次世代高性能計算機への昨今動き (富田さんの代理的なにか) 今後の計算機と計算科学 —なぜ我々は7年前の間違いを繰り返{す、した}のか

牧野淳一郎

東京工業大学 地球生命研究所

理化学研究所 計算科学研究機構

計算惑星科学シンポジウム 2013/11/23

何故今日これをやってるのか？

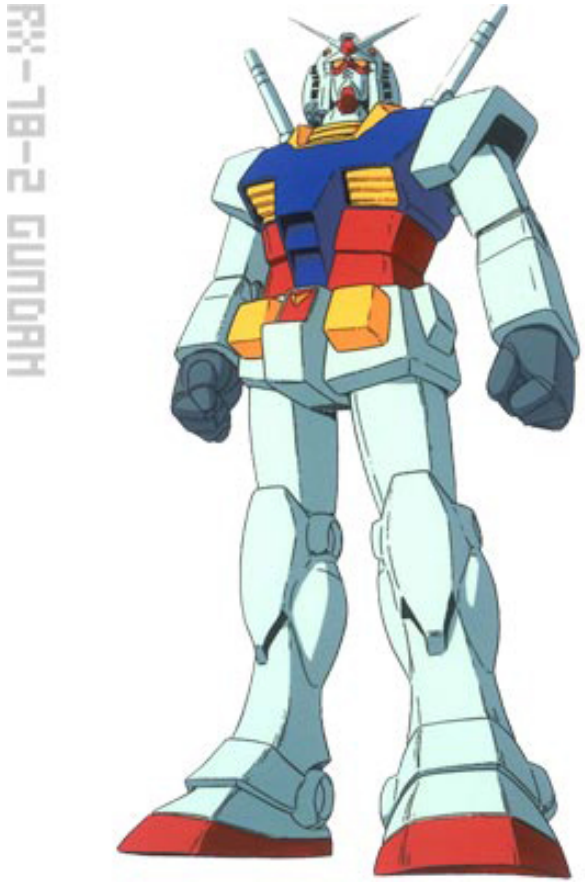
- 「計算惑星科学」をポスト「京」プロジェクトの中に位置付ける
- ポスト「京」プロジェクトに対する惑星科学コミュニティの理解を得る

で、牧野の話は

計算機に関するもろもろ的な話

「京」とその次の計算機の状況

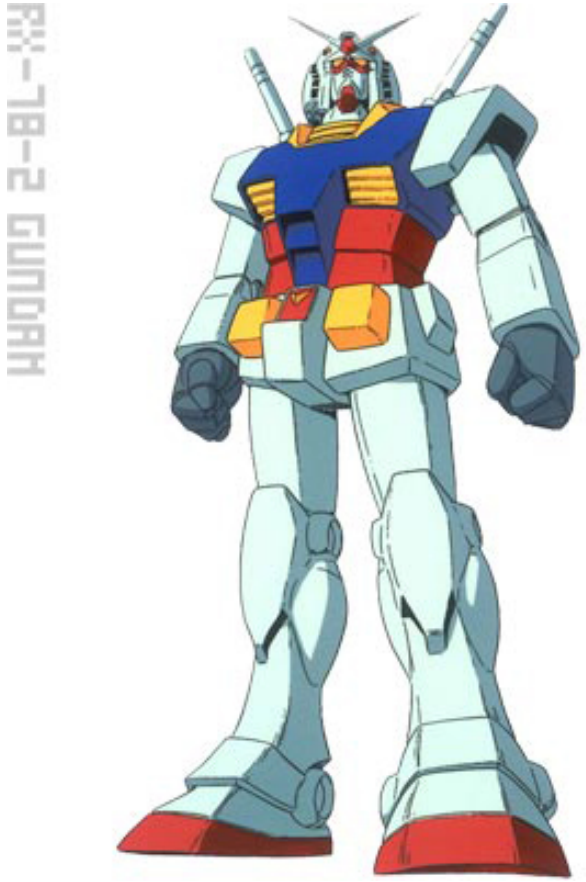
欲しかったのは



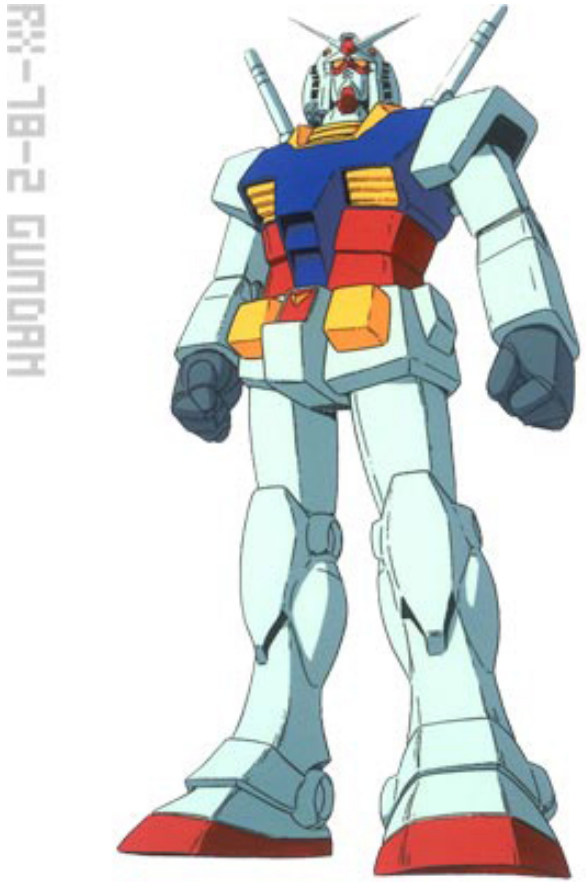
届いたのは

(http://www.zariganiworks.co.jp/korejanairobo/)

何か違う、、、



どうしてこうなった、、、



となるとまだ決まったわけではないという
可能性もまだあるかも

概要

- 「京」プロジェクトのデザインと概念設計
- もうちょっと広い視点から: 半導体技術と計算アーキテクチャ
- 現在進行中のプロセス
  - ポスト「京」検討の経緯 (公開資料等から)
  - 4つのアーキテクチャイメージ
  - FS と「理研提案」
- おまけ: 今後の方向について

「京」プロジェクトのデザインと概念設計

- 大体の時系列
- ターゲットアプリケーション
- 「概念構築に関する共同研究」

大体の時系列

- 2004 年くらいから文科省の下の情報科学技術委員会、計算科学技術推進ワーキンググループで議論
- 2005 年の「中間報告」
  - 8 分野からの要求をまとめた (ことになっている)
  - ベクトル 2PF +スカラー 4PF +「特定処理計算加速機」20PF
- 2006 年度「次世代スーパーコ ンピュータ：概念構築に関する共同研究」実質的にはデザインコンペ。NFH の他、東大 (+天文台)、九大、筑波大等も参加

時系列続き

- 2005 年から 2006 年にかけて CSTP 評価委員会からボロクソに言われる。目標、アーキテクチャを文句がでないように色々変更。
- 2006 年初め (だったと思う) 開発実施本部設置、ヘッド:渡辺 (NEC OB)
- 2007 年夏: ベクトル (?PF) +スカラー (?PF) 合計 10PF 以上、と決定
- 2009 年春: ベクトル担当の N が撤退
- 2009 年 11 月 13 日 (金) 仕分け。「2 位じゃいけないんですか？」
- 2010 年 10 月。プロトタイプ機が Green500 4 位にランクイン
- 2011 年 6 月。8 割完成で Top500 1 位
- 2011 年 11 月。全ノード動作で Top500 1 位。

ターゲットアプリケーション

http://www.nsc.riken.jp/target-application/target-application.html
名前がでたのは 2007/4 だが選定は 2006/4 までになされていた:

1　Cavitation　有限差分法によるキャビテーション流れの非定常計算
2　COCO　超高解像度海洋大循環モデル
3　FrontFlow/Blue　Large Eddy Simulation (LES) に基づく非定常流体解析
4　FrontSTR　有限要素法による構造計算
5　GAMESS　ＦＭＯ分子軌道法計算
6　GNISC　遺伝子発現実験データからの遺伝子ネットワークの推定
7　LANS　航空・宇宙機解析における圧縮性流体計算
8　LatticeQCD　格子ＱＣＤシミュレーションによる素粒子・原子核研究
9　MC-Bflow　血流解析シミュレーション
10　MLTest　オーダーメイド医療実現のための統計的有意差の検証

ターゲットアプリケーション(2)

11　Modylas　高並列汎用分子動力学計算ソフトウェア
12　NICAM　全球雲解像大気大循環モデル
13　NINJA/ASURA　天体の起源を探る超大規模重力多体シミュレーション
14　Octa　粗視化分子動力学計算
15　PHASE　平面波展開第一原理分子動力学解析
16　ProteinDF　巨大タンパク質系の第一原理分子動力学計算
17　RISM/3D-RISM　溶液内タンパク質の電子状態の 3D-RISM/FMO 法による解析
18　RSDFT　実空間第一原理分子動力学計算
19　Seism3D　地震波伝播・強震動シミュレーション
20　sievgene/myPresto　タンパク質・薬物ドッキングシミュレーション
21　SimFold　タンパク質立体構造の予測

但し

実際に性能評価に使われたのは HPL, FFT と以下の 7 個

- SimFold, Modylas: **古典** MD
- GAMESS, RSDFT: **量子化学**
- LANS, NICAM: **流体**
- LQCD: 4 **次元格子での** CG **反復**

この時点で、

- **したい計算**
- **高効率でできる計算**
- **高効率でできるがオーバースペックな計算**

の乖離

大雑把なアプリケーションの特性

- 古典MD、量子化学: メモリバンド幅もネットワークもあまりいらない
- 流体: メモリバンド欲しい。ネットワークはそこそこ
- QCD: メモリ量いらない。メモリバンド幅もネットワークも欲しい

これからわかること:

- 一台で全て満たそうとすると帯に短し襷に長しになる
- 低レイテンシ要求があるアプリケーションははいってない

「京」のデザインポイント

- CPU:128Gflops, 64GB/s, 16GB
- ICC: リンク 5GB/s x 2 x 10, CPU 20GB/s, レイテンシ:論文に書いてない
- 最近の計算機にしては B/F 重視
- 最近の計算機にしてはネットワーク重視
- CPU, ICC 別チップ

BG/Q との比較

	「京」	BG/Q	比率
ピーク速度 (GF)	128	204.8	–
メモリバンド幅 (GB/s)	64	42.6	2.4
ネットワーク (GB/s)	5	2	4
電力あたり性能 (GF/W)	0.85	2.1	(1/2.5)

演算性能あたりで、「京」は BG/Q の 2.4 倍のメモリバンド幅、 4 倍のネットワークバンド幅を持ち、 2.5 倍電気を食う。

どうしてこうなったか？

形式的な理由: 要求仕様にそう書いてあった

- HPL 10PF
- 電力30MW以下
- アクセラレータ付けるならやはり10PF
- この範囲でアプリケーションの性能を上げること

アプリケーションの側からは

欲しいもの:

俺のコードが速く走る計算機をよこせ!

但し: 「俺のコード」には温度差あり

- **必要ならアセンブラででも**
- **コンパイラの出力見るのは基本だよね？**
- **プロファイラ使ったことある**
- **OpenMP という言葉聞いたことがある**
- **並列化ってコンパイラがしてくれるんでしょ？**

最近の問題

アセンブラで書くとかくらいではどうにもならないことがある

- メモリ階層への対応
- 分散メモリ並列化

アルゴリズム、データ構造どちらも根本から見直す必要あり

1ステップもどって考える

何故分散メモリ＋深いメモリ階層なのか？

チップ内の処理能力の増加に比べてチップ間データ転送能力が上がらないから

これは、言い換えると、現在の典型的マイクロプロセッサの設計では

- **演算能力はデータ転送能力に比べて相対的に安い**
- **これは製造コストだけでなく電力コストも**

ということ

さらに言い換えると

- 「京」より演算性能を大幅に上げるのは、メモリバンド幅やネットワークバンド幅をあげなくてもいいなら電力・コストを大きくは増やさなくてもできたはず
- 逆に、 HPL 10PF だけならもっと安価にもできたかもしれない
- メモリバンド幅を増やすなら全く別の作りかたもあったかも

万能輸送機

こんなのが欲しかったのかも？

もうちょっと広い視点から:
半導体技術と計算アーキテクチャ

あるいは:

我々はどこからきて、どこにいくのか？

計算機アーキテクチャの変化

- **1950-60 年代: スカラー計算機 ENIAC — CDC7600**
- **1970-80 年代: ベクトル計算機 Cray-1 — NEC SX-3**
- **1990 年代以降: 分散メモリ計算機。 色々。**

何故変化するか？

基本的には、デバイス技術の変化による。
真空管-(パラメトロンとかもあったけどまあ)-バイポーラトランジスタ-TTL(ECL 等)IC-ECL VLSI -CMOS VLSI
色々境界条件が変わる。

- 使える素子の数 (価格、電力)
- 配線 (信号を動かす) コストと処理 (素子動作) コストの関係

大雑把な傾向

- 素子数は増える
- 配線コストは相対的に上がる

配線コストが上がる理由

要するに、素子数を増やすから。(少なくともCMOSでは)

CMOSトランジスタの「スケーリング則」(但し、これ自体は2002年頃に成り立たなくなるが):

寸法を半分にすると

- 速度2倍
- 消費電力 $1/4$ (速度倍にしても)

ところが、計算機(あるいはチップ1つ)の大きさはあまりかわらない。

- 短い(トランジスタサイズ程度)の配線については同じスケーリング
- 長い配線は消費電力へらない。面積もへらない(RC遅延を小さくするためには細くできない)
- それでも長い配線は速くならない

何故CMOSスケーリングは終わったか

理由は単純: 動作電圧を下げられなくなった。

- CMOSスケーリングの仮定:サイズに比例して電圧下げる
- $90\mu\mathrm{m}$ で $1\mathrm{V}$、そこからほとんどさがってない
- 半導体バンドギャップとの関係、リーク増大、トランジスタばらつきの影響など

話を戻して、まずスカラー計算機から

「スカラー計算機」って何？

- 1 つの命令で 1 演算 (最大)
- 命令を順番に実行 (パイプラインでクロックサイクル毎に、というのはある)

何故スカラー計算機だったか？

- 完全パイプラインの演算器を作るには使える素子が足りなかった
- 1965 年くらいまでの話 (CDC6600 まで)

ベクトル計算機

技術的な要因は2つ

- 完全パイプラインの演算器がつくれるようになった
- メモリが磁気コアから半導体に変わって、今までよりずっと高速になった(1チップの容量が小さかったので、メモリのトータルのピン数が多く、バンド幅がありあまるほどあった)

高速な演算器と高速なメモリを効率良く使える命令、制御回路が必要。

「同じ演算をメモリの並んだ要素に順番にやる」ベクトル命令は都合がいい

もちろん、ベクトルに対する操作=並列演算 ではある

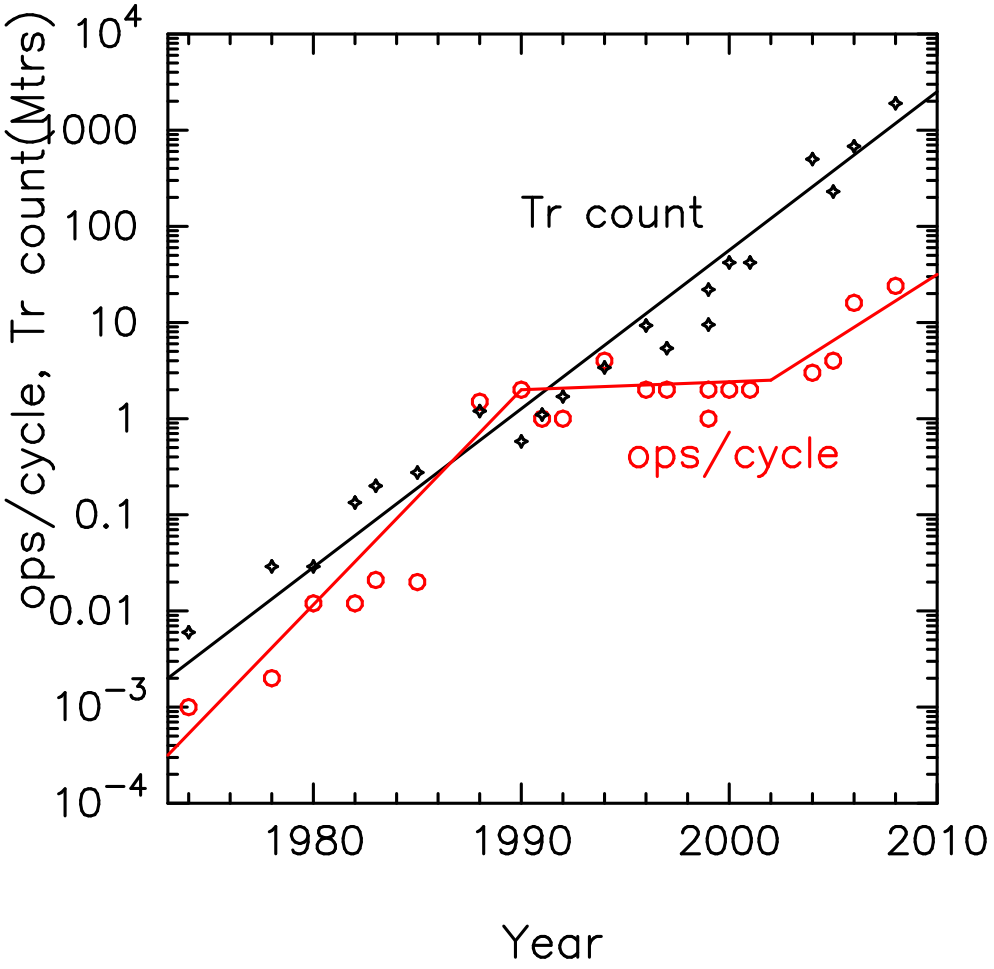
そのあとの半導体技術の発展

- 80年代初めまでのベクトル計算機の半導体: ECL 16-200ゲート程度。乗算器1つに数百チップ。
- CMOS なら当時すでに数千トランジスタ以上。80年代中頃には「乗算器チップ」
- 使えるトランジスタの数は10年で大体100倍に増えてきた。90年代中頃には演算器100個くらいは入るようになったはず

90年前後に起こったこと

- 米国の ECL ベースマシンベンダの崩壊(Cray T-90, ETA-10)
- 日本メーカーの残存者利益 (SX-3,4, VPP-500)
- 分散メモリ CMOS マイクロプロセッサ並列機への移行 (Intel Paragon, Cray T3D)

プロセッサはどうなったか？



黒: マイクロプロセッサのトランジスタ数。10年で100倍弱

赤: 演算器の数(90年以前は必要サイクル数の逆数)。90年代に増えなかった。ここ数年も停滞

なぜ演算器が増えなかったか

メモリバンド幅を用意できないから

- 95年には、100MHz の演算器100個を1チップにいれるのは計算上は可能
- でも、それぞれを64ビット幅でメモリにつなぐと、ピンが信号だけで1 万本くらい、メモリチップも1000個くらい必要。

対応策:

- 演算器は減らす。あまったトランジスタをキャッシュメモリにして、アプリケーションが上手くキャッシュを使うことを期待 (スカラプロセッサ)
- メモリへのピンは可能な限り沢山つけ、それでたりるところまで演算器は減らす。(ベクトルプロセッサ)

当然だが前者は安くて後者は高い。

この傾向は現在までずっと続いている。但し、演算器の数は2002 以降増え始めた

なぜ最近は演算器が増えるのか

- クロック上がらないので演算器増やす以外に性能あげる方法がない
- メモリインターフェースのクロックはまだあげることができ、ある程度のメモリバンド幅増加はまだ可能

GPU はもっと演算器多いじゃないか？

確かにそう。なぜそれが可能か？

- DDRx よりクロックの高い GDDRx を使っている (その代わり、総メモリ容量が小さい)
- 内部クロックは最近の CPU よりずっと低い。

同じ CMOS VLSI の制限の中でのバリエーション

今までの話を単純化すると

- 磁気コアメモリから半導体メモリへの移行によってスカラー計算機からベクトル計算機への移行 (1970 年代)
- ECL ロジックから大規模集積が可能な CMOS への移行によってベクトル計算機からスカラー分散メモリ並列計算機への移行 (1990 年代)
- CMOS スケーリング則の終焉によってマルチコア化 (2000 年代)

が起こった。

「京」のポジション

- 当初の計画: ベクトル+スカラー。 1970 年代+1990 年代。相当時代遅れ
- 途中でベクトル脱落したのでそれほど時代遅れではなくなった
- 時代の先ではない

「時代の先」はあるのか？

- もちろんそんなのは将来になってみないと分からない。
- とはいえ、色々なことを色々な人がやって、上手くいったものが時代を作る

なにがありそうか？ $=$ なにが限界にきているか、何はまだ伸びるか

- 限界の1つ $=$ DRAMチップ容量

1970	1k bits
1998	64M
2006	2G
2013	4G

最近ほとんど増えなくなった。キャパシタ容量の下限に近づいてきたため

DRAM以外のメモリ

- といっても、バンド幅が減るのでは話にならない
- バンド幅はしょせんチップ間の線の数なので、メモリセルがなんでも関係ない

というわけでもない。メモリがプロセッサチップの中ならどうか？

- (大きなチップなら) 現在でも 1G ビット、2018 年頃には 8G ビットくらいは入る。
- 面積効率で DRAM との差が小さくなる。電力、バンド幅では圧倒的に優位

これはまあ 1 つの方向ではありえる。DRAM から SRAM への移行。

さて、ポスト「京」

- 2011 年から検討始まった
- 最初に、 2011/7 くらいまでの短期のワーキンググループ。結論はこんな感じ:
  - アーキテクチャ、システムソフトウェア、アプリケーションの 3 作業部会で年度内にもうちょっと検討しろ
  - 適したアプリケーションが違う複数アーキテクチャもありでは?

2011年度のアプリケーション部会

- 夏頃に突然つくれという話が発生した。
- アーキテクチャ・コンパイラ・システムソフトウェアのほうは SDHPC 検討グループが横すべり
- アーキテクチャは戦略分野等から人を集めて急拠立ち上げ。9-11 月に集中的にミーティング、検討。

以下11/15 の合同部会での牧野の報告から抜粋

予備検討の方針(1)

- アーキテクチャ部会での検討では、B/F、メモリ量、ネットワークバンド幅等について非常に狭い範囲しか想定していないように思われた
- 消費電力当り性能は、エクサスケール実現にとって大きな壁である。
- B/F はアプリケーションの効率に大きく影響する一方、消費電力当り性能にも大きな影響をもつ
- メモリ量、ネットワークバンド幅も、大きく変えれば電力に影響する

アプリケーション側で、 B/F、メモリ量、ネットワークバンド幅の必要量をだしておこう

What I learned from Steve Jobs

— Guy Kawasaki

1. Experts are clueless
2. Customers cannot tell you what they need
3. Jump to the next curve
4. The biggest challenges beget best work
5. Design counts
6. Changing your mind is a sign of intelligence
7. ”Value” is different from ”price”
8. A players hire A+ players

(**いくつか省略**)

予備検討の方針 (2)

- アプリケーション、アルゴリズムにより B/F 等への要求は変わる
- 特に、同じアプリケーションでもアルゴリズムが変われば、またアルゴリズムが同じでも系のサイズ等だけによっても要求は変わる

といった問題があるので、

- 各分野に、重要なアプリケーション (計算法、系のサイズ等含めて) を選定してもらい、それぞれについて要求を見積もってもらう
- それらをいくつかのタイプに分類できるかどうか検討する

という方針を考えた。

アプリケーション

38 アプリケーション

分野	数
1	7
2	13
3	4
4	8
5	7

検討結果

(詳しい話は今日は省略)

- B/F とネットワークバンド幅は関係あり。
- B/F 要求高いがネットワークは弱くていいものはある。逆はない
- ランダムアクセス、非数値計算等、この軸ではよく表現できない要求もある
- 大雑把に数種類にタイプわけできそう

タイプわけの観点

観察:

- $\mathrm{B/F}$ は 0.1 以上の高いものと、桁で小さいものに分かれる
- メモリ要求にも非常に幅がある

注意事項:

- 分野によってはまだ十分な検討が進んでいない
- 分野によっては、そもそもアプリケーション・アルゴリズムの進化が速いために定量的な要求を明確にしにくいところもある

タイプわけの観点

「アプリケーションタイプ」でなくて「アーキテクチャタイプ」としてみた。

- **そのほうが物理的制約をイメージしやすい**
- **アーキテクチャ側との議論もしやすい？**

タイプわけ

以下の 4 タイプ

タイプ	B/F	メモリ量 (1TF)	消費電力 (1EF)	演算性能) (20MW)	バンド幅 (20MW)
ベースライン	0.1	10-100GB	20MW	1EF	0.1EB/s
SoC	4	5-10MB	2-5MW	4-10EF	16-40EB/s
アクセラレータ	0.001	1-10GB	4-10MW	2-5EF	2-5PB/s
バンド幅重視	1	1TB	120MW	0.15EF	0.15EB/s

最終レポートではこんな感じ(1)

2. サイエンスロードマップ「アプリケーションからの要求の概要」(9/9)

ネットワークレイテンシ

タンパクMD	1時間ステップがマイクロ秒程度。同期等がこれより十分短い必要あり
格子QCD	大域縮約をマイクロ秒程度
他の多く	もう少し余裕あり

ネットワークバンド幅

格子QCD	隣接ノードとの通信速度が B/F で 0.01 程度
大域FFT	バイセクションバンド幅で性能が決まる。普通の構成では効率 1% 以下 ハードウェアだけでなく、アルゴリズム面からの検討も重要

メモリ容量・バンド幅

地震波動解析 圧縮性流体計算 有限要素解析の 防災・工学応用	100ペタバイト前後のメモリ、高いメモリバンド幅(B/F 0.5 以上)が必要
タンパクMD 格子QCD	メモリ必要量は極めて小さい。大きなバンド幅(B/F 1以上) が必要
大規模粒子系計算 量子化学計算	バンド幅、メモリ量とも比較的要求小さい

ストレージ容量 速度

DNA	シーケンサデータ処理 50EB, 500TB/s 程度が必要
他の多く	1桁程度下の要求

多様な要求

・複数アーキテクチャも視野にいれる必要あり？

・メモリ・ネットワークバンド幅については新しいアルゴリズムの研究開発も重要

最終レポートではこんな感じ(2)

4. ロードマップ達成に向けて(アプリ要求性能)

- サイエンスロードマップ基づいて2018年ごろのアプリケーションに必要な性能を調査
  - 演算性能要求・総メモリ容量・演算性能あたりメモリ帯域・ネットワーク要求を調査

要求性能の解析結果

- 演算性能・メモリ容量・メモリ帯域に関する要求
  - 演算性能は800PFLOPS～2500PFLOPS
  - メモリ容量は10TB～500PBの幅があり、帯域も1000倍程度の差がある
  - 特徴的なもの
    - メモリ容量が少なくても良い: MD・気候・宇宙物理・素粒子物理
    - メモリ帯域が少なくても良い: 量子化学・原子核物理
    - メモリ容量・帯域が両方必要: 構造解析・非圧縮流体解析など
- ネットワークに対する要求　※トポロジに依存する部分もあり継続検討が必要
  - レイテンシ・帯域とも強い要求はないが、性能必要なアプリもあった
    - タンパク質の構造解析などでは1us以下での通信が必要な見込み
    - 物質化学分野ではBisection帯域が必要なアプリもある
    - 1us以下での高速な同期・放送・縮約などが要求されるアプリケーションもあり、専用のハードウェアによるサポートが必要になる可能性もある
- ストレージに対する要求
  - 要求容量に対して他の課題に比べて大きな課題はない
  - 性能要求に対しては今後のストレージデバイス技術に応じて構成方法を検討

- 要求性能をトレンドから予想される性能にマッピングした(図1)
  - サイエンスロードマップの達成には、前スライドの4分類とも、技術トレンドから予想される性能よりも高い数値が要求されている
- ロードマップ達成のためにアプリケーションの特性をさらに詳細化・定量化し、将来のスーパーコンピュータの設計目標を提示していくことが必要である

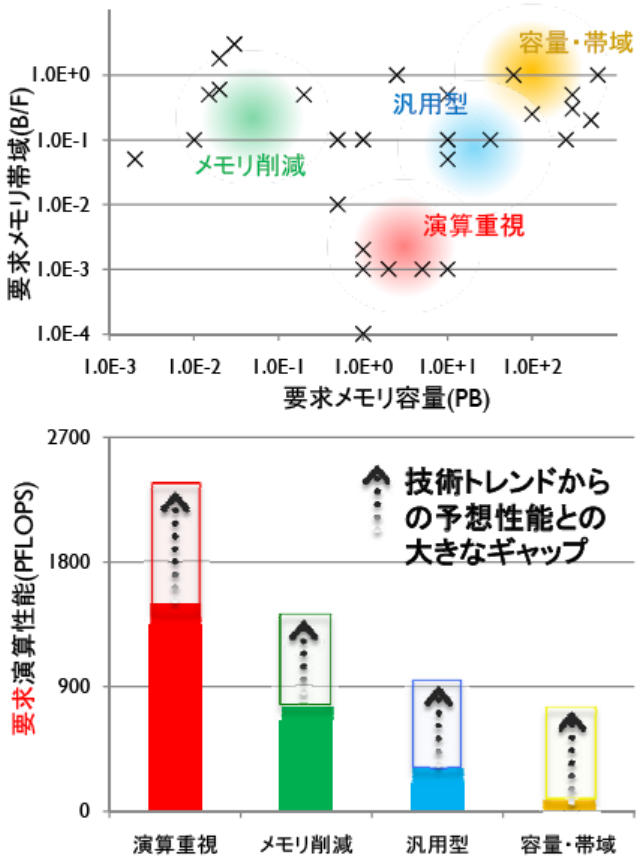


図1. 要求性能と予測性能の相関
上: 各アプリの要求するB/F値とメモリ容量
下: 各分類に対する要求性能値と予想性能

要するに

報告書での用語	アプリ部会のつけた名前	具体的イメージ
汎用	ベースライン	「京」の延長
容量・帯域	バンド幅重視	NEC SX-9 の延長
演算重視	アクセラレータ	GRAPE-DR,GPGPU
メモリ削減	SoC	...

注意

- 「汎用」は汎用ではない (「京」でうまく効率がでないアプリケーションはいくらでもある)
- 「容量・帯域」が本当にそのどっちかでも実現できる設計解があるかどうかは自明ではない
- 演算重視は要するに $\mathrm{B/F}$ 要求が低いものを対象にする
- 「メモリ削減」はオンチップメモリないし 3 次元実装でバンド幅を増やすのが本質。小容量になるのは結果

汎用/ベースライン

- 「京」の延長
- B/F $0.1 \sim 0.2$?
- 富士通さん頑張って

演算重視/アクセラレータ

- メモリ帯域が少なくて良いとなったアプリケーションの大半が量子系 (密行列の直交化や対角化が計算量のほとんどを占める)
- 後は大規模な粒子系
- GPGPU 的なものでいいが、アクセラレータ側に外部メモリはあまりなくてもいい (そちらにメモリあるならホストはなんのため?という問題も)
- GRAPE-DR ベース？

メモリ削減/SoC

- 想定アプリケーション: 小サイズMD、流体、QCD等
- 大サイズ差分法を out-of-core でできるかどうかは要検討
- 外付けメモリがないか、極端に低バンド幅
- 軽量コアを非常に多数集積して、電力当り性能を上げる
- オンチップメモリに対しては 高B/F
- メモリはチップ当り 1GB以下程度？
- ネットワークは 100GB/s 程度？

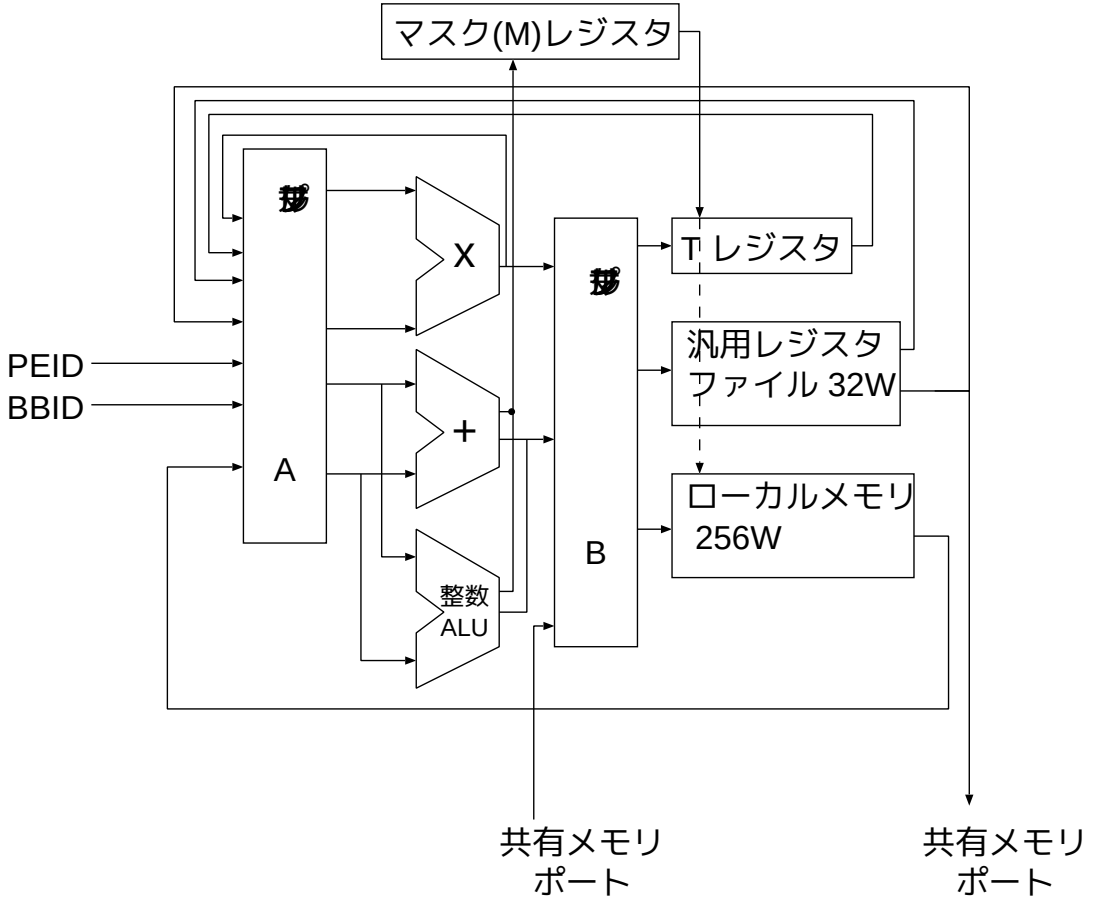
容量・帯域/バンド幅重視

- SX-9 の延長
- NEC さん頑張って、、、

演算重視・メモリ削減のもうちょっと具体的なイメージ

- 大雑把には: 70-80年代の大規模SIMDマシンを1チップ化。Goodyear MPP, CM, MasPar 等
- 例: CM-2。 2048 FPU, トータル 512MB メモリ
- 14nm だと 16384 FPU, 256-512MB メモリくらいが入るかも
- 考えるべきことは
  - コア内部アーキテクチャ
  - ネットワーク
  - プログラミング環境

コアアーキテクチャ:GRAPE-DR PE の構造



- 浮動小数点演算器
- 整数演算器
- レジスタ
- メモリ 256 語 (今回増やす)

多分これからあんまり変えなくても大丈夫そう

ネットワーク:何が問題か？

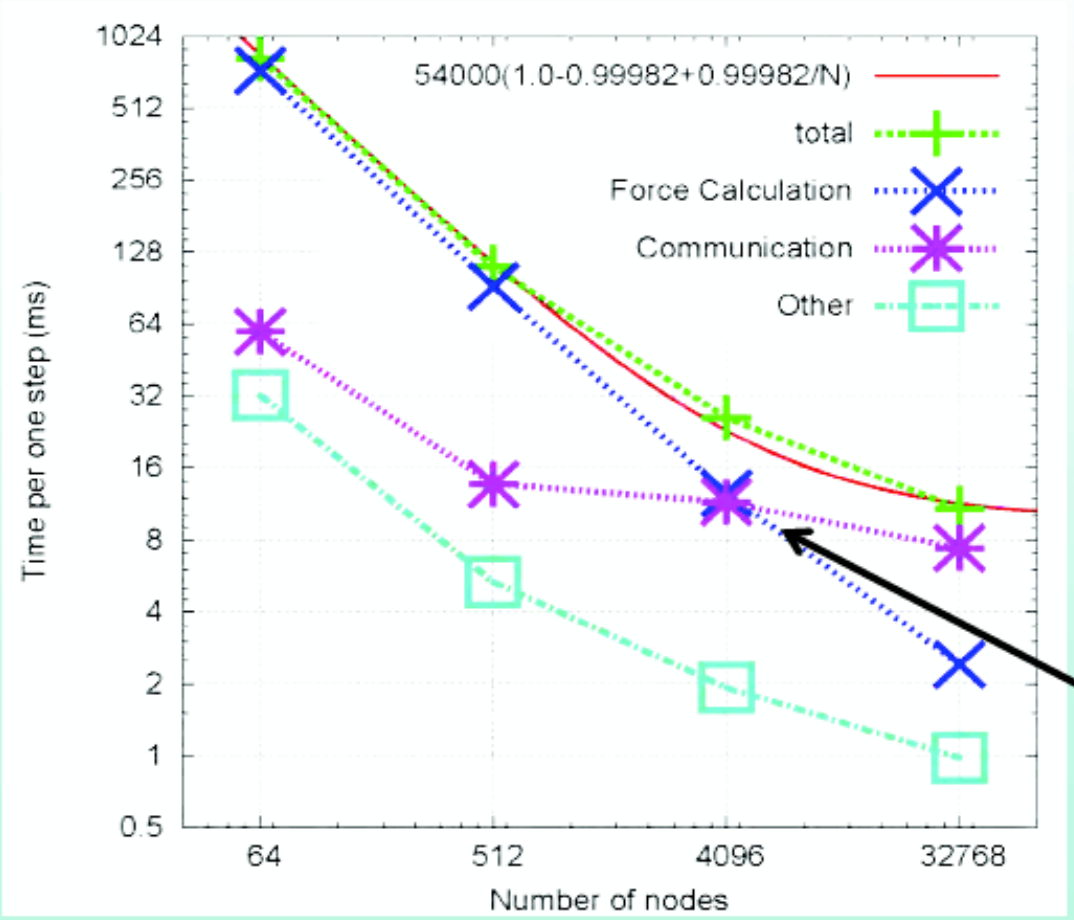
浮動小数点演算器の数 (乗算+加算で1つ)

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

もう一つの問題:
どうやってプログラムするの？

- 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。

話としては、メモリバンド幅が高く、同期・通信オーバーヘッドが小さいなら割合簡単にプログラム書ける

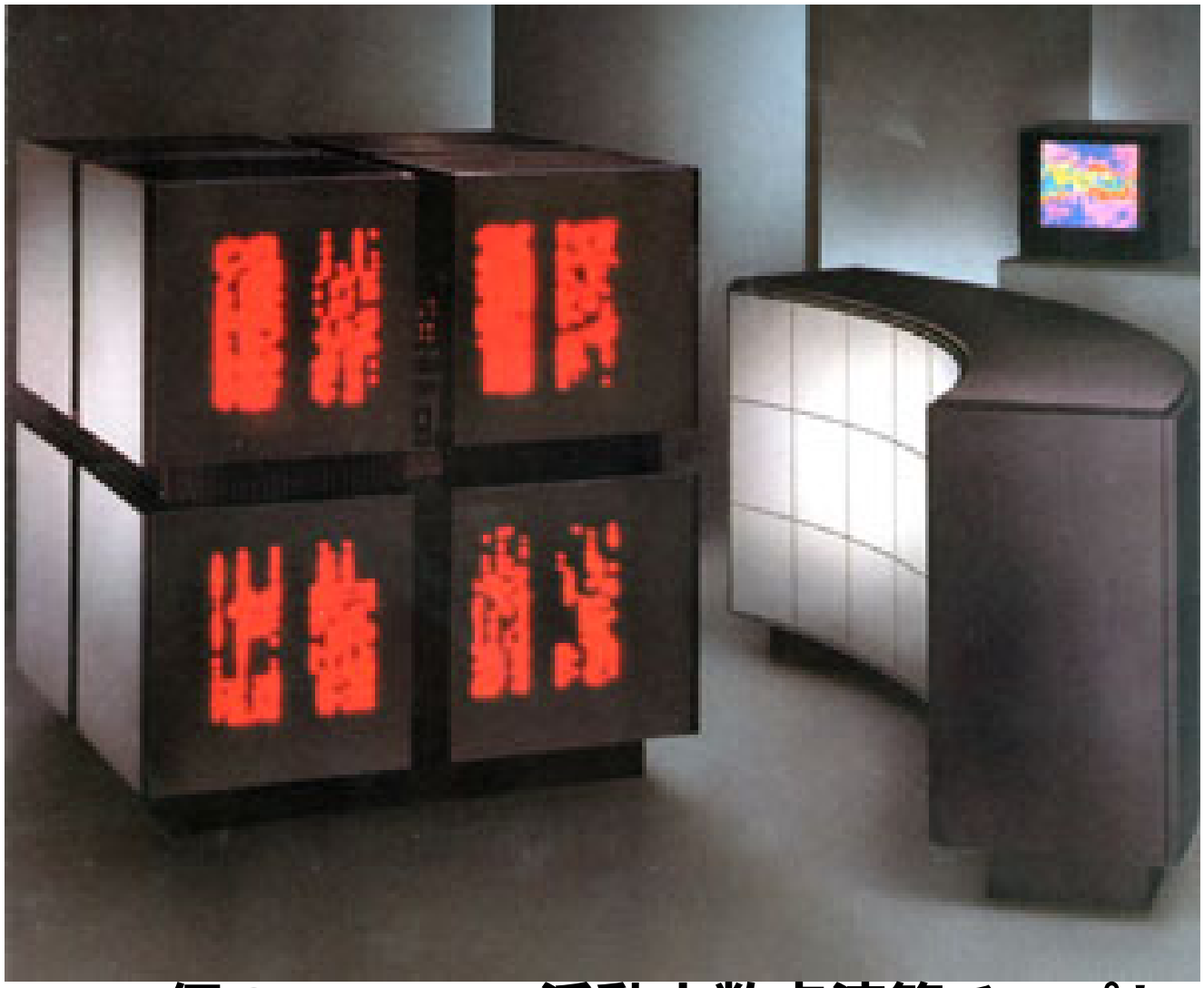
超並列SIMD計算機

— 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 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。

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

FS始まったあと

- 3FSはそれなりに進んできた
- 概算要求を来年度開発開始予定で進めている。来年度30億、総額1200億

公式資料1
公式資料2

大体どんな感じか

- ベクトルはさすがに落ちる方向
- スカラー: 「京」の後継の次
- 「加速部」: 我々が検討しているもの。1024 チップくらいまで独自ネットワークでつなげる (これで 16PF くらい)
- 加速部はスカラー (「汎用部」) に、GPGPU 的にもつなげる
- 2019 年度に 1EF くらいを目標

予想される批判

- 汎用部
  - Intel に比べてどこがいいの？
  - 売れる見込みないんじゃないの？国の予算そんなのに使うの？
- 加速部
  - 日本で作ったことないものつくるの？
  - NVIDIA、Intel に比べてどこがいいの？

どうやってよくするか？(加速部)

- 基本的思想としては、制御プロセッサを減らして大規模SIMDにすることで電力削減、通信レイテンシ減少を図る
- オンチップメモリをメインメモリとすることで容量増大、高速化、効率的利用を図る
- アプリケーション開発と並行して進めることで、アーキテクチャのボトルネックが生じないようにしていく(これが結構重要)

アプリケーション、ターゲットサイエンス

- そもそもなんのために作るのか？
- 汎用なので、「なんでもできる速い計算機」でいいのかもしれないが、そうはいっても具体的にアプリケーションを決めないと設計できない
- 汎用といっても、それでどんなことをやってどんな成果がでそうか、という作文は必要。

2011年度のアプリケーション作業部会を引き継いで、「アプリケーションFS」が発足、「計算科学ロードマップ」を作成中

というわけでここからが本題だったり

- 「分野連携による新しい科学の創出」の中に「計算惑星科学」が
- 計算惑星科学の研究拠点を立ち上げる機会にできるかも
- 加速部使ってサイエンスを、というのをよろしくお願いいたします、みたいな

おまけ:前回のキーポイント：概念構築のためのなんとか

- 概念構築といいつつすでに枠が決まっていた
  - HPL 10PF
  - 電力30MW以下
  - アクセラレータ付けるならやはり10PF
  - この範囲でアプリケーションの性能を上げること
- その結果、各社の提案が収斂した

以下、2012/6 に公開になった当時の会議資料から

(2012/6 公開)

秘 回収資料

資料2-2

次世代スーパーコンピュータの概念設計について 続き)

平成19年3月27日

理化学研究所
次世代スーパーコンピュータ開発実施本部

(2012/6 公開)

アーキテクチャ案の概要（汎用システム）【6月末時点】

- 基本要求仕様（性能評価の基準とするシステム構成）
  - 理論ピーク性能：10PFLOPS
  - 総メモリ容量：2.5ペタバイト

アーキテクチャ案	NEC	日立	富士通	筑波大学
コア数 （コア：1演算プロセッサ）	中並列 10万以下	高並列 10～50万	超並列 50～100万	
計算ノード数	～5万		10～15万	
高速演算機構	ベクトル		SIMD	
消費電力（本体のみ）	20-30 MW	10-20 MW	20-30 MW	10-20 MW
設置面積	3000 ㎡以上	1000-2000 ㎡	3000 ㎡以上	1000-2000 ㎡

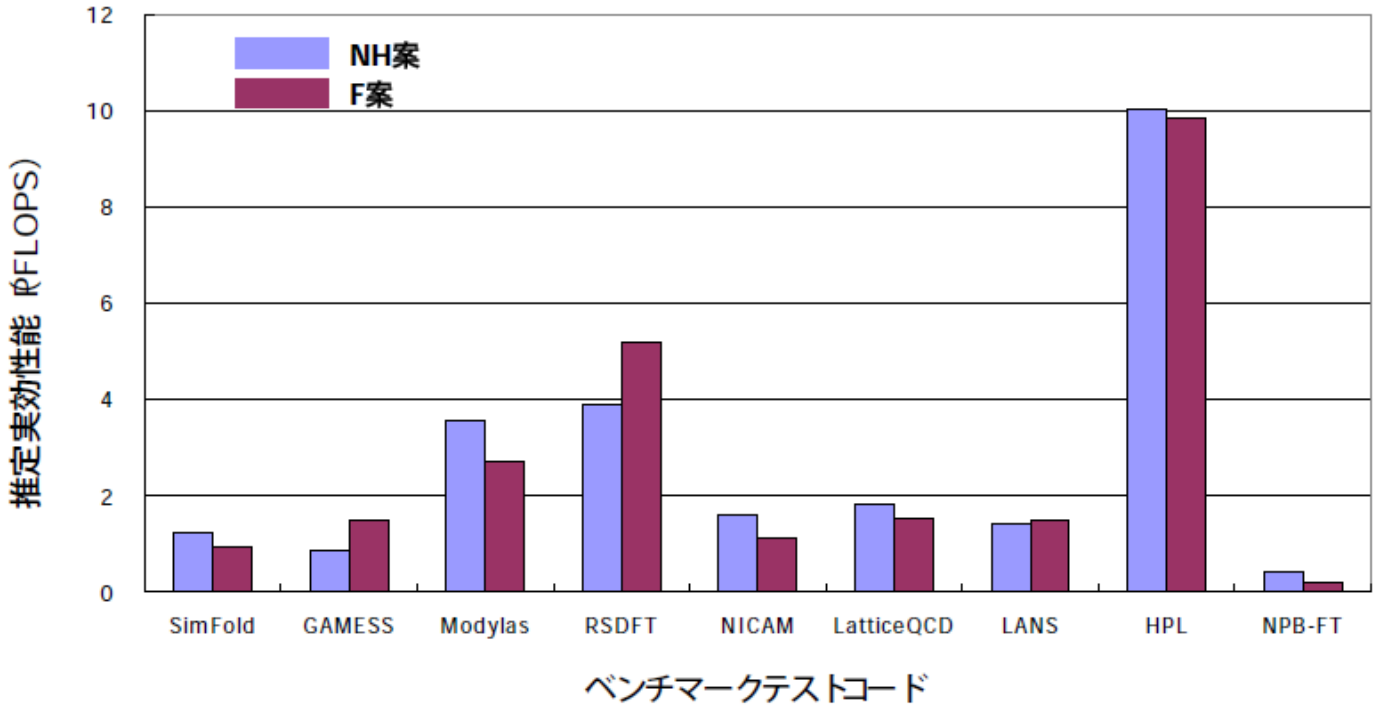
(2012/6 公開)

提案システムの演算部性能の比較

			NH案	F案
演算コア	動作周波数（GHz）		2	
	演算性能（GFLOPS）		64	16
	演算加速機構 （演算器数）		ベクトル型 （16: 2FMA x 8VPP）	SIMD型 （4FMA）
	レジスタファイル		ベクトルレジスタ 256要素×64本	スカラレジスタ 128本
CPUチップ （計算 ノード）	演算性能（GFLOPS）		256	128
	演算コア数		4	8
	メモリバンド幅 （Byte/Flop）		0.5	
	L2 キャッシュ	容量（MB）	8	6
		Byte/Flop	4	2
		特殊機構	選択的登録機構	ライン・ロック機構

(2012/6 公開)

ベンチマーク・テストによる性能予測 (詳細9本)



- ターゲット・アプリケーションから7本のベンチマーク・テスト, 及びHPL, NPB-FTについて, 実効性能を推定.
- いずれのベンチマーク・テストもほぼ同等の性能.

NH/F の比較

- ベクトル・スカラーで違うはずだったが、アーキテクチャパラメータはほとんど同じものに収斂
- さらに消費電力等も収斂
- さらにベンチマーク性能も収斂

おまけ：アクセラレータについては？

アーキテクチャ案の概要（アクセラレータ）（6月末時点）

- 基本要求仕様
  - アクセラレータ部の理論ピーク性能：10PFLOPS
  - 汎用サーバ（ホスト）のI/Oインターフェースに接続
  - ホストより指定された演算処理をアクセラレータで実行し，結果をホストに格納

アーキテクチャ案		国立天文台	東京大学
アクセラレータ	アーキテクチャ	SIMD型 プロセッサアレイ	
	プロセッサチップ数	約 15,000	約 20,000
	ボード数	4,000	2,500
ホストサーバ数		2,000	2,500
アクセラレータ部 消費電力		-10 MW ※平成24年6月公開時の注意書き 消費電力については、10MW以下、10-20MW、20-30MWの範囲でまとめたもの。提案は、ホスト部を除いて、1.7MWであった。	-10 MW ※平成24年6月公開時の注意書き 消費電力については、10MW以下、10-20MW、20-30MWの範囲でまとめたもの。提案は、ホスト部を除いて、0.68MW（案1）、0.88MW（案2）であった。

(2012/6 公開)

概念設計評価時の判断

- アクセラレータ案の説明資料には消費電力「-10MW」という謎の表現が。議事録には「10MW 以下くらい」とある。
- 実際の提案の数字は公開時注意書きにあるように 0.88-1.7MW

アクセラレータを採用しなかった理由

公式の説明

2者のシステム構成により、目標性能達成の見込みが確認できたため、アクセラレータの採用は考慮しない

- 目標 = LINPACK 10PF + HPCC 4 種1位
- アプリケーションのことはすっかり忘れられた、、、

教訓的なもの

- プロジェクトは (実現可能なら) 設定した目標通りのものが実現される
- 「プログラムは思った通りではなく、書いた通りに動く」と同じ
- なので、Top500 1 位とか HPCC 1 位とかを目標にしてアプリケーションのことを忘れられては困る
- 目標をちゃんと決めよう