CS 5803
Introduction to High Performance Computer Architecture: Performance Metrics

A.R. Hurson
323 Computer Science Building,
Missouri S&T
hurson@mst.edu

Introduction to High Performance Computer Architecture

- Instructor: Ali R. Hurson
  323 CS Building
  hurson@mst.edu
- Text: Computer Organization, Design, and Architecture, 5th edition (Shiva)
- Course material is available at:
  https://sites.google.com/a/mst.edu/hurson/courses/cs388

Grading Policy

- Periodic Exams and quizzes : 50%
- Final Exam (Comprehensive): 30%
- Project(s): 10%
- Home works: 10%

Introduction to High Performance Computer Architecture

- Hardcopy of homeworks are collected in class,
- It is encouraged to work as a group (at most two people per group) on homeworks and project (grouping is fixed through out the semester,
- Groups are not allowed to discuss about homework assignments and project with each other,
- December 3 is the deadline for filing grade corrections; no requests for grade change/update will be entertained after this deadline.

Introduction to High Performance Computer Architecture

- Outline:
  - 1. Performance measures
  - 2. High Speed Arithmetic Techniques
    - a) Fast Adder/Subtractor
    - b) Fast Multiplier/Divider
  - 3. Memory Hierarchy, Organization and Design
    - Virtual Memory
    - Cache Memory
    - Interleaved Memory
    - d) Associative Memory
  - 4. Study of Advanced Processor Features
    - a) Uniprocessor (RISC, CISC)
    - b) Stack Machines
    - c) Pipelining and Pipeline Design
    - d) Fine Grain Parallelism
  - 5. Instruction Level Parallelism
  - 6. Study of a multifunctional system
  - 7. How to break RISC barrier/superscalar/VLIW/super pipeline
  - 8. Study of Pentium/Power PC/Multicore architecture

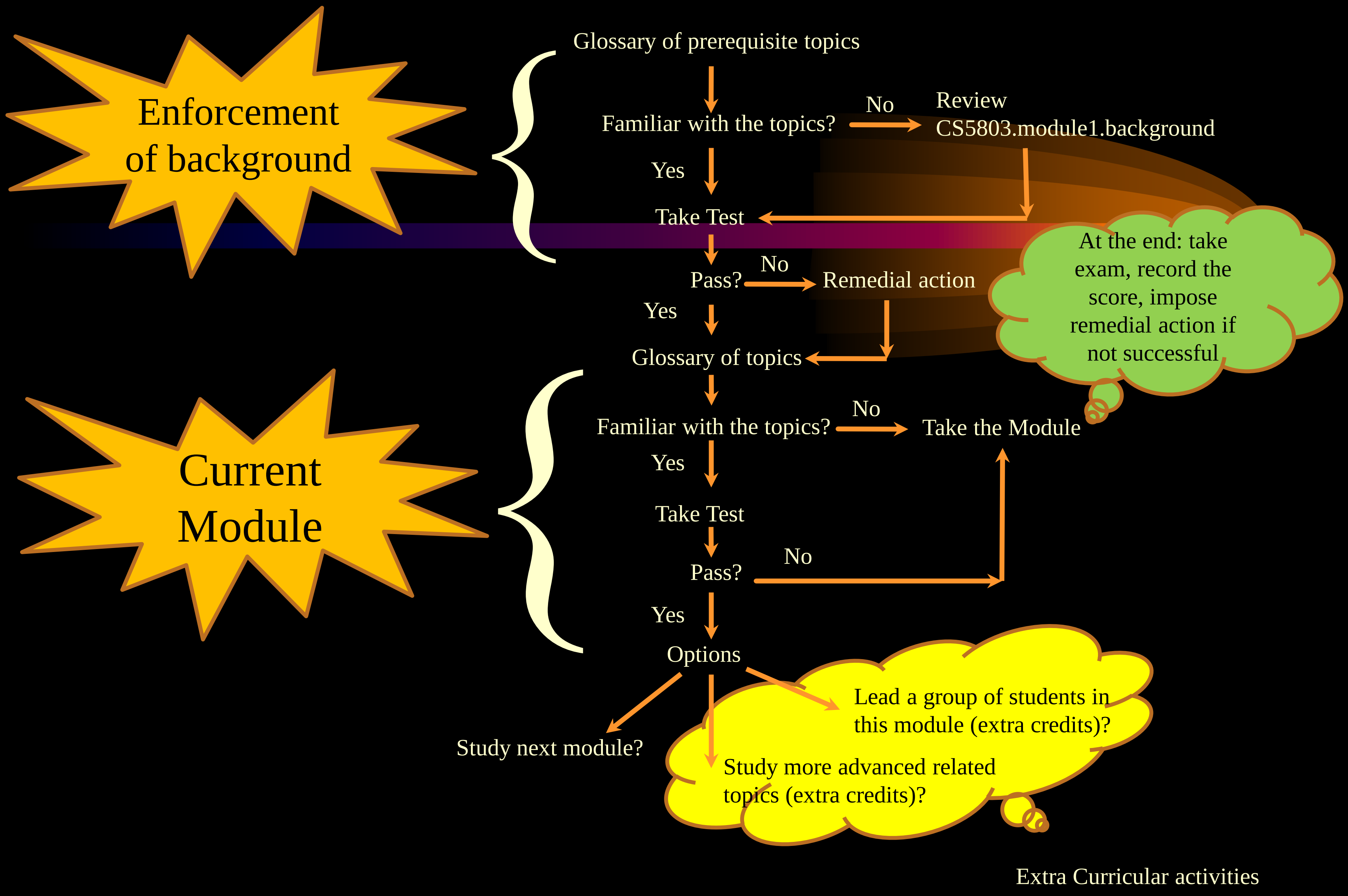
Introduction to High Performance Computer Architecture

- Outline
  - Some Performance Measure and Performance Metrics (I am expecting you to be familiar with this subject, if not, please see module1.background)
  - Amdahl's law
  - Green computing
  - CPU Time
  - Formulation of CPU time in terms of Instruction count, clock cycle time, and number of clock cycles per instruction
  - Formulation of CPU time in terms of Instruction count, clock cycle time, number of clock cycles per instruction, and role of different components in a simple computer organization
  - How to improve performance?

Note, this unit will be covered in two lectures. In case you finish it earlier, then you have the following options:

1) Take the early test and start CS5803.module2
2) Study the supplement module (supplement CS5803.module1)
3) Act as a helper to help other students in studying CS5803.module1

Note, options 2 and 3 have extra credits as noted in course outline.


Enforcement of background
Glossary of prerequisite topics
Familiar with the topics?
No
Review CS5803.module1.background
Yes
Take Test
Pass?
No
Remedial action
Yes
Glossary of topics
At the end: take exam, record the score, impose remedial action if not successful
Current Module
Familiar with the topics?
No
Take the Module
Yes
Take Test
Pass?
No
Yes
Options
Study next module?
Lead a group of students in this module (extra credits)?
Study more advanced related topics (extra credits)?
Extra Curricular activities

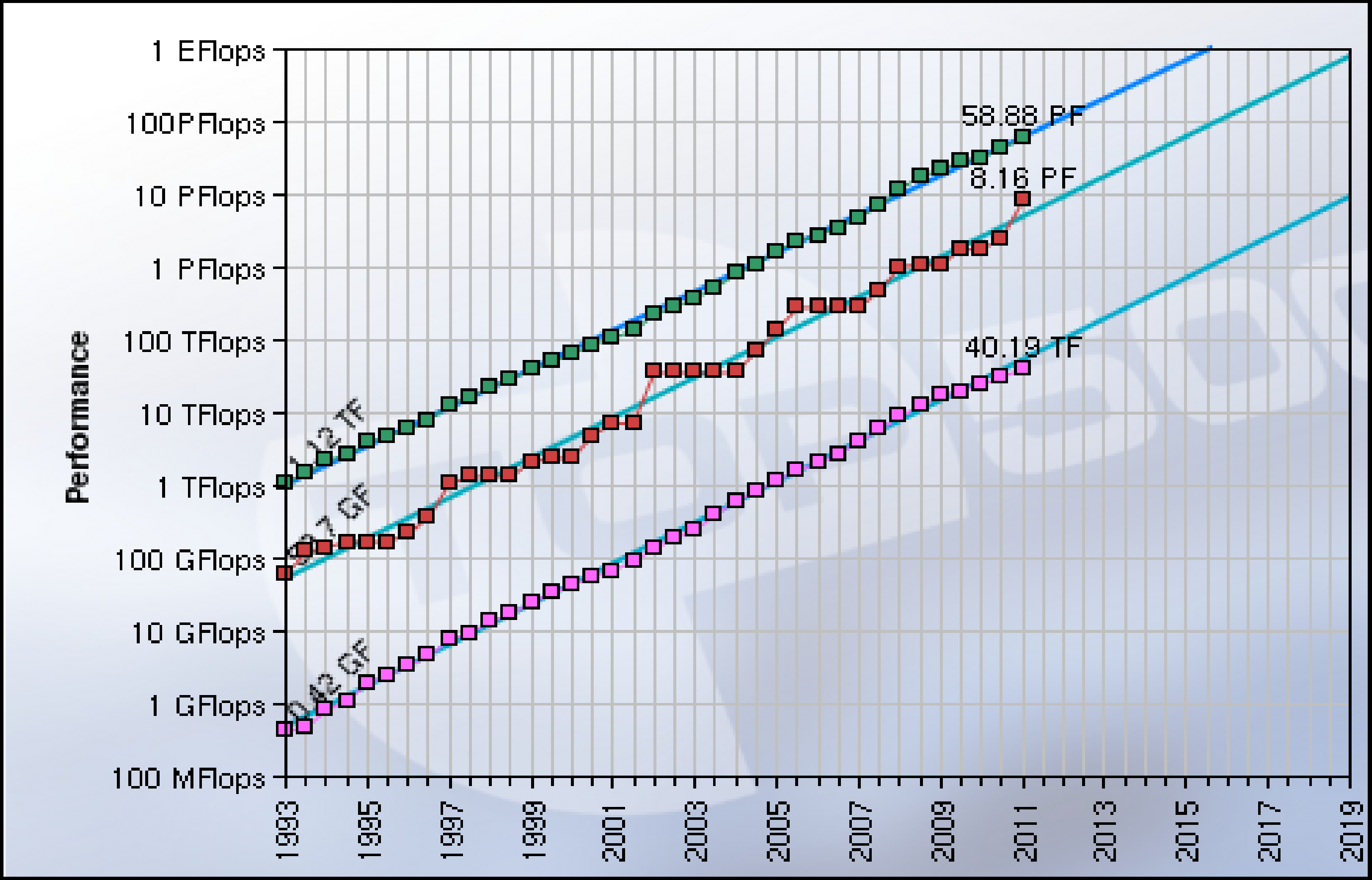
- You are expected to be familiar with:
  - Major components of a computer,
  - Flow of operations and control in a simple computer,
  - Some performance metrics
- If not, you need to study CS5803.module1.background

- Who has the fastest
  - The first Top500 list was created in 1993.
  - In 2012, Top500 list was dominated by IBM Blue Gene/Q with 4 systems in the top 10. The largest of which at Lawrence Livermore National Laboratory with more than 16 Petaflops sustained performance.

The Top10 supercomputers (as of 2011)

Rank	Site	Computer/Year Vendor	Country	Cores	R_{max} (Pflops)	R_{peak} (Pflops)	Power (MW)
1	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz,/ 2011 Fujitsu	Japan	548,352	8.162	8.774	9.899
2	National Supercomputing Center in Tianjin	Tianhe-1A - NUDT / 2010 NUDT	China	186,368	2.566	4.701	4.040
3	DOE/SC/Oak Ridge National Laboratory	Jaguar - Cray XT5-2.6 GHz / 2009 Cray Inc.	USA	224,162	1.759	2.331	6.951
4	National Supercomputing Centre in Shenzhen	Nebulae - Dawning TC3600 Blade/ 2010 Dawning	China	120,640	1.271	2.984	2.580
5	GSIC Center, Tokyo Institute of Technology	TSUBAME 2.0/2010 NEC/HP	Japan	73,278	1.192	2.288	1.399
6	DOE/NNSA/LANL/SNL	Cielo - Cray XE6 8-core 2.4 GHz /2011Cray Inc.	USA	142,272	1.110	1.365	3.980
7	NASA/Ames Research Center/NAS	Pleiades - 2.93 Ghz,/ 2011 SGI	USA	111,104	1.088	1.315	4.102
8	DOE/SC/LBNL/NERSC	Hopper - Cray XE6 12-core 2.1 GHz / 2010 Cray Inc.	USA	153,408	1.054	1.289	2.910
9	Commissariat a l'Energie Atomique (CEA)	Tera-100 - Bull bullx super-node S6010/S6030 / 2010 Bull SA	France	138,368	1.050	1.255	4.590
10	DOE/NNSA/LANL	Roadrunner - 3.2 Ghz /2009 IBM	USA	122,400	1.042	1.376	2.346

Trend in Supercomputer technology (as of 2011)

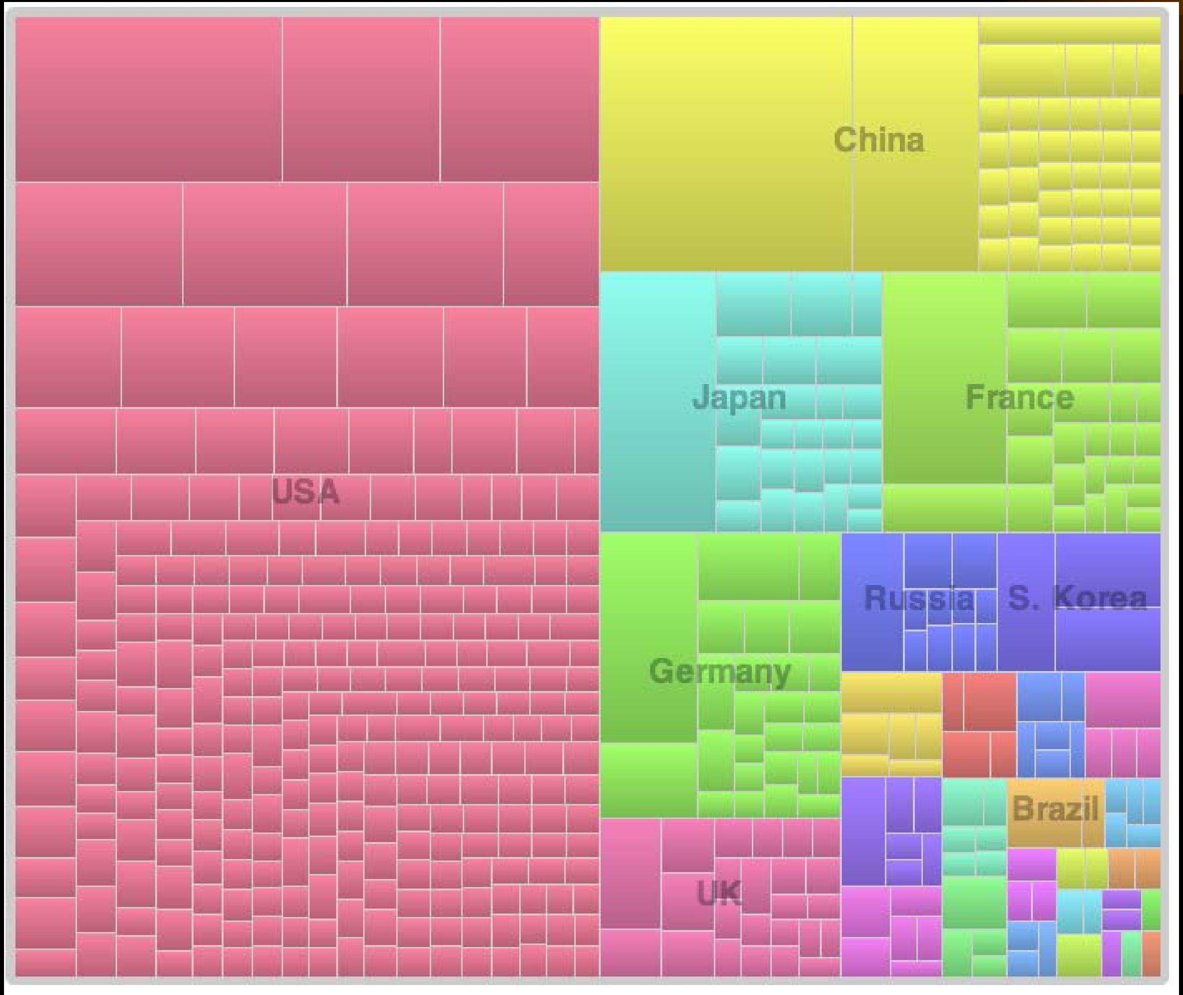


🟥 #1

🟩 Sum

🟪 #500

Countries Share

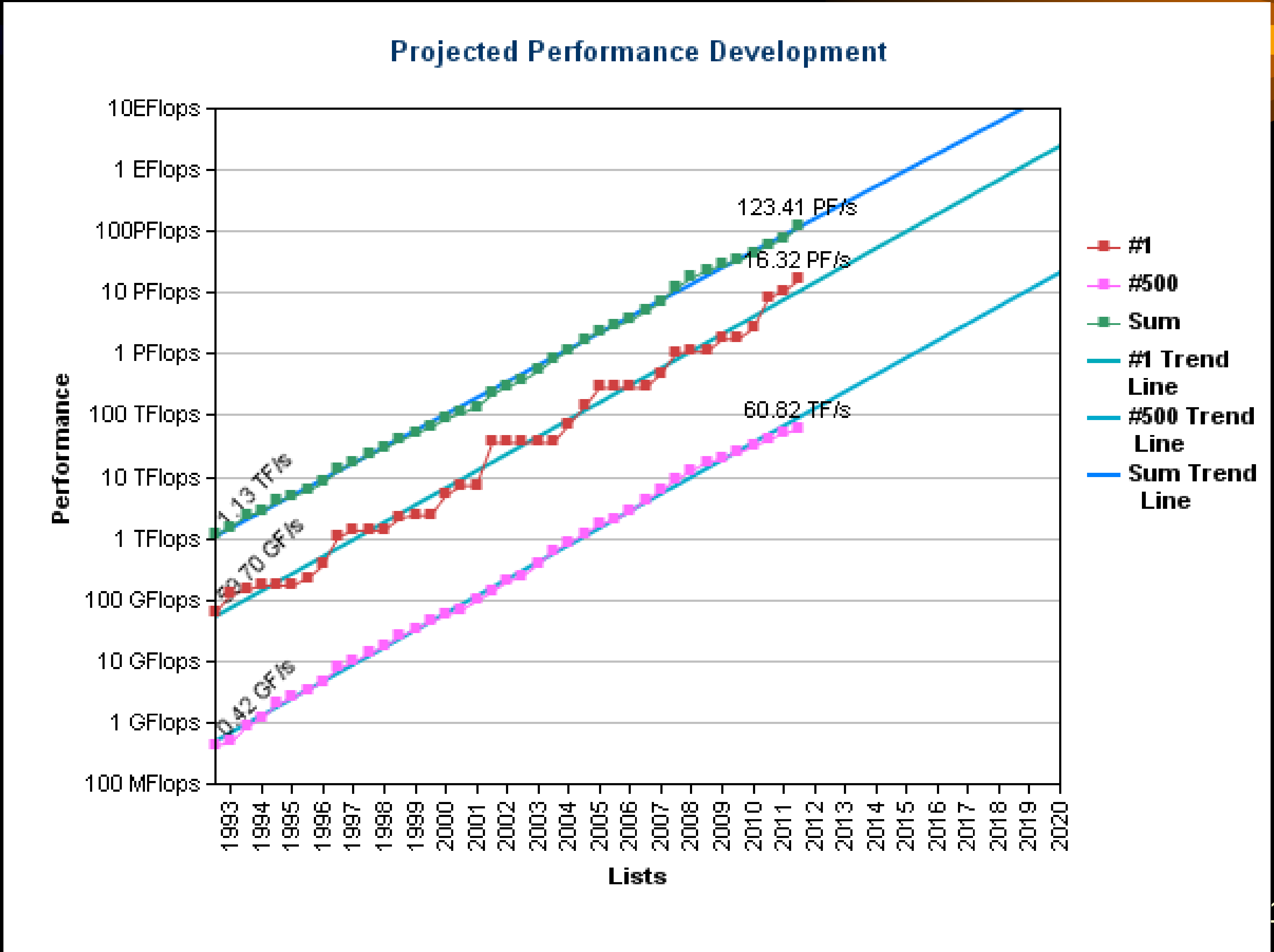


Absolute Counts
US: 274
China: 41
Germany: 26
Japan: 26
France: 26
UK: 25

Rapid change in Countries Share

Countries	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Australia	6	1.20 %	400406	552142	40344
Austria	2	0.40 %	188670	243386	26172
Belgium	2	0.40 %	83840	151472	16704
Brazil	2	0.40 %	269730	330445	37184
Canada	8	1.60 %	640129	890598	82684
China	61	12.20 %	7136315	14331013	881832
Denmark	2	0.40 %	198408	260395	22218
Finland	2	0.40 %	117858	180690	18640
France	25	5.00 %	3180744	4100571	454928
Germany	30	6.00 %	3242111	4181323	568952
India	2	0.40 %	187910	242995	18128
Ireland	1	0.20 %	40495	76608	7200
Israel	2	0.40 %	135361	280436	23928
Italy	5	1.00 %	471746	748248	42080
Japan	26	5.20 %	11182236	13641290	832838
Korea, South	4	0.80 %	950833	1126280	123384
Netherlands	1	0.20 %	50924	64973	3456
Norway	1	0.20 %	40590	51060	5550
Poland	5	1.00 %	315075	448204	44274
Russia	12	2.40 %	1341586	2290994	115120
Saudi Arabia	4	0.80 %	359240	414841	81920
Singapore	2	0.40 %	94073	144562	13192
Spain	2	0.40 %	135860	197696	14160
Sweden	5	1.00 %	489530	661642	75280
Switzerland	4	0.80 %	317895	383373	49480
Taiwan	2	0.40 %	220504	313570	32148
United Kingdom	27	5.40 %	1872107	2806546	260572
United States	255	51.00 %	25265849	36064596	3887556

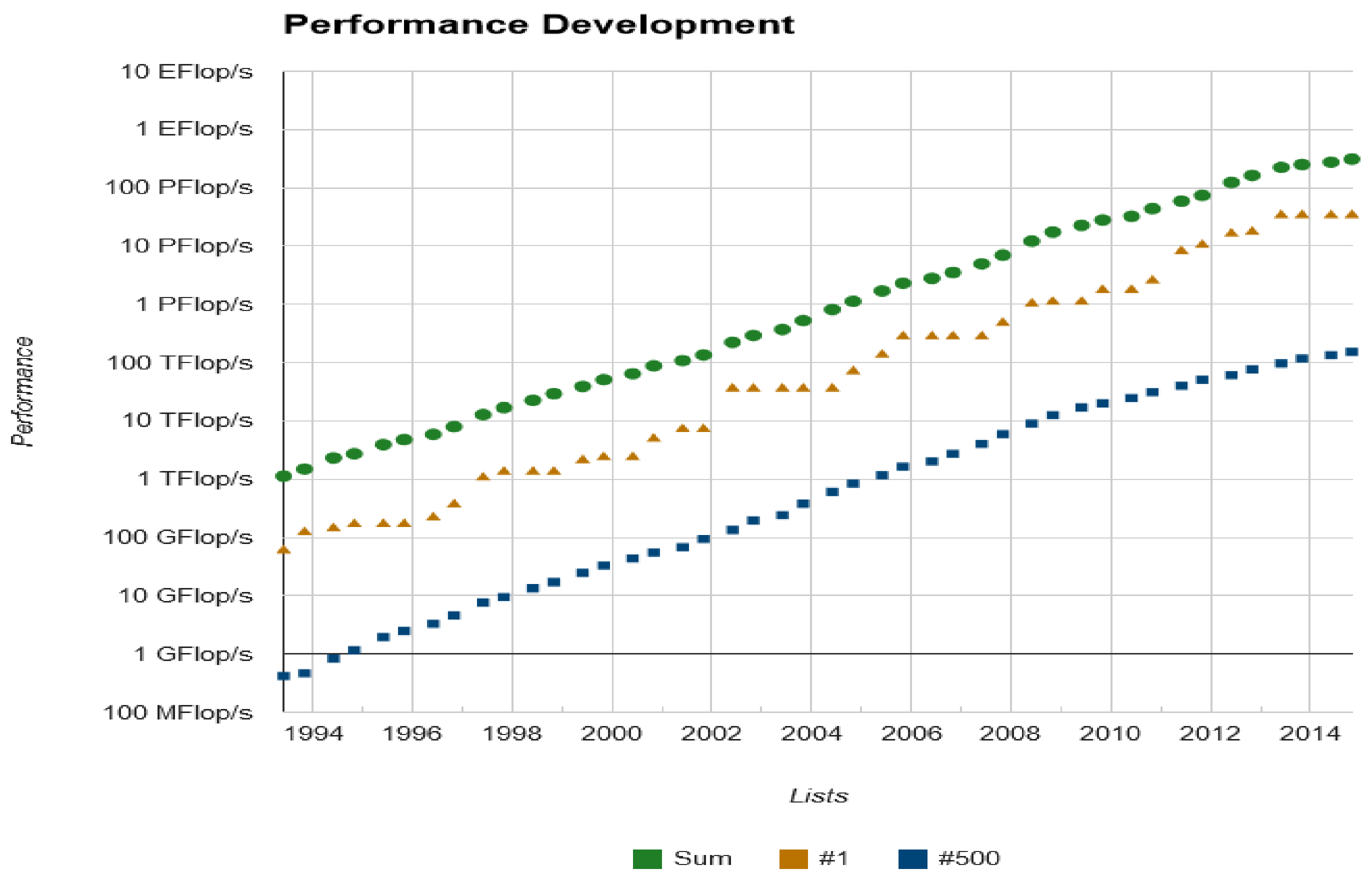
Trend in Supercomputer technology (as of June 2012)



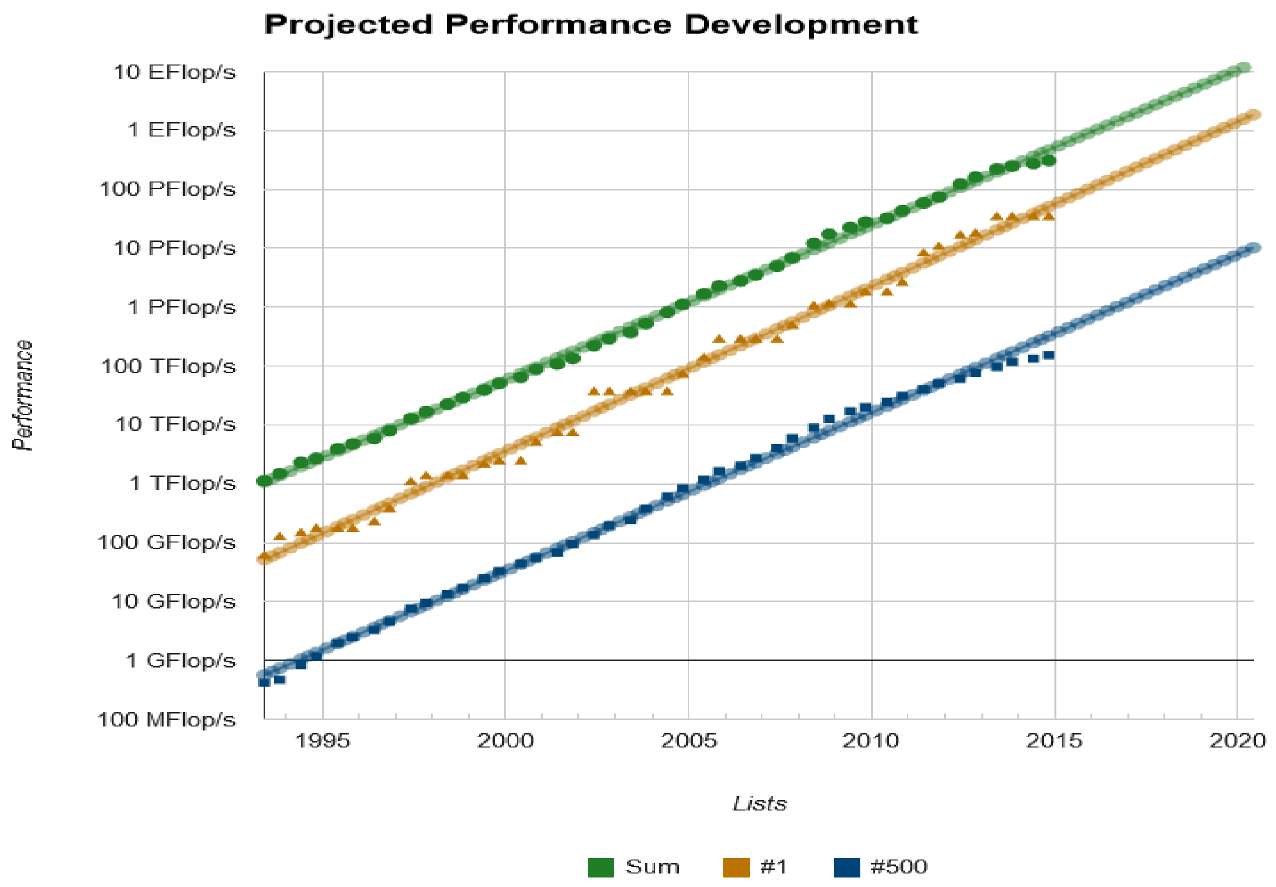
Introduction to High Performance Computer Architecture

Period	Site	Computer/Year Vendor	Country	Architecture	R_{max} (flops)	Power (W)
11/93-12/95	National Aerospace Laboratory	Numerical Wind Tunnel	Japan	Superscalar/vector processor	170 G	
6/1993	Los Alamos	CM-5	USA	Vector processor	59.7 G	
6/94	Sandia	Intel Paragon	USA	MPP	143 G	
6/1996	University of Tokyo	HITACHI SR2201	Japan	Distributed memory	232 G	
11/96	Center for Computational Science	CP-PAC: Hitachi	Japan	MPP	368 G	
6/97-6/00	Sandia	ASCI RED: Intel	USA	Distributed memory	1.068 T	
11/00-11/01	Lawrence Livermore	ASCI White: IBM	USA	Hierarchical	7.2 T	
6/02-6/04	Earth Simulator Center	Earth Simulator: NEC	Japan	Vector processor	9 T	20 K
11/04-11/07	Lawrence Livermore	BlueGene/L: IBM	USA		478 T	
6/08-6/09	Los Alamos	Roadrunner: IBM	USA	Hybrid	1.026 P	
11/09-6/10	Oak Ridge	Jaguar: Cray XT5	USA	+200,000 cores	1.759 P	
11/2010	National Supercomputer Center	TIANHE-1A	China	Hybrid	2.57 P	
2011	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 Fujitsu	Japan	548,352 cores	8.162 P	9,899 K
6/2012	Lawrence Livermore	SEQUIA: IBM BlueGene/Q	USA	1,572,864 cores	16.32 P	7,840 K
11/2012	Oak Ridge national Laboratory	Titan	USA	552,960 cores	17.6 P	8,210 K
6/13-	National University of Defense Technology	TINAHE-2	China	3,120,000 cores	33.86 P	17,808 K

Introduction to High Performance Computer Architecture



Introduction to High Performance Computer Architecture



- Who has the fastest
  - If the projection holds we can expect Exaflops system by 2019.

Introduction to High Performance Computer Architecture

- Introduction
  - Computer Architecture refers to the attributes of a system visible to a programmer — i.e., attributes that have a direct impact on the logical execution of a program.
  - Architectural Attributes include the instruction set, the number of bits used to represent various data types, I/O mechanisms, and techniques for addressing memory.

- Introduction
  - Computer organization refers to the operational units and their interconnections that realize the architectural specifications.
  - Organizational attributes include those hardware details transparent to the programmer: control signals, interfaces between the computer and peripherals, the memory technology, ...

- Introduction
  - Whether or not a computer can support a multiplication instruction is an architectural issue.
  - However, whether the multiplication is performed by a special multiply unit or by a mechanism that makes repeated use of the add unit is an organizational issue.

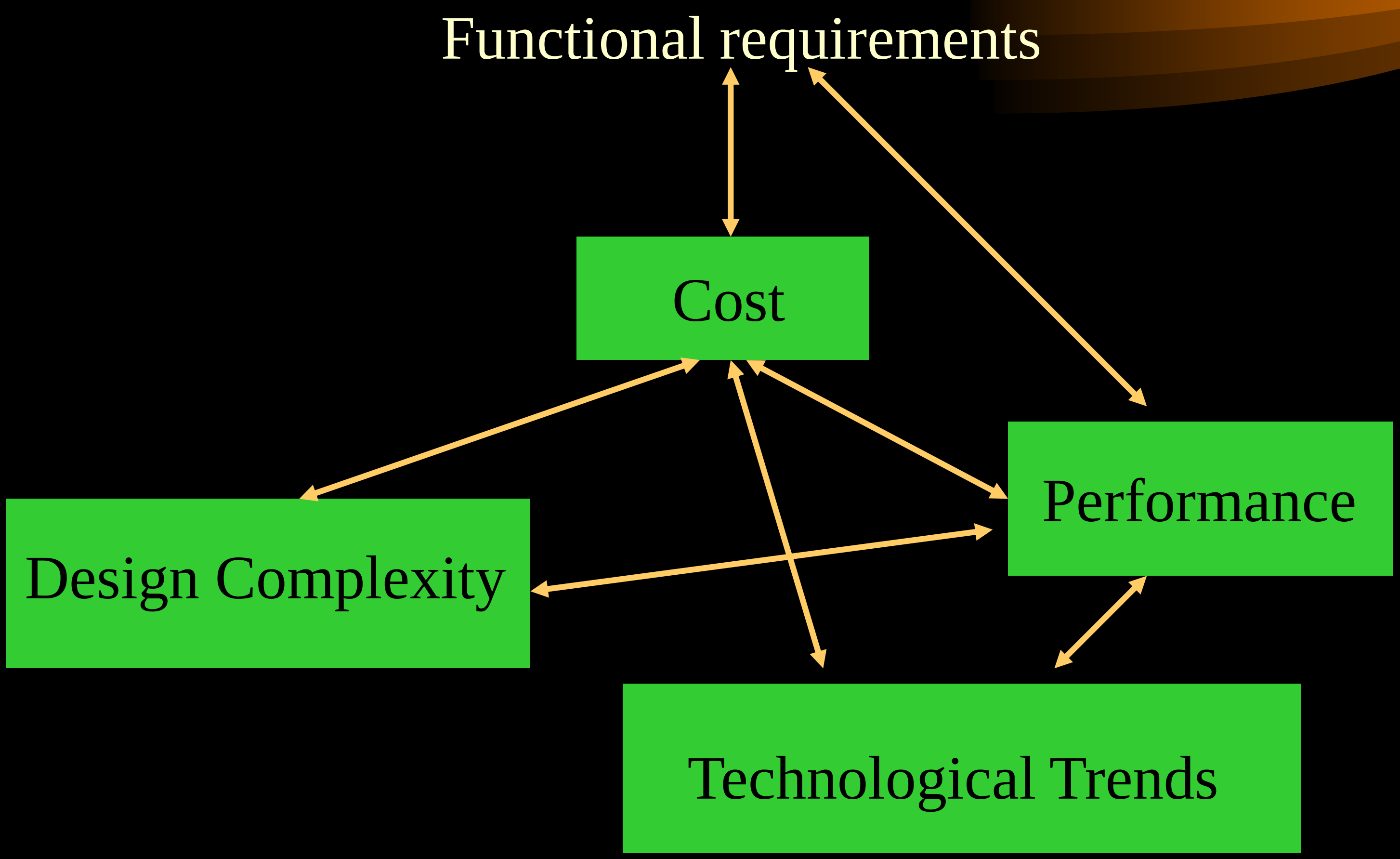
Introduction

- Following IBM, many computer manufacturers offer a family of computer models — all with the same architecture but with differences in organization.
- An architecture may survive many years but its organization changes with changing technology.
- In short, as the technology changes, the organization changes while architecture may remain unchanged.

- Introduction
  - A computer architect is concerned about:
    - The form in which programs are represented to and interpreted by the underlying machine,
    - The methods with which these programs address the data, and
    - The representation of data.

Introduction

- A computer architect should:
  - Analyze the requirements and criteria — Functional requirements
  - Study the previous attempts
  - Design the conceptual system
  - Define the detailed issues of the design
  - Tune the design — Balancing software and hardware
  - Evaluate the design
  - Implement the design — Technological trend


Functional requirements
Cost
Performance
Design Complexity
Technological Trends

Performance Measures

- Speed up — How much faster a task will run using the machine with enhancement relative to the original machine.

$$S = \frac{\text{Execution time on Original Machine}}{\text{Execution time on Enhanced Machine}}$$

- Performance Measures
  - Efficiency — It is the ratio between speed up and number of processors involved in the process:

$$E_p = \frac{S_p}{p}$$

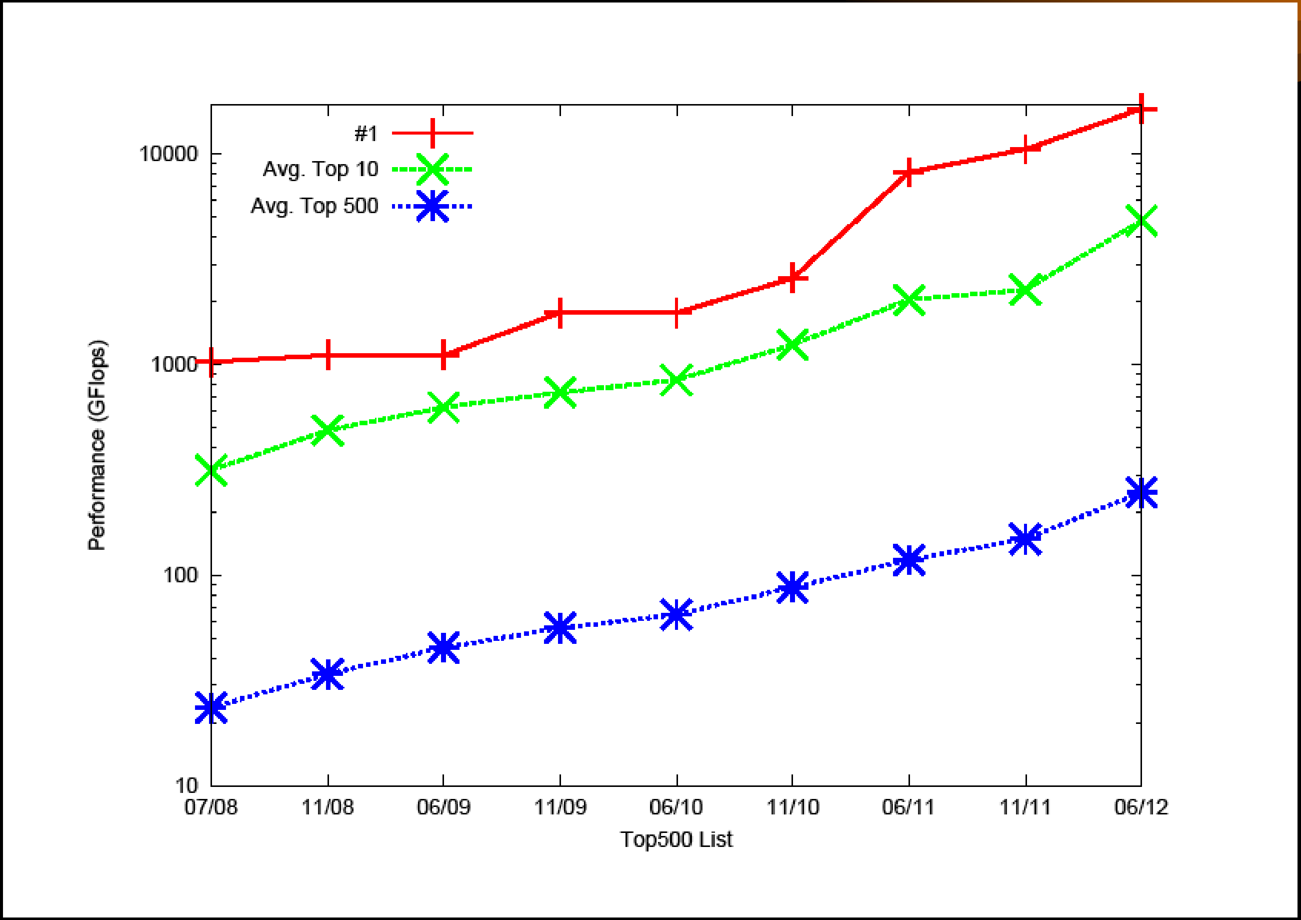
Performance Measures

- Efficiency can be discussed, mainly, within the scope of concurrent system.
- Efficiency indicates how effectively the hardware capability of a system has been used.
- Assume we have a system that is a collection of ten similar processors. If a processor can execute a task in 10 seconds then ten processors, collectively, should execute the same task in 1 second. If not, then we can conclude that the system has not been used effectively.

- Performance Measures
  - Green computing
  - Power consumption and power management

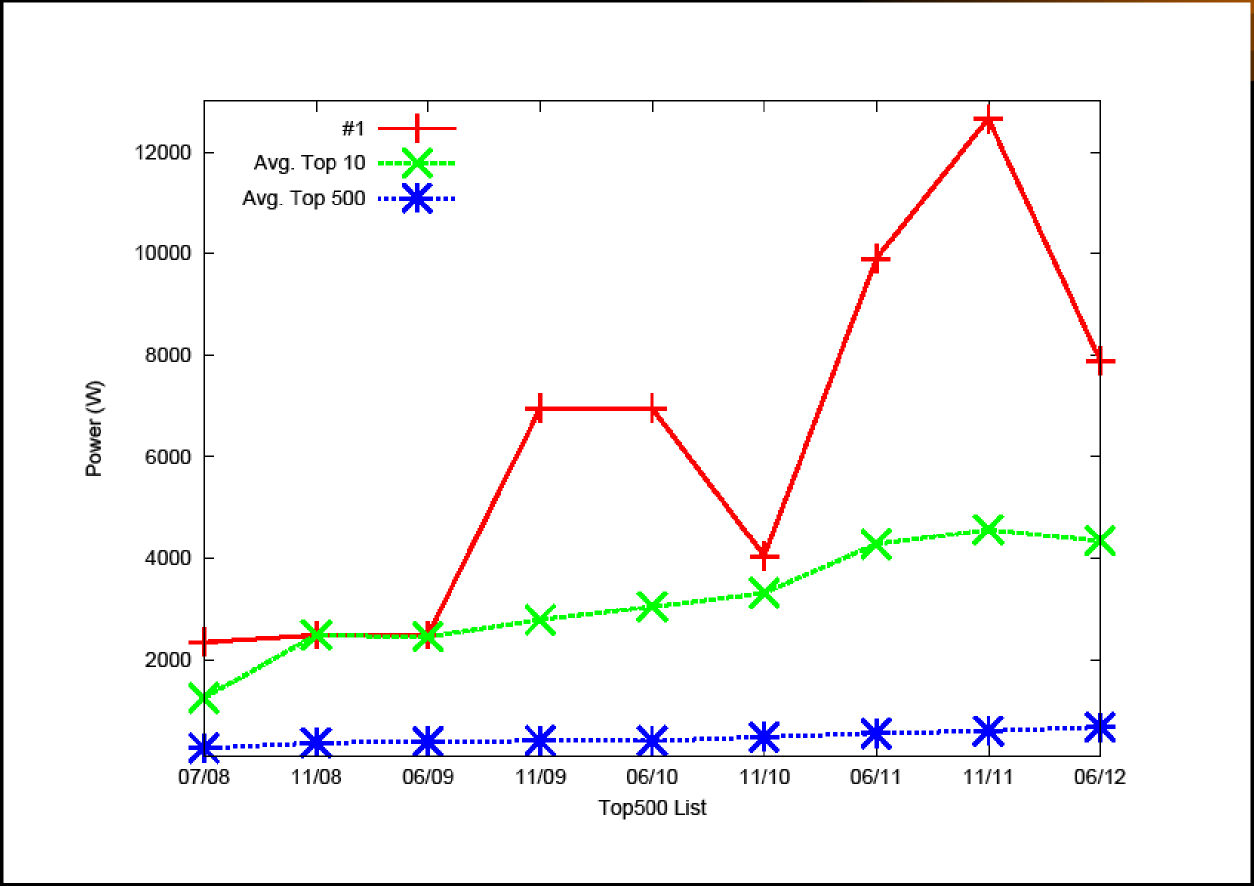
- Is one number enough?
  - As per our discussion, so far, performance was the major design constraint. However, the power is becoming a problem.
  - Power consumption became an issue with the growth of wireless technology and mobile devices. However, it is becoming even more of concern since feeding several Magawatt of power to run a supercomputer is not a trivial task and requires a great amount of supporting infrastructure

◆Is one number enough?



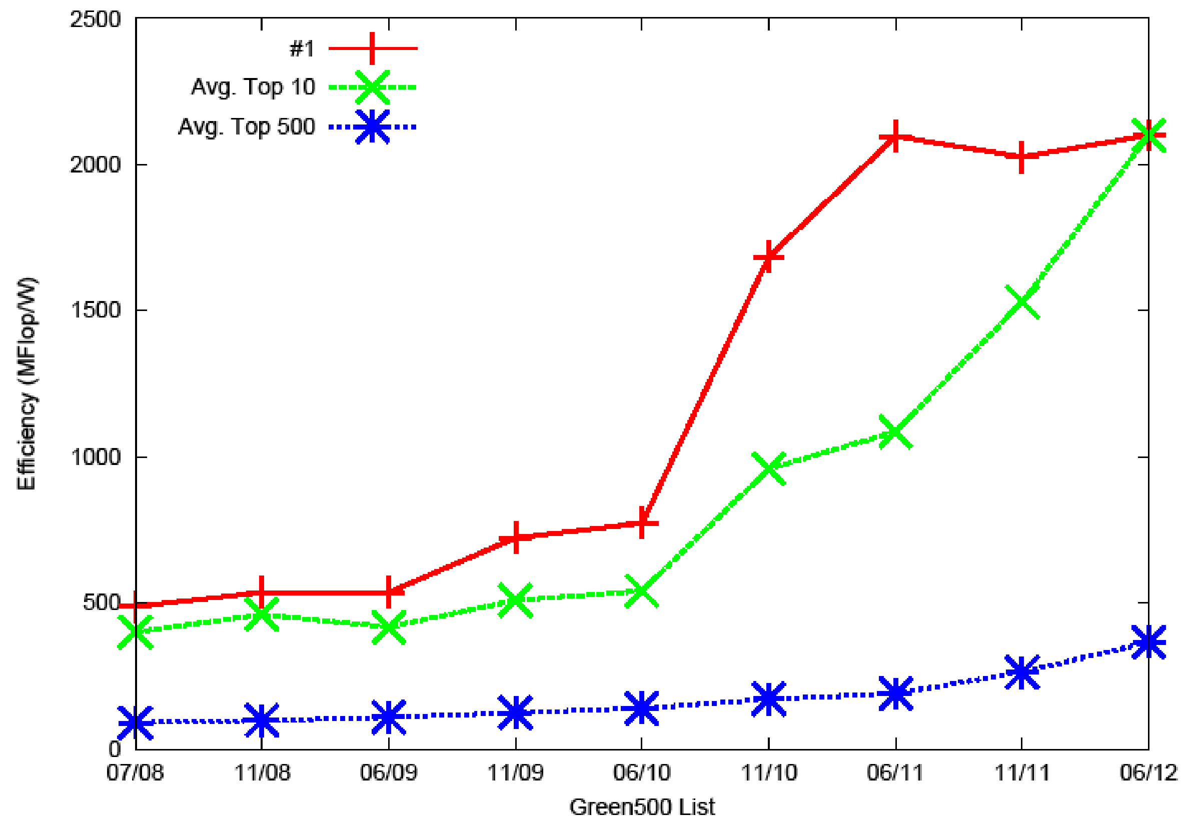
Top500 Performance

Is one number enough?



Top500 Power

◆ Is one number enough?



- Challenges for creating Exaflops machine are:
  - Energy and Power,
  - Memory and Storage,
  - Concurrency and locality, and
  - Resiliency
- An Exaflps machine should consume at most 20 Megawatt of power which corresponds to 50 Gflop/W. To reach this goal, power efficiency needs to be increased by a factor of 25 compared to today’s most power efficient system (IBM Blue Gere/Q)

Performance Measures

- Amdahl's law — The performance improvement gained by improving some portion of an architecture is limited by the fraction of the time the improved portion is used — a small number of sequential operations can effectively limit the speed up of a parallel algorithm.

Performance Measures

- Amdahl's law allows a quick way to calculate the speed up based on two factors — The fraction of the computation time in the original task that is affected by the enhancement, and, the improvement gained by the enhanced execution mode (speed up of the enhanced portion).

◆Performance Measures — Amdahl's law

$$Execution\text{–}time_{new} = Execution\text{–}time_{old} \times \left((1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right)$$

$$Speedup_{overall} = \frac{Execution\text{–}time_{old}}{Execution\text{–}time_{new}} = \frac{1}{\left((1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right)}$$

$$S \le \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{enhanced}}} = \frac{1}{\mathbf{f} + (1 - \mathbf{f}) / \mathbf{p}}$$

Where ***f*** and ***p*** represent the unchanged portion and the speed up of the enhanced portion, respectively.

- Performance Measures
  - Example — Suppose we are considering an enhancement that runs 10 times faster, but it is only usable 40% of time. What is the overall speed up?

$$S = \frac{1}{\left(.6 + \frac{.4}{10}\right)} = \frac{1}{.64} = 1.56$$

Performance Measures

- Example — If 10% of operations, in a program, must be performed sequentially, then the maximum speed up gained is 10, no matter how many processors a parallel computer has.

Performance Measures

- Example — Assume improving the CPU by a factor of 5 costs 5 times more. Also, assume that the CPU is used 50% of time and the cost of the CPU is 1/3 of the overall cost. Is it cost efficient to improve this CPU?

◆Performance Measures

$$S = \frac{1}{.5 + \frac{.5}{5}} = \frac{1}{.6} = 1.67$$

$$\cos t - of - the - new - machine = \frac{2}{3} \times 1 + \frac{1}{3} \times 5 = 2.33 times - of - the - original - machine$$

Performance Measures

- Response Time is defined (Execution time, Latency) as the time elapse between the start and the completion of an event. It is the latency to complete a task that includes disk accesses, memory accesses, I/O activities, operating system overhead, …

- Is response time a good performance metric?

Performance Measures

- The processor of today's computer is driven by a clock with a constant cycle time (τ).
- The inverse of the cycle time is the clock rate (f).
- The size of a program is determined by its instruction count (I_c) — number of the machine instructions to be executed.

Performance Measures

- Let us define the average number of clock cycle per instruction (*CPI*) as:

$$\mathrm{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}} = \frac{\sum_{i=1}^{n} (\mathrm{CPI}_i * I_i)}{I_c}$$

$$= \sum_{i=1}^{n} \left(\mathrm{CPI}_i * \frac{I_i}{I_c}\right)$$

Where I_i is the number of time instruction i is executed and CPI_i is the average number of clock cycles for instruction i.

Performance Measures

- For a given instruction set, one can calculate the CPI over all instruction types, if the frequencies of the appearance of the instructions in the program is known.
- *CPI* depends on the organization/architecture and the instruction set of the machine.
- Clock rate depends on the technology and organization/architecture of the machine.
- Instruction count depends on the instruction set of the machine and compiler technology.

Performance Measures

- The CPU time (T) is the time needed to execute a given program, excluding the time waiting for I/O or running other programs.
- CPU time is further divided into:
  - The user CPU time and
  - The system CPU time.

Performance Measures

$$\text{CPU}_{\text{Time}} = \text{CPU}_{\text{Clock cycles}} * \text{Clock}_{\text{Cycle time}}$$

$$\text{CPU}_{\text{Time}} = \frac{\text{CPU}_{\text{Clock cycles}}}{\text{Clock}_{\text{Rate}}}$$

- The CPU time is estimated as

$$T = I_c * CPI * \tau = \sum_{i=1}^{n} \left(CPI_i * I_i \right) * \tau$$

- Performance Measures
  - Example — It takes 10 seconds to run a program on machine *A* that has a 400 MHz clock rate.
  - We are intended to build a faster machine that will run this program in 6 seconds. However, machine *B* requires 1.2 times as many clock cycles as machine *A* for this program. Calculate the clock rate of machine *B*:

◆Performance Measures

$$CpuTime_A = \frac{CPUClockCycle_A}{ClockRate_A}$$

$$10 = \frac{CPUClockCycle_A}{400 * 10^6 \, Cycles/Second}$$

$$CPUClockCycle_A = 4000 * 10^6$$

$$CPUTime_B = \frac{1.2 * CPUClockCycle_A}{ClockRate_B}$$

$$ClockRate_B = \frac{1.2 * 4000 * 10^6}{6} = 800MHz$$

Performance Measures

- Example — Two machines are assumed: In machine *A* conditional branch is performed by a compare instruction followed by a branch instruction. Machine *B* performs conditional branch as one instruction.
- On both machines, conditional branch takes two clock cycles and the rest of the instructions take 1 clock cycle. 20% of instructions are conditional branches.
- Finally, clock cycle time of *A* is 25% faster than *B*'s clock cycle time. Which machine is faster?

Performance Measures

$CPI_A = .8*1+.2*2 = 1.2$

$t_B = t_A*1.25$

$CPU_A = I_{CA}*1.2*\ t_A$

$CPI_B = .25*2+.75*1 = 1.25$

$CPU_B = .8I_{CA}*1.25t_A*1.25 = I_{CA}*1.25*t_A$

- So *A* is faster.

- Performance Measures
  - Example — Now assume that cycle time of *B* can be made faster and now the difference between the cycle times is 10%. Which machine is faster?

    $CPU_A = I_{CA} * 1.2 * t_A$

    $CPU_B = .8 I_{CA} * 1.1 t_A * 1.25 = I_{CA} * 1.1 * t_A$

  - Now *B* is faster.

- Performance Measures
  - The execution of an instruction requires going through the instruction cycle. This involves the instruction fetch, decode, operand(s) fetch, execution, and store result(s):

$$T = I_c * CPI * \tau = I_c * (p+m*k)* \tau$$

Performance Measures

- The equation

$$T = I_c * CPI * \tau = I_c * (p+m*k) * \tau$$

is the major basis for this course. We will refer to this equation through out the course.

- Performance Measures
  - *P* is the number of processor cycles needed to decode and execute the instruction, *m* is the number of the memory references needed, and *k* is the ratio between memory cycle time and processor cycle time, memory latency.

- With respect to the *CPU* time

$$T = I_c * CPI * \tau = I_c * (p+m*k)* \tau$$

in the following sections we will study two major issues:

- Design and implementation of ALU in an attempt to reduce *P*,
- Design and implementation of memory hierarchy in an attempt to reduce *m* and *k*.

- Question
  - With respect to our earlier definition of *CPU* time, discuss how the performance can be improved?

$$T = I_c * CPI * \tau = \sum_{i=1}^{n} \left(CPI_i * I_i\right) * \tau$$

- In response to this question, the *CPU* time can be reduced by reducing the I_C, *CPI*, and/or τ.
- Note the performance improvement with respect to the τ due to the advances in technology is beyond the scope of this discussion.