Grid：広域分散並列処理環境での高精度分子シミュレーション —— C_{20} 分子のレプリカ交換モンテカルロ

池上 努[†] 武宮 博[†] 長嶋雲兵[†]
田中良夫[†] 関口智嗣[†]

高精度なポテンシャルエネルギー関数を用いた分子シミュレーション環境を構築するため，一連のエネルギー計算をグリッド上に分散して実行するライブラリ（GON ライブラリ）を作成した．GON ライブラリは，グリッド上の複数の並列計算機に粗粒度のジョブを非同期に発行し，また各ジョブを細粒度並列に実行することで，グリッド上の計算資源を統合的に利用する手段を提供する．ライブラリは計算手法の変更に柔軟に対応でき，かつ利用者から見てリモートでの実行をなるべく隠蔽する方向で設計した．このライブラリを用い，エネルギー計算に分子軌道法を直接，用いたレプリカ交換モンテカルロシミュレーションプログラムを作成し，小さなフラーレンである C_{20} 分子に応用した．テスト計算では 150 時間にわたってほぼ安定に計算を続行し，最大 16 個の計算サーバに分散した 860 個の CPU を同時に使用した．

Accurate Molecular Simulation on the Grid —— Replica Exchange Monte Carlo Simulation for C_{20} Molecule

Tsutomu Ikegami,[†] Hiroshi Takemiya,[†] Umpei Nagashima,[†]
Yoshio Tanaka[†] and Satoshi Sekiguchi[†]

The new library was developed to perform a high precision molecular simulation by distributing energy calculations over the grid. Those calculation jobs are submitted asynchronously to parallel machines on the grid, while each job is executed in a fine grained fashion. The library obscures the remote execution from users, yet is flexible enough to be adopted to various calculation methods. The library was used in the replica exchange Monte Carlo (REXMC) simulation with ab initio molecular orbital (MO) method for the energy calculation, and was tested on a small fllerene molecule, C_{20}. In the course of 150 hours of the test run, the maximum of 860 CPUs over 16 calculation servers were operated simultaneously, showing the grid capability to perform a large scale simulation that cannot be run on a single computer.

1. はじめに

近年，様々な産業分野において，ナノ材料を用いた精密製品の設計や新規物質の創成が強力に押し進められている．また，医学・生理学の分野でも，生体高分子や DNA レベルの研究が中心的な課題に据えられるようになった．これらナノスケールの領域では，計算機シミュレーションが研究に果たす役割はけっして小さなものではない．実験的な測定手段の限界もあいまって，分子レベルでの大規模かつ精密なシミュレーションの必要性は，今後ますます高まっていくと予想される．

分子動力学法やモンテカルロ法に代表される分子シミュレーションでは，数千から数十万回に及ぶ力場計算が計算時間の大半を占める．計算コストを抑えるため，一般的なシミュレーションでは力場関数として，経験的に定められた簡便な関数のセットを用いることが多い．特に生体高分子の分野では，AMBER や CHARMM などの優れた力場関数が提案され，広く用いられている．一方，活性分子種や金属微粒子，フラーレン類など，既存の力場関数が存在しない系や，そもそも簡便な関数形が期待できない系も数多く存在する．これらの系について，信頼性の低い力場関数のもとでシミュレーションを実行すると，科学的にナンセンスな結果を導きかねない．そこで近年，分子軌道法などの高精度手法により，力場を直接計算する分子シミュレーションが行われるようになってきた．

† 産業技術総合研究所グリッド研究センター
National Institute of Advanced Industrial Science and Technology

分子軌道法を用いた力場計算は，一般の力場関数と比較して桁違いの計算機資源を要求する．最近の計算化学パッケージには並列処理に対応したものが多く，並列計算機を用いることで個々の力場計算に要する時間（wall clock time）を短縮することができる．しかし，計算には多量のデータ通信をともなうのでグリッド上での並列処理には適しておらず，またその並列化効率は扱う系や計算手法に強く依存する．一般に分子軌道計算の並列化効率は並列度の上昇にともなって比較的，早い段階で飽和してしまうので[1]，単体の並列計算機を用いて力場計算の高速化を図る限り，シミュレーションを現実的な時間で解くのは困難である．このため，シミュレーションのアルゴリズムそのものを見直し，複数の力場計算を独立・並行に実行するような設計が必要となる．そのような方法の 1 つ，レプリカ交換モンテカルロ（REXMC）法[2] は複数のモンテカルロ計算を緩く結合して緩和を加速する手法であり，各モンテカルロ計算に含まれる力場計算は，ほぼ独立に実行可能である．このようなアプローチは特に，複数のサイトに散在する並列計算機を統合した仮想的な並列計算機クラスタでの実行に適している．すなわち，力場計算を個々の並列計算機上で並列処理し，さらに各力場計算を複数の並列計算機を用いて同時に実行することで，シミュレーション全体の並列化効率が高まり，処理時間が現実的な範囲に抑えられると期待できる．

各力場計算の独立性は高く，データ通信量は最小限に抑えられるので，力場計算の同時実行には，近年のグリッド構築技術の利用が適している．しかし，Globus Toolkit [3] などのミドルウェアを直接利用したシミュレーション環境の構築は，一般の自然科学者にはハードルが高い．そこで我々は，複数のサイトに散在する並列計算機を用いた並列力場計算の同時実行支援を目的として，新たに GON ライブラリを開発した．GON ライブラリはシミュレーションプログラムを開発する際に，グリッド上の資源を用いて簡単に力場計算の実行制御を行うためのプログラム群である．本稿では，シミュレーションのアルゴリズムに REXMC 法を採用し，GON ライブラリを用いて高精度な分子シミュレーション環境を構築した事例を紹介する．以下，GON ライブラリの設計方針とその実装を示し，次いで REXMC シミュレーションへの応用例について述べる．

2. GON ライブラリの設計方針

我々は，典型的な大規模系分子シミュレーションが以下のように進行すると想定している．まず，利用者の手元にある計算機（クライアント計算機）上でシミュレーションプロセスが立ち上がる．シミュレーションは複数の力場計算を，十数〜数十台の並列計算機（バックエンド計算機）を用いて同時に実行する．個々の力場計算は各バックエンド計算機上で，さらに十数〜数十程度の CPU を用いて並列処理される．各力場計算が終了すると，シミュレーションプロセスはその結果に基づいて新たな力場計算を準備する．この処理を数千〜数万回繰り返すことで，1 つのシミュレーションが終了する．力場計算は並列計算機を用いても数分から数十分かかるので，グリッド上の計算資源を用いるとしても，1 回のシミュレーションに数日から数週間を要する．このような複雑かつ大規模な処理を，グリッドに関する詳細な知識を要求せずに実行可能とすることを目的とし，以下の諸点を考慮して GON ライブラリを設計した．

バッチキューシステムを模した API の提供

計算化学パッケージの典型的な使用パターンは，(1) 入力ファイルを準備し，(2) ジョブ内容を記述したスクリプトを作成して，(3) バッチキューに登録する，という経過をたどる．シミュレーションプログラムの開発者がこのような利用形態に慣れ親しんでいることを想定し，GON ライブラリでは同等な処理形態のサービスインタフェースを提供することにした．すなわち，GON ライブラリはバッチキューシステムインタフェースに基づいた API を提供する．

スケジューリング機能の提供

一般にバックエンド計算機の性能は均質ではないので，力場計算の所用時間は使用する計算機に応じて変化する．したがって，力場計算を効率よく処理するには，バックエンド計算機の稼働状況を管理してジョブを発行するスケジューリング機能が必要になる．そこで GON ライブラリには内部的なキューを用意し，力場計算が計算インテンシブであることを前提に，キュー内のジョブを逐次バックエンド計算機に割り付けていく pure self scheduling アルゴリズムを採用することにした．

バックエンド計算機の動的追加/削除機能の実現

典型的な分子シミュレーションは，グリッド上の計算機資源を用いても数日から 1 カ月程度の計算時間を必要とする．この間，一定の計算機資源を常に使えるとは限らない．一部のバックエンド計算機が定期的なメンテナンスのために停止したり，計算機利用やネットワークの負荷が一時的に高まって利用に適さなくなることがありうる．この問題を解決するため，バック

エンド計算機の利用管理機能を持ったブックキーパーサブシステムを用意し，利用可能な計算機の動的な追加/削除を可能とした．ブックキーパーの詳細については次章で述べる．

シミュレーションの実行制御機能の提供

分子シミュレーションは一般に，途中結果を解析してパラメータを変更したり実行を放棄するなどのステアリング操作をともなう．また，パラメータを変えて複数のシミュレーションを走らせ，様子を見ることも少なくない．そこでブックキーパーを利用したシミュレーションの一時停止/再開機能を提供するとともに，複数のクライアント計算機が同じバックエンド計算機群を共有する仕組みを提供することにした．これにより，1つのシミュレーションを中断しても他のシミュレーションがバックエンド計算機を引き継いで使用することができ，利用効率が向上する．

標準グリッドミドルウェアの採用

クライアント計算機とバックエンド計算機間のデータ転送や計算の発行には，グリッド環境における標準的な技術を用いることが望ましい．このような機能を提供するグリッドミドルウェアが現在いくつか開発されているが，GON ライブラリはそのうちの 1 つ，Ninf システム[4] を用いて実装した．Ninf システムはクライアント–サーバモデルに基づく GridRPC[5] の実装の 1 つで，グリッド上のサーバに実装された手続き（サーバプログラム）をクライアントプログラムから容易に呼び出す機構を提供する．Ninf システムではサーバプログラムの動的な呼び出しが可能なので，上で述べたブックキーパー機能の実現と親和性が高い．MPICH-G[6] などのメッセージパッシングを実現するグリッドミドルウェアでは，実行開始時にバックエンド計算機を固定してしまうため，利用計算機の動的な変更が困難となる．

ファイルの自動転送機能の実現

計算化学パッケージの機能は多岐にわたり，計算に必要な入力パラメータや結果の出力は，パッケージの種類だけでなく計算対象や計算手法にも依存する．このため，クライアント計算機とバックエンド計算機の間で転送されるデータは，シミュレーションの種類に応じて数も形式も変化しうる．Ninf システムは IDL（Interface Description Language）で記述した関数定義を基に引数データの転送を行うが，化学計算パッケージの提供する全機能に対して関数定義を記述するのは現実的でないし，かといってシミュレーションごとに固有の IDL を記述するのでは柔軟性に欠ける．そこで GON ライブラリでは，計算化学パッケージとのデータの受渡しをすべてファイルを介して行うこととした．その際，あたかもすべての処理がクライアント計算機の上で行われたかのように擬装するため，クライアント側で準備した入力ファイルをバックエンド計算機へ自動的に転送し，計算の終了を待って出力ファイルをクライアントに転送する機能を実装した．

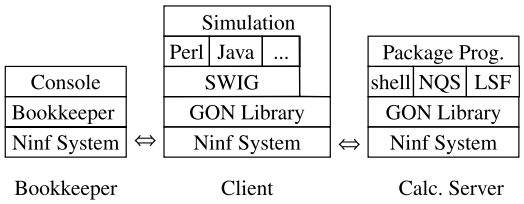

図 1 GON ライブラリの構成図
Fig. 1 Structure of GON library.

3. GON ライブラリの実装

3.1 構　　成

全体のシステム構成を**図 1** に示す．

GON ライブラリは，クライアント計算機上の GON クライアント，バックエンド計算機上の計算サーバ，およびブックキーパーの各サブシステムから構成される．この 3 者が連係して動作し，複数サイトに設置された計算機上で力場計算が非同期に実行される．

3.2 GON クライアント

GON クライアントには，利用者に対してバッチキューシステムインタフェースに基づく API が提供される．主要なサービスインタフェースは以下のとおりである．

`gon_setup()` GON ライブラリの初期化を行う．引数としてブックキーパーの URI を渡す．

`gon_submit()` 計算サーバで実行するジョブを登録する．引数として，計算に必要なファイル類を格納したディレクトリとそこで実行すべきコマンドライン引数，およびジョブの優先準位を渡す．返り値としてジョブ ID を返す．

`gon_wait_any()` `gon_submit()` で投入したジョブのいずれかの終了を待ち，そのジョブの ID を返す．

`gon_submit()` によりジョブが内部的なキューに登録されると，GON クライアントはブックキーパーに対して計算サーバの割当てを要求する．ブックキーパーから計算サーバが割り当てられると，`gon_wait_any()` はキューの中から優先準位に従ってジョブを選び，サーバ上でジョブを実行して結果を受け取る．これらの API は C 言語用に実装されているが，SWIG[7] で作成したインタフェースを介し，Perl などのスクリプト言語や Java 言語から利用することができる．

3.3 計算サーバ

計算サーバは，GON クライアントから転送された入力データを受け取り，指定されたプログラムを実行する．プログラムの実行には，その計算機の運用形態に応じてインタラクティブな実行，あるいは NQS や LSF などのキューシステムを介したバッチ処理を行う．実行終了後，生成された出力ファイルを GON クライアントに転送する．

3.4 ブックキーパー

ブックキーパーは GON クライアントによる計算サーバの利用を管理し，サーバの排他利用や動的な追加/削除を実現する．つねにグリッド環境上の任意の場所で動作し，事前に登録された計算サーバのリストを保持する．シミュレーションが開始され，GON クライアントとの接続が確立すると，クライトからの計算サーバ取得要請に応じてアイドル状態にあるサーバを渡す．GON クライアントは取得した計算サーバでのジョブが終了し次第，ブックキーパーに対してそのサーバがアイドル状態になったことを通知する．

ブックキーパーはまた，力場計算を制御するパラメータの一部を環境変数の形で保持することができる．環境変数のセットは計算サーバごとに別個に管理され，ジョブ実行時に各サーバに伝播される．

複数の GON クライアントが同じブックキーパーに接続すると，登録された計算サーバ群はクライアント間で共有され，排他的に利用される．現在のところ，特定の GON クライアントを優先するようなスケジューリングは行われず，全クライアントには均等に計算サーバが割り当てられる．なお，`gon_submit()` の引数として渡す優先準位は，GON クライアント内でのみ考慮される．

ブックキーパーは利用者との対話的なインタフェースを有し，計算サーバの割付け状況をモニタしたり，特定サーバの割付けの停止や GON クライアントからの懇請の無視を指示することができる．この機能により，一部のバックエンド計算機をメンテナンスのため停止したり，特定のクライアント計算機上のシミュレーションを安全に中断することが可能となる．

3.5 利 用 例

Perl スクリプトで書いたシミュレーションプログラムの例を**図 2** に示す．また，処理の流れを GON ライブラリ内部の動きとあわせて**図 3** に図示した．図中，GON クライアントの太線が，スクリプトの処理の流れに相当する．

まず初めに `gon_setup()` を呼び出し，ブックキーパーとの接続を確立する．次に，独立に実行される力

```
#!/usr/bin/perl
use GON;

Gon_setup("ninf://uri_to_Bookkeeper");
@JobList = setup_job();

for (@JobList) {
  $id = Gon_submit($_->{Nice}, $_->{Dir},
                   $_->{ARGV});
  $Job{$id} = $_;
}

while ($id = Gon_wait_any()) {
  $_ = delete $Job{$id};
  after_care($_);
  next if $_->{End};
  $id = Gon_submit($_->{Nice}, $_->{Dir},
                   $_->{ARGV});
  $Job{$id} = $_;
}
```

図 2 GON ライブラリの使用例：Perl から利用する場合
Fig. 2 Example Perl script for the usage of GON library.

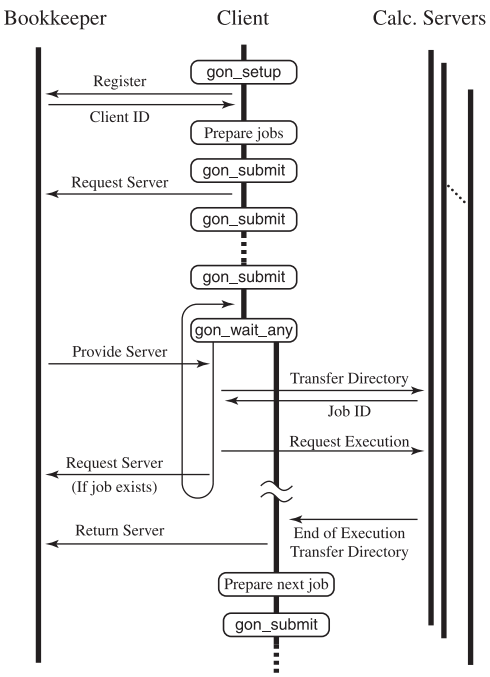

図 3 GON ライブラリの内部処理の流れ
Fig. 3 Typical workflow of GON library.

場計算ごとに専用のディレクトリを作成し，計算サーバで実行するジョブの内容を記述したスクリプトや入力ファイル類を格納する．仮想的な関数 `setup_job()` はこうした一連の前処理を行い，準備したジョブのリストを作成する．

準備したジョブは `gon_submit()` で GON ライブラリのキューに登録される．その際，ジョブの優先準位，

格納ディレクトリとあわせて，計算サーバ上で実行されるコマンドラインを引数として渡す．なお，シミュレーション側からはジョブがどの計算サーバで実行されるか，前もって知ることができないので，各サーバ上におけるコマンドラインレベルの互換性を，利用者の責任で確保しておかなければならない．

ジョブが投入されると，GON クライアントはブックキーパーに対して計算サーバの割当てを要求する．この要求はブックキーパー側で計算サーバが利用可能となるまでブロックされるため，非同期に発行される．ブックキーパーから計算サーバが提供されると，GON クライアントはそのサーバに `gon_submit()` で指定したディレクトリをまるごと転送し，次いでジョブの実行依頼を非同期に発行する．

`gon_wait_any()` は登録されたジョブのどれかが終了するまで待ち，計算サーバ側で変更を受けたディレクトリを GON クライアントに書き戻す．仮想的な関数 `after_care()` は終了したジョブの結果を検分し，次のステップで必要とされる力場計算用のジョブを準備する．再構成されたジョブは，シミュレーションの終了条件が満たされない限り GON ライブラリのキューに再登録される．

GON クライアントと計算サーバ間のデータの転送はディレクトリ単位で行われるので，クライアント側で `gon_submit()` で指定するディレクトリよりも上位に位置するファイルへのアクセスは，サーバ側では保証されない．また通信量を減らすため，力場計算の過程でディレクトリ内に作成される一時作業用のファイルは，計算終了時に計算サーバ側で削除しておくことが望ましい．

4. 応 用 例

4.1 計 算 手 法

SC02 において日米欧にまたがって構築された Metacomputing テストベッド[8] を利用し，GON ライブラリを用いて実際に高精度な分子シミュレーションが実行可能であることを示した．その際，ブックキーパーの機能として，

(1) 計算サーバの動的な追加/削除，
(2) 複数の GON クライアント間の調停，

が有効に働くことを確認した．

シミュレーションのアルゴリズムにはレプリカ交換モンテカルロ法（REXMC 法）[2] を用い，これを GON ライブラリを用いて広域分散処理した．おおまかな処理の流れを**図 4** に示す．REXMC 法では，対象となる分子のレプリカを複数，生成して異なる温度を与え，

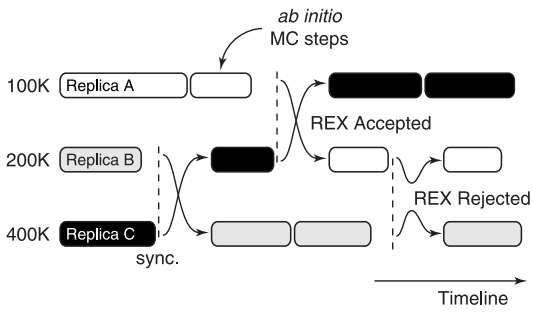

図 4 レプリカ交換モンテカルロ（REXMC）法の処理の流れ
Fig. 4 Workflow of replica exchange Monte Carlo (REXMC) method.

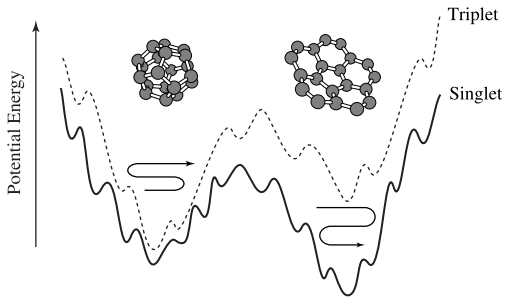

図 5 C_{20} 分子のポテンシャルエネルギー面の模式図
Fig. 5 Schematic potential energy surfaces of C_{20} molecule.

並行してモンテカルロ計算を実行する．その際，温度の隣接するレプリカ間で規定のステップごとに同期をとり，エネルギーを比較して統計的に温度を交換することで，平衡状態への緩和を加速する．すべてのレプリカの同期を一度にとることはないので，計算サーバの利用効率は高くなる．また，処理の遅れているレプリカの力場計算を優先的に実行し，対となるレプリカの同期待ち時間を低減している．なお，図では各モンテカルロステップごとに同期をとっているが，実際のシミュレーションでは 5 ステップごとに同期をとり，温度交換の評価を行った．

計算の対象にはフラーレンの一種，C_{20} 分子を選び，力場計算に分子軌道法を用いた．C_{20} 分子の代表的な構造とそのポテンシャルエネルギー面の模式図を**図 5** に示す．C_{20} 分子は計算の困難な系で，計算量の少なくて済む半経験的方法や密度汎関数法では，ポテンシャルエネルギー曲面を正確に再現できない[9]．より高精度な計算と比較した結果，電子一重項基底状態については RMP2/6-31G(d) 理論を選んだ．一般にフラーレン分子の電子的安定性は電子三重項状態とのエネルギー差より評価することができるので，三重項についても一重項と独立に REXMC 計算を走らせた．

表 1 C_{20}の REXMC 計算に利用した計算機のリスト
Table 1 Machines used for the REXMC simulation of C_{20}.

Site☆1	Machine	Mul☆2	Comm☆3	NProc☆4	NCPUs☆5
AIST	1 SGI Origin 2000	1	MPI	16	32
	2 Linux Cluster (Dual PentiumIII 1.4 GHz)	1	Socket	22	44
JAERI	3 Compaq Alpha Cluster (ES40)	1	Shmem	16	16
	4 Compaq Alpha Cluster (ES40)	11	Shmem	64	704
	5 Fujitsu PrimePower	3	MPI	16	96
PSC	6 Compaq Alpha Cluster (ES45)	1	Shmem	256	256
CFS	7 Cray T3E	1	Shmem	64	64

☆1 From top to bottom: National Institute of Advanced Industrial Science and Technology, Japan Atomic Energy Research Institute, Pittsburgh Supercomputing Center, and Computation for Science. ☆2 Multiplicity of batch queue. ☆3 Communication mechanism for parallel ab initio calculation. ☆4 Number of calculation processes run in parallel. ☆5 Total number of CPUs used for the calculations.

三重項については参照できる高精度な理論計算が存在しないが，エネルギー準位を一重項と比較することから，同レベルの理論 ROMP2/6-31(d) を用いた．

力場計算には SPECchem96[10] でも用いられている計算化学パッケージ GAMESS[1] を使用した．個々の分子軌道計算の並列処理には GAMESS に実装されている機能をそのまま用い，GON ライブラリを用いて各レプリカの力場計算を同時に実行した．

4.2 計算機環境

今回のテストに用いたバックエンド計算機の機種，ジョブの多重度（Mul），力場計算の並列処理に用いられる通信機構，並列度数（NProc），および計算に用いた CPU 数（NCPUs）を**表 1** にまとめる．多重度は独立に実行可能な GAMESS ジョブの数，並列度は GAMESS が立ち上げる計算プロセスの本数である．GAMESS は MPI および Socket を用いた並列化において，計算プロセスに加えて通信用のプロセスを立ち上げるので，たとえば MPI で NProc = 16 の場合には計 32 個の CPU が使用される．GAMESS の並列度数はブックキーパー側で一括管理する環境変数として設定した．また，表 1 の計算サーバを常時使用していたわけではなく，時間帯によって利用可能なサーバを切り替えて用いた．GON クライアントとブックキーパーは，AIST 内の別の計算機の上で運用した．

多重度の合計値 19 が力場計算に利用できる計算サーバ数の上限である．今回は一重項と三重項の 2 本のシミュレーションでそれぞれ 16 個のレプリカを生成したので，合計 32 個のレプリカが最大 19 個の計算サーバを奪いあう形で計算が進行した．

4.3 予備的性能評価

Metacomputing テストベッドを用いたシミュレーション計算に先立ち，各バックエンド計算機について通信性能と力場計算の実行時間を計測した．

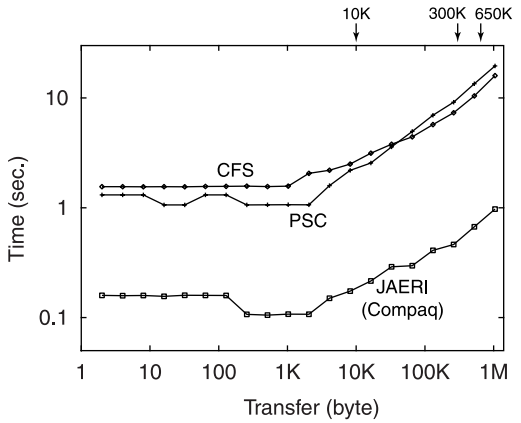

図 6 クライアント計算機とバックエンド計算機間の通信性能
Fig. 6 Data transfer performances between a client and calculation servers.

通信時間の評価

力場計算にともなう通信時間を見積もるため，Ninf システムを直接用いてクライアント計算機とバックエンド計算機間の pingpong テストを実施した．実測値を**図 6** に示す．通信にともなうオーバヘッドのため，転送量が 2 KB 未満では通信時間に差がでなかった．

1 回の力場計算にともなう通信量は一重項と三重項の場合で異なる．一重項の計算では計算サーバ側で分子軌道計算の出力結果を処理しており，通信量は上り下りとも 10 KB 程度に収まっている．一方，三重項の計算は分子軌道計算の収束性に問題があるため，前回の計算結果を利用して入力ファイルを作成しており，行きで 300 KB，帰りで 650 KB の通信が生じる．図 6 より，通信に費やされる時間は一重項の計算で 4 秒以内，三重項状態で最大 20 秒程度と推測される．

クライアント計算機とブックキーパーの間には 1 回の力場計算あたり 2 回，それぞれ数バイトの通信が発生する．両者は一般に同一サイト内に設置されるので，

表 2 力場計算 1 回あたりの所要時間（分）．括弧内の数字は下限を推測した値．計算サーバの番号は表 1 の 2 列目の数字を用いた

Table 2 Time required for one ab initio energy calculation in unit of minute. Estimated lower bounds are parenthesized. The server indices are listed in the second column of Table 1.

Server	NProc	Singlet	Triplet
1	16	10.20	14.55
2	22	6.10	12.55
3	16	8.13	14.80
4	64	(2.03)	(3.70)
5	16	22.32	55.90
6	256	(0.45)	–
7	64	8.20	–

ブックキーパーとの通信時間は計算サーバとのそれに比べて無視してさしつかえない．

力場計算時間の評価

各計算サーバ上で力場計算を実行し，計算に要する時間を見積もった．分子軌道法による力場計算は非線型方程式の数値解法を含むため，その計算時間は分子軌道の収束性に依存して一定ではない．比較的，収束の良い条件を選び，シミュレーションで用いる並列度数のもとで実計算時間（Wall clock time）を測定した結果を**表 2** に示す．事前の測定ができなかった部分については，計算時間の推測値を括弧内に記した．たとえば PSC 一重項の計算時間は，32 並列での計算時間を単純に外挿して求めている．32 並列未満での観察より，CPU 数の倍増に対する計算速度の改善は 1.6〜1.8 倍程度なので，推定値は計算時間の下限と考えてよい．

計算時間と比較すると通信時間はおおむね無視することができ，REXMC シミュレーションの広域分散化にともなうコストは小さいと期待される．

4.4 実 行 結 果

今回のテストでは，1 つのブックキーパーを共有して一重項と三重項のシミュレーションを 2 本，独立に走らせた．この構成では，2 本のうち 1 本をメンテナンスのために停止しても，残りのシミュレーションプロセスが余った計算サーバを引き継いで利用することができる．このため，計算サーバを効率よく利用することができ，150 時間にわたってほぼ途切れることなく計算を実行した．テストは日本時間で 2002/11/18 14:00 から 2002/11/24 20:00 にわたって行った．

対話型のインタフェースを通じてブックキーパーを操作し，適宜，利用可能な計算機資源を切り替えたので，テスト期間中，使用した CPU の数は一定していない．平常時は 1〜3 個の計算サーバ上で合計 50〜100 個の CPU を使用し，ピーク時には最大で 16 サーバ，860 個の CPU を使用した．

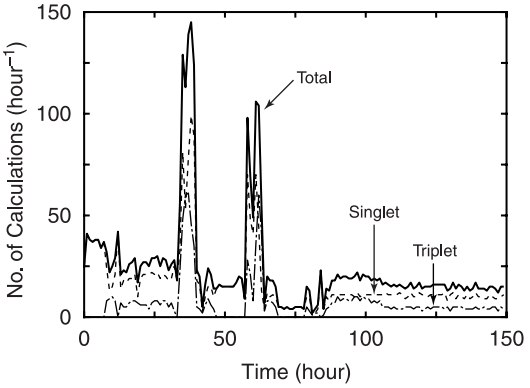

図 7 力場計算の処理頻度の時間変化

Fig. 7 Time variation of the number of the processed energy calculations during the test run.

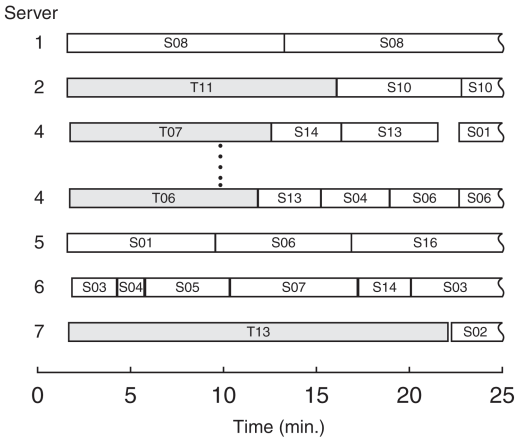

図 8 計算サーバへのジョブの割付け状況．計算サーバの番号は表 1 の 2 列目の数字を用いた．ジョブは S/T が一重項/三重項，それに続く数字がレプリカの通し番号を表している

Fig. 8 Running status of the calculation servers. The server indices are listed in the second column of Table 1. Each box corresponds to an ab initio energy calculation.

力場計算の実行回数を 1 時間ごとに集計し，時刻の関数としてプロットしたものを**図 7** に示す．図中，全計算回数を実線で，一重項と三重項の計算回数をそれぞれ破線と一点鎖線で表示した．計算サーバは 2 本のシミュレーションに均等に割り付けられるが，力場計算は三重項のほうが重いため，三重項の計算点数が全般的に少なくなっている．2 個の大きなピークは JAERI Compaq 機（表 1 の 4 番）を利用した期間にあたり，この間，1 時間に最大で一重項 99 点，三重項 46 点，合計 145 点の力場計算を処理した．

実際の計算におけるマシンの占有状況を，ある時間

帯を切り取って**図 8** に図示した．箱の 1 つ 1 つが力場計算に対応しており，たとえば PSC 機（6 番）を見ると 1 回の一重項計算に要する時間は 1〜7 分の間に分布していることが分かる．この計算時間は表 2 で推測した下限値よりもかなり大きく，個々の力場計算の並列度を上げるアプローチではシミュレーションの高速化に限界があることを示している．1 回の力場計算に要する時間は，実時間で数分〜十数分と通信時間を大幅に上回っており，遠隔地の計算サーバを用いる場合についても，通信のコストはおおむね無視してさしつかえない．また，個々の力場計算のつど，計算サーバの取得と返却を行っているので，各レプリカの力場計算が必ずしも同じサーバで計算されるとは限らない．このため，シミュレーションに影響を与えることなく，計算サーバを追加/削除することができた．

5. 結　　語

世界各地の並列計算機をグリッド上で結合し，高精度な力場関数を用いた大規模分子シミュレーションを実施した．一連の力場計算を各計算サーバ上で並行に処理するため，Ninf システム上に GON ライブラリを作成して用いた．個々の力場計算は，さらに計算サーバ内で粒度の細かな並列化のもとで実行される．このような階層的並列化を採用したことで，多数の CPU を効率良く運用することができた．また，ブックキーパーを利用することで，計算サーバの切替えや共有を可能とした．今回のテストでは，2 本のシミュレーションプロセスが計算サーバを共有し，平均 50〜100 個の CPU を使って，6 日間ほぼ途切れることなく計算を続行した．利用した CPU の数は，ピーク時で 860 個に達した．

GON ライブラリを他のシミュレーションプログラムと結合することで，様々なアプリケーションがグリッド上で実行可能となる．実際の利用経験を踏まえ，今後，以下のような改善を進める予定である．

- セキュリティの強化．Ninf-G へ移行する．
- ブックキーパーの改良．計算サーバのリストを動的に変更可能にする．また，スケジューリング機能を組み込む．
- 強制終了機能．シミュレーションを異常終了する際，計算サーバで実行中の JOB の終了を待つのではなく，能動的に中断する．

謝辞 計算機資源を提供していただいた，以下の各機関に感謝します．日本原子力研究所，Pittsburgh Supercomputing Center，Computation for Science, Korea Institute of Science and Technology Information.

参 考 文 献

1) Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.H., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S.J., Windus, T.L., Dupuis, M. and Montgomery, J.A.: General Atomic and Molecular Electronic Structure System, *J. Comp. Chem.*, Vol.14, pp.1347–1363 (1993). 並列化についてはパッケージ付属の PROG.DOC に詳しい．
2) Sugita, Y., Kitao, A. and Okamoto, Y.: Multidimensional replica-exchange method for free-energy calculation, *J. Chem. Phys.*, Vol.113, pp.6042–6051 (2000). and references there in.
3) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *Int. J. Supercomputer App.*, Vol.11, pp.115–128 (1997).
4) Nakada, H., Sato, M. and Sekiguchi, S.: Design and implementations of Ninf: towards a global computing infrastructure, *Future Generation Computing Systems*, Vol.15, pp.649–658 (1999).
5) Seymour, K., Nakada, H., Matsuoka, S., Dongarra, J., Lee, C. and Casanova, H.: Overview of GridRPC: A Remote Procedure Call API for Grid Computing, *Grid Computing — Grid 2002*, LNCS, Vol.2536, pp.274–278 (2002).
6) Foster, I. and Karonis, N.T.: A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems, *Proc. SC'98* (1998).
7) Simplified Wrapper and Interface Generator. http://www.swig.org
8) Presentation at SC2002 in Baltimore. http://www.hlrs.de/news-events/2002/sc2002
9) Grimme, S. and Mück-Lichtenfeld, C.: Structural Isomers of C20 Revisited: The Cage and Bowl are almost isoenergetic, *Chem. Phys. Chem.*, Vol.2, pp.207–209 (2002).
10) Standard Performance Evaluation Corporation. http://www.specbench.org

(平成 15 年 1 月 24 日受付)
(平成 15 年 5 月 28 日採録)

池上　努

昭和 42 年生．平成元年東京大学理学部化学科卒業．平成 3 年東京大学大学院理学系研究科修士課程修了（化学専攻）．平成 4 年より慶應義塾大学理工学部化学科助手，岡崎国立共同研究機構分子科学研究所助手，Universidad de Puerto Rico ポスドクを経て，平成 14 年 6 月より独立行政法人産業技術総合研究所グリッド研究センター非常勤職員．博士（理学）．少数多体系の研究に従事．日本化学会，日本物理学会各会員．

武宮　博（正会員）

日立東日本ソリューションズ（株）公共ソリューション本部サイエンス&テクノロジーセンター主任研究員．昭和 61 年東北大学大学院理学研究科天文学博士前期課程修了．平成元年同博士後期課程中退．同年日立東日本ソリューションズ（株）入社．平成 14 年より産業技術総合研究所グリッド研究センターへ派遣．

長嶋　雲兵（正会員）

昭和 30 年生．昭和 58 年北海道大学大学院理学研究科博士後期課程化学第二専攻修了．理学博士．同年，岡崎国立共同研究機構分子科学研究所電子計算機センター助手．平成 4 年お茶の水女子大学理学部情報科学科助教授，平成 8 年同教授，平成 10 年通産省工業技術院物質工学工業技術研究所基礎部理論化学研究室長．平成 11 年同産業技術融合領域研究所計算科学研究グループ長，平成 13 年独立行政法人産業技術総合研究所先端情報計算センター情報基盤研究開発室長，平成 14 年同グリッド研究センター統括研究員．筑波大学連携大学院大学教授．計算化学，情報化学，大規模数値計算，広域分散並列処理の研究開発に従事．日本化学会，IEEE，応用数理学会，コンピュータ化学会各会員．

田中　良夫（正会員）

昭和 40 年生．平成 7 年慶應義塾大学大学院理工学研究科後期博士課程単位取得退学．平成 8 年技術研究組合新情報処理開発機構入所．平成 12 年通産省電子技術総合研究所入所．平成 13 年 4 月より独立行政法人産業技術総合研究所．現在同所グリッド研究センター基盤ソフトチーム長．博士（工学）．グリッドにおけるプログラミングミドルウェア，計算ポータル，およびテストベッド構築に関する研究に従事．IC'99 論文賞．ACM 会員．

関口　智嗣（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 59 年筑波大学大学院理工学研究科修了．同年電子技術総合研究所入所．情報処理アーキテクチャ部主任研究官．以来，データ駆動型スーパーコンピュータ SIGMA-1 の開発等の研究に従事．平成 13 年独立行政法人産業技術総合研究所に改組．平成 14 年 1 月より同所グリッド研究センターセンター長．並列数値アルゴリズム，計算機性能評価技術，グリッドコンピューティングに興味を持つ．市村賞受賞．日本応用数理学会，ソフトウェア科学会，SIAM，IEEE 各会員．