ECSE 420 Parallel Computing

Zeljko Zilic
McConnell Engineering Building
Room 546

Parallel Computing & World History

- Computers: human invention –a “general purpose” tool
- Parallelism – obvious right from the start
  - Even before computers existed
    - E.g.: pyramids in ancient Egypt
- A Necessity! Especially in High-Performance Computing (HPC)
- Right now: not postponed to future
  - M. Flynn: “Future is parallel” (circa 1996)

Parallelism in World History

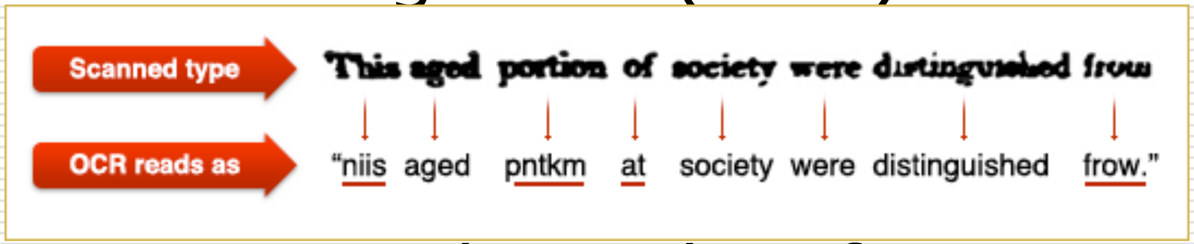
- End of feudal era:
  - Pipelining applied by craftsmen
- Spread of automobiles:
  - Synchronized production line at Ford
- Quantity -> quality concept in Hegel's philosophy:
  - Marx and followers, revolutions, upheavals
- Internet, open source, Google, …,

Parallelism in Nature

- Think of insects, microbes, viruses, plants, …,
  - Quantity -> quality concept at work again
- First objective: survival of the species
- More subtle objectives: getting work done
  - Look at ants, bees, pack of wolves, whales
- Regardless whether large or small
  - Animals, plants, other forms of life benefit by exploiting their strength in numbers

Parallelism and Beings: Farfetched

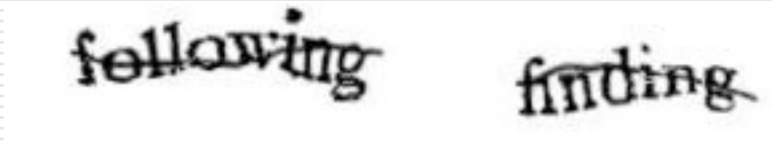
- Ongoing harvesting of human computation (free of charge!)
  - Digital archiving of NYT, word literature, radio
- Optical character recognition (OCR): ~90% accuracy



- Human typists: ~95%, but takes forever, expensive
- Good alternative?

CAPTCHA! to the Rescue

- CAPTCHA!: Completely Automated Public Turing test to tell Computers and Humans Apart



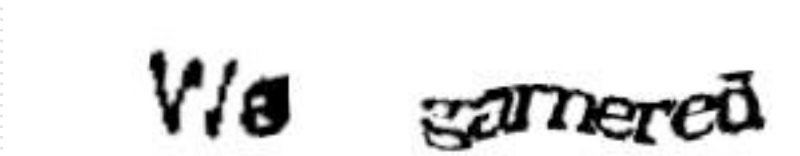
- Humans can recognize distorted text
- Great role in protecting from spam, protecting registrations
  - authorizing joining e-mail accounts, discussion groups, …

Underground Beating CAPTCHA!

- Include captcha's at entrance to pornographic sites/pictures
  - Free workforce
  - Limited in scope
- Hiring human readers - sweatshops
  - Costs money
  - IP detection issue
  - Time to react

Doing Useful Work for Free: reCAPTCHA

- OCR: humans better than machines
- Place scanned text as captcha's
- Two words at least: known + unknown



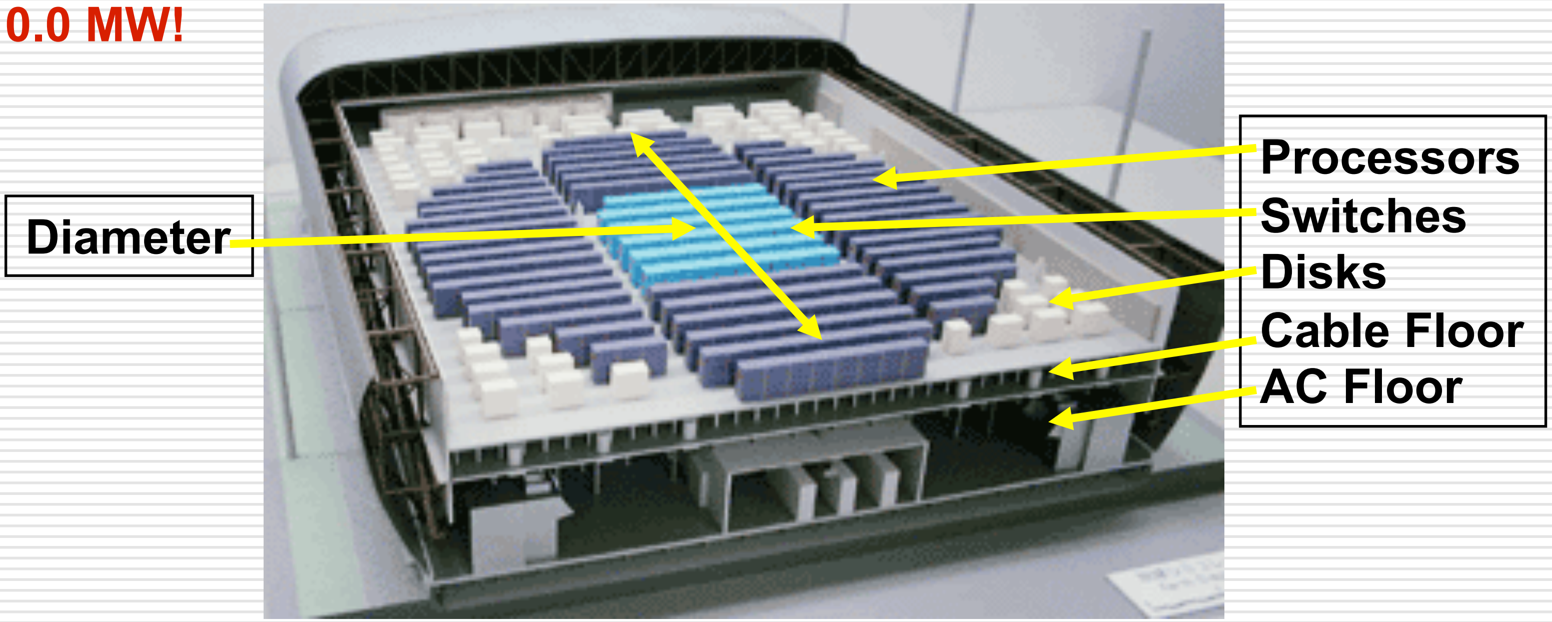
- Known captcha sweatshops get whole paragraph to type
- NYT almost done, others moving fast

Parallel Computing: Goals

- Pulling together compute resources to solve challenging computing tasks
- Keeping execution correctness while doing above (w.d.a.)
- Keeping productive w.d.a.
- Keeping electric distribution alive w.d.a.
- Having sufficient cooling w.d.a.
- Keeping existing computer room w.d.a. or
- Having enough money for sustaining the above

Case in Point: Earth Simulator

35.86 Tflop/s (#4), Footprint — 34,000 ft² (4 tennis courts x 3 floors)
10.0 MW!



Crossbar Interconnection Network
83000 Copper Cables
1800 Miles of Cable
http://www.es.jamstec.go.jp/esc/eng/index.html

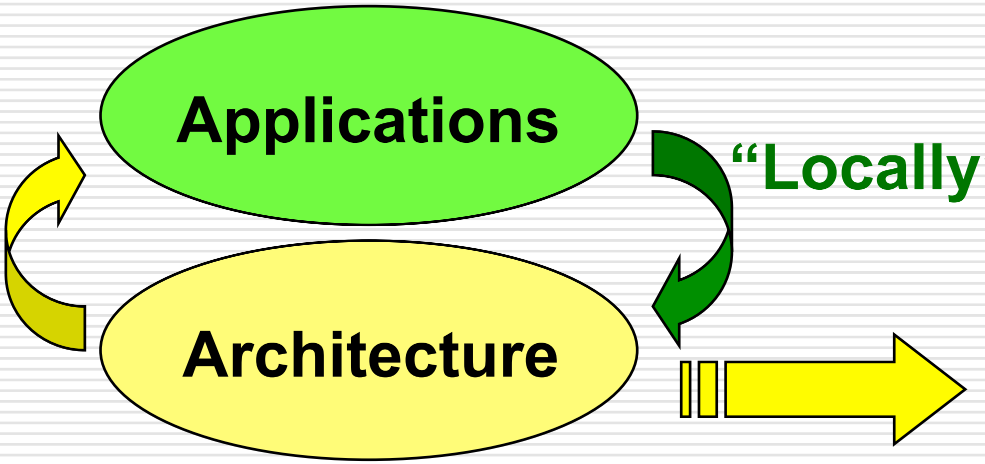
↔ **High Interprocessor Latency (11 in = 1ns)**

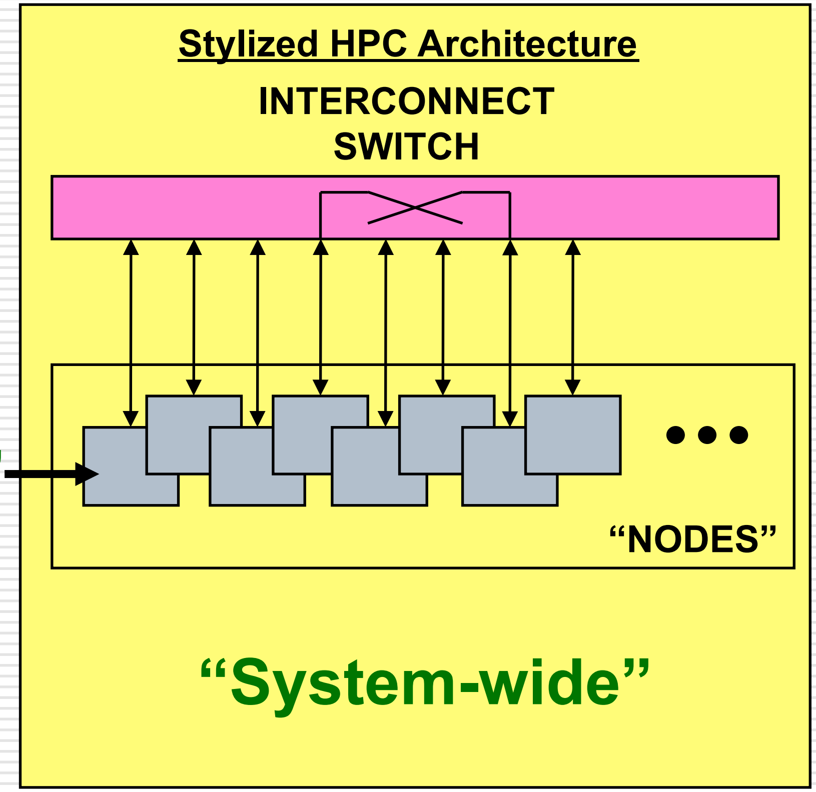
Parallel Computing Disciplines

- Architecture
- Operating Systems
- Programming Languages
- Compilers
- Programming techniques
- Algorithms (conceptual)
- Important application types
  - Databases, numerical linear algebra, modeling, intelligence (both meanings), CAD, visualization
- Opportunistic parallelism exploitation
  - SETI, spam generators, all SW on multiple-core PCs
- Remember: parallel/concurrent computing is a necessity

Application/Architecture Challenge

- Performance beyond a single (commodity) processor is only possible as a result of concurrency (parallelism) in applications
- Hierarchical application characteristics
  - "In the small" i.e., "Locally"
  - "In the large" i.e., "System-wide"





This Course: Focus

- Realistic architecture exposure
  - Shared-memory multiprocessing, symmetric (SMP)
    - Dual and multiple core general purpose processors
  - Distributed memory
    - Message-passing paradigm
  - Some research exposure of lecturer:
    - On-chip multiprocessing
    - Non-Uniform Shared Memory (NUMA)
- Programming
  - Concurrent – still on one processor
  - Parallel – using explicitly multiple processors
  - Distributed – using multiple computers

Learning Objectives

- Critical understanding of parallel and distributed systems
  - Performance measures
  - Difficulties and tradeoffs
  - Trends
- Concurrency issues in SW
- Start-to-end parallel computing project

Quiz: Opposite to Parallel?

- a.) Perpendicular?
- b.) Meandering?
- c.) Serial?
- Answer: think “Turing Machine” (T.M.)
  - T.M. : universal processor/computer model
  - Scans infinite tape with symbols (program or data)
  - On every new symbol, moves and produces output by rules defined via a finite state machine (**sequential**)
- T.M. is as serial as it can be, but is usually referred to as “sequential”

What is Parallel (in Computing)?

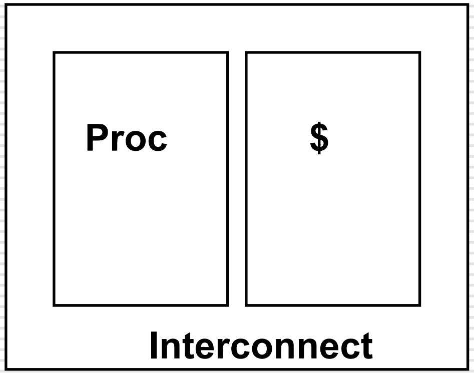
- Circuit
  - Analog, digital, quantum; combinational, sequential, …
- Single processor + specialized circuit (possibly reconfigurable)
  - FPGA computing machine
- Single pipelined processor, single superscalar processor, single multithreaded processor
- Single SIMD processor
  - Also: vector processor/machine
- Multiple processors executing single program
  - Shared- or distributed-memory, …,
- Multiple computers executing single program
  - Distributed or distributed-shared memory

Parallelism Available

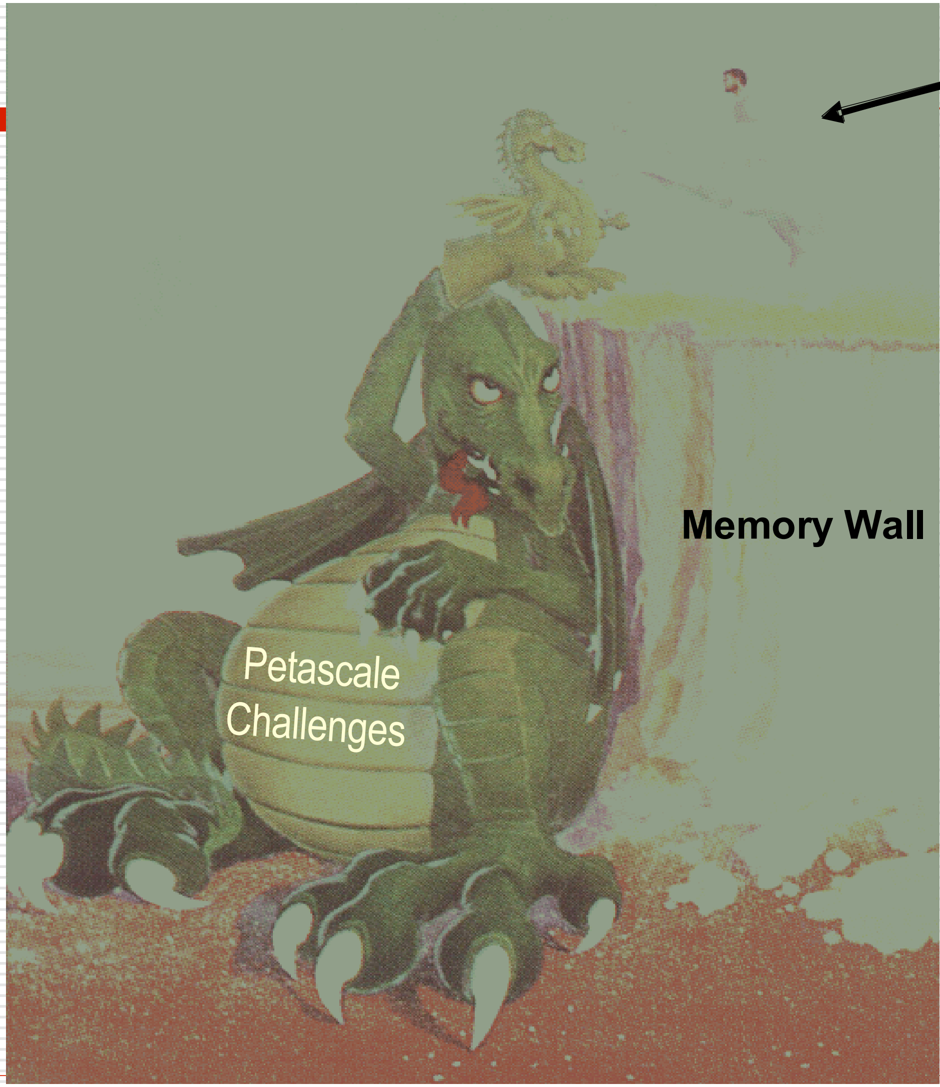
- Bits
- Operations
  - Add, subtract, multiply, …
  - Instruction-level (ILP)
    - How many processor instructions in parallel?
- Thread-level
  - How many threads at once
- Process-level – as above, less used
- Task-level
- Coarse-level: complete programs (or so)

Technology: A Closer (Rough) Look

- Basic advance is (was?) *decreasing feature size* (λ)
  - Circuits become either faster or lower in power
- Die size is growing too
  - Clock rate improves roughly proportional to λ
  - Number of transistors improves like λ^2 (or faster)
- Performance > 100x per decade
  - clock rate < 10x, rest is transistor count
- *How to use more transistors?*
  - Parallelism in processing
    - multiple operations per cycle reduces CPI
  - Locality in data access
    - avoids latency and reduces CPI
    - also improves processor utilization
  - Both need resources, so tradeoff
- *Fundamental issue- resource distribution, as in uniprocessors*



Challenges: Performance at Scale



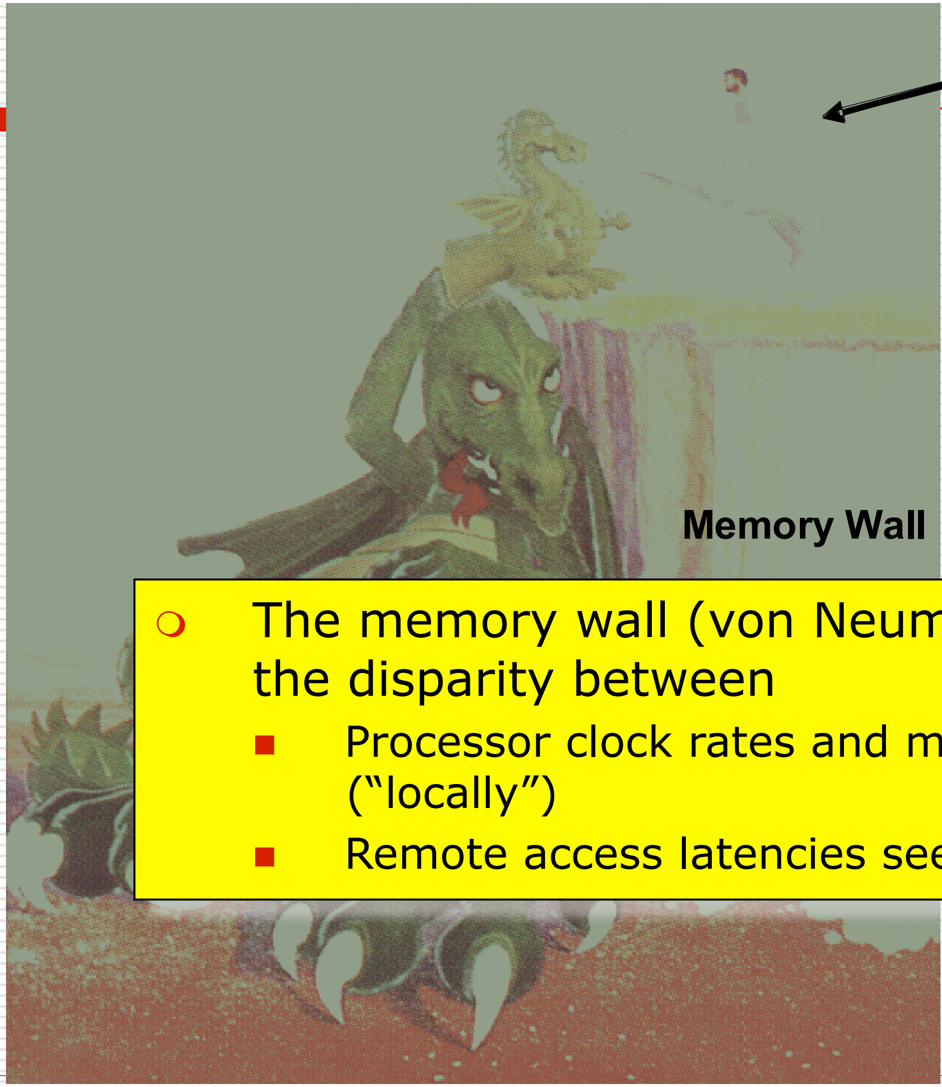
Advanced simulation and modeling apps

Conquering Terascale problems of today

Beware being eaten alive by the petascale problems of tomorrow.

Drawing by Thomas Zacharia (ORNL)

Performance at Scale



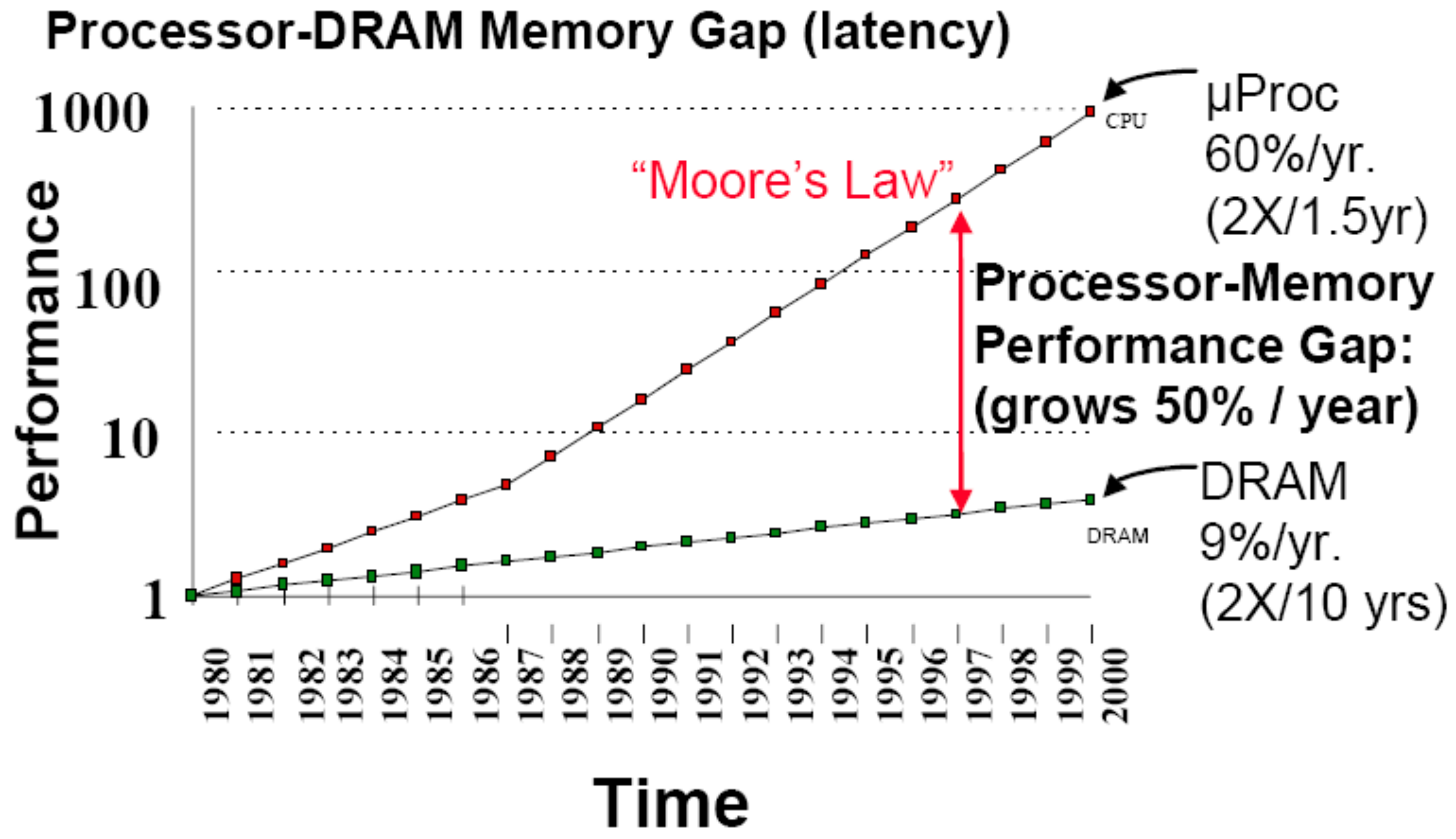
Advanced simulation and modeling apps

Conquering Terascale problems of today

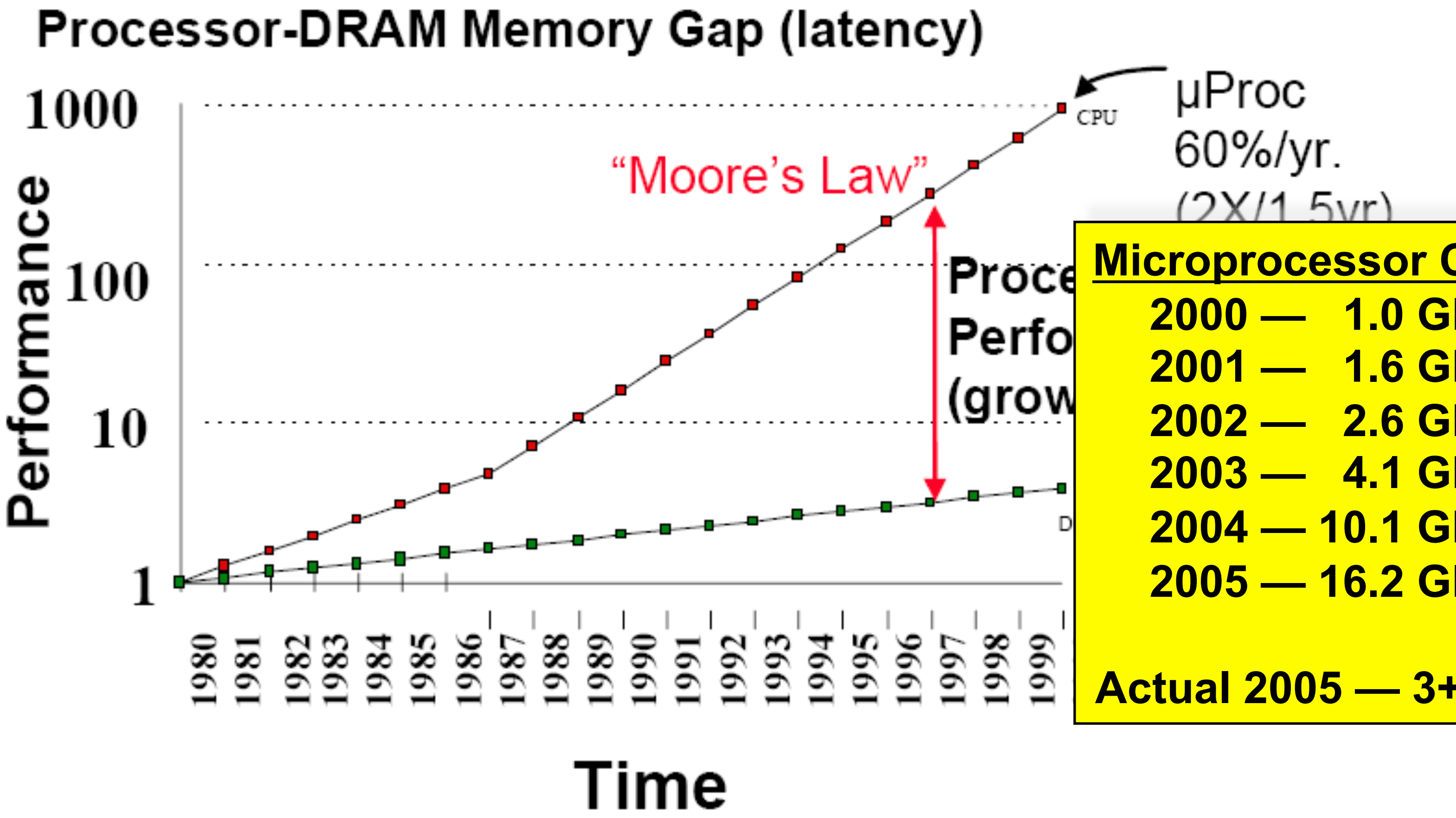
- The memory wall (von Neumann bottleneck) — the disparity between
  - Processor clock rates and memory cycle times ("locally")
  - Remote access latencies seen "system wide"

Drawing by Thomas Zacharia (ORNL)

The Memory Latency Wall



The Memory Latency Wall

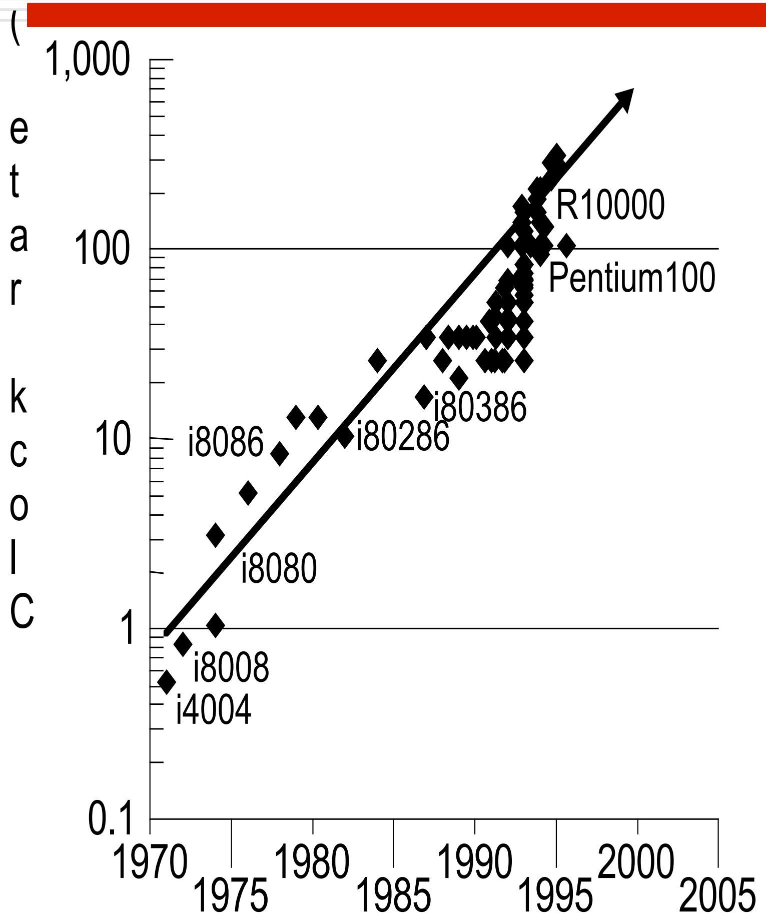


Microprocessor Clock

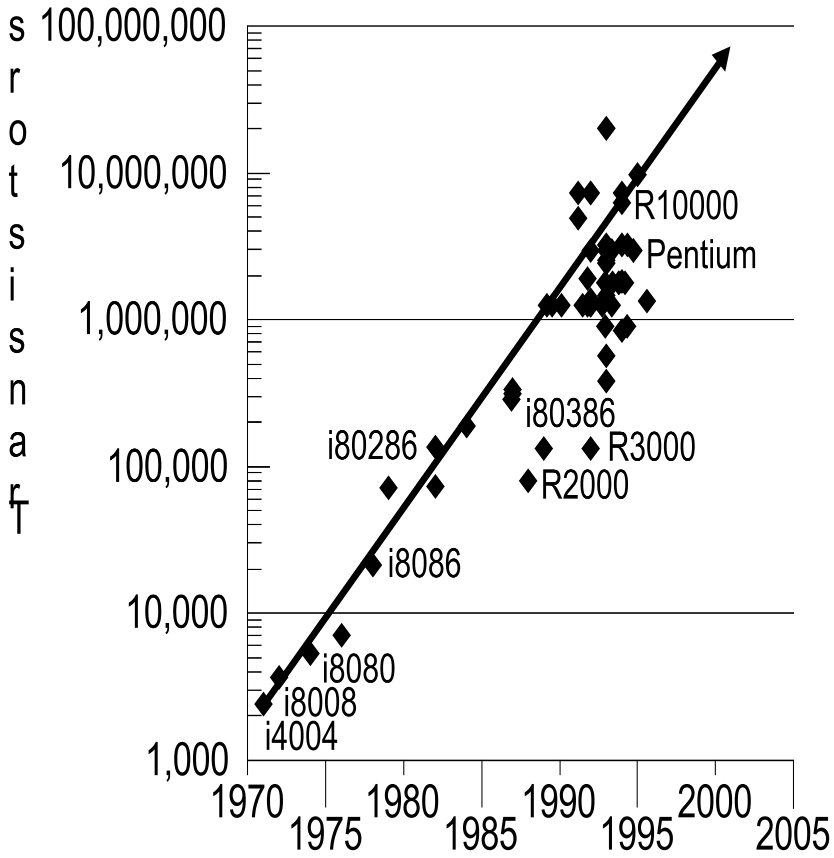
2000 — 1.0 GHz
2001 — 1.6 GHz
2002 — 2.6 GHz
2003 — 4.1 GHz
2004 — 10.1 GHz
2005 — 16.2 GHz

Actual 2005 — 3+ GHz

Growth Rates



- 30% per year



40% per year

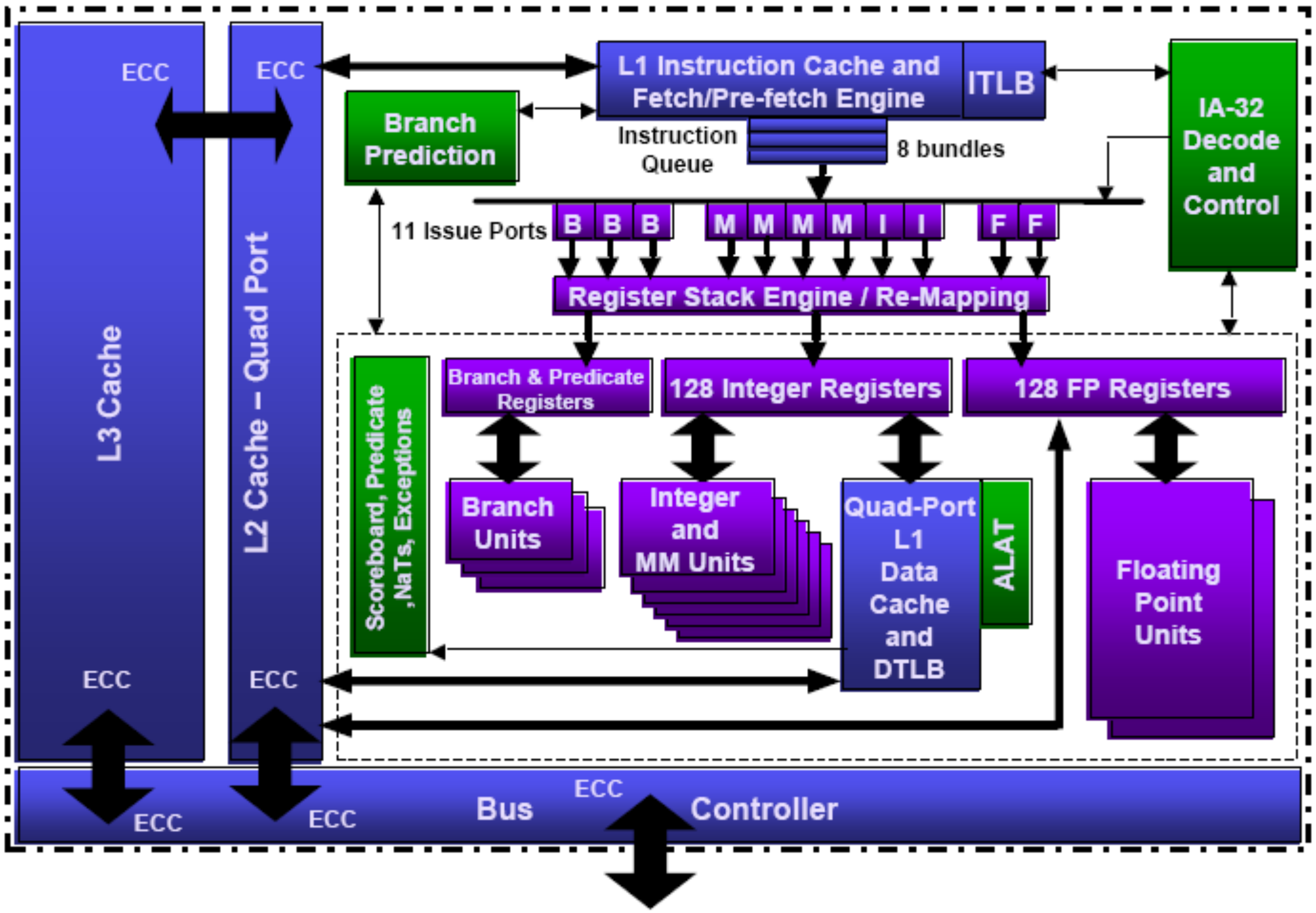
Architectural Trends

- Architecture: technology gains -> performance and functionality
- Tradeoff between parallelism and locality
  - Past microprocessors: 1/3 compute, 1/3 cache, 1/3 off-chip connect
  - Tradeoffs change with scale and technology advances
    - Most area taken by memories
- Understanding microprocessor architectural trends

  => Build intuition about design issues or parallel machines

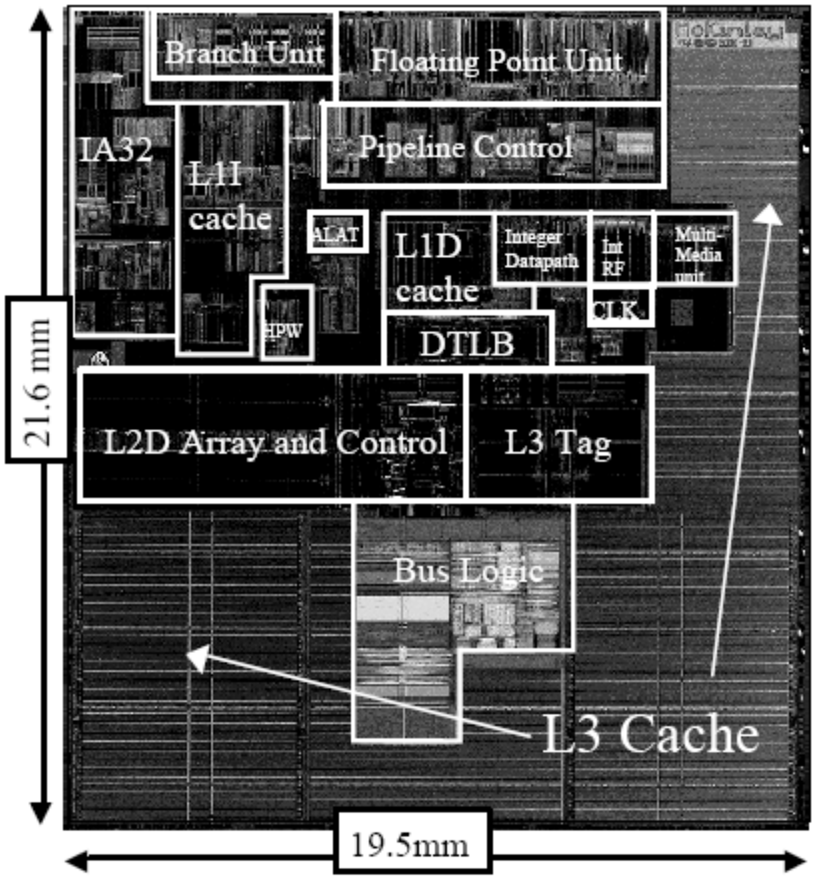
  => Fundamental role of parallelism even in “sequential” computers

Itanium Block Diagram

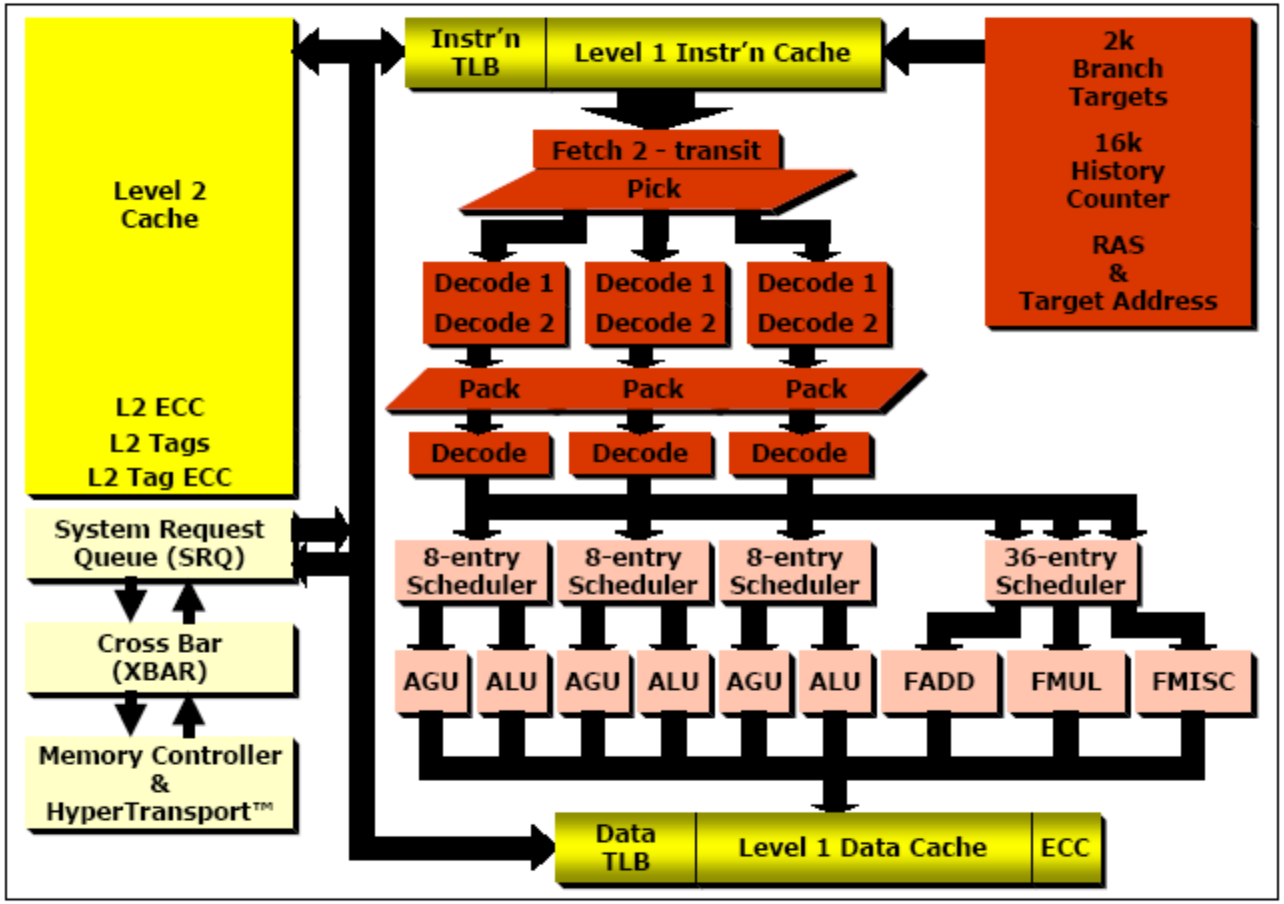


Itanium McKinley – A HPC Processor

- **.18µm bulk, 6 layer Al process**
- **8 stage, fully stalled in-order pipeline**
- **Symmetric six integer issue design**
- **3 levels of cache on-die totaling 3.3MB**
- **221 Million transistors**
- **130W @1GHz, 1.5V**



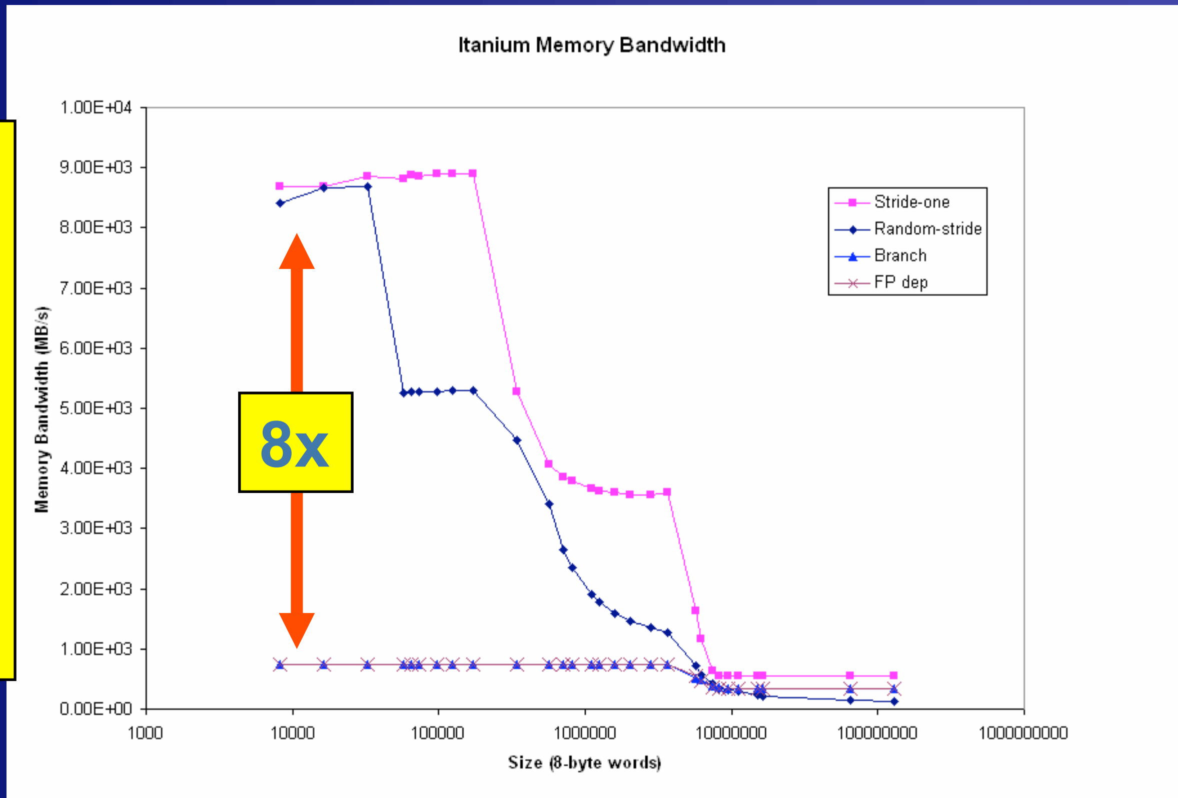
AMD Hammer uArchitecture



- **12-stage integer operation pipeline**
- **17-stage floating point operation pipeline**

Branch/Data Dependency - Itanium

Framework addition: Data Dependency

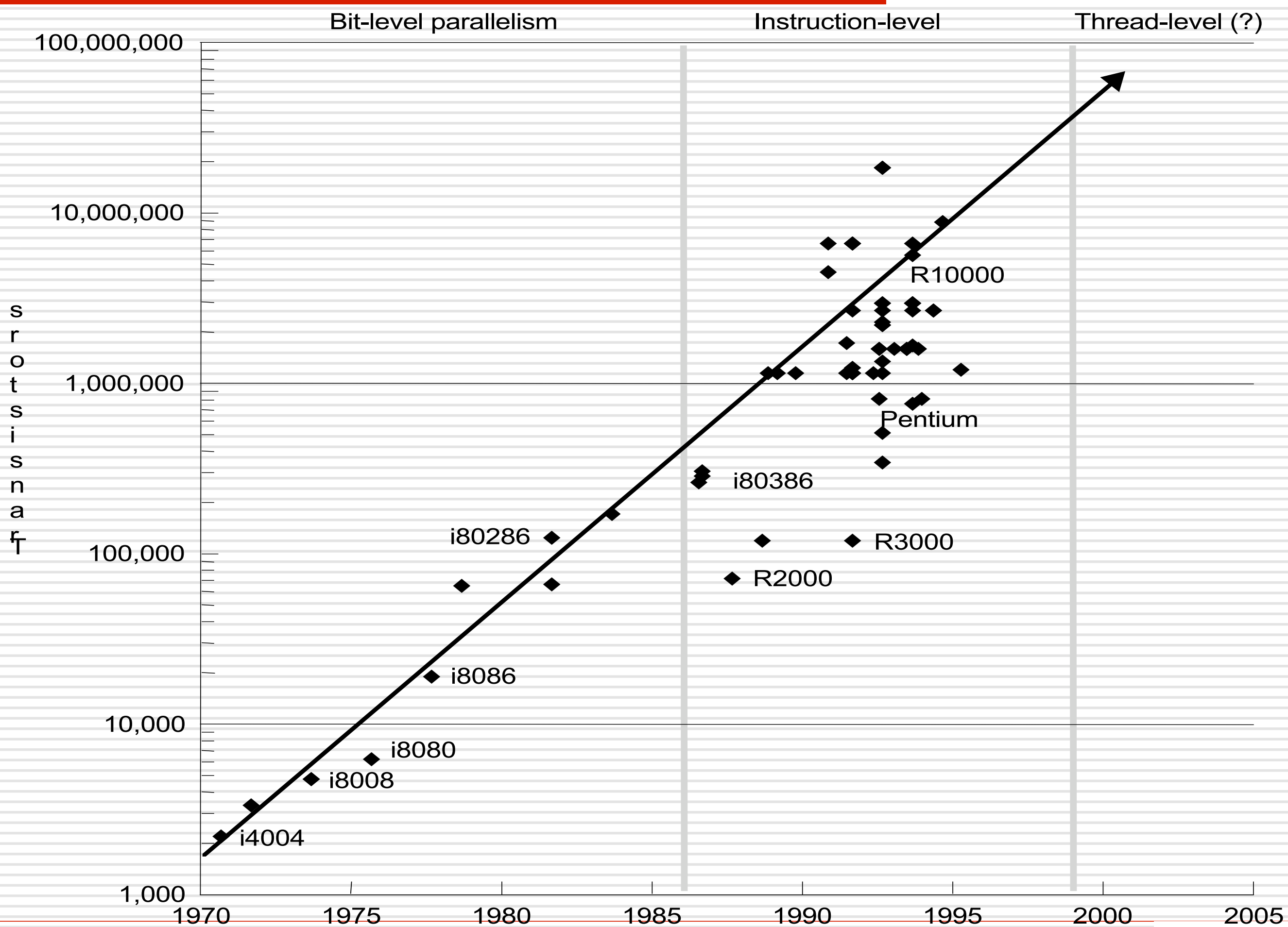


If you know about an 8x performance degradation

You may try to avoid it!

PMaC Performance Modeling and Characterization Lab
San Diego Supercomputer Center

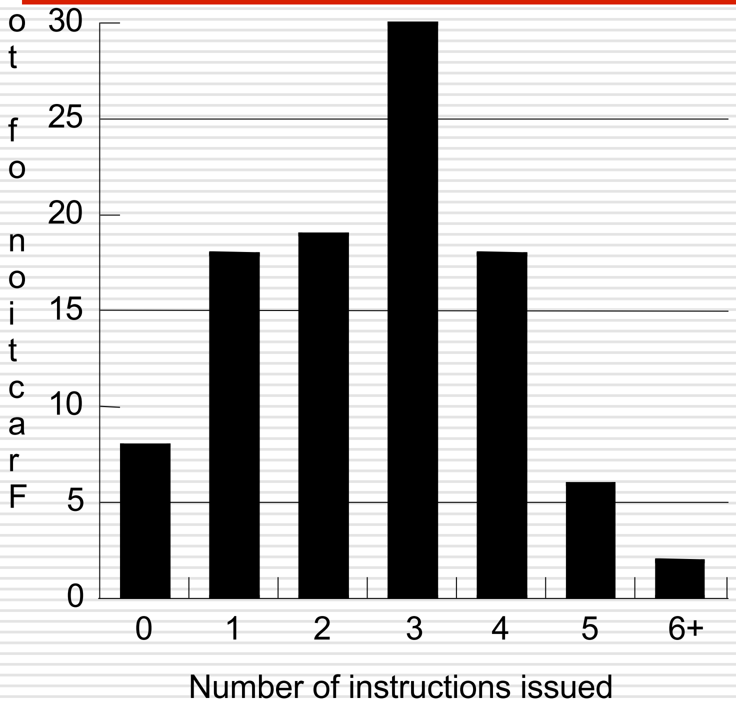
Phases in VLSI Generation

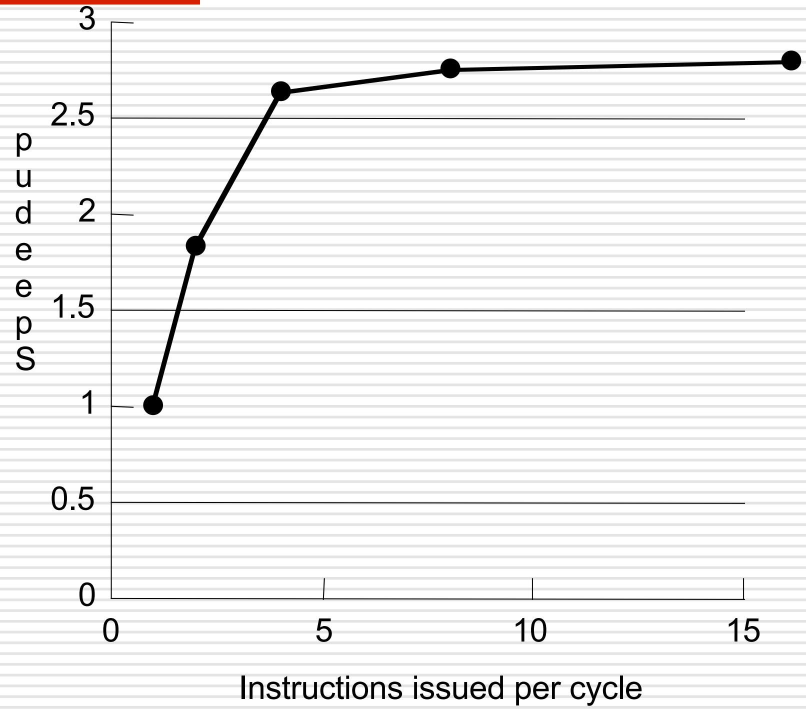


Architectural Trends

- Main trend in VLSI is increase in parallelism
  - Up to 1985: bit level parallelism: 8 bit -> 16-bit
    - Inflection at 32 bit – cache fits on a chip
    - Adoption of 64-bit under way, 128-bit far (no need)
  - Mid 80s to mid 90s: instruction level parallelism
    - Pipelining and RISC instruction sets + compiler
    - On-chip caches and functional units => superscalar
    - Out of order execution, speculation, prediction
      - Deals with control transfer and latency problems
  - Now: thread level parallelism
    - Hardware multi-threaded processors
    - Multi-core processors

How Far ILP Goes?

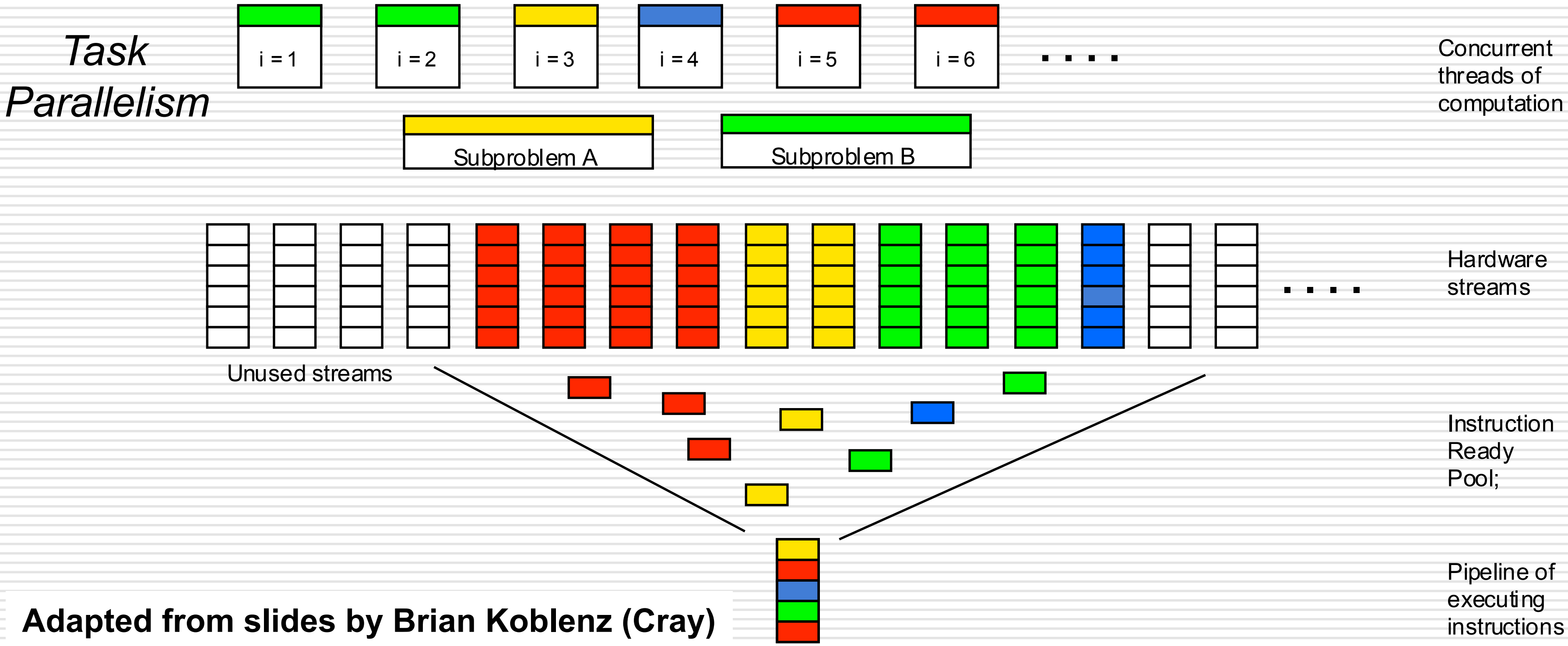




- Assumptions: infinite resources and fetch bandwidth, perfect branch prediction and renaming

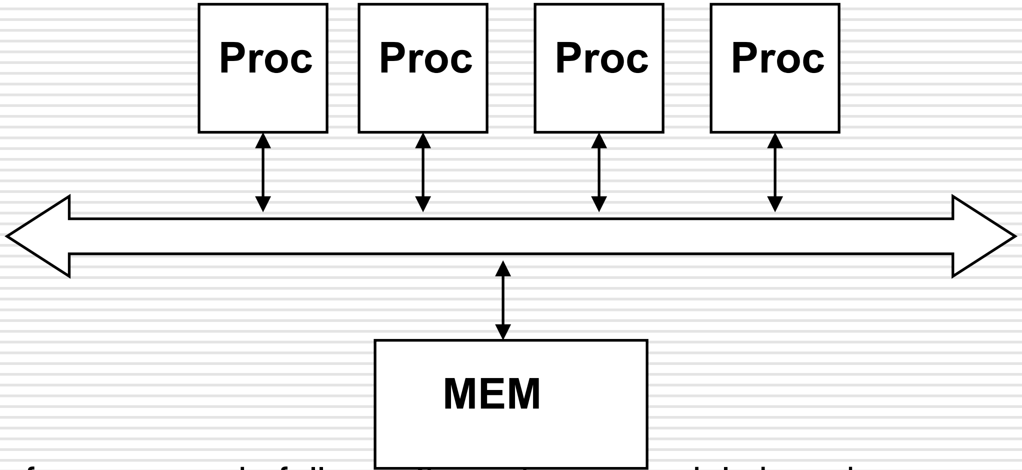
Tolerating Latency - Multithreading

- Multiple active threads - long latency filled by instructions in other threads
- Tolerate latency and keep pipelines full
- Also tolerate branch latency and memory-based synchronization



Adapted from slides by Brian Koblenz (Cray)

Thread Level Parallelism Use



No. of processors in fully configured commercial shared-memory systems

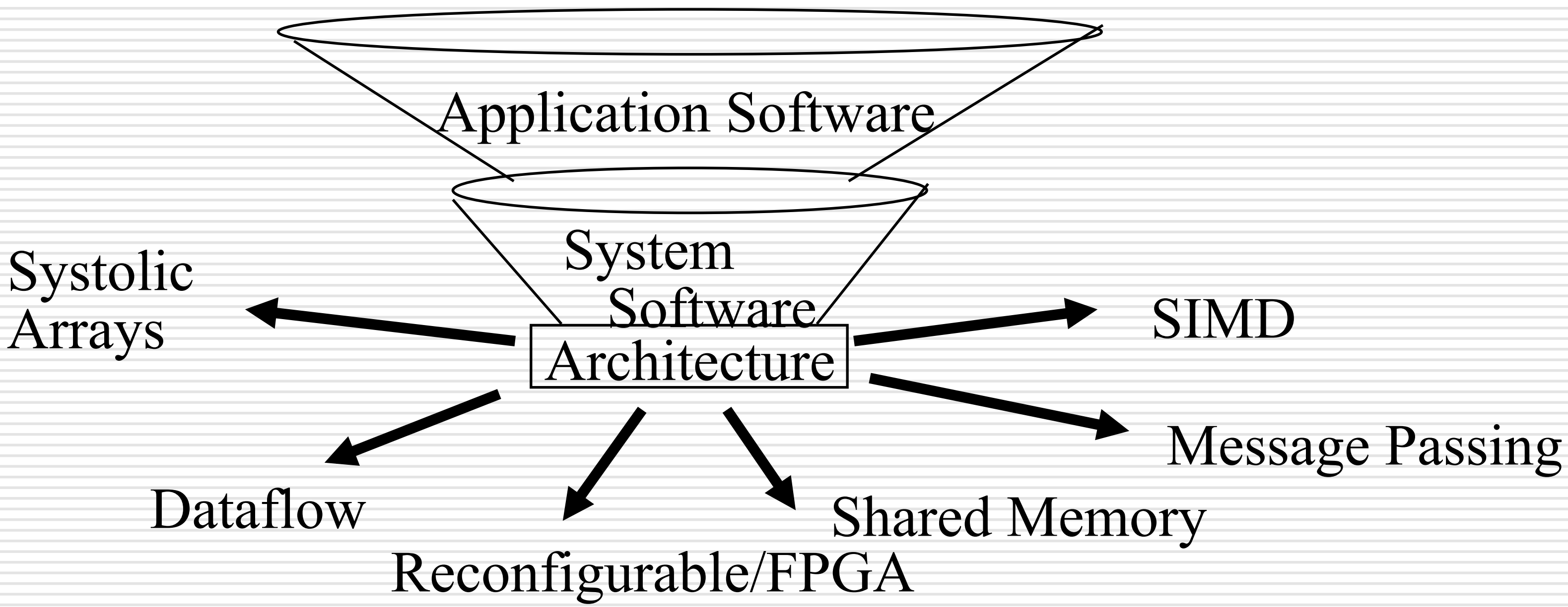
- Multiple CPUs with shared memory
  - dominates server and enterprise market, moving down to desktop
- More responsive with multiple CPUs
  - OS Kernel just picks another processor for next thread

Some Taxonomies

- Number of Instructions/Data
  - SISD – serial
  - SIMD
  - MISD
  - MIMD
- Parallelism planes
  - Control-flow
  - Data
- Others: Single Program, Multiple Data (SPMD)

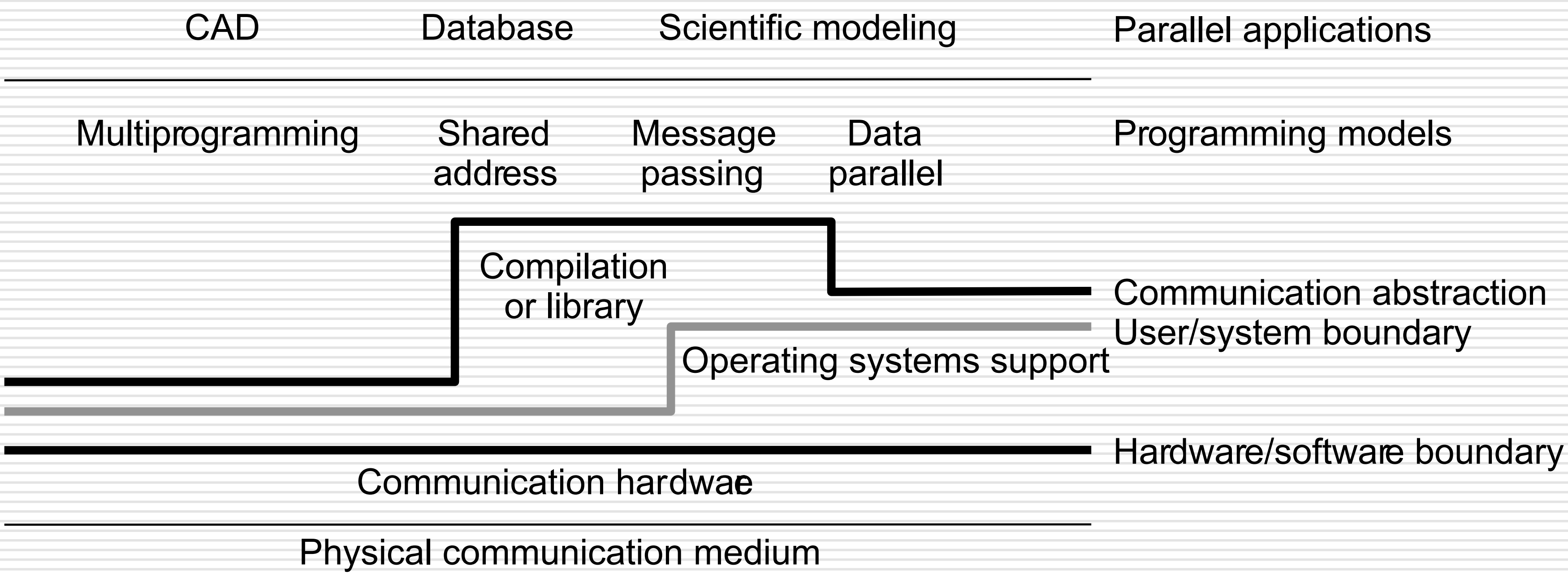
Parallel Architectures so Far

In Past: Divergent architectures, no predictable pattern of growth.



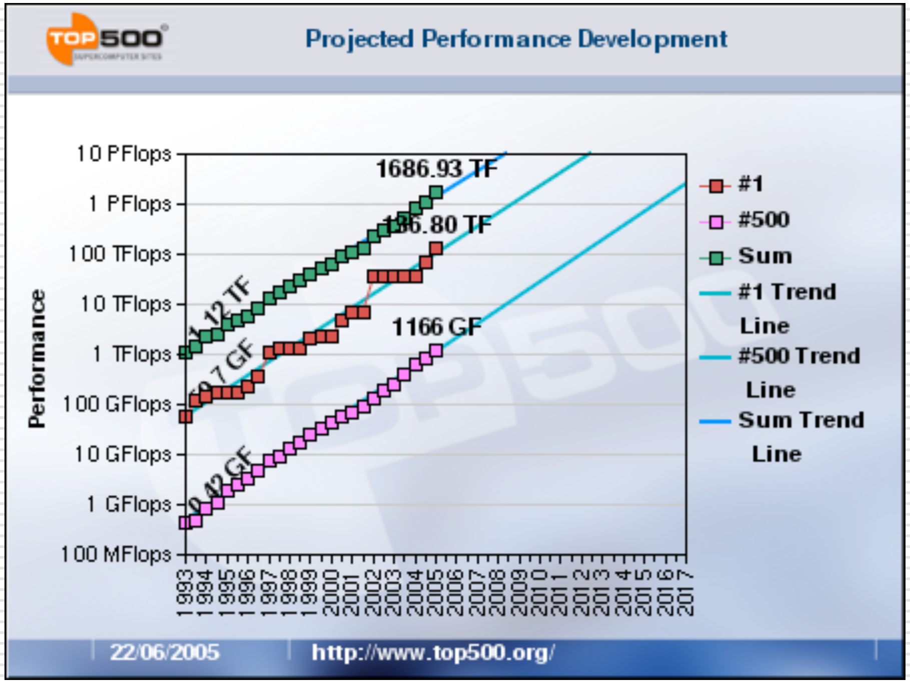
- Productivity crisis: software development suffers with dissimilar models

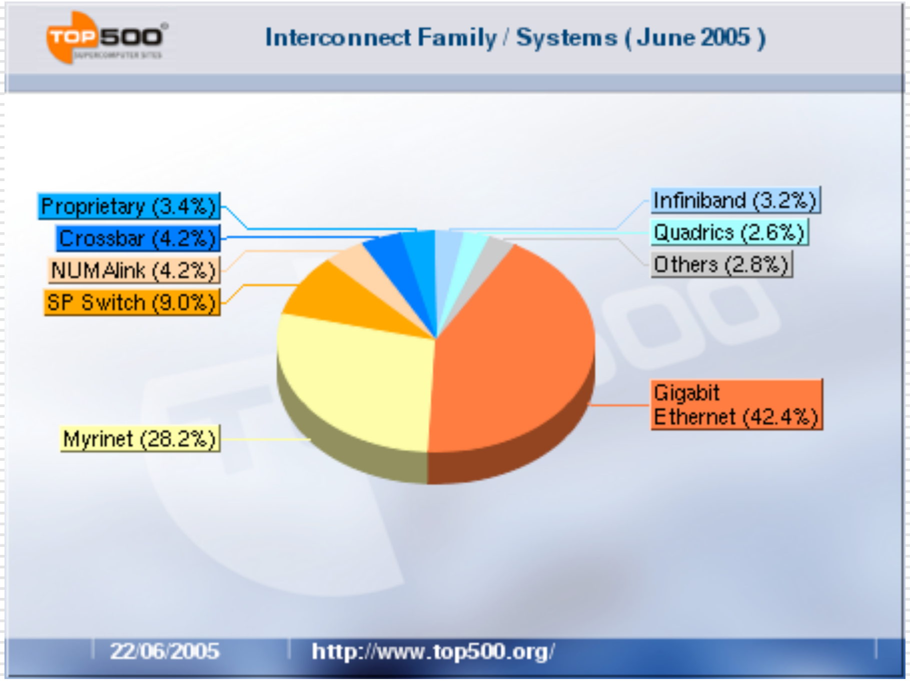
Modern Layered Framework



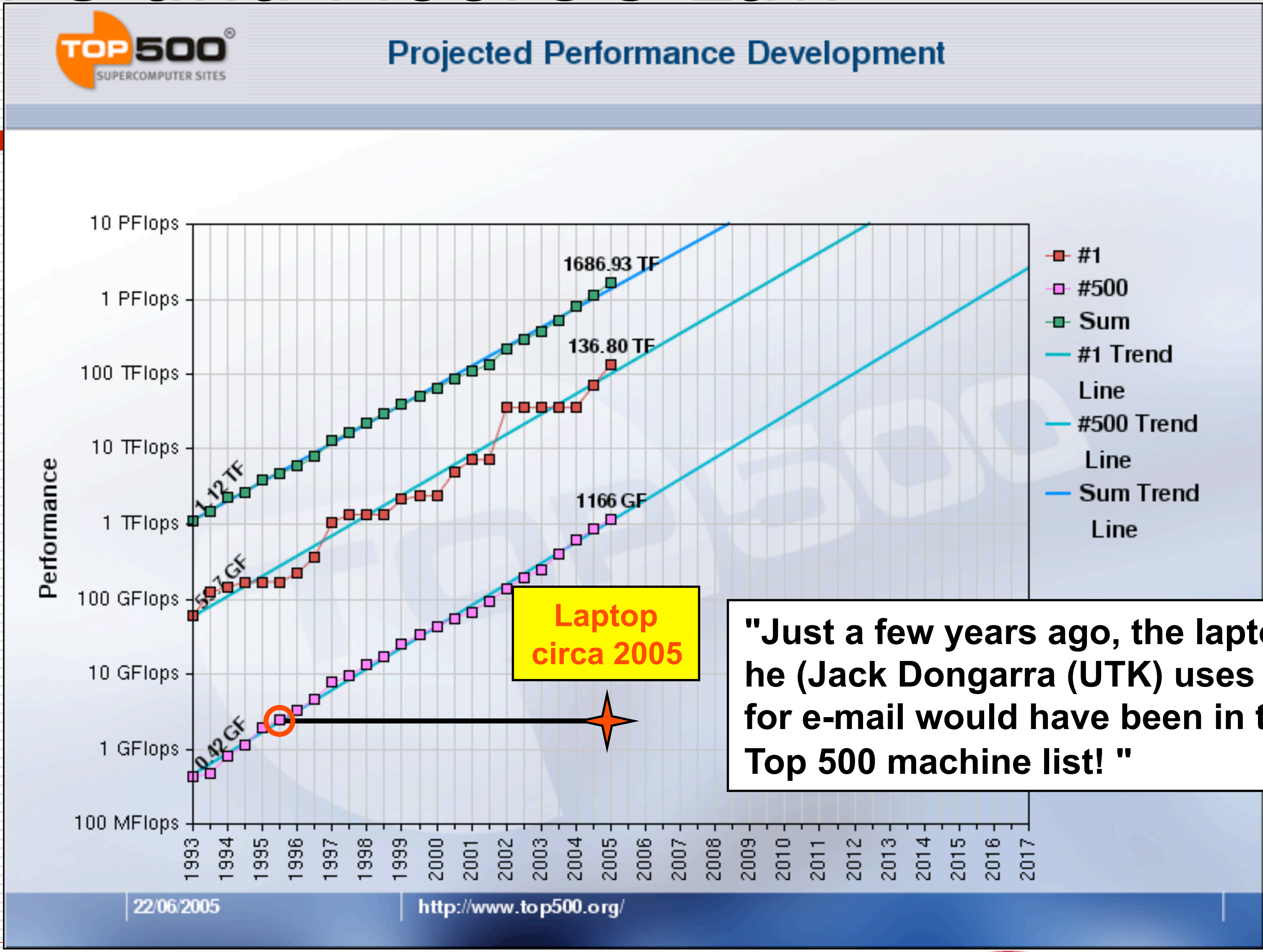
Fastest Computers

- www.top500.org – trends, makers, applications, users



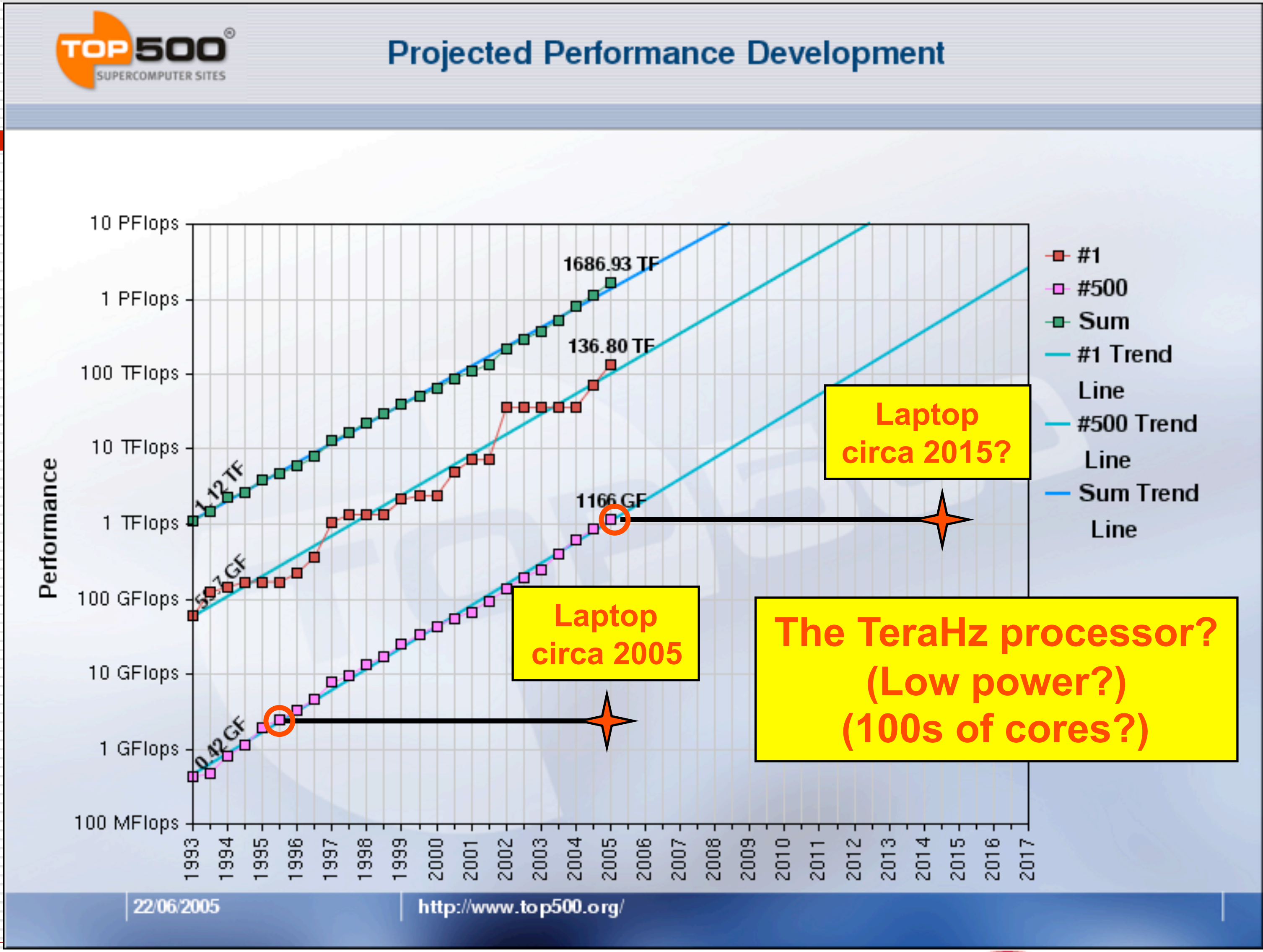


HPC and Moore’s Law



"Just a few years ago, the laptop he (Jack Dongarra (UTK) uses for e-mail would have been in the Top 500 machine list! "

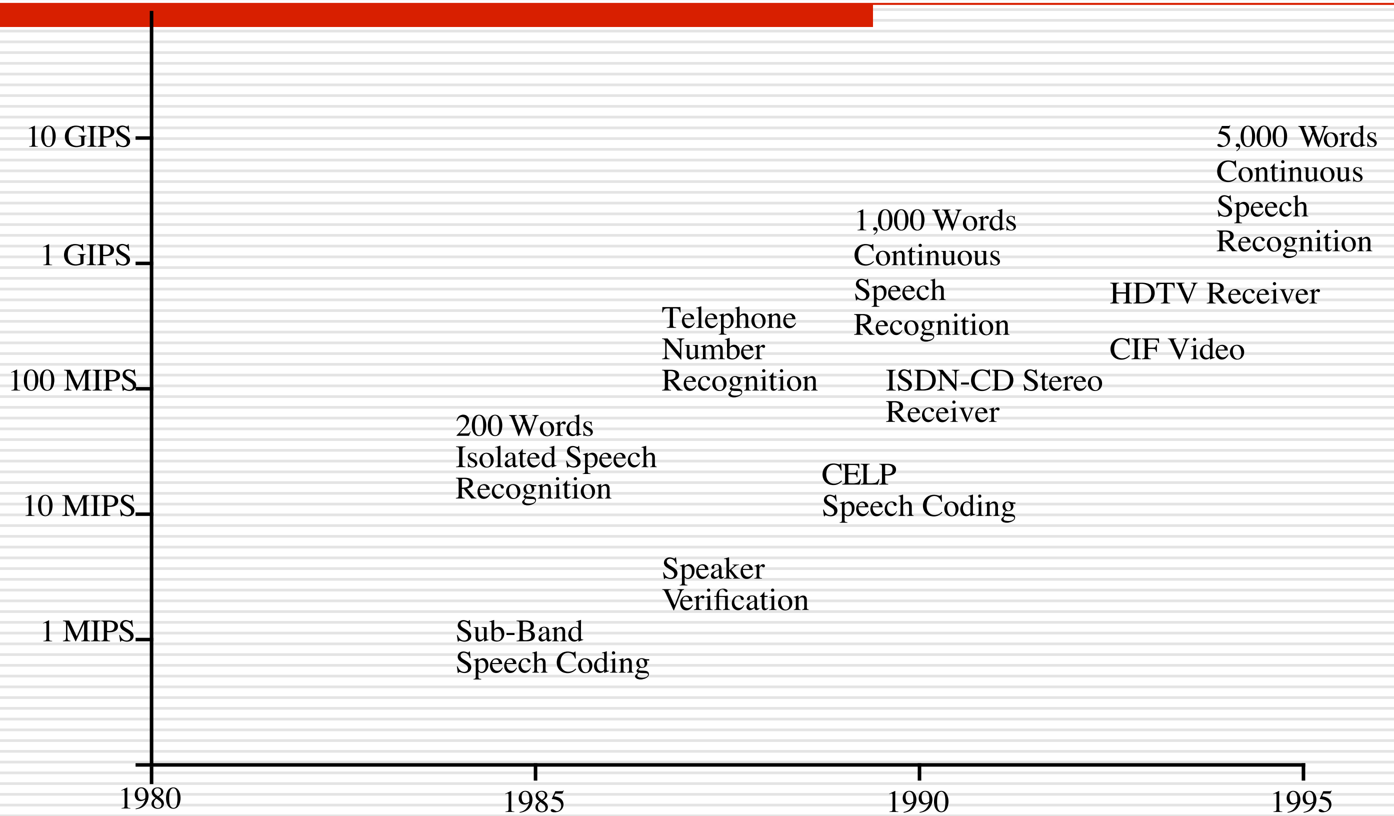
HPC and Moore's Law



Engineering Computing Demand

- Large parallel machines a must in many industries
  - Petroleum (reservoir analysis)
  - Automotive (crash simulation, drag analysis, combustion efficiency),
  - Aeronautics (airflow analysis, engine efficiency, structural mechanics, electromagnetism),
  - Computer-aided design
  - Pharmaceuticals (molecular modeling)
  - Visualization
    - in all of the above
    - entertainment (films like Toy Story)
    - architecture (walk-throughs and rendering)
  - Financial modeling (yield and derivative analysis)

Applications: Speech and Image Processing



- Also CAD, Databases, . . .
- *100 processors gets you 10 years, 1000 gets you 20 !*

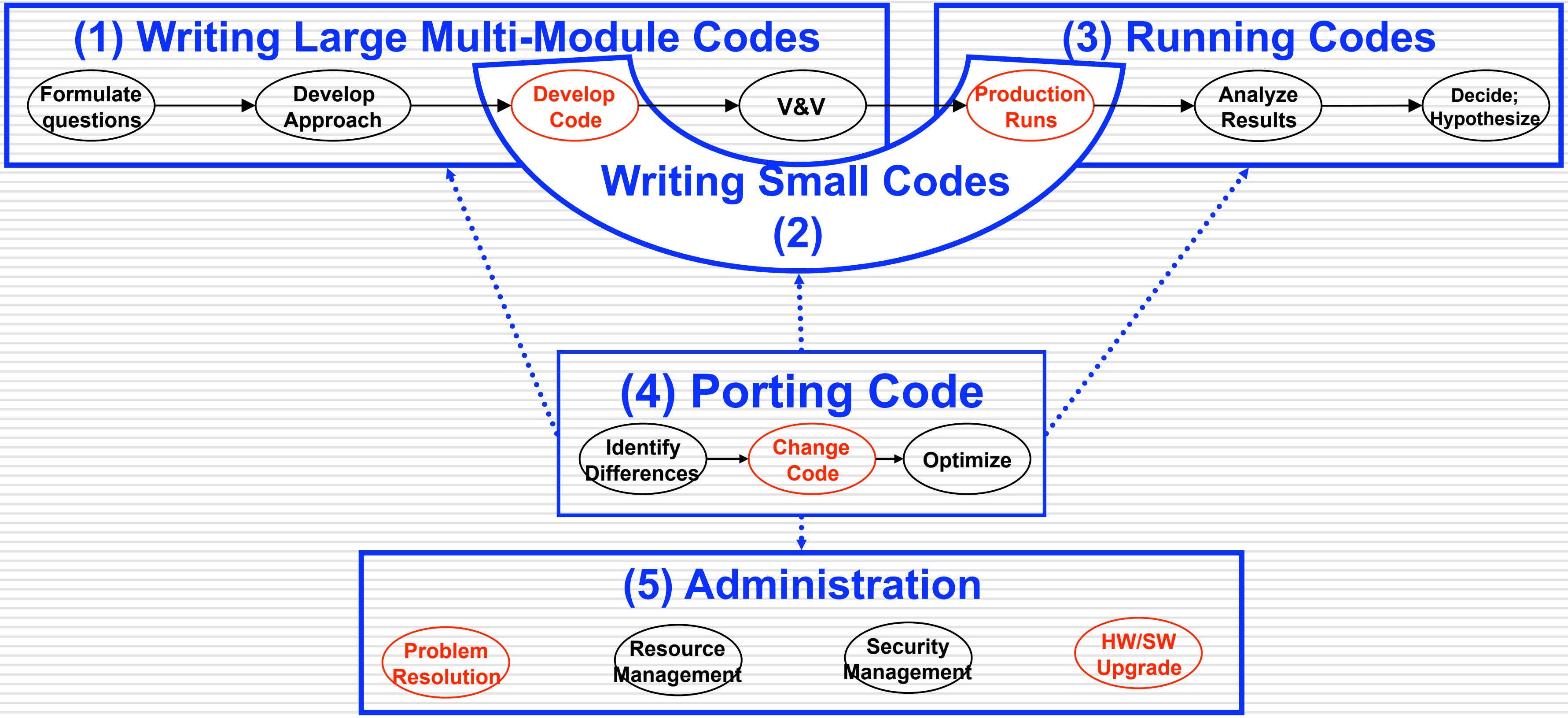
HPC Programming - Challenges

```
// Gaussian Elimination
for (k = 0 to n-1)
{
        for (i = k+1 to n)
        {
                z = A[i][k] / A[k][k];
                for (j = k+1 to n)
                        A[i][j] = A[i][j] - z * A[k][j]; y[i] = y[i] - z * y[k];
        }
}
```

HPC

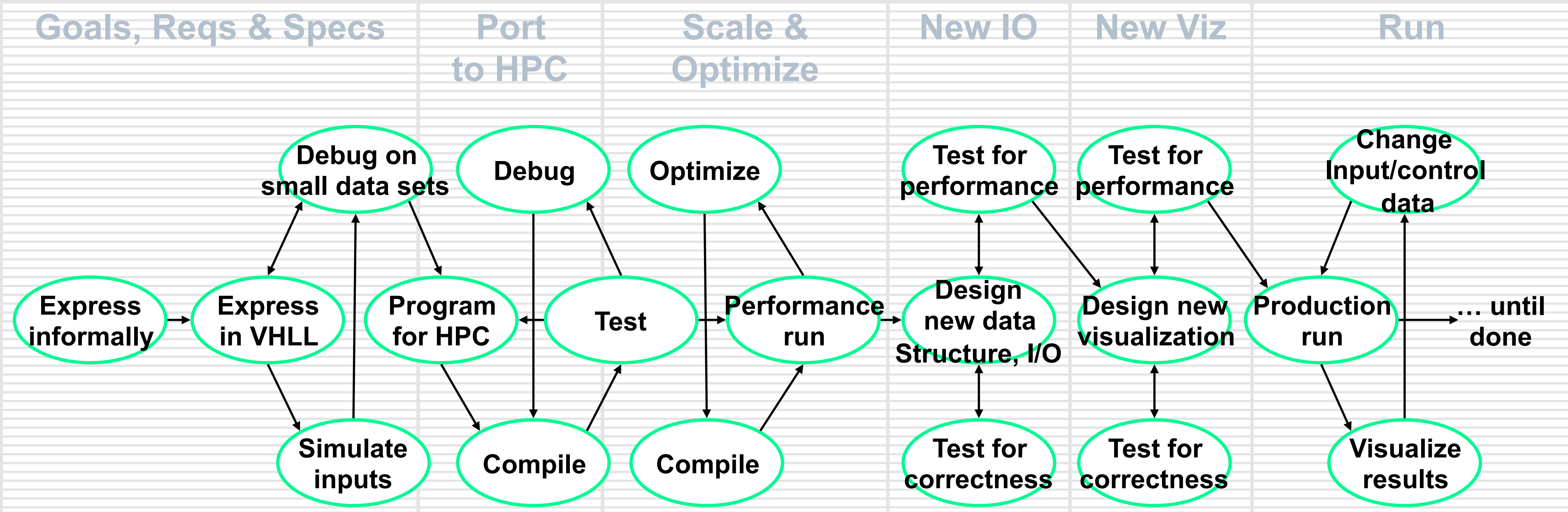
- Top500 High Performance Linpack (HPL)
  - Parallel, portable, block oriented
  - Exploit high-levels of temporal locality (reuse data)
  - Code size — >10,000 lines of FORTRAN + MPI
  - One of the most highly optimized codes for performance!

US Government HPC Workflows



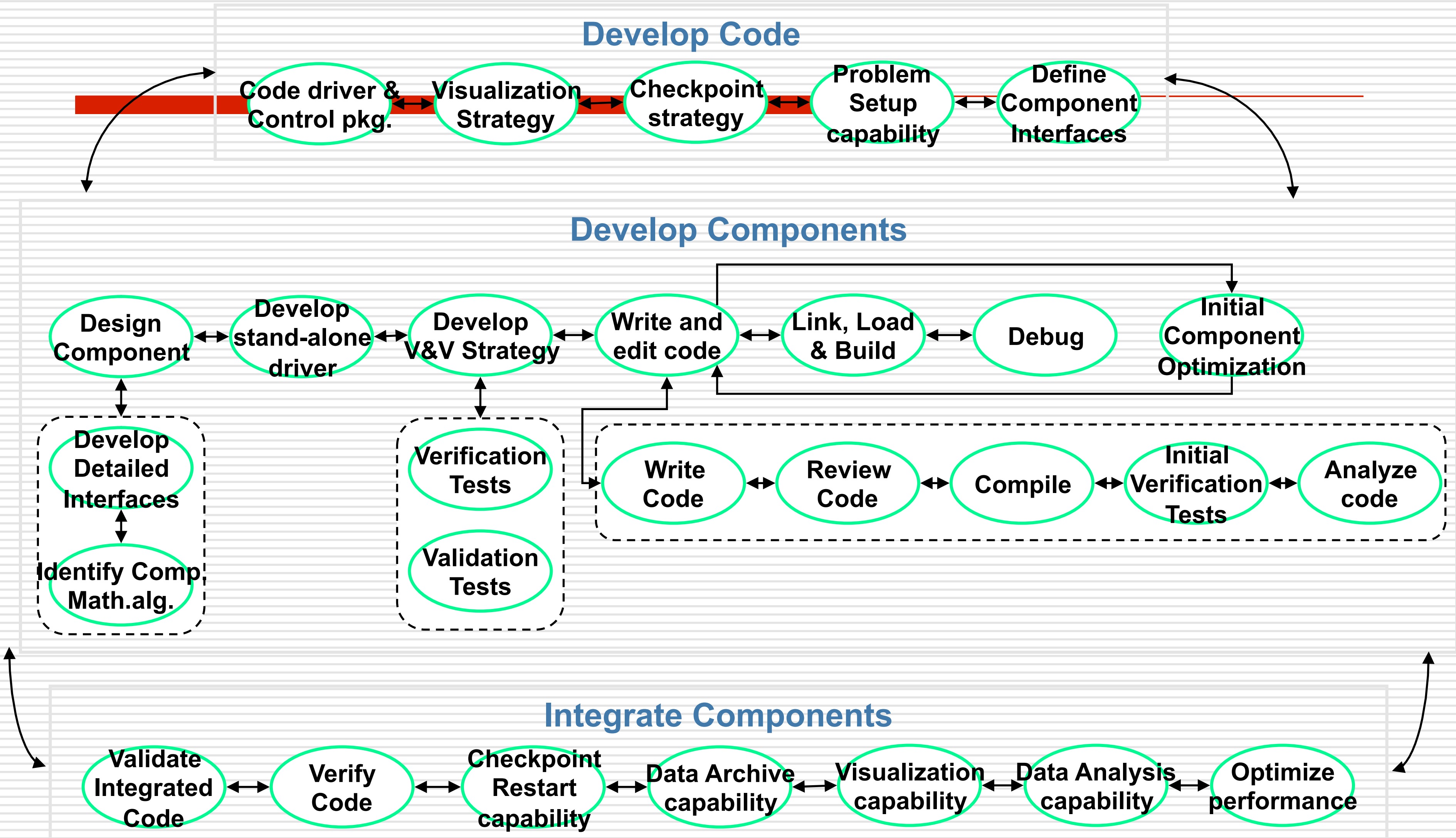
- Many overlapping steps. Item in red - HPC specific interest

Coding Programs Quickly

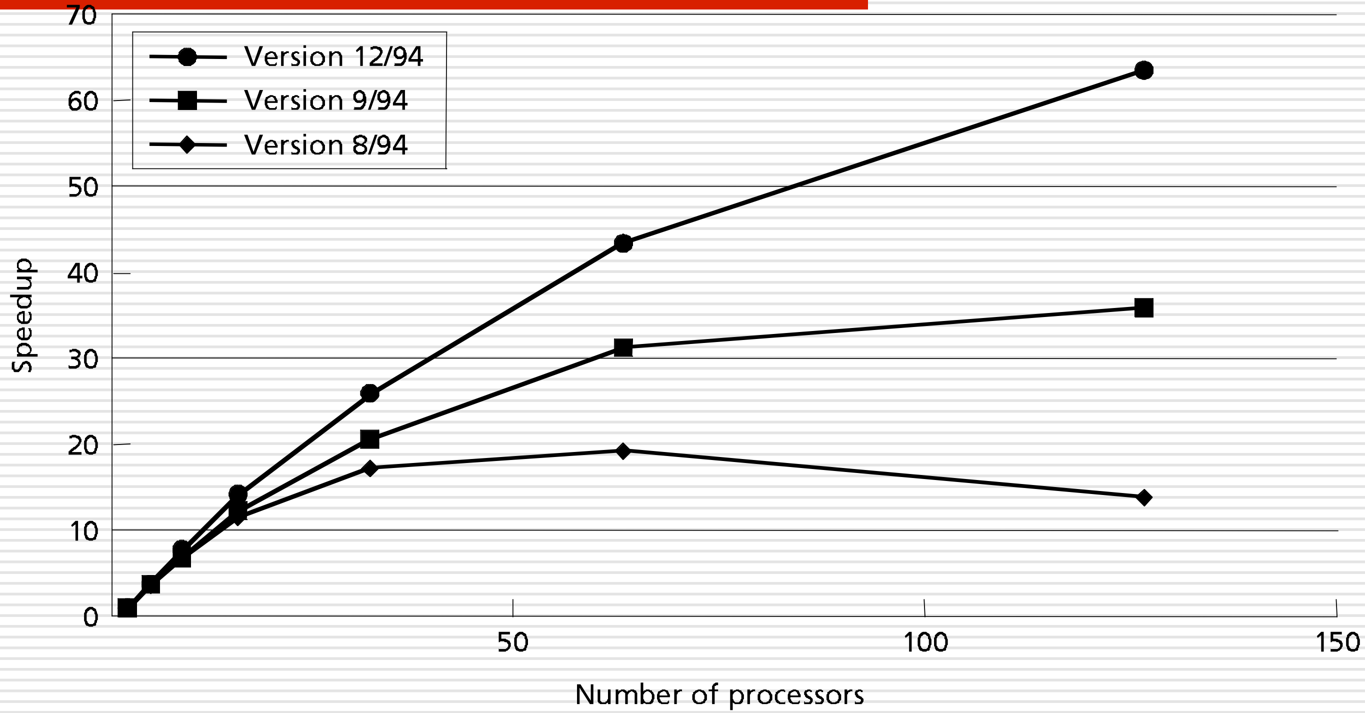


- Allows vendors to specifically identify which steps they are addressing with their technologies

Multi-Module Development



Is better parallel arch enough?

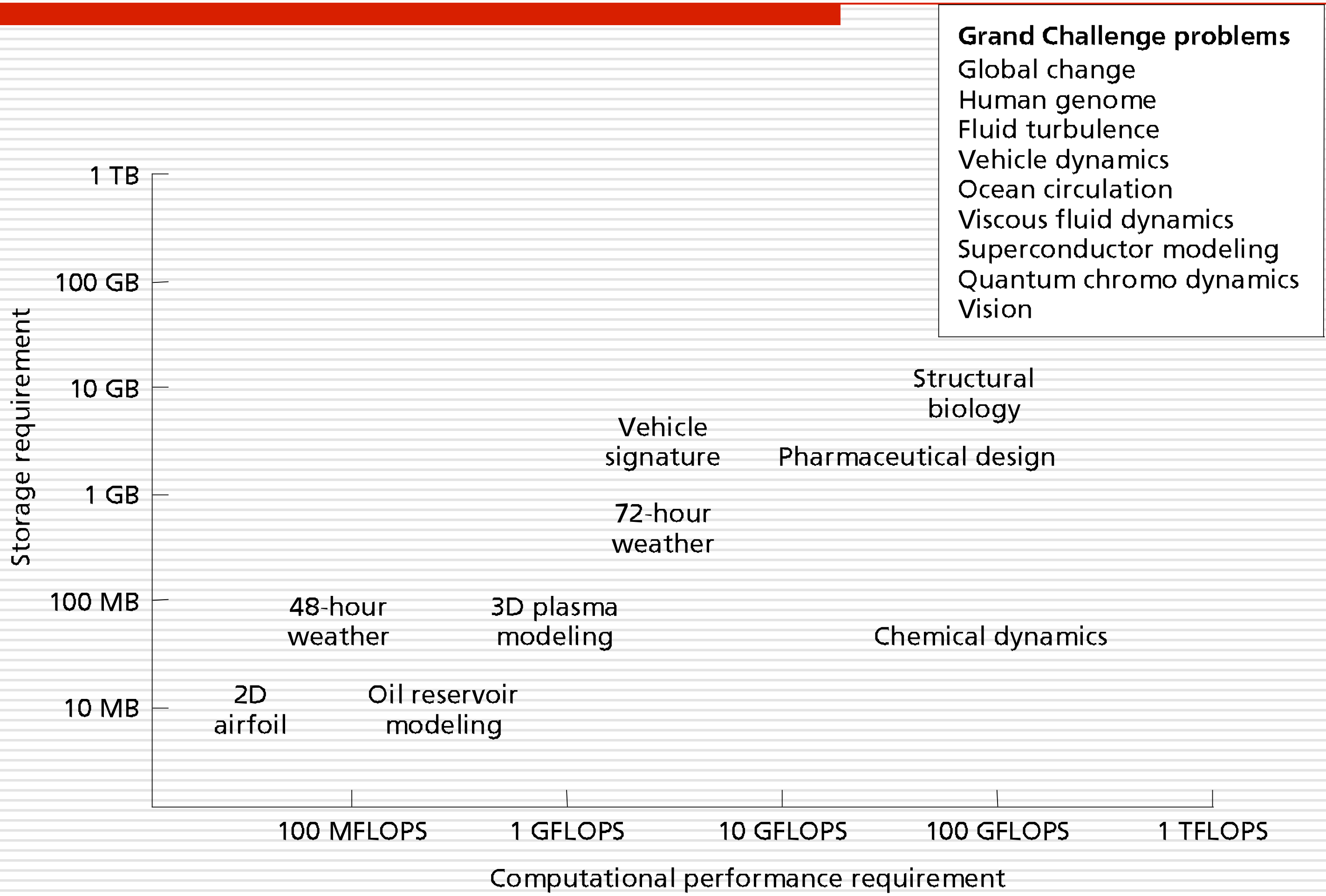


- AMBER molecular dynamics simulation program
- Starting point was vector code for Cray-1
- 145 MFLOP on Cray90, 406 for final version on 128-processor Paragon, 891 on 128-processor Cray T3D

Summary of Application Trends

- Transition to parallel computing has occurred for scientific and engineering computing
- In rapid progress in commercial computing
  - Database and transactions as well as financial
  - Usually smaller-scale, but large-scale systems also used
- Desktop also uses multithreaded programs, which are a lot like parallel programs
- Demand for improving throughput on sequential workloads
  - Greatest use of small-scale multiprocessors
- Solid application demand exists and will increase

Scientific Computing Demand



Main Measure: Speedup

- Speedup (p processors) = $\frac{\textit{Performance (N processors)}}{\textit{Performance (1 processor)}}$
- For a fixed problem size (input data set), Perf = 1/time
- Speedup (N processors) = $\frac{\textit{Time (1 processor)}}{\textit{Time (N processors)}}$
- Issue: comparison to uniprocessor version

Tale of Two Laws

- Amdahl – control-flow parallelism
  - Sequential part -> SP
- $$S = \frac{T1}{TN} = \frac{T1}{SP*T1 + (1-SP)T1/N} = \frac{N}{SP*N + (1-SP)}$$
- Pessimistic – no data parallelism
  - Does not apply to SIMD, SPMD
- Gustafson-Barsis
  - Normalized: $TN=1$
- $S=T1=N-(N-1)*SP$

Finally: Grading Scheme

- 40% homeworks (4)
- 30% midterm exam
- 30% project (teams of 1-2)

Acknowledgments

- D. Koester, MITRE
- NUMAchine group
- Authors of recommended textbooks