CS315A/EE382B

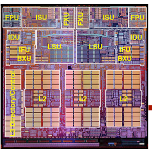
Parallel Computer Architecture and Programming

Kunle Olukotun
Stanford University

http://eeclass.stanford.edu/cs315a

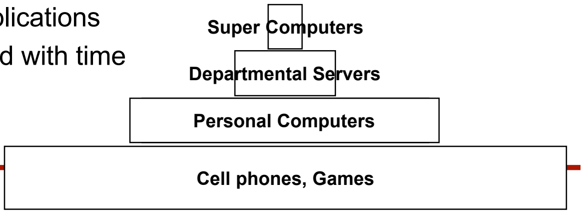
What Will You Get Out of CS315A

- In-depth understanding of the design and programming of modern parallel computers
  - Fundamental parallel architecture issues
    - naming, replication, communication, synchronization
  - Parallel Programming
    - shared memory, thread-level speculation, transactions
  - Parallel applications
    - scientific computing, enterprise, desktop
  - Design tradeoffs
    - Performance, cost, complexity, ease-of-use
  - Parallel systems from small to large scale
    - Single-chip
    - Racks



Will it be worthwhile?

- Absolutely!
  - A few of you will become parallel architecture designers
  - Many will program parallel machines
- Parallelism is becoming widespread
  - "The free lunch is over"
  - "Concurrency revolution"
- The fundamental issues and solutions translate across a wide spectrum of systems.
- Ideas pioneered at the high-end of the platform pyramid on the most-demanding applications
  - migrate downward with time



Super Computers

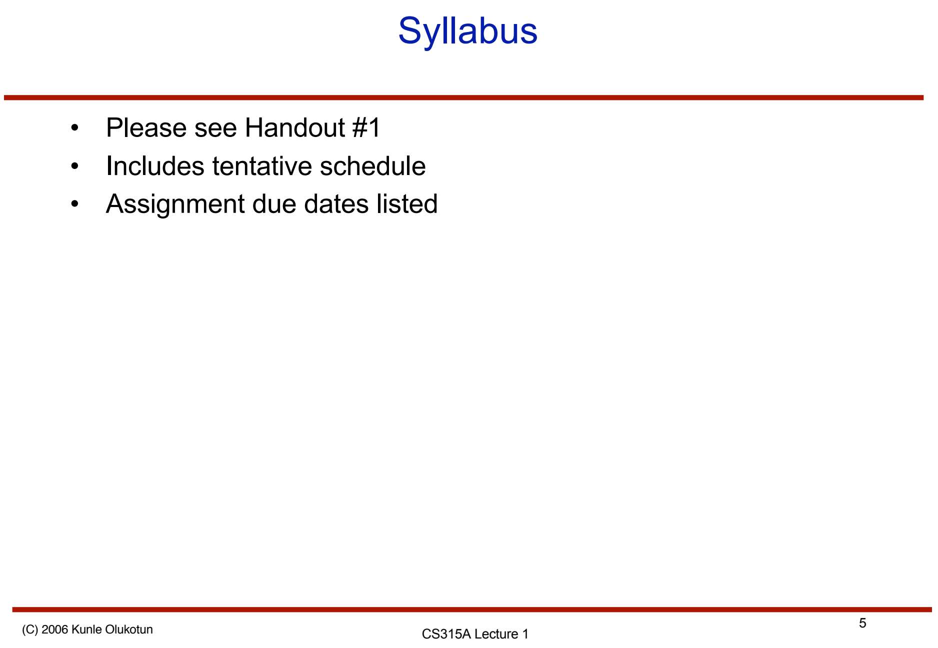
Departmental Servers

Personal Computers

Cell phones, Games

Major Topics

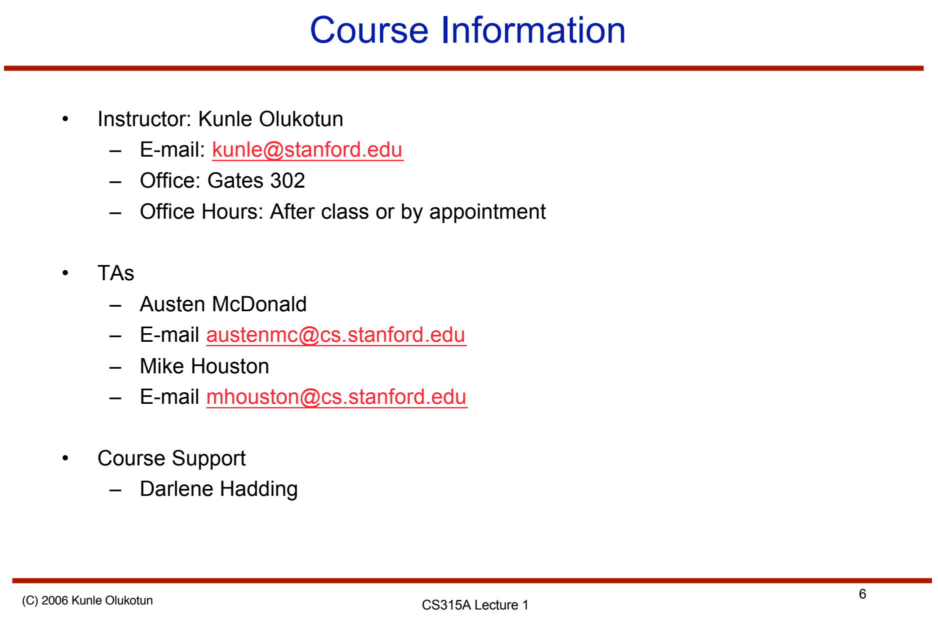
- Emphasize both programming and architecture
- Parallel Programming
  - Parallel analysis and algorithm design
  - Application characteristics
  - Programming with shared memory
  - Programming with speculative threads and transactions
- Parallel Architectures
  - Symmetric shared memory
  - Synchronization and consistency
  - Chip multiprocessors
  - Thread-level speculation
  - Transactional memory
  - Distributed shared memory



Syllabus

- Please see Handout #1
- Includes tentative schedule
- Assignment due dates listed

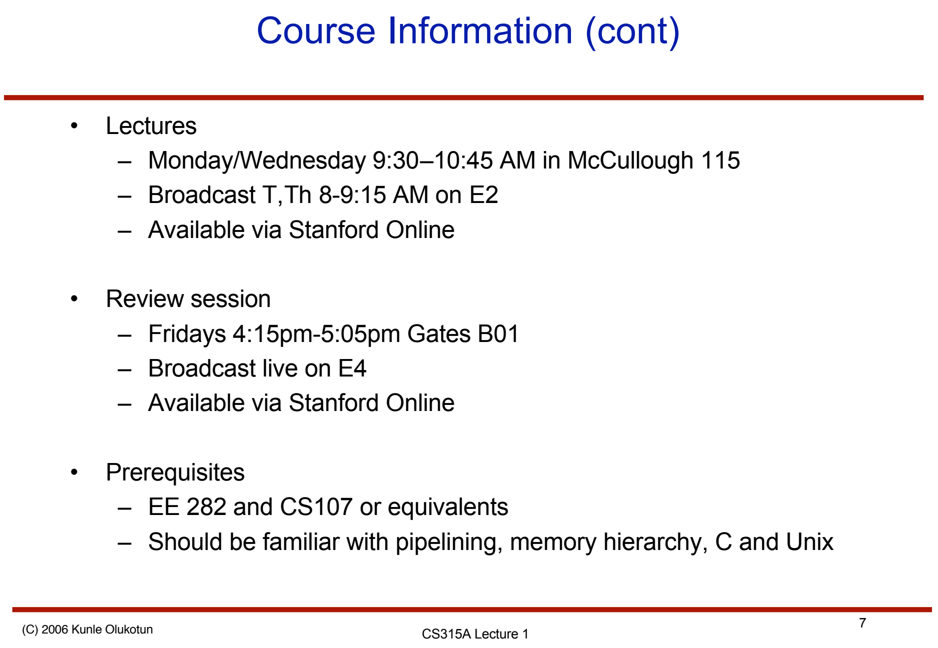
(C) 2006 Kunle Olukotun CS315A Lecture 1 5



Course Information

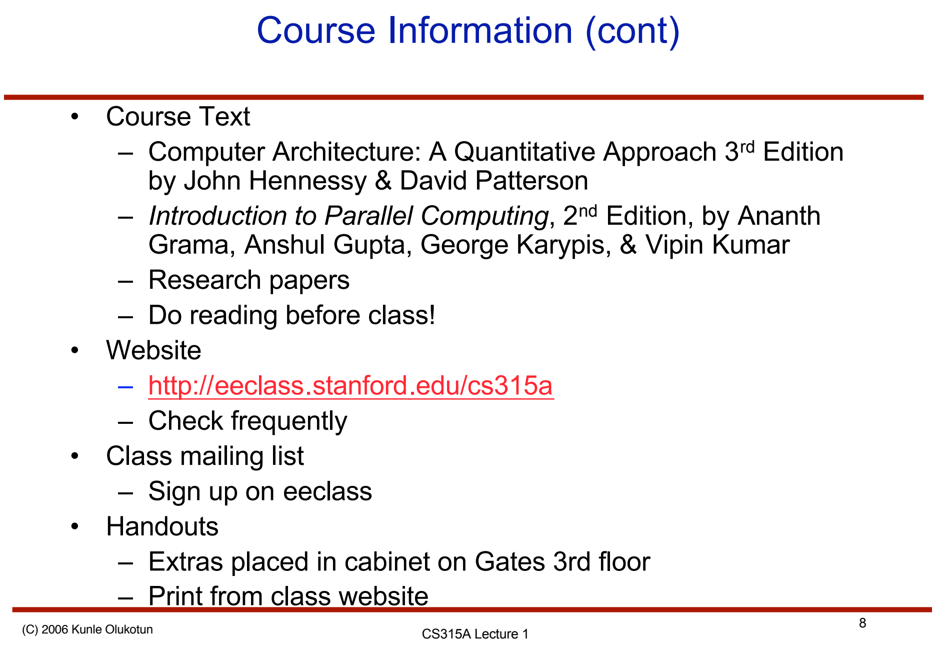
- Instructor: Kunle Olukotun
  - E-mail: kunle@stanford.edu
  - Office: Gates 302
  - Office Hours: After class or by appointment
- TAs
  - Austen McDonald
  - E-mail austenmc@cs.stanford.edu
  - Mike Houston
  - E-mail mhouston@cs.stanford.edu
- Course Support
  - Darlene Hadding

(C) 2006 Kunle Olukotun CS315A Lecture 1 6



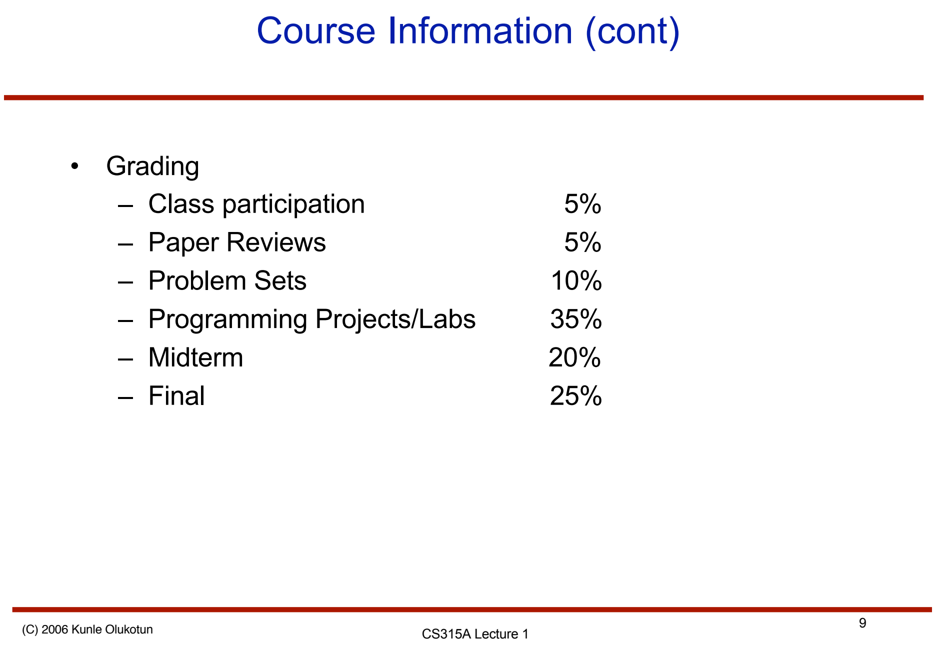
Course Information (cont)

- Lectures
  - Monday/Wednesday 9:30–10:45 AM in McCullough 115
  - Broadcast T,Th 8-9:15 AM on E2
  - Available via Stanford Online
- Review session
  - Fridays 4:15pm-5:05pm Gates B01
  - Broadcast live on E4
  - Available via Stanford Online
- Prerequisites
  - EE 282 and CS107 or equivalents
  - Should be familiar with pipelining, memory hierarchy, C and Unix



Course Information (cont)

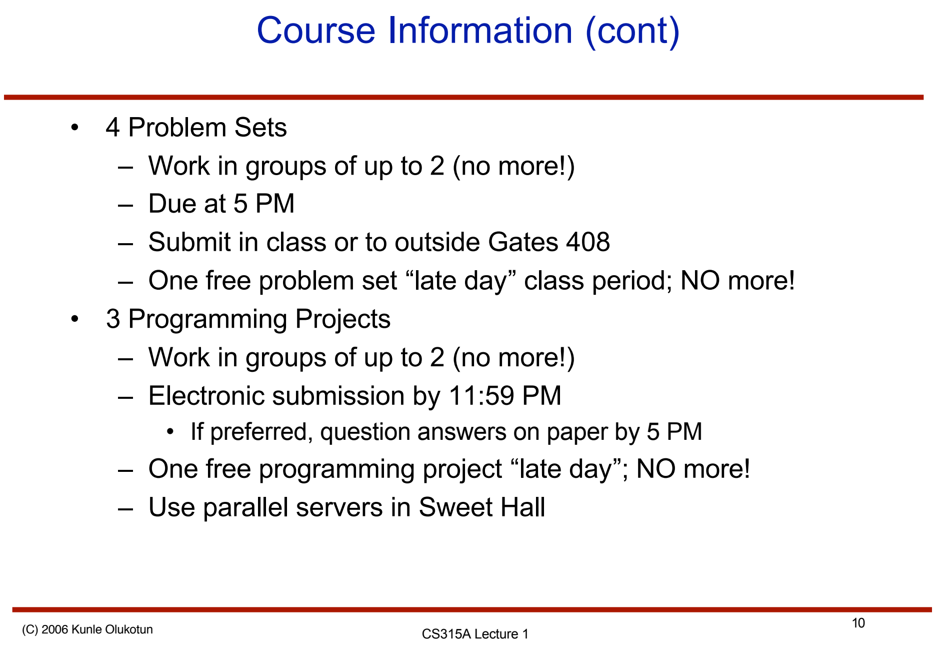
- Course Text
  - Computer Architecture: A Quantitative Approach 3rd Edition by John Hennessy & David Patterson
  - *Introduction to Parallel Computing*, 2nd Edition, by Ananth Grama, Anshul Gupta, George Karypis, & Vipin Kumar
  - Research papers
  - Do reading before class!
- Website
  - http://eeclass.stanford.edu/cs315a
  - Check frequently
- Class mailing list
  - Sign up on eeclass
- Handouts
  - Extras placed in cabinet on Gates 3rd floor
  - Print from class website



Course Information (cont)

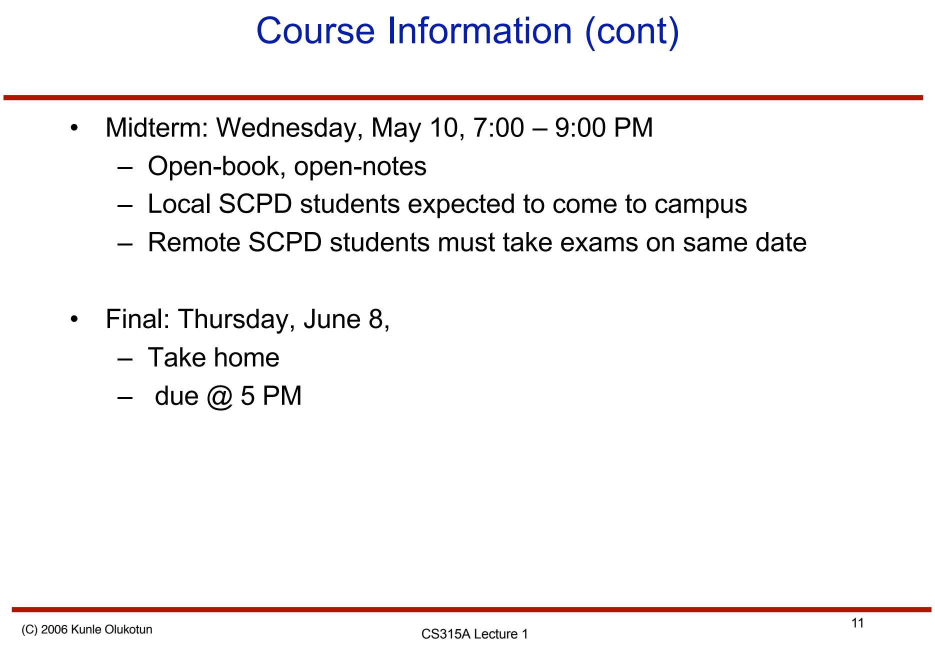
- Grading

– Class participation	5%
– Paper Reviews	5%
– Problem Sets	10%
– Programming Projects/Labs	35%
– Midterm	20%
– Final	25%



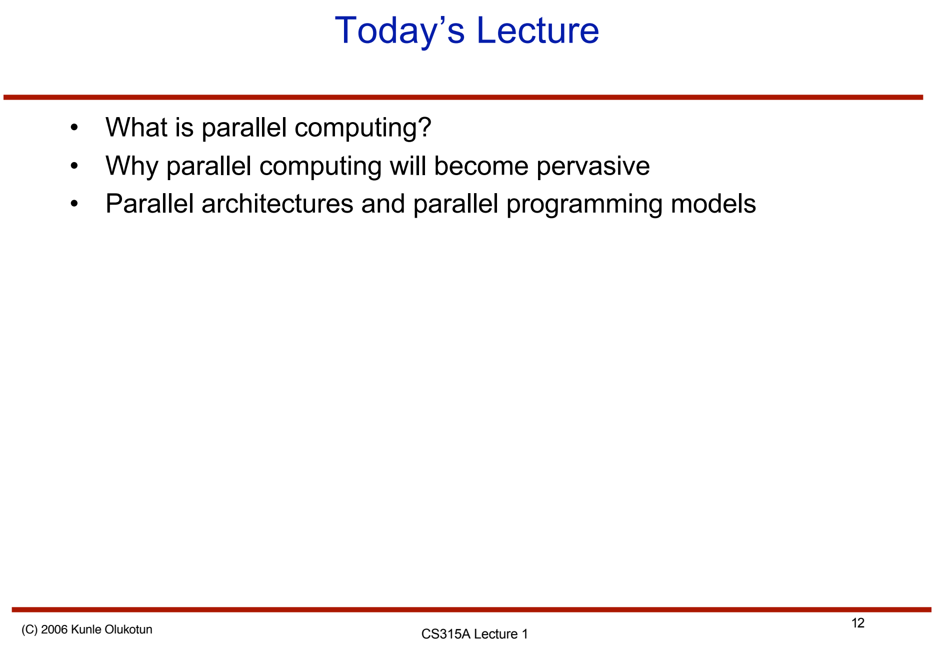
Course Information (cont)

- 4 Problem Sets
  - Work in groups of up to 2 (no more!)
  - Due at 5 PM
  - Submit in class or to outside Gates 408
  - One free problem set “late day” class period; NO more!
- 3 Programming Projects
  - Work in groups of up to 2 (no more!)
  - Electronic submission by 11:59 PM
    - If preferred, question answers on paper by 5 PM
  - One free programming project “late day”; NO more!
  - Use parallel servers in Sweet Hall



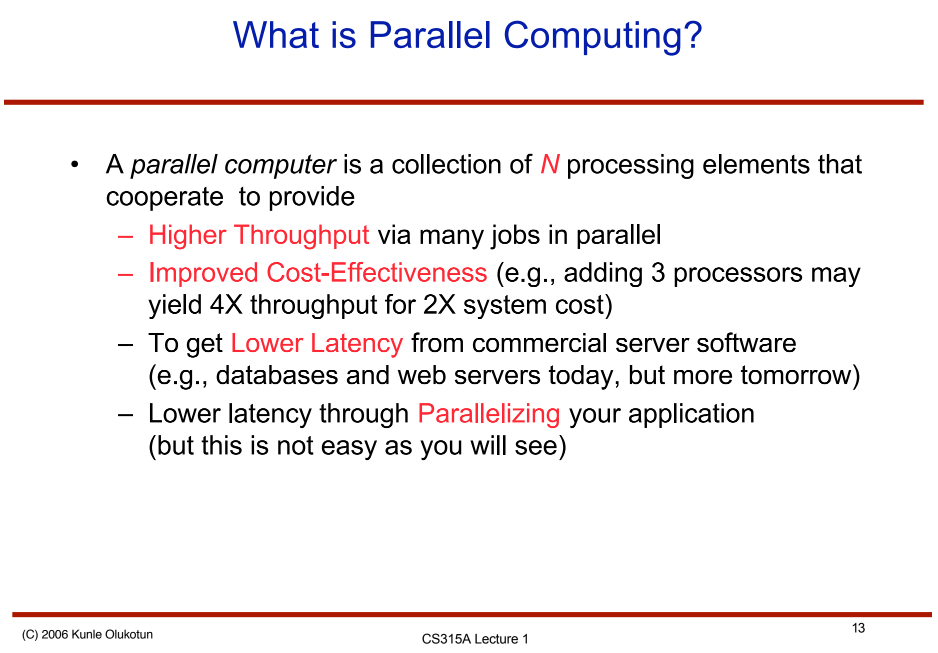
Course Information (cont)

- Midterm: Wednesday, May 10, 7:00 – 9:00 PM
  - Open-book, open-notes
  - Local SCPD students expected to come to campus
  - Remote SCPD students must take exams on same date
- Final: Thursday, June 8,
  - Take home
  - due @ 5 PM



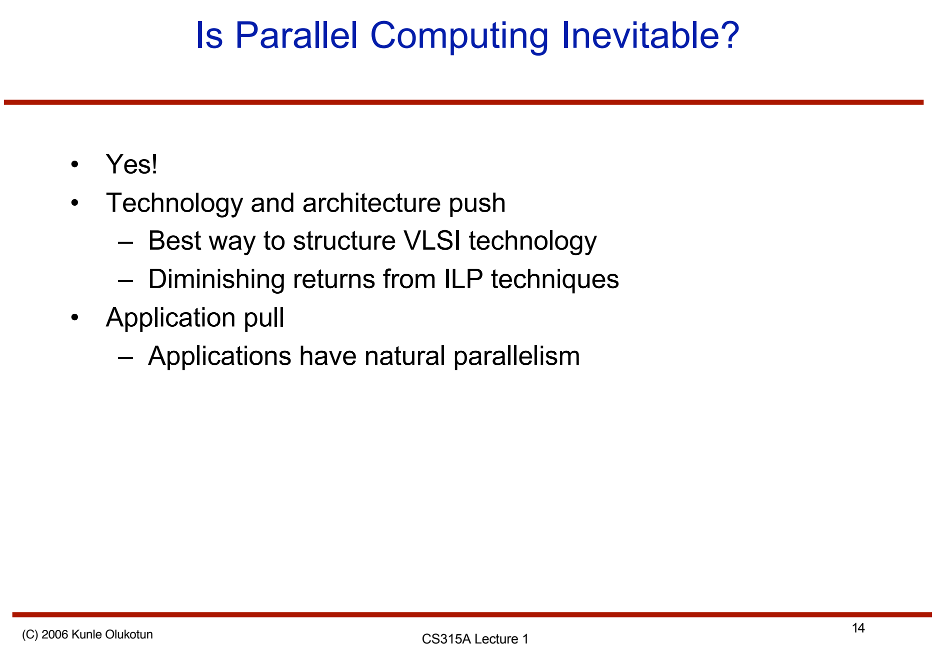
Today’s Lecture

- What is parallel computing?
- Why parallel computing will become pervasive
- Parallel architectures and parallel programming models



What is Parallel Computing?

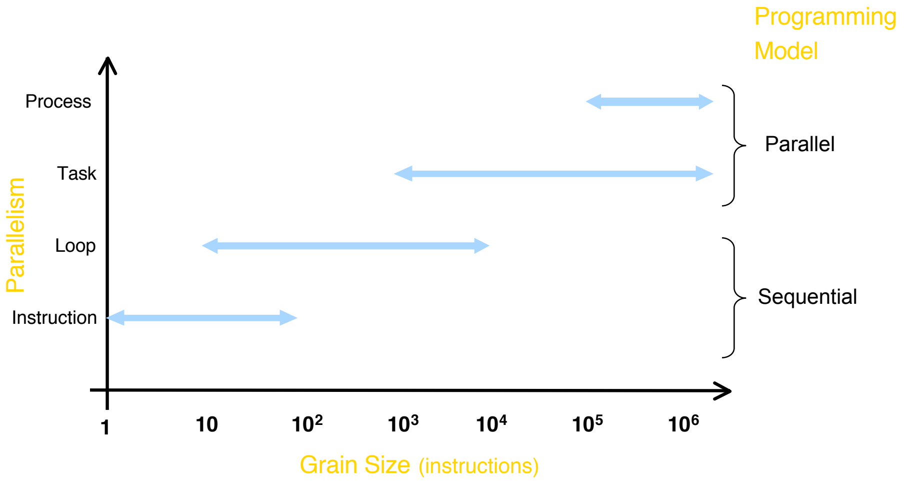
- A *parallel computer* is a collection of *N* processing elements that cooperate to provide
  - Higher Throughput via many jobs in parallel
  - Improved Cost-Effectiveness (e.g., adding 3 processors may yield 4X throughput for 2X system cost)
  - To get Lower Latency from commercial server software (e.g., databases and web servers today, but more tomorrow)
  - Lower latency through Parallelizing your application (but this is not easy as you will see)



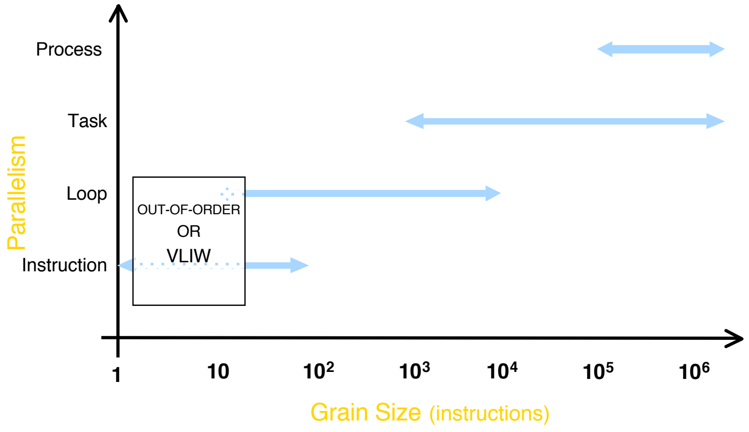
Is Parallel Computing Inevitable?

- Yes!
- Technology and architecture push
  - Best way to structure VLSI technology
  - Diminishing returns from ILP techniques
- Application pull
  - Applications have natural parallelism

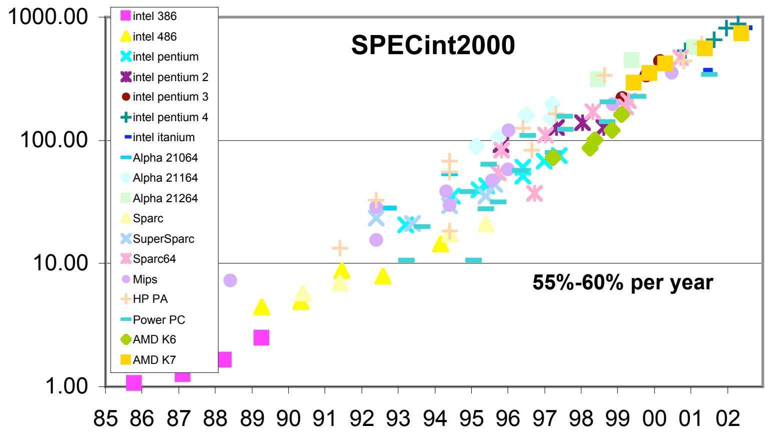
Where is Application Parallelism?



Exploiting Instruction Level Parallelism



Processor Performance Improvement

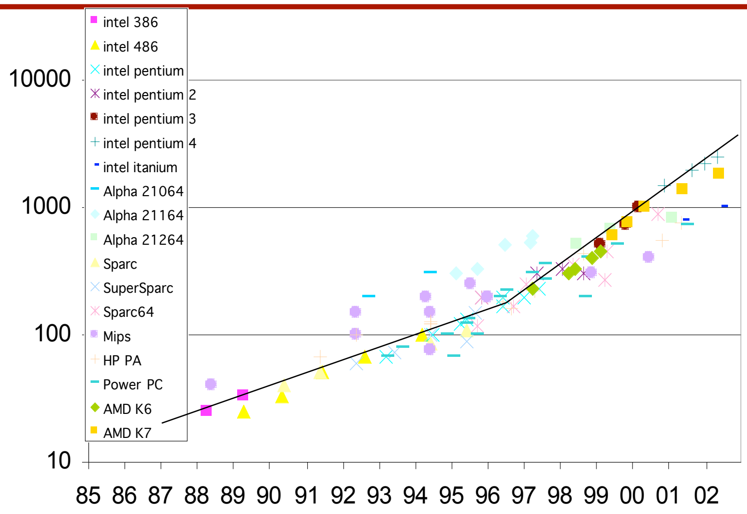


(C) 2006 Kunle Olukotun

CS315A Lecture 1

17

Clock Frequency Scaling

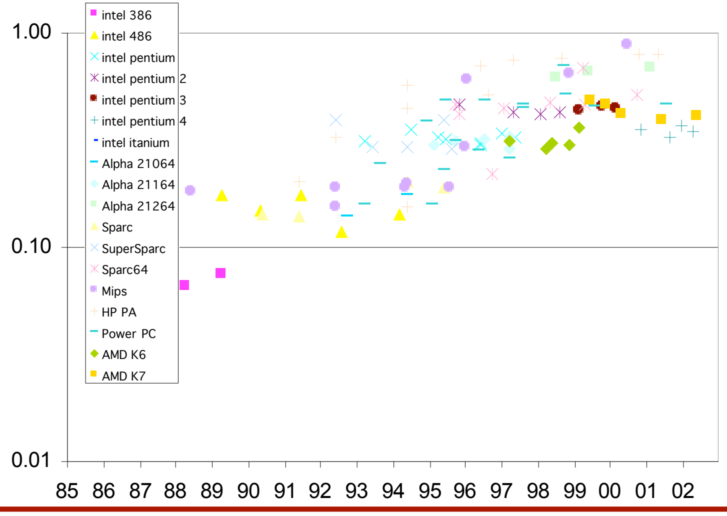


(C) 2006 Kunle Olukotun

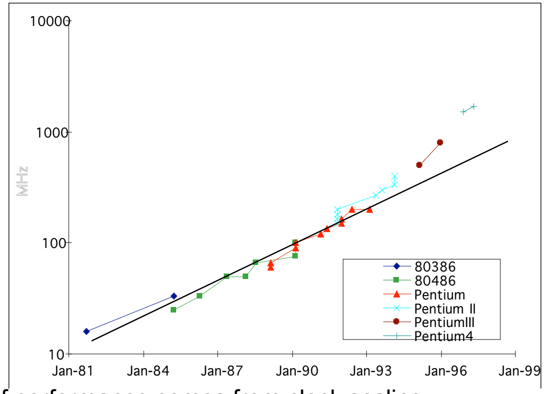
CS315A Lecture 1

18

SPECint/MHz



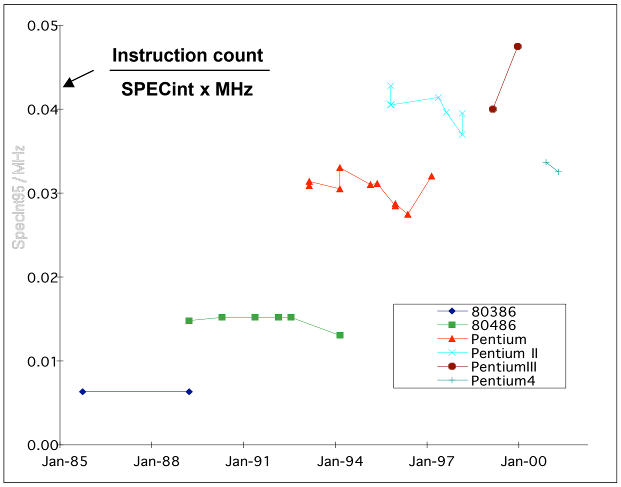
Clock Frequency Improvements Dominate



- Most of performance comes from clock scaling
  - Clock frequency double each generation
- Two factors contribute: technology (1.4x/gen), circuit design (deeper pipelining)

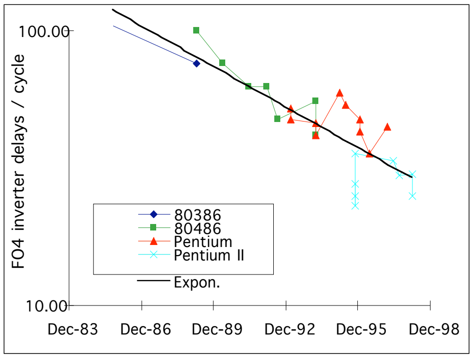
Architecture Scaling from ILP

- Plot of IPC
  - Compiler + IPC
  - 1.5x / generation
  - Until PIII, now falling
- There is a lot of hardware to make this happen
  - Many transistors
  - Lots of power
  - Lots of designers



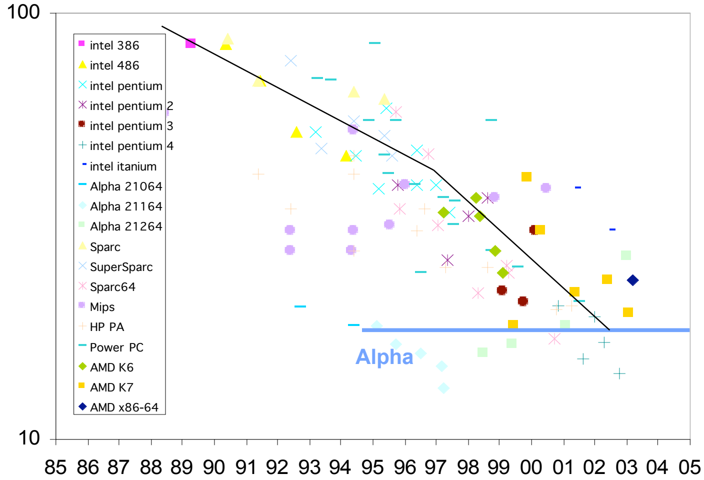
Gates Per Clock

- Clock speed has been scaling faster than base technology
- Cycles measured in FO4 delays has been falling by 1.4x/generation



- Caused by:
  - Better adder/mem arch
  - Faster circuit families
  - Better optimization
  - Better micro-architecture
- All this generally requires more transistors
- Cannot continue this trend

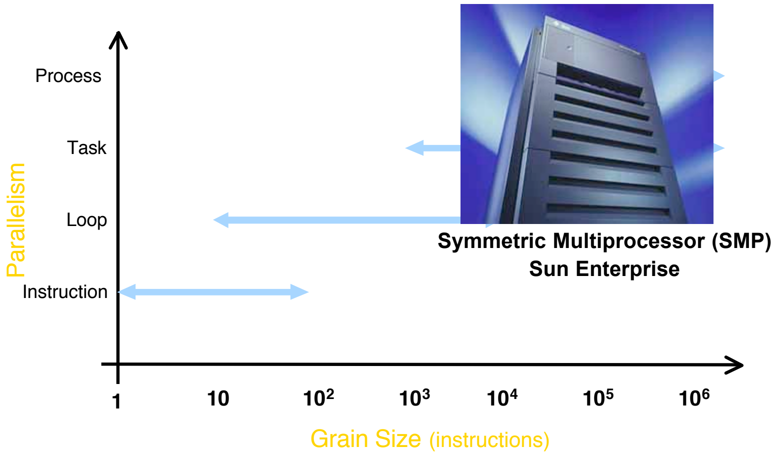
Clock Cycle in ‘FO4’



Single Processor Performance Reached Limits

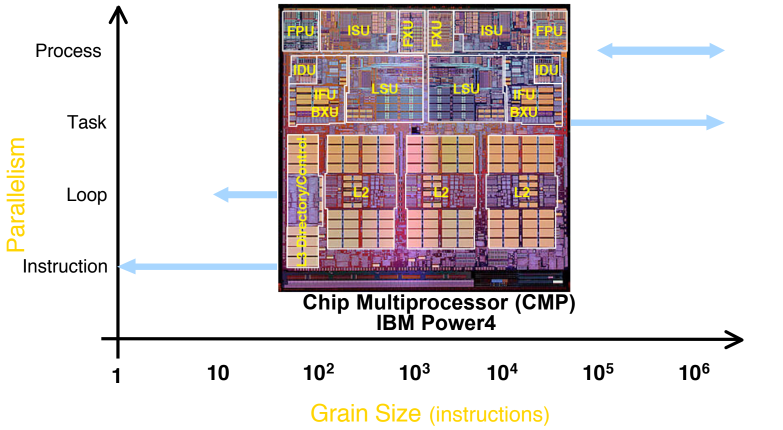
- I predicted this in the mid 90’s and it has finally happened
- ILP and deep pipelining have run out of steam:
  - ILP parallelism in applications has been mined out
  - Communication delays are hurting complex microarchitectures
  - Frequency scaling is now driven by technology
  - The power and complexity of microarchitectures taxes our ability to cool and verify
- Latest example
  - Pentium 4: 24 stages, 3+ GHz
  - Prescott (aka Pentium 5) : 35 stages, 4+ GHz
  - Comparable performance
- How do we continue to improve performance?

Exploit Coarse-Grain Thread-Level Parallelism



Symmetric Multiprocessor (SMP)
Sun Enterprise

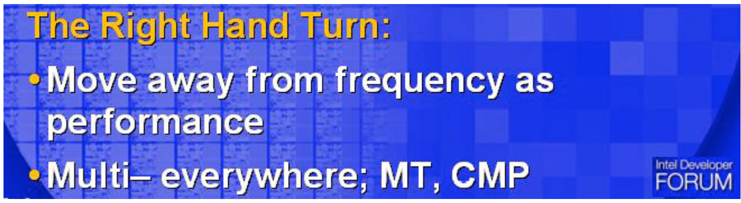
Exploit Fine and Coarse Grain Thread-Level Parallelism



Chip Multiprocessor (CMP)
IBM Power4

The End of the Word As We Know It

- Single thread performance plateau and …
- Process Technology Stops Improving
  - Moore's law but …
  - Transistors don't get faster (65nm vs. 45nm)
  - Wires are much worse



The Right Hand Turn:

- Move away from frequency as performance
- Multi– everywhere; MT, CMP

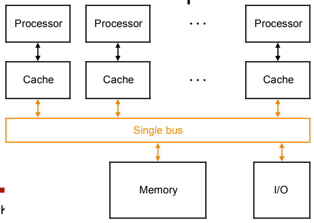
Intel Developer
FORUM

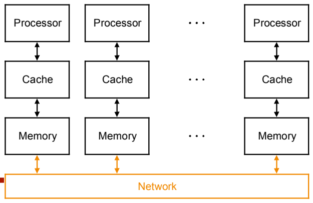
From Intel Developer Forum, September 2004

Now need parallel applications to take full advantage of microprocessors!

Key Multiprocessor Questions

- How do parallel processors share data?
  - single address space: Symmetric MP (SMP) vs. NonUniform Memory Architecture (NUMA)
  - message passing: clusters, massively parallel processors (MPP)
- How do parallel processors coordinate?
  - synchronization (locks, semaphores)
  - built into send / receive primitives
  - OS primitives (sockets)
- How are the processors interconnected?



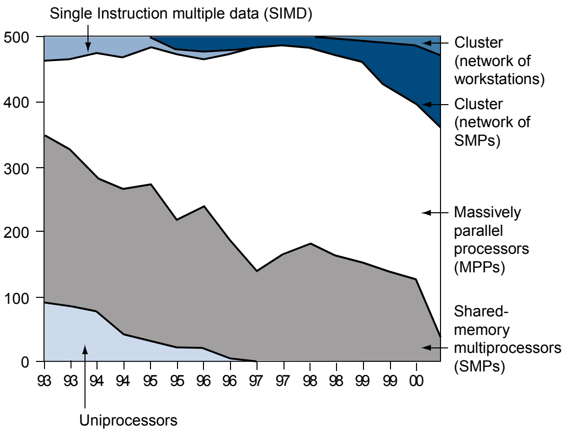


Parallel Applications: Science and Engineering

- Examples
  - Weather prediction
  - Evolution of galaxies
  - Oil reservoir simulation
  - Automobile crash tests
  - Biology
  - VLSI CAD
  - Physical modeling in computer games!
- Typically model physical systems or phenomena
- Problems are 2D or 3D
- Usually requires “number crunching”
- Involves “true” parallelism

Supercomputers

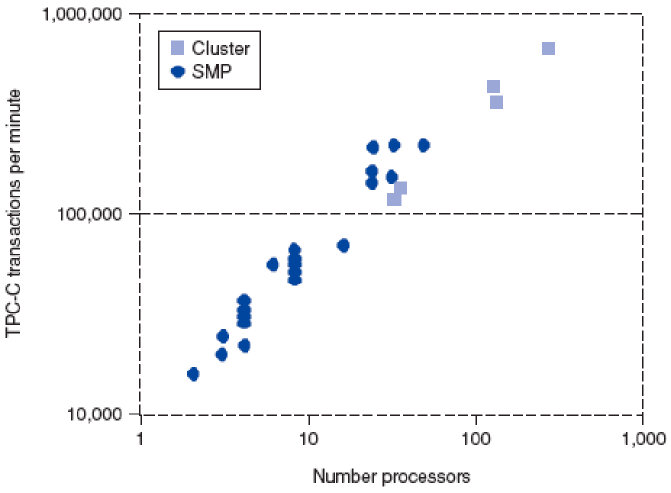
Plot of top 500 supercomputer sites over a decade:



Parallel Applications: Commercial

- Examples
  - On-line transaction processing (OLTP), TPC-C
  - Decision support systems (DSS), TPC-H
  - Application servers, SPEC JBB
  - Web servers, SPEC web99
- Involves data movement, not much number crunching
  - OTLP has many small queries
  - DSS has fewer large queries
- Involves throughput parallelism
  - inter-query parallelism for OLTP
  - intra-query parallelism for DSS

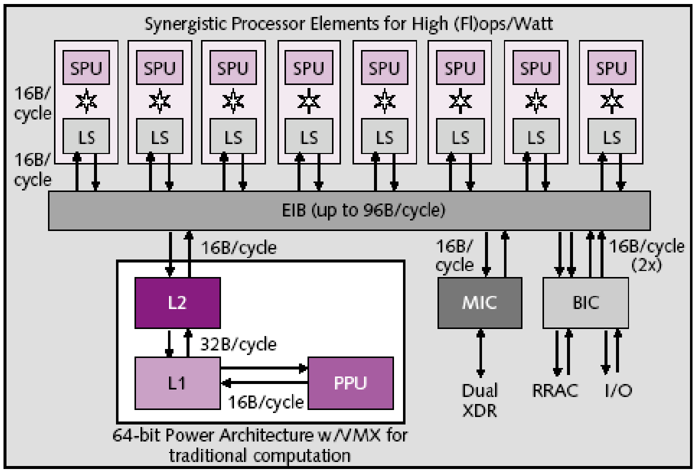
TPC-C Performance

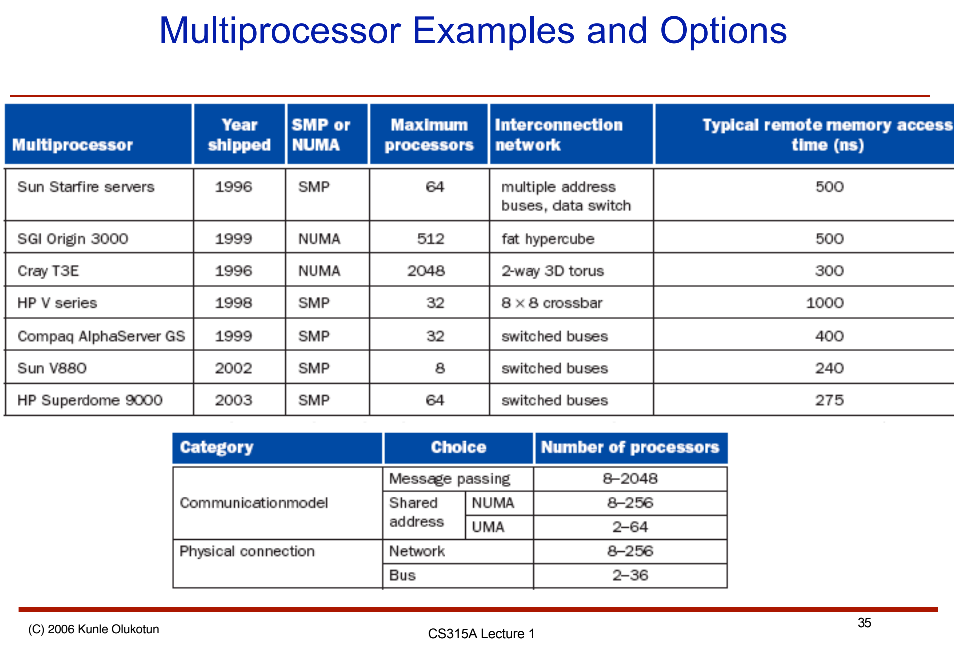


Parallel Applications: Multi-media/home

- Examples
  - speech recognition
  - data compression/decompression
  - video/graphics processing
  - 3D graphics
  - games
- Will become ubiquitous
- Involves everything (crunching, data movement, true parallelism, and throughput parallelism)

IBM Cell Processor

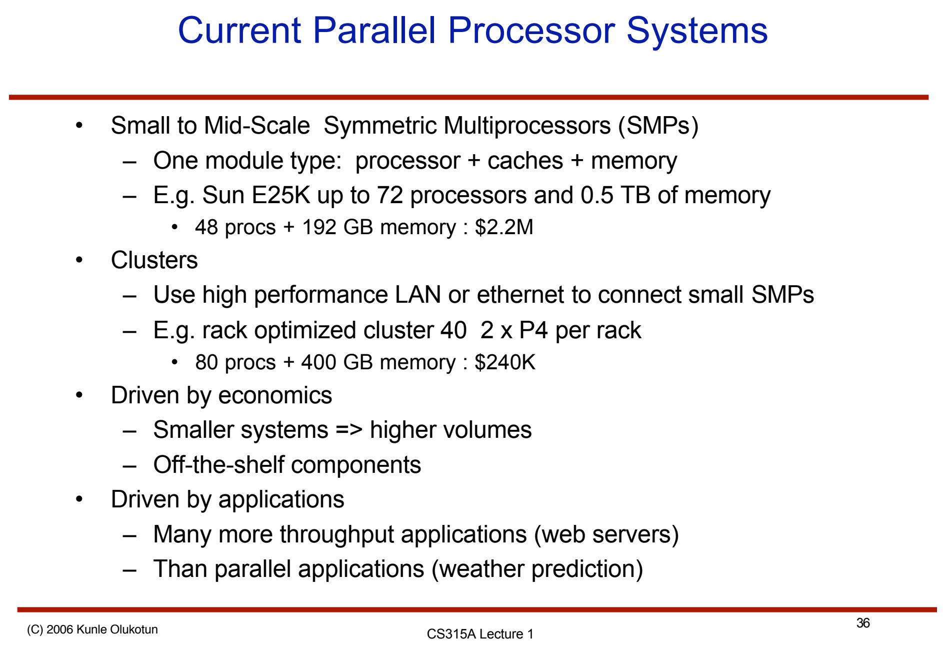




Multiprocessor Examples and Options

Multiprocessor	Year shipped	SMP or NUMA	Maximum processors	Interconnection network	Typical remote memory access time (ns)
Sun Starfire servers	1996	SMP	64	multiple address buses, data switch	500
SGI Origin 3000	1999	NUMA	512	fat hypercube	500
Cray T3E	1996	NUMA	2048	2-way 3D torus	300
HP V series	1998	SMP	32	8 × 8 crossbar	1000
Compaq AlphaServer GS	1999	SMP	32	switched buses	400
Sun V880	2002	SMP	8	switched buses	240
HP Superdome 9000	2003	SMP	64	switched buses	275

Category	Choice		Number of processors
Communicationmodel	Message passing		8–2048
	Shared address	NUMA	8–256
		UMA	2–64
Physical connection	Network		8–256
	Bus		2–36



Current Parallel Processor Systems

- Small to Mid-Scale Symmetric Multiprocessors (SMPs)
  - One module type: processor + caches + memory
  - E.g. Sun E25K up to 72 processors and 0.5 TB of memory
    - 48 procs + 192 GB memory : $2.2M
- Clusters
  - Use high performance LAN or ethernet to connect small SMPs
  - E.g. rack optimized cluster 40 2 x P4 per rack
    - 80 procs + 400 GB memory : $240K
- Driven by economics
  - Smaller systems => higher volumes
  - Off-the-shelf components
- Driven by applications
  - Many more throughput applications (web servers)
  - Than parallel applications (weather prediction)