Memory-Mapped I/O Continued
Multiprocessor Architectures

Dr. Igor Ivkovic

iivkovic@uwaterloo.ca

[with material from “Computer Organization and Design” by Patterson and Hennessy, and “Digital Design and Computer Architecture” by Harris and Harris, both published by Morgan Kaufmann]

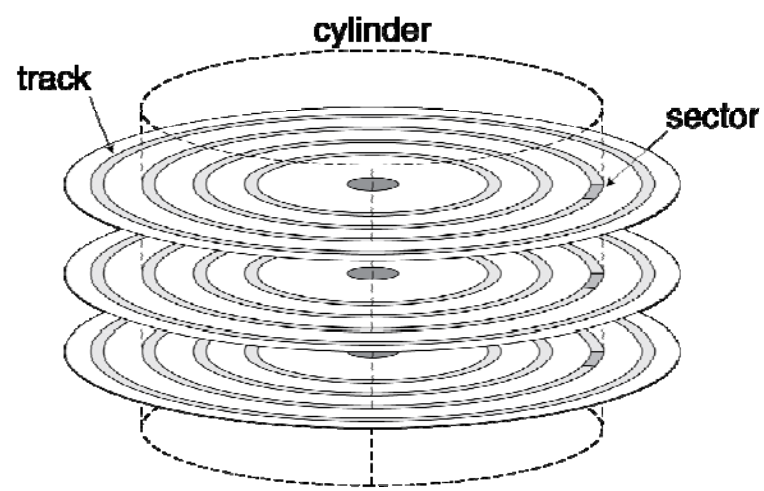
Objectives

- Additional notes on memory-mapped input and output
- Multiprocessor architectures
- Final exam overview

More on Memory-Mapped I/O /1

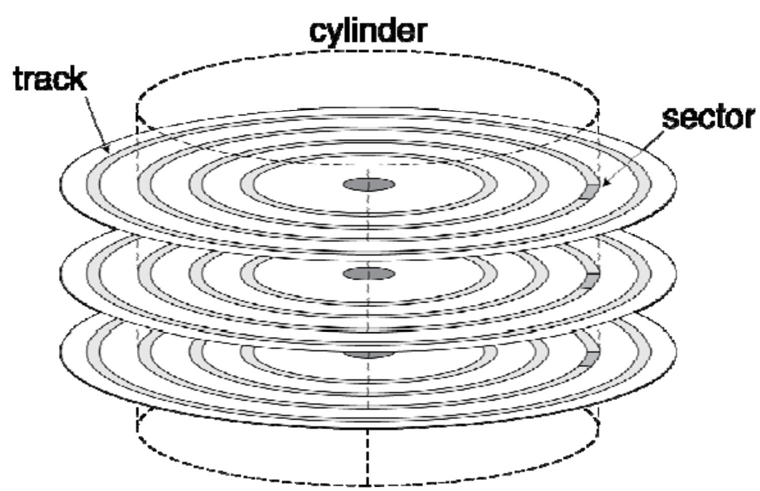
- **Disk Storage:**
  - Nonvolatile, rotating magnetic storage
  - There are 1 to 4 platters present
  - Platters rotate at 5,400 to 15,000 RPM
  - There are 10,000 to 50,000 tracks per surface
    - Higher density disks have higher density of tracks
  - Each track typically contains 100 to 500 sectors
  - Sectors are typically 512 bytes in size (can range to 4096 bytes)

- **Each sector records:**
  - Sector ID
  - Gap
  - Data (sectors are typically 512 bytes in size)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
  - Gap



More on Memory-Mapped I/O /2

- **Access to a sector involves**
  - Queuing delay if other accesses are pending
  - Seek: Move the head over the required track
  - **Seek time:** Delay required to reach the required track
  - **Rotational latency:** Delay required to reach the required sector
    - For 5,400 RPM HDD, average latency is 5.6 ms
    - For 15,000 RPM HDD, average latency is 2.0 ms
- **Access to a sector involves (continued)**
  - **Data transfer:** Delay that is a function of the sector size, the rotation speed, and the recording density of the track
  - **Controller overhead:** I/O overhead for transferring data to and from disk



More on Memory-Mapped I/O /3

- **Disk Access Example:**
  - **Given:**
    - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
  - **Average read time can be computed as follows:**
    - 4ms seek time
      + ½ / (15,000/60) = 2ms rotational latency
      + 512 / 100MB/s = 0.005ms transfer time
      + 0.2ms controller delay
      = 6.2ms
  - If actual seek time is 1ms, then average read time = 3.2ms

More on Memory-Mapped I/O /4

- **Manufacturers quote average seek time:**
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average seek times
- **SCSI, ATA, SATA interfaces also include a microprocessor to handle disk access optimization**
- **Disk drives also include caches:**
  - Prefetch sectors in anticipation of access
  - Avoid seek and rotational delay

More on Memory-Mapped I/O /5

- **Flash Storage:**
  - Nonvolatile semiconductor storage
  - 100 to 1000 times faster than hard disk
  - Smaller and requires less power
  - More cost per GB than hard disk but less than DRAM
- **NOR flash: A bit cell like a NOR gate**
  - Random read/write access
  - Used for instruction memory in embedded systems
- **NAND flash: A bit cell like a NAND gate**
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage, and so on

More on Memory-Mapped I/O /6

- **Flash Storage Limitations:**
  - Flash bits wear out after 1000’s of accesses
    - Some studies quote up to 100,000 reliable cell writes
    - But for USB flash storage, many manufacturers quote only 1,500 reliable connections of the device
  - Hence, flash storage is not suitable for direct RAM or disk replacement
  - **Wear levelling:** Steps taken to extend the lifetime of flash storage by remapping data to less used blocks

More on Memory-Mapped I/O /7

- **What is RAID?**
  - Redundant Array of Independent (Inexpensive) Disks
  - Uses multiple smaller disks as one larger disk
  - Parallelism of disk drives improves performance
  - Extra disks can be used for redundant data storage
- **RAID systems provide fault tolerant storage**
  - Especially if failed disks can be hot swapped (replaced while the system is active)
- **Disk striping:**
  - Split data over multiple disks (up to 32 disks)
  - For example, to write 256 KB of data over 4 disks, split the data and write 64 KB into each disk

More on Memory-Mapped I/O /8

- **RAID 0:**
  - No redundancy
  - Just stripe data over multiple disks
  - It does however improve performance
- **RAID 1: Mirroring**
  - N + N disks, replicate data
  - Write data to both data disk and mirror disk
  - On disk failure, read from mirror
- **RAID 2: Error correcting code (ECC)**
  - N data disks + E error-correcting disks (e.g., 10 + 4)
  - Split data at bit level across N disks
  - Generate E-bit ECC
  - Generally too complex, and not often used in practice

More on Memory-Mapped I/O /9

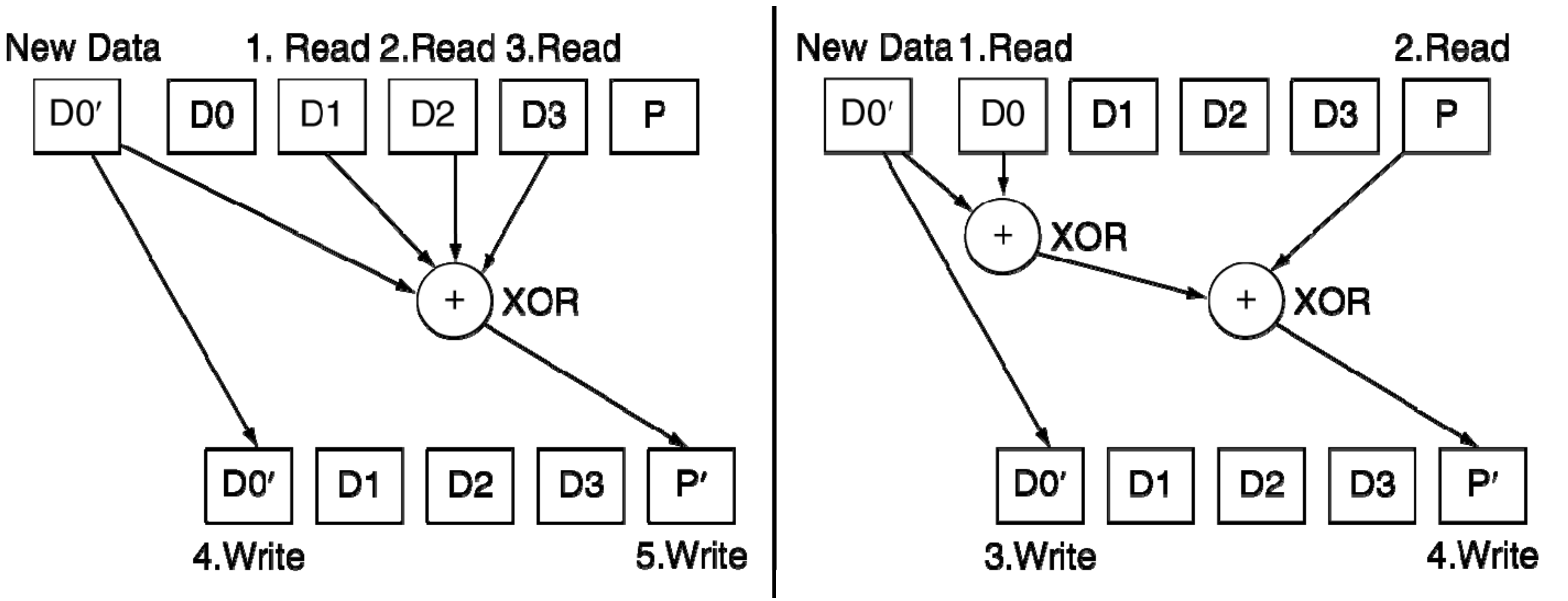
- **RAID 3: Bit-Interleaved Parity**
  - N + 1 disks
  - **Data striped across N disks at byte level**
  - Redundant disk stores parity
  - Read access
    - Read all disks
  - Write access
    - Generate new parity and update all disks
  - On failure
    - Use parity to reconstruct missing data

More on Memory-Mapped I/O /10

- **RAID 4: Block-Interleaved Parity**
  - N + 1 disks
  - **Data striped across N disks at block level**
  - Redundant disk stores parity for a group of blocks
  - Read access
    - Read only the disk holding the required block
  - Write access
    - Just read disk containing modified block, and parity disk
    - Calculate new parity, update data disk and parity disk
  - On failure
    - Use parity to reconstruct missing data

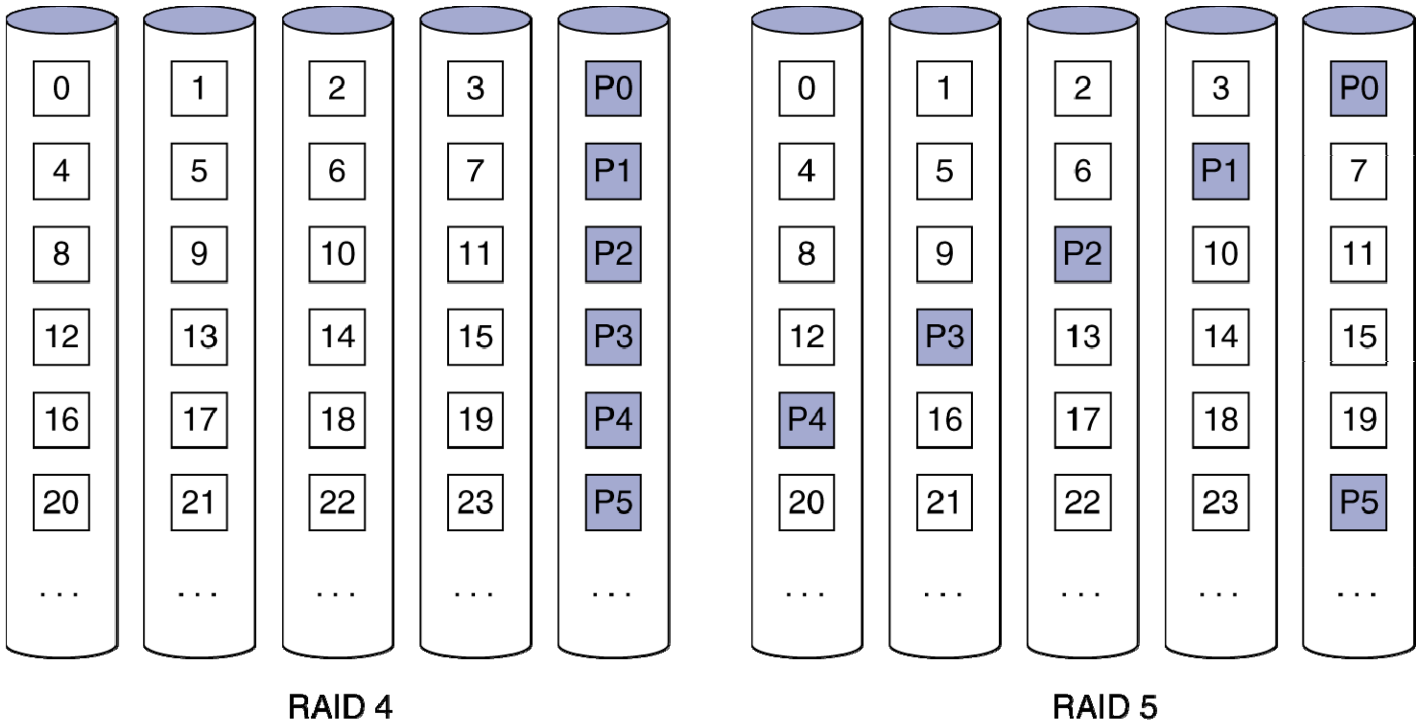
More on Memory-Mapped I/O /11

- **RAID 3 vs. RAID 4**



More on Memory-Mapped I/O /12

- **RAID 5: Distributed Parity**
  - N + 1 disks
  - Like RAID 4, but single parity blocks distributed across disks
  - Avoids parity disk being a bottleneck
  - Widely used

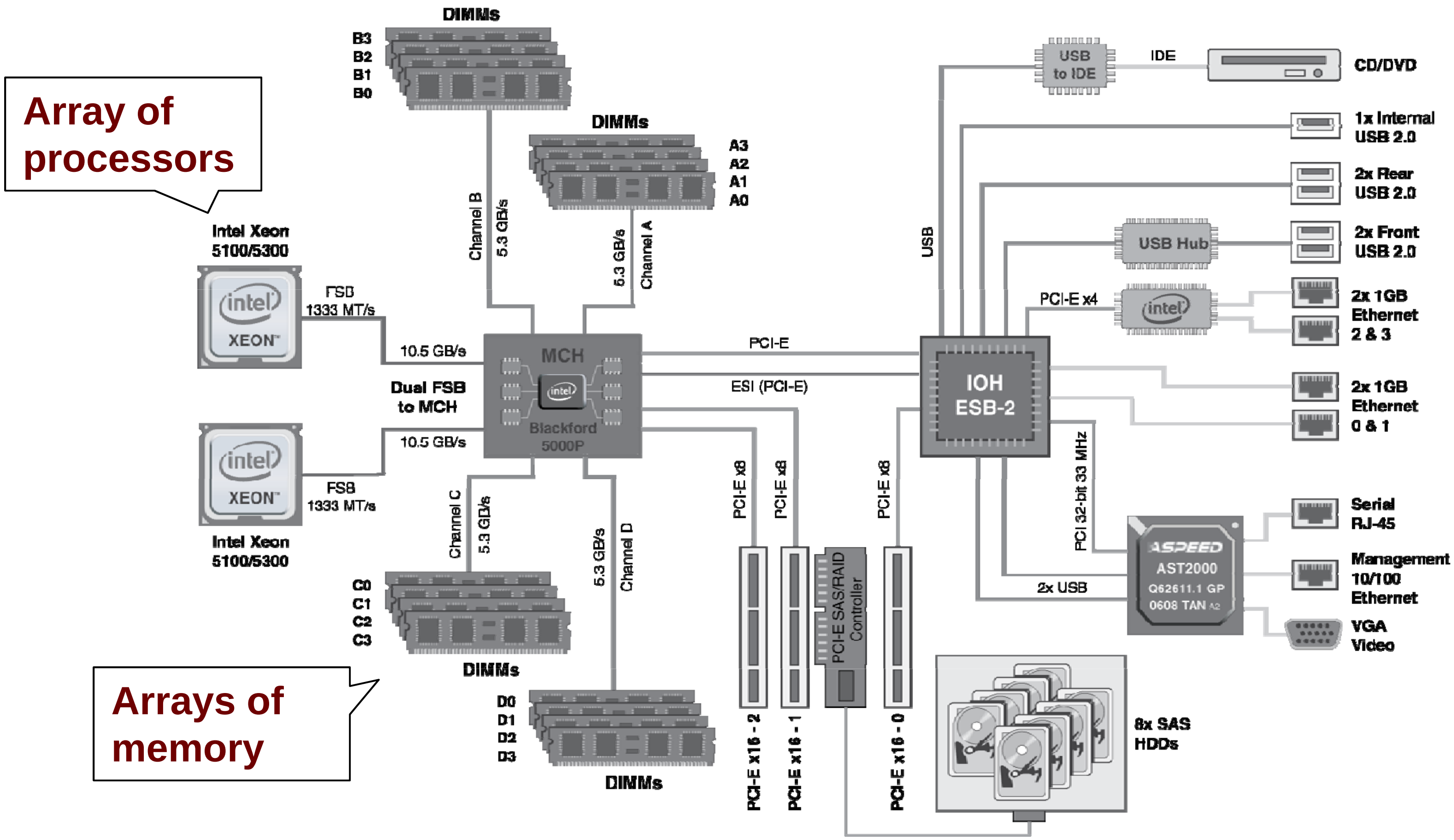


More on Memory-Mapped I/O /13

- **RAID 6: Distributed Parity**
  - N + 2 disks
  - Like RAID 4, but two parity blocks distributed across disks
  - Greater fault tolerance through more redundancy
  - Also widely used
- **RAID 0+1: Stripe then Mirror**
  - Striped Raid 0 drives are composed into Raid 1 array
- **RAID 10: Mirror then Stripe**
  - Mirrored Raid 1 drives are composed into Raid 0 array

More on Memory-Mapped I/O /14

- **Memory-Mapped I/O and Server Design:**
  - Applications are increasingly run on servers
  - Multiple processors, networks connections, massive storage



Multiprocessor Architecture /1

- **Parallelism in execution so far:**
  - We have already discussed processor-level parallelism with respect to pipelining
  - Multiple instructions are executed in parallel by using different components of the processor datapath
  - Out-of-order architectures and superscalar architectures further extend this idea, by allowing different scheduling of instructions and loading of multiple instructions at once, respectively
  - We have also explained **Single Instruction Multiple Data (SIMD)** approach that allows multiple data elements (packets) to be processed by the single instruction
  - How do we extend the parallelism idea further?

Multiprocessor Architecture /2

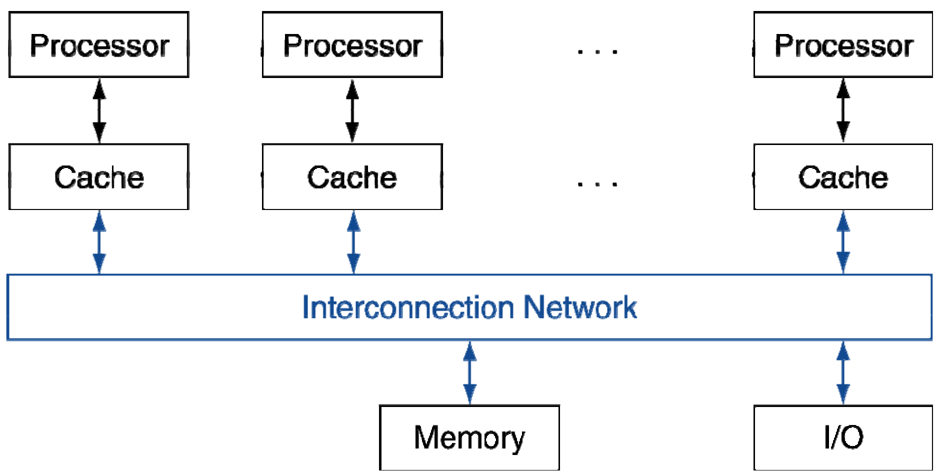
- **Parallelism in execution continued:**
  - Use multicore architectures, where a process is partitioned into multiple processing tasks
  - Each task is then run concurrently on different processor cores
    - Also called Multiple Instruction, Multiple Data (MIMD)
- **Parallel software is most often the obstacle**
  - **Need to get significant performance improvement**
  - Otherwise, just use a faster single processor since the implementation is easier with less overhead

Multiprocessor Architecture /3

- **Typical parallel execution difficulties:**
  - **Partitioning** – splitting the program into individual components that can execute in parallel
    - This may not be always possible (see parallel algorithms)
    - Compiler needs to recognize code that can be run in parallel
  - **Coordination** – manage program component execution and synchronize results
    - Ensure load balancing that keeps program tasks distributed across processing cores
    - Synchronize the results at the end of each parallel unit (e.g., after executing multiple threads) to allow the program to proceed
  - **Communications overhead** – ensuring that individual processing results are mutually consistent
    - Use shared memory access, data locking, message passing

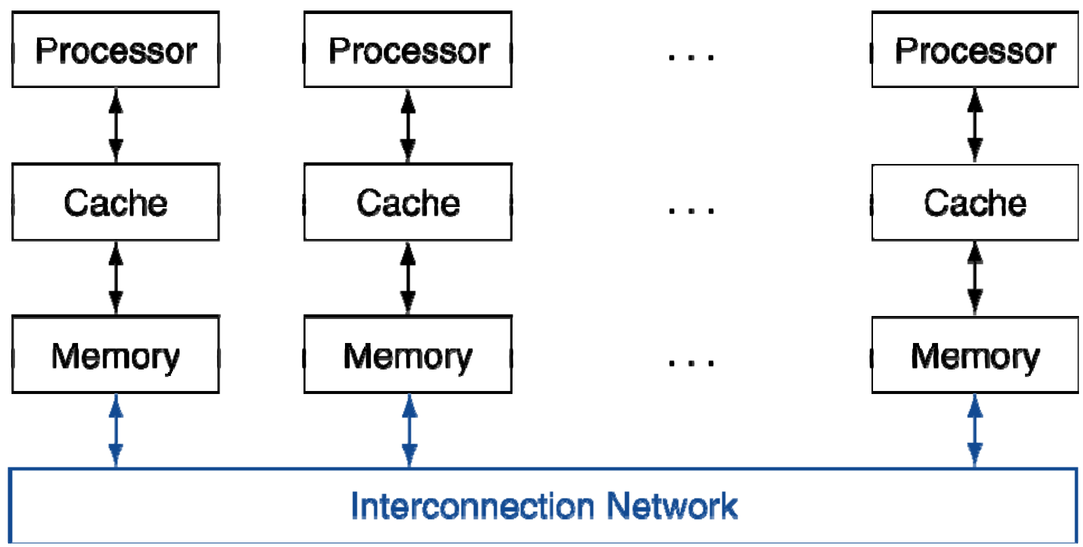
Multiprocessor Architecture /4

- **Shared Memory Multiprocessor (SMP):**
  - **Hardware provides single physical address space for all processors**
  - **Synchronize shared variables using locks**
  - Uniform Memory Access (UMA) Model:
    - The same access time for all processors for all included memory
  - Nonuniform Memory Access (NUMA) Model:
    - Some processors get faster access to specific types of memory (e.g., memory local to the processor is accessed faster)



Multiprocessor Architecture /5

- **Message Passing:**
  - **Each processor has private physical address space**
  - **Controller sends/receives messages between processors**
  - Due to the higher setup cost, message-passing architectures are generally implemented as clusters
    - Clusters are collections of commodity computers that are connected to each other over I/O interconnect



Multiprocessor Architecture /6

- **Loosely Coupled Clusters:**
  - Network of independent computers
  - Each has private memory and OS
  - Connected using I/O system
    - For example, Ethernet/switch, Internet
  - Suitable for applications with independent tasks
    - Examples include web servers, databases, simulations, etc
  - Quality attributes: high availability, scalable, affordable
- **Challenges:**
  - Administration cost per machine
    - Can be addressed using virtual machines
  - Low interconnection bandwidth

Multiprocessor Architecture /7

- **Grid Computing:**
  - Separate computers interconnected by long-haul networks
    - For example, Internet connections
  - Work units are farmed out, and then the results are sent back
  - Can make use of idle time on PCs
- Some examples of grid computing:
  - **SETI@home**
    - Search for Extraterrestrial Intelligence (SETI) project run by University of California at Berkeley
    - Distributes radio telescope data for processing by idle PC nodes
    - Over 5 million independent users quoted by the project hosts
  - **World Community Grid**
    - Non-profit computing grid that is intended to support scientific research that benefits humanity (supported by IBM)
    - For example, searching for better treatments of muscular dystrophy and other neurological diseases

Food for Thought

- **Read:**
  - Chapter 8 from the Course Notes
    - Review the material discussed in the lecture notes in more detail
    - Our course schedule follows the material in the Course Notes
  - Recommended: Chapter 7 from the course textbook
  - **Prepare for the final exam**
    - **Held on Mon Apr 22nd, from 12:30 to 3:00pm in PAC 4 & 5**

Final Examination Overview /1

- **Final Exam will cover:**
  - Lecture Notes #4 to #10
    - Review notes #1 to #3 as necessary
  - “Food for Thought” Readings and Exercises
  - Assignments #2 (excluding Q1) to #5
  - <u>Carefully Review Midterm Solutions</u>
- **Main Topics – before the midterm:**
  - Arithmetic Logic Unit (Notes #4)
    - One question on rotator/shifter circuitry
    - Tracing instructions through the ALU (similar to Assignment #2)
  - Floating Point Representation (Notes #5)
    - Calculations similar to Assignment #2
  - Single-Cycle Datapath (Notes #5)
    - Tracing instructions through the datapath

Final Exam Overview /2

- **Main Topics – after the midterm:**
  - Single-Cycle Processor Control Unit (Notes #6)
    - Adding instructions to the control unit
    - Questions similar to Assignment #3
  - Multi-Cycle Processor Datapath and Control Unit (Notes #6)
    - Tracing instructions through the datapath
    - Questions similar to Assignment #3
  - Pipelined Processor (Notes #7)
    - Tracing instructions through the datapath
    - Addressing data and control hazards
    - Questions similar to Assignment #4
  - Memory Management (Notes #8)
    - Using cache memory, including direct-mapped, n-way associative, and fully-associative cache
    - Using virtual memory, including page faults and TLB cache
    - Questions similar to Assignment #5

Final Exam Overview /3

- **Main Topics – after the midterm:**
  - Memory-Mapped Input and Output (Notes #9 and #10)
    - Using memory-mapped input and output, including address decoding and control of memory-mapped I/O
    - Disk storage details and RAID configurations
  - Multiprocessor Architectures (Notes #10)
    - The basics of parallelism, including parallelism in pipelined architectures, SIMD, MIMD, and Shared-Memory Processor

Concluding Remarks

- By the time you've sorted out a complicated idea into little steps that even a stupid machine can deal with, you've certainly learned something about it yourself. – Douglas Adams

- It's hardware that makes a machine fast. It's software that makes a fast machine slow.
  – Craig Bruce

- Any intelligent fool can make things bigger and more complex... It takes a touch of genius - and a lot of courage to move in the opposite direction.
  – Albert Einstein

[BrainyQuote. Online, 2013. http://www.brainyquote.com]