の分岐を書く必要はないようにする。実際にワークフローにパラメータを変えることのあるプログラムの実行を入力する場合に、入力すべき内容は以下の例のようになる。

ア．＜ワーク名＞

イ．＜プログラム名＞　パラメータ1

ウ．パラメータ1の候補

そして、途中結果の判定プログラムの出力から、データを取り込む処理とそれによる判定方法を入力しておく。また処理の判定記述例は次のような形式にする。

IF（変数による条件式）THEN TRY（ワーク名，パラメータ1）

この例では、条件式を満たせば、ワーク名のパラメータ1の候補から再実行を行うことを意味する。そして、先に変えたように、計算機資源が多くそのワークを2つ以上並列実行できると判断した場合、条件式を満たす前に予め次のパラメータ候補での実行を行う。

4.3 ユーザインタフェース

次にユーザインタフェースについて、構成と必要な機能を説明する。

■*ユーザインタフェースの構成*

ユーザは、結果を見ながら関連するパラメータの修正を行いたい。また、何度か再実行した際には、パラメータとその結果を比較してどのパラメータを採用するかを決定する。このため、結果の表示とワークフローの制御は同一の画面で操作できることが望ましい。また、このような操作をどこからでも行える環境が必要となる。この点について、我々はWebインターフェースによって実現することを提案する。

■*結果表示とパラメータ変更用Webの生成機能*

結果の表示を行うためには、計算結果の可視化が必要になる。ワークフローには、各プログラムによってどのようなデータが出力されるか、またそれを可視化するためプログラムの実行を指定できるようにする。ワークフローシステムは、出力結果を指定された方法で画像に変換する。そして、設定されたパラメータと画像情報をWebサーバに転送する。Webサーバは、パラメータと結果の画像、そして新たにパラメータを変更するためのインターフェースが並んだWebページを生成するようにしておく。

データによっては、Webブラウザからダウンロードして、ユーザが使っているPCなどで可視化する場合もある。この場合は、出力されたデータのリストを表示しダウンロードできるようなWebページを生成する。

次節では、本提案システムのシステム構成と現在の実装について説明する。

5.　システム構成

本提案システムの構成は以下の図4のようになる。ワークフローの入力を行うユーザインタフェースとワークフローの実行を制御する管理サーバ、実際にGrid計算機環境でジョブを実行する実行マネージャ、そして、結果の表示やパラメータの変更を行うことのできる結果表示ツールの4つのサービスに分けている。

ワークフローの入力を行うツールはJavaによるアプリケーションとして実現される。一方、結果を表示するツールは、Webブラウザと Webサーバによって構成している。各サービスの主要な機能と役割は以下のようにした。

A．ユーザインタフェース

①ワークフロー入力機能

②ワークフローの実行や停止などの操作

③現在の実行状況の表示

B．管理サーバ

①ワークフローから、プログラムの実行リストの生成

②パラメータ変更や計算機環境に応じてのプログラムの再実行リストや停止リストの作成

③パラメータ変更時の途中結果の保存処理

④実行マネージャを介して、プログラムの実行や停止を制御

⑤結果表示ツールへの結果ファイルの転送

C．結果表示

①Webサーバで管理サーバから結果ファイルを受け取り、結果用のWebページを生成。

②パラメータの変更を受付、管理サーバに転送

D．実行マネージャ

①実際に計算機環境にプログラムの投入を行う

また、本システムにおいて途中結果の判定情報を取り組む仕組みは、判定プログラムがファイルに変数と値を書き込み、それをワークフロー管理サーバが読み込む事で行っている。現在開発中のシステムではファイルの内容は、1 行に〈変数名〉〈値〉と並べた形式にしている。

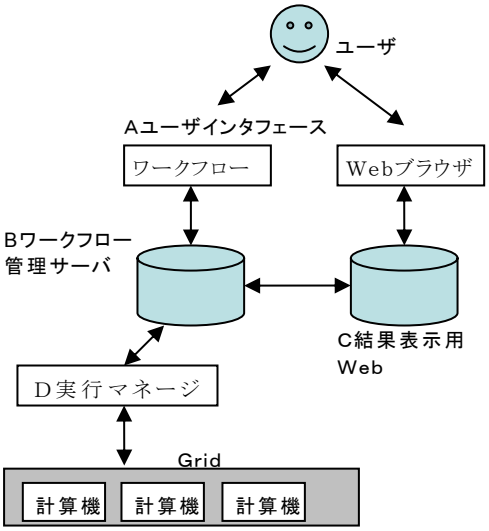

図 4 システム構成

6. 現状と今後の課題

ここでは、現時点でのシステムの実装された機能とその利用例、今後の課題について説明する。

現在の実装した部分で、ワークフローの入力と任意場所からの再実行を行うことができる。結果表示ツール(Webブラウザ)からのパラメータ変更は行うことができないが、ワークフローを入力するユーザインタフェースから、変更を行うことができる。また、ワークごとに、結果ファイルの自動保存を行い、再実行した時にデータを上書きしないようにしている。ここでは、「たんぱく質立体構造予測」においてパラメータを変更した実行例を示す。

処理の流れは、初期処理 → CM(Comparative Modeling)によるデータ検索→FR(Fold Recognition)によるデータ検索→NF(New Fold)による計算処理、となる。CM、FR、NFというのは、プログラムのアルゴリズムの名前である。CM、FRは、既に立体構造の知られているたんぱく質から似たもの探す手法で、NFは物理化学的な計算による手法である。CM、FRで候補がない部分についてNFを行う。そして、NFを実行する場合に、たんぱく質をいくつかに分割して処理するかどうかの問題があるため、複数の分割方法でNFをやり直すことができるようにした。

今回の実行ではアミノ酸配列の長さが 163、つまり 163 個のアミノ酸からなるたんぱく質に対し、52 番目と 53 番目のアミノ酸の間で分割した場合と、分割を行わない場合の実行を行い、両方の結果を比較することができた。この例では、分割を行った方が良い結果であった。

上に示したように、本システムは開発段階である。今後は以下の機能を追加し、効率的な実行環境を目指す。

まず、ユーザインタフェースの部分においては、計算結果の表示とパラメータの変更を統一する仕組みを用意する。また、現在は「たんぱく質立体構造予測」専用に結果表示を行っているため、汎用化していく予定である。

次に、ワークフローの実行に関しては、現在はパラメータをユーザが変更した時に特定の部分からやり直す機能しか実現していない。今後は、4. で説明した利用できる計算機資源に応じた自動処理や、実行時間の見積もり機能などを実現していく予定である。

謝辞

本研究中、「たんぱく質立体構造予測」のプログラムは、神戸大学高田研究室で開発された「Rokky システム」[2]を使わせて頂きました。協力し、プログラムを提供して下さった高田研究室の皆様に深く感謝いたします。

参考資料・文献

[1] Workflow: An Introduction
http://www.wfmc.org/information/Workflow-An_Introduction.pdf

[2] Rokky Protein Structure Suite
http://predict.chem.sci.kobe-u.ac.jp/rokky/

[3] UniCore http://www.unicore.org/

[4] MyGrid http://www.mygrid.org.uk/

[5] Globus http://www.globus.org/

[6] GridAnt: A Grid Workflow System
http://www-unix.globus.org/cog/projects/gridant/

[7] MyGrid Workflow system
http://twiki.mygrid.org.uk/twiki/bin/view/Mygrid/WorkFlow

[8] GGF (Global Grid Forum) http://www.ggf.org/