2003－HPC－94 (6)
2003／6／13

重力計算専用計算機GRAPE-6のリモートアクセス環境

小沼賢治[†] 朴泰祐[††] 佐藤三久[††]
高橋大介[††]

我々は，複数の他サイト上に分散した汎用並列計算機群と専用並列計算機を結合し，各種物理現象の複合シミュレーションを可能とするシステムを構築するため，重力計算専用計算機 GRAPE-6 クラスタを中心とする計算宇宙物理学用のシステムのリモートアクセス環境となる HMCS-R を実装した．HMCS-R の目的は複数のクライアントで GRAPE-6 を安全に効率良く共有することであるため，複数の並列クライアントの管理，robustness を上げるための種々の機能の追加，およびクライアントの処理スケジューリングや重力計算の入出力フェーズの効率化を図った．複数のクライアントで GRAPE-6 サーバへ同時にアクセスした場合の，それぞれのクライアントに生じる待ち時間を測定した結果，上記の効率化のために行った処理の効果が出ていることが確認できた．HMCS-R により，GRAPE-6 を他サイトから利用するための基礎が完成した．

Remote accessing environment of GRAPE-6 gravity engine

Kenji Onuma,[†] Taisuke Boku,[††] Mitsuhisa Sato[††]
and Daisuke Takahashi[††]

We have implemented HMCS-R which combines general purpose MPPs and special purpose one to realize large scale multi-physics simulations. The prototype of HMCS-R is specially dedicated for computational astrophysics with a gravity engine GRAPE-6. HMCS-R is well designed to handle multiple client MPPs (general purpose ones) efficiently to maintain the utilization ratio of GRAPE-6 very high. For this purpose, the server system with GRAPE-6 cluster is designed to handle multiple connections with well scheduled protocol as well as providing several features for robustness on the system. Through the performance evaluation with synthetic clients under various conditions, it is confirmed that HMCS-R works very efficiently making multiple clients to work without any stress by other clients. As a result, HMCS-R provides an ideal environment to enable the flexible remote access to GRAPE-6 from other sites.

1. はじめに

計算宇宙物理学において，重力計算は最も基本的で重要な計算であり，N 個の粒子に対して $O(N^2)$ の計算量が必要となる．この計算を大規模な並列計算機やクラスタなどを使用したとしても，大きい N に対して十分な計算速度で実行するのは難しい．そこで，この問題を高速に解くための手段として重力計算専用計算機 GRAPE-6 が開発された[1]．GRAPE-6 は重力計算用に内部通信網やパイプライン等が専用化された計算機であり，そのピーク性能は 1 ボードあたり 1TFLOPS に達している．またボード数を増やすことにより，ほぼ理想的に性能のスケーラビリティを得ることができる．

宇宙物理学において重要な問題である銀河形成は従来，重力のみを粒子間相互作用としてシミュレーションされてきたが，最近の研究により重力以外の複数の作用が無視できないことが分かり[2),3)]，より詳細なシミュレーションを行うにはそれらの作用全てを含める必要がある．しかし，GRAPE-6 は重力計算のみを超高速に処理することを目的として作られたものであるため，そのような複雑なシミュレーションを行うことはできない．

そこで，我々は HMCS(Heterogeneous Multi-Computer System) と呼ばれる，ハイブリッド型並列計算機システムを開発してきた[4]．HMCS は，重力計算を行う GRAPE-6 部（クラスタ）と，重力以外の粒子間相互作用を計算する汎用計算機部（汎用並列計算機やクラスタ）を，並列ネットワークで結合したシステムである．従来の HMCS は，ローカルサイト内に閉

† 筑波大学大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

じたシステムであり，同時には 1 つのシミュレーションのみを実行できる．これを HMCS-L(L は Local) と呼ぶ．

GRAPE-6 はその性質上，高価で貴重なものである故，これを保持できるサイトは限られている．また，大規模なシミュレーションを実行する際には，汎用計算機部として超高性能な並列計算機等が必要となってくる．そして，超高性能な計算機を汎用計算機部として使用したとしても，GRAPE-6 での重力計算時間は汎用計算時間の数十分の一で終ってしまい，GRAPE-6 の使用率は低いものとなってしまう場合が多い．

そこで，他サイトにある複数の並列計算機やクラスタを汎用計算機部とし，それらの間で GRAPE-6 クラスタを共有して使用するように HMCS のコンセプトを拡張することが考えられる．この場合，汎用計算機部はクライアント，GRAPE-6 クラスタは重力計算サーバと位置付けられる．しかし HMCS-L の GRAPE-6 サーバは，ローカルサイト内の単一のクライアントからの計算要求を処理することを前提として作られたものであるため，複数のクライアントで安全かつ効率良く共有できるように実装し直す必要がある．HMCS-L のマシン間通信プロトコルを改良し，これらの遠隔アクセス及びマルチクライアントに対応させたものを HMCS-R(R は Remote) と呼ぶ．本稿では，HMCS-R の設計・実装，およびその性能評価を行う．なお，HMCS-R に基づき，実際にグリッド向け RPC を用いて実現された HMCS も同時並行的に設計・実装されており，そちらのシステムは HMCS-G[5](G は Grid) と名付けられている．

2. HMCS(HMCS-L)

HMCS-R の説明に入る前に，HMCS-L について概説する．システム構成は図 1 のようになっている．GRAPE-6 ボードのホスト計算機は IA-32 ベースの PC であり，1 台の PC につき 1 枚の GRAPE-6 ボードが 32bit PCI バスを介して接続されている．そのホスト PC は 8 つあり，クラスタを形成している (図中では簡単のために 4 ノードとしている)．クラスタ内の各 PC は協調動作するよう MPI[6] でプログラムされており，クラスタ全体で 1 つのサーバとして機能する．HMCS では，クライアント側も何らかの並列システムであることを前提としている．また，通信ボトルネックを解消するために，クライアント側並列システムに複数の入出力プロセスが存在することを許す．この場合，それらの通信プロセスがクライアントを代表してデータ通信を行う．よって，このような通信のための代表プロセスのことを，特に agent と呼ぶ．従って，1 つ以上の agent の集合が，サーバから見たクライアントということになる．

システム全体の動作は以下の通りである．まずクライアントの各 agent からサーバの各ノードへ部分粒子データ（質量，座標など）を送る．次に，重力計算には全粒子データが必要となるため，サーバクラスタ内で粒子データを交換し，各ノードに全粒子データを置く．そして，各サーバノードで GRAPE-6 を用いて最初に受け取った分の粒子の重力計算を実行し，それぞれ agent に結果（加速度，ポテンシャル）を返す．得られた重力計算結果を元にクライアント側で他の粒子間相互作用を計算し，再びサーバに計算を依頼する．以上が 1 ステップの動作で，これを何回も繰り返すことでシミュレーションを進めていく．このように，HMCS では並列クライアント対並列サーバの関係により処理の高速化を図っている．

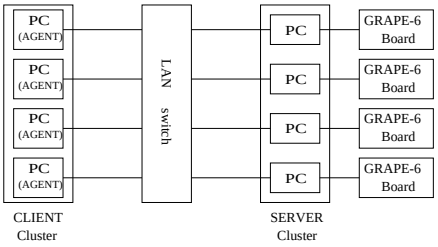

図 1 HMCS-L

3. HMCS-R の設計

GRAPE-6 ボードはホスト計算機に PCI バスを通じて接続され，ホスト計算機上のサーバプログラムによって全てのアクセスを制御しなければならない．GRAPE-6 は単一プロセスにより制御権を占有され，複数の重力計算を同時に実行することができないため，マルチスレッドによる制御に適していない．そのため，単一の GRAPE-6 サーバで複数のクライアントを時分割処理する（マルチクライアント）必要がある．本章では，GRAPE-6 サーバをマルチクライアント化するに当たっての基本設計について述べる．

3.1 複数並列クライアント

HMCS では並列対並列の処理を行うため，複数の agent を持つクライアントがさらに複数存在することになる．このため，GRAPE-6 サーバクラスタの各ノードには複数クライアントの各 agent がバラバラに接続してくる．各 agent から送られてくるのは部分粒子データであり，重力計算には全粒子データが必要となるため，サーバクラスタ内で粒子データの交換をしなければならない．それ故，サーバクラスタ全体で同一のクライアントを同時に処理しなければならない．そこで，どの agent がどのクライアントに属するのかを識別し，サーバクラスタ全体として同一のクライアントを同時に処理するようにする必要がある．

また，HMCS-L ではサーバを立ち上げる際，あらかじめクライアントのエージェント数を指定しなけれ

ばならず，agent 数はサーバノード数の倍数という制約があった．一般的にクライアントの agent 数は不明であるため，この制約を撤廃する必要がある．

3.2 複数クライアント間の影響

GRAPE-6 サーバが 1 つのクライアントを処理している間，他のクライアントはそれが終るのを待っていなければならない．このように，複数のクライアントで GRAPE-6 を時分割で使用する場合には待ち時間が生じる場合があり，クライアント数が多くなるほどその影響は大きくなる．そのため，サーバ側での処理クライアントのスケジューリングや重力計算処理を，クライアント間の影響がなるべく小さくなるようにする必要がある．

3.3 robustness

HMCS-L の GRAPE-6 サーバは，シミュレーションを開始する前にあらかじめ立ち上げ，1 つのクライアントが終了したらサーバも終了する．また，クライアントとの通信等における例外処理にあまり強くない．HMCS-R では複数のクライアントで GRAPE-6 を共有して使用するため，サーバは常に稼働していなければならない．それ故，複数クライアントとの通信等において，サーバの robustness を上げる必要がある．

4. HMCS-R の実装

HMCS では，クライアントとサーバの関係は単純な一対一の関係ではなく，複数対複数の関係を処理しなければならない．このため，HMCS サーバは MPI による並列プログラムで実装される．

4.1 複数並列クライアントの処理

HMCS-R の GRAPE-6 サーバは，並列クライアント（agent 群）が複数存在する状況に対応しなければならない．各サーバノードは，接続してきた agent それぞれに対して個別な処理をするのではなく，全サーバノードで統一して同じクライアントを処理する．

まず，どの agent がどのクライアントに属するものなのかを判別する必要がある．各 agent の所属を一意に判別するための識別子として，ホスト名と key 値を送ってもらうことにした．ホスト名によりクライアントマシンを特定でき，key 値に PID を送ってもらうことで同じホストで複数のクライアントプロセスが走っている場合にも対応できる．この識別子やその他クライアントに関するデータは，各サーバノードで管理テーブルに保存される．

次に，各 agent は非同期にアクセスしてくるため，各サーバノードが単純に TCP/IP の accept() や select() による受信処理を行っていたのでは，全サーバノードで統一して同一のクライアントを処理することができない．そこで，この同期をとるために，サーバノードの一つを代表ノードとし（MPI ランク０），この代表ノードが処理するクライアントを選択し，残りのサーバノードはこの選択に従って同一クライアントを処理する（図 2）．

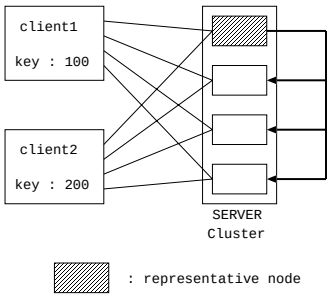

図 2 代表ノードの働き

この方法以外に，クライアント数やその粒子数に応じてサーバノードを空間分割することで，より効率的に処理することも考えられる．しかし，処理スケジューリングが大幅に複雑になるため，そのオーバヘッドを考えると得策ではない．よって，同一クライアントを全サーバノードで一斉に処理する，gang scheduling 的な処理を行う．

4.2 フェーズの細分化

重力計算の 1 ステップには，大きく分けて以下の 3 つの入出力フェーズがある．

- UNIT : クライアントから粒子に関するデータ（粒子数など）を受信する．
- CALC_G6P : クライアントから粒子データ（質量，座標など）を受信し，重力計算をする．
- WAIT : 重力計算の結果をクライアントに返す．

従来の HMCS は単一クライアントを対象としていたため，これらの入出力フェーズを一括して処理していた (図 3(a))．

クライアント側では，サーバが重力計算をしている間に他の計算をすることで，処理の多重化や通信オーバヘッドの隠蔽を図ることがある．重力計算が終るまでにクライアントがサーバに結果を受け取りに来れば問題は無いが，そうではない場合，サーバ側では図 4(a) の斜線部のような停止時間（クライアントが結果を受け取りにくるのを待っている）が生じ，クライアント側では図 5(a) の斜線部のような他のクライアントがサーバを占有している間の待ち時間が生じる．(図 4，図 5 は 2 つのクライアント C1・C2 を処理する場合で，太線は占有を表す.)

このような無駄なサーバの停止時間・クライアントの待ち時間を削減するため，図 3(b) のように 3 つのフェーズを分離し，各フェーズごとにリクエストを受け付けるようにした．フェーズを分離することで，図 4(b)・図 5(b) に示すように，クライアントによるサーバの無駄な占有が無くなり，フェーズを分けないときに比べて各クライアントの待ち時間を大幅に減らすことができる．また，サーバ側で強制的にフェーズを決定することで，クライアント側のプログラムミスに

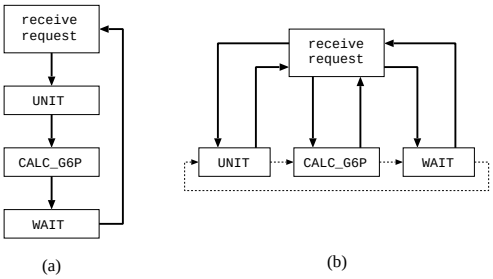

図 3 サーバの処理フェーズ

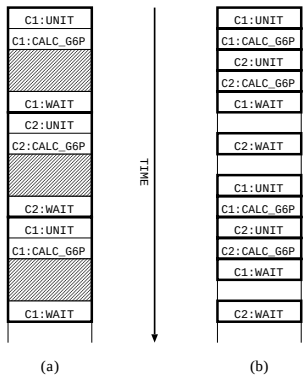

図 4 サーバの処理進行

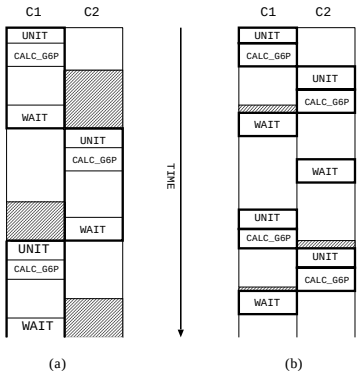

図 5 クライアントの処理進行

よるサーバへの被害を減らし，robustness を高めることにもなる．ただし，このようなプロトコルフェーズの細分化を行うと，HMCS-L の場合と異なり，クライアント別に色々な状態保持のためのバッファが必要となる．しかし，GRAPE-6 が処理可能な粒子数を考慮すると，512MB〜1GB 程度のメモリを各サーバノードが積んでいれば，数十組程度のクライアントを処理する上でバッファの容量は問題にならない．従って，HMCS-R ではプロトコルフェーズの細分化を採用する．

4.3 処理スケジューリング

ほとんどの場合，クライアントごとに処理する粒子数は異なり，サーバへリクエストを出すピッチも異なる．一般的に，粒子数が多いほど重力計算・その他の粒子間相互作用の処理量は多くなり実行時間が長くなるため，クライアントがサーバへリクエストを出す間隔は，そのクライアントの粒子数と概ね比例関係にあると考えられる．それ故，単純にラウンドロビンでクライアントを処理していたのでは，粒子数の少ないクライアントほど待ち時間が長くなってしまうし，サーバの停止時間が生じる可能性がある．そのため，select() によるアクティブなディスクリプタの検出を行い，リクエストを受け次第，つまり先着順にクライアントを処理し，複数のリクエストが同時に来た場合はラウンドロビンで処理するようにした．このように処理量の少ないクライアントを優先的に処理することで，それぞれのクライアントに生じる待ち時間の偏りが小さくなる可能性が高くなる．

4.4 タイムアウト

サーバの robustness を上げるため，以下の 2 つのタイムアウト処理を追加した．

(1) 接続・送受信タイムアウト

重力計算を始めるには，クライアントの全ての agent がそろっている必要があるため，全ての接続・送受信の完了を待たなければならない．したがって，もし何らかの原因で agent のどれか 1 つでも接続・送受信してこないものがあった場合，サーバは停止状態になってしまう．そこで，全サーバノードで agent との接続・送受信の待ち時間に上限をもうけ，1 つでもその時間を越えた場合，そのクライアントとの接続を切断する．

(2) タイムステップ間タイムアウト

タイムステップ間とは，クライアントが重力計算結果を受け取ってから次にサーバへリクエストを出すまでの間のことである．タイムステップ間にクライアント側が中断したり，全ステップが終ったにもかかわらず正しい終了処理を行わなかった場合，サーバのクライアント管理テーブルにはそのクライアントに関する情報が残ったままになってしまい，そういうクライアントが次々に溜ってしまうと管理テーブルの容量を圧迫することになる．そのため，タイムアウト処理をする必要があるが，クライアント側の処理にどれだけ時間がかかるかは分からないため，接続・送受信タイムアウトのように，サーバ側で待ち時間の上限を決めることはできない．そこで，クライアント側でタイムアウト時間を決め，サーバに申告してもらうことにした．クライアントから送られてきたタイムアウト時間を過ぎてもリクエストが来ない場合，管理テーブルからそのクライアントを削除する．

5. 性 能 評 価

5.1 性能評価環境

GRAPE-6 サーバクラスタの諸元を表 1 にまとめ

た．ノードにより多少 CPU の性能が異なるが，実際に重力計算をするのは GRAPE-6 であり，ホスト計算機の主な処理は通信であるため，どのノードを使用してもほとんど差は無いものと考えられる．

CPU	Intel Pentium4 1.9GHz または 1.6GHz
2nd Cache	256KByte または 512KByte
Memory	512MByte
OS	Red Hat Linux 7.1
Network	100base-TX

表 1 GRAPE-6 サーバクラスタの仕様

これに対し，疑似的なクライアントを用意し，通信性能・処理性能に関する評価を行う．実際のアプリケーションを走らせた場合の性能評価を行うことも可能ではあるが，ここではサーバの性能をベンチマーク的に評価するため，疑似クライアントを用いる．疑似クライアントのプログラムではまず，粒子に関して疑似データを生成し，シミュレーションのループに入る．シミュレーションループの中では，UNIT・CALC_G6P・WAIT の順番に入出力フェーズを繰り返し，GRAPE-6 サーバに重力計算を依頼していく．得られた重力計算結果に対しては何もせず，重力計算のタイムステップ間や，CALC_G6P フェーズと WAIT フェーズの間に適宜 sleep() による停止時間を設定し，GRAPE-6 サーバへのリクエストタイミングをずらすことで，疑似的に色々な場合のクライアント側処理を作り出す（停止時間を粒子に対する汎用処理と見立てる）．

GRAPE-6 サーバとの通信時間や処理性能を均等にするため，クライアントとしては全てのノードが同じハードウェアで構成されるクラスタを使用する．また，クライアントプログラムを用いて性能測定する際，複数のクライアントプロセスが同じノード上で走らないようにした．これは，GRAPE-6 サーバと通信するに当たって，1 つのネットワークインタフェースを複数のクライアントプロセスが使用することによる，通信時間の不均衡を無くすためである．

5.2 マルチクライアント性能

HMCS-R はローカルサイト内だけでなく，他サイトのクライアントからも GRAPE-6 を利用可能にし，複数のクライアントで GRAPE-6 を共有することを目的とするシステムである．それ故，性能測定をするに当たってはローカルサイト・リモートサイトの両方から GRAPE-6 サーバへアクセスすべきだと思われる．しかし，HMCS-G の実装では OmniRPC[7] を用いてローカルサイト内の計算機上にクライアントプロセスを走らせ，その計算機を中継して GRAPE-6 サーバへアクセスするため，実際の GRAPE-6 サーバとのやりとりはローカルサイト内に閉じている．また，GRAPE-6 を複数のクライアントで共有して使用する場合，ほとんどの場合には待ち時間が生じるため，HMCS-R の開発に当たっては，性能面に関して，なるべく他のクライアントの影響が小さくなるような工夫を施した．そのため，性能測定に際しては，ローカルサイト内に複数のクライアントプロセスを走らせ，それぞれのクライアントに生じる待ち時間を測定することで，GRAPE-6 サーバの性能を調べる．

それぞれ規模の異なるシミュレーションをする疑似クライアント A・B・C・D を，表 2 のように用意した．TIME STEP はタイムステップ間の停止時間 (秒)，CALC_G6P-WAIT は重力計算を依頼してから結果を受け取りに行くまでの停止時間 (秒) であり，その両方を足したものが総停止時間 (秒) である．実際のシミュレーションにおいては一般的に，これらの停止時間は粒子数に概ね比例するため，粒子数に応じて停止時間を設定した．また，この測定ではサーバノード数を 4，各クライアントの agent 数を 4 とした．

クライアント名	A	B	C	D
粒子数	25000	50000	75000	100000
TIME STEP	1	2	3	4
CALC_G6P-WAIT	0	1	2	3
総停止時間	1	3	5	7

表 2 各疑似クライアントのパラメータ

クライアント間の影響を小さくする工夫として，前章においてフェーズの細分化を行った．よって，マルチクライアント性のを評価に当たっては，フェーズを分けない場合と分けた場合の両方の性能測定をし，両者を比較することで評価をする．

図 6 はフェーズの細分化を行わない場合，すなわち HMCS-L と同じプロトコルで処理を行った場合で，クライアント A・B・C・D それぞれについて，GRAPE-6 サーバを単独で使用した場合と，全てのクライアントを同時に走らせた場合の，1 タイムステップあたりの実行時間を示している．実線の部分が単独で実行した場合の実行時間であり，破線の部分はマルチクライアントにより生じた待ち時間を足した実行時間である．

4 つのクライアント全てを同時に走らせた場合では，どのクライアントもほぼ同じ実行時間となっている．マルチクライアントによる増加時間を見てみると，D に関しては非常に少ないが，その他のクライアントでは D に合わせるような形でそれぞれ増加している．これは，CALC_G6P・WAIT フェーズ間の停止時間が GRAPE-6 での重力計算時間よりも長くなっているにもかかわらず，そのクライアントによりサーバが占有されてしまい，タイムステップ間の停止時間をおいてリクエストを出しても，結局クライアントの処理スケジューリングがラウンドロビンになってしまっているためだと考えられる．

図 7 はフェーズの細分化を行った場合の測定結果であり，グラフの構成は図 6 と同様である．実線部分のデータは図 6 と同じものである．4 つのクライアント全てを同時に走らせた場合の増加時間を見てみる

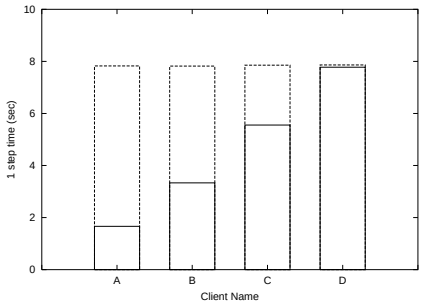

図 6 フェーズを分けない場合の 1 タイムステップの平均処理時間

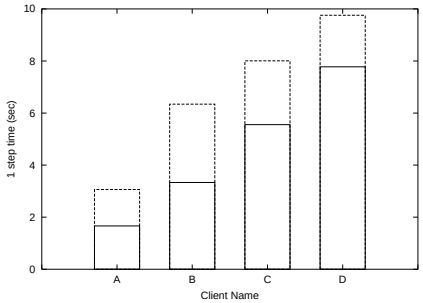

図 7 フェーズを分けた場合の 1 タイムステップの平均処理時間

と，どのクライアントも 1.5〜2.5 秒程度増加している．フェーズを分けない場合は，規模の小さいクライアントほど待ち時間が長くなっていたが，フェーズを分けた場合では，どのクライアントにも比較的バランス良く待ち時間が配分されている．これは，クライアント側の CALC_G6P・WAIT フェーズ間の停止時間によるサーバの占有が無く，フェーズ間の停止時間に他のクライアントの処理を挟むことができ，先着順の処理スケジューリングがうまく働いて，リクエストを出す間隔が短いクライアントを優先して処理しているためだと考えられる．

クライアント D に関しては，フェーズを分けない場合の方が待ち時間が少なく，クライアント C ではほぼ同じくらいであるが，それ以外のクライアントに関しては，フェーズを分けた場合の方が大分少なくなっている．全クライアントの待ち時間の合計を比較すると，フェーズを分けた方が 4 秒以上短くなっている．これは 1 タイムステップの待ち時間であり，これを数千〜数十万回繰り返すことを考えれば，フェーズを細分化することによる待ち時間の削減の効果は大きいものだと言える．また，クライアント数が増えれば，その効果はさらに大きなものになると考えられる．

6. おわりに

我々は，複数の他サイトに分散した汎用並列計算機と専用並列計算機を結合し，各種物理現象の複合シミュレーションを可能とするシステムの基礎を構築するため，重力計算専用計算機 GRAPE-6 クラスタを中心とする計算宇宙物理学用のシステムの，リモートアクセス環境である HMCS-R の実装を行った．複数の agent から成るクライアント群の管理，および agent 群の処理スケジューリングを設計・実装した．複数のクライアントで GRAPE-6 を時分割に使用するに当たって，クライアントの処理スケジューリングや重力計算の入出力フェーズの効率化を図った．規模の異なる複数のクライアントで GRAPE-6 サーバへ同時にアクセスした場合の，それぞれのクライアントに生じる待ち時間を測定した結果，上記の効率化のために行った処理の効果が出ていることが確認できた．また，複数のクライアントで GRAPE-6 を安全に共有するため，robustness を上げる種々の機能をサーバに追加した．

現在，グリッド環境から GRAPE-6 サーバへアクセスするシステムである HMCS-G が実装されており，HMCS-R はその基礎となっている．今後は，サーバノードを切り分け，複数のサーバプロセスを走らせておき，自動に効率良くクライアントを振り分ける，メタサーバの開発をしていく．

謝辞 本研究を進めるに当たり，御助言・御協力を頂いた，東京大学牧野淳一郎助教授，立教大学須佐元講師，筑波大学梅村雅之教授ならびに宇川彰教授に深く感謝致します．

参 考 文 献

1) the GRAPE project: . http://grape.astron.s.u-tokyo.ac.jp/grape/.
2) M. Umemura: Three-dimensional hydrodynamical calculations on the fragmentaion of pancakes and Galaxy formation, *The Astrophysical Journal, 406*, pp. 361–382 (1993).
3) T.Nakamoto: A 3-D Radiative Transfer Solver using a Massively Parallel Computer, *Numerical Astrophysics 1998* (1998).
4) 朴泰祐他: Heterogeneous Multi-Computer System における重力効果を含む宇宙輻射流体計算, 情報処理学会論文誌ハイパフォーマンスコンピューティングシステム, Vol. 43, No. SIG6(HPS5), pp. 219–229 (2002).
5) T. Boku et al.: HMCS-G:Grid-enabled Hybrid Computing System for Computational Astrophysics, *Proc. of CCGrid2003(GAN03 Workshop)* (2003).
6) The Message Passing Interface (MPI) standard: . http://www.mcs.anl.gov/mpi/.
7) M. Sato et al.: OmniRPC: a Grid RPC facility for Cluster and Global Computing in OpenMP, *Proc. of Workshop on OpenMP Applications and Tools 2001(LNCS 2104)*, pp. 130–135 (2001).