# GRAPE-6 の次 — 主に計算法の面から

牧野淳一郎

# 概要


1. 何を考えるか
2. 独立時間刻みとツリー法
3. GRAPE と ツリー法
4. 三題話
5. まとめ ( るのはちょっと無理かも )


要約：


GRAPE-6 の次ではやはり必要になってくると思われる、ツリー法と独立時間刻みを組み合わせて高精度・高速な計算を実現する方法について、予備的な検討をする。

# 何を考えるか

**GRAPE-6 の現状：**

**ツリー法では shared time step**

独立時間刻みでは高精度直接計算

原理的には、大粒子数で空間構造も発達する系 ( あるいはソフトポテンシャルが使えない系 ) の高速計算は結構大変

## 大変というのは本当か

**GRAPE-6 の単体 ( 箱一つ ) 性能: 8 Tflops。**
**フロントエンド ( 1CPU として ) の実効性能 : 100 Mflops。**

**独立時間刻みの処理速度**
**FE: $4 \times 10^5$ particles/sec (だいぶ楽観的)**
**GRAPE-6 : $1.4 \times 10^{11}$ pairs/sec**

- **40 万体を超えると GRAPE-6 の方が遅い**
- **ホストと GRAPE-6 の価格差: 1:50**

**100 万体程度の計算ではホストの数を増やす必要があるような**
**ややこしいことをしてもスループットがあげられれば得になる**
**可能性がある。**

## GRAPE の方の仕事を減らす方法は？

汎用計算機で使われている / 検討されているアプローチは次の 2 つ。

1. ネイバースキーム
2. ツリー法

ものになるかどうか、というより、 GRAPE と組み合わせられるかどうか考えてみる。

# ネイバースキーム

大体どういう計算法だったか？

- 各粒子が近接粒子のリストを持つ
- 普通のステップでは近接粒子からの力だけを計算
- 全体からの力を計算する時に、近接粒子のリストを作り直す
- 原理的にはタイムステップの比が $(\text{リストの長さ})^{1/3}$

# ネイバースキームのハードウェア化

問題：

- 近接粒子リストをどこにどうやってしまうか
- しまったものをどうやって使うか

- パイプラインが一本なら話は簡単：単に大きなメモリをつけておけばいい。
- 複数パイプラインでも、力を受ける方の粒子がみんな同じなら話は同様
- 複数のパイプラインで別々の粒子への力を計算するのは???

# 複数パイプラインとネイバーリスト

リストを作る方は問題ない。ネイバーリスト用の比較的小さなメモリユニットがパイプラインの数だけあればいい

使う方は問題。

複数のパイプラインはそれぞれ違う粒子からの力を計算したい

↓

粒子データをしまうメモリ自体が複数ポートからの独立な読みだしを許す必要がある

キャッシュとかいろいろ考えると面白いけど、複雑過ぎ

# ネイバーリスト用ハードウェアのまとめ

- 小規模なハードウェアならいろいろ工夫の余地はあるかもしれない
- パイプライン本数が非常に多い時に、メモリバンド幅を沢山用意しなくてよいような方法があるかどうかよくわからない。

# ツリー法+独立時間刻み+GRAPE

ツリー法と GRAPE の組み合わせ自体は昔からある。(Makino 1991)

基本的な考え方:

物理的に近くの粒子に対して一回だけツリーをたどり、同じリストを使って複数の粒子への重力を計算すること。

独立時間刻みでない限りまあまあうまく働く。

## 独立時間刻みの時の問題

ある時刻で動かす粒子数は少ないので、普通にやる ( ステップ毎にツリーを作り直す ) と

- ツリーを作る
- ツリーをたどる

のそれぞれに時間が掛かり過ぎる。

ツリーを作るのもたどるのもなるべく少なくできないか？ということを考えてみる。

現在の宇宙論用ツリーコードは実は大体これ： **GRAPE** を使ってないので別にこのへんに時間が掛かっても構わない。

## ツリーを作る方の回避

基本的にはしばらく同じものを使い回すことで回避できる。

使い回している間に粒子が動く：相互作用リストを作る時にどれくらい動いたかを考慮して作ればいい。

もうちょっと具体的には：セルの大きさを、現在の粒子の広がりでもって計算しなおす。

もちろん、あまり使い回す時間が長いと結局相互作用リストが非常に長くなって計算量がかえって増える。ので、適当なところで手を打つ。

## ツリーをたどるほうの回避

ツリー構造を変えないなら、相互作用リストも変えなくたって構わないのではないか？

使い回している間に粒子が動く：相互作用リストを作る時にどれくらい動くかを考慮して作ればいい。

細かいチューニングはいろいろ考えられる

- 速度が速い粒子はツリーにいれないで直接計算する
- 局所的にツリーを作り直す

## 目をつぶっていた問題

- 相互作用リストは誰がどこに持つのか？
- リストはどうやって使うのか？

この辺の問題はネイバーリストと本質的には変わらない。

本質的ではないとはいえ定量的には重要な違いはいろいろある。以下、そのあたりを思いつくままに考える。

## 相互作用リストとネイバーリストの違い

- 相互作用リストは多数の粒子が同じものを見る ( ようにできる )。
- 相互作用リストは全部合わせてもそれほど大きくない。
- ツリー法のほうが、ネイバースキームよりも効率が高い ($N$ が大きければ)

## 相互作用リストの共有

**GRAPE-5** で現在使っているグループの平均サイズ： **2000** 程度。

- 計算時間のグループサイズへの依存性は弱い
- **GRAPE** が速いほど最適なグループサイズは大きい

**GRAPE-6** や、その次の機械での最適なグループサイズは $10^4$ 以上になるものと思われる。

これくらい大きければ、独立時間刻みでもなんとかならないか？

## 共有リストと独立時間刻み

同じグループの粒子は空間的に近い

↓

例えば球状星団コアのような高密度領域は数個のグループで
カバーできる。

↓

特に非常にタイムステップが短い粒子は比較的少数のグループ
にしかない。

（楽観的に考えると）GRAPE で一度に1つのグループへの
力しか計算できなくてもなんとかなるかもしれない。

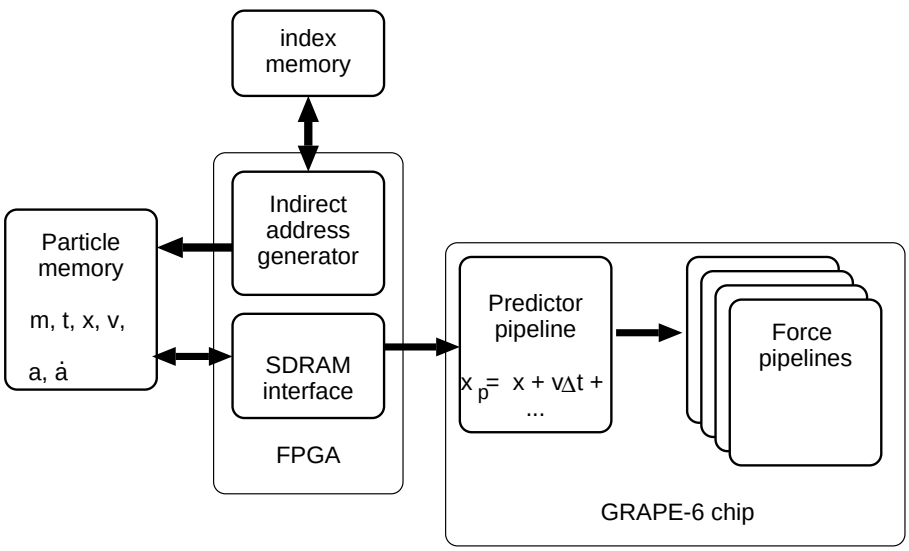
## 相互作用リストの大きさ

Makino 1991 の怪しい理論式では、$n_g$(グループの粒子数)が $10^4$ を超えるとリスト長 $n_t$ は $n_t \sim 2n_g$ くらい。

まあ、もっと小さい $n_g$ で使うとしても、 $10n_g$ もとってあれば十分であろう。
1 粒子当たり 40 バイト程度。これはもともとの予測子データと同程度。

# ハードウェアの実際

## とりあえず、パイプラインチップ一つのもの：



## ハードウェアの実際 (続き)

とりあえず、今の GRAPE-6 チップをそのまま使うとして、

- 全粒子 ( グループ ) 用の相互作用リストを持たせる ( 容量 $20N$ 語程度 )
- 粒子の他、ツリーのノード全部の予測子も持たせる。
- メモリは SDRAM かなんかを使う ( 安いので )
- メモリコントローラは FPGA チップ

256MB の SDRAM モジュールをつければ 200 万体くらいまでは 1 モジュールに入る。

# アルゴリズムと並列化

1チップの絵がでてきてもよくわからないので、以下

- どうやって使うか
- 並列化

の話。

# どうやって使うか(1)

- 全粒子が同期したタイミングでツリーを作る。
  - 粒子をグループわけし、各グループについて相互作用リストを作っておく。
  - 作ったツリーと相互作用リストを GRAPE 側に送っておく。
  - このステップでの重力計算は、従来のツリー法と同じ。

## どうやって使うか(2)

- 以下を、次に全粒子が同期するまで繰り返す。
  - 次の時刻を決める
  - 次に動かす粒子を決める。
  - 次に動かす粒子を、もともとの相互作用リストに対応するグループにわけ、それぞれ相互作用リストを使って重力を計算する。
  - 重力が求まったら、軌道積分して粒子とその上のノードの予測子を作り直し、 GRAPE のほうの予測子データを更新する。

## 通信コストの分析

ツリーを作り直さないステップでの通信：基本的には $O(n)$ ($n$ は同時に積分する粒子数)。ノードの予測子の更新をいい加減にやるとコストがかかる。

ツリーを作り直すステップでの通信：基本的に $O(N)$。増えるのはリストの分だけ。

というわけで、そんなに通信は増えない。

ホストの仕事自体はいろいろ増える。
もっとも大きいのは、( 多分 ) 粒子を積分した時にその親 ( 祖先 ) を全部更新する必要があること。これは無駄な更新をしないような工夫が必要。

## 並列化 — GRAPE 側

多数のパイプラインを使って一つの粒子 ( 一つの相互作用リスト ) を並列に計算するのが実際には必要。

リストを分割する → 粒子メモリは重複して必要になる。

粒子・ノードデータ自体を分割する → 確率的には load imbalance が発生。

# GRAPE 側並列化 ( つづき )

なるべく均等になるように分ける

DFS 順でノード、粒子を 1 次元的に並べて、それを「横」に
切って複数パイプラインに配置する

例えば、自分のグループの中の粒子は均等になる。

他は大したことは ( 多分 ) ない。

## 並列化 ( ホスト側 )

- 共有メモリの並列 CPU 機なら特に考えることはない。
- 分散メモリ、、、ちょっとまだ考えたくない。

## まとめ

- とりあえず、紙の上では、 tree+GRAPE+individual timestep というのは出来なくもなさそうである。
- 基本的な思想は、相互作用リストを時間方向にも使い回すため、GRAPE のほうにそのメモリをつけるというもの。
- ハードウェアは、現在の GRAPE-6 チップはそのままでメモリ容量を増やし、さらにもうちょっといろいろ制御するための FPGA をつければなんとかなるような気がする
- 複数パイプラインでの並列化も紙の上ではいけそうな気もする。
- 次のステップは実際のコード ( GRAPE は使わないで ) を使ったもうちょっと精密な評価になる。