ハイパフォーマンス
コンピューティング 65－8
（１９９７．３．６）

並列処理による重力多体問題の解法の高速化

野中 恵三, 平田 博章, 新実 治男, 柴山 潔
京都工芸繊維大学 工芸学部 電子情報工学科

概要

重力多体問題は本来, 粒子数の２乗回に比例する計算が必要であり多大な処理時間を要する. この計算回数を減らすために粒子の存在する空間を階層化して粒子をまとめるBarnes-Hut法が知られている. 階層化はポインタによるリンクが一般的だが, 本論文ではバイナリキーを使用することによって処理を高速化しようと試みた. バイナリキーは, 空間の座標値を２進数によって表現する. これによってベクトル処理が容易になる.

プログラムはFortran90を用いて記述し, ベクトル並列計算機上に実装した. 評価はこのプログラムに対するもので, 重力計算の所要時間, 組み合わせの所要時間及び台数効果について行なった. 重力計算については, 粒子数10万に対して約７分の結果を得た.

To Speed-up Gravitational N-body Problem Solving by Parallel Processing

Keizo Nonaka, Hiroaki Hirata, Haruo Niimi, Kiyoshi Shibayama
Kyoto Institute of Technology

Abstract

The Barnes-Hut algorithm is the best known way to solve the gravitational N-body problem. It groups the particles by hierarchizing the space. This study tries to speed up the processing time using "Binary keys." This improves vector processing. The program was written in Fortran90 and executed on VPP300, a vector parallel computer. The gravitational calculation time for 100,000 particles was recorded to be 7 minutes.

1. はじめに

重力多体問題はニュートン力学に基づいている．すべての粒子に対して

$$\vec{F}_i = -Gm_i \sum_{j=1}^{n} \frac{m_j(\vec{x}_i - \vec{x}_j)}{|\vec{x}_i - \vec{x}_j|^3} \qquad (1)$$

の計算を行ない，時間を微小時間前進させて系の変化を追う．この問題は粒子数が増えるとその計算回数が急激に増大するため高性能の並列計算機を長時間使用しないと解くことができない．

この計算回数を少なくするために粒子の含まれる空間に対して階層化を行なう，Barnes-Hut 法 (以下 BH 法) というアルゴリズムがよく知られている．BH 法ではどの粒子と階層中のどのノードを組み合わせるかを決定した後，式 (1) を計算する．

階層化にはポインタによるリンクが一般的であるが，本論文では，ベクトル化を優先させるためにポインタは使用しなかった．代わりに 2 進数の組み合わせによって 3 次元座標を表すバイナリキーを用いた．バイナリキーはビット操作によって容易に作成することができるという利点がある．ベクトル計算機およびベクトル並列計算機にとってバイナリキーは扱いやすいのではないかというのが本研究の着目点である．

すなわち，本論文の目的は「バイナリキーを用いた BH 法は，ベクトル計算機およびベクトル並列計算機と親和性がよく重力多体問題の解法を十分に高速化できる.」という仮説を検証することである．

並列計算機を用いた重力多体問題の解法の高速化に関する研究は，

1. スタンフォード大，SPLASH による多角的な研究

2. 東京大学の専用計算機 GRAPE3：20 万体 1 ステップ 116 秒

3. カリフォルニア工科大，Intel Touchstone Delta 512PE 上でのバイナリキーを用いた研究：880 万体を多極分解 1 ステップ 114 秒

等，数多くある．カリフォルニア工科大の研究はバイナリキーを使用している [1].

2. Barnes-Hut 法

式 (1) において片方の粒子が重くて遠い場合と近くて軽い場合，その比が同じであれば他方の粒子に及ぼす力は同一である．この距離と質量の関係から遠方の粒子はその位置と質量をまとめてグループとして取り扱うことができる。グループ化を行なうためには空間を階層的に分割すればよい．階層化の手順は次の通りである [2].

1. すべての粒子が入る立方体を想定し，これを親セルとする．親セルの各辺を 2 等分すれば親セルは 8 個の子セルに分割される．

2. 子セルの中に粒子が 2 個以上あればその子セル をさらに 8 分割する．

この手順を再帰的に繰り返すことによって，すべての粒子はいずれかのセルに属することになる．セルの親子間をリンクで結ぶと全体はツリー構造をなす．2 次元での例を図 1 に示す．

このツリーを探索してどの粒子とどのセルを組み合わせるかを決定して重力計算を行なう．そのアルゴリズムは，次の通りである．

if (粒子とあるセルが十分離れている) **then**
　そのセル内の粒子の質量の合計とセルの重心位置から重力計算
else
　そのセルを子セルに分割したものそれぞれについて各子セルの粒子の質量の合計と重心位置を求めて重力計算
end if

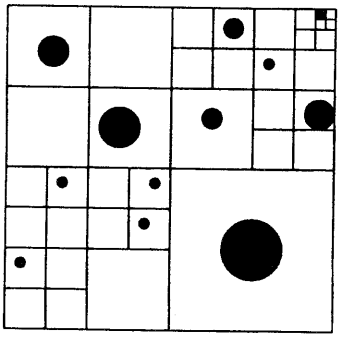

図 1: 分割された空間の 2 次元での表現

粒子とセルが十分離れているかの条件は,

$$l/D \quad < \quad C \qquad (2)$$

で判断する. lはセルの一辺の長さ, Dは粒子とセルの重心の距離, Cは定数で 0.5 ～ 1 である. Cを大きくすると計算回数は減り処理時間は短縮されるが, 誤差が増える. Cを大きくするとその逆になる. 本研究はこの Cを変化させて高速化を図るものではないので中間の 0.75 に固定した.

本来 $O(n^2)$回の計算回数を, このアルゴリズムによって $O(n \log n)$ 回に減らすことができる.

従来の方法では粒子と粒子をループの中で無条件に組み合わせて力を計算する. これに対して BH 法では組み合わせを先に決定したうえで力を計算する. BH 法を用いて重力多体問題を解くには,「セル組み合わせ」と「重力計算」の双方が高速化される必要がある.

3. バイナリキー

プログラミング言語には Fortran90 を使用した. Fortran90 はポインタをサポートしているのでセルの階層構造をポインタで表現することは可能である. しかしポインタの使用はメモリアクセスのパターンの予測が困難になりベクトル化による高速化を期待することができない.

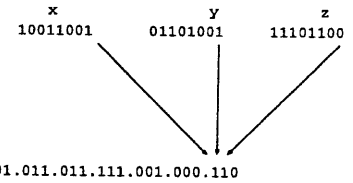

図 2: バイナリキーの作成 (IEEE Supercomputing'93, pp15 より)

そこで粒子の座標値 x, y, z からバイナリキーとばれる 1 個の値を作る [1]. BH 法によって作られたセル階層構造は最上位レベル, 中間レベル, 最下位レベルに分類できる. バイナリキーはこれら各レベルのセルの空間内の位置関係を数値として表す. 数値による位置関係の表現はベクトル化にとって効率的に働く.

バイナリキーの作成は次の手順で行なう.

1. 実数で表わされた粒子の座標値 x, y, z を整数値に変換する.

2. 整数値となった各座標値の 2 進数表現を最下位ビットから順に図に示すように組み合わせる. これを最上位ビットまで繰り返す.

図 2 はこの手順を図式化したものである. バイナリキーのすべてのビットが 0 の時は, 右左いずれ方向へのシフトによっても値が変わらない. バイナリキーの最上位ビットは, これを防ぐために設けた特別ビットであり常に 1 にしておく. これを「プレースホルダビット」という.

バイナリキーを最下位ビットから 3 ビットずつ区切って読めば 8 進数表現となる. このバイナリキーの 8 進表現を「オクタルキー」という. これはちょうど 8 個に分割された子セルのそれぞれに相当する. 最上位のプレースホルダビットは全体空間を表す. いま仮にオクタルキーが 123 であるとすると, 親セル 1 の中の子セル 2, さらにその中の孫セル 3 を意味する. このようにバイナリキー

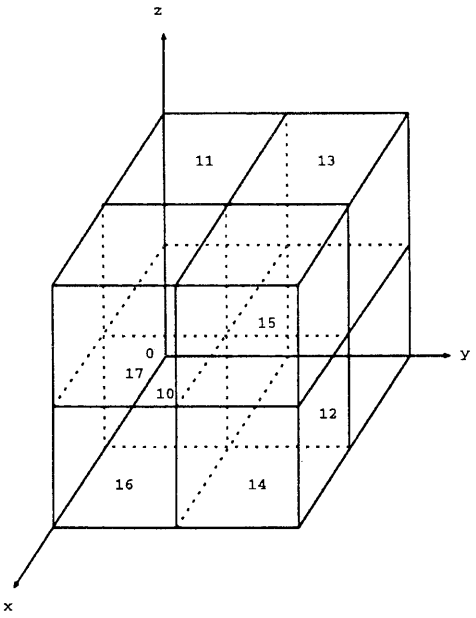

図 3: オクタルキーによる座標表現

はセル空間の親子関係, すなわち系内のセルのトポロジを表す. どの空間にあるどのセルなのか, どの空間に属する粒子なのか瞬時に判別できる. 図 3 にオクタルキーによる座標表現の例を示す.

1 語 32 ビットの場合, プレースホルダビットを含めて 11 桁のオクタルキーを作ることができる. これによって最下層の小セルが 8 の 10 乗個になるレベルまで表すことができるので, 約 10 億個の粒子を含む問題まで対応できる.

バイナリキーを用いるとこれをキーとして粒子をソートすることができる.

4. プログラムの構成

本プログラムは, 次の 3 つの段階に大きく分けられる.

1. 粒子の座標値と質量データの入力
2. 式 (1) の計算
3. 2. の結果のファイルへの出力

本論文では, 富士通株式会社製のベクトル並列計算機, VPP300/16R を使用した. メインルーチンはセル組み合わせに関する処理全体を行なった後重力計算サブルーチンを呼んで式 (1) の計算を行なう. メインルーチンは 1PE で担当させた. これに対して重力計算サブルーチンで使用する PE 台数 (1〜15 台) は任意に決定できるようにした. メインルーチンの全命令行の 20%の行がベクトル化できた. 計算サブルーチンでは 100%ベクトル並列化できた.

メインルーチンの構成は,

1. バイナリキーの作成
2. ソートによる有効なバイナリキーの抽出
3. 中間セル内の質量の総和と重心座標の算出
4. 速見表の指定
5. セル組み合わせ

の 5 パートである.

ソートを行なう理由は有効なバイナリキーを抽出して, 速見表を作成するためである. 速見表は, ツリー探索に代わって階層を横方向に分解し, セル内の質量の総和と重心座標を配列に書き込んだものといえる. ある粒子に有効なバイナリキーが存在するとはこの分解された階層が存在することでもある. つまり有効なバイナリキーが存在する粒子は必ず速見表を有する. ツリー探索の代わりとして速見表を指定するようにした.

前後 2 個のバイナリキー

15267365024 .....B1

15233333333 .....B2

があるとき, 有効なバイナリキーは 152 である. B1 と B2 は, 親セルから 2 回の分割を経た空間までは同一のセルに属することがわかる. ツリー構造でいえば, ルートから第 2 番目のノードまでは共通

ということである. 他のある粒子とセル 152 間の力を計算する時, 予めセル 152 内の粒子の質量の総和と, セル 152 の重心座標とを計算しておく必要がある. このように前後 (2 個以上) のバイナリキーの最上位から連続して重複する桁を有効なバイナリキーと呼ぶ.

粒子の質量の総和の算出は, バイナリキー 152 に 0 を付加した 15200000000 と, 7 を付加した 1527777777 の 2 個のバイナリキーを用意し, すべてのバイナリキーを走査して 15200000000 と 15211111111 の範囲内にあるバイナリキーに対応する粒子の質量を合計すれば求まる. 重心座標は親セルの重心からバイナリキーに従って 8 つの子セルのいずれかを選択していくことで求まる.

各粒子に対してこの速見表を指定した後, 式 (2) の条件に従って組み合わせを行なう. 組み合わせとは, 例えば, 粒子 A, 粒子 B 及び粒子 B に指定されたセルがある時, 粒子 A に対してこのセルと組み合わせて計算するか, 直接粒子 B と組み合わせて計算するかを決定することである.

5. 実行結果と評価

評価の軸として, 1. 重力計算サブルーチンの所要時間 2. セル組み合わせの所要時間 3. 並列処理の台数効果の 3 点を据えた. 計時は, 富士通社の性能測定用のツールを使用した. VPP300 では, usr 時間は稼働 cpu 台数の合計時間を表す. この場合総経過時間 real の方が高速化の評価指標としては適当である [3]. 粒子データは 100,000 体分を乱数を発生させて作成した.

重力計算サブルーチンのみの所要時間は, 重力計算サブルーチンも含めた総所要時間から計算サブルーチンを実行しなかったときの所要時間を引けば求められる. 重力計算サブルーチンの有無による所要時間を 6 ページの表 1 と表 2 に示す. 1PE 実行時, 4PE 実行時ともに約 7 分が計算サブルーチンの所要時間である.

1PE でセル組み合わせまでを行なっているためパート毎の経過時間は PE 数を変化させてもほぼ一定である. パート毎の経過時間を表 3 に示す.

またセル組み合わせのみの所要時間は, 計算を行なわない総時間から速見表の指定までの時間を引けば求まる. 約 11 分である.

重力計算サブルーチンの並列処理の台数効果をみるために, 使用 PE 台数を 2, 8, 15 と変化させて実行した結果を表 4 に示す. 台数が増えるにつれて処理時間全体は増加している. 通信の増加が台数効果を妨げたと考えられる.

6. まとめ

計算サブルーチンのみの所要時間は 100,000 体で約 7 分間という結果を得た. VPP300 のピーク時浮動小数点演算速度からいえば妥当である.

セル組み合わせの前準備が約 1 分である. これに対してセル組み合わせは, 約 11 分の長時間を要する. セル組み合わせは式 (2) の条件判断が中心で逐次処理となる. セル組み合わせは, 速見表の指定と重力計算の間でボトルネックとなっている. 処理全体を高速化するにはセル組み合わせをさらに高速化する必要がある.

残念ながら並列処理の台数効果は現れず, 1PE で実行するのが最も高速である.

参考文献

[1] M.Warren, J.Salmon : "A Parallel Hashed Oct-Tree N-Body Algorithm", Supercomputing '93 (1993).

[2]J.Barnes, P.Hut : "Hierarchcal *NlogN* force-calculation algorithm", Macmillan Journals (1986).

[3] 富士通 (株)HPC 本部 : "VPP FORTRAN プログラミング " (1996).

表 1: 計算サブルーチンの有無 (1PE)

単位 [分:秒. ミリ秒]

	有	無	差
real	18:37.61	11:51.20	6:46:41
user	17:21.98	11:01.83	6:20.15
sys	0:30.33	0:22.75	0:07.58

表 2: 計算サブルーチンの有無 (4PE)

単位 [分:秒. ミリ秒]

	有	無	差
real	19:40.67	12:16.42	7:24.25
user	77:28.34	43:27.48	28:00.86
sys	1:19.63	0:52.32	0:27.31

表 3: パート毎の所要時間

単位 [分:秒. ミリ秒]

バイナリキーの作成	35,962	36,217	255
バイナリキーのソート	45,283	45,664	381
質量と中心座標の算出	58,343	58,613	270
速見表の指定	58,511	58,774	263

表 4: 並列処理の台数効果

単位 [分:秒. ミリ秒]

	2PE	8PE	15PE
real	19:02.74	20:35.20	24:18.36
user	34:52.04	149:09.40	331:57.51
sys	0:46.28	2:31.65	5:17.91