エクサスケールシステムに向けて —— 分散メモリ超並列アーキテクチャの復活

牧野淳一郎

東工大 地球生命研究所
理研 計算科学研究機構 粒子系シミュレータチーム

第 5 回アクセラレーション技術発表討論会会津大学 2013/9/5-6

話の構成

- スパコンの進化:: 1950-2010
- 現状の問題
  - 消費電力
  - 並列処理オーバーヘッド
  - ソフトウェアの開発/メンテ
- 解決？
- 日本の「ポスト「京」」

スパコンの進化: 1940-2000

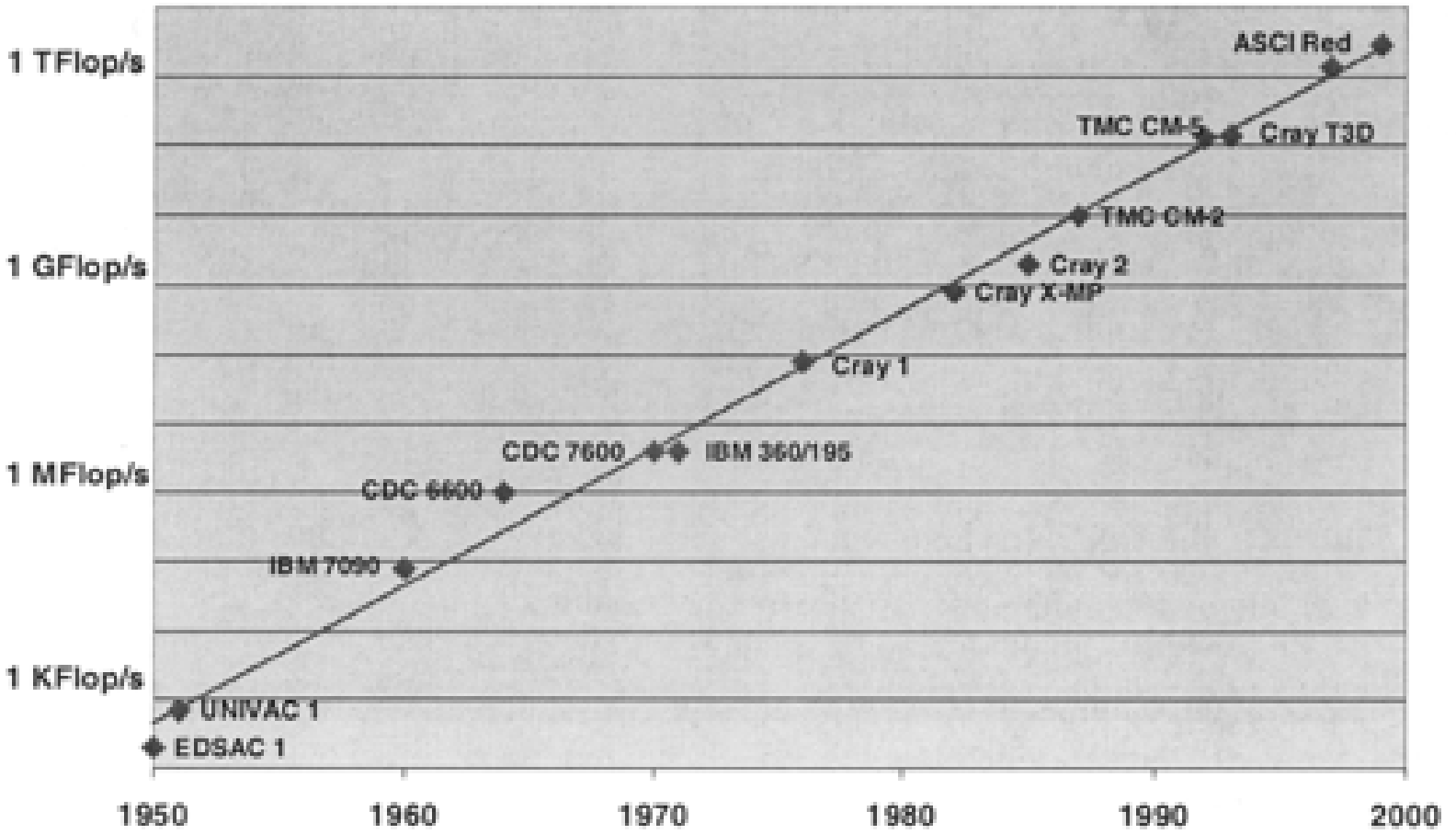
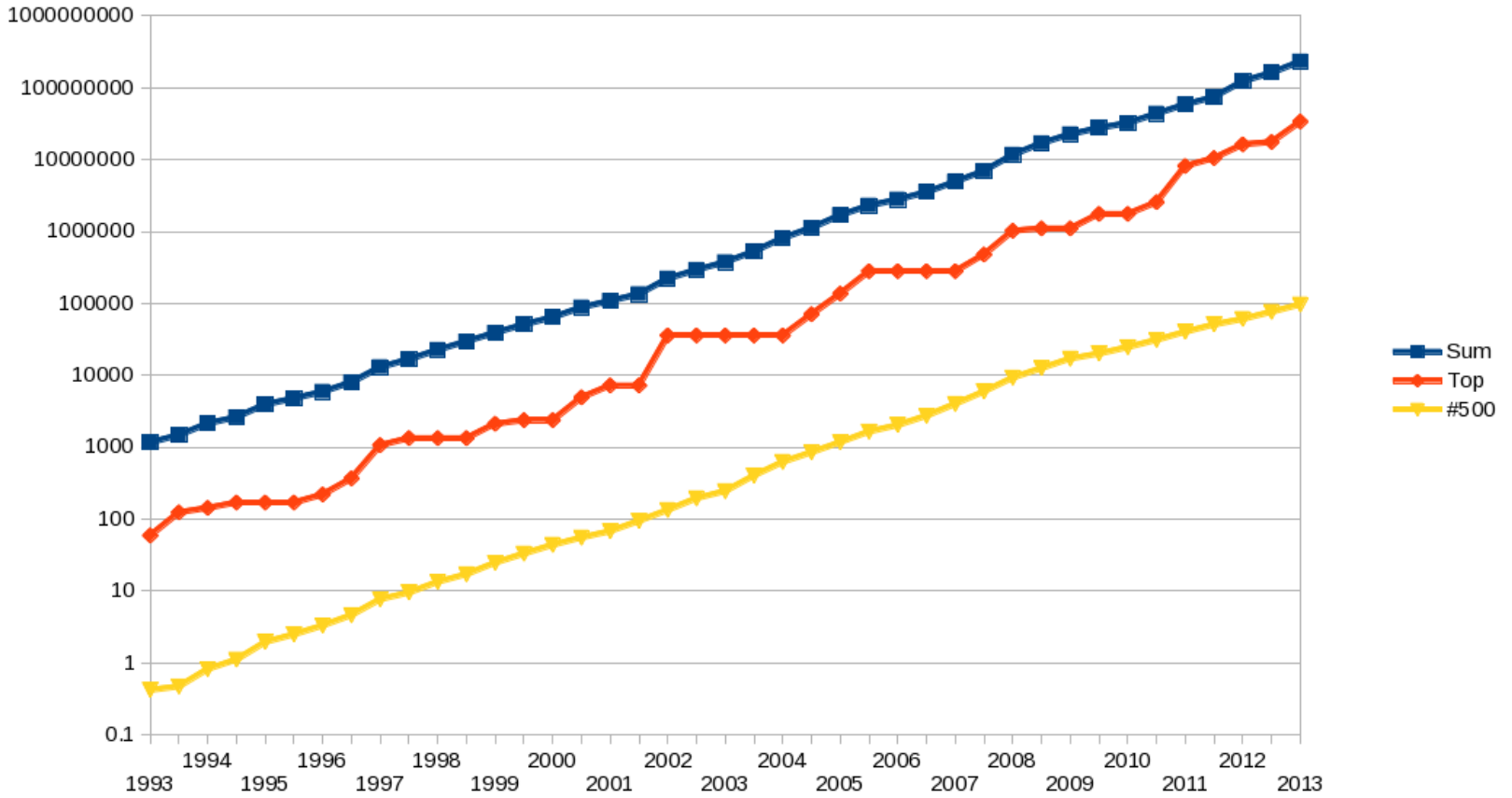


Figure 1. *Moore's law and peak performance of various "supercomputers" over time.*

1940-2000: 10 年に 100 倍

スパコンの進化: 1993-2013

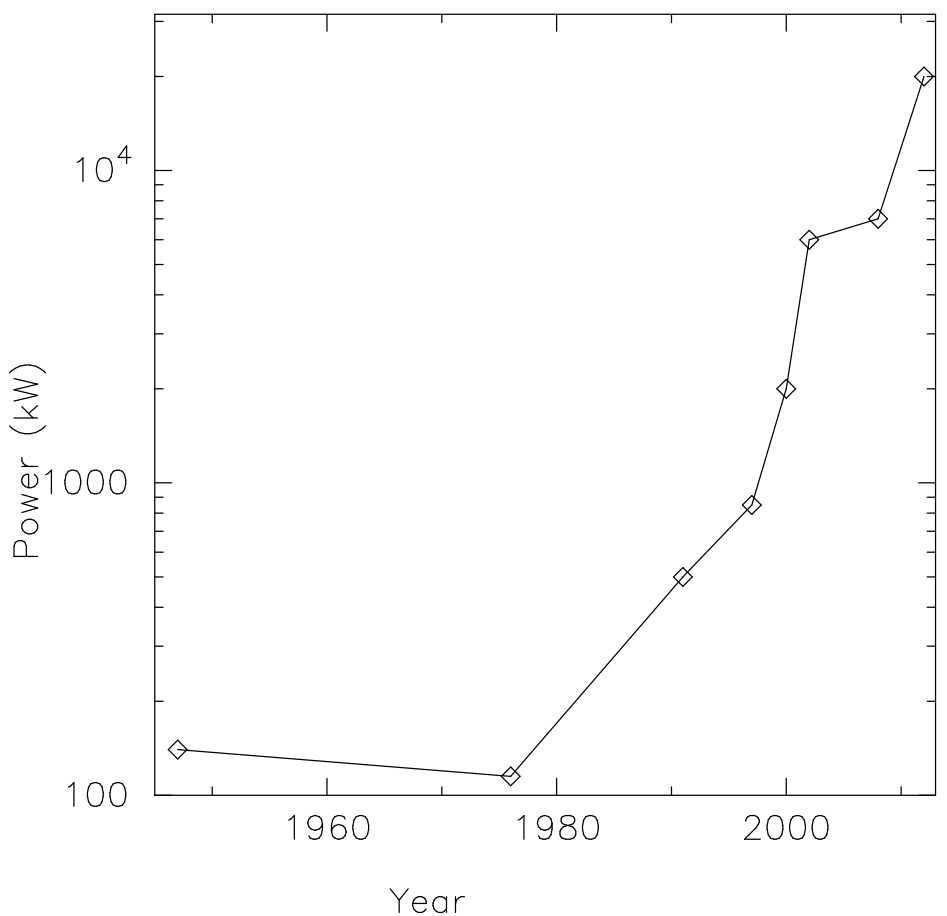


1993-2013: 10年に 500倍!?

問題1: 消費電力

ENIAC	1947	140kW
Cray-1	1976	115kW
Cray C90	1991	500kW
ASCI Red	1997	850kW
ASCI White	2000	2MW
ES	2002	6MW
ORNL XT5	2008	7MW
「京」	2012	20MW

グラフにすると……



ENIAC から Cray-1 まではたいして変わらない

1975-95 の 20 年間に 10 倍

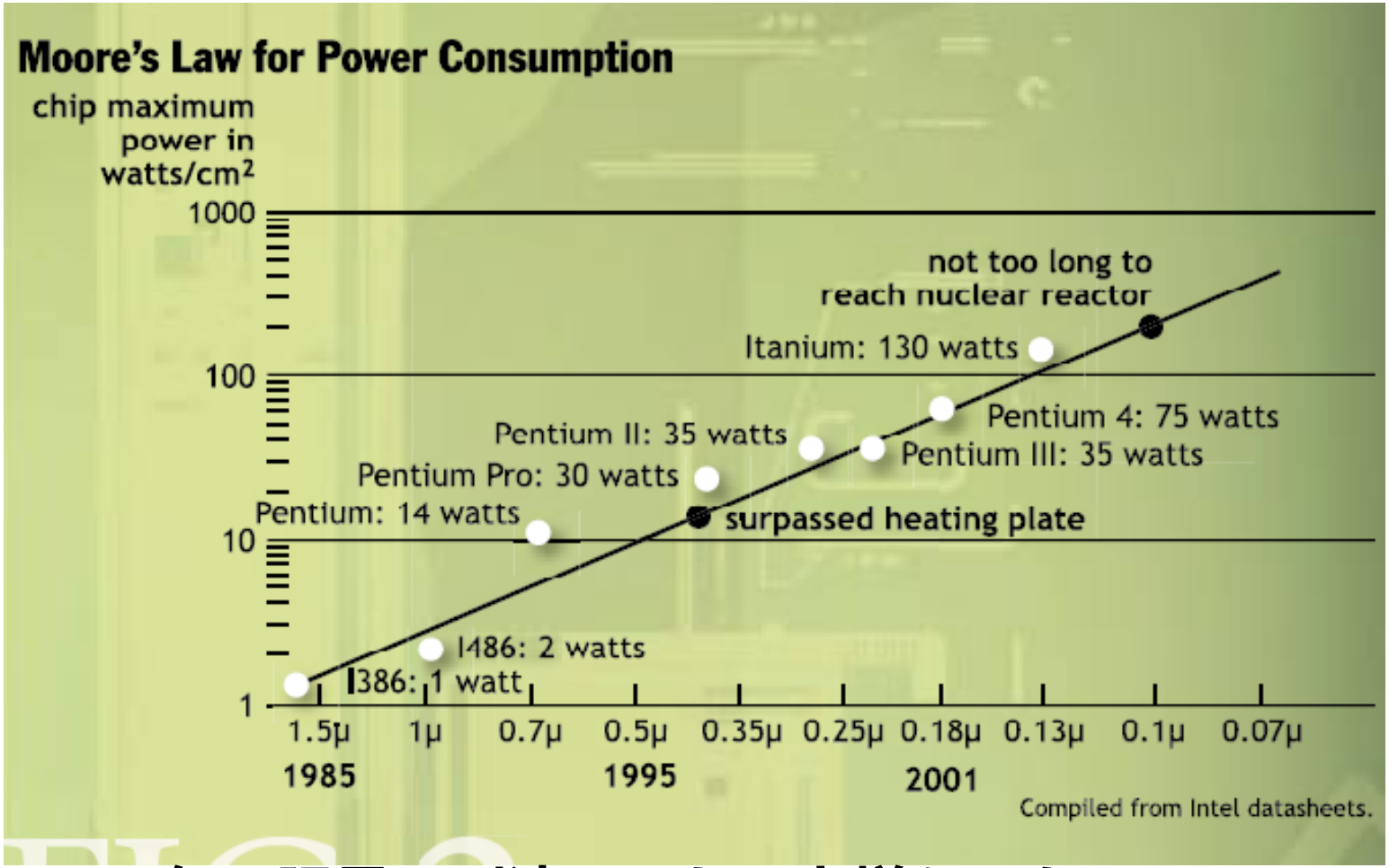
1995-2012 の 17 年間に 30 倍

指数関数より速い増加 (有限時間でクラッシュ)

何故?

- 沢山お金掛けるようになった:: ASCI Red: 50億, 「京」×百億
- チップあたり (ないし面積あたり) の消費電力増加
- チップ面積あたりの価格低下

シリコン面積あたりの消費電力



2003 年に限界に到達、それ以上増えてない (BTX の失敗)

問題 2: 並列化オーバーヘッド

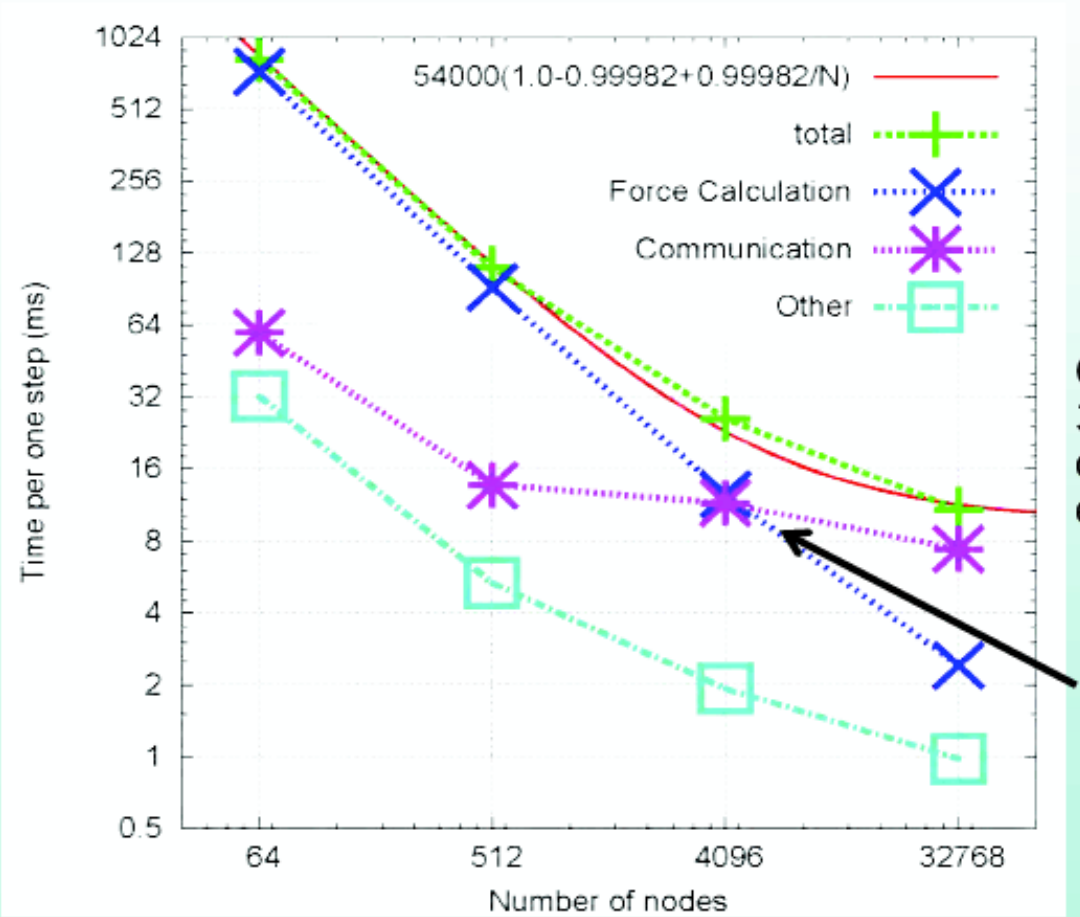
浮動小数点演算器の数 (乗算+加算で1つ)

Cray-1	1976	1
Cray C90	1991	16
ASCI White	2000	16,384
地球シミュレータ	2002	40,960
「京」	2012	2,820,096

「京」は、大自由系の短時間計算はできても、小 (といっても100万とか、、、) 自由度系の長時間計算にはむかない

性能スケーラビリティの例

Strong Scaling (内訳)



Cutoff 28Å,
3,349,656 atom ,
calculate energy
every 4 step

Cross over at
817 atom/node

「京」での分子動力学計算

- 1 タイムステップが 5ms を切らない
- 通信オーバーヘッドが問題

5ms で十分速いか？

- タンパクとかだとマイクロ秒くらいしか計算できない
- 専用機 ANTON は 100 倍以上速い

現在のアーキテクチャの延長では大きな改善は難しい。

並列化オーバーヘッドの起源

演算器の数自体が問題なわけではない

- 「通信時間」のほとんどは、メモリ読み書きのオーバーヘッド
- CPU→キャッシュ→メモリ→NIC→NIC→メモリ→キャッシュ→CPU
- 同期や総和になるともっと大変なことに

問題 3: どうやってプログラム書くの？

- MPI
- OpenMP
- SIMD 拡張
- Cache の有効利用
- アクセラレータ
- ...
- ...

解決の方向は?

- 消費電力と通信オーバーヘッドの両方を減らしたい
- メモリはそんなにいらない (という問題も多い)。100 万原子:100MB。1 兆になっても 100TB。タイムステップが少し多いと 1 兆粒子はエクサでも終わらない。

一つの方向:

- 「小さな」オンチップメモリしかもたないプロセッサチップ (「小さなといっても 256MB とか、、、))
- 単純なコアを多数 SIMD で動かすことで、同期、通信のオーバーヘッドを減らす。

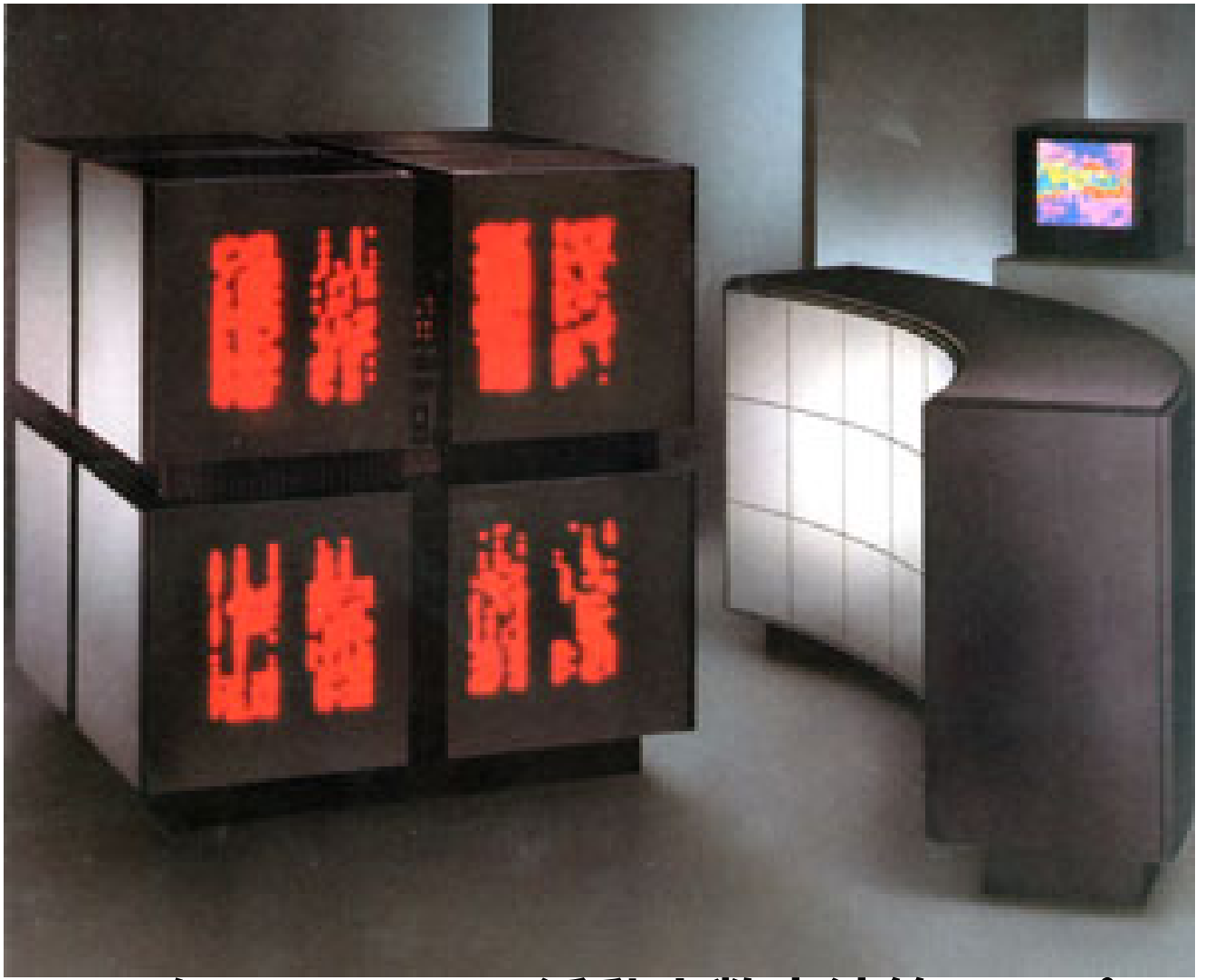
超並列SIMD計算機

— A lost technology —

- Goodyear MPP (1970s)
- ICL DAP (Late 1970s)
- Thinking Machines Connection Machine-1/2 (Late 1980s)
- Maspar MP-1/2 (Early 1990s)

CM-2 はそれなりに成功だった

TMC CM-2



2048 個の Weitek 浮動小数点演算チップセットを SIMD で動かす

TMC CM-2

- 64k 個の 1-bit プロセッサ。メモリ量 64k ビット
- 2048 個の浮動小数点演算器。1 つが 32 プロセッサにシェアされる
- 12 次元ハイパーキューブネットワーク (16 プロセッサが 1 チップ)

現代 (というかちょっと未来) の半導体技術なら、 4-8 台くらいの CM-2 を 1 チップにいれて、100 倍のクロックで動作させ、 10-20 Tflops くらいを消費電力 100W 以下で実現可能

CM-2 ソフトウェア

- *Lisp: データ並列 Lisp
- C*: C++ で実装したデータ並列 C
- CM-Fortran (ほぼ HPF)

もちろん「天才」Guy Steele がいたからできた話ではある。

私のハードディスクから発掘された C* コードの残骸

```
typedef domain node_domain {
    EXTERN int index;              /* index for particle/node */
    EXTERN REAL mass;              /* mass of the node */
    EXTERN REAL position[NDIM]; /* position */
    EXTERN REAL velocity[NDIM]; /* velocity */
    EXTERN REAL acc[NDIM];         /* accelleration */
    EXTERN REAL acc_old[NDIM];  /* accelleration of previous ste
    EXTERN REAL potential;         /* current potential of the part
    EXTERN REAL out_potential;
    EXTERN REAL r_work[NDIM+1];
} NODE_DOM, *NODE_PTR;
```

C* コードの残骸の続き

```
void node_domain::calc_accel()
{
  int myindex;
  mono int i,k;
  REAL dx[NDIM];
  acc[0] = acc[1] = acc[2] = 0.0;
  potential = 0.0;
  myindex = (int)this - (int)&node[0];
  if(this && this < &node[nbody]){
    for(i = 0; i < nbody; i++){
      REAL rsq;
      REAL pot, rsqinv, rinv;
      if(myindex != i){
        rsq = eps2;
        for (k = 0; k < NDIM; k++){
          dx[k]=node[i].position[k]
                -position[k];
          rsq += dx[k]*dx[k];
        }
        rsqinv = 1.0/rsq;
        rinv = sqrt(rsqinv);
        pot=node[i].mass *rinv;
        potential+=pot;
        pot *=rsqinv;
        for (k=0; k<NDIM; k++) {
          acc[k]+=pot*dx[k];
        }
      }
    }
  }
}
```

いいとこ

- 「データ並列」を表現。通信/プロセッサ内データの切り分け、データレイアウトその他はコンパイラ+ランタイムが面倒みてくれる
- 通信は単に配列への間接アクセスで書ける
- 実際に相当複雑な処理が書ける。Barnes-Hut ツリー:構築、粒子毎にバラバラにツリー探索、といったことも。

汎用コアの SIMD との違い: メモリが独立、独立な範囲内ではランダムアクセスできる。

実際には

コンパイラがバグばっかりだったのでこんなの書く羽目に

```
for(k=0; k<4; k++){
    host_work[k] = CM_u_read_from_processor_1L(&node[i],
            &r_work[k], REAL_LEN);
    CM_u_move_constant_1L(&r_work2[k],host_work[k], REAL_LEN);
}
rsq = eps2;
for (k = 0; k < NDIM; k++){
   CM_f_subtract_3_1L(&dx[k],&r_work2[k],&position[k],
                    SIG_LEN,EXP_LEN);
   CM_f_mult_add_1L(&rsq,&dx[k],&dx[k],&rsq,
            SIG_LEN,EXP_LEN);
}
CM_f_divinto_constant_3_1L(&rsqinv,&rsq,1.0,
            SIG_LEN,EXP_LEN);
CM_f_sqrt(&rinv, &rsqinv, SIG_LEN, EXP_LEN);
CM_move(&pot, &r_work2[NDIM], REAL_LEN);
CM_f_multiply(&pot, &rinv, SIG_LEN, EXP_LEN);
```

```
    CM_f_add(&potential, &pot, SIG_LEN, EXP_LEN);
    CM_f_multiply(&pot, &rsqinv, SIG_LEN, EXP_LEN);
    for (k=0; k<NDIM; k++) {
       CM_f_mult_add_1L(&acc[k],&dx[k],&pot,&acc[k],
                SIG_LEN,EXP_LEN);
    }
}
```

非構造格子?

- 非構造疎行列だって書くのは難しくない。全節点でデータ並列動作。ポインタでアクセスすればPE間通信でデータ取ってくる。
- 性能は実は結構でる: 実はほとんどのアクセスは PE メモリ内ですむ。(ように節点番号ふって欲しいな) メモリバンド幅は高い (しアクセス粒度も小さい) から。

汎用スカラー並列に比べてどう改善？

- 消費電力:
  - データ移動が減る。キャッシュ階層がなく、チップ外メモリも (第一義的には) ない
  - 命令フェッチ・デコードのコストも減っている。1 チップに 1 ユニット。
- 通信オーバーヘッド
  - データ移動が減る。外部メモリ、キャッシュ階層がない
  - ハンドシェイク、同期のオーバーヘッドも減る。SIMD で動いている範囲では始めから同期しているので同期操作不要。細粒度通信が可能。
  - チップ間はもうちょっと考える必要がある。データフローマシン的動作をさせたい

SIMD超並列機のネットワーク

ポスト「京」の FS で一応「検討中」あんまりまだ考えてないですが、、、以下は検討中の 1 例

- チップ内: 64 コア程度をクロスバー結合したものが基本ユニット。ユニット間は多段ネットワーク
- チップ間: 16 チップ程度をルータにつなぐ。ルータ間は色々できるように作るがポスト「京」むけは 4D トーラスの通信パターンで性能でるように、、、
- このネットワークでつなぐのは 2048-4096 チップ程度。
- ルータの upward link は総バンド幅で 400GB/s 程度

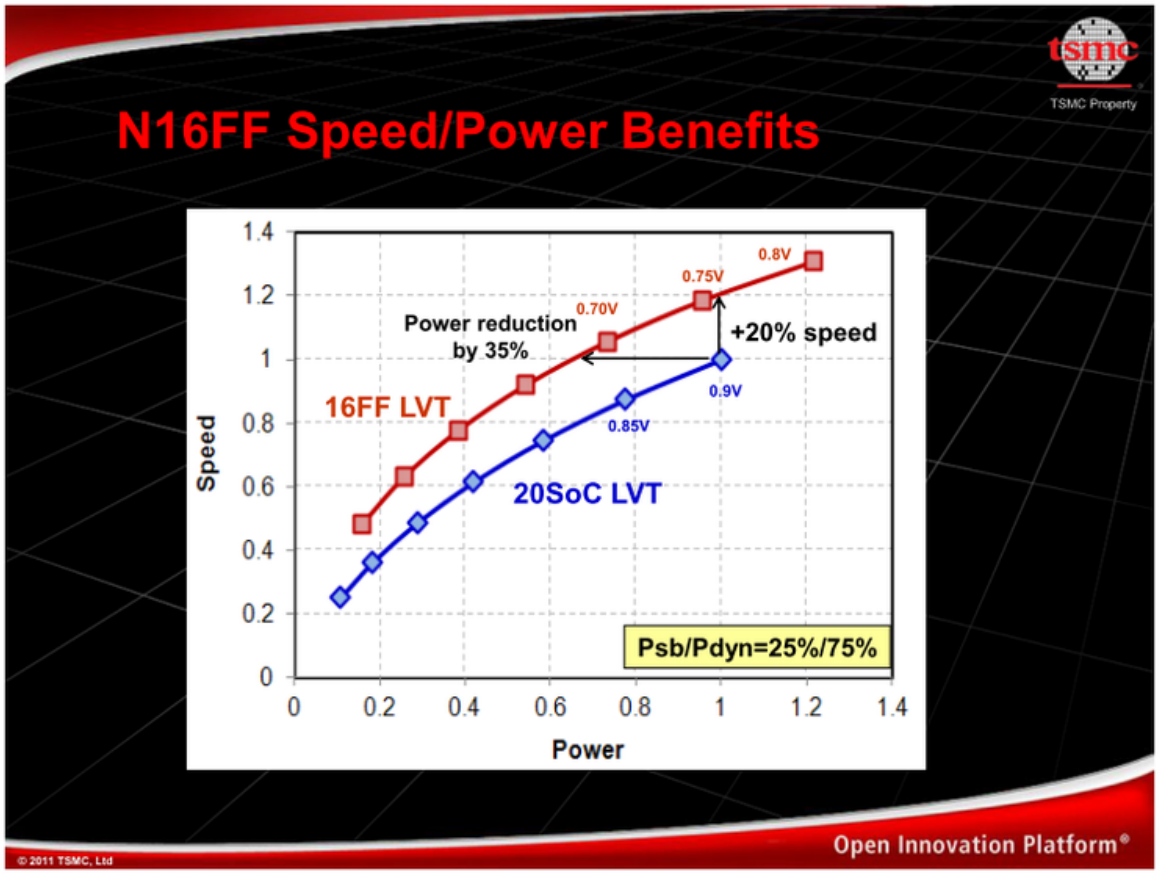
脳の話だと、、、

- ニューロン 10^{11} 個
- シナプス 10^{15} 個
- シナプスの結合情報を全部オンチップでもつのはちょっと無理
- リアルタイムくらいの速さでいいなら、総バンド幅 $10\mathrm{PB/s}$ くらいでいいはず。これは大したことない。
- ネットワークは問題。ランダム結合なら少なくとも (データ量/バイセクションバンド幅)*平均ホップ数 くらいの時間かかる？
- ある程度でも階層化されてるなら、ネットワークはほぼモジュール毎に考えるのでいいはず。

(楽観的な) 評価例

- 昨日の五十嵐さんの例: 神経細胞 17 億。
- 1 ステップの通信は秒オーダー?
- 原理的には、1 チップが受け取るデータは $100\mathrm{GB}$ くらい? オーダーとしては 1 秒スケールで受け取ることができる。
- シナプス演算を例えば 8 ビット加算 1 つ程度で表現「できるなら」、消費電力は倍精度浮動小数点演算の $1/500$ くらい。$50\mathrm{Tops/W}$ くらいはできる。
- 結合データを外部メモリに置くと、それの読出しの電力消費ははるかに大きい。

低電圧動作(TSMC側の主張)



低電圧動作(TSMC側の主張)

- 16nm 0.75V に比べて 0.5V だと電力 1/6、性能 4 割。電力あたり性能は 2 倍以上。
- これ使わない手はない、、、

ポスト「京」


京の100倍 新スパコン開発 NHKニュース - Mozilla Firefox
ファイル(F) 編集(E) 表示(V) 履歴(S) ブックマーク(B) ツール(T) ヘルプ(H)
京の100倍 新スパコン開発 NHKニュース
スパコン - YouTube
www3.nhk.or.jp/news/html/20130508/k10014434551000.html
NHK NEWSWEB
2013年(平成25年)5月8日[水曜日]
文字サイズ:
トップページ > 科学・医療ニュース一覧 > 京の100倍 新スパコン開発
ニュース詳細
京の100倍 新スパコン開発
5月8日 17時55分
文部科学省
2020年ごろの完成目指す
01:20 / 01:20
文部科学省は、おととし計算速度の世界一を達成したスーパーコンピューター「京」の100倍の性能を持つ、新型のスパコンの開発を来年度から始めることになりました。
日本のスーパーコンピューターは、理化学研究所などが開発した「京」がおととし計算速度で世界一となりましたが、その後はアメリカのスパコンに相次いで抜かれ、現在は世界で3番目となっています。こうしたなか、文部科学省は、来年度から「京」の100倍の処理能力を持つ新型のスパコンの開発に乗り出すことを決め、8日開か
主要ニュース
首相 抑止力の観点で厳基地攻撃議論
トヨタ 営業利益増加を発表
川口委員長解任決議案 可決の公算
東京 最後の同潤会アパートを解体へ
長嶋・松井両氏 首相と夕食会
アジア最大規模 東京でバイオ展
大阪「ふたりっ子」舞台の閉鎖
TOPへ戻る
Facebookページはこちらか
気になるニュースは いいね! をクリック
WEB特集
ネット報道の新潮流(1)「まとめ」て読
5月7日(火)
米軍制服組トップ 同行取材記
5月2日(木)
世界遺産登録へ 明暗の訳は？
5月1日(水)
鳥インフル感染確認1か月 対策は
スタート
ネットワーク接続
コマンド プロンプト
iTunes
2 Firefox
20:33

今後の HPCI 計画推進のあり方に関する検討ワーキンググループ (牧野もはいってる)

現在の計画

- 汎用スカラーで系の後継: 2020年でもエクサには 60-80MW いくらなんでも大き過ぎる。
- SIMD 「加速部」をつけたす。
  - 電力を 1/2-1/3 に
  - (加速部ネットワーク内では) 通信オーバーヘッドを 1 桁以上削減
  - (加速部オンチップメモリにのるなら) B/F=4 を実現

まとめ

- 現代の大規模スカラー並列機は「小さい」問題にむかない (現在の「小さい」$=$自由度 10^7、2020 年だと 10^9)
- この問題の解決の一つの方向: SIMD 超並列
- SIMD 超並列による加速部が今のところポスト「京」プロジェクトにはいっている。