GRAPE 上の並列ツリーコード

牧野淳一郎

概要

1. 動機
2. 従来の並列化アルゴリズム
3. GRAPE との組合わせ
4. 新しい並列化アルゴリズム
5. 実装上の問題
6. 性能評価

動機

これまでの宇宙論等の大規模 N 体 / SPH 計算のほとんどが

- 汎用の並列計算機 (T3E 等)
- ホスト 1 台 + GRAPE

で行なわれている
並列ホスト+GRAPE ならどちらよりも規模が大きな計算
ができるはず。

どれくらいのことができるはずか？

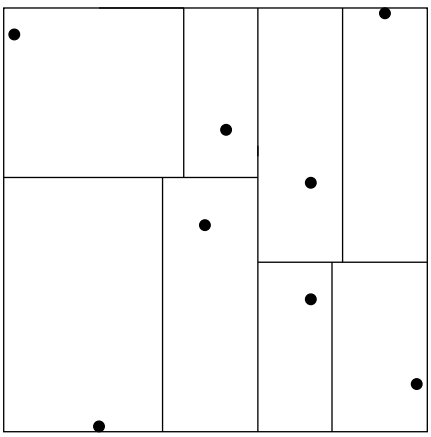
GRAPE-5/6 と現状の Alpha: 1M 粒子 1 step 10 秒 (計算精度にあまり依存しない)
Alpha だけ: 1M 粒子 1 step 200-1000 秒
T3E 1024 CPU: 100M 粒子 1 ステップ 30-200 秒
GRAPE-5/6 なら 10 台程度でこれと同等の速度
三鷹 16 台、本郷 8 台
メモリ：三鷹は 32 GB 200M 粒子くらいはできる

従来の並列化アルゴリズム

実用になっている並列化法は以下の 2 つ。どちらももと Caltech Hypercube のグループの Salmon と Warren によるもの

- **Orthogonal Recursive Bysection (ORB)**
- **Hashed Oct Tree (HOT)**

ORB

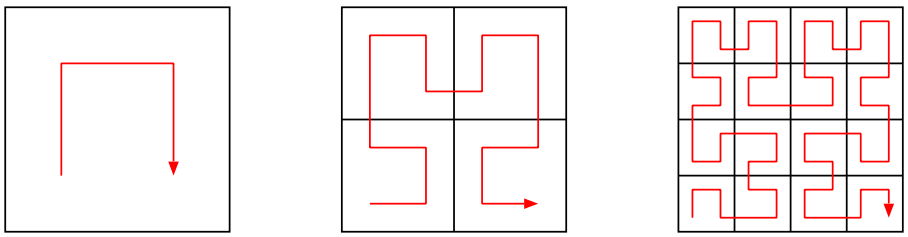


- まず、粒子が半分ずつにわかれるように x 軸に垂直な平面で切る
- 切ったそれぞれについて、また半分になるように y 軸に垂直な平面で切る 3 次元なら次に z、2 次元なら次は x 軸になる

というのを、プロセッサ数になるまで繰り返す

HOT

ペアノ曲線：N 次元空間を 1 次元にマッピング



- 粒子をペアノ曲線上の順番で並べる
- 各プロセッサに連続した粒子を割り当てる

どっちがいいか？

- **ORB**
  - 実装は簡単。速度もまあまあ
  - プロセッサ数が 2^n でないといけない
- **HOT**
  - プロセッサ数は任意
  - なんだかややこしい。速度は？

普通の計算機上では性能は似たようなもの

GRAPE を使う場合に考えるべきことは？

通常の並列計算機
計算時間のほとんどはツリーをたどって重力を計算するところ
そこさえ速くなれば、他のところはあまり速くならなくても構わない。

GRAPE
ツリーを作るところも無視できない (全体の 2-3 割)
この部分も効率良く並列化されないといけない
通信に時間が掛かったりするのは嬉しくない

今回使うことにした方法

ORB を多少変更

- 深さを 3 層までに制限
- 2 分割以上も許す

例えば、 12 ノードなら $3 \times 2 \times 2$ とかいうふうに分ける。
もちろん、 2, 4, 8 ノードでは従来の ORB と同じ。

空間分割の実際

ORB も実装はいろいろ。良く知られているのは Dubinski の論文によるもの。粒子はすでに各プロセッサに適当に分配されているとする。

1. まず 2 分法で分割点を捜す。各プロセッサで粒子を数えた後、関係するプロセッサ全部での合計をとって新しい分割点を決める。
2. 各プロセッサで分割点の右と左に粒子を振り分ける
3. 各プロセッサは、 ORB ツリーの現在の階層で自分とペアになるプロセッサと粒子を交換する

というのを、階層分繰り返す。
この方法の問題点：計算量、通信量（回数）ともに大きい。

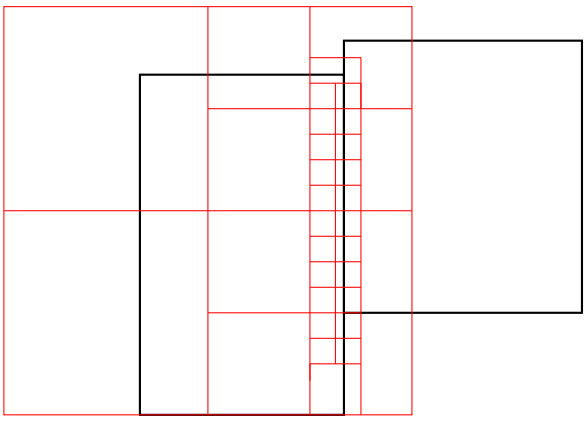
空間分割の実際2

今回使った方法： Brackston & Suel によるものを踏襲

- 各プロセッサは適当に選んだサンプル粒子のセットを ノード 0 に送る。
- ノード 0 は、集めたサンプルセットを使って分割を生成する。できた分割は全ノードに放送する。
- 各プロセッサは、もらった分割を使って粒子を振り分けて、担当するプロセッサに送る。

分割に多少の誤差が入るが、計算時間としては問題にならない程度。
途中で受信バッファ領域が足りなくなった時の処理が面倒

他のプロセッサにある粒子からの力の計算



他のプロセッサから、「必要ないところをカットした」ツリーをもらってくる (local essential tree, LET)
それらと自分のところを適当にまとめて全体ツリーを作る

「適当にまとめる」って？

Dubinski の実装：この上の階層は ORB ツリーそのもの
まあ、これで問題はないけど、実装が面倒

- ORB tree のところと local tree のところでツリーの構造が違う
- LET はツリー構造 (ポインタとか) もちゃんと保持して送る必要あり

今回の実装

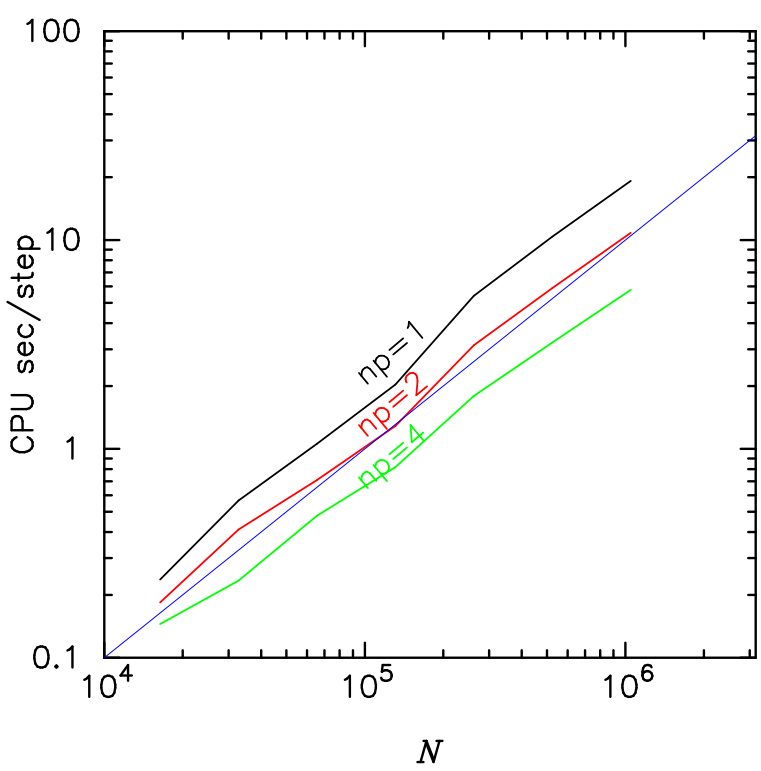
- **LET** ではなく、**LET** の最下層にある「粒子」だけを送る。
- 受けとった「粒子」を、自分のツリーに挿入する。

ツリーに粒子を挿入：もともとの **Barnes** のプログラムで使われている木の作り方。実装は簡単
通信量は最小
計算量は多少増える。

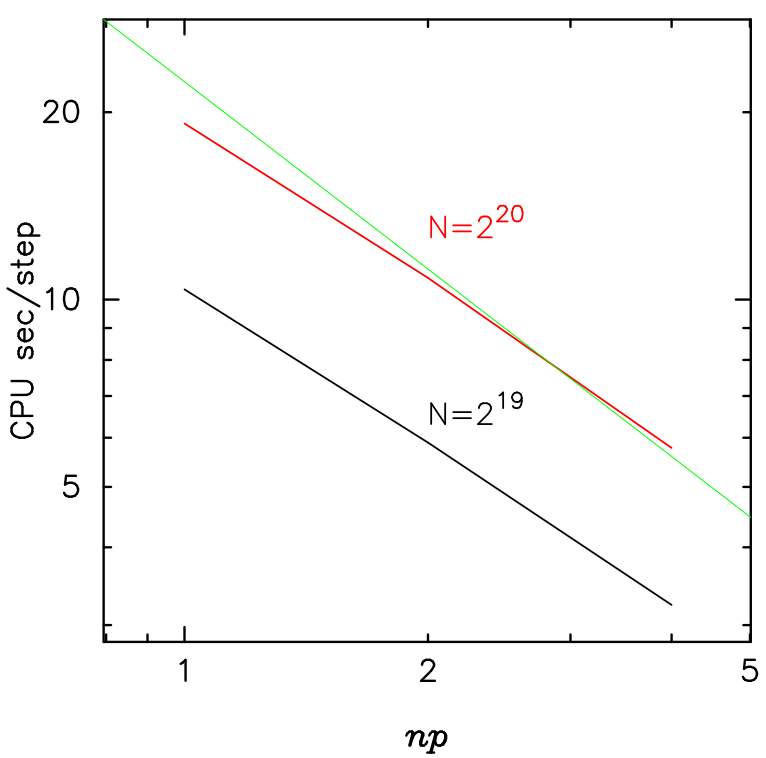
実装と評価

前に書いた C++ 版のコードをいじり回して MPI 版を作った。元のコード：2600 行。 MPI 版： 4300 行。
とりあえず GRAPE-6 対応にして、手元の GRAPE-6 クラスタで性能評価した。
通信は 100BASE-T 上の MPICH/p4。バースト転送速度は 10MB/s, レイテンシは 100 μs 程度。
1, 2, 4 CPU でしかやってないので、実質は ORB と同じ。

計算速度



並列加速率



結果まとめ

- 計算時間は粒子数に比例。特におかしいところはなさそう
- 1CPU であんまり速くない。これは主には P4+g++ の組合せが遅いため。ほかにもまだチューニングの余地はある。
- 並列効率はまあまあ。 $1 \to 2$ であまり速くならないが、これは1 ノードでは全くなかった通信とか並列化のための仕事が発生するためなのでやむをえない。$2 \to 4$ は悪くない。

もうちょっと細かい時間分析

4 ノード、 1M 粒子

項目	時間 (1 ノード, 秒)	時間 (4 ノード)
時間積分等	**0.831**	**0.212**
空間分割	**0**	**0.292**
ツリー構築 (LET 含む)	**5.523**	**1.721**
加速度計算	**12.875**	**3.549**

ツリー構築のうち LET 関係： 0.3 秒程度
local tree をつくるところ自体が遅い。

まとめ

- GRAPE で動く並列ツリーコードを作った。
- プロセッサ数は基本的には任意である。
- GRAPE-6 クラスタではネットワークが 100BT でも
  良好な並列加速率を実現した。
- 天文台の GRAPE-5 クラスタでも性能評価したい。