専用計算機による計算科学

牧野淳一郎
東京大学理学系研究科天文学専攻

大雑把な予定

1. 何故専用計算機を考えるか？
2. 専用計算機とはどのようなものか？
3. GRAPE プロジェクトの歴史
4. 専用計算機が「うまくいかない」のは何故か？

専用計算機を考えるのは何故か？

それ以前に、、、

そもそも「計算科学」とは何で、何を目指すものか？

というような話をするべきかもしれないですが、

- シミュレーション
- 数値実験

といっても、その目的、意味は対象分野によって非常に違う。

以下、理論天文学での話。

天文学における多体シミュレーションの役割

- 観測の解釈 — 理論＋モデル計算
  知られている物理法則から天体現象を理解する
- もうちょっと原理的な問題
  カオス
  熱力学

そもそもどんな対象を考えるか？

大抵の天文学的対象：自己重力系
一つの星：重力とガスの圧力がつりあう
それよりも大きい構造：　重力を構成要素の運動エネルギーで支える

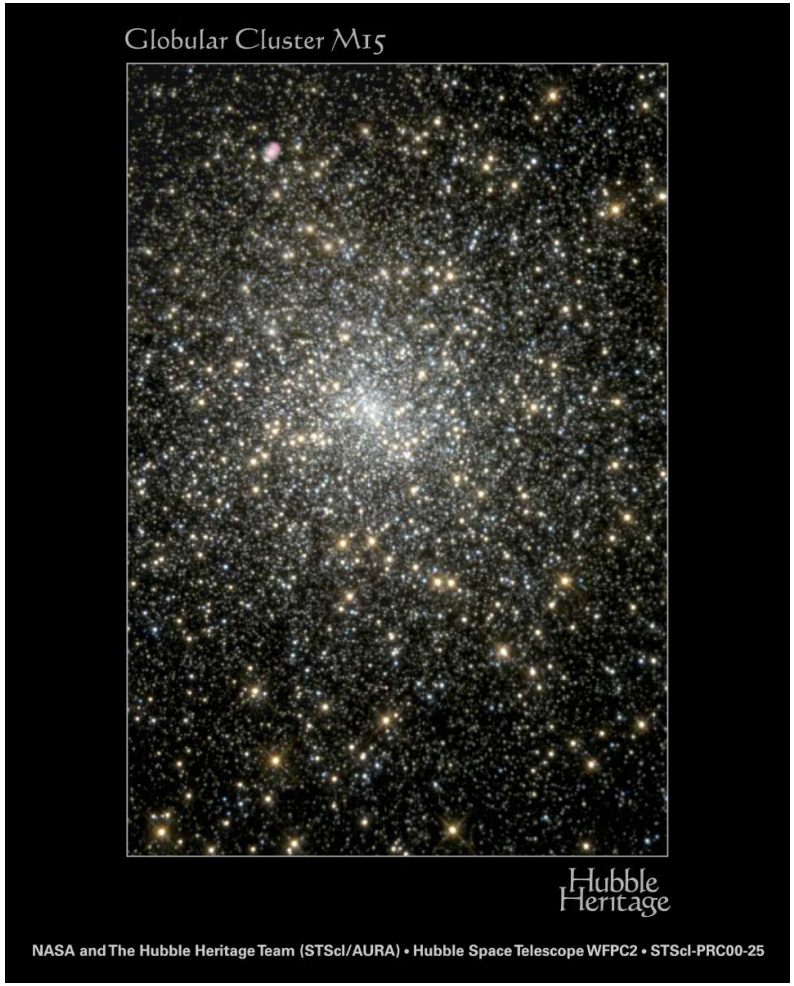
- 太陽系
- 星団
- 銀河
- 銀河団
- 宇宙の大規模構造

例: 星団

- 球状星団
- その他、丸い星団

丸い: ある程度熱力学に緩和していることを意味する。(円盤銀河なんかとは違う)
しかし、熱平衡状態ではない (自己重力系には熱平衡状態は存在しない)

球状星団 M15



地球からの距離 10kpc (1pc: 3 光年くらい)
星の数: 200 万くらい
質量: 太陽の 50 万倍
半径 (質量の半分がある): 4 pc

球状星団 G1

アンドロメダ (M31) にある、局所銀河団で最大の球状星団

質量:
見積もり (a) 1.5×10^7 太陽質量 (Meylan et al. 2001)
見積もり (b) 8×10^6 太陽質量 (Baumgardt et al. 2003)
2 倍の違い: 力学モデルの違いから。

球状星団を研究する理由

沢山あるが、、、

- 銀河でもっとも古い星の集団 — 球状星団がどうやってできたかは銀河自体がどうやってできたかと関係
- 出来てから星形成やガスとの相互作用が基本的にはない — 理想的な「自己重力質点系」に近い
- より複雑な銀河中心等を理解するための鍵
- 激しい星形成領域 — 比較的大きな星団になっているものが多いらしい。何が見えるか、何が起こるかを理解することは星形成過程の理解にも重要。
- 中間質量ブラックホール ???

もうちょっと原理的な問題

重力だけで相互作用する質点の集まりには一体なにが起きるのか？

粒子数 $N = 2$: ニュートンが解決
$N > 2$ ポアンカレが「一般には解析解はない」ことを示した

球状星団：$N \sim 10^6$
銀河: $N \sim 10^{11}$
なにが起きるか？
数値計算なら、初期条件を与えれば答はでる。

計算すればなんでもわかるか？

原理的には答は **YES**。
現実の問題としては **NO**。

- 計算機の能力 (今日の話の主題)
- 素過程の理解 (次に簡単に触れる)

素過程の理解 — 例:銀河の進化

典型的な銀河: 1000 億個くらいの星、それと同程度の質量のガス、その数倍の質量の「ダークマター」からなる (ということになっている)

- 1000 億体問題は今の計算機では全く無理
- 1000 億体問題が解けても、それだけでは駄目
  - ガスから星ができる過程
  - 超新星爆発等で星がガスに戻る過程
  - ダークマターはどうするか

球状星団は比較的こういう問題が少ない、クリーンな系。

解く方程式 — 重力多体系の基礎方程式

もとの方程式自体はもちろん、各粒子の運動方程式

$$\frac{d^2 \boldsymbol{x}_i}{dt^2} = \sum_{j \neq i} G m_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^3}, \tag{1}$$

数値計算は基本的にはこれを使う

何故多体問題を直接数値計算するのか？

もうちょっと違う方法

- 解析的ななんか
- 分布関数に対するフォッカープランク方程式

現実問題として、、、

- 解析的には解けない
- フォッカー・プランク近似は正しいとは限らない
- 多次元ではフォッカー・プランク方程式は計算量的に解けない

で、多体問題を直接数値積分すれば、「原理的には」なんでもわかるはず。

数値計算の方法等

この種の問題：基本的に

- より大粒子数で
- より正確な

計算をすることで、「新しいことがわかる」(こともある)。

↓

どうやって今までより「良い (大きく、正確な)」計算ができるようにするかが問題

「良い」計算をする方法

基本的な事実: 速く計算できればそれだけ良い計算が可能になる。(例は時間に余裕があればいくつかあげたい)
それしか方法がないわけではないが、「速く計算できるようにする」のは極めて重要な方法。

- 計算法を改良する
- 速い計算機を買う
- 速い計算機を作る

計算法

少しだけ計算法の話を。
原理的には、多体シミュレーションはとっても単純：
運動方程式

$$\frac{d^2 \boldsymbol{x}_i}{dt^2} = \sum_{j \neq i} G m_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^3}, \tag{2}$$

を数値積分するだけ。

右辺を計算するプログラム： 2 重ループで 10 行くらい
時間積分： なにかルンゲクッタとか適当なものを使えばいい

というだけで話が済めばいいけれど、もちろん世の中はそんなに簡単ではない。

何が問題か？

- 計算精度の問題： 2 粒子の近接散乱、自己重力による構造形成 — 時間刻みをどんどん短くしないとちゃんと計算できなくなる。
  積分時間が長いので高精度の公式を使いたい。
- 計算量の問題： 右辺の計算量が $O(N^2)$ — N が少し大きくなるとすぐに計算時間が現実的ではなくなる

というわけで、どんな方法を使っているかという話を簡単に。

計算法 — 時間領域

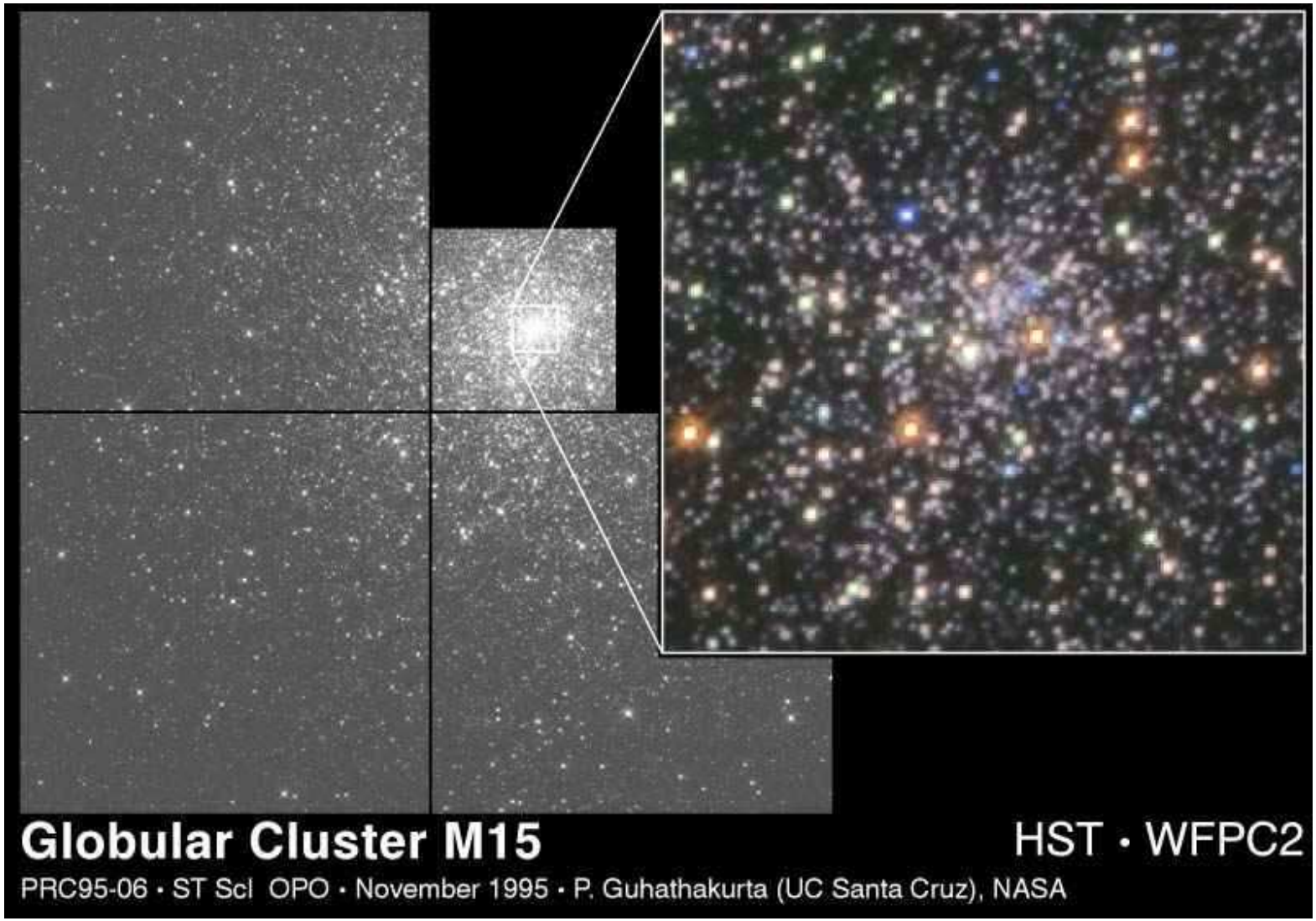
算数としては、単に常微分方程式の初期値問題の数値解。
ナイーブに考えると、いろんな公式がライブラリであるので、
それを使えば済みそうな気がする。
それだけでは済まないのが問題。
済まない理由:

- 粒子によって非常に大きく軌道のタイムスケールが違うことがある
- 連星とかそういったものができる

軌道タイムスケールの問題

- 構造形成による効果
- 一様な系でも起きる問題

構造形成による効果：球状星団の例



構造形成による効果 (続き)

- 「コア崩壊型」星団 — 星の数密度が中心までべき (半径の -1.8 乗) で増加
- 中心にブラックホールがある？

星の運動のタイムスケール: 中心に近いほど短くなる。
等温カスプ (密度が半径の -2 乗に比例): タイムスケールの分布がべき乗になる。

一様な系でも起きる問題

重力が引力であるために確率的には2つの粒子がひじょうに近付くような近接散乱が起こる

インパクトパラメータが 0 に近い2体衝突：非常に短い時間刻みが必要

これは重力多体系特有の問題：相互作用が引力で、しかも距離 0 で発散するため。

例えば分子動力学計算ではこういう問題はおこらない。

計算量への影響

単純な可変時間刻みでは計算量が大きくなりすぎる。

理由: どちらの場合も、タイムステップの分布がべき乗的なテイルをもつようになる。

粒子数が増えるに従って、タイムステップが短くなる。
構造形成の効果: 最悪 $O(N^{1.3})$
2 体衝突の効果: $O(N^{1/3})$ 程度

対応:

- 粒子毎に時間刻みをバラバラに変化させる。(独立時間刻み)
- 2 体衝突、連星は座標変換して扱う。

独立時間刻みの原理

粒子毎にばらばらの時刻 t_i と時間ステップ Δt_i を与える

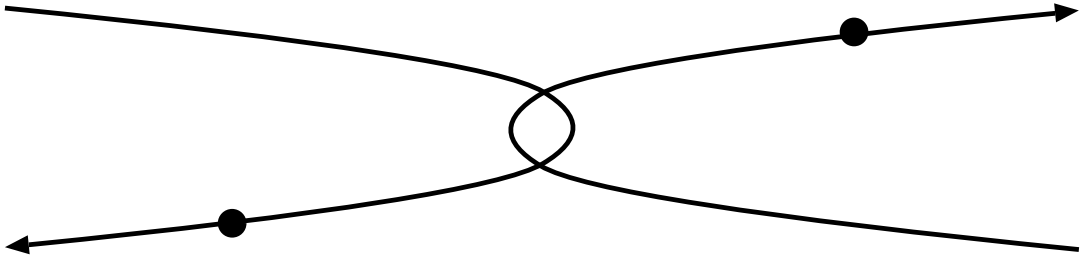
1. $t_i + \Delta t_i$ が最小の粒子を選ぶ。
2. その粒子の軌道を新しい時刻まで積分する。
3. その粒子の新しい時間刻みを決める。
4. ステップ 1 に戻る。

ある粒子の時刻 $t_i + \Delta t_i$ で他の粒子の位置が高精度で必要:
予測子・修正子型の公式を使う。

近接遭遇

連星に限らず、2 つの星の距離が極端に近づくと計算が破綻する (相互作用ポテンシャルが発散するので数値誤差も発散する)。なんらかの対処が必要。

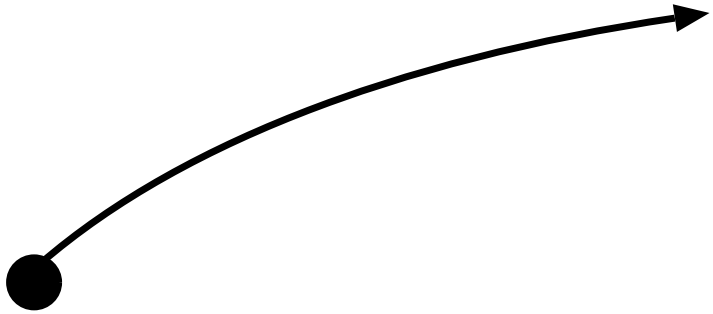
原理的には “straightforward”: 2 体が十分近づいたら、解析解 (またはそれ+摂動) でおきかえればいい。



具体的には

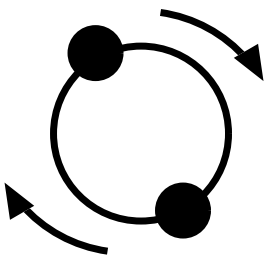
- 2つの粒子が近付いたかどうかをチェックする
- 近付いていたら、その2粒子の重心および相対位置(内部運動)を新しい変数として時間積分する
- 相対運動に対して、回りの粒子からの摂動が十分に小さければ数値積分しないで解析的に解く
- 2つの粒子が十分離れたらその重心と相対位置から元の粒子を復元する

連星



普通の星:他の星全体が作るポテンシャルの中を運動

連星: 2 つが重力的に結合できかた:



- 生まれる時に連星 (結構多い)
- 3 体相互作用

連星と計算量

連星は軌道周期が短い
連星の熱力学: ほぼ「等温」のバックグラウンドに埋めこまれた自己重力系
一旦周りよりも「温度」が上がると、周りに熱を与えて無制限に温度が上がる。
(3体相互作用作用でエネルギーを与える)
周りと温度が等しい $=$ 軌道長半径 $\sim$ 系の大きさ$/N$
軌道周期が他の星の $1/N$
軌道周期が他の星の $1/1000N$ くらいまでは系内にいられる。
こんなのを計算してたら日が暮れるどころではない。

連星の扱い

十分コンパクトな連星で、回りの粒子からの摂動が無視できるほど小さい場合：要するに単に摂動を無視すればよい

どれくらい摂動が小さいなら無視していいか？ ―目的による

エネルギー：断熱不変量

角運動量：そうではない

角運動量について正確な結果を得るのは現状では難しい

弱い摂動の扱い

まともにやるなら、軌道要素で表して、外力を軌道平均して、、、

画期的に安直な方法

- 摂動論的扱いができる範囲内で連星の軌道周期を遅くする
- 摂動項は、実時間での軌道要素の変化率が同じになるようにスケールして運動方程式に入れる。

摂動論の範囲内では摂動論と同じ結果を、軌道を直接数値積分して得ることができる。
(平均運動共鳴は扱えない)

うまくいかないケース

階層的な3重星：連星の外側をもう一つ星が回っているようなもの

- 安定条件は数値的にしかもとまっていない。
- 共鳴が効くので時間のスケーリングはできない

いろいろ研究中

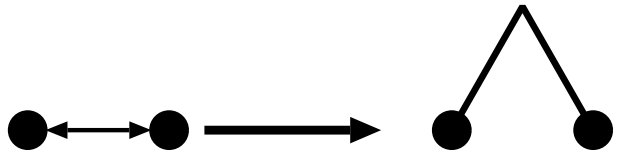
2体以上の系の扱い

例えば3体の系を重心運動と内部運動に分離するにはどうするか？

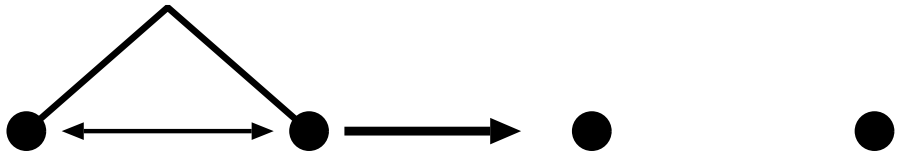
- 「3体」として認識、それ用の正則化
- 2体+1体としては認識、再帰的に扱う

今のところ広く使われているコードは前者
後者のコードもそれなりに動くようになってきている

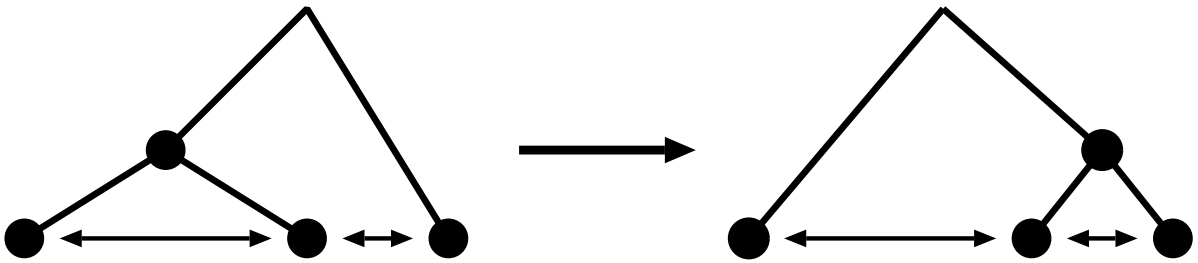
構造の組み換え



$r < r_{close}$



$r > \gamma r_{close}$



計算法 — 空間領域

運動方程式の右辺をどうやって評価するか？という問題。

以下、独立時間刻みのことはとりあえず棚上げにして話をすすめる。

広く使われている方法： **Barnes-Hut treecode**

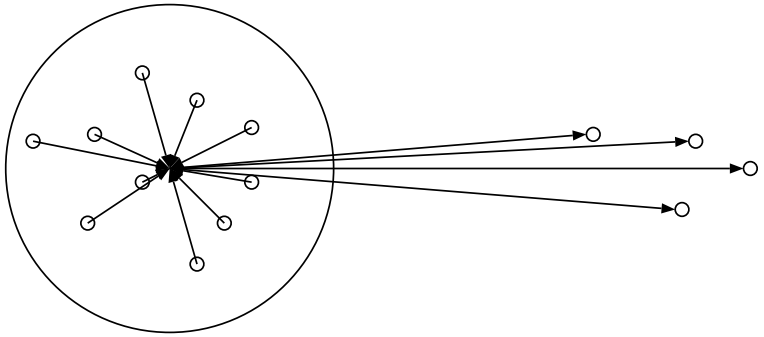
有名な方法： 高速多重極法

ツリー法、FMMの基本的発想

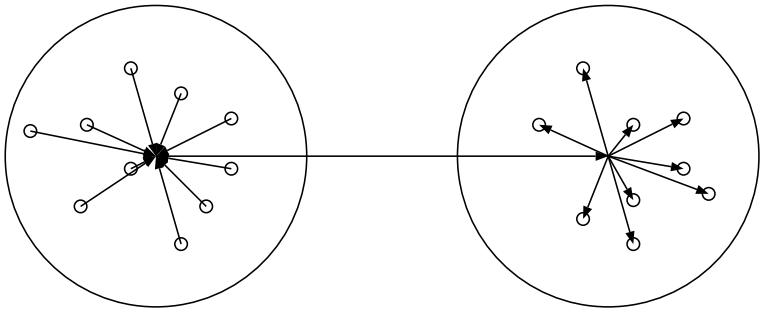
遠くの粒子からの力は弱い

↓

まとめて計算できないか？



Tree



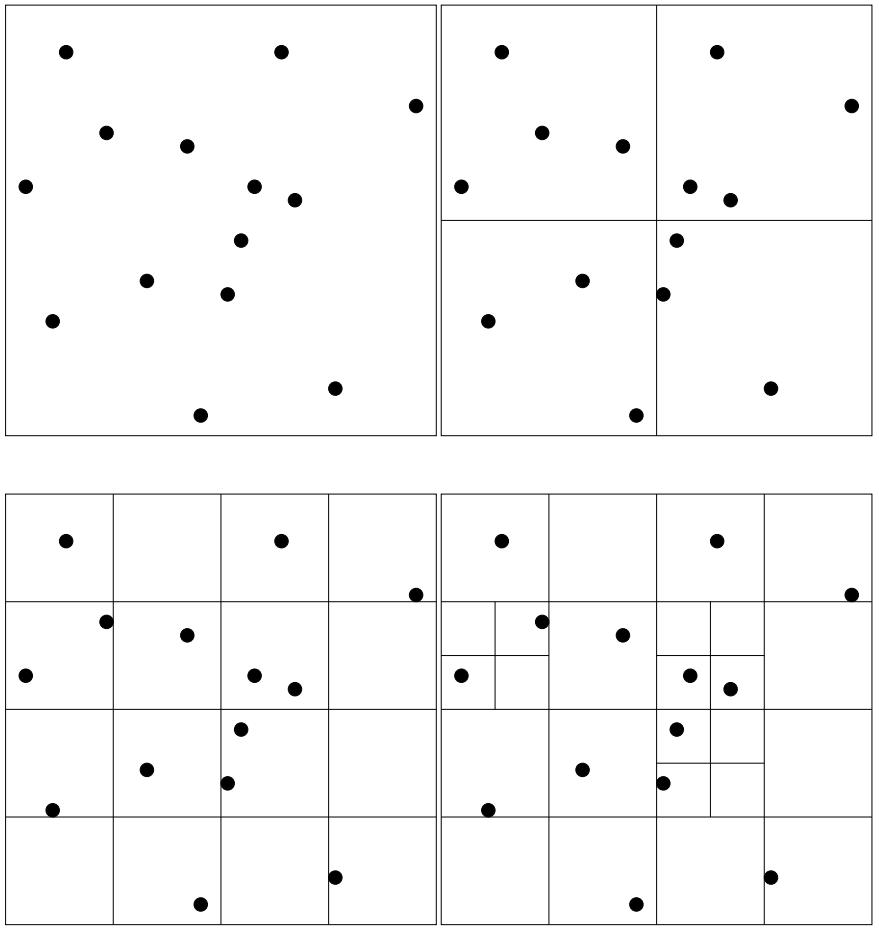
FMM

- ツリー：力を及ぼすほうだけをまとめて評価
- FMM：力を受けるほうもまとめて評価

どうやってまとめるか? — ツリー法の場合

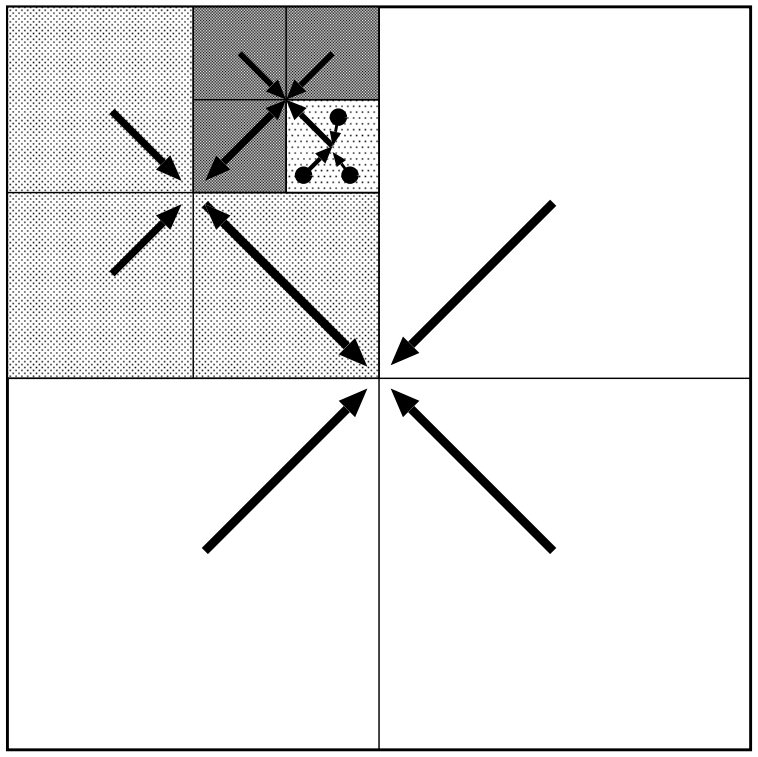
階層的なツリー構造を使う。

- まず、全体が入るセルを作る
- それを再帰的に 8 (2 次元なら 4) 分割する
- 中の粒子がある数以下になったら止める (上の例では 1 個)



多重極展開の構成

まず、ツリーの各セルのなかの粒子がつくるポテンシャルの多重極展開を計算する。

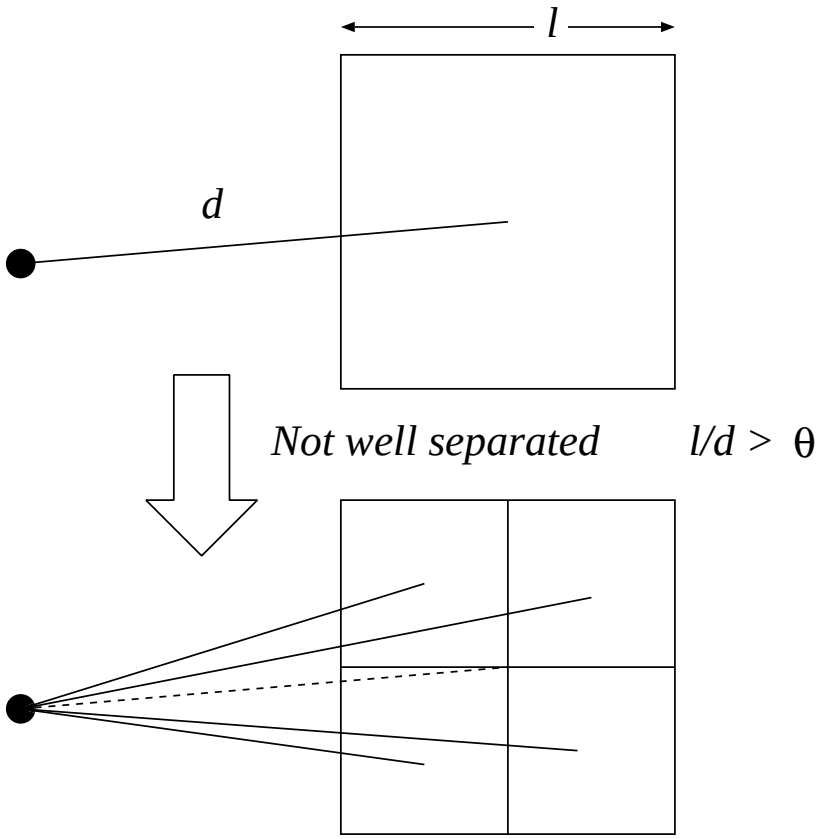


- 最下層のセル:そのなかの粒子が作るポテンシャルを多重極展開
- それ以外 : 子セルの多重極展開の展開中心をシフトして加算

下から順に計算していけばよい。計算量は $O(N)$。展開をシフトする式はかなり複雑。

ツリー法での力の計算

再帰的な形に表現すると格好がいい。



- 十分に離れている:重心(あるいは多重極展開) からの力
- そうでない : 子ノードからの力の合計

系全体からの力 ＝ ルートからの力

速い計算機を買う

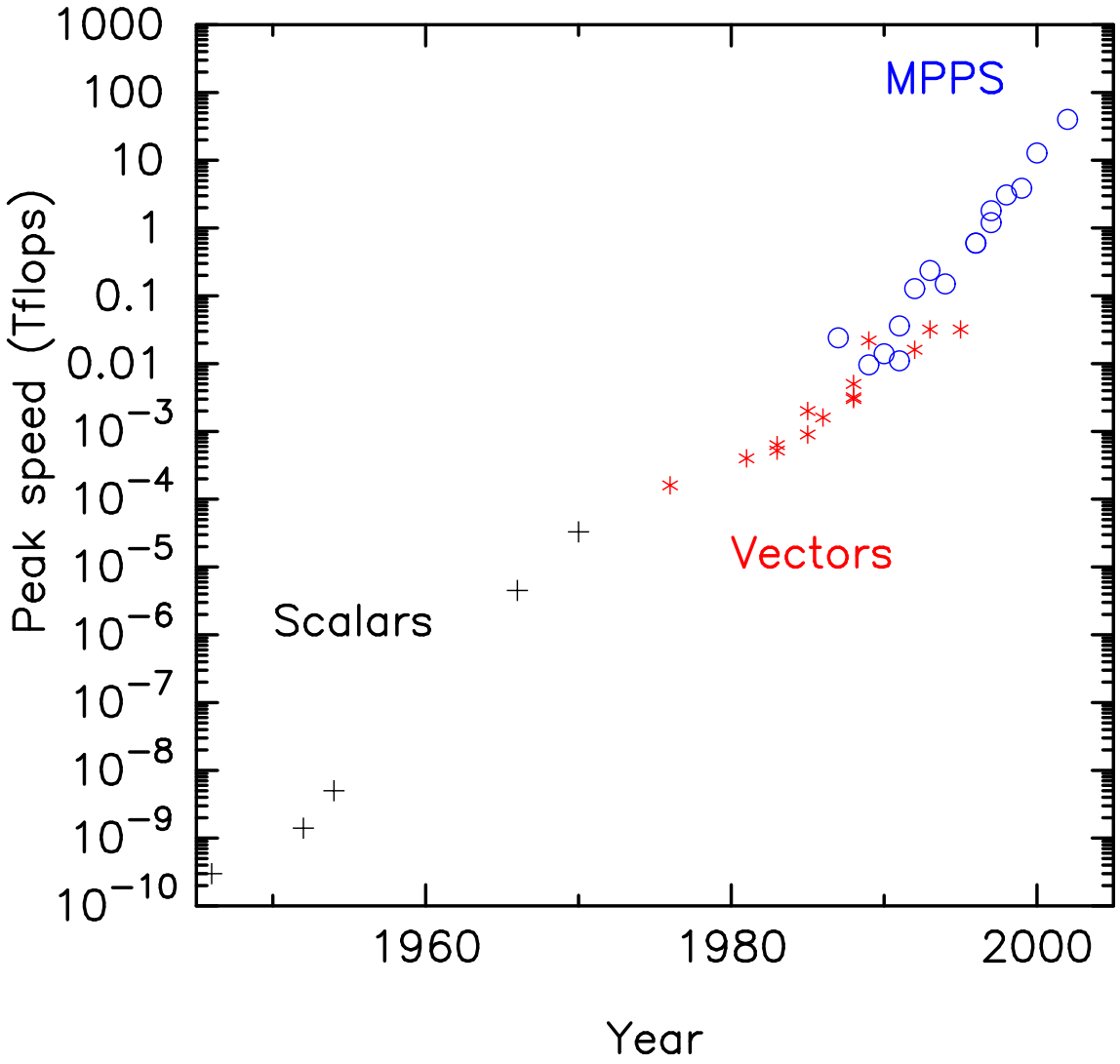
計算法の話はこれくらいにして本題に入る。
速い計算機を使えば速く計算できるというのはなんかトートロジーみたいで当たり前という気もするが、実はそうでもない。

根本的な理由:
最近 30 年間の計算機の「発展」のおかげで、あるアルゴリズムを速い計算機でちゃんと性能がでるように動かすのはどんどん大変になってきた。

計算機の発展

速度の進化: 50 年で 10^{10} 倍

ほぼ時間の指数関数



指数関数的進化を可能にしたもの

1. ムーアの法則:トランジスタの大きさ: 3 年で半分
   - トランジスタの数: 4 倍
   - 速度: 2 倍
2. 計算アーキテクチャの革新
   スカラー計算機 $\rightarrow$ ベクトル $\rightarrow$ 並列

というわけで、現状での課題:(割合ネットワークが遅い並列計算機上での) 並列化

多体シミュレーションの並列化

- 独立時間刻み
- ツリー・FMM

独立時間刻みの並列化

実はそれ以前の問題: 「全粒子から全粒子への力の計算」の並列化

計算量: $O(N^2)$、あるいはタイムステップが増えることも入れると$O(N^{3.5})$ くらい。

$O(N)$ 以上のプロセッサを使いたい。

$O(N)$ でも係数が小さいと使えるプロセッサ数は結構小さいことがある

ツリー法のベクトル化、並列化

ベクトル化、並列化の技法はツリー法も FMM も同じ。

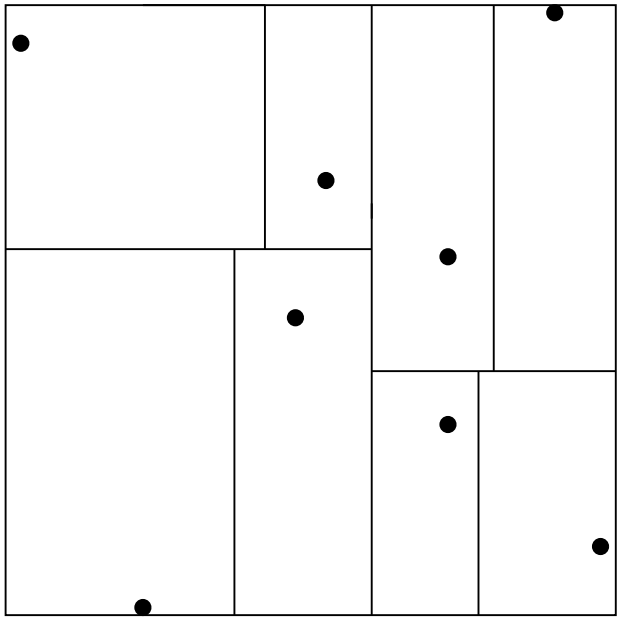
- 共有メモリの機械では「容易」(アルゴリズムの並列度は実際上 N)
  - Splash2 ベンチマークとかで効率がでないのはツリー生成のアルゴリズムが無能なせい
- メッセージパッシングの機械 : 効率の良い空間分割が必要。Salmon & Warren が様々な方法をテストした。プログラムは面倒だが、作ってしまえば効率はよい。

分散メモリでの並列化

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

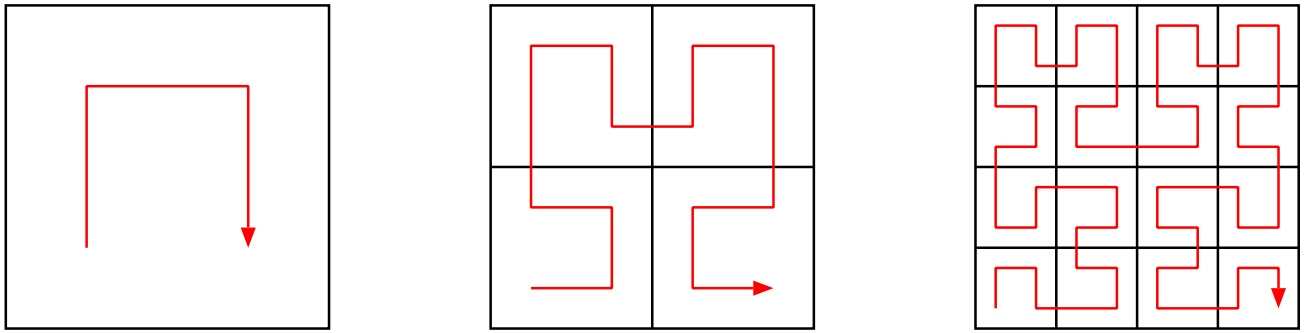


- まず、粒子が半分ずつにわかれるように x 軸に垂直な平面で切る
- 切ったそれぞれについて、また半分になるように y 軸に垂直な平面で切る 3 次元なら次に z、2 次元なら次は x 軸になる

というのを、プロセッサ数になるまで繰り返す

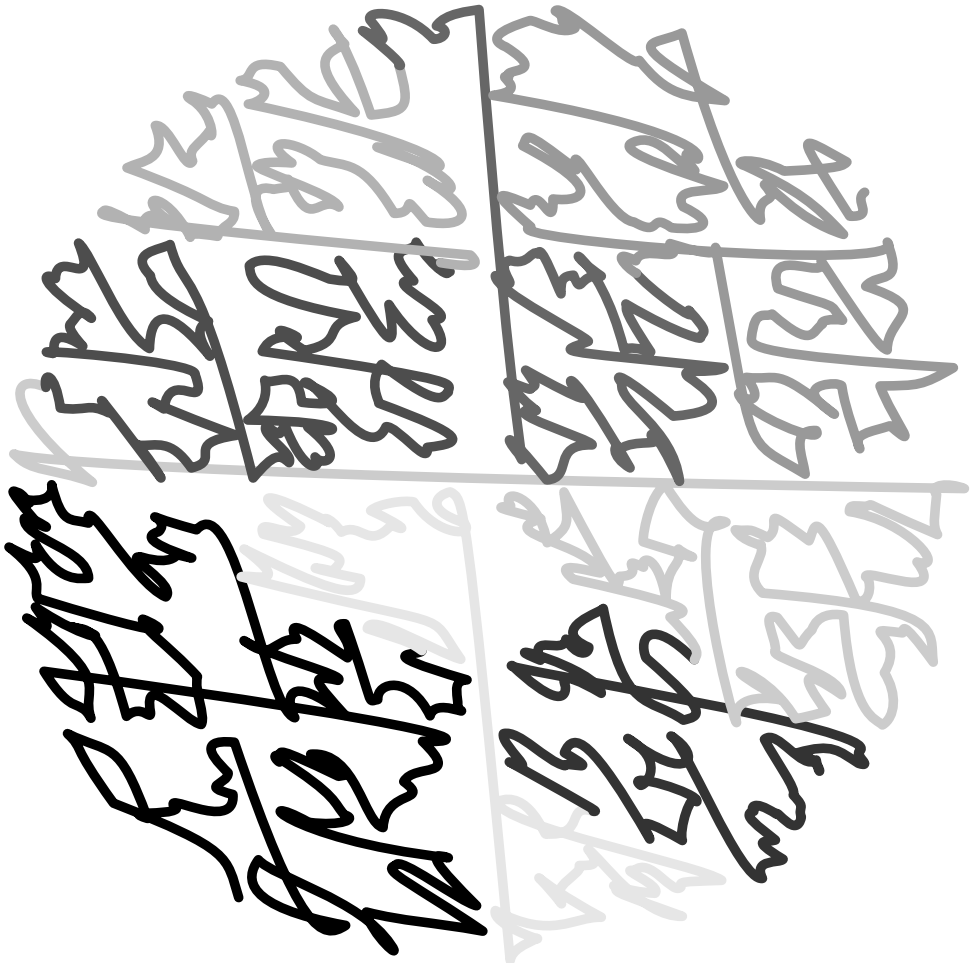
HOT

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

HOT の例



ただしこれはペアノ曲線ではなくて Morton Ordering

高速化の第3の方法 — 計算機を作る

速い計算機を買ってきて動かすというのもなかなか大変。

- 10年もたつと計算機アーキテクチャが変わるので昔頑張って作ったプログラムが水の泡になる
- 最近考えないといけないことが増えた
  - 分散メモリでの並列化
  - キャッシュの有効利用による高速化
  - その他なんだか分からないテクニック

高速化の第3の方法 — 計算機を作る

速い計算機を買ってきて動かすというのもなかなか大変。

- 10年もたつと計算機アーキテクチャが変わるので昔頑張って作ったプログラムが水の泡になる
- 最近考えないといけないことが増えた
  - 分散メモリでの並列化
  - キャッシュの有効利用による高速化
  - その他なんだか分からないテクニック

もうちょっと違う人生はないか?

一つの考え方: 計算機を自分で作る

人が作った計算機を苦労して使おうと思うから面倒。
ハードウェアから好きなように作ればむしろ簡単にならないか?

なぜ専用計算機を考えるのか

もうちょっと真っ当な動機:

基本的には、汎用計算機に比べて、「速く、安く」できる（かもしれない）から。

なぜ「速く、安い」か？

- 問題自体の特性
- 技術的な要因
- 歴史的、経済的な要因 (今日は省略)

問題自体の特性

粒子系シミュレーションの特徴:一つの粒子が多数の粒子と相互作用する

- 計算量が多い (メモリ必要量に比べて)
- 計算が比較的単純な繰り返しである
- 通信パターンが規則的である

(独立時間刻み、ツリー法では細かい考慮が必要にはなる)

対象システムの分類

(川合、星野)

連続系 (流体等): 規則的、近傍通信、計算量小

粒子系：規則的 ($N \times N$ 通信)、計算量大

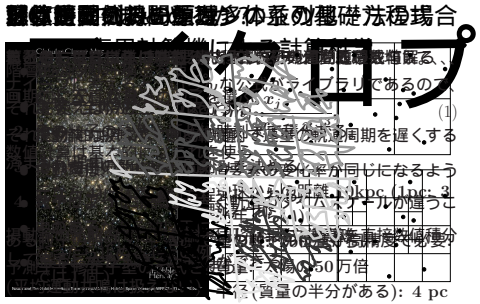
その他 (離散不規則系): 回路シミュレーション等？

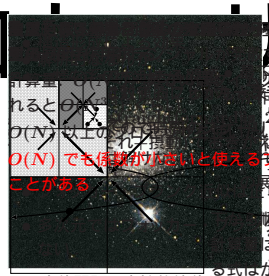
規則的で計算量が大きい = 専用ハードウェアに適している

技術的な要因

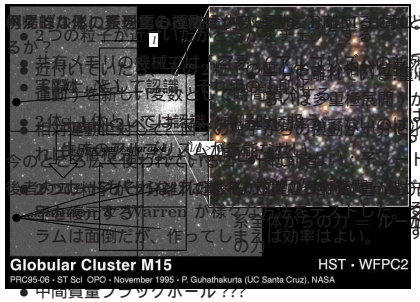
- 半導体製造技術の進歩 ＝ 多数の演算回路を集積する大規模回路が実現可能
- 半導体設計技術の進歩 ＝ 半導体技術の「素人」でも高レベル記述で設計ができる (???)
- 汎用計算機の設計技術の限界 (?) ＝ トランジスタ利用効率の急速な低下

ツイクロプロセッサの「進化」





で、多体問題を直接数値積分すれば、「原理的には」なんでもわかるはず。



ただしこれはペアノ曲線ではなくて Morton Ordering

高速化の第3の方法 — 計算機を作る

速い計算機を買ってきて動かすというのもなかなか大変。

- 10年もたつと計算機アーキテクチャが変わるので昔頑張って作ったプログラムが水の泡になる
- 最近考えないといけないことが増えた
  - 分散メモリでの並列化
  - キャッシュの有効利用による高速化
  - その他なんだか分からないテクニック

高速化の第3の方法 — 計算機を作る

速い計算機を買ってきて動かすというのもなかなか大変。

- 10年もたつと計算機アーキテクチャが変わるので昔頑張って作ったプログラムが水の泡になる
- 最近考えないといけないことが増えた
  - 分散メモリでの並列化
  - キャッシュの有効利用による高速化
  - その他なんだか分からないテクニック

もうちょっと違う人生はないか?

一つの考え方: 計算機を自分で作る

人が作った計算機を苦労して使おうと思うから面倒。
ハードウェアから好きなように作ればむしろ簡単にならないか?

なぜ専用計算機を考えるのか

もうちょっと真っ当な動機:

基本的には、汎用計算機に比べて、「速く、安く」できる（かもしれない）から。

なぜ「速く、安い」か？

- 問題自体の特性
- 技術的な要因
- 歴史的、経済的な要因 (今日は省略)

問題自体の特性

粒子系シミュレーションの特徴:一つの粒子が多数の粒子と相互作用する

- 計算量が多い (メモリ必要量に比べて)
- 計算が比較的単純な繰り返しである
- 通信パターンが規則的である

(独立時間刻み、ツリー法では細かい考慮が必要にはなる)

対象システムの分類

(川合、星野)

連続系 (流体等): 規則的、近傍通信、計算量小

粒子系：規則的 ($N \times N$ 通信)、計算量大

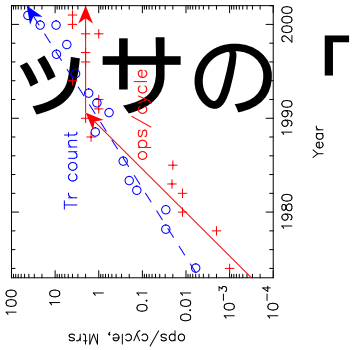
その他 (離散不規則系): 回路シミュレーション等？

規則的で計算量が大きい = 専用ハードウェアに適している

技術的な要因

- 半導体製造技術の進歩 ＝ 多数の演算回路を集積する大規模回路が実現可能
- 半導体設計技術の進歩 ＝ 半導体技術の「素人」でも高レベル記述で設計ができる (???)
- 汎用計算機の設計技術の限界 (?) ＝ トランジスタ利用効率の急速な低下

マイクロプロセッサの「進化」

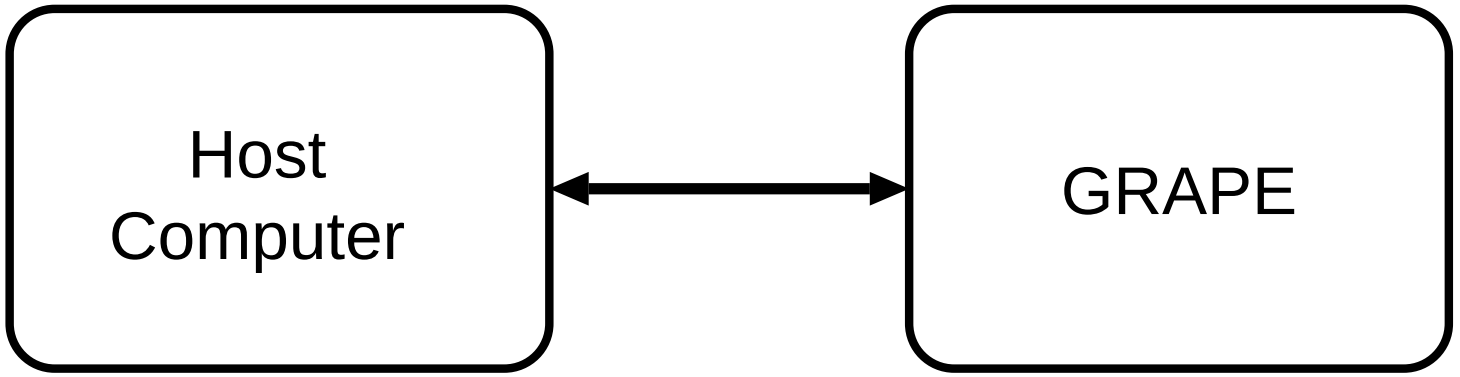


代表的なマイクロプロセッサの、サイクル当たりの浮動小数点演算数

1 を超えてから、ほとんど止まっている ＝ 汎用の並列処理は難しい

ここがつけめ。

GRAPE の基本的考え



Time integration etc.　　Interaction calculation

専用ハード: 相互作用の計算

汎用ホスト: 他のすべての計算

専用ハードウェア

- 相互作用計算のための専用パイプラインプロセッサ
  - 多数の演算器を集積可能
  - すべての演算器が常時並列動作

  $\longrightarrow$ 非常に高い性能

重要な条件: **Memory wall** の回避

汎用ホスト計算機

- 高レベル言語 **(Fortran, C, C++...)**
- すでにあるプログラムが「少しの」変更で使える。
- 独立時間刻み、ツリー法等も使える

Memory Wall

元々は M. Wilkes が使った言葉 (EDSAC の設計者)。
CPU の処理能力に、主記憶とのデータ転送能力が追いつかなくなる。
3 つの要因

- プリント基板は LSI のようには微細化しない。
- チップ間接続の数は微細化に比例。ゲート数は 2 乗に比例。
- 長い配線では高い周波数の転送は大変。

例: Intel Pentium 4 3GHz、 クロック 1 演算としても

- 72GB/s の転送速度が欲しい。
- メモリは理論ピークで 6.4GB/s しかない

ベクトル計算機の対応

ベクトル計算機における **Memory Wall** への対応:
物理的にメモリを沢山準備してバンド幅を増やす。

ベクトル計算機の対応

ベクトル計算機における **Memory Wall** への対応:

物理的にメモリを沢山準備してバンド幅を増やす。

数字を比べてみると、、、

計算機	ピーク演算性能	メモリバンド幅	比
Intel P4 (3.8GHz)	**7.6Gflops**	**0.8GW/s**	**0.1**
SX-8	**16Gflops**	**8GW/s**	**0.5**

ベクトル計算機の対応

ベクトル計算機における **Memory Wall** への対応:

物理的にメモリを沢山準備してバンド幅を増やす。

数字を比べてみると、、、

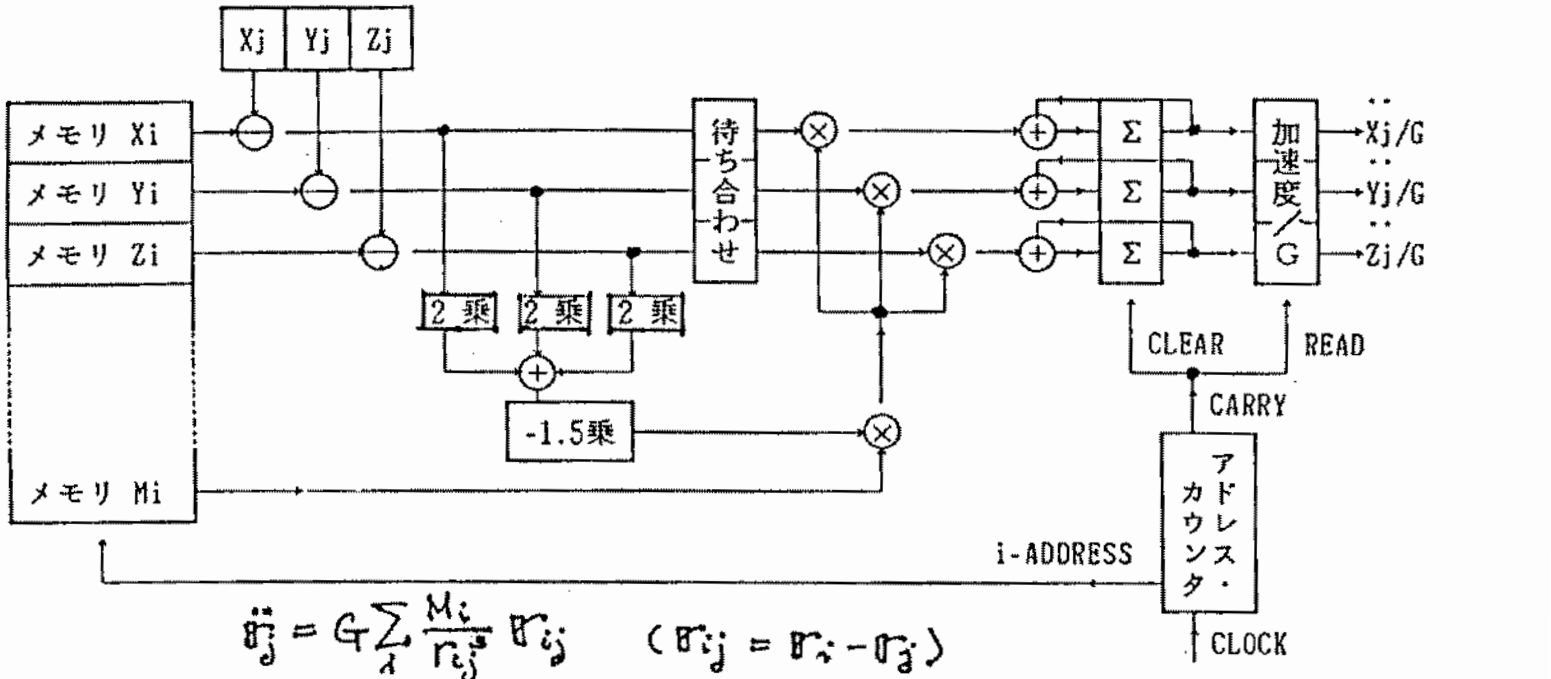
計算機	ピーク演算性能	メモリバンド幅	比
Intel P4 (3.8GHz)	**7.6Gflops**	**0.8GW/s**	**0.1**
SX-8	**16Gflops**	**8GW/s**	**0.5**

実は桁では変わらない。

性能差の例: **VPP-5000** に最適化された陽解法流体コードで**10**倍とか

(VPP5000, Opteron 1.6GHz 1CPU)

GRAPE パイプライン



$$\ddot{\vec{r}}_j = G\sum_i \frac{M_i}{r_{ij}^3}\vec{r}_{ij} \qquad (\vec{r}_{ij} = \vec{r}_i - \vec{r}_j)$$

+, −, ×, 2乗は1 operation, -1.5乗は多項式近似でやるとして10operation 位に相当する.
総計24operation.
各operation の後にはレジスタがあって, 全体がpipelineになっているものとする.
「待ち合わせ」は2乗してMと掛け算する間の時間ズレを補正するためのFIFO(First-In First-Out memory).
「Σ」は足し込み用のレジスタ. N回足した後結果を右のレジスタに転送する.

図2. N体問題のj-体に働く重力加速度を計算する回路の概念図.

(近田 1988)

GRAPE における Memory wall の回避

あるいは通信量の削減

ホスト — GRAPE 間: N 個の粒子、 N^2 の計算

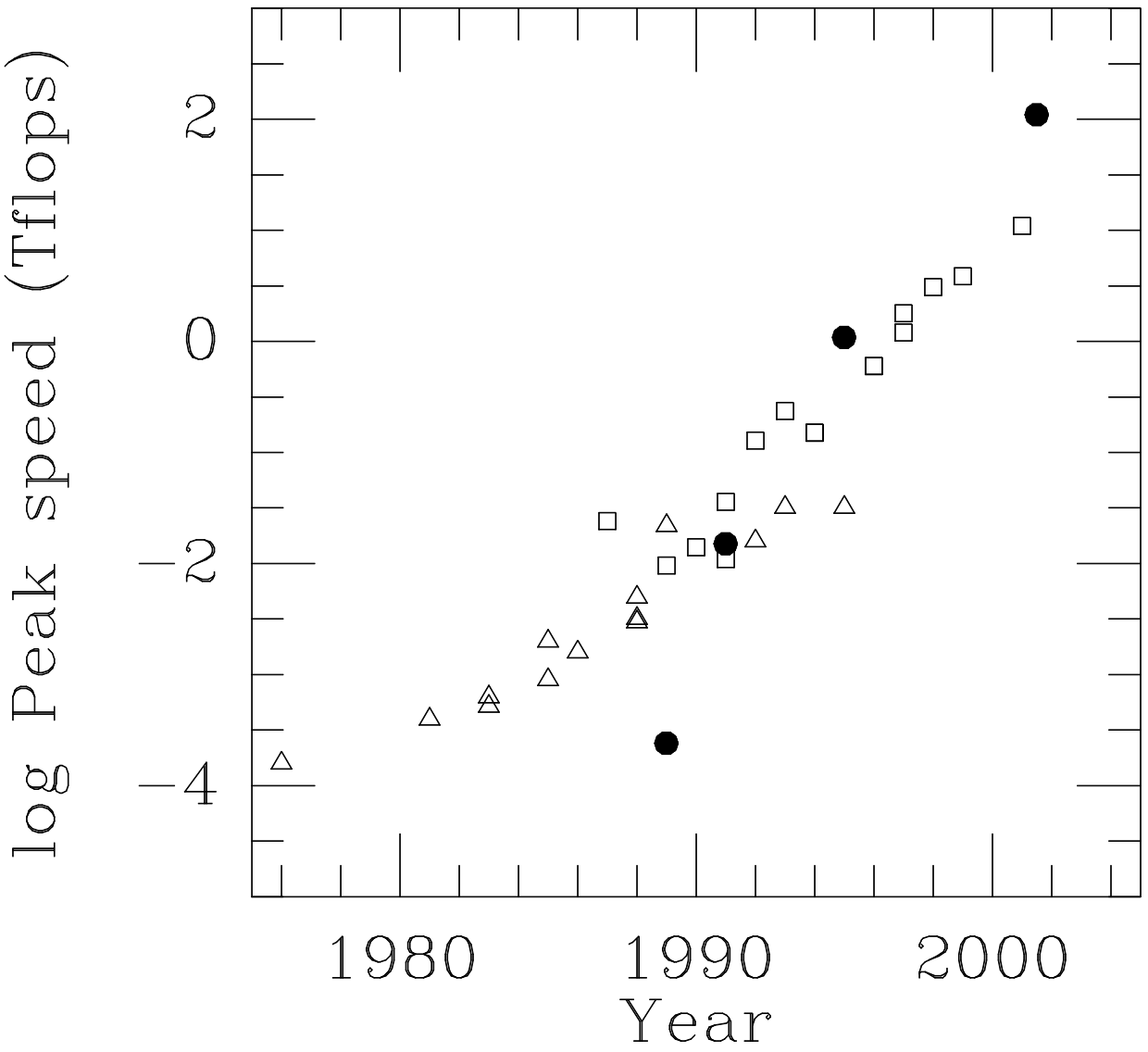
ボード/チップレベル:

- i 並列 (複数のパイプラインがメモリ 1 つから同じデータを受け取って、別の粒子への力を計算)
- 仮想マルチパイプライン

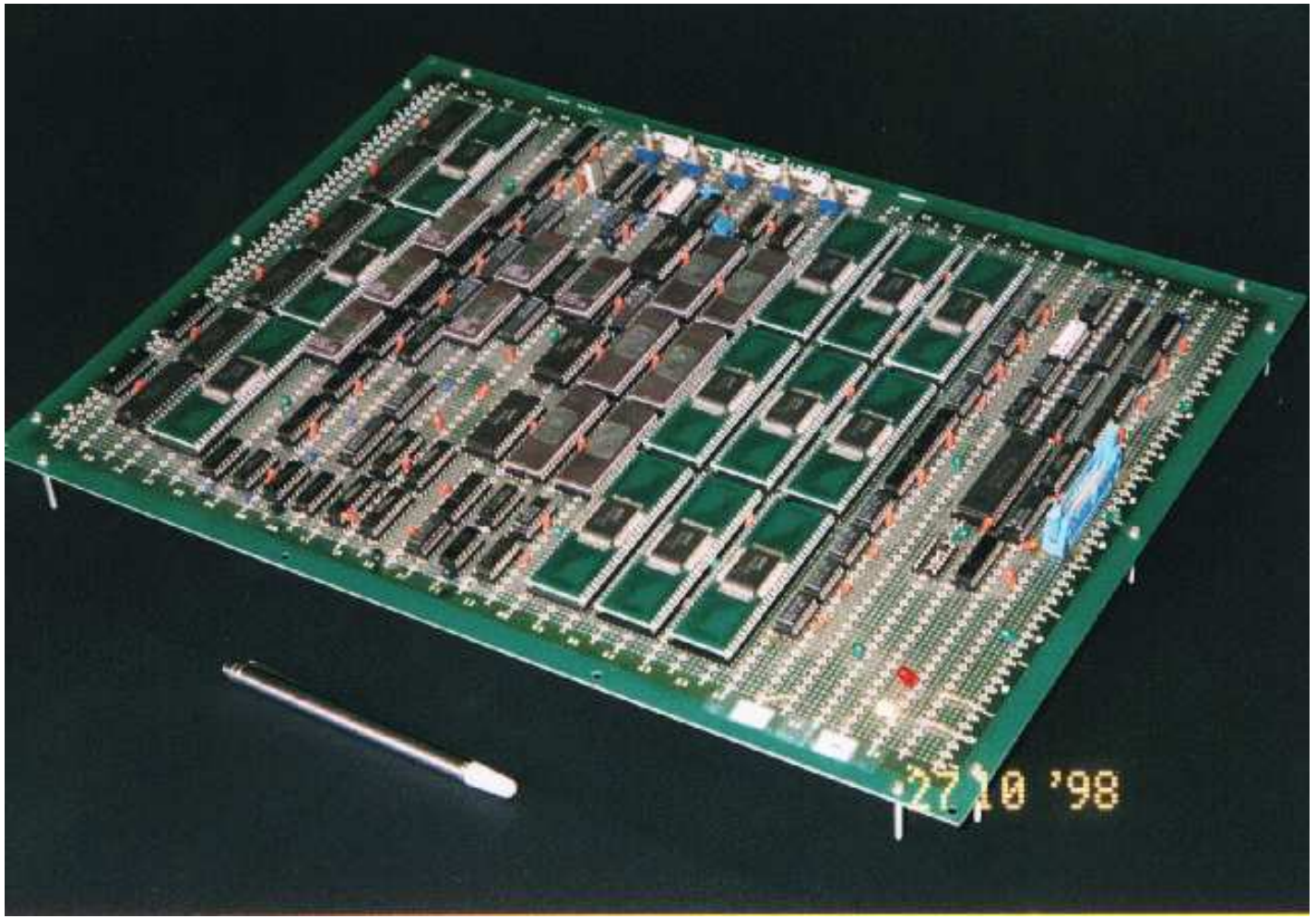
GRAPE アーキテクチャの発展

1989 GRAPE-1 低精度、EPROM で演算
1990 GRAPE-2 高精度、浮動小数点演算 LSI
1991 GRAPE-3 低精度、カスタム LSI
1995 GRAPE-4 高精度、カスタム LSI 、超並列
1998 GRAPE-5 低精度、複数パイプラインを集積
2001 GRAPE-6 高精度、複数パイプラインを集積

計算速度の発展



GRAPE-1



GRAPE-1 の中身

「初めてのデジタル回路」

当初の目標

- とりあえず演算パイプラインのようなものを作る
- ホスト計算機につないで動かす
- 意味がある計算ができるかどうかはあまり気にしない

最初の構成

- 演算は ROM テーブルルックアップ。データは 8 ビット
- 通信は GPIB

計算/通信比: 粒子数1万くらいで性能がでればいいことにすると

動作クロック 5 MHz くらい = 1粒子のデータのやりとりに 1ms くらいは使っていい

データ量が数十バイトくらい → データ転送速度は100KB/s くらい欲しい

RS-232C では不足

SCSI は難しくて良く分からなかった

実用になるものへ

途中の計算 (2 粒子間の力の相対精度) は低くても、

- 最初の座標の引き算
- 最後の足し上げ

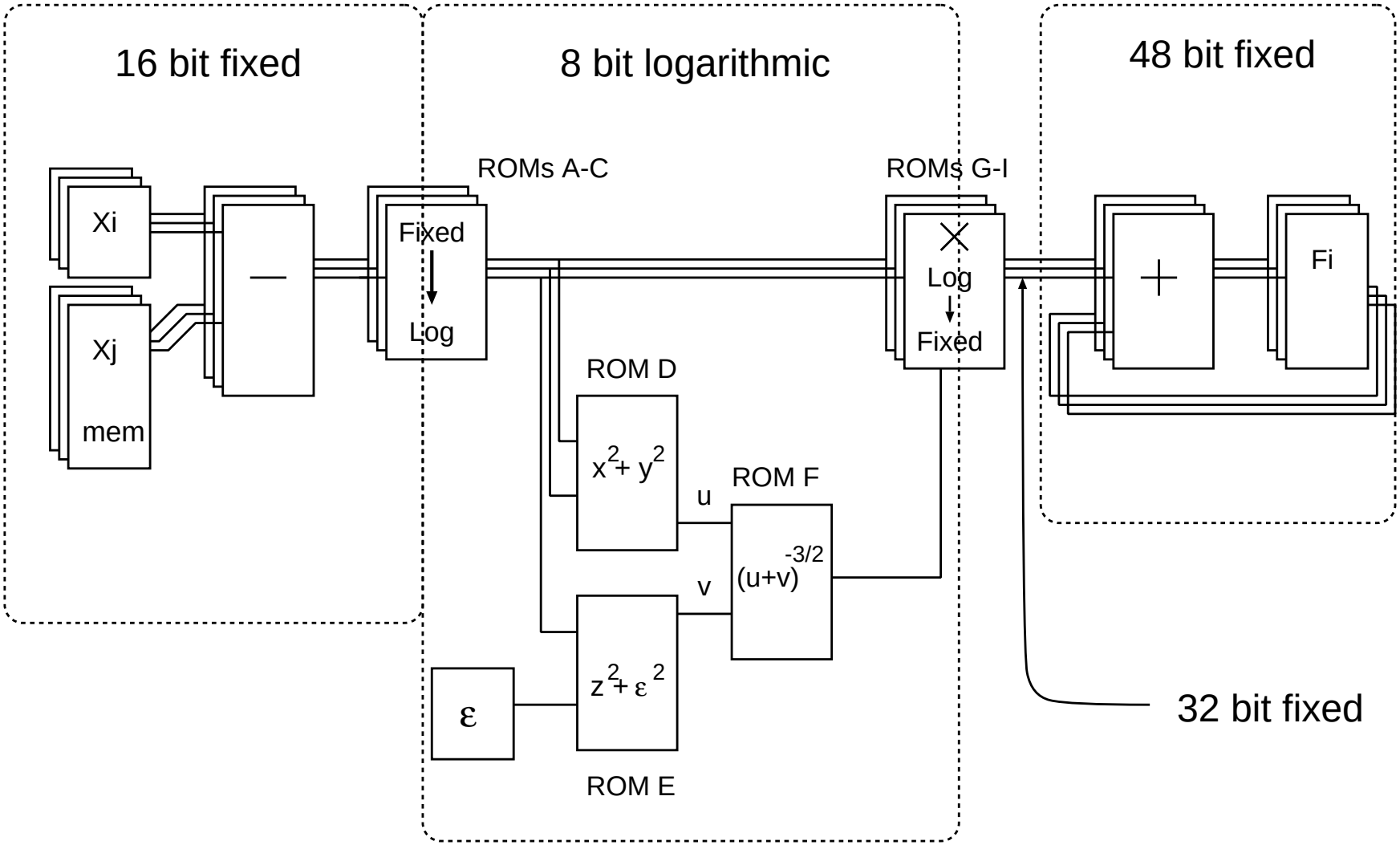
だけを (そこそこ) 高い精度でやれば、問題によっては十分使える (ツリーコードに比べれば高い精度になる)

ということに気が付いたので、そっちに変更

最初の座標の引き算: 16bit 固定小数点、最後の足し上げ:48bit 固定小数点

とする。

GRAPE-1 パイプライン



開発中のトラブル

ハードは割合順調に出来たらしい (実は伊藤君が担当だったので良く知らない)

できてから、性能がでない、、、

当初: PC-9801 (i286) ホスト、どこかの GP-IB ボード (TI の GP-IB インターフェースチップがのってるだけのもの)。
通信速度は問題なし

ホストが遅い、メモリものらない、、、、というので何故か GP-IB インターフェースカードがあった Sony NEWS-830 につなぐと、、、、

通信オーバーヘッド

NEWS につなぐと全くまともな性能が出ない。

原因: GPIB を使った通信は read/write システムコールを使う。で、これが 1 回呼ぶと 1ms くらいかかる。

GRAPE-1: 1 粒子もらって、それへの力を計算、結果返す。

NEWS 830 は I/O プロセッサ付き: オーバーヘッドは一層大きい

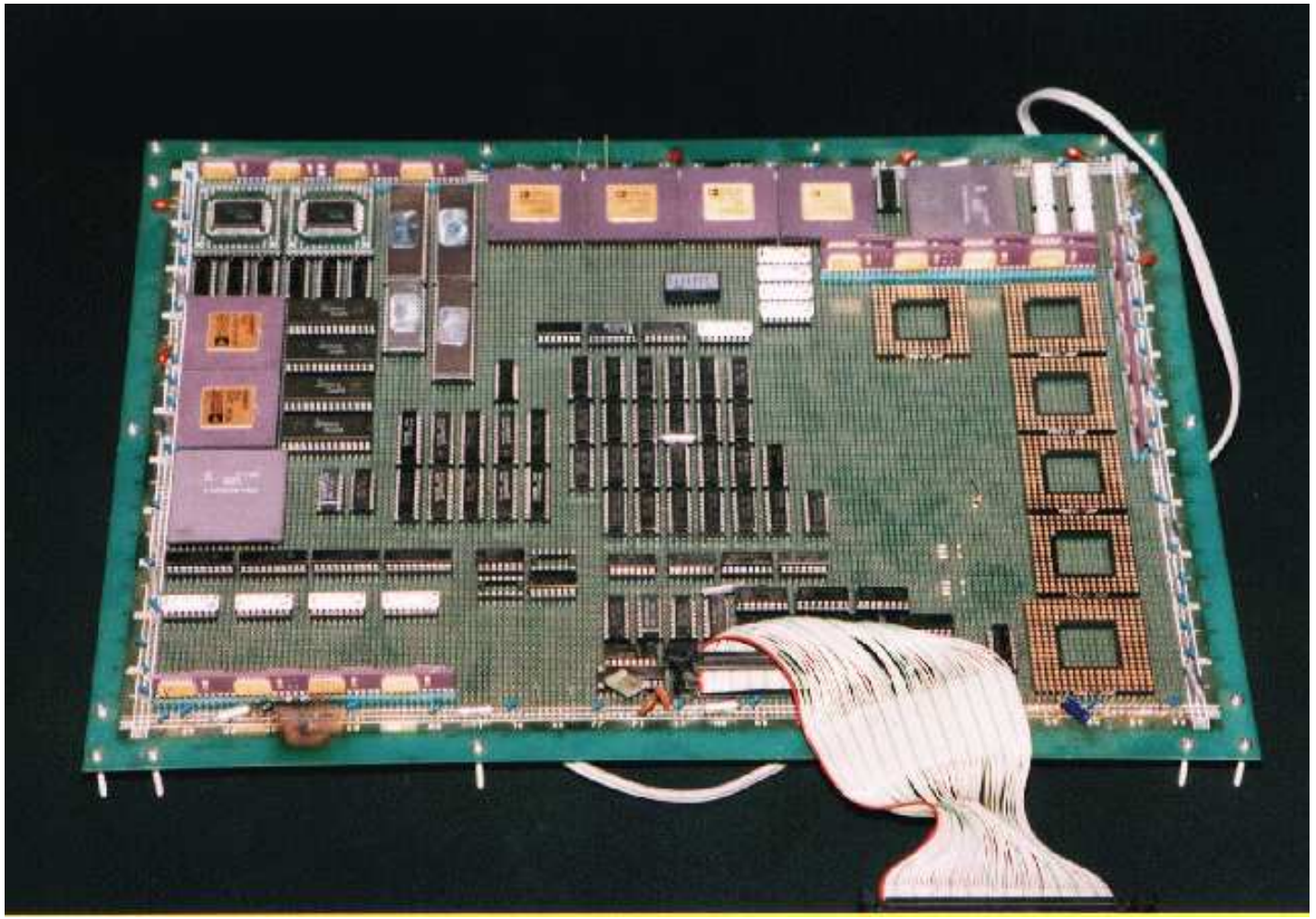
最初の対応: PC-9801 をバッファリングに使う。

次の対応: I/O プロセッサのない安い NEWS にして、 GP-IB コントローラのレジスタを OS を通さないでユーザープロセスが直接アクセスする。これで 100KB/s くらいでるようになった。

教訓

- 通信ソフトウェアは厄介である
- 教科書的に「良い」方法が良い結果になるとは限らない
- 結果が良ければ「邪道な」ことでもなんでもかまわない

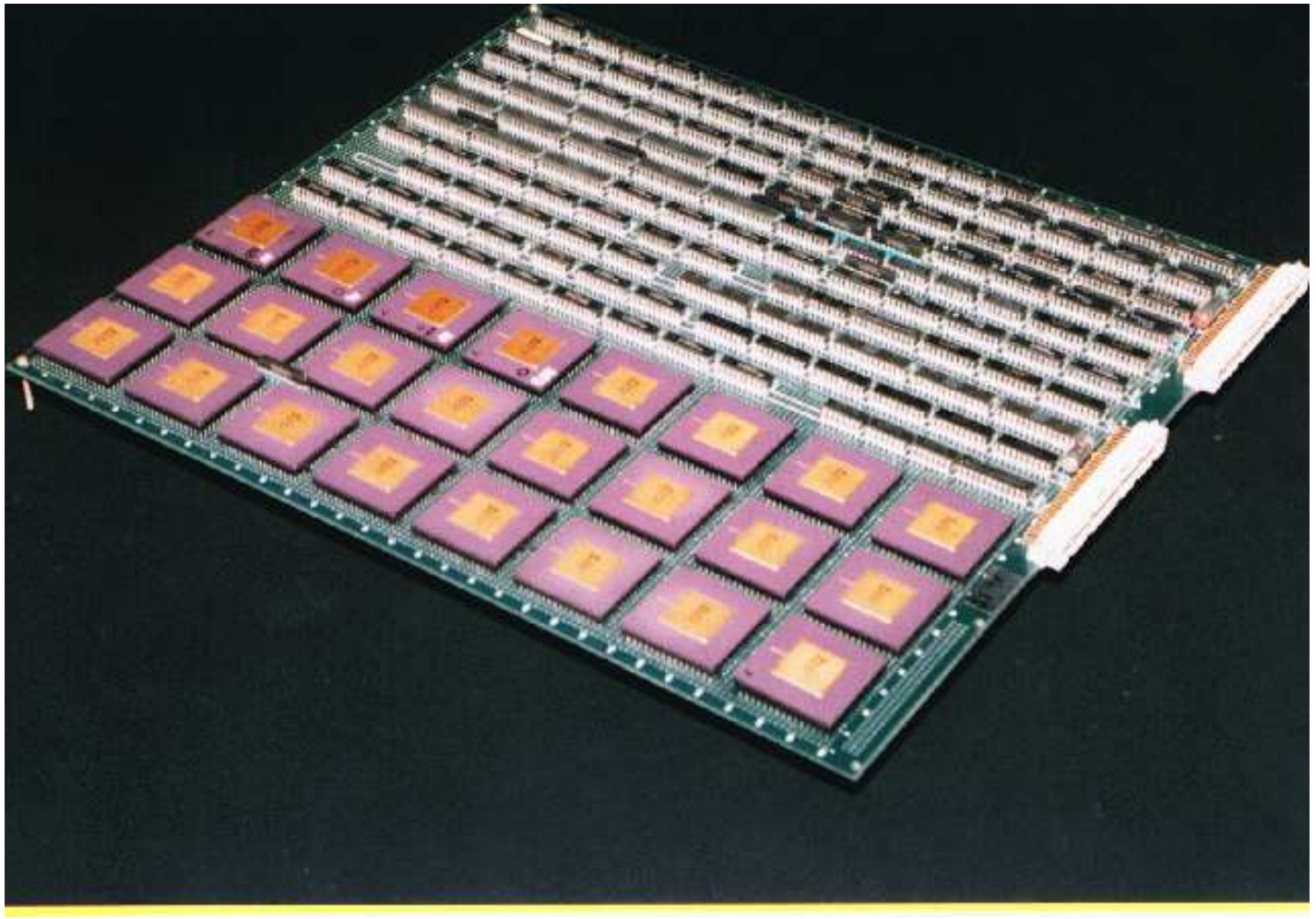
GRAPE-2



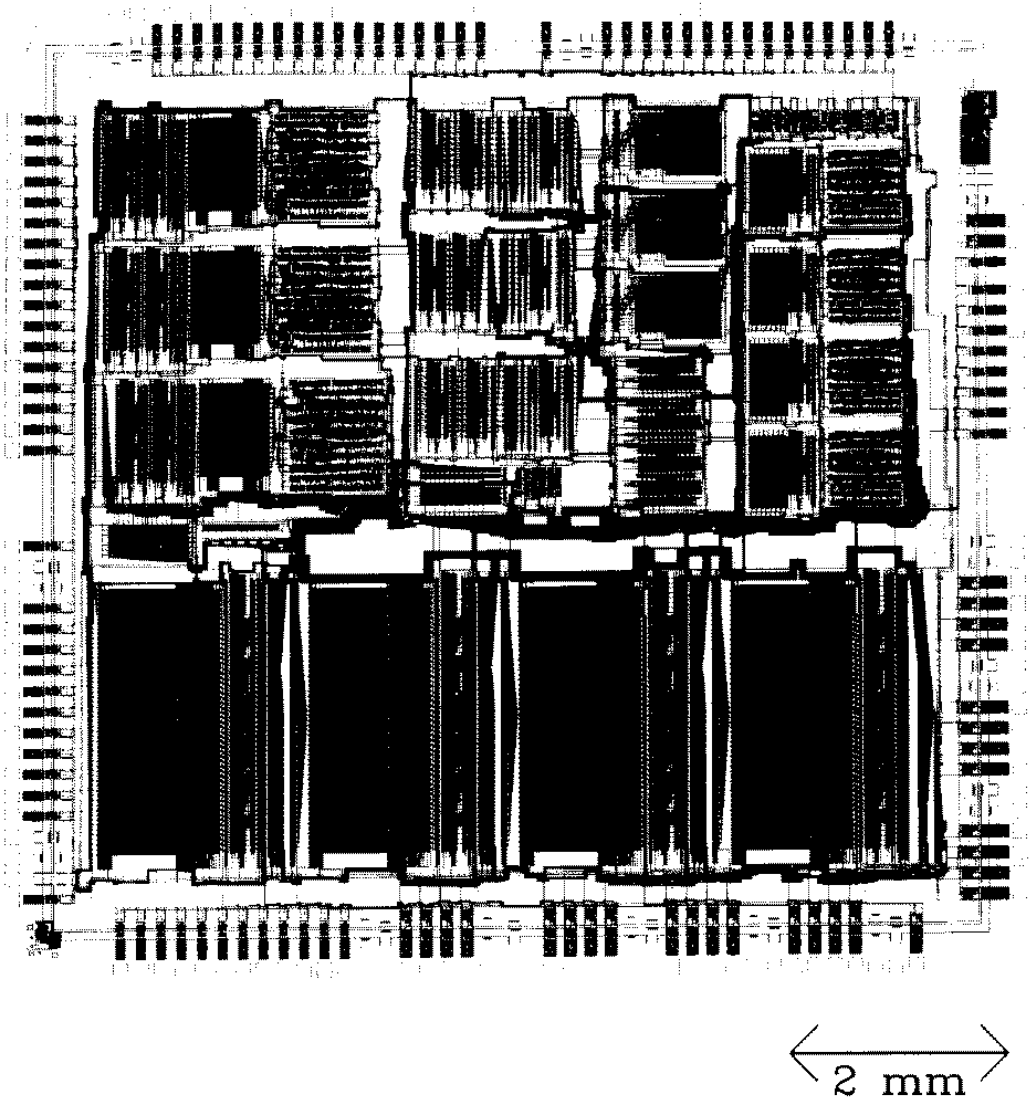
GRAPE-2 のまとめ

- 8 ビット演算とかは止めて普通に浮動小数点演算 (倍精度は最初と最後だけ)
- 通信は VME バス
- ホストは VME バスに PIO アクセス。 DMA は使わない (GRAPE はマスターにならない)

GRAPE-3



GRAPE-3 チップ



GRAPE-3 チップ

- 仕様決定、シミュレータ (C で記述) は牧野がやった
- 論理設計以降は富士ゼロックス (橋本、富田)
- SCS Genesil で設計
- ファブは NS. 1μm

チップを作るのはどれくらい心臓に悪いか?ということ

これまで、2回チップを作る予算があったことはない=一発勝負

普通の責任分担:

- テストベクタが通らなければ向こうのせい
- テストベクタが通ってまだ問題があればこっちのせい

「理論上は」「完璧な」テストベクタを作ればこちらのせいである問題は起こりえない。

実際はそうはいかないけど、、、

普通のチップとの違い

マイクロプロセッサとかに比べるとはるかに楽

「内部状態」というものがほとんどない。計算中/停止の違いくらい。これはアキュムレータの制御だけ。

演算回路の仕様自体が問題。小規模なシミュレーションでは動いても、実際にハードウェアができて初めて可能になる大規模な計算で問題なく動くかどうかはわからない。

その他、予想もしない問題は起きる。

GRAPE のチップはどんなふうに作るか

1. まずパイプラインの仕様を決める。
   - 何を計算するか
   - 各演算器の演算精度
   - 目標性能
   - メモリインターフェース、ホストインターフェースの物理仕様 (ビット幅、速度)
2. パイプラインの機能検証: 動作 (計算結果) をソフトウェアでシミュレーションするプログラムを書く。その動作が理屈に合っていて、実際の多体計算プログラムにいれても動くことを確認する。
3. あとは普通にハードウェア記述言語で設計

CPU の設計との違い: 物理設計の検証は動作記述と合わせる。サイクルレベルの設計はハードウェア記述言語のものだけ。

GRAPE-4

GRAPE-4 における通信

演算性能:GRAPE-3 の 100 倍

通信:単一ホストでできるベストに近いところで済ませられればそうしたい。

とすると 100 MB/s くらい。

1 粒子データ 200 バイトくらい

10^5 粒子くらいでまあまあの性能がでる計算になる。

まあ、ゴードンベル賞でも取ろうという時以外は機械を分けて使うので、これくらいの見積りで十分。

100MB/s をどうやって出すか? (1993年に)

- DMA は必須
- ホストは、、、、 DEC Alpha、バスは TurboChannel

オーバーヘッドを避けるためにはDMA のたびにカーネルにいったりしては困る。
というわけで
ユーザープロセスのバッファに使うページの物理アドレスをカーネルの内部関数をなんかして調べて、それをユーザープロセスがインターフェースカードの DMA アドレスレジスタに直接書く。

最近なら？

最近のまともな OS では、カーネルが確保したバッファをユーザー空間にマップしてくれるみたいなので同様のことがもうちょっと安全に実現できる。

GRAPE-6 について

- 設計思想
- プロセッサチップ
- プロセッサボード
- ネットワークボード
- 全体

設計思想

プロジェクト目標 (お金をもらう時の公約)

粒子系専用でいいけど世界一の速度を達成する

「当然」の目標:

サイエンスとして、研究にちゃんと使える機械にする。おもちゃとは違う。

境界条件

- 総予算: 5 億 (参考:地球シミュレータ 400 億、ASCI Q 250 億？)

で、初期目標: 200Tflops (地球シミュレータ 40T)

境界条件

- 総予算: 5 億 (参考:地球シミュレータ 400 億、ASCI Q 250 億？)

で、初期目標: 200Tflops (地球シミュレータ 40T)

「常識」では考えられない

予算獲得時には結構問題

境界条件

- 総予算: 5 億 (参考:地球シミュレータ 400 億、ASCI Q 250 億？)

で、初期目標: 200Tflops (地球シミュレータ 40T)

「常識」では考えられない

実際のところできる話かどうか？

だいたいできたから結果論だけど

GRAPE-6 の性能予測

演算性能の予測: 基本的は GRAPE-4 から外挿しただけ

	G4	G6 (予測)	G6 (実際)
設計ルール	1μm	0.25μm	0.25μm
動作クロック	32 MHz	125 MHz	90MHz
パイプライン本数	1/3	5-10	6
演算性能	600Mflops	36-72 Gflops	31 Gflops
開発費	2500万	7000万くらい	1億以上
チップ単価	8000円	1-2万	3万

楽観的には 3000チップで 200 Tflops。チップ以外はボード当り 100万くらい(これは大体合ってた)

チップについては結構予測が甘かった、、、 200Tflops が 64 Tflops に落ちた理由

全体アーキテクチャの制限

もっとも厄介な制約: ホスト計算機との通信速度

GRAPE-4の実効速度: 50 MB/s 程度

理想的には 100倍、でなくても 20 倍くらいは欲しい。

- 並列ホストは必須
- 単純に並列にホストとGRAPEをつなぐのでは駄目

チップアーキテクチャの制限

GRAPE-4 ボード１枚分くらいがチップに入る。

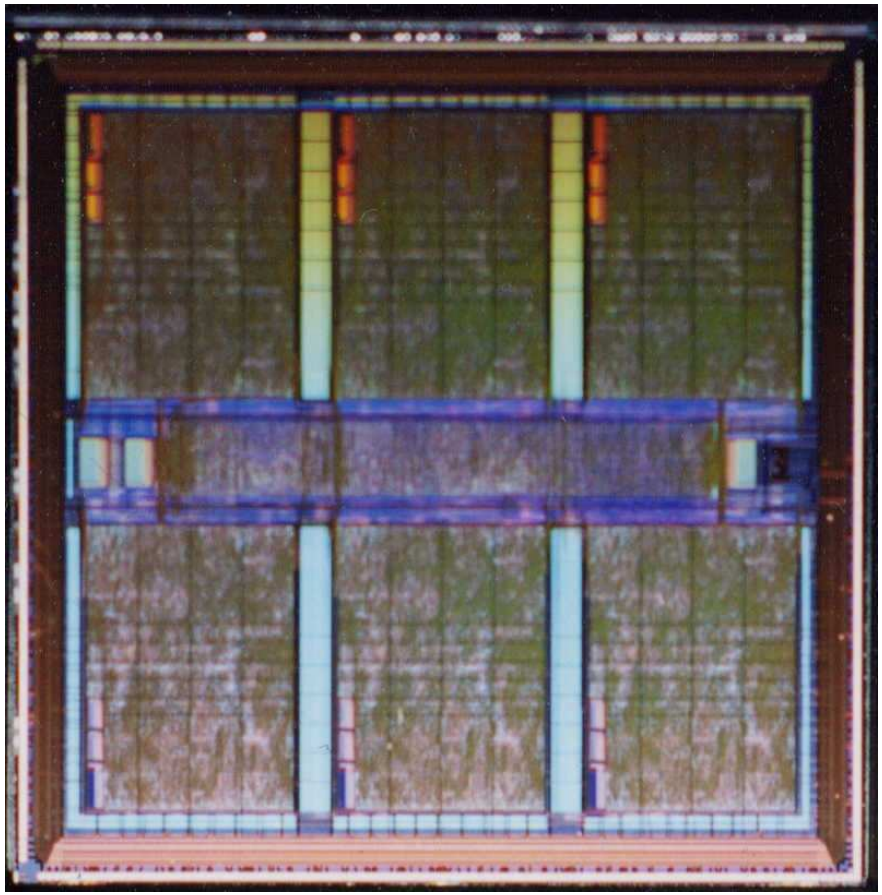
独立時間刻みを使うことを考えると、メモリを共有できるチップ数に制限がつく。

チップ毎にメモリを持つことで、この問題を回避。

複数のチップが同じ粒子への、別の粒子からの力を計算、ボード上で合計してホストに戻す。

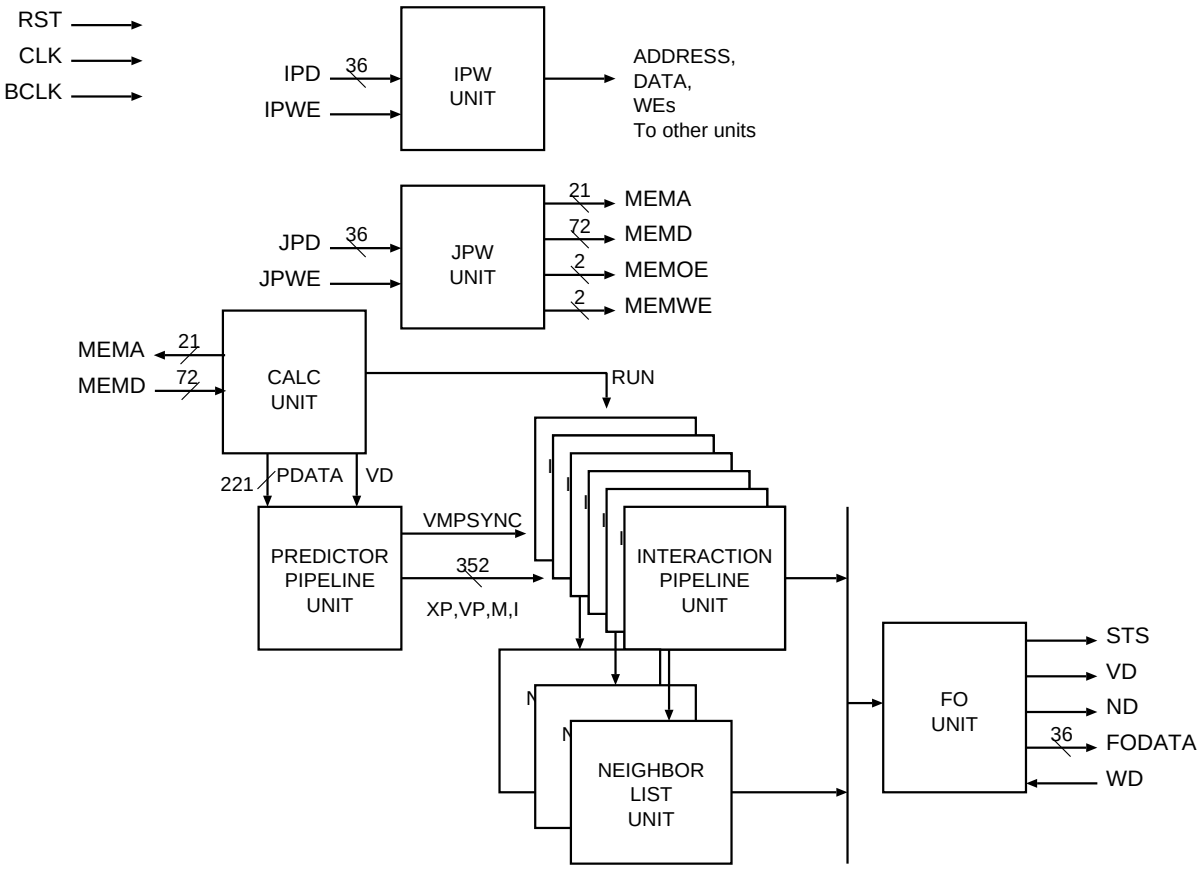
指数を固定して加算することで、結果が加算順序に依存しないようにする。(細かいことだけど割合大事)

パイプライン チップ



- 0.25 μm ルール
  (東芝　TC-240,　1.8M ゲート)
- 90 MHz 動作
- 6 パイプラインを集積
- チップあたり 31 Gflops

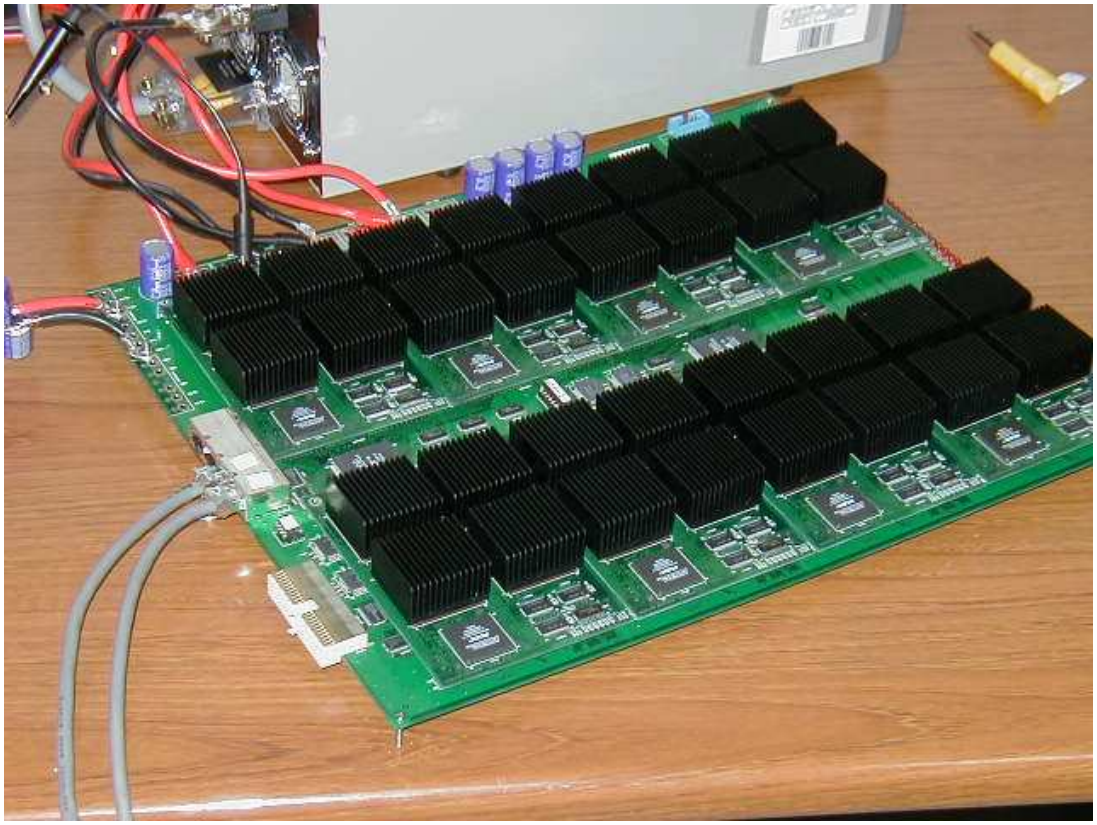
パイプライン LSI詳細



GRAPE-4 プロセッサボードの全機能を集積

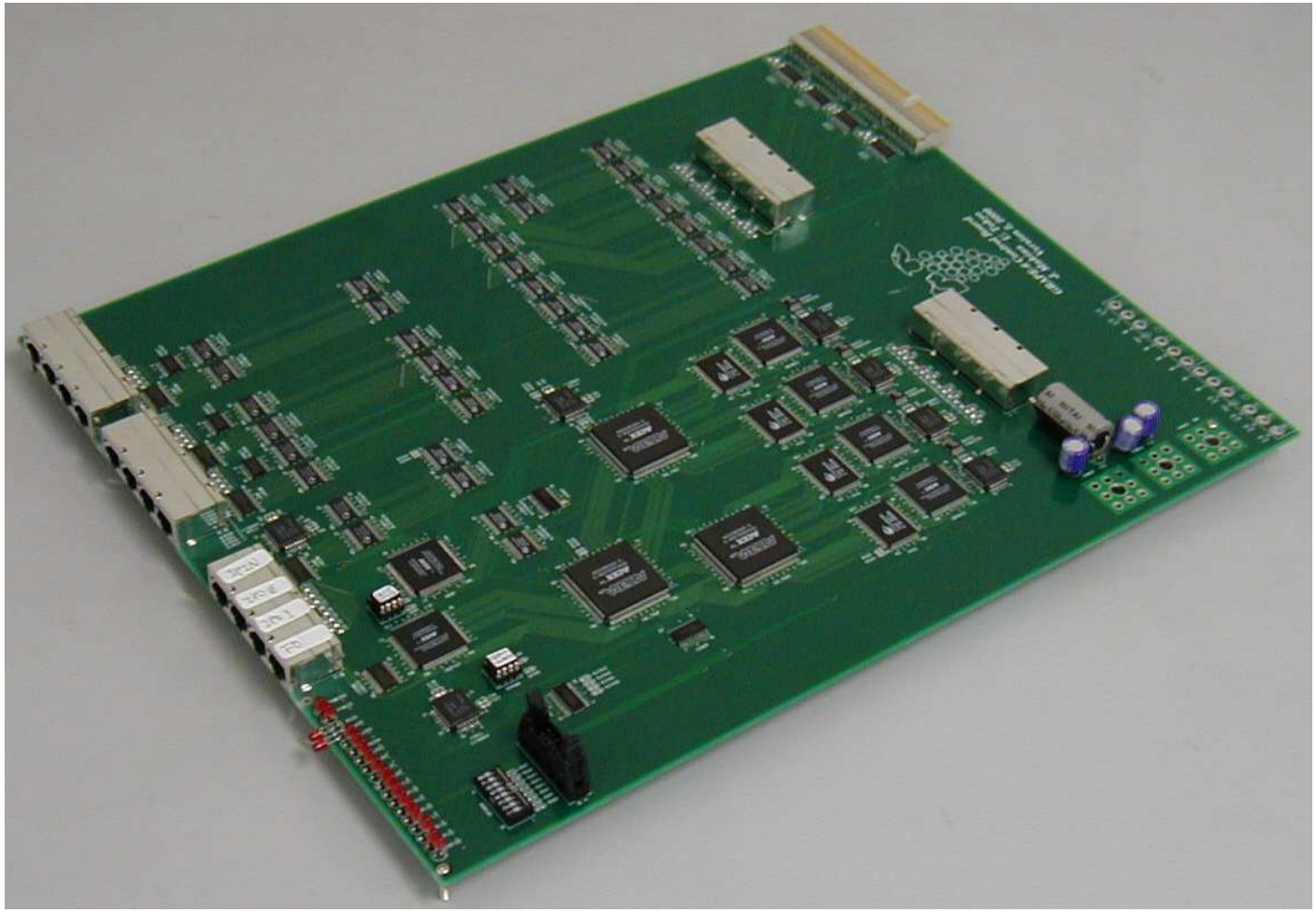
- ホストIF
- メモリIF
- パイプライン本体
- 制御回路

GRAPE-6 processor board



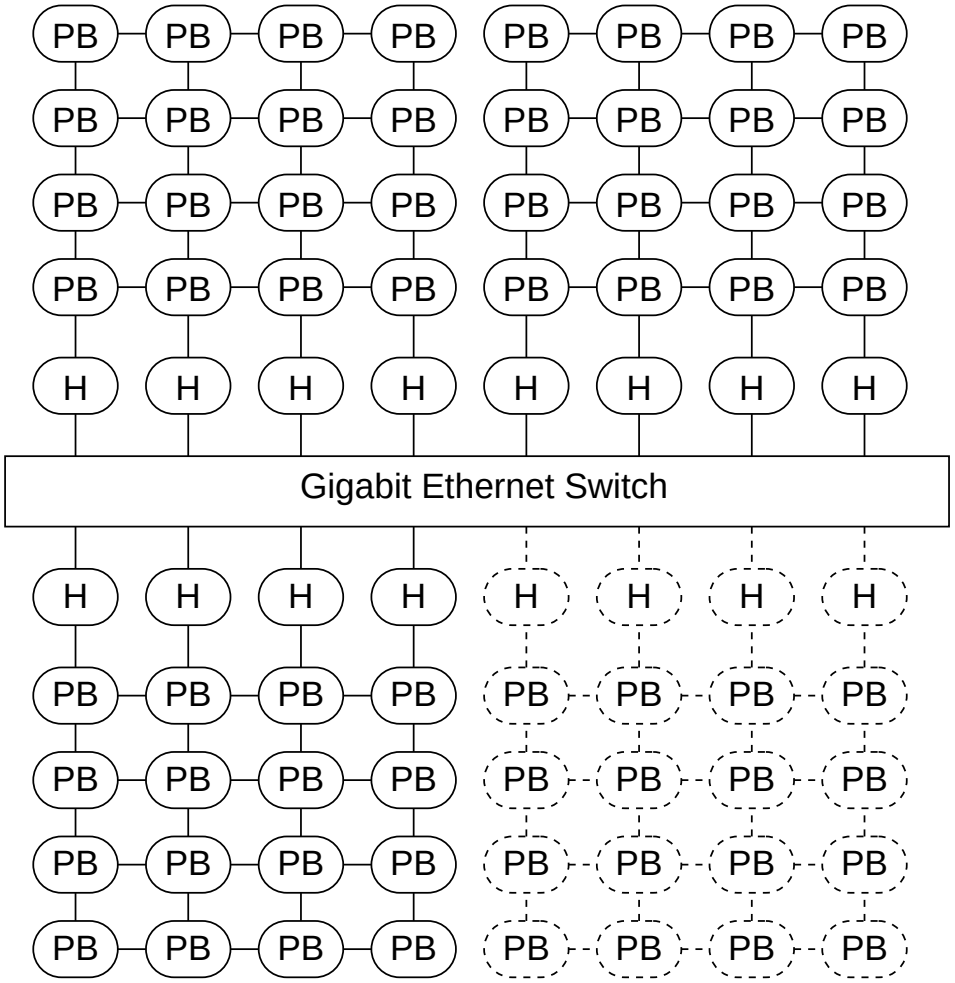
- ボード 1 枚に 32 チップ
- セミシリアル (LVDS) インターフェース (350MHz clock, 4 wires)

GRAPE-6 network board



GRAPE-6 host interface board

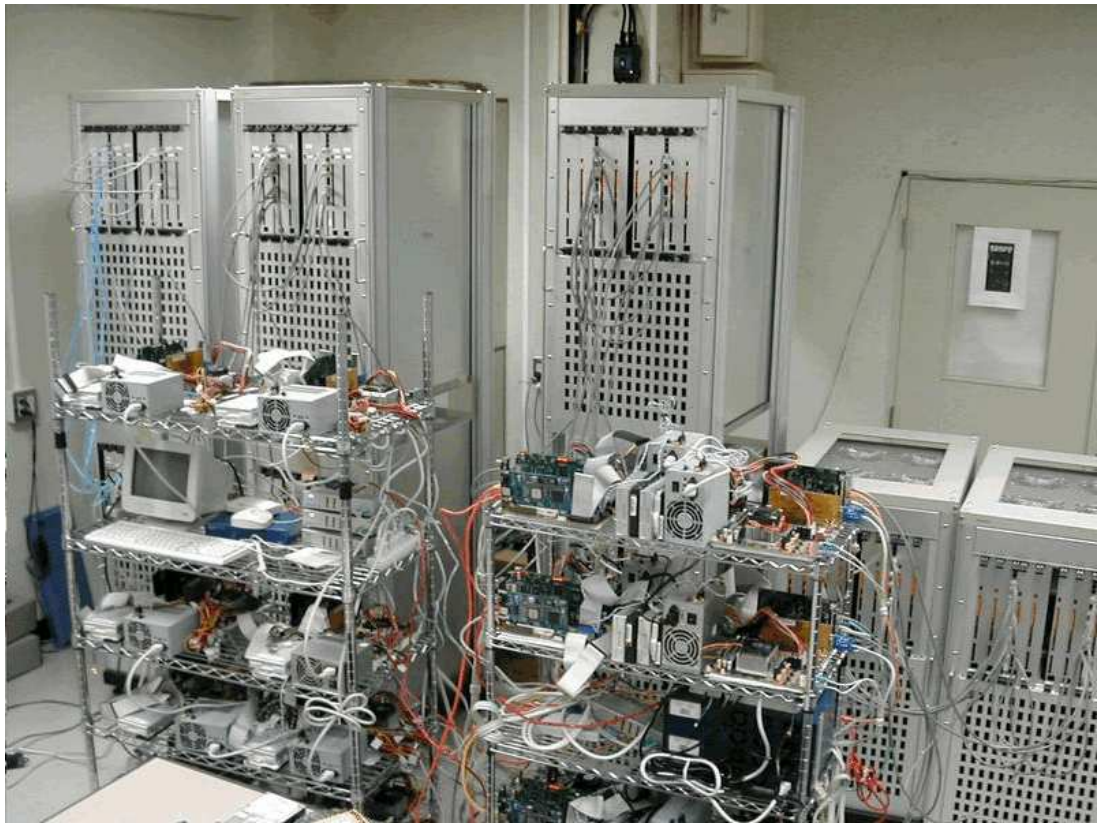
The full 64 Tflops GRAPE-6 system



- 4-host, 16-board "block" with dedicated network
- 4 (currently 3) "blocks" connected through GbE network

Combination of host network solution and dedicated network solution.

The 64-Tflops GRAPE-6 system



Present 64-Tflops system.

4 blocks with 16 host computers.

個人的な感想

GRAPE のようなものを「うまく作る」のは結構大変である。

- 対象となる問題
- アルゴリズム・要求精度
- 計算機アーキテクチャ
- 論理設計
- 実装
- OS、デバイスドライバ等のソフトウェア

の全般についてまあまあの理解がないと全体設計が出来ない。

普通の計算機を作るのに比べると

普通の計算機では

- 対象となる問題、アルゴリズム、要求精度

「必要ない」（本当？）

- それ以外

「まあまあ」ではまあまあの計算機しか出来ない。

他の誰かと同じようなことをすれば済むという面もある（１番になれるか？という問題はある）

専用計算機は難しい？

「成功」といえるようなものはそんなに多くない。

多体系に話を限るとアプローチは2つ。どちらも結構いろいろある。

専用パイプライン

- DMDP (Delft)
- FASTRUN
- GRAPE
- MD-Engine
- MDM

並列計算機

- Digital Orrery
- Transputer-based projects...
- HaMM
- 他にも沢山

あんまりうまくいかないのはなぜ？

どれが成功でどれが失敗かというのはさておき、、、
基本的な理由は 2 つ。

1. 出来たものが実は使えなかった。
2. 出来た時には速度が汎用計算機より速くなかった。

理由 (1) は専用計算機に固有だが、例は少ない (公表されていないのがあるのかもしれないが)
理由 (2) は普通の計算機と同じ。開発に時間がかかるとムーアの法則の分相対的に遅くなる。

開発期間の問題

大抵の人は (私を含めて) 見積もりが甘いというのは確か。

本質的な問題：
専用計算機の場合、開発期間が多少 (1-2 年) 延びたくらいで意味がなくなるようなプロジェクトは、そもそもあまり意味がない (出来てからのハードウェアの寿命がどうせ短い)。
大雑把には、計画時点で

- 価格性能比が 1000 倍 — 問題なし
- 価格性能比が 100 倍 — ちょっと苦しい
- 価格性能比が 10 倍 — やめたほうがいい

開発に 5 年、マシンの稼働期間が 5 年とするとこれくらい。

価格性能比 1000 倍は可能か？

あくまでも「計画時点」。とはいえ、、、

半導体技術は1-2年先を見込める：1.5-2倍得

大量生産されるマイクロプロセッサほどクロックはあげられない：3-5倍損

結局、

- チップ当り演算数が同じならチップ当りの値段を 1/1000 にする
- チップ当り値段が同じなら演算数を1000 倍にする

必要がある。

チップ当りの値段を下げる

ある程度は可能

PC クラスタに使うような計算機のノード単価: 50-100 万

ASIC の量産単価 1-10 万

ASIC の値段が半分以上を占めるようにシステムを設計できれば 5-50 倍は下がる。(開発費があるので実際には 10 倍以上は難しい)

メモリ・ネットワーク周りに手を抜けないと値段は下がらない。

チップ当りの演算数を増やす

マイクロプロセッサはクロックあたり 1 演算程度しかしない

$\longrightarrow$ 100 個以上の演算器が 1 チップに入れば OK。(GRAPE-6 の時。今だと 1000 個いれないと駄目)

入れるだけならなんということはない。 10^7 トランジスタもあれば十分。

どうやって使うか

- チップ外への通信速度
- 演算器間の通信速度
- 内部メモリ、レジスタと演算器の間の通信速度

GRAPE の場合

- 実 / 仮想マルチパイプラインによるチップ外通信速度の削減
- ハードワイヤドパイプラインによる最小コストでの演算器間接続
- 同じくハードワイヤドのため内部メモリ不要

通常 on-chip multiprocessor で起きる問題をほぼ完全に回避している。
基本的にはそういうことが出来る問題だから。
逆にいえば、そういうことが出来る問題 (アルゴリズム) でないと専用計算機はうまくいかない。

専用計算機で上手くいった例

科学技術計算用では実は結構少ないかも、、、

Google してでてきたもの:

科学技術計算専用ロジック組込み型プラットフォーム・アーキテクチャの研究 (EHPC, 九州大学)

50Tflops の分子動力学専用計算機完成 (理研)

SPH 専用計算機プロジェクト (国立天文台)

超並列 FEM-MD 専用計算機の試作 (上の九州大学の)

分子動力学シミュレーション専用計算機 MDGRAPE-3(理研)

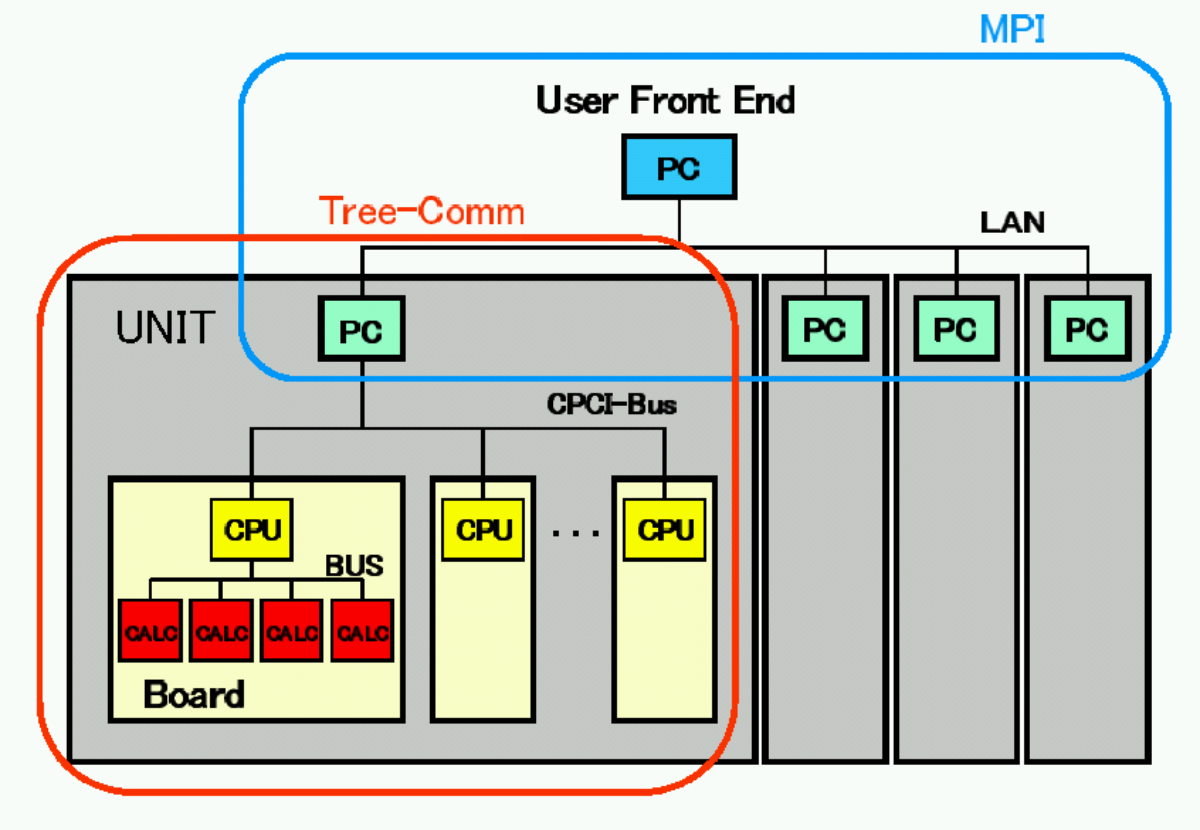
....

INFN APEmille

MOE: A Special-Purpose Parallel Computer for High-Speed, Large-Scale Molecular Orbital Calculation

EHPC システム

佐々木、長嶋 (日本コンピュータ化学会論文誌 Vol 2, No.3, 2003)



EHPC システム 詳細

(PC(互換CPCIボード)+「専用」ボード7枚)×n の構成

専用ボード: “CALC” SH4 4個 (セガドリームキャストのCPU)

“CPU” 日立超LSI製PCIブリッジ

専用ボードの名目ピーク (単精度) 5.6 Gflops

理研分子動力学専用計算機

MDM (戎崎、成見他): GRAPE-6 と同時期、ほぼ同等の性能。

GRAPE-2A, MDGRAPE の後継。

Protein Explorer (MDGRAPE-3, 泰地、成見他)

MDM の後継、現在チップ完成、量産中のはず。

チップ単体ピーク性能: 230 Gflops

目標性能: 1Pflops

SPH 専用計算機プロジェクト

SPH: Smoothed Particle Hydrodynamics

メッシュレスなラグランジュ的粒子による流体計算法

原理的には粒子間相互作用を速く計算できればいいので専用計算機向き？

92 年くらいに始まった

実用計算をするにはいたらなかった

- 計算精度の問題
- 計算法の問題 重力の法則は変わらない。差分スキームは変わる、、、
- 開発時間の問題

INFN APE

格子 QCD 計算専用計算機

- APE - INFN special project (1983-1988)
- APE100 - INFN special project (1988-1994)
- APEmille - INFN special project (with collaboration of Desy) (1995-2000)
- apeNEXT - Project driven by an international collaboration

Columbia QCD

Since 1982

QCDSP (400+600Gflops) 1998 完成 — TI DSP を CPU に使用

QCDOC 1999- 現在組み立て中のはず。 BG/L の原型 (何故 BG/L のほうが先に出来たかは不明)

他の専用計算機プロジェクトのまとめ

MD 用: GRAPE の眷族なので、、、

QCD 計算用: 20 年くらい続いている。使える機械が出来ている。

QCD 計算以外への波及効果: 元々分散メモリ並列計算機自体がここから来たもの。

他の応用: ???

うまくいった専用計算機

- 計算量が多い、メモリはあんまりいらない
- 計算の目標が割合明確で単純
- 根性がある
- 政治力がある (予算が取れる)

専用計算機の今後

GRAPE アーキテクチャ：汎用計算機に比べた相対的優位はだんだん大きくなる

ムーアの法則でトランジスタは増える

汎用計算機では増えたトランジスタの全部を演算器には使ってない

GRAPE は全部使える

チップの初期コストが上がるのが問題：予算をどうやって取る？

次期 GRAPE

振興調整費 GRAPE-DR プロジェクト (2004-2008)

基本的な考え

- チップに演算器を 2000 個くらい入れる
- それを (GRAPE が得意なタイプの問題に対しては) ある程度のプログラム可能性をもった形で使う。 GRAPE のようなハードワイヤードなパイプラインにはしない。

アーキテクチャを変える理由

- 色々できるならそのほうが面白い 予算が欲しい
- Top 500 に載りたい
- ハードワイヤードなパイプラインでなくても汎用計算機にかなり余裕で勝てる (まあ、私の見積もりは甘いわけですが)
- 同じようなのを 3 回も作るのは飽きる

専用計算機の「限界」

開発費の高騰

1990	1μm	1500万円
1997	0.25μm	1億円
2004	90nm	3億円以上？

ある程度広い応用を持つものでないと予算獲得が難しい

別のアプローチはないか？

目標:

- 応用に特化し、多数の演算器を１チップに集積、並列動作させて高い性能を得た専用計算機の特徴を生かす
- しかし広い応用範囲を実現する

明らかな矛盾？

GRAPE の高性能: 多数の演算器を１チップに集積し、小さなメモリバンド幅で全演算器を有効に働かせたことによる

相互作用の計算式に専用化したパイプラインが必須というわけではない。

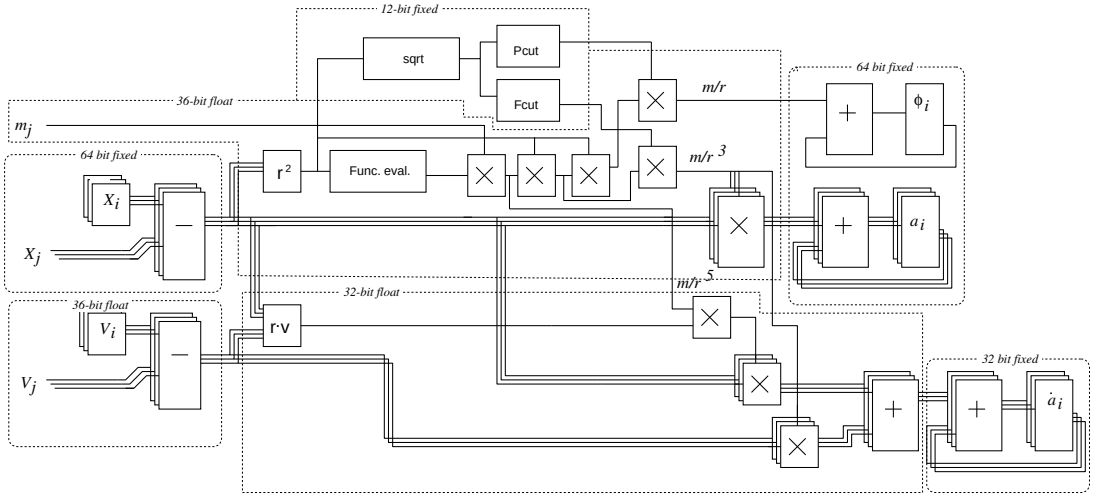
多数の演算器を詰め込む方法

境界条件: メモリバンド幅は増やしたくない (システムコストはほぼメモリバンド幅で決まる)

可能な方策

1. GRAPE 的専用パイプラインプロセッサ
2. 再構成可能プロセッサ
3. SIMD 並列プロセッサ

GRAPE的専用プロセッサ



利点:

- シリコンの利用効率は極めて高い
- 動作クロックも上げやすい

欠点

- アプリケーション限られる。多種類作るのはリソースがかかり過ぎる

再構成可能プロセッサ

FPGA ベース

- 任意のロジックを実現可能
- 集積度、速度は大きなペナルティがある
- 精度が低くてもいい応用には向く

いわゆる「動的再構成可能プロセッサ」

IPFlex DAP/DNA 等

- 8-32 ビットの単純な ALU を多数集積
- その間をプログラマブルな配線でつなぐ
- 集積度、速度はやはり大きなペナルティ

パイプラインは必要？

パイプラインを使う (使える) 理由:

- 規則的で並列性の高い計算をする
- 入力データ一つに対して多数の計算をする
- 専用パイプラインの場合には配線リソース等が不要になる

パイプライン (ベクトル) 処理と並列処理

ベクトル化できる処理 (プログラム) と並列化できる処理:
基本的は同じ。

パイプラインの代わりに単純な SIMD 並列処理ではいけない
のか？

「教訓」: SIMD 超並列プロセッサは歴史上の存在

Goodyear MPP

ICL DAP

CM-1/2

MasPar MP-1

何故 SIMD 超並列は消えたか？

基本的な理由: Memory Wall

多数のプロセッサを LSI に集積 → プロセッサ数にみあうだけのメモリバンド幅が必要。

現在ではこれは全く不可能。

例: 2億トランジスタ → 64ビット乗算器が 1000個。

19万本の配線 が必要

しかし、 GRAPE ではそんな問題は起きていない

GRAPE と同じようなことをやらせるなら I/O 速度への要求は何桁も下がる。

GRAPE における並列性

する計算:

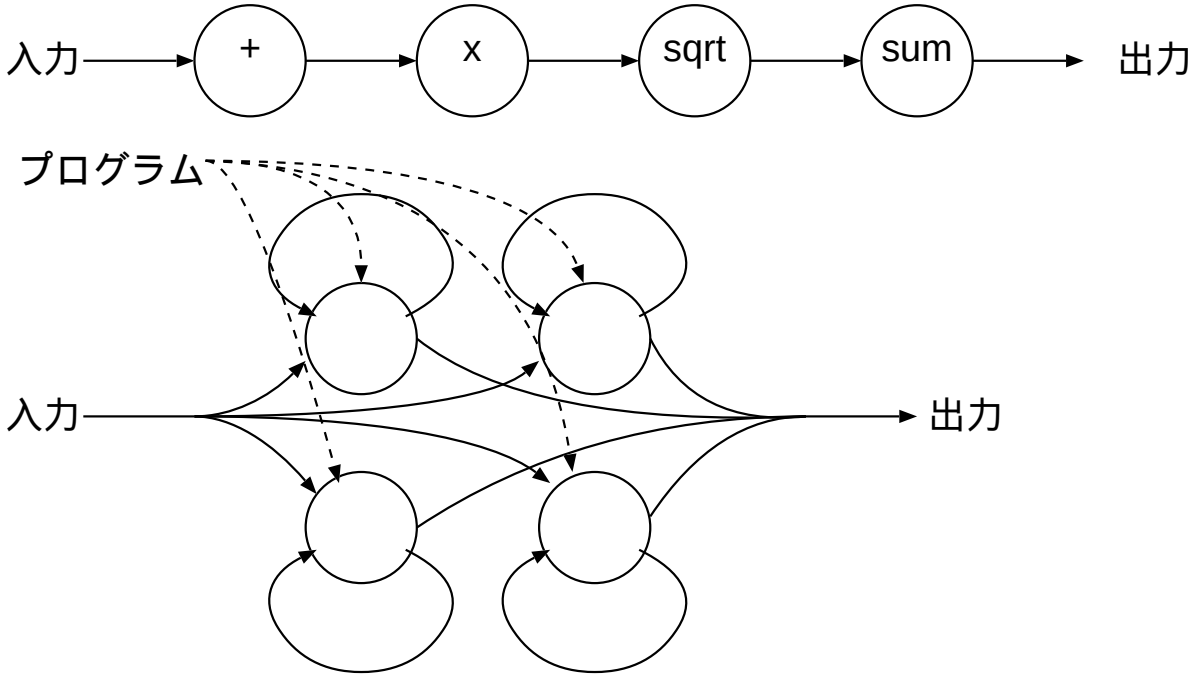
$$a_i = \sum_j f(r_i, r_j, m_j)$$

i についても j についても並列 (j は総和が必要)

1. パイプライン演算 (j 並列)
2. 物理/仮想マルチパイプライン (i 並列)
3. 複数チップ (i 並列/j 並列)

「パイプラインであること」はあまり本質的ではない?

パイプライン処理と SIMD 並列処理

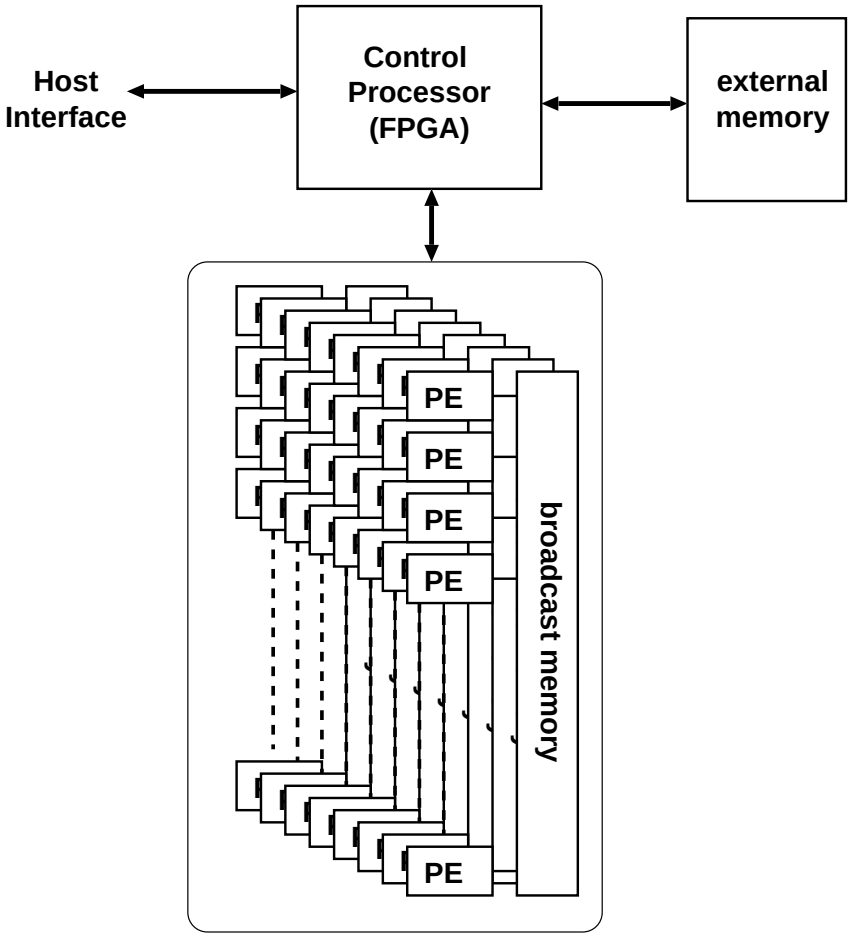


パイプラインでできることは SIMD 並列でもできる (縮約演算は追加ハードウェア必要)

専用パイプラインはいろいろメリットある

再構成可能だと、、、

GRAPE-DR の アーキテクチャ



- 非常に多数のプロセッサエレメント (PE) を 1 チップに集積
- PE = 演算器 + レジスタファイル (メモリをもたない)
- チップ内に小規模な共有メモリ (PE にデータをブロードキャスト)。これを共有する PE をブロードキャストユニット (BU) と呼ぶ。
- 制御プロセッサ、外部メモリへのインターフェースを持つ

「GRAPE として」使う

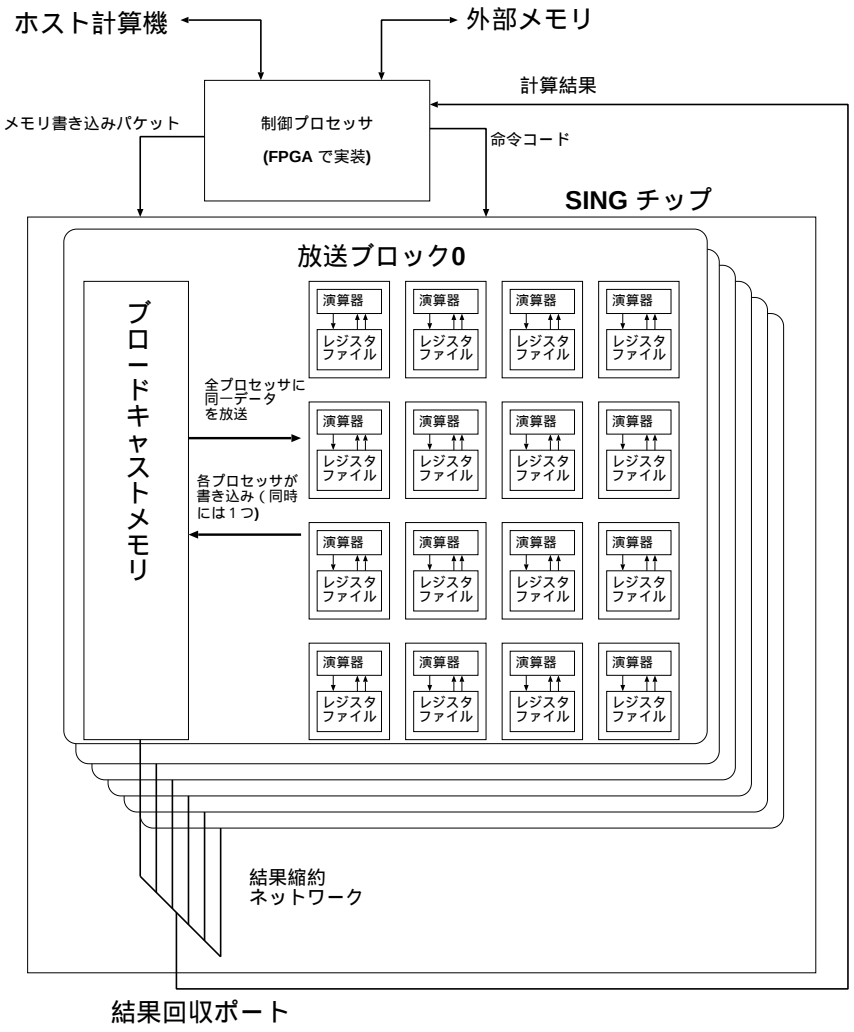
最も単純には

- 全ての PE が、自分の粒子への、同じ粒子からの力を計算
- 力を及ぼすほうの粒子データは外部メモリから供給

現実問題としては、実用アプリケーションで性能を出すのはもうちょっと面倒くさい。

- 違うブロードキャストユニットの同じ位置の PE には同じ粒子データを書く
- 力を及ぼすほうの粒子データはブロードキャストユニット毎に違うものにする。

ブロードキャストユニットの構成



- 命令コード、ブロードキャストメモリのアドレス等は外から供給される。
- 各 PE は同じデータを受け取る。

PE の詳細

演算:(最低限) 以下をサポート

- 倍精度浮動小数点加減算
- 倍精度固定 (ブロック) 小数点加減算
- 単精度浮動小数点乗算。倍精度はマルチサイクルで
- 区分多項式近似・スケーリングに必要なビット操作
- 条件付き実行

メモリオペレーション

ローカル共有メモリ：ブロードキャスト、シリアルな読出し、書き込み

汎用ポート：リダクション演算、放送

ターゲットアプリケーション

今までの GRAPE 的なものの置き換え

- 粒子法: 重力、分子間力、流体的相互作用、、、
- 境界要素法: ポアソン方程式、ヘルムホルツ問題、、、
- ルジャンドル展開による極座標流体計算

プロジェクトとしての目標

- 密行列ソルバ — Linpack で 500 Tflops 以上

もうちょっとまともな目標

- 分子軌道法での 2 電子積分
- 密行列の対角化、直交化

予算

平成１６年度科学技術振興調整費新規課題

「分散共有型研究データ利用基盤の整備」

総額 3 億 $(-\alpha)\times$ 5 年

最終システムのイメージ

2007 年度に基本的には完成 (2008 年度に増強)

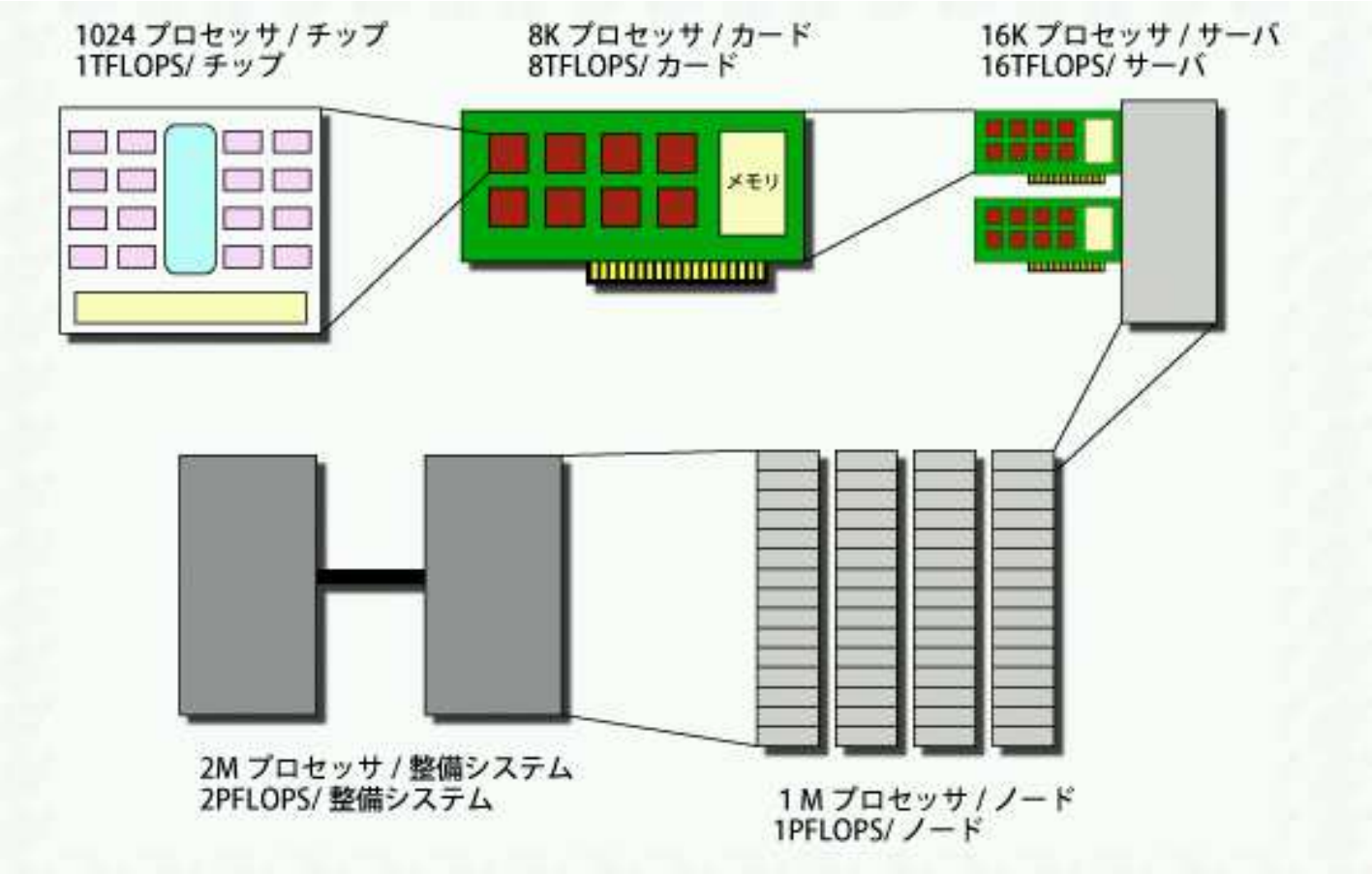
ピーク性能 2Tflops (単精度)

プロセッサチップ 2048 個。

プロセッサボード当り 4 チップ (PCI-X/Express)

ホストは 512 ノードの PC クラスタ

最終システムのイメージ



GRAPE に比べるとどれくらい損か？

GRAPE-6: 200 万ゲート、 400 演算 = 5K ゲート/演算

PE の大きさ (推定):

ユニット	サイズ
fadd	7K
fmul	15K
register file 等	30K?
合計	47K

乗算・加算同時実行可能だとすると、合計では 25K ゲート/演算。純粋な GRAPE に比べると 5 倍損

共有メモリやインターフェース回路はサイズとしては無視できる。

類似のアプローチとの比較 (1)

- **SIMD 超並列計算機 (Goodyear MPP, CM-1/2)**
  - 外部メモリへのバンド幅、通信ネットワークでプロセッサ数が制限
  - プロセッサがメモリを共有することでプロセッサ数の制限を回避。応用範囲には制限。
- **FPGA**
  - トランジスタ利用率が低い
  - ソフトウェア開発が困難
  - どちらも回避

類似のアプローチとの比較 (2)

再構成可能計算機

- **DSP ブロック入り FPGA**
  - ゲート効率 **FPGA** よりはちょっといい？
- **DAP/DNA, DRP**
  - ゲート効率悪い
  - 動作速度遅い

他のアプローチとの基本的な違い:

チップ内にスイッチングネットワークをもたない

- (多少) できないことも発生

- デザインの単純化 → 高集積、高速動作 (理論上は、、、)

ソフトウェア

PE のプログラムは、基本的にシーケンシャル
(SIMD アプローチの利点)

- アセンブリ言語
- 単純なコンパイラ

上位のソフトウェア

- 複数ホスト+超並列ボード上の並列化
- ホスト計算機/超並列ボードのタスク割り当て

大規模科学技術計算の今後

GRAPE-DR のような非常に計算量の多い規則的計算に向いたアーキテクチャ:
多分結構順調に進歩する (CELL みたいなのも含めて)

規則的だけどメモリバンド幅が必要な (ということになっている) 種類のアプリケーション
本当に不規則なアプリケーション

- 普通の PC (クラスタ) で我慢？
- もうちょっと違う何かを考える？

ポスト地球シミュレータ ???

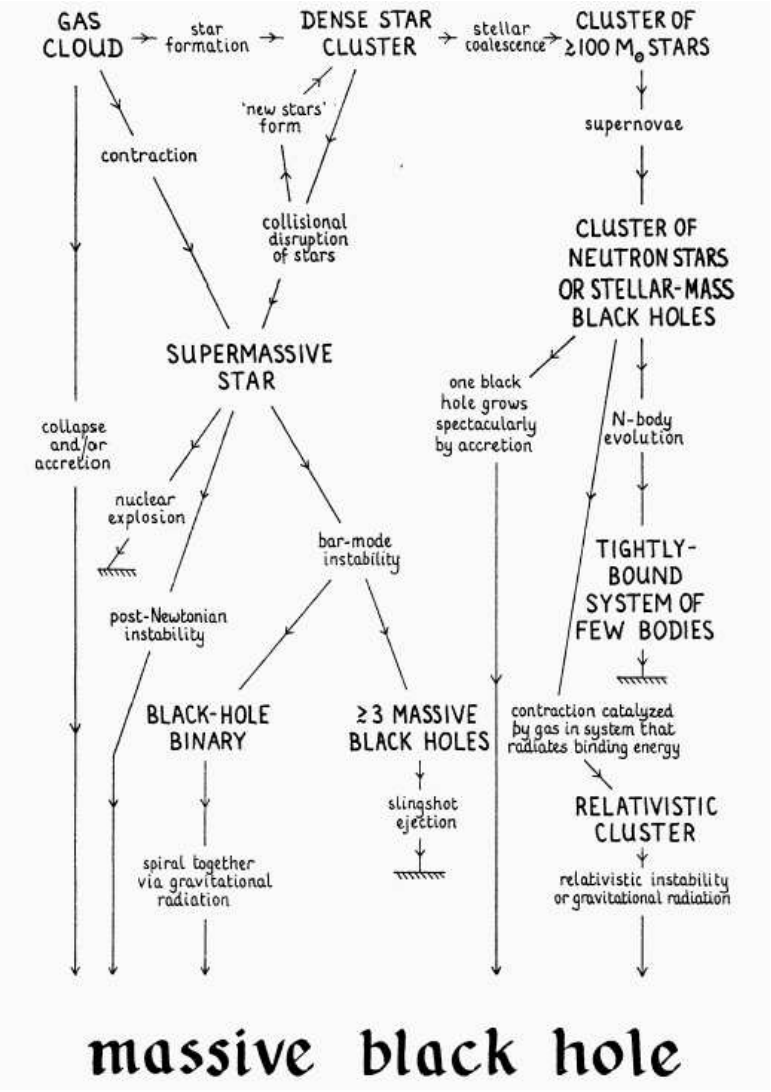
まとめ

- GRAPE では、もっとも演算量の多い相互作用の計算だけを専用化し、他の計算は汎用計算機に回すことで高い性能と柔軟性を両立させている。
- この分担のためにハード開発は比較的容易である。
- このような分担をうまく機能させるためには、採用するアルゴリズムがそういうふうにできている必要がある。可能ならアルゴリズムの変更も考えるべき。
- とはいえ、次世代プロジェクトは専用性が薄くなる。

中間質量ブラックホール — シミュレーションと観測の関係の例として

1. はじめに:大質量ブラックホールの作り方
2. M82 の IMBH 候補
3. 合体シナリオ
4. 他のモデル？
5. 他の IMBH 候補
6. 普通の球状星団に IMBH はあるか？
7. まとめ

はじめに:大質量ブラックホールの作り方



Classic View (Rees 1984)

本質的には 2 通り

- 単一の超大質量星
- コンパクト星の高密度クラスターの熱的な進化

どちらも簡単ではない...

理論はともかく

観測的にギャップがあるのが問題: 完全に空中楼閣を作るのは
(よほど物理が単純でないと) 難しい。

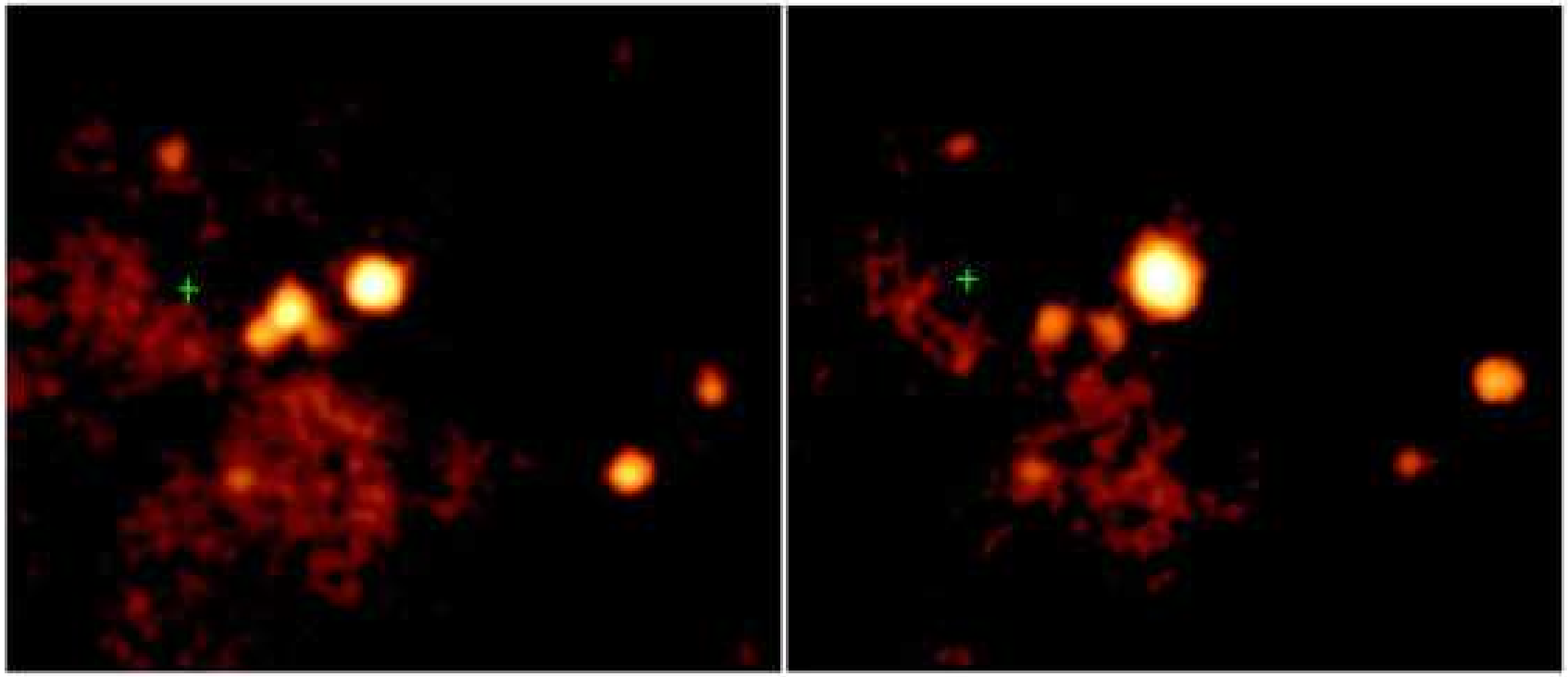
観測的なギャップ

- 恒星質量 $\mathrm{BH} \sim 10 M_\odot$
- 超大質量 $\mathrm{BH} > 10^6 M_\odot$

中間は ???

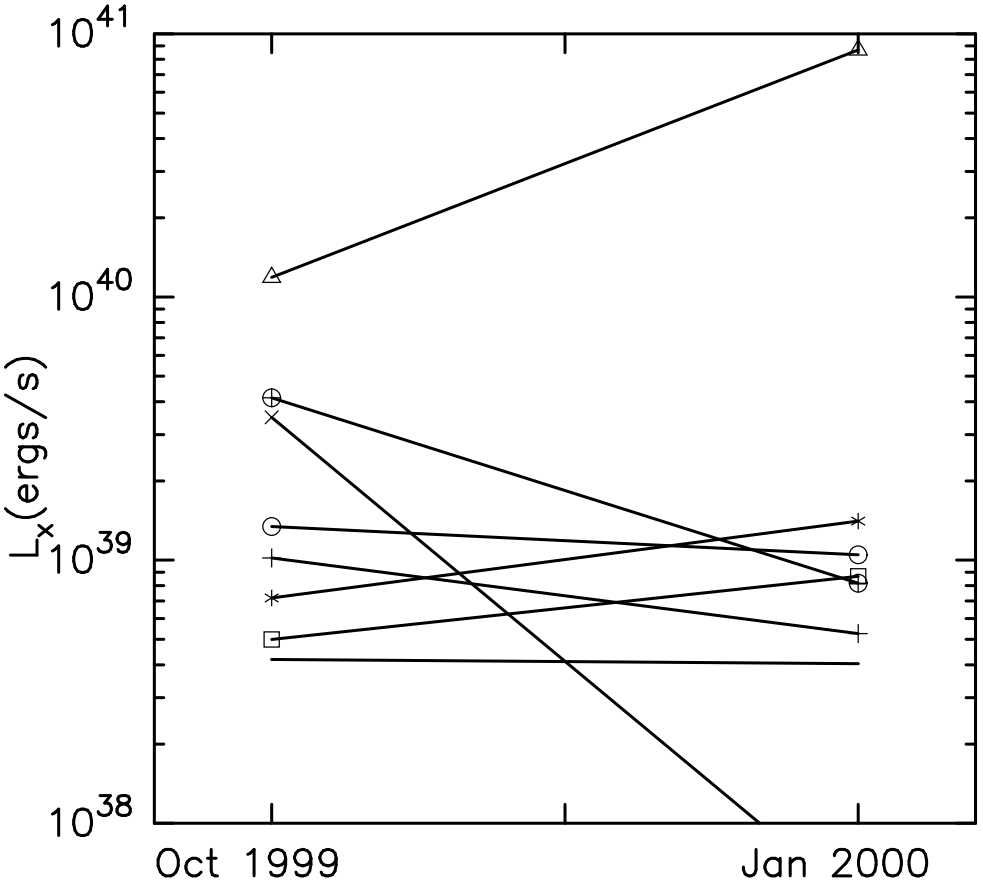
M82 の中間質量 BH 候補

Matsumoto *et al.* ApJL 547, L25



大きな時間変動を示す複数のソース

時間変動



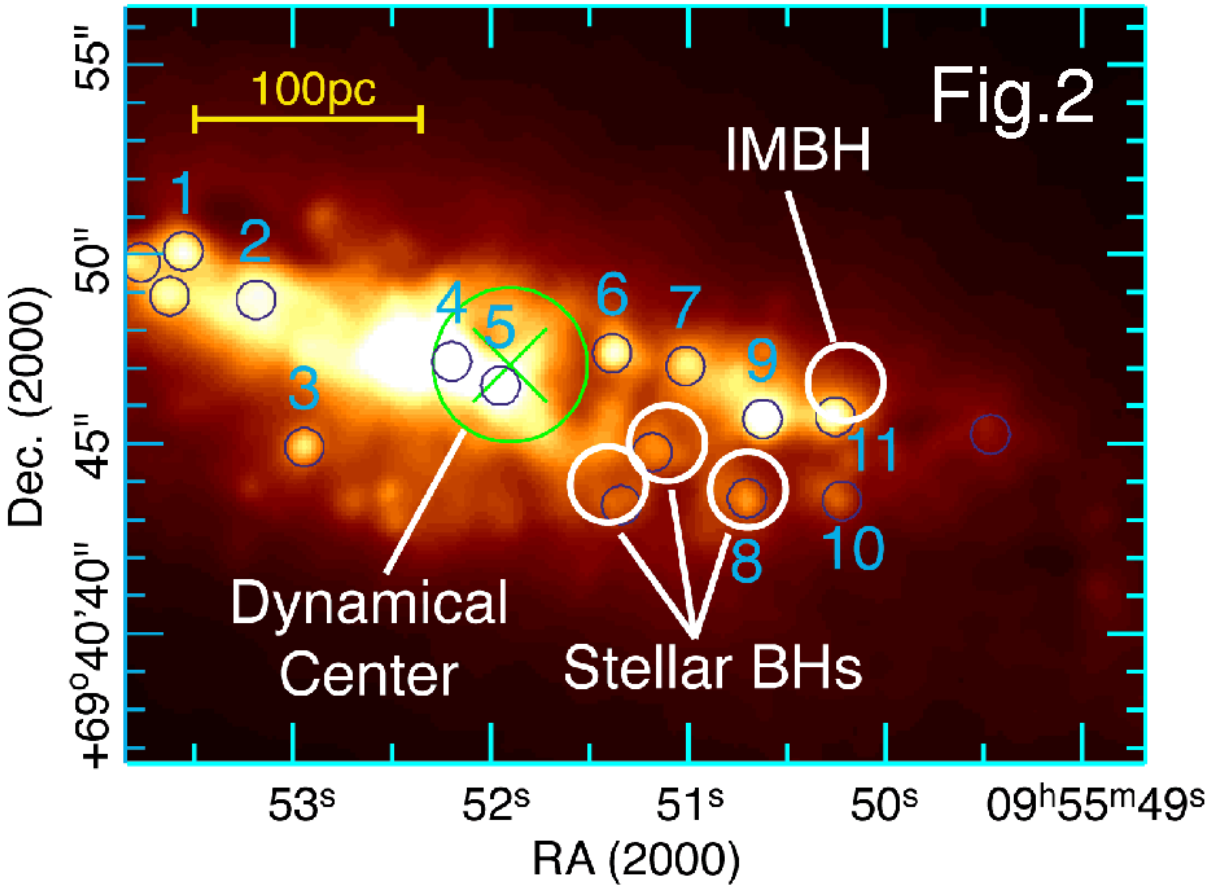
最大のソースが大きな時間変動

M82 IMBH (候補) の意味

- 最初の 質量が $>> 10, << 10^6$ の BH 候補天体
- エディントン質量 $\sim 700 M_\odot$ = 最初の IMBH (intermediate-mass BH).
- M82 の中心から 200 pcs くらい離れている: 銀河中心の BH そのものではない

赤外 カウンターパート

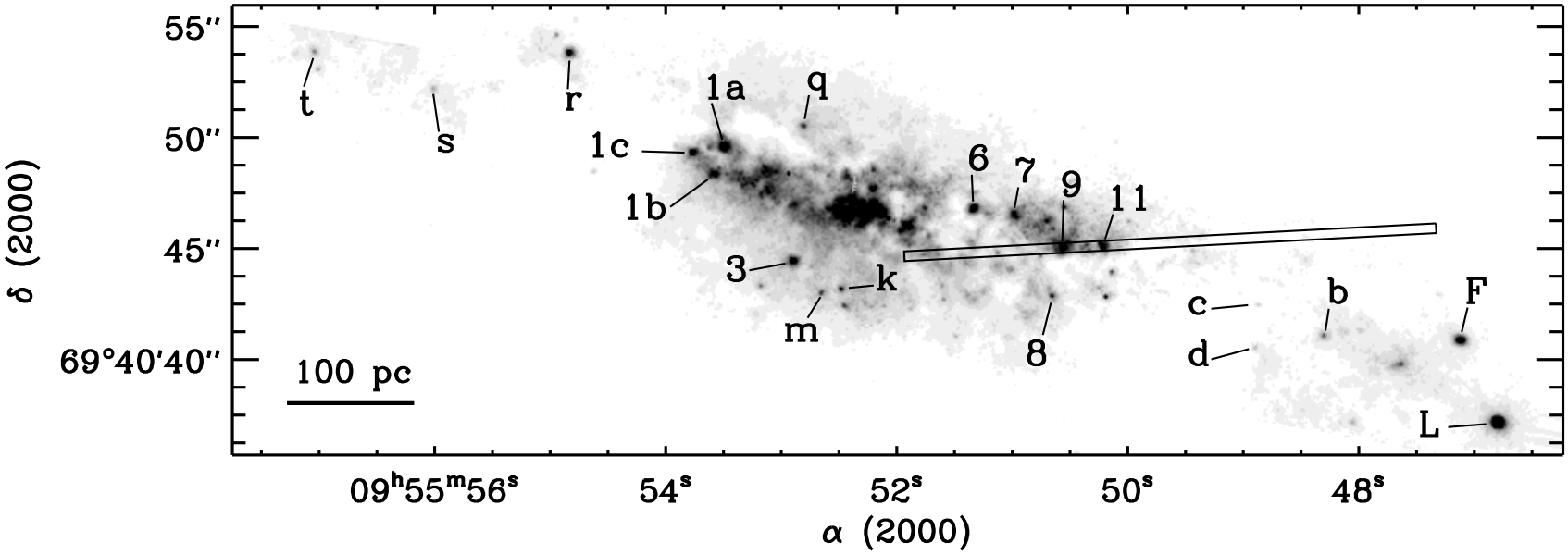
M82 のすばるによる観測 (K' band)



赤外 カウンターパート(2)

HST NICMOS/Keck NIRSPEC

McCrady et al. (astro-ph/0306373)



スターバーストで最近出来た非常に若い星団。

IMBH は星団の中にある？

*(*理論家の目には*)* IMBH は若くてコンパクトな星団の中にあるというのは「明らか」。

では、 IMBH はどうやってできたか？

- 星団と同時に出来た？(あんまりありそうにない)
- 星団の中で作られた。

ホスト星団の力学的な特徴

McCrady et al. 2003 (astro-ph/0306373)

Cluster #11 (MGG-11)

- 星のランダム速度 $\sigma_r = 11.4 \pm 0.8\mathrm{km/s}$
- 半径 $1.2 \pm 0.17\mathrm{pc}$
- 質量 $3.5 \pm 0.7 \times 10^5$ 太陽質量
- 年齢 $\sim 10^7$ 年

質量光度比が割合低い (重さの割りに明るい == 重い星が多い)

熱力学的緩和時間非常に短い ($< 10^7$ 年)

BH がない MGG9 は緩和時間 4 倍くらい長い

一つの可能なシナリオ

1. 熱力学的な緩和 (力学的摩擦) により大質量星が中心に沈む
2. 星団の中心で星の暴走的な合体で大質量星ができる
3. この星のコラプスで IMBH (の種) ができる
4. この IMBH (の種) がさらに他の星と合体して成長

シミュレーション

Portegies Zwart et al., Nature, 2004/4/15 号

初期条件

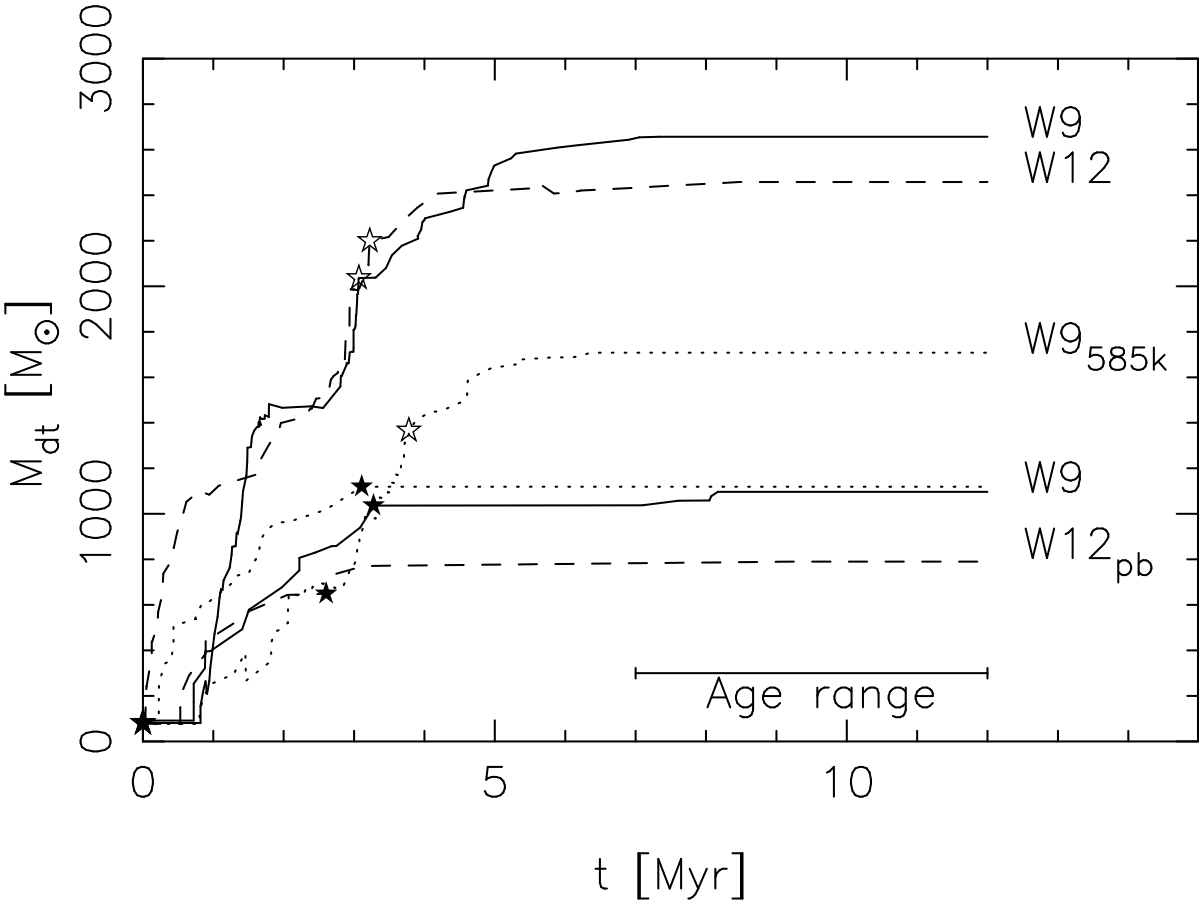
- **King model with $W_0 = 7$-12** (中心ポテンシャルの深さ)
- **Star-by-star simulation for MGG-11 (MGG-9 is scaled)**
- 星の進化は **Eggleton et al.** によるテーブル。
- **50** 太陽質量以上は、、、「外挿」、一部の計算では石井・加藤による半径を利用。
- 星同士は接触したら合体、適当な **mixing** を仮定。

あんまりサイエンスではない話

いかにしてこの論文が Nature に載るにいたったか？

1. まず Simon (Portegies Zwart) が 1 ページくらいの概要を L. Sage に送った。
2. Sage からは「つまらんからいらない」みたいな返事が来た。
3. Simon が Sage に電話して談判した。
4. 何故か「ではレフェリーに回すから論文送って」という話になった。

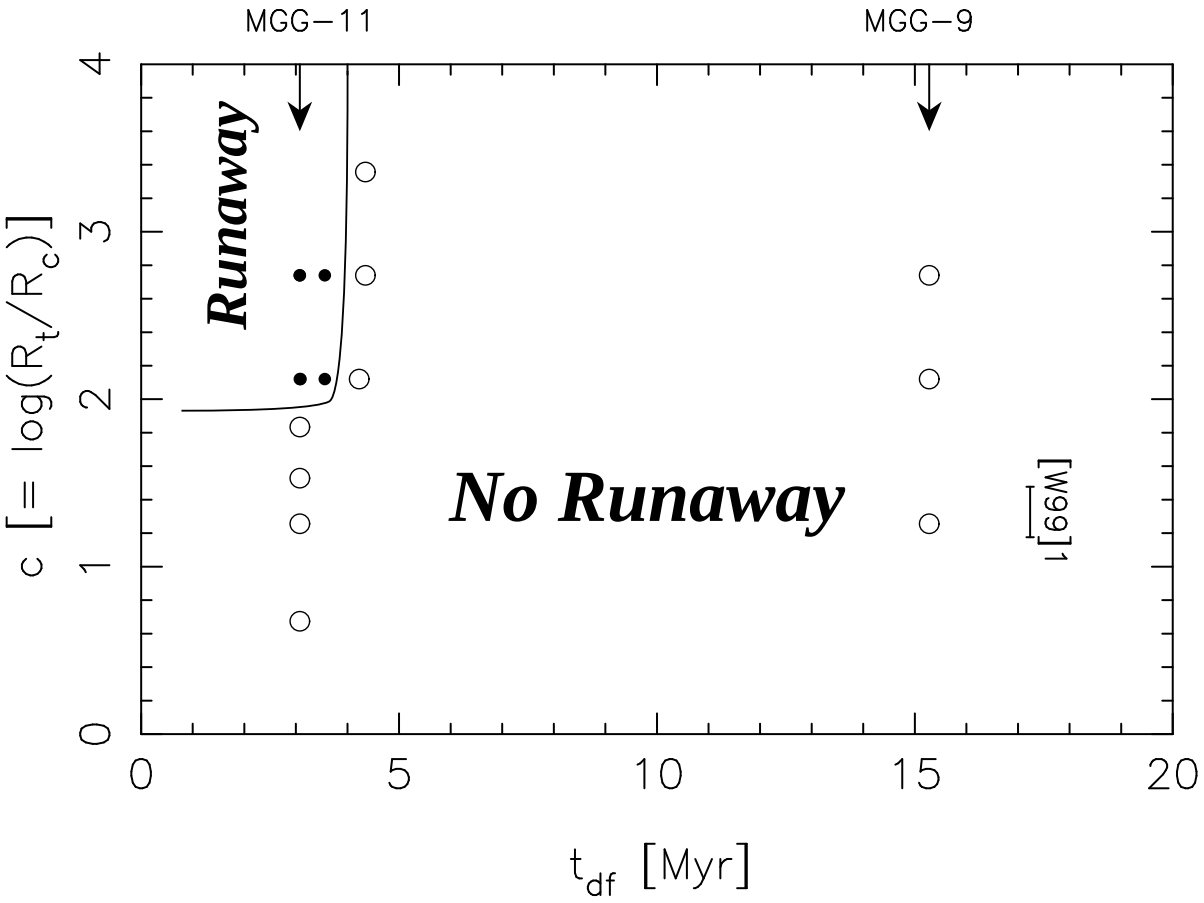
結果



$W_0 \geq 8$ なら暴走的合体 (MGG-11 では)

MGG-9 (緩和時間長い) では暴走的合体はおきない

結果のまとめ



緩和時間が短く、かつ初期に小さいコアを持つ必要がある

IMBH 形成についてのまとめ

- 暴走的な合体による IMBH の形成は、理論/数値実験の結果を見る限りありそう。
- 暴走的な合体が起きる条件は星団の緩和時間が短く高密度なコアを持つこと

暴走的合体は、少なくとも、M82　の星団の中にある IMBH 候補を説明する極めてもっともらしくモデルではある。(他になんかあるわけでもない)

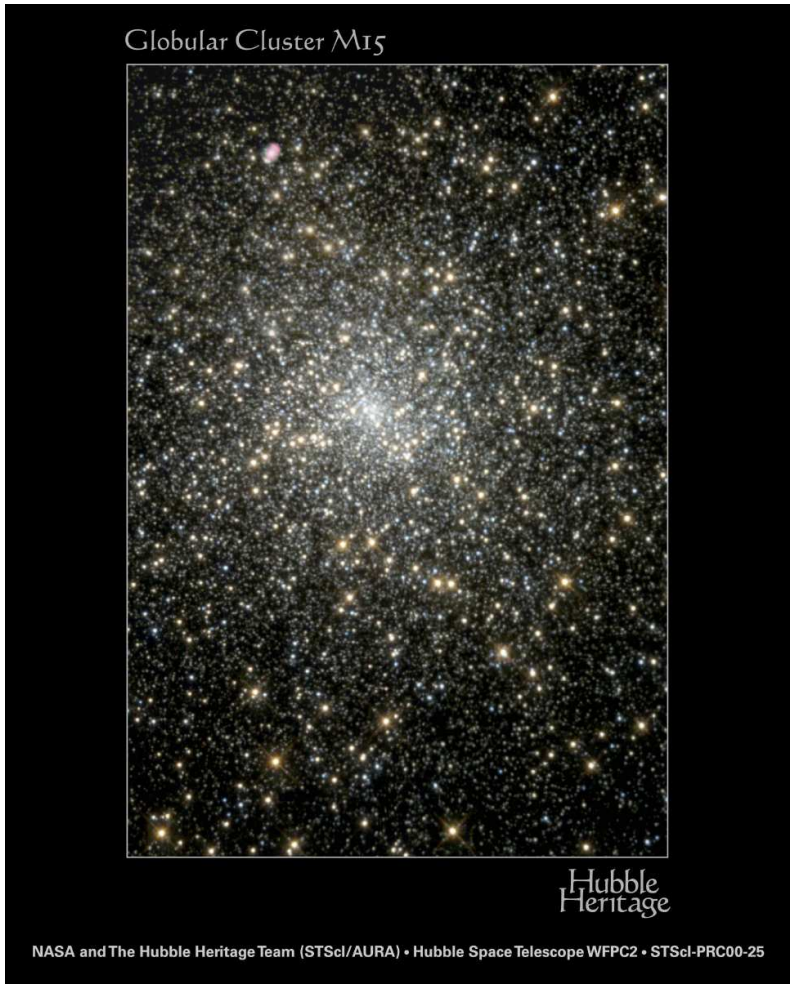
球状星団に IMBH がないのは何故？

教科書的回答:

1. 球状星団の緩和時間は長い $\sim 10^9$ yrs.
2. コアも多分初期に小さくない

本当に? M15 と G1 は？

M15



古典的 “Core Collapse” 星団。

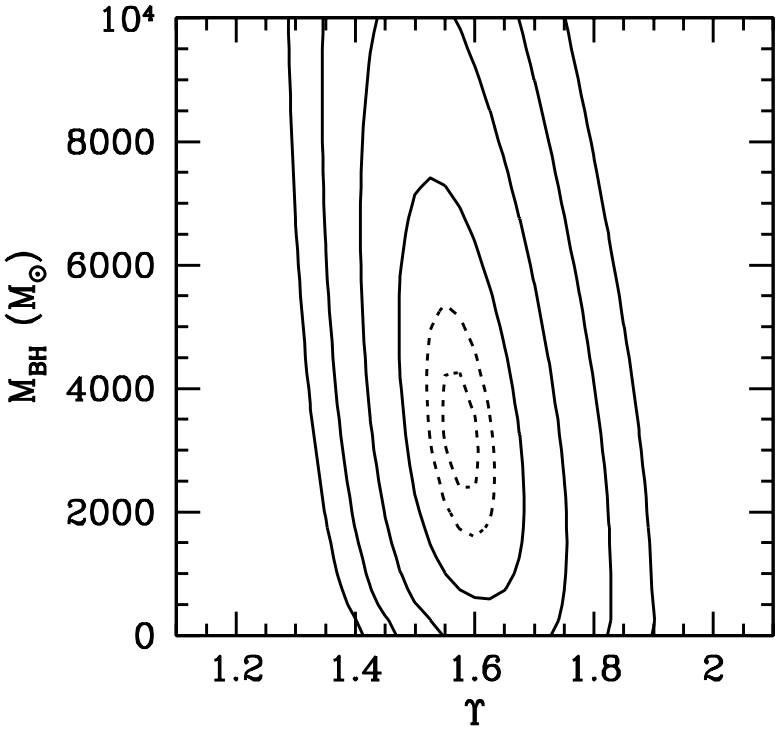
個々の星まで分解して star count で密度プロファイルが書ける。
コアはなく中心までカスプ。

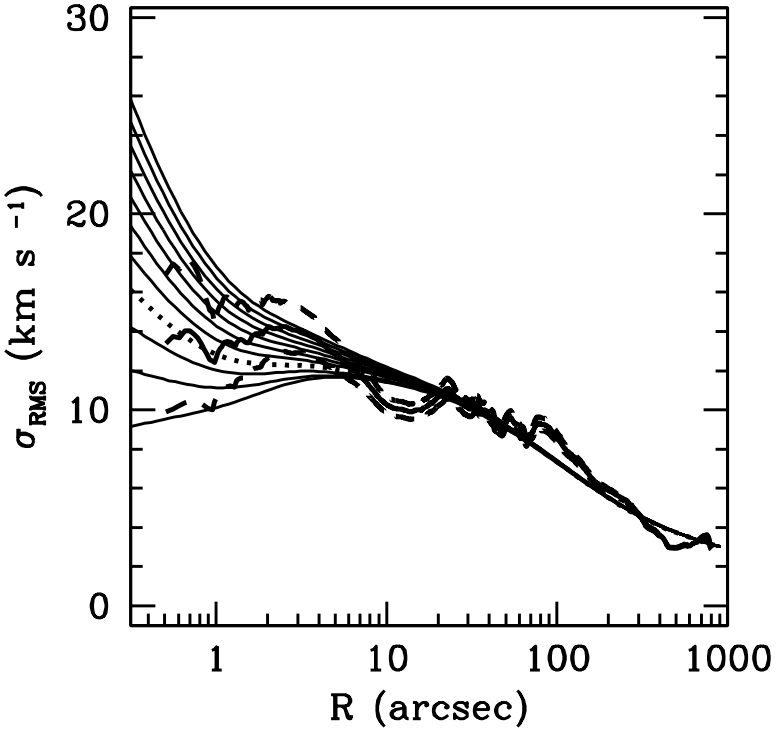
カスプの傾き ~ -0.8,

中心 BH と 重力熱力学的崩壊のどちらとも「矛盾しない」。

IMBH ありの力学モデル

Gerssen et al. 2002

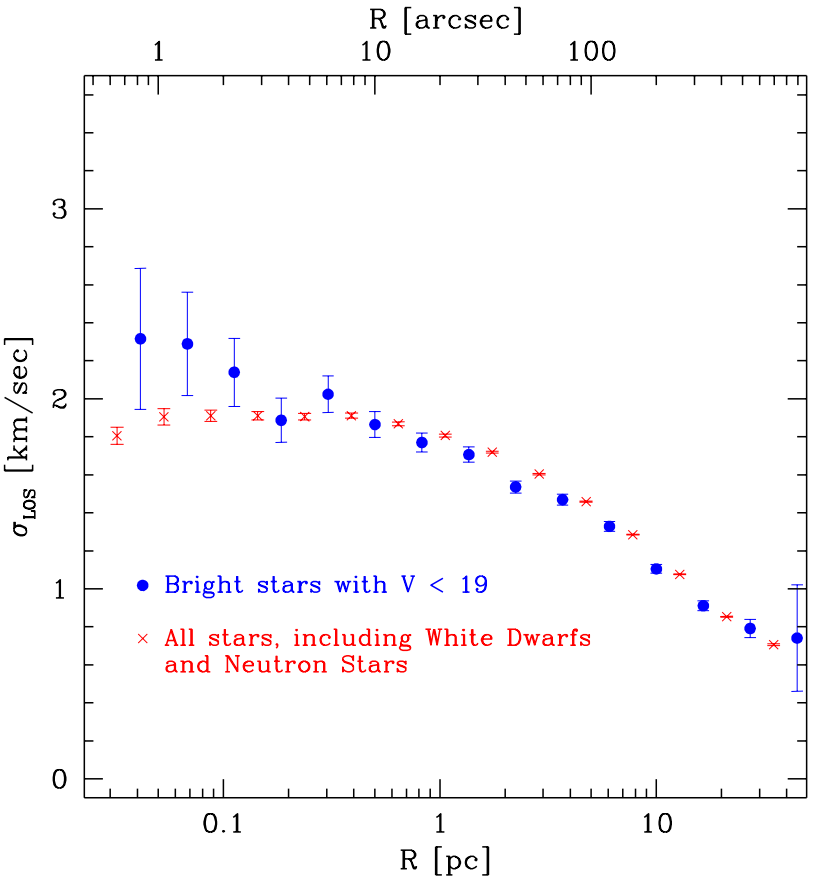


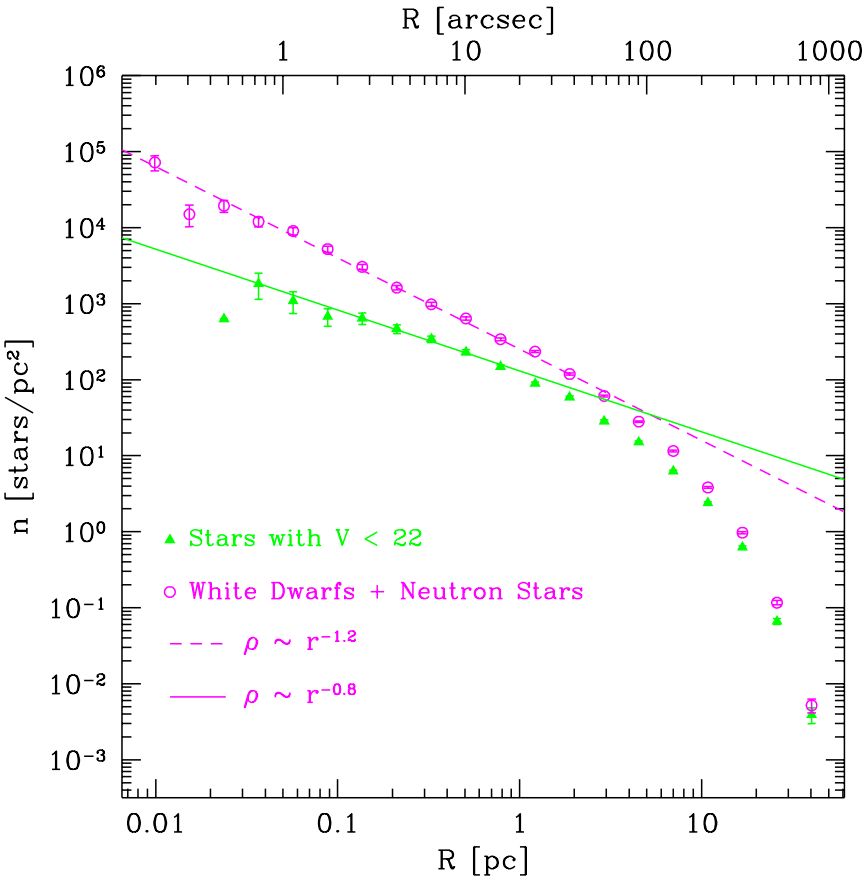


3000 $M_\odot$ black hole !?

BH なしの進化モデル

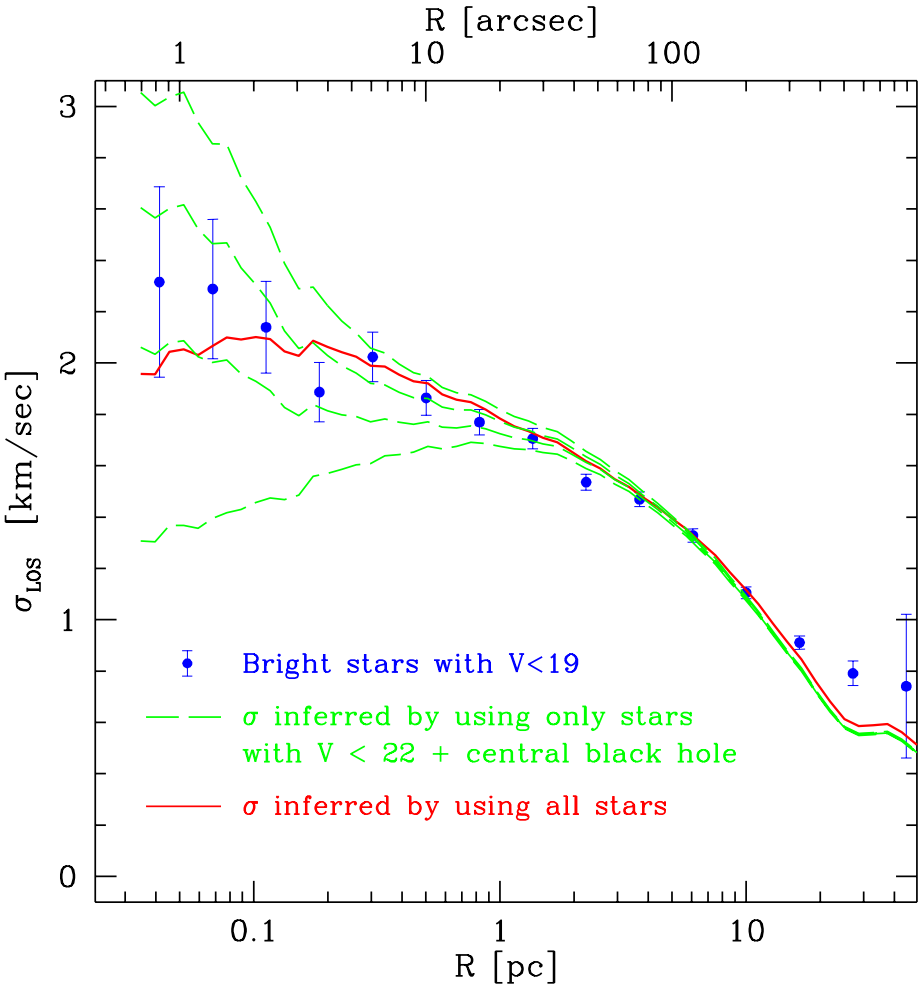
N体計算 Baumgardt et al.(ApJ 2003, 582, L21)





左:速度分散 右:表面輝度

我々のモデルに「ブラックホール発見」



我々のシミュレーション結果を観測と同じ方法で解析すると「ブラックホールがある」という結論になる

mass segregation によって中心に集まった中性子星と重い白色矮星が「ダークマス」として振舞う。

球状星団に IMBH はあるのか？

M15 には「極めてありそうにない」。標準的なモデルで完璧に説明できる。

(詳しい話はしませんが)
G1 はもっとありそうにない (Baumgardt et al. ApJL, 2003, 589, L25)

球状星団に IMBH はあるのか？(続き)

探していた場所は間違ってないか？
そもそも BH のある球状星団はどんなふうに見えるのか？

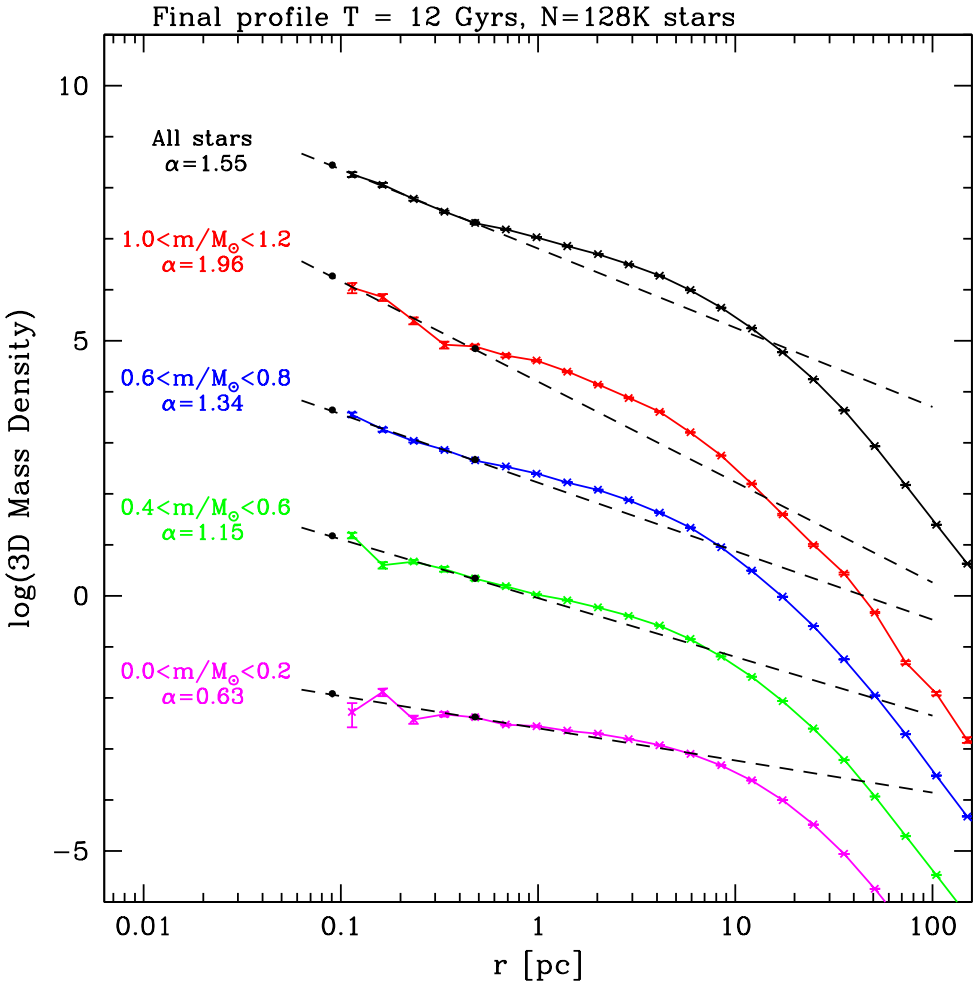
理論 (Bahcall and Wolf 1976): $\rho \propto r^{-7/4}$.
表面輝度のカスプの傾きは -0.75 のはず。

注意:

- -0.75 はあくまでも漸近的な傾き
- 全ての星が同じ質量と仮定

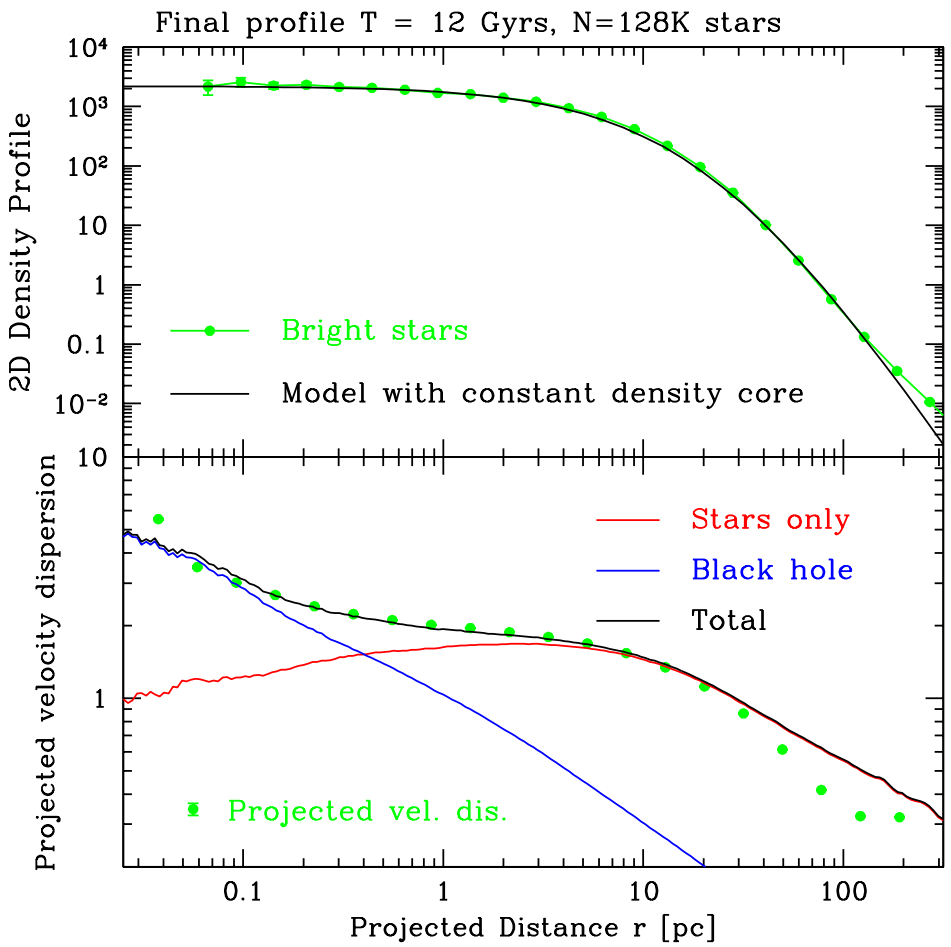
もっともらしい初期条件から、 IMF と恒星進化モデルを入れて、IMBH がある球状星団を進化させてみた。(Baumgardt et al 2004 a, b, ApJ)

3次元 密度分布



- 軽い星はスロープ浅い
- 見える星 (太陽質量の 0.8 倍くらい) はスロープ −1.3 くらい。単純な理論モデルより浅い。

投影



- どう見てもフラットなコアがある
- 速度分散は「コア」の中心近くで微妙に増加

IMBH があるのは一見普通にコアがある球状星団のほうと思われる

まとめ

- M82 に発見された IMBH 候補は、恒星質量ブラックホールと大質量ブラックホールをつなぐ「ミッシング・リンク」である (可能性が高い)。
- 我々は、星団内での大質量星の合体で IMBH を形成し、銀河中心での IMBH の合体で SMBH を形成する 2 段階シナリオを提案する。
- 数値シミュレーションでは IMBH が形成するかどうかは星団の初期の構造に強くよるが、 観測と合う自然なパラメータで IMBH ができる。
- 球状星団 M15, G1 の観測結果は BH なしの進化モデルで良く説明できる。
- 仮に球状星団に BH があるとすれば、 M15 のような PCC クラスタではなく一見普通に見えるものの中心に隠れている。精密な分光観測で発見できる可能性がある。

以下は補助スライド

GRAPE における並列化

パイプライン1本が速くても、並列化できなければたいした意味はない。

ホストは1台として、 GRAPE の側の並列化を考える。

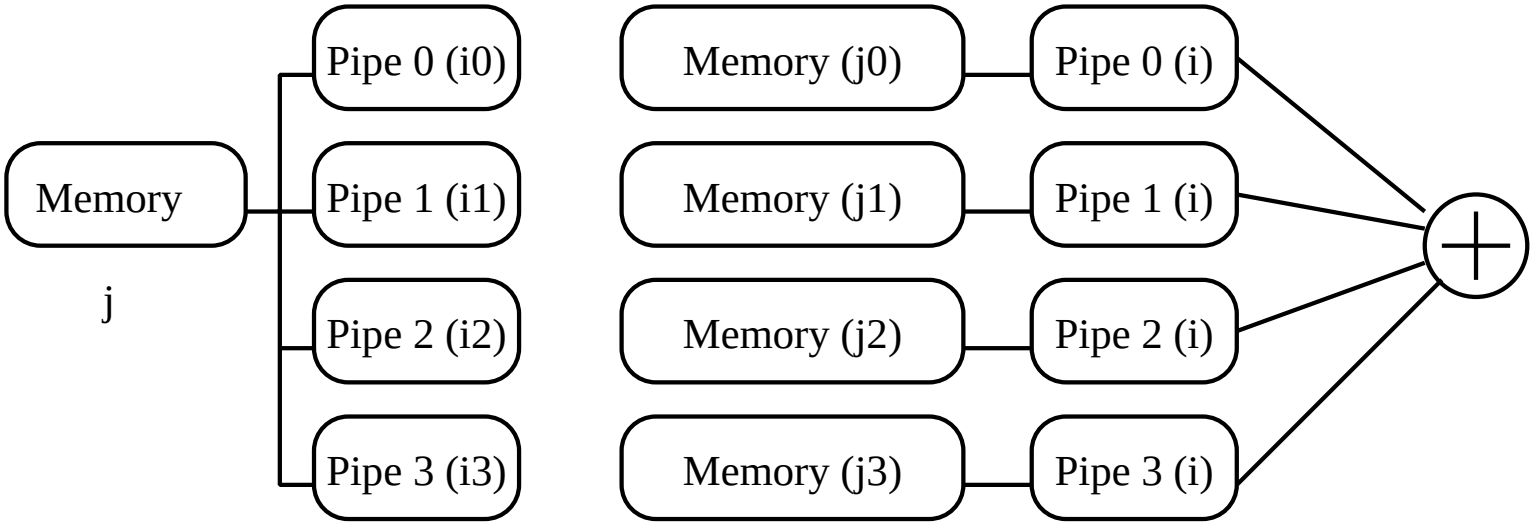
方針は 2 つ

- 複数の粒子への力を並列に計算する (i 並列)
- 複数の粒子からの力を並列に計算する (j 並列)

実現方法

i 並列 : 一つのメモリからのデータを複数パイプラインに配る。

j 並列 : 複数の (メモリ + パイプライン) に同じ粒子への力を計算させて、終ってから合計する。



i-parallel ***j-parallel***

並列パイプラインの実装

メモリを共有できる範囲 (ボード / チップ):

i 並列

それより上： j 並列

メモリデータの複製を許すなら i 並列も使える

GRAPE-4: ボード上 i 並列、ボード間 j 並列

GRAPE-6: チップ内 i 並列、ボード上 j 並列

Case Study： GRAPE-6 の場合

演算方式、論理設計： GRAPE-4 のものを多少改良 / 変更。
大きな問題はなかった。
実装 : テクノロジ微細化、低電圧化のためにいろいろ問題が起きた。

- レイアウトが収束しない (タイミング要求を満たせない)。
- チップ内で電源電圧が下がる。
- 負荷変動に電源が対応できない。

レイアウトが収束しない

(最近はましになってきてるかもしれないけど) ディープサブミクロン設計でありがちな問題。

配線遅延 ＞＞ スイッチング遅延

→ レイアウトしないと遅延時間がわからない

従来の論理設計段階では仮配線長を仮定するやりかたでは駄目。

本来、レイアウトと論理設計を並行して進めるような手法が必要

GRAPE での配線遅延

GRAPE では

- 長い配線はパイプラインとインターフェース回路の間くらいしかない
- そこのタイミング要求はあんまり厳しくない (マルチサイクルでいいのが多い)

ので比較的楽で、大幅なレイアウト修正なしでなんとか誤魔化していた、、、

実際、ほぼ同じ時期に同じ会社が作っていた高速マイクロプロセッサに比べれば半分以下の期間でレイアウトが終った。

チップ内で電源電圧が下がる。

最初のチップでは

- パッケージの電源ピンからダイのパッドまでの配線抵抗
- ダイ上の電源供給ネットの抵抗

が大きかったために、内部で電圧が下がってまともに動かなかった。

計算中と非動作時では2倍程消費電力が違うので、電圧もその分変わる。

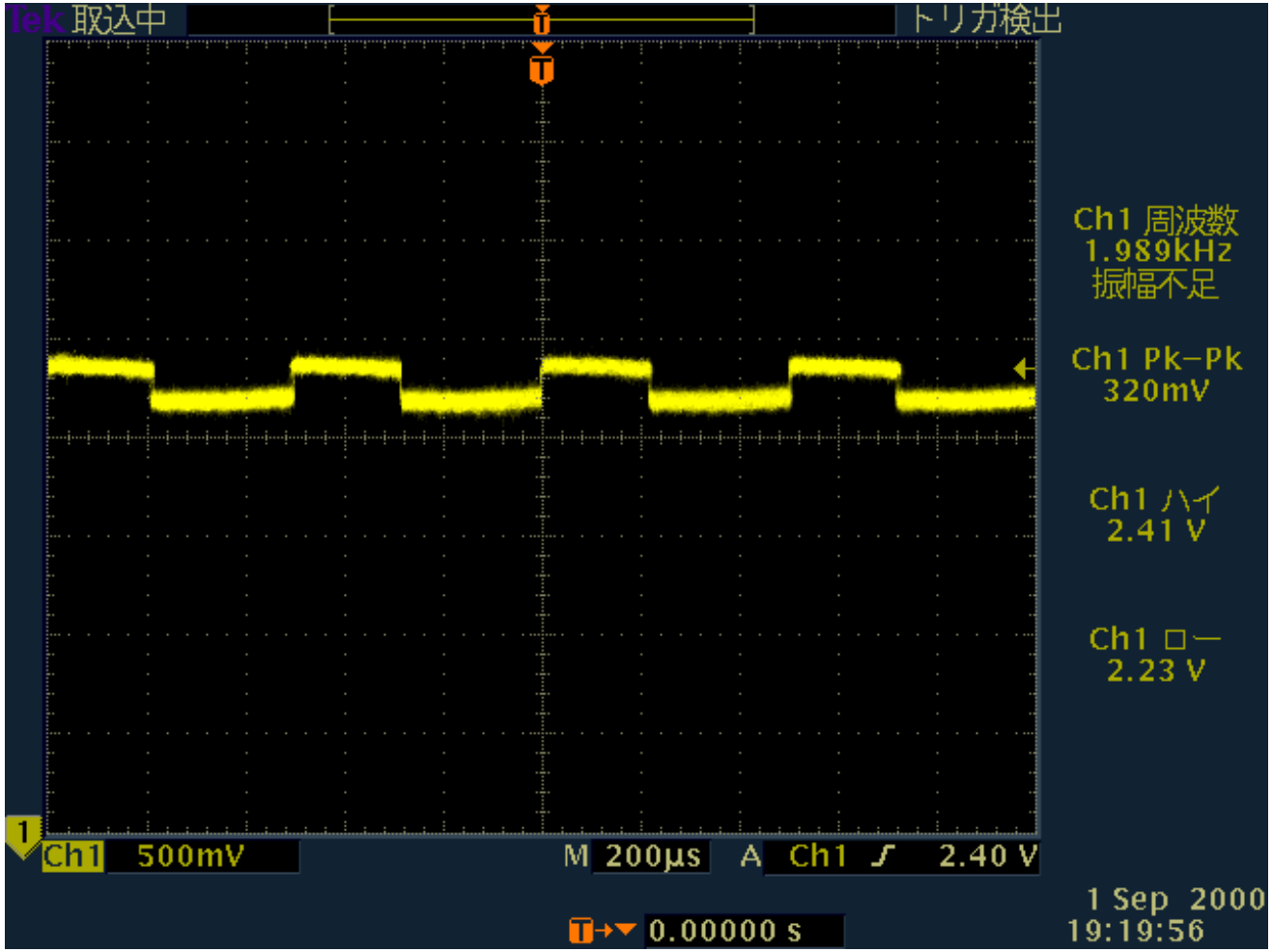
形式的な原因 : チップ製造を担当した会社の設計ガイドラインに、コアロジックの消費電力から電源ピン数等への制約がなかった (らしい)

負荷変動に電源が対応できない。

チップが動作時と非動作時で2倍くらい消費電力が違う。動作時間は$10\mu s$ から $1ms$ 程度。ボード1枚では $70A$ と $140A$ の間で変わる

- 普通のスイッチング電源では電流変化に応答できない
- 大容量の電解コンデンサをつけてもほとんど気休めにしかならない

電源波形


Tek 取込中
トリガ検出
Ch1 周波数
1.989kHz
振幅不足
Ch1 Pk-Pk
320mV
Ch1 ハイ
2.41 V
Ch1 ロー
2.23 V
Ch1 500mV
M 200µs A Ch1 2.40 V
0.00000 s
1 Sep 2000
19:19:56

負荷変動への対応

今から設計するなら

- スイッチング電源をフィードフォワード制御する
- 非動作時でも消費電力が下がらないように設計する

現在の対応

- ある程度応答の速い電源に変える
- 内部抵抗の小さい電解コンデンサを大量につける

Case Study からの教訓

結局は：

専用計算機も大きくなると大規模システムで起きるようなあらゆる問題が起きる。

というだけ。

あとは、、、

担当エンジニアのいうことは正しいとは限らない

↓

ボードやチップは専門家にまかせておけば出来るというものでもない

とはいえ、これはあまり専用計算機固有の問題というわけではない。

計算法 ── 時間領域

基本的には、 大規模な常微分方程式の初期値問題。

しかし、普通のライブラリにあるような方法 (高次ルンゲクッタ / 線形多段階法) は、そのままではほとんど 使いものにならない。

重力は特徴的なスケールを持たない引力

↓

現れる距離、時間スケールに制限がない。

2つのキーアルゴリズム

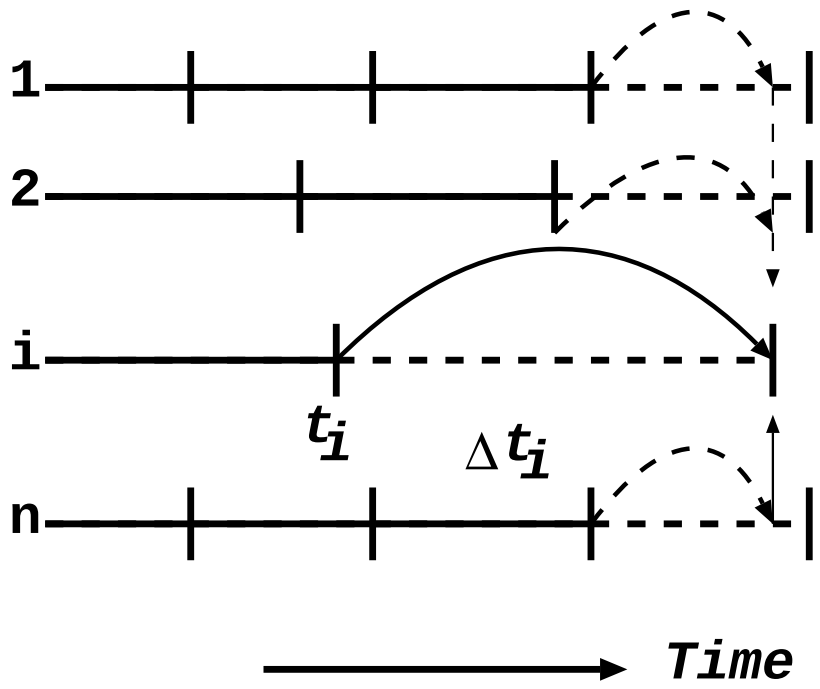
- 独立時間刻み
  - 近接散乱 $\longrightarrow$ 短いタイムステップ
  - 高密度中心核 $\longrightarrow$ 短いタイムステップ

  通常の適応時間刻みに比べた加速: $O(N^{1/3} \sim N)$.
- 近接散乱や連星系のための特別な扱い:
  - 独立時間刻みだけでは近接散乱の全てをカバーできない (桁落ちが問題になる)
  - 準安定な連星系 (および、準安定な階層的なサブシステム) : 摂動論的扱いが必要。

計算精度 と計算時間の両方に大きな効果（これらなしでは計算不可能）

独立時間刻み

(Aarseth 1963)



- 各粒子にそれぞれ時刻と時間刻みを与える
- 「イベント駆動」時間積分 — $t_i + \Delta t_i$ がもっとも小さい

粒子が積分される

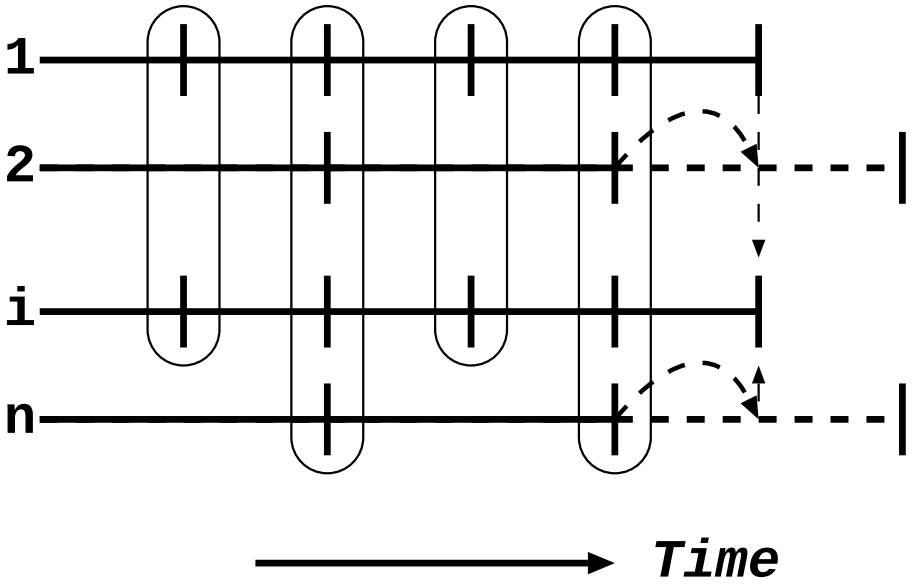
問題点

- 並列化が困難 (一度に一粒子)

ブロック時間刻み法

(McMillan 1986) ベクター / パラレル計算機のための改良

最初は Cyber205 のために導入された



- 時間刻みを 2^{-k} に制限する。
- $t_i + \Delta t_i$ が同じ (Δt_i は違ってもよい) 粒子は同時に積分される。

$O(N_c^{2/3})$ 個の粒子（N_c は高密度コアのなかの粒子数）を並列計算

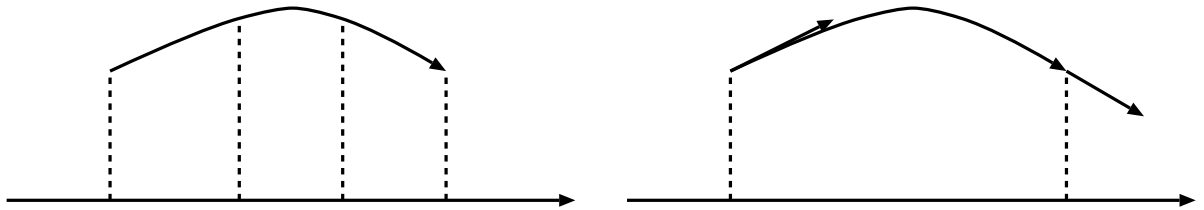
— そんなに大きな数ではない。

時間積分公式

粒子が同期していない → 可変刻み多段階法以外の選択肢は事実上ない。

次数： 4 次くらいが適当 (Makino 1990)。

- **Aarseth scheme (Aarseth 1963):** 可変刻みのアダムス法、 PEC モード、4 次、2 階の方程式用。
- **Hermite scheme (Makino 1990):** ラグランジュ補間 (ニュートン補間) の代わりに、加速度の一階時間導関数も使ってエルミート補間を構成。



連星等の特別な扱い

問題：

- 桁落ち $|x_i| >> |x_i - x_j|$
- (準) 安定な連星等：直接積分では計算時間がかかり過ぎる。

解決法 — 主に Aarseth とその共同研究者らによる

桁落ち → 相対座標 (と正則化) を使う

連星等 →「孤立した」系では積分をとばす。弱い摂動を受けている系では摂動論的扱い、、、

(もちろん、ほとんど安定だけれども孤立しているとはみなせないところにもっとも計算時間がかかる、、、)

今後の方向（？）

- シンプレクティック積分スキーム、時間対称型スキームとの組み合わせ
  - 潜在的な利点：エネルギー等の時間 1 次の誤差がなくなる
  - 困難：普通に考えると「未来」の情報が必要？
- 時間方向の並列化 (常微分方程式の初期値問題一般の研究方向)

応用例

というのか、最近どんな研究がされているのかを脈絡なく紹介。

- 球状星団
  - 星の進化、親銀河との相互作用
  - 蒸発のメカニズム
- 大規模構造・銀河団
  - “Universal” profile
- 惑星形成
  - 暴走成長

– 暴走成長の停止とその後

というような話をちょっとします。

球状星団

球状星団とはどんなものか：

- 銀河内にある（系外銀河にもたくさん見つかっている）
- 典型的な質量は $10^{5\sim6}$ 太陽質量。星の数ではもう数倍程度
- 典型的な大きさは 10 パーセクくらい
- 星の年齢：決定誤差の範囲内で宇宙年齢と一致（宇宙より古いとかそういう話もないではないが、、、）
- ほぼ球形で、回転もあまりない
- 銀河内の球状星団の分布も、ほぼ球形
- 年齢：緩和時間より大部長い

おおむねここまで述べてきたような熱力学的な進化をしてきているはず。
銀河とかと違って丸く、あまり特徴がない形をしている：緩和している？

但し：

- もちろん、壁があったりはしない。逆に、親銀河の作る tidal field で星が出ていきやすくなっている
- 星の質量はいろいろである
- 星は進化して、超新星になったり白色矮星になったりする

というあたりが違う。

球状星団の数値計算

理想化された単純な系でも、結局は数値計算しないとどうなるかはよくわからない
現実的なものがどうなるかの理解にはいっそう数値計算が重要

が、実はそういう研究は最近まであまりされていなかった。

理由：

- 理想化されたものがどうなるのかあまり良くわかってなかったため
- 計算機の能力不足 (上のことの理由でもある)

比較的初期の現実的な計算の例：

Chernoff and Weinberg (1990): 1次元フォッカー・プランク方程式に星の質量分布と進化、銀河の tidal field の効果を近似的にとり入れ、初期条件の広い範囲でどのような進化が起きるかをサーベイした。

数値計算の方法について

特に多体計算の方法については、後でもう少し詳しくやるとして、もうちょっと原理的にどういう方法があるかについて。

- 流体近似 : 形式的には、ボルツマン方程式のモーメント方程式を作ると流体の方程式が出てくる。

  が、平均自由行程は違うので、、、

- 軌道平均フォッカー・プランク近似 $(\mathbf{FP})$: 分布関数の時間発展を差分法で数値積分。衝突項は $\mathbf{2}$ 次までのモーメントでよいというフォッカー・プランク近似を使う。衝突項自体は、粒子の軌道にそって局所的に密度、速度分布から決まるものを積分 (軌道平均) して求める。

  分布関数の次元を落さないと使えない。 $f(E)$, 最近になっ

て $f(E, J)$. いずれも球対称のみ。

- モンテカルロ法：上の、分布関数の時間発展を、モンテカルロ積分する。

  $f(E, J)$（球対称）のみ

- N 体シミュレーション： 多体系の運動方程式を直接積分。

  原理的には何でもできる。計算時間が、、、

それぞれ、一長一短というか、万能とはいかない、、、

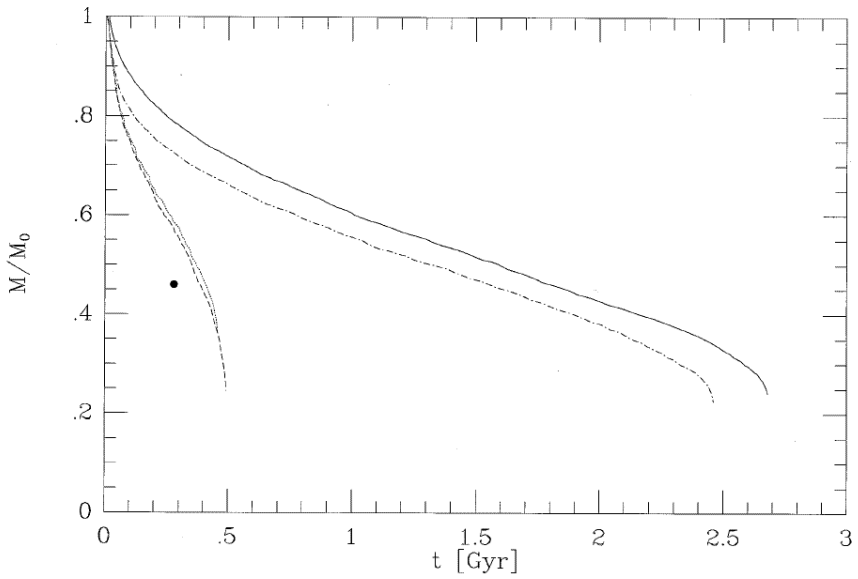
現実的な計算の例

Chernoff and Weinberg (1990): $f(E)$ を使う FP 計算。

Fukushige and Heggie (1995): N 体計算で「追試」。全然結果が違う、、、ただし、原因を明確にはしていない。

Takahashi and Portegies Zwart (1998, 2000): $f(E, J)$ を使う FP コードを N 体計算の結果を使ってキャリブレーション。多分、そこそこ正しい。

Chernoff and Weinberg とはやはり全然違った。



黒丸： CW

点線、破線： Takahashi の $f(E)$ の計算実線、鎖線： Takahashi の $f(E, J)$ の計算

違いの原因

基本的な違い：粒子が逃げる条件

$f(E)$ の計算:エネルギーでしか条件を付けられない。粒子がどこにあるのかは？

$f(E, J)$ の計算:実際に粒子がどこまで遠くにいくかがわかる。

但し、実際の銀河の効果はさらに複雑 (時間変化、 $\mathbf{tidal\ heating}$) これらはまだよく調べられていない。

大規模構造、銀河団

宇宙全体の構造：
大雑把には、宇宙の質量のほとんどは銀河が担っている。
銀河の宇宙内での分布は一様からは程遠い (宇宙の大規模構造)。
また、銀河団のような集団を作っている銀河というのも結構多い (ちょっとこれは定義の問題でもあるが、、、)
これらの構造はどのようにしてできたのか？

現在の標準的な理解：ビッグバンからの宇宙膨張の間に、もともとあったゆらぎが重力不安定 (これは熱力学的なものではなくて力学的な不安定) によって成長してできた。

これは、基本的には球状星団とは違って熱力学的な進化として

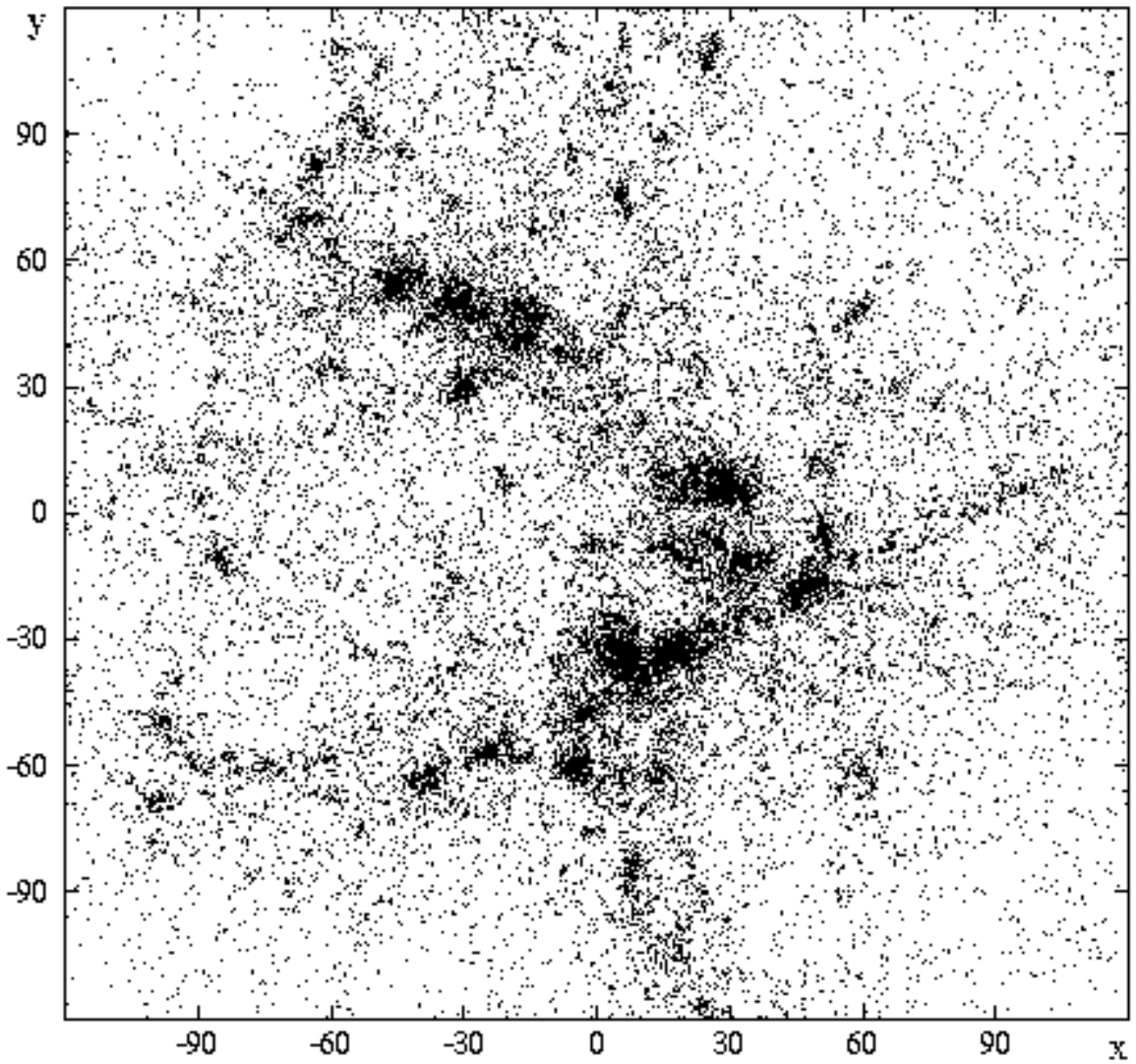
理解できるわけではない。

理由：
宇宙年齢にくらべて緩和時間が圧倒的に長いため。

が、そういうことを全然考えなくてもいいかというと、そうでもない

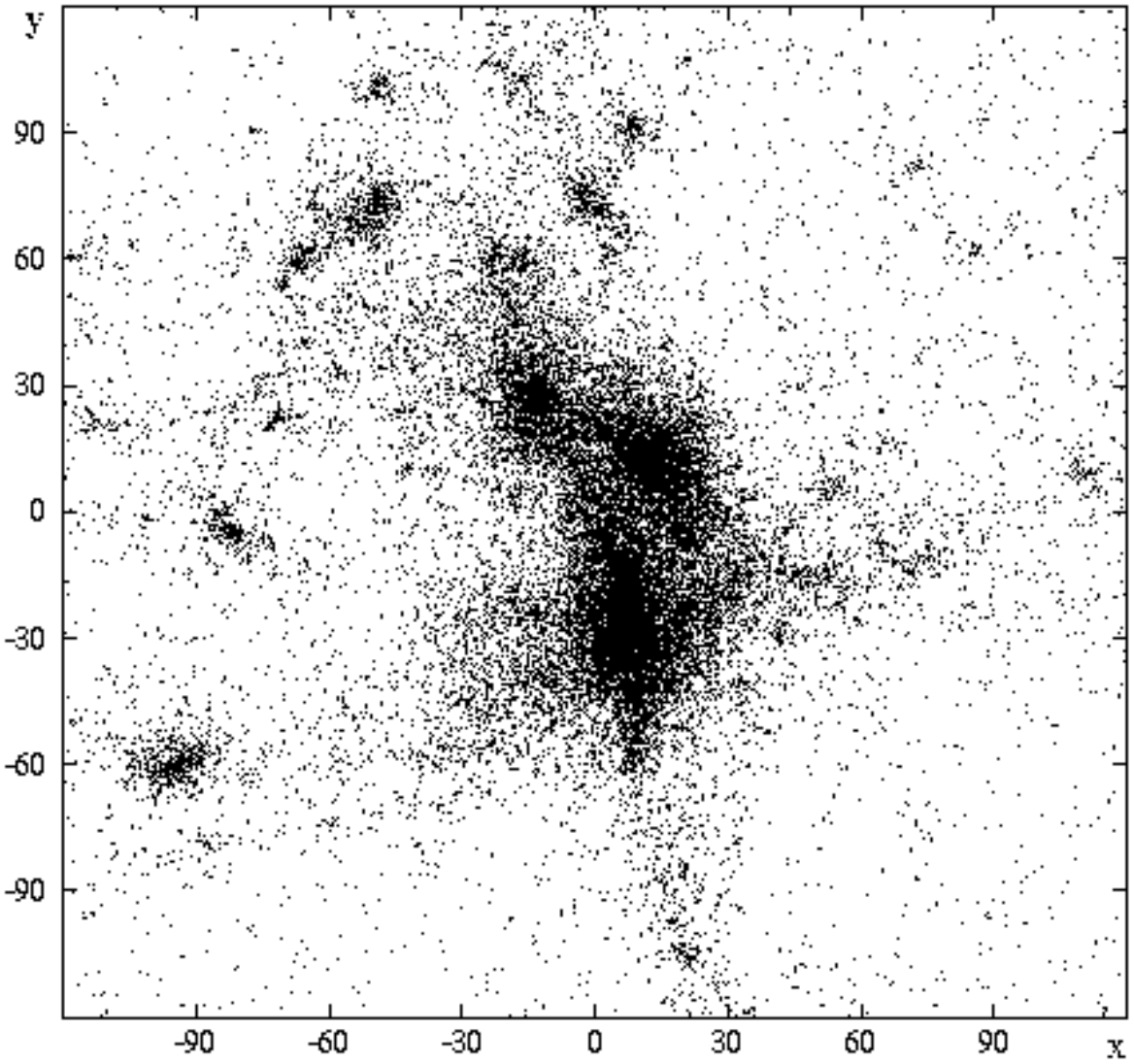
- 銀河団や大規模構造を考える時には、銀河が構成要素：実効的な粒子数は小さい
- 現実では効かなくても、数値計算では熱力学的な進化のために結果が違ってきているかもしれない

計算例（絵）1



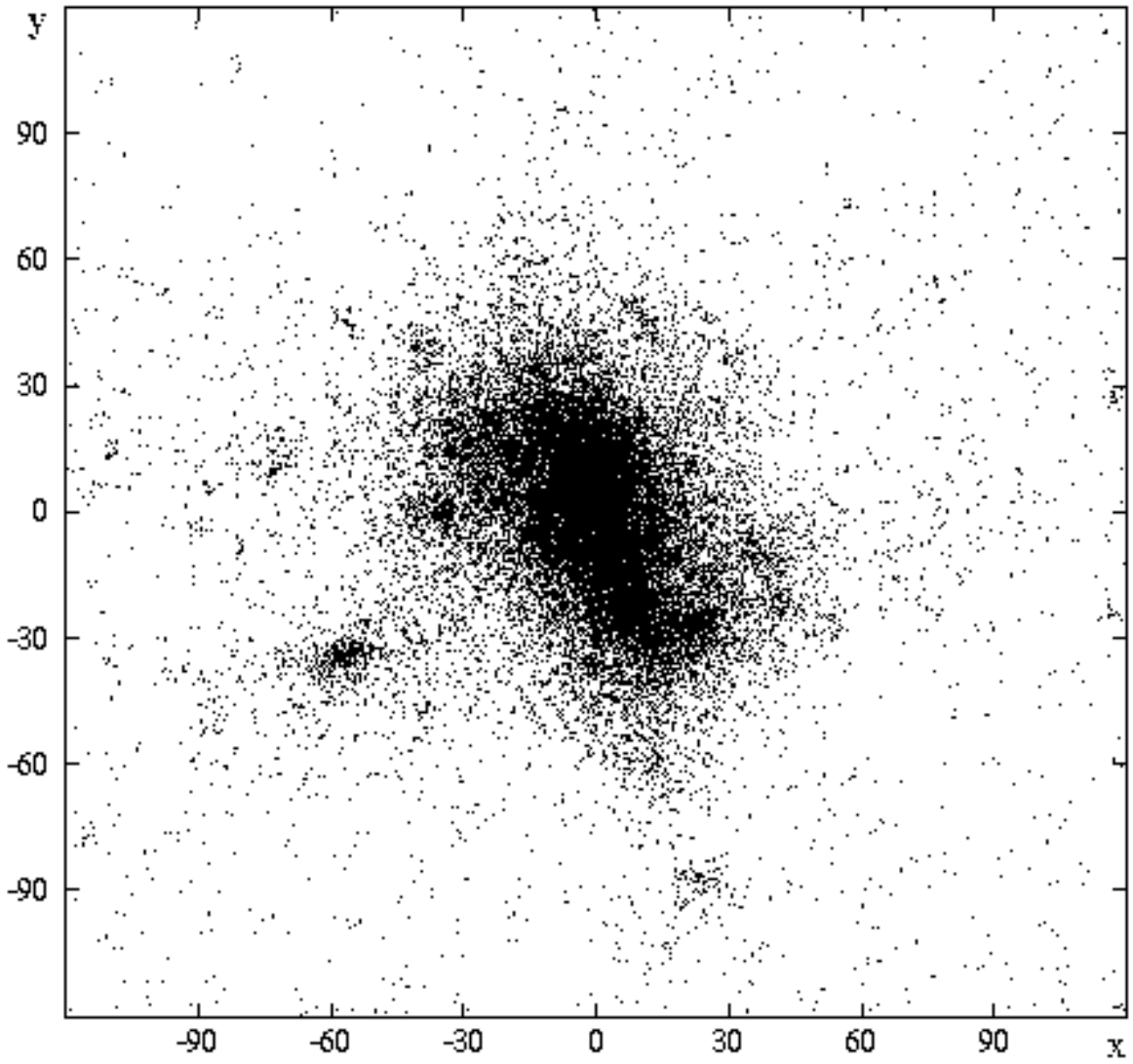
Fukushige and Makino 1996 から。銀河形成の計算例 $T = 0.5 (z = 8.7)$

計算例（絵）2



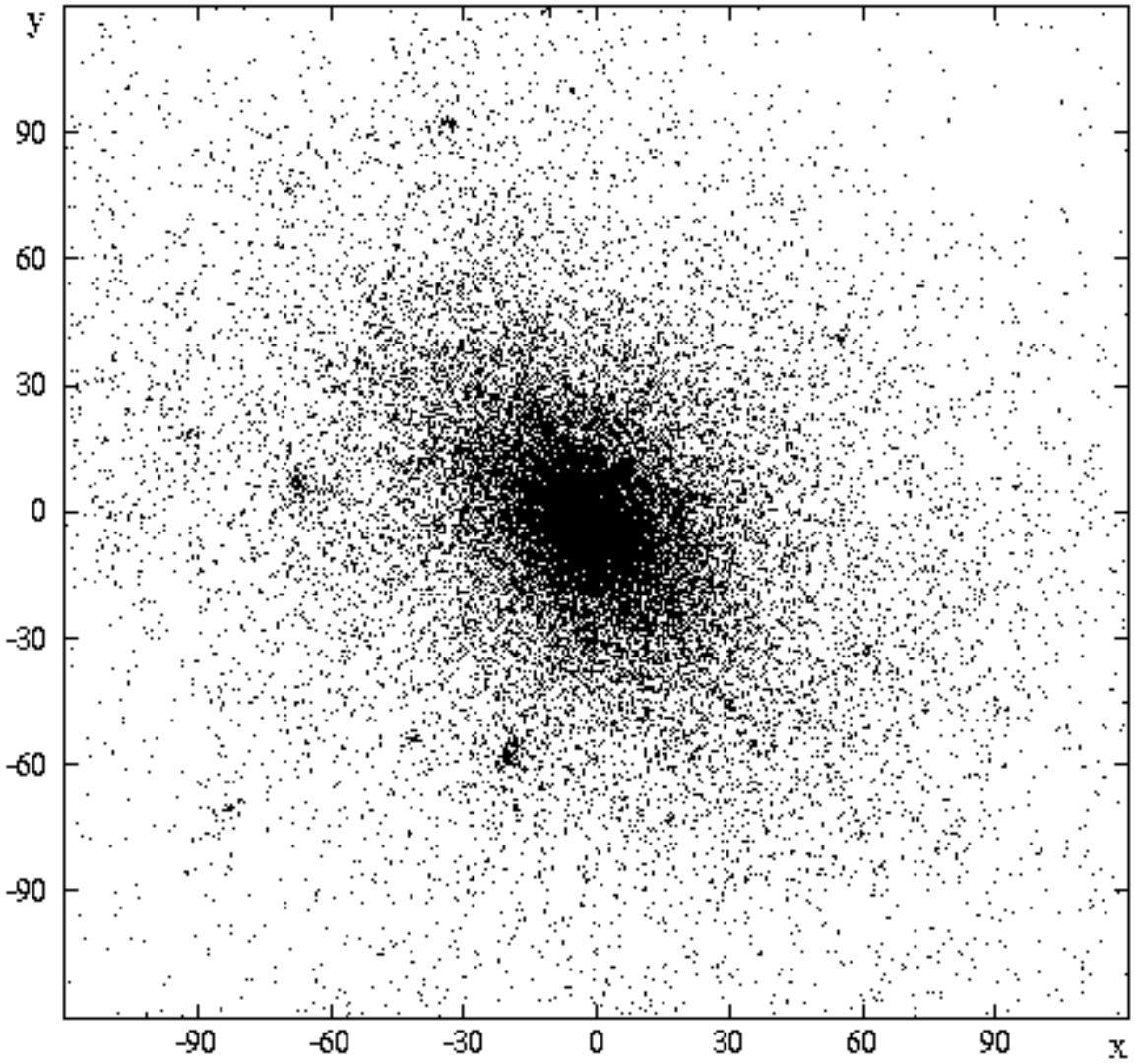
$$T = 1$$

計算例（絵）3



$T = 1.375$

計算例（絵）4



$$T = 3.125 (z = 1.8)$$

数値計算の問題の例

Navarro, Frenk and White (1996)

様々な宇宙モデルで与えられる初期条件から、銀河団の形成の数値シミュレーションをして、結果としてできる銀河団の密度構造が、ほとんど初期条件によらないで

$$\rho = \rho_0 \frac{1}{(r/r_0)(1 + r/r_0)^2} \tag{3}$$

で良く近似できるということを「発見」した。

使った粒子数：2万くらい。普通の計算機で、ある程度の数シミュレーションを流そうとおもうとこれくらいが当時の限界。

Fukushige and Makino (1997): 専用計算機を使ってほぼ同じ計算を 77万粒子でやった。

$\longrightarrow$ 答が違う!

Moore et al. (1998): 並列計算機で 300 万粒子

$\longrightarrow$ **Fukushige and Makino と傾向は同じ。**

$$\rho = \rho_0 \frac{1}{(r/r_0)^{1.5}(1 + r/r_0)^{1.5}} \tag{4}$$

と、中心で密度が高い。

Navarro, Frenk and White が間違った理由 : 粒子数の不足のため、中心では熱力学的な緩和がおきていた。

惑星形成

惑星形成の標準モデル (京都モデル) — 1980 年代の理解

原始太陽系星雲の中でダストが赤道面内に沈澱

↓

重力不安定でダスト円盤が分裂、微惑星に ($10^{18}g$ くらい)

↓

互いの重力で微惑星同士が合体を繰り返す

↓

そのうちに惑星になる

標準モデルの問題点：惑星が出来るのに時間がかかり過ぎる。
外側程時間がかかって、木星あたりで太陽系の年齢を超える。
が、木星はある

ので、なにかがおかしいはず。

何故時間がかかるか？

微惑星が合体・成長するにつれて、成長が遅くなる

成長する

↓

ガスの抵抗が相対的に弱くなる

微惑星間の相互作用の効き方が大きくなる（緩和時間が短くなる）

↓

ランダム速度が大きくなって、実効的な衝突断面積が小さくなる

↓

成長が遅くなる

1980 年代末に、Wetherill 他が「これはちょっと違うのではないか」といい出した。

上の議論：「微惑星はみなおなじように成長する」と暗黙に仮定されていた。

まあ、「大きくなると成長が遅くなる」んだから、大きさが揃ってくると思うのは一見自然。

Wetherill の議論

世の中には dynamical friction というものがある

↓

重い粒子と軽い粒子が混在していれば、重い方のランダム速度が小さくなる

↓

重い粒子のほうが成長が速くなって、暴走的に成長する

彼らは、速度分散の質量依存性をモデル化したモンテカルロ計算で上のようなことが起きることを示した。

これだと、木星でもなんでもあっというまにできてしまうことになる。

問題：本当にそんなにうまくいくのか？

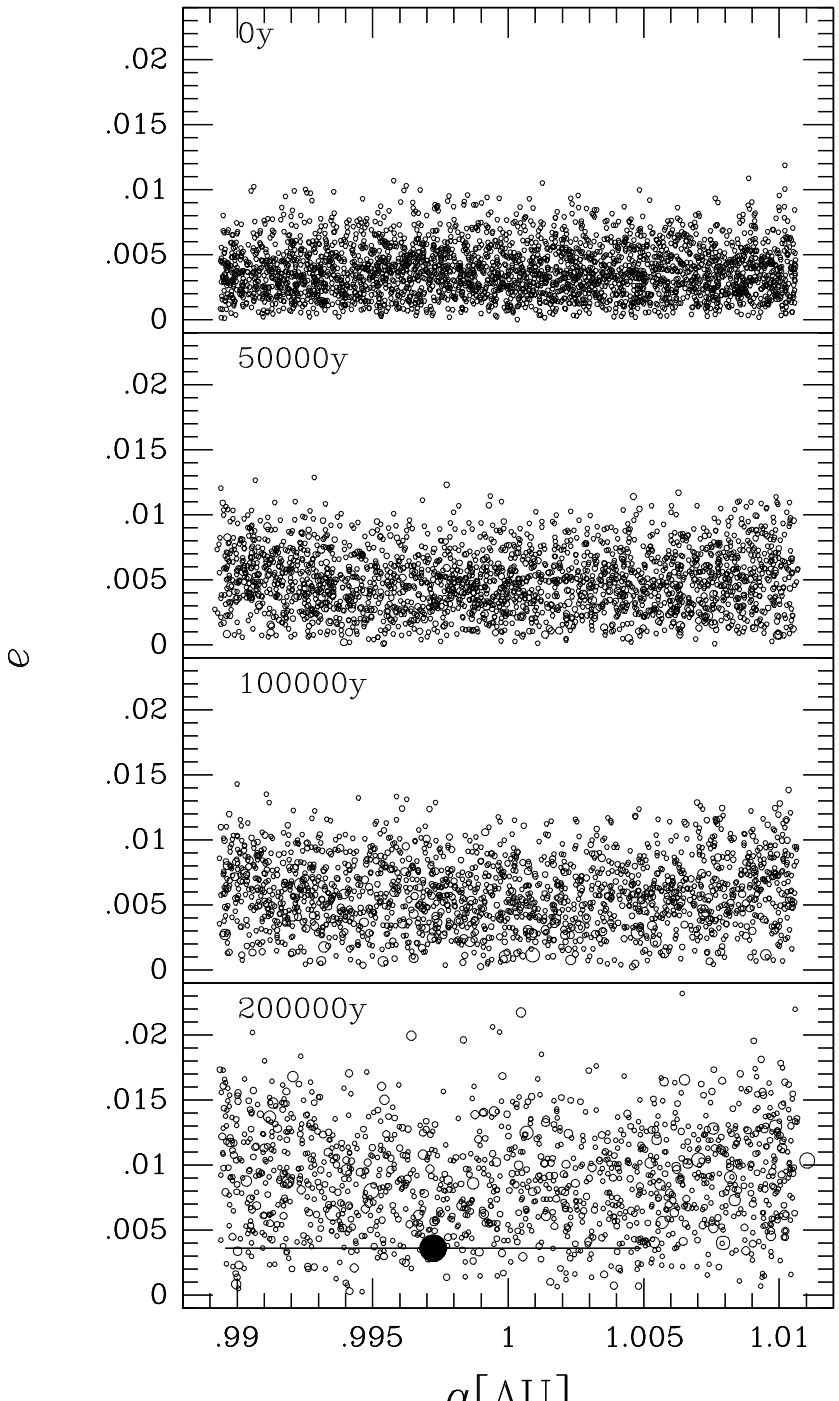
多体シミュレーションでの結果

1990年代・井田、小久保らによる N 体計算での研究：

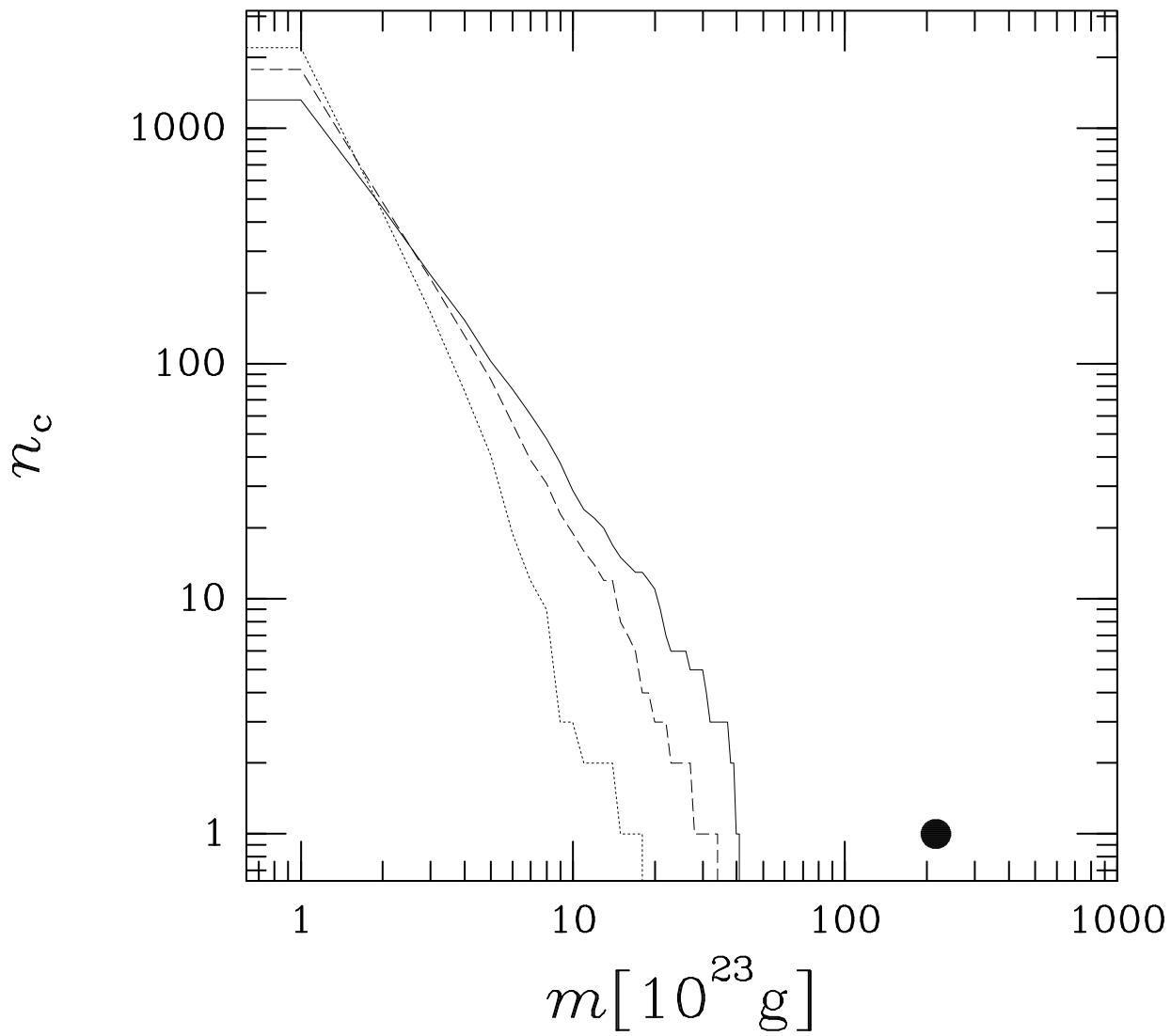
- Dynamical friction は確かに効く
- 但し、暴走成長したものがある程度重くなると、それが回りを加熱して結局成長速度が下がる
- 従って、ある程度重いものが少しづつ離れてできる、「寡占的成長」が実現する

- 木星コアくらいまでは成長できる
- 地球は、 $1/10$ くらいのところで止まる
  そこからどうやってできたかはまだ良くわかっていない

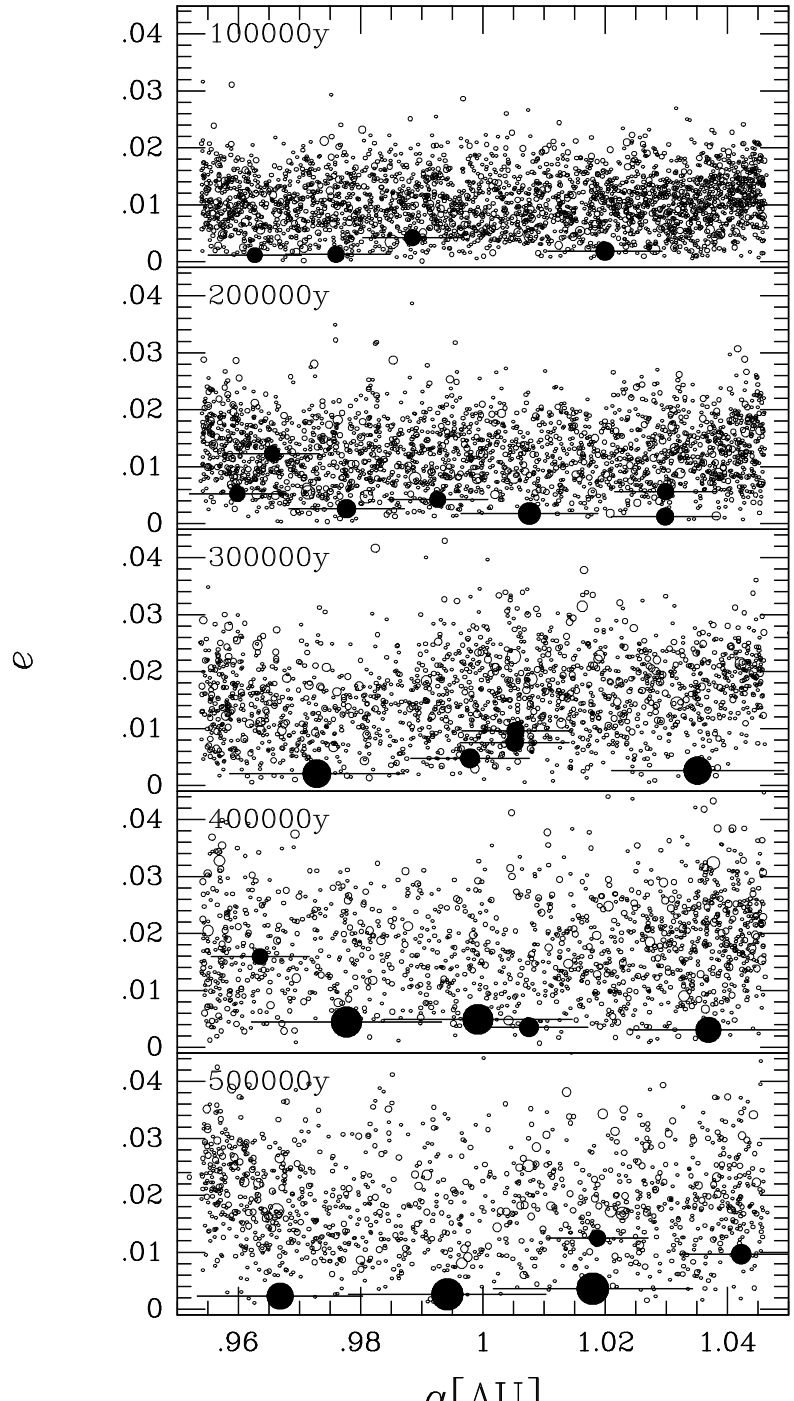
暴走成長の計算例


0y
50000y
100000y
200000y
e
.02
.015
.01
.005
0
.99
.995
1
1.005
1.01
a[AU]

微惑星の質量分布の進化



寡占的成長

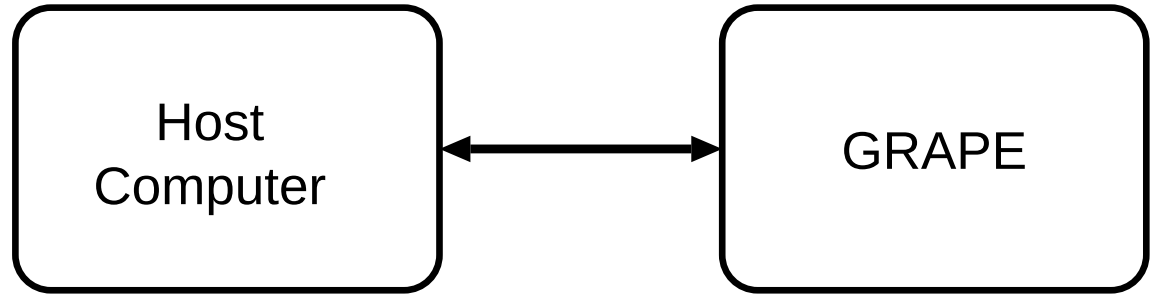

100000y
200000y
300000y
400000y
500000y
e
.04
.03
.02
.01
0
.96
.98
1
1.02
1.04
a[AU]

計算機を速くする

- 寝て待っていれば速くなる (10 年で 100 倍) まあ、少しは働かないといけない
  - 並列化
  - PC クラスタを組み立てたり、、、
- 自分で作る (GRAPE)

GRAPE の基本的な考え

(GRAPE, GRAvity PipE)



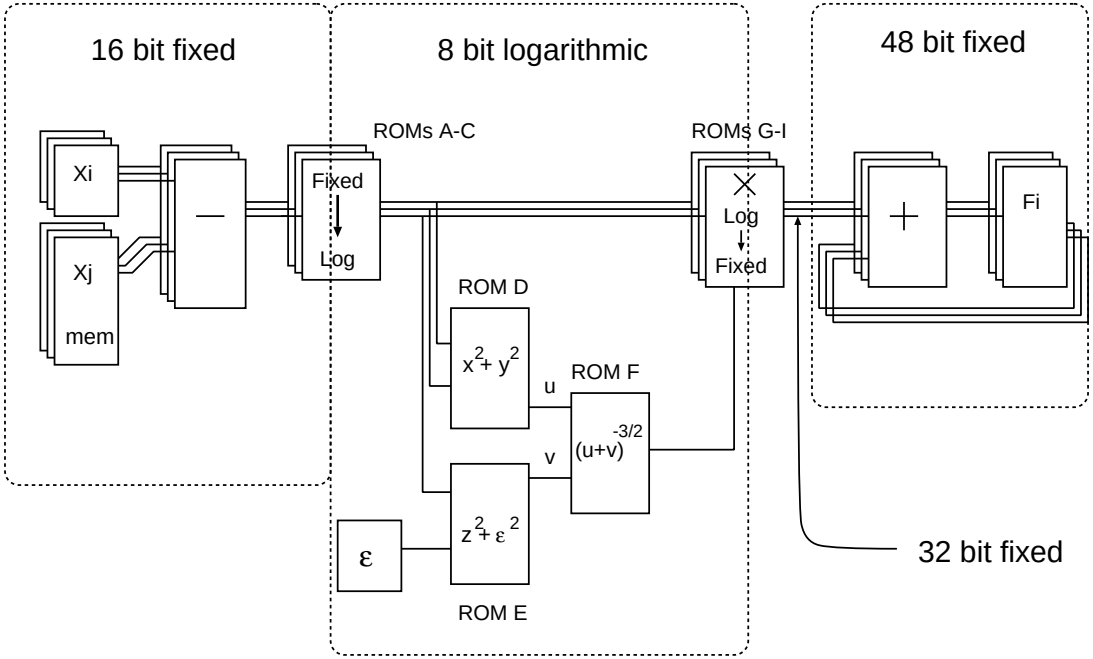
- 専用ハードウェア
  - ハードウェア化した論理 → ソフトウェア不要
  - 相互作用計算専用のパイプライン → 高い性能
- 汎用ホスト計算機
  - 高レベル言語が使える (Fortran, C, C++...)

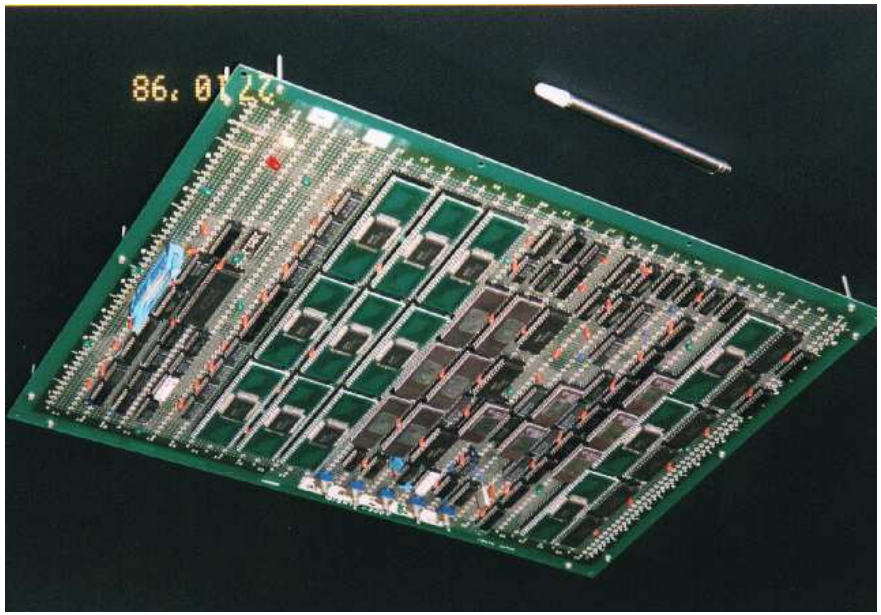
・既存のプログラムが多少の手直しで使える

・ツリーコード、独立時間刻みなども使える

GRAPE-1 パイプライン

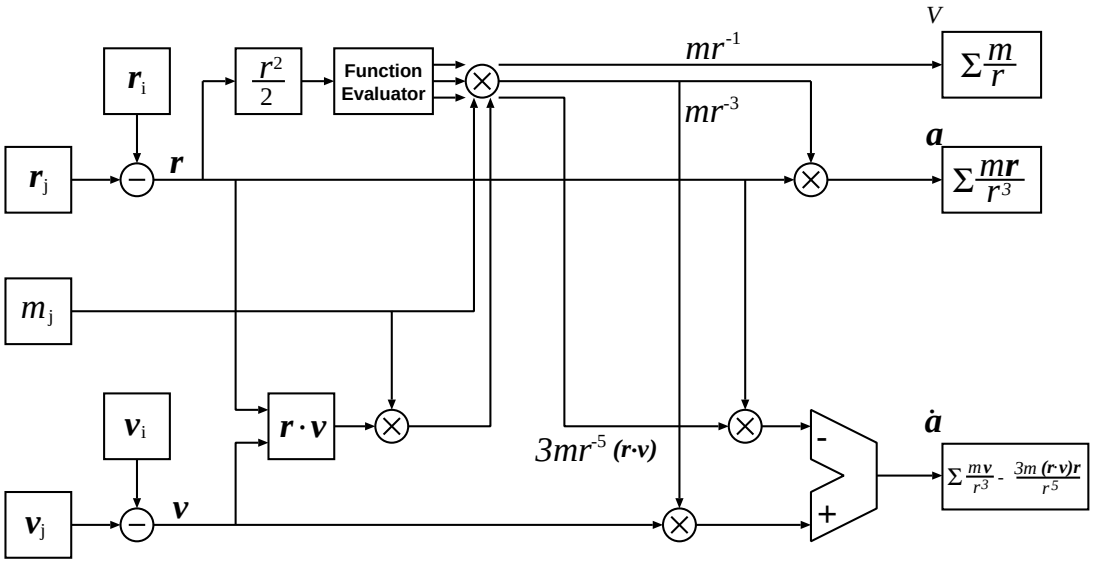
(1989, 240 Mflops)

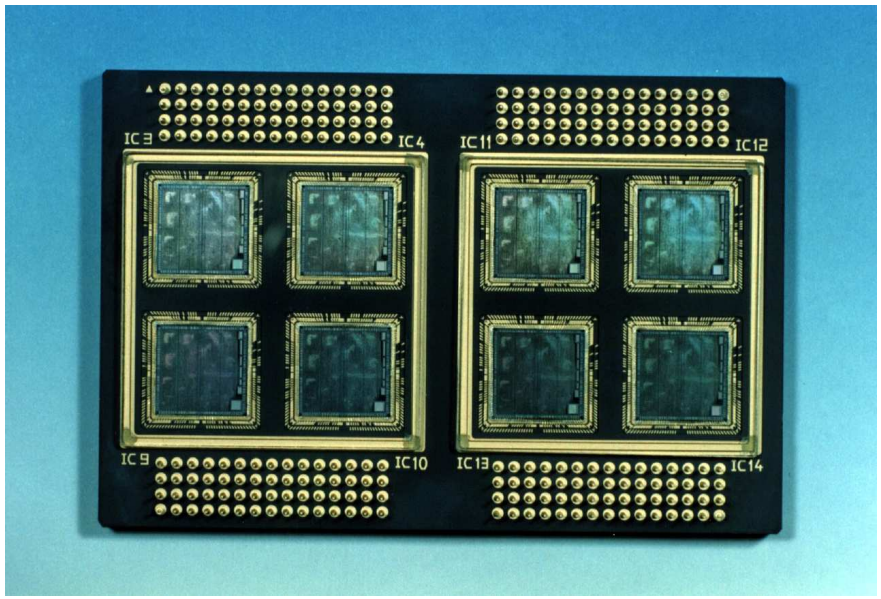



86. 01. 22.

GRAPE-4 パイプライン

(1993, 600 Mflops/chip)

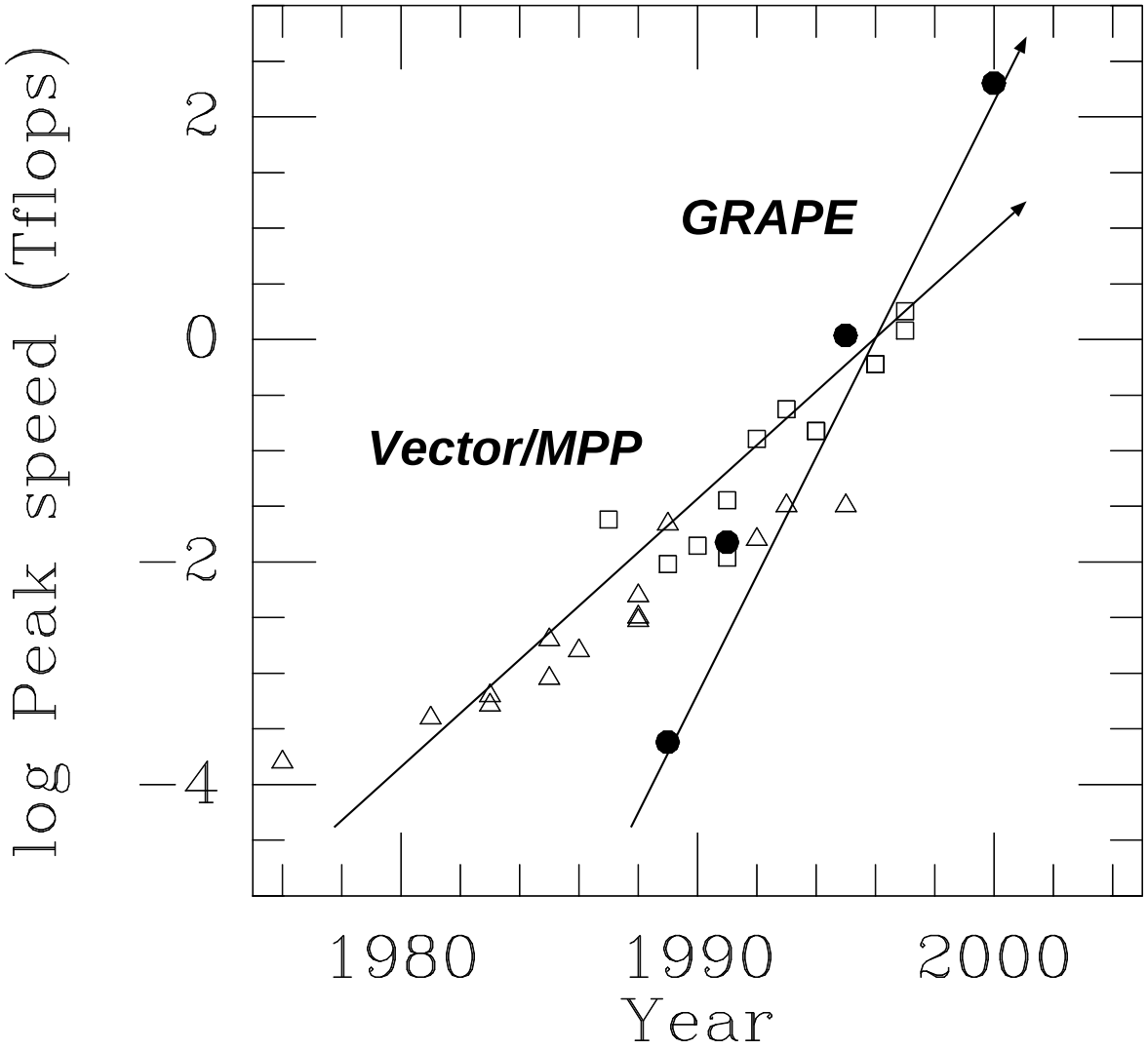



IC3
IC4
IC11
IC12
IC9
IC10
IC13
IC14

GRAPE-4

(1995, 1.08 Tflops)

汎用計算機との速度比較



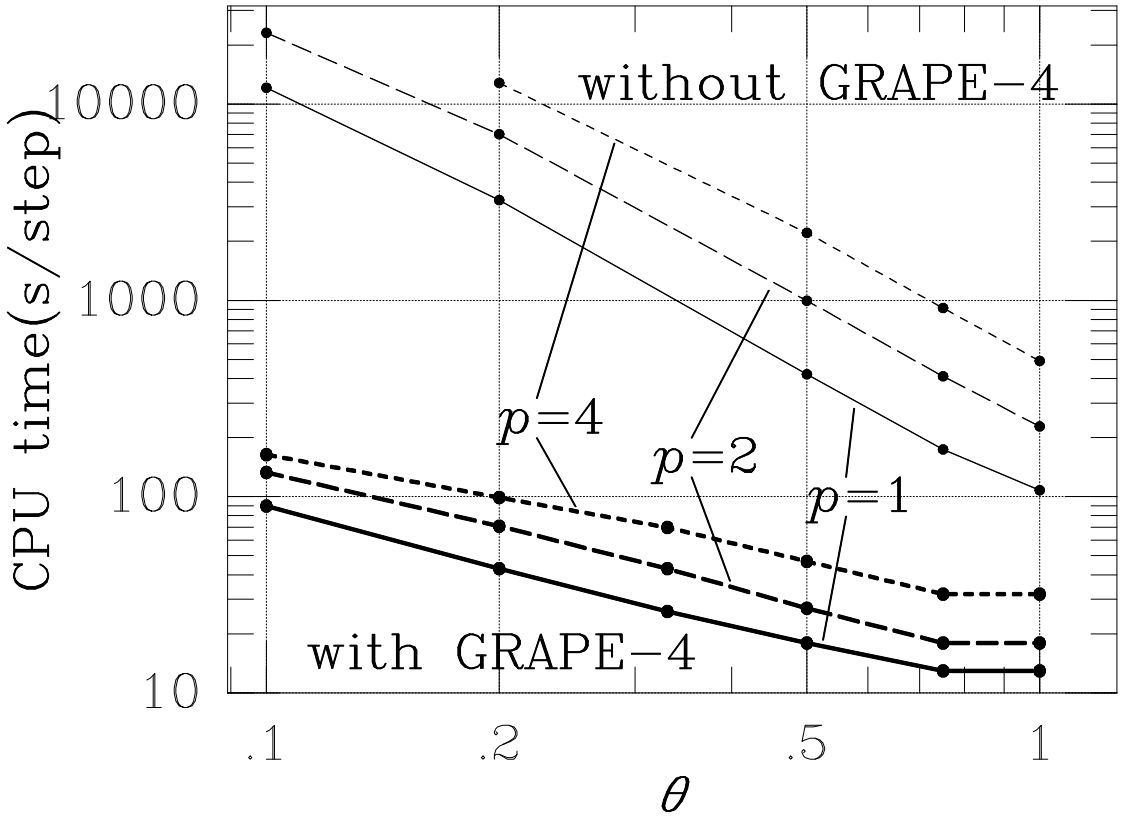
ツリー・FMM と GRAPE

通常のツリー、 FMM: 粒子間相互作用 (モノポール) の計算にしか使えない

疑似粒子法： セルからの重力の計算にも使える

疑似粒子法は GRAPE で効率的に加速出来る (ツリーでも FMM でも)

実際の計算速度 (ツリーの場合)



GRAPE なし： DEC Alpha EV5 300 MHz

GRAPE ： GRAPE-4

粒子数： 262144

GRAPE を使うことで、　10-100 倍の高速化が可能。

GRAPE-6 計画

目標：

1. 重力計算用にGRAPE-4 から100倍以上性能を向上させた、サブペタフロップス計算機を実現し、粒子系向け専用計算機の可能性を実証する
2. FPGA を使った粒子系向け多用途計算機を構築する
3. 上の2つを組み合わせて、高速で応用範囲の広いシステムを構築する

一応、2001年度で完成の予定（「ほぼ」予定通り進行中）

並列化の 2 方法

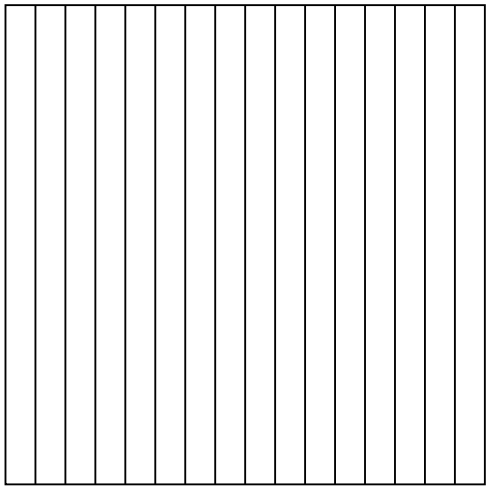
- 各ノードが全粒子のコピーを持って、重力計算、時間積分は分担する
- 各ノードが自分の分担の分だけを持つ。重力計算のためには他のノードから粒子を貰ってくる。

どちらにしても、各ノードが全粒子の情報を使って力を計算する。

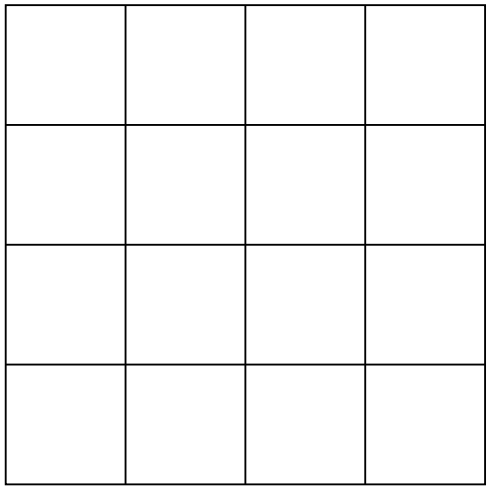
ノード一つの通信バンド幅でスケーラビリティが制限される。

もうすこし違う考え方はないか？

近接相互作用の時（２次元の例）



1-D decomposition



2-D decomposition

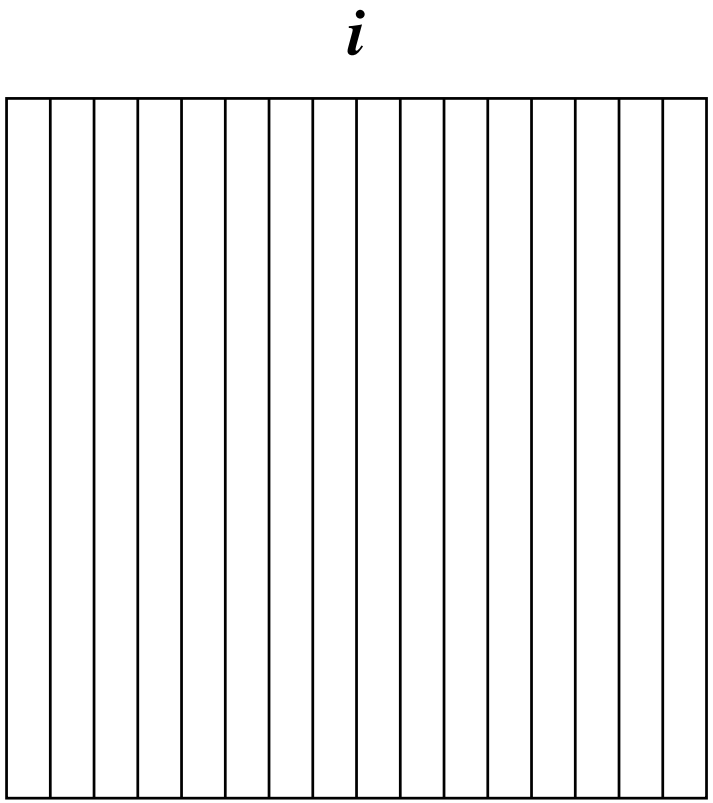
１次元空間分割：スケーラブルでない

ノード数 ∝ １次元方向のメッシュ数

２次元空間分割：スケーラブル

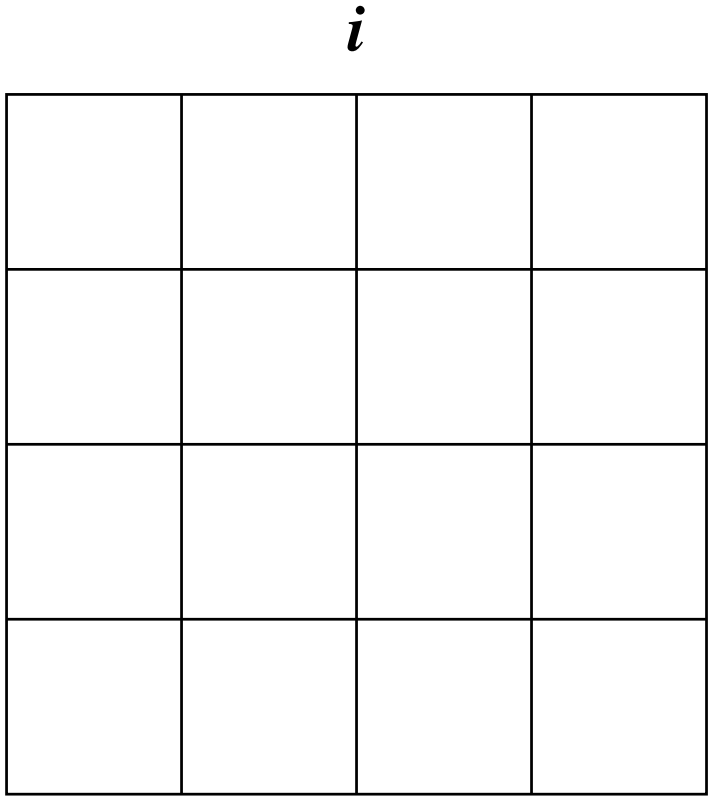
ノード数 ∝ 全体のメッシュ数

遠距離相互作用 ── 相互作用行列を考える



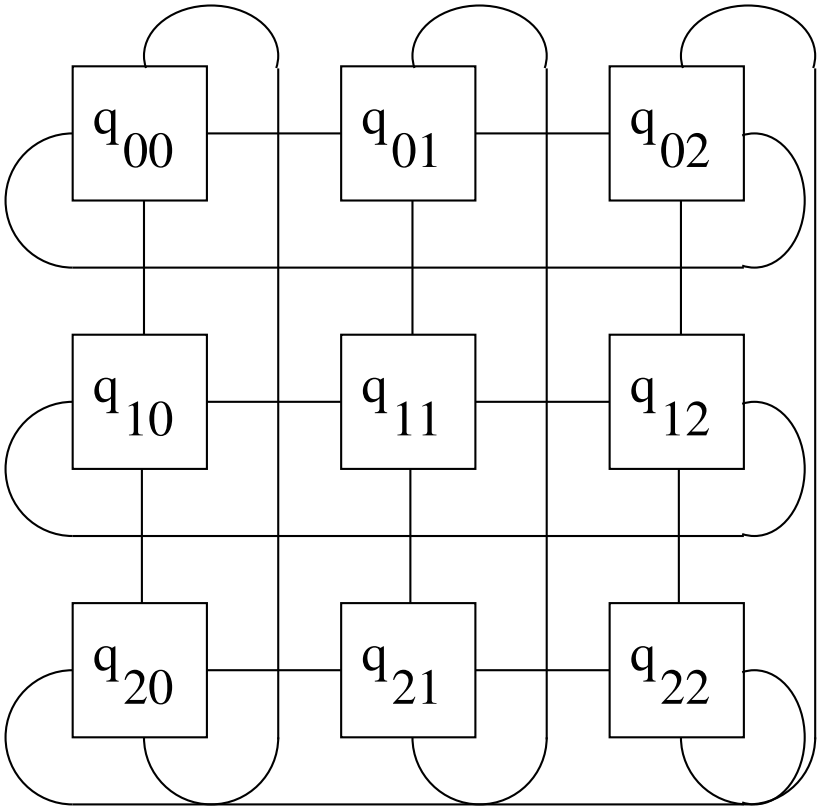
- 相互作用の行列を見ると、普通のアルゴリズムは1次元分割になっている。
- これがスケーラブルでない理由
- 2次元分割は可能か？

相互作用の2次元分割



- 形式的には左のような絵を書ける。
- 実際にはこれはなにを意味しているのか？

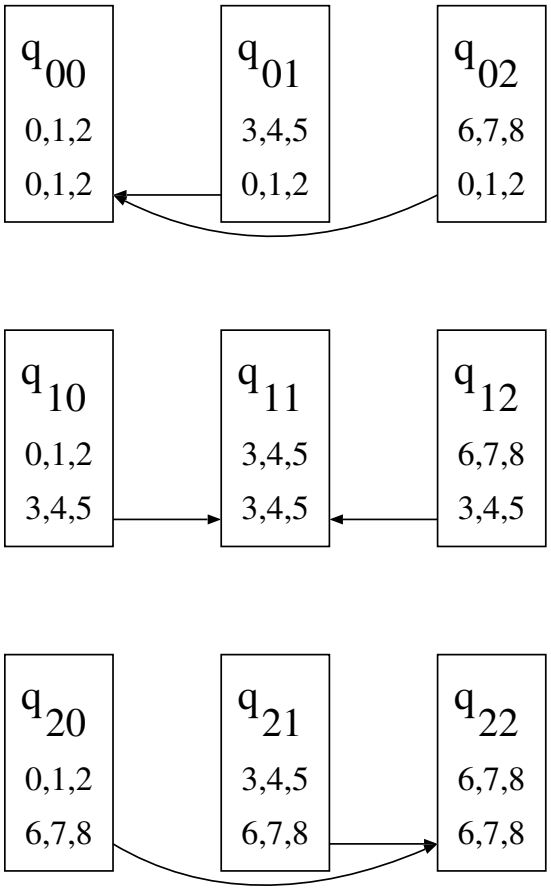
ハードウェアとしては、、、



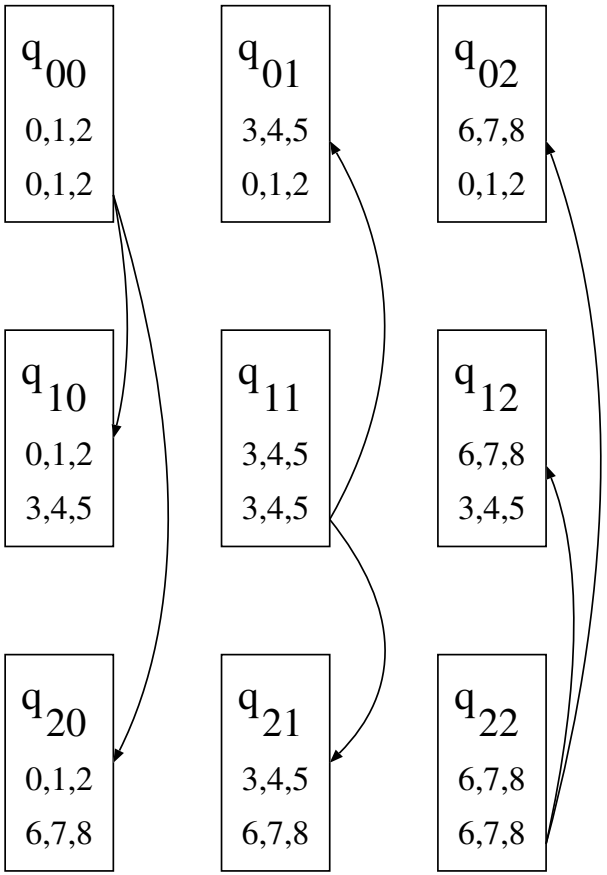
- 分割した相互作用行列にプロセッサを対応させる。
- トポロジ：2D トーラスなり「ハイパークロスバー」なり、、、

計算手順

Step 1



Step 2



横に総和、縦に放送

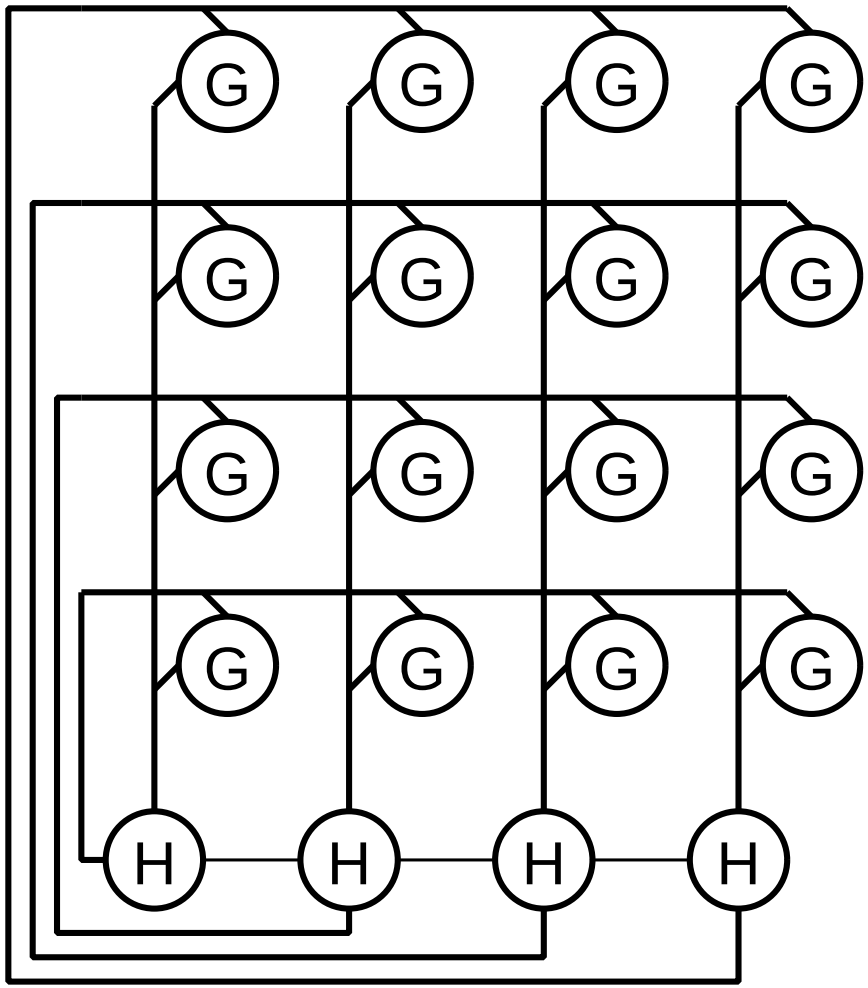
使えるプロセッサ数

・効率が $50\ \%$ になるプロセッサ数

$$p_{\mathrm{half,2Dbcast}} \sim (NC_f/2C_c)^2 \tag{5}$$

(通信のスタートアップが無視できる場合) ここから先でも速度は向上する (プロセッサ数の平方根に比例)

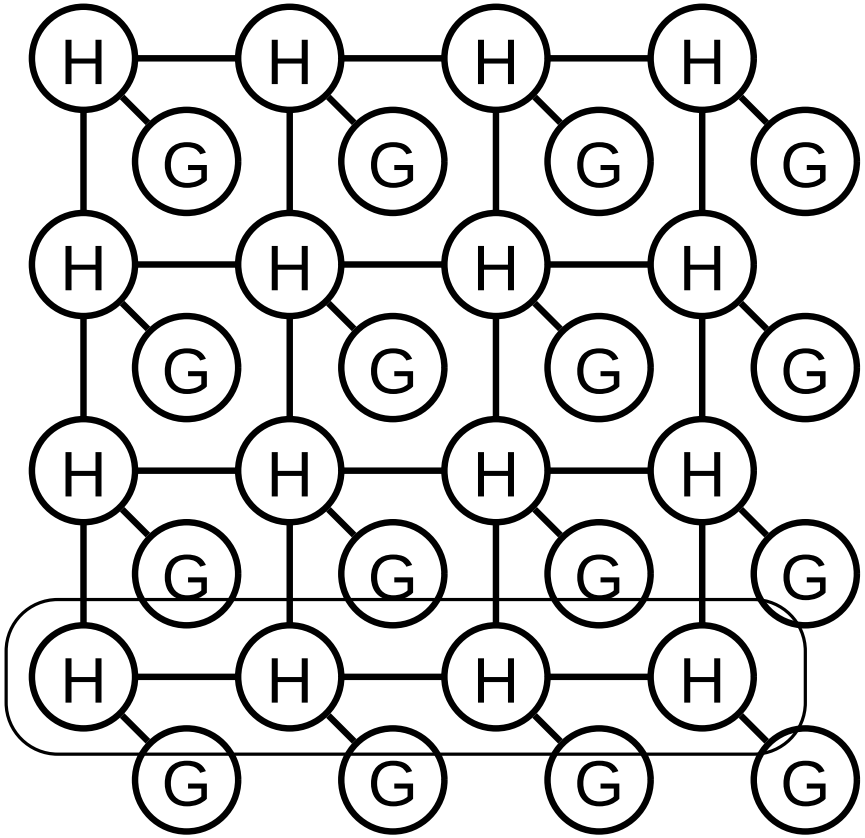
GRAPE-6 の2次元ネットワーク



GRAPE 側で2次元アルゴリズムを実現

実際のネットワークはもうちょっとややこしい

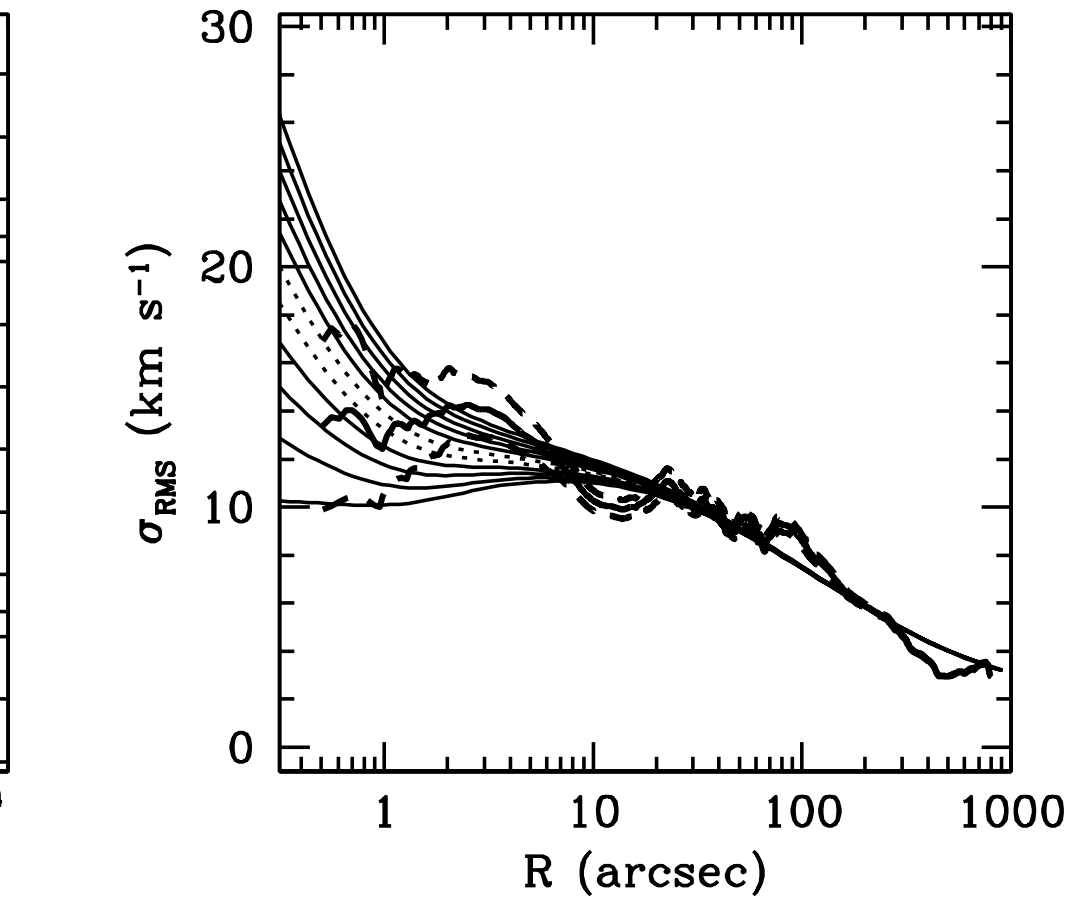
別の解



今 (2002 年以降) ならこれ。1996 年の時点では採用は現実的ではなかった (と思う)

- 速いネットワークは高価だった (GbE が本当に安くなったのは 2003 年)
- 速いホスト計算機も高価だった (Alpha, HP-PA は x86 の数倍の速度があった)

「現実的な」 M/L



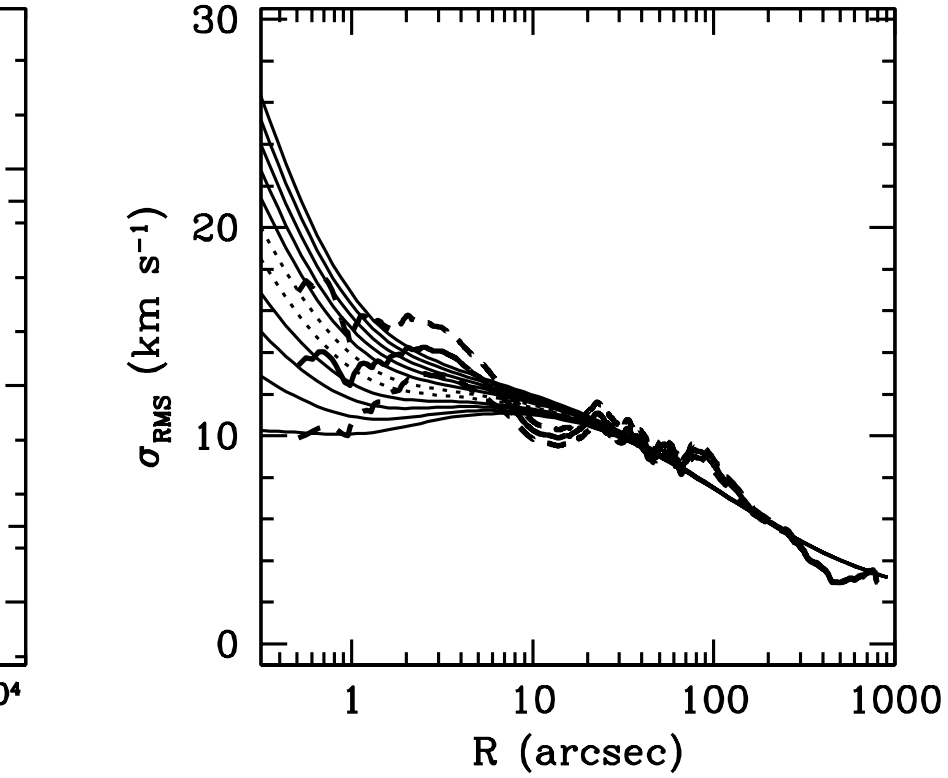
Dull *et al.* **(1997)** の **multi-mass Fokker-Planck** 計算による M/L プロファイル

速度プロファイルはあんまり変わらない

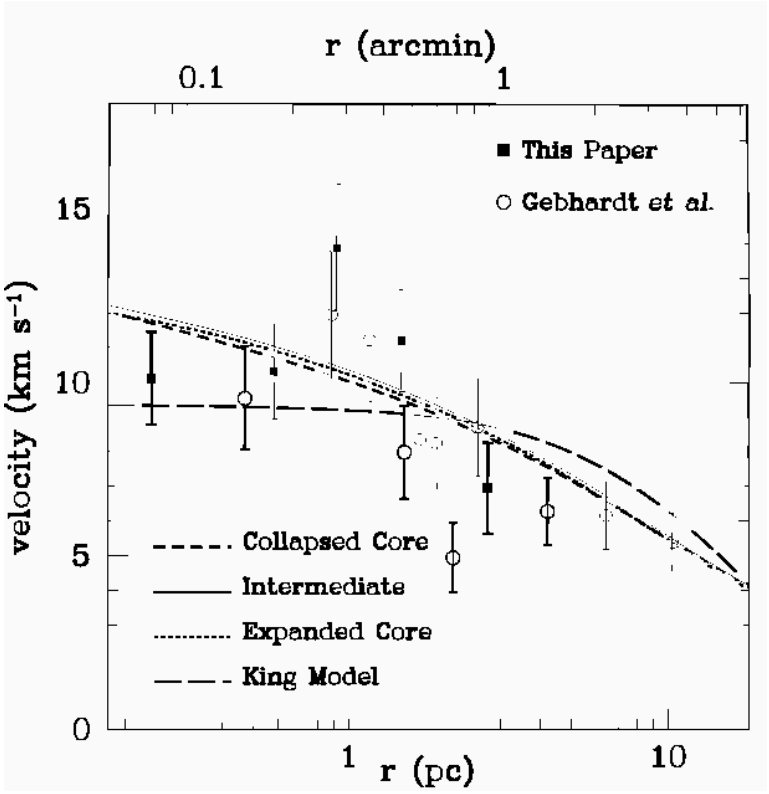
依然大きなブラックホールが必要

何故結果が違うのか？

Gerssen



Dull



「同じ」 M/L で、 Dull 他 と Gerssen 他 は全く違った速度プロファイルを出した。

Addendum :

Gerssen et al. astro-ph/0210158 (Oct 8, 2002):

After the completion of our paper, the authors of the D97 paper discovered an unfortunate error in their Figures 9 and 12. The labeling along the abscissa of these figures is incorrect due to a coding error in supermongo plotting routines (H. Cohn and B. Murphy, private communication, 2002). The units along the top axis should have read 'arcmin' instead of 'pc', and the labeling in arcmin along the bottom axis is incorrect. The net result is that the radial scale of these figures in the D97 paper is too compressed by a factor 2.82.

つまり、、、

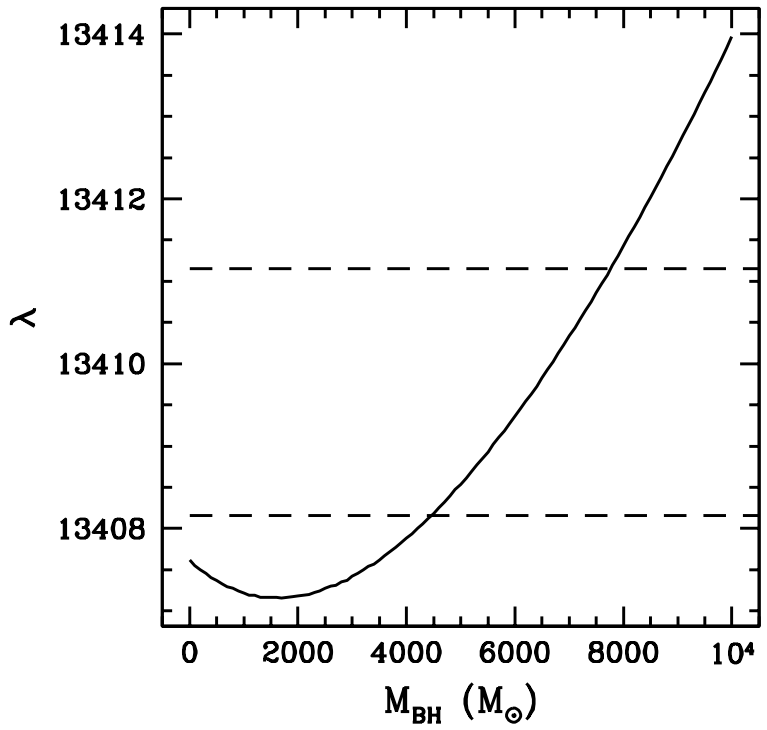
彼らがモデルに使った理論計算の論文のグラフの横軸の単位が間違っていた

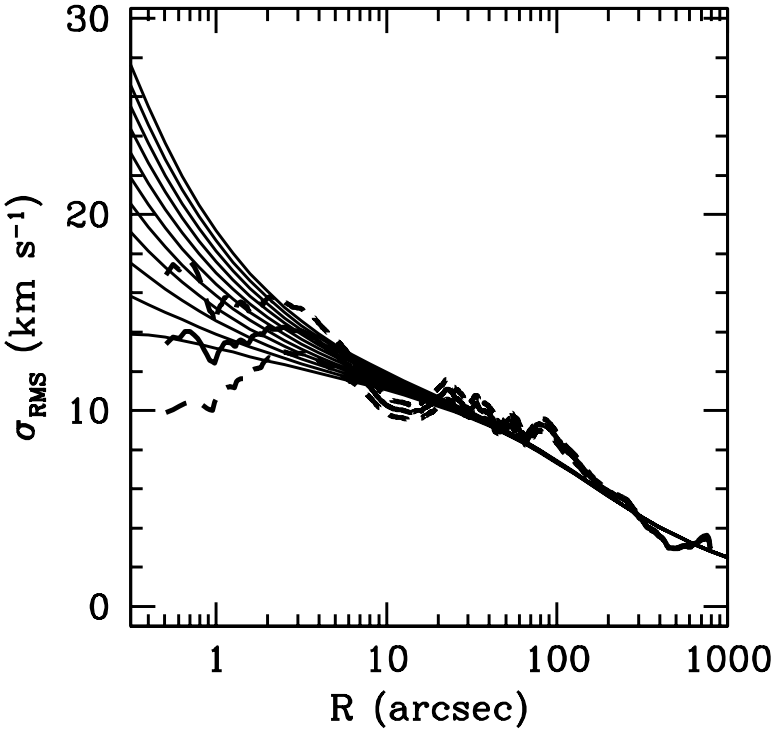
という主張。

落ち着いて計算チェックしていれば、なんか間違ってたのがわかったはず、、、

New plot

New plot (astro-ph/0210158):





ブラックホール消滅、あるいは *"1-σ で存在"*.

正則化

正則化の基本的な考えは、座標と時間を適当に変換して数値解の性質をよくしようというもの

2 次元の例。 2 次元座標 $\mathbf{r} = (x, y)$ を、

$$x = u_1^2 - u_2^2 \tag{6}$$

$$y = 2u_1 u_2 \tag{7}$$

という関係を満たす新しい変数 $\mathbf{u} = (u_1, u_2)$ に変換し、さらに独立変数を時間 t から $rds = dt$ という変数変換で得られる変数 s に置き換える。

正則化された方程式

すると2体問題の運動方程式

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\mathbf{r}}{r^3} \tag{8}$$

が単振動の方程式

$$\frac{d^2\mathbf{u}}{ds^2} = \frac{E}{2}\mathbf{u} \tag{9}$$

に変換される。ここでEはシステムの全エネルギー。

これは、軌道角運動量が 0 の直線解も扱える。

解析解でつなぐ必要はない。

歴史

Levi Civita (1905) が2次元の変換を提案

3次元で摂動がある場合に拡張: Kustaanheimo and Stiefel (1965)

2次元の場合の本質: 位置ベクトルを複素数とみなしてその平方根をとる

Kustaanheimo and Stiefel はスピノルを使って4次元に軌道を拡張

ハミルトンの4元数を使っても書けるということが数年前に示された。