PH 354: Computational Physics

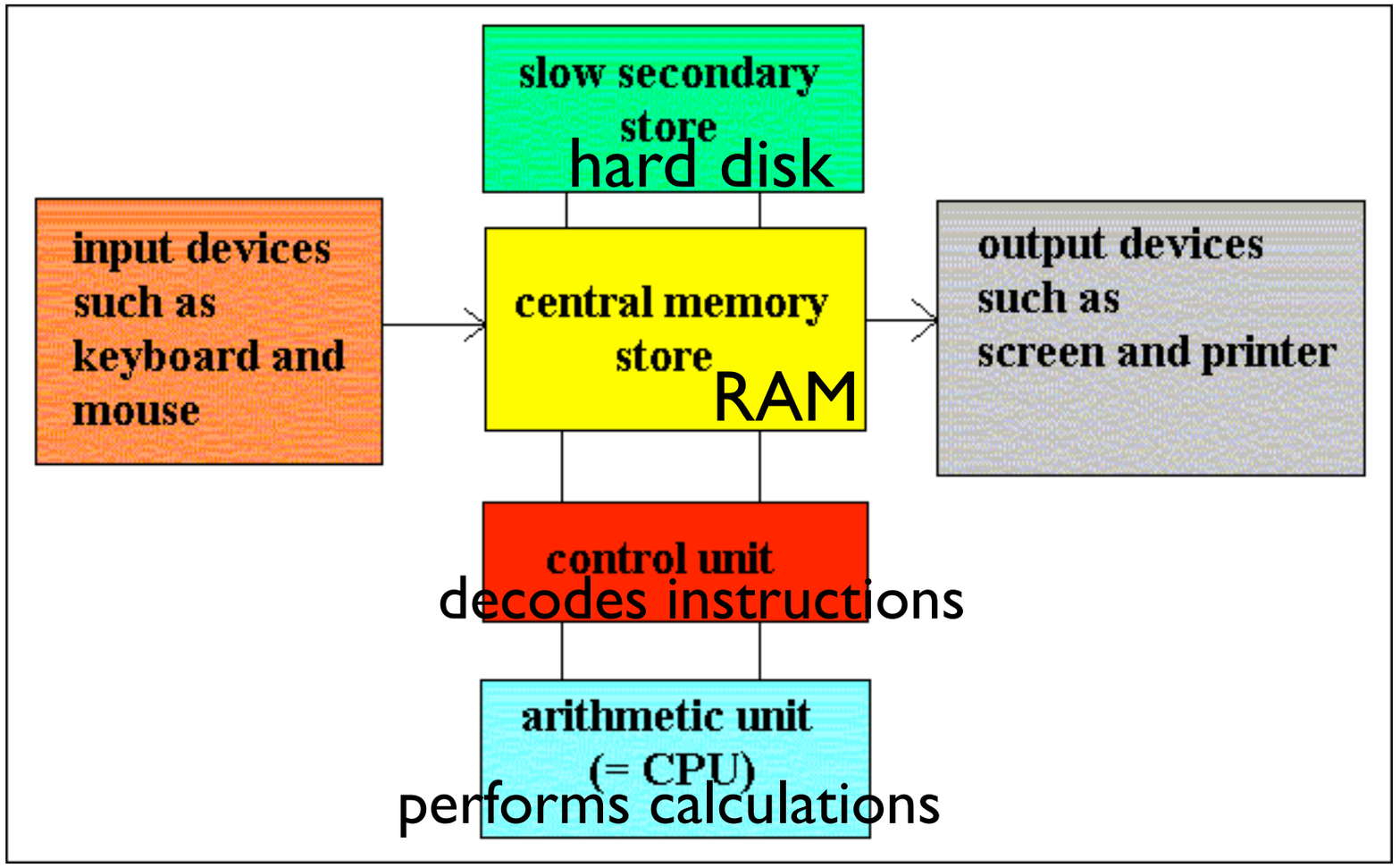
Instructors:
Prateek Sharma (prateek@iisc.ac.in)
Manish Jain (mjain@iisc.ac.in)
TA:
Sanat Gogoi (sanatgogoi@iisc.ac.in)

Philosophy & Approach

- Getting used to write code
- Practise, practise & practise more
- Good programming practices
- Algorithms for Physical applications: root-finding, linear algebra, ODEs, PDEs, etc.
- Interpreted language with little overhead
  Python (similar to MATLAB but free & widespread)
- No exams. Heavy HWs & course-project.
  Submission via **GitHub**

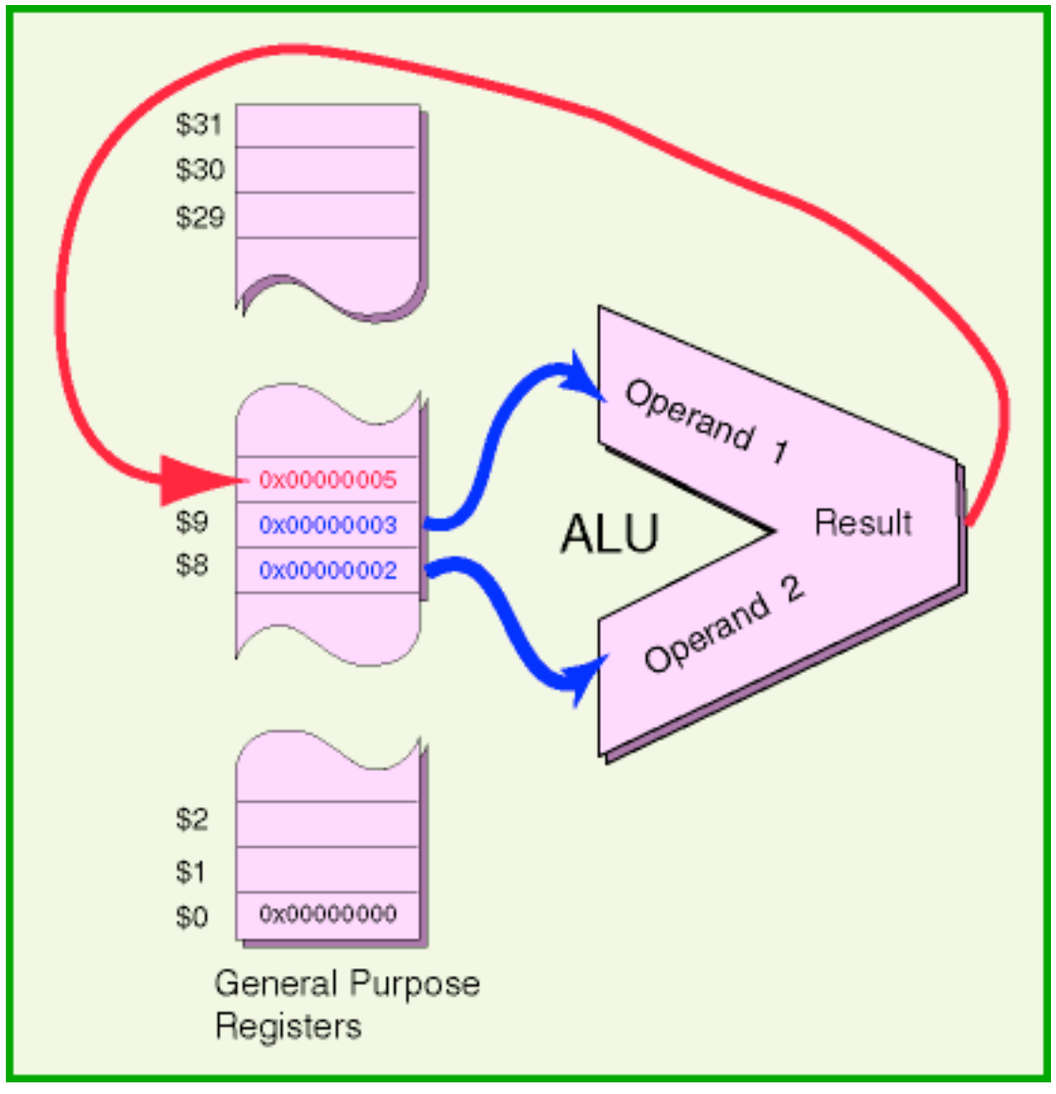
some basics about how modern computers work

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

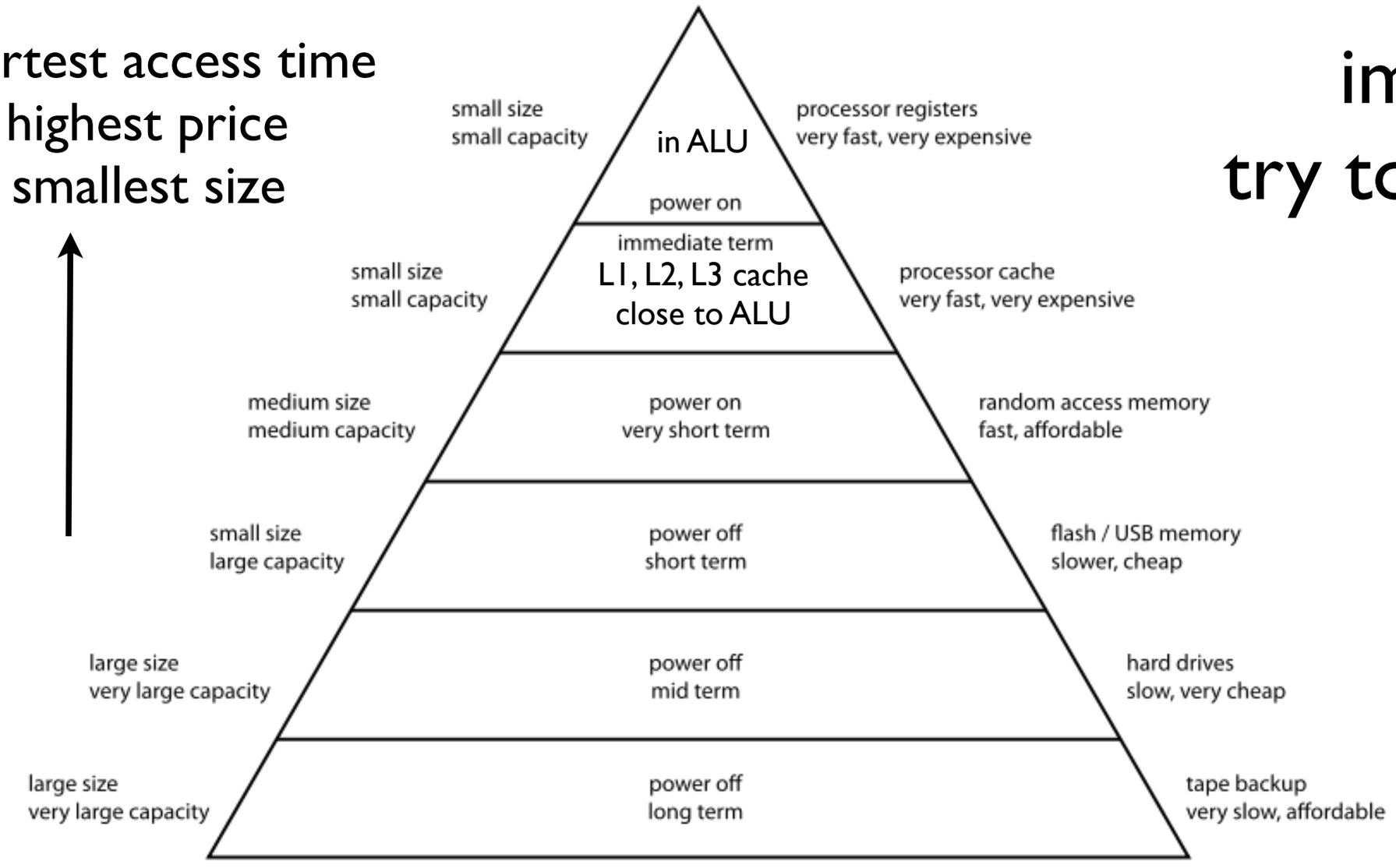
integer arithmetic ops.
add,subtract,multiply,divide

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

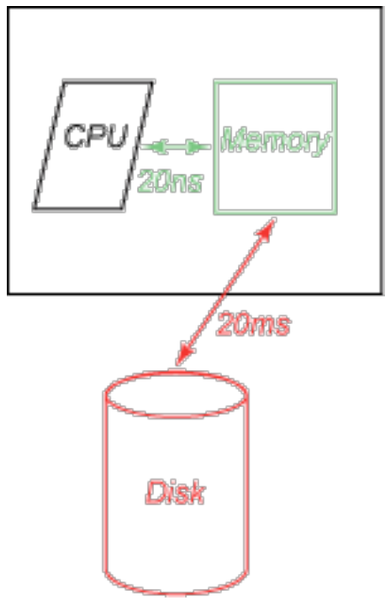
Hierarchical Memory



implication:
try to reuse cache

Cache Utilization

data stored in memory as a 1-D array



column major in fortran

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

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

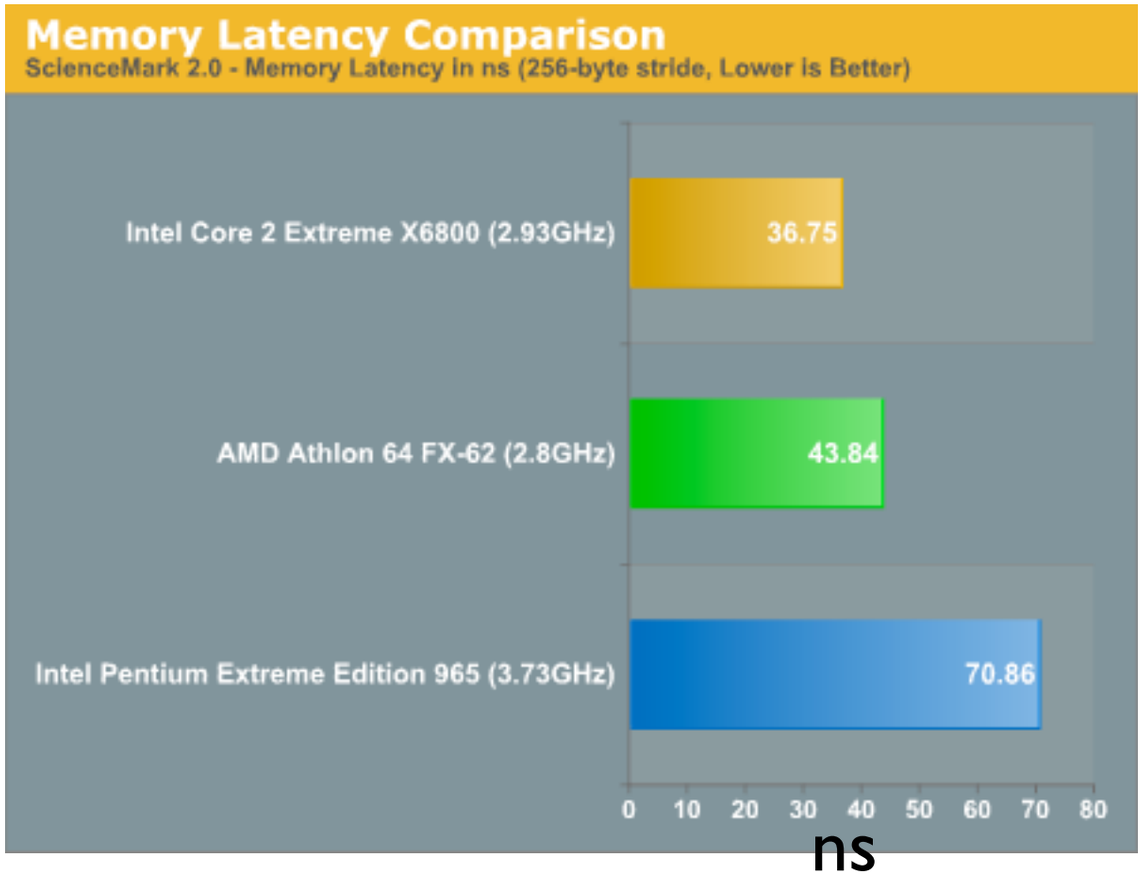
```
do i = 1, imax
do j = 1, jmax
  a(i,j) = float(i-j)/float(i+j)
enddo
enddo
```

sometimes compilers do these optimizations (-O3)

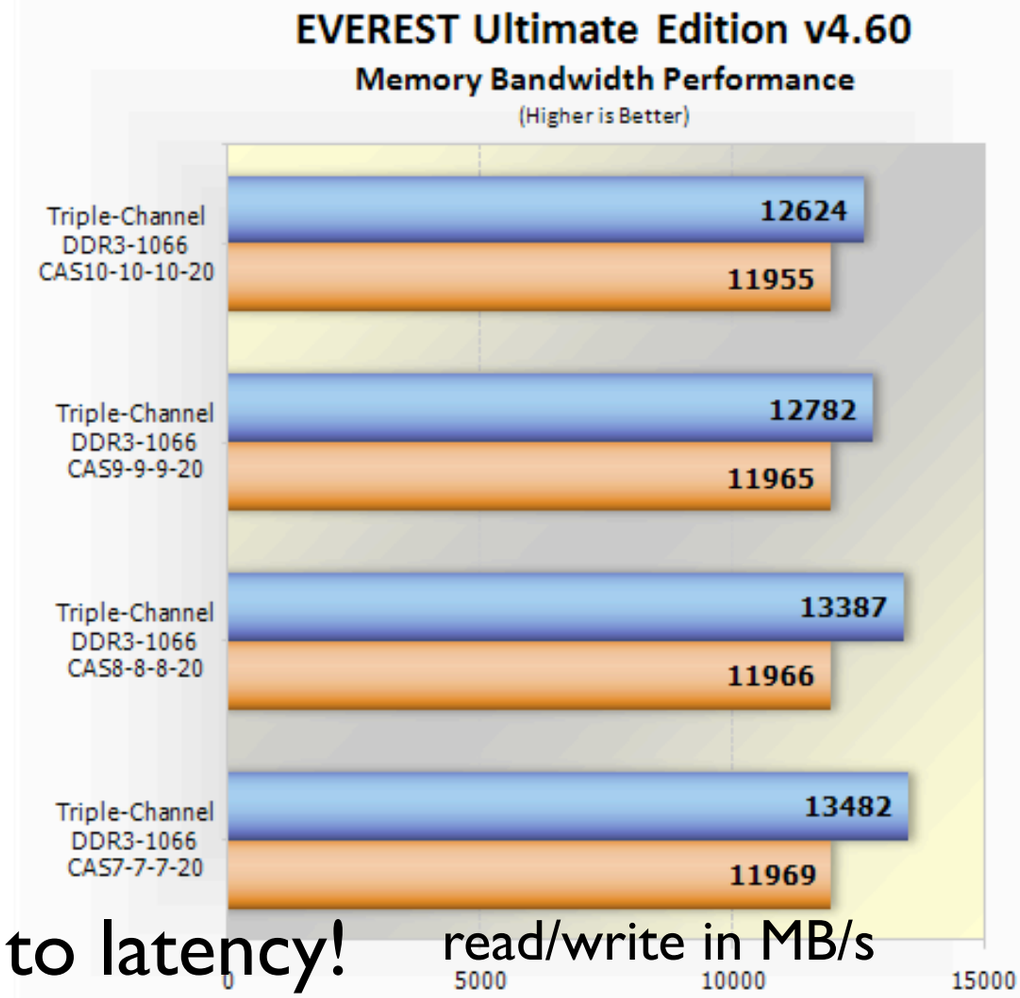
Latency & Bandwidth

minimum time to do an action
(access time)

rate of action once action is initialized

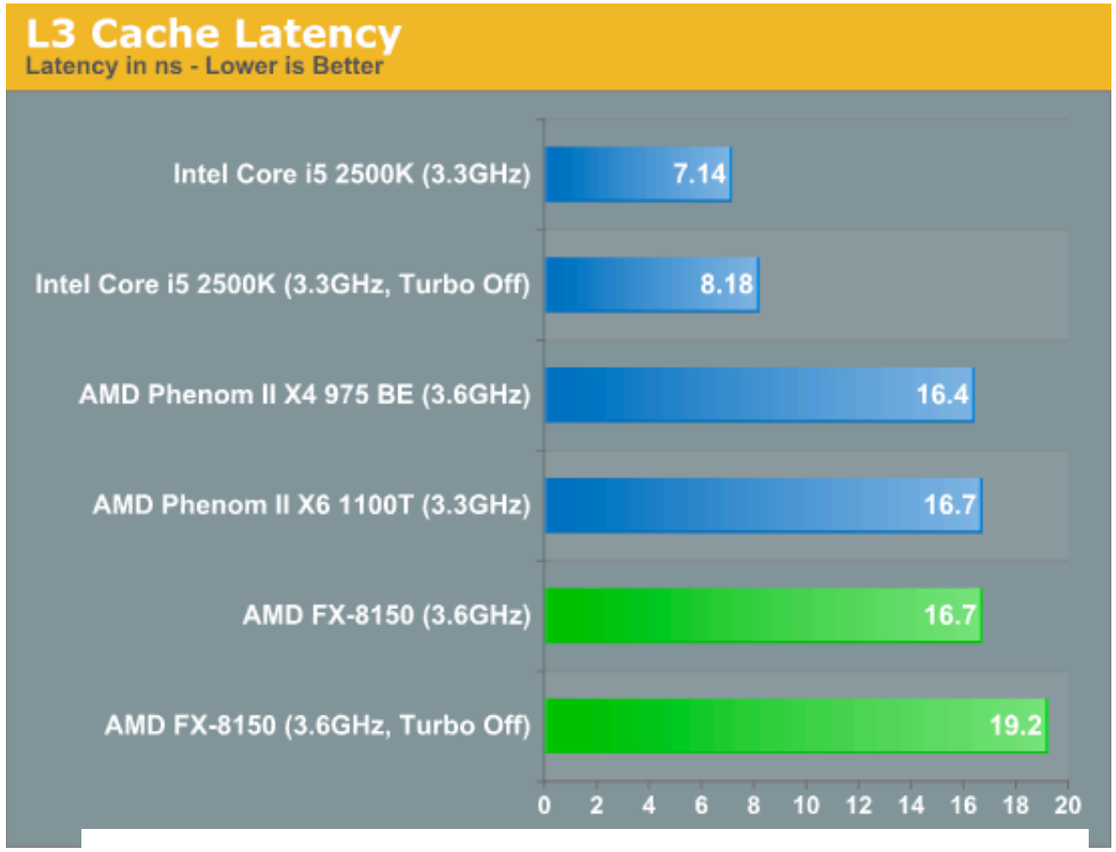


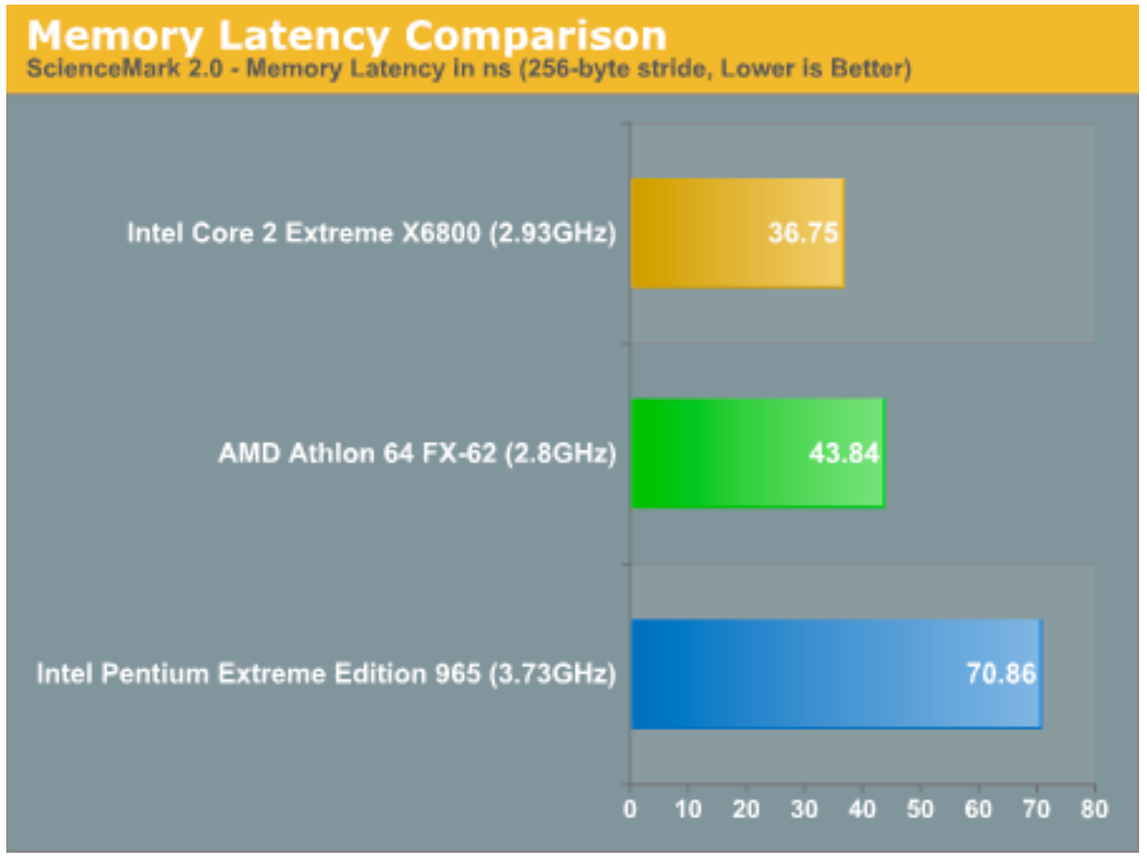
ns

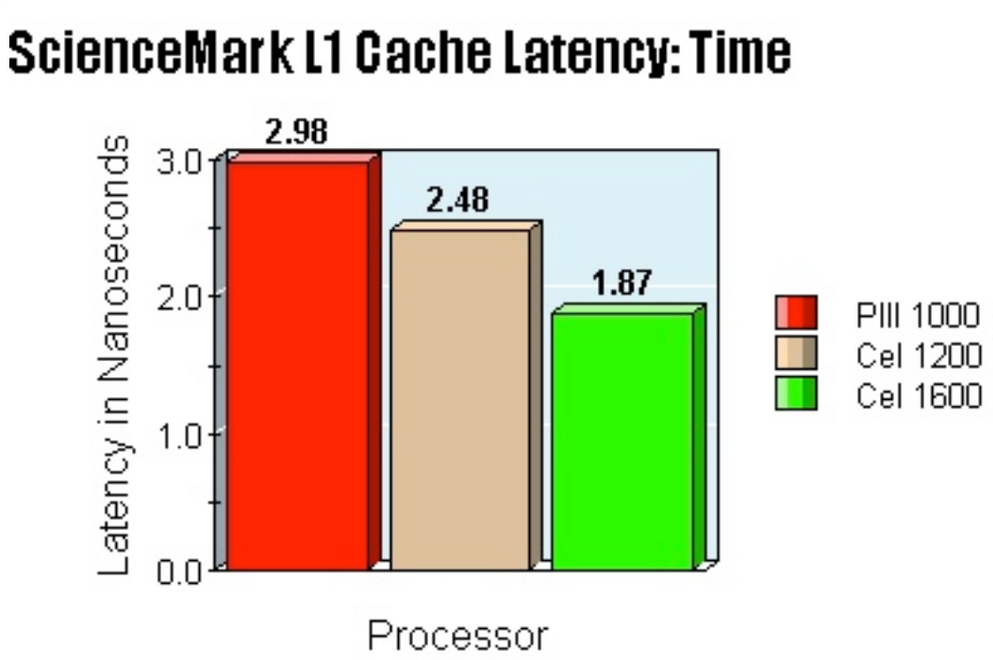


read/write in MB/s

better to minimize memory access due to latency!



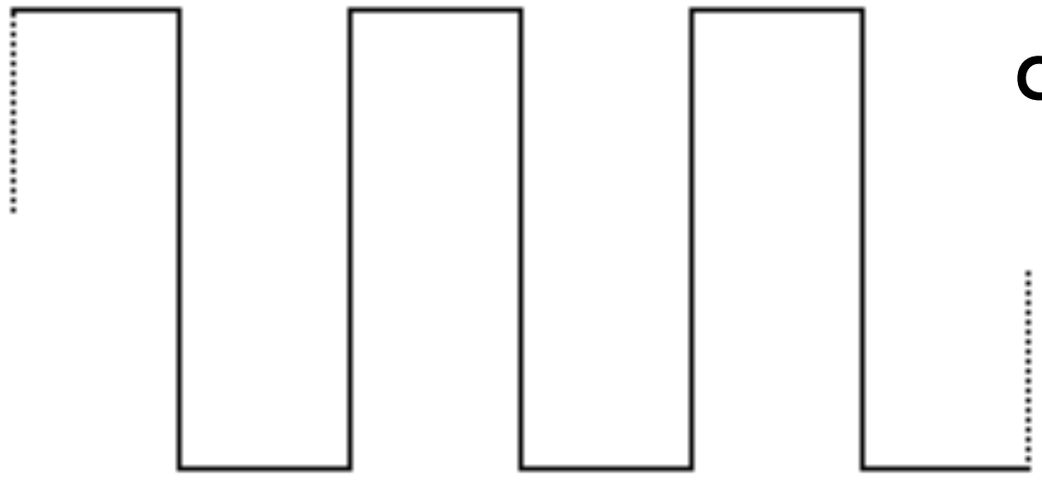




Nehalem processor:
L1~64 kB
L2~2 MB
L3~30 MB

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

Clock Rate



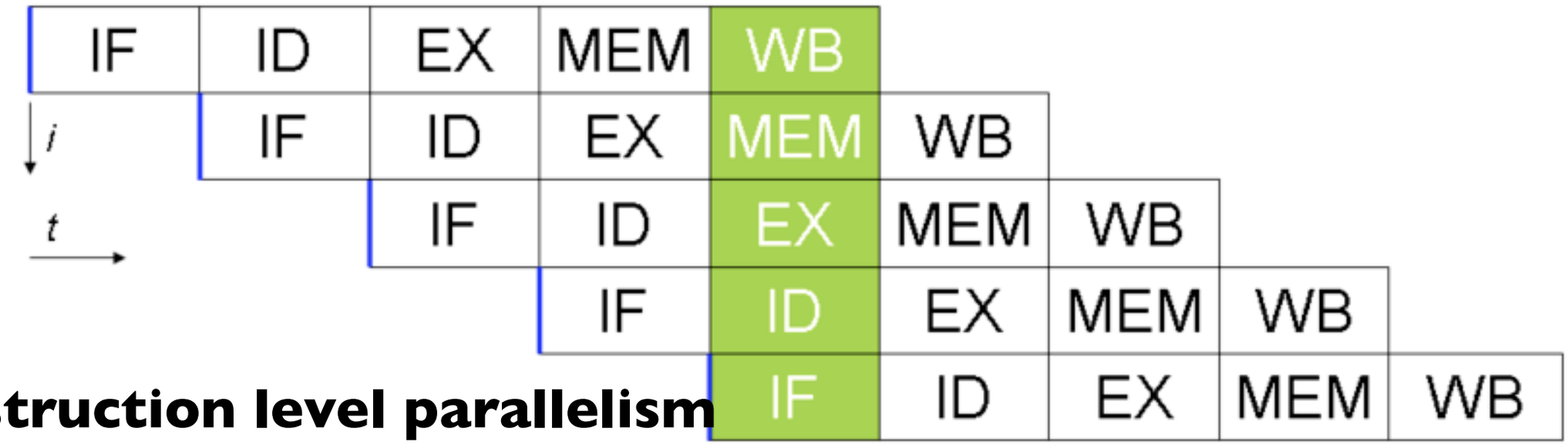
clock coordinates
different actions

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)

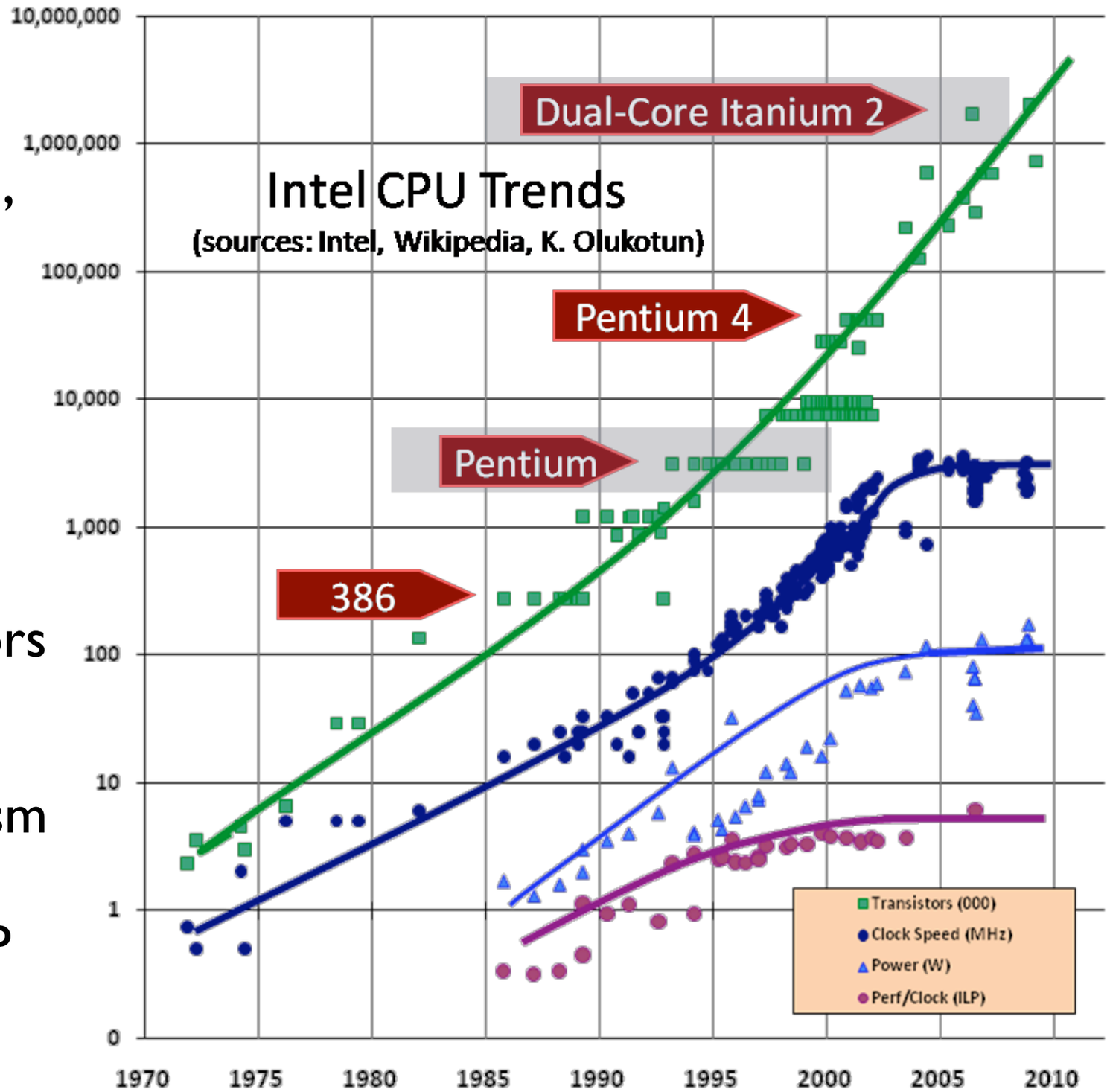
Moore's Law

saturation of clock-speed, power efficiency, ILP

=> paradigm shift

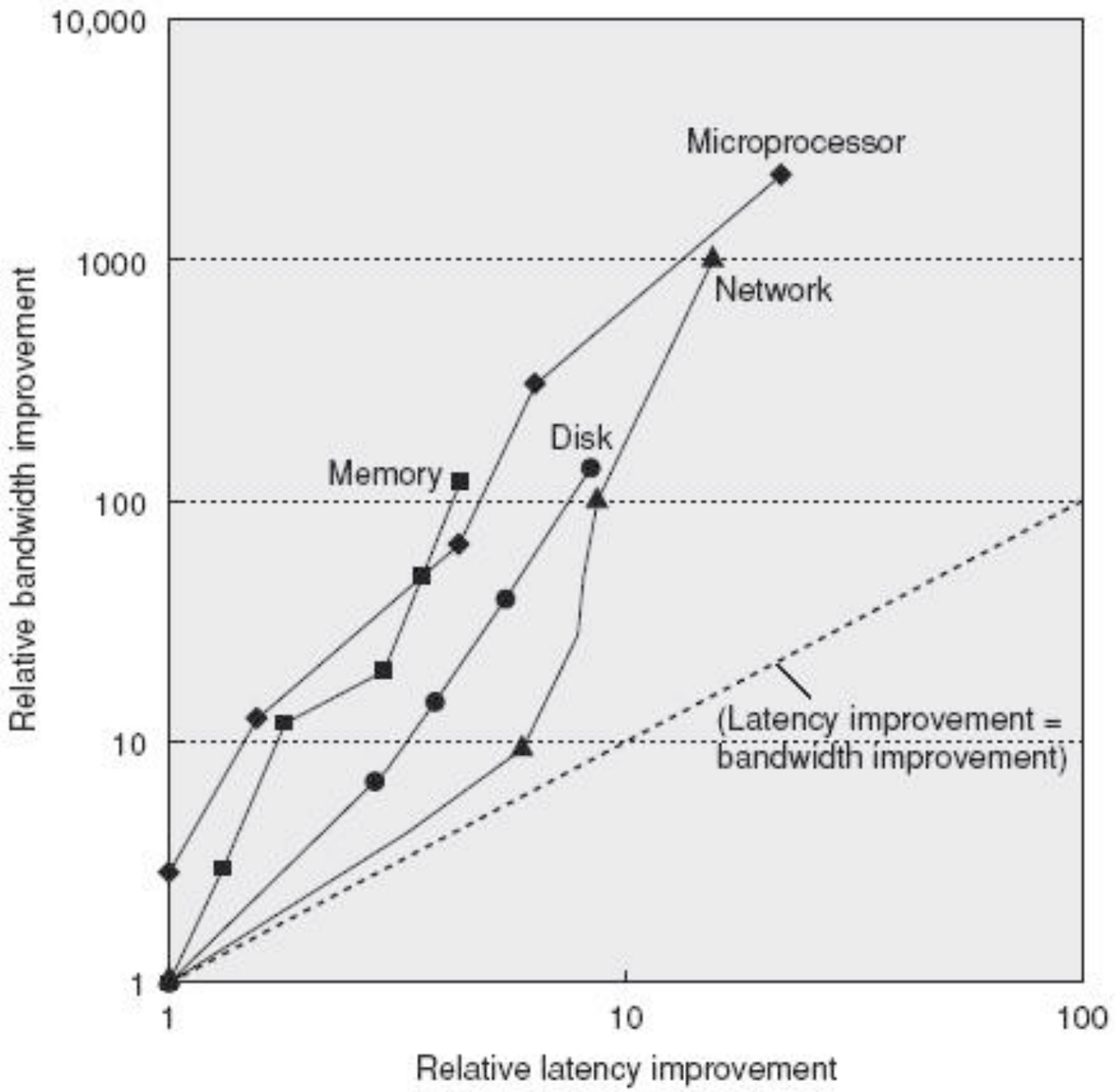
multiple processors/chip rather than faster processors

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



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

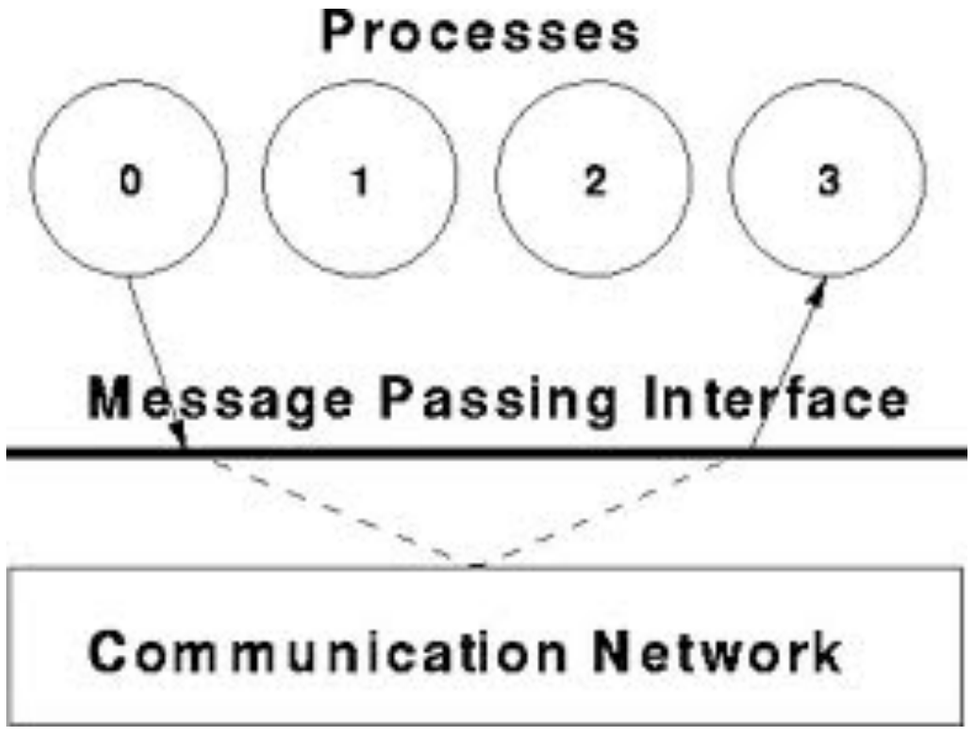
Grid computing: computers communicating over the internet; e.g., SETI@home

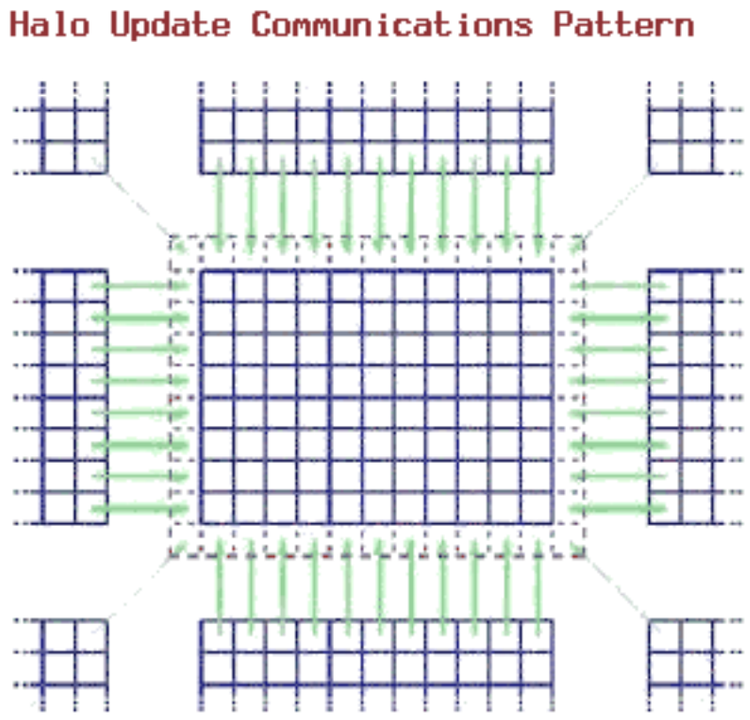
GPUs (graphics processing units): driven by games/graphics industry, fast FP operations

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

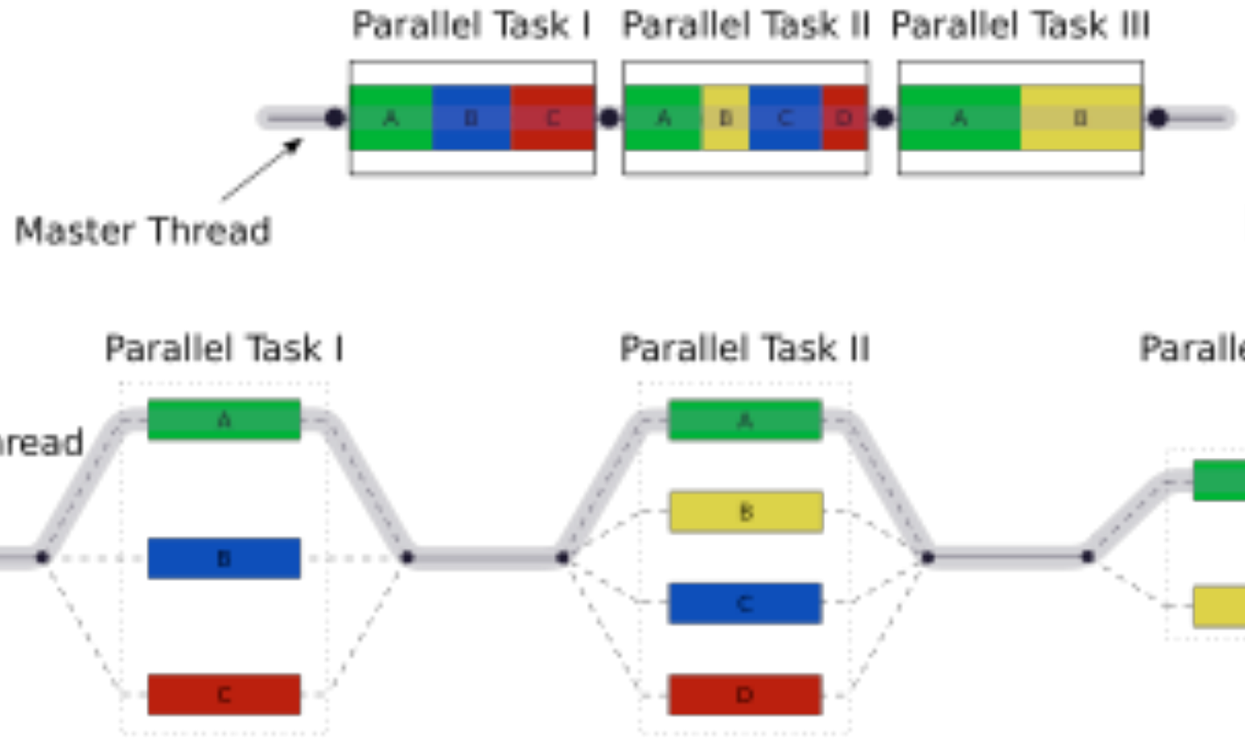
distributed memory programming model

message passing; MPI





shared memory programming model



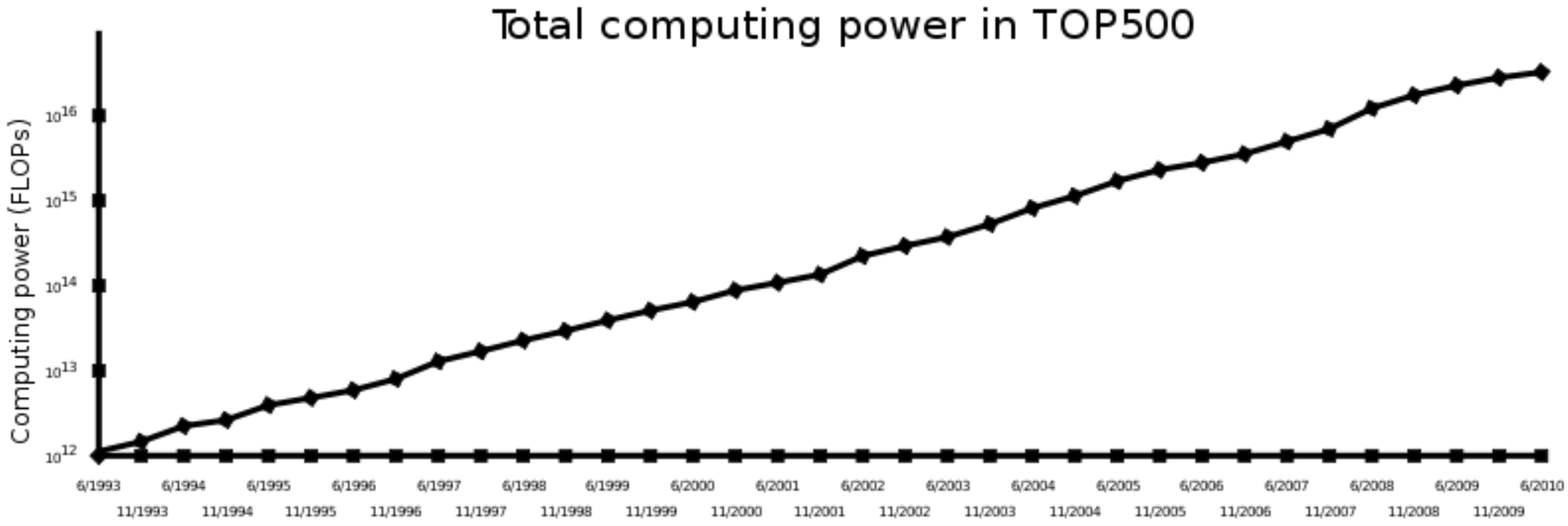
multithreading; openMP

Trends in Supercomputing

http://top500.org/:

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

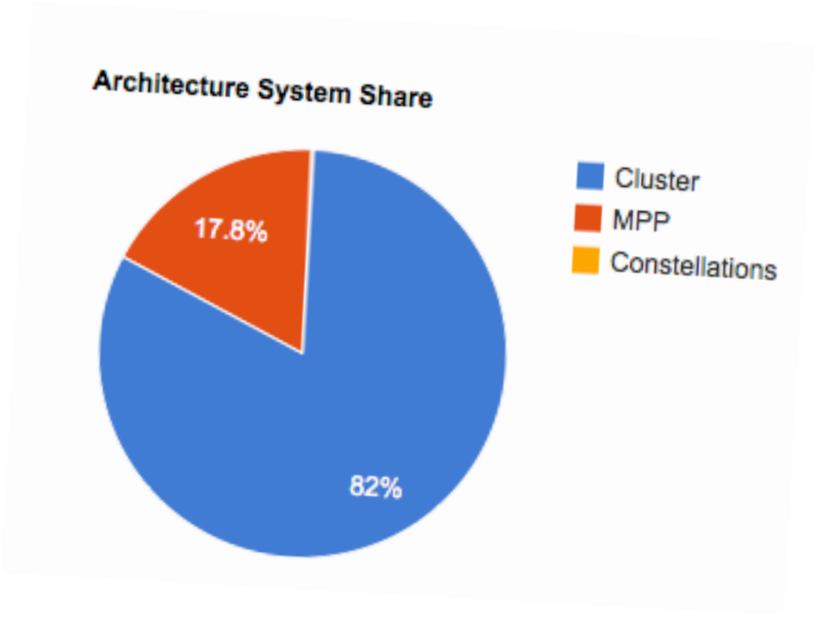
shows trends in architecture, interconnects, vendors, etc.

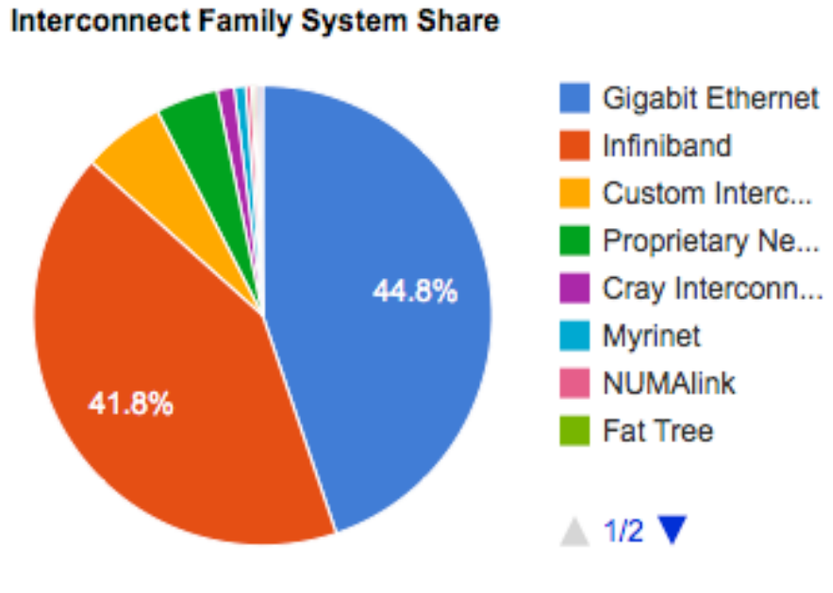


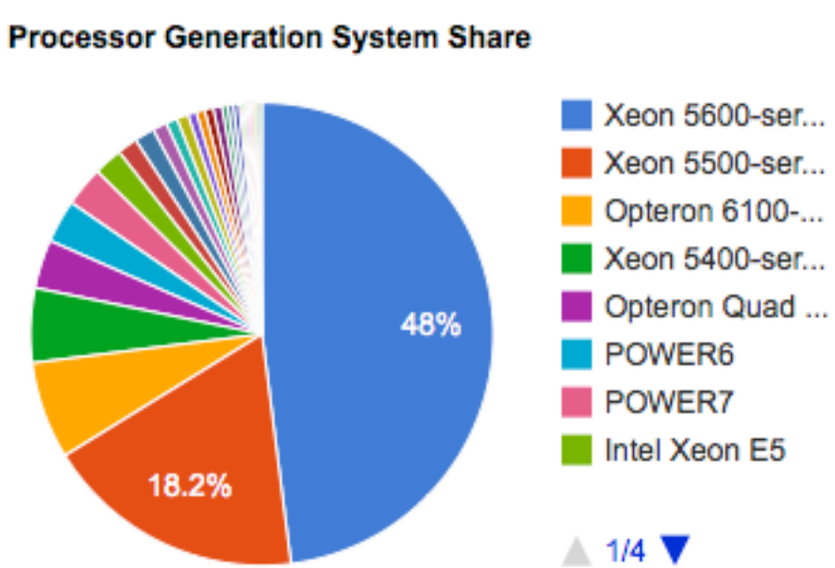
the list

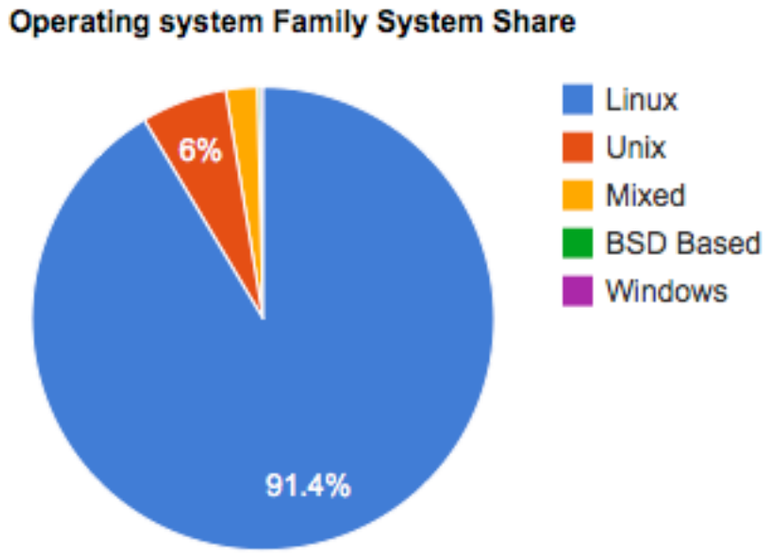
Rank	Site	Computer
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT
3	DOE/SC/Oak Ridge National Laboratory United States	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

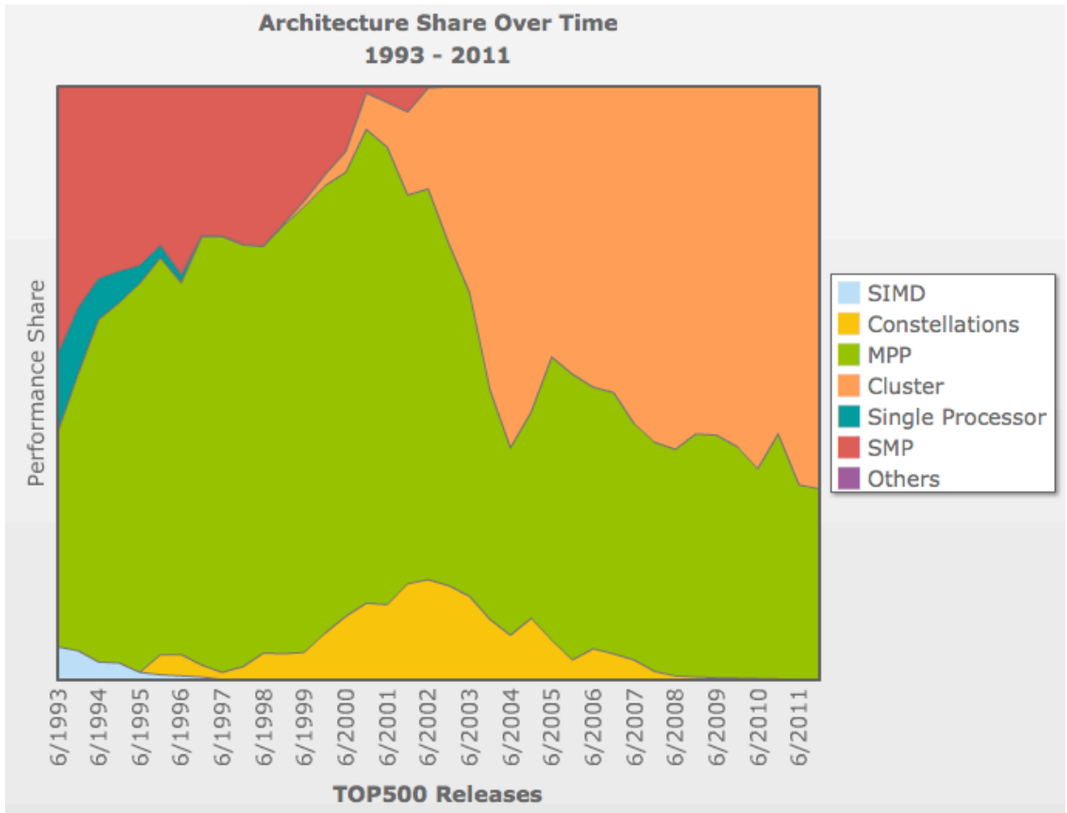
Some Statistics

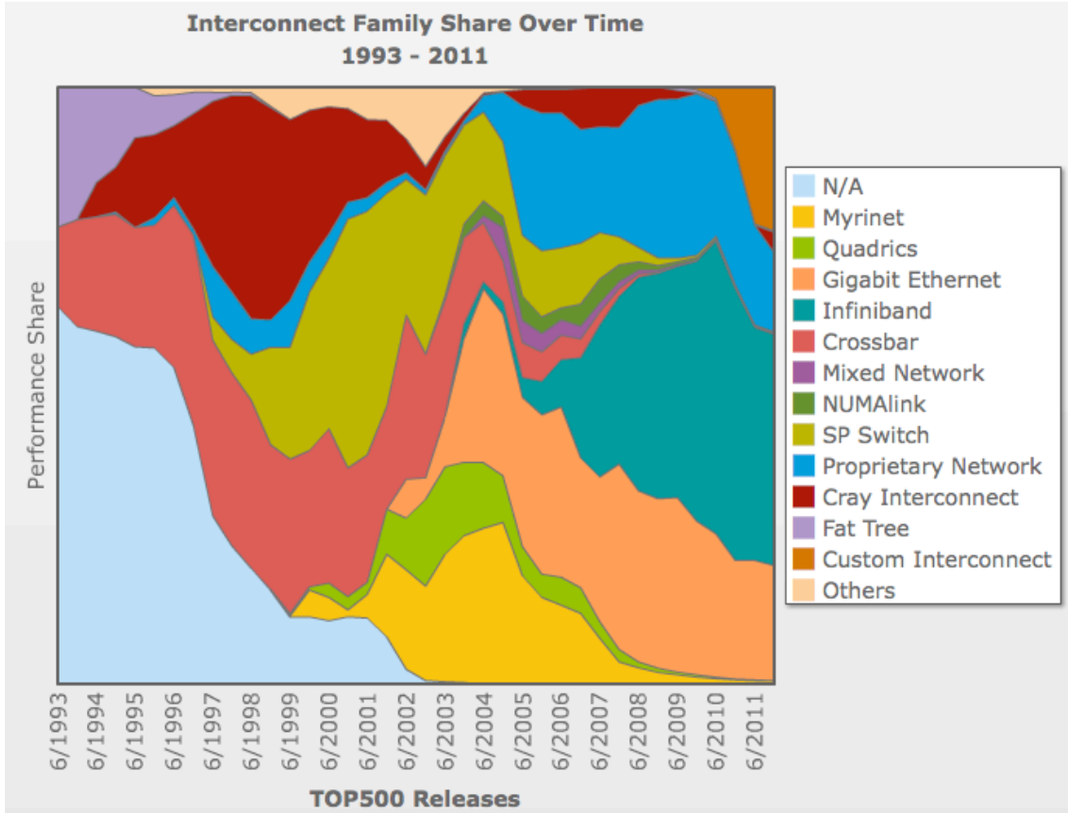


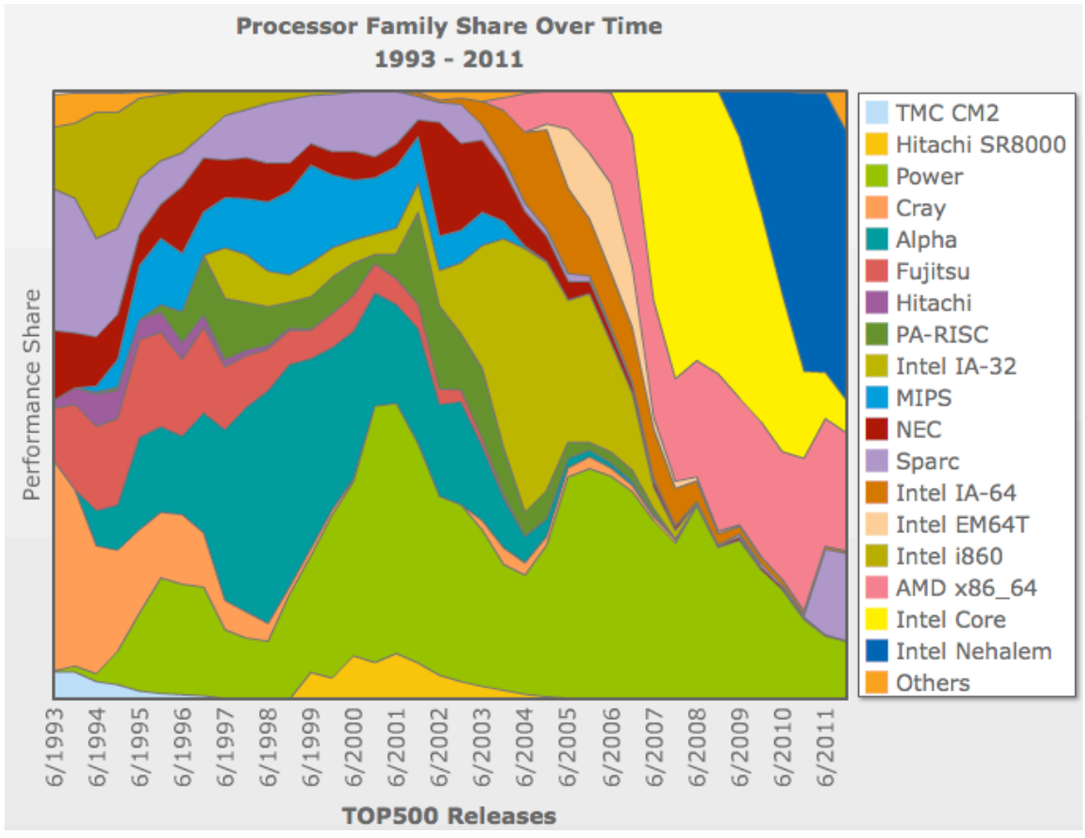


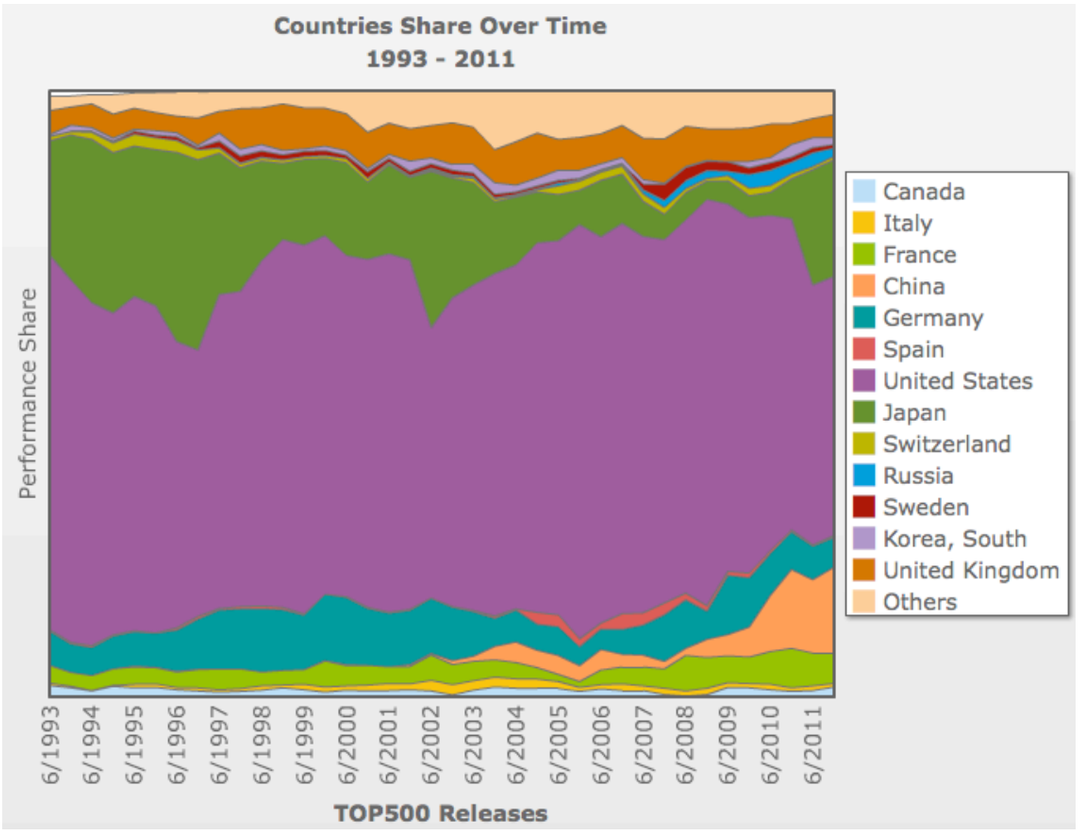





Architecture Share Over Time
1993 - 2011
Performance Share
SIMD
Constellations
MPP
Cluster
Single Processor
SMP
Others
6/1993
6/1994
6/1995
6/1996
6/1997
6/1998
6/1999
6/2000
6/2001
6/2002
6/2003
6/2004
6/2005
6/2006
6/2007
6/2008
6/2009
6/2010
6/2011
TOP500 Releases


Interconnect Family Share Over Time
1993 - 2011
Performance Share
N/A
Myrinet
Quadrics
Gigabit Ethernet
Infiniband
Crossbar
Mixed Network
NUMAlink
SP Switch
Proprietary Network
Cray Interconnect
Fat Tree
Custom Interconnect
Others
6/1993
6/1994
6/1995
6/1996
6/1997
6/1998
6/1999
6/2000
6/2001
6/2002
6/2003
6/2004
6/2005
6/2006
6/2007
6/2008
6/2009
6/2010
6/2011
TOP500 Releases


Processor Family Share Over Time
1993 - 2011
Performance Share
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
6/1993
6/1994
6/1995
6/1996
6/1997
6/1998
6/1999
6/2000
6/2001
6/2002
6/2003
6/2004
6/2005
6/2006
6/2007
6/2008
6/2009
6/2010
6/2011
TOP500 Releases


Countries Share Over Time
1993 - 2011
Performance Share
Canada
Italy
France
China
Germany
Spain
United States
Japan
Switzerland
Russia
Sweden
Korea, South
United Kingdom
Others
6/1993
6/1994
6/1995
6/1996
6/1997
6/1998
6/1999
6/2000
6/2001
6/2002
6/2003
6/2004
6/2005
6/2006
6/2007
6/2008
6/2009
6/2010
6/2011
TOP500 Releases

testing performance

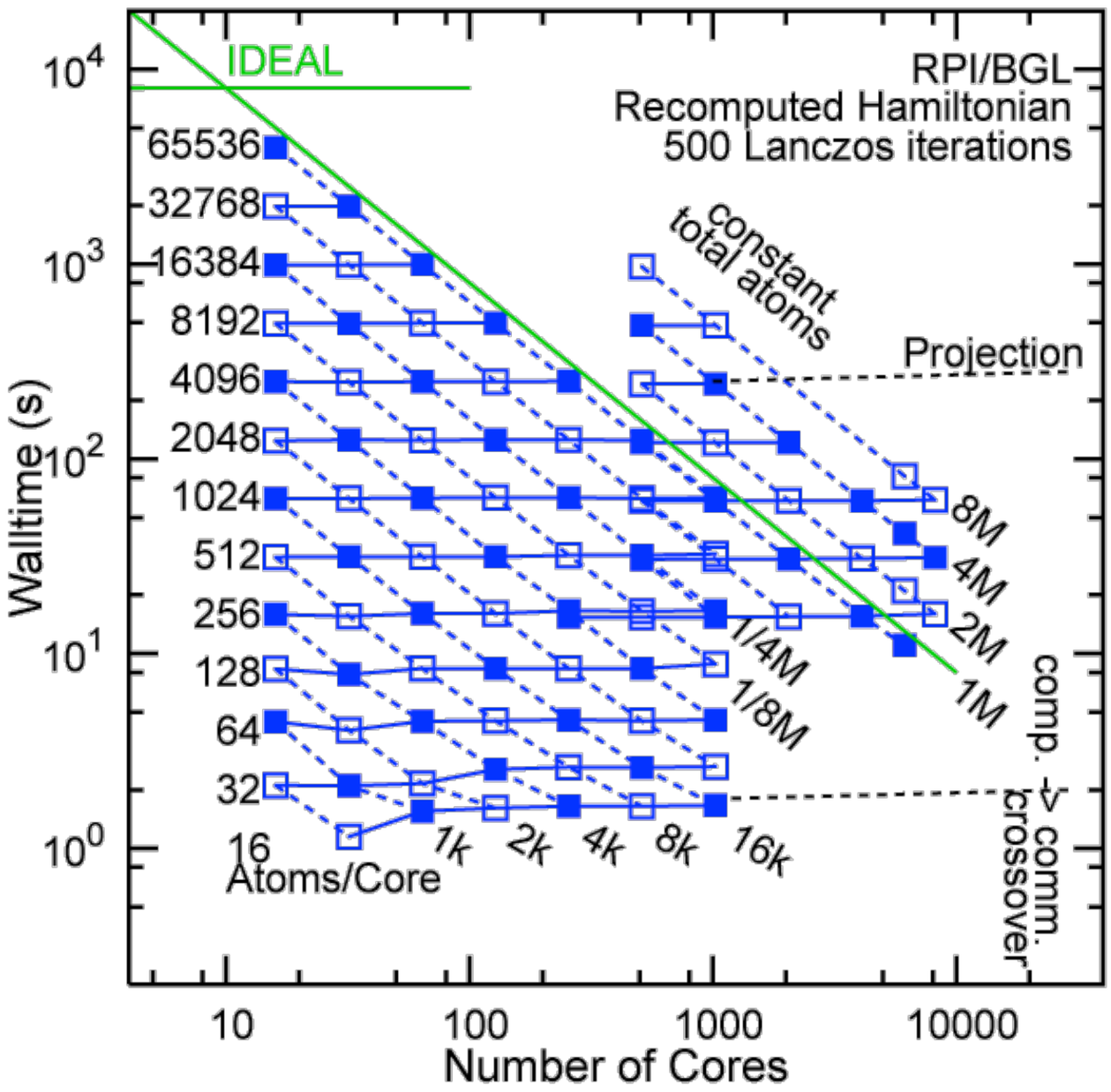
strong scaling: increase the no. of procs. on a fixed problem-size

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