流線形計算機 その2 — 実践篇 (+ 時間があまりそうなので: 新ベンチマークの提案)

牧野淳一郎
理化学研究所 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

話の構成

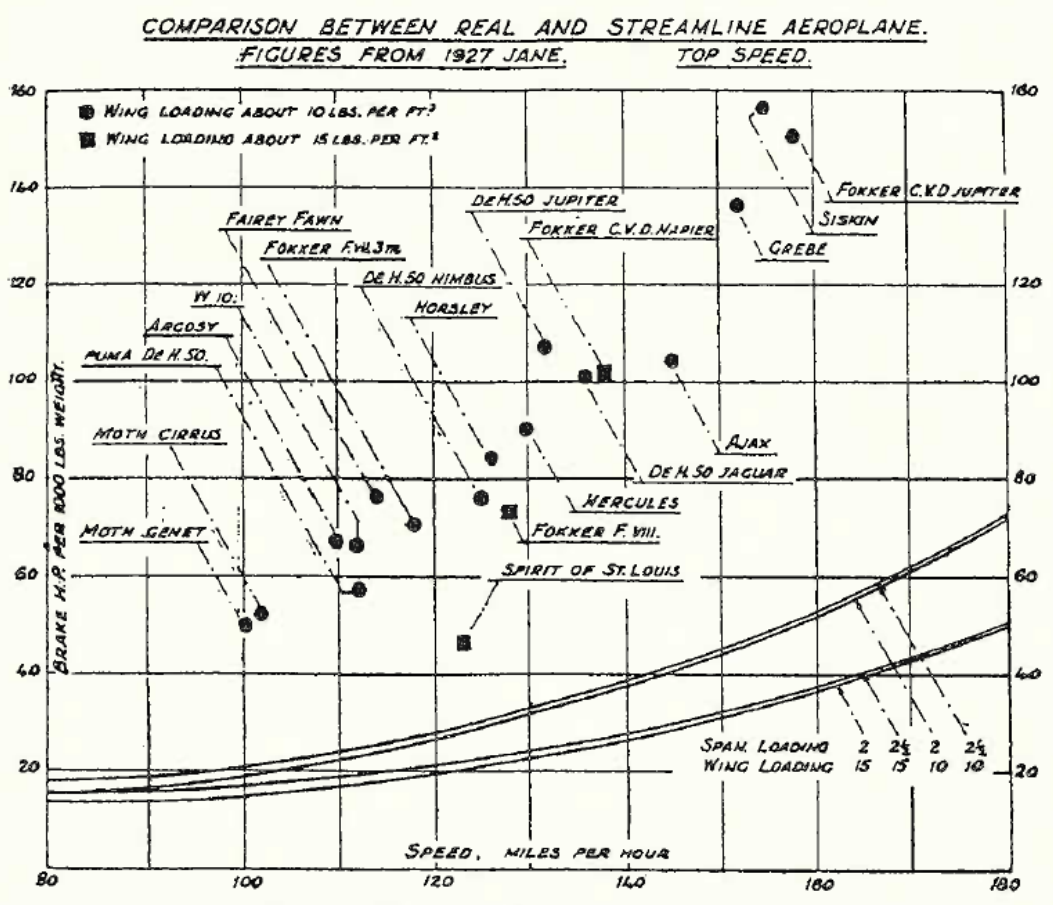
- 前回のお話のおさらい
  - 「流線形」とは？
  - モデルとして:「流線形航空機」
  - 計算機にとって「流線形」とは何か
  - 個別アプリケーションに対して
  - 行列乗算と深層学習
  - まとめ
- (あとベンチマークの話)
- まとめ

飛行機における流線形

問題: 空気抵抗をどこまで減らすことができるか
(原理的な) 解答:
減らせるものと減らせないものがある。

- 抗力
  - 誘導抗力 — 有限幅の翼ではゼロにはならない
  - 有害抗力
    - 圧力抗力 — 原理的にはどこまでも減らせる
    - 摩擦抗力 — 表面積で決まる限界あり

定量化した結果



横軸:速度

縦軸:重量あたり馬力

下の線:理想値。翼面荷重と翼幅荷重が違う４種

点は実機。一番下(良い)のはリンドバーグの Spirit of St.Louis

Spilit of St. Louis

流線形化のため操縦席から正面には窓がない。
良いといっても理想の3倍の抵抗
エンジンカウリング、引っ込み脚、片持ち翼、、、

現代の航空機

左: グライダー 右:ボーイング 787
実用機の B787 も随分スマート、理想に近いものになっている。

では、計算機にとって流線形とは何か

- (少なくとも 1929 年当時の飛行機にとって) 燃料コストは主要なもの
- 計算機にとっても、やはりエネルギーコストであろう。
- 特に最近の計算機では実際にハードウェア製造コストより電気代が大きくなりつつあるので、「演算あたりの必要パワー」を最小化する、というのが本質的に重要な、計算能力の進歩をそのまま決める要因になっている。

つまり

ある計算処理を実行するのに必要な理論上の最小必要エネルギーがあり、そのエネルギーで実際に実行できるのが「流線形計算機」**Streamline Computer** である

「流線形計算機」に反論と再反論

1. 半導体テクノロジーが変われば必要エネルギーは変わるから意味がない。

   残念ながらもう CMOS スケーリングは終わった

2. 仮に「理想の計算機」があるとしても、それはアプリケーション毎に違うものであり、だからといってアプリケーション毎に計算機を作ることができるわけでもないんだから絶対実現できない理想であり、意味がない。

   CMOS スケーリング終わったので、実は専用計算機を問題毎に作る解もある。何年でも使えるから。もちろん、なるべく汎用、という考え方もある

3. アプリケーションで使われるアルゴリズムは日夜進歩するので、「アプリケーション毎に理想の計算機が決まる」というのがそもそも間違いである。

   まあそれが本当ならそもそも GRAPE とかない、、、

消費電力の分類

飛行機における

- 抗力
  - 誘導抗力 — 有限幅の翼ではゼロにはならない
  - 有害抗力
    - 圧力抗力 — 原理的にはどこまでも減らせる
    - 摩擦抗力 — 表面積で決まる限界あり

にあたるものが必要。例えばこんな感じ。

- エネルギー消費
  - 演算組合せ回路
    - 動的
    - 静的 (リークとか)
  - 記憶素子 (メモリ、レジスタ)
  - データ移動 (クロック、ラッチ、配線)
  - 制御回路 (命令に関する全て)

原理的にゼロにできないものは演算組合せ回路の動的消費電力のみ

個別アプリケーションに対する「流線形計算機」

大体以下の4種くらいを考えればいい (物理というよりデータアクセスのパターンとして)

1. 規則格子 (陽解法差分)
2. 粒子
3. 密行列
4. 不規則格子

粒子法と密行列はつくればできるのはほぼ自明。計算精度をどうしたいかは微妙 (特に密行列)。
というわけで、今日の残りはまずは行列専用計算機の話。

行列乗算専用回路

- 意外に (でもない？) 重要なアプリケーションがある
  - 量子化学計算一般
  - 深層学習
  - **FEM (DDM** とか使うと結局、、、)
  - **stiff** な系がある話 (東大の心臓シミュレーションとか)
- **28nm** で、倍精度乗算器+加算器だけなら **0.02** 平方ミリくらい。**300** 平方ミリで **1** 万ペア。**500MHz** で回して **10TF**、**100-150GF/W** くらいまでいけるはず。単精度ならさらに **3-4** 倍。
- **28nm** の **GPU** の **10** 倍くらいの電力性能。
- 多分 **2030** 年くらいまで使える。
- なんかもうこれだけでいい気がしてきた。

深層学習と行列乗算

- 深層学習は多段ニューラルネット。ということは
  - 計算の $\mathbf{99.9\%}$ くらいが行列乗算 (BP 学習でも)
  - GPU が使われてるのは結局単精度行列乗算専用マシンとして
  - NVIDIA が倍精度なしの Maxwell も高く売るのを始めたくらいには需要がある。
- 学習は多段ニューラルネットに対して BP が最適かどうかは (私には) 良くわからない。まあでも多分行列乗算ができればいいような方法にはなる。

深層学習専用にもうちょっと頑張ってみると、、、

数千 × 数千 の行列積を「可能な限り短い TAT で」やりたい。

- BP 学習はあんまり上手く並列化できてない。
- 1つのニューラルネットを多数の GPU でとかは全然無理
- チップ間を高速ネットワークでつなげばなんとかならないか？

例

- 1チップ 30TF (単精度)
- 行列サイズ 2048
- 36 チップを 6×6 の2次元グリッドに
- 行列乗算が 16 マイクロ秒で終わる。
- 1方向の転送速度が 16MB を 16 マイクロ秒で、 1TB/s。6 でわって 170GB/s
- すぐ隣のチップとならできるかも。5GHz x 300 本。
- GPU より 100 倍速いとかいって 5000 万円くらいで売れるかも。

チップ間通信

- $\mathbf{170GB/s}$ が絶対にできないとはいわないが、もうちょっと下げたい。
- データ圧縮を考えるのは重要。音声や動画データでは時間相関があるので、予測演算が可能なはず。
- $\mathbf{DNN}$ の中間層でそんなことができるかどうかは自明ではないが、研究の価値はあるかも。

今さら専用回路？今こそ専用回路？

CMOS スケーリング終焉のインパクト：おそらく誰もまだよくわかっていない。

- 半導体プロセスが微細化しても電力がちょっとしか下がらない (デザインルールに比例)。
- 半導体プロセスが微細化すると初期費用もトランジスタあたりの価格も「上がる」。

これらの意味:

- アーキテクチャとして消費電力を下げること (流線形化) が、先端プロセスを使うことより重要。
- アーキテクチャを改良することしか計算機の性能向上の方法がない。
- 専用機が容易なことでは汎用機に追いつかれなくなる。

これからは実は専用機の時代かも？

まとめ

- 「流線形計算機」とは、アプリケーションにおける演算の組み合わせ論理を駆動するのに必要な電力に限りなく近い消費電力で実際の計算を行える計算機である。
- 普通の汎用計算機では2桁くらい余計に電気を喰っている。
- 現実のアプリケーションは結構、密行列乗算、粒子、規則格子、不規則格子くらいのバリエーションしかない。これくらいについて「流線形計算機」に近いものはつくれるのではないか？
- 1種で複数の機能 (プログラム可能) と単機能のどちらがいいか？
- 密行列乗算を例に考えると、単精度で $28\mathrm{nm}$ プロセスでチップレベル $300\mathrm{GF/W}$ くらい、300平方ミリくらいのチップで $30\mathrm{TF}$ くらいはつくれそう。

まとめ(続き)

- さらに、深層学習でありがちな問題サイズで **1PF** くらいのものなら基板１枚(日立サイズを想定、、、)でできるかも。

おまけ: 新ベンチマーク提案

- どういうベンチマークを提案するか？
  - **HPL** を制限時間付き (**10**秒、**300**秒、**10,000**秒とか) でやる
- 何故そうするか？
  - 計算機の複数の側面を測定する
  - 特に大規模並列機がもうちょっとまともなものになるようにしむける

話の順番

- **HPL** の現状と問題
- **HPCG** はどうか？
- **HPL** の性能特性
- 我々は何を知りたいか
- 問題サイズ・実行効率・実行時間
- 「新ベンチマーク」

HPLはどういうものか

- **High Performance Linpack: 20**年以上 **Top 500** のランキングに用いられる「ベンチマーク」
- 仕様: 直接法で密行列を解く。演算量が減る最適化 (ストラッセンのアルゴリズムとか) は×。それ以外は何をやってもいい。
- 問題サイズもなんでもいい (昔の「**Lipack** ベンチマーク」は問題サイズが **100** と **1000** だった)

HPL のメリット

- プログラムソース、アルゴリズム、問題サイズの指定がない：アルゴリズム、ハードウェアの進化にルールを変えないで対応できる。
- 規模が非常に違うマシン間でも一応意味がある比較ができる。

これらは非常に重要。これらのために 20 年以上にわたって使われつづけてきた。

HPLのデメリット

- 素晴らしく最適化が進んだこと、問題サイズに制限がないことのため、HPL でそれなりの効率がでるために必要な主記憶バンド幅やネットワークバンド幅は非常に小さくなった
- つまり、演算に特化していて足回りが極度に弱いマシンでも問題サイズが大きければ良い数値がでる
- マシンが大きくなると、ちゃんと性能がでるための問題サイズが大きくなり、計算時間も非常に長くなる。「京」では約30時間。

この、実アプリケーションとの乖離は近年問題。HPL ではピークの 50-70% でても「私のプログラム」では 5% とかいうのが普通になっている。

ドンガラ先生その他の対応

「**HPCG**」

- 有限要素法に対する **CG** 法のマルチカラー反復のベンチマーク。
- グリッドは規則格子だが、間接アクセスをなくすようなプログラムの書換えは禁止。

現状の数値は「京」で **5%**、**SX-ACE** で **10%** くらいだったような。

では HPCG でいいか？

良いところはある。 HPL の大きな問題を確かに解決している。

- メモリシステム、特に gather operation の速度が最重要。演算性能だけでは決まらない。
- 実行時間短い。

HPLの悪いところ

- HPL と違ってアルゴリズム (従ってメモリアクセスパターン) が固定されている。将来的なアーキテクチャ、アルゴリズムの進歩の余地がない。
- このため、アプリケーションが全体としてメモリアクセスを減らす方向に進化すると、 HPL とは逆の意味で現実と乖離したベンチマークになる。
- おそらく既になっている。
- 理論的には、コード書換えとコンパイラの改良によってルールを破ることなく間接アクセスを回避することが可能。ベンチマークの意味をなしていない。(多分。まだ詳細なルールチェックはできてない)

際どい最適化

- ソースプログラムから間接アクセスを残したまま、実際のアクセスはすべて連続アクセスになるようにデータ構造、ループ構造を書換えることは可能。
- そうすると、以下のような「最適化」が理論的には可能
  - 間接アクセスがあるベクトル化可能なループについて、間接アクセスに見えても実はリストベクトルが連続値になっているかどうかをループにはいる前にチェック
  - 連続値だったらそれ用に間接アクセスを消去したループを実行。
- **gather** 命令の実装レベルでの対応もありえる。リストベクトルが完全に連続アドレスだったら 通常のロードに切換えるだけ。

というわけで

- 個人的には **HPCG** は危険ではないかと思う。
- とはいえ現状の **HPL** だけでは話にならない。

では何が必要か？

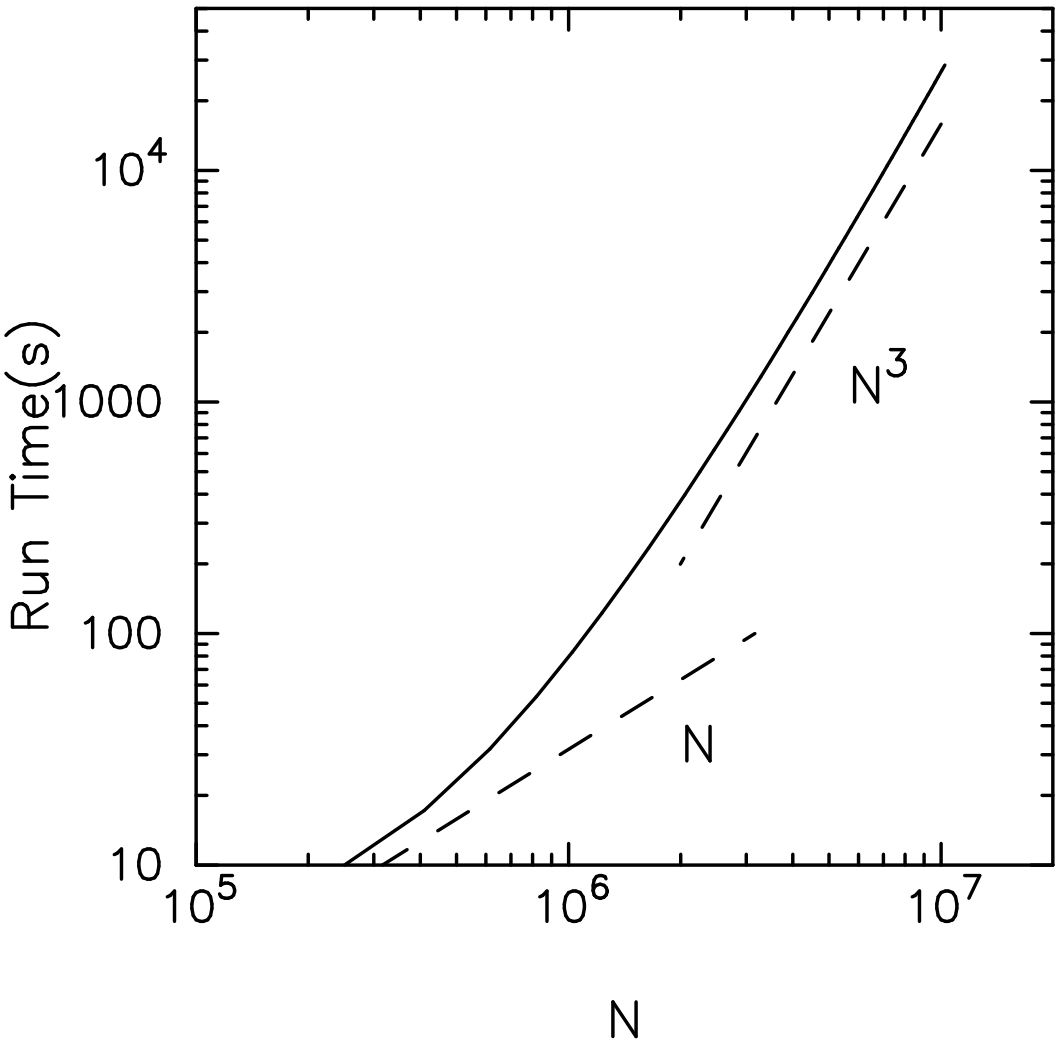
- 現在の **HPL** は少なくとも演算性能は測っているし、ネットワークバンド幅も結構測っている (弱いマシンもある)。
- 明らかに測定できてない重要なパラメータ：ネットワークレイテンシ。
- 時間がかかりすぎるのはやはり問題。

ではどうするべきか？

対応案

- 時間がかかりすぎるなら、時間制限をつければいいのではないか？
- レイテンシを測るなら、「非常に短い時間」に計算が終わるサイズで実行すればいいのではないか？

どんな感じになるはずか？



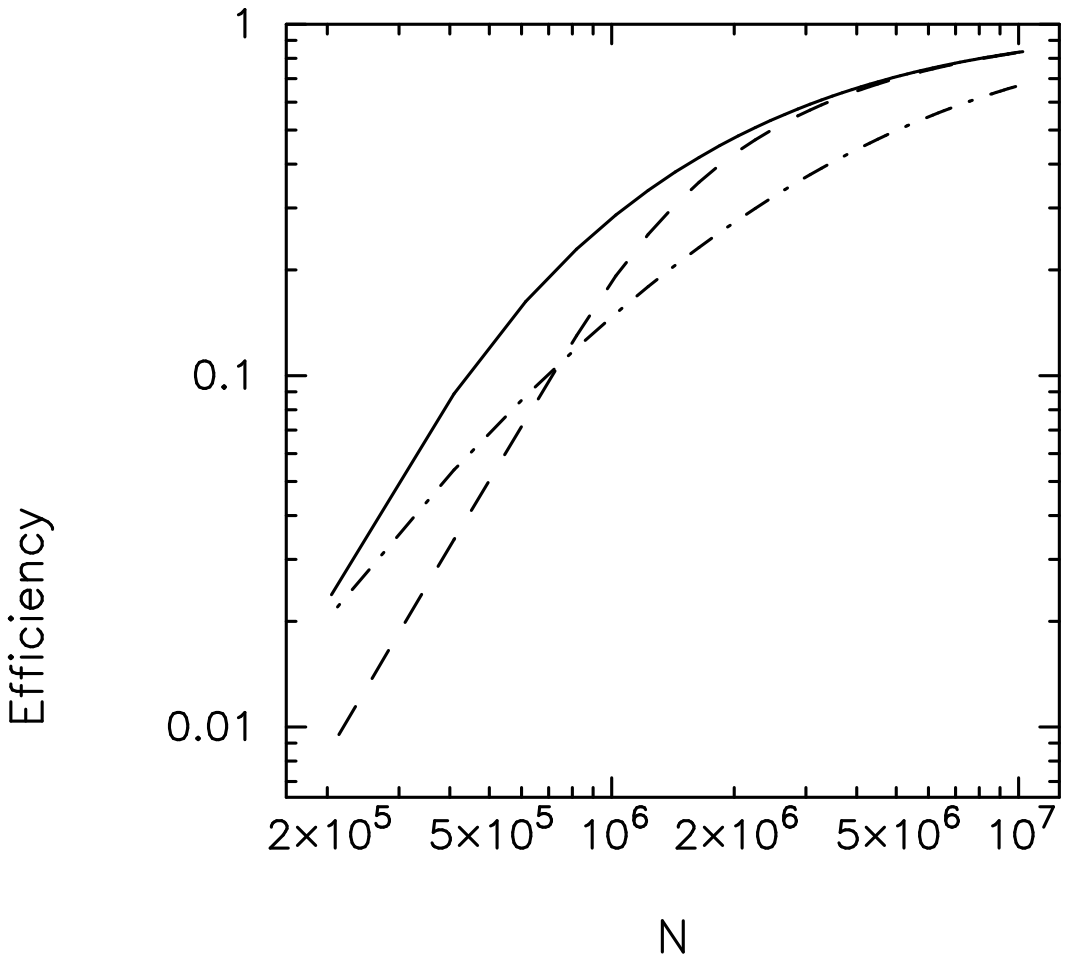
仮想的マシン (ノード性能 **3TF**、**1** 万ノード) の **HPL** 問題サイズと実行時間

牧野が適当に作った性能モデル (**netlib** の **HPL** ページにあるものの改良版)

ネットワークは実効 **30Gbps**

ノード間レイテンシは **5** マイクロ秒

仮想マシンの実行効率



サイズ小さい時にはレイテンシリミット
途中はネットワークバンド幅リミット
最終的には演算速度リミット
破線は大レイテンシ、一点鎖線は低バンド幅を仮定

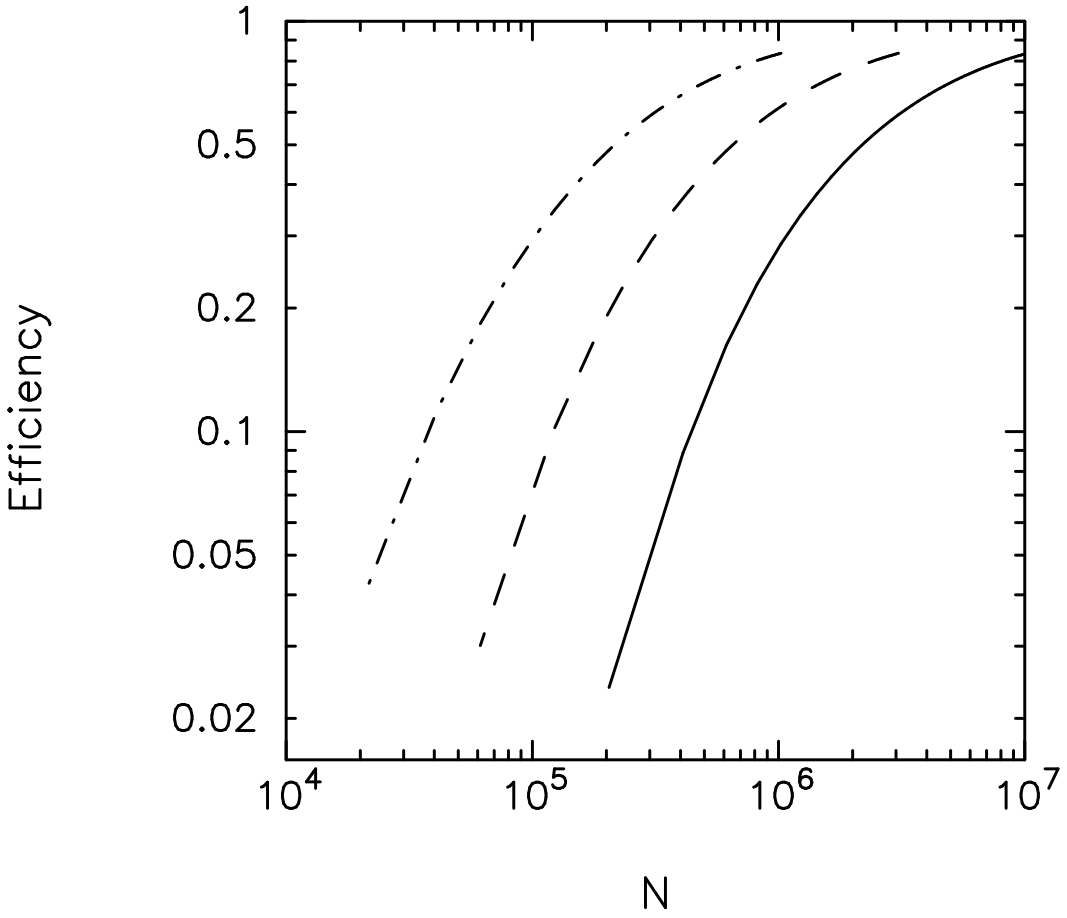
つまり

- (おそらく大抵のマシンでは) **3**つの違う問題サイズで **HPL** の性能を測定すれば大体どういう代物かわかる。
- あまり大きな問題サイズを許すと無意味に高い効率になる。

とはいえ、問題点:

どれくらいの問題サイズが適当かはマシンサイズによるのではないか？

マシンサイズと実行効率

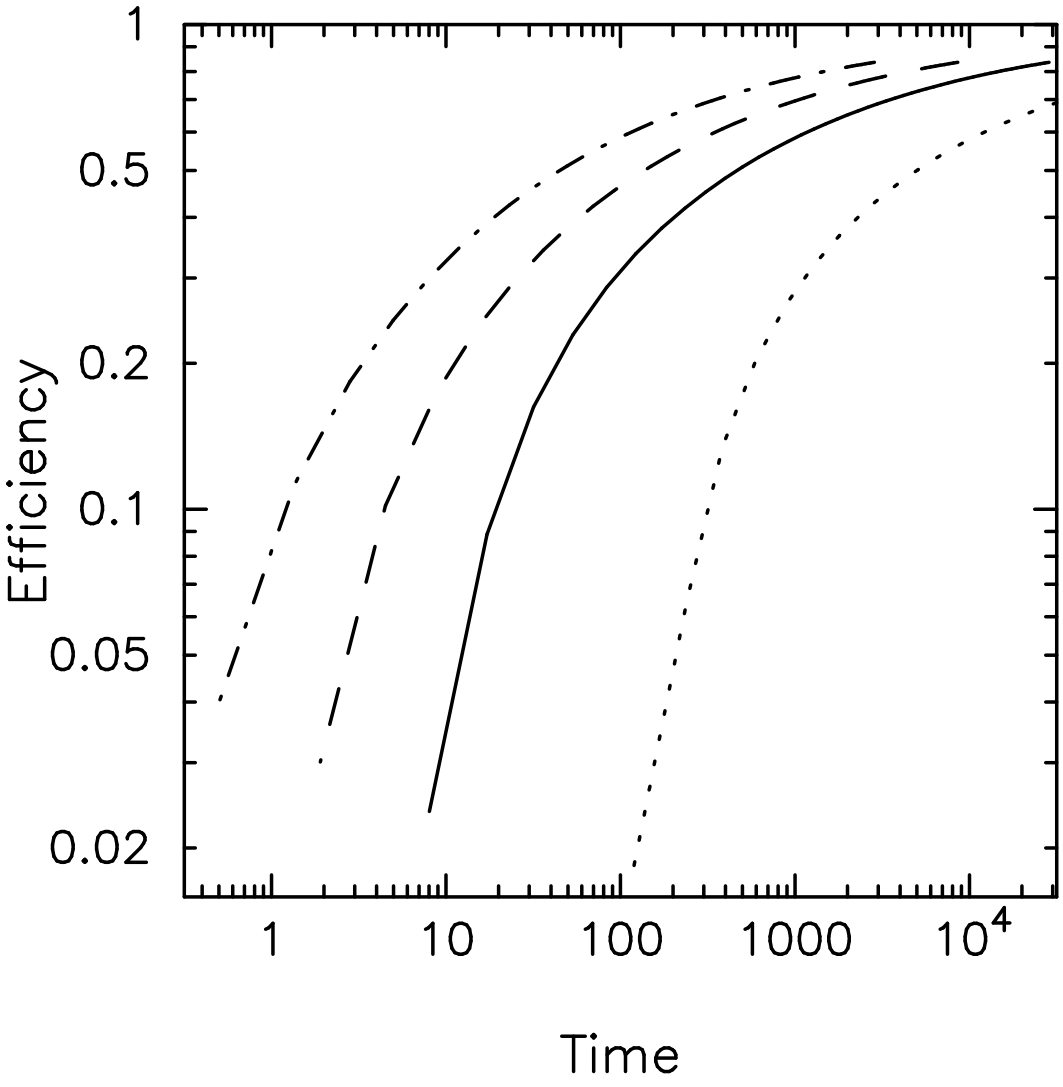


サイズ 1/10, 1/100 のマシン

確かに、性能が下がる問題サイズが全然違う。あんまり具合よくない

ではどうするか？

実行時間を合わせる



サイズを合わせるほどひどくない
実行時間が短いところは大きなシステムは効率悪いが、それはむしろ解決するべき問題 (レイテンシを下げるべき) ではないか？

実行時間合わせる (続き)

- レイテンシ 1 桁下げられればかなり効率上がる。但しネットワークバンド幅も見える
- ちょっと意外だったこと:エクサスケールでも 1 万秒もあれば効率はあんまり下がらない
- 言い換えると、30 時間とかの実行時間は、おそらく効率を例えば 85% くらいから 90% 以上にあげるためだけにやられている。見栄はわかるが本当に意味があるかと言われると？

順位付けは？

どうしても1位から番号つけたい、という人はいるであろう。いくつか考え方はある。

- 3つの数値の幾何平均をとる。
  - ネットワークが同じ程度なら、大きなマシンは時間短い時でも遅くはならないので、それほど影響ないはず。
  - ネットワークが非常に良いマシン(特に低レイテンシマシン)は良い評価になる。これはまあ あってもいいのではないか
- それぞれのクラスでの順位自体の幾何平均や調和平均をとる。こっちのほうが平均して速いマシンに有利？

「新ベンチマーク提案」のまとめ

- 現状の **HPL** はあんまりよくない。
- **HPCG** は色々な問題がある。あんまり勧められない。
- **HPL** の **10** 秒、**300** 秒、**10000** 秒以下の実行時間での性能を測定すると、現在から将来の典型的なマシンで、ネットワークレイテンシ、ネットワークバンド幅、演算性能の実力が見える。
- レイテンシがはいることは特に大規模マシンでの強スケーリング性能にとって重要。
- **3** つの数値から **1** つの順位付けをする方向は色々ありえる。検討必要。