Review: Motivation for and challenges of PNA

- Future applications will increasingly rely on numerical algorithms
- Ready or not, parallelism is here
- Algorithmic and hardware innovation go hand-in-hand
- Parallelization challenges:
  - Finding parallelism: Speedup(p) * (Serial fraction) ≤ 1
  - Parallel overheads: *e.g.*, communication, synchronization, redundant work
  - NUMA: “Mops” not flops
  - Load imbalance; numerical issues “at scale”

Why does hardware matter and how is it changing?

Prof. Richard Vuduc

Georgia Institute of Technology

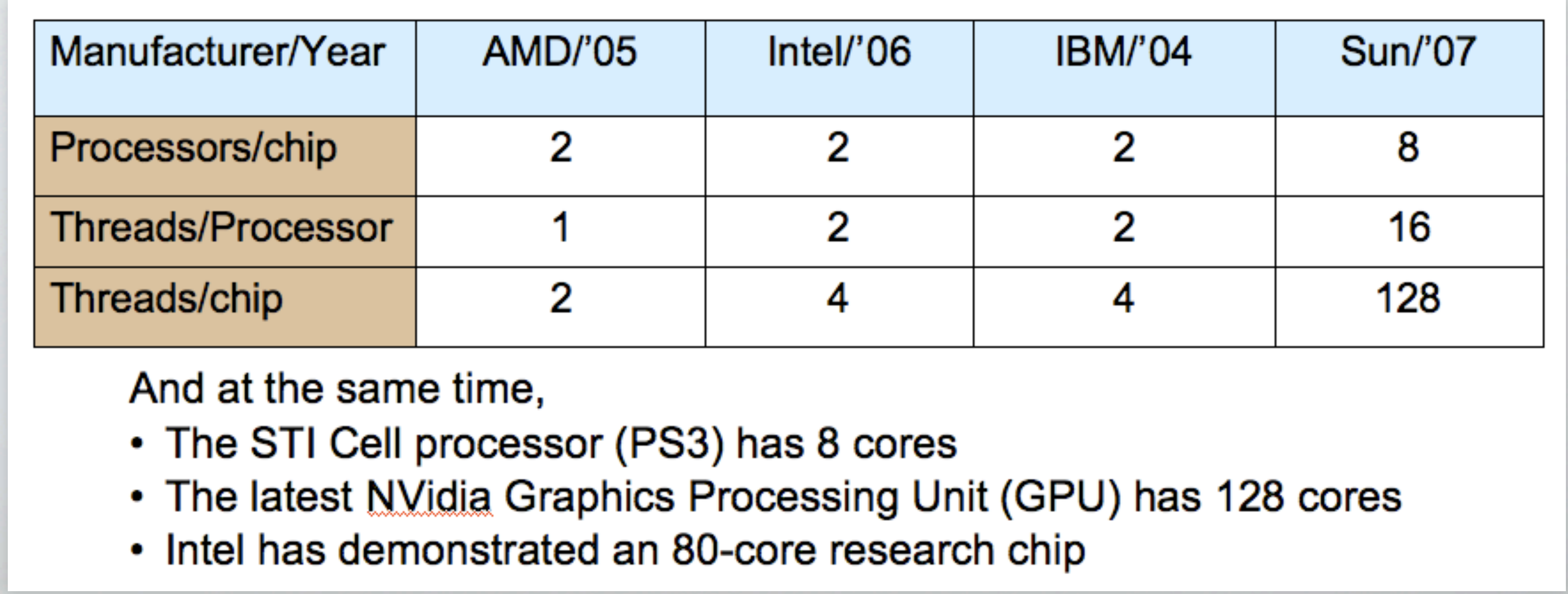
CSE/CS 8803 PNA, Spring 2008

[L.02] Thursday, January 10, 2008

Sources for today’s material

- CS 267 / CS 194 (Yelick @UCB)
- Burton Smith’s talk at MSR Manycore Workshop (summer ’07)
- Patterson/Culler @ UCB
- Yotov and Pingali (UT Austin)
- Kamil, *et al*. (UCB)

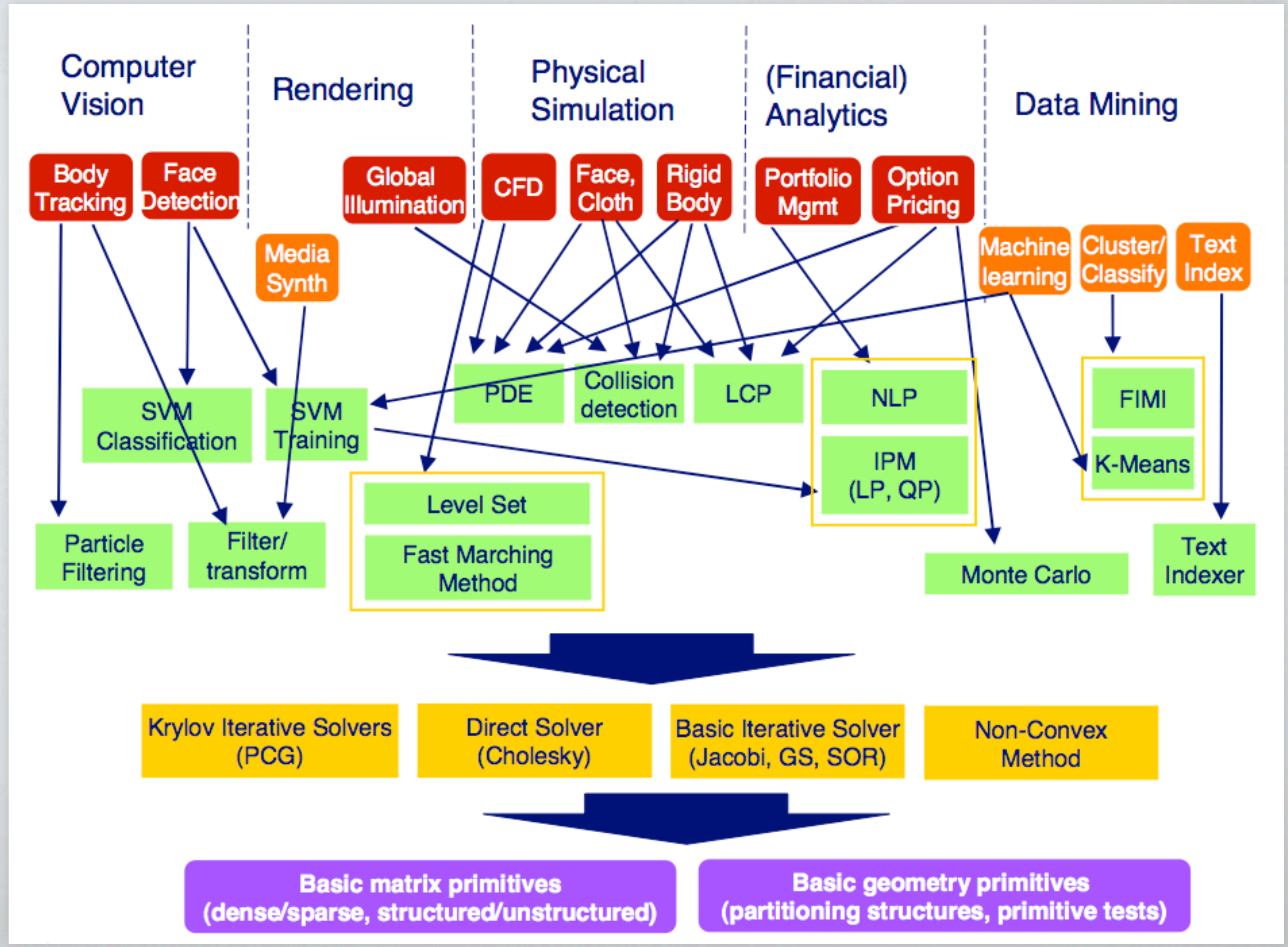
Current landscape of widely available parallel architectures



Manufacturer/Year	AMD/'05	Intel/'06	IBM/'04	Sun/'07
Processors/chip	2	2	2	8
Threads/Processor	1	2	2	16
Threads/chip	2	4	4	128

And at the same time,

- The STI Cell processor (PS3) has 8 cores
- The latest NVidia Graphics Processing Unit (GPU) has 128 cores
- Intel has demonstrated an 80-core research chip



Computer Vision

Rendering

Physical Simulation

(Financial) Analytics

Data Mining

Body Tracking

Face Detection

Global Illumination

CFD

Face, Cloth

Rigid Body

Portfolio Mgmt

Option Pricing

Media Synth

Machine learning

Cluster/ Classify

Text Index

SVM Classification

SVM Training

PDE

Collision detection

LCP

NLP

IPM (LP, QP)

FIMI

K-Means

Level Set

Fast Marching Method

Particle Filtering

Filter/ transform

Monte Carlo

Text Indexer

Krylov Iterative Solvers (PCG)

Direct Solver (Cholesky)

Basic Iterative Solver (Jacobi, GS, SOR)

Non-Convex Method

Basic matrix primitives (dense/sparse, structured/unstructured)

Basic geometry primitives (partitioning structures, primitive tests)

Source: Dubey, *et al*., of Intel (2005)

Why does hardware matter?

Hardware constrains concurrency

- Amdahl’s law: Serial speed matters
- Communication costs: Latency and bandwidth
- Parallel overheads: Cost of synchronization

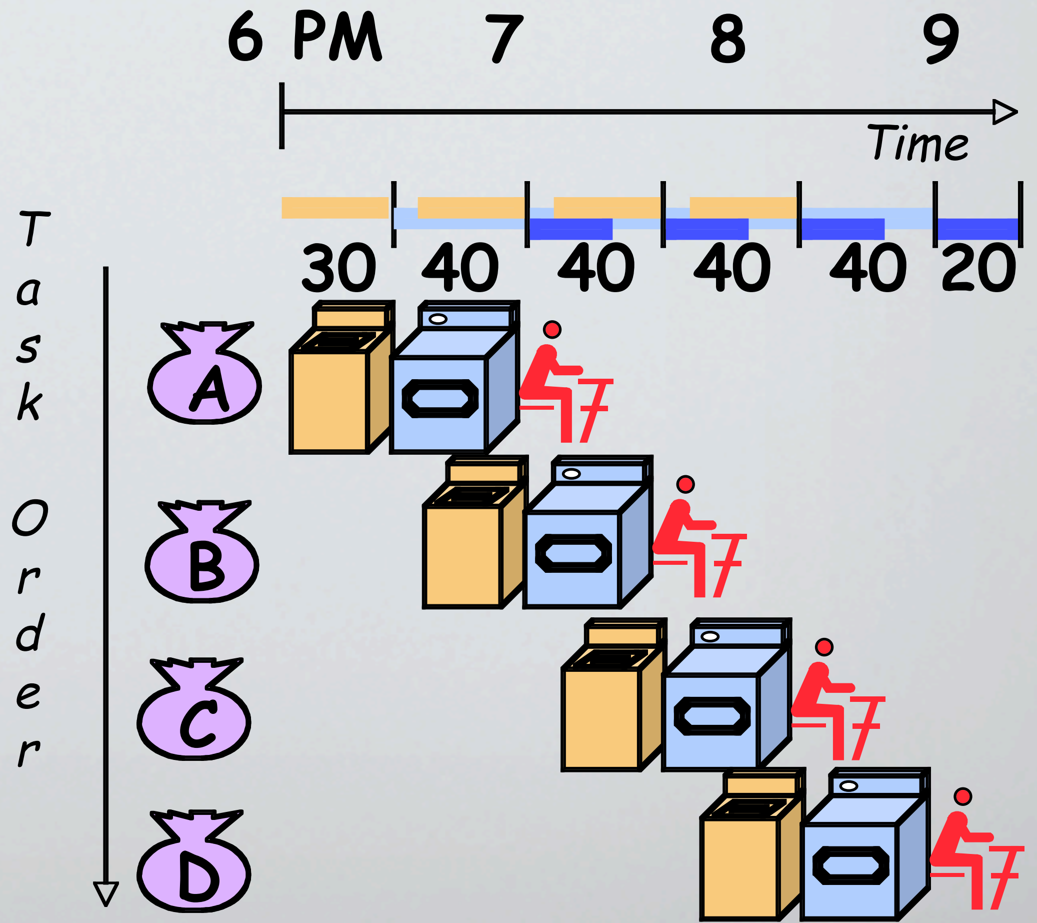
An idealized uniprocessor

- Address space: named bytes/words
- Basic operations: reading, writing, and arithmetic/logical
- Operations executed in program order
- Idealized cost: All operations have roughly same cost

A real uniprocessor

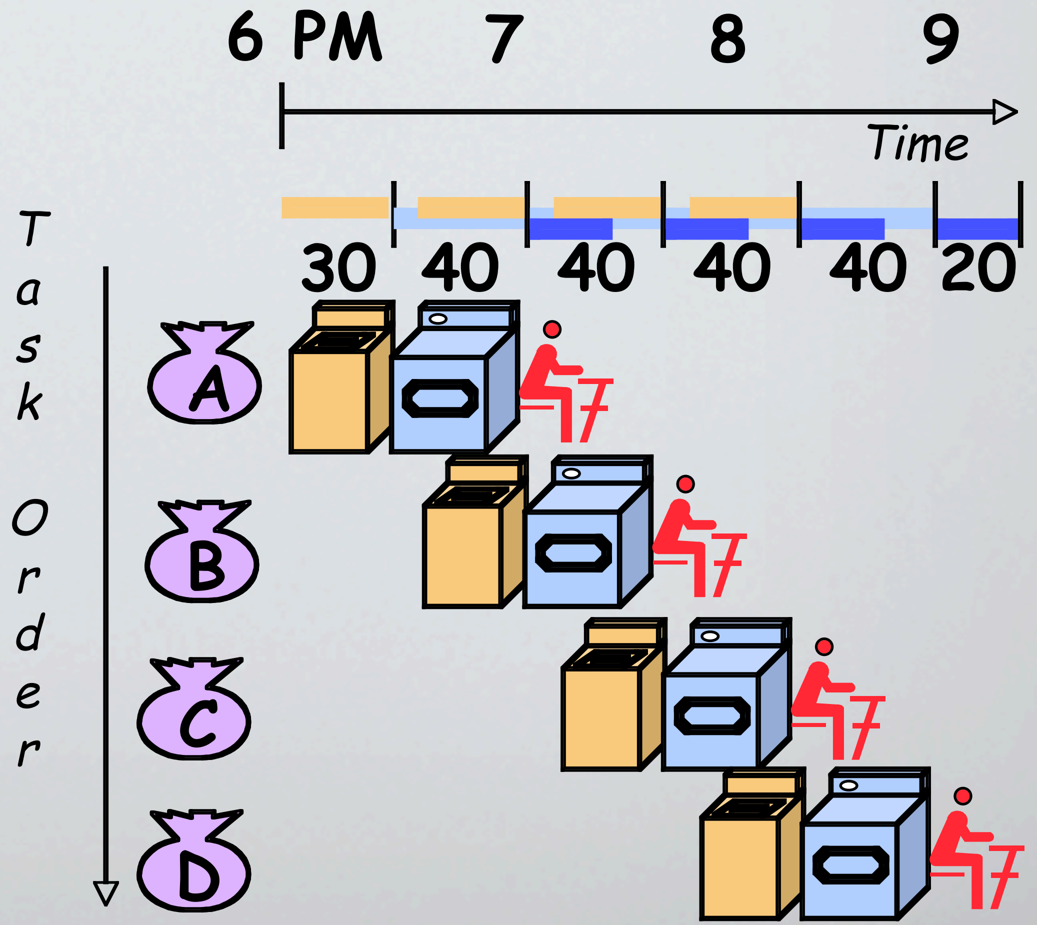
- Has registers and caches, and so access costs vary
- Instruction-level parallelism
    - Multiple “functional units” that run in parallel
    - Different instruction mixes and orders (schedules) have different costs
    - Pipelining
    - Superscalar, out-of-order execution, branch prediction, ...
- Data parallelism: SIMD units (*e.g.*, SSE3)
- Compilers deal with this stuff in theory, but not always in practice

Uniprocessor parallelism: Pipelining

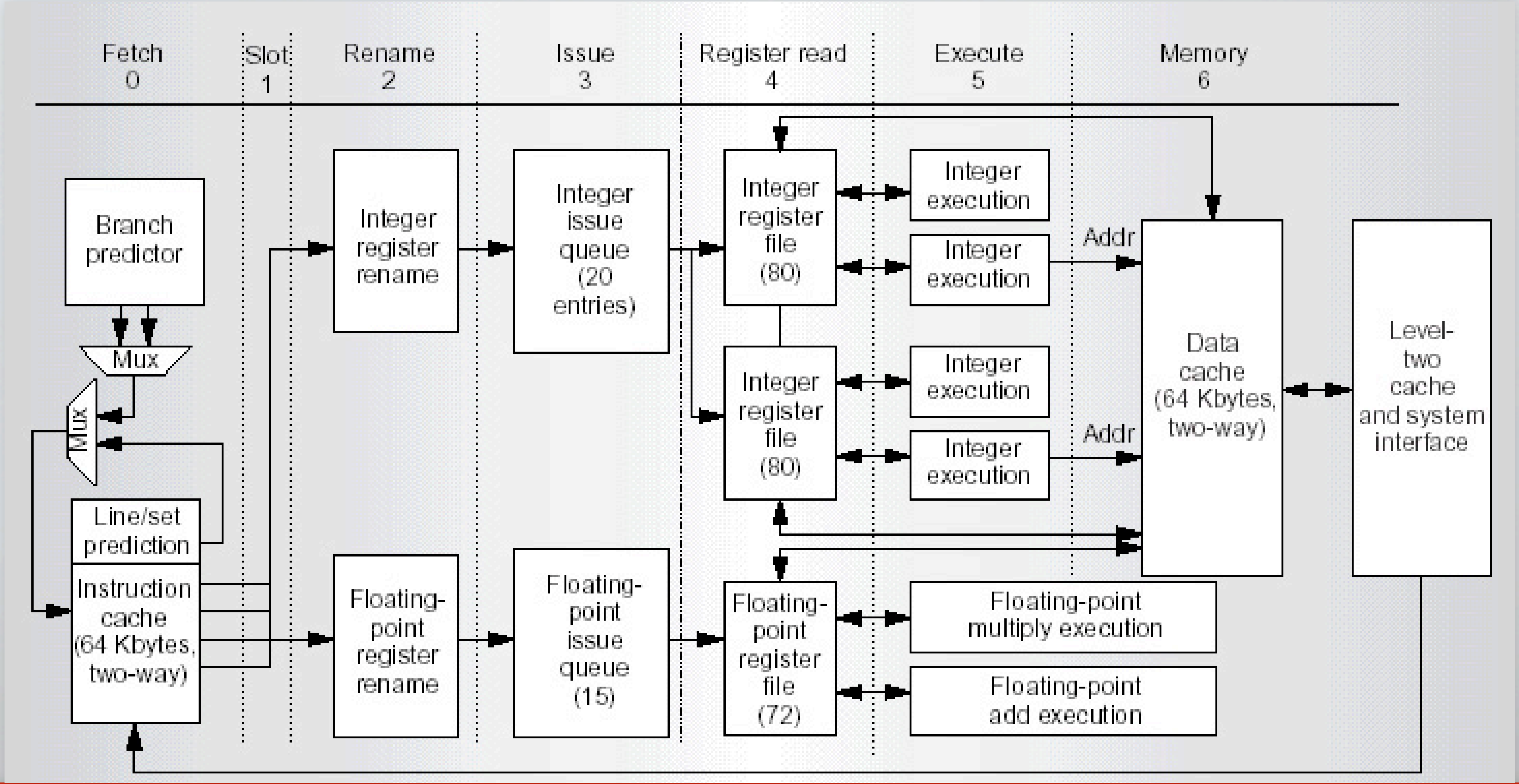


Uniprocessor parallelism: Pipelining

- *E.g.*, laundry: 90 min **latency**
- Wash = 30 mins
  - Dry = 40 mins
  - Fold = 20 mins
- 4 loads sequentially = 6 hours
- 4 loads pipelined = 3.5 hours
- **Bandwidth** = loads / hour

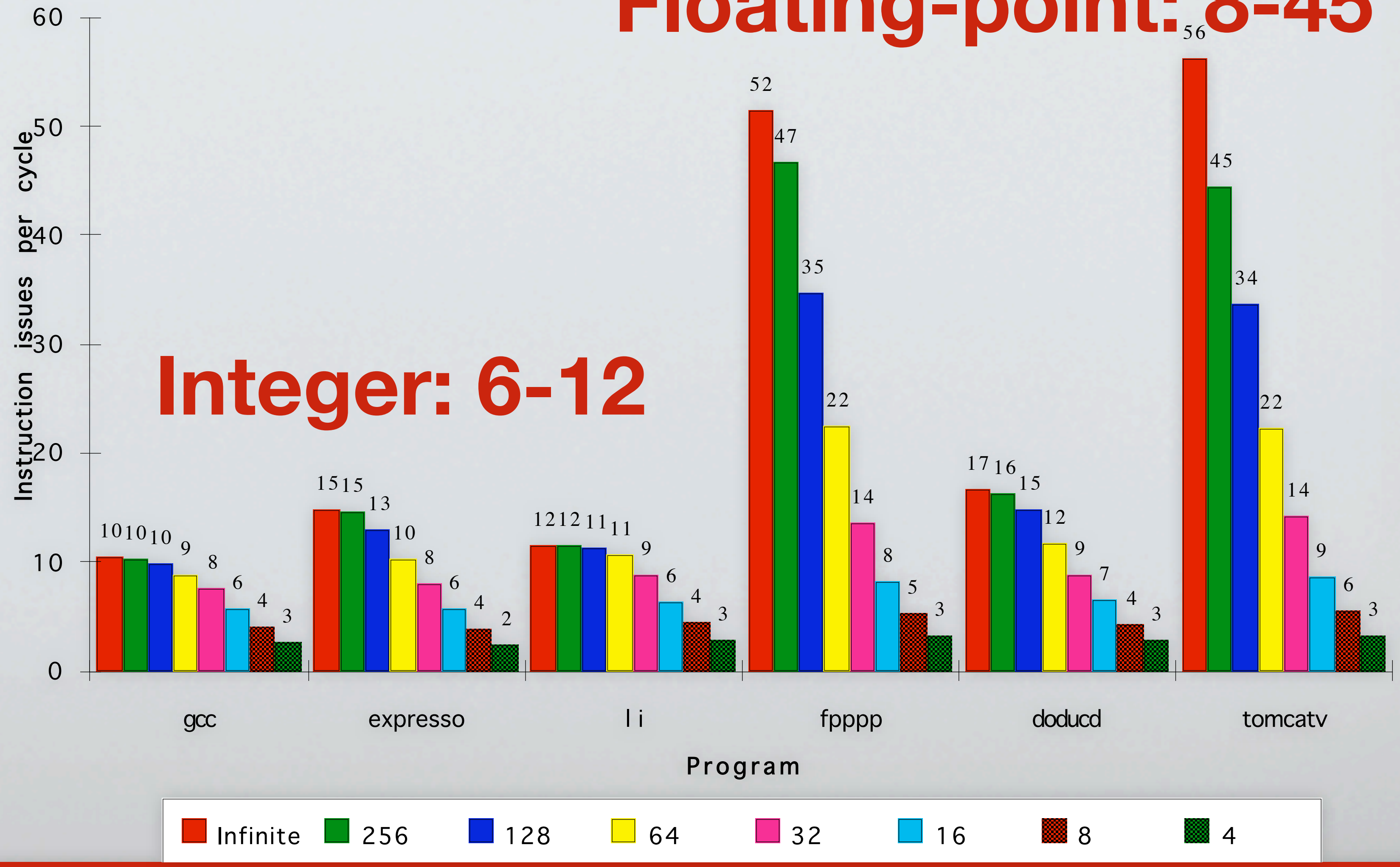


Example: Alpha 21264 pipeline



Fetch
0
Slot
1
Rename
2
Issue
3
Register read
4
Execute
5
Memory
6
Branch predictor
Mux
Mux
Line/set prediction
Instruction cache (64 Kbytes, two-way)
Integer register rename
Floating-point register rename
Integer issue queue (20 entries)
Floating-point issue queue (15)
Integer register file (80)
Integer register file (80)
Floating-point register file (72)
Integer execution
Integer execution
Integer execution
Integer execution
Floating-point multiply execution
Floating-point add execution
Addr
Addr
Data cache (64 Kbytes, two-way)
Level-two cache and system interface

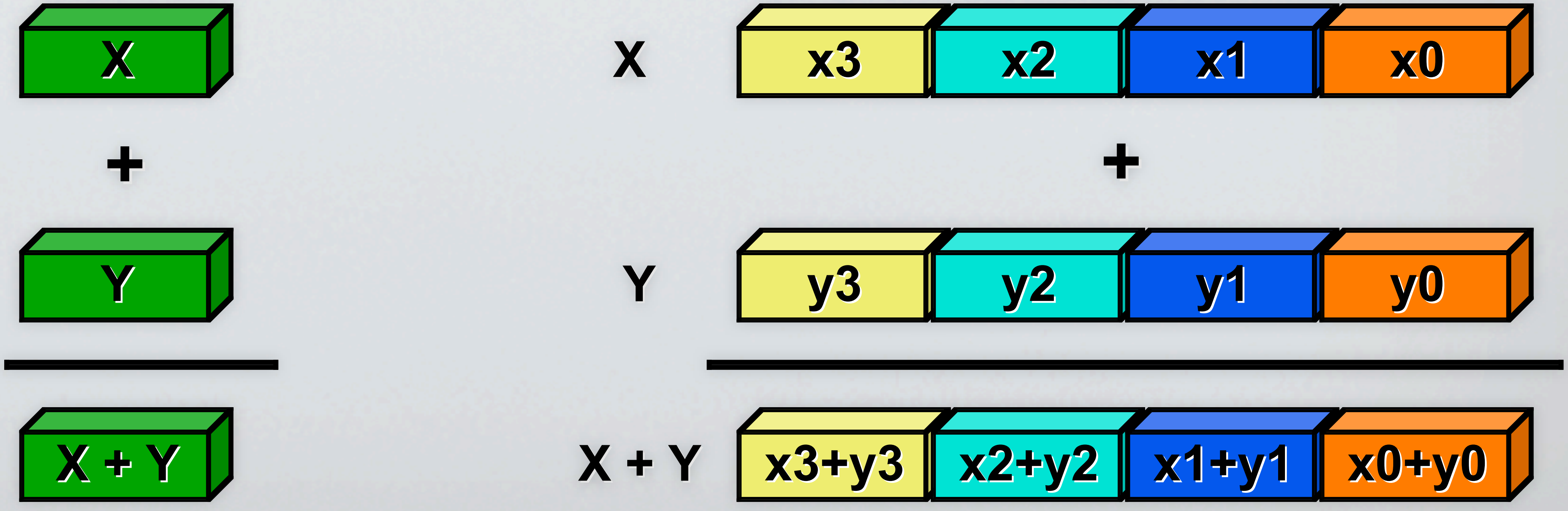
ILP gain likely to be limited.



Floating-point: 8-45

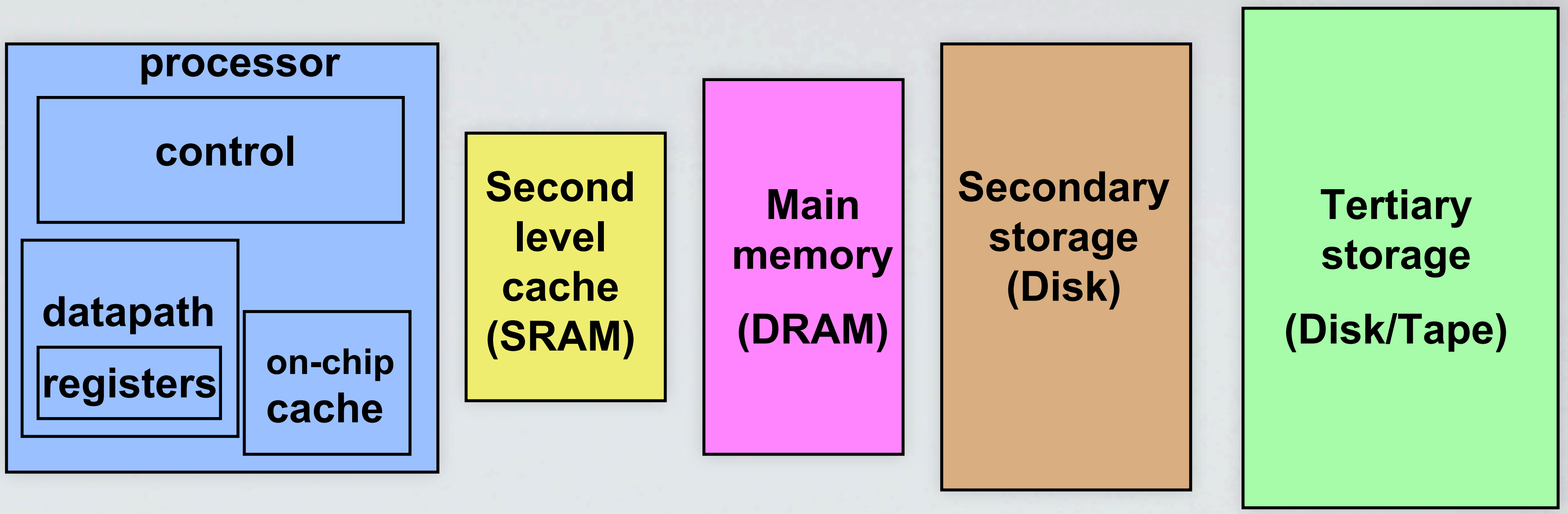
Integer: 6-12

Uniprocessor parallelism: SIMD (Single Instruction, Multiple Data)



Special instructions: Consequences and limits

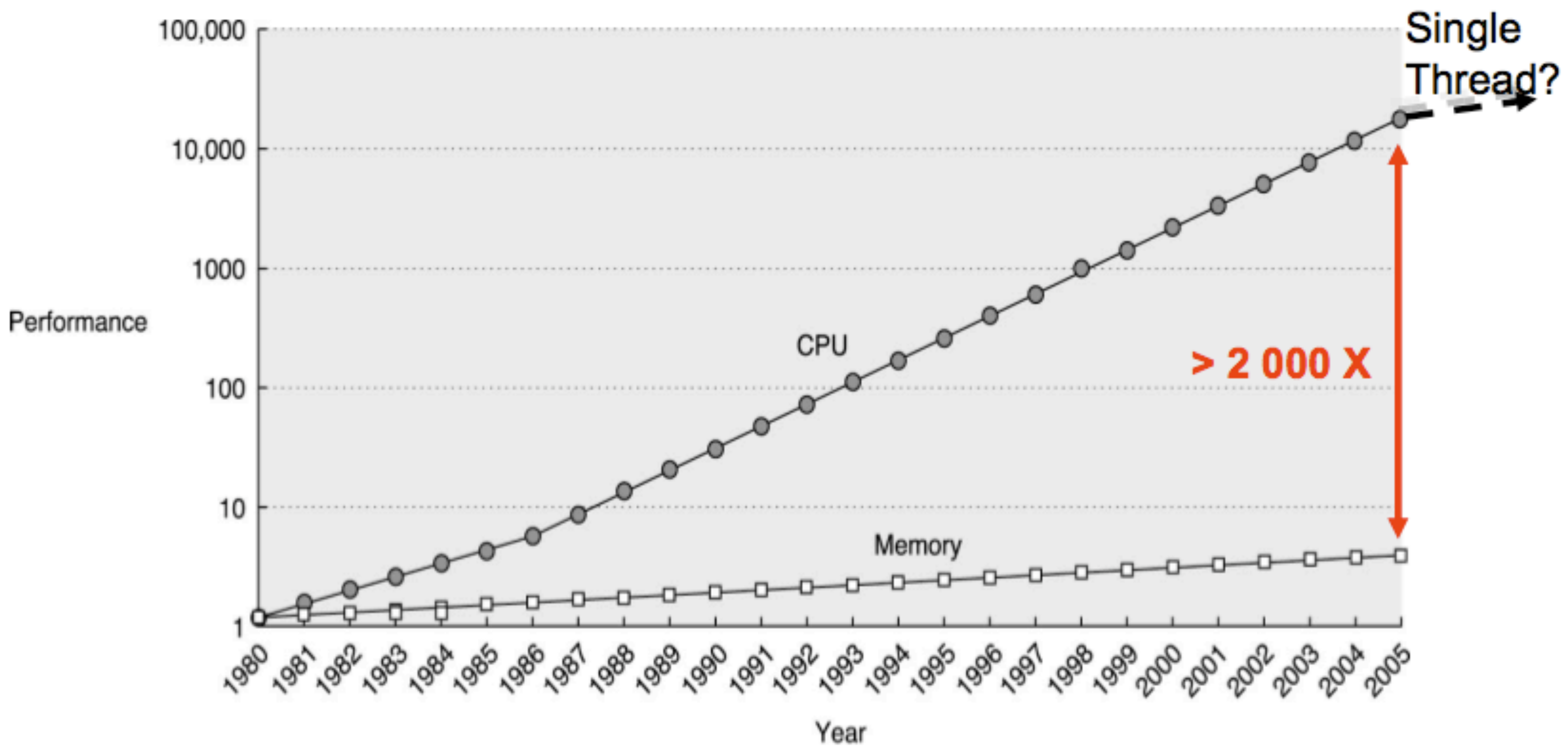
- Improve instruction bandwidth
  - SIMD
  - Other example: Single-cycle **fused multiply-add (FMA)**, z = y + a*x
- In theory, compiler handles everything, but may not in practice
  - May require special flags
  - May require code reorganization to make parallelism “obvious”
  - May need to use “instrinsics” or assembly language
- Subtle difference: floating-point semantics



Cost	1ns	10ns	100ns	10ms	10sec
Size	B	KB	MB	GB	TB

Recall: Memory hierarchies.

Cost of accessing data depends on where data lives.



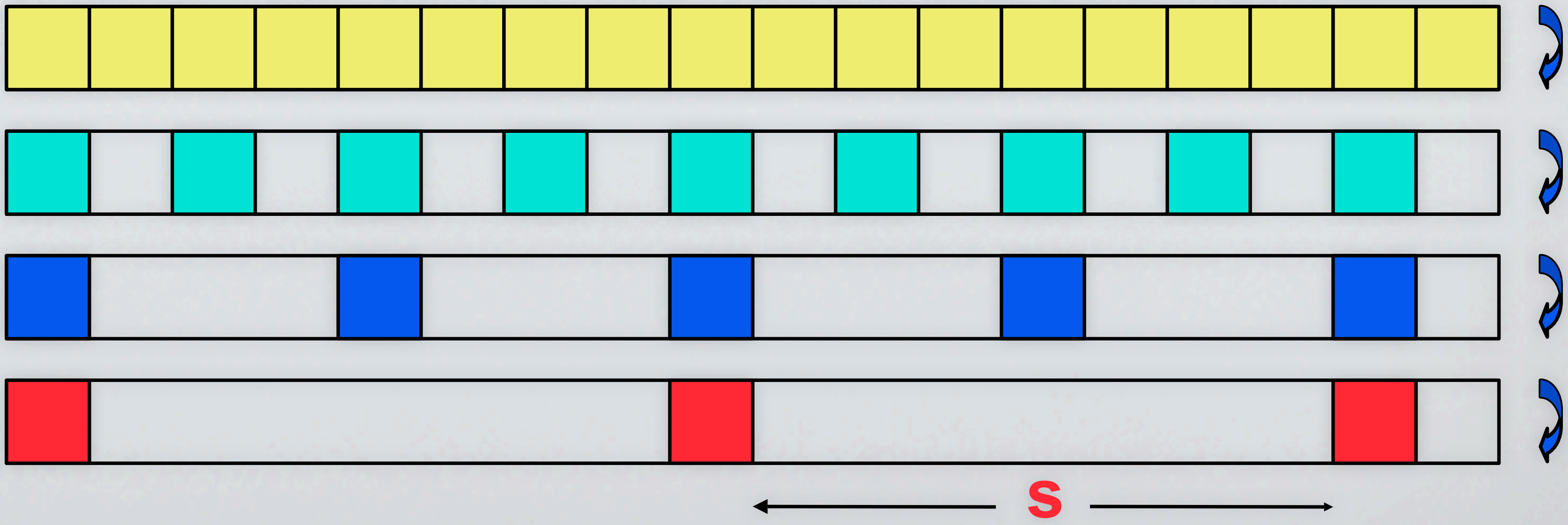
Memory hierarchies reflect growing processor-memory speed gap.

Dealing with high memory latency

- Use caches as fast memory
  - Store data that will be reused many times: **temporal locality**
  - Save chunks of contiguous data: **spatial locality**
- Exploit fact that bandwidth improves faster than latency: **prefetch**

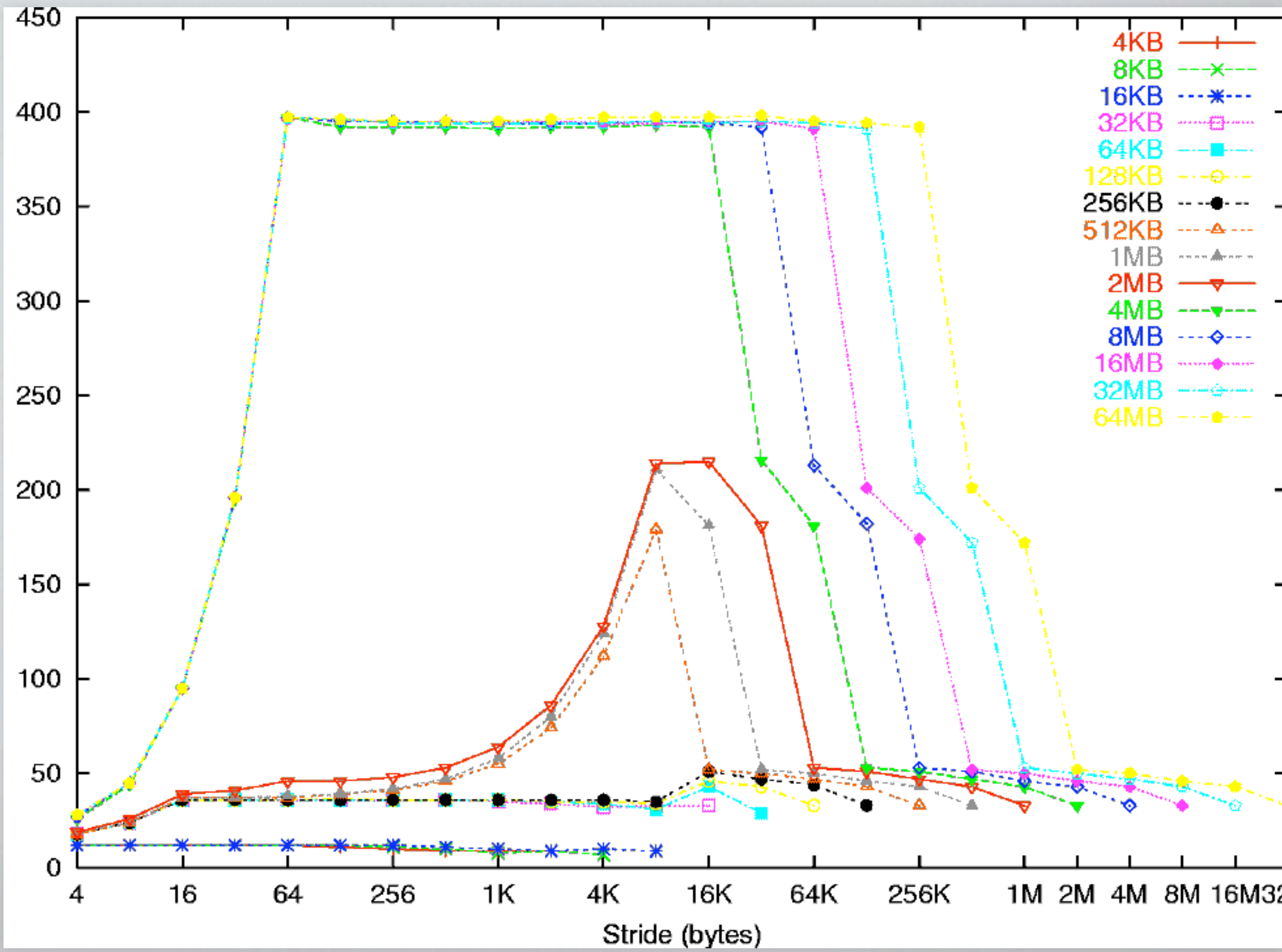
- Modern processors automate cache management
  - All loads cached automatically (LRU), and loaded in chunks (*cache line size*)
  - Typical to have a hardware prefetcher that detects simple patterns

Experiment to observe memory parameters.



Strided-stream through array; measure average access time.

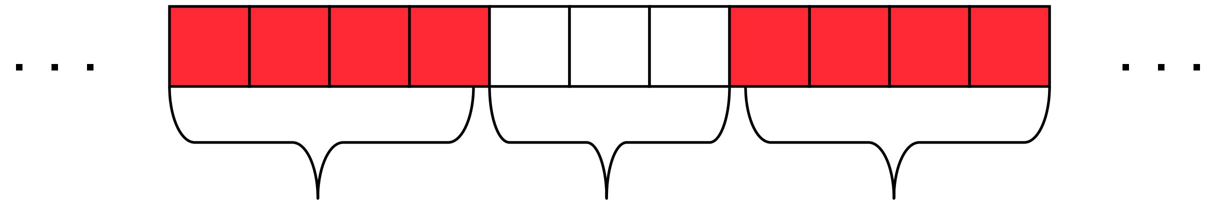
(Saavedra-Barrera benchmark)


450
400
350
300
250
200
150
100
50
0
4
16
64
256
1K
4K
16K
64K
256K
1M
2M
4M
8M
16M32
Stride (bytes)
4KB
8KB
16KB
32KB
64KB
128KB
256KB
512KB
1MB
2MB
4MB
8MB
16MB
32MB
64MB

Experimental effect of prefetching

- Stanza benchmark (Kamil, *et al.*, 2005)

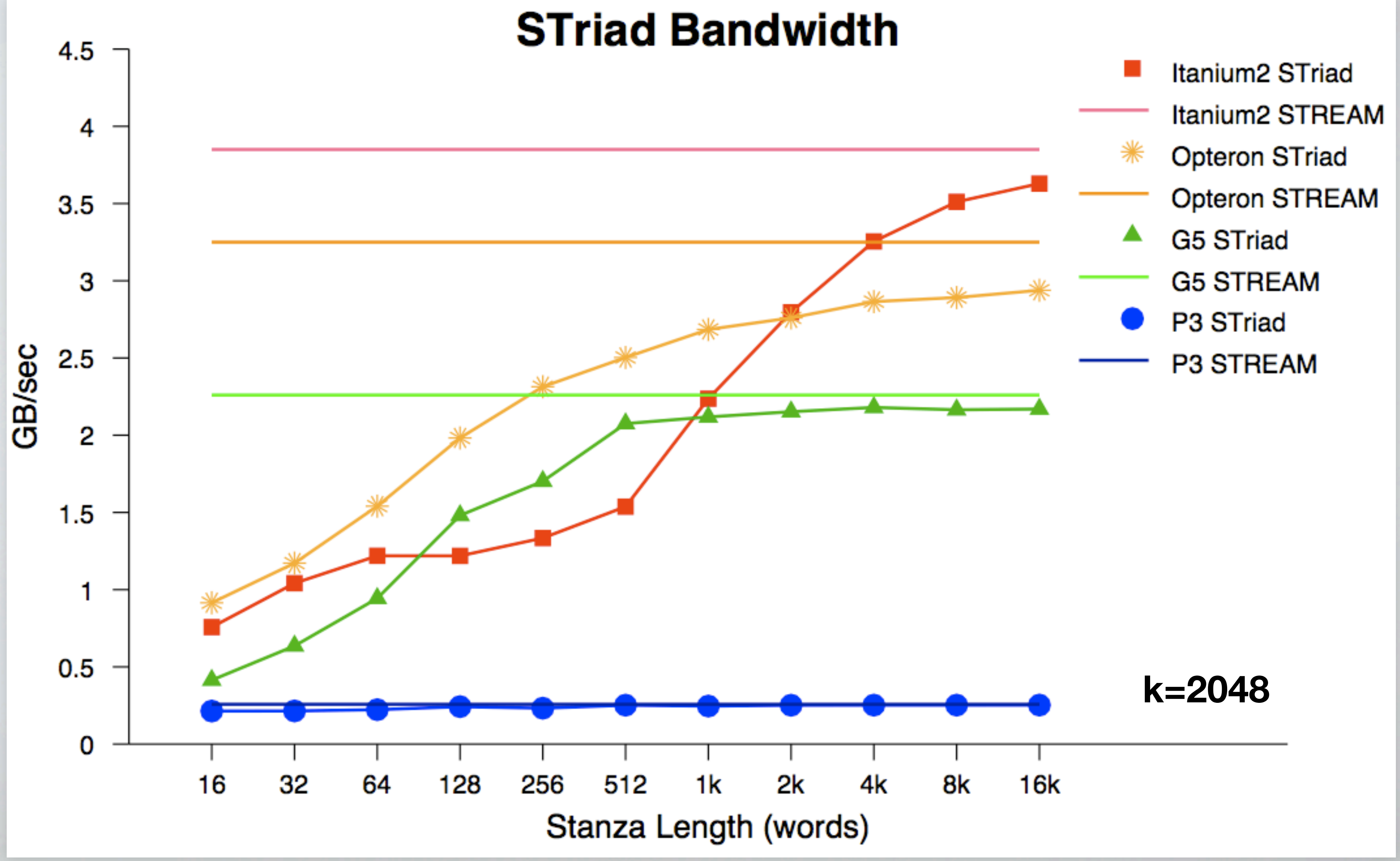
```
while i < n do:
    for each L element stanza do:
        A[i] = a*X[i] + Y[i]
skip k elements
```



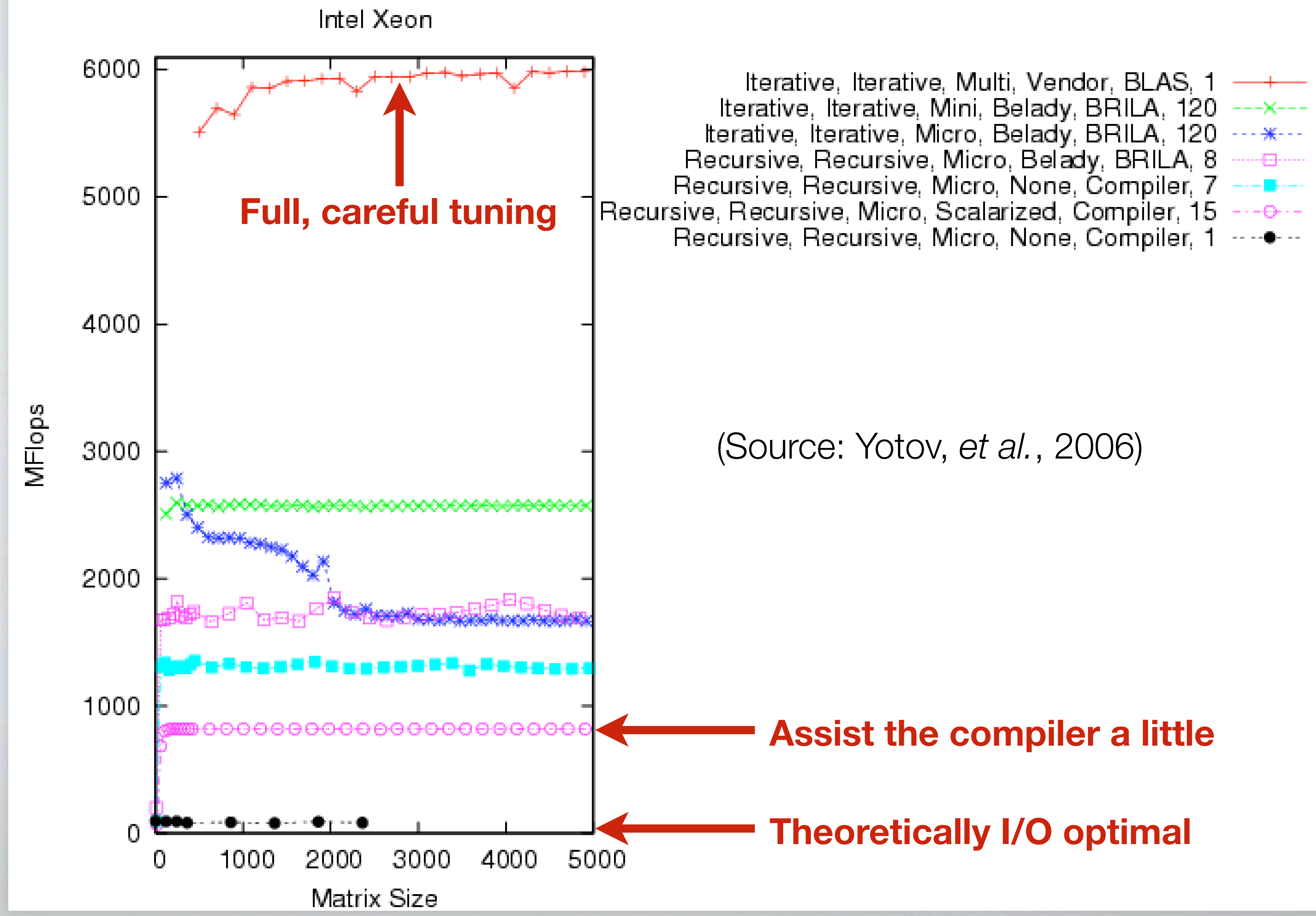
1) do L triads
stanza

2) skip k
elements

3) do L triads
stanza


STriad Bandwidth
GB/sec
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
16
32
64
128
256
512
1k
2k
4k
8k
16k
Stanza Length (words)
Itanium2 STriad
Itanium2 STREAM
Opteron STriad
Opteron STREAM
G5 STriad
G5 STREAM
P3 STriad
P3 STREAM
k=2048

Putting it all together (preview):
Dense matrix-matrix multiply



Full, careful tuning

Assist the compiler a little

Theoretically I/O optimal

(Source: Yotov, *et al.*, 2006)

Summary: Need to consider hardware to get high performance

- Two simple benchmarks show complex behavior
- Performance can be a sensitive function of machine
- Carefully exploiting architectural knowledge can lead to big gains in speed
- Whence simple guiding models?

Administrivia

Administrivia

- Oops, went over time last time (10:55a)
- T-Square site “hosed” ; try again
- Cluster accounts?
- Attendance at SIAM PP is optional (no write-up required), but I may be able to cover registration fees for a few people
- Office hours: T/Th 11a-12p
  - “Physical”: Klaus 1334
  - “Virtual”: via AOL Instant Messenger at “VuducOfficeHours”

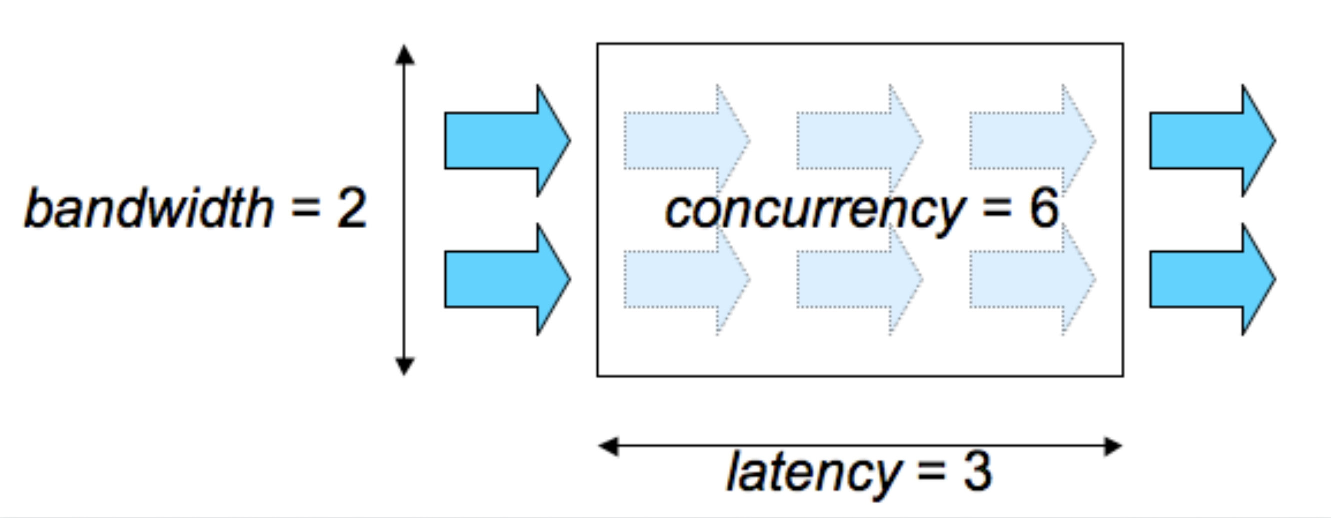
What drives multicore/manycore systems?

Fear of tunnel vision?

- “I think there is a world market for maybe five computers.” T.J. Watson, chairman of IBM, 1943.
- “There is no reason for any individual to have a computer in their home.” Ken Olson, president and founder of Digital Equipment Corporation, 1977.
- “640K [of memory] ought to be enough for anybody.” Bill Gates, chairman of Microsoft,1981.
- “On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it.” Ken Kennedy, CRPC Directory, 1994

Hardware trends driving parallelism

- Architectual limits
  - ILP wall: Gains from ILP seem limited
  - Memory wall: Growing processor-memory gap
  - Power wall: Chips are too hot
- Lots of physical and cost limits

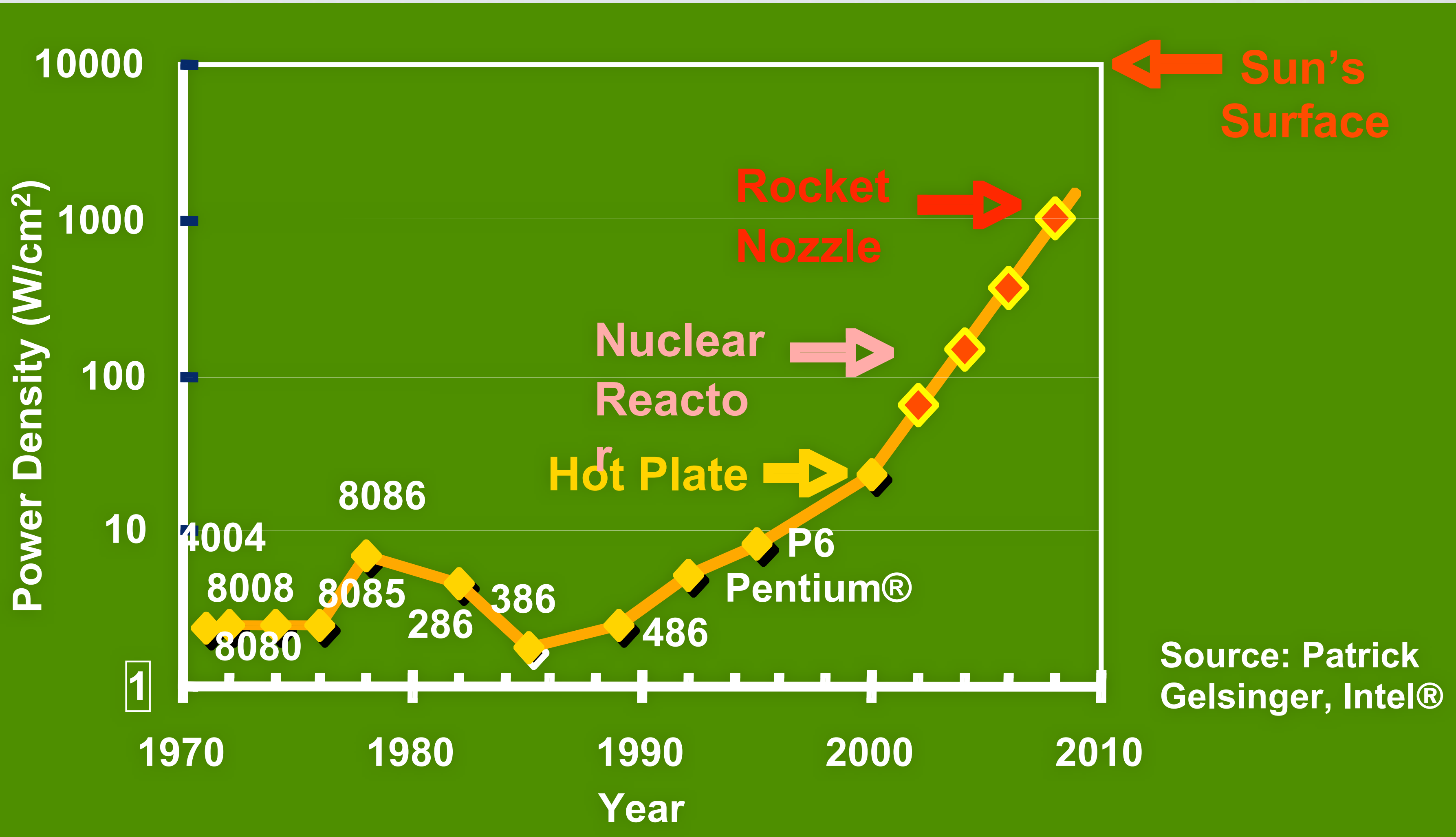


Latency, bandwidth, and concurrency

Little's Law (queuing theory): In any system that transports items from input to output without creating or destroying them,

$$\text{latency} \times \text{bandwidth} = \text{concurrency}$$

Power density

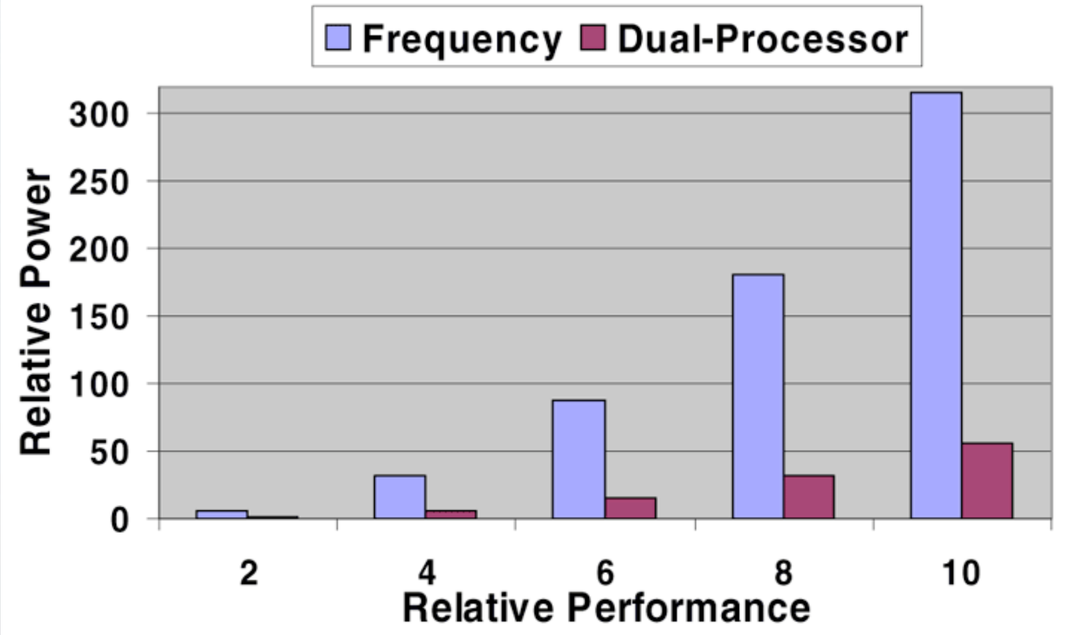


Source: Patrick Gelsinger, Intel®

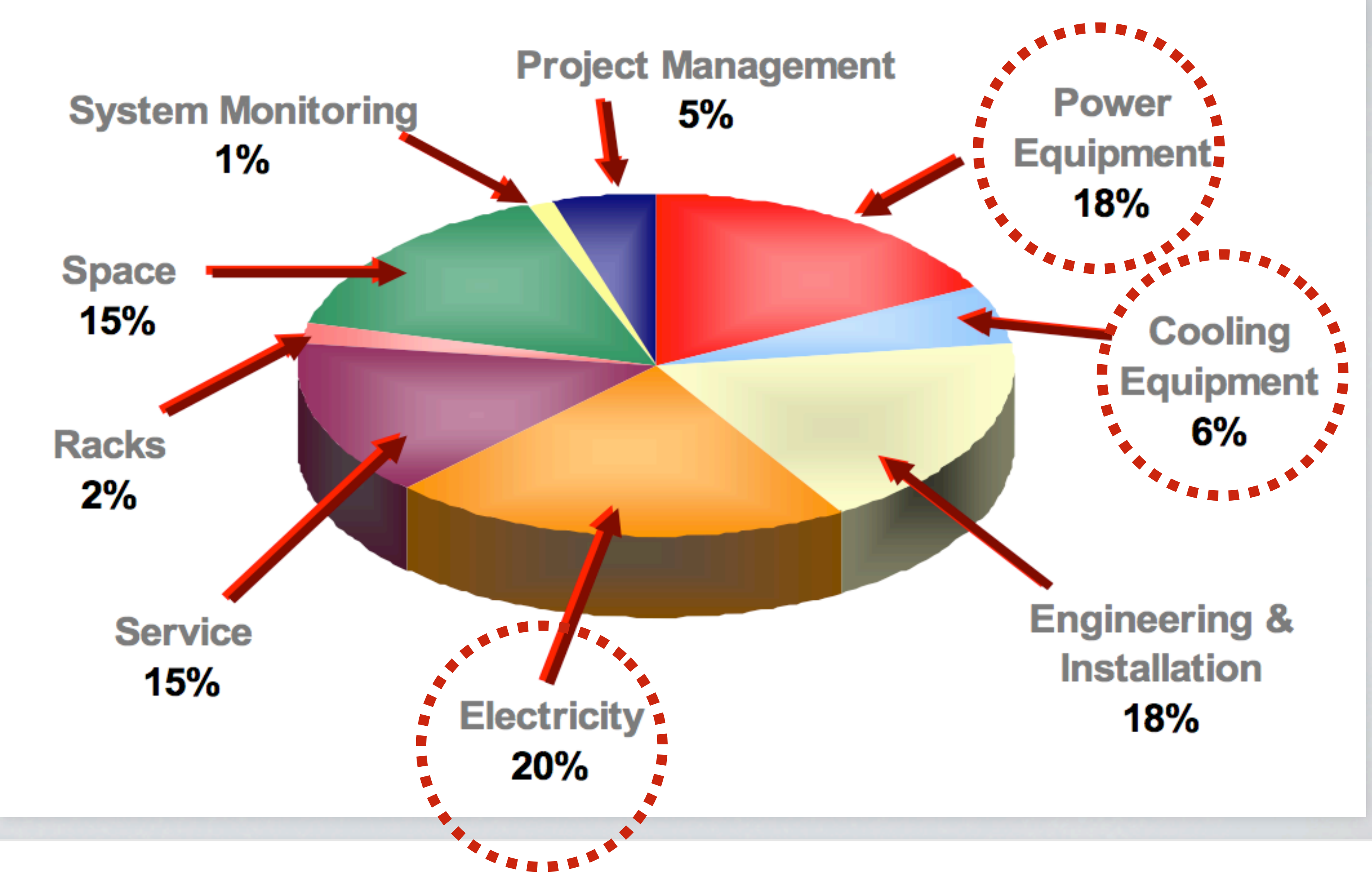
Parallelism saves power

- Rules of thumb
  - Perf. ~ (# cores) * (frequency)
  - Power ~ (capacitance) * $(\text{voltage})^2$ * (frequency)
    - **Power ~ (# cores) * $(\text{freq})^{2.5}$**

- 2x cores, 1/2 freq ⇒ Same perf at ~ 1/3 power



Source: Mendelson, et al., *Intel*



Data center operating costs: Electricity+power+cooling: 44%!

Electricity + power & cooling equipment = 44%!   *Source*: apc.com (2003)

By 2010, servers will consume 3% of all U.S. electricity.

Up from 1.2% in 2005, and 0.6% in 2000.

Source: IDC

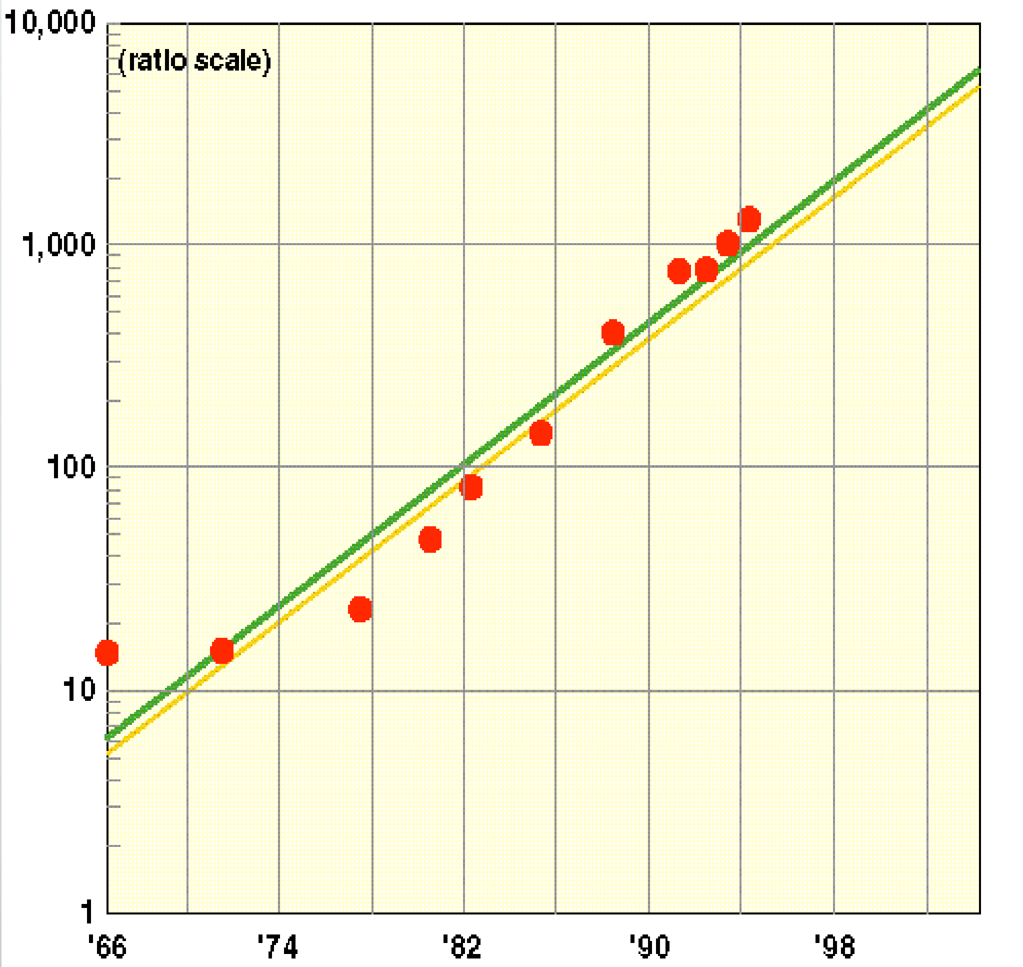
Power usage in data centers is increasing

- Moore’s Law: flop/s ~ 3x every 2 years
- Power efficiency: flop/Joule ~ 2x every 2 years
- Between 2000 and 2006:
  - Computational efficiency increased by 27x
  - Power efficiency increased by 8x
  - At-the-plug power consumption increased by 27/8 ~ **3.4x** in 6 year period!
- By 2009, cost of electricity and cooling will exceed server cost
- Source: K. Brill, Up Time Institute (2006)

Limits on chip yield

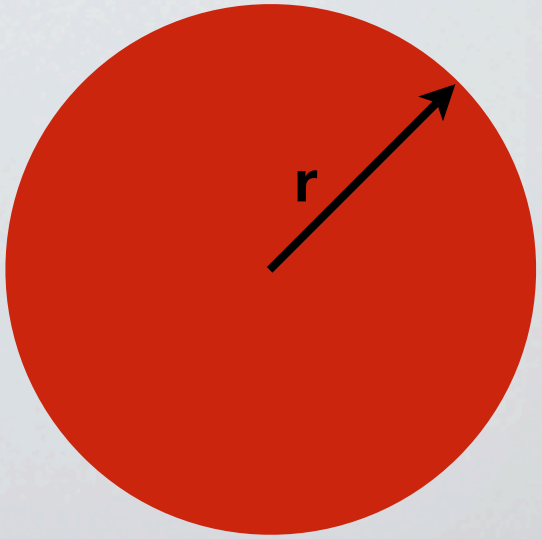
- Moore’s 2nd (Rock’s) Law: fab costs increase
- Yield (usable chips) decreases
- How parallelism helps
    - Using more smaller and simpler cores simplifies verification
    - Can use partially working chips, *e.g.*, PS3’s Cell uses 7 of 8 cores

Cost of semiconductor factories in millions of 1995 dollars

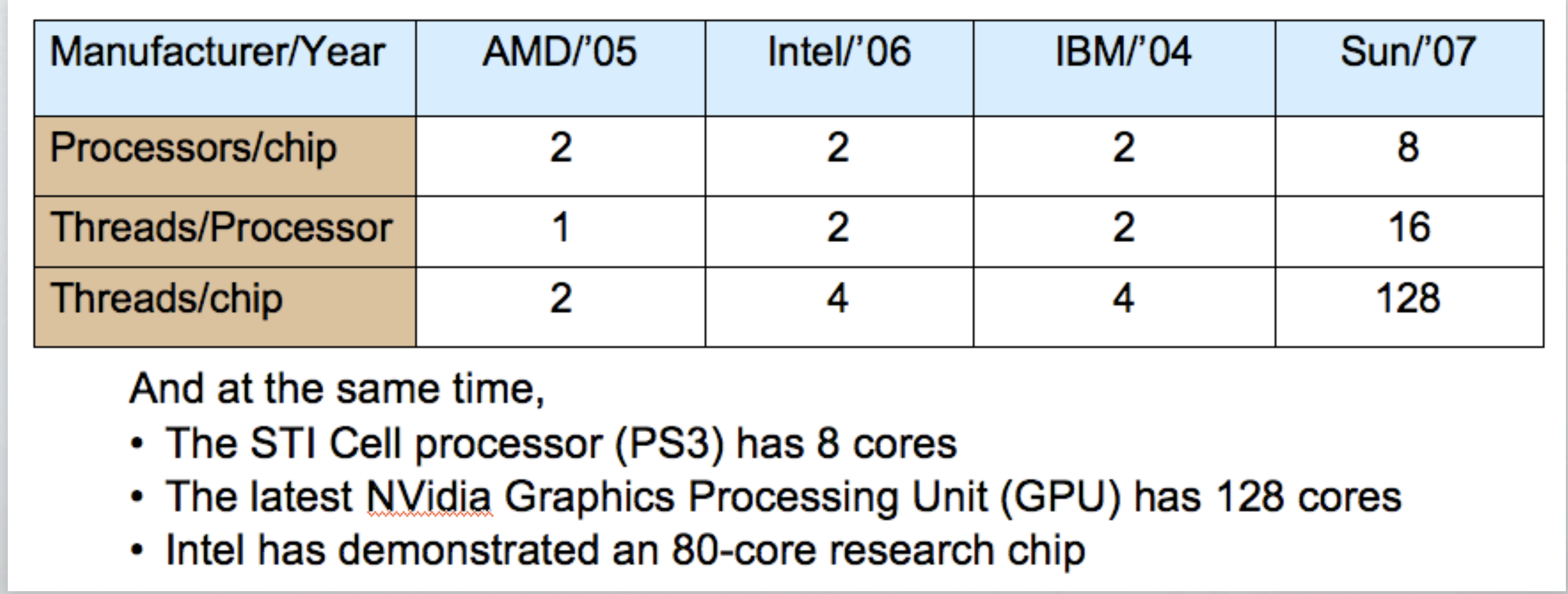


Limits due to the speed of light

- Sequential machine with speed **P** = 1 TFlop/s on a “chip” of radius **r**
- Speed of light **c** ~ $3*10^8$ m/s
- So, **r** < **c** / **P** ~ 0.3 mm
- 1 TB of memory on π***r**^{2} chip area ⇒ ~ 3.5 Angstrom^2 / bit



Current landscape of widely available parallel architectures



Manufacturer/Year	AMD/'05	Intel/'06	IBM/'04	Sun/'07
Processors/chip	2	2	2	8
Threads/Processor	1	2	2	16
Threads/chip	2	4	4	128

And at the same time,

- The STI Cell processor (PS3) has 8 cores
- The latest NVidia Graphics Processing Unit (GPU) has 128 cores
- Intel has demonstrated an 80-core research chip