

重力多体問題専用計算機 GRAPE

ー その動作原理と使用方法

福重 俊幸

(株)K&F Computing Research

イントロダクション

動作原理(専用アクセラレータ)

動作原理(汎用アクセラレータ)

アクセラレータの特色

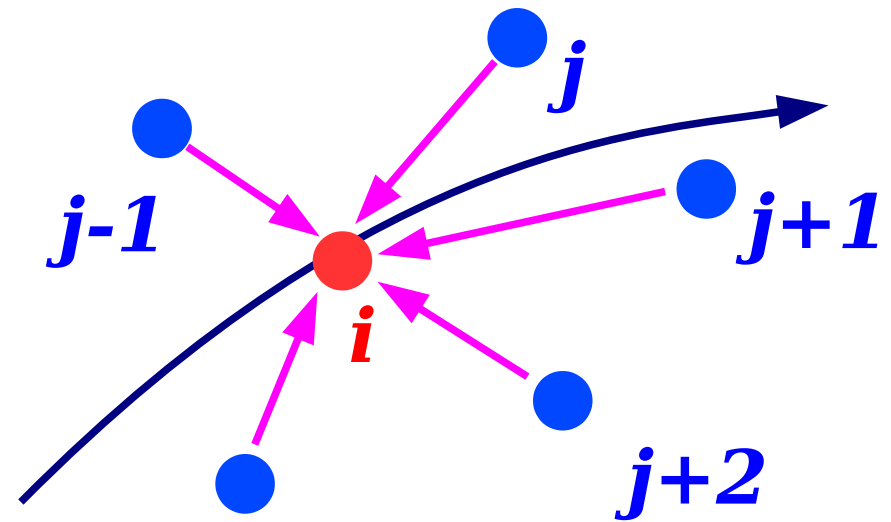
基礎的な使用法(実際)

まとめ

多体シミュレーションの計算コスト

$$\left\{ \begin{array}{l} \frac{d^2 \vec{r}_i}{dt^2} = \vec{f}_i \quad O(N) \text{ 粒子数} \\ \text{(運動方程式)} \end{array} \right.$$

$$\left\{ \begin{array}{l} \vec{f}_i = \sum_{j \neq i}^N G \frac{m_j (\vec{r}_j - \vec{r}_i)}{|\vec{r}_j - \vec{r}_i|^3} \quad O(N^2) \\ \text{(重力計算)} \end{array} \right.$$



重力計算が全計算量の大半を占める。

高速化の手法

計算量を減らす(アルゴリズム的に)

ツリー法、 P^3M 法など (重力計算の節約)

独立時間刻み法など (時間積分計算の節約)

より高速な計算機をつかう

CPU内並列化(AVX,コア)、ノード並列化

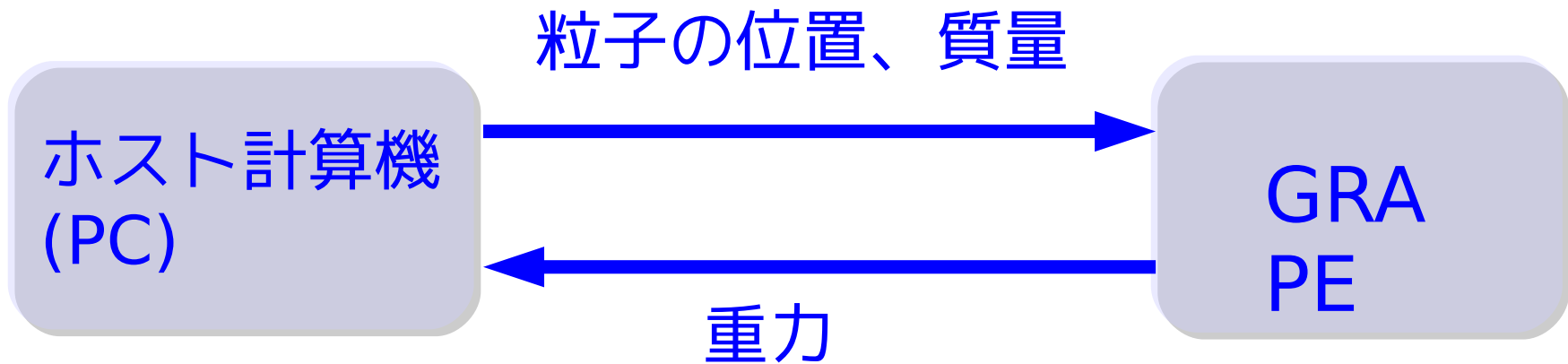
専用アクセラレータ(GRAPE)、

汎用アクセラレータ(GPU, GRAPE-DR)

その両方

多体問題専用計算機 GRAPE

(専用アクセラレータ)



重力計算に特化したハードウェア。

イントロダクション

動作原理(専用アクセラレータ)

動作原理(汎用アクセラレータ)

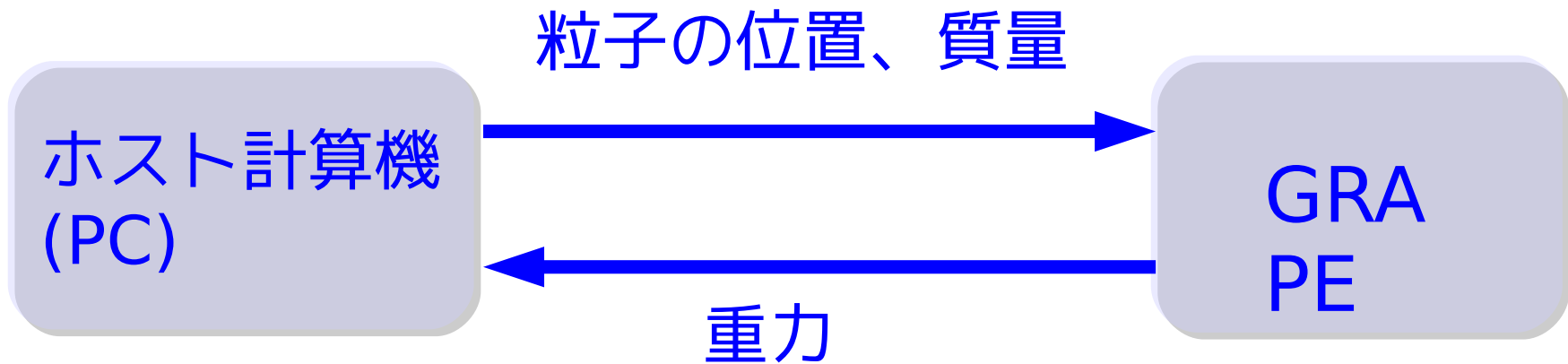
アクセラレータの特色

基礎的な使用法(実際)

まとめ

多体問題専用計算機 GRAPE

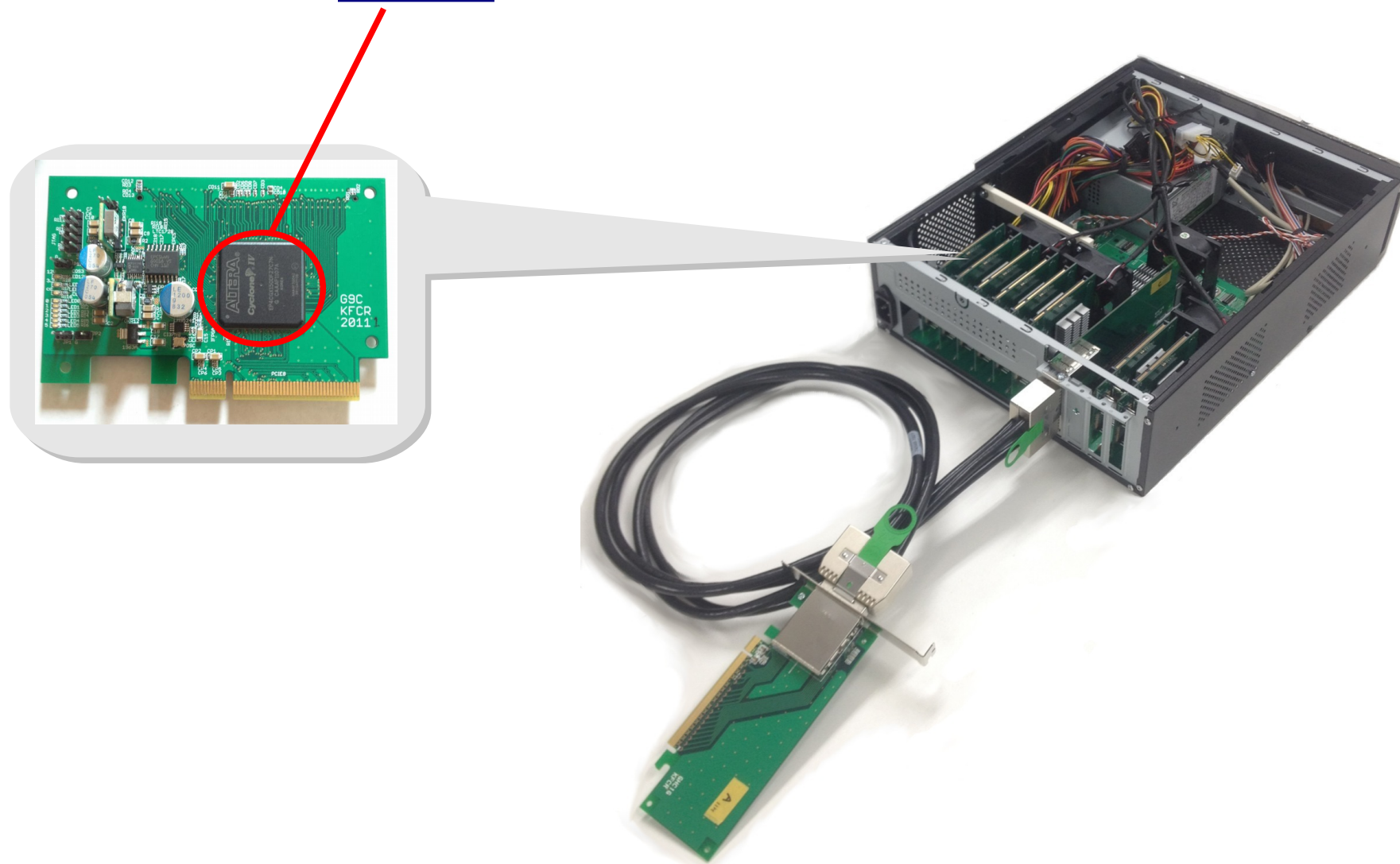
(専用アクセラレータ)



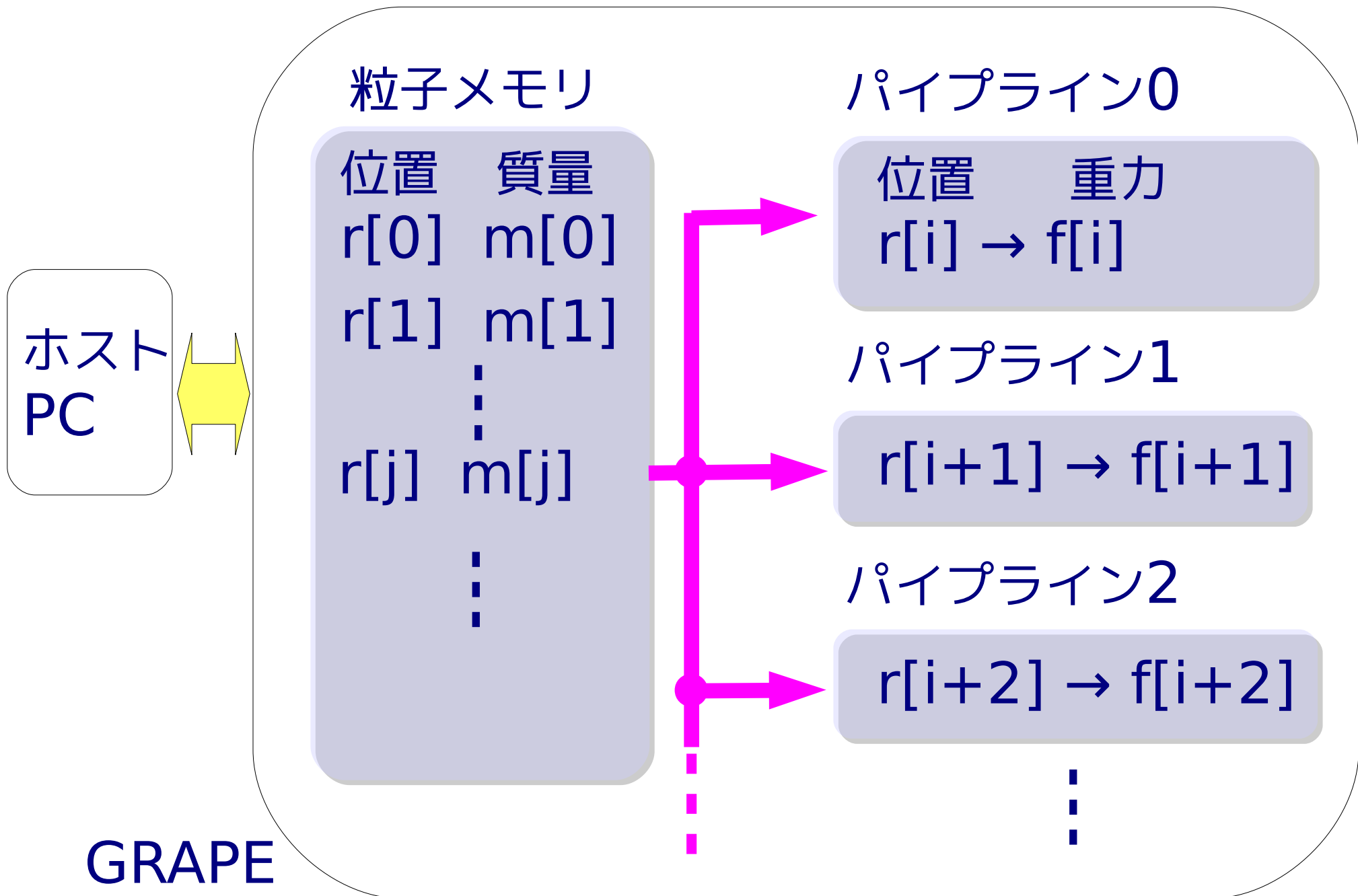
重力計算に特化したハードウェア。

GRAPE-9 model 800/5000

重カパイプライン (GRAPE-5, GRAPE-6 互換)
を集積したFPGAを8-16個搭載。

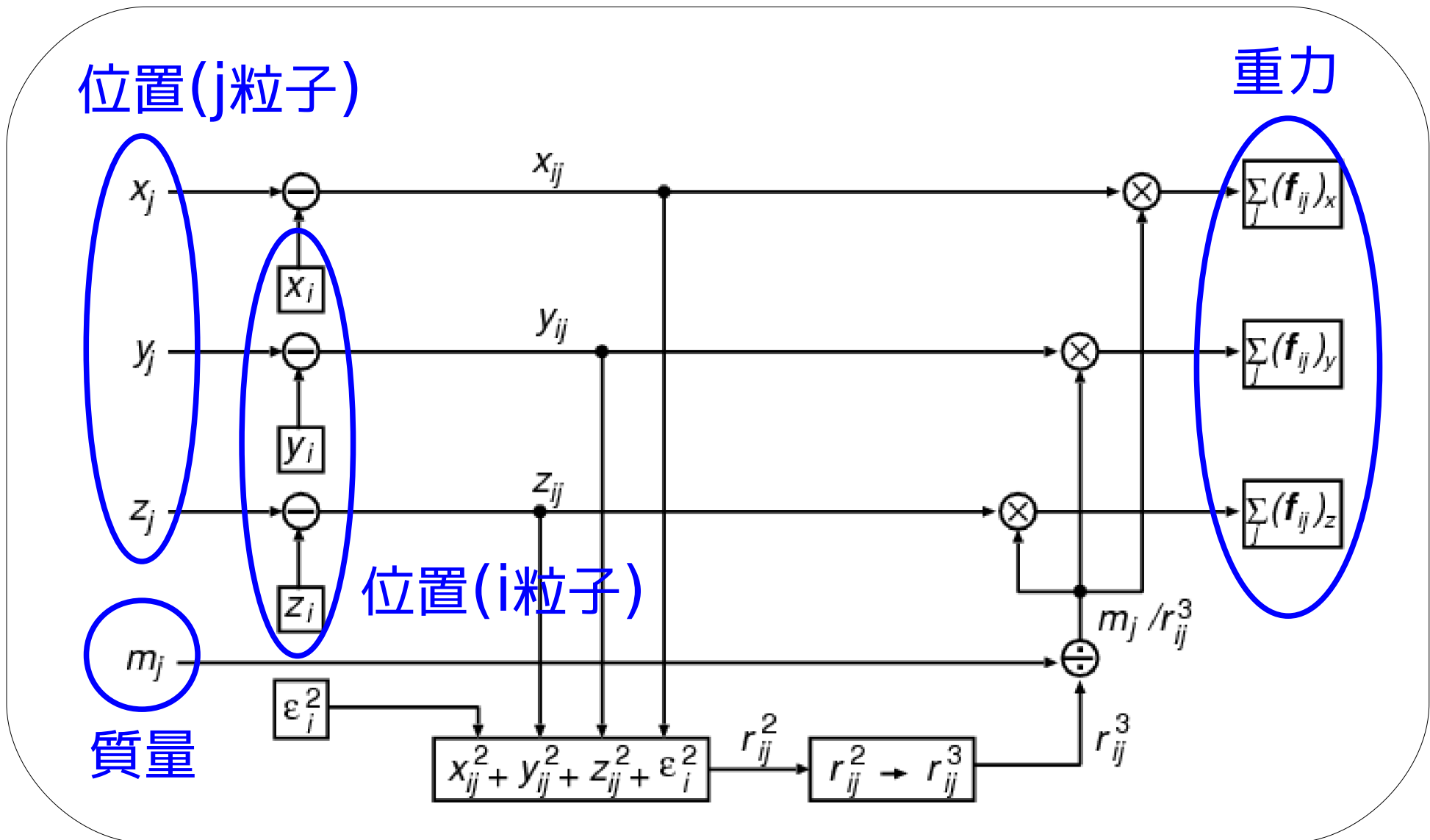


基本構成(GRAPE)



重力計算パイプライン

プログラムではなく、論理回路で演算を記述する。

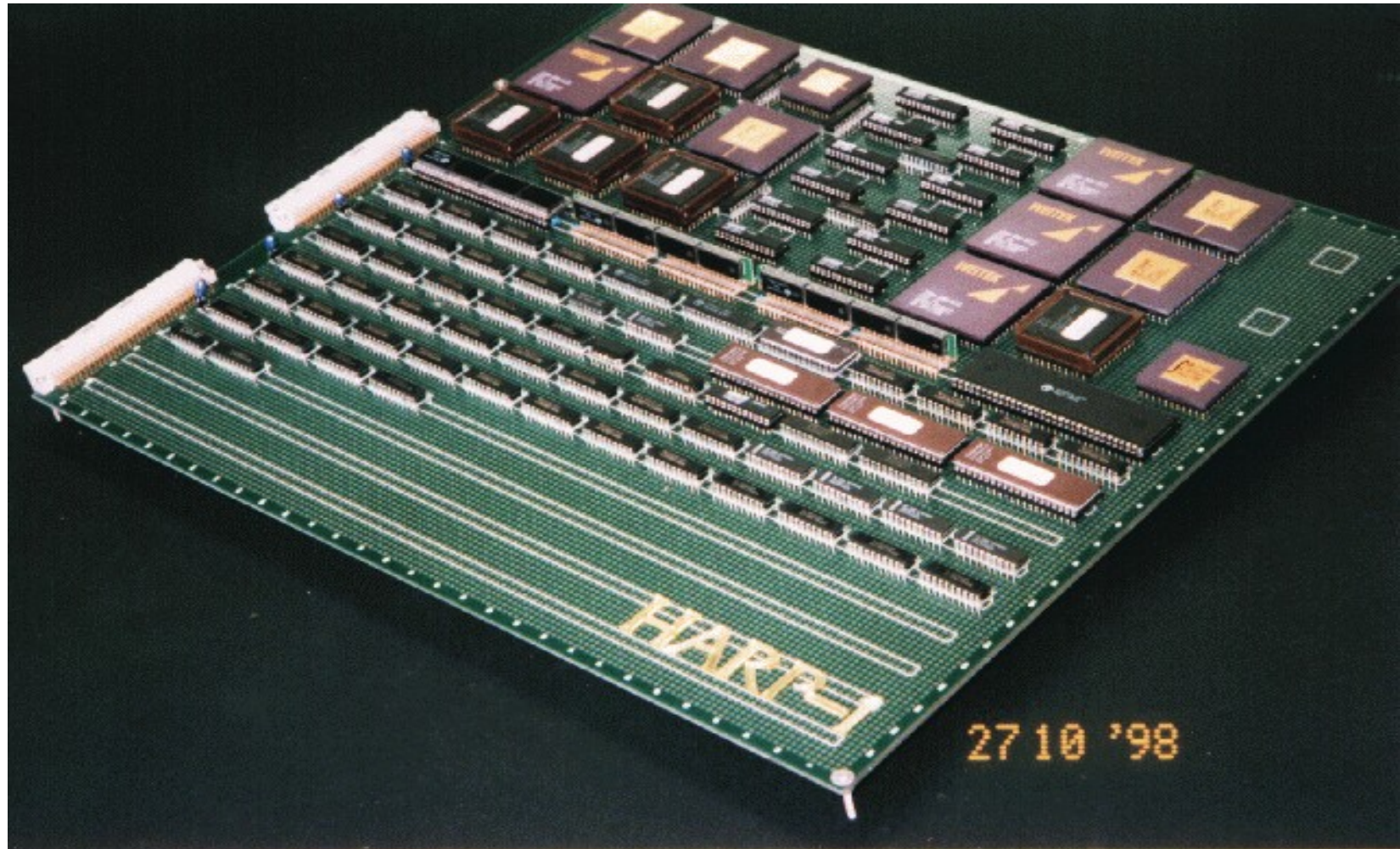


シミュレーションの手順(GRAPE)

1. 重力を及ぼす側の全粒子(j 粒子)の位置と質量を粒子メモリに送る。
2. 全粒子への重力が求まるまで以下を行う。
 - 1) 重力を受ける側の粒子(i 粒子)の位置を各パイプライン内のメモリ(レジスタ)に送る。
 - 2) すべての j 粒子との重力を各パイプラインで計算し積算する。
 - 3) 求めた i 粒子への重力を各パイプライン内のメモリ(レジスタ)から送り返す。
3. 求めた重力を用いて運動方程式を積分し、次の位置を求める。1. に戻る。

初期のGRAPEボード

HARP-1(1993)



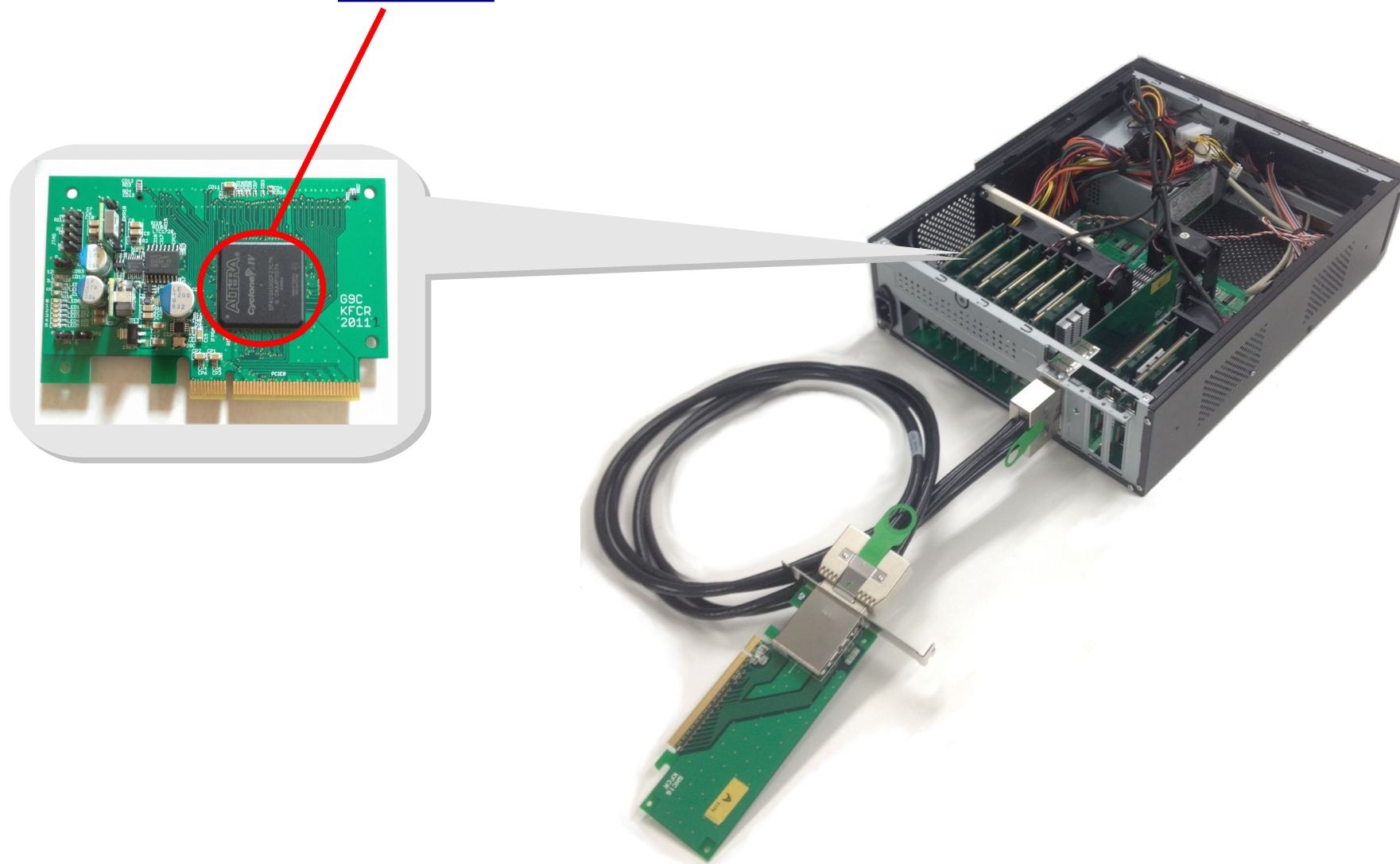
初期のGRAPEボード

GRAPE-5, GRAPE-6 (2005頃? at MUV)



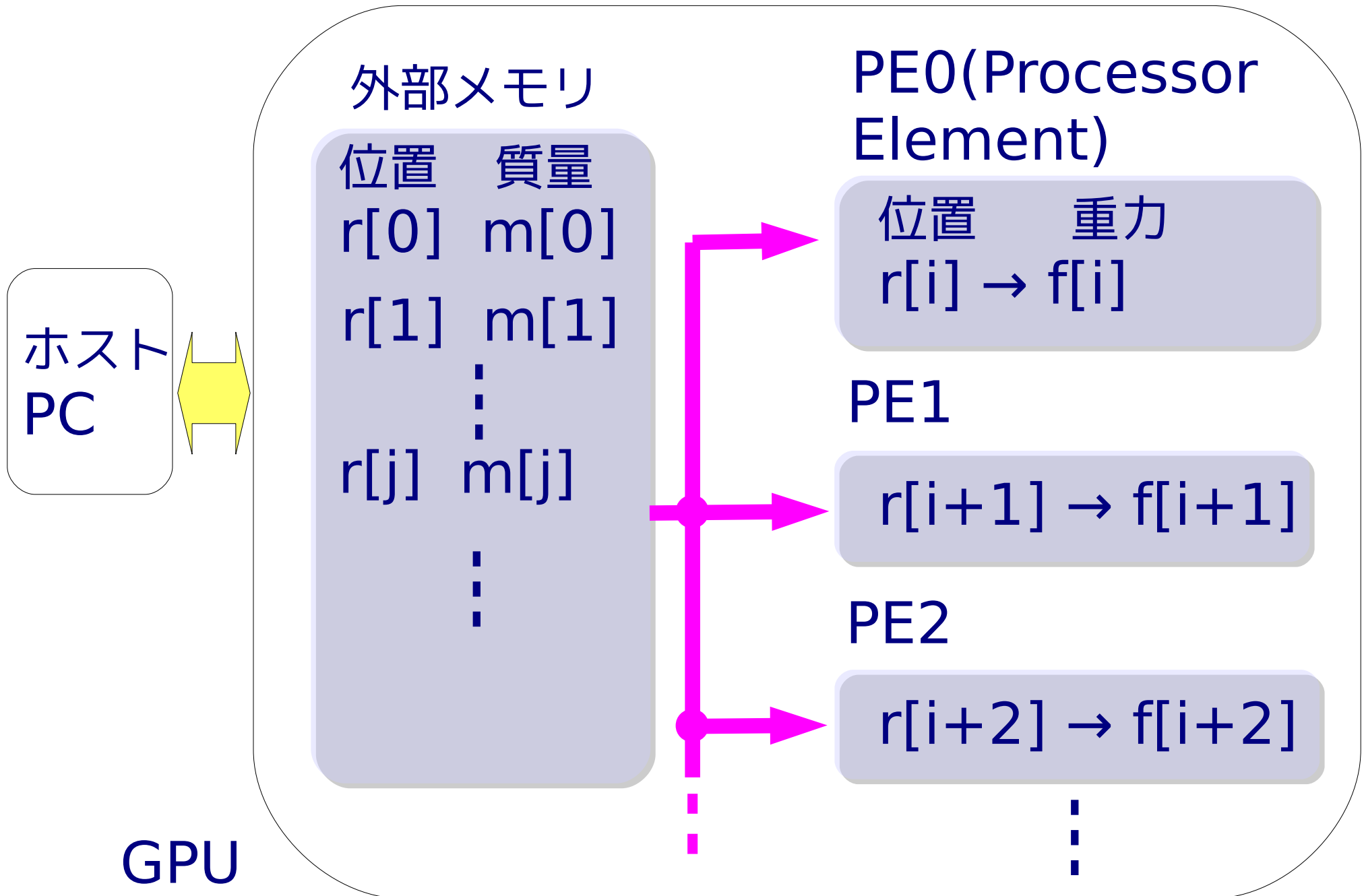
GRAPE-9 model 800/5000

重カパイプライン (GRAPE-5, GRAPE-6 互換)
を集積したFPGAを8-16個搭載。



イントロダクション
動作原理(専用アクセラレータ)
動作原理(汎用アクセラレータ)
アクセラレータの特色
基礎的な使用法(実際)
まとめ

基本構成(GPU)



シミュレーションの手順(GPU)

1. 重力を及ぼす側の全粒子(j 粒子)の位置と質量を外部メモリに送る。
2. 全粒子への重力が求まるまで以下を行う。
 - 1) 重力を受ける側の粒子(i 粒子)の位置を各PE内のメモリ(レジスタ)に送る。
 - 2) すべての j 粒子との重力を各PEでプログラムに基づいて計算し積算する。
 - 3) 求めた i 粒子への重力を各PE内のメモリ(レジスタ)から送り返す。
3. 求めた重力を用いて運動方程式を積分し、次の位置を求める。1. に戻る。

イントロダクション

動作原理(専用アクセラレータ)

動作原理(汎用アクセラレータ)

アクセラレータの特色

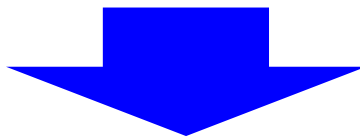
基礎的な使用法(実際)

まとめ

汎用機よりも速い理由

例) 重力計算(64k) CPU 数Gflops 程度
GRAPE-9 3Tflops程度
GPU(P100) 4Tflops程度

- 計算対象があらかじめ決まっている。
- 計算対象に高い並列性がある。



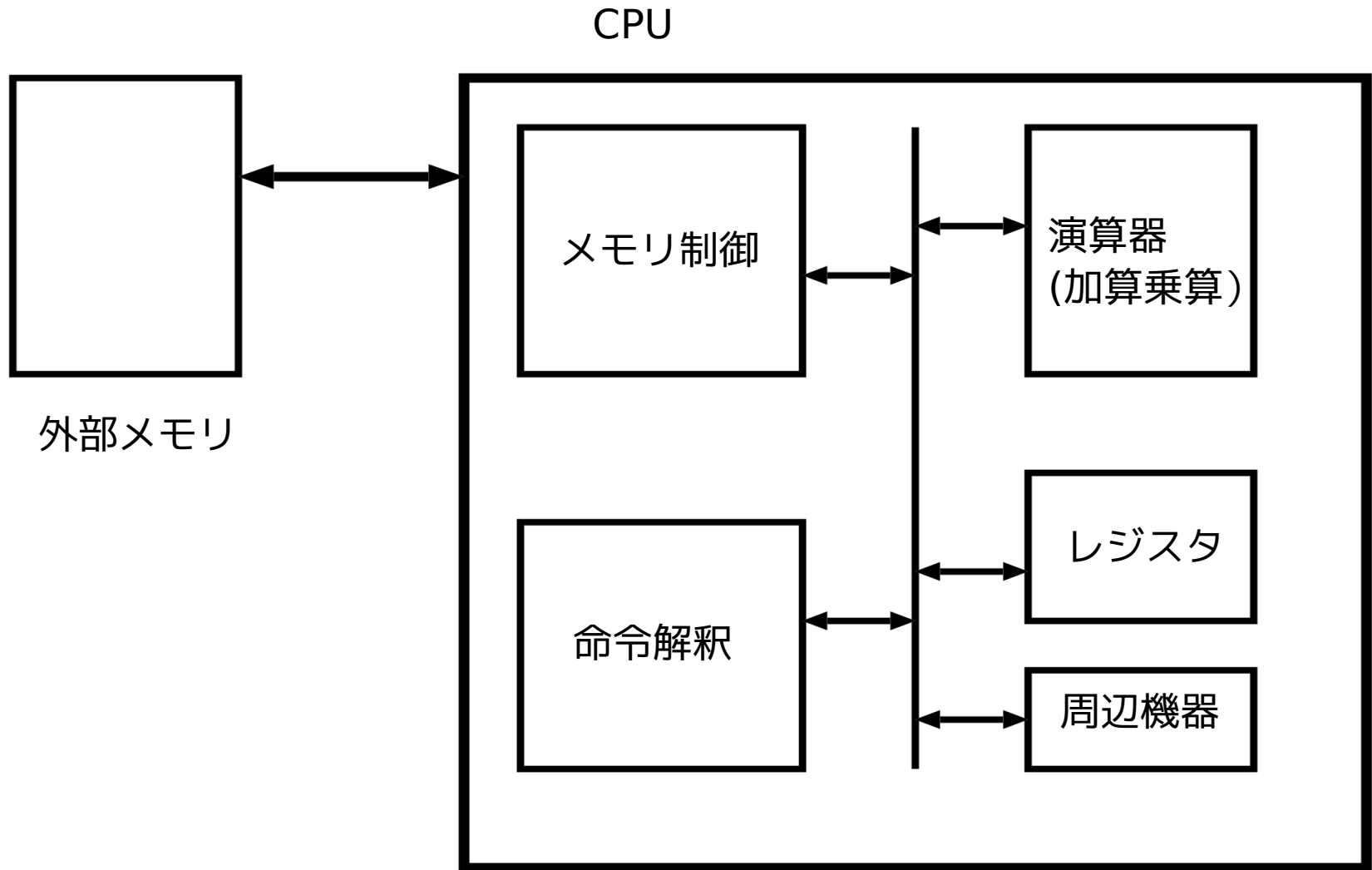
命令解釈部・メモリ制御部が必要ない(小さい)

その分を演算回路に割くことができる。

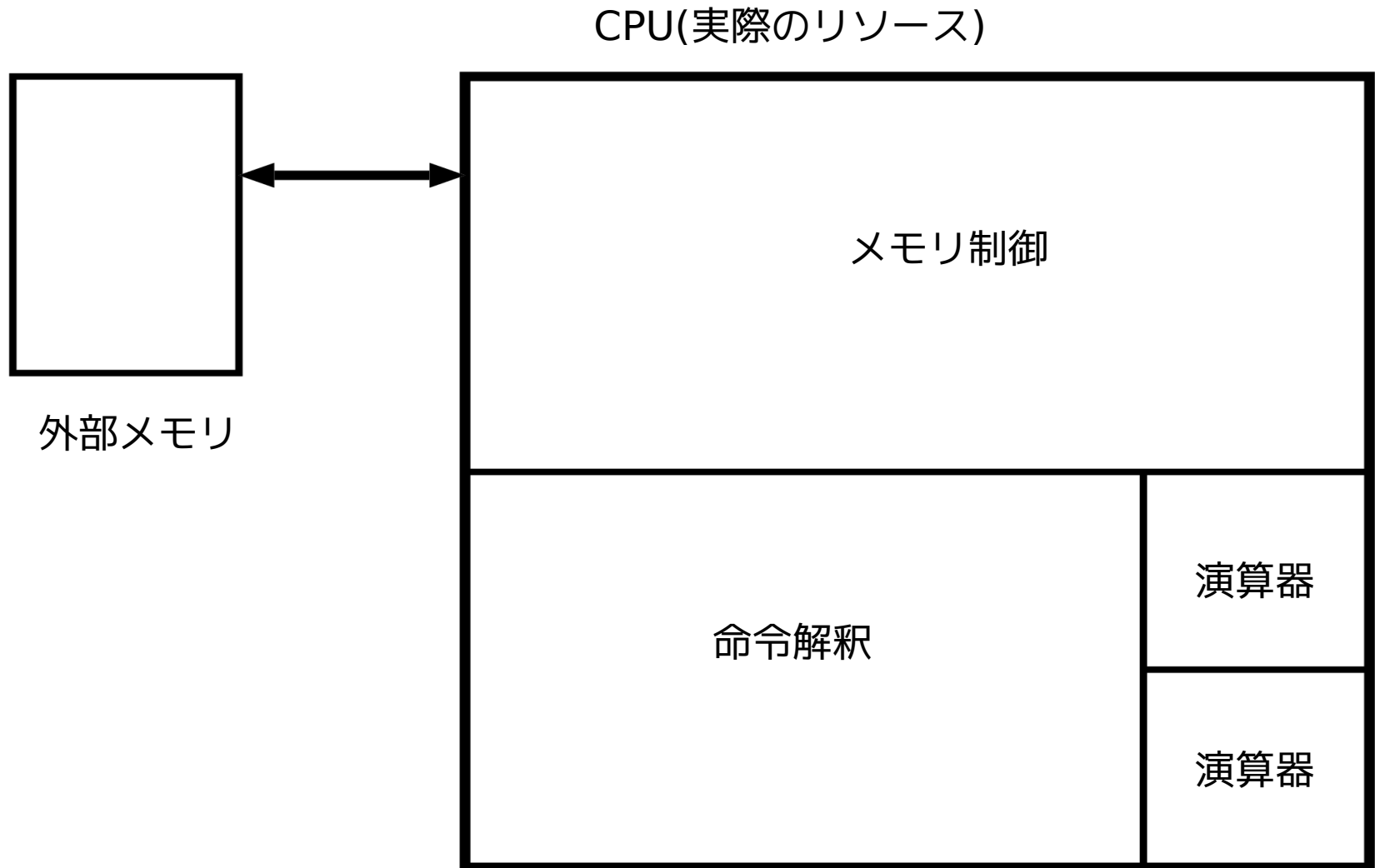
数値フォーマットの最適化

例) 17-bit 対数フォーマット

汎用機(CPU)の構造

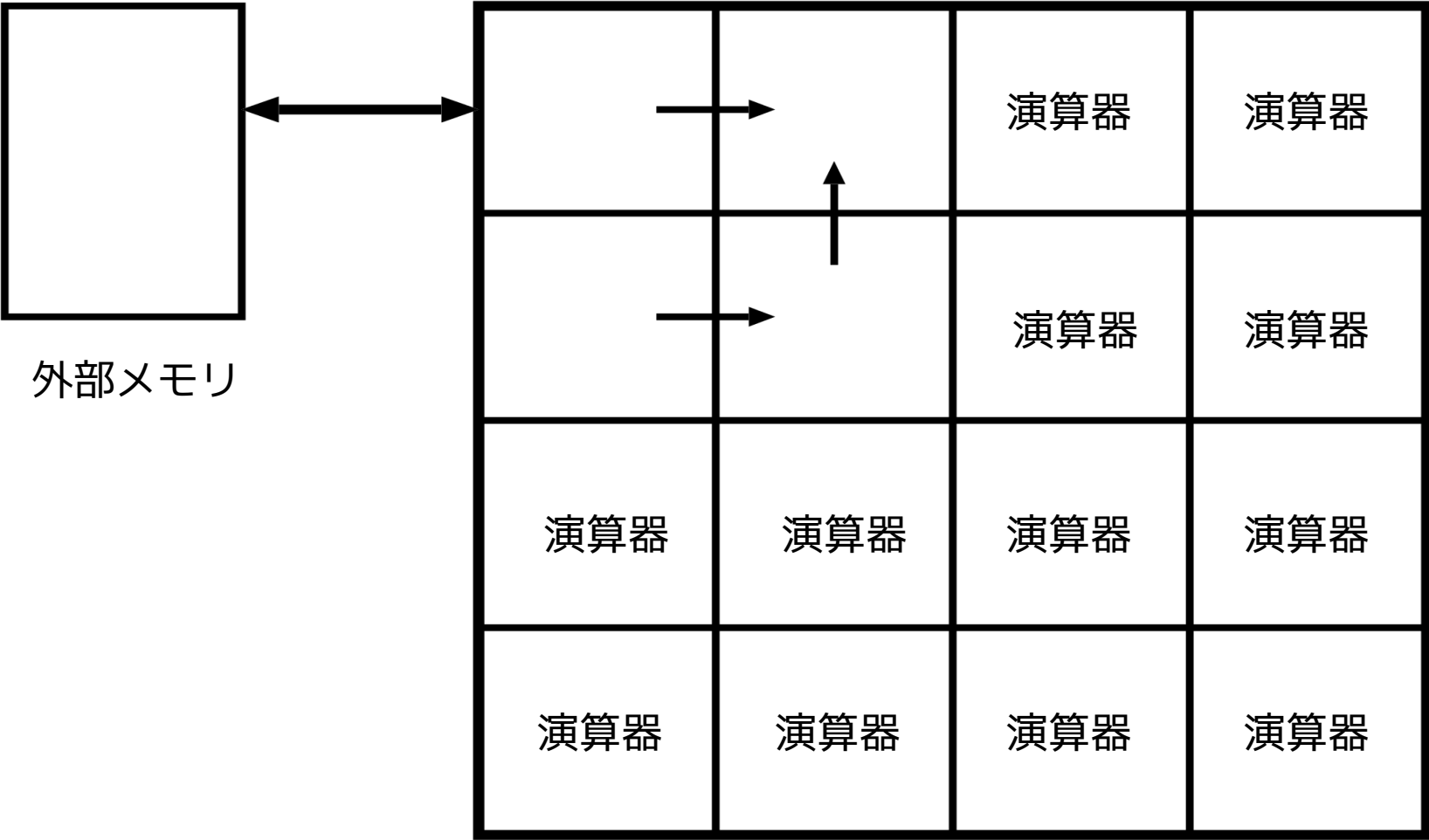


汎用機(CPU)の構造 (つづき)



専用計算機

GRAPE(リソース)



外部メモリ

演算器

演算器

演算器

演算器

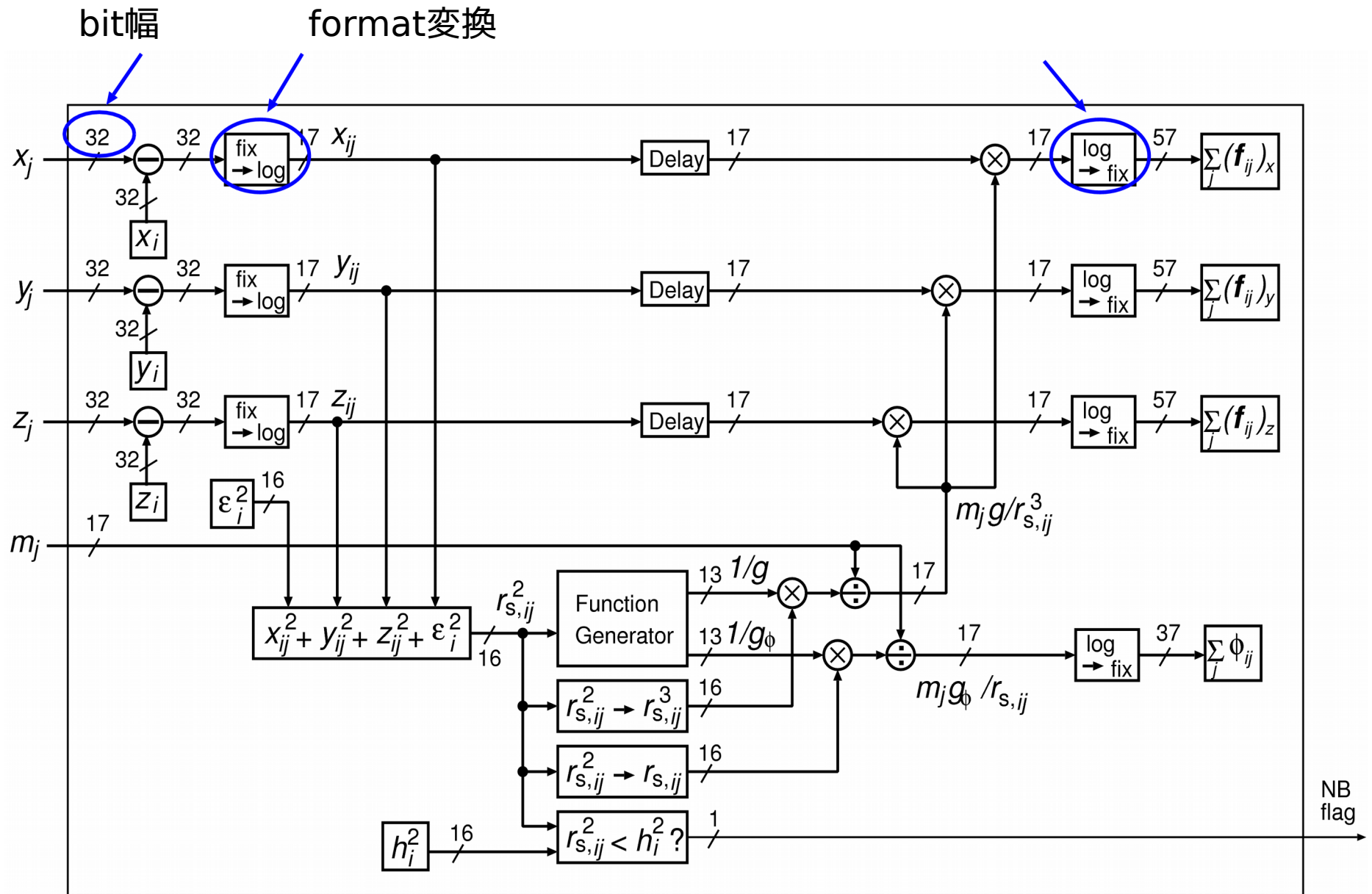
演算器

演算器

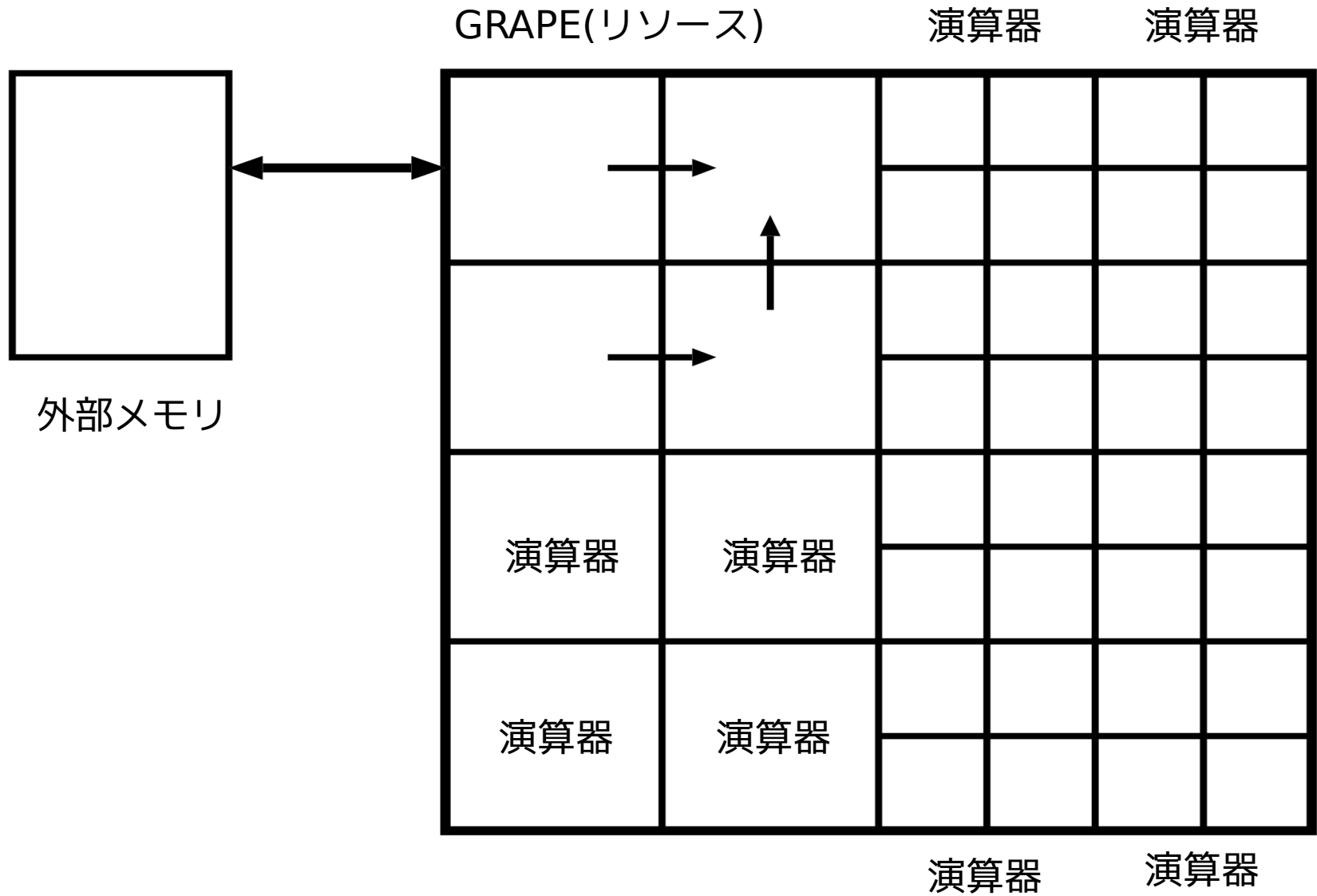
演算器

演算器

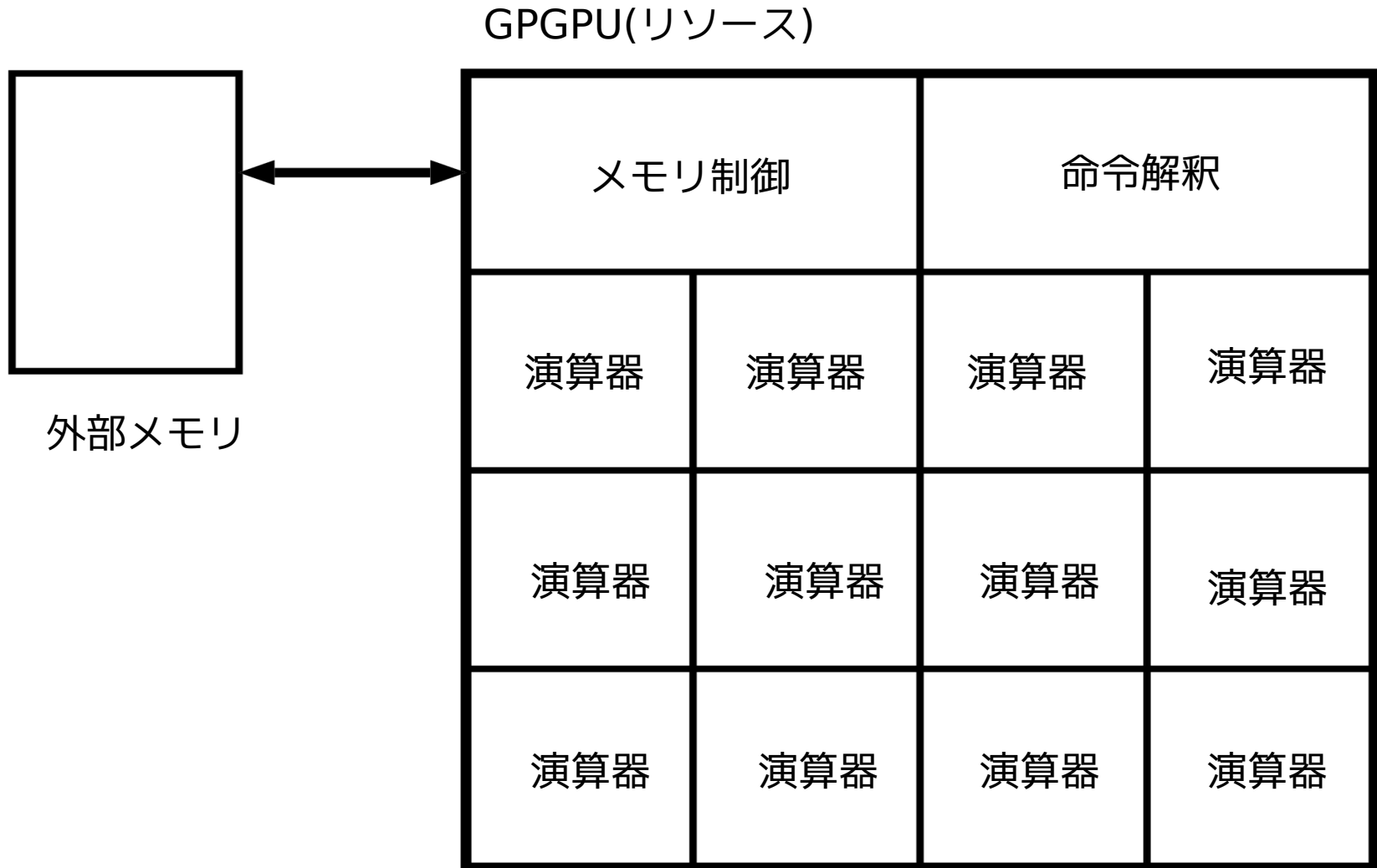
GRAPE-7のパイプライン



専用計算機(つづき)



汎用アクセラレータ (GPGPU)

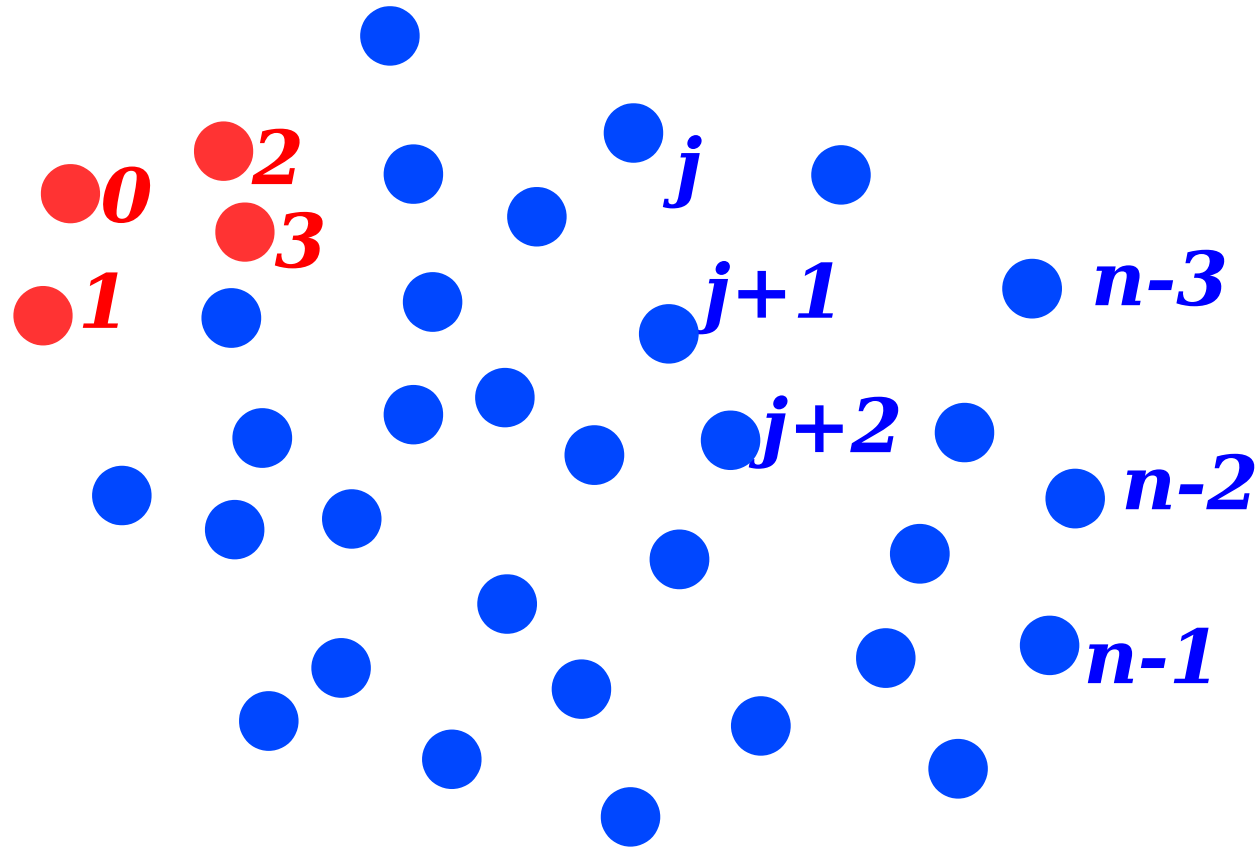


イントロダクション
動作原理(専用アクセラレータ)
動作原理(汎用アクセラレータ)
アクセラレータの特色
基礎的な使用法(実際)
まとめ

基礎的な計算の手順

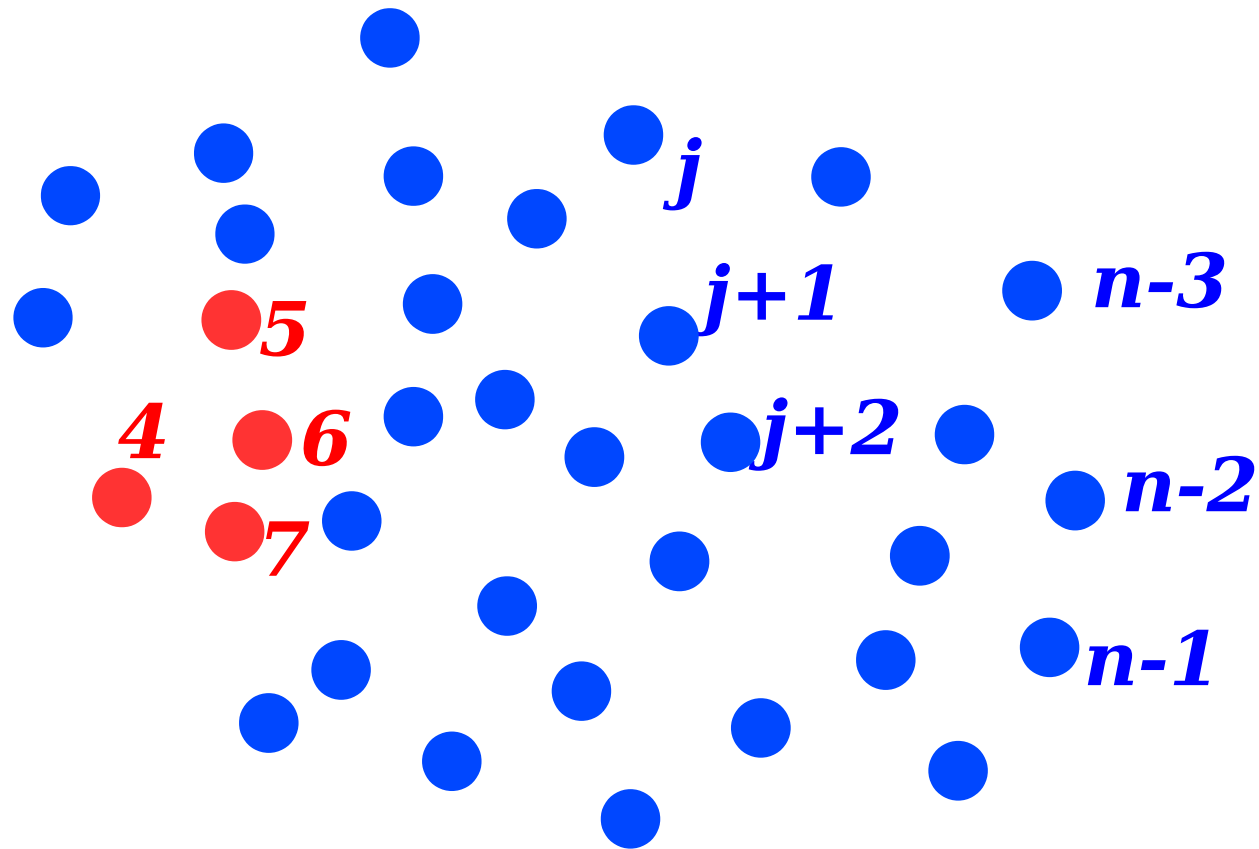
1. 重力を及ぼす側の全粒子(j 粒子)の位置と質量を粒子メモリに書き込む。
2. 全粒子への重力が求まるまで以下を行う。
 - 1) 重力を受ける側の粒子(i 粒子)の位置を各パイプライン内のレジスタに書き込む。
 - 2) 計算開始信号を送る。
 - 3) 計算終了を待つ。
 - 4) 求めた i 粒子への重力を各パイプライン内のレジスタから読み出す。

基礎的な計算の手順(i並列)



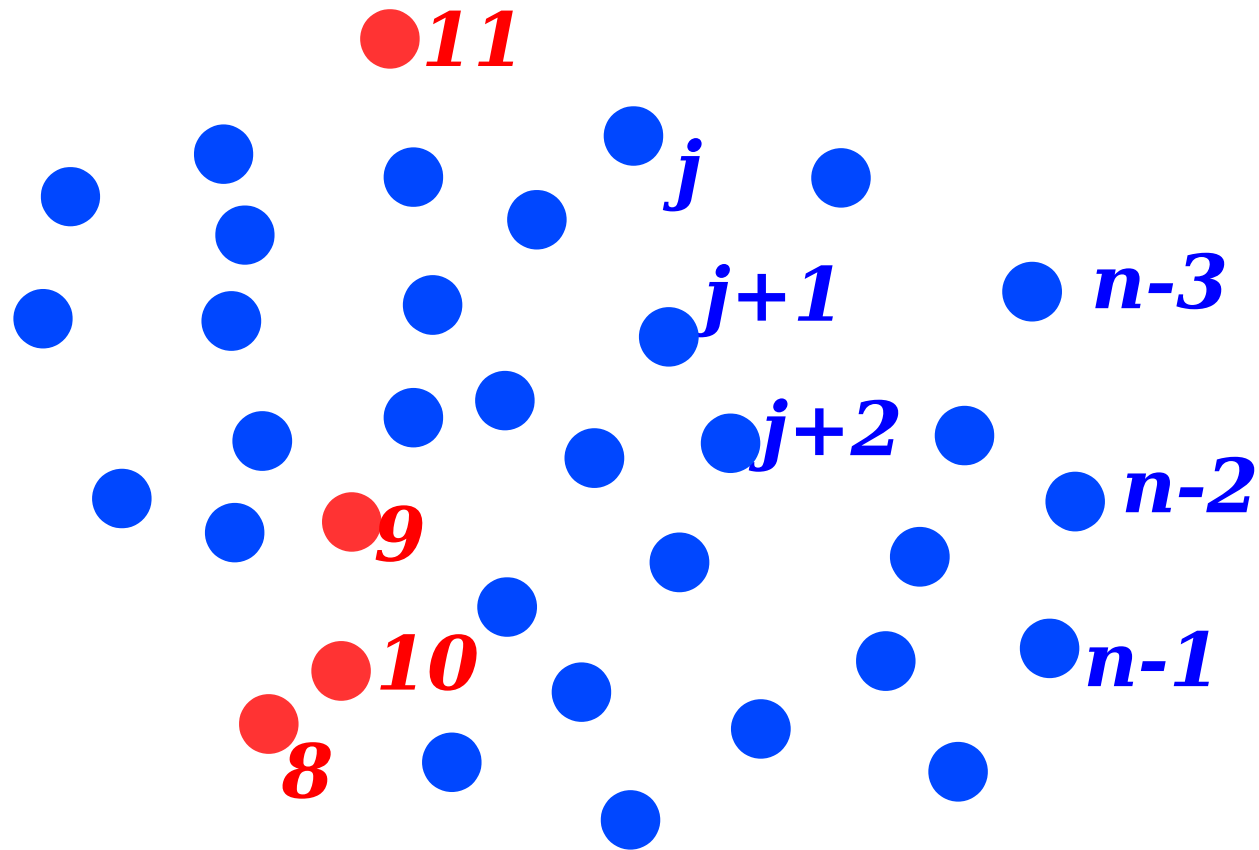
```
for (i=0; i<n; i+=4) {  
    calc_force_on(i..i+3);  
}
```

基礎的な計算の手順(i並列)



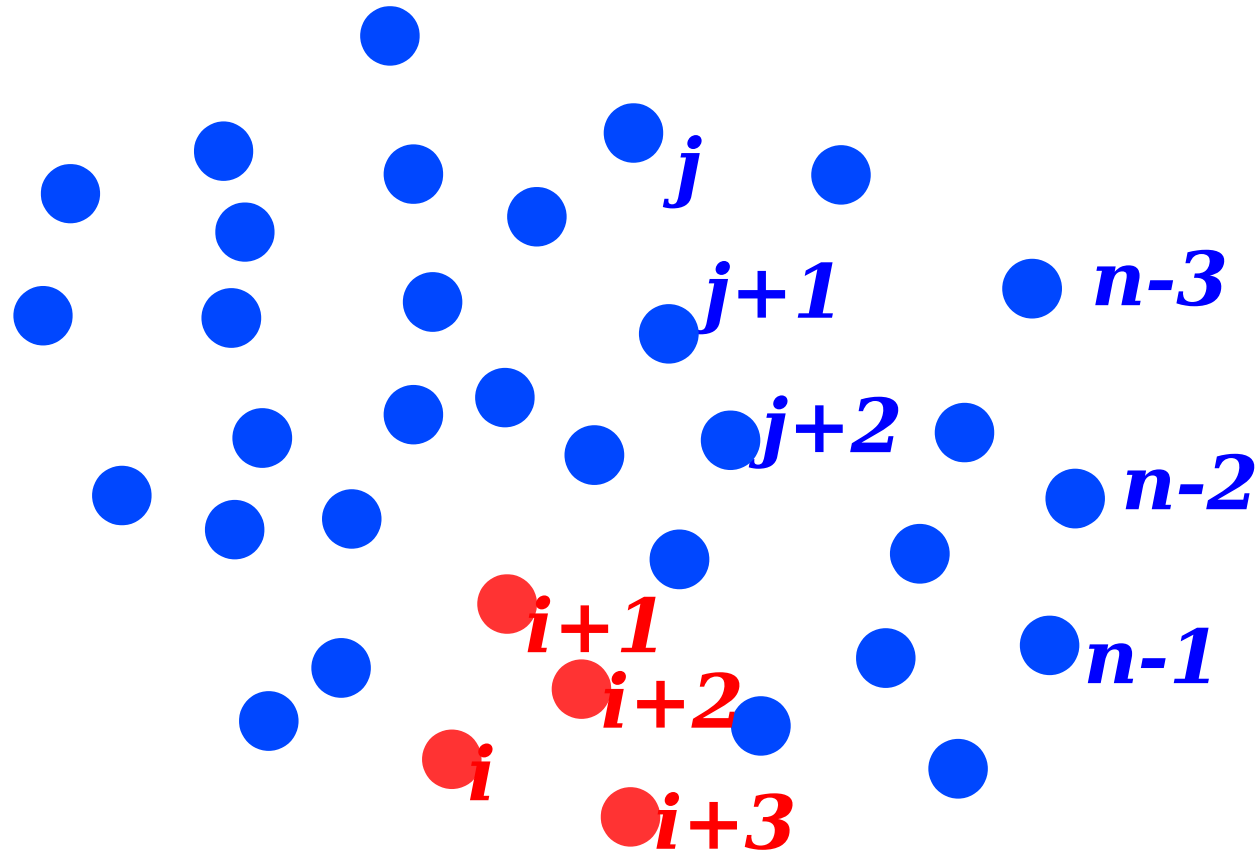
```
for (i=0; i<n; i+=4) {  
    calc_force_on(i..i+3);  
}
```

基礎的な計算の手順(i並列)



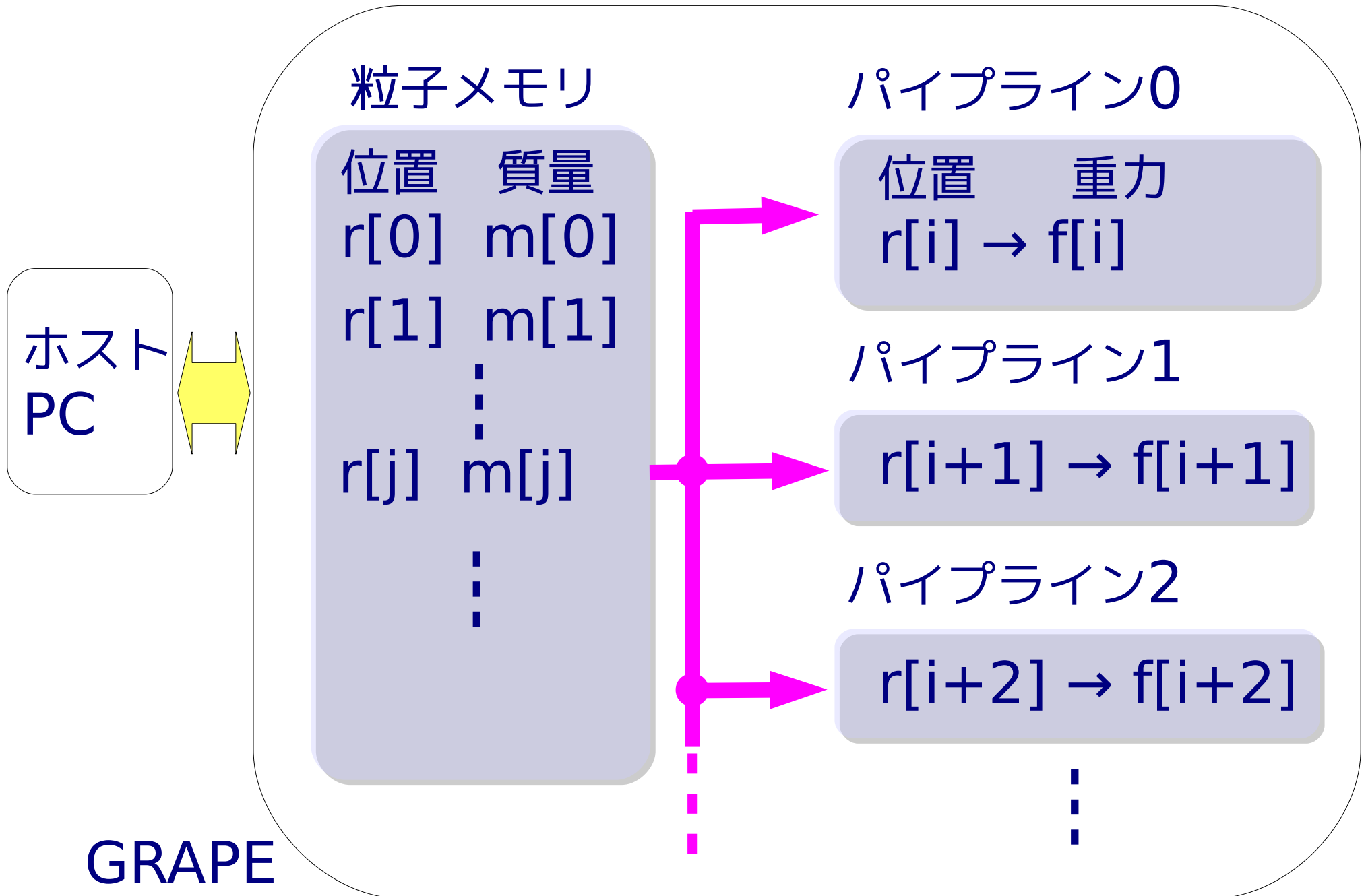
```
for (i=0; i<n; i+=4) {  
    calc_force_on(i..i+3);  
}
```

基礎的な計算の手順(i並列)



```
for (i=0; i<n; i+=4) {  
    calc_force_on(i..i+3);  
}
```

基本構成(GRAPE)



コーディングの例 (直接計算法)

```
npipes = g5_set_number_of_pipelines();
for (step = 0; step < final_step; step++) {
    g5_open();
    g5_set_range(xmin, xmax, mmin);
    g5_set_xj(0, n, x);
    g5_set_mj(0, n, mj);
    g5_set_eps_to_all(eps);
    g5_set_n(n);
    for (i = 0; i < n; i += npipes) {
        if (i + npipes > n) npipes = n - i;
        g5_calculate_force_on_x(xj + i,
            acc + i, pot + i, npipes);
    }
    g5_close();
    // orbital integration
    .....
}
```

位置、質量のスケーリング

```
g5_set_range(xmin, xmax, mmin)
```

位置座標の上限、下限

質量の分解能

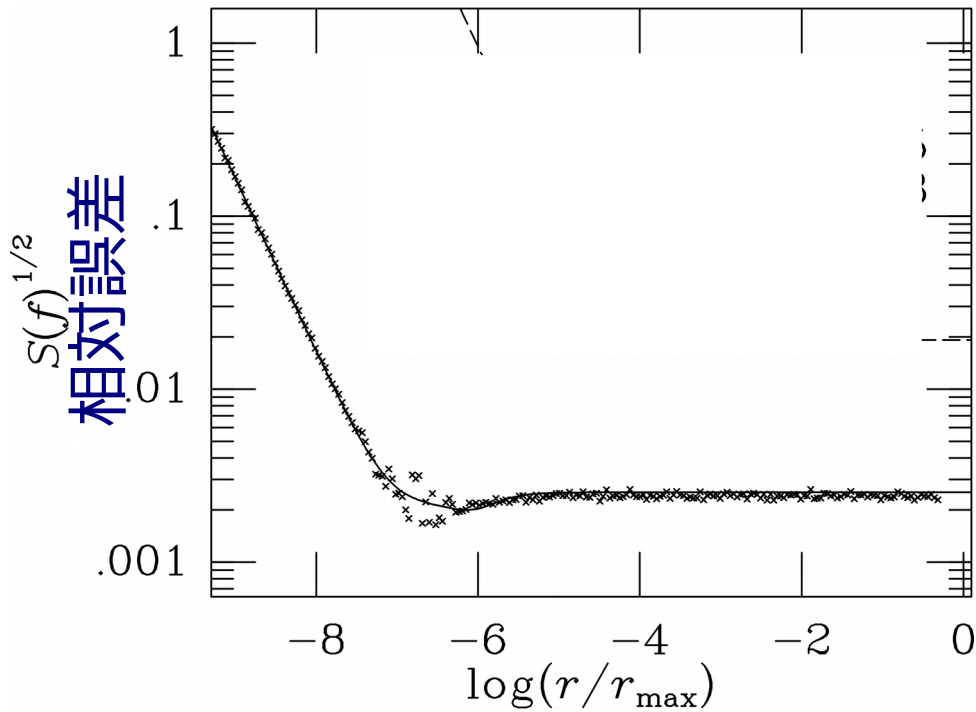
座標の分解能は

$$\text{約 } 10^{-10} \times (\text{xmax} - \text{xmin})$$

ソフトニング値の下限は

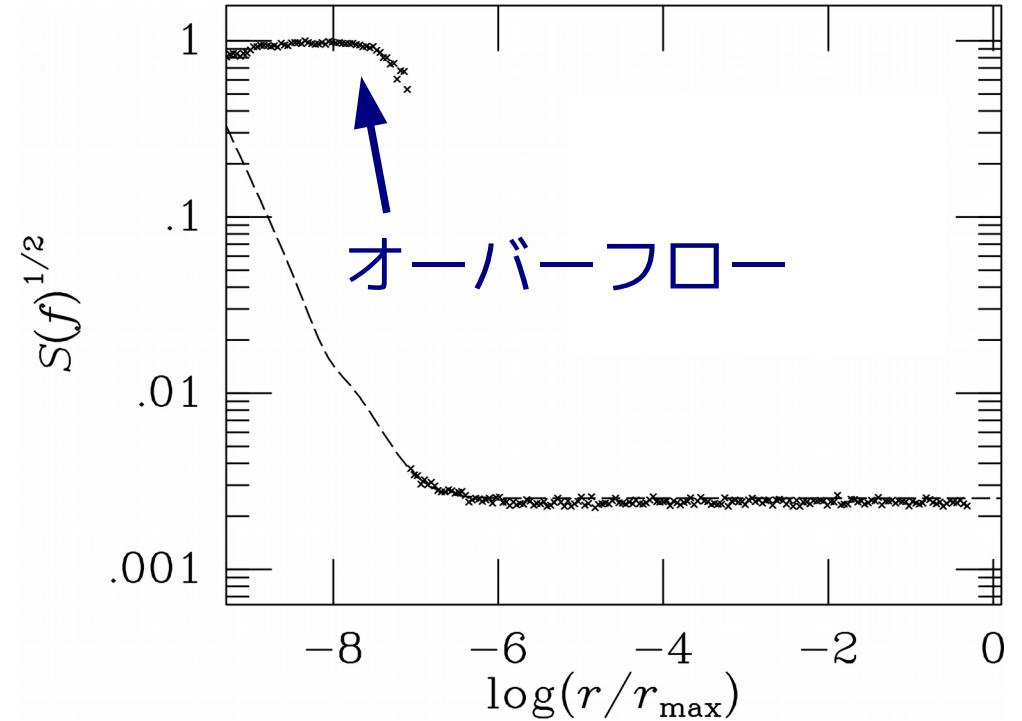
$$\epsilon \gtrsim 10^{-7} \times \left(\frac{m}{mmin} \right)^{1/2} (\text{xmax} - \text{xmin})$$

位置、質量のスケーリング (続き)



2粒子間の距離

$$m / m_{\min} = 1$$
$$\epsilon / r_{\max} = 10^{-6}$$



$$m / m_{\min} = 1$$
$$\epsilon / r_{\max} = 10^{-8}$$

小さすぎるソフトニング値

イントロダクション
動作原理(専用アクセラレータ)
動作原理(汎用アクセラレータ)
アクセラレータの特色
基礎的な使用法(実際)
まとめ

まとめ

- + GRAPE(専用アクセラレータ)の動作原理
 - プログラムではなくハードウェアで演算を記述する。
 - 対象を限定することで高速化。
- + GRAPE(専用アクセラレータ)の使用法
 - ホスト計算機上の API を介して使用する。
- + 最近の計算機(アクセラレータ)
 - 用途によっては GRAPE 的な手法が採用されている。