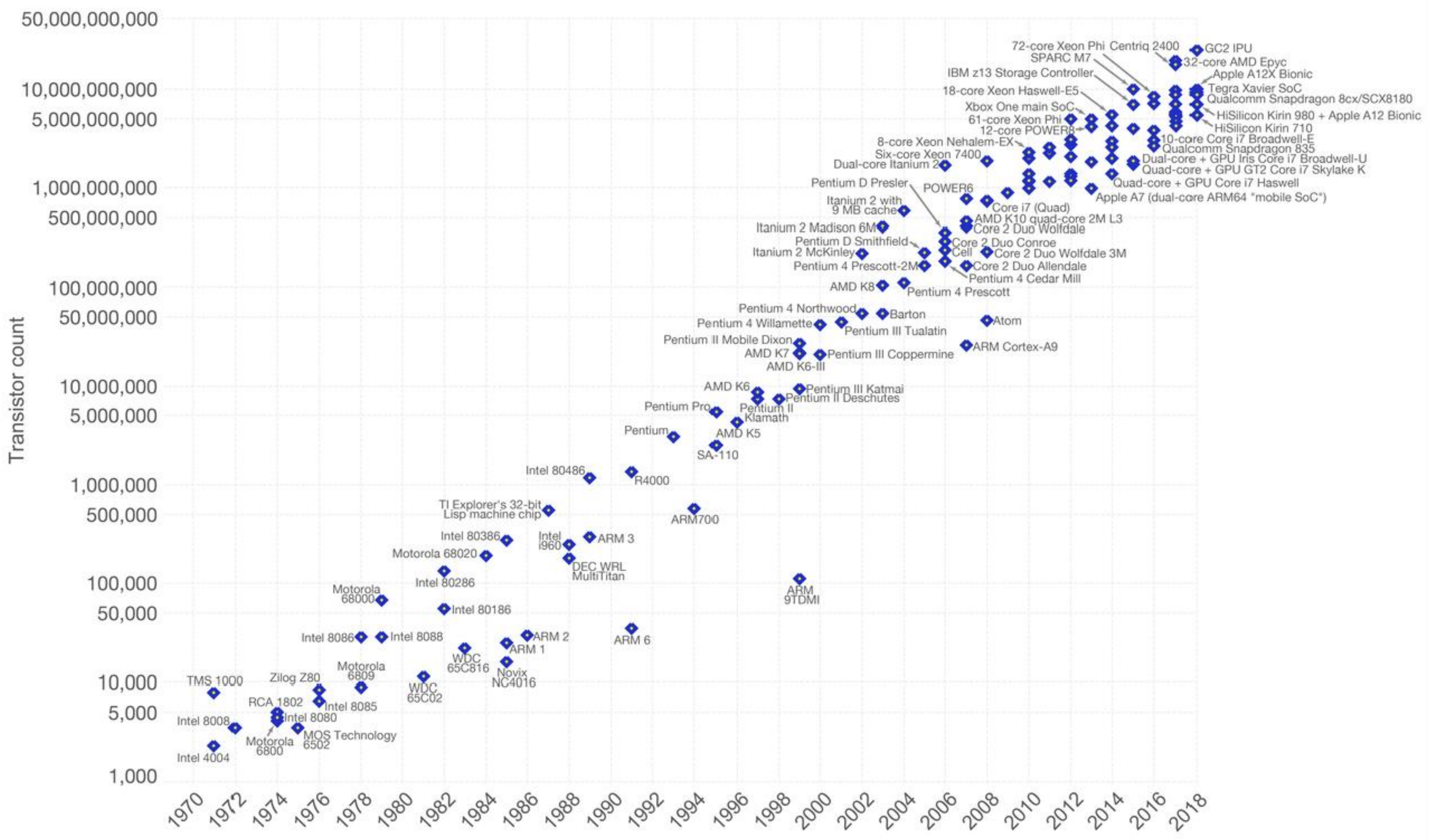
# EE 542
# Lecture 8: Single Processor to Cloud

Internet and Cloud Computing

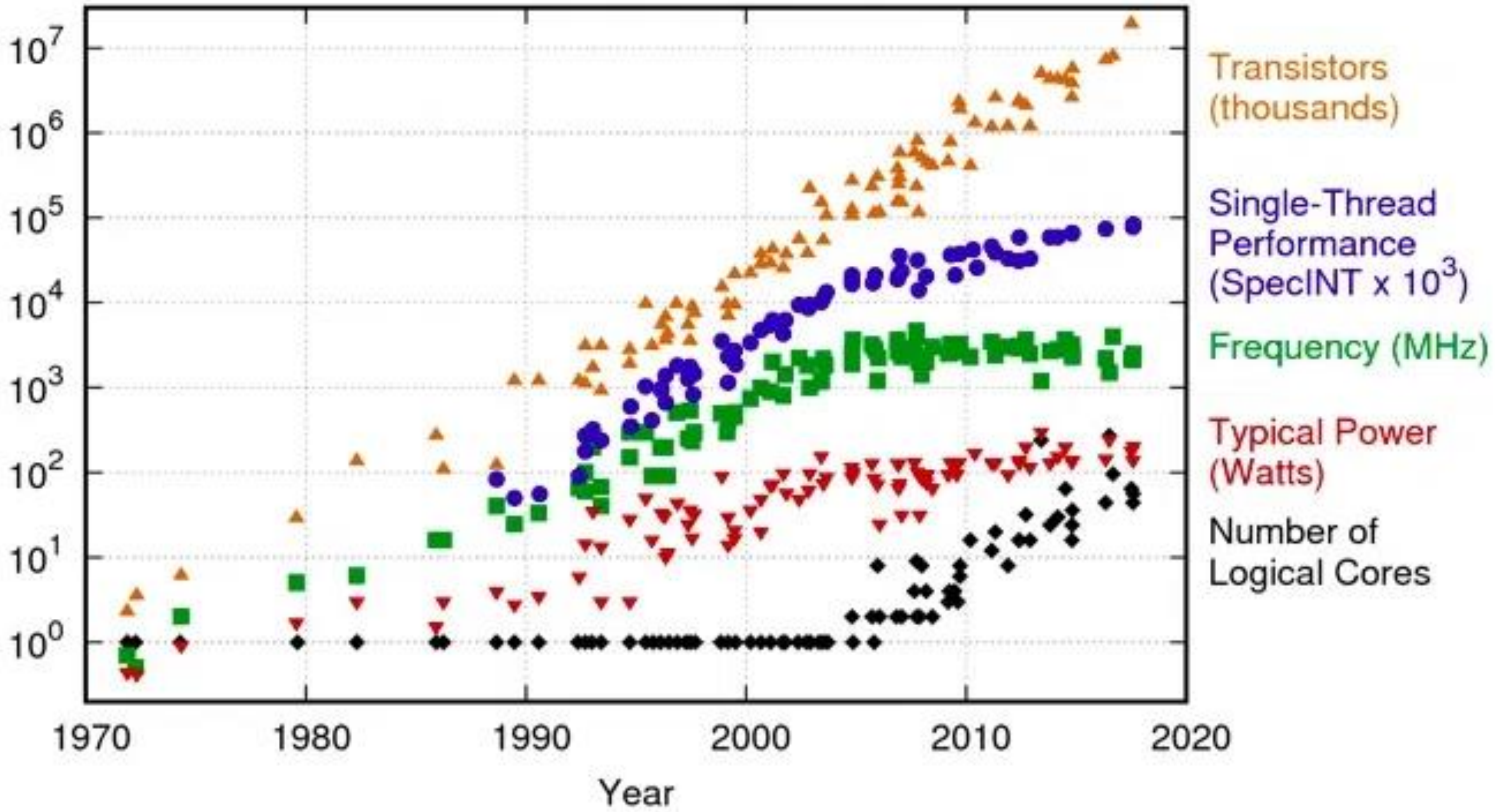
Young Cho
Department of Electrical Engineering
University of Southern California

# Moore's Law



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

# CPU Scaling



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

# Parallelism

- Bit Level
  - Parallel Circuit Design
- Instruction Level
  - Microarchitecture
  - Example: Pipelining and Superscalar
- Thread/Task Level
  - Independent Processes
  - Multiple cores, register files, and shared memory
- Application Level
  - Independent Programs
  - Multiple cores and Multiple chips

## Instruction-Level Parallelism (ILP)

- Parallelism at the machine-instruction level
- The processor can re-order, pipeline instructions, split them into microinstructions, do aggressive branch prediction, etc.
- Instruction-level parallelism enabled rapid increases in processor speeds

# Thread-Level Parallelism (TLP)

- This is parallelism on a more coarser scale
- Server can serve each client in a separate thread (Web server, database server)
- A computer game can do AI, graphics, and physics in three separate threads
- Single-core superscalar processors cannot fully exploit TLP
- Multi-core architectures are the next step in processor evolution: explicitly exploiting TLP

# Application Level Parallelism

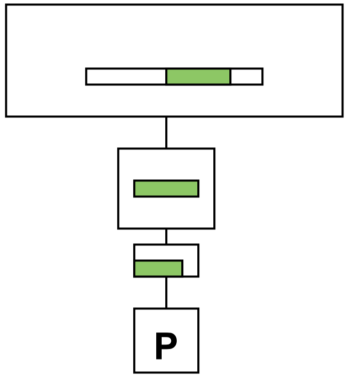
- Multicore Processor
  - All processors are on the same chip
  - Multi-core processors are MIMD
    - Multiple Instructions Multiple Data
  - Extensive Shared Memory Management
- Clustered Computers over Network
  - Scales based on the number of parallel programs
  - Potential for tighter integration - difficult to achieve

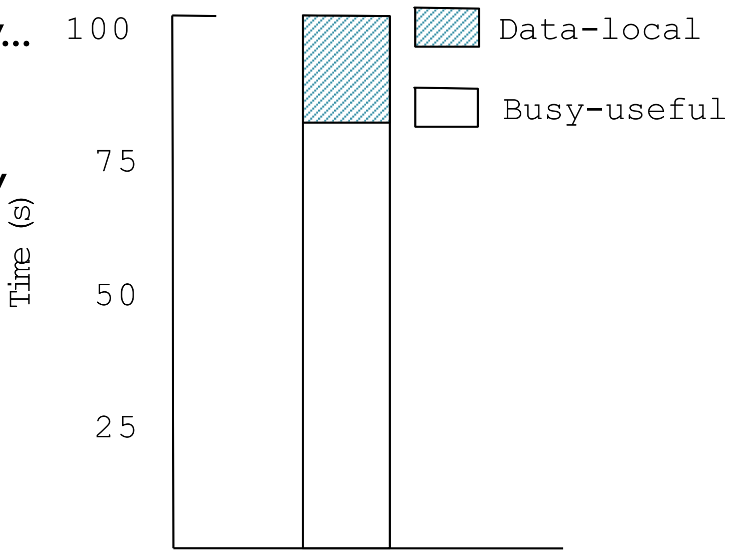
## The Ultimate Goal

- Inifinite Scalability
  - More HW => Higher Performance
- Difficulty in Parallelization
  - Need to parallelize a single thread into independent threads
  - Limits dependent on the algorithm
  - Or is there a limit??

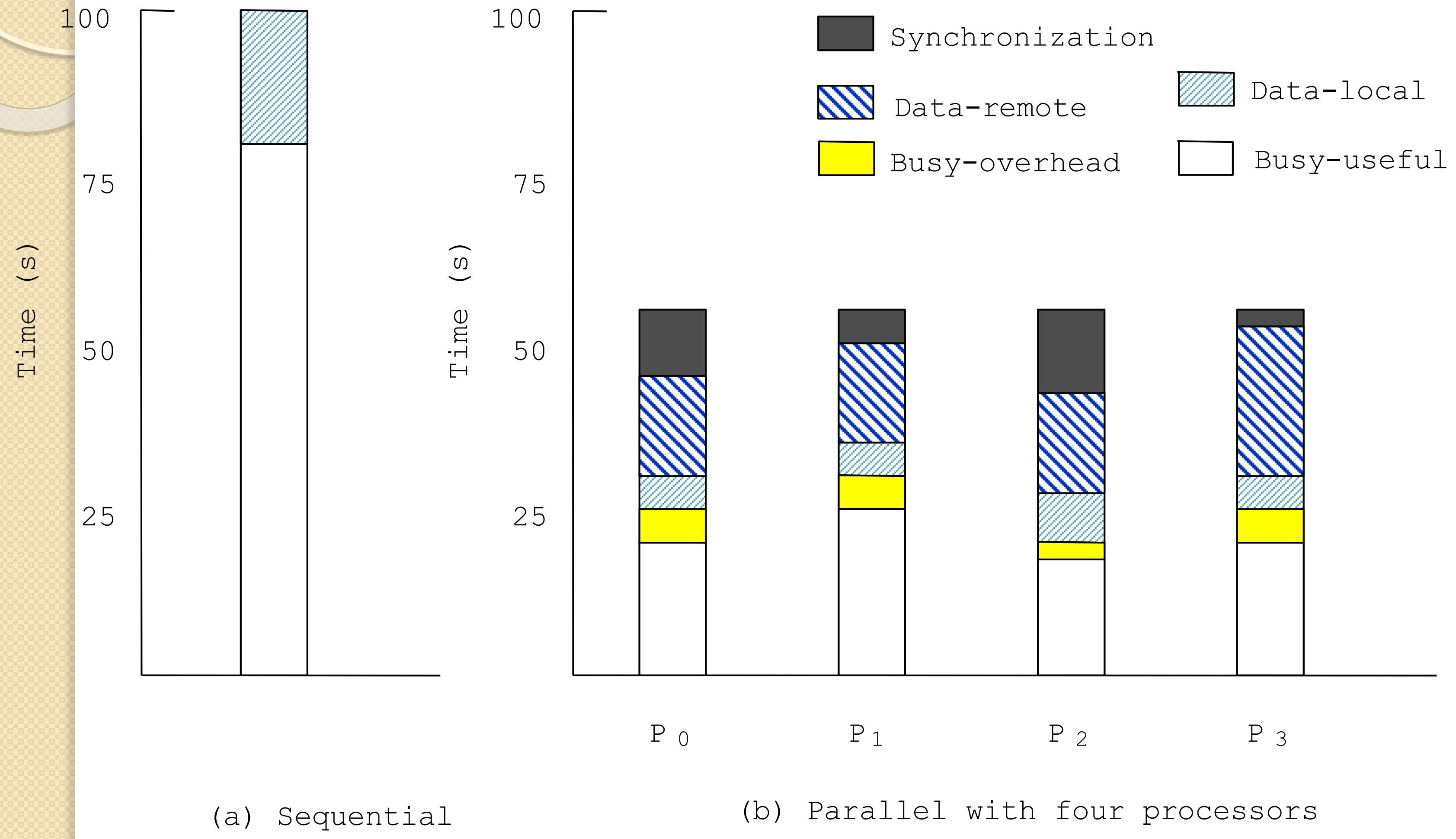
# Uniprocessor View

- Performance depends heavily on memory hierarchy
- Time spent by a program
  - Timeprog(I) = Busy(I) + Data Access(I)
  - Divide by cycles to get CPI equation
- Data access time can be reduced by:
  - Optimizing machine
    - bigger caches, lower latency...
  - Optimizing program
    - temporal and spatial locality





# Uniprocessor vs. Multiprocessor

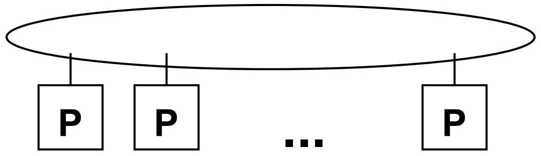


(a) Sequential

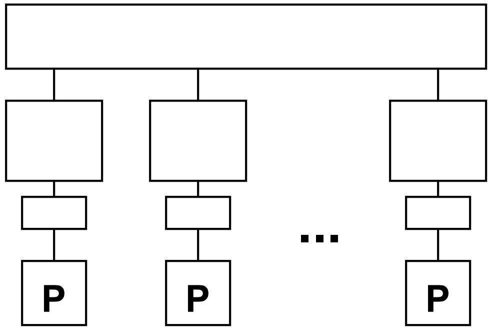
(b) Parallel with four processors

# Multiprocessor

- A collection of communicating processors
  - Goals: balance load, reduce inherent communication and extra work



- A multi-cache, multi-memory system
  - Role of these components essential regardless of  programming model
  - Prog. model  and comm. abstr. affect specific performance tradeoffs



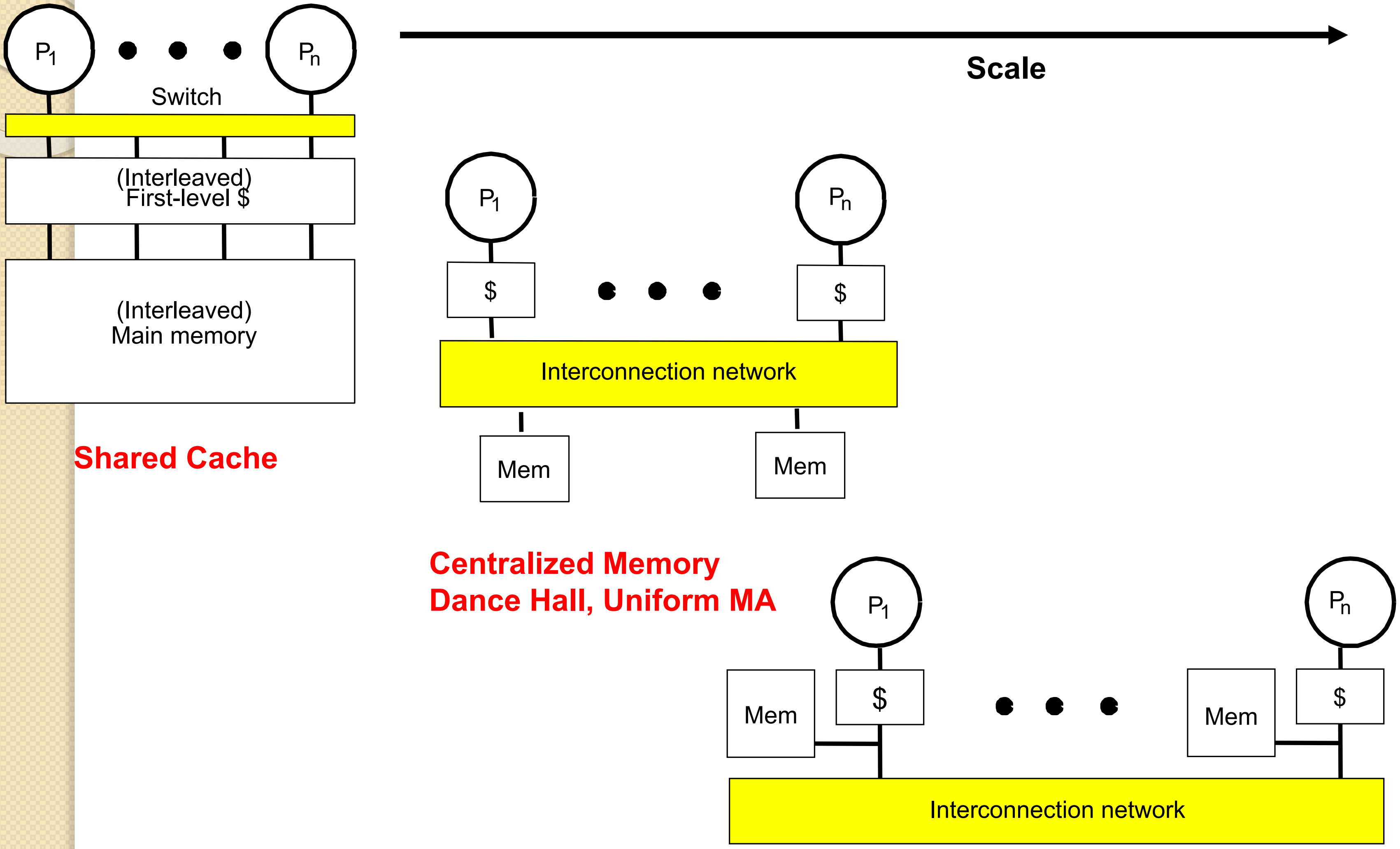
# Parallel Architecture

- Parallel Architecture
  - Computer Architecture
  - Communication Architecture
- Small-scale shared memory
  - Extend the memory system to support multiple processors
  - Good for multiprogramming throughput and parallel computing
  - Allows *fine-grain sharing* of resources
- Characterization
  - Naming
  - Synchronization
  - Performance

## Characterization

- Naming
  - Global physical address space
  - Global virtual address space
  - Segmented shared address space
- Synchronization
  - Local and Shared Memory
  - BUS and Network Latencies
- Performance of Parallel Algorithms
  - Total Latency
  - Total Bandwidth
  - Often not the processor

# Memory Configurations



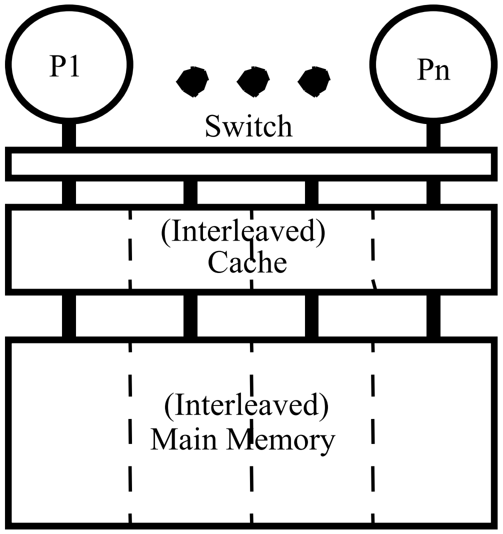
Shared Cache

Centralized Memory
Dance Hall, Uniform MA

Distributed Memory (Non-Uniform Mem Access)

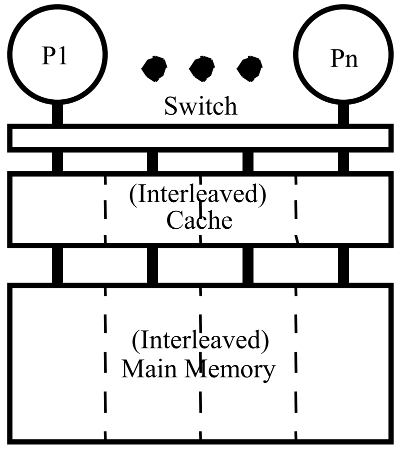
# Shared Cache

- Alliant FX-8
  - early 80's
  - eight 68020s with x-bar to 512 KB interleaved cache
- Encore & Sequent
  - first 32-bit micros (N32032)
  - two to a board with a shared cache



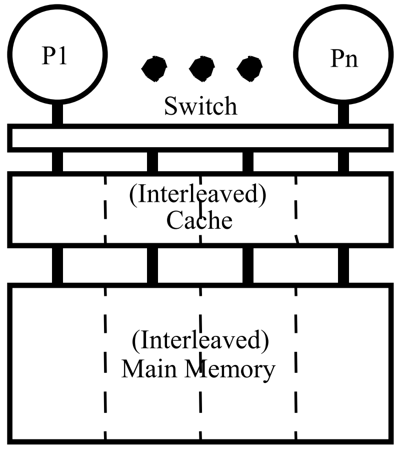
## Advantages

- Single Cache
  - Only one copy of any cached block
- Fine-Grain Sharing
  - Cray Xmp has shared registers!
- Potential for Positive Interference
  - One proc pre-fetches data for another
- Smaller Total Storage
  - Only one copy of code/data
- Can share data within a line
  - Long lines without false sharing

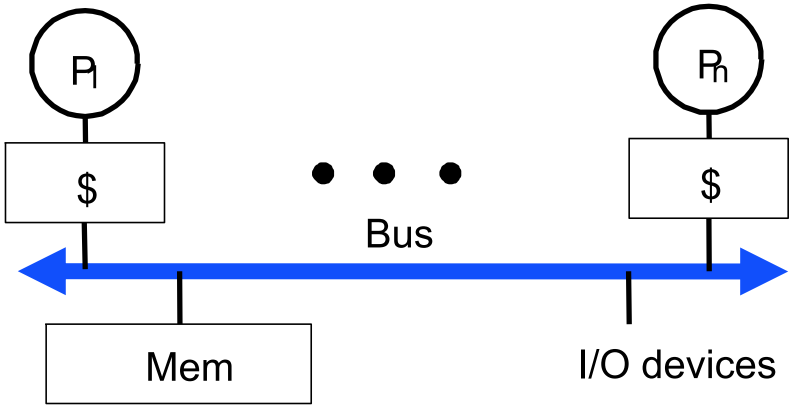


# Disadvantages

- Fundamental BW limitation
- Increases latency of all accesses
  - Cross-bar (Multiple ports)
  - Larger cache
  - L1 hit time determines proc. Cycle
- Potential for negative interference
  - one proc flushes data needed by another
- Many L2 caches are shared today
- CMP makes cache sharing attractive



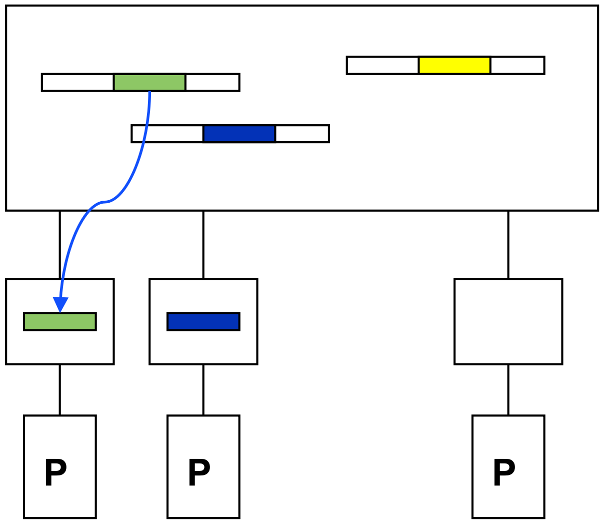
# Bus-Based Symmetric Shared Memory



- Dominate the server market
  - Building blocks for larger systems; arriving to desktop
- Attractive as throughput servers and for parallel programs
  - Fine-grain resource sharing
  - Uniform access via loads/stores
  - Automatic data movement and coherent replication in caches
  - Cheap and powerful extension
- Normal uniprocessor mechanisms to access data
  - Key is extension of memory hierarchy to support multiple processors
- Chip Multiprocessors

# Caches are Critical for Performance

- Reduce average latency
  - automatic replication closer to processor
- Reduce average bandwidth
- Data is logically transferred from producer to consumer to memory
  - store reg --> mem
  - load  reg <-- mem
- Many processors can shared data efficiently
- What happens when store & load are executed  on different processors?



# Cache Coherence

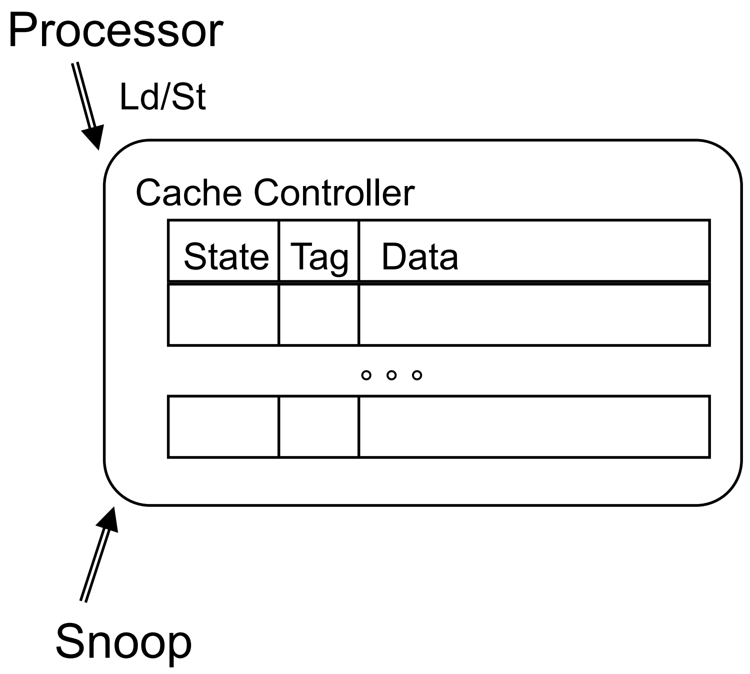
- Caches play key role in all cases
  - Reduce average data access time
  - Reduce bandwidth demands placed on shared interconnect
- Private processor caches create a problem
  - Copies of a variable can be present in multiple caches
  - A write by one processor may not become visible
  - Accessing stale value in their caches
  - *Cache coherence* problem
- What do we do about it?
  - Organize the mem hierarchy to make it go away
  - Detect and take actions to eliminate the problem

# Architectural Building Blocks

- Bus Transactions
  - fundamental system design abstraction
  - single set of wires connect several devices
  - bus protocol: arbitration, command/addr, data
  - Every device observes every transaction
- Cache block state transition diagram
  - FSM specifying how disposition of block changes
  - invalid, valid, dirty

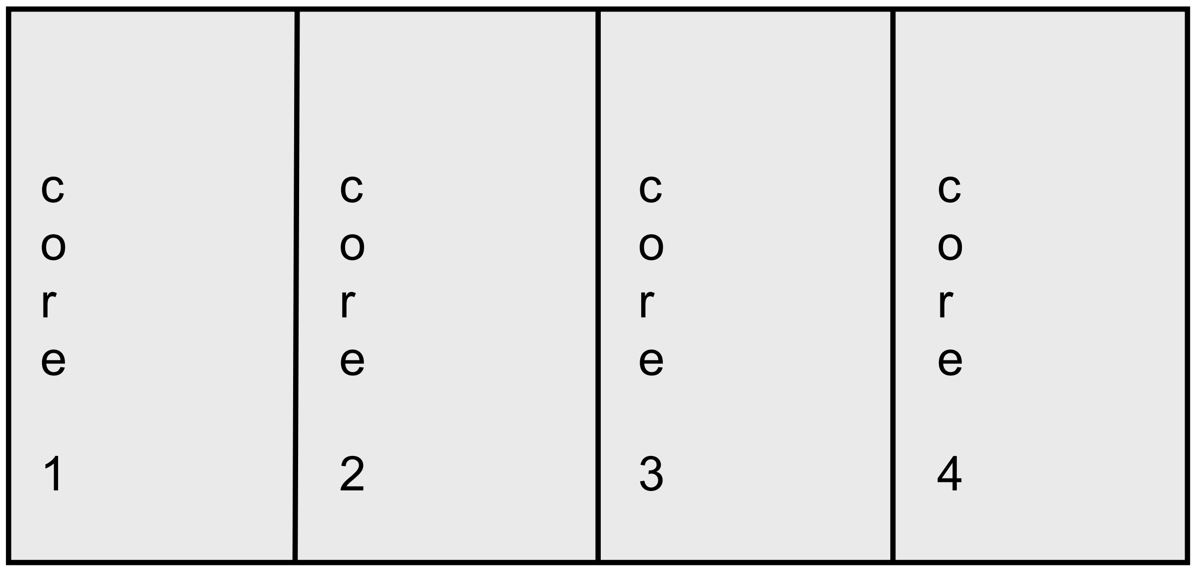
# Design Choices

- Cache Controller
  - Updates state of blocks in response to processor and snoop events
  - Generates bus transactions
- Action Choices
  - Write-through vs. Write-back
  - Invalidate vs. Update



## Multi-core CPU chip

- All multiprocessor concepts apply
- The cores fit on a single processor socket
- Also called CMP (Chip Multi-Processor)

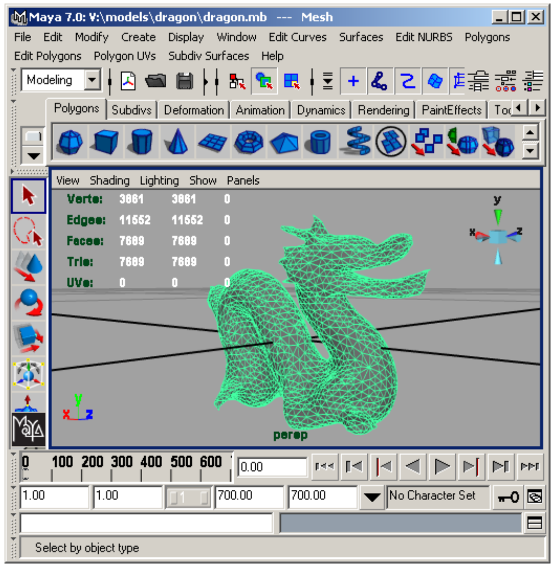


# Why Multiple Processors

- Moore's Law in CMOS
- Deeply Pipelined circuits
  - Large size processors
  - Difficult hardware design and verification
  - Large design teams necessary
- Many Dumb But Useful Applications
  - Many new applications are multithreaded
  - General trend in computer architecture
- Marketing Strategy
  - Increased number of processors
  - Potential for instantaneous high performance

# Today's Cloud Application

- Database servers
- Web servers (Web commerce)
- Compilers
- Multimedia applications
- Scientific applications, CAD/CAM
- In general, applications with *Thread+ level parallelism* (as opposed to instruction-level parallelism)


Maya 7.0: V:\models\dragon\dragon.mb --- Mesh
File Edit Modify Create Display Window Edit Curves Surfaces Edit NURBS Polygons
Edit Polygons Polygon UVs Subdiv Surfaces Help
Modeling
Polygons Subdivs Deformation Animation Dynamics Rendering PaintEffects
View Shading Lighting Show Panels
persp
Select by object type

Each can run on its own core