

ν GCに関するN体計算の解説と展望

矢作 日出樹

Introduction

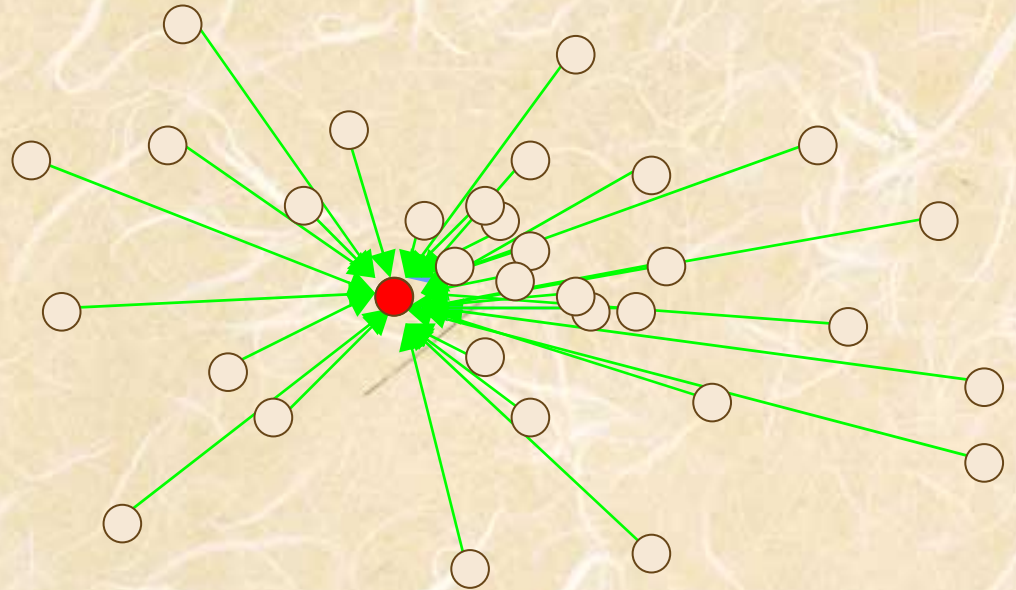
- N-body Methods
- Adaptive Mesh Refinement (AMR)
- Optimization of the AMR code
 - Why should I optimize the code?
 - How was the code changed?

N-body Methods

- Open Boundary Problems (Isolated system, monolithic collapse)
 - Direct Sum Method
 - Tree method
- Periodic Boundary Problems (statistics of objects)
 - Particle-Mesh (PM) Method
 - Particle-Particle Particle-Mesh (P³M) Method
 - Tree-PM Method
 - Adaptive Mesh Refinement (AMR)

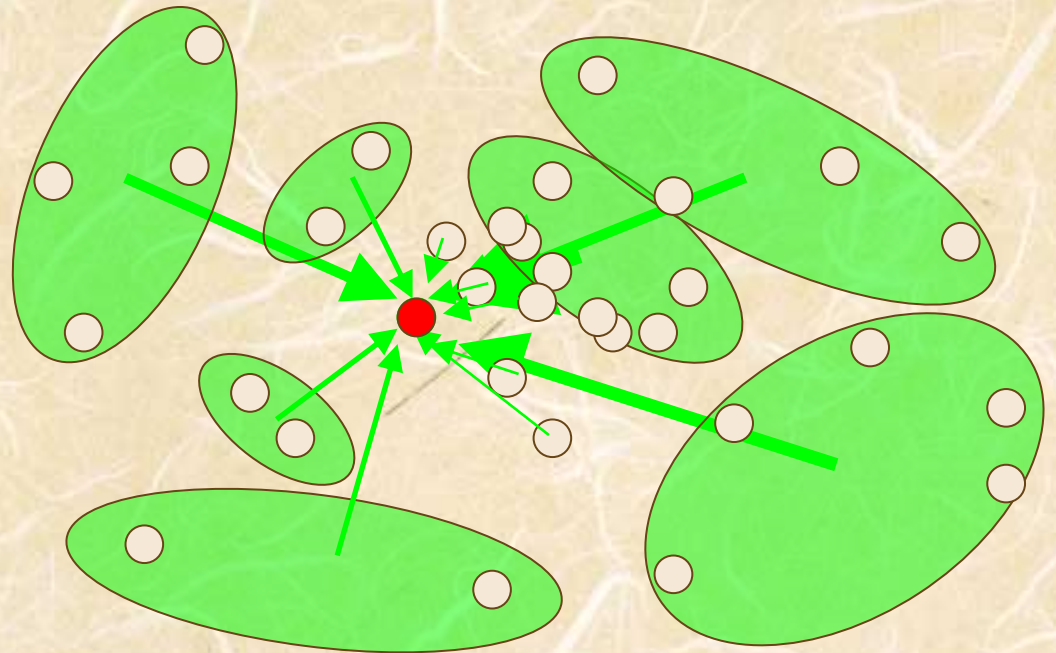
N-body Methods

- Direct sum method
 - Calculate force from each particle
 - $O(N^2)$



N-body Methods

- Tree method
 - Distant particles are bundled up
 - $O(N \log N)$



N-body Methods

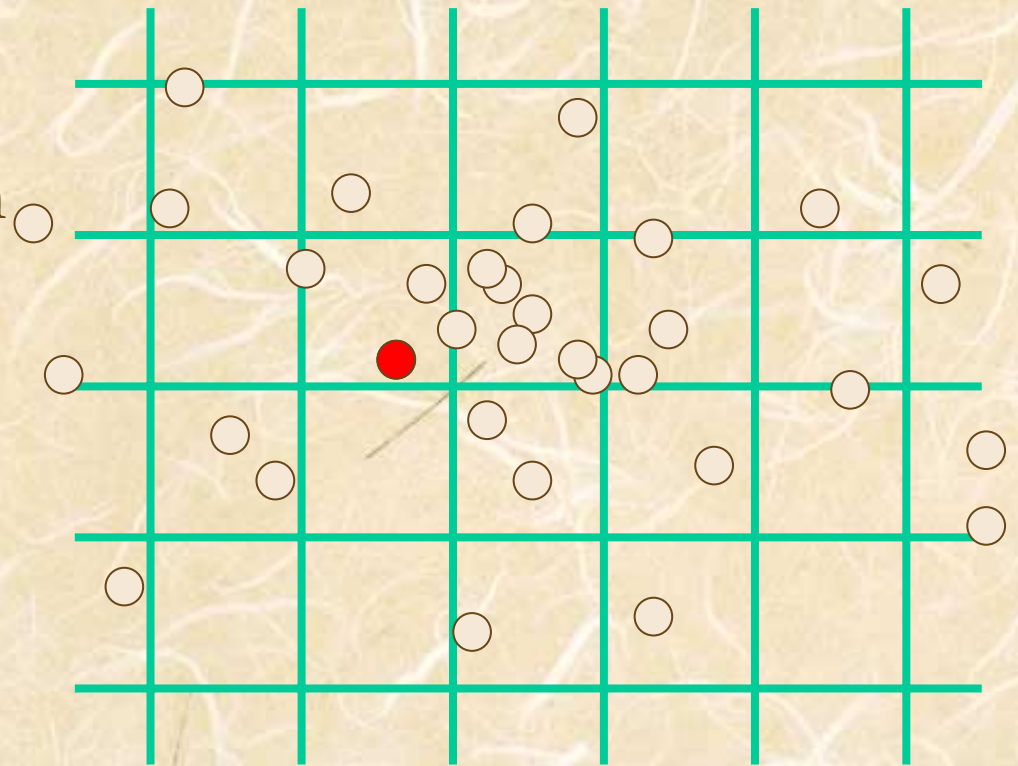
- Particle-mesh (PM) method

Mass assignment

Poisson solver

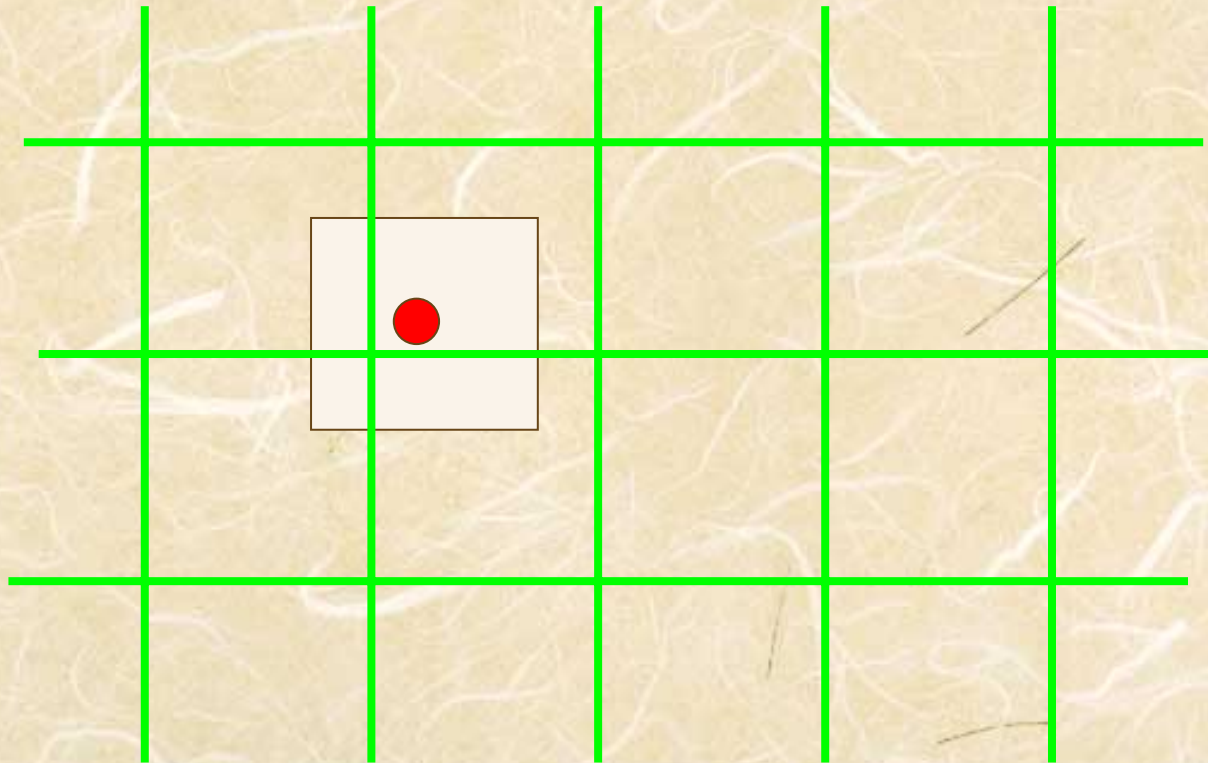
Force interpolation

$O(N \log N)$



N-body Methods

- Particle-Mesh Method
 - Mass assignment (Cloud-in-Cell Scheme)



N-body Methods

- Particle-Mesh Method

- Mass assignment (Cloud-in-Cell Scheme)

```
for (i=0; i<np; i++){
    ix=(int) pos[ip][0]; dx=pos[ip][0] - (double)ix;
    iy=(int) pos[ip][1]; dy=pos[ip][1] - (double)iy;
    iz=(int) pos[ip][2]; dz=pos[ip][2] - (double)iz;
    rho[ix ][iy ][iz ] += mass[ip] * (1-dx) * (1-dy) * (1-dz);
    rho[ix ][iy ][iz+1] += mass[ip] * (1-dx) * (1-dy) * ( dz);
    rho[ix ][iy+1][iz ] += mass[ip] * (1-dx) * ( dy) * (1-dz);
    rho[ix ][iy+1][iz+1] += mass[ip] * (1-dx) * ( dy) * ( dz);
    rho[ix+1][iy ][iz ] += mass[ip] * ( dx) * (1-dy) * (1-dz);
    rho[ix+1][iy ][iz+1] += mass[ip] * ( dx) * (1-dy) * ( dz);
    rho[ix+1][iy+1][iz ] += mass[ip] * ( dx) * ( dy) * (1-dz);
    rho[ix+1][iy+1][iz+1] += mass[ip] * ( dx) * ( dy) * ( dz);
}
```


N-body Methods

- Particle-Mesh Method

- Poisson Solver (Convolution method)

- $\Delta \phi = 4 \pi G \rho$

- $k^2 \phi_k = 4 \pi G \rho_k$

- $\rho \xrightarrow{\text{FFT}} \rho_k \xrightarrow{\text{(conv.)}} \phi_k \xrightarrow{\text{FFT}} \phi$

```
fft3d_real_forward (n, rho);
```

```
for (kx=0; kx<n; kx++){
```

```
    for (ky=0; ky<n; ky++){
```

```
        for (kz=0; kz<n; kz++){
```

```
            phi[kx][ky][kz]=4.0 * PI * G * rho[kx][ky][kz] / (kx*kx+ky*ky+kz*kz);
```

```
        }
```

```
    }
```

```
}
```

```
fft3d_real_backward(n, phi);
```

N-body Methods

- Particle-Mesh Method

- Force interpolation (Cloud-in-Cell Scheme)

```
for (i=0; i<np; i++){
    ix=(int) pos[ip][0]; dx=pos[ip][0] - (double)ix;
    iy=(int) pos[ip][1]; dy=pos[ip][1] - (double)iy;
    iz=(int) pos[ip][2]; dz=pos[ip][2] - (double)iz;
    acc[ip] =
        grv[ix ][iy ][iz ] * (1-dx) * (1-dy) * (1-dz)+
        grv[ix ][iy ][iz+1] * (1-dx) * (1-dy) * ( dz)+
        grv[ix ][iy+1][iz ] * (1-dx) * ( dy) * (1-dz)+
        grv[ix ][iy+1][iz+1] * (1-dx) * ( dy) * ( dz)+
        grv[ix+1][iy ][iz ] * ( dx) * (1-dy) * (1-dz)+
        grv[ix+1][iy ][iz+1] * ( dx) * (1-dy) * ( dz)+
        grv[ix+1][iy+1][iz ] * ( dx) * ( dy) * (1-dz)+
        grv[ix+1][iy+1][iz+1] * ( dx) * ( dy) * ( dz);
}
```


N-body Methods

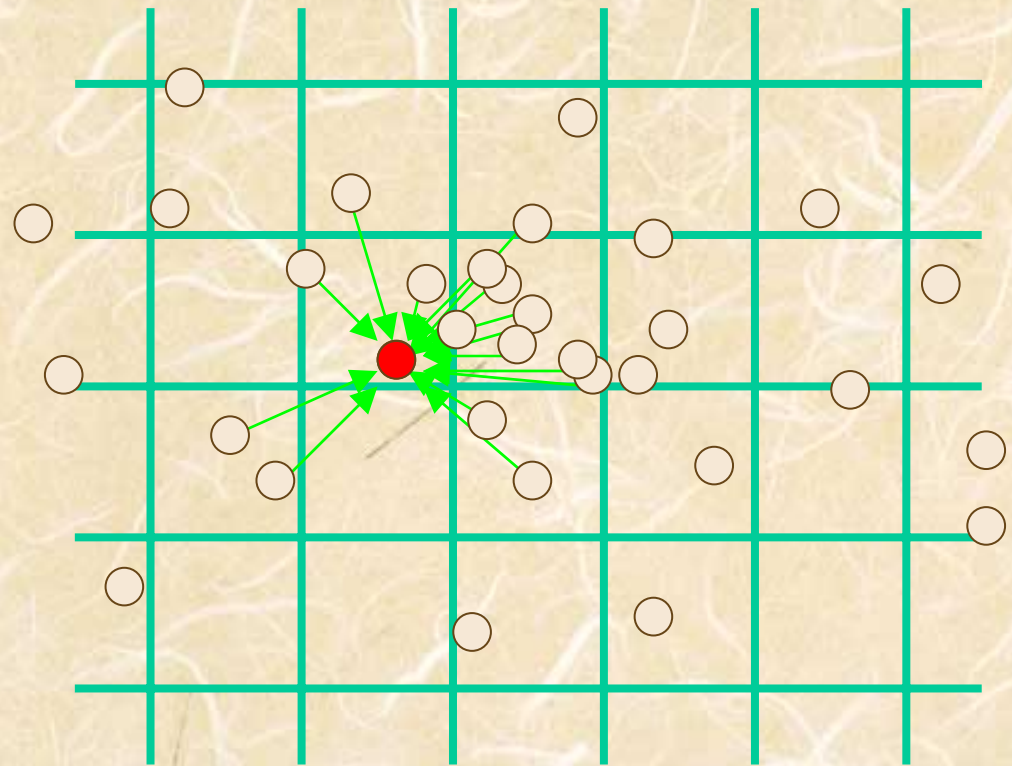
- Particle-particle particle-mesh (P³M) method

- Distant particles

- PM*

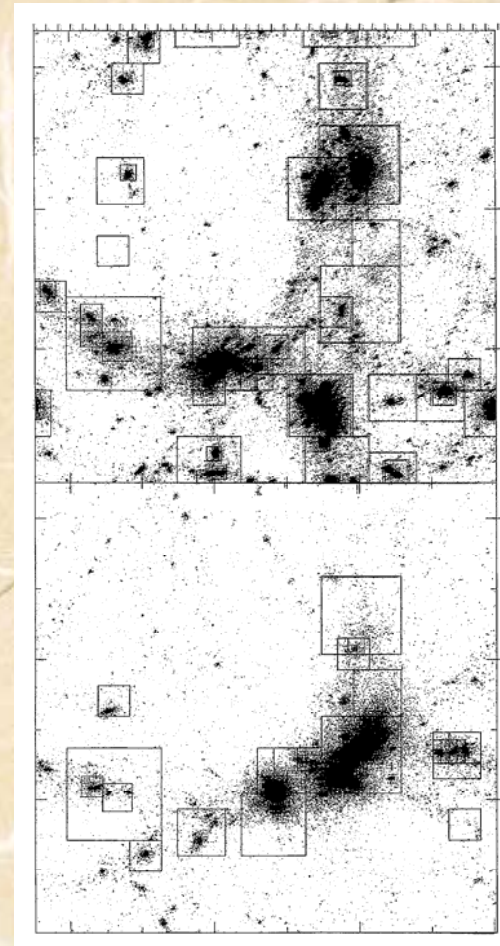
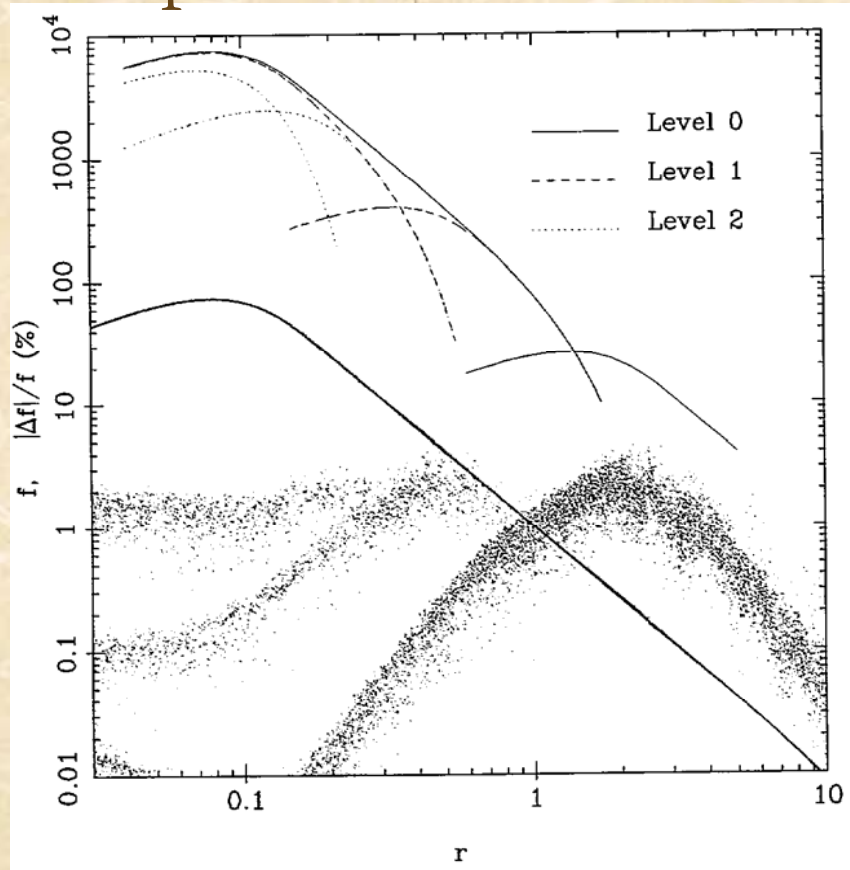
- Nearby particles

- Direct sum*



N-body Methods

- Adaptive P³M Method



Couchman (1991)

N-body Methods

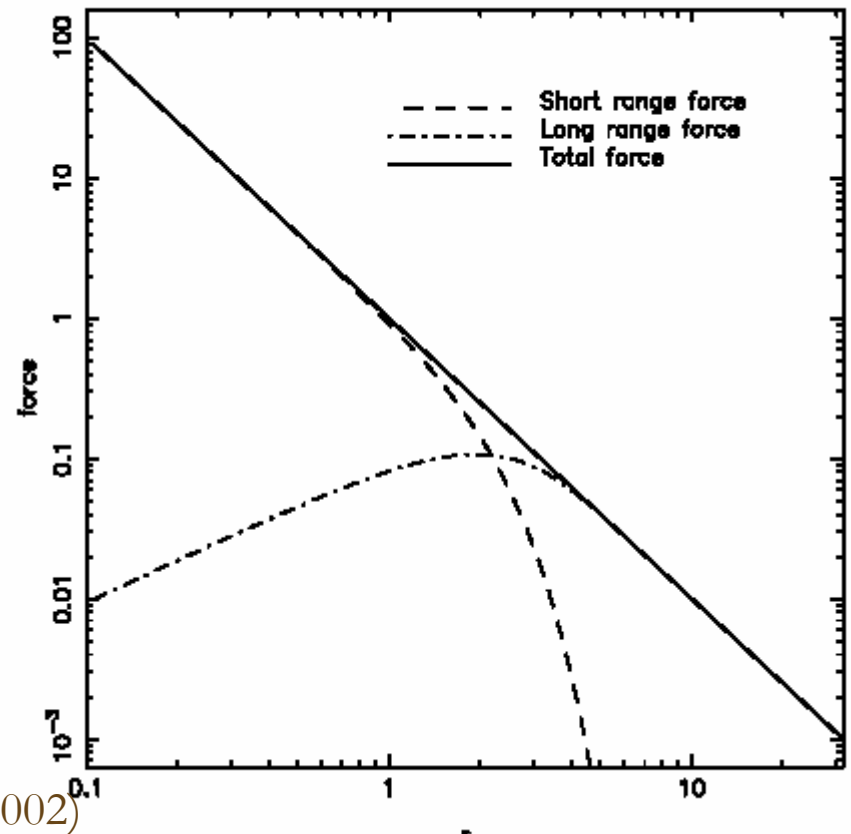
- TPM (Xu 1995; Bode+ 2000; Bode&Ostriker2003)
 - P³MのPPをTreeにする
- TreePM (Bagla 2002; Dubinski+ 2004; Yoshikawa&Fukushige 2005; Springel 2005)
 - P³Mとは異なる重力の分解をする

can be split into two parts in Fourier space (Ewald 1921).

$$\begin{aligned}\varphi_k &= -\frac{4\pi G\rho_k}{k^2}, \\ &= -\frac{4\pi G\rho_k}{k^2} \exp(-k^2 r_s^2) - \frac{4\pi G\rho_k}{k^2} (1 - \exp(-k^2 r_s^2)), \\ &= \varphi_k^l + \varphi_k^s, \\ \varphi_k^l &= -\frac{4\pi G\rho_k}{k^2} \exp(-k^2 r_s^2), \\ \varphi_k^s &= -\frac{4\pi G\rho_k}{k^2} (1 - \exp(-k^2 r_s^2)),\end{aligned}$$

(Bagla 2002)

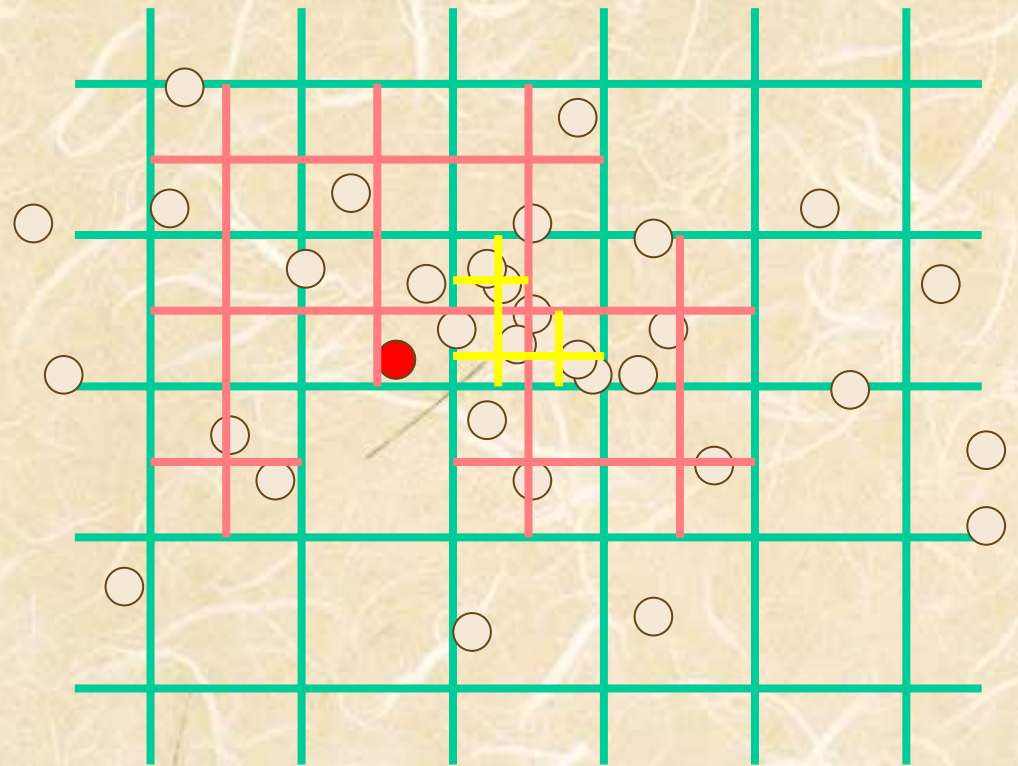
TreePM: A Code for Cosmological N-Body Simulations



N-body Methods

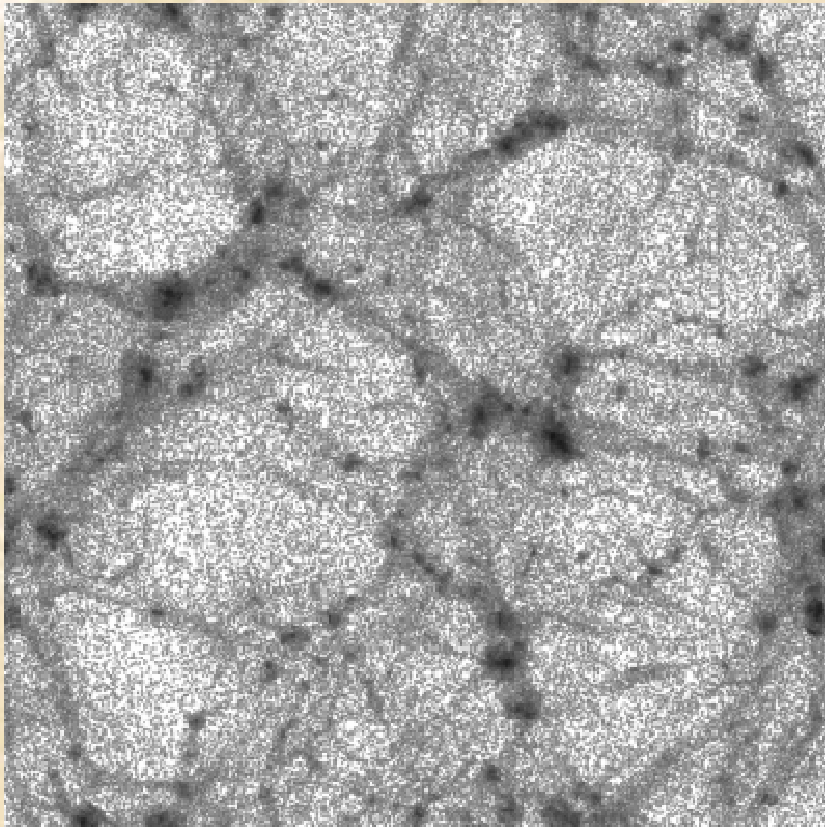
- Adaptive Mesh Refinement (AMR)

- Divide cells only where higher resolution required

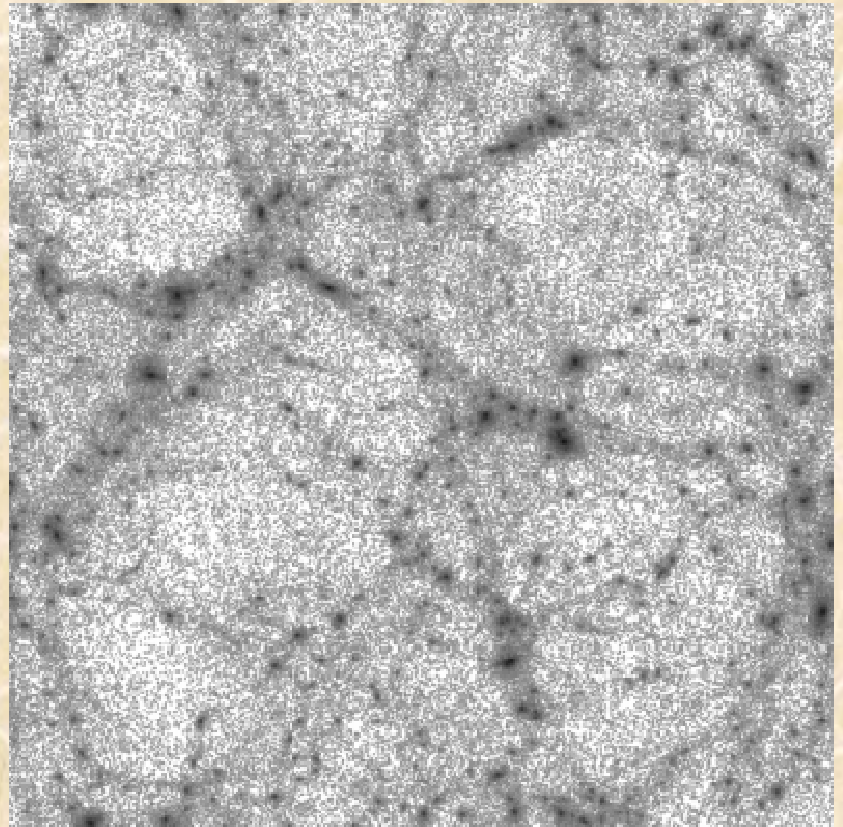


N-body Methods

- AMR

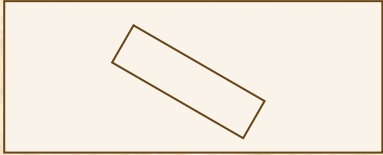


PM



AMR

AMR の歴史

- Berger & Collela (1989)
 - 数値流体法で分解能が必要な個所に、より細かい構造格子を再帰的に配置
 - BC(89)では斜めに格子を配置している。
 - この手法は、その後余り受け入れられていないようである。
 - Moving Mesh との違い
 - *AMR*: 決まった分解能を最小の計算機資源で実現
 - *Moving Mesh*: 決まった計算機資源で最大の分解能を実現

AMRの宇宙論業界での歴史

- Villumsen (1989)
- Anninos, Norman, & Clarke (1994)
- Kravtsov, Klypin, & Khokhlov (1997; ART)
- Norman & Bryan (1999; Enzo)
- Knebe, Green, & Binney (2001)
- Yahagi & Yoshii (2001)
- Teyssier (2002; RAMSES)

Villumsen (1989)

NO. 3, 1989

HIERARCHICAL PARTICLE-MESH N-BODY CODE

419

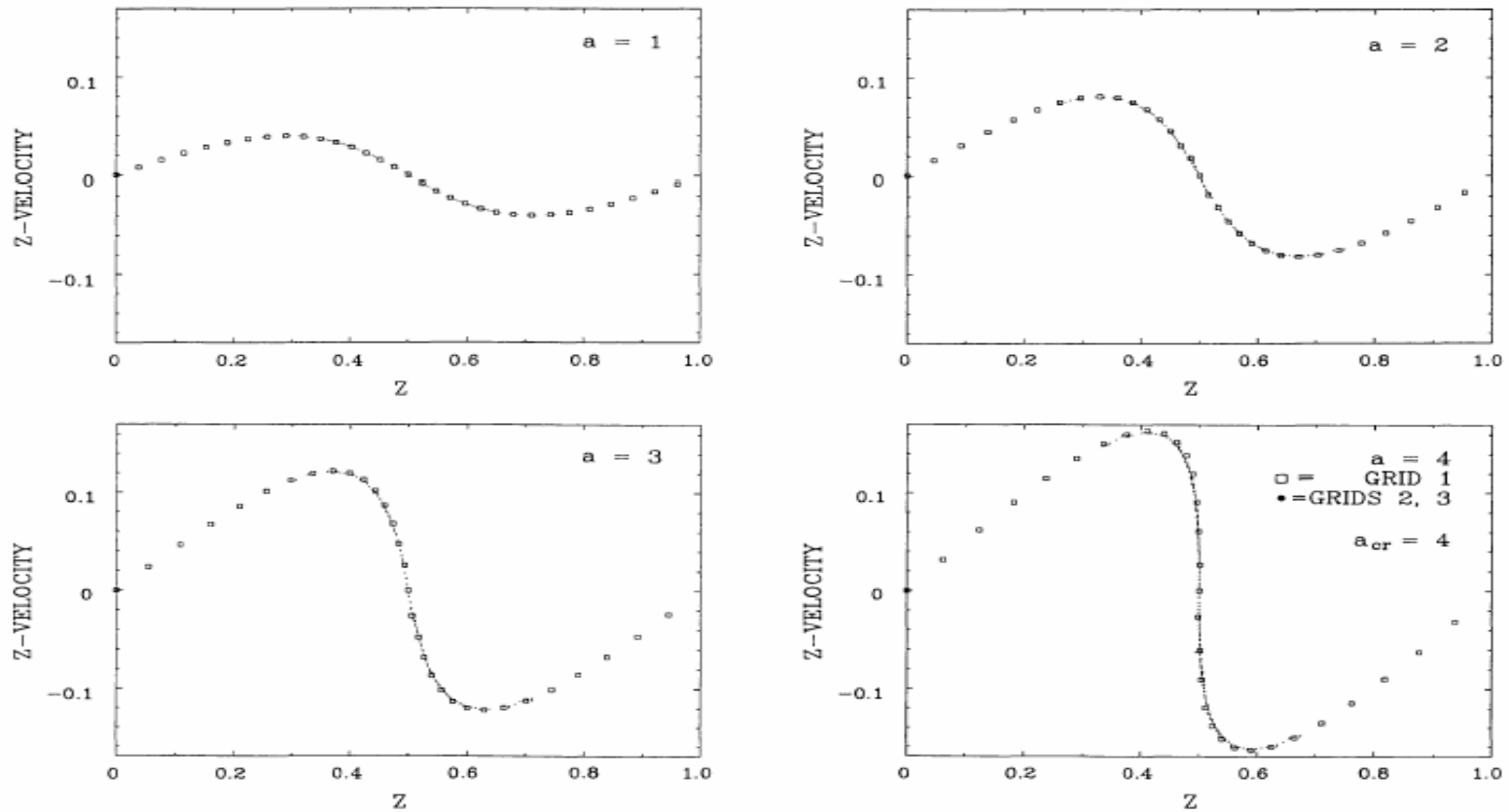


FIG. 5.—Reduced phase-space plot for one-dimensional pancake collapse where cresting occurs at $a = 4$. Open squares are for particles in the top grid; dots are particles in the subgrids. The four panels show the simulation at $a = 1, 2, 3, 4$.

Anninos, Norman, & Clarke (1994)

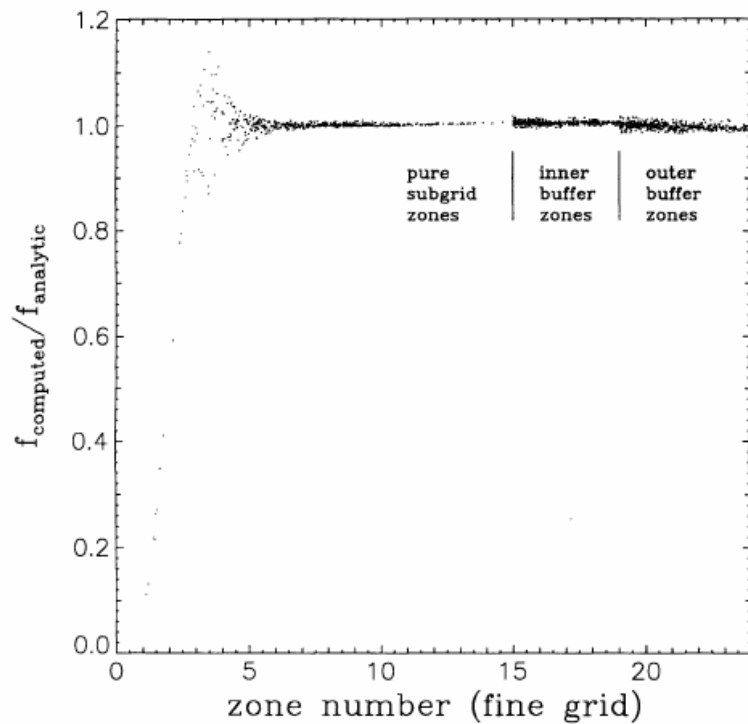


FIG. 4.—Same as Fig. 3, except that the forces computed on the subgrid are tested here.

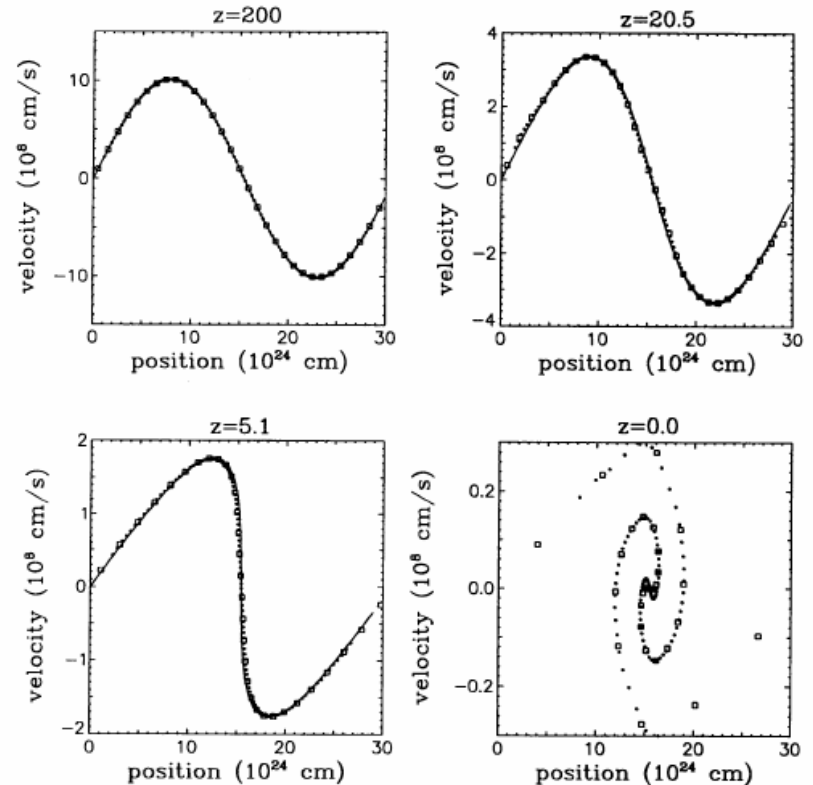


FIG. 5.—Time-sequence displays of phase space for the dark matter particles with the Zel'dovich solution as initial data. Open squares represent particles on the coarse grid, small dots are particles evolved on the subgrid scales, and the solid line is the analytic solution.

Norman & Bryan (1999)

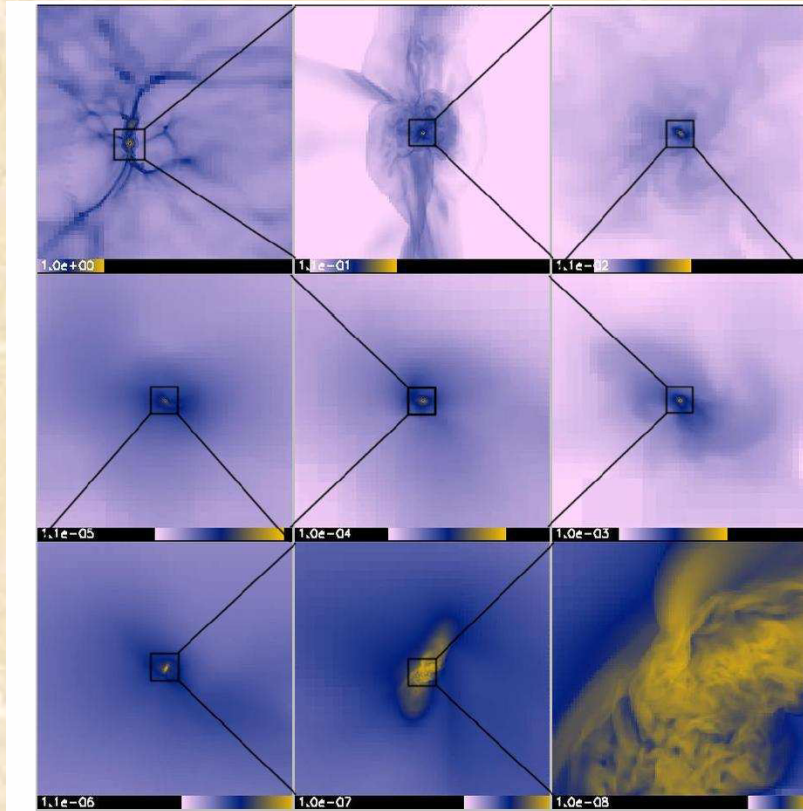


Figure 3: In these frames we show a zoom into the star forming region. Each panel shows a slice of the logarithm of the gas density magnified by a factor of ten relative to the previous frame, starting at the upper left.

Bryan, Abel, & Norman (astro-ph/0112089)

Kravtsov, Klypin, & Khokhlov (1997)

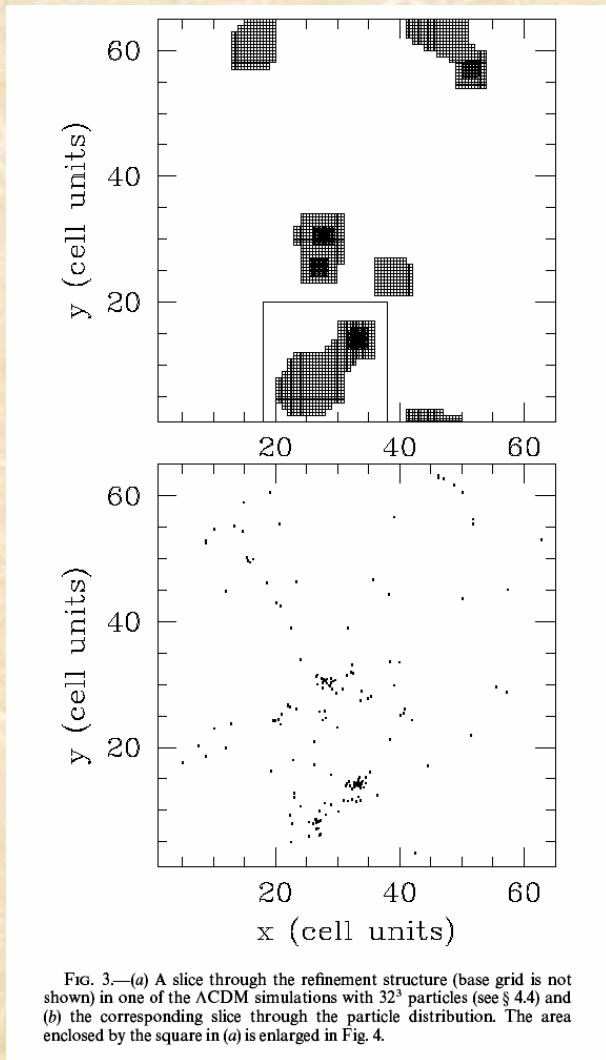


FIG. 3.—(a) A slice through the refinement structure (base grid is not shown) in one of the Λ CDM simulations with 32^3 particles (see § 4.4) and (b) the corresponding slice through the particle distribution. The area enclosed by the square in (a) is enlarged in Fig. 4.

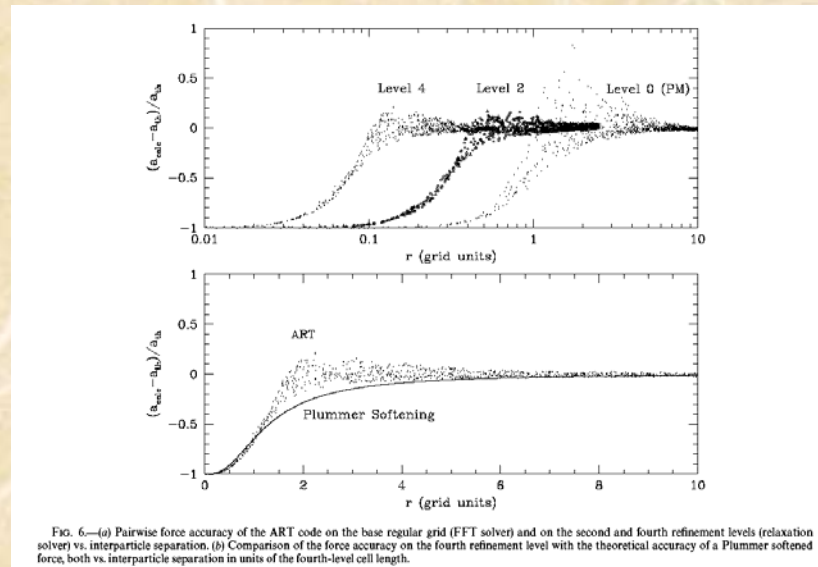


FIG. 6.—(a) Pairwise force accuracy of the ART code on the base regular grid (FFT solver) and on the second and fourth refinement levels (relaxation solver) vs. interparticle separation. (b) Comparison of the force accuracy on the fourth refinement level with the theoretical accuracy of a Plummer softened force, both vs. interparticle separation in units of the fourth-level cell length.

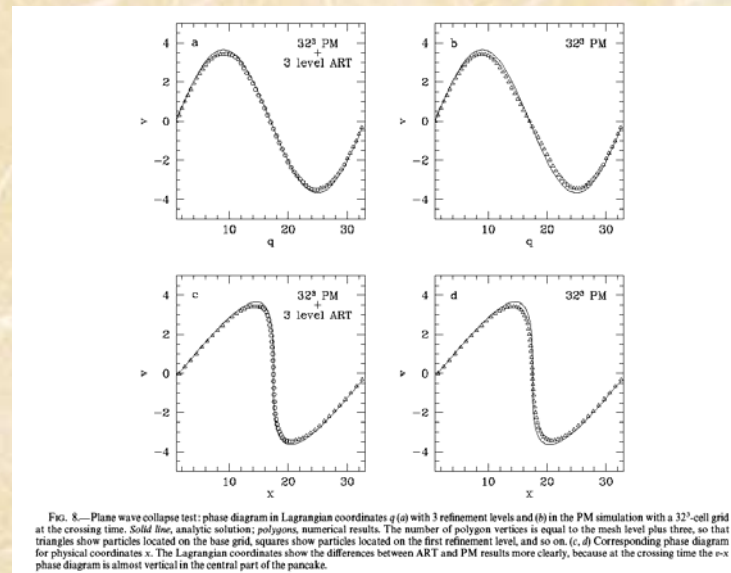


FIG. 8.—Plane wave collapse test: phase diagram in Lagrangian coordinates q (a) with 3 refinement levels and (b) in the PM simulation with a 32^3 -cell grid at the crossing time. Solid line, analytic solution; polygons, numerical results. The number of polygon vertices is equal to the mesh level plus three, so that triangles show particles located on the base grid, squares show particles located on the first refinement level, and so on. (c, d) Corresponding phase diagram for physical coordinates x . The Lagrangian coordinates show the differences between ART and PM results more clearly, because at the crossing time the x - q phase diagram is almost vertical in the central part of the pancake.

Yahagi & Yoshii (2001)

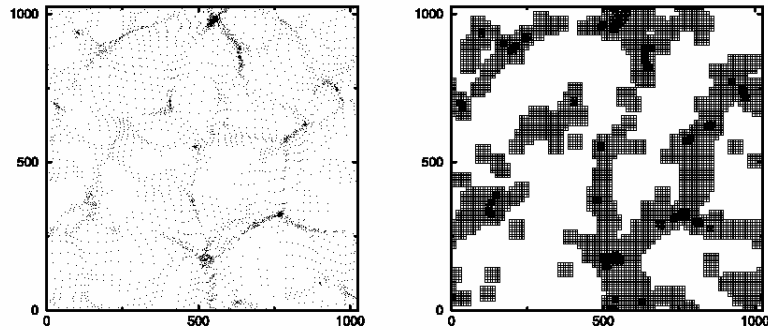


FIG. 4.—Example of the hierarchical mesh distribution in the N -body code with AMR for the case of the LCDM universe described in § 3.4. When particles are distributed as shown in the left-hand panel, hierarchical meshes are placed as shown in the right-hand panel. Shape of the hierarchical meshes is not geometrically restricted.

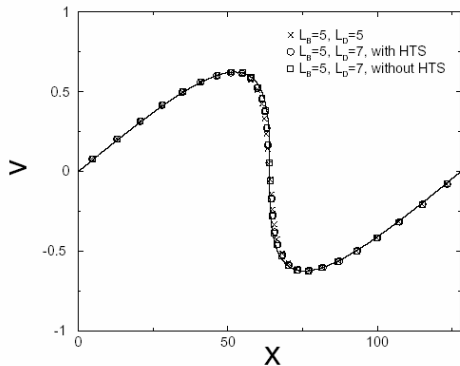


FIG. 10.—Snapshot of the phase diagram from the single-plane wave test at the epoch of first caustic generation. The solid line shows the exact solution calculated by the one-dimensional code with 1024 sheets using the code described in Yano & Gouda (1998). Crosses, circles, and squares show the results obtained by the code without AMR, with AMR and HTS, and with AMR but without HTS, respectively. Because the meshes are refined only once at this epoch, the difference between the AMR and non-AMR results is minor. We note, however, that this difference becomes larger as time proceeds beyond the first caustic generation (see Fig. 11).

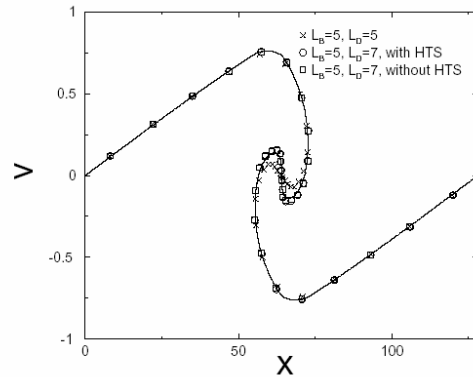


FIG. 11.—Snapshot of the phase diagram from the single-plane wave test at the epoch of second caustic generation. Same as Fig. 10 but for $a_2 \approx 2.34 a_1$, where a_1 and a_2 are the scale factors at the epoch of first and second caustics generation, respectively. Although blunt in the non-AMR run, the second caustics are well captured in the AMR result, irrespective of whether the HTS is included or not.

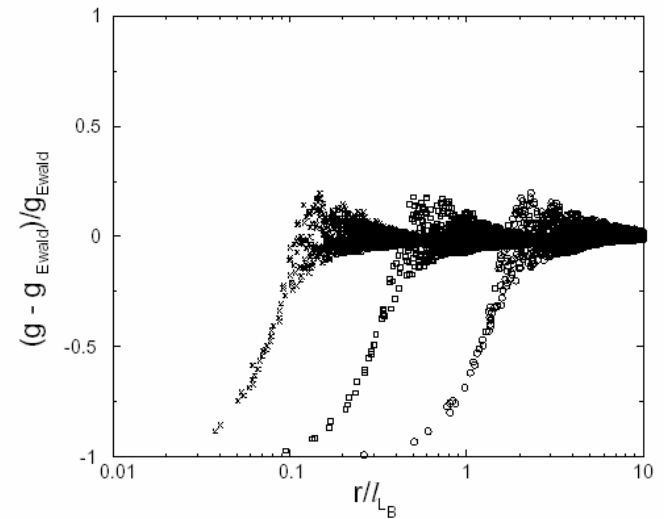
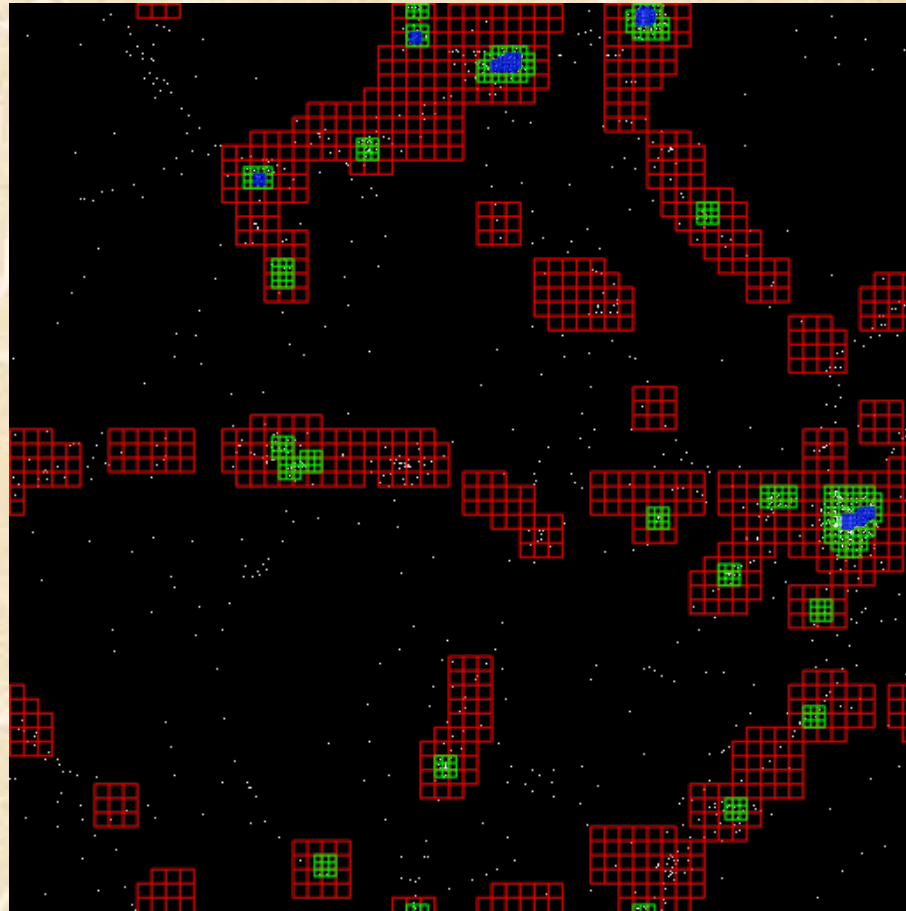


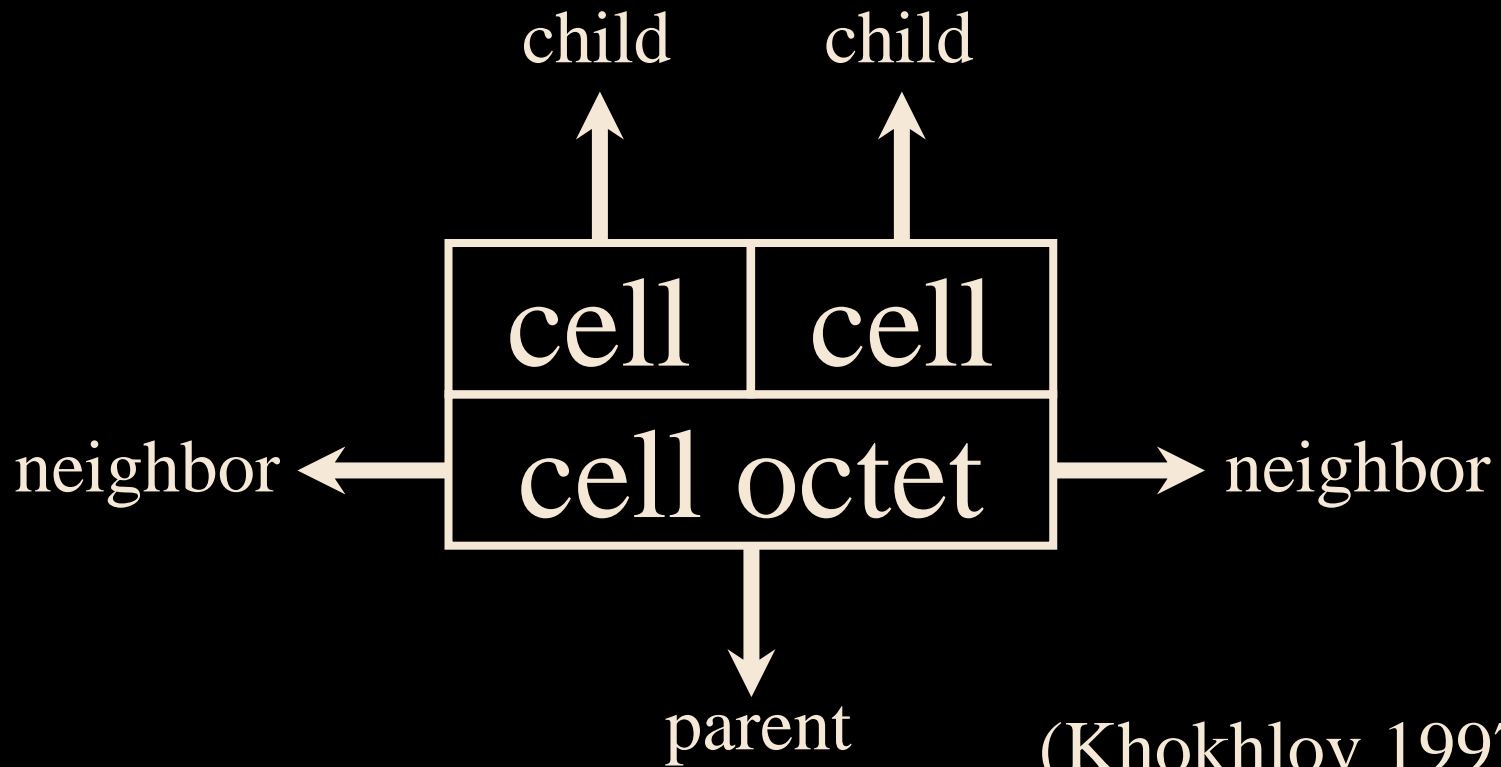
FIG. 9.—Error of the pairwise force calculated by the AMR code. Circles, squares, and crosses are for $L_D = 6, 8, 10$, respectively, with $L_B = 6$ in common. Error is defined as the relative error of the calculated force using the AMR code in comparison with that calculated by the Ewald expansion. In all cases, the error is maximized at $r \sim 2/L_D$ but is kept within 20%.

Adaptive Mesh Refinement



Adaptive Mesh Refinement

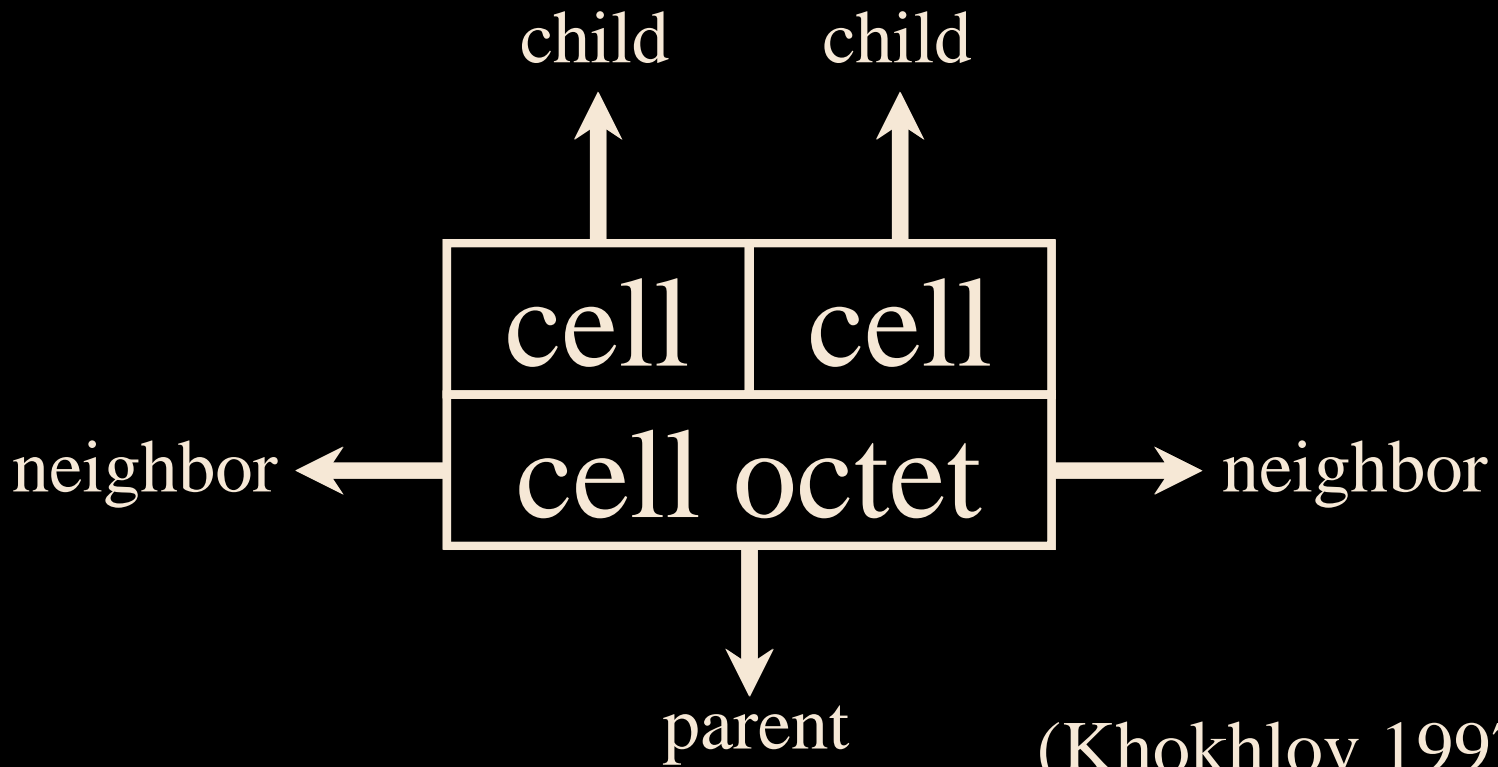
- Data structure



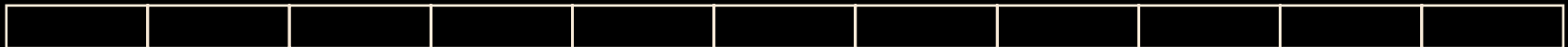
(Khokhlov 1997)

Adaptive Mesh Refinement

- Data structure

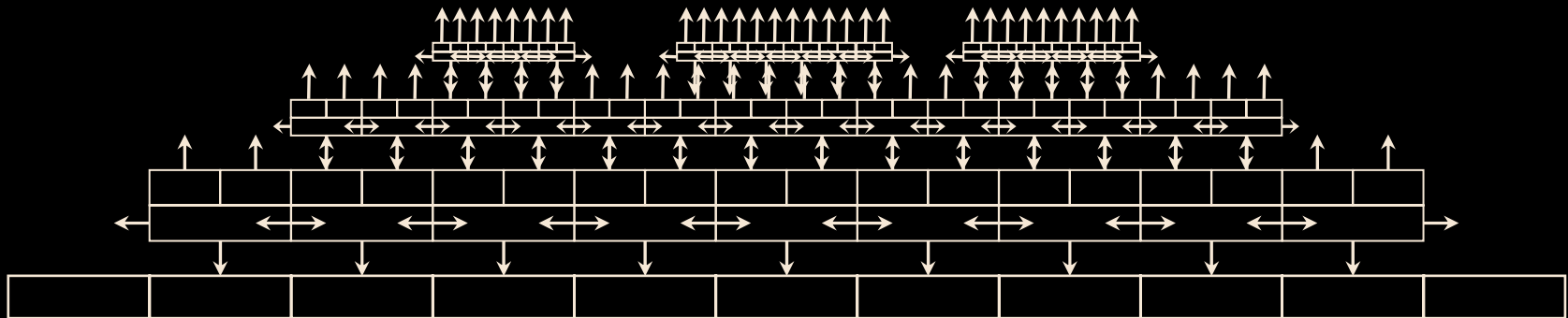


(Khokhlov 1997)



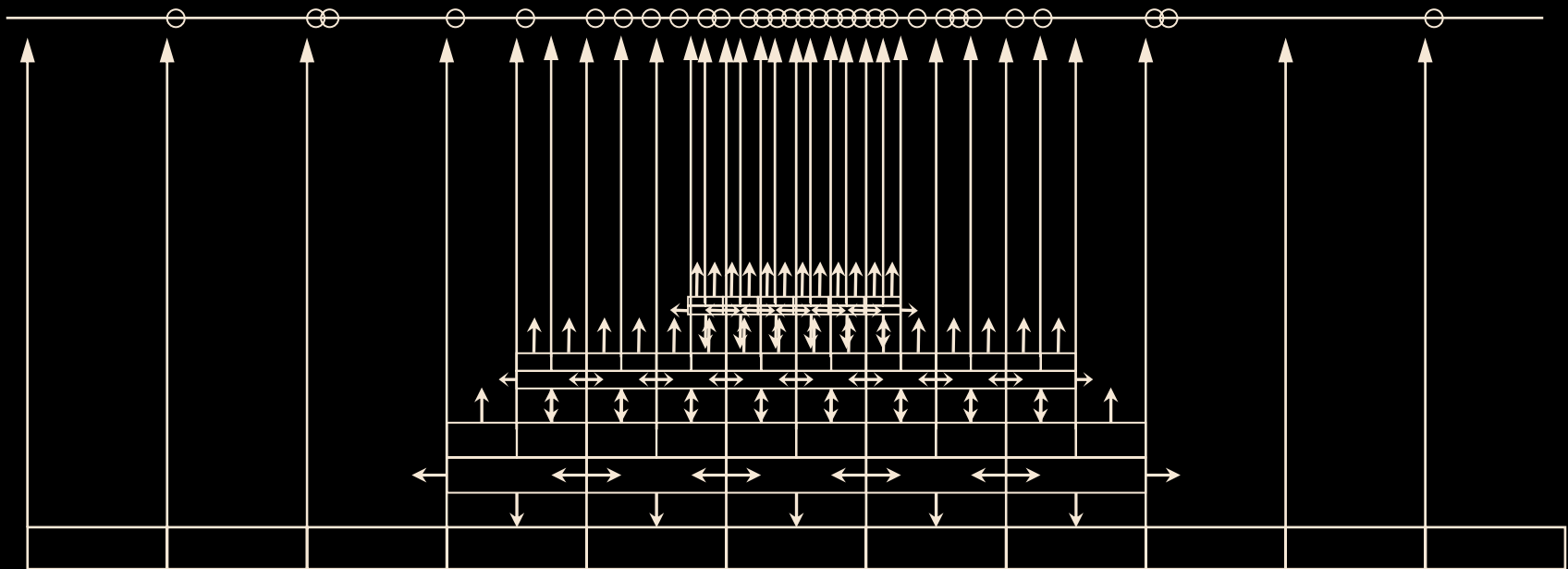
Adaptive Mesh Refinement

- Data structure



Adaptive Mesh Refinement

- Data structure

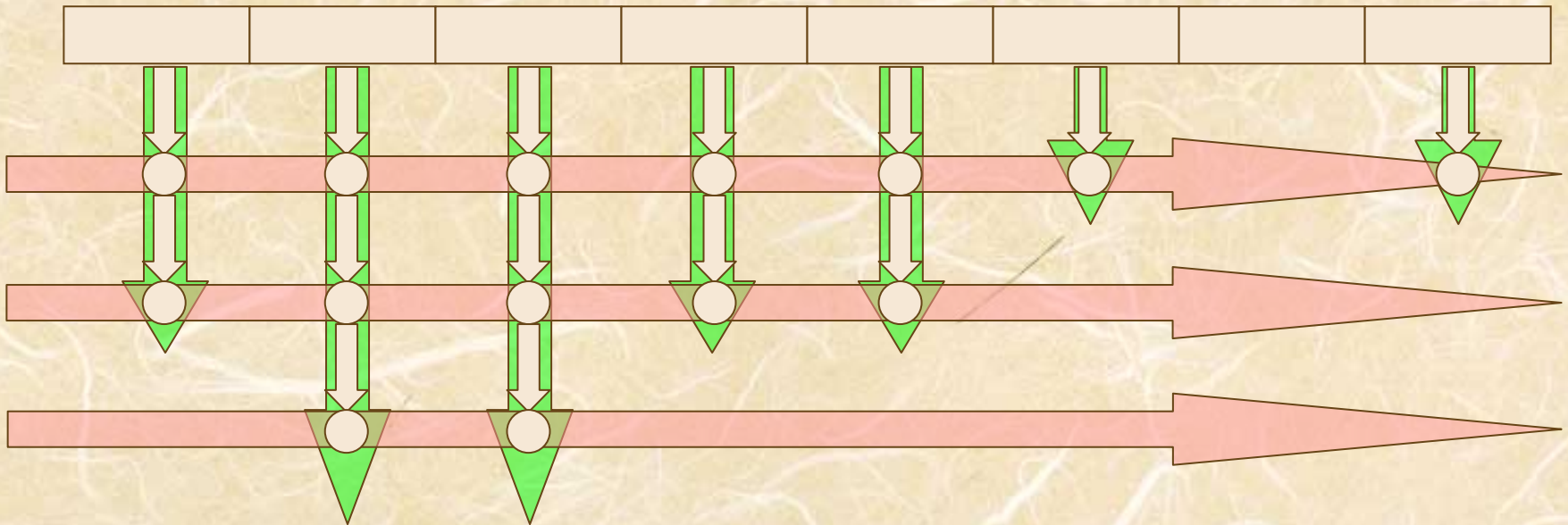


Vectorization

- Particle loops
 - Integration of the equation of motion
- Mesh loops
 - Poisson equation
- Particle-Mesh loops
 - Mass assignment
 - force interpolation
 - particle sieve

Vectorization

- Loop order exchange



- *Tree* (Hernquist 1990)
- *Hash* (Yabagi 2002)

Vectorization

- Loop split

- Vectorized mass assignment



AMRコードの速度

Acronym	Type	Ref.	Computer(Nproc)	Box size [Mpc]	
AMR	PM+AMR	(a)	VPP5000(1)	64	$z = 0$, periodic
ART	PM+AMR	(b)	SPP-1200(8)	$15h^{-1}$	$z = 0$, periodic
Mac	P ³ M	(c)	T3E-900(128)	512	$a \sim 0.9$, periodic
TPM	PM+Tree	(d)	Origin2000(256)	$150h^{-1}$	$z = 0.5$, periodic
FS	Tree	(e)	GRAPE5(1)	50-200	1k-2k steps, $q = 0.4$
FM	Tree	(f)	GRAPE5(2)	2-16	6k-8k steps, $q = 0.75$
W98	Tree	(g)	Avalon(70)	200	700 steps
W97	Tree	(h)	ASCI Red(4096)	200	last 286 steps

(a) This work; (b) Kravtsov, Klypin, & Khokhlov 1997; (c) MacFarland et al. 1998; (d) Bode, Ostriker, & Xu 2000; (e) Fukushige & Suto 2001; (f) Fukushige & Makino 2001; (g) Warren et al. 1998; (h) Warren et al. 1997

AMRコードの速度

(1) Acronym	(2) Time/step [sec]	(3) N _{ptcl}	(4) particle/sec [10 ³]	(5) R _{max} [GFlops]	(6) (4)/(5) [MFlops ⁻¹]
AMR	47.18	256 ³	355.60	9.475	37.53
ART	185.3	64 ³	1.415	~0.9305	1.52
Mac	~ 110	256 ³	152.52	79.59	1.92
TPM	293.9	256 ³	57.09	101.4	0.56
FS	~ 150	8:8 10 ⁶	58.67
FM	21	~ 2 10 ⁶	95.25
W98	134.7	9753824	72.41	48.60	1.49
W97	118.3	322159436	2723.24	~ 598.8	4.55

高速化の動機

● 地球シミュレータ

- 並列化効率が50%になる台数まで利用可能
- VPP5000/32PEで約50%
- VPP5000とSX-6ではプロセッサあたりのFlops値はほぼ同じ
- プロセッサあたりのメモリは地球シミュレータの方が小さい
- つまり、天文台でできる計算より大きい計算が出来ない

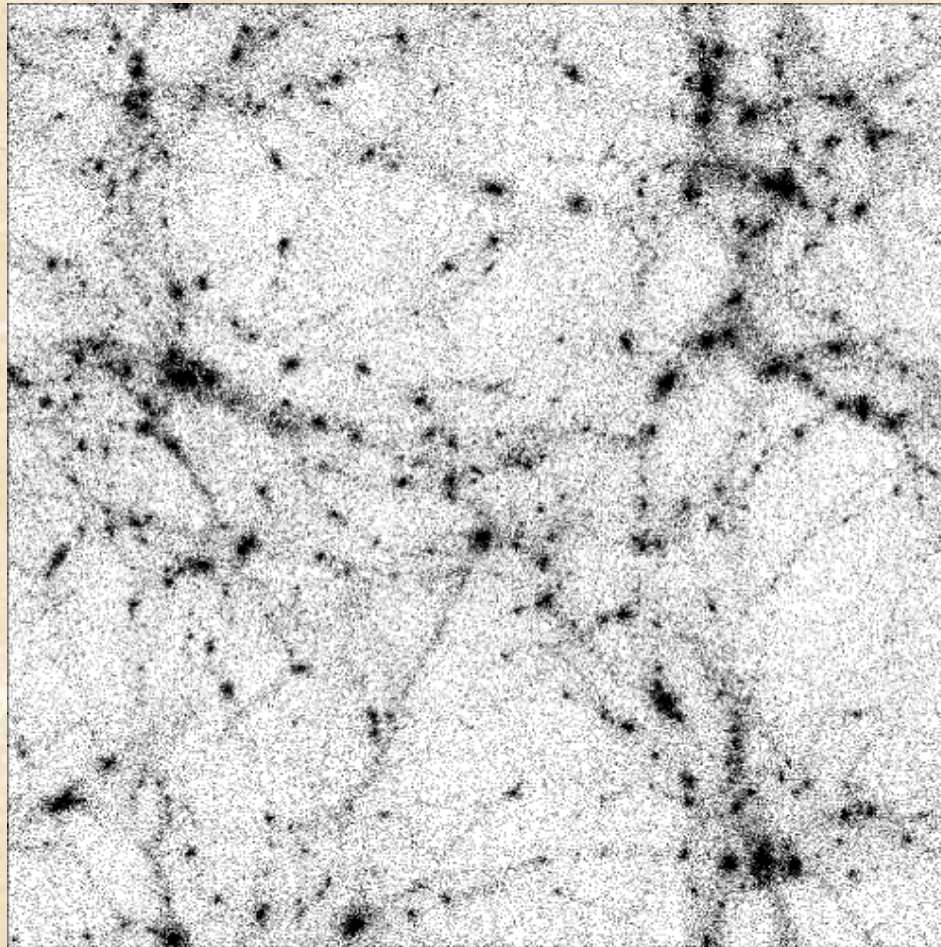
階層格子のデータ分割

- Morton ordering
 - Warren & Salmon (1995), Fryxell et al. (2000)

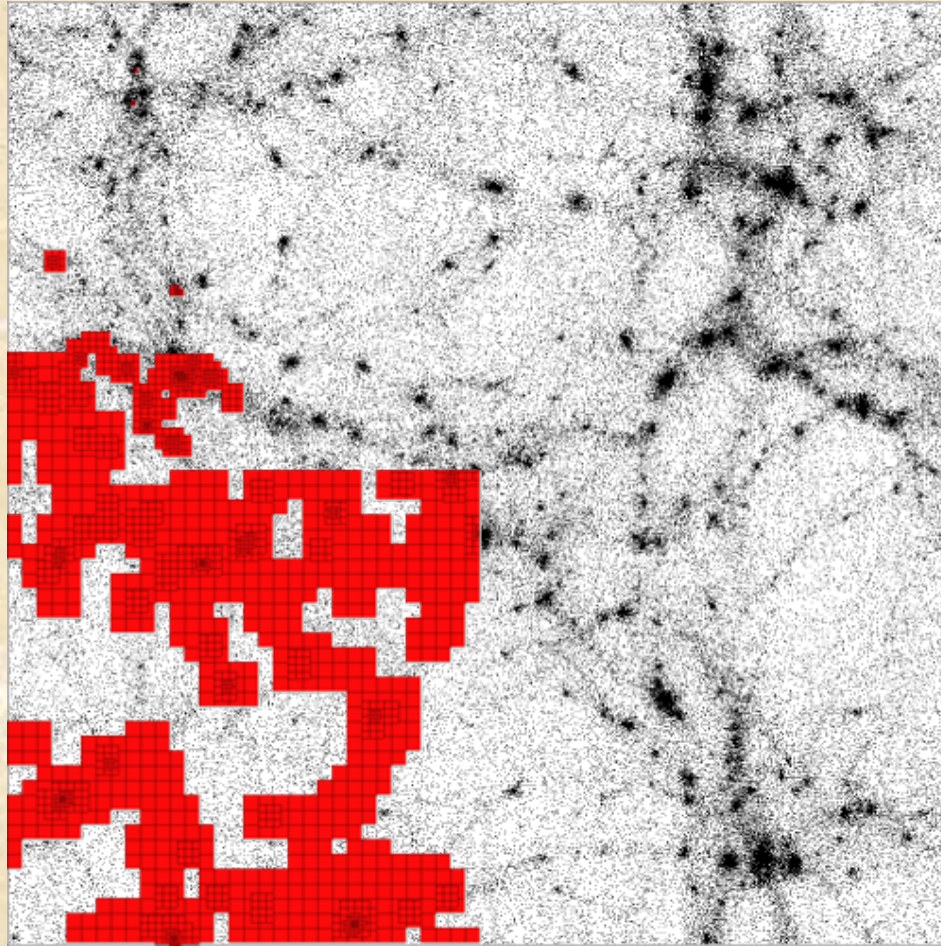


FIG. 1. *The self-similar space filling curve (Morton order) connecting the particles which is used to obtain the domain decomposition. Eight processor domains are shown in different levels of gray.*

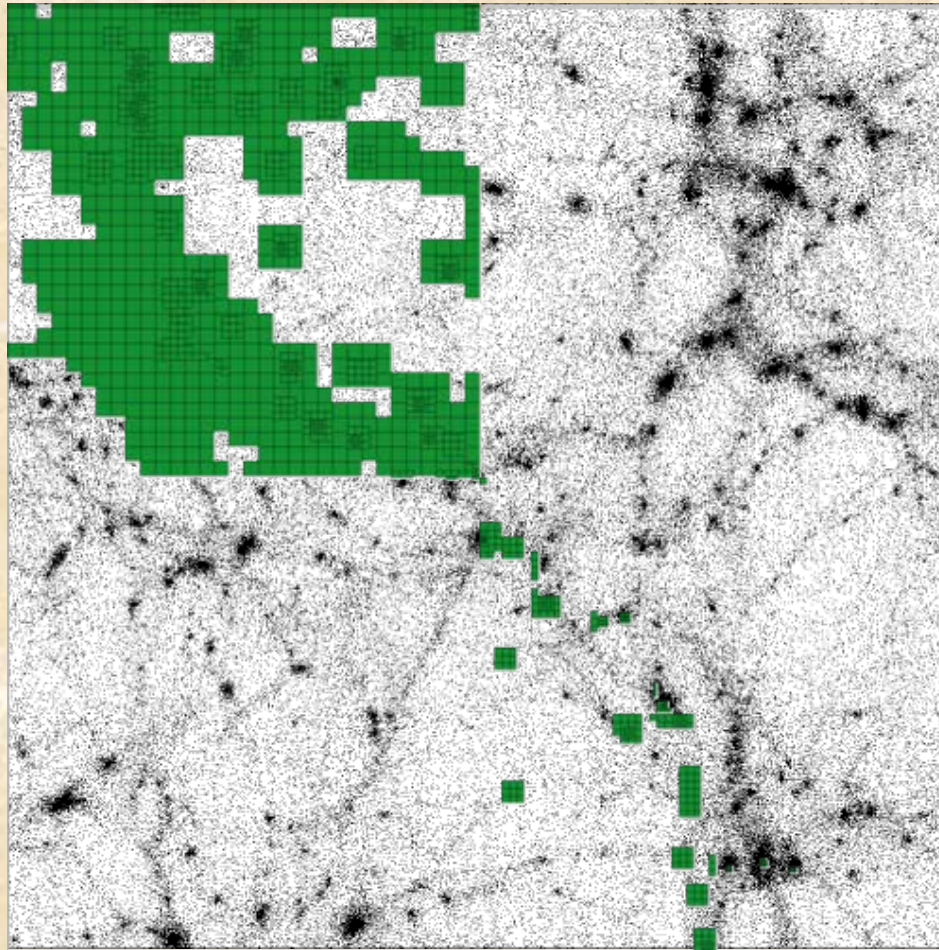
階層格子のデータ分割



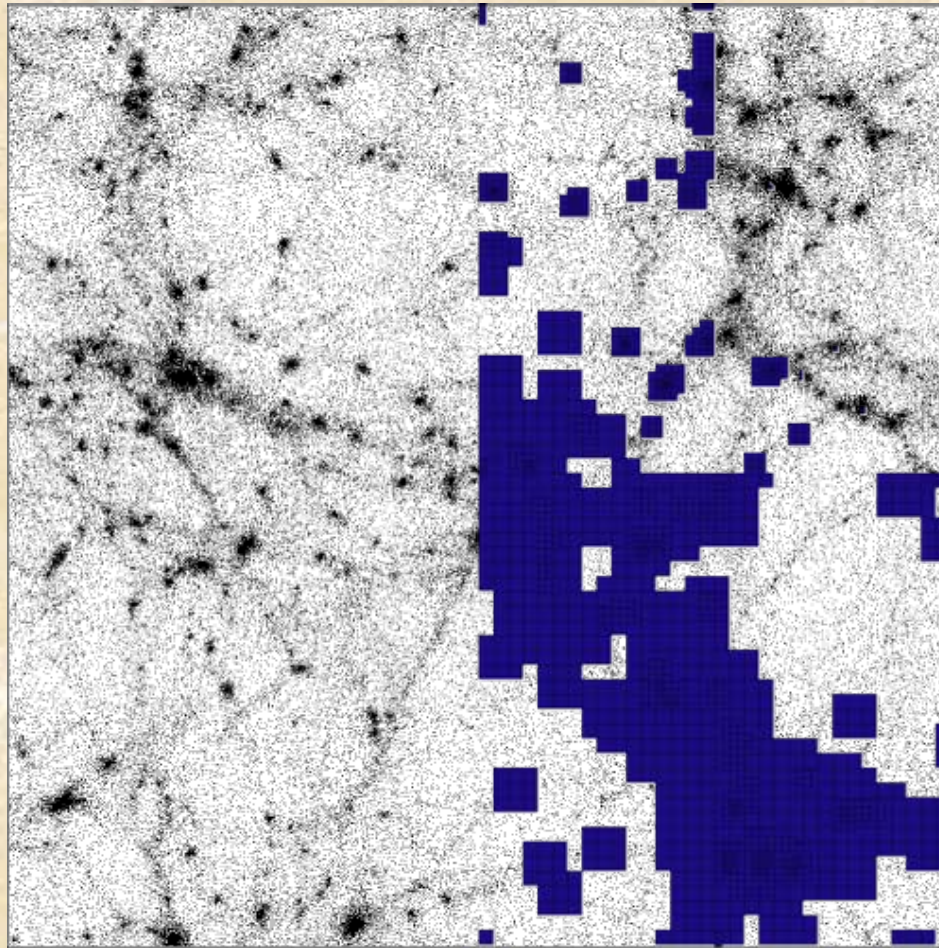
階層格子のデータ分割



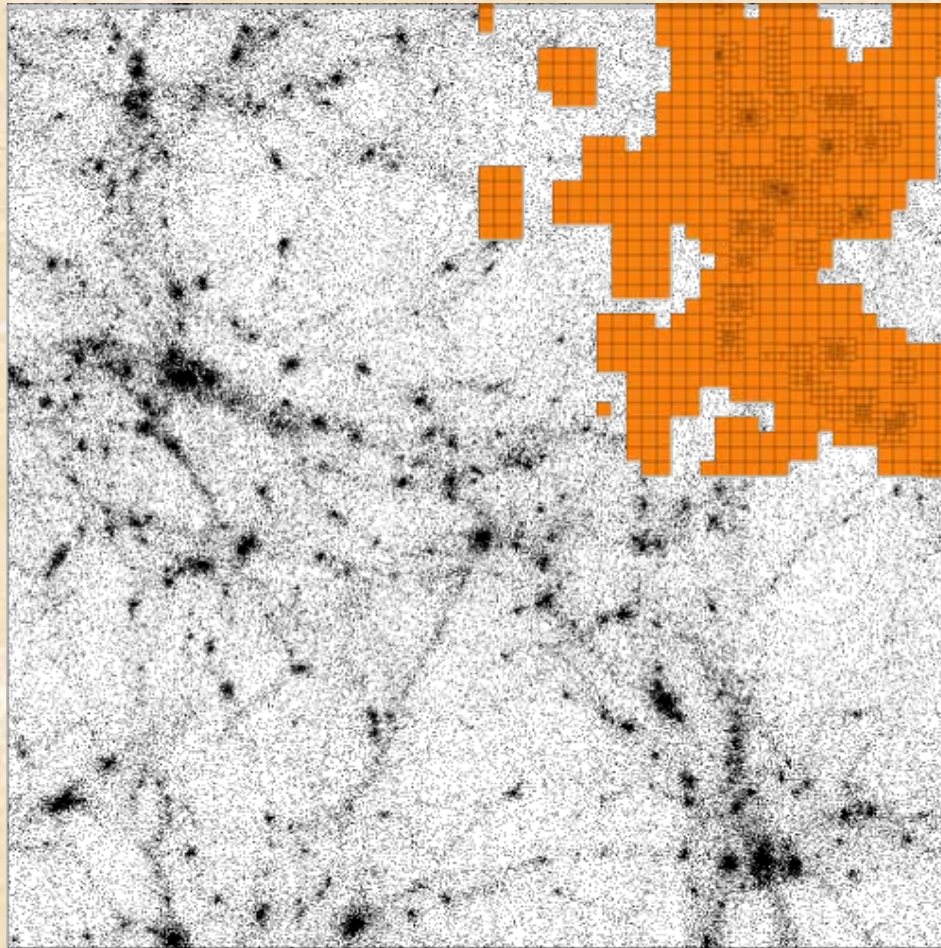
階層格子のデータ分割



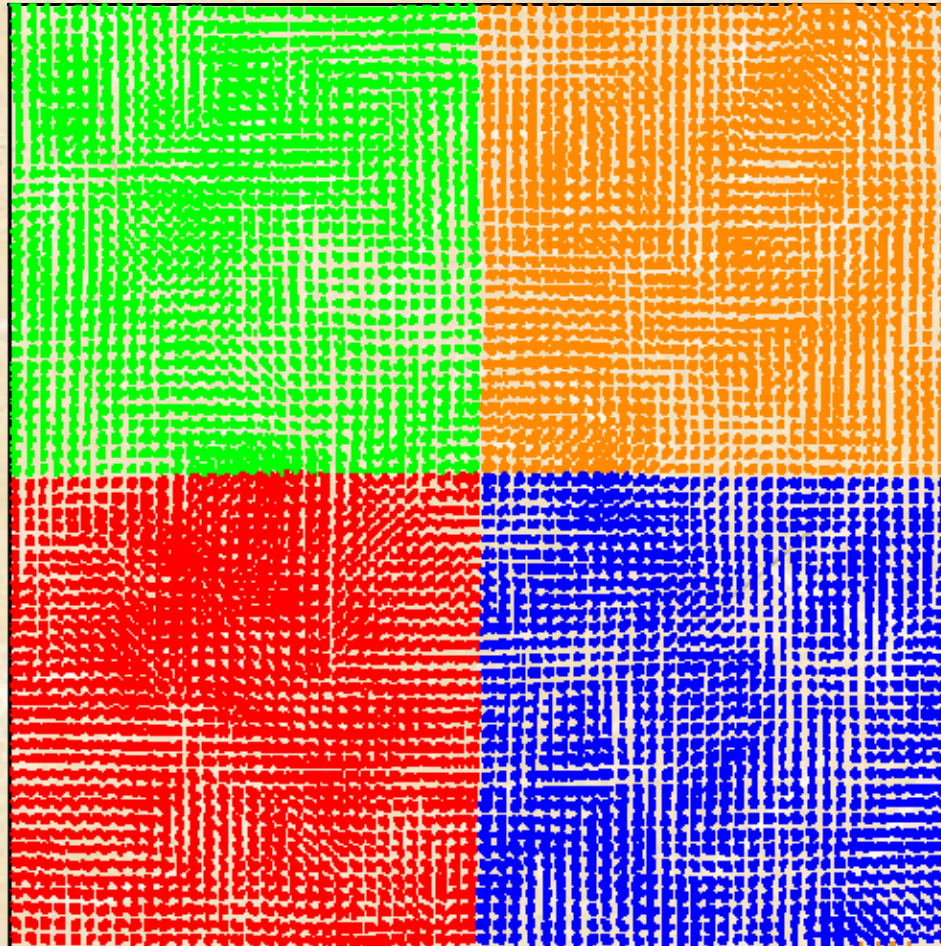
階層格子のデータ分割



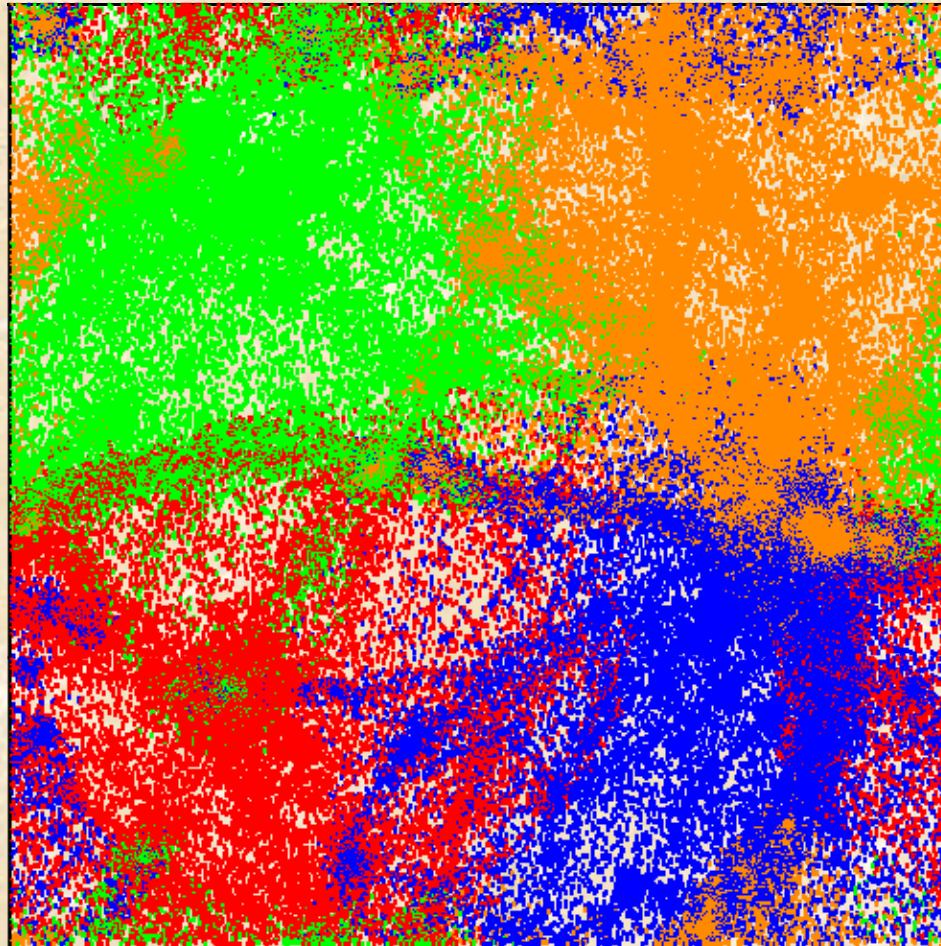
階層格子のデータ分割



粒子のデータ分割(旧コード)

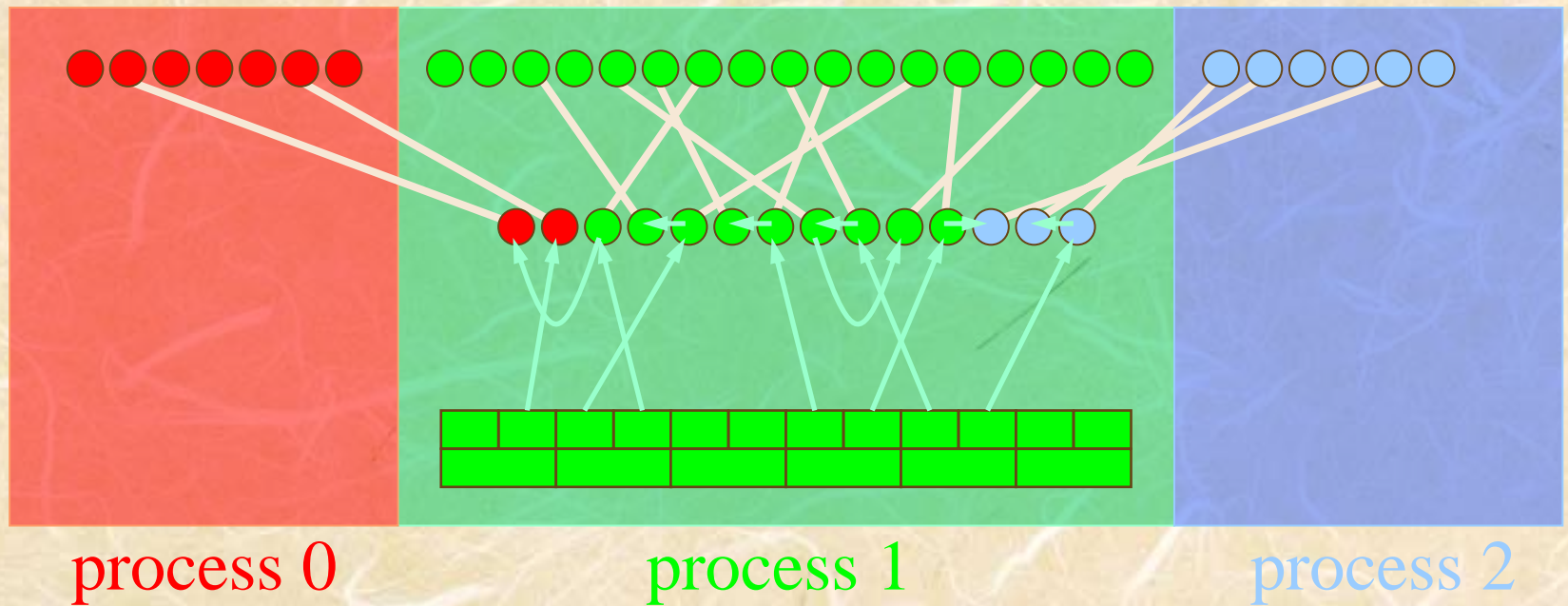


粒子のデータ分割(旧コード)

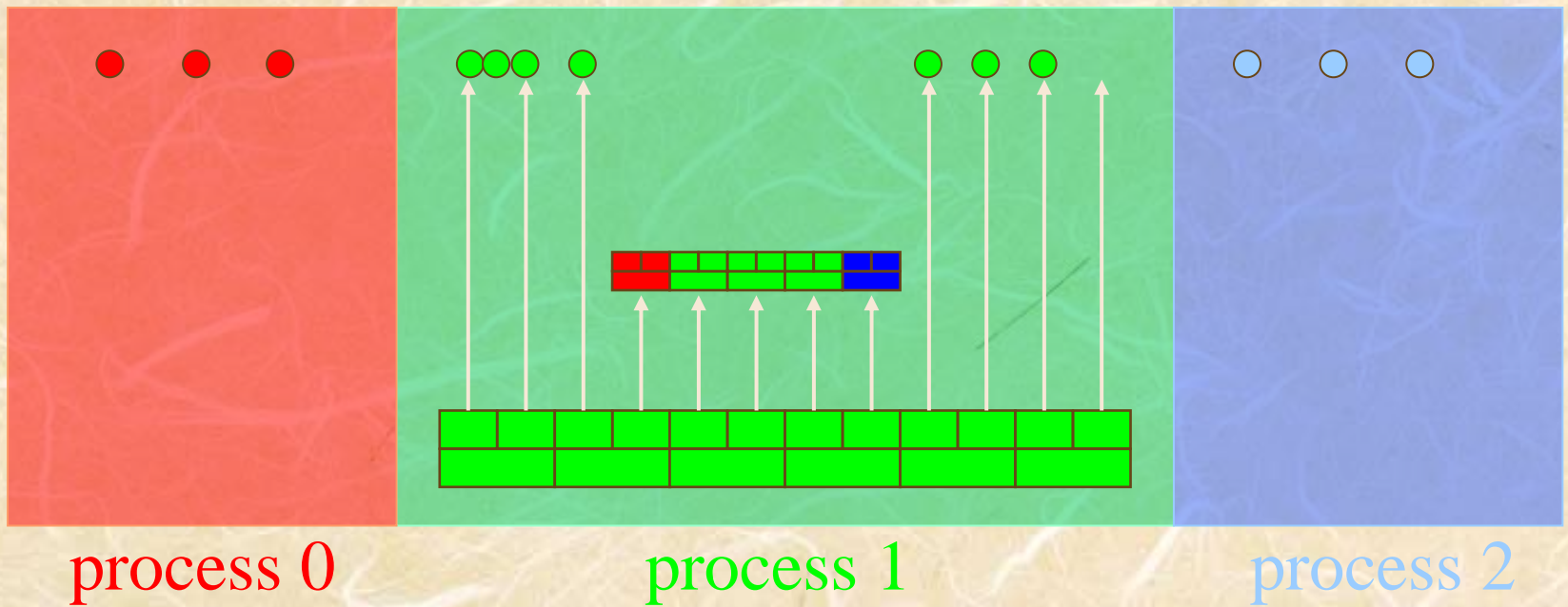


粒子のデータ分割(旧コード)

Force interpolation



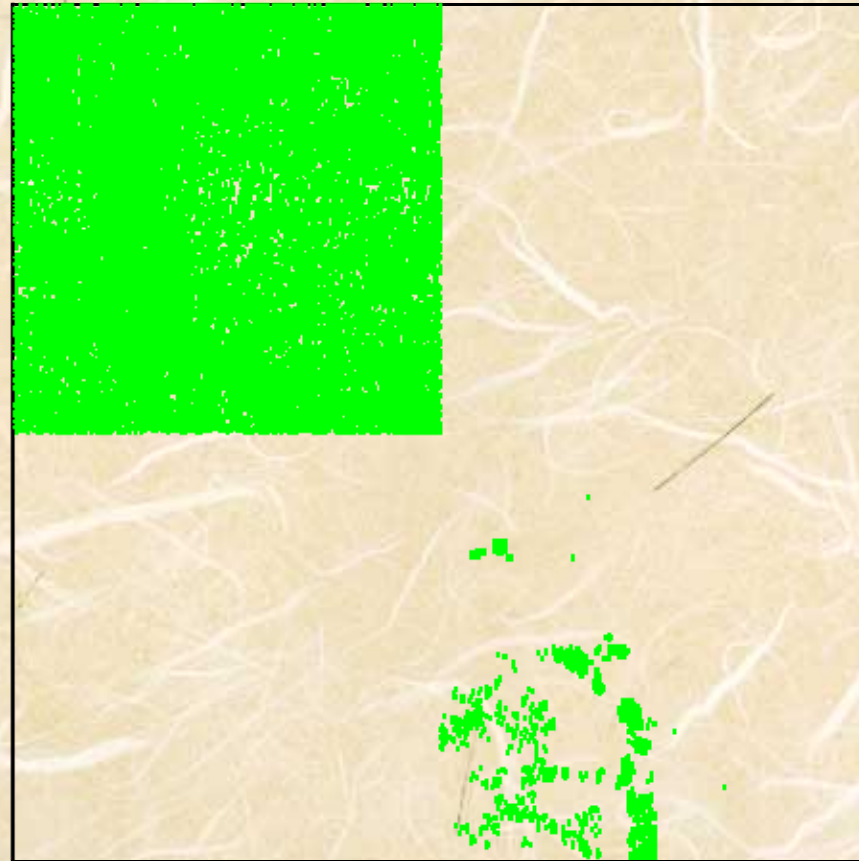
粒子のデータ分割(新コード)



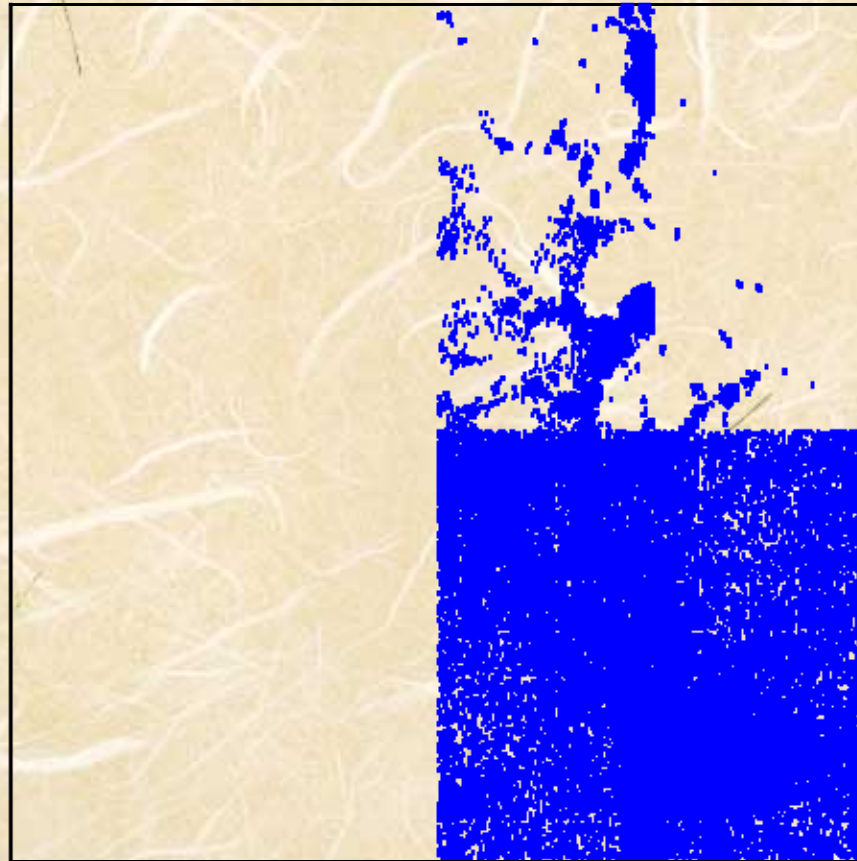
粒子データの分割(新コード)



粒子データの分割(新コード)



粒子データの分割(新コード)



粒子データの分割(新コード)



この先危険

- 私が遠くに言ってしまったら、この世に戻ってくるよう、遠慮なく呼び戻してください。

粒子データの分割

- 粒子の速度を更新
- 粒子の位置を更新
- 階層格子を更新

$$v_{i+1/2} = v_{i-1/2} + a_i \Delta t$$

$$x_{i+1} = x_i + v_{i+1/2} \Delta t$$

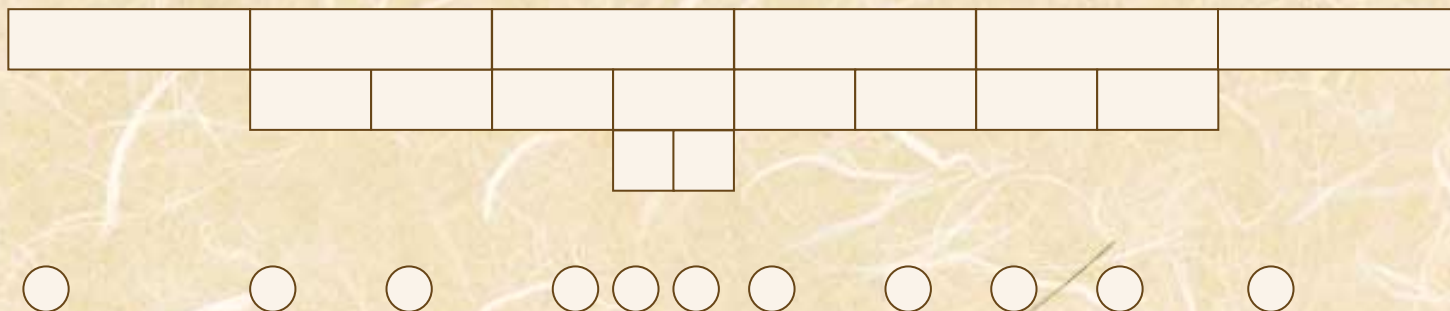
- 粒子の速度を更新
-
-
-

階層格子の更新と粒子データの転送

- 現在の階層格子に合わせて粒子をふるいわけ
- 階層格子の消去
- 階層格子の追加
- 粒子リンクリストの局所化
- 粒子データの転送

階層格子の更新と粒子データの転送

- 現在の階層格子に合わせて粒子をふるいわけ

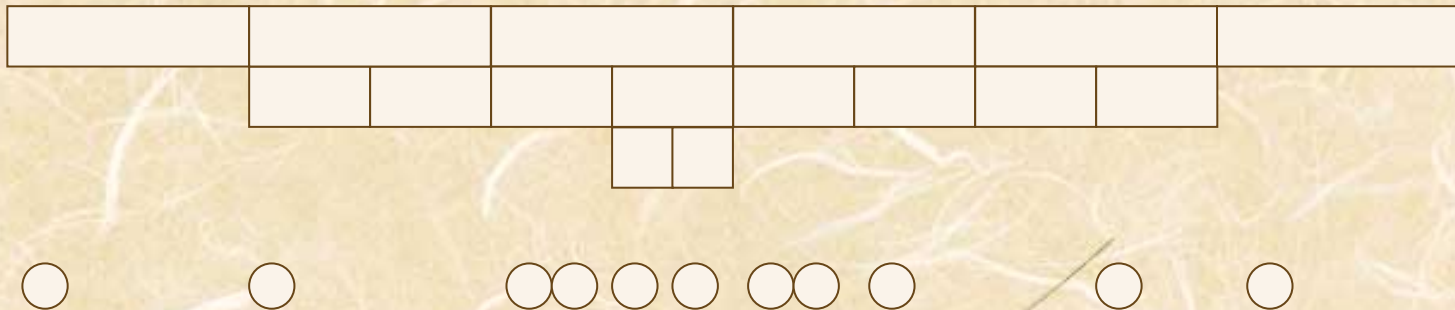


階層格子の更新と粒子データの転送

- 現在の階層格子に合わせて粒子をふるいわけ
- 階層格子の消去
- 階層格子の追加
- 粒子リンクリストの局所化
- 粒子データの転送

階層格子の更新と粒子データの転送

- 階層格子の消去

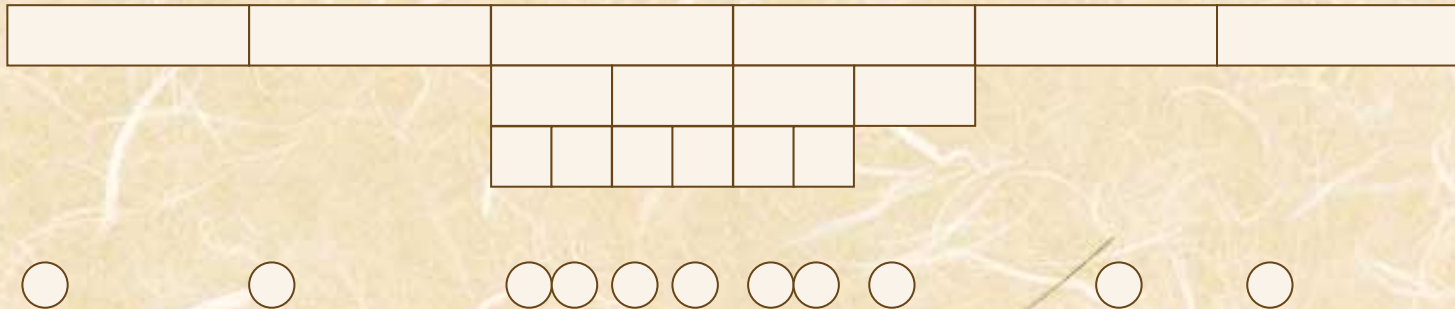


階層格子の更新と粒子データの転送

- 現在の階層格子に合わせて粒子をふるいわけ
- 階層格子の消去
- 階層格子の追加
- 粒子リンクリストの局所化
- 粒子データの転送

階層格子の更新と粒子データの転送

- 階層格子の追加



階層格子の更新と粒子データの転送

- 現在の階層格子に合わせて粒子をふるいわけ
- 階層格子の消去
- 階層格子の追加
- 粒子リンクリストの局所化
- 粒子データの転送

階層格子の更新と粒子データの転送

- 粒子リンクリストの局所化

- 階層格子の追加が行われた後に、階層格子は Morton 順序に従って整列させられる。
- その結果、階層格子とその配下にある粒子のリンクリストが異なるプロセッサ上にくることがある。
- そのため、階層格子と同じプロセッサ上へ粒子のリンクリストを転送する必要が生じる。

階層格子の更新と粒子データの転送

- 現在の階層格子に合わせて粒子をふるいわけ
- 階層格子の消去
- 階層格子の追加
- 粒子リンクリストの局所化
- 粒子データの転送

階層格子の更新と粒子データの転送

- 粒子データの転送

- 今まで転送されているのは粒子のリンクリストとインデックスだけである。
- そこで、仕上げに、粒子の位置及び速度情報を送りあう。

AMR N体コードの改造

- 32ビットの壁

```
#define LEV_NPTCL 23
#define MAX_NPTCL (1<<LEV_NPTCL)
#define MASK_PTCL_PROC (MAX_NPTCL-1)
...
for (ip=0; ip<np; ip++){
    proc[ip] = ptcl[ip] >> LEV_NPTCL;
    local[ip] = ptcl[ip] & MASK_PTCL_PROC;
}
```



プロセッサID

プロセッサ内の粒子ID

VPP5000(32PE) 512³体

2⁵=32

2*2²²=8M

ES(128PE) 1024³体

2⁷=128

2*2²³=16M

- ES(128PE)1024³体はぎりぎり入るが、粒子数分布の平均からのずれが1を超えると破綻
 - そこで、二つの4バイト整数に分割する

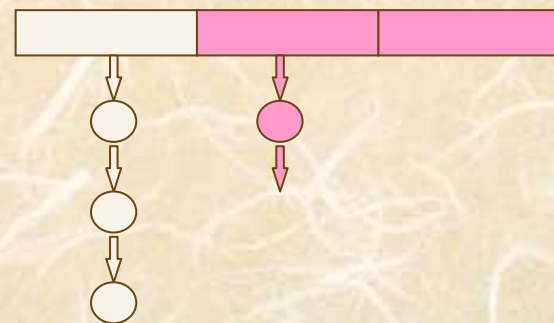


変更箇所

- 粒子のデータ構造(配列から延びるリンクリスト)の変更
 - 粒子の移動
 - セルは変更せずに、粒子のリンクリストのつなぎ替えをする
 - オープンセルにフラグを立てる
 - フラグの無いセルに入っている粒子を上階層に渡していく
 - セルを上階層から消したり足したりしていく、と同時に粒子を下階層へ渡していく

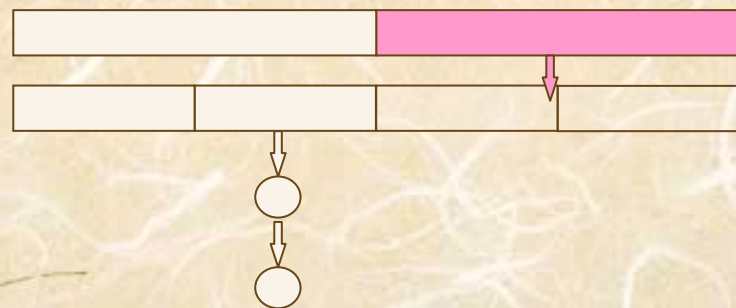
セルは変更せずに、粒子のリンクリストのつなぎ替えをする

- 下のレベルから上のレベルへ向けて以下の操作を行う
 - セル内に留まる粒子はそのままにして、セルの壁を越える粒子をリンクリストから抜き出す
 - 抜き出した粒子を含むセルが異なるPEにある場合は、その粒子を転送する。
 - 抜き出した粒子を含むセルがバッファセルの場合、粒子を上階層へ転送する。



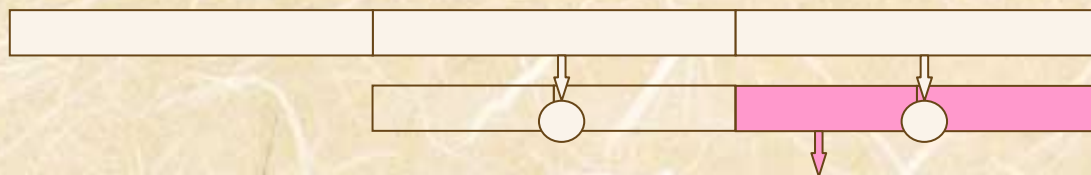
セルは変更せずに、粒子のリンクリストのつなぎ替えをする

- 下のレベルから上のレベルへ向けて以下の操作を行う
 - セル内に留まる粒子はそのままにして、セルの壁を越える粒子をリンクリストから抜き出す
 - 抜き出した粒子を含むセルが異なるPEにある場合は、その粒子を転送する。
 - 抜き出した粒子を含むセルがバッファセルの場合、粒子を上階層へ転送する。



セルは変更せずに、粒子のリンクリストのつなぎ替えをする

- 上のレベルから下のレベルへ向けて以下の操作を行う
 - 粒子を含むセルに子供がない、又はいてもバッファセルである場合は、粒子をリンクリストに残す
 - 粒子を含むセルの子供がオープンセルなら下の階層へ粒子を転送



変更箇所

- 粒子のデータ構造(配列から延びるリンクリスト)の変更
 - 粒子の移動
 - セルは変更せずに、粒子のリンクリストのつなぎ替えをする
 - オープンセルにフラグを立てる
 - フラグの無いセルに入っている粒子を上階層に渡していく
 - セルを上階層から消したり足したりしていく、と同時に粒子を下階層へ渡していく

フラグの無いセルに入っている粒子を上 上の階層に渡していく

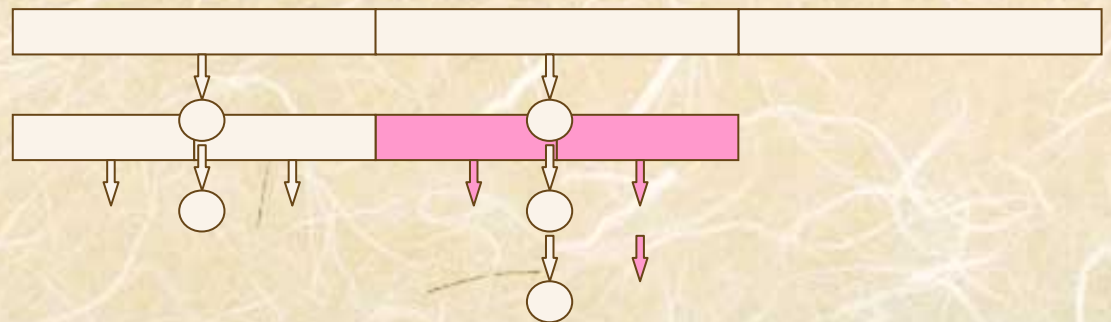
- 下の階層から上の階層へ以下の操作を行う
 - オープンセルだったが次のステップで消える、又はバッファセルになるセルオクテット配下のセル中の粒子のリンクリストを結合する
 - 結合した粒子のリンクリストを上上の階層へ転送する
 - オープンセルのままにいつづける粒子のデータを詰める
- 上の階層から下の階層への操作は次にまとめて行う

変更箇所

- 粒子のデータ構造(配列から延びるリンクリスト)の変更
 - 粒子の移動
 - セルは変更せずに、粒子のリンクリストのつなぎ替えをする
 - オープンセルにフラグを立てる
 - フラグの無いセルに入っている粒子を上階層に渡していく
 - セルを上階層から消したり足したりしていく、と同時に粒子を下階層へ渡していく

セルを上レベルから消したり足したりしていく、
と同時に粒子を下階層へ渡していく

- 上の階層から下の階層へ以下の操作を行う
 - 子セルオクテットがオープンセルオクテットであるセル中の粒子をリンクリストから取り出す
 - それら粒子が新しい子セルオクテット中のどのセルに入るかを調べる
 - 粒子を新しいセルに振り分ける



まとめ

- 32ビット整数を使う場合、粒子のインデックスにローカルのインデックスとPEのインデックス両方入れるのは限界である。
- ローカルのインデックスとPEのインデックスを分けることにした。
- これで、いよいよES向けチューニングを始められる。

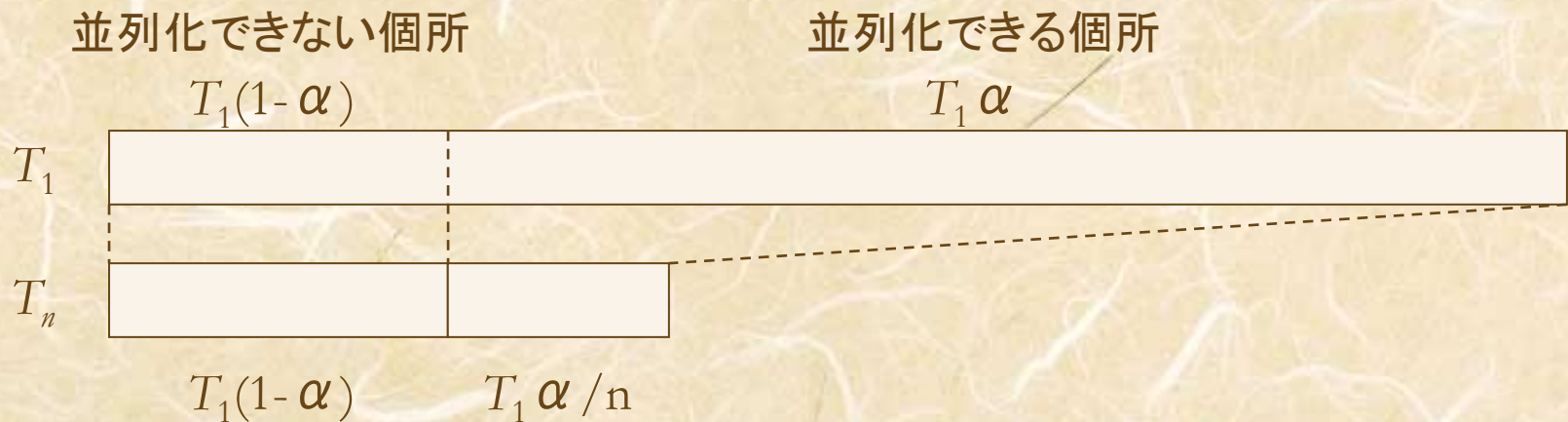
HPC用語の基礎知識

- 高速化率(speed-up)
 - 台数を増やして、何処まで速くなったか
 - $s_n = T_1 / T_n$
- 並列化効率
 - $p_n = s_n / n$
- 並列化率
 - 経過時間のうち並列実行された個所の割合
 - α

HPC用語の基礎知識

- Amdahlの法則(並列版)

$$T_n = T_1(1 - \alpha + \alpha/n)$$
$$= T_1(1 - \beta_n\alpha), \quad \beta_n = 1 - \frac{1}{n}$$



並列化率の求め方

- T_1 が計算できない場合は良くある
 - 異なるプロセッサ数で二回計測することにより、並列化率(α)を求めることができる。

$$T_n = T_1(1 - \beta_n\alpha)$$

$$T_m = T_1(1 - \beta_m\alpha)$$

$$\alpha = \frac{T_m - T_n}{\beta_n T_m - \beta_m T_n}$$

並列化効率の求め方

$$\begin{aligned} p_n &= \frac{s_n}{n} \\ &= \frac{1}{n} \left(\frac{T_1}{T_n} \right) \\ &= \frac{1}{n} \left(\frac{1}{1 - \beta_n \alpha} \right) \\ &= \frac{1}{n} \left(\frac{\beta_m T_n - \beta_n T_m}{\beta_m T_n - \beta_n T_n} \right) \end{aligned}$$

ある並列化効率を満たす台数

$$p_{min} \leq p_n = \frac{1}{n(1 - \beta_n \alpha)}$$

$$n \leq \frac{p_{min}^{-1} - \alpha}{1 - \alpha} = n_{max}$$

要求されている並列化率

$$n \leq \frac{p_{min}^{-1} - \alpha}{1 - \alpha}$$

$$\alpha \geq \frac{n - p_{min}^{-1}}{n - 1}$$

$$\geq 99.2 \quad (n = 128, p_{min} = 0.5)$$

白色雑音問題での計測

- $N=256^3$

Number of Nodes	Time per step (sec)
8	7.725
32	2.510

$$\alpha = 0.986$$

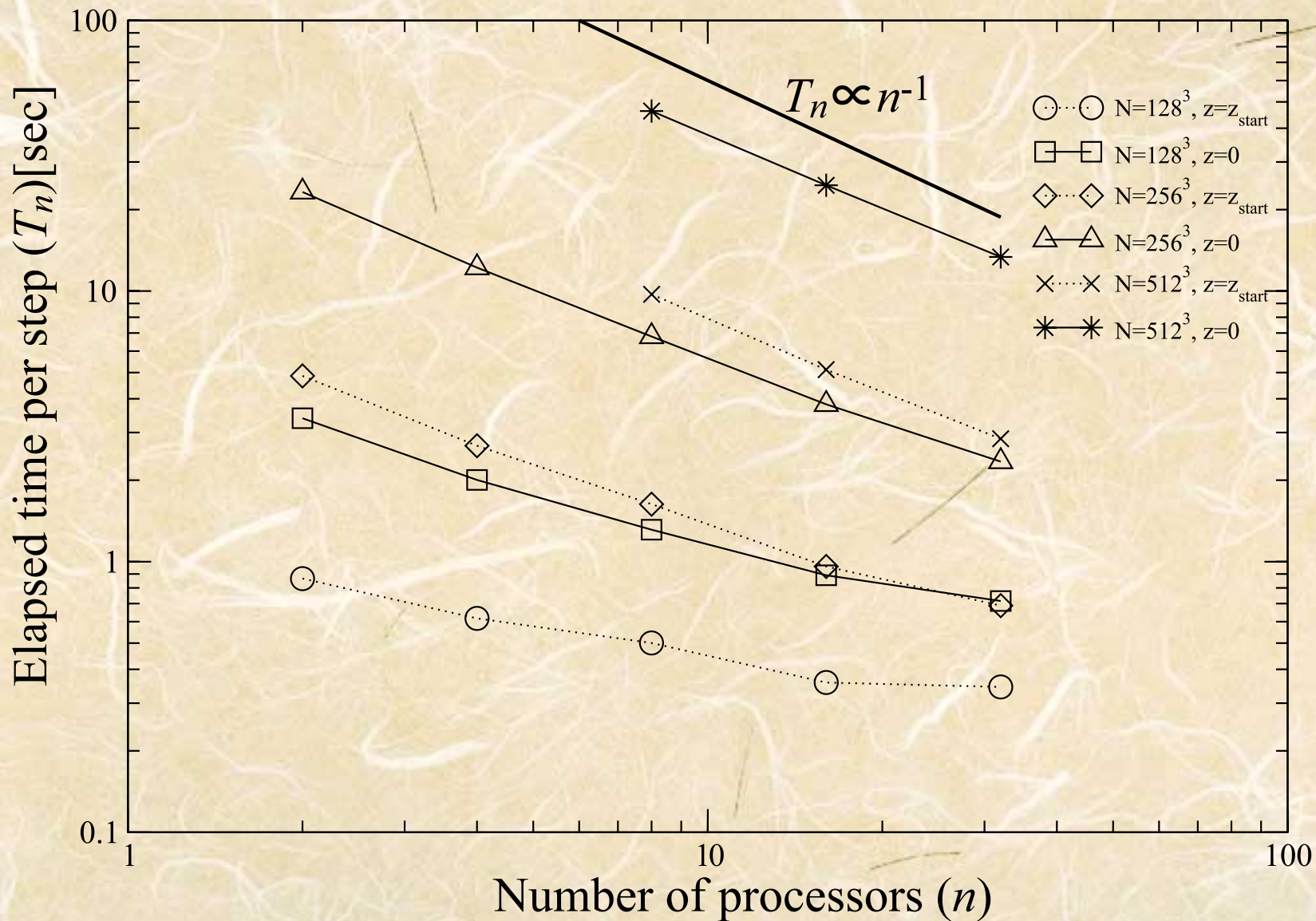
$$p_{32} = 0.702$$

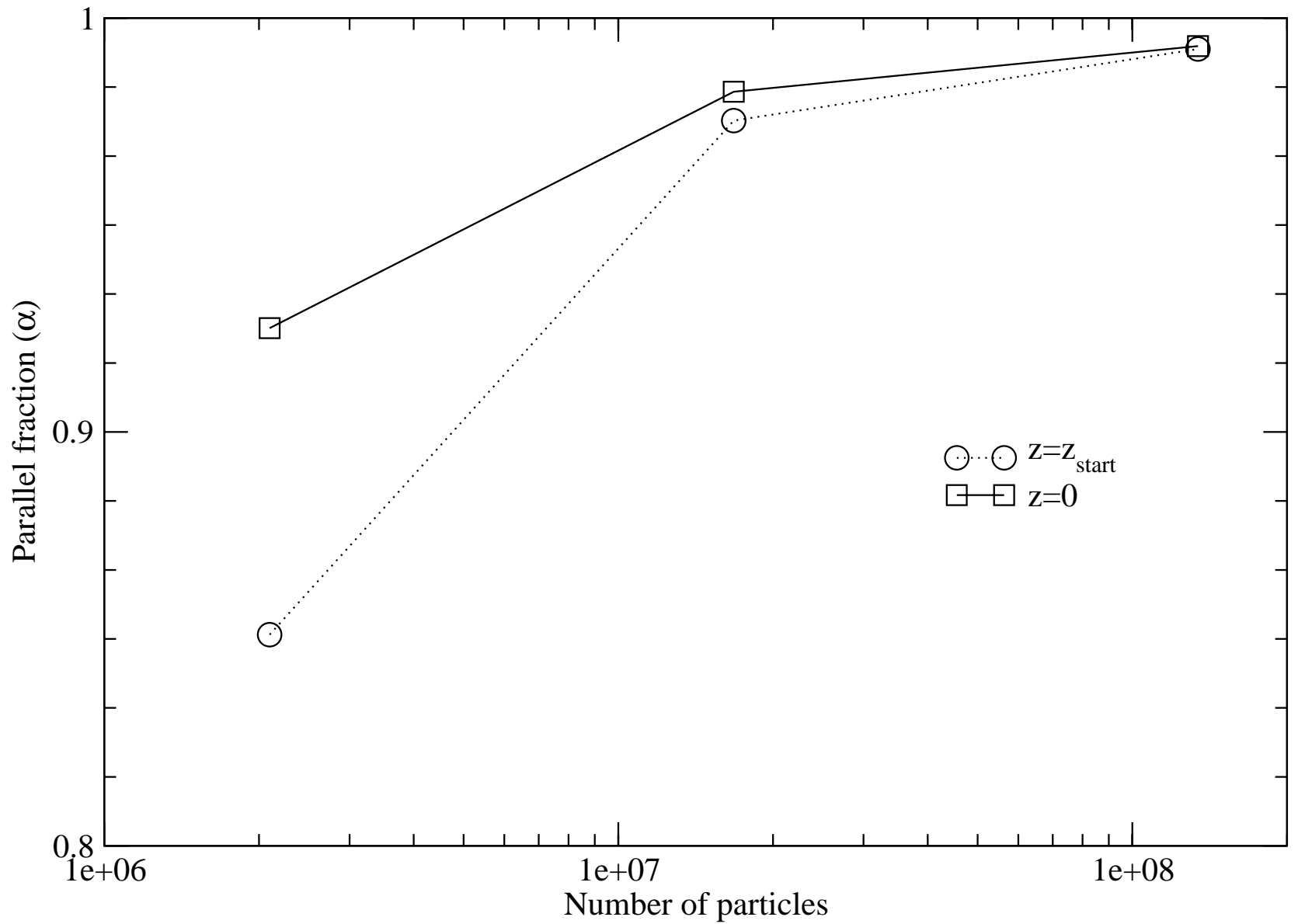
$$n_{max} \sim 74 \quad (p_{min} = 0.5)$$

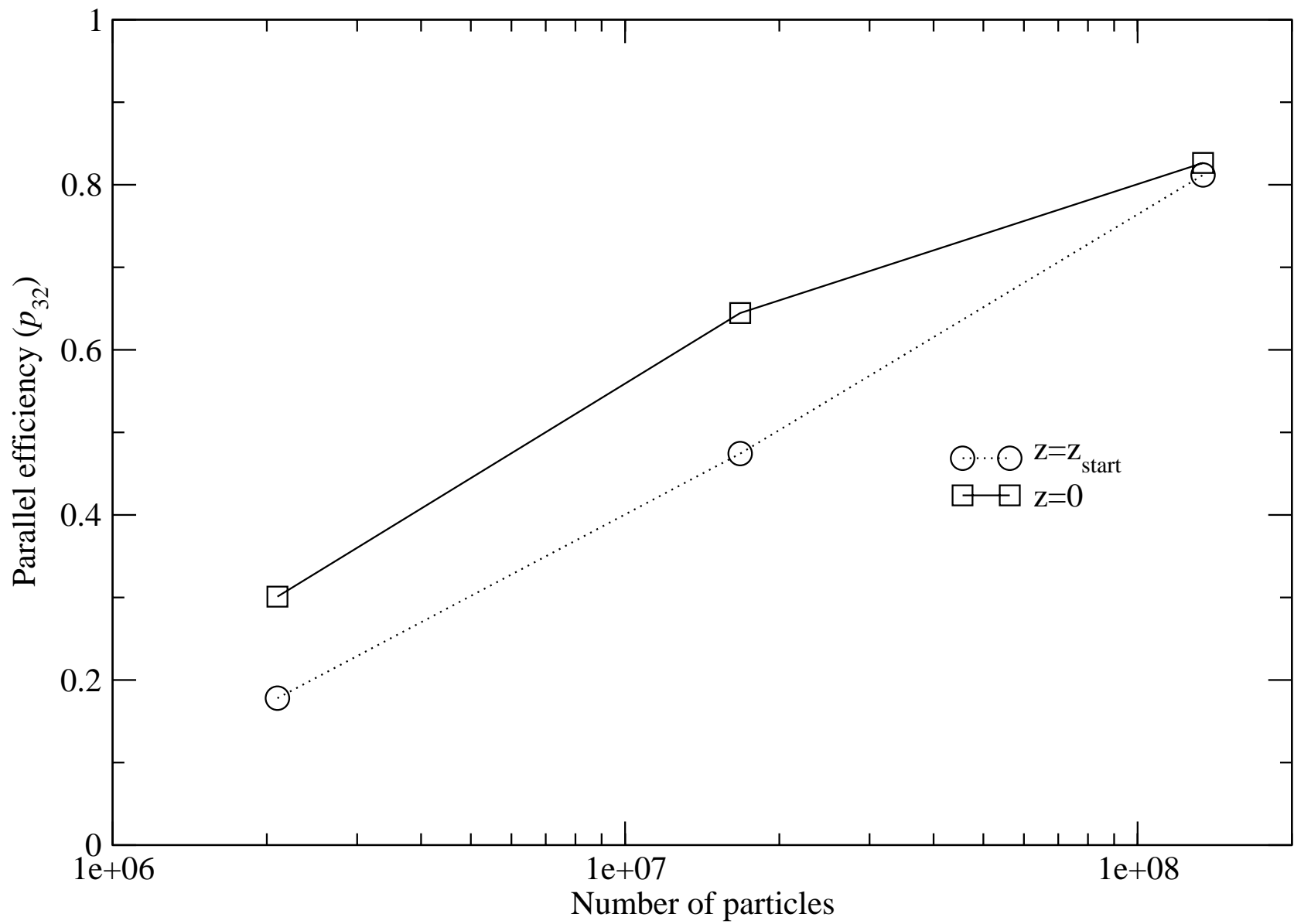
計時解析 I

- Λ CDM simulations
 - $\Omega = 0.3, \lambda = 0.7, h = 0.7, n = 0.9, \Omega_b = 0.048$
- 使用機種
 - VPP5000@ADAC/NAOJ
- 粒子数
 - $N = 128^3, 256^3, 512^3$
- CPU数
 - $N_{\text{proc}} = 2, 4, 8, 16, 32$

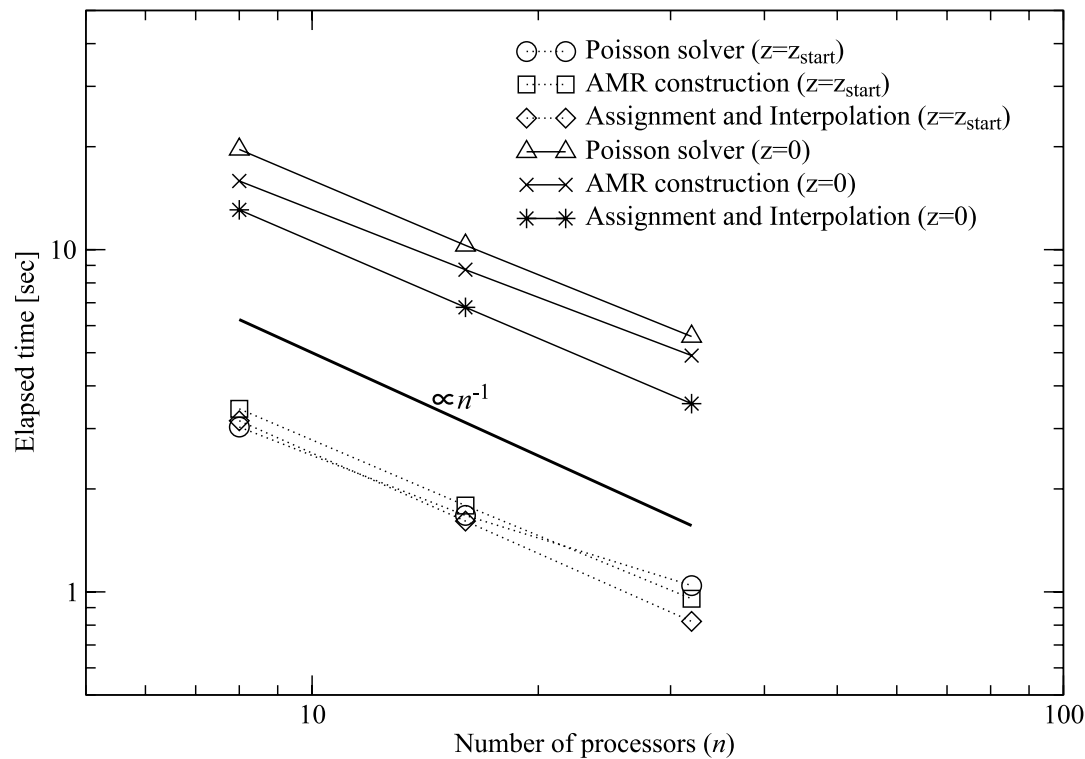
計時解析 II







各ルーチン毎の計時解析



- AMR construction 及び Assignment and Interpolation では粒子データを取り扱うが、これら二つのケースでも理想的な状況に近いスピードアップが実現できている。

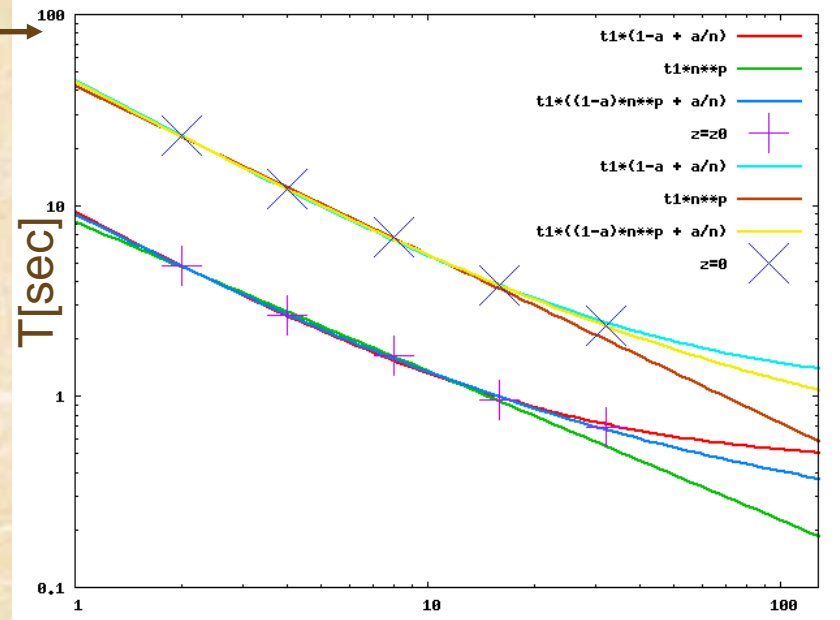
Λ CDMシミュレーションの経過時間を以下の三つの関数型でフィッティングした。

$$T_n = T_1(1 - \alpha + \alpha/n)$$

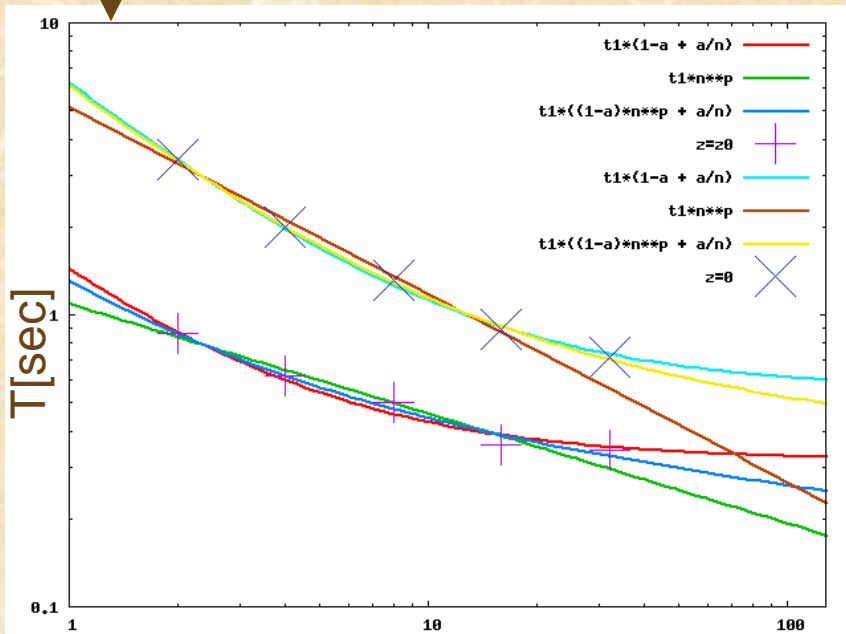
$$T_n = T_1 n^p$$

$$T_n = T_1((1 - \alpha)n^p + \alpha/n)$$

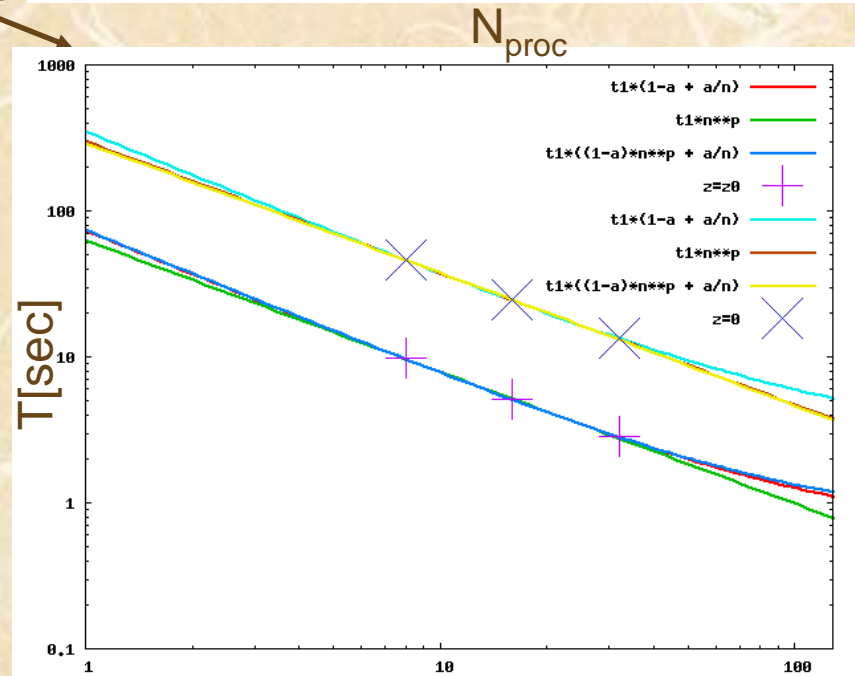
$N=256^3$



$N=128^3$



$N=512^3$



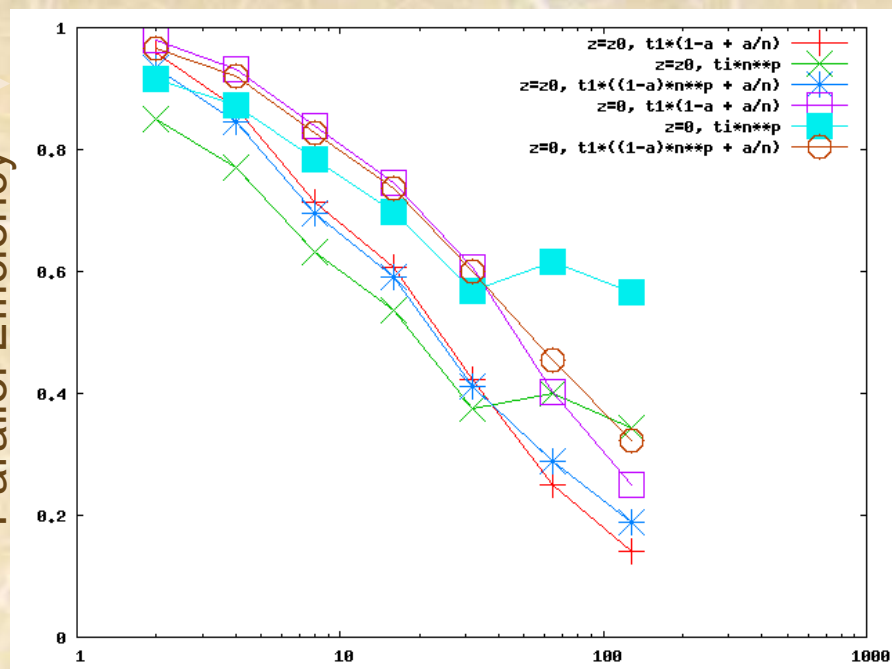
N_{proc}

N_{proc}

$N=256^3 \rightarrow$

前項のフィッティングから T_1 を見積もり、
 並列化効率 $p_n = (T_1 / (nT_n))$ を見積もった。
 また、 T_{64} , T_{128} も前項のフィッティング
 から見積り、 p_{64} , p_{128} を算出した。その
 結果から、 $n \leq 128$ まで、どのような見積
 りをしても、50%以上の並列化効率を維持
 できるものと予測される。

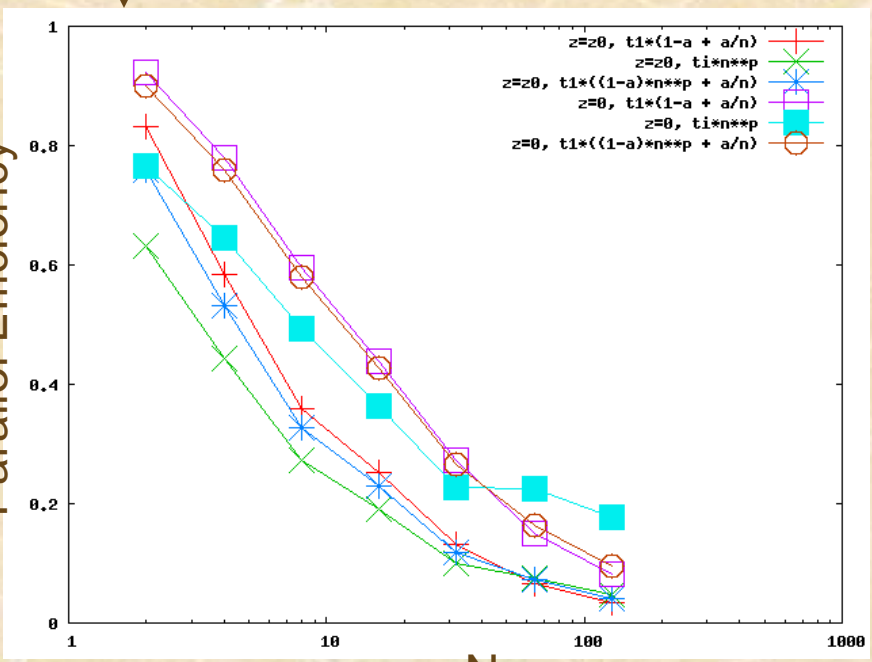
Parallel Efficiency



$N=128^3$

$N=512^3$

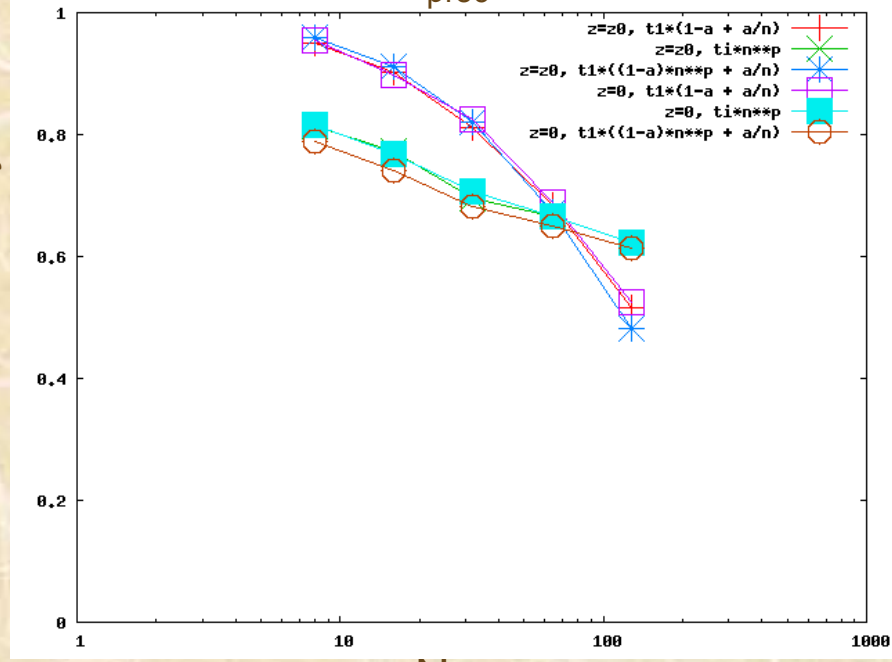
Parallel Efficiency



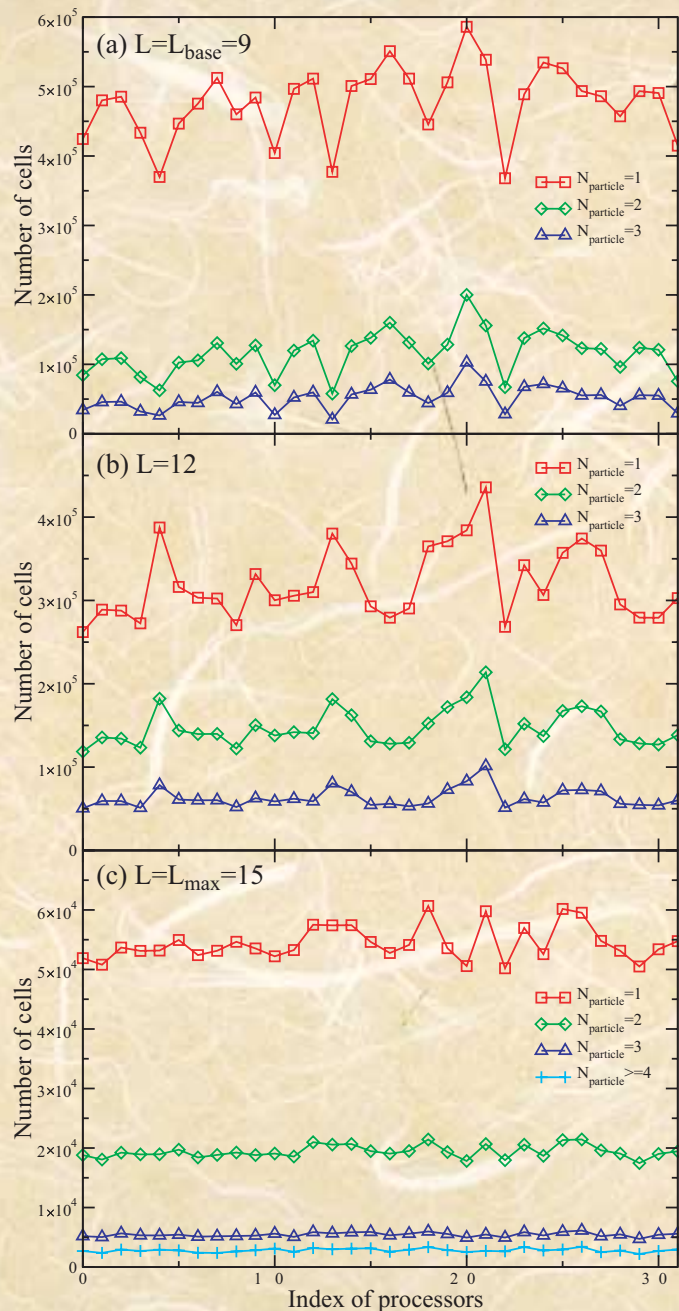
N_{proc}

N_{proc}

Parallel Efficiency



N_{proc}

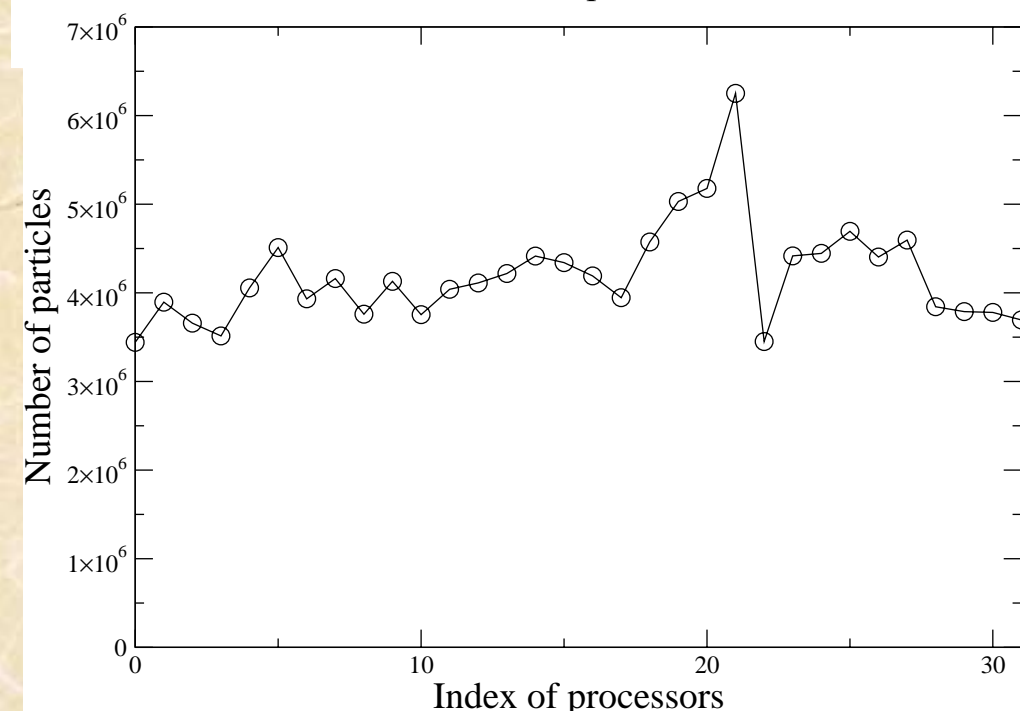
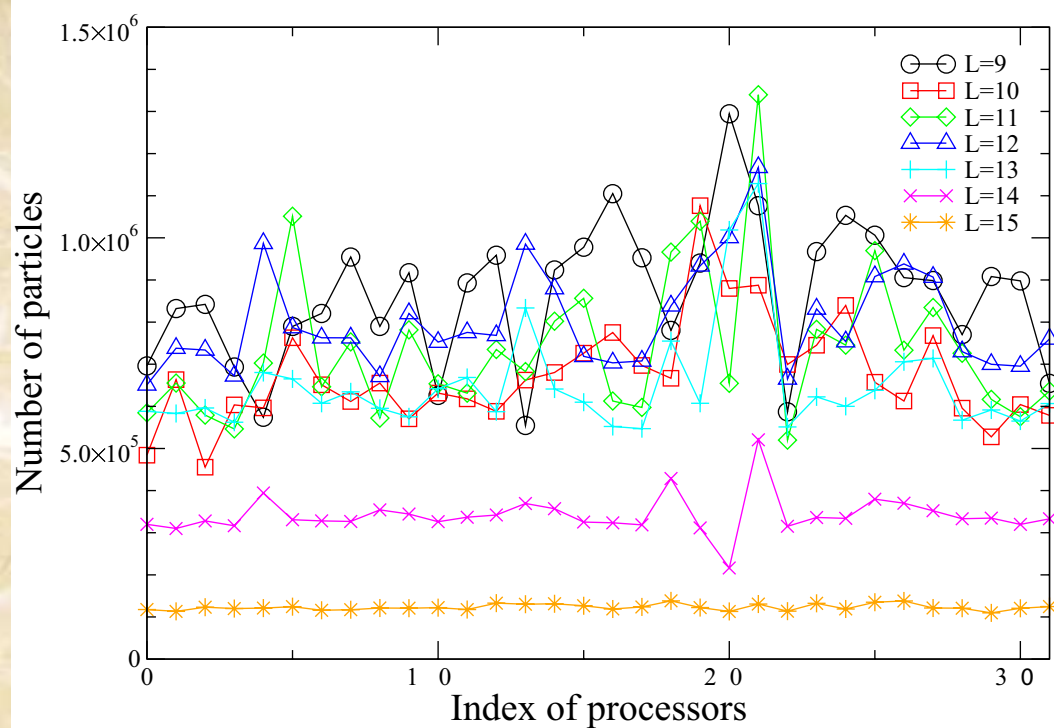


粒子分割

- 上段が一番粗い階層の格子数、下段が一番細かい階層の格子数、中段はその中間の階層の格子数を表している。
- 赤、緑、青、水色は、粒子を一つ含む格子数、二つ含む格子数、三つ含む格子数、四つ以上含む格子数をそれぞれ表している。
- 粒子分割は深い階層の方が浅い階層より均等に分割されている。

● 粒子分割

- 上段は各階層毎の各CPUの担当する粒子数、下段は各CPUの全粒子数を表している。
- 各格子に入る粒子数に制限のない一番細かい格子が均等に分割できないと思われるが、実際には、逆に最も良く均等分割されている。
- 均等分割から、最大で約50%の不均等が生じている。



誤差解析 I

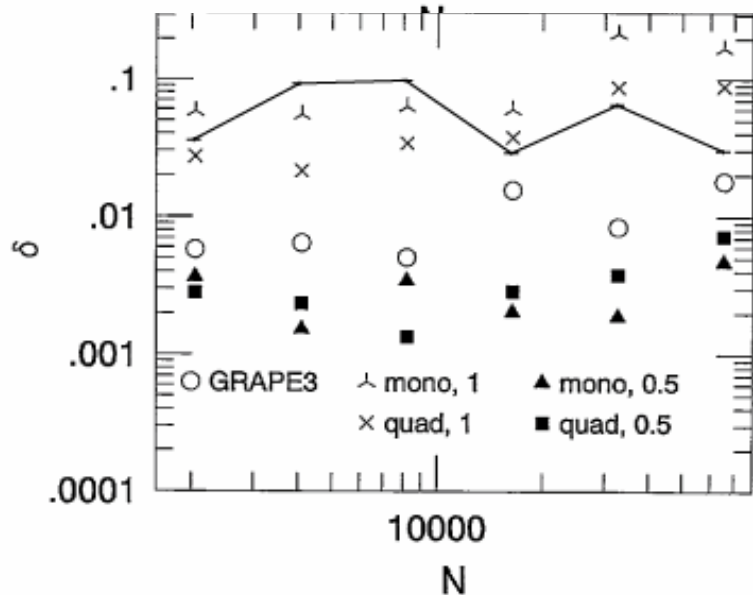
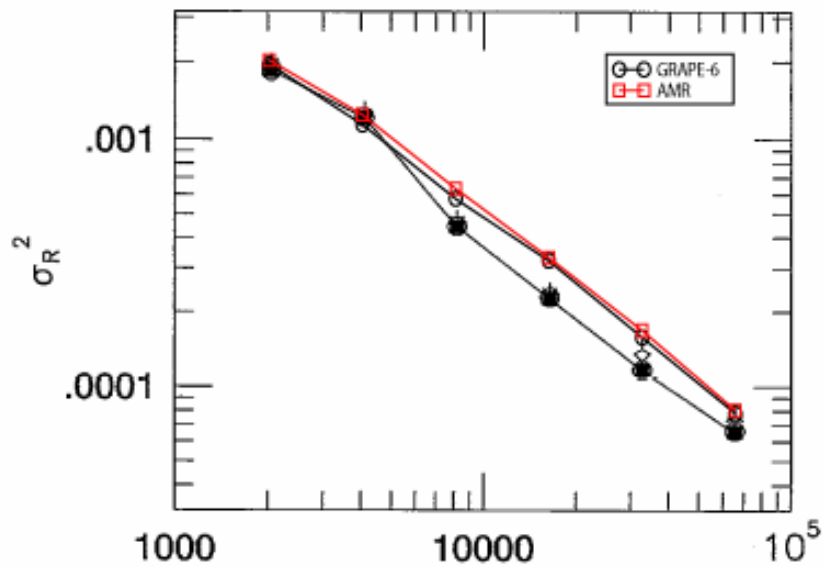
- 共同研究者: 牧野淳一郎
- Hernquist, Hut, & Makino (1993; HHM93)
 - 定常解では、全体の力学的エネルギーだけでなく、個々の粒子の力学的エネルギーが保存する
 - N体シミュレーションで Plummer 解を計算し、個々の粒子の力学的エネルギーの変化を調べた

$$\sigma_R = \left(\frac{1}{N} \sum_{i=1}^{i=N} \left(\frac{E_{i,t} - E_{i,0}}{E_{i,0}} \right)^2 \right)^{1/2}$$

誤差解析 II

- $G=1, M=1, E=-1/4$
 - $\varepsilon = 1/32, \delta t=2$
 - $t=0$ から $t=8$ までの4区間について平均を取る
- スケーリング
 - AMRN体コードでは、最も細かい格子間隔を1とする単位系を採用。
 - 以下のスケーリングをした。
 - $G=1, E=-1/4, R = 32 \frac{3\pi G}{64 |E|} \quad M = \left(\frac{64 |E| R}{3\pi G} \right)^{1/2}$
 - $T \propto M^{5/2} |E|^{-3/2}$.

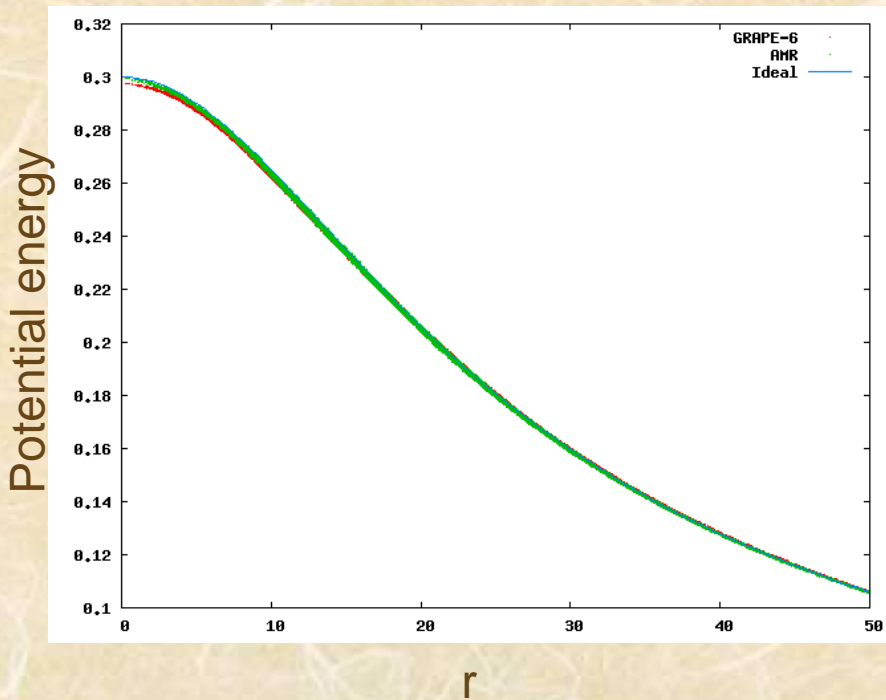
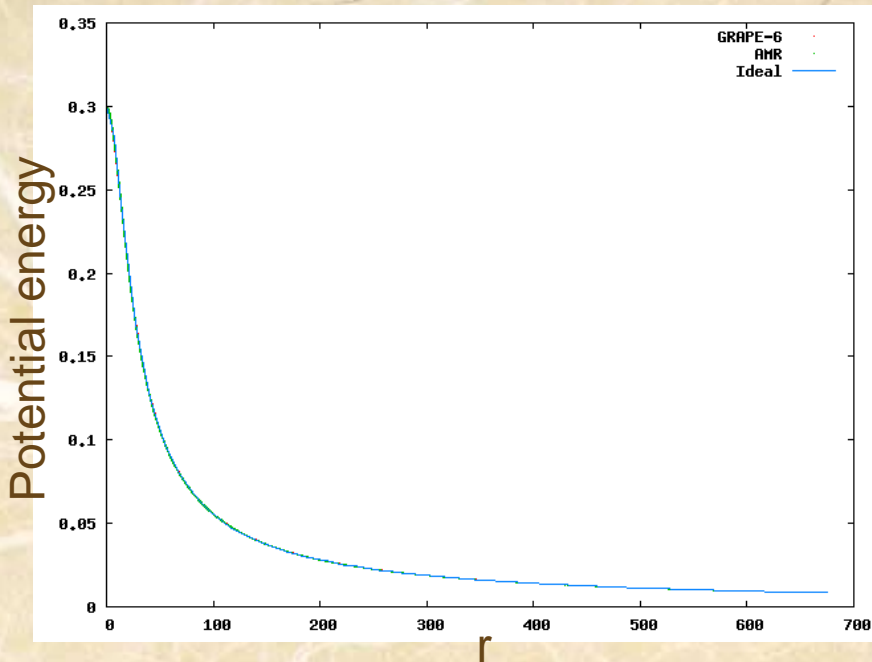
誤差解析 III



- HHM93の図の上に、今回の結果を重ね合わせた(上図)。およそ近い値の誤差になっている。

● $\delta = \left(\frac{\sigma_R}{\sigma_{R, \text{GRAPE-6}}} \right)^2 - 1$ の上にプロットしたものである(下図)。AMRの結果はツリー法の $\theta = 1.0$ の結果に近い結果となった。

ポテンシャル



- 大局的にはGRAPE-6の結果と、AMRの結果に差は無い。N=65536.

- 中心付近を拡大すると、GRAPE-6よりAMRの結果の方が解析解に近い。

$$\sigma^2 = \frac{1}{N} \sum_{i=0}^N \left(\frac{\phi_i - \phi(r)}{\phi(r)} \right)^2$$

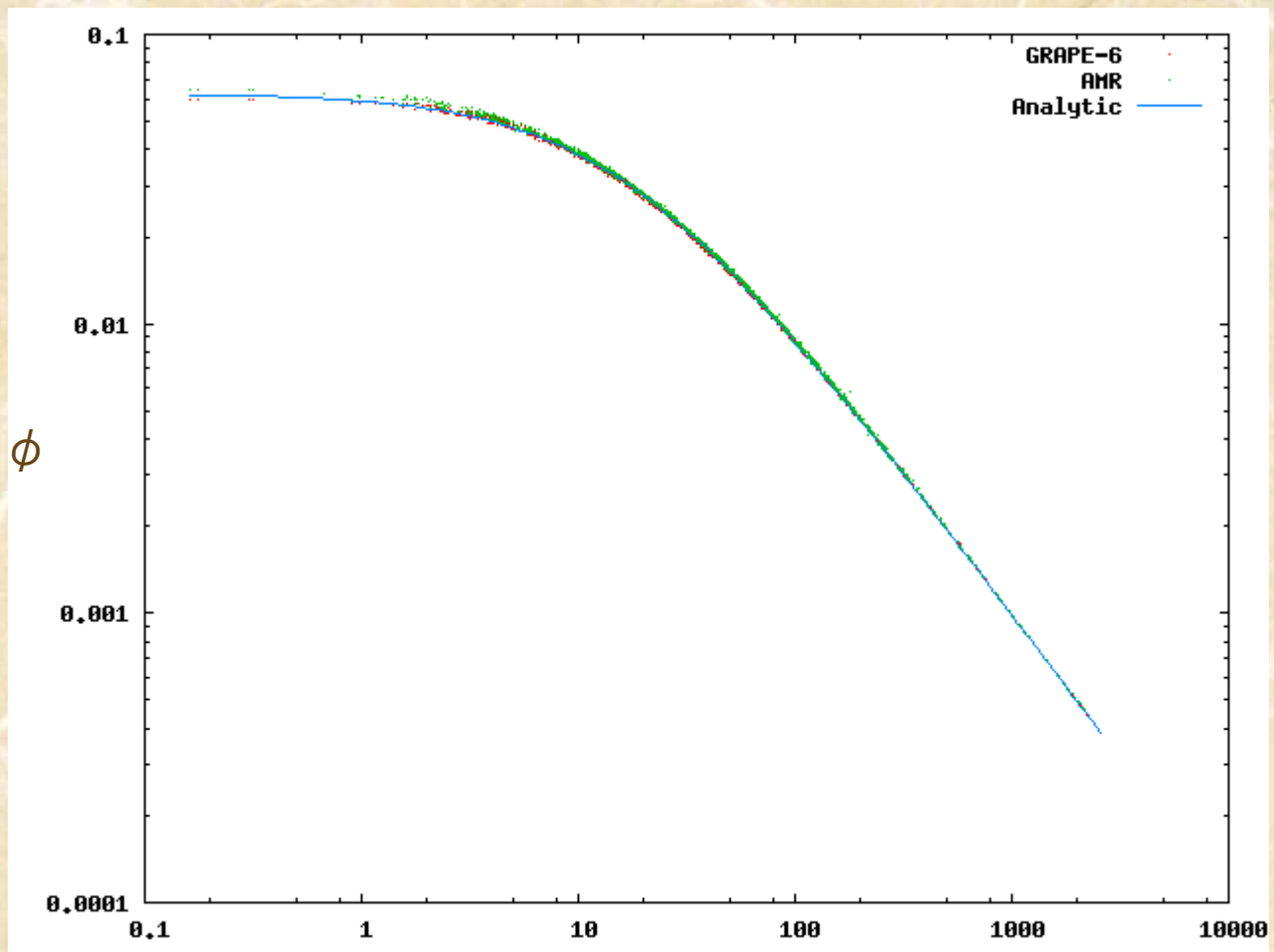
- GRAPE-6: 1.91367e-05
- AMR: 1.51096e-05

Hernquist Model

$$\rho(r) = \frac{M a}{2\pi r} \frac{1}{(r+a)^3}$$
$$M(r) = M \frac{r^2}{(r+a)^2}$$

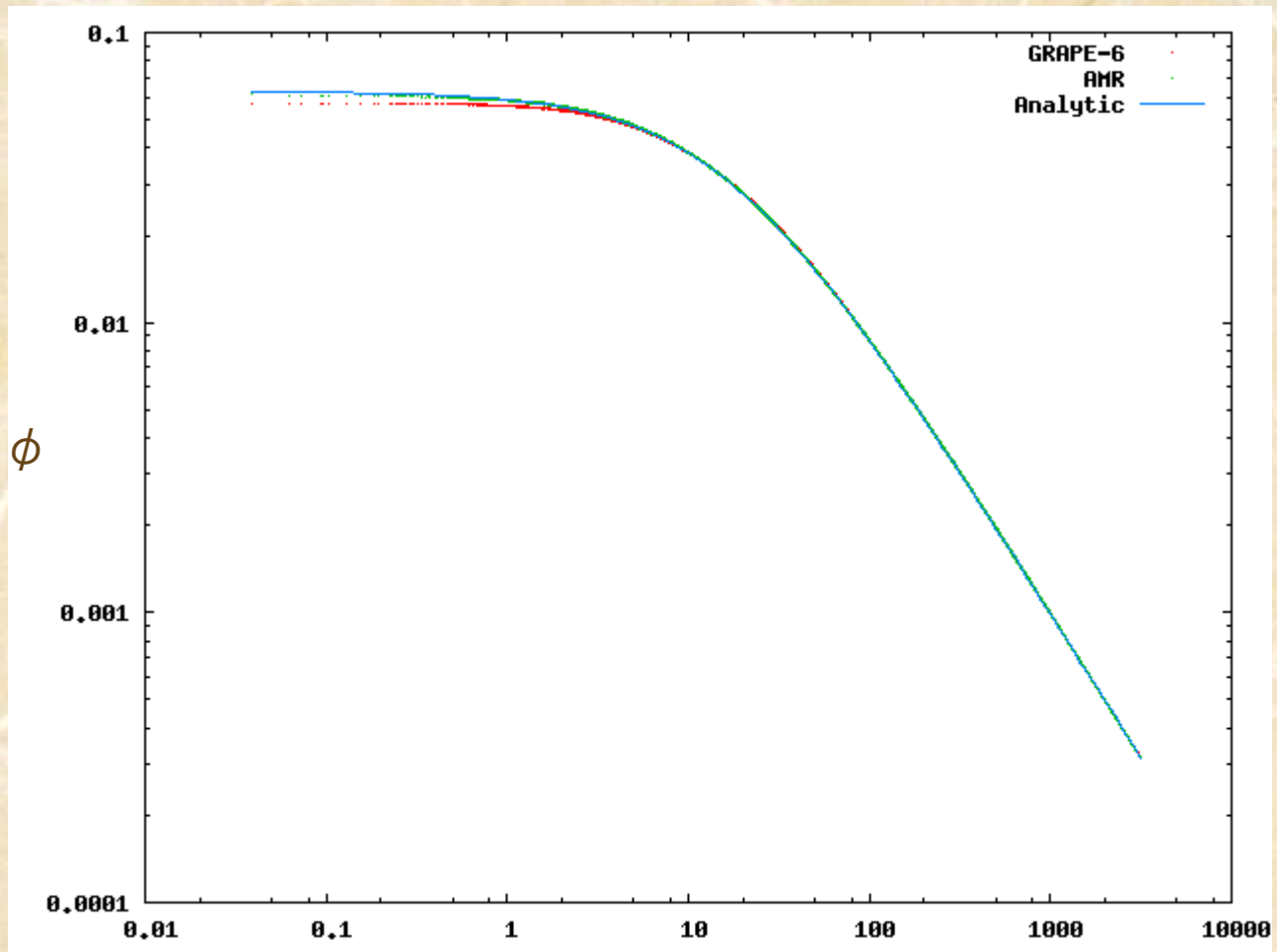
- NEMOのmkhernquistを使って初期データを生成し、外側の粒子 ($M(r)/M > 0.99$) を取り除き、スケーリングを行う。(a=16, M=1)
- AMR
 - $L_{\text{BASE}}=8, L_{\text{DYNR}}=15$

$N=2048$



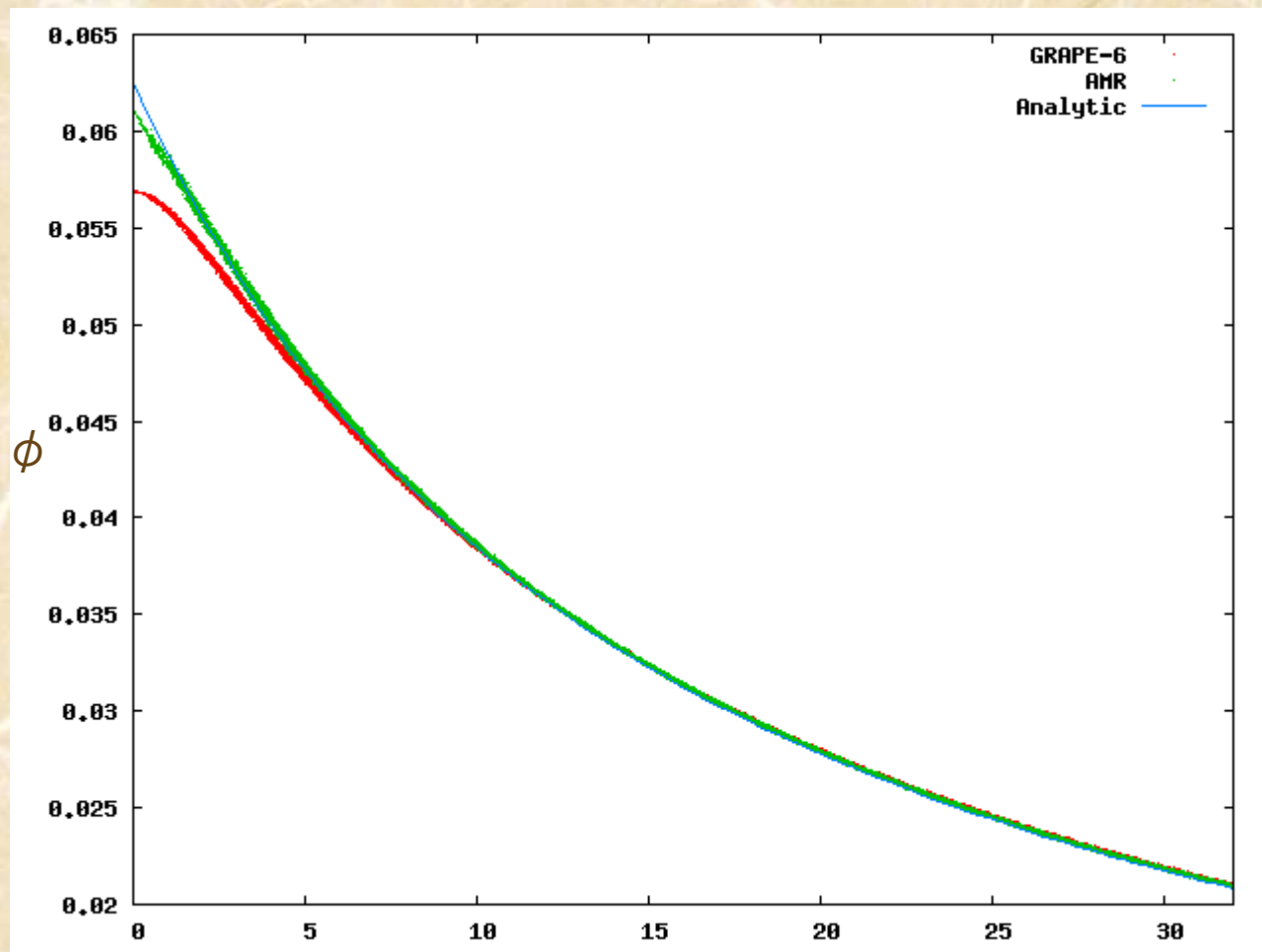
r

$N=65536$



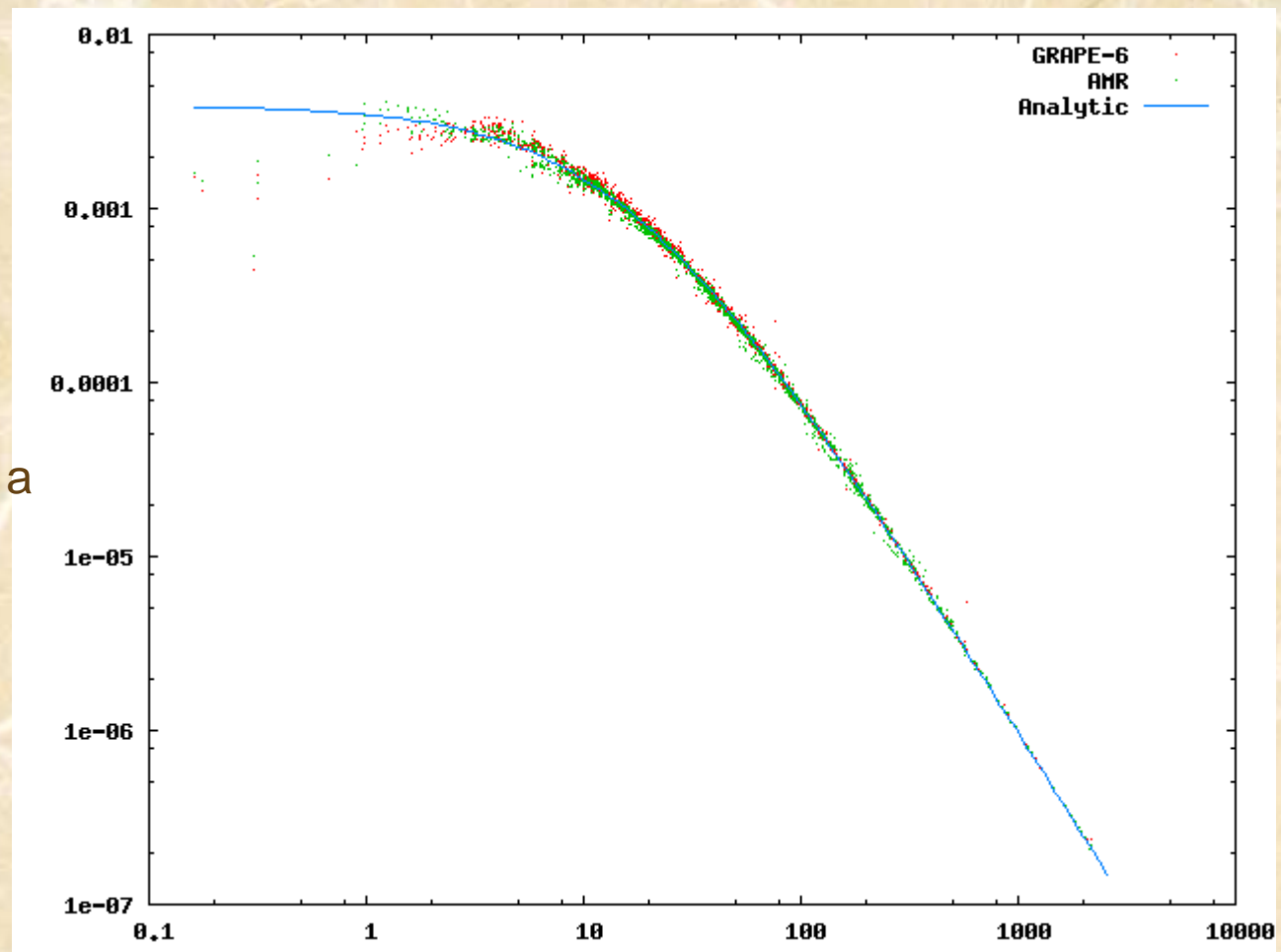
r

$N=65536$



r

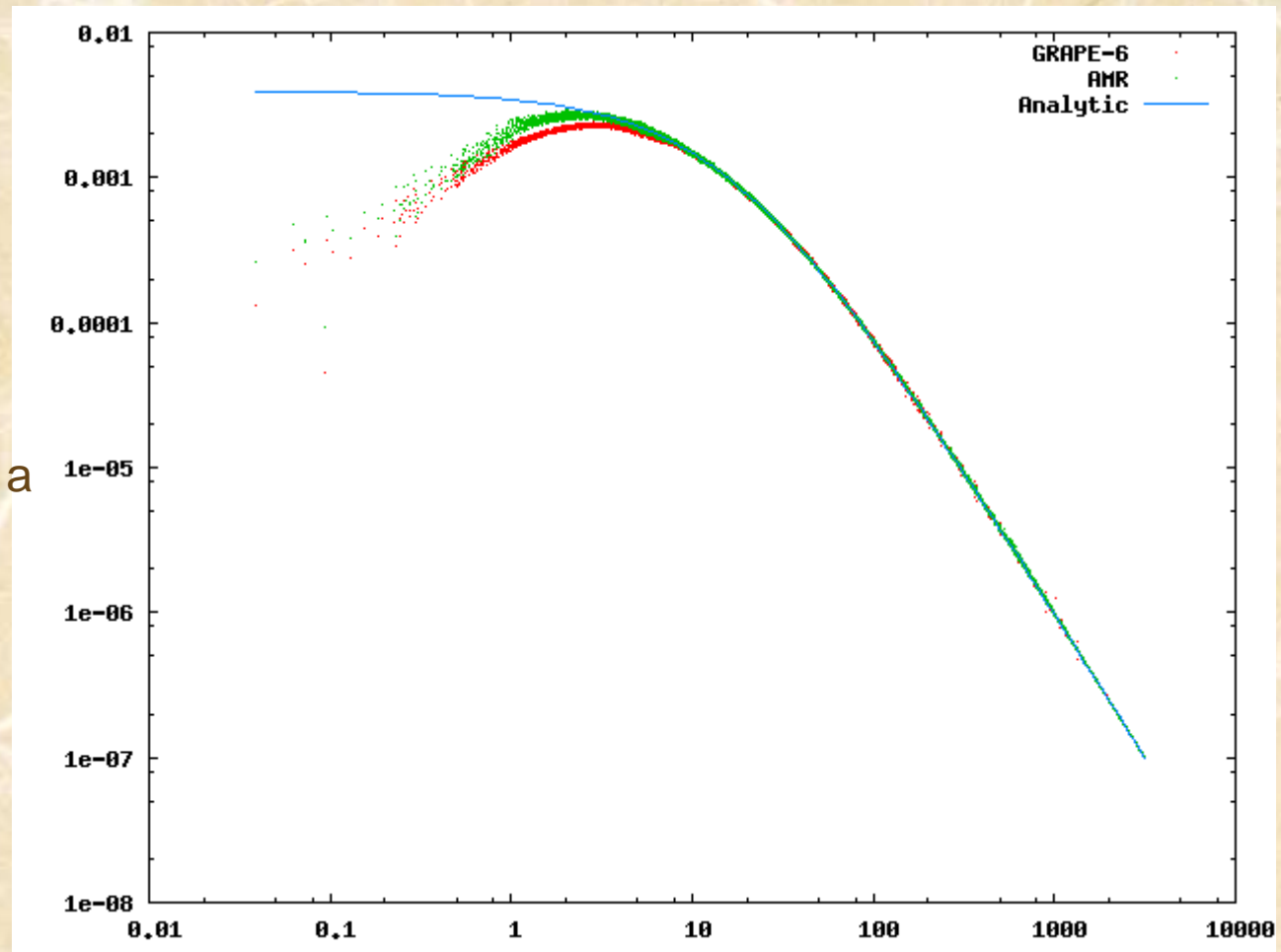
$N=2048$



a

r

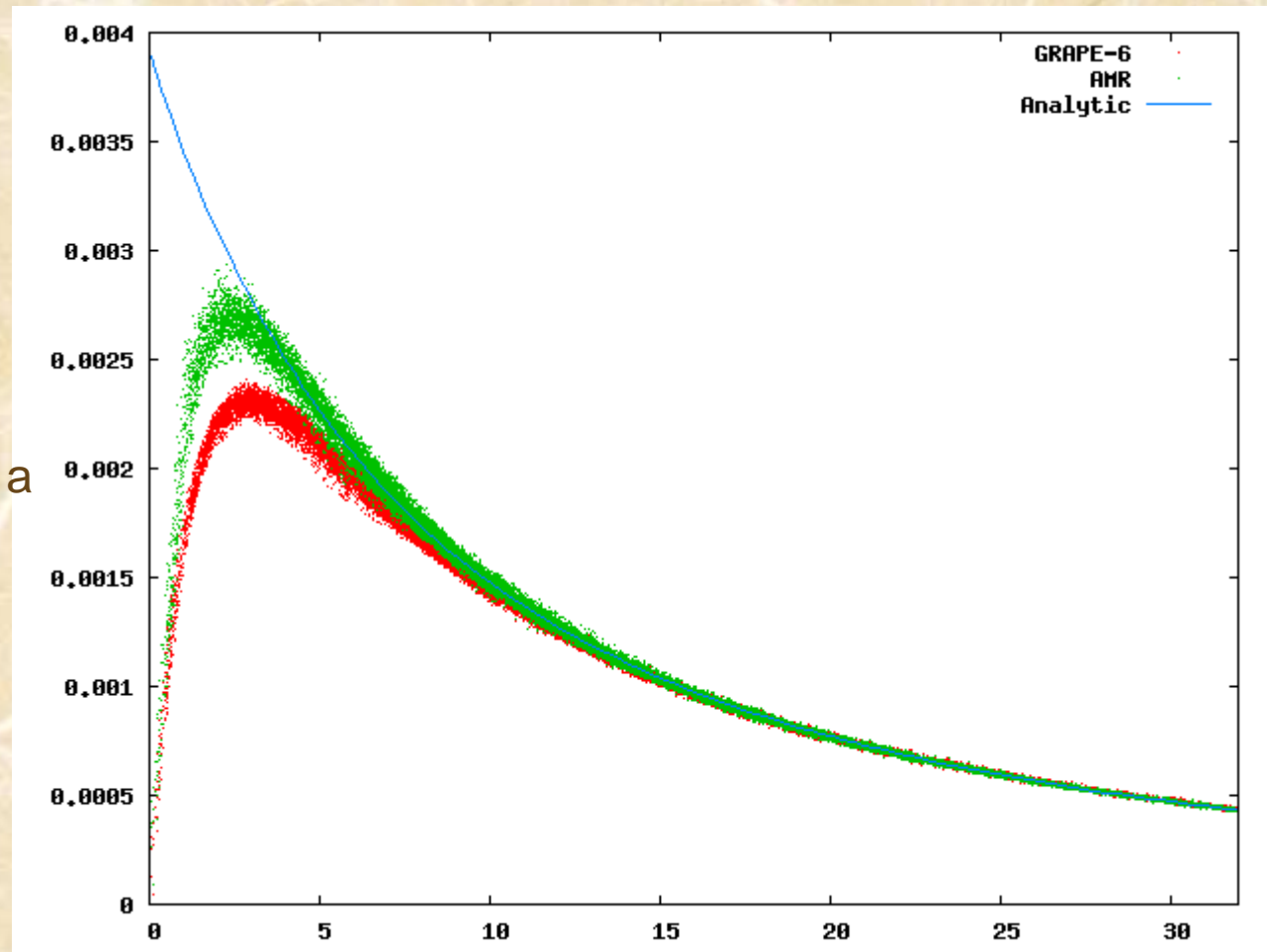
$N=65536$



a

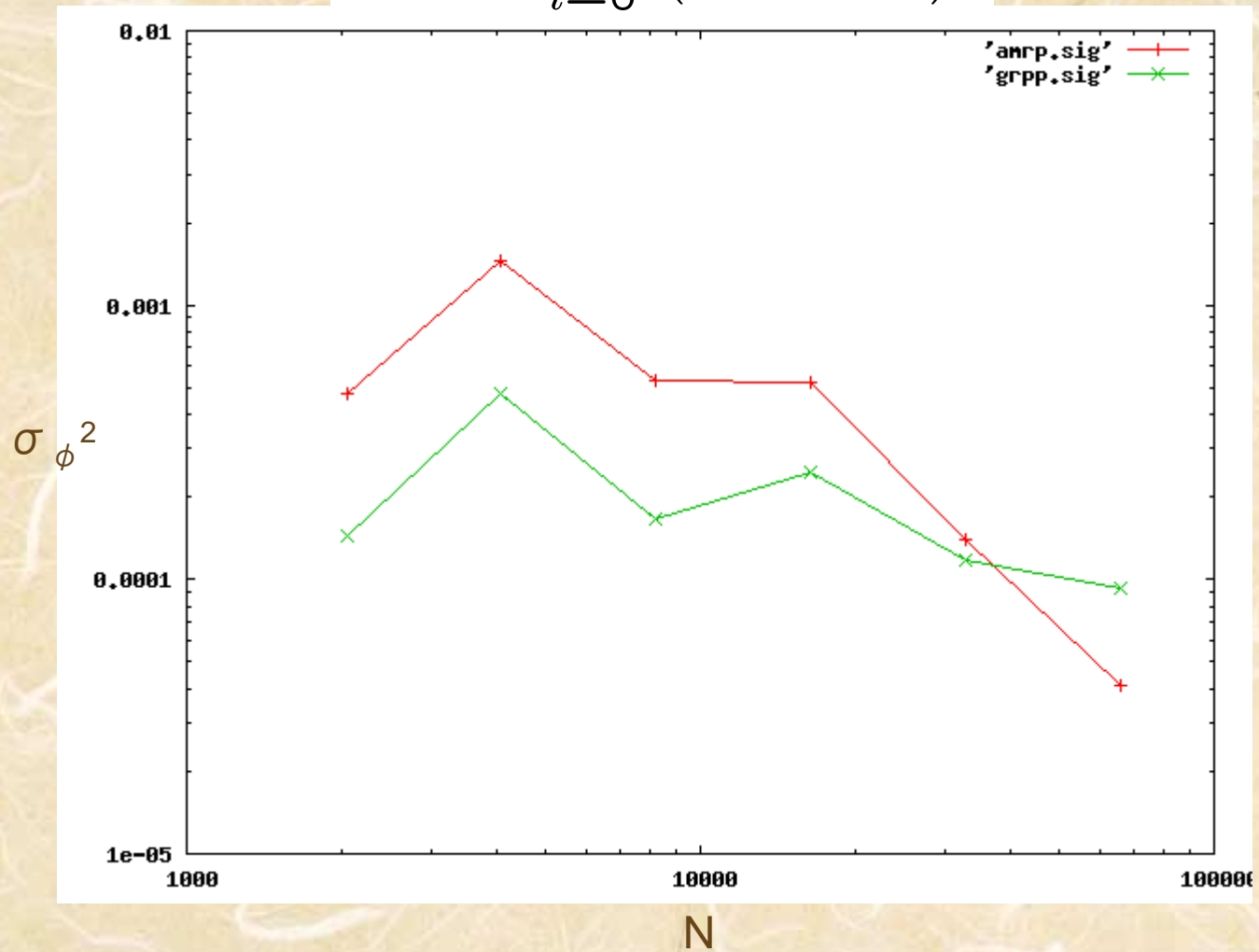
r

$N=65536$

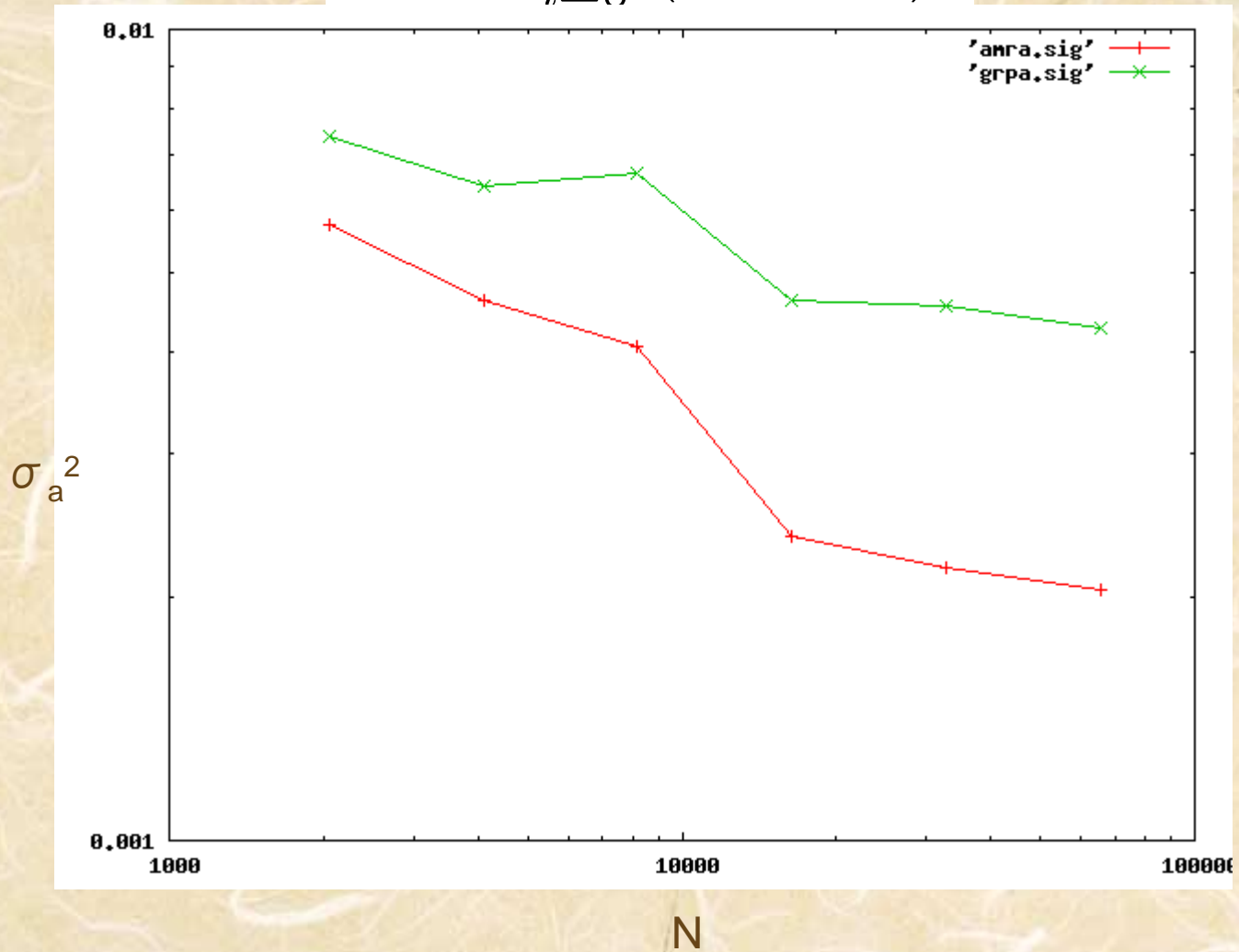


r

$$\sigma_{\phi}^2 = \frac{1}{N} \sum_{i=0}^N \left(\frac{\phi_i - \phi(r)}{\phi(r)} \right)^2$$



$$\sigma_a^2 = \frac{1}{N} \sum_{i=0}^N \left(\frac{a_i - a(r)}{a(r)} \right)^2$$



まとめ

- Hernquist モデルの力とポテンシャルの誤差を求めた。
 - 力、ポテンシャルの両方で、中心付近ではAMRコードの方が解析解に近いふるまいをする。
 - ポテンシャルについては、粒子が少ない場合はGRAPEの方が誤差が小さい。
 - 力については、どの粒子数でもAMRコードの方が誤差が小さい。
 - 何故、力とポテンシャルで結果が変わるのか?

この先危険(再)

- 地球シミュレータ固有の話です
- 飛ばしますか、そうですか

並列化チューニング

- メモリによる制限のため、 1024^3 体計算には64ノード(512AP)が必要
- しかし、並列化でつまずき相談員に相談する
- そうしたら、ESでは通信中に計算を実行することが出来ていないことが判明。I系非同期通信を全てSendrecvに置き換えることに。

標準の集団通信を使いましょう

- MPIに含まれる集団通信で置き換えられる個所は置き換える。
 - for (jproc=0; jproc<N_PROC; jproc++)
MPI_Irecv(nrip+jproc,1,MPI_INT,jproc
TAG_POST_MOD_SIEVE_HM_NI,
MPI_COMM_WORLD, rnireq+jproc);
 - for (jproc=0; jproc<N_PROC; jproc++)
MPI_Issend(tnsip+jproc,1,MPI_INT,jproc,
TAG_POST_MOD_SIEVE_HM_NI,
MPI_COMM_WORLD, snireq+jproc);
 - MPI_Waitall (N_PROC, rnireq, MPI_STATUSES_IGNORE);
 - MPI_Waitall (N_PROC, snireq, MPI_STATUSES_IGNORE);
 - を以下のように置き換える
 - MPI_Alltoall (tnsip, 1, MPI_INT, nrip,1,MPI_INT,MPI_COMM_WORLD);

I系ではなくSendrecvを使いましょう

- 集団通信で置き換えられない、MPI_Issend, MPI_Irecv, MPI_Wait の組み合わせは MPI_Sendrecv で置き換える

```
• for (jproc=0; jproc<nsiproc; jproc++){
•     kproc = lsiproc[jproc];
•     MPI_Issend (slp+hsip[kproc], nsip[kproc], MPI_INT, kproc,
•                 TAG_POST_MOD_SIEVE_HM_I, MPI_COMM_WORLD, ireq->sreq+jproc);
• }
• for (jproc=0; jproc<nriproc; jproc++){
•     kproc = lriproc[jproc];
•     MPI_Irecv (rlp0+hrip[kproc], nrip[kproc], MPI_INT, kproc,
•                 TAG_POST_MOD_SIEVE_HM_I, MPI_COMM_WORLD, ireq->rreq+jproc);
• }
```

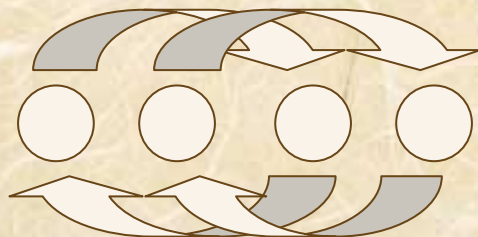
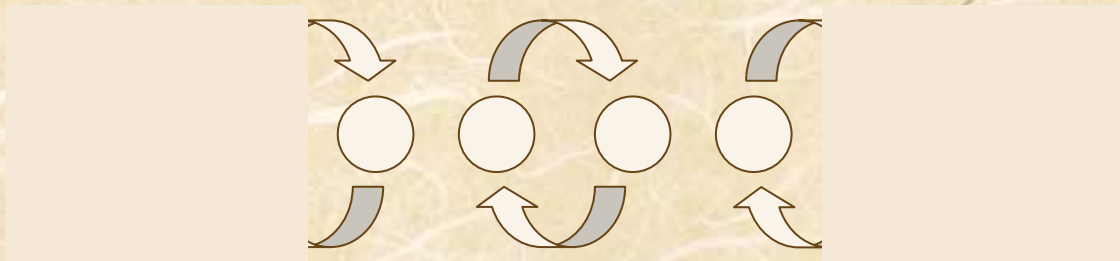
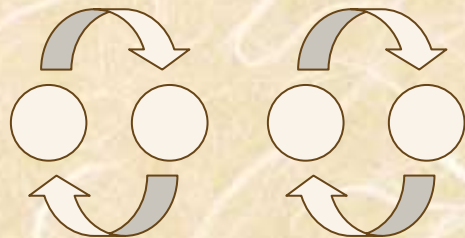
を以下のように置き換える

```
• nsrproc = 0;
• for (jproc=0; jproc<N_PROC; jproc++){
•     kproc = lkproc2[jproc];
•     if (nsip[kproc] > 0 || nrip[kproc] > 0) lsrproc[nsrproc++] = kproc;
• }
• for (jproc=0; jproc<nsrproc; jproc++){
•     kproc = lsrproc[jproc];
•     MPI_Sendrecv (slp+hsip[kproc], nsip[kproc], MPI_INT, kproc,
•                 TAG_POST_MOD_SIEVE_HM_I, rlp0+hrip[kproc], nrip[kproc],
•                 MPI_INT, kproc, TAG_POST_MOD_SIEVE_HM_I, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
• }
```

- 一見なんてことないように見えるが、実は、lkproc2 を決めるのは自明ではない。

Sendrecvペアの作り方

- 前と通信するのか、後ろと通信するのか、それが問題だ
- $N_p=4$



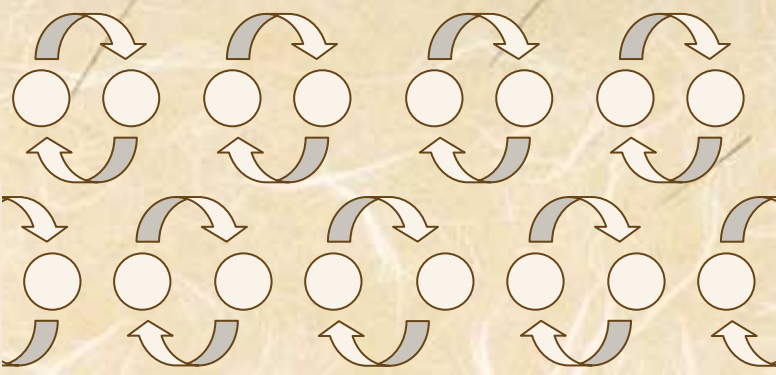
Sendrecvペアの作り方

- lkproc2の構築箇所 (!(N_PROC & 1))

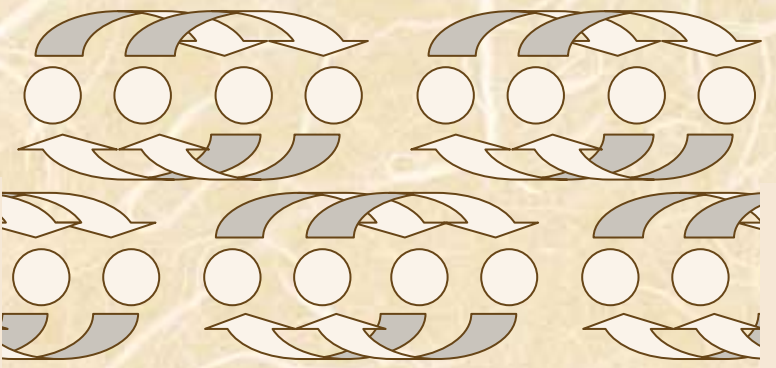
- lkproc2[0] = ikproc;
- for (jproc=1; jproc<N_PROC/2; jproc++){
- kproc = 1;
- while (!(jproc & kproc)) kproc <<= 1;
- if ((ikproc / kproc) & 1){
- lkproc2[jproc*2-1] = lkproc[N_PROC-jproc];
- lkproc2[jproc*2] = lkproc[jproc];
- } else {
- lkproc2[jproc*2-1] = lkproc[jproc];
- lkproc2[jproc*2] = lkproc[N_PROC-jproc];
- }
- }
- lkproc2[N_PROC-1] = lkproc[N_PROC/2];

$N_p=8$

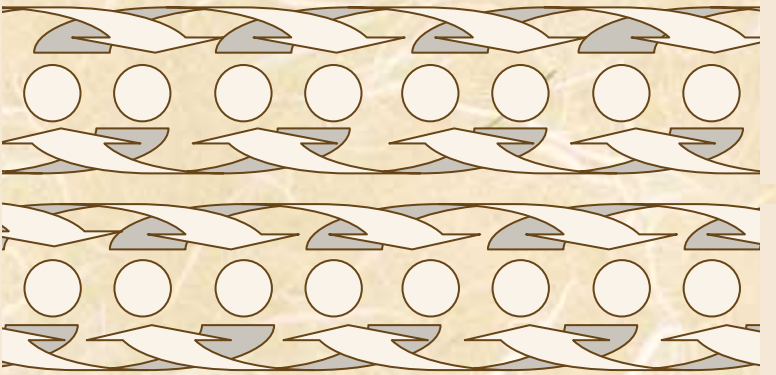
$j_{proc}=1$



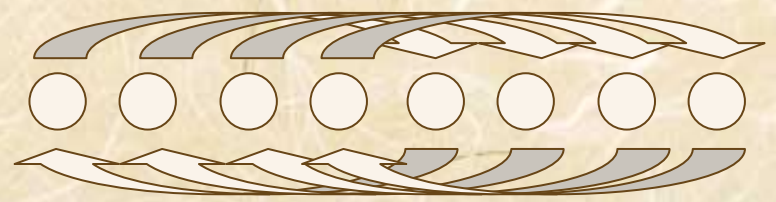
$j_{proc}=2$



$j_{proc}=3$



$j_{proc}=4$



もっとエレガントなペアの作り方

- 似鳥君の提案

- for (jproc=0; jproc<N_PROC; jproc++)
 lkproc2[jproc] = ikproc ^ jproc;
- $A \wedge J = B$ なら $B \wedge J = A$ なのでペアになる。
- ペア間の距離は一定ではない
 - PE間の通信量がMorton順序の距離に応じて減少するならば、通信待ちが生ずることになるが、実際に試していないので何ともいえない。

時間計測

日時	8/04:12	9/04:14	9/05:15	9/05:15	9/05:15
粒子数	256 ³	512 ³	512 ³	512 ³	512 ³
ステップ	20	20	20	20	80
32AP _(sec)	65.189	274.364	271.311	300.848	1116.7
64AP _(sec)	58.355	183.577	205.549	193.414	679.557
α	0.8946	0.9843	0.9679	0.9877	0.9914
MAX-AP	10.49	64.62	32.11	81.97	117.4

mpi_ssend_init mpi_send_init mpi_sendrecv

地球シミュレータ

- 前回までのあらすじ

- 通信を隠蔽するMPIの登場

- 以前のMPIは系コマンドを使っても、計算によって通信を隠蔽することが出来なかった
- その為、*Sendrecv* を使うことが推奨されたいた

- 8ノード64CPUを使い並列化率を計測

- 20ステップの経過時間から2ステップの経過時間を引くと、並列化率0.99733(375.9CPU)が達成
- 暫定利用申請へ

地球シミュレータ

- 並列化率的には376CPUにしか達していなかったが、512CPUを使う場合、計算規模が大きくなるので、512CPUを利用するだけの並列化率に達する見込みは大きい、と申請したところ、無事通過。
- 暫定利用64ノード(512CPU)をした結果、128ノード(1024CPU)でも並列化効率50%を達することが判明

地球シミュレータ

- 暫定128ノードで計算したところ、今度は1024ノードでも並列化効率が50%を維持することが判明。
- 暫定1024ノードで計算をしようとしたところ、計算が止まる。原因を究明しようにもジョブはなかなか回らない。そうこうしている間に暫定利用期間が終了。

地球シミュレータ

- 128ノード本申請が許可された段階で、 1024^3 体計算を64ノード使って開始。
- 現在は $z=3$ 辺りまで計算が進んでいるが、計算が止まっている。原因は調査中。といっても、5月から中断中なので、この研究会から帰ったら、この問題に取り掛かる。

展望

- 無衝突系

- 開放境界条件

- *Tree*

- GRAPEで加速させることもできる

- 周期境界条件

- *Tree-PM*

- メモリ使用量が少ない

- SPH

- *AMR*

- 高速(少なくとも私のは)

- Euler法

展望

- SuprimeCAM
 - 100Mpc立法の計算の視野と同程度
- Hyper-SuprimeCAM
 - 11月の研究会では、嶋作さんが、 z がなんぼかで、一回の撮像で300Mpcの領域が撮像できるようになると言っていた
 - 質量分解能を上げるのではなく、領域を大きくする必要あり

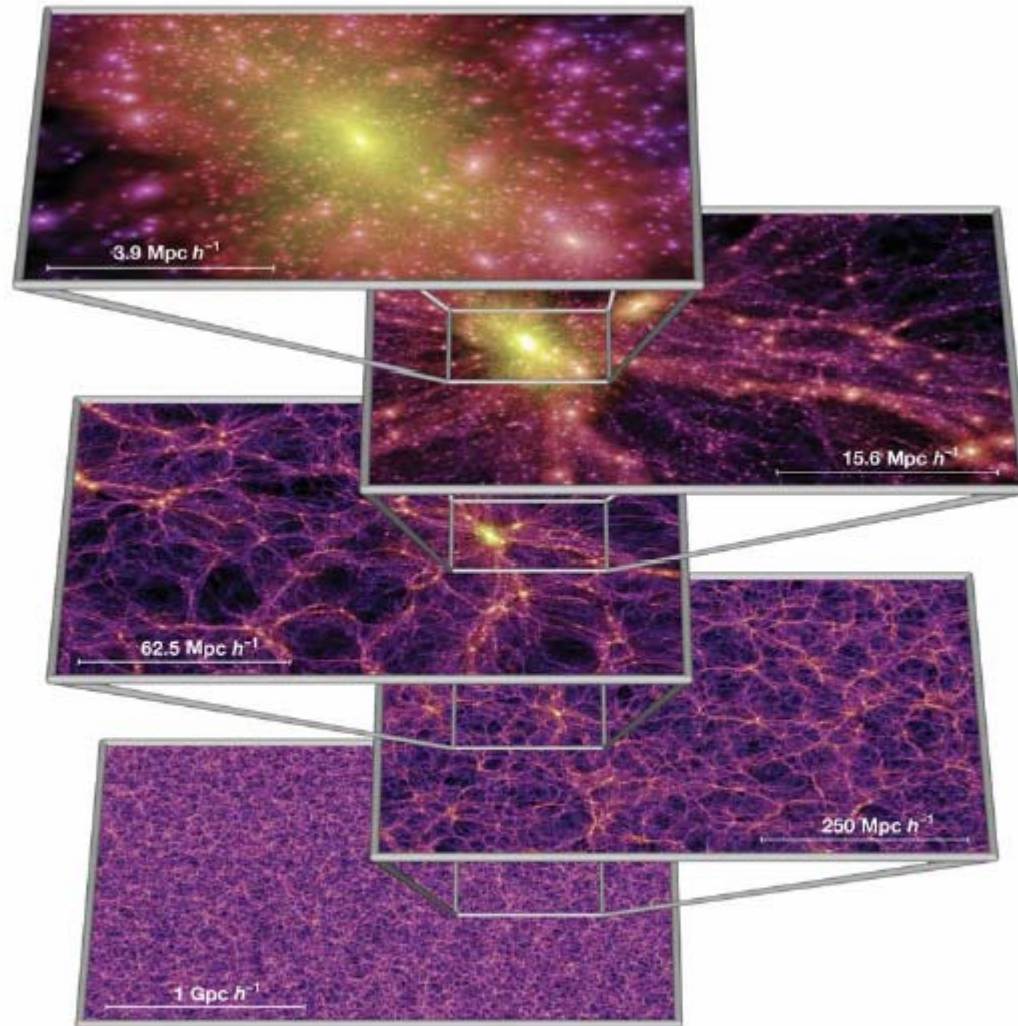


Figure 1 | The dark matter density field on various scales. Each individual image shows the projected dark matter density field in a slab of thickness $15h^{-1} \text{ Mpc}$ (sliced from the periodic simulation volume at an angle chosen to avoid replicating structures in the lower two images), colour-coded by

density and local dark matter velocity dispersion. The zoom sequence displays consecutive enlargements by factors of four, centred on one of the many galaxy cluster haloes present in the simulation.