Cray XD1での星団進化の高性能「小規模」シミュレーション

似鳥 啓吾[†] 牧野 淳一郎[††] 阿部 譲司[†††]

本論文では，400 個のデュアルコア Opteron プロセッサを用いた Cray XD1 システム上での高性能な N 体シミュレーションコードの実装と，64 k 粒子の星団のシミュレーションでの性能について述べる．これまでにも多くの天体物理学的 N 体計算の並列化が報告されているが．その中でも数十プロセッサ以上を用いた実装の性能評価には，大きな粒子数が使われる傾向がある．たとえば，これまでのゴードン・ベル賞へのエントリでは，少なくとも 70 万粒子が用いられている．この傾向の理由は，並列化効率にある．というのも，大規模並列機で小さな粒子数で性能を出すのは非常に困難であるからである．しかしながら，多くの科学的に重要な問題では計算コストは $O(N^{3.3})$ に比例するため，比較的小さな粒子数の計算に大規模並列計算機を用いることが非常に重要である．我々は，64 k 粒子の $O(N^2)$ 直接計算独立時間刻み法の計算で 2.03 Tflops（対ピーク 57.7%）の性能を実現した．これまでの 64 k 粒子での同様の計算における最大の効率は，128 プロセッサの Cray T3E-900 での 7.8%（9 Gflops）である．今回の実装では従来の方法より高スケーラブルな 2 次元並列アルゴリズムを用いている．さらに今回のような高性能を達成するためには Cray XD1 の低レイテンシネットワークが本質的に重要であった．

High-Performance *Small-Scale* Simulation of Star Clusters Evolution on Cray XD1

Keigo Nitadori,[†] Junichiro Makino[††] and George Abe[†††]

In this paper, we describe the implimentation and performance of N-body simulation code for a star cluster with 64 k stars on a Cray XD1 system with 400 dual-core Opteron processors. There have been many reports on the parallelization of astrophysical N-body simulations. For parallel implementations on more than a few tens of processors, performance was usually measured for very large number of particles. For example, all previous entries for the Gordon-Bell prizes used at least 700 k particles. The reason for this preference of large numbers of particles is the parallel efficiency. It is very difficult to achieve high performance on large parallel machines, if the number of particles is small. However, for many scientifically important problems the calculation cost scales as $O(N^{3.3})$, and it is very important to use large machines for relatively small number of particles. We achieved 2.03 Tflops, or 57.7% of the theoretical peak performance, using a direct $O(N^2)$ calculation with the individual timestep algorithm, on 64 k particles. The best efficiency previously reported on similar calculation with 64 K or smaller number of particles is 7.8% (9 Gflops) on Cray T3E-900 with 128 processors. Our implementation is based on highly scalable two-dimensional parallelization scheme, and low-latency communication network of Cray XD1 turned out to be essential to achieve this level of performance.

1. はじめに

太陽系，星団，銀河，銀河団，宇宙の大規模構造といった多くの天体は，ニュートン重力によって相互作用をする多粒子系として表現される．このような系の数値シミュレーションは「重力 N 体シミュレーション」とよばれ，計算天体物理学の最も強力な道具の 1 つである．重力 N 体シミュレーションは，月の形成[1]から宇宙全体[2]まで，すべてのスケールの系の研究に使われている．

これらのシミュレーションは，以下の 3 つの理由により高い計算能力を必要とする．1 つは，粒子数が N のとき（少なくとも単純な実装では）1 ステップあたりの計算コストが $O(N^2)$ となることである．

2 つめの理由は，多くの場合で，系の進化のタイムスケールが典型的な粒子の軌道のタイムスケールと比べて何桁も大きくなることである．たとえば，太陽系の長期間シミュレーションでは惑星とその他の天体の軌道を 4.5×10^9 年にわたって追う必要があり，適切な精度を得るには 1 年につき 100 タイムステップ前後が必要である．また，惑星形成のシミュレーションの場合，微惑星の軌道を数 100 万軌道周期にわたって

† 東京大学
The University of Tokyo
†† 国立天文台
National Astronomical Observatory of Japan
††† ソニー株式会社情報技術研究所
Information Technologies Laboratories, Sony Corporation

追う必要がある．このような長時間積分が必要となるため，GRAPE-4[3] や GRAPE-6[4] のような専用計算機を用いてもこの種の問題では数万粒子までしか扱えなかった[5]．このことは星団のシミュレーションについても同様である．この場合，系の進化のタイムスケールは粒子数に比例し，1 軌道周期あたりのタイムステップは $O(N^{1/3})$ に比例する（1 ステップに粒子が進む距離が，最近接粒子との距離に比べて十分に小さくなるようにとるため）．全体では，計算コストは $O(N^{3.3})$ となる．

3 つめの理由は，異なる粒子が異なる軌道のタイムスケールを持ちうることである．重力は引力であるので，2 つの粒子は任意の距離まで近づくことができる．このため，時間刻みもこれにあわせる必要がある．しかも，時間刻みを変えるだけでは十分ではない．可変ではあってもすべての粒子を積分するのに同じ時間刻みを使うと，星団のシミュレーションでは，N を増加させたとき平均の刻み幅は $1/N$ かそれ以上の速さで減少してしまう．このため，計算コストは $O(N^{3.3})$ ではなく $O(N^4)$ となる．この 3 つめの問題は，粒子ごとに異なる時間刻みを許すことで解決できる[6),7)]．しかし，この手法では並列に積分できる粒子数は N よりもずっと小さくなり，N がそれほど小さくなくても大規模並列計算機を用いることは難しくなる．

Dorband ら[8] は，この独立（ブロック）時間刻みのアルゴリズムと組み合わせた $O(N^2)$ 直接計算法のコードを並列化し，Cray T3E での性能を報告した．彼らは通信と計算が最大限にオーバラップするようにした通信アルゴリズムを用いたが，$N = 64\,\mathrm{k}$ で初期粒子分布の非等方性が強い場合には，64 プロセッサ以上では性能は上がらないという結果であった．より最近の PC クラスタでの研究[9] では，PC クラスタの計算性能あたりの通信性能は Cray T3E に比べて桁で劣るため，さらに悪いスケーリングとなっている．

専用計算機 GRAPE を用いることで，単一ノードの性能を著しく向上させることができ，64 k もしくはそれ以下のような比較的小さな粒子数での長時間積分シミュレーションも実現することができる．しかしこの場合でも，全体の性能は 1 台のホスト計算機によって制限されることになる．なぜなら，複数の GRAPE 搭載ノードをまたぐ並列化は，相対的に PC クラスタよりずっと大きな通信オーバヘッドをかかえることになるからである．実際にも，粒子数がかなり大きくない限りは並列化による加速が得られないという結果が得られている[10]．

本論文では，64 k 粒子の星団のシミュレーションに対し 1000 近くのプロセッサを用い，世界で初めて理論ピーク性能の 50%以上を達成した計算について，その結果を報告する．400 個の 2.2 GHz dual-core Opteron を搭載した Cray XD1 での平均実測性能は 2.03 Tflops，理論ピーク性能は 3.52 Tflops である．先行研究では，並列化効率が良い場合でも，理論ピークの 10%以下しか達成していない．よって我々の実装は以下の 2 点で画期的である．1 つは小さな粒子数で高いスケーラビリティを実現したことであり，もう 1 つはきわめて高い実性能を達成したことである．

以下，どのようにして高いスケーラビリティと実性能を実現したのかについて，次のような構成で述べる．2 章では独立（ブロック）時間刻みのアルゴリズムの概略を述べる．3 章では 2 次元並列アルゴリズムについて述べる．これは当初は Makino[11] によって提案されたものである．さらに，今回，より高性能を得るために加えた改良についても述べる．4 章では高度に最適化された重力計算ループについて概説する．詳細は Nitadori ら[12] にある．5 章では，2 成分からなる 64 k 粒子の星団の長時間計算の結果を述べる．6 章はまとめとする．

2. ブロック時間刻みアルゴリズム

独立時間刻みアルゴリズムの基本的な考え方は，粒子ごとに時間と時間刻みを持たせることである．すなわち，粒子 i は現時刻 t_i と時間刻み Δt_i を持つ．計算手続きは以下のとおりである：

(1) $t_i + \Delta t_i$ が最小になるような粒子を選ぶ．
(2) この粒子を新しい時刻 $t_{i,new} = t_i + \Delta t_i$ まで積分し，新しい時間刻みを決定する．
(3) ステップ (1) に戻る．

粒子 i を積分するためには，時刻 $t = t_i + \Delta t_i$ でのすべての粒子からの力を計算する必要がある．粒子 i 以外のすべての粒子について，$t_j \le t \le t_j + \Delta t_j$ であることが保証されている．時刻 t での粒子 i への力を計算するためには，他の粒子の時刻 t での「予測子」を求めて使う．この予測子の計算を可能にするため，積分法としては線形多段階法を用いる．

この独立時間刻みアルゴリズムの実用上の問題は，1 度に 1 つの粒子しか積分されず，並列化の余地が少ないことである．そこで，McMillan[13] はブロック時間刻みと呼ばれるアルゴリズムを導入した．この場合，時間刻みは 2 の冪乗に量子化され，同じ時間刻みの粒子は同じ時刻を共有し並列に積分されることができる．このようになるためには，時間刻みは現時刻を割り切れる必要がある．この単純な制限を課すことによって，

最大限の並列度を得ることができる[14]．この複数の粒子がまとめて積分されるステップを「ブロックステップ」と呼ぶ．

積分法には 1990 年代初頭までは，4 次の可変刻み幅の Adams-Bashforth-Moulton 線形多段階法を二階微分方程式用に適合したものが用いられてきたが，今日では 4 次の Hermite 積分法[15] が広く用いられている．4 次の Hermite 積分法はオリジナルの ABM 法と比べて実装が単純であり計算効率も高い．

3. 最適な負荷分散の 2 次元並列アルゴリズム

3.1 1 次元並列化

既存のブロック時間刻みアルゴリズムの並列実装[8)~10)] はすべて，ここで 1 次元並列化と呼ぶ，重力計算を 1 つのループ変数について分割する手法に基づいている．ブロック時間刻みアルゴリズムの重力計算ループは次のようなかたちをしている：

```
for(ii=0; ii<nblock; ii++){
  i = list_of_particles_in_current_block[ii];
  clear the force on particle i;
  for(j=0; j<n; j++){
    if(j!=i){
      粒子 j から粒子 i への力を計算して積算
    }
  }
}
```

ここで，nblock は現在のブロック時間刻みで同時に積分される粒子の数，list_of_particles_in_current_block は積分される粒子のリストである．既存の並列実装はいずれも，内側のループか外側のループのどちらか一方でしか並列化されていないという意味で 1 次元的である．

粒子数 N がプロセッサ数 p よりも十分に大きければ，1 次元並列化で事足りるという考え方も可能かもしれない．しかしブロック時間刻みアルゴリズムの場合はそうはならない．なぜなら，ブロックステップあたりに積分される粒子数 N_b は N よりもずっと小さいからである．5 章で述べるように，64 k 粒子の計算では N_b の平均値は 400 ぐらいである．したがって，内側のループに対する並列化（i 並列とも呼ぶ）は数百を超えるプロセッサ数には適さない．外側のループに対する並列化（j 並列）は全プロセッサにまたがる総和を要求するが，ネットワークが比較的低速な分散メモリ計算機ではこの処理が非常に低速になるという問題がある．

さらに，i 並列でも j 並列でも 1 次元並列の場合，分散メモリ計算機におけるブロックステップあたりノードあたりの通信量は，ノード数によらず $O(N_b)$ である．一方ノードあたりの計算量は $O(NN_b/p)$ である．それゆえ，適当な並列化効率を得られる最大のノード数は $O(N)$ ということになる．通信が比較的遅い場合，N の前の係数はとても小さなものとなる．これが，先行研究ではスケーラビリティが良くない理由である．

3.2 基本 2 次元アルゴリズム

i 並列化と j 並列化を同時に行うことで，スケーラビリティを改善することができる．以下では，Makino[11] の記述を要約する．「regular-bases の Hyper-systolic アルゴリズム」[16] も本質は同じである．この考え方の起源は Hillis ら[17] による，N 体計算に $O(N^2)$ 個のプロセッサを使うアルゴリズムにまでさかのぼることができる．

p 個のプロセッサが $r \times r$ のように 2 次元に配置されている状態を考える（$r^2 = p$）．N 個の粒子は r 個のサブセットに分割され，プロセッサ q_{ij} は i 番目と j 番目のサブセットを保持する．計算手順は次のとおりである：

(1) それぞれのプロセッサ q_{ij} はサブセット j からサブセット i への力を計算する．
(2) 水平方向（i 方向）に力の総和をとる．結果は対角のプロセッサ q_{ii} に格納される．
(3) プロセッサ q_{ii} は総和された力を鉛直方向（j 方向）に放送する．
(4) すべてのプロセッサはサブセット i の粒子を更新し，対角プロセッサは更新分を水平方向に放送する．各プロセッサは j 番目のサブセットを受信することになる．

実際には，独立時間刻み法ではブロックステップ中の粒子（以降「活動粒子 [*active particles*]」と呼ぶ）への力のみが計算され，上述の手続きの通信も活動粒子についてのみ行われる．すなわち，このアルゴリズムでは，それぞれのプロセッサは $O(N_b/r)$ のデータを送信／受信する（1 次元のときは $O(N_b)$）．

3.3 改良 2 次元アルゴリズム

基本 2 次元アルゴリズムにはブロック時間刻み法と組み合わせたとき 2 つの問題点がある．1 つは，活動粒子の数が少ないと計算負荷が不均一となることである．各ステップで，プロセッサ q_{ij} が計算しなければならない重力は，サブセット j 中の全粒子からの，サブセット i 中の全活動粒子への力である．それゆえ，活動粒子の個数の揺らぎが，不均一な計算負荷状況を生じさせてしまう．この効果は，1 辺のプロセッサ数 r が大きいときに深刻となる．なぜなら，活動粒子数

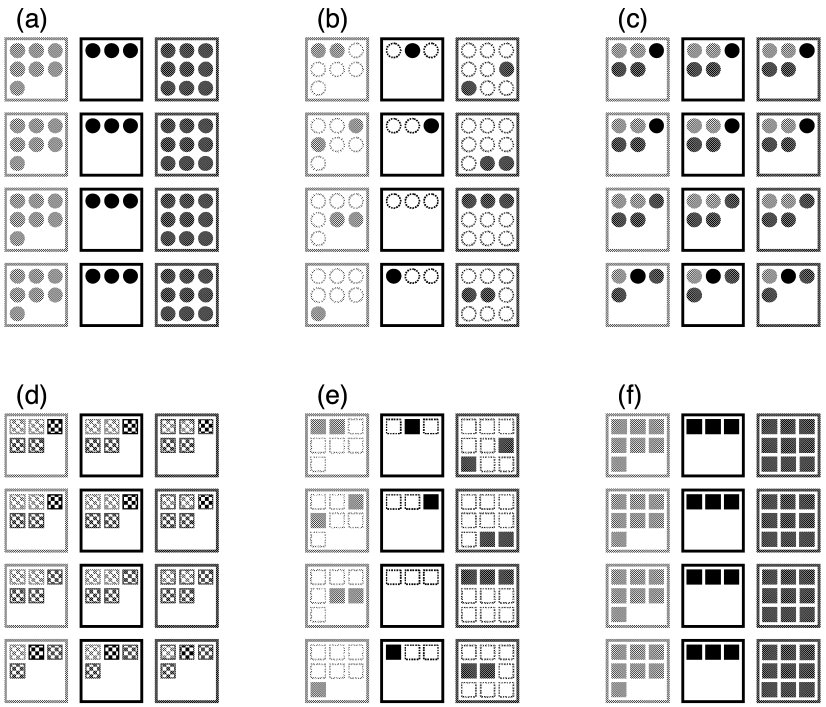


図 1 $(a, b) = (4, 3)$ のプロセッサ配置での改良 2 次元アルゴリズム．円は粒子の位置を，半塗りの四角は粒子への部分力を，塗りつぶされた四角は粒子への合計力を示す．(a) 第 1，2，3 列にそれぞれ 7，3，9 個の活動粒子がある．(b) 列の中でそれぞれのプロセッサは担当する粒子を決める．(c) 水平方向に位置を放送．(d) 力の計算．(e) 水平方向に力を総和．(f) 鉛直方向に合計力を放送

Fig. 1 Improved 2D algorithm for processor array of $(a, b) = (3, 4)$. Circles denote positions, half-toned and filled square denote partial and full forces, respectively. (a) 7, 3 and 9 active particles are found in columns 1, 2 and 3. (b) Each processor in one column decides which particles to handle. (c) Horizontal broadcast of the positions. (d) Force calculation. (e) Summation of the partial forces. Completed forces are sent back to the original locations. (f) Vertical broadcast of the forces.

が最大となるサブセットが計算全体を律速してしまうからである．もう 1 つの問題は，平方数のプロセッサ数しか使えないことである．

ここで，上記の 2 つの問題を解決した改良 2 次元アルゴリズムを紹介する．i 並列と j 並列を同時に行うという基本的な考え方はこれまでと同様であるが，通信パターンとデータ分割は，前節で述べた基本 2 次元アルゴリズムとは根本的に異なる．

2 次元状に $a \times b$ に配置された p 個のプロセッサを考える（**図 1**）．通信は水平方向もしくは鉛直方向に発生する．実装を簡潔にするために，縦横方向の通信パターンにあわせて MPI コミュニケータを生成し（`MPI_Comm_split` を使用），MPI の集団通信機能を利用している．

N 個の粒子は b 個のサブセットに等分割され，q_{xj} となるすべてのプロセッサは j 番目のサブセットを共有する．計算の手順は以下のとおりである：

(1) 各プロセッサは各自のサブセット中の最小の時間刻みを求め，サブセットから活動粒子のリスト（活動リスト）を作る．ここで，プロセッサ q_{xj} のリストはどの x についても同一である．

(2) 各プロセッサは活動リスト長と（ローカルの）最小時間刻みを水平方向に放送する．ここでは `MPI_Allgather` を使う．もし他のプロセッサがより小さな時間刻みを持っているようなことがあれば，そのプロセッサの活動粒子はこのステップでは積分されない．

(3) この段階で全プロセッサは各列の持つ活動粒子数が分かっている．これをもとに，各列の持つ活動粒子の列内での分配を決定する．このためには，全活動粒子に仮の番号を与えて全活動粒子のリストを作り，これを a 個に（割り切れる限り）等分割すればよい．一例を**図 2** に示す．

(4) これに従って各プロセッサは活動リストを a 個のサブセットに分割し，プロセッサ q_{ij} は i 番目のサブセットを各自の活動リストとして選択

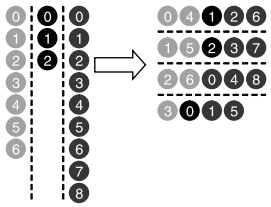


図 2 図 1 (b) を与える活動粒子の分配方法．各列に 7，3，9 個の活動粒子がある場合（左），各行は，5，5，5，4 個の活動粒子を担当する（右）．各列の活動粒子の個数が分かった段階で分配を決定できる点に注意

Fig. 2 Making the balanced vertical distribution of active particles of Fig. 1 (b). 7, 3 and 9 active particles are found in columns 1, 2 and 3 (left), and 5, 5, 5 and 4 active particles are assigned to row 1, 2, 3 and 4 (right). Note that the distribution can be fixed after all numbers of active particles in each column are known.

する．

(5) 各プロセッサは各自の活動リスト（の粒子の座標）を水平方向に放送する．これで同じ行のプロセッサはすべて同一の活動粒子の座標のリストを持つことになる．ここでは，`MPI_Allgatherv` を使う．

(6) プロセッサ q_{ij} は自分のサブセット j 中の全粒子からの，活動リスト中の全粒子への，(部分) 力を計算する．

(7) 水平方向に部分力の総和をとり，結果はその粒子を当初から保持していたプロセッサに格納する．ここでは `MPI_Reduce_scatter` を使う．この段階で，各プロセッサは各自の活動粒子への合計の力を持っている．

(8) 各プロセッサは，各自の活動粒子への力を鉛直方向に放送する．ここでは，`MPI_Allgatherv` を使う．この段階で，すべてのプロセッサ q_{xj} はサブセット j 中のすべての活動粒子への力を持っている．

(9) 各プロセッサは活動粒子の軌道積分を行う．ここで，q_{xj} となるすべてのプロセッサは x について冗長な演算を行う．

図 1 にこの改良 2 次元アルゴリズムの挙動を示す．この例では，活動粒子の数が列ごとに大きく異なっても，各プロセッサの計算負荷はよくバランスしている．

4. 重力計算ループの最適化

4.1 関数と演算数

重力 N 体シミュレーションでは，並列化効率が良好であれば，計算時間の大半は名前どおり粒子間の重力の計算に費やされる．Hermite 積分法[15] を用いる場合，計算する必要があるのは以下の加速度，その時間微分，ポテンシャルであり，

$$\mathbf{a}_i = \sum_{j\neq i}^{N} \frac{m_j \mathbf{r}_{ij}}{(r_{ij}^2+\varepsilon^2)^{3/2}}, \tag{1}$$

$$\dot{\mathbf{a}}_i = \sum_{j\neq i}^{N} m_j \left[\frac{\mathbf{v}_{ij}}{(r_{ij}^2+\varepsilon^2)^{3/2}} - \frac{3(\mathbf{v}_{ij}\cdot\mathbf{r}_{ij})\mathbf{r}_{ij}}{(r_{ij}^2+\varepsilon^2)^{5/2}} \right], \tag{2}$$

$$\phi_i = -\sum_{j\neq i}^{N} \frac{m_j}{(r_{ij}^2+\varepsilon^2)^{1/2}}, \tag{3}$$

ここで，$\mathbf{a}_i$ は粒子 i の重力加速度，ϕ_i はポテンシャル，$\dot{\mathbf{a}}_i$ は Hermite 積分法用の加速度の時間微分であり，$\mathbf{r}_i$, $\mathbf{v}_i$, m_i はそれぞれ位置，速度，質量である．また，ε はソフトニングパラメータ，$\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$ であり，$\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$ である．

$\mathbf{a}$ と ϕ の計算には，9 回の乗算と 10 回の加減算，それぞれ 1 回の除算と平方根が含まれる．Warren ら[18] はこの相互作用計算に 38 回の浮動小数点演算を用い，この換算はその後の多くの研究にも使われた．今回我々もこれに従う．

これに加え，$\dot{\mathbf{a}}_i$ の計算にはさらに 11 回の乗算と 11 回の加減算が必要になる．したがって，エルミート法のための重力相互作用の浮動小数点演算数は 60 とした[12]．

4.2 最 適 化

今回用いた最適化の詳細は，Nitadori ら[12] に述べられている．基本的な考え方は，必要な精度を保持しつつ最近の SSE/SSE2 命令セットの機能を最大限に活用することである．SSE2 命令セットは，x87 のようなスタックベースの命令より最適化しやすい，レジスタベースの命令を提供している．さらに，2 つの倍精度実数への SIMD 演算をサポートしている．SSE は 4 つの単精度実数への SIMD 演算をサポートし，さらに重力計算には非常に有用な逆数平方根の近似命令を持っている．我々の最適化された重力計算ループでは，最初の座標の差分と最後の力の積算でのみ倍精度の SSE2 命令を用い，残りの部分については単精度の SSE 命令を用いている．逆数平方根の部分では，近似命令と 1 回のニュートン・ラフソン法反復を用い，単精度相当で計算している．この混合精度のアプローチは，GRAPE-4[3] や GRAPE-6[4] でも用いられてきた．

AMD k8（Athlon64 または Opteron）プロセッサでは，この重力計算ループは 4 つの重力相互作用を 120 CPU サイクルで計算できる．いい換えれば，120 サイクルで 240 回相当の浮動小数点演算を実行する

ことができる．すなわち 2.2 GHz のプロセッサでは 4.4 Gflops で重力を計算できることになり，偶然にも k8 プロセッサの倍精度ピークの値と同一となっている．

今回用いたのは最大 400 個のデュアルコア Opteron プロセッサを搭載した Cray XD1 である．1 ソケット（2 プロセッサコア）あたり 8.8 Gflops の性能を持ち，全体としては 3.52 Tflops の性能を持つ．

5. 星団進化のシミュレーション

本章では，今回開発した並列独立時間刻みコードを用いた星団の熱力学的進化のシミュレーションの性能について報告する．

5.1 モデル

今回我々が調べるのは，2 成分からなる星団の進化である．このような系に対しては軌道平均をとったフォッカー・プランクモデルを用いた研究はいくつかあるが[19]，N 体計算を用いた例は非常に少ない．

初期モデルとしては 64 k 粒子からなる力学平衡の Plummer モデルを作った．単位系は Heggie ユニット[20] である．「重い」粒子 1 個は「軽い」粒子 1 個の 5 倍の質量を持ち，重い粒子が全質量の 10%を占めている．すなわち，初期モデルには 64,111 個の軽い粒子と 1425 個の重い粒子が存在する．今回はソフトニング付きの重力ポテンシャルを用いた．ソフトニングパラメータは $\varepsilon = 4/N$ である．

5.2 結果

図 3 は質量分離の進化を表している．重い粒子は運動エネルギーを失い系の中心へ沈んでいく．**図 4** は Casertano ら[21] の手法を用いて計算したコアの大きさと密度である．コアが縮んでいき，コアが小さくなるほど進化も速くなるのが分かる．現在この N 体計算の結果とフォッカー・プランク計算との結果の詳細な比較を行っている．

5.3 性能

全計算時間は 26.5 時間，合計のタイムステップ数は 4.93×10^{10}，合計の浮動小数点演算数は 1.94×10^{17} であり，平均の性能は 2.03 Tflops であった．

表 1 は $1046 < t \leq 1047$（シミュレーション系での時間）での計算時間の詳細である．実行時の性能に（主にハードウェアの調子により）時間変化があったので，最も高い性能が測定された時点のものを記した．ブロックステップ平均の消費時間は約 700 μs である．前章で述べたように，1 ブロックステップにつき 4 回の MPI 集団通信が行われている．それにもかかわらず通信時間が非常に短いことから，Cray XD1 の通信ハードウェアと MPI ライブラリの性能が良好であることが分かる．

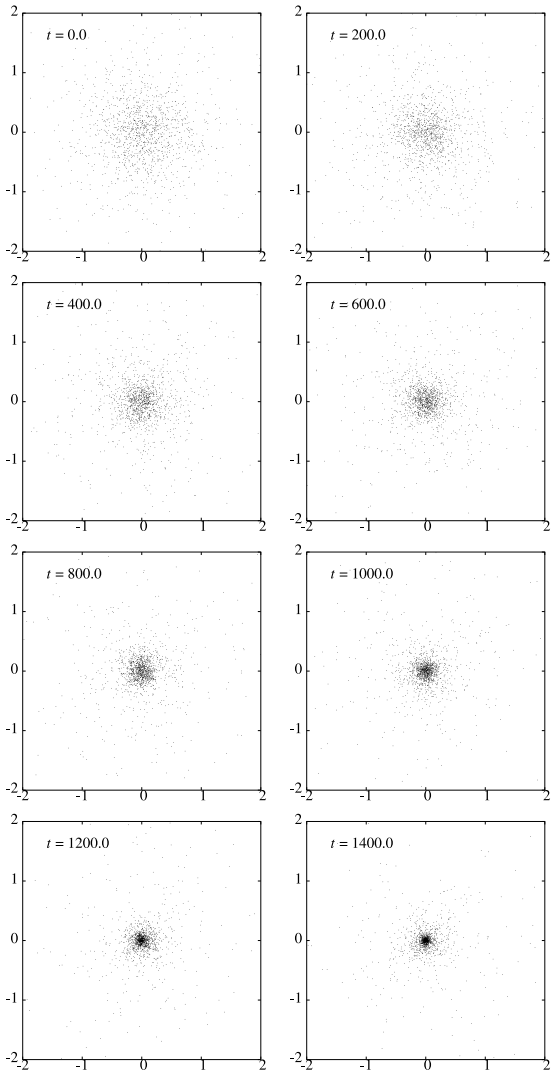


図 3 星団の進化．重い粒子のみをプロットしてある
Fig. 3 The evolution of the star cluster. Only the heavy particles are plotted.

ここで 1 ブロックステップあたりの通信回数と通信量を簡単に見積もってみる．p 個のプロセッサ間の集団通信は，p が 2 の冪乗であれば $\log_2 p$ 回の 1 対 1 通信で実現できる．より一般的には，p が n 個の素数 r_i の積 $p = \prod_{i=1}^{n} r_i$ で表されるとき，$\sum_{i=1}^{n}(r_i - 1)$ 回の通信で実現できる．ただし実際の通信回数は MPI 実装に依存する．今回我々は $25 \times 32 = 800$ 個の MPI プロセスを用いた．よって，横方向に 3 回，縦方向に 1 回の集団通信によって合計 23（$= 5 \times 3 + 8$）回の 1 対 1 通信が発生すると予想できる．1 回の集団通信で 1 プロセッサが送受信するメッセージ長の合計は，平

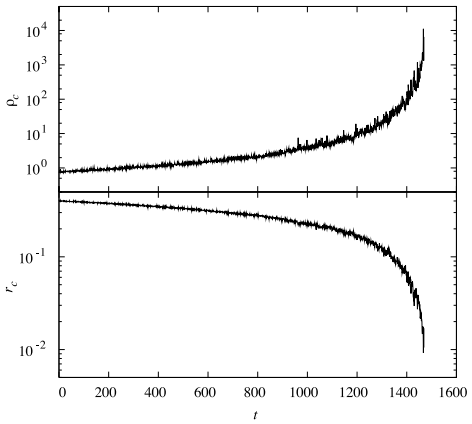


図 4 N 体単位時間に対するコア密度（上）とコア半径（下）
Fig. 4 The core density (top) and the core radius (bottom) for N-body unit time.

表 1 $1046 < t \leq 1047$ での計算時間の詳細
Table 1 Breakdown of the calculation time.

演算性能	2.37 Tflops
実消費時間	54.7 s
合計タイムステップ数	3.26×10^7
合計ブロックステップ数	8.07×10^4
ブロックステップ平均の計算時間	490 μs
ブロックステップ平均の通信時間	179 μs
平均の活動粒子数 N_b	404

均の活動粒子数が 400 程度，1 粒子あたりの大きさが 50 バイト程度であることから（これを 25 または 32 分割するので）片方向あたり 800 バイト以下である．よってこのプロセッサ数ではほとんどのネットワークにおいてレイテンシが律速となる．

5.4 スケーラビリティ

図 5 に，最大 800 プロセッサ（400 dual-core）の Cray XD1 での N=16 K と 64 K での我々の並列 N 体コードの性能を示す．同じ N と p での複数のプロットは異なるプロセッサ縦横比での結果である．最適の縦横比での結果をみると，N=64 k での並列化効率は 512 プロセッサまで 75%以上を維持している．16 k 粒子の場合でも，256 プロセッサまでは効率は 60%以上であり，512 プロセッサではほぼ 1 Tflops の性能に達している．

今回得られた性能の位置づけを明確にするため，いくつかの先行研究を要約してみる．Dorband ら[8] は 128 プロセッサの Cray T3E-900 で N=16 k のとき約 6 Gflops，理論ピーク性能の 5.2%を得ている．GRAPE-6[4] で N=16 k のときに得られた最大の性能は 130 Gflops であり理論ピークの 13%である．我々は同じ粒子数で 256 プロセッサのとき理論ピークのほぼ 60%を達成している．

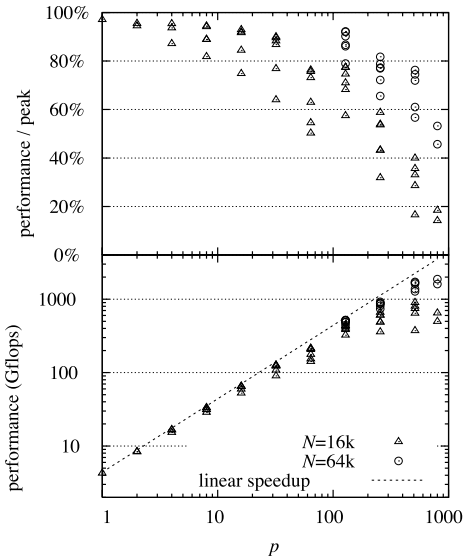


図 5 N=16 k（三角）と N=64 k（円）でのプロセッサ（コア）数 p に対する対ピーク性能（上）と実効性能（Gflops）（下）．破線はリニアスピードアップ．同じ N と p での複数のプロットは，異なるプロセッサ縦横比での結果を示す．対ピーク性能はほぼ並列化効率と見なしてよい
Fig. 5 Performance per peak (top) and performance in Gflops (bottom) for the number of processors, for N=16 k (triangle) and N=64 k (circle). The dotted line indicates the ideal linear speedup. Multiple symbols for the same values of N and p are results for different processor geometries. The performance per peak is almost equivalent to the parallel efficiency.

6. ま と め

我々は $O(N^2)$ 直接計算独立（ブロック）時間刻み積分法用の，新しい高スケーラブルなアルゴリズムを開発した．これを Cray XD1 に実装し，16 k や 64 k といった小さな粒子数においても，800 プロセッサ（測定当時 Cray XD1 の最大構成）まできわめて良好な性能のスケーリングを達成した．512 プロセッサでは 70%以上の並列化効率を達成し，800 プロセッサでの長時間計算では 2.03 Tflops を得た．

今回我々は初めて，最近の 1000 プロセッサ近くの超並列計算機が 10^5 粒子以下の天体物理学的 N 体計算に対して使用できることを明らかにした．計算コストが $O(N^{3.3})$ に比例する問題に対しては $N = 10^5$ が数 Tflops の計算機では実用的な上限なので，今回のように小さな N での高い効率を達成することはきわめて重要である．

謝辞 数多くの有益な議論をしてくださった KFCR 社の福重俊幸氏と川井敦氏に感謝します．SONY の松元秀樹氏，ならびにクレイ・ジャパンのスタッフの

方々には Cray XD1 の使用にあたって多大なるご協力，ご指南をいただきました．ここに感謝の意を表します．ユーテン・ネットワークス社の光成滋生氏からは SIMD 命令をはじめとする様々なの最適化技術をご教示いただきました．この場を借りてお礼申し上げます．本研究の一部は，日本学術振興会科学研究費補助金特別研究員奨励費（課題番号：18・11376）の援助を受けて行われました．

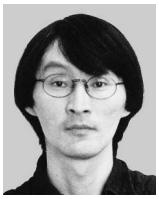
参 考 文 献

1) Kokubo, E., Makino, J. and Ida, S.: *AAS/Division for Planetary Sciences Meeting Abstracts*, Vol.31 (1999).
2) Springel, V., et al.: *Nat*, 435, 629 (2005b).
3) Taiji, M.: *Highlights of Astronomy*, Vol.11, No.600 (1998).
4) Makino, J., Fukushige, T., Koga, M. and Namura, K: *PASJ*, Vol.55, No.1163 (2003).
5) Kokubo, E. and Ida, S.: *ApJ*, Vol.581, No.666 (2002).
6) Aarseth, S.J.: *Monthly Notices Roy. Astron. Soc.*, Vol.126, No.223 (1963).
7) Aarseth, S.J.: *Multiple Time Scales*, Brackbill and Cohen (Eds.), Academic Press, New York, p.377 (1985).
8) Dorband, E.N., Hemsendorf, M. and Merritt, D.: *J. Comp. Phys.*, Vol.185, No.484 (2003).
9) Gualandris, A., Portigies Zwart, S. and Tirado-Romos, A.: submitted to *IEEE Transactions on Parallel and Distributed Systems* (astro-ph/0412206) (2004).
10) Fukushige, T, Makino, J. and Kawai, A.: *PASJ*, Vol.57, No.1009 (2005).
11) Makino, J.: *New Astronomy*, Vol.7, No.373 (2002).
12) Nitadori, K., Makino, J. and Hut, P.: *New Astronomy*, Vol.12, No.169 (2006).
13) McMillan, S.L.W.: *The Use of Supercomputer in Stellar Dynamics*, Vol.156, Hut, P. and McMillan, S. (Eds.), Springer, New York (1986).
14) Makino, J.: *PASJ*, Vol.43, No.859 (1991b).
15) Makino, J. and Aarseth, S.J.: *PASJ*, Vol.44, No.141 (1992).
16) Lippert, T., Ritzenhöfer, G., Glässner, U, Hoeber, H., Seyfried, A. and Schilling, K.: *International Journal of Modern Physics C*, Vol.7, No.485 (1998).
17) Hillis, W.D. and Barnes, J.: *Nature*, Vol.326, No.27 (1987).
18) Warren, M.S., Salmon, J.K., Becker, D.J., Goda, M.P., Sterling, T. and Winckelmans, G.S.: *Proc. Supercomputing '97*, IEEE Computer Society Press, Los Alamitos (1997).
19) Inagaki, S. and Wiyanto, P.: *PASJ*, 36, 391 (1984).
20) Heggie, D.C. and Mathieu, R.D.: *The Use of Supercomputer in Stellar Dynamics*, Hut, P. and McMillan, S. (Eds.), Springer, New York, 233 (1986).
21) Casertano, S. and Hut, P.: *ApJ*, 298, 80 (1985).

(平成 18 年 10 月 13 日受付)
(平成 19 年 1 月 29 日採録)

似鳥 啓吾

1981 年生．2006 年東京大学大学院理学系研究科天文学専攻修士課程終了．現在，同博士課程 2 年．重力多体問題の数値計算アルゴリズム，並列化についての研究に従事．2006 年より日本学術振興会特別研究員 DC1．日本天文学会会員．



牧野淳一郎

1963 年生．1990 年東京大学大学院総合文化研究科広域科学専攻博士課程修了．学術博士の学位を取得．同年東京大学教養学部助手．1994 年より東京大学教養学部助教授．1999 年より東京大学大学院理学系研究科助教授．2006 年より自然科学研究機構国立天文台教授．理論天文学の研究，シミュレーション用専用計算機の開発に関する研究に従事．1995 年から 2003 年まで計 6 度 Gordon Bell 賞，1998 年日本天文学会林忠四郎賞．日本天文学会会員．

阿部 譲司（正会員）

1959 年生．1984 年慶應義塾大学大学院修士課程電気工学専攻修了．同年日本鋼管株式会社入社．1991 年ソニー株式会社入社．現在，情報技術研究所 HPC 領域担当部長．並列行列解法の大規模シミュレーションへの応用，並列アーキテクチャ上でのアルゴリズムの特性，並列アーキテクチャ評価等の研究に従事．日本応用数理学会評議員．