# GRAPE の将来計画

牧野淳一郎
東大天文

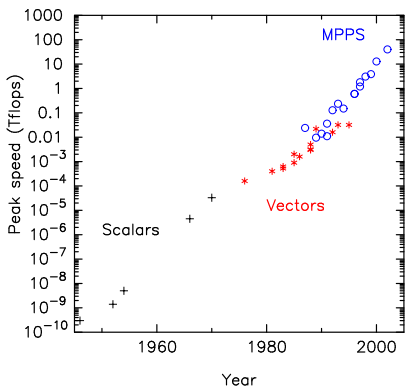
## 講演構成

- 科学技術計算用 (汎用) 計算機の進化方向
- 専用計算機の進化方向
- 次世代 GRAPE の方向
  - GRAPE-7
  - GRAPE-8?

## 計算機の発展

速度の進化: 50 年で $10^{10}$ 倍

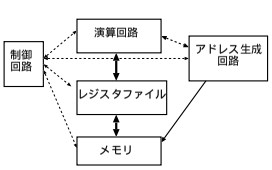
ほぼ時間の指数関数




## 指数関数的進化を可能にしたもの

1. ムーアの法則:トランジスタの大きさ: 3 年で半分
   - トランジスタの数: 4 倍
   - 速度: 2 倍
2. 計算アーキテクチャの革命
   スカラー計算機 → ベクトル → 並列

## スカラー計算機: 1946–1975




要するに普通に「計算機」だと思うもの。
「プログラム」があって、それを 1 命令づつ実行する。
「フォン・ノイマン」アーキテクチャ

命令は基本的なもの:
load: メモリからレジスタへ
store: レジスタからメモリへ
演算: レジスタからデータを演算器に、結果をまたレジスタに
分岐: 次に実行する命令のアドレスを (条件により) 変更

## 実行速度の制限要因: memory wall

演算命令とかの実行は速くできる
速くするのが難しい: メモリからの読み込み (load)

- DRAM のサイクルタイムはあまり速くない
- CPU とメモリの間のデータ幅 (信号線の数) はあまり多くできない

例:
Intel P4 データ 64 bit, クロック 800 MHz (DRAM チップ内部はもっと遅い)
AMD K8 データ 128 bit, クロック 400 MHz
CPU の演算速度に比べると 1/5 (必要なバンド幅に比べると 1/30 程度)

## メモリを速くする

遅いものは速くできないので、沢山並べて

- 順番にアクセス (マルチバンク)
- データ幅を広くする

「ベクトル計算機」の考え方:沢山並べて、並列にアクセスする:
ずーっと先まで次に何をするかがわかっていれば簡単
キャッシュでは具合の悪い

```
do i=1,10000000
      a(i) = b(i) + c(i)
enddo
```

みたいなのが理想的

## ベクトル計算機の高速化の限界

演算パイプライン、プロセッサを増やす:
どちらの方向も「共有メモリ」の構成をとった。
メモリのどこにあるデータも。どの CPU (あるいはパイプライン) からでも同じ速さで読み書きできる。
言い換えると: メモリと演算器がクロスバースイッチみたいなものでつながる。

- 長い配線が避けられない
- スイッチのハードウェア規模、遅延が大きくなる

性能をあげるのが難しくなる
共有メモリ: 16-32 CPU くらいが物理的な限界?
1980 年代後半から 90 年代になって、ベクトル計算機の性能向上は遅くなった。

## 並列計算機

並列計算機といっても、共有メモリならベクトル計算機と同じ問題が発生。

共有メモリにしない → 問題が (とりあえず) 回避される。
つまり: 1 CPU + メモリの簡単な計算機を構成要素にして、それを沢山並べる。
その間は、なんか適当なネットワークとかでつながる
という方向だと、ややこしい問題を (計算機を作る人は) 考えなくてもいい。
=分散メモリ型並列計算機

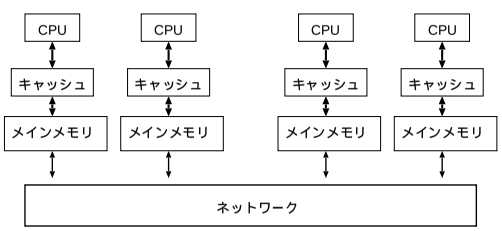
## 分散メモリ型並列計算機

(数値計算業界からみて) 開発の重要な源流は

- QCD 計算のための計算機 (Caltech Hypercube)
- 星野-川合らによる PACS/PAX
- イリノイ大学の ILLIAC IV

どれも、割合単純な連続系 (偏微分方程式系) を解くことだけを目標に設計・製作された。

## 分散メモリ型計算機の構成




どれもおなじようなもの。ILLIAC IV は特殊: SIMD 型 (全部の CPU が同じ演算を実行)。
あとの流れは MIMD (みんなバラバラに計算する)

## 分散メモリ型計算機の (偏った) 歴史

1970 年代: 汎用マイクロプロセッサを使った計算ノード
1980 年代: 多様

- 汎用マイクロプロセッサ/専用プロセッサ
- SIMD/MIMD
- 分散メモリ/分散共有メモリ/AllCache/...

## 分散メモリ型計算機の歴史 (2)

1990 年代: 淘汰

- ベクトル型分散メモリ— 富士通、NEC
- Cray T3x — Alpha
- Beowulf — Intel x86

2000 年代: ???

- Beowulf、SMP クラスタ以外は死滅?

## 死滅した理由

- 価格性能比
- プログラミングモデル

## 価格性能比

消費者マーケットに出るもの: 生産数が何桁も違う。
それだけ、開発費を投入して量産コストを下げることができる。
2002 年時点で:
生産数 $< 10^4$ 個のカスタム LSI: 開発費 数億、量産コストチップあたり 5-10 万、動作クロック 300-500 MHz
Intel Pentium 4: 開発費 ??? 売値: 安いものなら 1 万、動作クロック: 2GHz 以上
同じようなものを作ると、価格性能比で 10 倍は損
良いから高いわけではなくて、生産量が少ないから高い

## プログラミングモデル

多様なアーキテクチャ → プログラミングモデルへの要求:

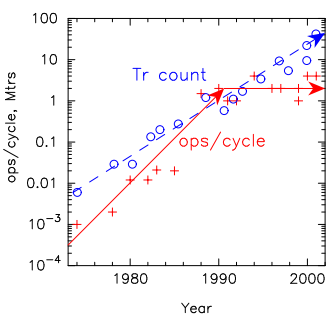
- 移植性
- 性能の高さ

生き残ったモデル: message-passing
要するに: 各ノードでは普通にプログラムが走る。必要に応じて、通信ライブラリを使って通信する。
自動並列化とか並列処理言語とかそういったものには頼らない。
もっとも安直なハードウェアに有利な状況
将来は?

## 進化の限界

- 「汎用性」の低下
  アーキテクチャの複雑化
  - メモリ階層
  - スカラー → ベクトル → ベクトル並列 (分散メモリ)
    「あらゆる応用プログラムで万能に速い」計算機はなくなった
- 半導体利用効率の低下
  - チップあたりトランジスタ数は 10 年で 100 倍
  - チップあたりの演算器数は過去 15 年間で 2 倍程度

## 半導体利用効率の低下




マイクロプロセッサのトランジスタ数と 1 クロックあたりの浮動小数点演算数

演算数は 1 に達したところで進歩がほとんど止まっている

## 専用計算機の方向性

アプリケーションに専用化したプロセッサ:多数の演算器を使える

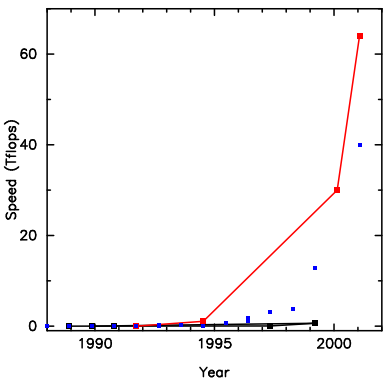
商用で成功しているもの

- PC 用グラフィックカード
- Sony PS2 などのゲーム機

科学技術計算用専用計算機

- GRAPE
  - 粒子間相互作用計算に特化した演算パイプライン
  - 400 個程度の演算器を 1 チップに集積 (GRAPE-6, MDM)

## GRAPE の進化




赤: GRAPE

青: 汎用ベクトル/並列

速度はほぼ同等。
開発費は 2 桁違う。

## 専用計算機の「限界」

開発費の高騰

| | | |
|---|---|---|
| 1990 | 1$\mu$m | 1500万円 |
| 1997 | 0.25$\mu$m | 1億円 |
| 2004 | 90nm | 3億円以上？ |

開発の方向

1. 頑張って大きな予算を取る
2. 小さな予算でできるように何か考える
3. 頑張らないでも大きな予算を取れる方法を考える

## 頑張って大きな予算を取る

## 頑張って大きな予算を取る

小久保君とか、
私より偉い誰かに任せたい

## 小さな予算でできるように何か考える

カスタムチップを起こすのにはお金がかかり過ぎる。
FPGA 「リコンフィギャラブルデバイス」は？

トランジスタ利用効率で大きく劣る
↓
データ語長が短い応用でないと高性能は難しい

というわけで、低精度 GRAPE (その他力学的時間スケールで済むような話) なら結構いける。

## GRAPE-7

- FPGA 利用
- 通信は PCI-X の利用で高速化 (台坂、川井、福重)
- 1-2 年先の技術だと、ボード製造原価 20 万円で 400Gflops くらいが可能 (低精度なら)
- SPH 計算も可能 (福重ら)
- Treecode ならホストも現在の MUV の 10 倍速くなれば全体で 10 倍速くなる
- 10 ノード並列で $10^8$ 粒子 1 ステップ 10 秒、100 ノードなら $10^9$ 粒子 10 秒

## 頑張らないでも大きな予算を取れる方法を考える

そんなうまい話があるか？

## 頑張らないでも大きな予算を取れる方法を考える

そんなうまい話があるか？

「技術的」アプローチ:
応用範囲を広げる (天文の中/外両方に)

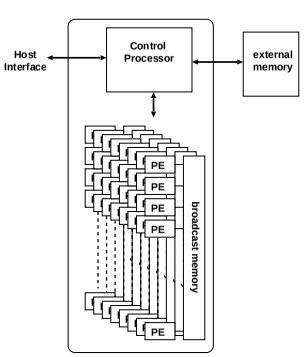
## GRAPE-8？

- 応用に特化し、多数の演算器を 1 チップに集積、並列動作させて高い性能を得た専用計算機の特徴を生かす
- しかし広い応用範囲を実現する

GRAPE の高性能: 多数の演算器を 1 チップに集積し、小さなメモリバンド幅で全演算器を有効に働かせたことによる

相互作用の計算式に専用化したパイプラインが必須というわけではない。

## 1 チップ 超並列プロセッサ




- 非常に多数のプロセッサエレメント (PE) を 1 チップに集積
- PE = 演算器 + レジスタファイル (メモリをもたない)
- チップ内に小規模な共有メモリ (PE にデータをブロードキャスト)
- 外部メモリへのインターフェースを持つ

## アプリケーション

「GRAPE として」使う

- 全ての PE が、自分の粒子への、同じ粒子からの力を計算
- 力を及ぼすほうの粒子データは外部メモリから供給

「多対多」の相互作用計算ならばなんでも扱うことができる

- 粒子法: 重力、分子間力、SPH、、、
- 境界要素法: ポアソン方程式、ヘルムホルツ問題、、、
- ルジャンドル展開による極座標流体計算
- 密行列ソルバ (内積型を利用)

## アプリケーション (続き)

多数の要素の各点で重い計算をするようなのにも使える (かも)
流体での cooling function
規則格子での流体計算 (?)

## 類似のアプローチとの比較 (1)

- SIMD 超並列計算機 (Goodyear MPP, CM-1/2)
  - 外部メモリへのバンド幅、通信ネットワークでプロセッサ数が制限
  - プロセッサがメモリを共有することでプロセッサ数の制限を回避。応用範囲には制限。
- FPGA
  - トランジスタ利用率が低い
  - ソフトウェア開発が困難
  - どちらも回避

## 類似のアプローチとの比較 (2)

再構成可能計算機

- DSP ブロック入り FPGA
  - ゲート効率 FPGA よりはちょっといい?
- DAP/DNA, DRP
  - ゲート効率悪い
  - 動作速度遅い

他のアプローチとの基本的な違い:

チップ内にスイッチングネットワークをもたない

- (多少) できないことも発生
- デザインの単純化 → 高集積、高速動作

## ソフトウェア

PE のプログラムは、基本的にシーケンシャル
(SIMD アプローチの利点)

- アセンブリ言語
- 単純なコンパイラ

上位のソフトウェア

- 複数ホスト+超並列ボード上の並列化
- ホスト計算機/超並列ボードのタスク割り当て

## 将来展望

- 多対多の相互作用向け SIMD 超並列計算機という新しい概念
  - 専用計算機並の高いコストパフォーマンス
  - 特定の物理系に縛られない広い適用範囲
  - 将来の半導体技術の発展を有効に利用可能
- ベクトル並列、スカラー並列と相補的
- compute-intensive な問題にブレークスルーをもたらす

## まとめ

GRAPE の将来計画
半導体技術的には先に進める
問題は初期費用
FPGA で初期費用を減らす — GRAPE-7
高い初期費用に見合うだけのある程度の応用範囲をもたせる — GRAPE-8

## COTS アプローチ

**Intel Pentium 4 等の “off the shelf” プロセッサ**

- 大量生産 → 低コスト
- 大量生産 → 莫大な開発費 → 高い性能

単一スカラープロセッサで COTS の性能を超えるのは非常に難しい
性能が memory limited — プロセッサを別のものにしても速くならない

**COTS の限界: backward compatibility.**
**VLIW すら難しい (Itanium の失敗)。**

## チップ内部構成

- 共有メモリユニットの数 (共有メモリユニットあたりの PE 数)
- PE の構成
  - 演算方式 (語長、データ形式、関数計算のサポート)
  - 制御方式 (ベクトル命令/スカラー命令)
  - レジスタ構成
- メモリインターフェース方式
- 汎用プロセッサとのインターフェース