# 第4回
# 2012/5/14
# 「コンピュータの性能はどう考えるの?」

# 性能に関して

- 計測し、報告し、指標をまとめるには?
- アーキテクチャのデザインの賢い選択のための指標
- 宣伝にまどわされず、アーキテクチャの善し悪しの本質を見透かす
- 「どうしてこのようなアーキテクチャの構成なのか?」という本質を見抜く手助けともなる

  **Q:**
  *さまざまなプログラムに関して、どうしてあるハードウェアは他のより速いのであろう?*

  *性能のどの点がハードウェアが本質的に影響しているのであろう?*
  *(e.g., あるマシンが遅いとき、新しいマシンが必要なのか、それともOSを新しくすべきか?)*

  *マシンのISAがどのように性能に効いてくるか?*

- *単に実行効率以外の性能メトリックはあるか?*

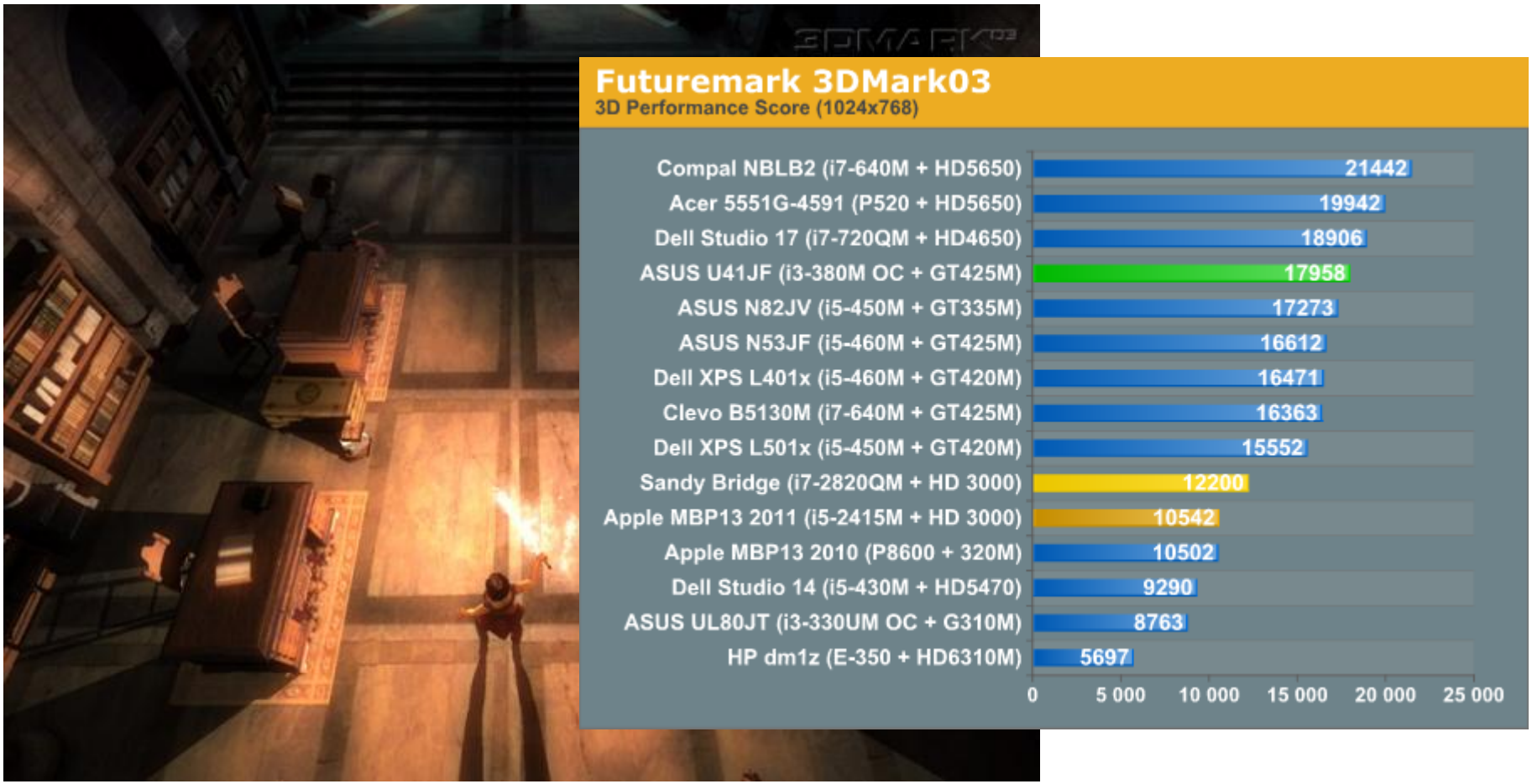
# この飛行機のうち、どれが一番「性能」が良いか？

| 飛行機 | 定員 | 航続距離 (km) | 速度 (Mach) | 燃料消費(l/h) |
|---|---|---|---|---|
| Boeing 737-500 | 132 | 4444 | 0.74 | 2933l/h |
| Boeing 747-400 | 416 | 13,450 | 0.89 | 12,788 l/h |
| BAC/Sud Concorde | 120 | 7250 | 2.2 | 30,000l/h |
| Lockheed Martin F16 | 1 | 3900 | 2.2 | 3000l/h |

- **Concord, F16は747と比較して、どの程度速い?**
- **747 は 737 と比較して、どれだけ乗客数を運べる?**
- **747は、737と比較して、何倍の距離を飛べる?**
- **単位時間あたり、もっとも乗客を長い距離運べるのは?**
- **最も燃費効率よく乗客を運べるのは?**


# 計算機の性能は重要

- 「グラフィックスベンチマーク」→グラフィックスカードの購買のベース

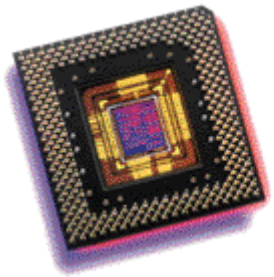


# 計算機における指標: 時間と時間と時間

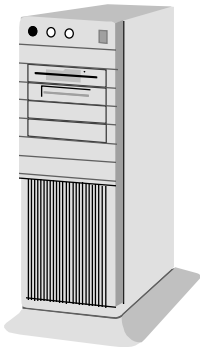
- **Response Time (latency)**
  - — 私のジョブが走るのにどの程度時間がかかる?
  - — この計算はいつおわるの?
  - — 私のデータベースの問い合わせはいつまで待てば良い?
- **Throughput**
  - — あるマシンが一度に走らせられるジョブの数は?
  - — 単位時間あたりのジョブの実行数は?
  - — どのような速さでジョブが処理されている?
- その他種々の「指標」

*新しいプロセッサでマシンをアップグレードしたら、どの指標が良くなる?*

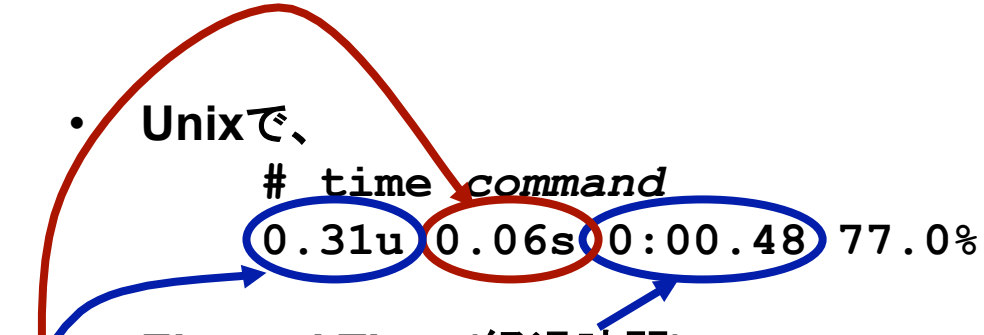
*もし新しいマシンを研究室に買ったら、どの指標が良くなる?*




対



# 実行時間


• Unixで、
# time command
0.31u 0.06s 0:00.48 77.0%
Elapsed Time (経過時間)
– 実際のプログラムの実行の経過時間
= 全ての時間のトータル (CPUの実行時間+ディスクのアクセス待ち/読み書き時間、OS内部の時間、その他入出力、他のユーザの実行、etc.)
– 有用な指標であるが、精密な比較のためにはあまり役に立たない。
• CPU time
– I/Oや他のプログラムの実行時間は数えない
– system time (OSの内部の実行時間)と、
– user time(OSの外部でのユーザプログラム自身の実行時間)
• われわれの注目すべきもの: user CPU time

# 一般的な性能の定義

- マシン X で動作しているプログラムに対し、,

  $性能_X = 1 / 実行時間_X$

- 「X は Yより n 倍性能が良い」

  $性能_X \ / 性能_Y = \ 実行時間_Y \ /実行時間_X = n$

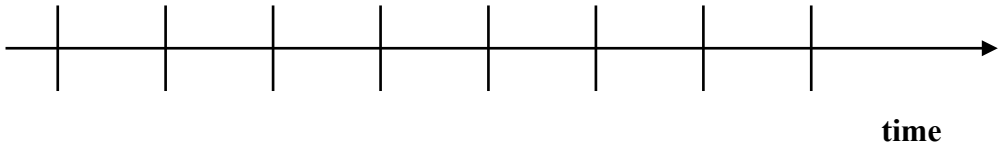
- 問題点:
  - あるマシン A はあるプログラムを20秒で実行する
  - 別なマシンBは同じプログラムを25秒で実行する
  - Q: 果たして、全てのプログラムで、マシンAはマシンBより1.25倍性能が良いと言えるであろうか?

# クロックサイクル

- 実行速度を秒数で表現するより、より精密にはクロックサイクルを用いる

$$\frac{\text{秒}}{\text{プログラム}} = \frac{\text{サイクル}}{\text{プログラム}} \times \frac{\text{秒}}{\text{サイクル}}$$

- クロックの「一刻み」はいつ実行を開始するか、ということをあらわす（一種の抽象化ではある)：




- サイクルタイム = クロックの刻みの間隔 = 1 / 毎秒あたりのサイクル数
- クロックレート (周波数) =毎秒あたりのサイクル数　(1 Hz. = 1 cycle/sec)

A 200 Mhz. のクロックは $\frac{1}{200\times10^{6}}\times10^{9} = 5$ ナノ秒　のサイクルタイム

# どのように性能を向上させれば良いか?

$$\frac{\text{秒}}{\text{プログラム}} = \frac{\text{サイクル}}{\text{プログラム}} \times \frac{\text{秒}}{\text{サイクル}}$$

**Q:** したがって、性能を向上させるには(他の部分が等しかったとすると)

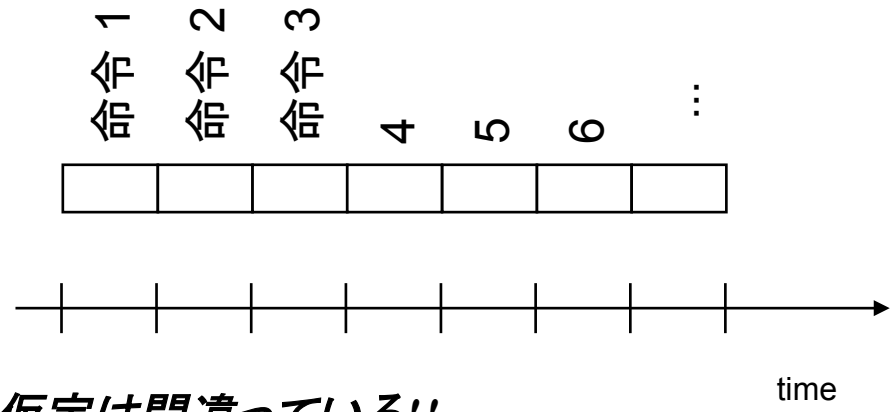
プログラムが要求するサイクル数を_____ 、または

クロックのサイクルタイムを_____、つまり、

クロックレートを_____ 。

# プログラムの実行には何サイクルかかるのか?

- 仮定:プログラムの総サイクル数
  = プログラムで実行される命令数




*この仮定は間違っている!!*

*異なる命令は、異なるマシン上で異なる時間がかかる*

*何故か?*

*メモリアクセスはレジスタ間操作より時間がかかる*
*ジャンプがあると、時間がかかる(何故?)*
*浮動小数点命令は、時間がかかる、などなど*

# サイクルタイムによる性能比較の例

- **Q: 我々がよく実行するプログラムPは200Mhzのクロックを持つあるマシンA上で10秒で実行される。我々はデザイナーが新型のマシンBを設計するのを手伝っており、設計のターゲットとしては同じプログラムを6秒で実行できなくてはならない。デザイナーは新しいチップの技術を使えるので、クロックを上げることができるが、他のCPUの部分のデザインが影響を受け、Pを実行するには1.2倍のクロックサイクルが必要になる、と報告している。我々は、設計上クロックレートをどこまで上げれば良い、と伝えるべきか?**

# 回答

- 仮にマシンA上でプログラムP実行に必要な総サイクル数を k とすると、
  - CPU時間$_A$ = k / 周波数$_A$
  - k = 10 x 2 x $10^8$ = 2 x $10^9$ サイクル
- マシンBでのCPU時間は、
  - CPU時間$_B$ = k x 1.2 / 周波数$_B$
  - 周波数$_B$ = 1.2 x 2 x $10^9$ / 6 = 4 x $10^8$ = 400Mhz

# クロックとサイクルのまとめ

- プログラムの実行の尺度
  - ある数の命令(機械命令)
  - ある数のクロックサイクル
  - ある時間(秒)
- これらの定量的尺度を表現する用語:
  - サイクルタイム　cycle time (1サイクルあたりの時間(秒))
  - クロックレート　clock rate (1秒あたりのサイクル数)
  - CPI (cycles per instruction、1命令あたりの平均クロックサイクル)
    *浮動小数点演算を多用するアプリはCPIが高くなるかもしれない*
  - MIPS (毎秒何百万命令実行数 millions of instructions per second)
    単純な命令を用いるプログラムでは高い→良い尺度ではない
    c.f., 相対MIPS: あるマシンの速度を基準(1MIPS)とする (Vax MIPS)

# 性能を決定する尺度・指標

- あるプログラムに対しては性能は実行時間で決定されるのは先に延べた
- ほかのパラメータは性能を決定する有効な尺度になるであろうか?
  - プログラムを実行するのに必要なサイクル?
  - プログラムで実行される命令数?
  - クロックレート?
  - CPI?
  - MIPS?

- 良くある誤信: 性能をある一つのパラメータのみで評価してしまうこと
  ←本当はさまざまなパラメータがある

# CPI の例

- **Q: 同じ命令セットアーキテクチャ(ISA)の二つの異なる実装があったとしよう (例: Pentium と Pentium Pro)**

  **あるプログラムに対して,**
  **マシン A はクロックサイクルタイムが10 ns で、CPI が 2.0**
  **マシン B はクロックサイクルタイムが20 ns で、CPI が 1.2**

  **どちらのマシンがこのプログラムに対してはどの程度速い?**

- ***ヒント: もし二つのマシンが同じISAであったとすると、我々の定量的尺度のどれが同一になる? (e.g., クロックレート, CPI, 実行時間, 総命令実行数, MIPS)***

- **回答: *総*命令実行数をkとすると、それぞれの実行時間 $t_A$, $t_B$は、**
  - **$t_A = k /(10 \times 10^{-9}) \times 2.0$**
  - **$t_B = k / (20 \times 10^{-9}) \times 1.2$**
  - **性能$_A$/性能$_B$ = $t_B / t_A$ = 1/1.2**

# 命令数の例

- Q: コンパイラのデザイナがある特定の言語機能を機械語に変換するのに、あるマシンに対して二つの機械語のコードの列の選択肢があるとする。実装の性質により、命令は三種類に分けらる。それぞれのサイクル数は以下の通り:
  - クラスA: 1サイクル、クラスB: 2 サイクル、クラスC: 3 サイクル
- また、コード列は、以下の命令のクラスを含んでいるとする
  - 一番目のコード列は5命令: Aが2、Bが1、Cが2
  - 二番目のコード列は6命令: Aが4、Bが1、Cが1
- この時、
  - それぞれのコード列のCPIを求めよ
  - どちらのコード列がどれだけ速いか?

# MIPS の例

- Q: 二つのコンパイラが100Mhzのマシン上でテストされており、大規模アプリケーション用のコードを出力する
  - 命令のクロックサイクルは前問同様に、クラスA: 1サイクル、クラスB: 2サイクル、クラスC: 3 サイクル
  - 第一コンパイラのコードでは、クラスA 500万命令、クラスB 100万命令、クラスC 100万命令、が実行される
  - 第二コンパイラのコードでは、クラスA 1000万命令、クラスB 100万命令、クラスC 100万命令、が実行される

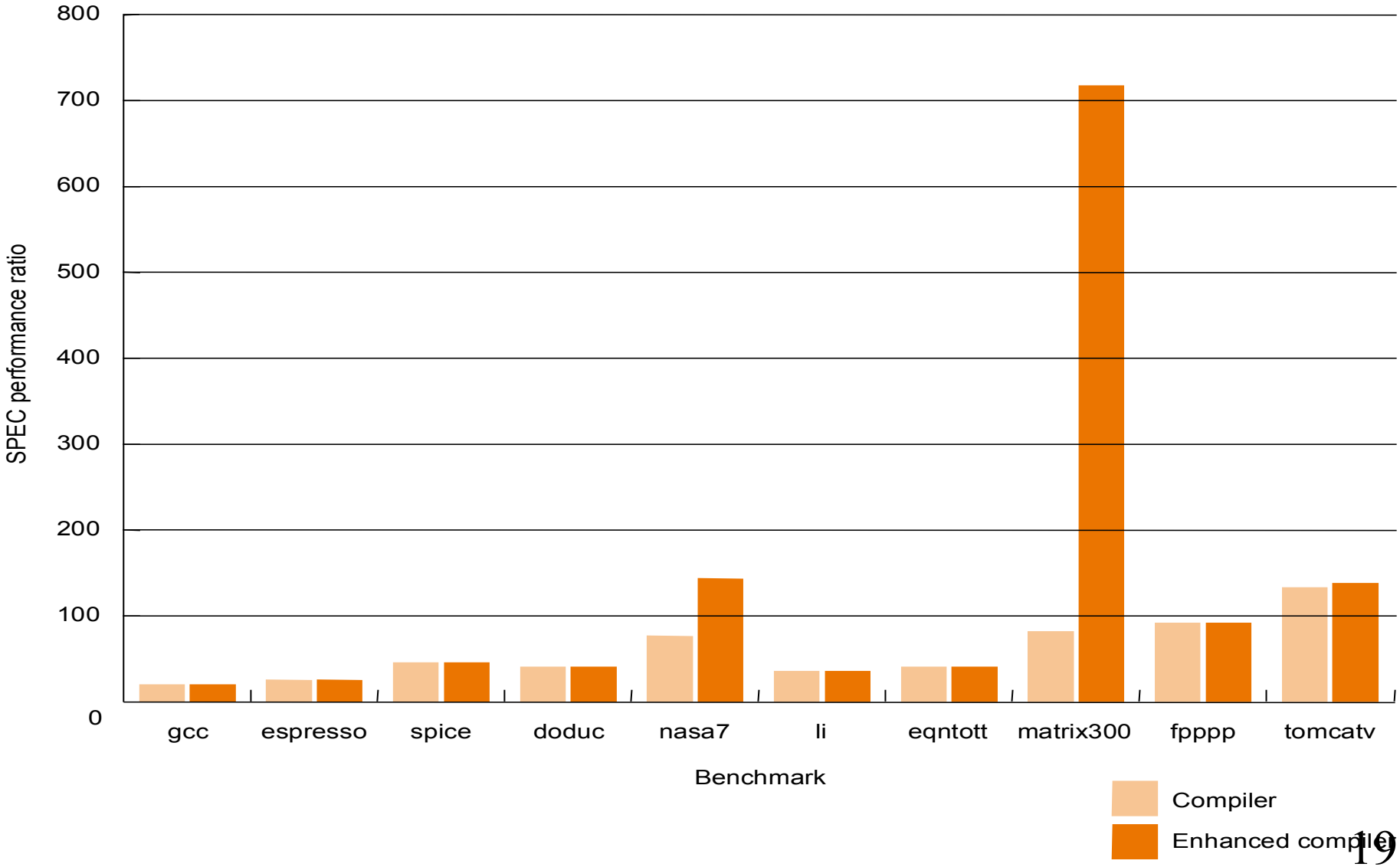
- この時、
  - MIPS値の高い方はどちらのコンパイラ?
  - 実行時間が短い(性能が高い)のはどちらのコンパイラ?

# ベンチマークに関して

- 性能は実際のアプリケーションを動かすのがベストな尺度となる
  - 期待されるワークロードに対して典型的なプログラムを用いる
  - または、一般的に典型的なクラスのアプリケーションを用いる
    e.g., コンパイラ/エディタ、科学技術計算、グラフィックス、など.
- 小さいベンチマーク (Small benchmarks)
  - アーキテクトやハードウェアデザイナにとっては扱いやすい
  - 標準化は容易: 例: Dhrystone, Whetstone, Sieve
  - 乱用される恐れ: 針小棒大な結果、チューニングが可能
    (benchmark tuning)
- SPEC (System Performance Evaluation Cooperative)
  - 多数の企業間で、本物のプログラムと入力を標準化 (数千~数万行)
  - それでもやはり乱用される可能性がある(次のスライド)
  - 性能の非常に重要な尺度。アーキテクチャだけでなく、コンパイラ技術の評価にも用いられる
  - SpecCPU (SpecINT, SpecFP) 89, 92, 95, 00
  - 現在は Spec CPU 2006
  - 現在ではさらに様々なベンチがある (SpecHPC, SpecJVM, …)
    - www.spec.orgを参照

# SPEC ’89

- ベンチマークチューニングの結果　→matrix300は以後不採用に



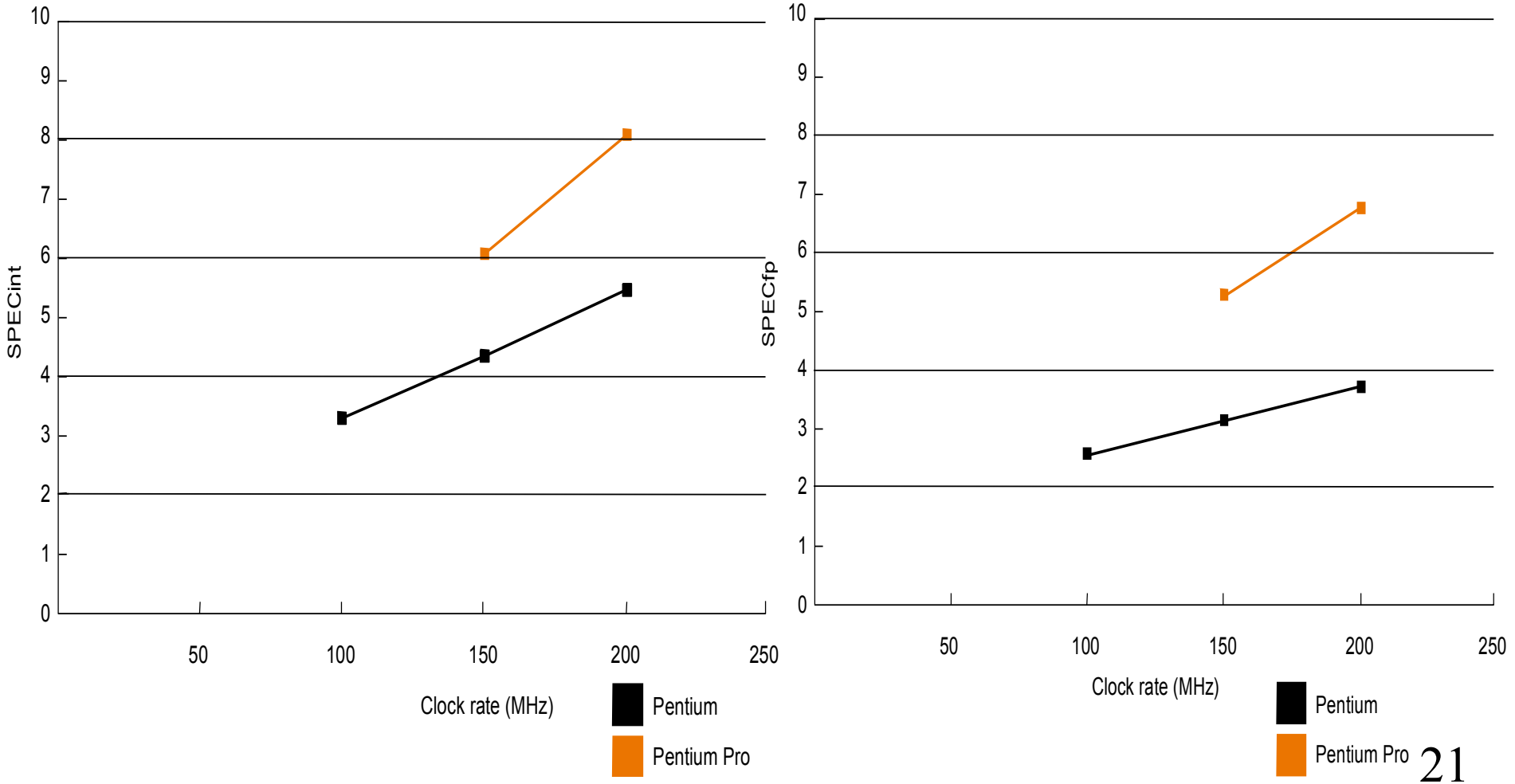
# 改良：SPEC ’95

| Benchmark | Description |
|---|---|
| go | Artificial intelligence; plays the game of Go |
| m88ksim | Motorola 88k chip simulator; runs test program |
| gcc | The Gnu C compiler generating SPARC code |
| compress | Compresses and decompresses file in memory |
| li | Lisp interpreter |
| ijpeg | Graphic compression and decompression |
| perl | Manipulates strings and prime numbers in the special-purpose programming language Perl |
| vortex | A database program |
| tomcatv | A mesh generation program |
| swim | Shallow water model with 513 x 513 grid |
| su2cor | quantum physics; Monte Carlo simulation |
| hydro2d | Astrophysics; Hydrodynamic Naiver Stokes equations |
| mgrid | Multigrid solver in 3-D potential field |
| applu | Parabolic/elliptic partial differential equations |
| trub3d | Simulates isotropic, homogeneous turbulence in a cube |
| apsi | Solves problems regarding temperature, wind velocity, and distribution of pollutant |
| fpppp | Quantum chemistry |
| wave5 | Plasma physics; electromagnetic particle simulation |

# SPEC ’95

*クロックレートを倍にすると性能も倍になるか?*

*遅いクロックレートのマシンの方が性能が良い場合があるか?*



# さらなる改良: SPEC 2000

- CINT2000 and CFP2000 are based on compute-intensive applications provided as source code. CINT2000 contains eleven applications written in C and 1 in C++ (252.eon) that are used as benchmarks:

| Name | Ref Time | Remarks |
|---|---|---|
| 164.gzip | 1400 | Data compression utility |
| 175.vpr | 1400 | FPGA circuit placement and routing |
| 176.gcc | 1100 | C compiler |
| 181.mcf | 1800 | Minimum cost network flow solver |
| 186.crafty | 1000 | Chess program |
| 197.parser | 1800 | Natural language processing |
| 252.eon | 1300 | Ray tracing |
| 253.perlbmk | 1800 | Perl |
| 254.gap | 1100 | Computational group theory |
| 255.vortex | 1900 | Object Oriented Database |
| 256.bzip2 | 1500 | Data compression utility |
| 300.twolf | 3000 | Place and route simulator |

- CFP2000 contains 14 applications (6 Fortran-77, 4 Fortran-90 and 4 C) that are used as benchmarks:

| Name | Ref Time | Remarks |
|---|---|---|
| 168.wupwise | 1600 | Quantum chromodynamics |
| 171.swim | 3100 | Shallow water modeling |
| 172.mgrid | 1800 | Multi-grid solver in 3D potential field |
| 173.applu | 2100 | Parabolic/elliptic partial differential equations |
| 177.mesa | 1400 | 3D Graphics library |
| 178.galgel | 2900 | Fluid dynamics: analysis of oscillatory instability |
| 179.art | 2600 | Neural network simulation; adaptive resonance theory |
| 183.equake | 1300 | Finite element simulation; earthquake modeling |
| 187.facerec | 1900 | Computer vision: recognizes faces |
| 188.ammp | 2200 | Computational chemistry |
| 189.lucas | 2000 | Number theory: primality testing |
| 191.fma3d | 2100 | Finite element crash simulation |
| 200.sixtrack | 1100 | Particle accelerator model |
| 301.apsi | 2600 | Solves problems regarding temperature, wind, velocity and distribution of pollutants |

# Specの報告の例

最近SPEC CPU 2006

## CFP2000 Result

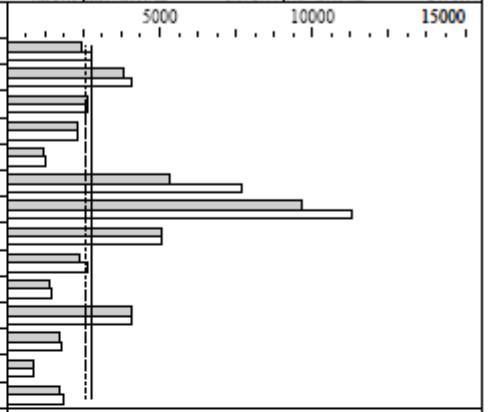


**IBM Corporation**
IBM eServer p5 595 (1900 MHz, 1 CPU)

SPECfp2000 = 2796
SPECfp_base2000 = 2585

SPEC license #: 11 Tested by: IBM Test date: Jan-2005 Hardware Avail: Nov-2004 Software Avail: Dec-2004

| Benchmark | Reference Time | Base Runtime | Base Ratio | Runtime | Ratio |
|---|---|---|---|---|---|
| 168.wupwise | 1600 | 65.4 | 2448 | 57.1 | 2800 |
| 171.swim | 3100 | 81.4 | 3809 | 75.8 | 4088 |
| 172.mgrid | 1800 | 68.6 | 2623 | 68.1 | 2643 |
| 173.applu | 2100 | 90.5 | 2320 | 90.5 | 2320 |
| 177.mesa | 1400 | 116 | 1207 | 110 | 1268 |
| 178.galgel | 2900 | 54.1 | 5356 | 37.9 | 7659 |
| 179.art | 2600 | 27.0 | 9623 | 23.1 | 11279 |
| 183.equake | 1300 | 25.6 | 5068 | 25.6 | 5076 |
| 187.facerec | 1900 | 79.8 | 2381 | 72.7 | 2615 |
| 188.ammp | 2200 | 160 | 1373 | 150 | 1462 |
| 189.lucas | 2000 | 49.3 | 4060 | 49.3 | 4060 |
| 191.fma3d | 2100 | 120 | 1743 | 116 | 1804 |
| 200.sixtrack | 1100 | 131 | 839 | 125 | 877 |
| 301.apsi | 2600 | 152 | 1712 | 140 | 1858 |




### Hardware

| | |
|---|---|
| CPU: | POWER5 |
| CPU MHz: | 1900 |
| FPU: | Integrated |
| CPU(s) enabled: | 1 core, 1 chip, 1 core/chip (SMT off) |
| CPU(s) orderable: | 16,24,32,40,48,56,64 |
| Parallel: | No |
| Primary Cache: | 64KBI+32KBD (on chip) |
| Secondary Cache: | 1920KB unified (on chip) |
| L3 Cache: | 36MB unified (off-chip)/chip, 1 chip/MCM, 8 MCM/SUT |
| Other Cache: | None |
| Memory: | 256 GB DDR2 |
| Disk Subsystem: | 2x36GB SCSI, 15K RPM |
| Other Hardware: | None |

### Software

| | |
|---|---|
| Operating System: | AIX 5L V5.3 |
| Compiler: | XL C/C++ Enterprise Edition Version 7.0 for AIX<br>XL Fortran Enterprise Edition V9.1 for AIX |
| | Other Software: ESSL 4.2 |
| File System: | AIX/JFS2 |
| System State: | Multi-user |

### Notes/Tuning Information

```
Portability Flags:
     -qfixed used in:        168.wupwise, 171.swim, 172.mgrid, 173.applu,
                             178.galgel, 200.sixtrack, 301.apsi
     -qsuffix=f=f90 used in: 178.galgel, 187.facerec, 189.lucas, 191.fma3d

Base Optimization Flags:
  Fortran:    -O5 -lhmu -blpdata -lmass
  C:          -qpdf1/pdf2
              -O5 -blpdata -qalign=natural

Peak Optimization Flags
  168.wupwise:  fdpr -q -O3
                -O5 -q64 -blpdata -lmass -qalign=struct=natural -qfdpr
  171.swim:     fdpr -q -O3
                -O5 -q64 -qarch=pwr3 -qtune=pwr3 -blpdata -lmass -qalign=struct=natural -qfdpr
                F77=xlf90
  172.mgrid:    -qpdf1/pdf2
```

Standard Performance Evaluation Corporation
info@spec.org
http://www.spec.org

# Top500 とは? (その1、概要)


- “The Top500 Supercomputing Sites” http://www.top500.org/
- 1993年6月にHans Meuer(写真), Erich Strohmaier (Manheim 大学), Jack Dongarra (Tennessee大学), Horst Simon (NERSC)が全世界の計算機のベンチマークをアーカイブするプロジェクトとして開始
  - C.f. “Dongarra List”
- 年二回更新、6月はHeidelberg開催のInternational Supercomputing Conf.にて、11月は米国IEEE Supercomputingにて発表、上位3位を表彰、500位まで賞状

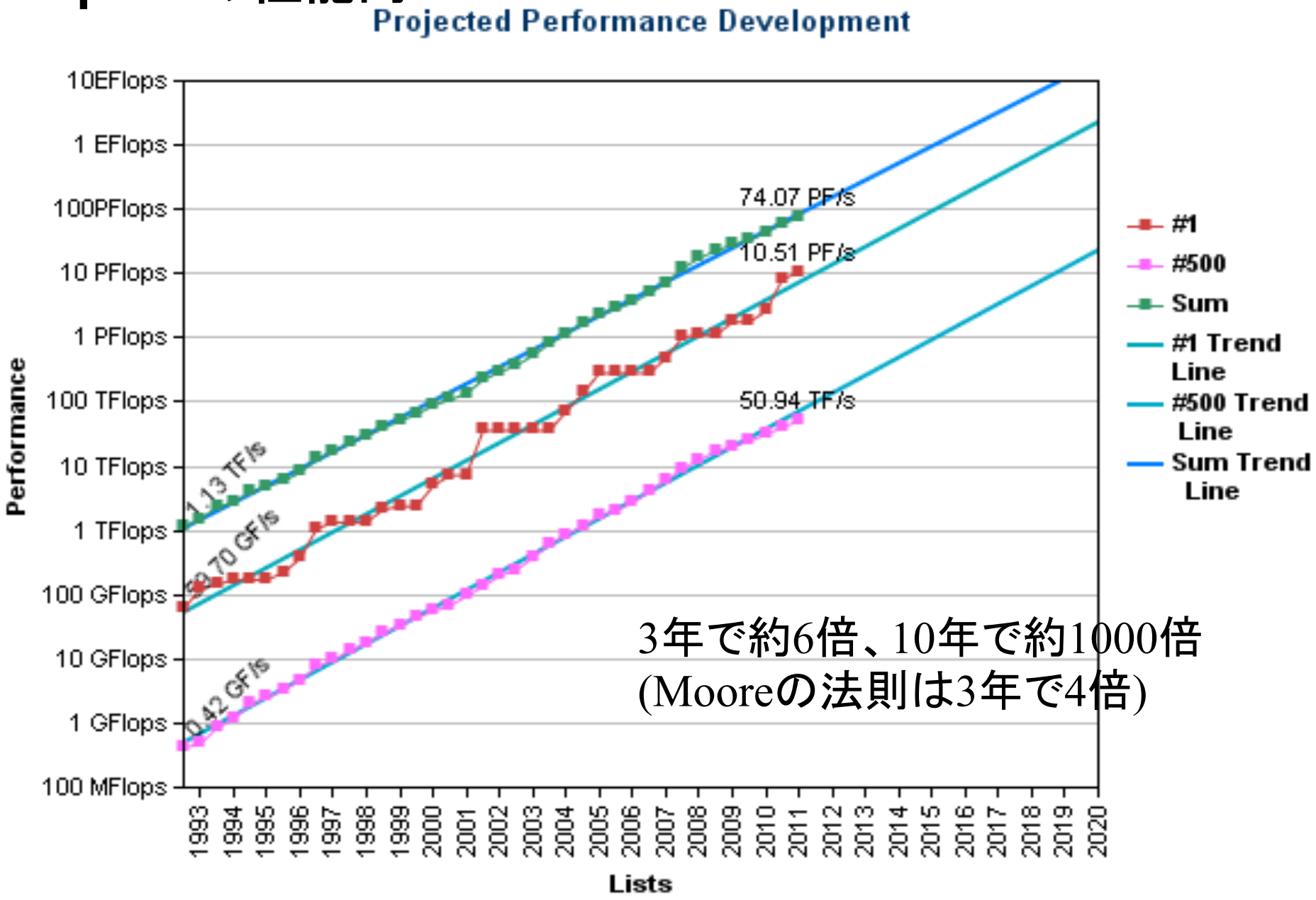
# Top500 とは? (その2、技術内容)


- ランキング指標: Linpack (Partial pivotingを行うLU分解)実行時の毎秒の平均浮動小数点演算回数(FLOPS)　(行列サイズnに対し 2/3 n^3 + O(n^2) 演算)
  - 並列計算機では、HPLなどの並列化されたLU分解のプログラムを用いる
- 用語集
  - Rmax – 計測されたLinpack実行のFLOPS値
  - Nmax – Rmaxを達成したときの行列サイズ
  - N ½ - Rmaxの半分の性能を達成したときの行列サイズ
  - Rpeak – 理論最高性能FLOPS値
  - Efficiency (実行効率) – Rmax / Rpeak
  - #proc – プロセッサ数
  - Nworld – 世界全体でのRmaxによるランキング(1-500位)

# Top500とは?(その3、競争)


- 実際の科学技術計算のシナリオだけではない
  - 高性能なLinpackは計算機のハードウェア(Disk I/Oを除く)にまんべんなく多大な負荷
- 1993年Nworld =1のCM-5(59GigaFlops)から始まり、特に近年争いが激しい
  - C.f. Mooreの法則3年で4倍、Top500は6倍
  - 上位は、メーカー、研究所、国をあげての熾烈なプライドの戦い
  - 2002年4月1位はNEC の地球シミュレータ(Rmax = 35.86 TeraFlops)
  - 2005年11月1位は米国Lawrence Livermore National LaboratoryのBlueGene/L Rmax = 280.60 TeraFlops
  - 各部門でも激烈な争い
    - 「何台Top500に入ったか」
  - 2008年ははじめてPetaflopsの年に？ ($10^{15}$FLOP/秒)

# Top500の性能向上




3年で約6倍、10年で約1000倍
(Mooreの法則は3年で4倍)

# The Green 500

- 最もTop500の性能に対して電力性能が良いスパコンのランキング
- Linpack実行の性能値をその際の電力で割った値。
- http://www.green500.org

# TSUBAME2.0世界ランキング
# スパコン二大リスト (2010年11月)


## The Top 500 (ベンチマーク絶対性能、ペタフロップス)

1位: 2.566 中国防衛大 Tianhe 1-A (11)
2位: 1.758 :米国オークリッジ国立研究所 Cray Jaguar (81)
3位: 1.271 :中国深圳国立スパコンセンター Dawning Nebulae (13)
4位: 1.192 :日本 東工大/HP/NEC TSUBAME2.0 (2)
5位: 1.054 :米国ローレンスバークレー国立研究所 Cray Hopper (30)
6位: 1.050 :仏CEA国立研究所 Bull Bullx (97)
7位: 1.042 :米国オークリッジ国立研究所 IBM Roadrunner (16)
33位(日本2位): 0.1914:日本原子力研究開発機構/富士通 (95)

(Green500 rank)


## The Green 500 (ベンチマーク電力性能、メガフロップス/W)

1位: 1684.20 : 米国 IBM研究所 BlueGene/Q プロトタイプ (116)
2位: 958.35 :日本 東工大/HP/NEC TSUBAME2.0 (4)
3位: 933.06 :米国 NCSA Hybrid Cluster実験機 (403)
4位: 828.67 : 日本 理研 京 (170)
5-7位: 773.38 : ドイツ ユーリッヒ大等 IBM QPACE SFB TR (207-209)
10位(日本3位): 636.36 :日本 環境研 (102)

(Top500 rank)

"Little Green 500" では TSUBAME2.0の実験構成が
1.037 Gigaflops/W 達成 (米Microsoftとの共同研究)

THE GREEN 500™

sponsored by

SUPERMICRO

This certificate is in recognition of your organization's achievements in reducing the environmental impact of high-performance computing.

**GSIC Center, Tokyo Institute of Technology**

Is recognized as the

**Greenest Production Supercomputer in the World**

on the world's Green500 List of computer systems as of

**November 2010**

Wu-chun Feng, Co-Chair

Kirk Cameron, Co-Chair



# Supercomputing 2010 @ New Orleans

## 東工大ブース


# ペタフロップス？ギガフロップス/W?

**6.6万倍高速**

**3倍省エネ**

**<<**

**4.4万倍データ**

Laptop: SONY Vaio type Z (VPCZ1)
CPU: Intel Core i7 620M (2.66GHz)
MEMORY: DDR3-1066 4GBx2
OS: Microsoft Windows 7 Ultimate 64bit
HPL: Intel(R) Optimized LINPACK Benchmark for Windows (10.2.6.015)
256GB HDD

**18.1 ギガ($10^9$)フロップス**

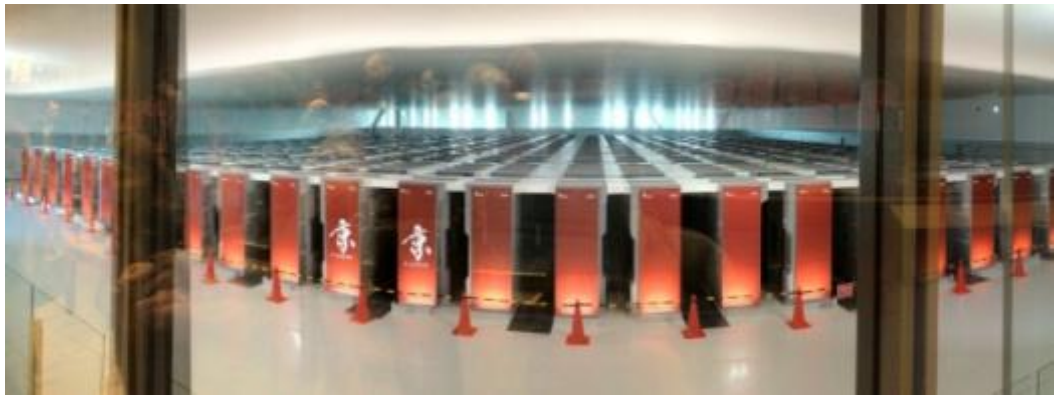
**369 メガ($10^6$)フロップス / Watt**

Supercomputer: TSUBAME 2.0
CPU: 2714 Intel Westmere 2.93 Ghz
GPU: 4071 nVidia Fermi M2050
MEMORY: DDR3-1333 80TB + GDDR5 12TB
OS: SuSE Linux 11 + Windows HPC Server R2
HPL: Tokyo Tech Heterogeneous HPL
11PB Hierarchical Storage

**1.192 ペタ($10^{15}$)フロップス**

**1037 メガ($10^6$)フロップス / Watt**

# さらに10ペタフロップス、100万CPUコア時代へ

2010年6月世界最速(11月は2位)
ORNL/Cray XT5 "Jaguar"
~250,000 AMD"Istanbul" Opteron CPU Cores, 2.3 Petaflops Peak, 1.8 Petaflops Linpack
~200 racks, ~580m2 floorspace
362TB Memory, ~7MW Power,
10 Petabytes HDD

我が国の神戸「京」(2011-12)
Fujitsu Sparc VIII-fx "Venus"
~700,000 Cores, > 10 Petaflops Peak
> 1PB Memory, ~20MW Power
> 1000 racks, 設置面積~6500m2
> 100PB HDD

> 1 エクサフロップス
> 10億CPUコア in 2018

# アムダールの法則 (Amdahl's Law)

改良後の実行時間 =

影響されない部分の実行時間 +(影響される部分の実行時間 / 改良の度合)

- 例:
  Q: あるプログラムの実行時間が100秒で、そのうち乗算が80%の時間を占めていたとする。実行性能を4倍にするには、乗算の性能を何倍にしなくてはならないか?
  - T = 1/5T + 4/5T => 1/4T = 1/5T + 4/5T*1/P => P = 16
- また、実行性能を5倍にするのには??
  - P -> ∞
- 並列処理などにも良くAmdahlの法則が用いられる

# アムダールの法則の例

- 以下、英語の練習を兼ねて、解いてみよ

- **Suppose we enhance a machine making all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?**

- **We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?**

# 性能評価・まとめ

- 性能は本来はある特定のプログラムに固有のものである
  - 全体の実行時間は性能の一貫性のある尺度となりうる

- あるアーキテクチャに関しては、性能向上は以下によってもたらされる:
  - クロックレートの増加 (CPIに対する著しい悪影響なしに)
  - CPIを低下させるためのアーキテクチャ構成の新技術
    - 例:パイプライン、スーパスカラなど
  - 命令数とCPIを減少させるようなコンパイラのコード生成

- 落とし穴: あるマシンの側面の改良が、必ずしもそれに比例した性能向上をもたらすとは限らない
- 性能のメトリックは、近年では実行時間以外にも、電力効率などいくつか重要な指標が出てきている。
- 性能に関して、新聞やカタログなどで読むことを全て信じてはいけない!!