

ペタフロップス級 宇宙論的N体 シミュレーション

石山 智明

筑波大学計算科学研究センター 神戸分室

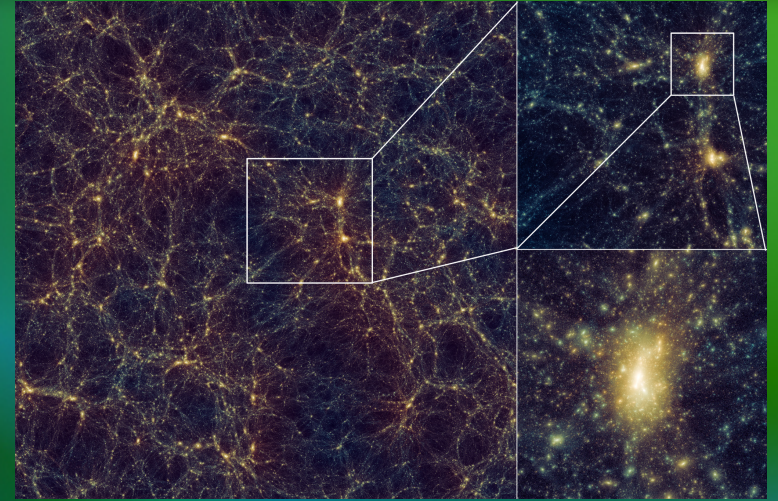
co-developer 似鳥 啓吾 (理化学研究所)

次期システム

- 次期システム
 - 1060ノード、25440 CPUコア、480Gflops / ノード
 - 当面は現行システムの**20倍**の性能
- 使い方はどう変わるか？
 - より大規模な計算を行う (大規模ジョブ)
 - たくさん計算する (ジョブ並列)
 - より複雑なことをする

宇宙論的N体の場合

現行システムでは 2048^3 粒子くらいが最大。次期システムでは最大 8192^3 粒子くらい

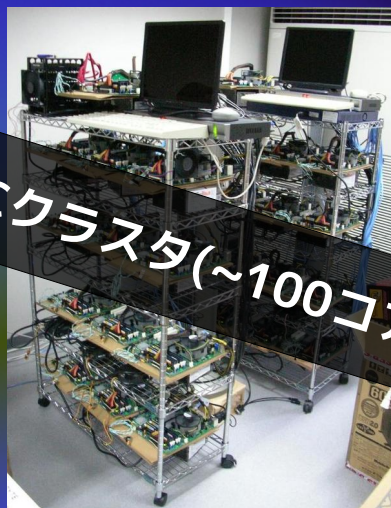


- より広い領域のシミュレーション
 - cosmology、大規模構造
- より微細な構造のシミュレーション
 - 銀河形成、ダークマター検出

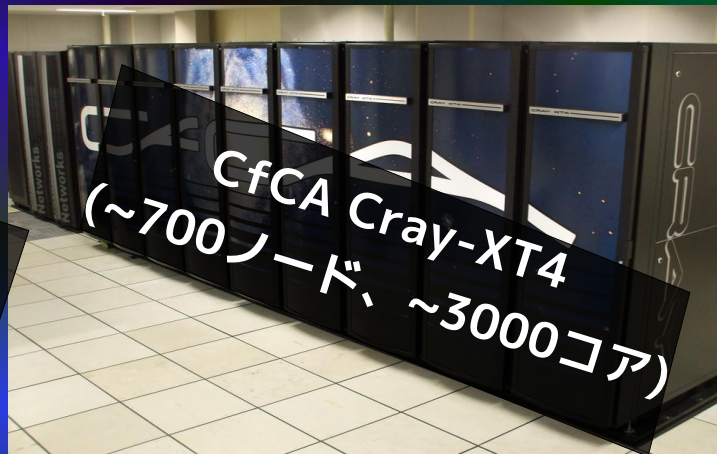


問題点

- ~1000億粒子以上の計算を“したければ”
~ 5000CPUコア以上が必要
- シミュレーションコードの**スケーリング**が問題



PCクラスタ(~100コア)



CfCA Cray-XT4
(~700ノード、~3000コア)



CfCA Cray-XC30
(~1060ノード、~25440コア)



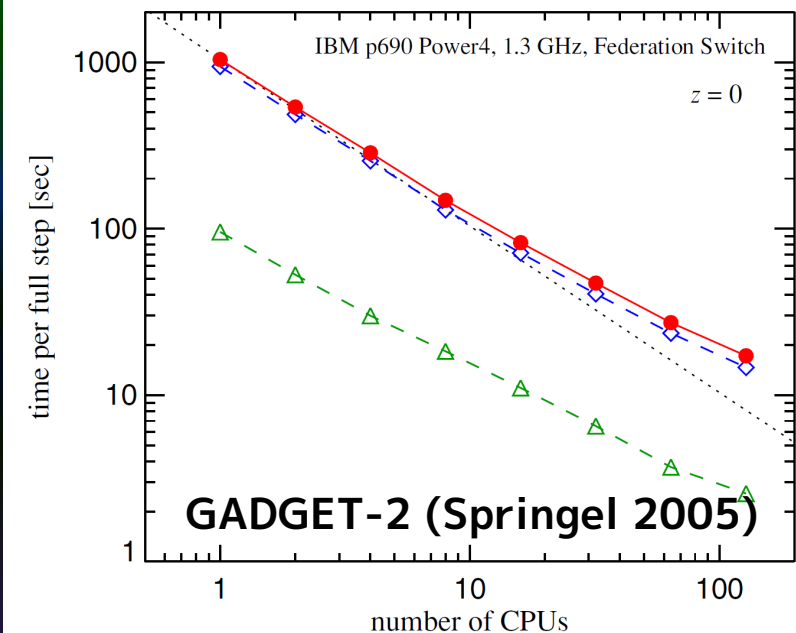
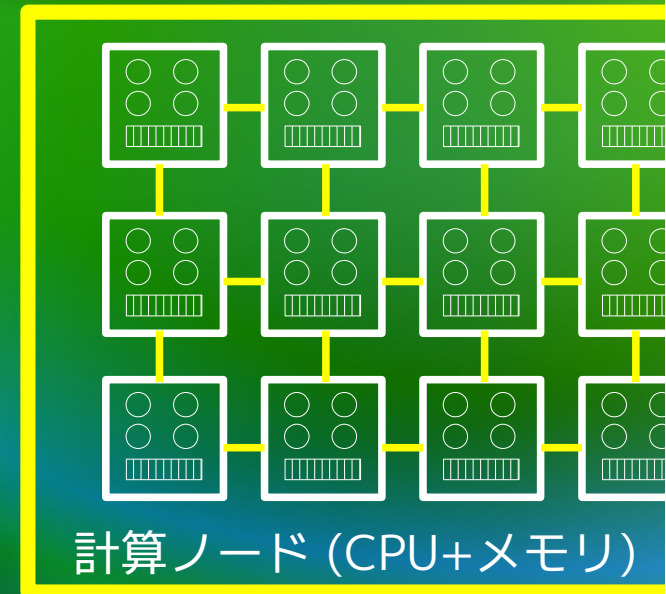
京速計算機(京)
~80000ノード
~66万コア
TOP500リストで世界No.1獲得

並列計算のスケーリング

- 理想的には CPU 数を n 倍にしたら n 倍の規模の計算が、または同じ規模の計算が $1/n$ の時間でできてほしい (スケーリング)
- 実際は重力計算に必要な通信が入るので無理
- 特に重力は遠距離力なので、ノード数が増えるほど辛くなる
 - 例えば GADGET-2 は 1000 並列以上でスケールする気がしない
- でも粒子数を増やしたい or もっと速く計算したい

→ **新しい並列アルゴリズム、コード開発**

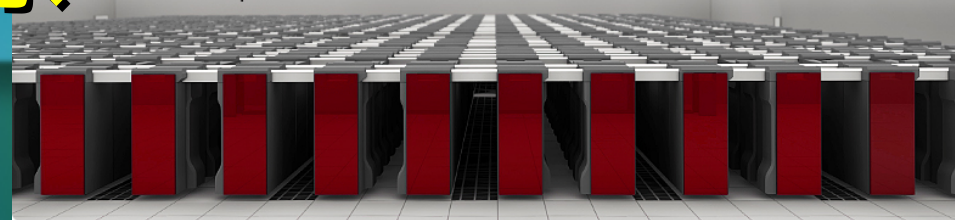
高速ネットワーク



テスト計算機としての"京"



TOP500リストで世界No.1 獲得



- SPARC64™ VIIIfx
2.0GHz octcore
(128Gflops / core)
- 16 GB memory / core
- 6D torus network
- Total 82944 nodes
(663552 CPU core)
- 1.3PB memory
- 10.6 Pflops peak speed

1	DOE/SC/Oak Ridge National Laboratory United States	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590	27113	8209
2	DOE/NNSA/LLNL United States	Sequoia - BlueGene/ Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	16325	20133	7890
3	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510	11280	12660
4	DOE/SC/Argonne National Laboratory United States	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom	786432	8162	10066	3945

2011年5、11月のTop 500 で世界 **No.1**

2012年5月に No.2 に転落

2012年11月に No.3 に転落

だが消費電力 No.1 は継続中

本研究

- 8万並列66万CPUコア (京の全システム) まで良くスケールする無衝突系重力多体(宇宙論的N体)シミュレーション用コードを開発した
 - さらに他のグループのコード(米アルゴンヌ)より5倍程度速い
- 重力計算アルゴリズムは ツリー法、TreePM 法
 - 構造形成、ダークマターハロー、銀河等幅広く使える
- 1兆粒子規模の無衝突系重力多体シミュレーションが現実的な時間で可能となった

重力計算アルゴリズム

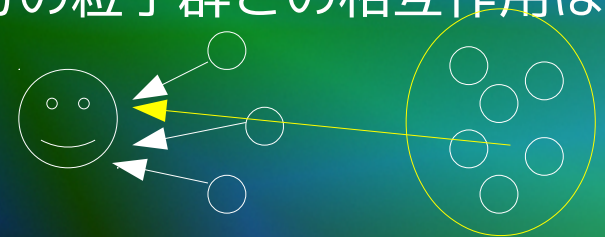
- 粒子間の相互作用重力を計算し、運動方程式を時間積分、系の時間発展を追う (時間積分は2次リーププロック)

- **直接計算** : 全粒子対全粒子の力を計算する

- $O(N^2)$

$$\frac{d^2 r_i}{dt^2} = \sum_{j \neq i}^N G m_j \frac{r_j - r_i}{|r_j - r_i|^3}$$

- **ツリー法** : 近傍の粒子との相互作用は直接、遠方の粒子群との相互作用はまとめて多重極展開で計算する



- $O(N^2)$ から $O(N \log N)$ へ

- **Modified algorithm**: 相互作用リストを粒子グループで共有する

- **PM法** : 一様格子上の密度場を計算し、FFTを用いてポアソン方程式を解く
 - 周期境界条件を自然に解ける

TreePM法

➤ 近距離力(カットオフつき)をTree法、遠距離力をPM法で解く

➤ $O(N^2) \rightarrow O(N \log N)$

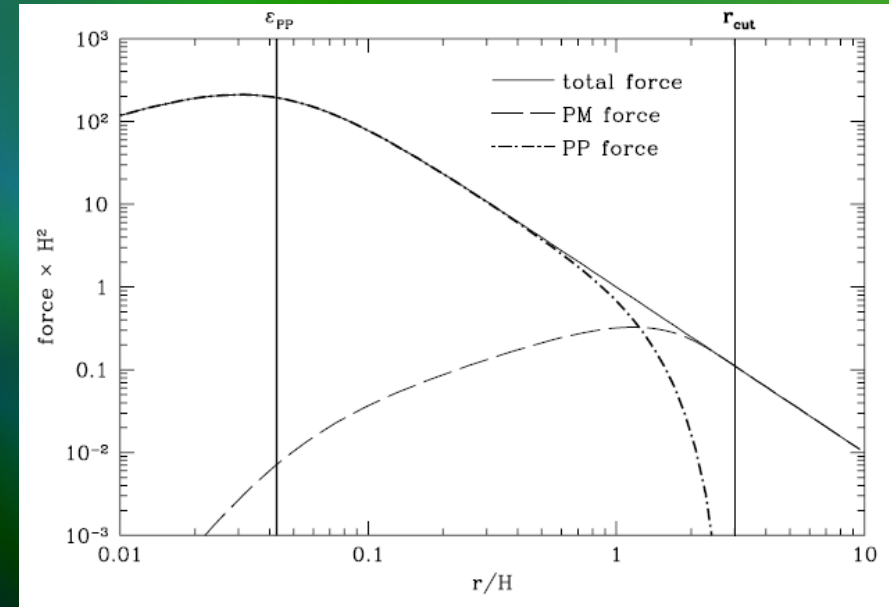
➤ 周期境界条件を実現

➤ 比較的新しいアルゴリズム

➤ GOTPM (Dubinski+ 2004)

➤ GADGET-2 (Springel 2005)

➤ GreeM (Ishiyama+ 2009)



$$\mathbf{a}_i = \sum_{j \neq i} \frac{m_j (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^3} g_{\text{P3M}}(|\mathbf{r}_j - \mathbf{r}_i|/\eta),$$

$$g_{\text{P3M}}(R) = \begin{cases} 1 - \frac{1}{140} (224R^3 - 224R^5 + 70R^6 + 48R^7 - 21R^8) & (0 \leq R \leq 1) \\ 1 - \frac{1}{140} (12 - 224R^2 + 869R^3 - 840R^4 + 224R^5 + 70R^6 - 48R^7 + 7R^8) & (1 \leq R \leq 2) \\ 0 & (2 \leq R) \end{cases},$$

典型的な TreePM 法の並列化

1. 全空間を分割し計算ノードに割り当て、
粒子を再配分する

- ・ サンプル粒子の集約 (全対通信)
- ・ 一部粒子情報を通信 (隣接通信)

1. 長距離重力の計算 (PM 法)

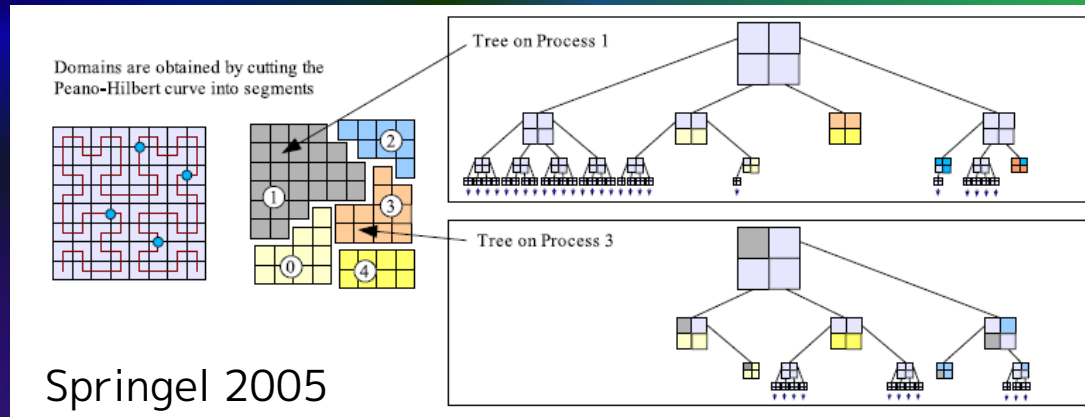
- ・ 全メッシュ情報を通信
(全対通信、粒子よりは通信量小)

2. 短・中距離重力の計算 (Tree法)

- ・ 一部ツリー情報を通信 (隣接通信)

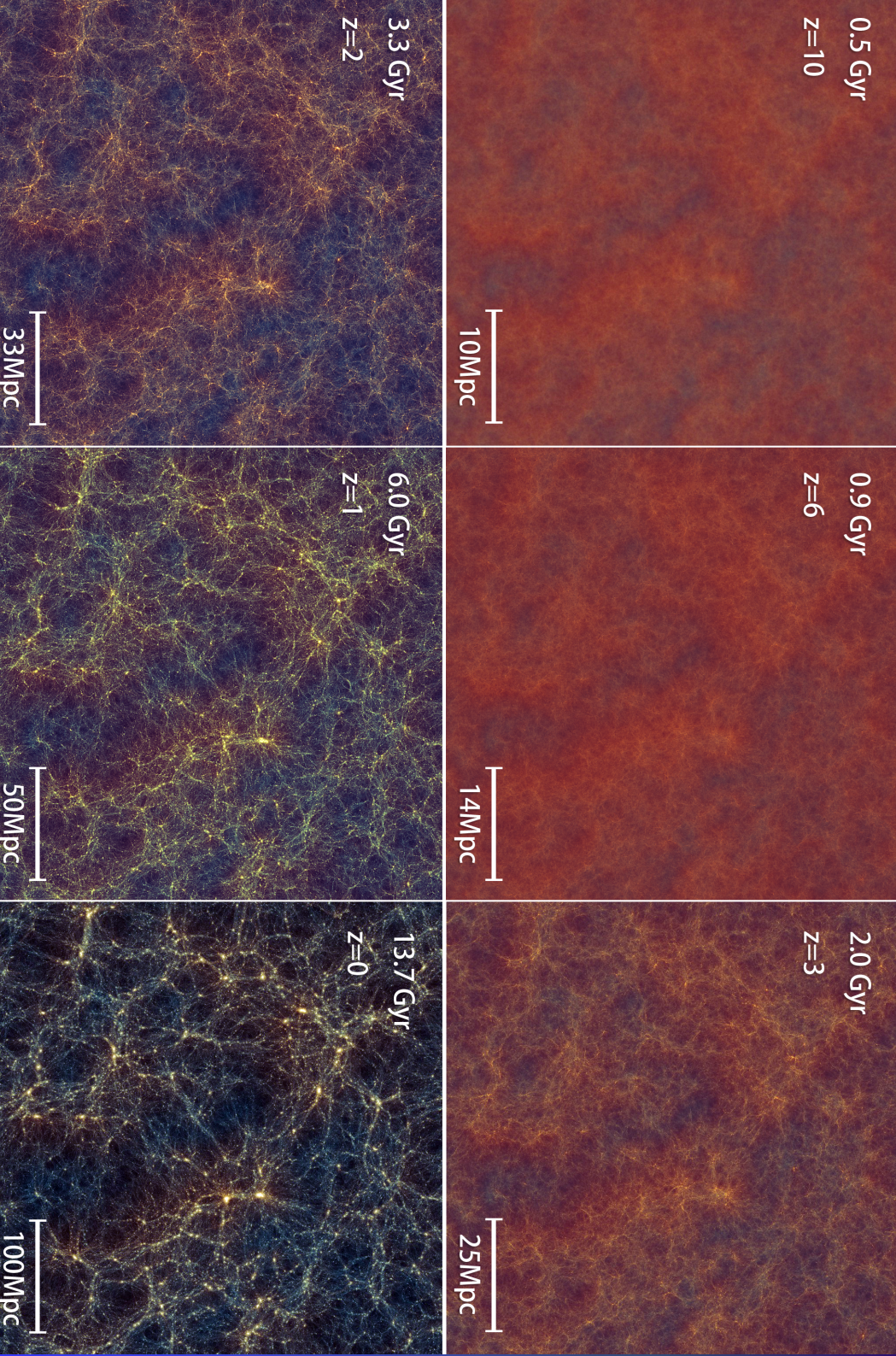
3. 粒子の時間積分

4. 1に戻る



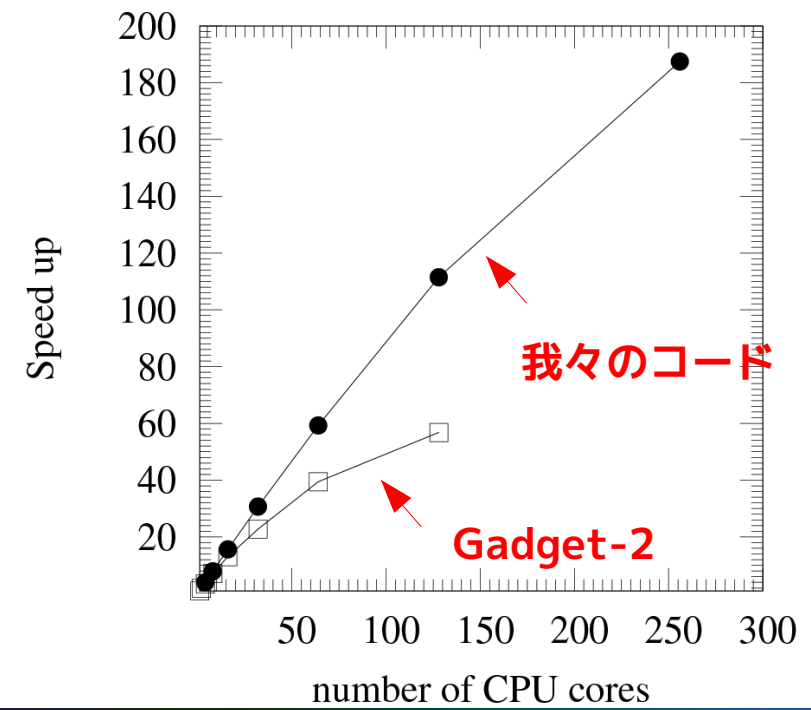
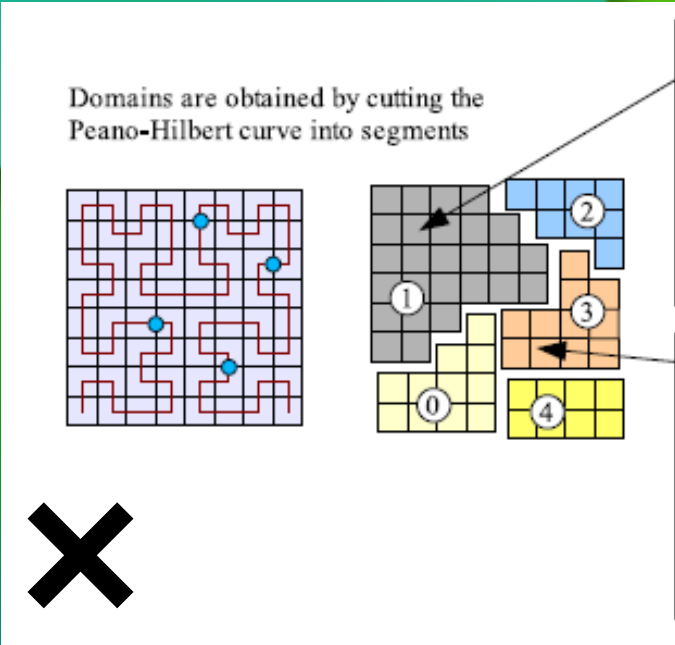
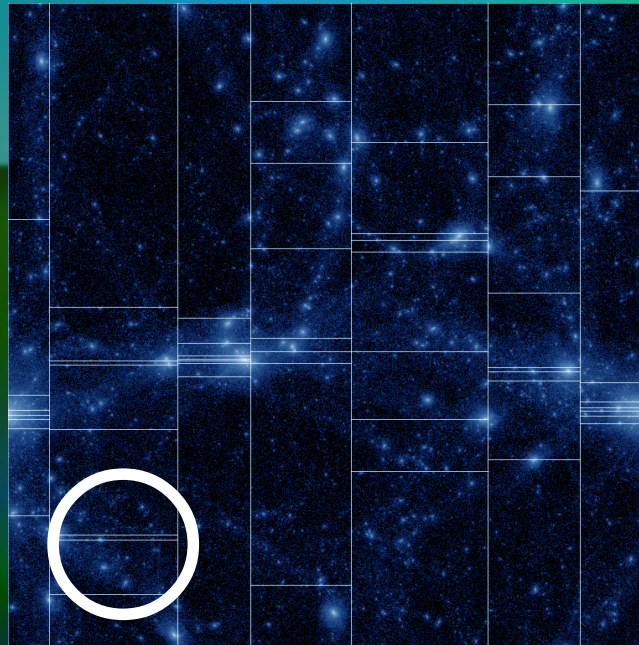
何が面倒か？

- 高密度構造が至るところで発展する
→ 動的領域分割
 - **通信パターンの不規則化**
- 重力は長距離力である
→ 通信アルゴリズムを工夫する
- ツリーで $O(N \log N)$ といっても重力計算は重い
 - SIMD
 - (GRAPEs, GPU)



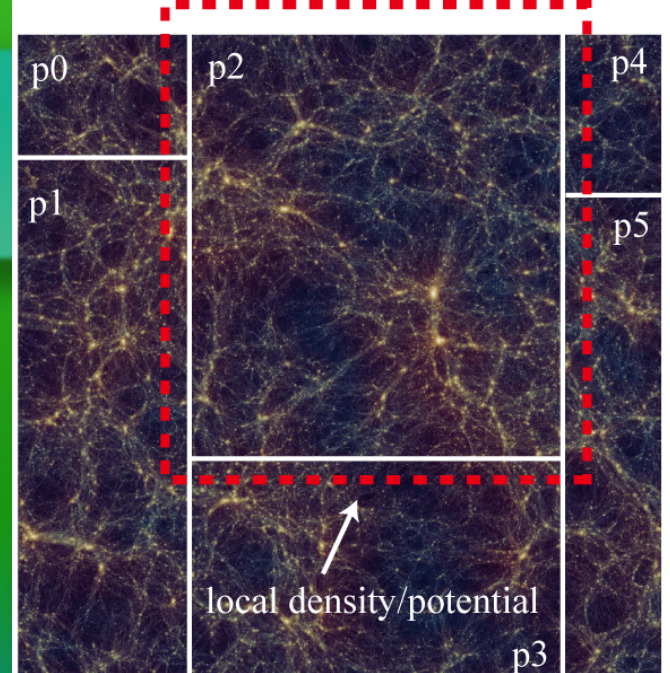
動的領域分割

- 領域分割の仕方が重要
- × 粒子数均等
- ○ 相互作用数均等
- ◎ 相互作用均等 + 補正
 - 計算時間が一定になるようにする
- △ 空間充填曲線
- ○ 再帰的多段分割
 - 隣接ノードが自明

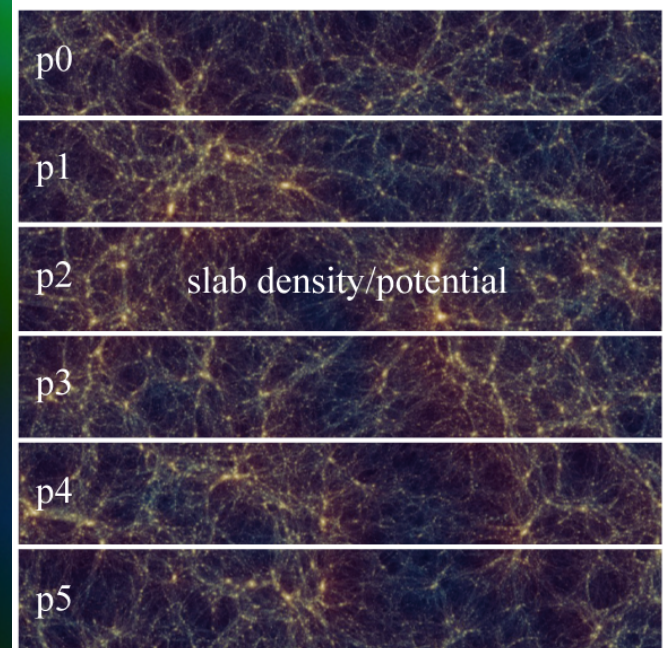


並列PM

1. density assignment
2. local density $\rightarrow$ slab density
 - 長距離通信
3. FFT (density $\rightarrow$ potential)
 - 長距離通信 (library 任せ)
4. slab potential $\rightarrow$ local potential
 - 長距離通信
5. force interpolation



2) density comm $\downarrow$ $\uparrow$ 4) potential comm



3) FFT & convolution

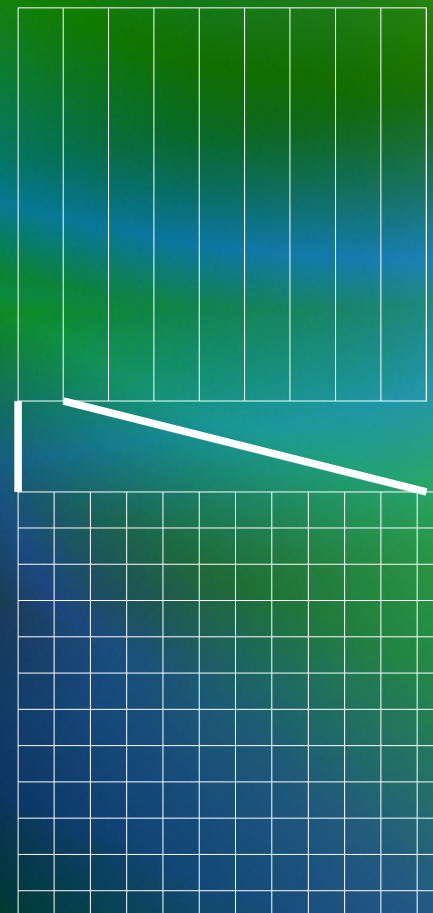
もう少し具体的に

- $\sim 10000^3$ 粒子、 4096^3 PMメッシュのシミュレーションを京のフルシステム ($48 \times 54 \times 32 = 82944$ ノード)で行う場合
 - 最適な領域分割は $48 \times 54 \times 32$ (network topology 依存)
 - 1D並列FFTW を使うと FFT は 4096 並列で可能



1FFTプロセスは **数千ノード** から
メッシュデータを受け取る必要有!!!

- MPI_Isend、MPI_Irecv は無理
- MPI_Alltoallv を使うと簡単に実装できるが、、、



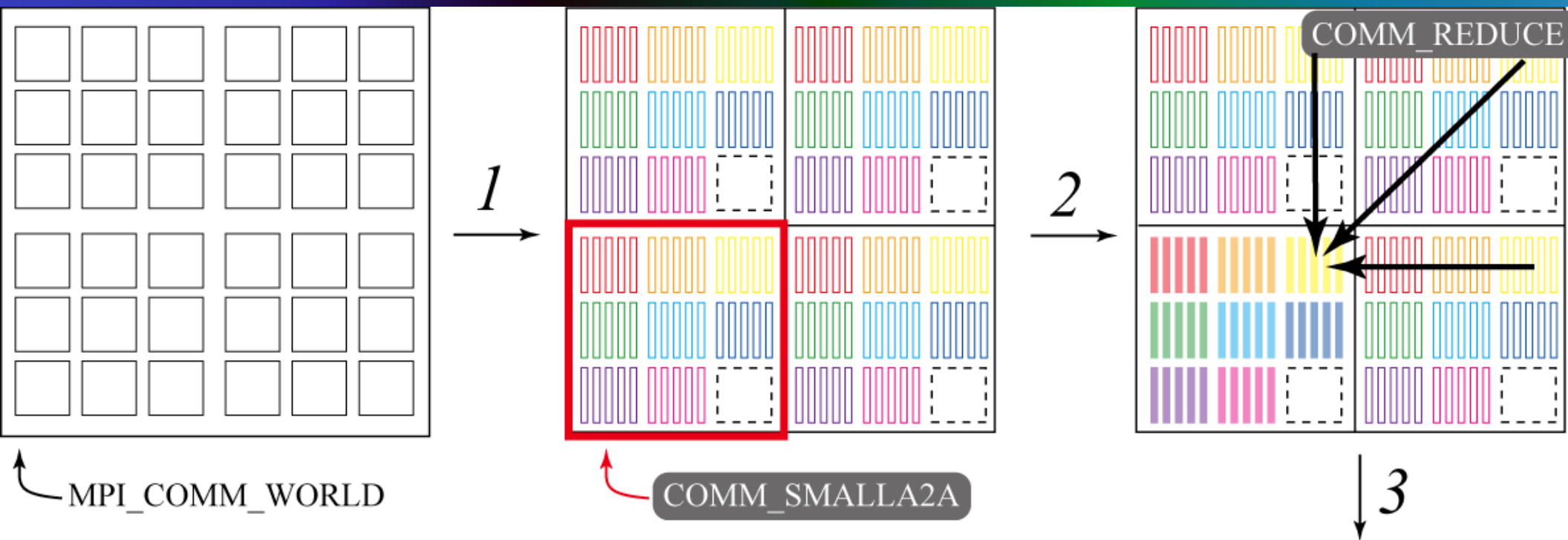
通信の階層化 (MPI_Comm_split を使う)

➤ MPI_Alltoallv (..., MPI_COMM_WORLD)

1. MPI_Alltoallv (..., COMM_SMALLA2A)

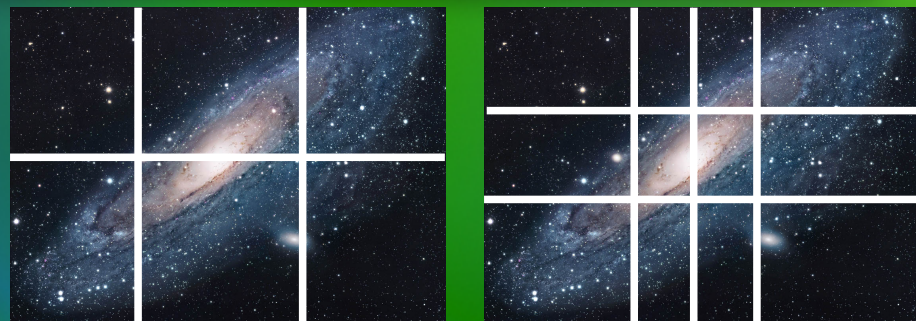
2. MPI_Reduce(..., COMM_REDUCE)

3~4 倍の通信高速化に成功!



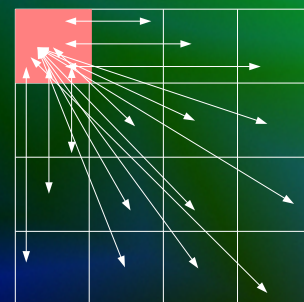
その他の通信最適化: 全データのシャッフル

- 全ノードのデータをシャッフルしたくなることもあるかもしれない
 - Restart 時にノード数が変わる時など



- 実装は Alltoallv が楽。だが数千～数万並列になると色々問題が……

- 通信コストは $O(p \log p)$ 、 p は並列数
- そもそも通信バッファは足りるのか?

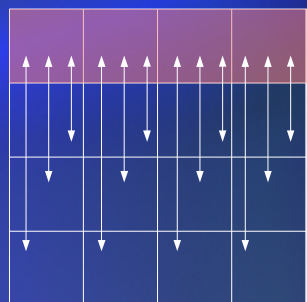
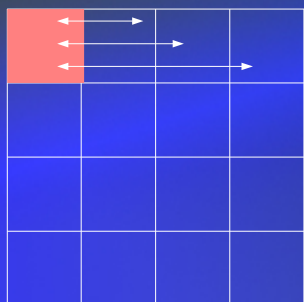


MPI_Alltoallv
(..., MPI_COMM_WORLD)

$$O(n p \log p)$$

n は通信分割数

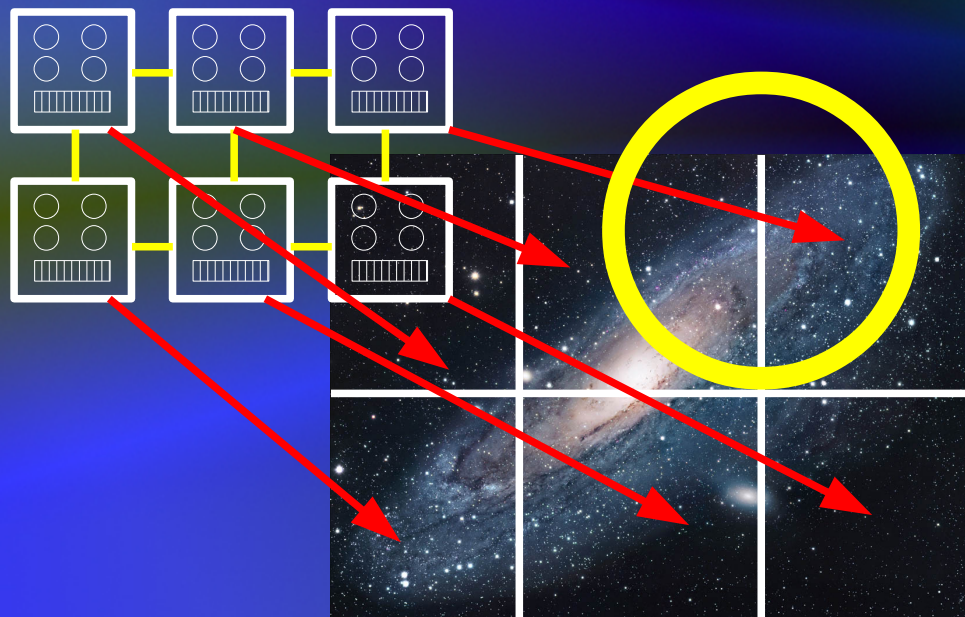
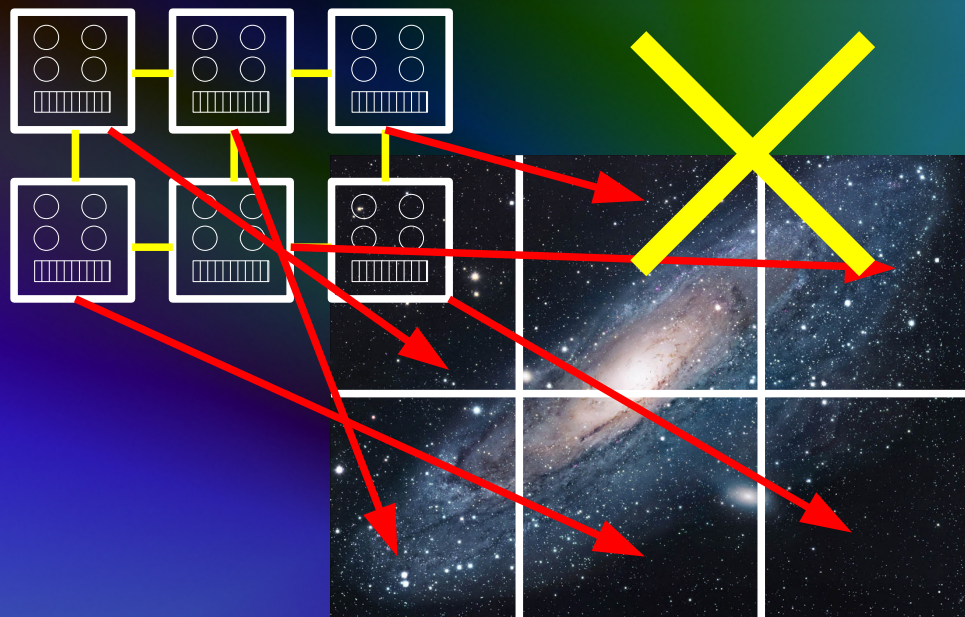
- 通信を階層的にすることで解決



MPI_Alltoallv (... , COMM_X) + MPI_Alltoallv (... , COMM_Y)
+ MPI_Alltoallv (... , COMM_Z)

$$O(3n^{1/3} p^{1/3} \log p^{1/3})$$

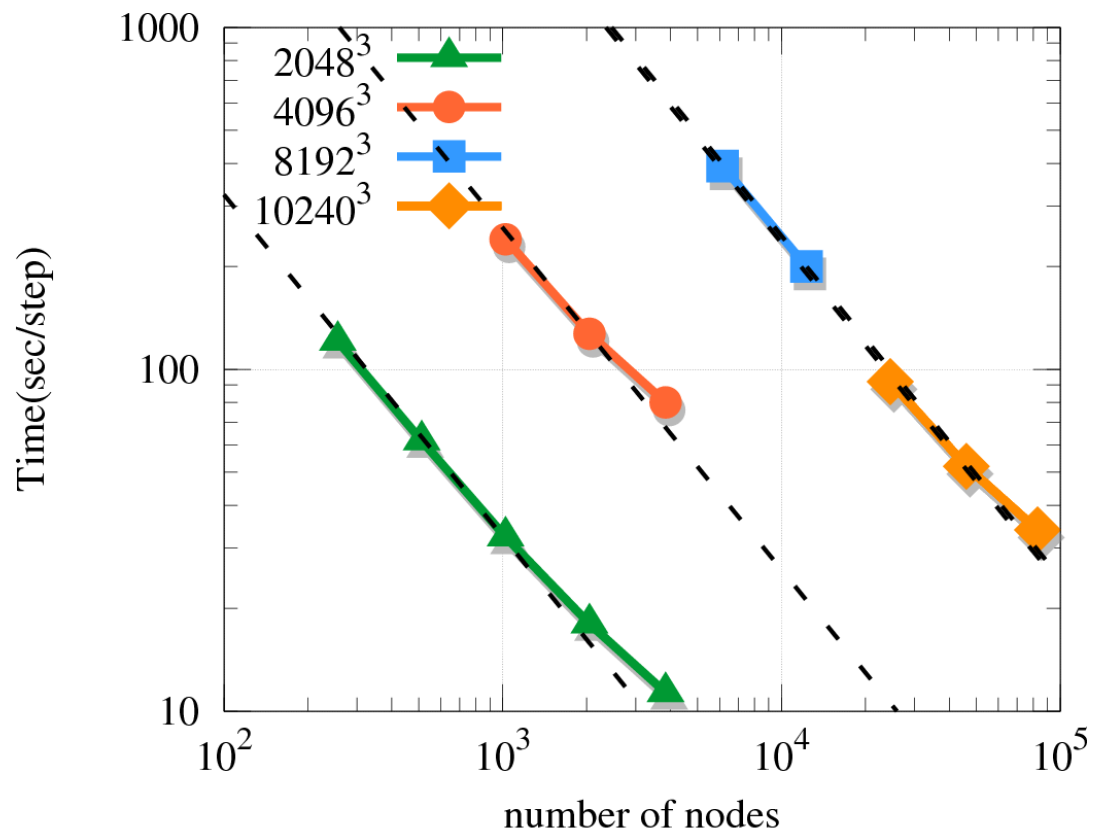
その他通信関係の最適化



京の場合 (torus network)

- 各領域を担当するノードの物理配置を、実際の領域分割にあわせる
 - 隣接した領域を担当するノードとの通信コストが最小に
- MPI + OpenMP のハイブリッド並列にする
 - 通信混雑の緩和
 - 計算機の仕様によってはMPIのみも可
- 通信と計算のオーバーラップ
 - 近距離力用の通信リスト作成と送受信を同時に行う

性能 (京速計算機)



- $2048^3 \sim 10240^3$ 粒子
 - 256並列から8万並列まで良くスケールする
- 12600^3 粒子
 - **55%**の対ピーク実行性能比

(**5.8** / 10.6 Pflops)

Ishiyama, Nitadori, and Makino,
2012 (arXiv: 1211.4406),
SC12 Gordon Bell Prize Winner

少し裏話

- 実はファイナリストには米アルゴンヌのグループが、同様の宇宙論的シミュレーションで Sequoia (BG/Q、ピーク 20Pflops) を用いて、14Pflops の実行性能を達成していた (Habib+ 2012)
- ところが、ピーク2倍の計算機を使っているにも関わらず、1粒子あたりの計算時間が我々のコードよりなぜか 2.4倍遅かった……

大規模データをどう解析するか？

- Output は 1粒子あたり 32bytes
 - 4096^3 粒子なら 2TB/snapshot、 10240^3 粒子なら 32TB/snapshot
 - 計算ノード数で分割
 - ただし個々の分割ファイルがどの領域の粒子を持っているかを記憶
 - 一部の領域の粒子が欲しいときに一部ファイルだけ読むようにする
- 粒子からの可視化は諦める (スナップショット数をそれほど多くできない)
 - 可視化に必要な情報だけを粗くして、細かい時間刻みで output
- 全粒子の情報が必要な解析は並列にやるか on the fly
 - Halo finding、 power spectrum など

まとめ

- 8万並列66万CPUコア (京の全システム) まで良くスケールする無衝突系重力多体(宇宙論的N体)シミュレーション用コードを開発した
 - さらに他のグループのコード(米アルゴンヌ)より5倍程度速い
 - MPI_COMM_WORLD のサイズを小さくすることが、通信最適化に必要不可欠
 - ツリー、TreePM法なので、構造形成、ダークマターハロー、銀河等幅広く使える
- 1兆粒子規模の無衝突系重力多体シミュレーションが現実的な時間で可能となった