# PEZY-SCクラスター睡蓮での計算科学アプリケーションの性能評価

中里直人(会津大学)

2016年2月11日

地球流体データ解析・数値計算ワークショップ

# 概要

- PEZY-SCプロセッサ概要
- Suiren(睡蓮)の仕様について
- 計算科学アプリケーションの性能評価

KEK 計算科学センター石川正 准教授

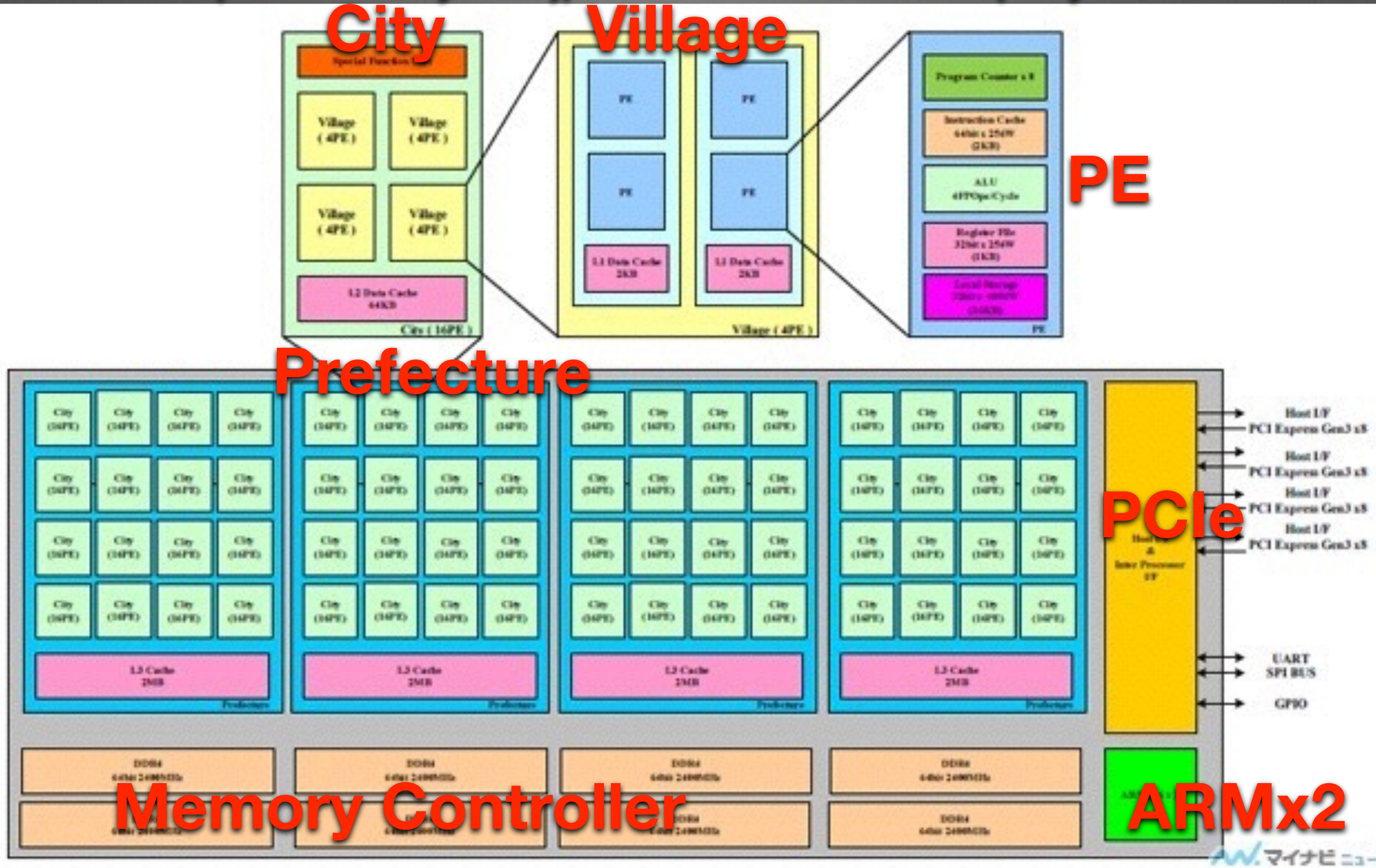
会津大学大学院 河野郁也さんらとの共同研究

# PEZY-SC processor

- MIMDアーキテクチャ
  - 1024 Processing Element (PE)がそれぞれ命令列を処理
  - dual-issue / SMT アーキテクチャ
  - 単精度 2 積和算/サイクル
  - 倍精度 1 積和算/サイクル
  - PEが階層構造で組み合わされている
- アクセラレータ
  - ホストCPUからPCIeを介して制御
  - 他のPEZY-SCとの通信もホスト経由

# PEの階層構造

http://news.mynavi.jp/articles/2014/09/17/pezy/ より



# PEの階層構造

- Villageブロック 4 PE
  - 2PEごとに一次データキャッシュを共有
- Cityブロック 4 Village (16 PE)
  - 二次データキャッシュを共有
  - Special Function Unit (SFU) : 除算, 剰余, 平方根など
- Prefecture 16 City (256 PE)
  - 三次データキャッシュを共有
- Top level 4 Prefecture (1024 PE)

# 命令実行の概要

- PEごとに8スレッドがSMT動作
  - それぞれがプログラムカウンタを保持する
  - それぞれがレジスタファイルを保持する
  - PEごとに命令キャッシュを持つ
  - ローカルメモリは共有
- 全体で最大8192スレッドが同時動作
  - 各データキャッシュ階層ごとに明示的に同期可能
  - これはOpenCL Kernel APIとは非互換
- OpenCL互換のPZCLでカーネル開発
  - ホストコードはOpenCL APIでI/Oおよびカーネル駆動

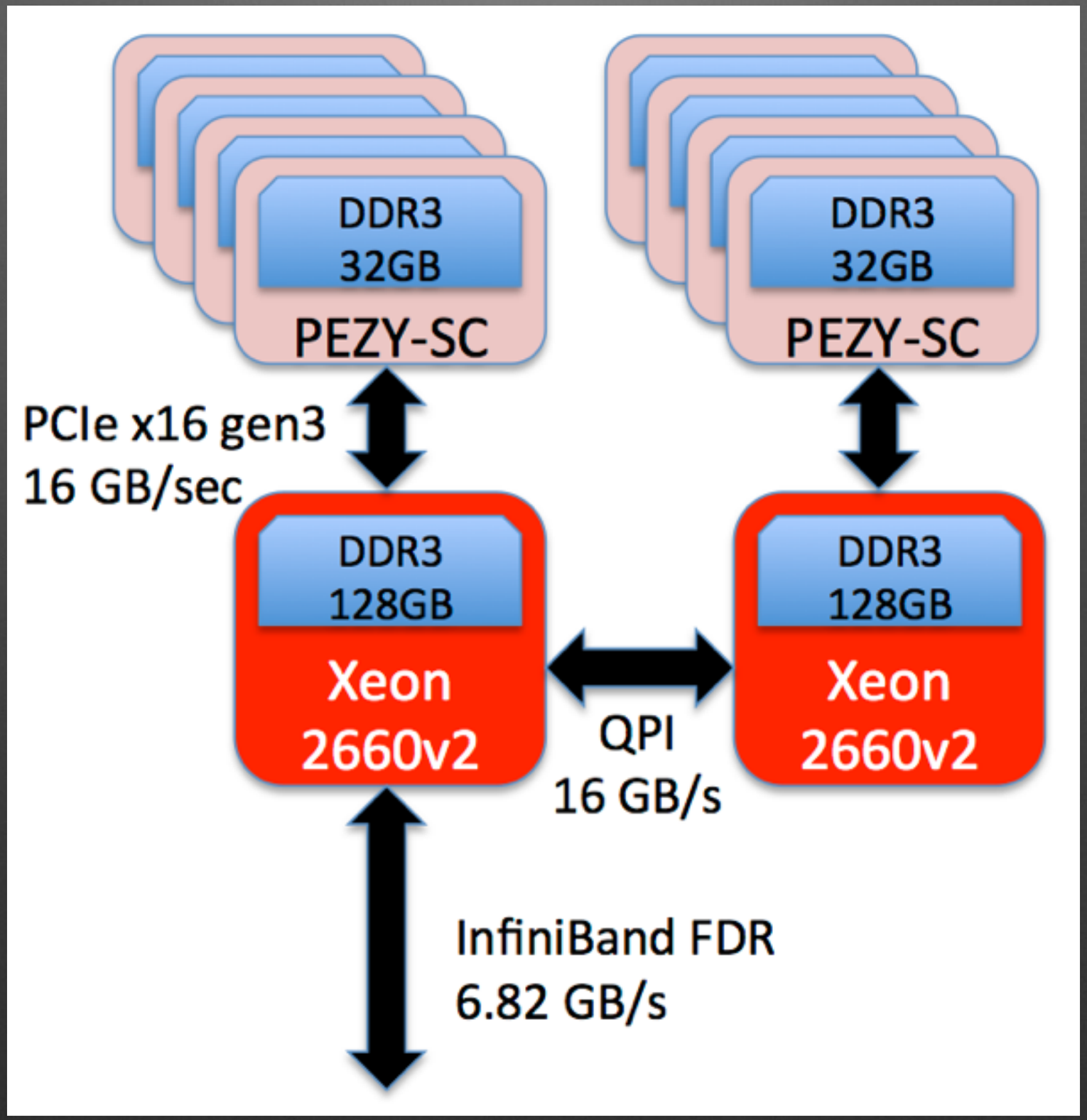
# Suiren(睡蓮)

- PEZY-SC利用の初めてのクラスタシステム
  - 高エネルギー加速器研究機構　計算科学センター
  - ZettaScaler-1.0 2014年10月稼働
  - Rmax 202.6 TFLOPS (2015年6月 TOP500)
  - 6.22 GFLOPS/W (Green 500 v1.2 rule 2015年11月)

**Listed below are the November 2014 The Green500's energy-efficient supercomputers ranked from 1 to 10.**

| Green500 Rank | MFLOPS/W | Site* | Computer* | Total Power (kW) |
|---|---|---|---|---|
| 1 | **5,271.81** | GSI Helmholtz Center | L-CSC - ASUS ESC4000 FDR/G2S, Intel Xeon E5-2690v2 10C 3GHz, Infiniband FDR, AMD FirePro S9150<br>Level 1 measurement data available | 57.15 |
| 2 | **4,945.63** | High Energy Accelerator Research Organization /KEK | Suiren - ExaScaler 32U256SC Cluster, Intel Xeon E5-2660v2 10C 2.2GHz, Infiniband FDR, PEZY-SC | 37.83 |
| 3 | **4,447.58** | GSIC Center, Tokyo Institute of Technology | TSUBAME-KFC - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.100GHz, Infiniband FDR, NVIDIA K20x | 35.39 |

# Suirenノード構成




98E
60h
2014/11/15 10:51:07

2014/10/30 16:39

# 性能評価の手法について

- OpenCLコードをPZCL用に修正
  - カーネルには大きな修正なし
  - 共有メモリ(__local)の利用はなし
  - PZCL特有のkernel APIは利用していない
  - ホストコードもOpenCLのまま。スレッド起動数は調整。

## 比較したアーキテクチャ

| Name | PEZY-SC | K20c | R280X |
|---|---|---|---|
| Architecture | PEZY-SC | GK110 | Tahiti XTL |
| Compute Unit | 1024 PE | 13 SMX | 32 CU |
| Clock (MHz) | 733 | 706 | 850 |
| Memory Type | DDR3 | GDDR5 | GDDR5 |
| Memory Size(GB) | 32 | 5 | 3 |
| Memory BW (GB/s) | 85.3 | 208 | 288 |
| SP Performance (GFLOPS) | 3000 | 3524 | 3482 |
| DP Performance (GFLOPS) | 1500 | 1170 | 870 |
| Programming Framework | PZCL | CUDA/OpenCL | OpenCL |

# PZCL/OpenCL カーネル共通化

```
#ifdef _PZC_KERNEL
#include "pzc_builtin.h"

#define KERNEL(xxx) void pzc_##xxx
#define __global

unsigned int get_global_id(int i)
{
  if (i == 0) return get_tid();
  if (i == 1) return get_pid();
  return 0;
}

unsigned int get_global_size(int i)
{
  return get_maxtid();
}
#else

#define KERNEL(xxx) __kernel void xxx

void flush() {}
#endif

KERNEL(force_pot_grav_jerk_11_DS)
(....
```

カーネルソースのヘッダ部分　抜粋

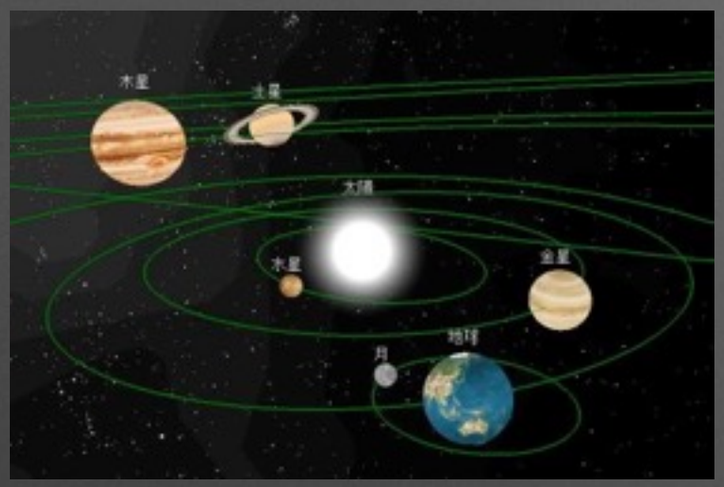
カーネルの定義

# 性能評価した計算科学アプリ

- 重力多体問題 $O(N^2)$
  - 粒子間相互作用計算カーネルの性能
  - Hermite積分法による軌道積分
  - MPI並列によるSuirenの性能評価
- 流体シミュレーション $O(N \log N)$
  - octree法による相互作用カーネルの高速化
  - Smoothed Particle Hydrodynamics(SPH)法
- ステンシル計算 $O(N)$
  - 津波進化計算：浅水方程式の解法MOSTの並列化

# 宇宙の粒子シミュレーション

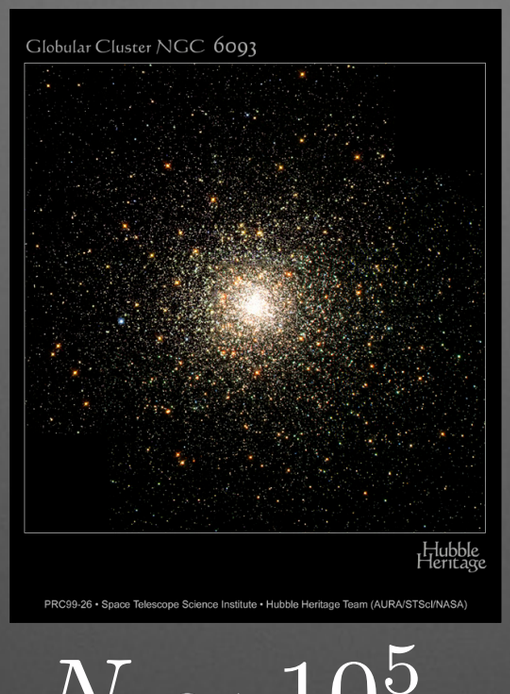
## solar system




$N \sim 10$

$t_{\rm lifetime} \sim 10^9 {\rm yr}$

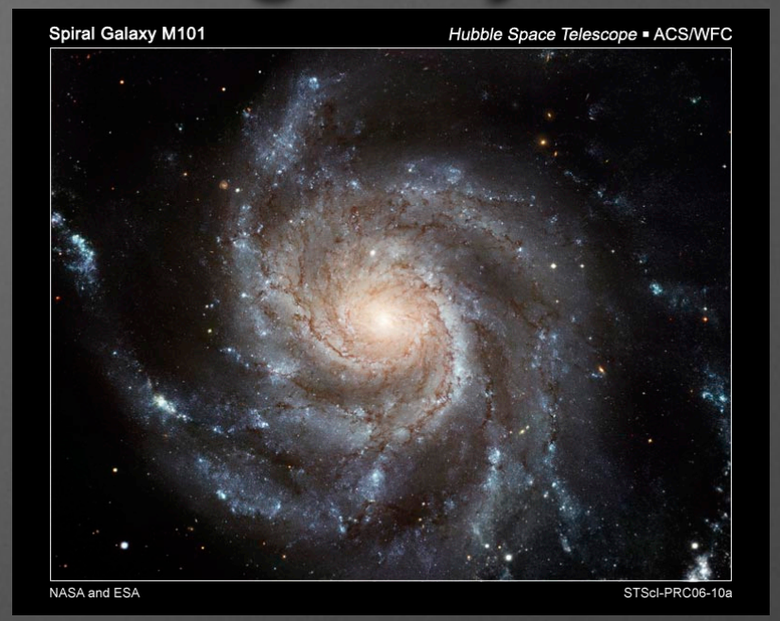
## star cluster




$N \sim 10^5$

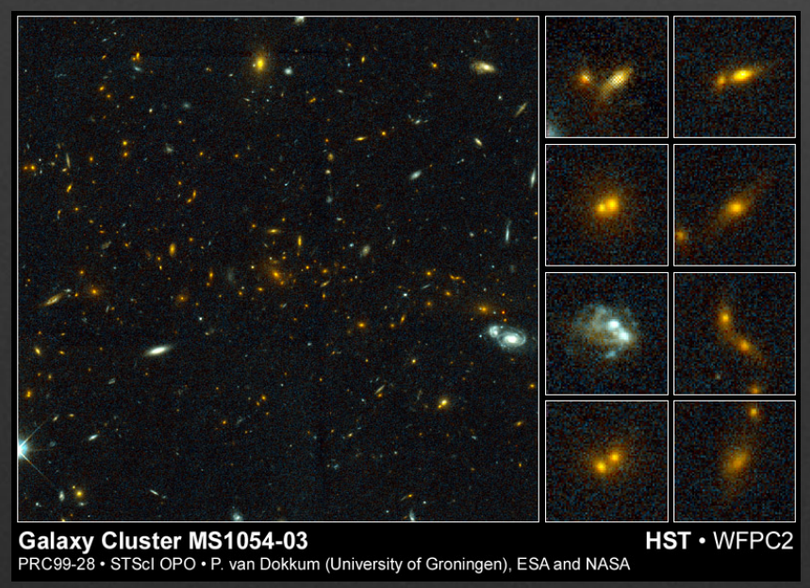
$t_{\rm lifetime} \sim 10^{10} {\rm yr}$

## galaxy




$N \sim 10^{11}$

$t_{\rm lifetime} \sim 10^{10} {\rm yr}$




## cluster of galaxies

$N \sim 10^3$

$t_{\rm lifetime} \sim 10^{10} {\rm yr}$

# Hermite積分法の性能評価(1)

- 重力多体問題専用計算機GRAPE用に開発
  - 予測子-修正子法
  - 今回の実装は時間精度4次の積分法
  - タイムステップブロック化積分

**予測子 (PEZY & HOST) *O(N)***

$$\boldsymbol{r}_i^{(\mathrm{p})} = \boldsymbol{r}_i^{(0)} + \boldsymbol{v}_i^{(0)}\Delta t_i + \frac{\boldsymbol{a}_i^{(0)}}{2}\Delta t_i{}^2 + \frac{\boldsymbol{j}_i^{(0)}}{6}\Delta t_i{}^3$$

$$\boldsymbol{v}_i^{(\mathrm{p})} = \boldsymbol{v}_i^{(0)} + \boldsymbol{a}_i^{(0)}\Delta t_i + \frac{\boldsymbol{j}_i^{(0)}}{2}\Delta t_i{}^2,$$

**重力計算 (PEZY) *O(N²)***

$$\boldsymbol{a}_i^{(1)} = \sum_{i\neq j}^{N} \frac{Gm_j}{\left[\left(r_{ij}^{(\mathrm{p})}\right)^2 + \varepsilon^2\right]^{3/2}} \boldsymbol{r}_{ij}^{(\mathrm{p})},$$

$$\boldsymbol{j}_i^{(1)} = \sum_{i\neq j}^{N} \frac{Gm_j}{\left[\left(r_{ij}^{(\mathrm{p})}\right)^2 + \varepsilon^2\right]^{3/2}} \left[\boldsymbol{v}_{ij}^{(\mathrm{p})} - \frac{3\boldsymbol{r}_{ij}^{(\mathrm{p})}\cdot\boldsymbol{v}_{ij}^{(\mathrm{p})}}{\left(r_{ij}^{(\mathrm{p})}\right)^2 + \varepsilon^2}\boldsymbol{r}_{ij}^{(\mathrm{p})}\right]$$

**修正子(HOST CPU)**

$$\boldsymbol{s}_i^{(0)} = 2\left[-3\left(\boldsymbol{a}_i^{(0)} - \boldsymbol{a}_i^{(1)}\right) - \left(2\boldsymbol{j}_i^{(0)} + \boldsymbol{j}_i^{(1)}\right)\Delta t_i\right]\Delta t_i^{-2},$$

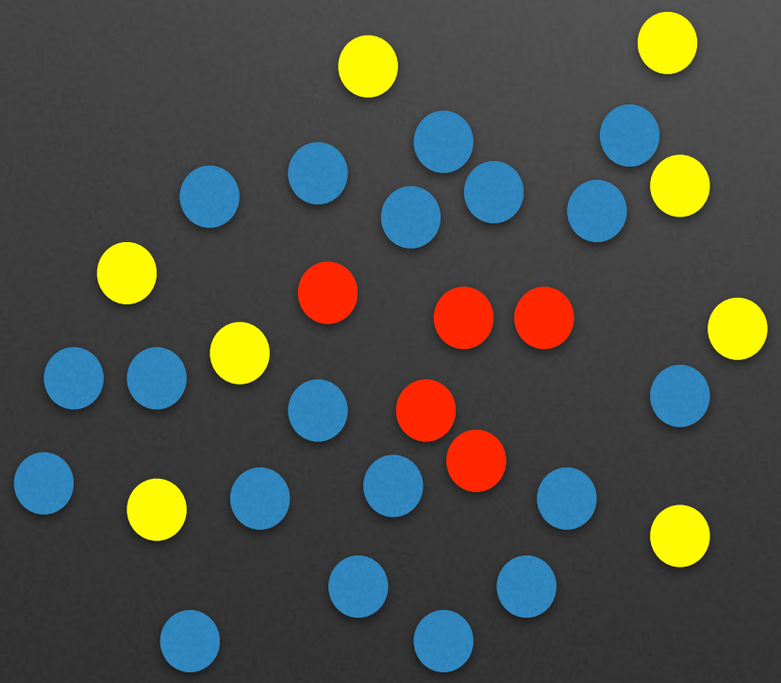
$$\boldsymbol{c}_i^{(0)} = 6\left[2\left(\boldsymbol{a}_i^{(0)} - \boldsymbol{a}_i^{(1)}\right) + \left(\boldsymbol{j}_i^{(0)} + \boldsymbol{j}_i^{(1)}\right)\Delta t_i\right]\Delta t_i^{-3}.$$

$$\boldsymbol{r}_i^{(1)} = \boldsymbol{r}_i^{(\mathrm{p})} + \frac{\boldsymbol{s}_i^{(0)}}{24}\Delta t_i{}^4 + \frac{\boldsymbol{c}_i^{(0)}}{120}\Delta t_i{}^5$$

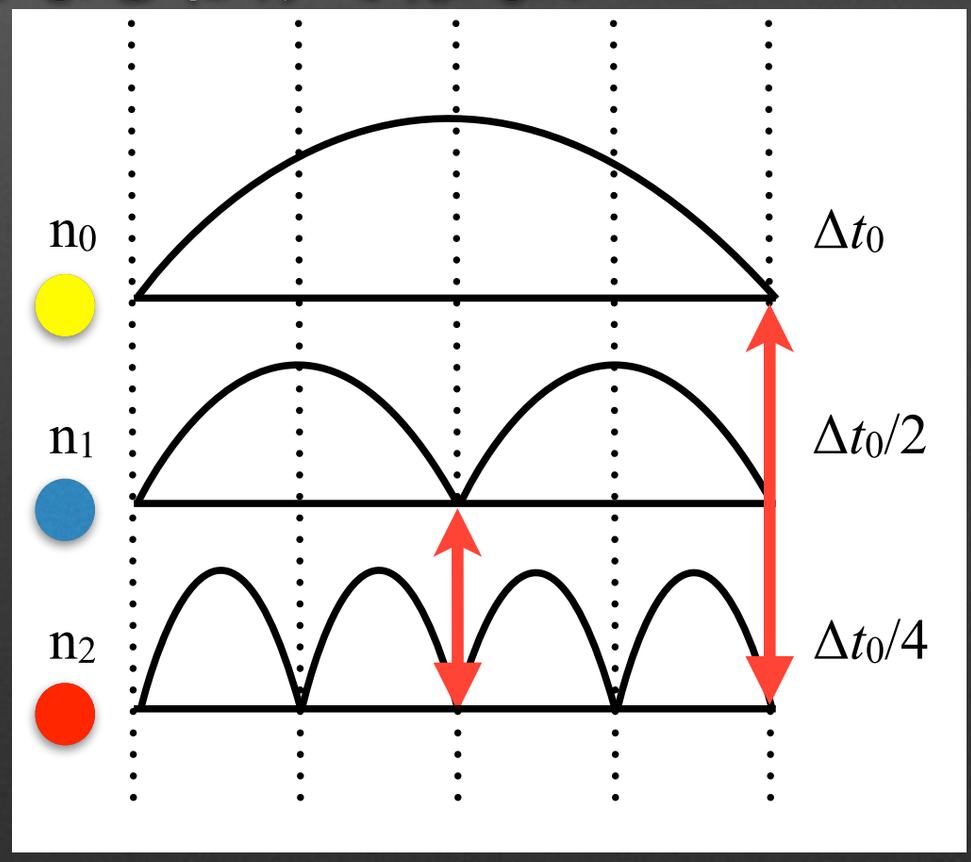
$$\boldsymbol{v}_i^{(1)} = \boldsymbol{v}_i^{(\mathrm{p})} + \frac{\boldsymbol{s}_i^{(0)}}{6}\Delta t_i{}^3 + \frac{\boldsymbol{c}_i^{(0)}}{24}\Delta t_i{}^4.$$

# Hermite積分法の性能評価(2)

- ブロック化積分法概要
  - 全粒子について積分タイムステップをローカルに評価
  - タイムステップを2のべき乗で量子化
  - それに基づき粒子をグループ(ブロック)化
  - 最もタイムステップの短いグループのみを積分
  - よって演算量が毎ステップ $O(N^2)$ となるわけではない

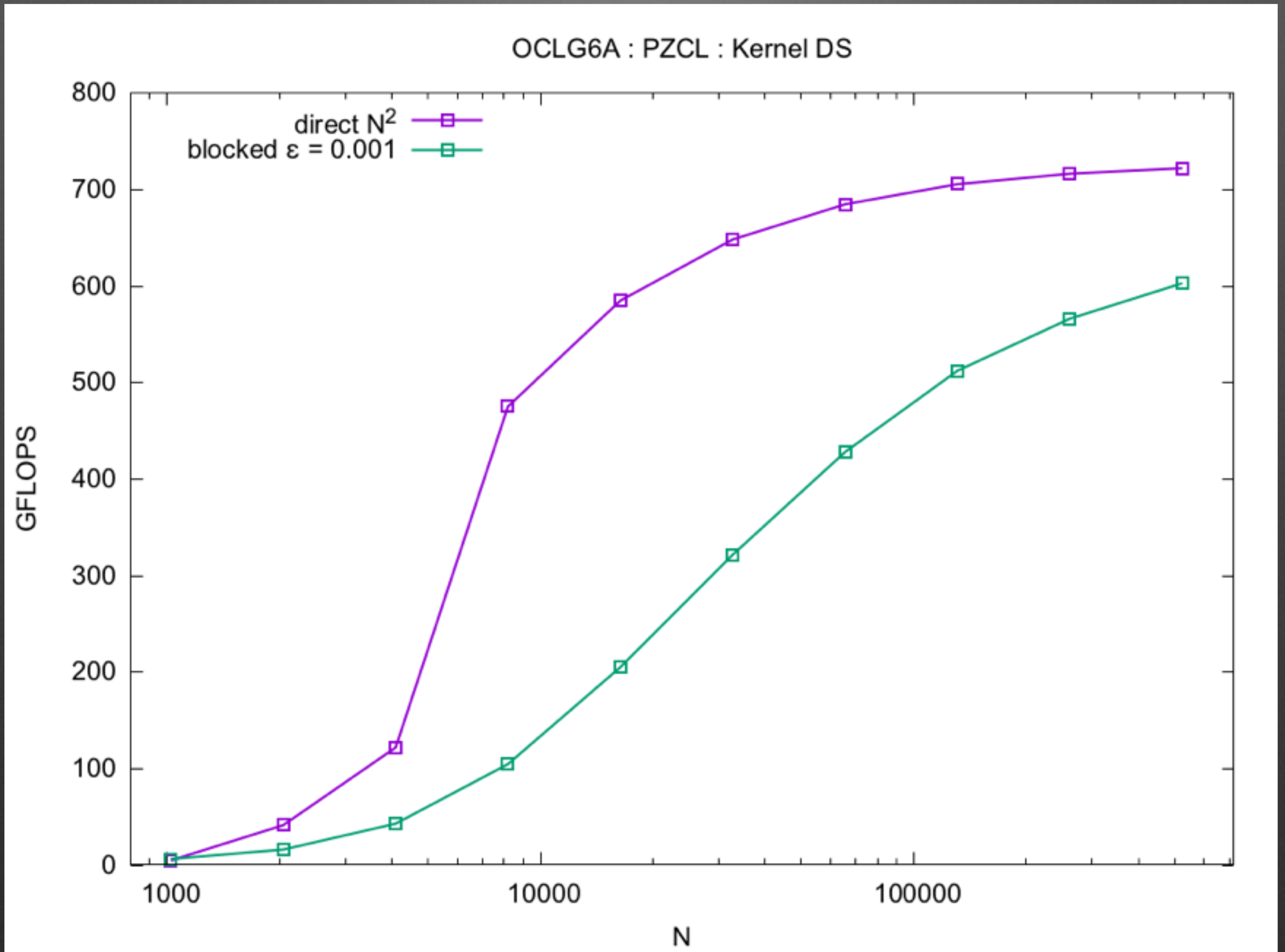


粒子のブロック化

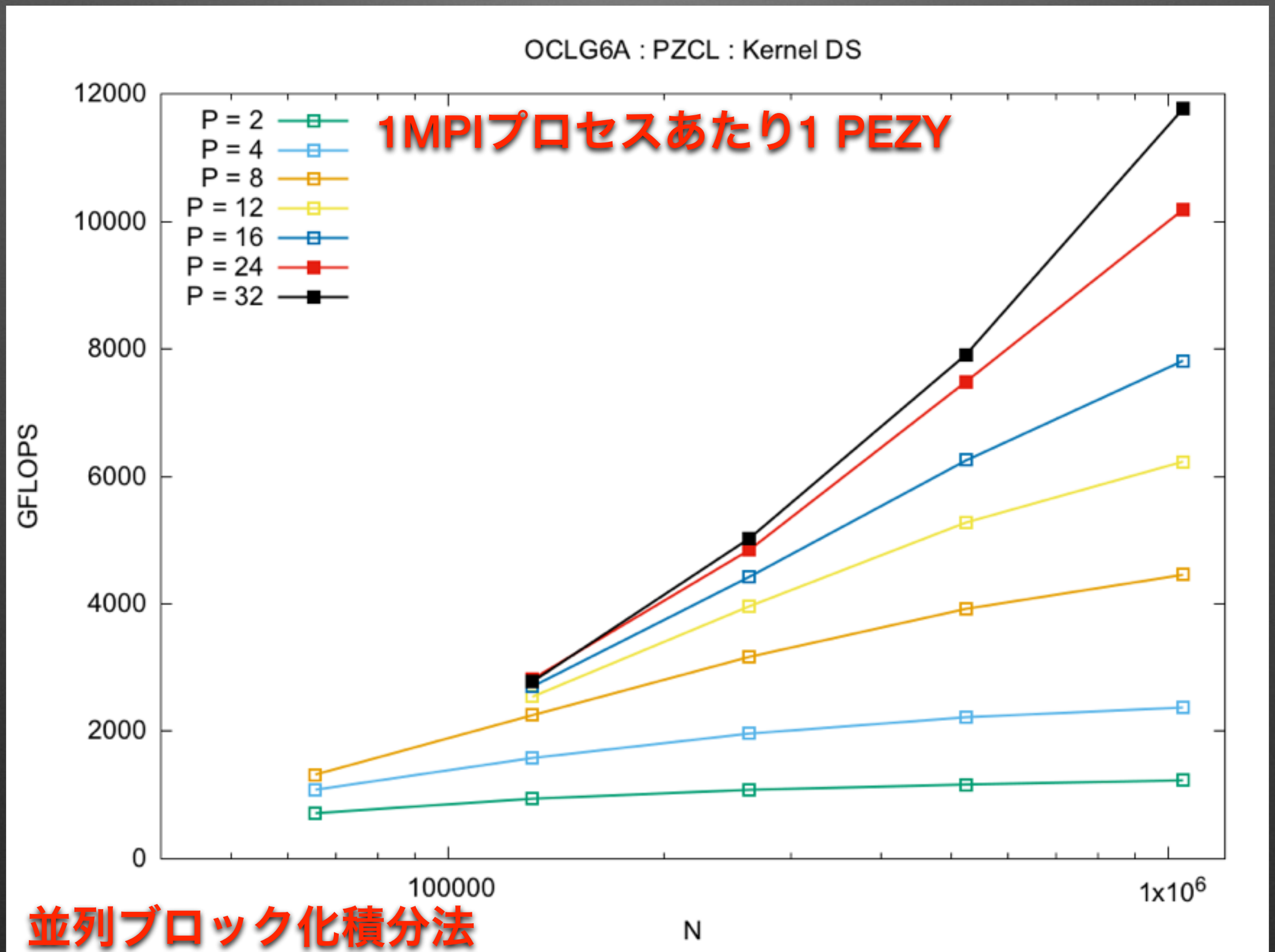



http://arxiv.org/abs/1105.1082 より

# Hermite積分法 1 PEZY-SC



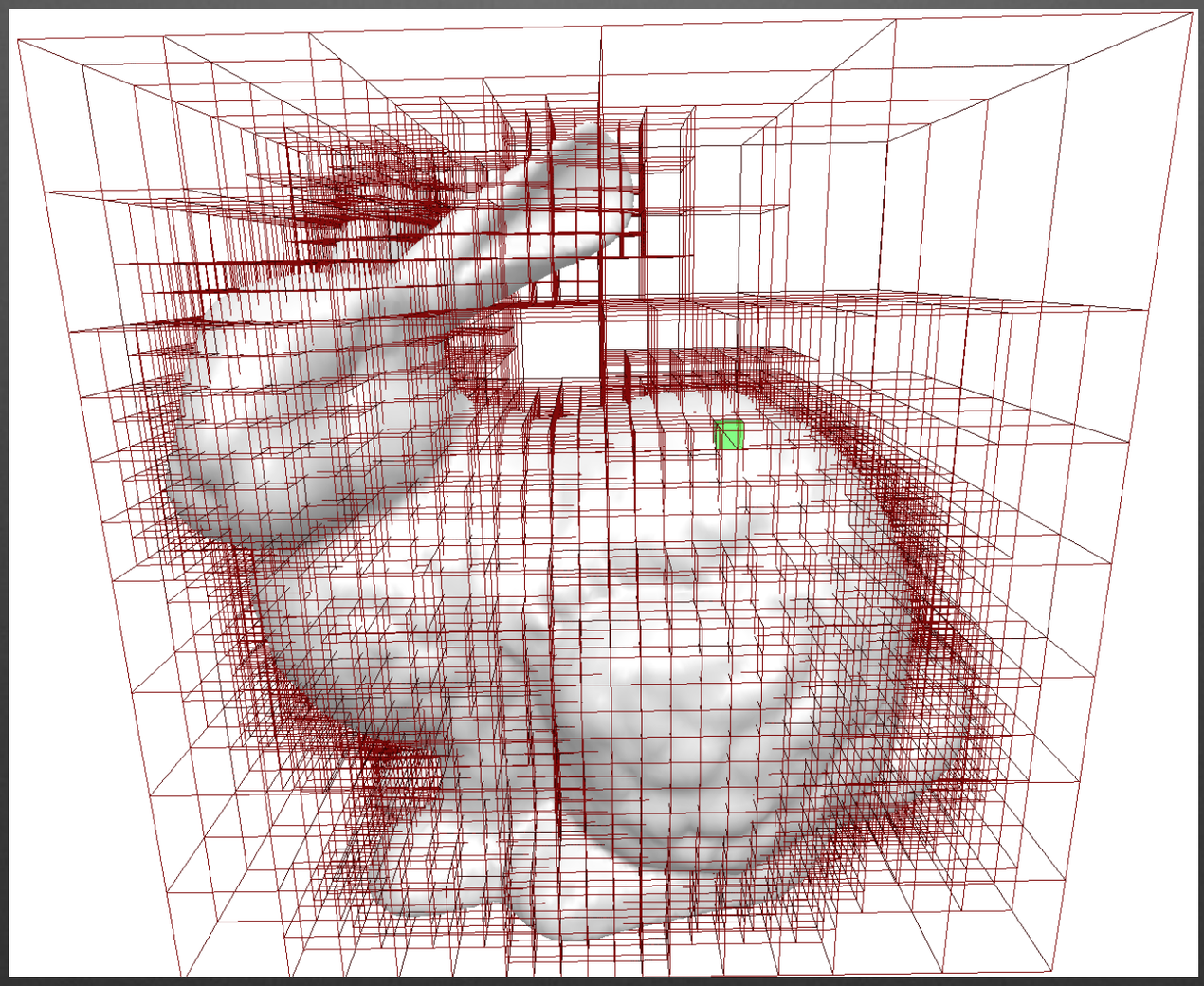
# Hermite積分 MPI 並列化




並列ブロック化積分法

# Octreeデータ構造

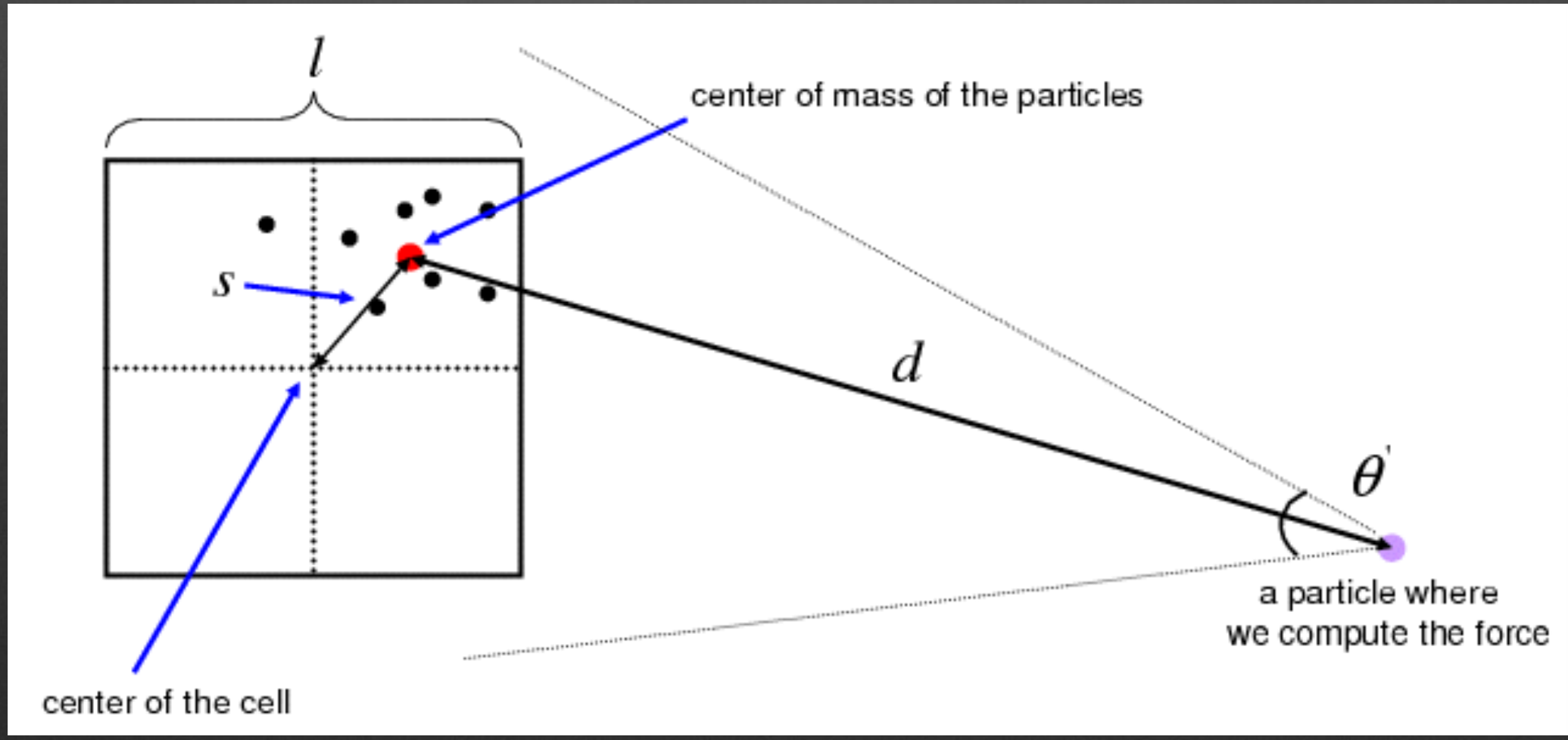
- 空間を再帰的に八分割
  - 疎な粒子分布を効率的に表すことができる
  - 距離判定・交差処理などが効率化できる



from Castro etal. 2008

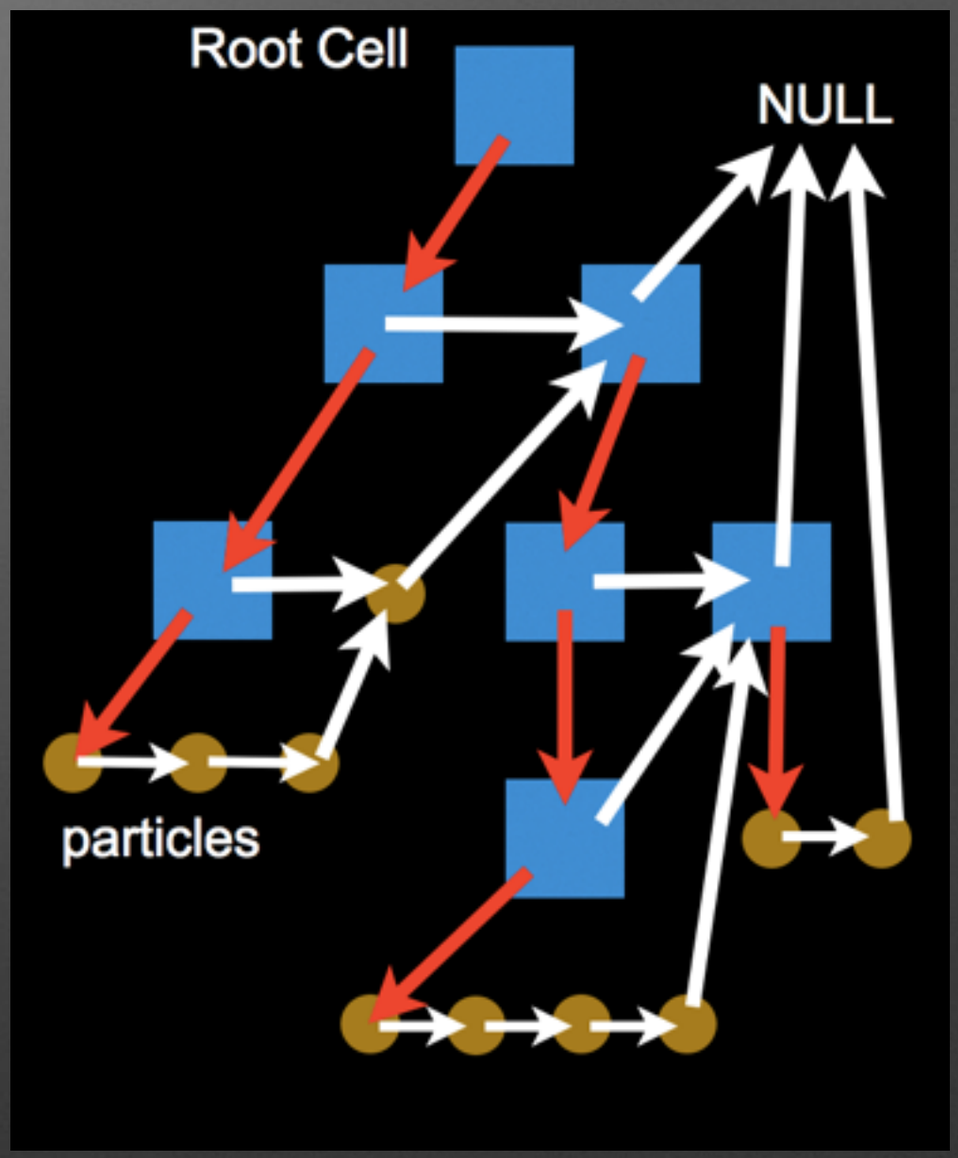
# Octreeによる重力計算

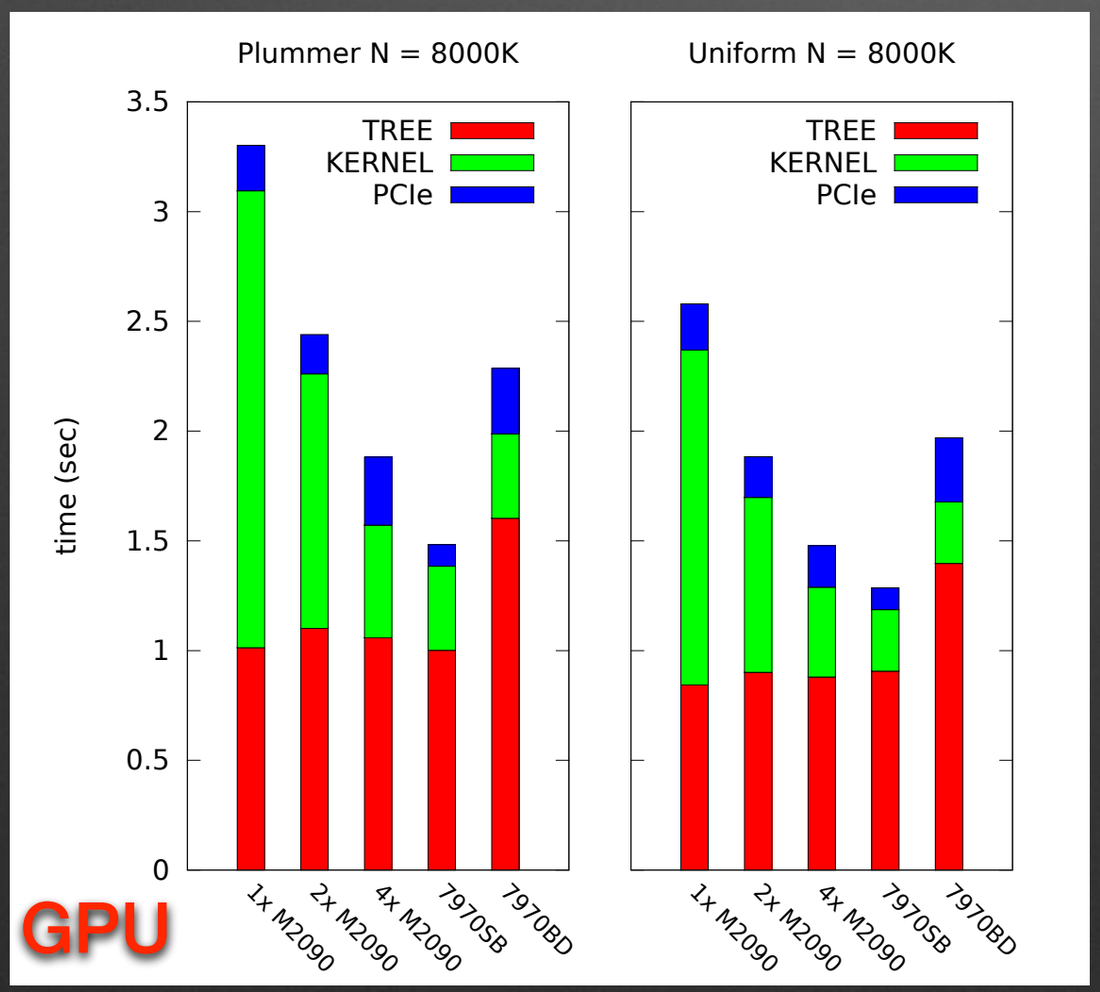
- 粒子によるポテンシャルを多重極展開
  - 遠方の粒子集団にポテンシャルを近似することができる
  - Octreeにより距離判定の高速化
  - 他の応用として近傍粒子探索の高速化も可能



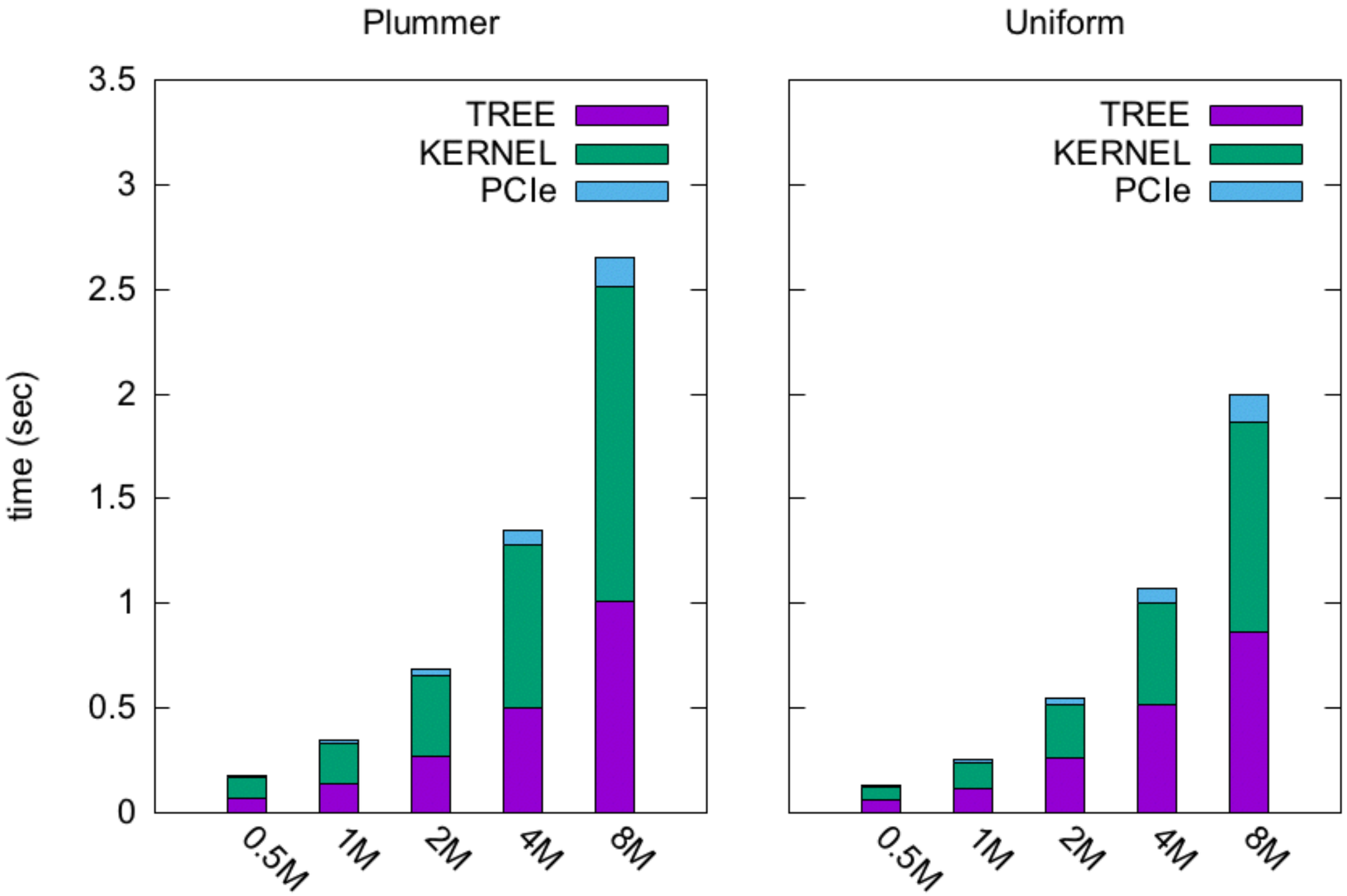
# Octreeの並列走査

- OctreeをOpenCLカーネルで走査
  - ツリー構造をlinked listに変換
  - ツリー走査をループにする
  - キャッシュが有効なら高性能
  - 実用的には八分木である必要もない



# 重力計算の性能評価


Plummer
Uniform
TREE
KERNEL
PCIe
time (sec)
3.5
3
2.5
2
1.5
1
0.5
0
0.5M
1M
2M
4M
8M

# SPH法のOpenCL実装

- SPH法による白色矮星シミュレーション
  - Octreeによる近傍粒子探索とSPHカーネル総和計算
  - 流体の圧力を状態方程式から計算
  - 軌道の数値積分

$$\rho_i = \sum m_j W(\boldsymbol{r}_i - \boldsymbol{r}_j; h)$$

$$\frac{D\boldsymbol{v}_i}{Dt} = -\sum m_j (\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}) \nabla W(\boldsymbol{r}_i - \boldsymbol{r}_j; h) - (\nabla \Phi)_i.$$

$$\frac{Du_i}{Dt} = \frac{1}{2} \sum m_j (\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}) (\boldsymbol{v}_i - \boldsymbol{v}_j) \nabla W(\boldsymbol{r}_i - \boldsymbol{r}_j; h)$$

# SPH法の性能評価

| $P$ | 0.5M | 1M | 2M | 4M |
|---|---|---|---|---|
| 1 | 3.922895e-01 | 8.193518e-01 | 1.662591e+00 | 3.382097e+00 |
| 2 | 2.756883e-01 | 5.440593e-01 | 1.086922e+00 | 2.210079e+00 |
| 4 | 2.216704e-01 | 4.461828e-01 | 9.096476e-01 | 1.843587e+00 |
| 8 | 2.214832e-01 | 4.851834e-01 | 9.945402e-01 | 2.058704e+00 |

表 **1**　PEZY-SC プロセッサでの SPH 法による白色矮星シミュレーションの性能評価．最初の列は利用した PEZY-SC プロセッサの数を示す．計算時間の単位は秒．

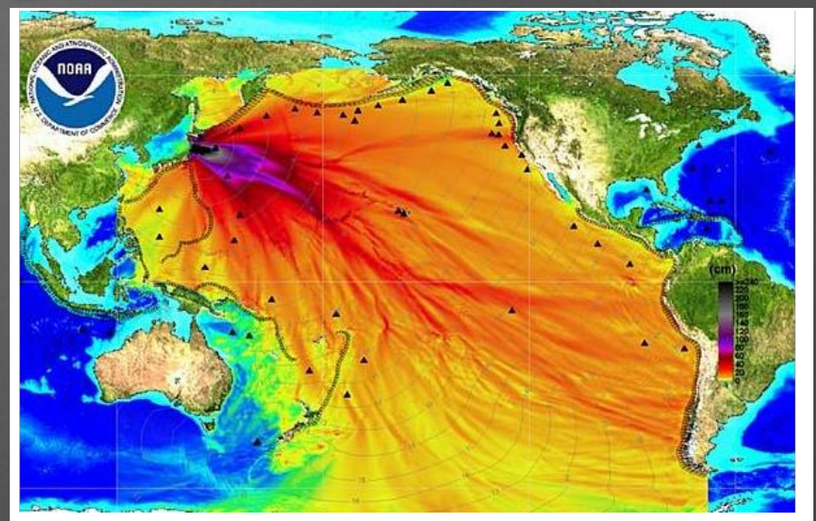
- 白色矮星シミュレーションの性質
  - Octree構築と状態方程式計算はHOST CPU
  - 他の部分はPEZY-SCで並列化
  - N = 4Mの場合、HOST部分には約1.4秒
  - P = 1,2,4,8の時、カーネル実行時間は2, 0.8, 0.4, 0.6秒
  - P = 4までは複数PEZY-SCでの並列化が有効

# FDPSの性能評価(テスト中)

```
hpc1[fdps/sample/nbody_pl] OMP_NUM_THREADS=20 ./nbody.out -T 1.0 -I ../init_data/phi_pl64k
platform 0 PEZY PZCL 2.0.2.11498
        device 0 PEZY-SC
        device 1 PEZY-SC
        device 2 PEZY-SC
        device 3 PEZY-SC
        device 4 PEZY-SC
        device 5 PEZY-SC
        device 6 PEZY-SC
        device 7 PEZY-SC
Selected:
     //==================================\\
     ||                                  ||
     || ::::::: ::::::. ::::::. .::::::. ||
     || ::      ::    : ::    : ::       ||
     || ::::::  ::    : ::::::'  `:::::. ||
     || ::      ::::::' ::      `......' ||
     ||     Framework for Developing     ||
     ||        Particle Simulator        ||
     ||     Version 1.1 (2015/08)        ||
     \\==================================//

       Home   : https://github.com/fdps/fdps
       E-mail : fdps-support@mail.jmlab.jp
       Licence: MIT (see, https://github.com/FDPS/FDPS/blob/master/LICENSE)
       Note   : Please cite Iwasawa et al. (in prep.)

       Copyright (C) 2015
         Masaki Iwasawa, Ataru Tanigawa, Natsuki Hosono,
         Keigo Nitadori, Takayuki Muranushi, Junichiro Makino
         and many others
******** FDPS has successfully begun. ********
time_end = 1
../init_data/phi_pl64k
./result/t-de.dat
Number of processes: 1
Number of threads per process: 20
PEZY PZCL 2.0.2.11498 ::PEZY-SC 0


Build options ::
np_ave=65536
used mem size for tree=147456384
used mem size for tree(0)=0.147456384
-0.499334146704456
time:  0.0000000 energy error: -0.000000e+00 -2.493341e-01 2.500000e-01 -4.993341e-01
time:  0.1250000 energy error: -3.130508e-05 -2.493263e-01 2.499216e-01 -4.992480e-01
```

# MOSTの性能評価 (1)

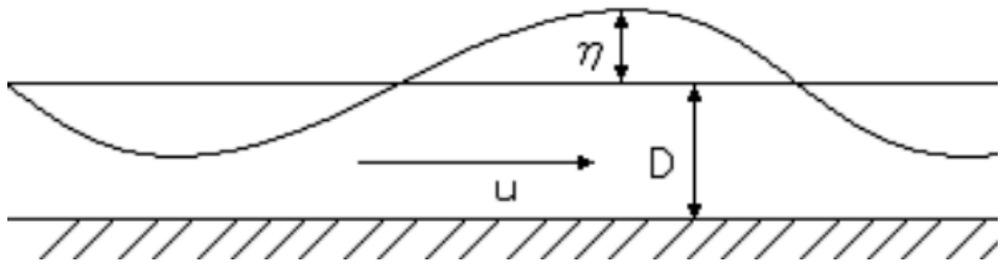
- Method of Tsunami Splitting (MOST)
  - NOAAで利用されている
  - 浅水方程式(偏微分方程式)を解く解法
  - 差分法 & 次元方向に演算子分離法
  - 時間方向にはEuler法



Tsunami image from the Japan Tsunami in 2011, © NOAA

$$\begin{cases} H_t + (uH)_x + (vH)_y = 0 \\ u_t + uu_x + vu_y + gH_x = gD_x \\ v_t + uv_x + vv_y + gH_y = gD_y \end{cases}$$

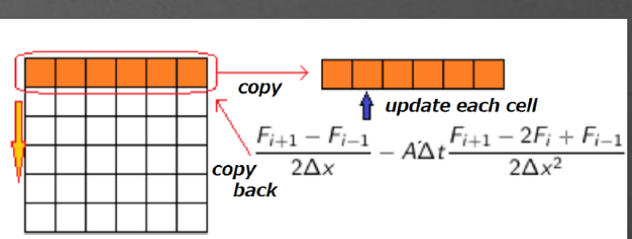
- H : 全波高 (= D + η )
- η : 波の高さ
- D : 水深
- u, v : 各方向の速度成分
- g: 重力加速度



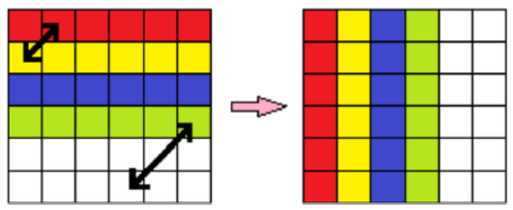
# MOSTの性能評価 (2)

- MOSTは各方向に3点ステンシル計算
  - MIC, GPUなどでの並列化について一部既報告
  - FPGAでの実装についても一部既報告
  - OpenMP, OpenACC, CUDA, OpenCLによる並列化と性能評価

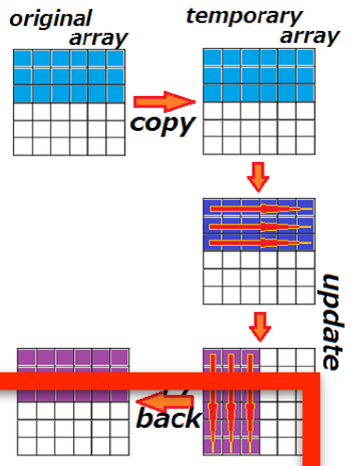
▸ Code1（オリジナル）
・1行または1列分のデータをコピーしてきて処理する
・並列度は O(N)



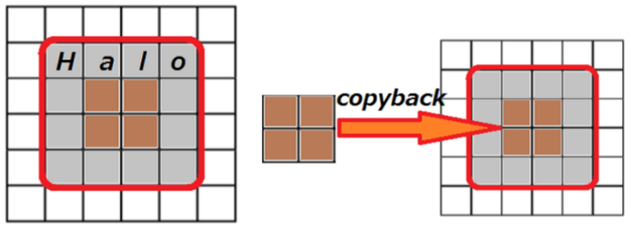

▸ Code2（転置）
・Code1におけるメモリアクセス効率の向上を図る
・並列度はO(N)



▸ Code3（2-D temporary array)
・Code4のための前準備
・データコピーの回数を軽減
・並列度はO(N^2)




▸ Code4（ブロック化）
・高い並列性が得られる（特にGPU向け）
・並列度はO((N/B)^2)　　B：ブロックサイズ



# MOSTの性能評価 (3)

| $N_x$ | 1 ステップ | 格子点あたり |
|---|---|---|
| 500 | 6.544603e-03 | 2.617841e-08 |
| 1000 | 2.504481e-02 | 2.504481e-08 |
| 2000 | 9.910859e-02 | 2.477715e-08 |
| 3000 | 2.213808e-01 | 2.459787e-08 |
| 4000 | 3.926668e-01 | 2.454167e-08 |
| 5000 | 6.118429e-01 | 2.447372e-08 |
| 6000 | 8.716483e-01 | 2.421245e-08 |
| 7000 | 1.197728e+00 | 2.444343e-08 |
| 8000 | 1.520423e+00 | 2.375661e-08 |
| 10000 | 2.428314e+00 | 2.428314e-08 |

問題サイズ：$N_x^2$の領域

条件：300ステップ

ブロックサイズ 1 x 1

- 格子点当たりの性能比較
- K20c (Tesla)
  - OpenCL ~ 8.0e-9 sec
  - CUDA ~ 2.5e-8 sec
  - CUDA(Shm) ~ 5.0e-9 sec
- R280X (Radeon)
  - OpenCL ~1.3e-9 sec

GPUの方が現状では高速

メモリ帯域の差が大きい

# PEZY-SC/PZCLの現状

- PZCLへのOpenCLコード移植は容易
  - OpenCLとはほぼ互換
  - ソースコードは共通化可能
  - 違い：オフラインコンパイルのみ
  - カーネル組み込み関数のサポート不足(rsqrt()/sqrt()のみ)
  - 共有メモリを利用したコードの取り扱い??

- **課題**
  - メモリ帯域が最新のGPUより遅い
  - SFUが比較的に少ないため、除算などが相対的に低速
  - PZCL コンパイラの最適化

# まとめ

- **Suirenで計算科学アプリの性能を評価した**
  - Hermite積分法：GPUと比べると若干遅い
  - Octree法：GPUとあまり遜色のない性能
  - MOST法：メモリ帯域に律速されている
  - 多倍長精度演算：GPUと遜色のない性能

- **今後の課題**
  - アーキテクチャに特化した最適化の調査
    - GPU用のコードでも比較的性能はよい
  - 他の計算科学アプリケーションの実装
  - 大規模計算の実現