エクサスケールマシン開発プロジェクト

牧野淳一郎
理研 計算科学研究機構
エクサスケールコンピューティング開発プロジェクト
コデザイン推進チーム チームリーダー
東工大 地球生命研究所

第 6 回アクセラレーション技術発表討論会 2014/06/20

とタイトルは書いてありますが

今日の話は

「個人の見解であり、所属する組織を代表
するものではありません」

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
- 日本の「ポスト「京」」
- 海外との比較
- 「加速部」
- まとめ

スパコンの進化: 1940-2000

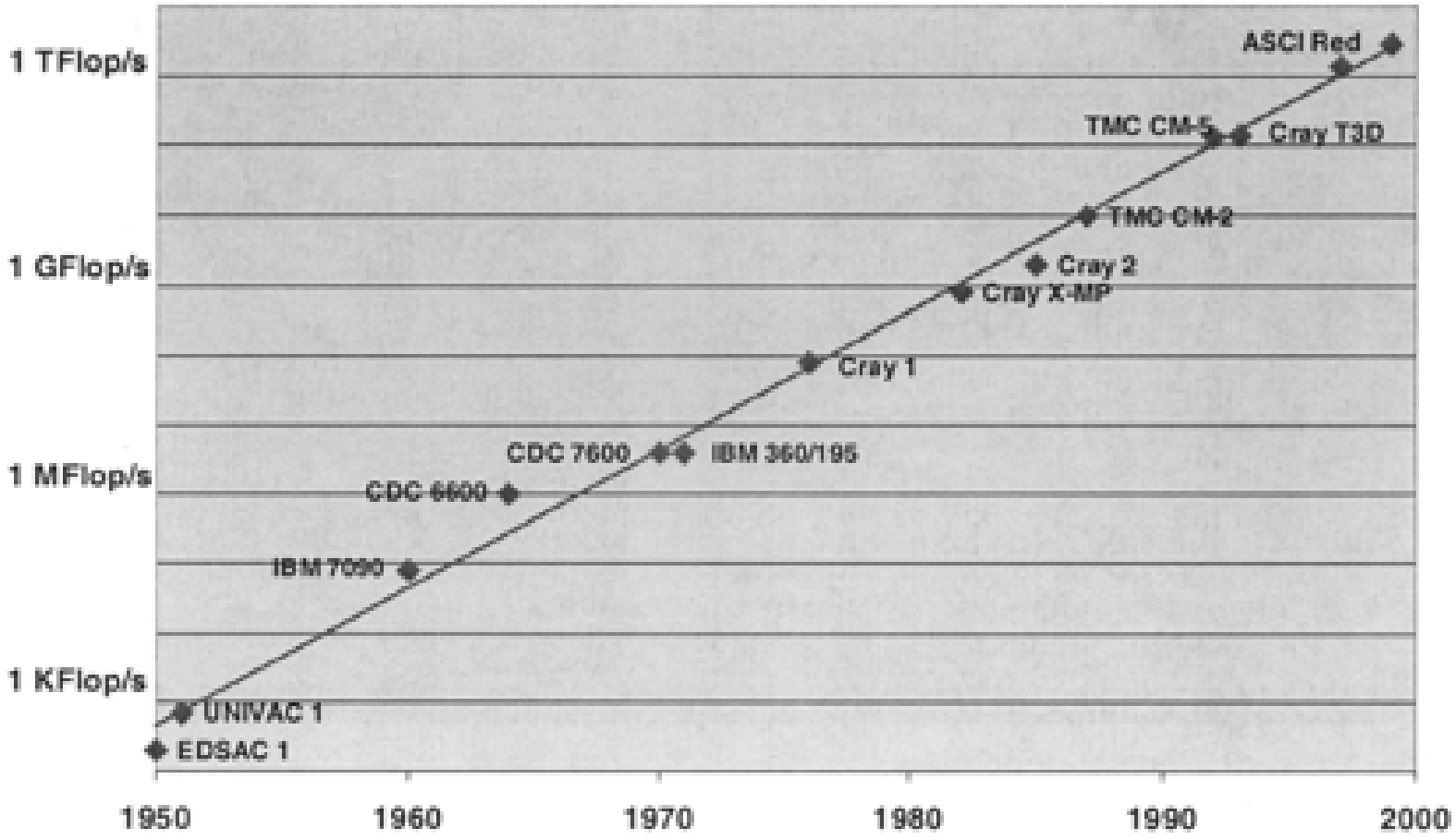
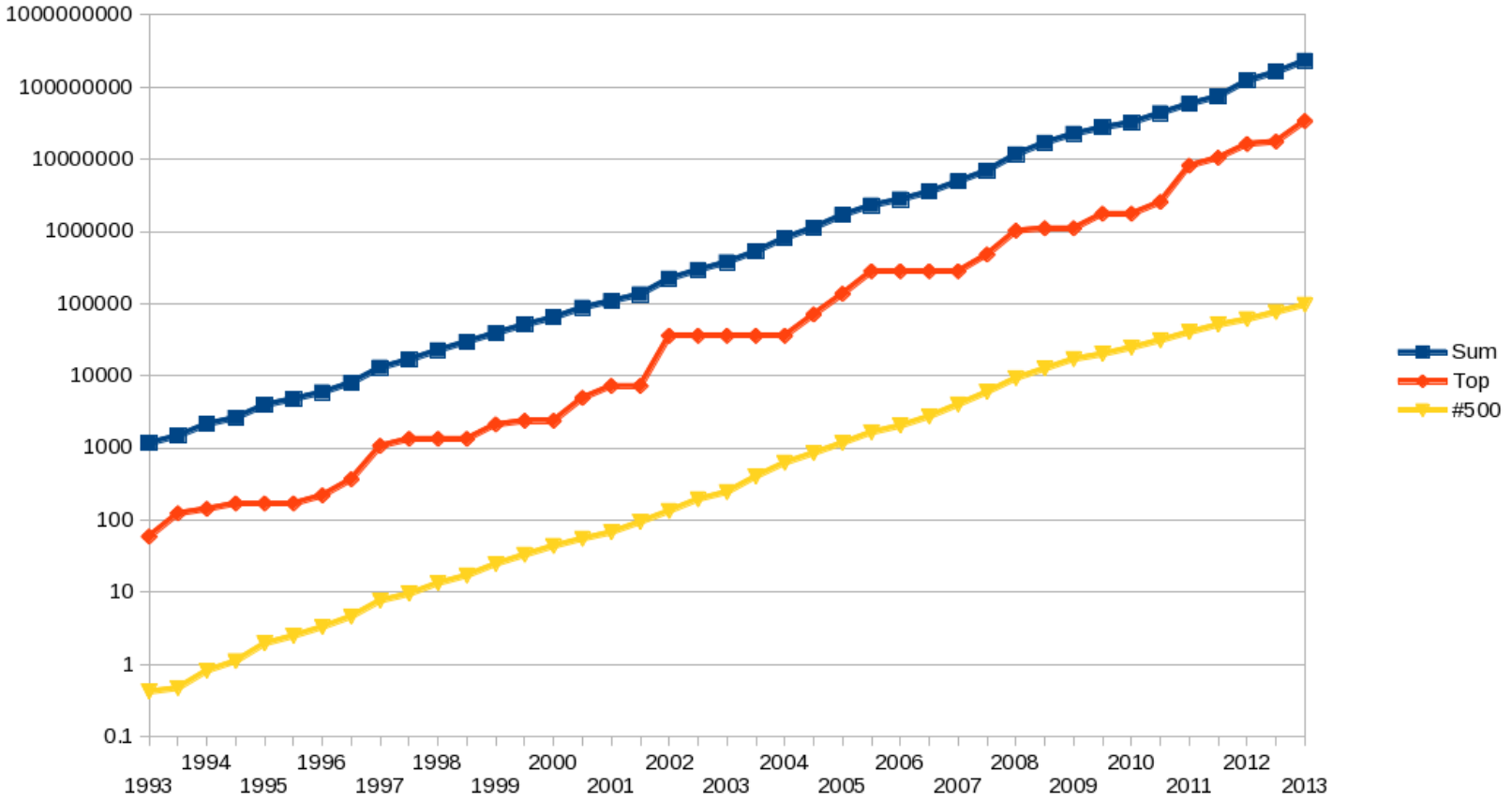


Figure 1. *Moore's law and peak performance of various "supercomputers" over time.*

1940-2000: 10年に 100 倍

スパコンの進化: 1993-2013

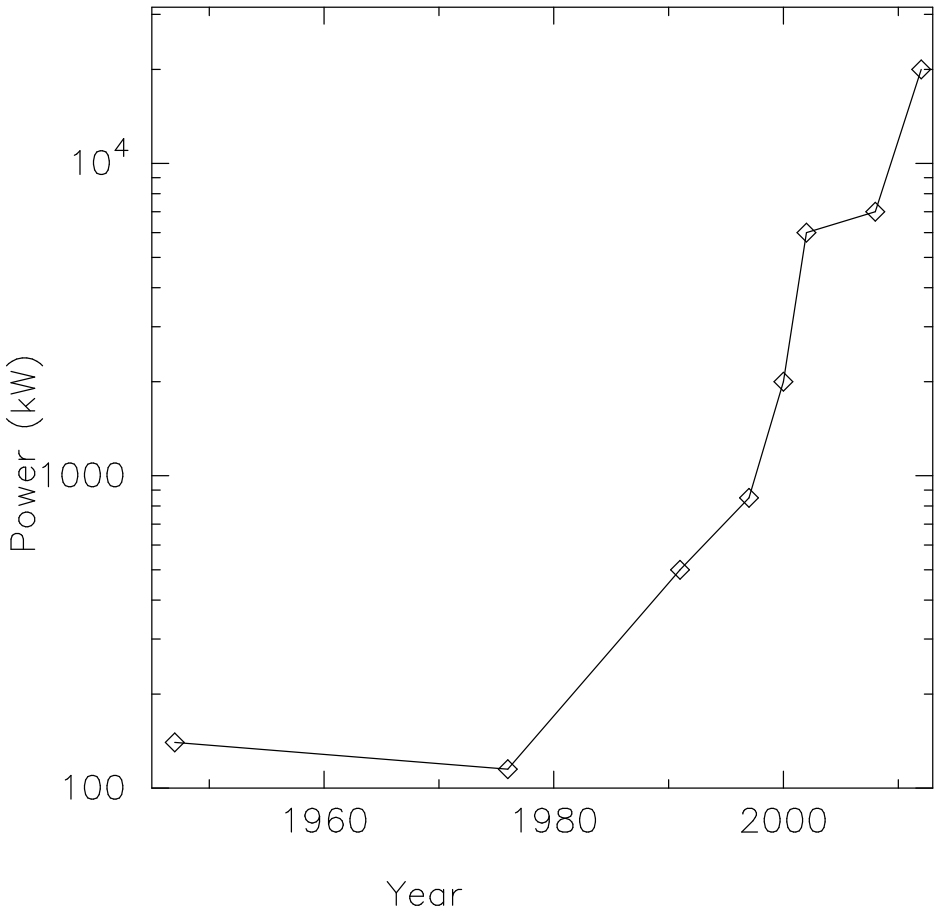


1993-2013: 10年に 500倍!?

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

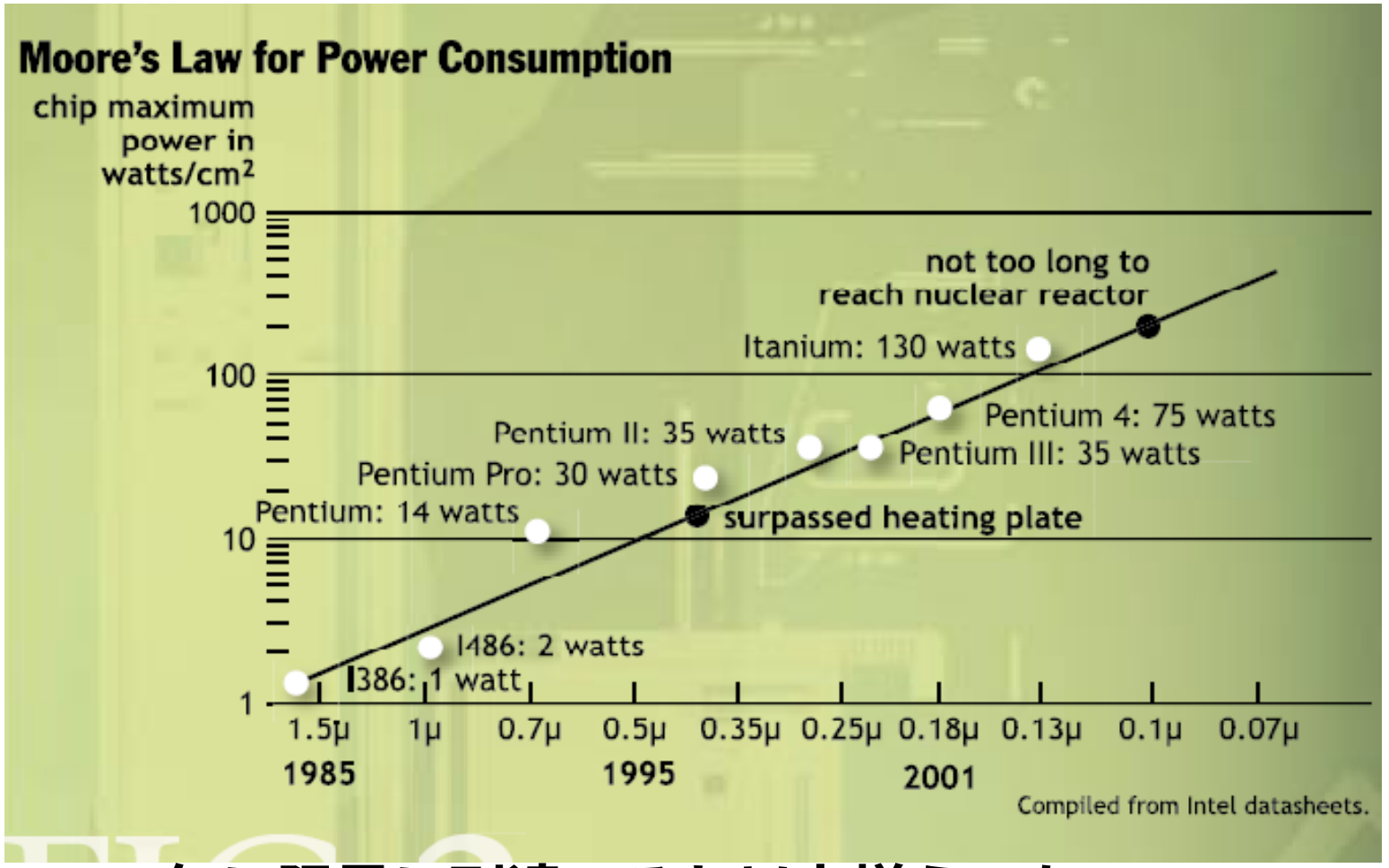
1995-2012 の 17 年間に 30 倍

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

問題 2: 並列化オーバーヘッド

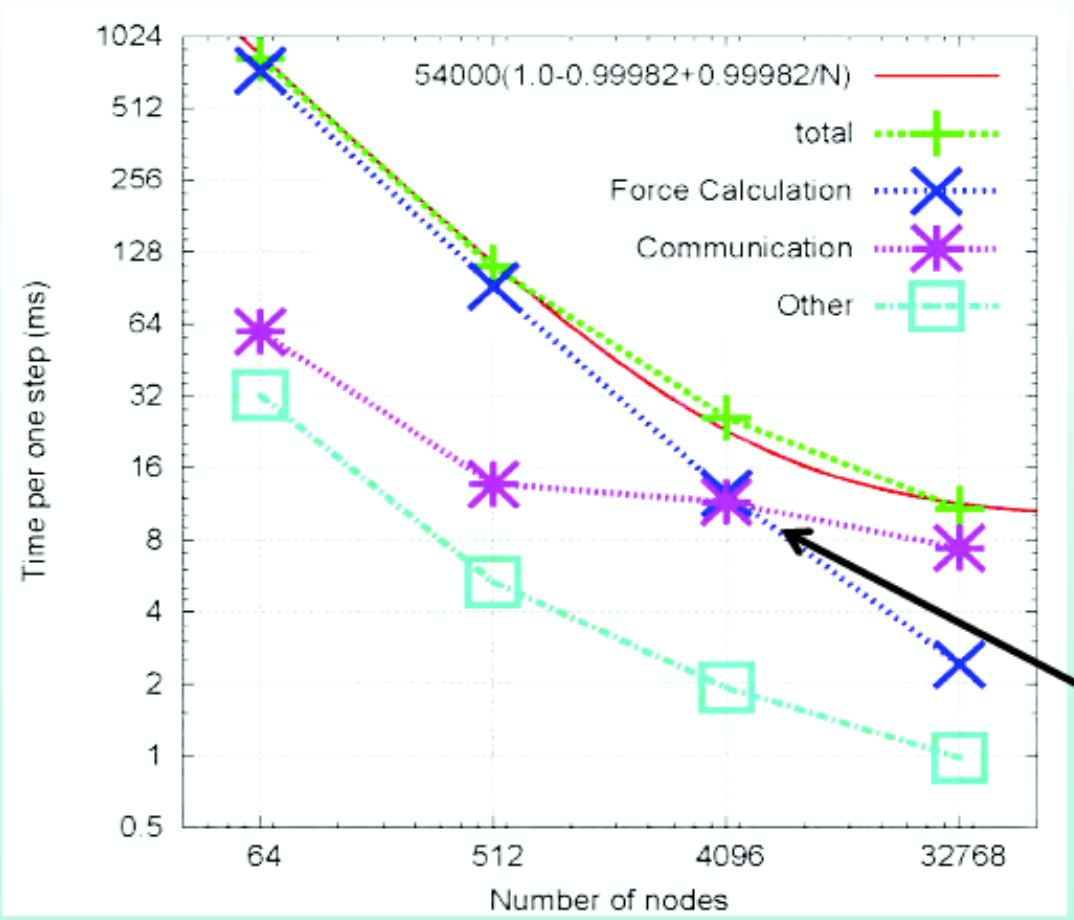
浮動小数点演算器の数 (乗算+加算で1つ)

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
- ...
- ...

で、エクサ(ポスト「京」)は
どうなるのか？

公開資料の最新版: 総合科学技術会議（第116回）議事次第
平成２５年１２月１７日（火）
http://www8.cao.go.jp/cstp/siryo/haihu116/haihu-si116.html
資料1-1 資料1-2 資料1-3
文科省事前評価

役所的プロセスの現状
次期フラッグシップシステムに係るシステム検討ワーキンググループ
第一回:平成26年6月11日（水曜日）17時〜19時

要点をまとめると

- 2020 年頃 エクサフロップス級
- 30-40MW
- 「汎用部」と「加速部」ノードレベルでつながるがそれぞれネットワークを持つ

汎用部

「京」からの「発展」。アプリケーション資産とかなんとかそういう話

	「京」	(FX10)	ポストFX10	エクサ汎用部
年	2011	2012	2014	2018?
プロセス	45	40	28?	10?
コア数	8	16	32?	64?
SIMD幅	2	2	4?	8?
演算性能(GF)	128	237	1,000?	4,000?
B/F	0.5	0.36	0.5?	???
電力性能(GF/W)	0.83	0.8?	?	?

問題と対応

1. 消費電力増加: 微細化とSIMD 幅増大による対応
   - 電力性能が微細化で 5 倍、電圧低下で 2-3 倍、SIMD 幅増加で 2 倍くらい？
2. 並列化オーバーヘッド: ?? 但し、「京」はチップ内はよくできている (同期オーバーヘッドが小さい)
3. どうやってプログラム書くの？: 「京」との連続性で対応

電力は問題。 (他が問題じゃないわけではないけど)

- ピーク性能100倍にしたら電力が5倍になってしまう (60MW 以上)
- 5 年間の電気代だけで 3-400 億。
- 製造コストも大きい

海外との比較

Intel MIC	KNC	KNL	KNH
プロセス	22	14	7??
システム稼働年	2013	2016?	2019?
演算性能 (TF)	1.1	3	>10?
電力性能 (ボード名目) GF/W	3	15?	40?

- KNL よりは高い電力性能。KNH よりかなり低い。
- 値段は KNL くらい (多分)
- Nvidia は Intel より良いであろう

海外との比較

Intel MIC	KNC	KNL	KNH
プロセス	22	14	7??
システム稼働年	2013	2016?	2019?
演算性能 (TF)	1.1	3	>10?
電力性能 (ボード名目) GF/W	3	15?	40?

- KNL よりは高い電力性能。KNH よりかなり低い。
- 値段は KNL くらい (多分)
- Nvidia は Intel より良いであろう

とはいえ KNC には世界中から落胆の声が、、、

Haswell Xeon と Xeon Phi

	KNC	E3-1240L v3
プロセス	22	22
演算性能 (TF)	1.1	0.128
消費電力 (名目 W)	300	25
電力性能 (名目 GF/W)	3.6	5.12

E3 の消費電力 はメモリ含んでないので公平な比較ではない

それでも、、、メニーコアって電力性能良かったのでは ??
結構混迷感が高い

話を戻して、、、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。
- 必要メモリバンド幅は本当はどれだけかはアプリの書き方に依存

一つの方向:

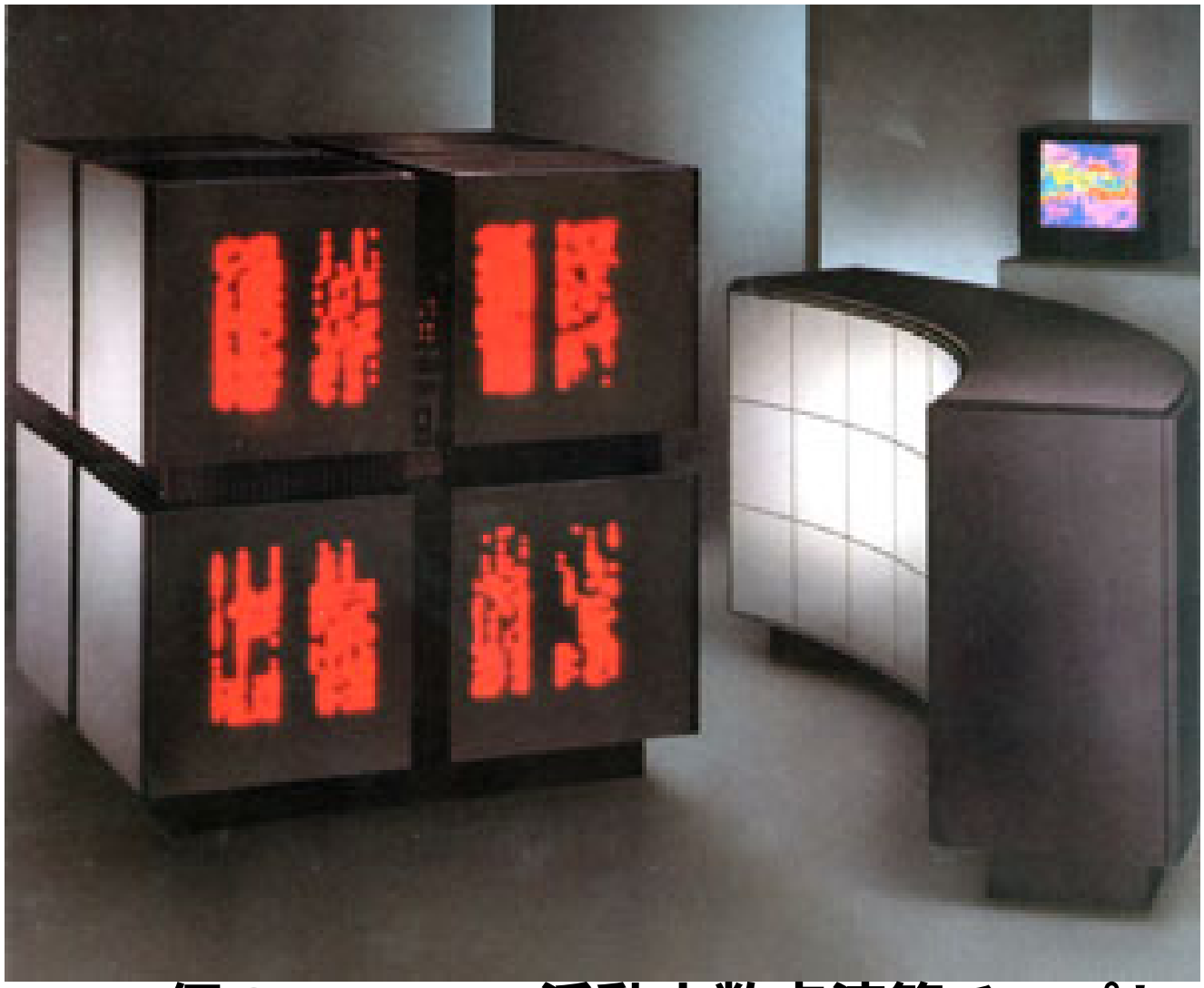
- 「小さな」オンチップメモリしかもたないプロセッサチップ (「小さなといっても 128MB とか、、、))
- 単純なコアを多数 SIMD で動かすことで、同期、通信のオーバーヘッドを減らす。

超並列SIMD計算機

- 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)

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

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 step*/
    EXTERN REAL potential;        /* current potential of the particle */
    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。ベクタプロセッサ的で、ストライドでもインダイレクトでも性能あまり落ちない。

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 で一応検討した。以下は例。

- チップ内: 16-64コア程度をバス結合したものが基本ユニット。ユニット間は隣接(4D)メッシュ+ファットツリー
- チップ間: 4Dトーラス程度のネットワーク。
- このネットワークでつなぐのは 2048-4096 チップ程度。
- チップ間リンクは 10-20GB/s 程度

性能皮算用

GRAPE-DR に比べてどこまでいけるか、という観点

	GRAPE-DR	ポスト「京」加速部
テクノロジー	90nm	10nm
DP 乗算器の数	256	16384?
クロック	500MHz	500MHz
演算性能	256GF	16TF
消費電力	60W	120W
動作電圧	1V	0.75V
メモリ量	1MB	256MB

- 牧野の計算なので楽観的
- でもまあ 28nm で 30GF/W はできた

GRAPE-X ボードとチップ

ボード設計、調整: 台坂
チップ設計: 牧野、中里

GRAPE-X チップ

- TSMC 28HPM
- 3mm x 2mm
- 32 コア、倍精度演算 2flops/cycle, 単精度だと 4
- 内積計算流した時で 30Gflops/W 程度に一応なった
- 10nm ならまあ 100GF/W 超えるのはあまり困難ではなさそう

QCD 性能見積もり (大きなチップ作ったとして)

- KEK の松古さん他と 2012 年度にやったもの
- 計算機想定: 8192 コア、単精度ピーク 16TF、チップ間 4D トーラス、リンク 10GB/s
- 1PE8 格子点 (16kB メモリ)
- クローバークォーク: チップに $16^3 \times 32$, 7.3TF/チップ
- ドメインウォールクォーク: チップに $8^4 \times 32$, 2.3TF/チップ

どちらも通信リミット。チップ間通信減らすようなアルゴリズムがもうちょっと頑張れるとするとまあまあの性能。

メモリ増えるともうちょっと通信は楽。

他に検討中したアプリケーション

- 古典分子動力学
- 規則格子での流体等 (宇宙 MHD、地震、……)
- 宇宙 N 体
- 量子化学計算

実行効率は結構でる (この辺は汎用より高いものが多い)

オンチップメモリでは足りない時

- 規則格子流体、地震波等:オンチップメモリに入らないと全く性能でない
  - 小さい領域を読み込んで複数時間ステップ進めるようなアルゴリズムが必要
  - そんなコードをアプリケーション毎に書くのは全然無理なので、フレームワークやDSLでの対応が必須
- 不規則格子: そもそもCG反復の内積計算がボトルネックだったりする
  - 規則格子と類似した、小さい領域を読み込んで局所的に反復するアルゴリズムが必要
  - CG法自体に対しては、QCDとかでは使っている

まとめ

- 日本のエクサスケールプロジェクトでは 2020 頃エクサということになっている。
- これは汎用 (「京」との連続性維持) だけだと製造コスト的にも電力的にも無理なので、アクセラレータを、というのが現在のところ公式の計画
- 無理とかいう以前に汎用は競争力的に厳しいかも。KNL はともかく KNH、それ以前にただの Xeon に比べてどうか？
- アクセラレータは単純に電力性能、価格性能比を上げるだけではなく、オンチップメモリにはいるアプリケーションに対してベクトル機並みの高い B/F を提供する。
- SIMD で同期オーバーヘッドを減らし、チップ間通信もキャッシュ階層を単純化することで減らす
- さて、、、