重力多体問題専用計算機GRAPE -その動作原理と使用方法

川井敦(KFCR)
台坂博(一橋大)

講義内容

- イントロダクション
- 動作原理
- 国立天文台GRAPEシステム
- GRAPE-7 ライブラリ関数
- 最近のトレンド
- まとめ

GRAPE上のツリー法については高度なN体シミュレーション法で解説

N体シミュレーションの計算コスト

$$\frac{d\mathbf{v}_i}{dt} = -\sum_{i \neq j} \frac{Gm_j(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

- 軌道積分には運動方程式を評価する必要
- 粒子同士の重力相互作用のペア数
  - 粒子数の2乗に比例
  - 軌道積分は粒子数に比例
- N=10000体の場合の計算量の比較
  - 重力計算 ~ 10^8 → 10 sec (if 10^{-7} sec per interaction)
  - 軌道積分 ~ 10^4 → 10^{-3} sec

重力計算が全計算量の大半を占める

高速化のポイント

いかに速く重力計算を行うか

手法:

- 計算量を減らすアルゴリズム:
  - –ツリー法、エバルト法、PM法、独立時間刻み(ITS)、、、
- 並列化:
  - –PCクラスター、最近ほとんどのスパコン、、、
- 計算機自体の高速化:
  - –クロックアップ、マルチコア化、ベクトル演算機、専用化、、、

講義内容

- イントロダクション
- 動作原理
- 国立天文台GRAPEシステム
- GRAPE-7 ライブラリ関数
- 最近のトレンド
- まとめ

GRAPEの設計思想

軌道積分、他

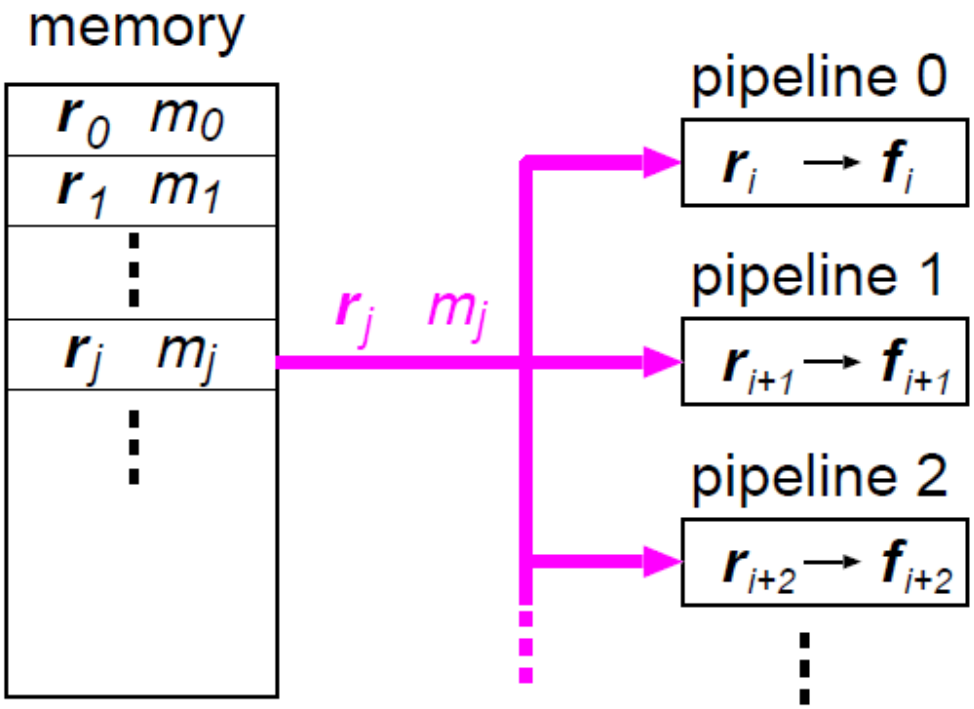
$O(N)$

粒子間相互作用

$O(N^2)$

専用機は重力計算だけを計算する

基本構成



重力計算パイプライン(レジスタ)＋粒子メモリ

- 1回に計算できる粒子数はメモリサイズに依存

重力パイプライン

$$\sum_{i \neq j}^{N} \frac{m_j(\mathbf{r}_j - \mathbf{r}_i)}{(|\mathbf{r}_j - \mathbf{r}_i|^2 + \varepsilon^2)^{3/2}}$$

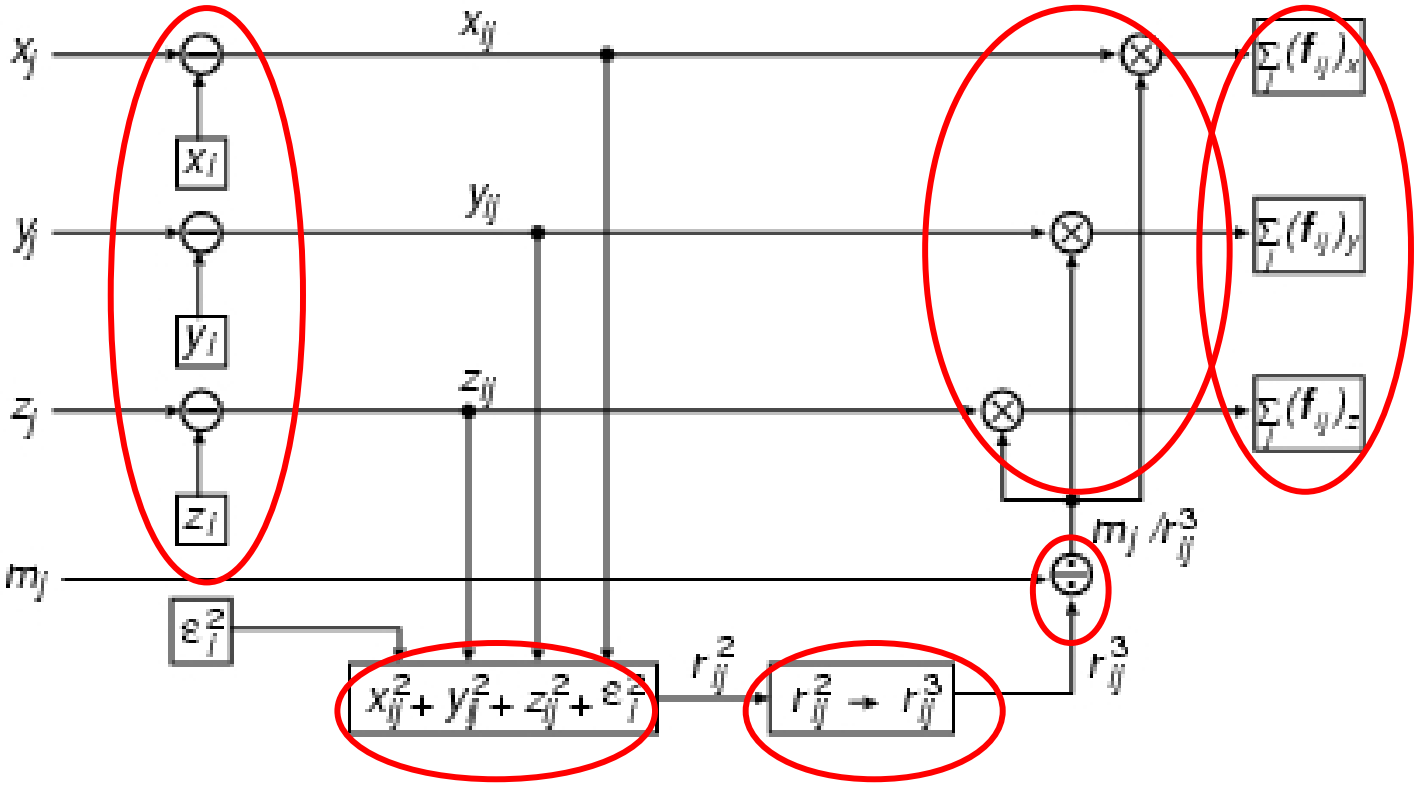


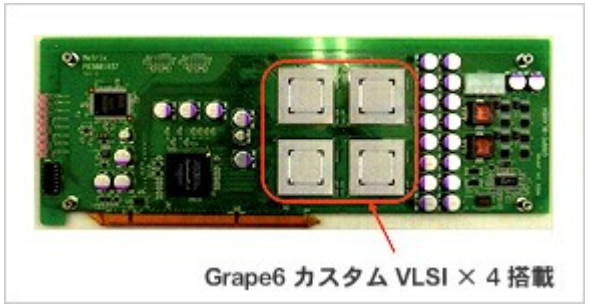
図 4.2: 重力計算パイプライン

必要な演算(約40演算)を同時に行う

汎用機よりも速い理由

- トランジスターの利用効率が高い
  - GRAPEの1チップは100演算以上の計算を同時に行っている
  - PCのCPUは何億トランジスターを使ってもオーダー1の演算

3GHz



100MHz

効率化のための工夫

- 数値フォーマットの最適化

  各演算に適したフォーマットと、必要最小限の語長を選べる

- パイプライン化

  複数の粒子からの重力計算をパイプライン処理できる

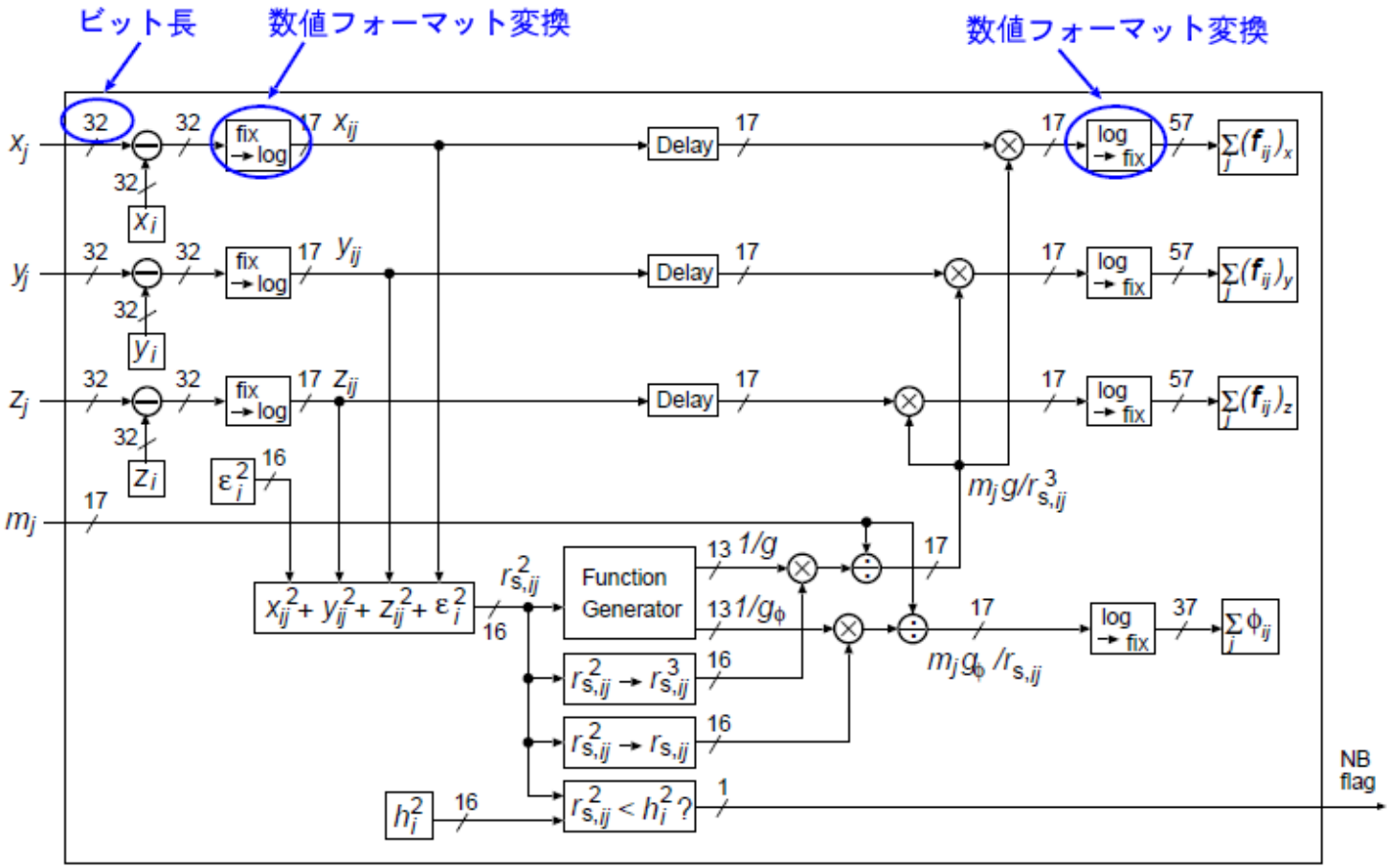
  (複数の演算を異なるj粒子に対して並列実行できる)

- パイプラインの並列化

  複数の粒子への重力計算を並列実行できる。特にメモリの動作速度をあげずにパイプライン数を増やせる

計算対象が決まっていること、計算対象に高い並列性があることを利用している

フォーマットの最適化



ビットを減らすことでパイプライン回路を小さくして、1つのチップに複数のパイプラインを実装

GRAPEファミリ

GRAPE-7

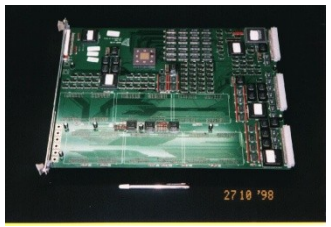
GRAPE-DR

GRAPE-6

MDM
(MDGRAPE-2 + WINE-2)

'99 GRAPE-5

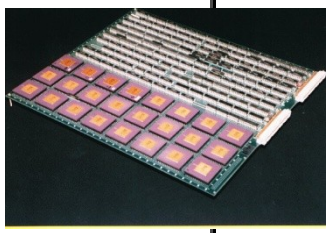
PROGRAPE-1
Arbitrary Interaction



GRAPE-4

MD-GRAPE

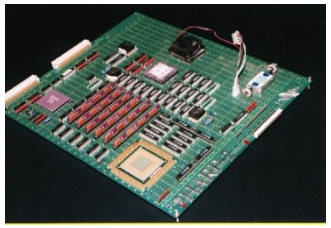
SPH



HARP-2

HARP-1

WINE-1
Ewald



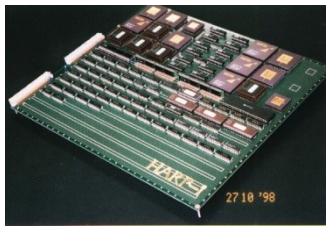
GRAPE-3

GRAPE-2A

GRAPE-1A

GRAPE-2

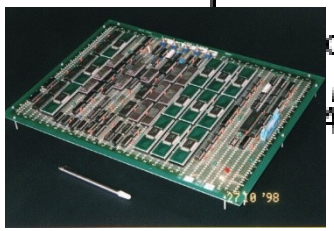
Arbitrary Central Force



Full Accuracy

GRAPE-1

Limited Accuracy



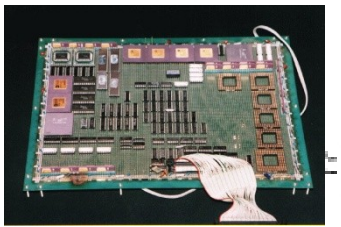


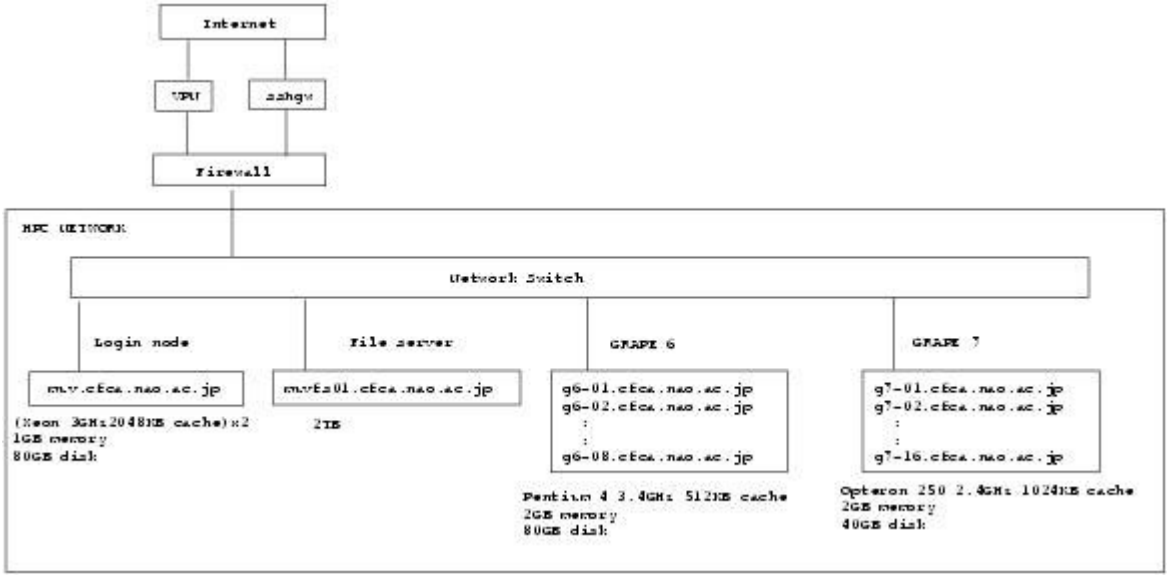
図 4.3:

ルールがくずれつつある...

講義内容

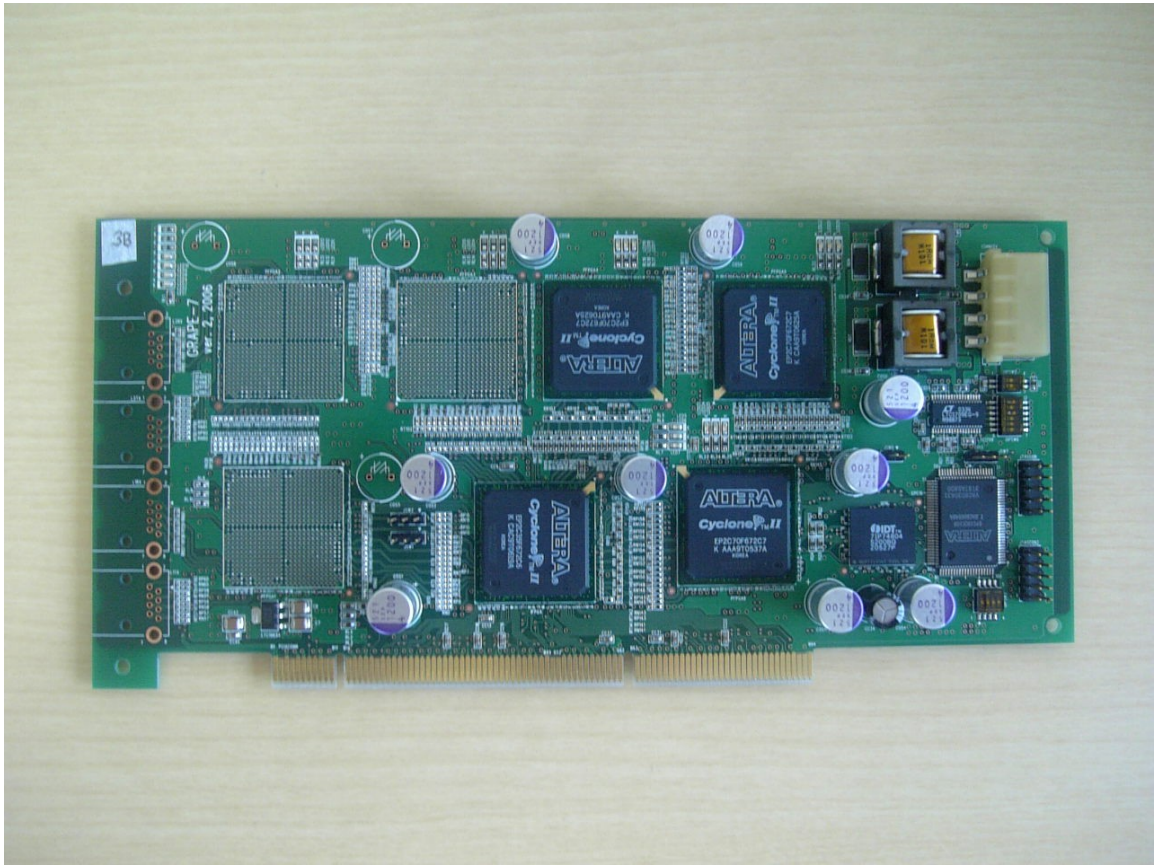
- イントロダクション
- 動作原理
- 国立天文台GRAPEシステム
- GRAPE-7 ライブラリ関数
- まとめ

国立天文台GRAPEシステムネットワーク構成



- 現在運用で使われているのは、GRAPE-7 model 300
- ホスト計算機
  - AMD Opteron 250, 2GB memory (古！！)
  - OS：CentOS 4.7 x86_64

GRAPE－7ボード概観（KFCR社製）



- GRAPE―5の後継機、低精度用GRAPE
- カスタムチップの代わりに再構成可能デバイス(FPGA)を使用
- 高速インターフェース（PCI-X）の採用

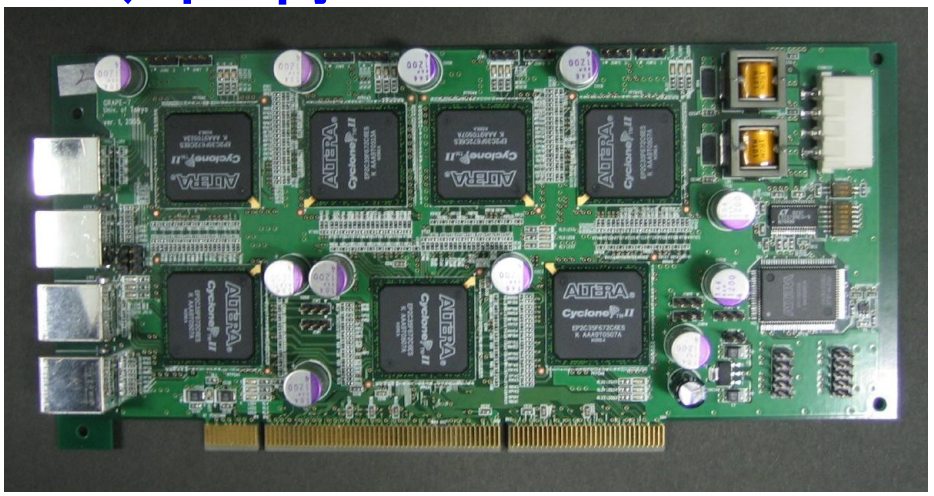
GRAPE－7（model 300）仕様

ピーク演算性能	182Gflops
データ転送速度	実行最大700MB/S
•重力演算チップ数	ボードあたり3個(Altera CyclonII EP2C70)
•重力演算パイプライン数	チップあたり16、ボードあたり48
•ボード動作周波数	100MHz
•仮想パイプライン数	ボードあたり256
•ホスト計算機との接続	PCI-X 100MHz
•ホスト計算機とのデータ転送速度	ピーク800MB/s、実行最大　700MB/s
•粒子メモリ	12k粒子
•近接粒子リストメモリ	1パイプライン当り72
•相互作用	カットオフをもつ中心力
•計算精度	pairwiseの相対誤差 0.3%

GRAPE－7（model 600）仕様

ピーク演算性能　364Gflops

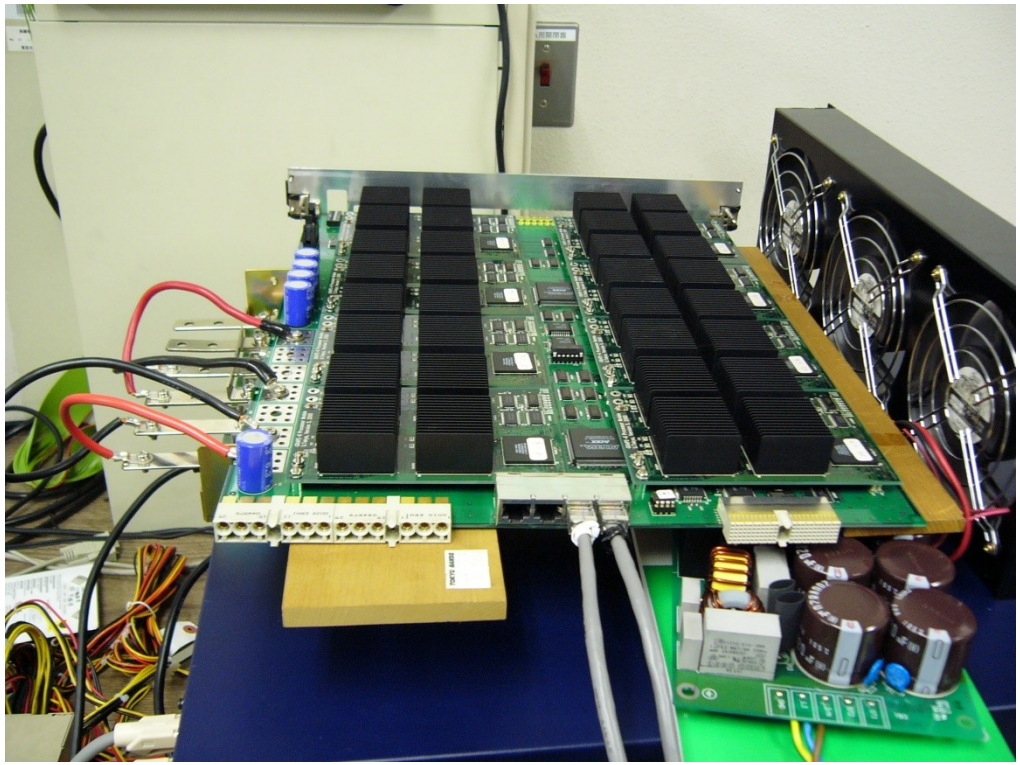
データ転送速度　最大700MB/S



- 重力演算チップ数　ボードあたり6個
- 重力演算パイプライン数　チップあたり16、ボードあたり96
- ボード動作周波数　100MHz
- 仮想パイプライン数　ボードあたり256
- ホスト計算機との接続　PCI-X 100MHz
- ホスト計算機とのデータ転送速度　ピーク800MB/s、実行最大　700MB/s
- 粒子メモリ　24k粒子
- 近傍粒子リストメモリ　1パイプライン当り144

GRAPE－6仕様

- 高い計算精度とindividual timestep法のための機能を実装
  - 加速度の時間微分を計算する機能
  - 予測子を計算する機能
- ピーク演算速度： 1037Gflops （32チップ搭載ボードの場合）



講義内容

- イントロダクション
- 動作原理
- 国立天文台GRAPEシステム
- GRAPE-7 ライブラリ関数
- 最近のトレンド
- まとめ

計算手順

1. 全粒子の座標と質量をGRAPE上のメモリに書き込む（j粒子：力を与える粒子）
2. パイプライン本数分の粒子（i粒子：力を受ける粒子）の座標、ソフトニング値をGRAPE上のパイプラインレジスターに書き込む
3. 計算開始信号をGRAPEに送る
4. 計算を終了を待つ
5. 計算結果をGRAPE上のパイプラインレジスタから読み出す
6. すべての粒子(i粒子)について計算を行う

コーディング例（直接計算法）

```
/* variable declaration */
static double mj[NMAX], xj[NMAX][3];
static double eps2, acc[NMAX][3], pot[NMAX];
double xmax = 10.0, xmin = -10.0, mmin = 0.001;
int i, n, nn, step, nstep;
int npipes = g5_get_number_of_pipelines();
.................

g5_open();
g5_set_range(xmin, xmax, mmin);
for (step = 0; step < nstep; step++) {
    /* force calculation */
    g5_set_jp(0, n, xj, mj);
    g5_set_eps2_to_all(eps2);
    g5_set_n(n);
    for (i = 0; i < n; i += npipes) {
        nn = npipes;
        if (i + nn > n) {
            nn = n -i;
        }
        g5_calculate_force_on_x(xj+i, acc+i, pot+i, nn);
    }
    /* orbital integration */
    .................
}
g5_close();
```

GRAPE－7ユーザーライブラリ

- ヘッダファイル

/misc/local/g7pkg2.1/include/g5nbutil.h

- ライブラリ

/misc/local/g7pkg2.1/lib/libg75nb.a
/misc/local/g7pkg2.1/lib/libhib.a

- マニュアル

/misc/local/g7pkg2.1/doc/g5user-j.pdf
/misc/local/g7pkg2.1/doc/g5nbuser-j.pdf

C, C++,Fortrunから使用可能

基本的にG5互換

座標、質量のスケーリングに関する注意

座標の最大値 $xmax$、最小値 $xmin$

全粒子のとり得る座標成分の上限と下限を指定する。

座標の分解能は $10^{-10} \times rmax$ 程度になる

$(rmax \equiv xmax - xmin)$。

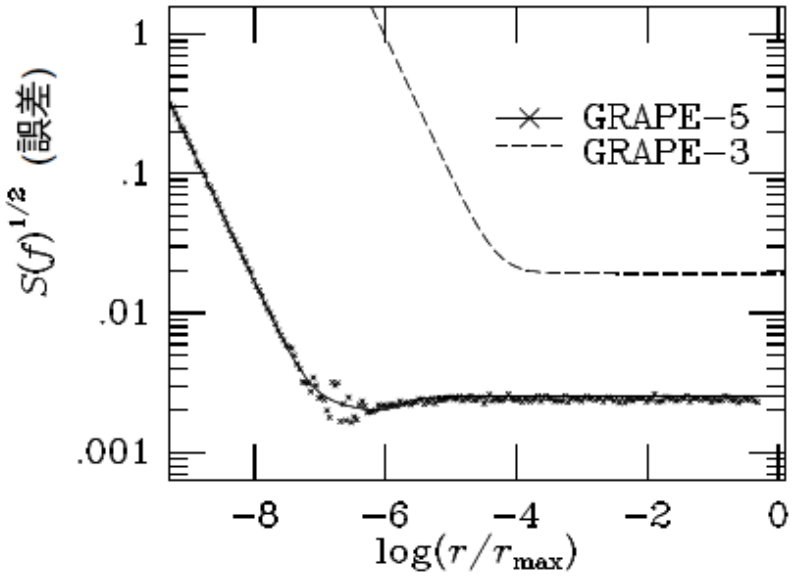
質量の最小値 $mmin$

質量の分解能を指定する。

このときソフトニング値は

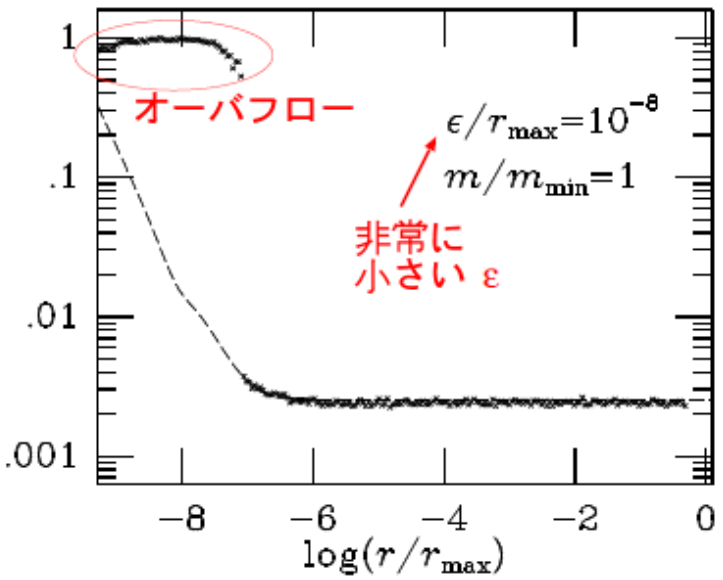
$$\epsilon \gtrsim 10^{-7} \times \left(\frac{m}{mmin}\right)^{1/2} rmax$$

となるように設定する。



$\epsilon/rmax = 10^{-6}$

$m/mmin = 1$



$\epsilon/rmax = 10^{-8}$

$m/mmin = 1$

コーディング例(近傍粒子探索)

```
static int nblist[NPIPESMAX][NBMEMSIZE], nnb[NPIPESMAX];
h = 0.01;
.................
g5_set_h_to_all(h);
.................
    for (i = 0; i < n; i += npipes) {
        nn = npipes;
        if (i + nn > n) nn = n - i;
        g5_calculate_force_on_x(xj+i, acc+i, pot+i, nn);
        nbof = g5_read_neighbor_list();
        if (nbof == 1) {
             fprintf(stderr, "NB list overflow\n");
        }
        for (ii = 0; ii < nn; ii++) {
             nnb[ii] = g5_get_neighbor_list(ii, nblist[ii]);
        }
    }
.....................
```

- プロセッサー内の1パイプライン当りに保存できる数(24粒子)を超えないように注意が必要

講義内容

- イントロダクション
- 動作原理
- 国立天文台GRAPEシステム
- GRAPE-7 ライブラリ関数
- 最近のトレンド
- まとめ

SIMD

- 今のコンピュータのほとんどはノイマン型
  - 命令も読み込む必要

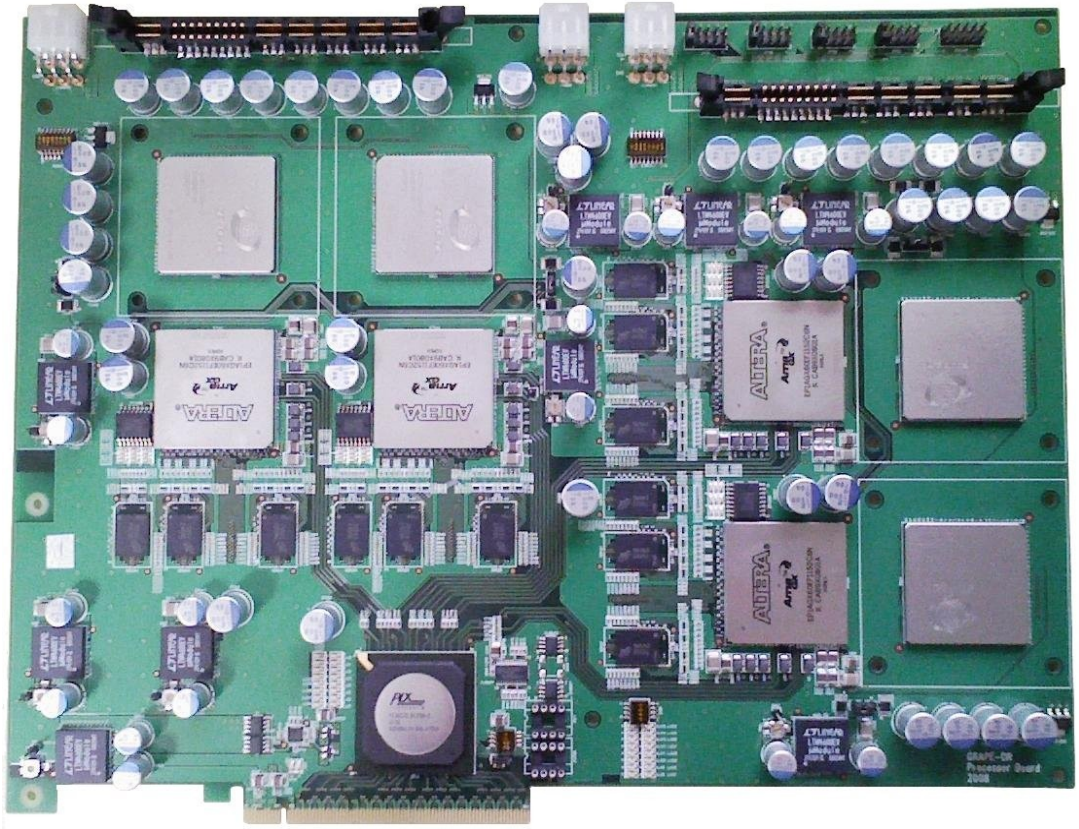
    a+b → aとbと”+”も読み込む
- CPU-メモリ転送がボトルネック
- 1回の命令で複数のデータ処理すれば効率があがる
  - a[3..0] + b[3..0] とベクトル的に処理
  - Single Instruction Multiple Data(SIMD)

CPU/GPUの高速化の主流な方法

“GRAPE” on SIMD

- CPU
  - MMX/SSE/AVX
  - AVXではdouble 4要素に同時演算
  - Phantom-GRAPE by K.Nitadori
- GPU
  - Processor Element(PE)を多数ならべて画像処理
  - 科学技術計算も可能
  - スパコンでも使われる(Tsubame, 出島、天河,,,)
  - 重力計算(nVidia by Nitadori, ATI by Nakasato)
- GRAPE-DR

GRAPE-DR



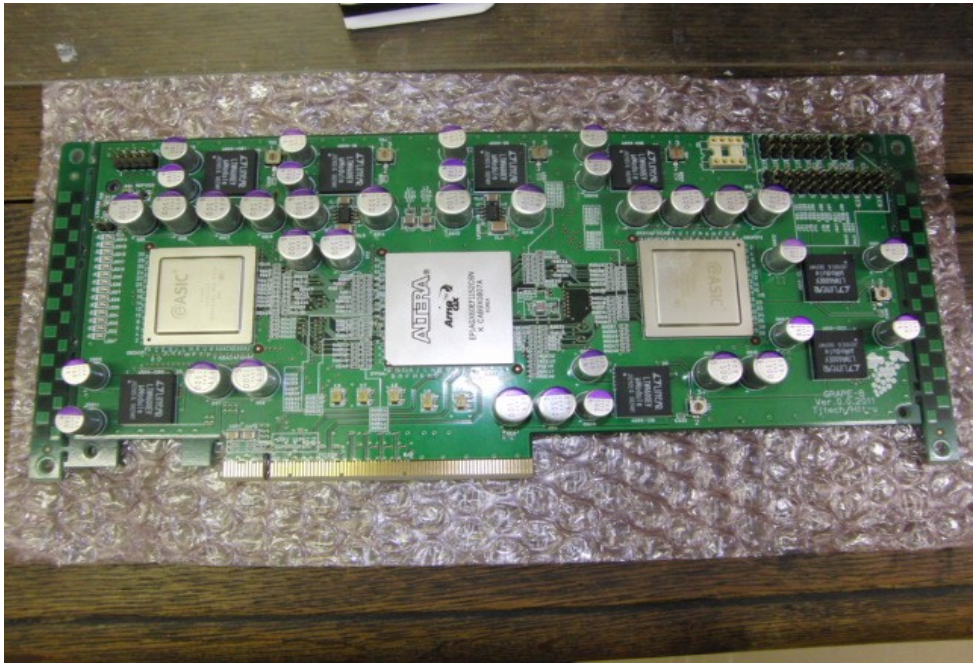
•512PE on chip

•ピーク性能：1.8Tflops(倍精度加算単精度乗算),

0.925Tflops(倍精度加算倍精度乗算)

・キャッシュなどの分の回路リソースもPEに→（電力）効率が高い

GRAPE-8



-Structured-ASIC上に重力パイプライン(低精度型)を実装

FPGAより効率がよい(より多くのパイプラインが入る)

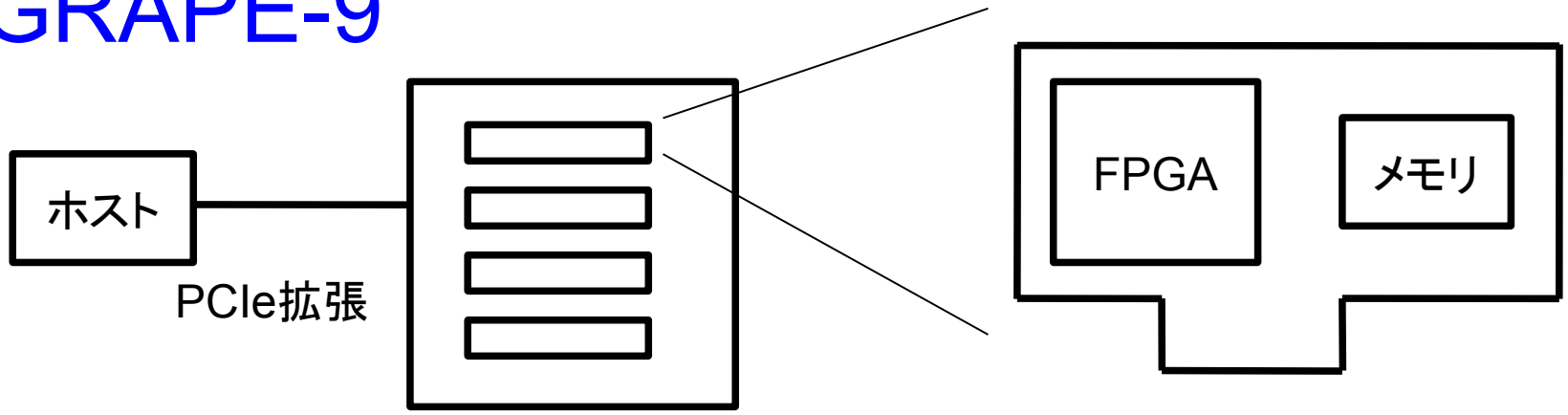
-ボード: 2個のG8 チップ,制御用FPGA,電源モジュール

-性能: 48 パイプライン 250MHz駆動→450Gflops/chip

ボードあたり 900Gflops/45W

-IF: PCIe Gen1 x8

GRAPE-9



-GRAPE-7の後継機、KFCRで開発中

(FPGAベース、という意味で)

-PCIe拡張ボックスに多数のFPGAボードが入ったもの

-個々のボード: FPGA + SO-DIMMメモリ

-FPGAに重力パイプライン(低/高精度型の両方)

-来年度にMUVに導入(のはず)

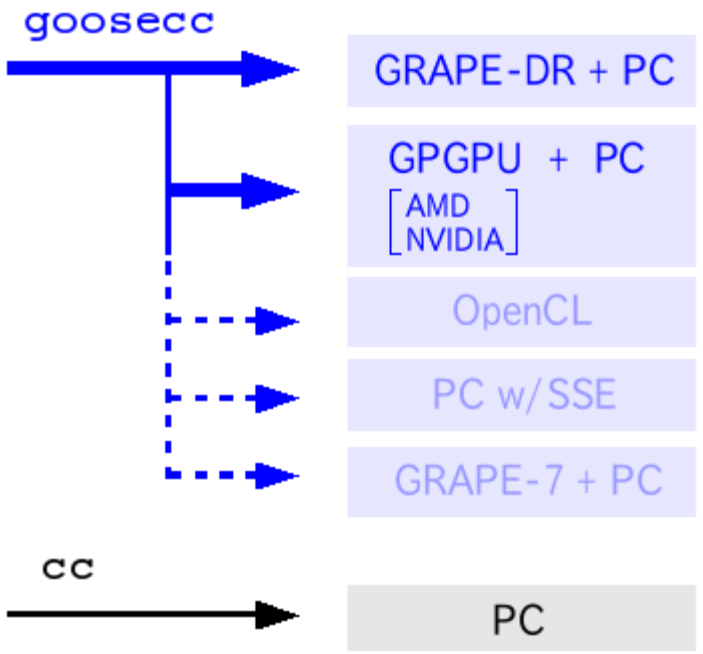
開発環境

- 重力計算については誰かがライブラリーを作ればよいが…
- GPGPU
  - NVIDIA　CUDA
  - ATI（AMD）　ATI stream SDK(brook+, CAL, IL)
- より汎用的な環境
  - OpenCL(ベンダーに依存しないプラットホーム)
  - LSUMP(ATI/GRAPE-DR用コンパイラー by 中里さん)
  - Goose(ドメイン特化型コンパイラ　by KFCR)

Goose記述例

```
#pragma goose parallel for loopcounter(i, j) \
precision("double")
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
       for (k = 0; k < 3; k++) {dx[k] = x[j][k] - x[i][k]; }
       r2 = dx[0] * dx[0] + dx[1] * dx[1] + dx[2] * dx[2]

         + eps2;
       rinv = rsqrt(r2);
      mrinv = m[j] * rinv;
      mr3inv = mrinv * rinv * rinv;
      for (k = 0; k < 3; k++) {
         a[i][k] += mr3inv * dx[k];
      }
      pot[i] -= mrinv;
    }
 }
```



まとめ

- GRAPEの動作原理
- 国立天文台GRAPEシステム
- GRAPE－7関数ライブラリの使用方法
- 最近のトレンド

について説明した