AA 372: Numerical & Statistical Techniques

Prateek Sharma (prateek@physics.iisc.ernet.in)
Office: D2-08
Office Hours: Fri. 2-3 pm (I expect to see you!)

Two parts

- Numerical Analysis by me at IISc

- Statistical Methods by Desh at RRI
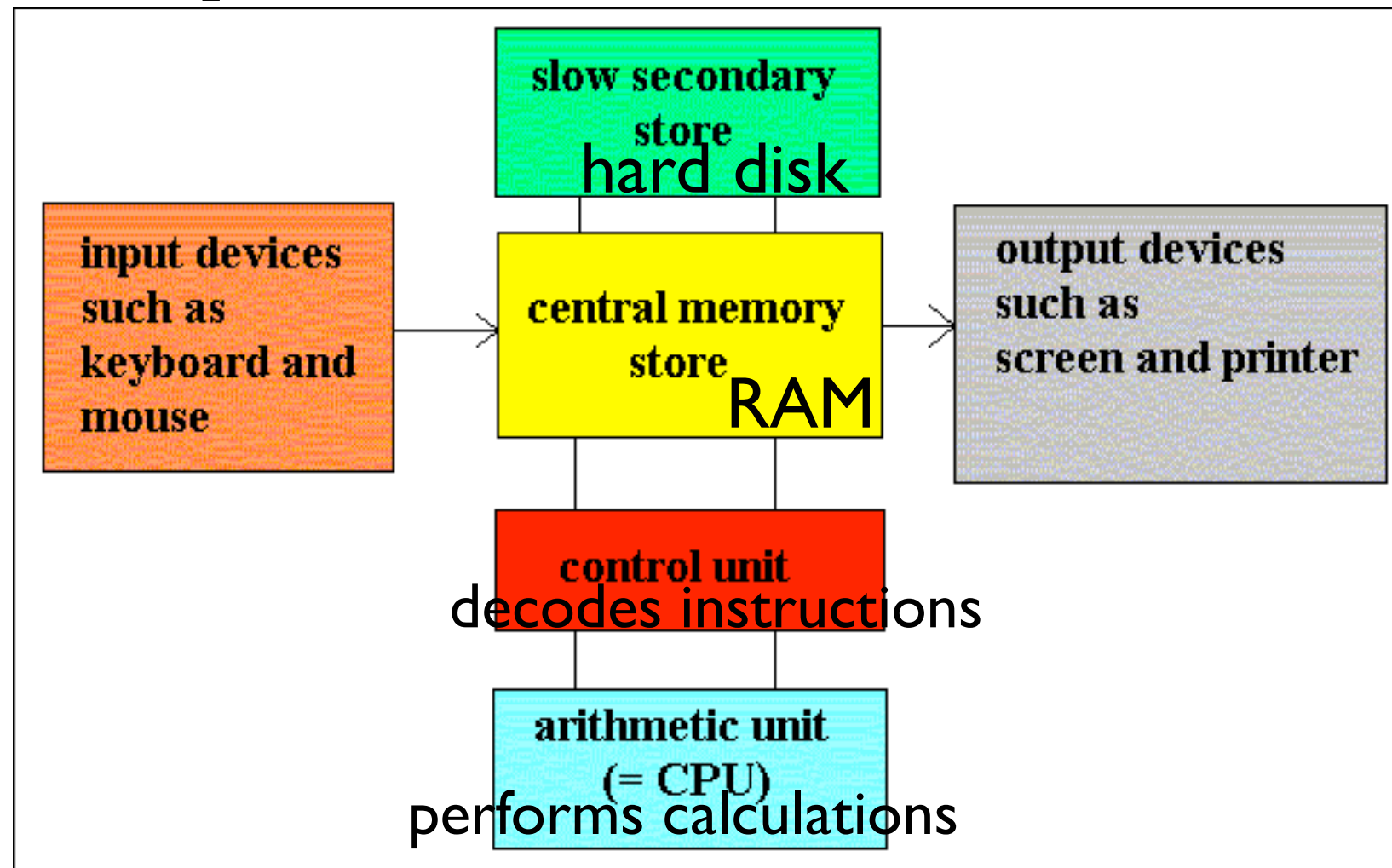
Organization

I've created a wikipage for this course:
http://ps-teaching.wikispaces.com/

I'll upload slides/problem-sets here
grading: weekly homeworks, project

Syllabus:
http://ps-teaching.wikispaces.com/AA+372+Syllabus

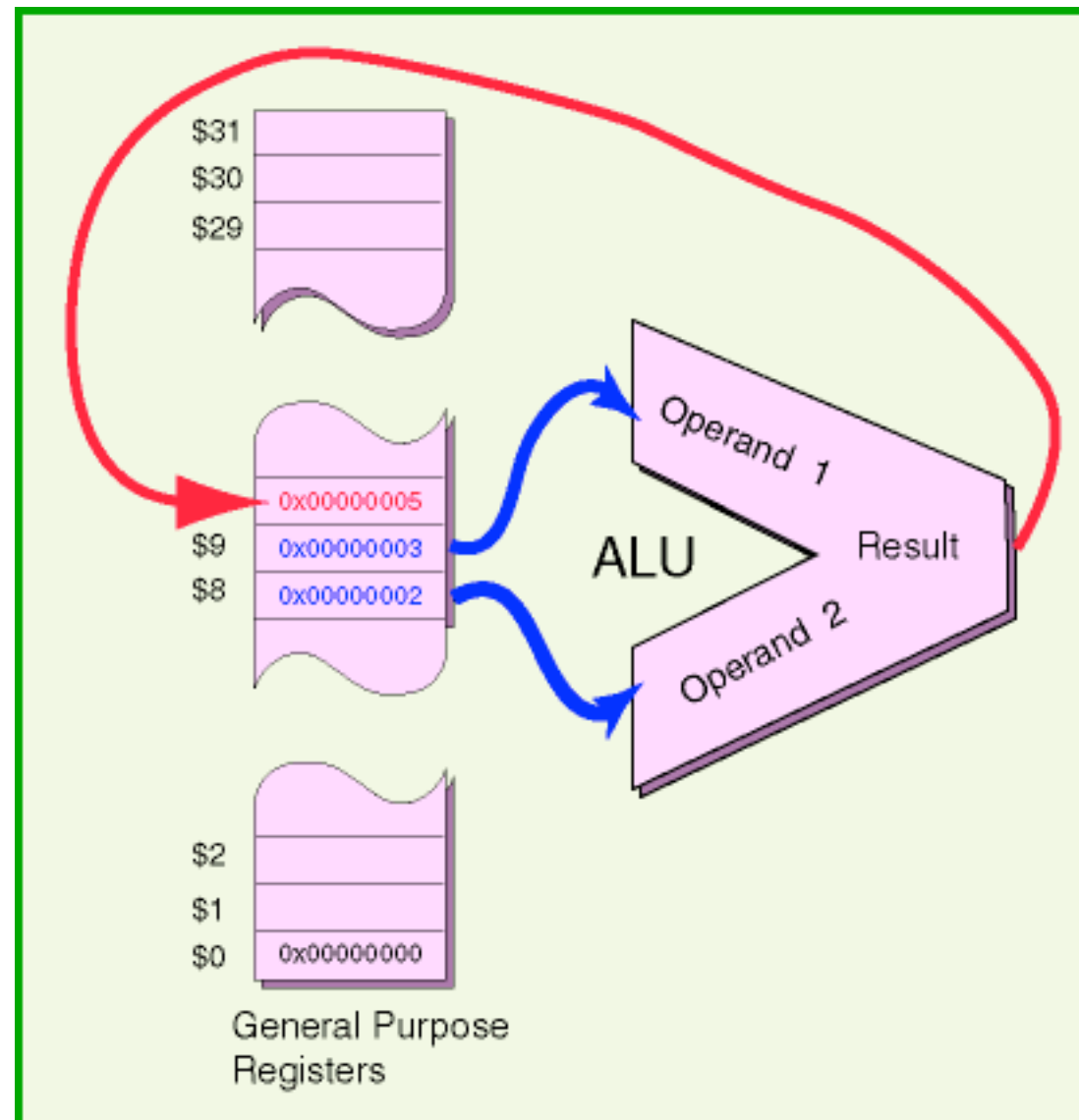some basics about how modern computers work

Computer Architecture

both instructions & data sent by input devices to memory
loaded from memory to CPU registers
Instruction Set Architecture (ISA): machine language instruction set,
word size, registers

ALU

bitwise logic ops.
AND, OR, NOT, XOR

integer arithmetic ops.
add, subtract, multiply, divide

bit shifting (* or / by 2^n)
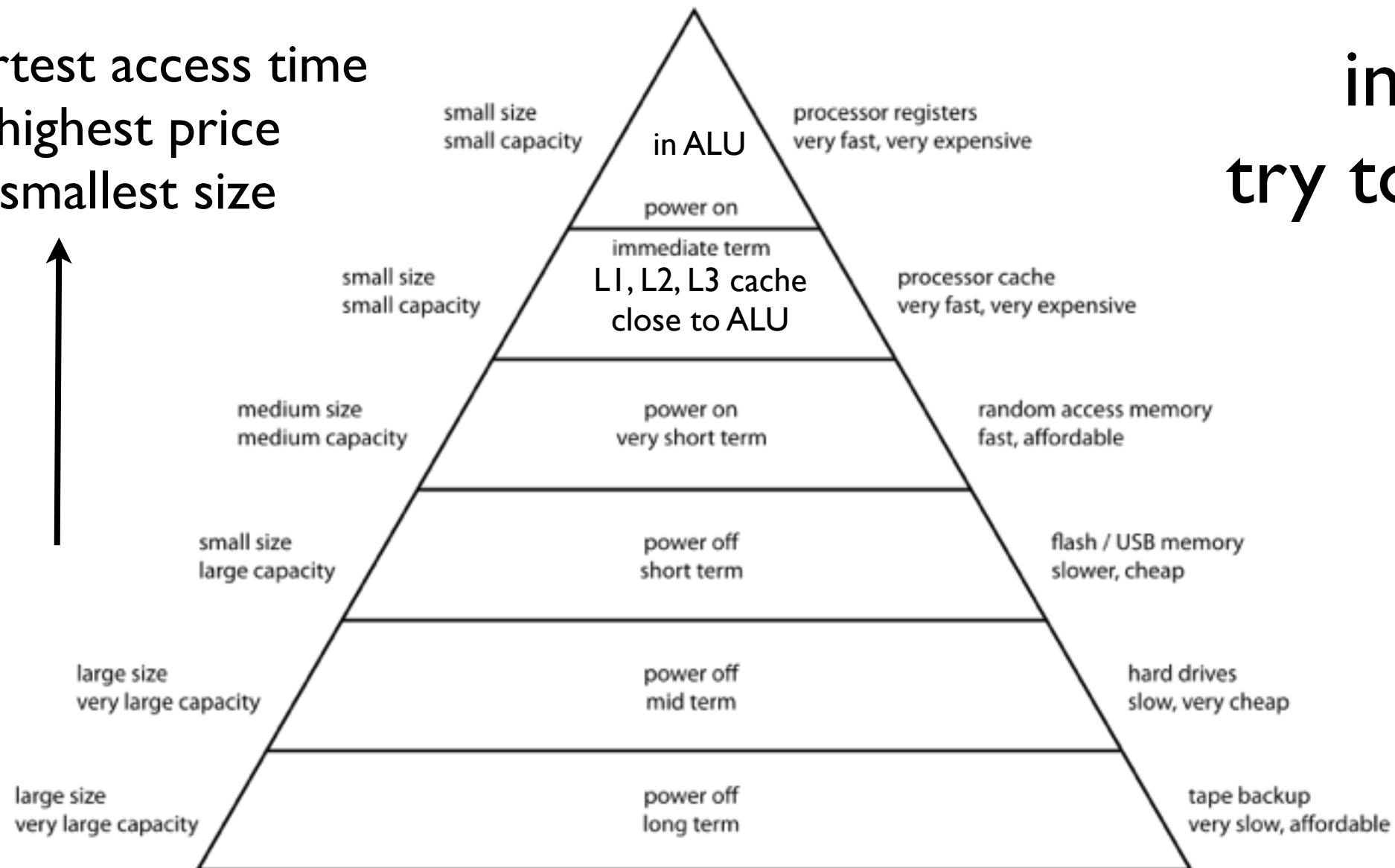
FPU: floating point unit

+,-,* fast / slow and so are exp, cos, & other transcendental fns.
commonly used function are coded in machine language
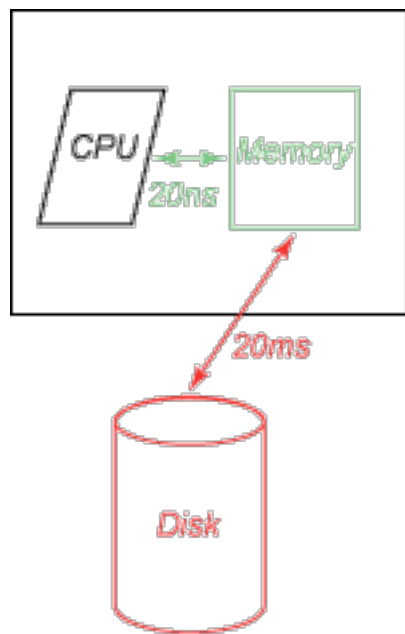
Hierarchical Memory

Computer Memory Hierarchy

shortest access time
highest price
smallest size

implication:
try to reuse cache

small size small capacity	in ALU	processor registers very fast, very expensive
	power on	
small size small capacity	immediate term L1, L2, L3 cache close to ALU	processor cache very fast, very expensive
medium size medium capacity	power on very short term	random access memory fast, affordable
small size large capacity	power off short term	flash / USB memory slower, cheap
large size very large capacity	power off mid term	hard drives slow, very cheap
large size very large capacity	power off long term	tape backup very slow, affordable

Cache Utilization

data stored in memory as a 1-D array

column major in fortran

$$\begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 6 & 2 & 5 & 3 & 4 \end{bmatrix}$$

row major in C!

```
do j = 1, jmax
do i = 1, imax
  a(i,j) = float(i-j)/float(i+j)
enddo
enddo
```
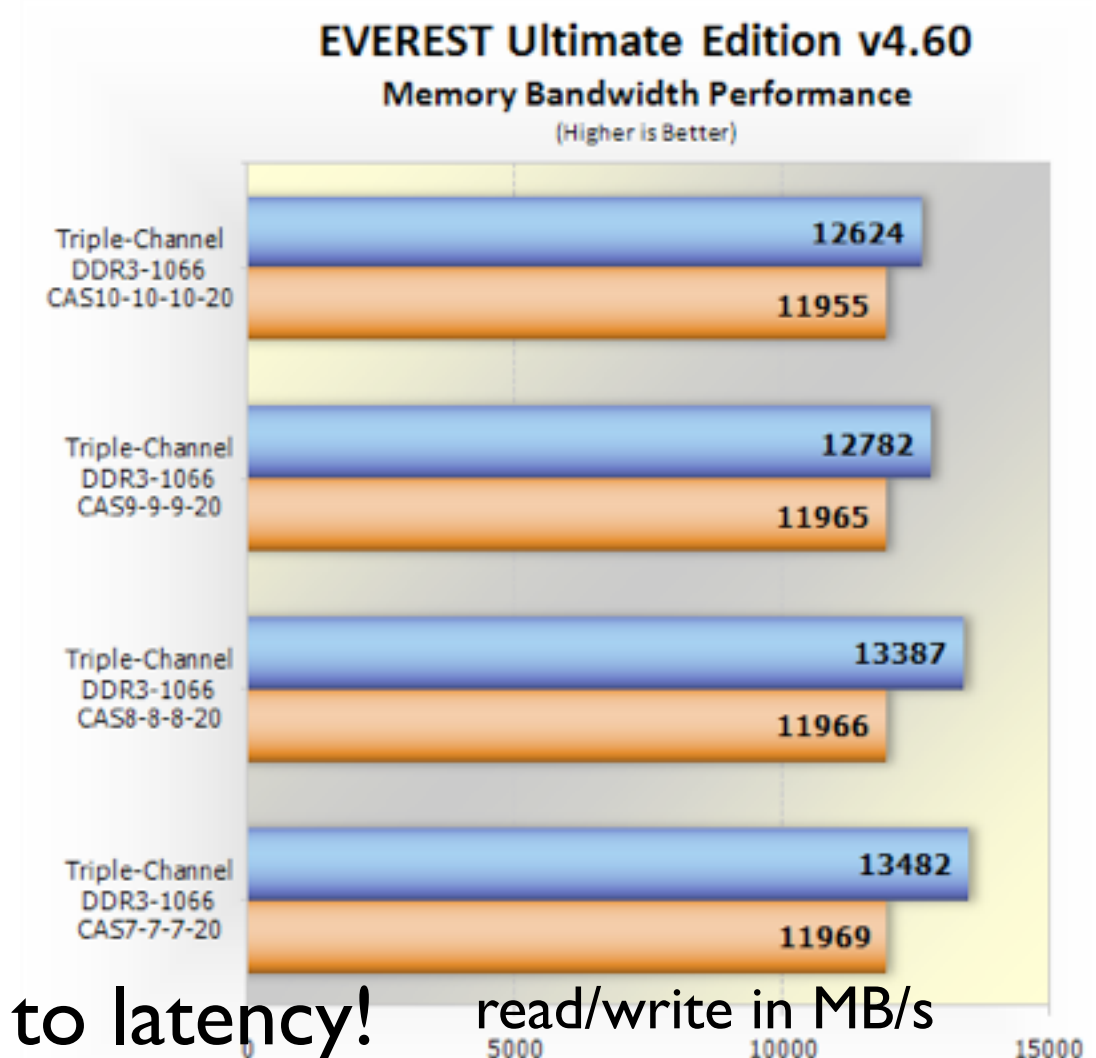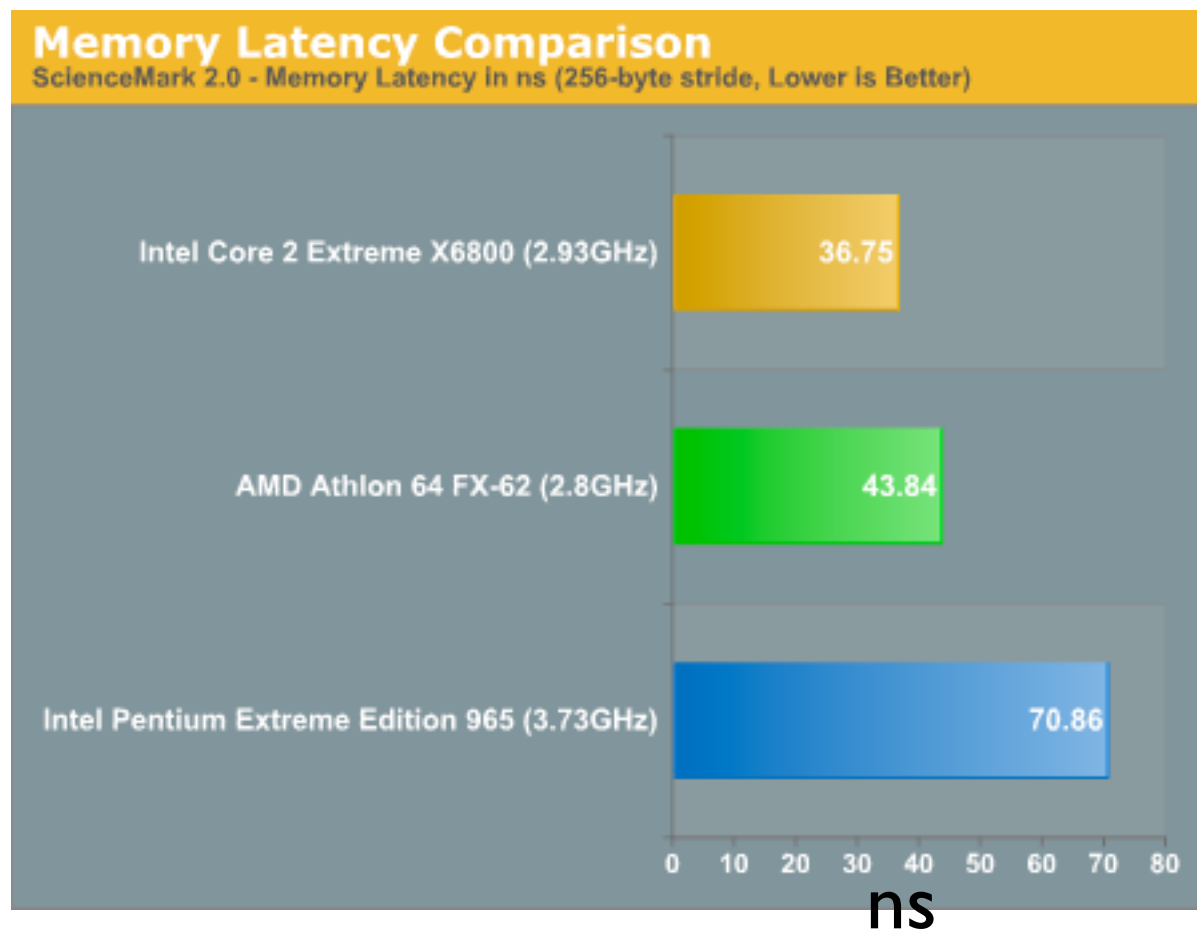
faster than

```
do i = 1, imax
do j = 1, jmax
  a(i,j) = float(i-j)/float(i+j)
enddo
enddo
```

sometimes compilers do these optimizations (-O3)

Latency & Bandwidth

minimum time to do an action
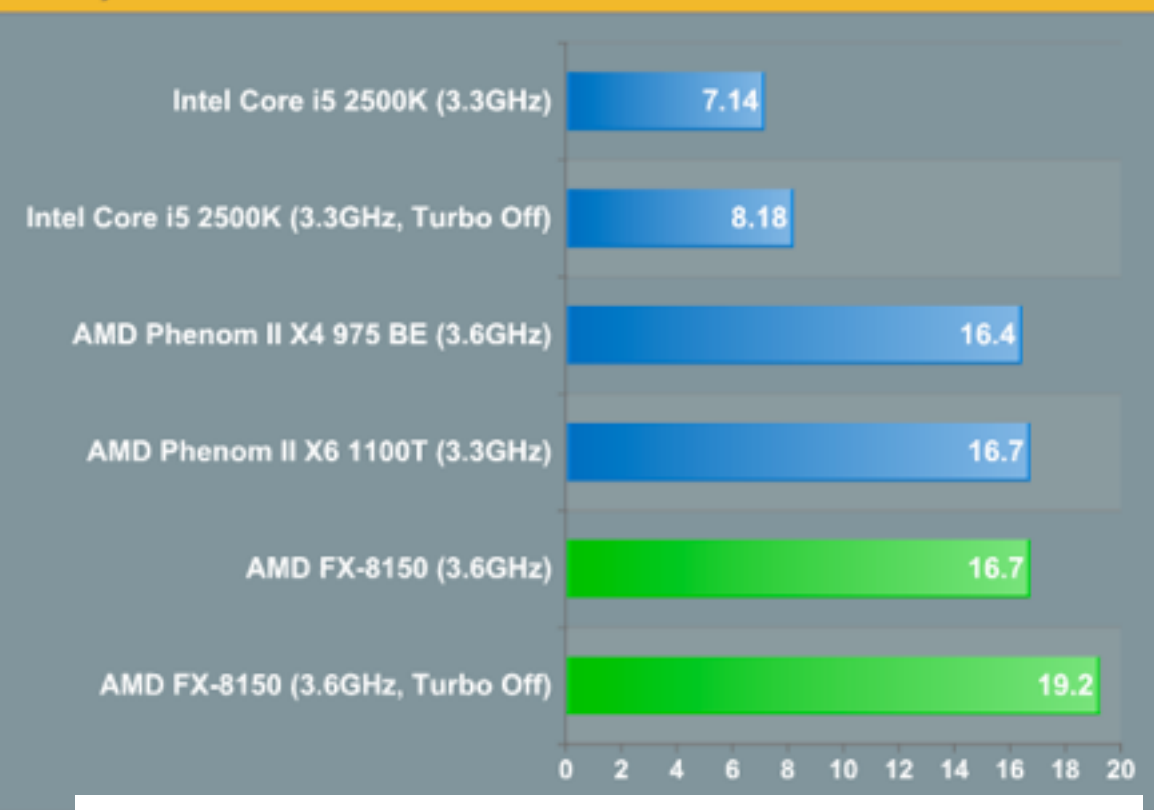(access time)
rate of action once action is initialized

Memory Latency Comparison
ScienceMark 2.0 - Memory Latency in ns (256-byte stride, Lower is Better)

	ns
Intel Core 2 Extreme X6800 (2.93GHz)	36.75
AMD Athlon 64 FX-62 (2.8GHz)	43.84
Intel Pentium Extreme Edition 965 (3.73GHz)	70.86

EVEREST Ultimate Edition v4.60
Memory Bandwidth Performance
(Higher is Better)

Triple-Channel DDR3-1066 CAS10-10-10-20	12624	11955
Triple-Channel DDR3-1066 CAS9-9-9-20	12782	11965
Triple-Channel DDR3-1066 CAS8-8-8-20	13387	11966
Triple-Channel DDR3-1066 CAS7-7-7-20	13482	11969

read/write in MB/s
5000   10000   15000

better to minimize memory access due to latency!

L3 Cache Latency
Latency in ns - Lower is Better

Processor	Latency (ns)
Intel Core i5 2500K (3.3GHz)	7.14
Intel Core i5 2500K (3.3GHz, Turbo Off)	8.18
AMD Phenom II X4 975 BE (3.6GHz)	16.4
AMD Phenom II X6 1100T (3.3GHz)	16.7
AMD FX-8150 (3.6GHz)	16.7
AMD FX-8150 (3.6GHz, Turbo Off)	19.2

Memory Latency Comparison
ScienceMark 2.0 - Memory Latency in ns (256-byte stride, Lower is Better)

Processor	Latency (ns)
Intel Core 2 Extreme X6800 (2.93GHz)	36.75
AMD Athlon 64 FX-62 (2.8GHz)	43.84
Intel Pentium Extreme Edition 965 (3.73GHz)	70.86
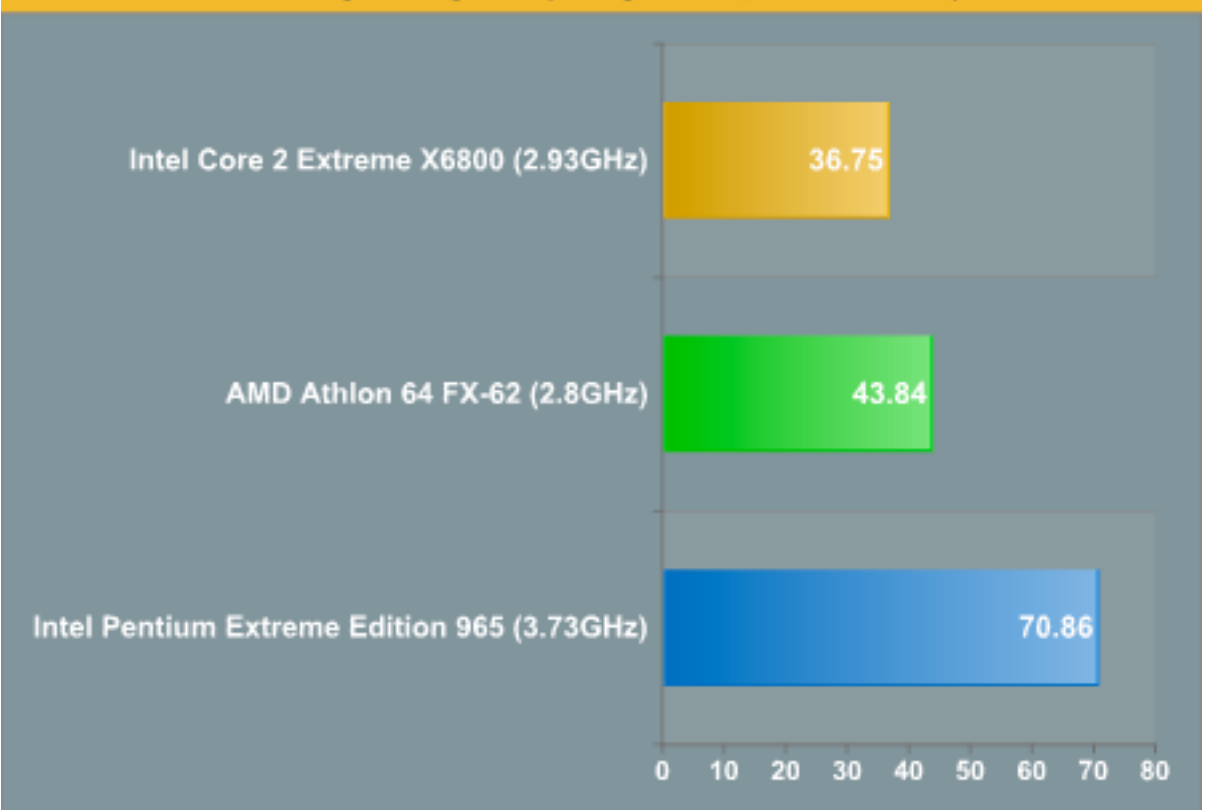
ScienceMark L1 Cache Latency: Time

Processor	Latency in Nanoseconds
PIII 1000	2.98
Cel 1200	2.48
Cel 1600	1.87
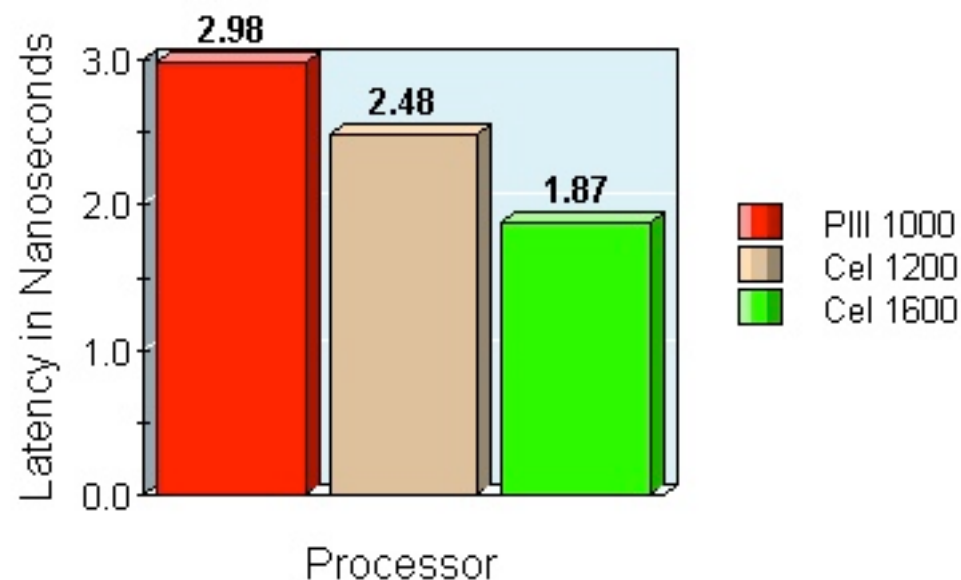
Nehalem processor:
L1~64 kB
L2~2 MB
L3~30 MB

L1 cache ~ 5 times faster than L3 cache ~5 times faster than RAM!

Clock Rate

clock coordinates
different actions
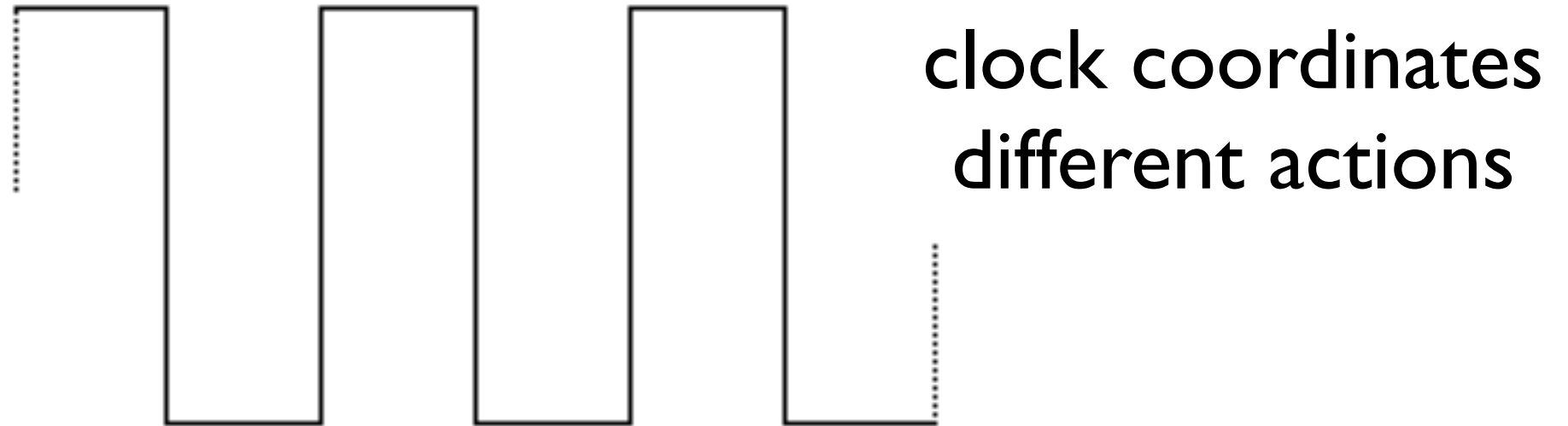
modern CPUs upto 4 FLOPs per cycle:
2.4 GHz => 4x2.4 10^9 ~10^{10} FLOPs/cycle/core (10 GF)
if the cluster has 80 cores => 800 GF machine

this is not the only parameter! since data access is more
time-consuming (40 ns) than FLOPs (0.1 ns); having larger RAM/
cache/interconnects more important than just clock speed

Architecture level Parallelism

bit level parallelism: 4 bit ... 32 to 64 bit word-size (=register size); more bits processed/cycle

instruction level parallelism

five-stage pipeline in a RISC (IF = Instruction Fetch, ID = Instruction Decode, EX = Execute, MEM = Memory access, WB = Register write back)

Moore's Law

saturation of clock-speed, power efficiency, ILP
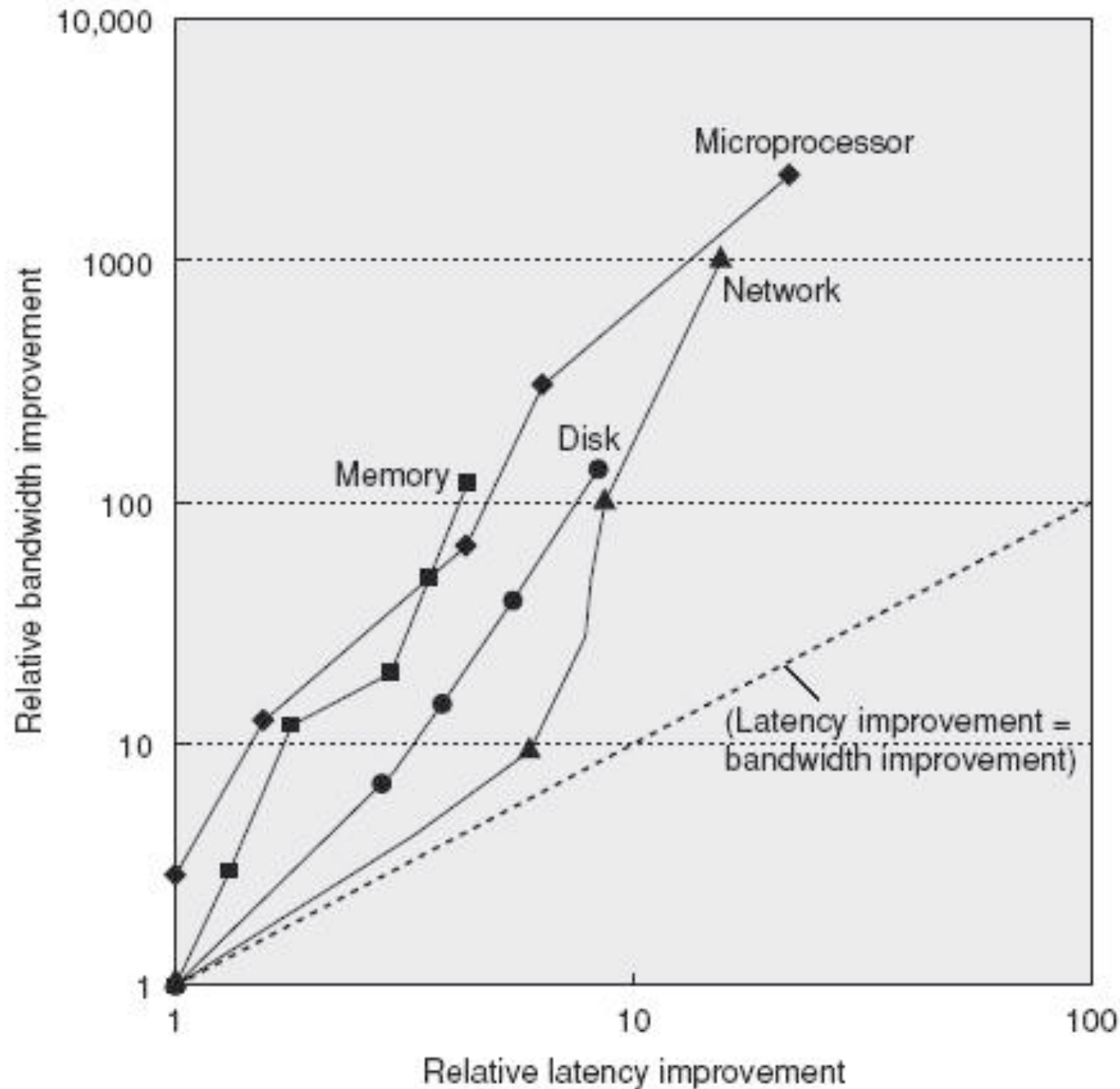
=> paradigm shift

multiple processors/chip rather than faster processors

thread/data level parallelism requires programming (MPI,openMP) unlike ILP

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)

Dual-Core Itanium 2

Pentium 4

Pentium

386

Transistors (000)
Clock Speed (MHz)
Power (W)
Perf/Clock (ILP)
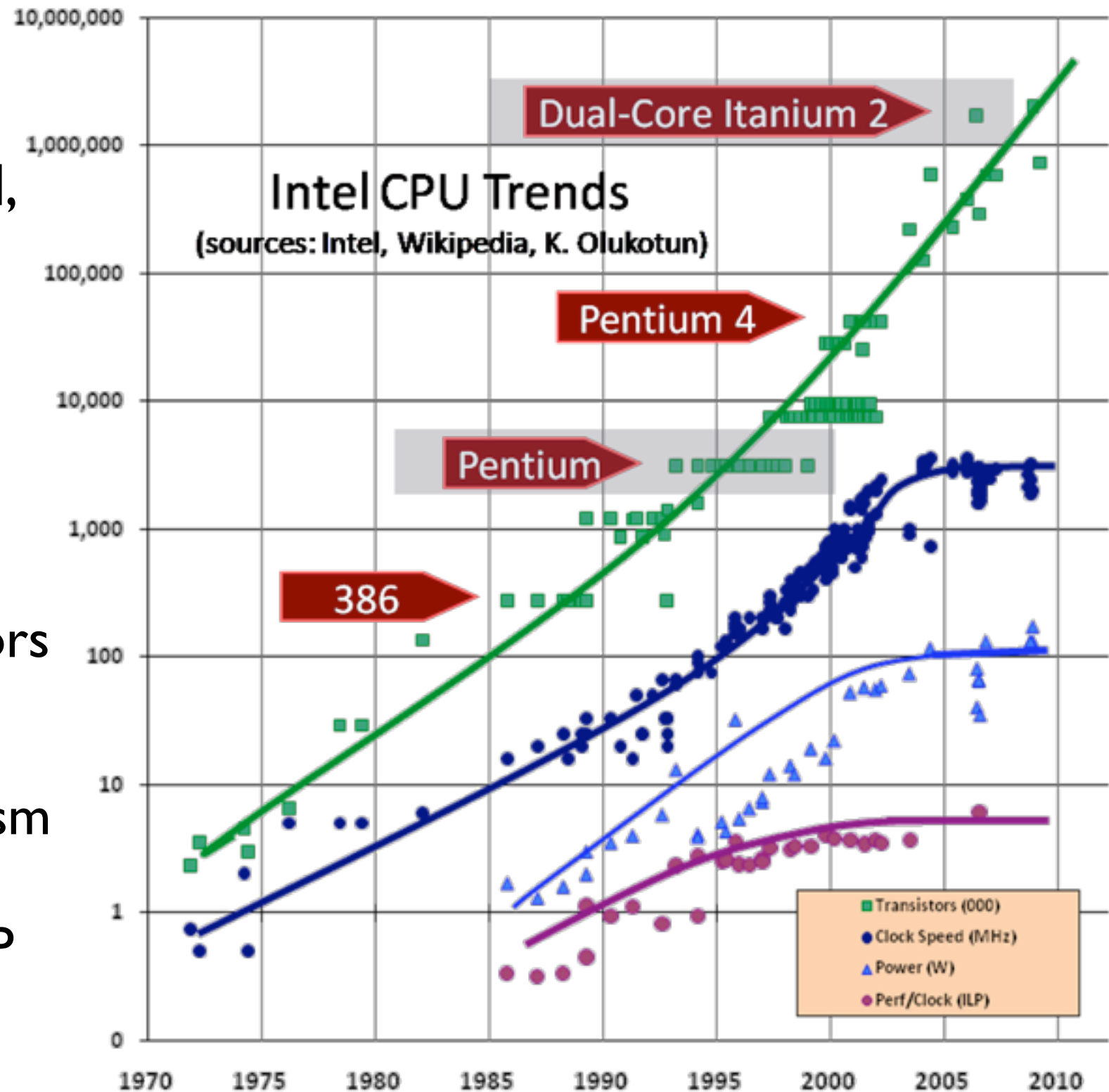
improvements governed by technology

architecture, compiler, programs reflect this

power issues!
chips becoming smaller and smaller

$$P = 1/2\ C\,V^2 f$$

higher frequency => more power
consumption & heating
can't be air cooled!
reduce operating voltage (transistor errors),
frequency (speed reduction)

software closely tied to hardware

esp. with parallel systems

source code: high level language (fortran, c, c++)

compiler (also optimizes the code, e.g., -O2, -O3 flags)

object code & executable (lower level assembly/machine code)

interpreted languages (e.g., python, perl, MATLAB, Mathematica, IDL scripting languages) slower but handy/easier

important to remember architecture to attain maximum performance

Parallel Computing

multicore: multiple processors on the same chip

shared memory (SMP): all processors have common main memory

Distributed memory: beowulf cluster, parallel clusters w. specialized interconnects

Grid computing: computers communicating over the internet; e.g., SETI@home

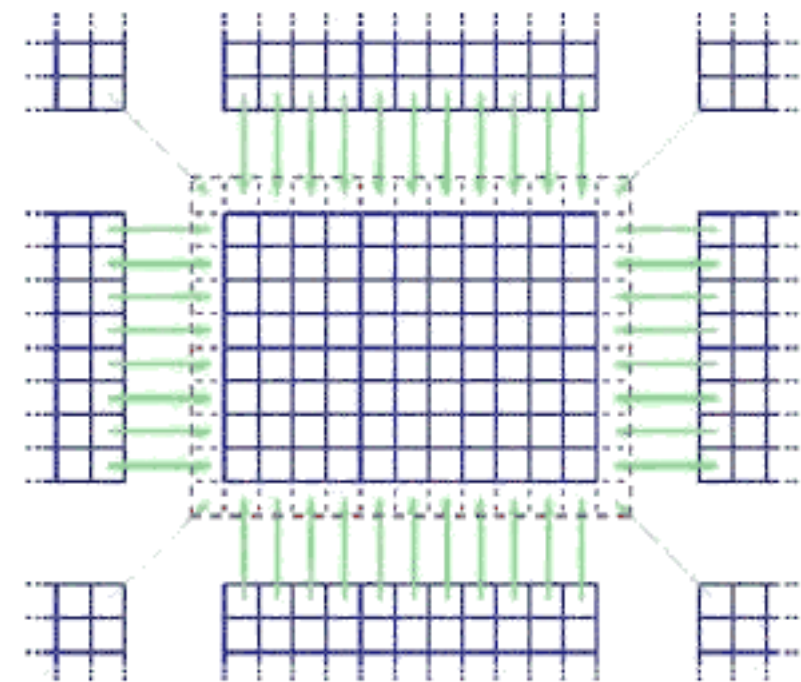GPUs (graphics processing units): driven by games/graphics industry, fast FP operations

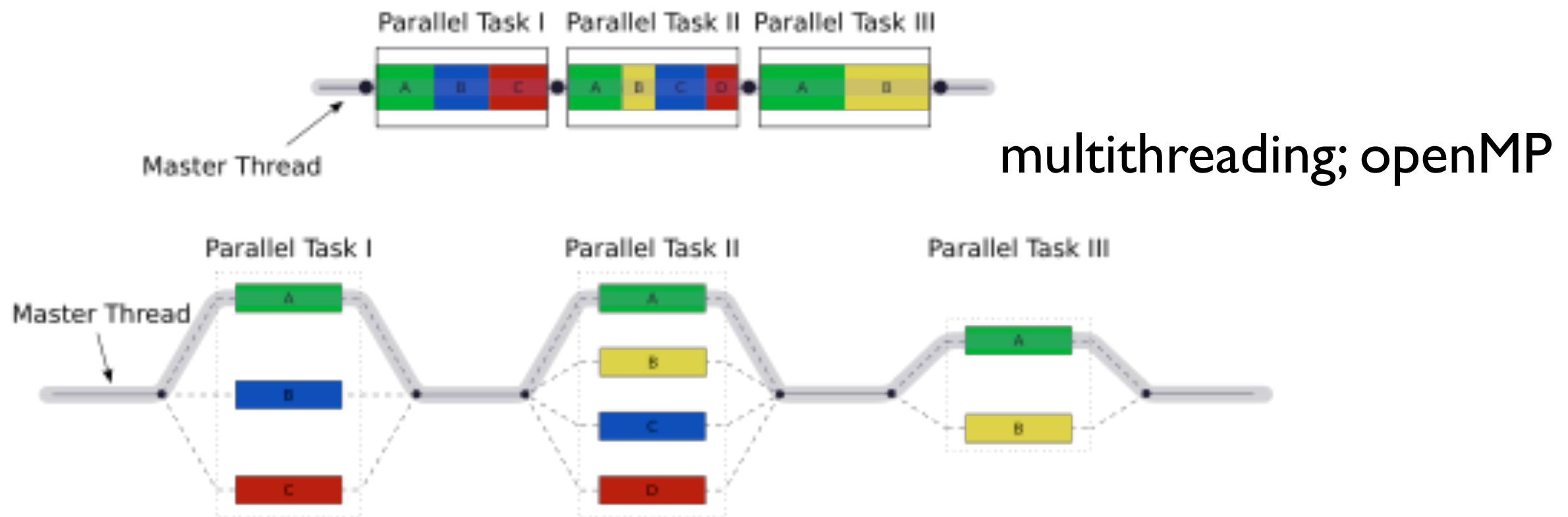Software: MPI (message passing interface; distributed systems), openMP (shared memory)

distributed memory programming model

message passing; MPI
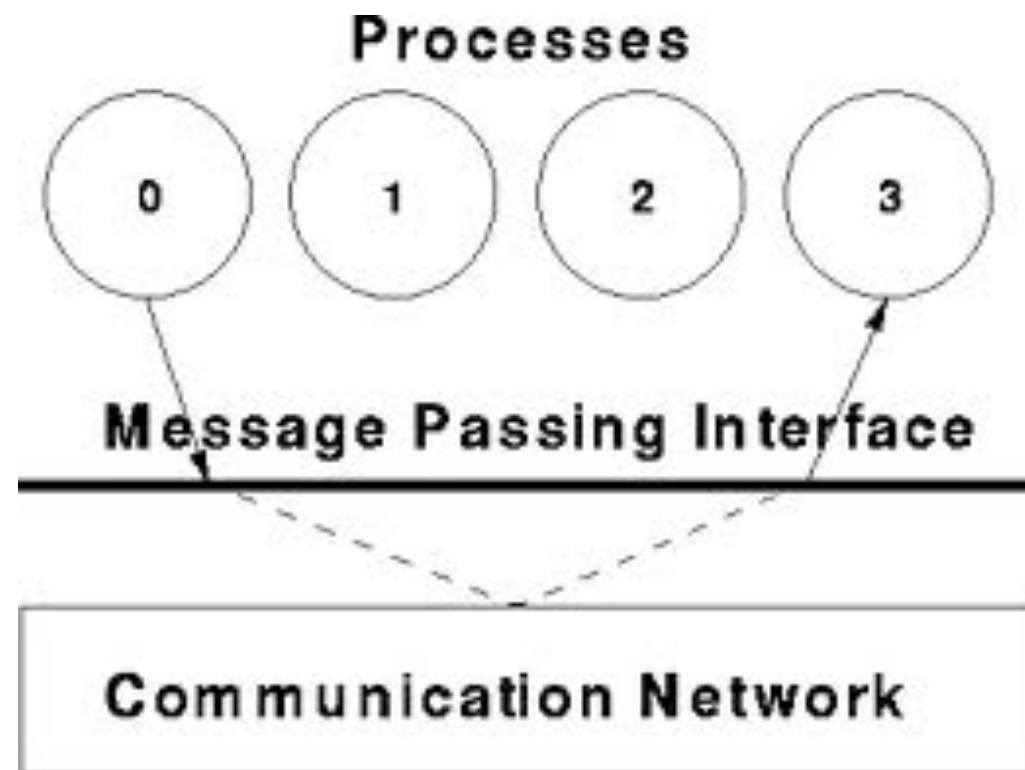
shared memory programming model
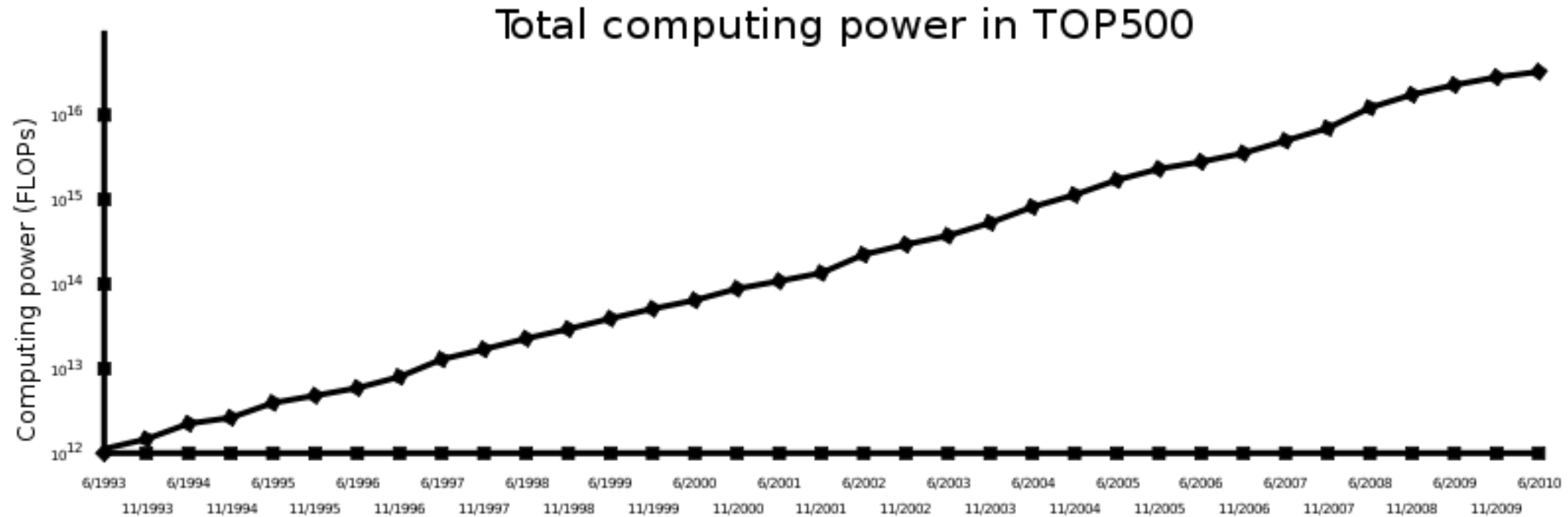

multithreading; openMP

Trends in Supercomputing

http://top500.org/:

list of 500 fastest (based on LINPACK benchmark) computers in the world

shows trends in architecture, interconnects, vendors, etc.

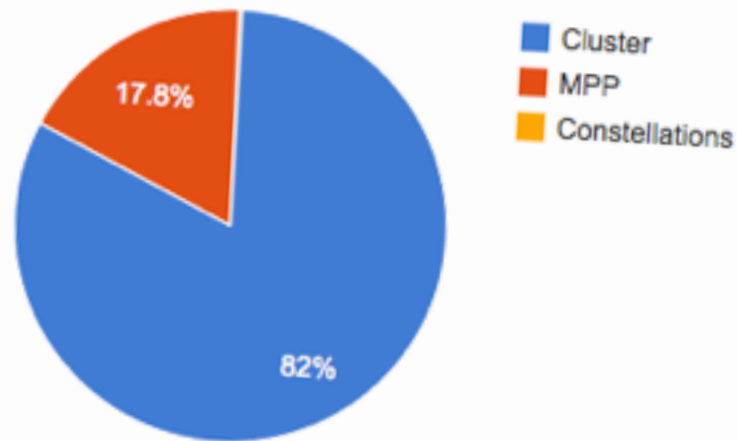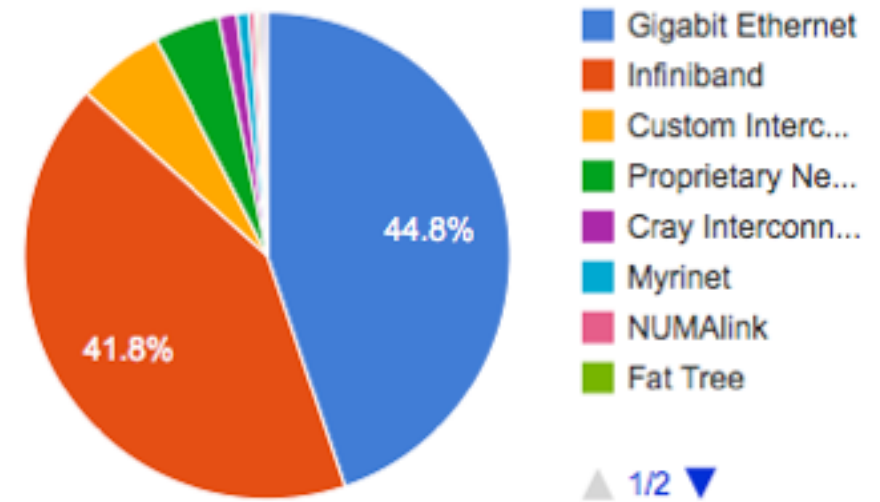Total computing power in TOP500

the list

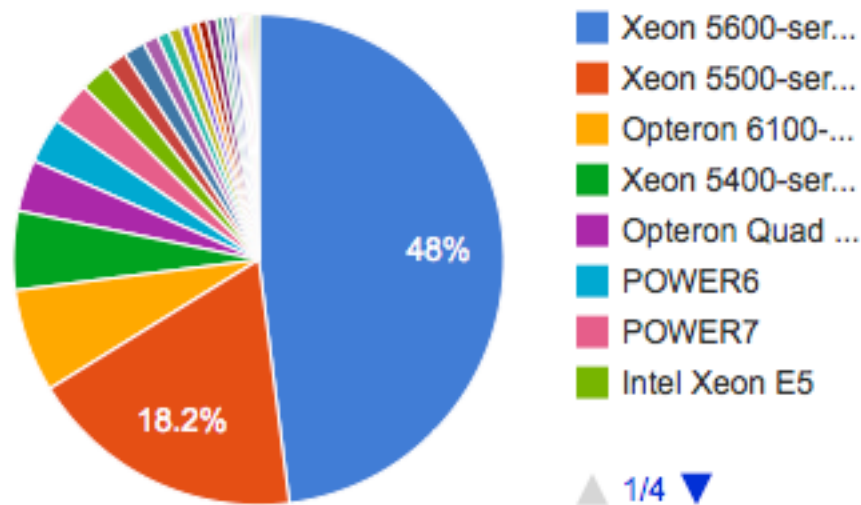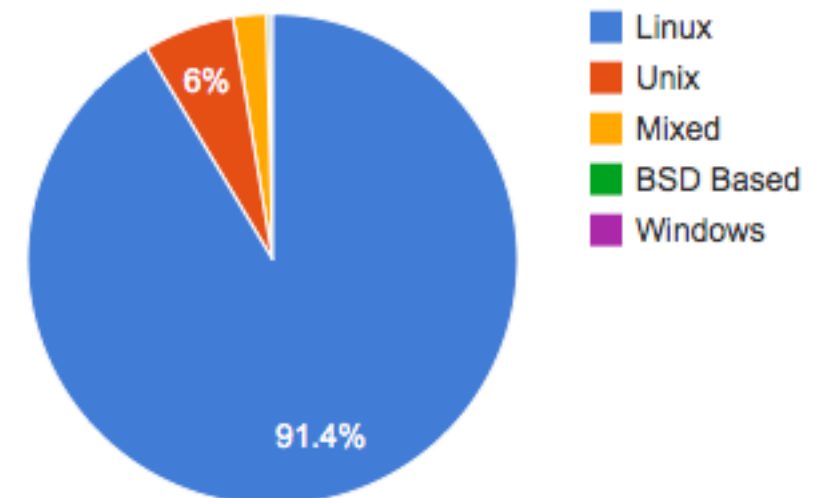| Rank	Site	Computer
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT
3	DOE/SC/Oak Ridge National Laboratory United States	Cray XT5-HE Opteron 6-core 2.6 GHz Cray Inc.
4	National Supercomputing Centre in Shenzhen (NSCS) China	Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 Dawning
5	GSIC Center, Tokyo Institute of Technology Japan	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows NEC/HP
6	DOE/NNSA/LANL/SNL United States	Cray XE6, Opteron 6136 8C 2.40GHz, Custom Cray Inc.
7	NASA/Ames Research Center/NAS United States	SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon 5570/5670 2.93 Ghz, Infiniband SGI
8	DOE/SC/LBNL/NERSC United States	Cray XE6, Opteron 6172 12C 2.10GHz, Custom Cray Inc.
9	Commissariat a l'Energie Atomique (CEA) France	Bull bullx super-node S6010/S6030 Bull
10	DOE/NNSA/LANL United States	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband IBM

Some Statistics

Architecture Share Over Time
1993 - 2011

Performance Share / TOP500 Releases

Legend: SIMD, Constellations, MPP, Cluster, Single Processor, SMP, Others
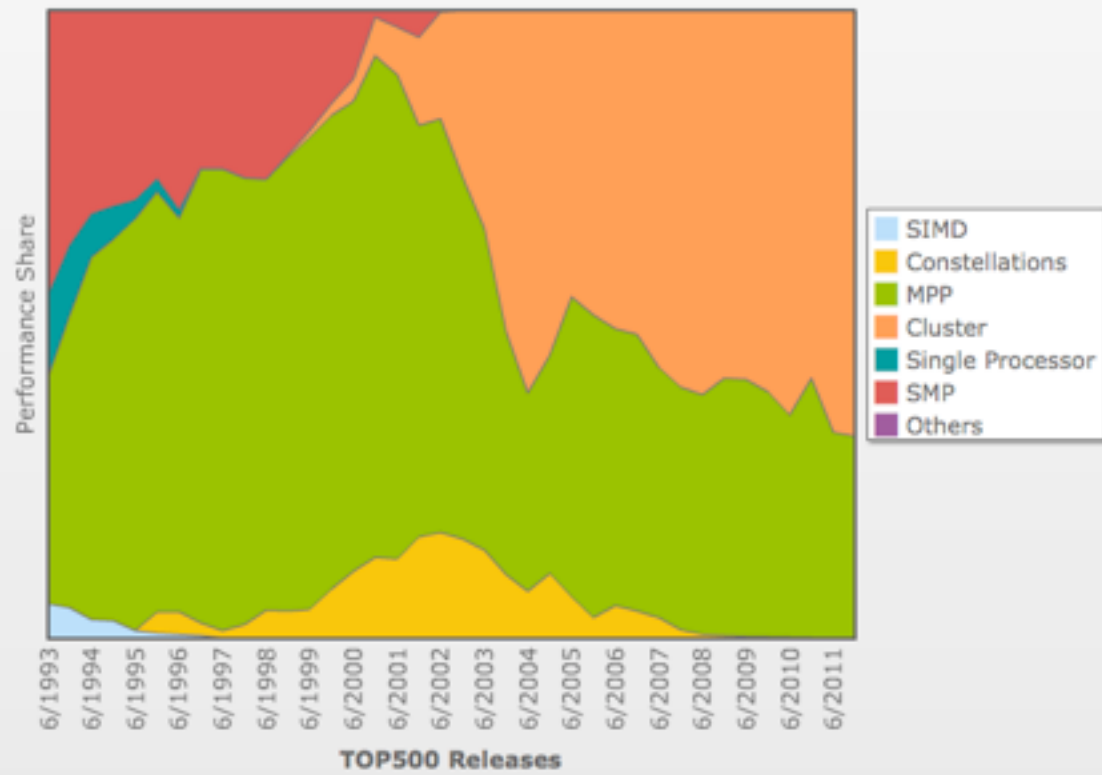
Interconnect Family Share Over Time
1993 - 2011

Performance Share / TOP500 Releases

Legend: N/A, Myrinet, Quadrics, Gigabit Ethernet, Infiniband, Crossbar, Mixed Network, NUMAlink, SP Switch, Proprietary Network, Cray Interconnect, Fat Tree, Custom Interconnect, Others
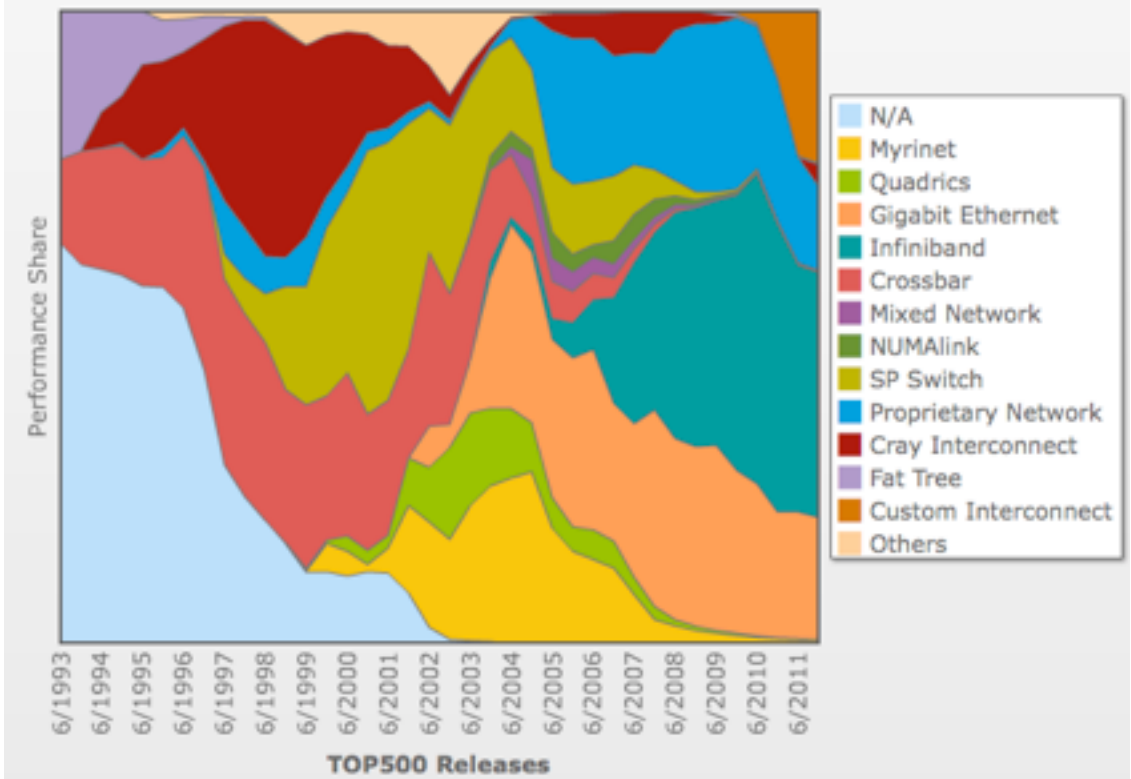
Processor Family Share Over Time
1993 - 2011

Performance Share / TOP500 Releases

Legend: TMC CM2, Hitachi SR8000, Power, Cray, Alpha, Fujitsu, Hitachi, PA-RISC, Intel IA-32, MIPS, NEC, Sparc, Intel IA-64, Intel EM64T, Intel i860, AMD x86_64, Intel Core, Intel Nehalem, Others
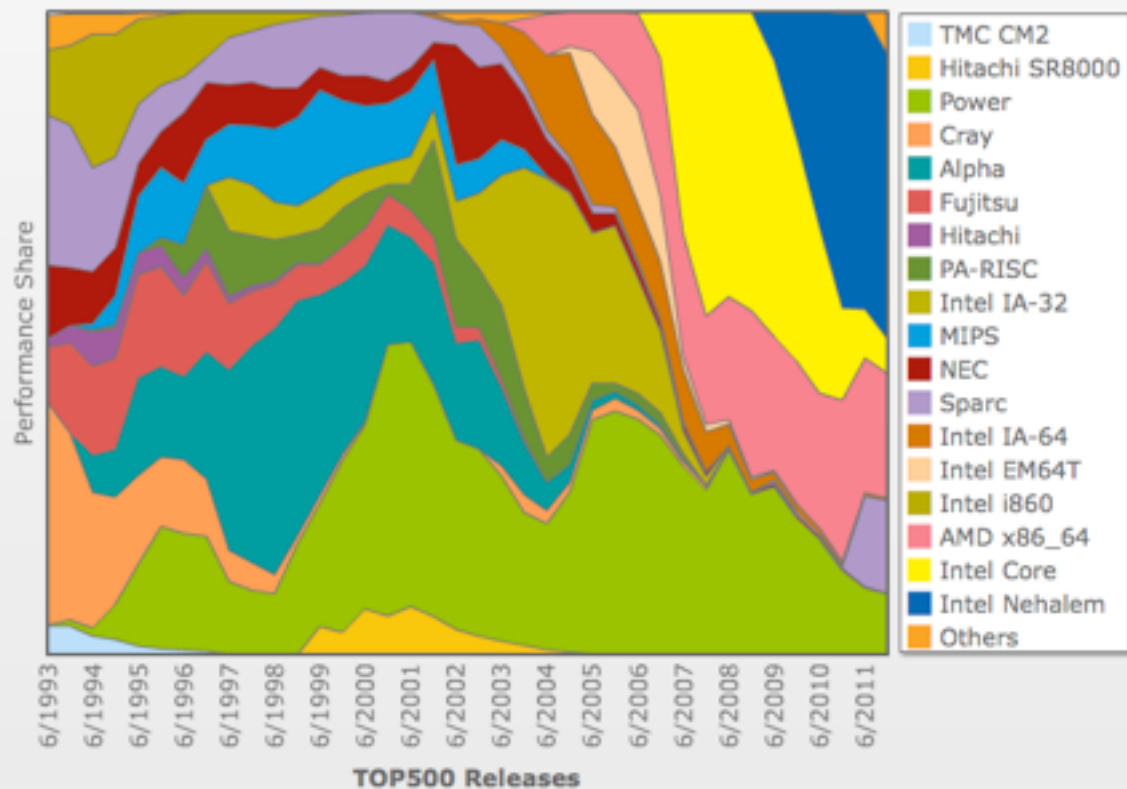
Countries Share Over Time
1993 - 2011

Performance Share / TOP500 Releases

Legend: Canada, Italy, France, China, Germany, Spain, United States, Japan, Switzerland, Russia, Sweden, Korea, South, United Kingdom, Others
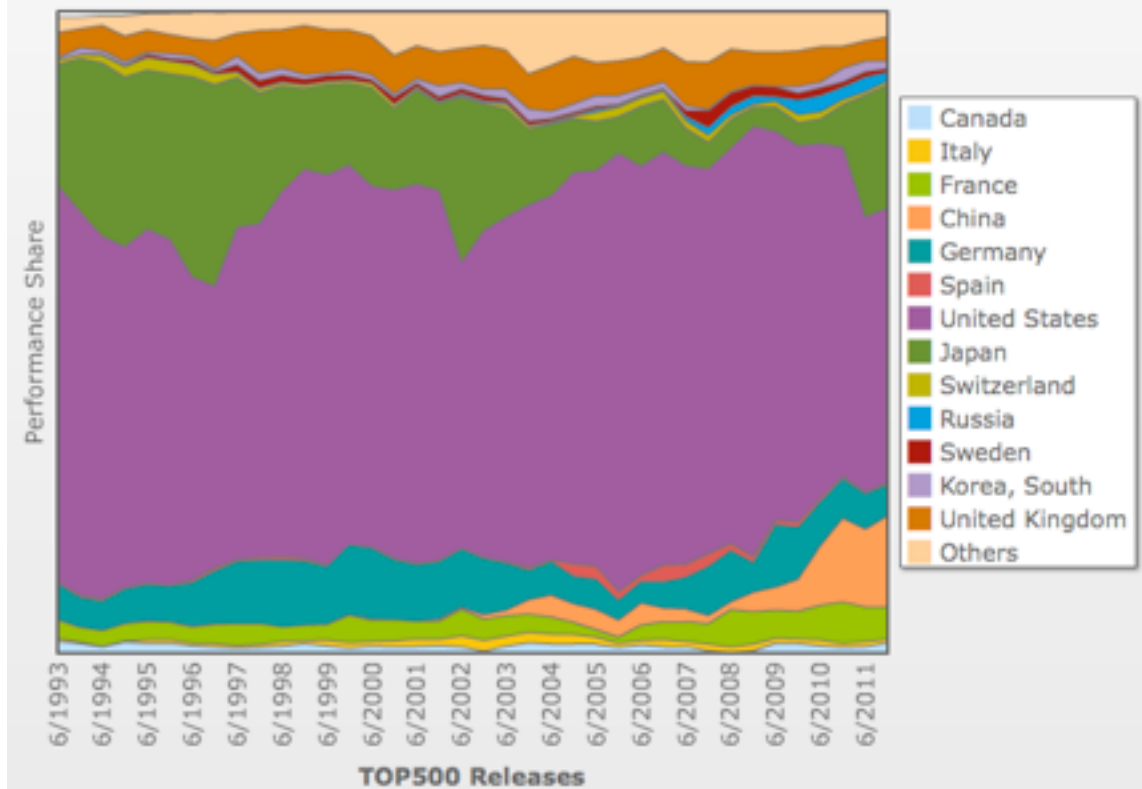
testing performance

strong scaling: increase the no. of procs. on a fixed problem-size

poor scaling for small problem size communication time >> computation time remember latency?

computation ~ N^3, communication~N^2

going to bigger problem size helps with communication overhead

weak scaling: keep the problem per processor the same and inc. the problem-size (& processor count)
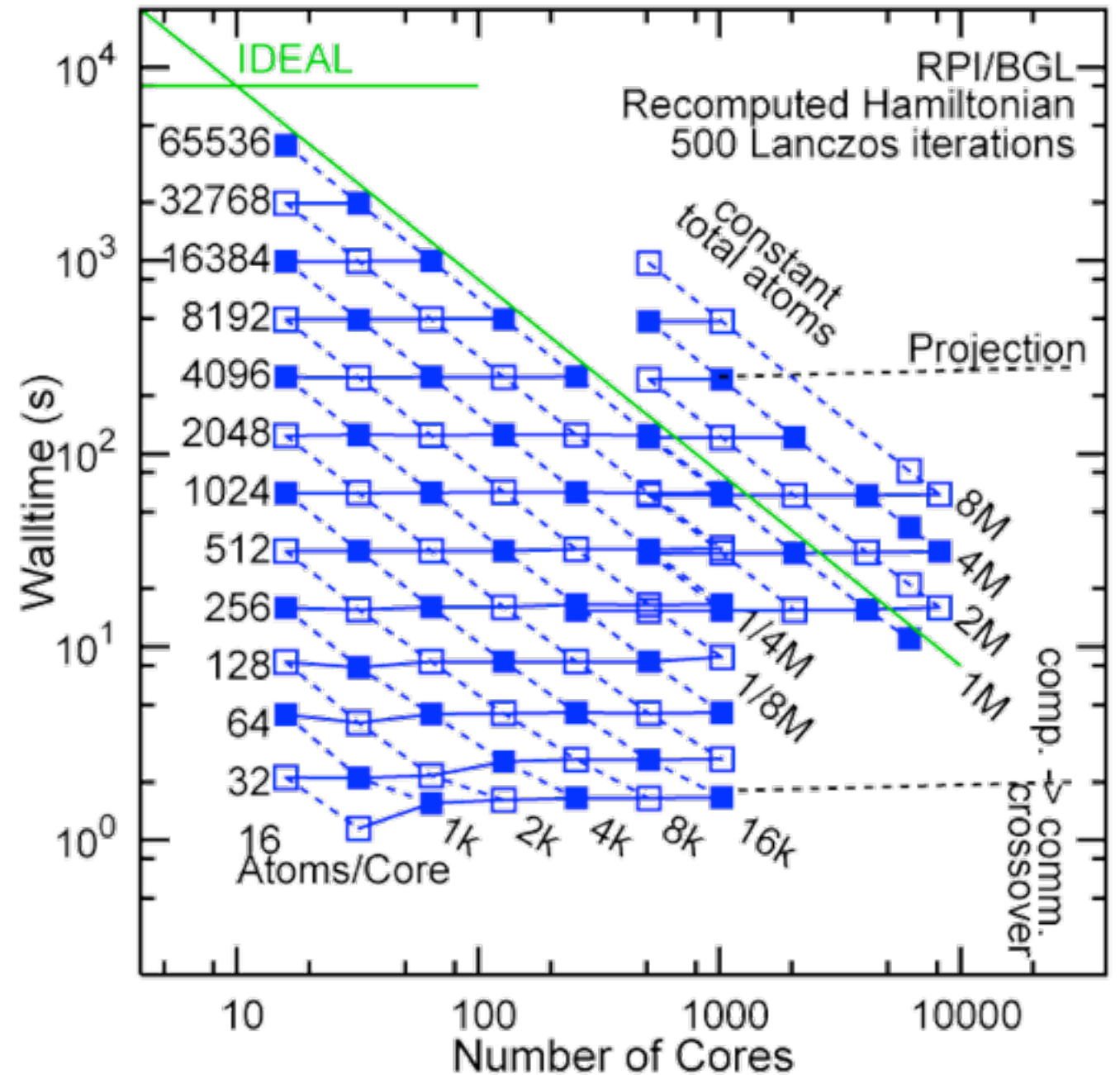
Summary

- just touched the tip of the iceberg

- lot of info online

- modern computer architecture: communication >> computation => cache contiguous data, minimize data access

- parallel systems: programming models

- free tools (e.g., LAPACK) online; no need to reinvent the wheel; good to know basics