大規模長時間実行 Grid アプリケーションの実装と評価

武宮　博 [†,††] 田中　良夫 [†,††]
中田　秀基 [†,†††] 関口　智嗣 [†]

動的に計算量が変動するという特徴を持つ大規模シミュレーションプログラムを GridRPC と MPI を組み合わせることにより Grid 化し，長期間にわたって Grid 環境上で継続実行させることを試みた．実装に際して，(1) 大規模計算機を事前に一定期間予約し，対象を変更する，(2) 複数のユーザにより共有されている計算機から，その時点で利用可能なものを動的に利用していく，という二種類の現実的な実行シナリオを想定し，このシナリオを満足するためにアプリケーションに実装すべき機能を検討した．また，本手法に基づき Grid 化したプログラムを環太平洋 Grid 環境上で半月から二ヶ月にわたり動作させることで，本手法により計算量の変動にあわせて動的に実行対象計算機を変更する柔軟性と，障害を検知・復旧する頑健性を実現できることを示す．

Implementation and Evaluation of Large-scale Long-run Grid Applications

Hiroshi Takemiya [†,††], Yoshio Tanaka [†,††]
Hidemoto Nakada [†,†††] and Satoshi Sekiguchi [†]

A large-scale adaptive simulation program has been "gridified" by combining GridRPC with MPI, and executed on the Grid environment for a long time. In the implementation of the program, we considered functions needed to satisfy two different execution scenarios such as (1) using computing resources according to the reservation schedule and (2) using idle computing resources in the shared environment. The long-run exeperiment of the program shows that our programming approach enables applications to be flexible enough to allow dynamic resource allocation and is robust enough to detect and recover from faults.

1. はじめに

1995 年に Grid の概念が提唱されて以来，Global Grid Forum(GGF) 等の活動を通して Grid 基盤の持つべき機能， Grid ミドルウェアの持つべき機能に関して広範な議論が行われ，インタフェースの標準化，実装が推し進められてきた．このような基盤部分に関する研究開発の進展に伴い，大規模な計算機資源を必要とするバイオサイエンス，ナノサイエンス，原子力物理などの応用分野でも Grid に対する大きな期待が寄せられてきている．しかしながら，これらの分野において Grid 技術を適用したアプリケーションを構築した事例はまだ少数にとどまっており，さらに実際に複数のサイトに散在する計算機を連携し，長期にわたって計算を行った例はほとんどない．これは Grid Computing が未だに研究レベルの技術にとどまっており，実用レベルの技術として成熟していないことを意味している．

未だに大規模 Grid アプリケーションの開発，実行が進展していない大きな原因として，実際にアプリケーションを開発，実行する際の方法論が確立していないという点が挙げられる．そのために，多くの応用分野の研究者は Grid Computing に対して高い期待を抱きつつも，自分たちの持つアプリケーションを Grid 化することができずにいる．

我々は，長期にわたり Grid 環境上で実行される大規模アプリケーション（以下，大規模長時間実行 Grid アプリケーション）の開発に適したプログラミングモデルとして，GridRPC と MPI を組み合わせた手法を提案した[3]．本手法は，GridRPC の持つ動的な遠隔実行プログラム実行機能及び障害検知機能と MPI の持つ効率的な並列実行機能を組み合わせるもので，Grid 環境上での長時間実行に必要な柔軟性，頑健性，効率性を満足する大規模アプリケーションを開発するための現実的な手法である．具体的には，Grid 環境上に散在する計算機に対し，MPI によって並列化されたシミュレーションプログラムを GridRPC により

† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
†† 科学技術振興機構，CREST
CREST, Japan Science and Technology Agency
††† 東京工業大学
Tokyo Institute of Technology

動的に起動し，クライアントプログラムを介して連携させる.[3] ではさらに，典型的な大規模長時間実行アプリケーションを上記の手法に基づき Grid 化し，日米に散在するクラスタから構成される Grid 環境上で最大 1793CPU を利用して実行することで，本手法に基づき Grid 化したプログラムが実際に大規模実行可能であることを示した．また，プログラムの実行中に人為的に障害を発生させ，本手法が種々の障害を検知可能であることを示した．

上記の実験は数時間から数日程度の期間で実施されており，大規模計算機を占有利用することが可能であったが，大規模 Grid アプリケーションを数週間から数ヶ月間継続実行することを想定すると，特定の計算機を占有利用することは現実的でない．それをふまえて，我々はまず大規模アプリケーションを Grid 上で長時間実行するためには，どのようなシナリオが現実的であるかを検討する．また，そのシナリオに沿って実行を行うためには，アプリケーションにどのような機能が必要とされるかを検討する．さらに，実際に長期にわたる実験を行い，シナリオの現実性や実装機能の有効性を評価する．

今回実験対象に用いる Adaptive Hybrid QM/MD シミュレーションコードは，前稿で用いた材料シミュレーションプログラム[1] に対して，より精度の高い計算を短時間で実行するために系の状況にあわせて動的に計算領域の再定義を行うという特徴を付加したものである．再定義により計算量が動的に変化するため，必要な計算機数が変化する．系の状況にあわせて必要な領域において詳細な計算を行う adaptive アプローチはこれまで多くの分野で行われているが，従来多く用いられている MPI を使ったプログラム開発では，計算量の変化に柔軟に対応することが困難であった．

我々の目的は，上記のような特質を持つ大規模 Grid アプリケーションの長時間実行を期待しているアプリケーションユーザに対し，実装の際の指針を示し，考慮すべき事項に関する情報を提供することにある．

2. 大規模 Grid アプリケーションの長時間実行

2.1 GridRPC と MPI の組み合わせによるプログラミング

我々の想定する大規模長時間実行 Grid アプリケーションは，数十から数百プロセッサを用いて並列に実行される複数の処理が互いに疎に連携しつつ，数週間から数ヶ月に及ぶ長期間の計算を必要とするものである．このような処理を実現するためには，アプリケーションに以下の 3 つの特質が要求される．

(1) 柔軟性

Grid 環境を構成している各サイトの計算機は複数の利用者によって共同利用されており，ジョブの実行時間が制限されているのが一般的である．また，突発的な障害が発生してダウンしたり，定期的にメンテナンスが行われたりする．そのような場合，動的に実行対象計算機を変更して処理を継続する柔軟性が求められる．

(2) 頑健性

単一計算機での実行と比較して，Grid 環境での計算は計算機やネットワークに障害が発生する可能性が高い．さらに，長時間の実行を想定するとその確率はますます高くなる．したがって，大規模長時間実行 Grid アプリケーションはネットワークや計算機の障害を検出し，障害から自動的に復旧する頑健性が求められる．

(3) 高効率性

大規模長時間実行 Grid アプリケーションでは多数の計算機を利用するため，それらを効率的に管理できることが求められる．また，それら計算機の効率的利用，計算機間の効率的通信も実現する必要がある．

我々が提案している GridRPC と MPI を組み合わせるプログラミングモデルは，GridRPC を用いて MPI によって並列化された遠隔実行プログラムを動的に起動し，連携する．GridRPC の持つ動的な遠隔実行プログラム機能と多様な障害検知機能を利用してプログラムの柔軟性および頑健性の実現を支援し，MPI の持つ遠隔実行プログラム内での効率的な並列処理機能と GridRPC の遠隔実行プログラム管理機能を組み合わせることにより高効率性の実現を支援する．

2.2 実行シナリオ

1. で述べたように，大規模長時間実行 Grid アプリケーションを現実的な環境で実行することを考えた場合，数ヶ月にわたって計算機を占有し，継続して利用することは困難である．各サイトの計算機は複数の利用者によって共同利用されており，ジョブの実行時間も制限されているのが一般的である．また，突発的な障害が発生してダウンしてしまうことがあるほか，定期的にメンテナンスが行われる可能性もある．

したがって，動的に実行対象計算機を変更しながらシミュレーションを継続していく必要がある．このような環境下で大規模シミュレーションを長期にわたり継続的に実施するための現実的な実行シナリオとして，我々は以下の 2 つのシナリオを検討した．

(1) 予約ベースの実行

事前に計算機を予約しておき，予約スケジュールに基づき実行を行う．ただし，全ての計算機を計算期間全体にわたって予約することはできず，一部の時間のみ利用可能であるとする．シミュレーションは予約スケジュールに従って動的に計算機を変更しながら継続実行される．

(2) 遊休計算機の動的利用

事前に予約することはせず，動的に遊休計算機を利用し計算を継続する．ただし，計算機は複数の利用者によって利用されていることから，一定時間継続して計算機を利用したらその計算機の利用を終了し，別の計

算機を利用することを試みる．

予約ベースの実行は，数百〜数千規模の CPU を持つ大規模計算機の利用を想定している．このような規模の計算機を事前予約せずに利用できる可能性は低い．一方，遊休計算機の動的利用は，数十〜百程度の CPU を持つ小中規模の計算機の利用を想定している．また，予約ベースの実行では，限られた時間範囲内ではあるが，計算機を占有して利用できることを想定している．それに対し，遊休計算機の動的利用では，一旦利用できたとしても再度利用を試みた場合に利用できる保証はない．これら想定している環境の違いは，3 章で述べる利用対象計算機の選択戦略に影響を及ぼす．

3. 大規模長時間実行アプリケーションの Grid 化

本章では，Adaptive Hybrid QM/MD シミュレーションコード及び GridRPC の実装系である Ninf-G の概要を説明した後，2. で述べた実行シナリオに沿って動作する大規模長時間実行 Grid アプリケーションの実装方針，実装詳細について述べる．

3.1 Adaptive Hybrid QM/MD シミュレーションコード

Adaptive Hybrid QM/MD シミュレーションコードは，材料工学分野における次世代電子デバイスやマイクロマシンの設計において，亀裂発生等ナノスケール特性を解析するために開発されたプログラムである．

本コードは，Density Functional Theory (DFT) に基づく詳細な QM (Quantum Mechanics) シミュレーションと経験的原子間ポテンシャルを用いた古典 MD (Molecular Dynamics) シミュレーションを組み合わせることによって，現実的な時間内に大規模系の挙動を精度よく解析可能としている．系全体の振る舞いは計算量の少ない MD シミュレーションにより計算され，亀裂発生の可能性のある領域等，詳細な解析が必要な領域の振る舞いは QM シミュレーションにより計算される．QM 領域は利用者の興味に従って複数定義可能で，各々独立に計算され，MD シミュレーションを介して間接的に連携する．QM シミュレーションと MD シミュレーション間ではデータ交換が必要であるが，計算量と比較して通信量はわずかであることから，Grid 環境上での効率的な実行が期待される．

また，本シミュレーションでは，系の挙動に従い計算実行中に QM 領域の再定義が行われ，QM 領域の数や領域に含まれる原子数が変化する．再定義は QM 領域の拡張，QM 領域の分割の 2 段階で行われる．QM 領域の拡張では，MD 領域の境界域に含まれる原子の位置を調べ，QM 領域に含まれる原子に充分近接している場合は QM 領域に加える．これは，時間進展に伴う系の状態変化に対応し，計算精度の悪化を防ぐためである．一方，QM 領域の分割では，定義した QM 領域の複数領域への分割可能性を調べる．QM シミュレーションの計算量は QM 領域に含まれる原子数の 3 乗に比例するので，領域を分割して個々の領域に含まれる原子数を少なくすれば総計算量が減少し，計算時間を節約できるためである．この再定義が終了すると，各 QM 領域に含まれる原子数に基づき必要とされる計算機の CPU 数が決定される．原子数と必要とされる CPU 数の換算は，ユーザがあらかじめファイルに記述しておいた変換ルールに従って行われる．

3.2 Ninf-G システム

Ninf-G[2] は GridRPC の参照実装の一つであり，Globus Toolkit 上に構築されたミドルウェアである．Ninf-G は Grid 環境での RPC 実現のために種々の機能を提供しているが，以下では大規模長時間実行 Grid アプリケーションの実装に必要な Ninf-G の機能について，柔軟性の実現及び頑健性の実現を支援する機能を中心に説明する．

- 柔軟性の実現支援機能

GridRPC において遠隔実行プログラムは必要に応じて動的に起動される．したがって，柔軟に実行対象計算機を変更することが可能である．また，長期にわたり実行を行う場合には，当初想定していなかった計算機を追加することになったり，利用可能プロセッサ数が予約期間によって変わる等，計算機の利用に関する情報にも変更が発生する場合がある．Ninf-G では計算機に関する種々の属性の設定を動的に変更可能にすることによって，そのような状況に柔軟に対応できる．属性の変更は，属性設定ファイルの再読み込みを行う `grpc_config_file_read_np()` 関数，あるいは遠隔実行プログラム起動時に属性を動的に設定する `grpc_handle_attr_set_np()` 関数を用いることで実現される．

- 頑健性の実現支援機能

Ninf-G では RPC 実行における障害検知のために 2 種類の機構を提供している．1 つはクライアントプログラムと遠隔実行プログラム間のネットワーク接続の切断に基づく障害検知，もう一つは RPC の各動作（初期化，実行，終了等）に関する時間制限 (タイムアウト) に基づく障害検知である．

遠隔実行プログラムが起動されると，クライアントプログラムとの間で TCP 接続が確立され，それを維持しながら RPC を実現する．ネットワークあるいは計算機の障害が発生すると，この接続が切断され，それにより Ninf-G は遠隔実行プログラムの異常を検知する．

また，Grid 環境を構成する計算機やネットワークは複数の利用者により共有されているため，動的にその負荷は変動する．そのため，バッチシステムを介して遠隔実行プログラムを起動しようとしても計算機が他の利用者によって使用されているため長時間待たされたり，ネットワーク性能が低下してデータの転送に長時間を

要することもある．Ninf-G ではこれも広い意味での障害と考え，利用者が設定した種々のタイムアウト時間を越えて処理が終了しない場合はエラーを返す．具体的には，起動時間タイムアウト，実行時間タイムアウト，ハートビートタイムアウトの 3 種類のタイムアウト時間を設定可能となっている．実行タイムアウトに関しては，さらに細かな設定が可能となっており，計算機の継続実行時間制限 (`job_maxTime`, `job_maxCpuTime`, `job_maxWallTime`), および一回の RPC 実行時間制限 (`session_timeout`) を個別に設定することができる．

3.3 Adaptive Hybrid QM/MD シミュレーションコードの Grid 化

まず，Grid 化する際の基本方針としてシミュレーションプログラム自体の変更を最小限にとどめるために，長期実行を実現するための機能をスケジューリングレイヤとして独立に作成し，シミュレーションプログラムからはそのレイヤの提供する API を介して Ninf-G システムにアクセスする構造を採用した．プログラムの全体構造を図 1 に示す．

スケジューリングレイヤの実装に際しては，Adaptive Hybrid QM/MD シミュレーションの特徴を活かしつつ Grid 上で長期にわたり実行を継続するために，以下の点を考慮した．

- 実行対象計算機変更の契機

実行対象計算機の変更契機は，(1) ネットワークあるいは計算機システムの障害が原因で，現在対象にしている遠隔実行計算機が利用できなくなった場合，(2) 予約時間あるいは継続利用時間の制限を超えた場合，(3) 計算規模が変化し，現在利用している計算機の持つ資源では計算できなくなった，あるいは他にさらに適した計算機が存在する場合，の 3 種類に大別される．これらの契機を検知するために，以下の実装を行った．

まず，(1) に関しては Ninf-G の提供する種々の障害検知機能を利用することとし，Ninf-G 用の設定ファイルに設定を行った．設定事項の動的な変更に対処するために，QM シミュレーション実行の際には必ず `grpc_config_file_read_np()` 関数を呼び出して設定ファイルの再読み込みをすることにした．

また，(2) に関しても，継続利用時間の制限に関しては，Ninf-G の提供する継続実行時間制限に関するタイムアウト機能を利用するが，予約時間のチェックに関しては，アプリケーションレベルで設定ファイルを作成し，情報を設定することにした．Ninf-G 用の設定ファイルと同様，本設定ファイルも QM シミュレーション実行の際に再読み込みされ，利用期間制限を越えていないかどうかのチェックを行う．

さらに，(3) に関しては，従来は終了処理の一環として QM 領域の再定義処理を行っていたが，これを時間進展のループ内に入れ込み，一定回数の時間進展を行う毎に QM 領域の再定義を行うことで計算規模が変化したかどうかのチェックを行うことにした．

- 計算機の選択戦略

(1) に述べた実行対象計算機変更の契機が発生した場合，何らかの方針で次に利用する計算機を選択する必要がある．選択戦略には種々のものが考えられるが，今回は実行シナリオに対応した 2 種類の選択戦略を実装することにした．

予約ベースの実行の場合，代替利用可能な余剰計算機を多数用意することはないであろう．したがって，障害が発生した場合にも予約した計算機の再利用を試みる．ただし，規定回数試みても再利用可能とならない場合にのみ，他の計算機を利用を試みる．

障害発生や予約時間の終了等の理由で実行対象計算機を変更する場合，あるいは QM 領域の計算規模が変化した場合には，予約計算機が占有利用可能であることから積極的に実行対象計算機の変更を行うこととする．具体的には，ある QM 領域の計算対象となっている計算機が変更される場合に，その領域だけではなく全ての QM 領域の計算規模をチェックするとともに予約されている計算機の計算性能を比較し，規模の大きいものから順に計算性能の高い計算機に再割り付けを行う．本戦略は，実行の効率性を重視した戦略である．

遊休計算機の動的利用の場合，障害発生時に障害の発生した計算機を再利用する利点はあまりない．占有利用されていないため，再利用を試みても他のユーザにより利用される可能性があるからである．したがって，この場合は計算機の安定性を重視した選択戦略を採用する．すなわち，全ての利用可能な計算機に対し過去の実行履歴を保持しておき，過去の実行履歴において最も障害発生の少ない計算機を優先して選択する．また，実行対象計算機を変更する場合にも，一つの QM 領域の実行対象計算機の変更を契機として全ての QM 領域の変更を検討することはせず，その領域を担当する計算機のみの変更を行うこととした．これも，計算機を変更した場合に利用できる保証がないことから，現在利用できているものを継続利用したほうが得策であると考えたためである．

- 障害発生時の対処

図 1 に示すように，Adaptive Hybrid QM/MD シミュレーションコードは計算の進行に伴い 7 種類の状態を遷移する．障害が発生した場合には，その時点の状態から QM シミュレーション起動待ち状態に遷移し，順次状態を回復していくことで障害発生状態まで回復していく．そのために，スケジューリングレイヤでは，シミュレーションプログラムが与える初期処理データ及び QM シミュレーション実行用入力データを保持しておき，障害復旧の際に利用する．

- 利用計算機数が不足した場合の対処

複数の QM 領域の計算は全て独立に実行可能であることから，通常これらの計算は異なる計算機に割りあてられ並列に実行される．しかし，長期実行を行っている場合には，利用可能な計算機の数が QM 領域の

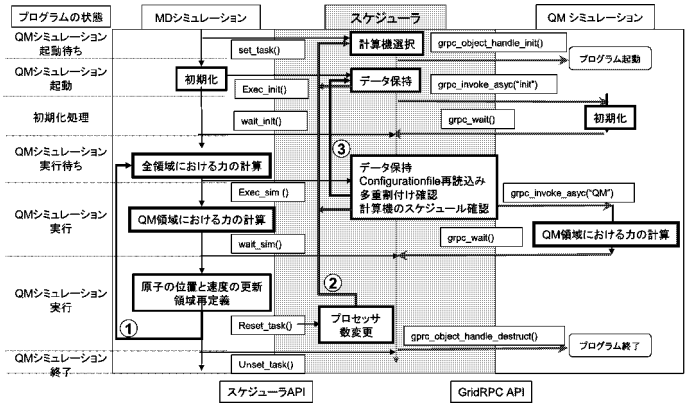

図 1 Grid 化 Adaptive Hybrid QM/MD コードの構造
図中①はアプリケーションの時間進展ループ，②は障害発生時の状態遷移，③は多重割り付け時の状態遷移を表す

数を下回ることがありうる．その場合，同一計算機に複数の QM 領域の計算を割り付け，逐次的に実行することを考える．

これを実現するための最も直感的な方法は，異なる領域計算のたびに遠隔実行プログラムを起動しなおし，初期化して領域計算を行うことである．しかしながら，この手法は起動コストが毎回発生するし，動的に遊休計算機を利用するシナリオでは，せっかく獲得した計算機を利用できなくなる可能性がある．また，これまでの経験から大規模並列プログラムを起動する際は計算実行時と比較して障害が発生する確率が高い．これらの理由から，遠隔実行プログラムは起動したまま異なる領域の計算を実行することにした．

ただし，異なる MPI プロセス数を要する領域の計算が割り付けられる可能性がある．利用プロセス数の異なる計算を割付け可能とするために，大域的な MPI のコミュニケータ (`MPI_COMM_WORLD`) 内に利用されるプロセスだけで構成されるサブコミュニケータ (`MPI_QM_WORLD`) を計算開始時に毎回作成し，サブコミュニケータ内で QM 領域の計算を行うようにした．

- 計算規模の動的変化への対処

計算規模の変化は，利用する MPI プロセス数の変化として反映される．Ninf-G において起動する MPI プログラムのプロセス数は，`GRPC_HANDLE_ATTR_MPI_NCPUS` という属性をハンドルに対して設定することにより指定できる．

MPI プロセス数を変更するには，一旦遠隔実行プログラムを終了して再起動する必要がある．アプリケーションからの要求にしたがって毎回再起動することは，利用計算機数が不足した場合に考察したように得策ではない．そこで，アプリケーションの要求する MPI プロセス数よりもあらかじめ多めにプロセス数を確保しておき，実際に要求されたプロセス数のみでサブコミュニケータを形成し処理を行わせ，そのプロセス数を超えた時のみ再起動を行うことにした．これにより再起動の回数を減少できる．実際にどの程度多めにプロセスを確保しておくかはユーザが設定ファイルに記述する．

4. 長時間実行実験

4.1 実 験 概 要

実験は，(1) 予約ベースの実行シナリオに基づくものと，(2) 遊休計算機の動的利用シナリオに基づくものの二種類を行った．

(1) の実験では， SIMOX(Separation by Implantion of Oxygen) と呼ばれる IC チップ製造技術のシミュレーションをおこなった．SIMOX は，高速に加速された酸素原子をシリコン基盤に入射し SIO2 絶縁層を形成する技術である．シミュレーションでは，シリコン基板上に入射位置の異なる 5 つの酸素原子を配置し入射させ，酸素原子と基盤内のシリコン原子の挙動を調べる．本シミュレーションでは，個々の酸素原子及びその周辺のシリコン原子が QM 領域として毎回動的に定義され，酸素原子がシリコン基盤の内部に侵入するにつれて QM 領域に含まれる原子の数は徐々に増加する．また，シミュレーション開始直後の QM 領域数は 5 つであるが，酸素原子とシリコン原子の相互作用の結果によっては，1 つの QM 領域が 2 つに分割されたり，複数の QM 領域が 1 つに統合されたりする．したがって，本シミュレーションでは QM 領域の数も大きさも動的に変化する．

(2) の実験では， 原子レベルの摩擦シミュレーションを行った．本シミュレーションは，ナノスケールのシリコン探針をシリコン基盤に接触させつつ移動させた場合の接触領域の酸化過程を調べるものである．接触領域には水分子や OH 分子を配置し，その周囲の領域を動的に QM 領域として定義し計算を実行する．シリコン探針が時間とともに基板上を移動するため，本シミュレーションでも QM 領域の大きさが動的に変

表 1 SIMOX シミュレーションにおける動的割り付け結果

原因	発生回数	同一計算機割付け 成功	同一計算機割付け 失敗	別計算機割付け 成功	別計算機割付け 失敗
起動時障害	712	34	706	12	18
実行時障害	58				
領域変更	244	13	7	195	29

化する．また，シリコン探針の形状の変化により QM 領域の数も動的に変化する．

4.2 実験結果

二種類のシミュレーション各々に関して，主に柔軟性及び頑健性の観点から結果を評価する．

4.2.1 予約ベースの実行

予約ベースの実行においては，国内及び TeraGrid[4] に設置された計 8 台のクラスタを事前に予約して，仮想的に 128 プロセッサを有する計 15 台のクラスタとみなし，産業技術総合研究所の P32 クラスタのみを利用する予備的計算フェーズ 9 日間を含め 19 日間にわたってシミュレーションを行った．

本計算に先がけて利用計算機のスケジュールを行った．スケジュールの際には，動的な QM 領域数の増加や障害発生に伴う利用計算機数の変化に備えて，5 台の (仮想) クラスタに加え 1 台の予備クラスタをスケジュールし，領域数の増加に備えることにした．スケジュールは大きく 5 つのフェーズに分類される．

フェーズ 1（2 日間）： 産業技術総合研究所 AIST Super Cluster(P32, M64 及び F32 クラスタ，以下 ASC) を用いる

フェーズ 2（3 日間）： ASC 及び NCSA と Pittsburgh Supercomputing Cneter(PSC) に各々設置された TeraGrid クラスタ 2 台を用いる

フェーズ 3（3 日間）： ASC 及び南カルフォルニア大学 (USC) のクラスタを用いる

フェーズ 4（2 日間）： ASC，東京大学及び東京工業大学のクラスタを用いる

本実験では，19 日間で計 270 ステップの計算が行われた．また，10 日間の本計算において最長 5 日間にわたって継続実行することができた．実験結果詳細を表 1 に示す．

実験期間中には計 1014 回の動的割り付けが発生した．そのうち，障害の発生により行われた割り付けは 770 回にのぼった．これらのうち，初期の遠隔実行プログラムの起動に失敗したケースが 712 回と大半を占める．失敗の主な原因は，Globus の job manager が起動されていなかった，MPI プロセス間の通信用メモリ資源の確保ができなかった，バッチキューが active になっていなかった，ディスク容量制限を越えてしまい起動に必要なファイルの生成ができなかった等である．

これらの障害が発生した場合，再割り付けを試みても成功する確率は低い．表からわかるように，障害発生時に同一計算機への割り付けを試みて成功する確率は 4.6%((成功 34 回，失敗 706 回) 程度である．今回採用した計算機選択アルゴリズムでは，規定回数失敗するまで同一計算機に対して再起動を試みる戦略を採用したが，規定回数を多く設定しても有効ではないことがわかった．

また，一台の計算機を仮想的に複数の計算機とみなして利用した場合，上記のような障害が発生すると全ての仮想計算機が一斉に遠隔実行プログラムの起動に失敗することになる．このような場合，正常に動作している計算機に対して多重に QM 領域の計算が割り付けられ逐次的に処理されるが，今回は計算効率を重視して一旦シミュレーションを中断し障害要因を除去した後，手動で再起動した．実験期間中の再起動回数は 22 回にのぼる．したがって，このような事態を避けるためには，可能な限り計算機の仮想的な分割利用は避けることが望ましい．

実験中発生した障害において，Ninf-G の提供するタイムアウト機構により 110 回の障害が検知された．また，アプリケーションレベルで実装された予約期間超過に関するタイムアウトに関しても，期間の超過を検出しその時点で予約されている別の計算機へと実行対象が変更されることが確認された．

タイムアウトに関する問題点としては，一回の QM 計算に関する計算時間のタイムアウト (session timeout) の有効性が挙げられる．典型的な QM 計算は，一回の計算に数時間を要し，かつ収束計算であるため計算時間が大きく変動する．このような特質を持つ計算に対してタイムアウトを設定する場合，タイムアウトの値として非常に大きな値を設定せざるを得ない．しかしながらタイムアウトにより検知される障害は，計算の初期の段階で発生する可能性もある．実際，今回検知されたセッションタイムアウトは，計算初期にディスク容量制限を越えてしまい，計算が止まってしまったことが原因であった．また，この事例では Ninf-G の提供する heart beat 機能も有効に機能しなかった．通常，ディスク容量制限を越えるとアプリケーションが書き込みに失敗し異常終了することでエラーを検知できるが，今回の場合は何らかのタイミングでアプリケーションがブロックしたままとなってしまいエラーが検知されなかった．また，heat beat 機能はアプリケーションとは異なるスレッドで実行されており，そのために heart beat 送信が中断しなかったものと考えられる．詳細は現在究明中である．

このような状況は，QM 計算プログラムが収束計算の状況を定期的にクライアントプログラムに通知する等，アプリケーションレベルで heart beat 機能を実装することで検知可能である．Ninf-G には遠隔実行プログラムが実行中にクライアントプログラムの関数を呼び出す call back 機能が実装されており，この機能を利用することにより上記問題を解決できると考えられる．

次に，QM 領域の再定義により 47 回にわたり領域

表 2 Friction シミュレーションにおける動的割り付け結果

原因	発生回数	同一計算機割付け 成功	同一計算機割付け 失敗	別計算機割付け 成功	別計算機割付け 失敗
起動時障害	126	3	104	27	31
実行時障害	39				
領域変更	27	2	0	21	4

原子数が変化し，その結果実行対象計算機の再選択が 244 回発生した．このような変化に対して，シミュレーションプログラムは自動的に MPI プロセス数を変更して遠隔実行プログラムの再起動を行い，シミュレーションを継続実行することができた．

4.2.2 遊休計算機の動的利用

遊休計算機の動的利用シナリオに基づく実験では，国内及び PRAGMA[5] テストベッドに設置された計 12 台の計算機を対象として， 52 日間にわたってシミュレーションを実行させた．

本シミュレーションでは 52 日間で計 386 ステップの計算が行われ，最長 22 日間の継続実行を実現することができた．実験結果詳細を表 2 に示す．

本シミュレーションでは，192 回の動的割り付けが発生した．そのうち 165 回が障害の発生が契機となって行われたものであり，SIMOX シミュレーション同様，同一計算機への再割り付けの失敗回数が非常に多い (104 回)．過去の実行履歴から障害発生件数の少ない計算機を選択するという計算機選択戦略に反する結果となったのは，上記 104 回の失敗のうち 100 回が最も多数の CPU(128CPU) を要する QM 計算の割付けに失敗して連続して発生したためである．障害発生時，128CPU を有する計算機は 1 台しか利用可能でなかったため，他の計算機への回避が不可能であった．このように他の計算機を選択する余地がない場合を除き，本実験では 52 日間に 2 回の手動による起動を行うだけで実行することができた．

5. 考　　察

今回の実験を通じて得られた知見および考察をまとめる．

(1)　より細かな非均質性への対応の必要性

グリッドミドルウェアはハードウェアやオペレーティングシステムなどの非均質性を考慮して設計されているが，今回の実験において，より詳細なシステム設定，環境における非均質性への対応が必要であることが明らかになった．

- 外部との通信路の確立手段について

Ninf-G はクライアントとサーバの通信路を確立する際には，サーバからクライアントに対して接続している．計算ノードにプライベート IP アドレスが割り当てられている場合でも NAT などの機能を用いれば外部に対して接続可能であるし，Firewall の設定においても外部への接続が許可されていれば問題ない．しかし，PSC のクラスタにおいては，外部に対してネットワーク接続を行う場合，アプリケーションゲートウェイ (AGW) と呼ばれるノードを介した通信を行う必要があり，Ninf-G で PSC のクラスタにサーバプロセスを起動する際には Globus の GRAM 呼び出しを行うコマンド列に `(agw_count=#)` という Resource Specification Language (RSL) 属性を渡してやる必要があった (# は確保したい Gigabit Ethernet インタフェースの数)．Ninf-G は Globus の RSL 仕様で規定されている属性を渡す機能は提供しているが，この `agw_count` という属性は PSC が独自に規定・利用している属性であるため，Ninf-G がこの属性を渡すことはできなかった．今回の実験においては外部との通信を行うノードは rank0 のプロセスが起動されるノードだけであったたため，AGW を介さずに外部との直接の通信が可能な特殊なノードが rank0 に割り当てられるような特殊な設定をしてもらうことにより対応した．この経験に基づき，最新の Ninf-G には任意の RSL 属性を扱う機能が実装されている．

また，USC においては Firewall の設定が非常に厳格であり，外部に対する接続も許可されなかったため，今回の実験では，全ての通信をこちらのクライアントホストに中継する特殊な Proxy ノードを提供いただいた．この場合，起動するサーバプロセスに対して実際のクライアントホストではなくこの Proxy ノードを接続先のクライアントホストとして通知する必要がある．Ninf-G はサーバに対して接続先のクライアントホスト名を明示的に指定する機能を提供しているが，複数のサーバを利用する場合に各サーバごとに異なるクライアントホストを指定する事ができなかったため，この機能を利用することができなかった (他のサーバに対してもこの Proxy ノードがクライアントホストとして見えてしまうため)．今回の実験においては，計算ノードの hosts ファイルに実際のクライアントホスト名と Proxy ノードのアドレスを対応づけるエントリを追加することによりこの問題を回避した．最新の Ninf-G にはサーバごとにクライアントホスト名の指定を可能とする機能が実装されているが，このような厳格な Firewall で守られた計算機の利用については，Ninf-G 自体に通信を中継する機能を実装するといった本格的な対応が必要である．

- その他細かな非均質性について

バッチシステムの最大ジョブ実行時間，大きなファイルを生成して良いディスクパーティション，ディスククオータ，などの細かな項目についてサイトごとに相違があり，各クラスタごとにファイルの生成先やジョブを起動する際にバッチシステムに渡すオプションなどを変更・指定する必要があった．エンドユーザがこれらの項目をいちいち指定して利用することは現実的ではなく，実行環境を仮想化する事によってこれらの相違点を吸収して RPC を行うといった新しい技術の

開発が必要である．

(2)　グリッドオペレーションの自動化の必要性

複数のサイトに跨る実験を行う際，TeraGrid は Web インタフェースを通じてクラスタノードの予約を申請し，申請が承認された場合に各サイトの管理者がバッチシステムの設定を変更して占有利用に対応している．しかし，今回の実験に際しては予約しているはずなのに実際には予約時刻が来ても投入したジョブがキューがアクティブにならないという現象が NCSA のサイトで数回，PSC のサイトでも 1 度確認された．いずれも管理者の作業ミスによるものであったが，設定を手作業で行う以上ミスが起る可能性は残る．今後複数サイトでのアプリケーションの実行をより使いやすく確実なものとするためには，予約システムの自動化，つまり複数サイトにまたがる計算機の事前予約を実現するスケジューラの実現が急務である．また，TeraGrid は我々ユーザからの要求受け付け窓口を設置しており，ここにメールを送ると内容に応じて適宜各サイトの管理者に伝達される仕組みになっている．また，NCSA のように 24 時間サポートをしているサイトにおいても，例えば 8 時間勤務を 3 交代で行うといった運営をしているため，要求を受け取った担当者と実際に作業を行う担当者が異なる場合があり，管理者間で情報をきちんと伝える必要がある．しかし今回はこの情報の伝達に不備があり，我々が何度かメールで確認したにもかかわらず対応されなかったという問題が何度かあった．グリッド基盤を円滑に管理・運営するためには，このような情報の伝達が確実に行われるような仕組みを確立する必要がある．

6. ま と め

動的に計算規模が変化するという特性を持つ大規模シミュレーションプログラムを GridRPC と MPI を組み合わせるプログラミングモデルに基づくことにより Grid 化することで，Grid 環境上で長時間実行なプログラムを実装できることを示した．実装に際しては，予約ベースの実行と遊休計算機の動的利用という 2 種類の実行シナリオを検討し，それらのシナリオに沿って長時間実行を行うためにアプリケーションに実装すべき機能，計算機選択戦略に関して考察した．また，実際に Adaptive Hybrid QM/MD シミュレーションコードを題材として環太平洋 Grid 環境上で長時間継続計算実験を行うことにより，我々のアプローチの有効性を検証した．長期間の継続計算に関しては，最長 22 日間の継続実行を実現し，過去の履歴に基づき安定した計算実績を持つ計算機を優先して利用する選択戦略が有効であることがわかった．

しかしながら，実用レベルで大規模長時間実行 Grid シミュレーションを実施するには解決すべき問題が数多く残っている．例えば，単に再起動を行うという単純な戦略では一旦発生した障害の復旧は困難であり，現段階ではユーザの介入による問題点の解決が必要である．より洗練された障害復旧の機構を考案することも重要であるが，より現実的な Grid 環境の運用管理形態を検討していくことがさらに重要である．5. で述べたように，今回の実験において発生した障害の原因は，運用管理技術及び組織間の情報伝達技術の未熟さに起因するものが多い．また Grid 環境に特有な問題として，計算機が広域に分散しているために，あるサイトの計算機に障害が発生した場合，原因がわかっているにもかかわらず時差の関係でサイト管理者と連絡がとれず，解決が遅れるということも発生した．このような場合，他の計算機に多重に計算を割り付けることで実行を継続するということが有効であることはわかったが，根本的な解決法ではない．広域な Grid 環境をどのように運用管理すればよいかを検討し，明確化することが急務である．そのために，今回のような実験を積み重ね，発生する障害に関する情報を蓄積していくことが重要である．

謝辞

本研究に際し，Hybrid QM/MD コードを提供いただいた名古屋工業大学 尾形 修司助教授に感謝いたします．また，本研究の遂行にあたり貴重な御助言をいただいた南カリフォルニア大学 中野 愛一郎助教授に感謝いたします．

また，SC2005 における実験に協力し計算資源を提供いただいた TeraGrid Exective Commitee メンバ諸氏，Pittsburgh SuperComputing Center, National Center for Supercomputing Applications, University of Southern California, 東京大学田浦研究室，東京工業大学松岡研究室に感謝いたします．

参 考 文 献

1) Ogata, S., Shimojo, F., Kalia, R., Nakano, A., Vashishta, P.:‘Hybrid Quantum Mechanical/Molecular Dynamics Simulations on Parallel Computers: Density Functional Thery on Real-space Multigrids, computer Physics Communications, Vol. 149, pp.30–38 (2002)
2) 武宮博，田中良夫，中田秀基，関口智嗣：“Ninf-G2:大規模 Grid 環境での利用に即した高機能，高性能 GridRPC システムの実装と評価”，情報処理学会論文誌コンピューティングシステム，Vol.45, No.SIG11(ACS 7), pp.144–159 (2004)
3) 武宮博，田中良夫，中田秀基，関口智嗣：“MPI と GridRPC を利用した大規模 Grid アプリケーションの開発と実行：Hybrid QM/MD シミュレーション”，SACSIS 2005, IPSJ Symposium Series Vol.2005, No.5, pp.153-160,(2005)
4) http://www.teragrid.org
5) http://www.pragma-grid.net