CS5375 Computer Systems Organization and Architecture

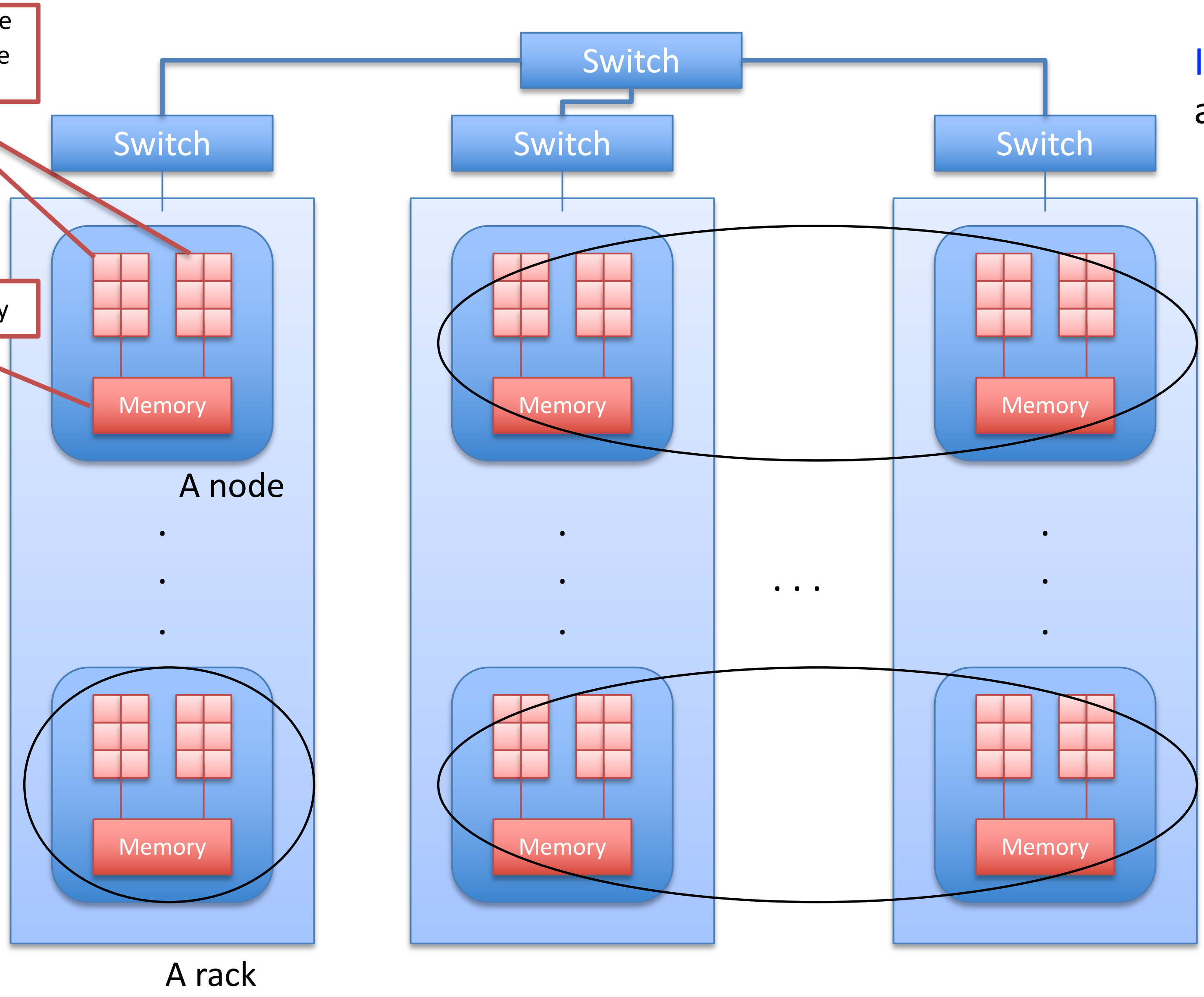
Lecture 23

Instructor: Yong Chen, Ph.D.
Department of Computer Science
Texas Tech University
Yong.Chen@ttu.edu, 806-834-0284

Outline

- Distributed Memory, Large-scale Computers
- Programming Models
- Infrastructure of Warehouse-Scale Computer

A Hrothgar@TTU Parallel Computer Example



Interconnect (topology also matters)

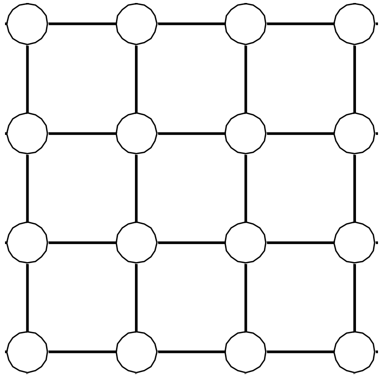
Potentially shared address space (specially NUMA) if with HW/SW support (note distributed memory!)

Shared address space (shared memory also) (only scales to 12 cores)

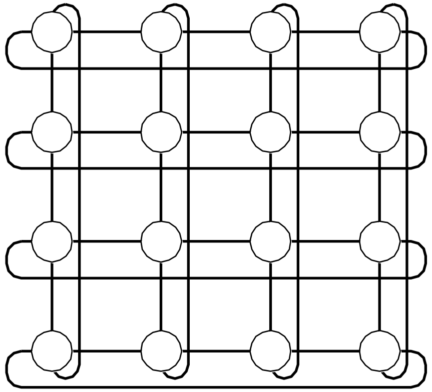
Distributed address space (distributed memory also) (can scale to 12*640=7680 cores)

Interconnect Topology

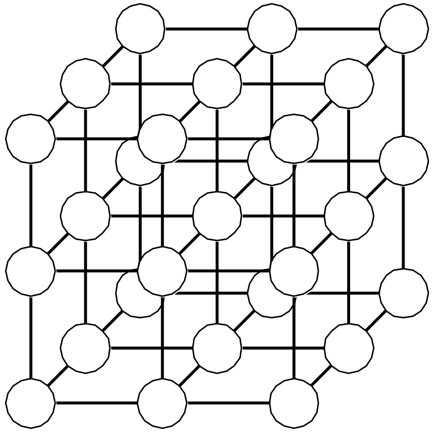
- Linear array, ring/1-d torus
- 2-d mesh, 2-d torus
- Tree, fat tree



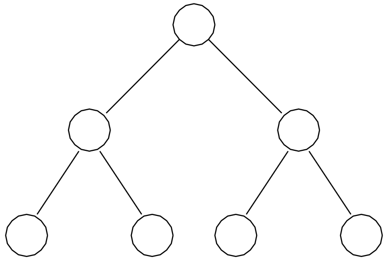
2-D mesh: without wraparound links



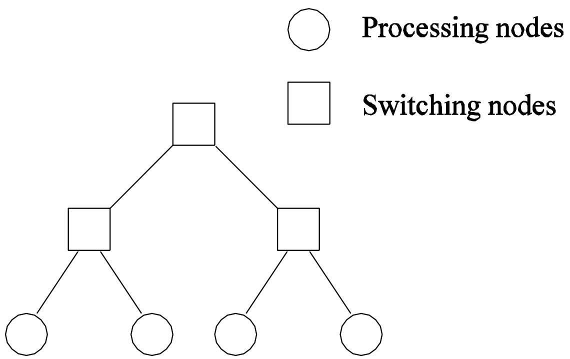
2-D torus: with wraparound links



A 3-D mesh with no wraparound links



A static tree network

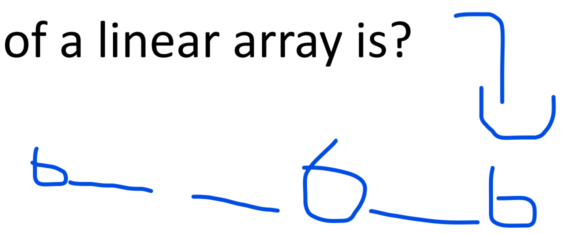


A dynamic tree network

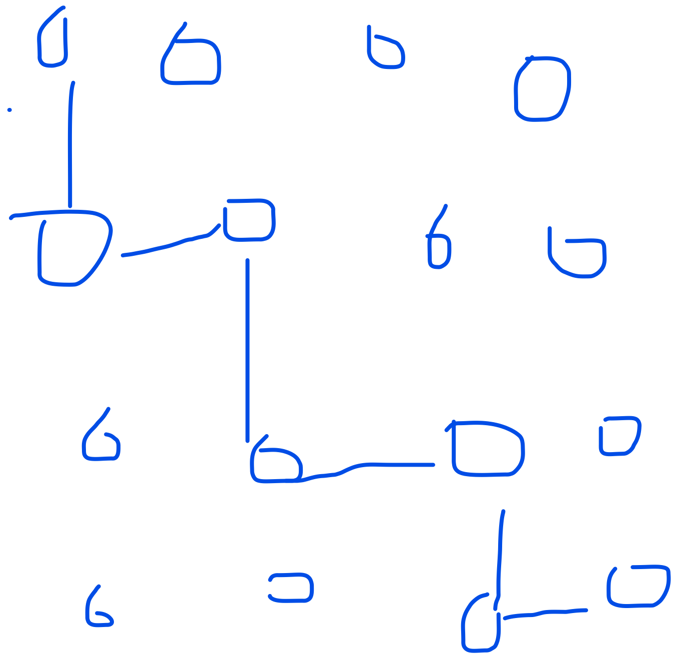
Evaluating Interconnect Topologies: Diameter

- Diameter: the maximum distance between any two nodes
  - The distance between two nodes is defined as the shortest path (in terms of the number of links) between them

- The diameter of a linear array is?
  $p - 1$



- Mesh?
  $2(\sqrt{p} - 1)$



- Tree (complete binary tree)?
  $2\log((p+1)/2)\ \ or\ 2\,(\log(p+1) - 1)$

arc ntg but link

Evaluating Interconnect Topologies: Arc Connectivity

- Connectivity: a measure of the multiplicity of paths between any two nodes
  - A network with high connectivity is desirable, because it lowers contention for communication resources

- Arc connectivity: one measure of connectivity is the minimum number of arcs that must be removed from the network to break it into two disconnected networks

Evaluating Interconnect Topologies: Arc Connectivity

- Linear array?

  1

- Ring?

  2

- Mesh?

  2

- 2-D torus?

  4

- Tree?

  1

Evaluating Interconnect Topologies: Bisection Width

- Bisection Width: minimum number of links you must cut to divide the network into two equal parts

- The bisection width of a linear array and ring, respectively?

  1, 2

- Mesh, 2D-torus?
- $\sqrt{p}$
- $2\sqrt{p}$

- Tree? Here tree is only binary tree i.e. one line tree

  1

Evaluating Interconnect Topologies: Cost

- Cost: many criteria can be used to evaluate the cost of a network

- One way of defining the cost of a network is in terms of the number of communication links or the number of wires required by the network

- However, a number of other factors, such as the ability to layout the network, the length of wires, etc., also factor into the cost

Evaluating Interconnect Topologies: Cost (Number of links)

This section is to total number of links respectively

- Linear array, ring?

  $p-1,\ p$

- 2-D mesh, 2-D torus

  $2(p-\sqrt{p}) \rightarrow 2 \cdot \sqrt{p}\,(\sqrt{p}-1$

  $\boxed{2p} \rightarrow \frac{p \cdot 4}{2}$

- Tree (complete binary tree)?

  $p-1$

Outline

- Distributed Memory, Large-scale Computers
- Programming Models
- Infrastructure of Warehouse-Scale Computer

Programming Models

- MPI (Message Passing Interface) programming model
- MapReduce programming model
  - “MapReduce: Simplified Data Processing on Large Clusters”, by Jeffrey Dean and Sanjay Ghemawat, Google, Inc.

Principles of Message-Passing Programming

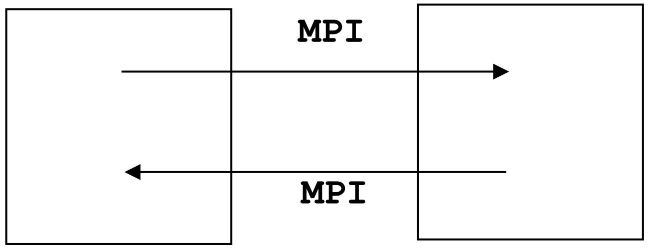
- The logical view of a distributed-address-space parallel computer supporting the message-passing paradigm consists of p processes, each with its own exclusive address space

- Each data element must belong to some process' address space; hence, data must be explicitly partitioned and placed

- All interactions (read-only or read and write) require cooperation of two (or more) processes - the process that has the data and the process that wants to access the data

Principles of Message-Passing Programming

- Message passing model is for communication among processes (with separate address spaces)
- Interprocess communication consists of
  - Movement of data from one process's address space to another's
  - Synchronization



- All parallelism is explicit: the programmer is responsible for correctly identifying parallelism and implementing parallel algorithms using MPI constructs

MPI (Message Passing Interface) Programming Model

- Point-to-point communication routines
  - Blocking: MPI_Send, MPI_Recv, etc.
  - Non-blocking: MPI_Isend, MPI_Irecv, etc.

- Collective communication routines
  - MPI_Bcast, MPI_Scatter, MPI_Gather, MPI_Reduce, MPI_Barrier, etc.

- Basic and derived data types

- Virtual topologies, communicators

- A tutorial: https://hpc-tutorials.llnl.gov/mpi/

MapReduce Programming Model

- Simplified data processing on large clusters

- Map: applies a programmer-supplied function to each logical input record
  - Runs on thousands of computers
  - Provides new set of key-value pairs as intermediate values

- Reduce: collapses values using another programmer-supplied function

MapReduce Programming Model (cont.)

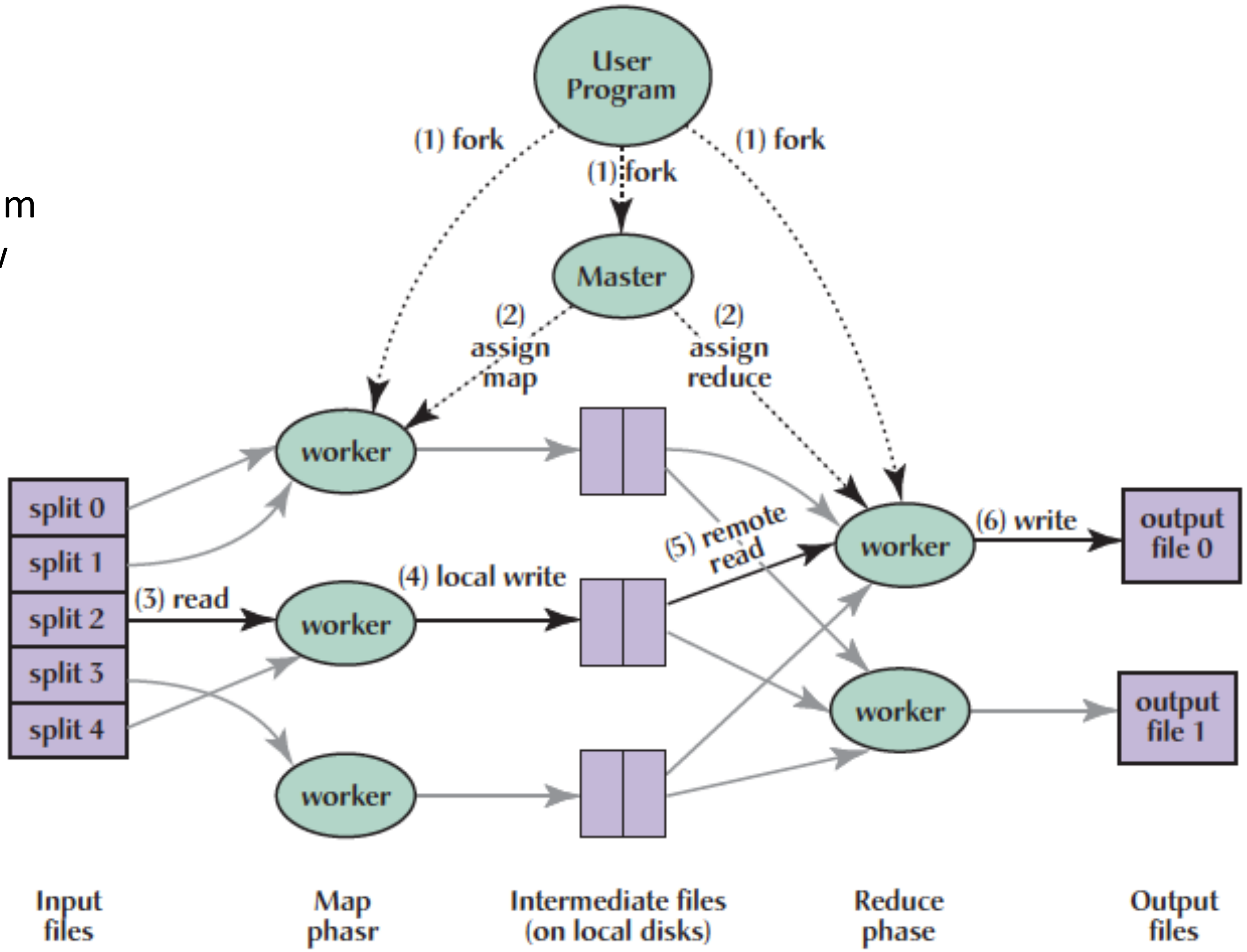
- Word count example

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
EmitIntermediate(w, "1");

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

MapReduce Programming Model (cont.)

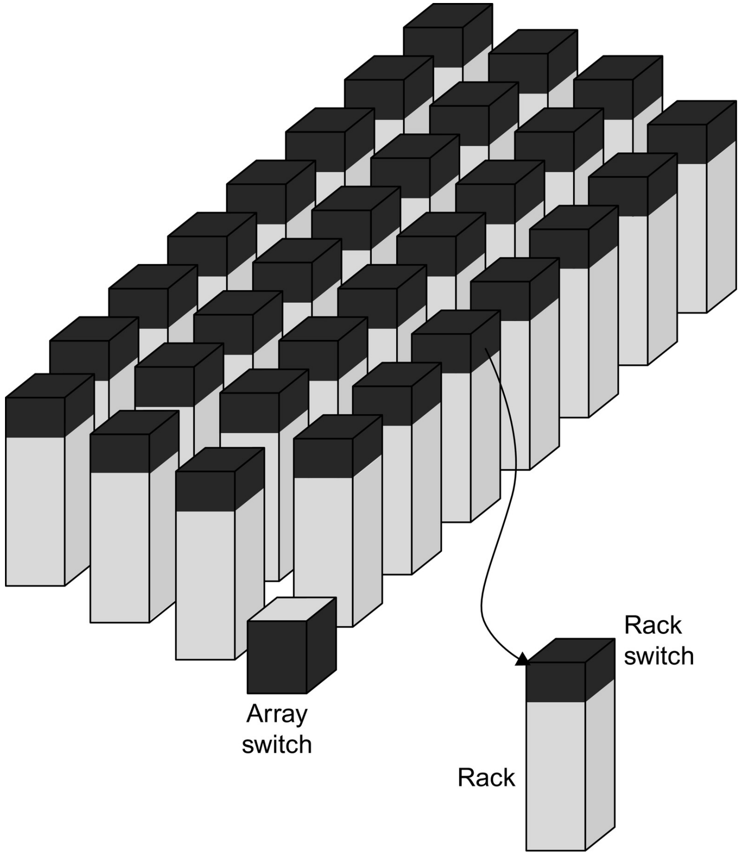
MapReduce program execution overview



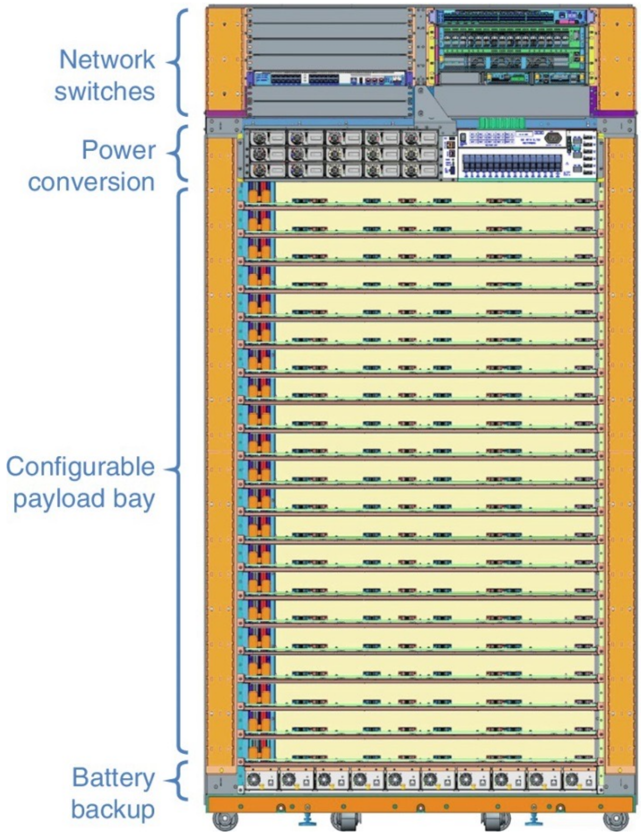
MapReduce Programming Model (cont.)

- MapReduce runtime environment schedules map and reduce task to WSC (warehouse-scale computer) nodes
  - Workload demands often vary considerably
  - Scheduler assigns tasks based on completion of prior tasks
  - Tail latency/execution time variability: single slow task can hold up large MapReduce job
  - Runtime libraries replicate tasks near end of job

Infrastructure of Warehouse-Scale Computer



Hierarchy of switches in a WSC



A Google rack for its WSC

Infrastructure of Warehouse-Scale Computer

An example server from a Google WSC. The Haswell CPUs (2 sockets × 18 cores × 2 threads = 72 “virtual cores” per machine) have 2.5 MiB last level cache per core.

Infrastructure of Warehouse-Scale Computer (cont.)

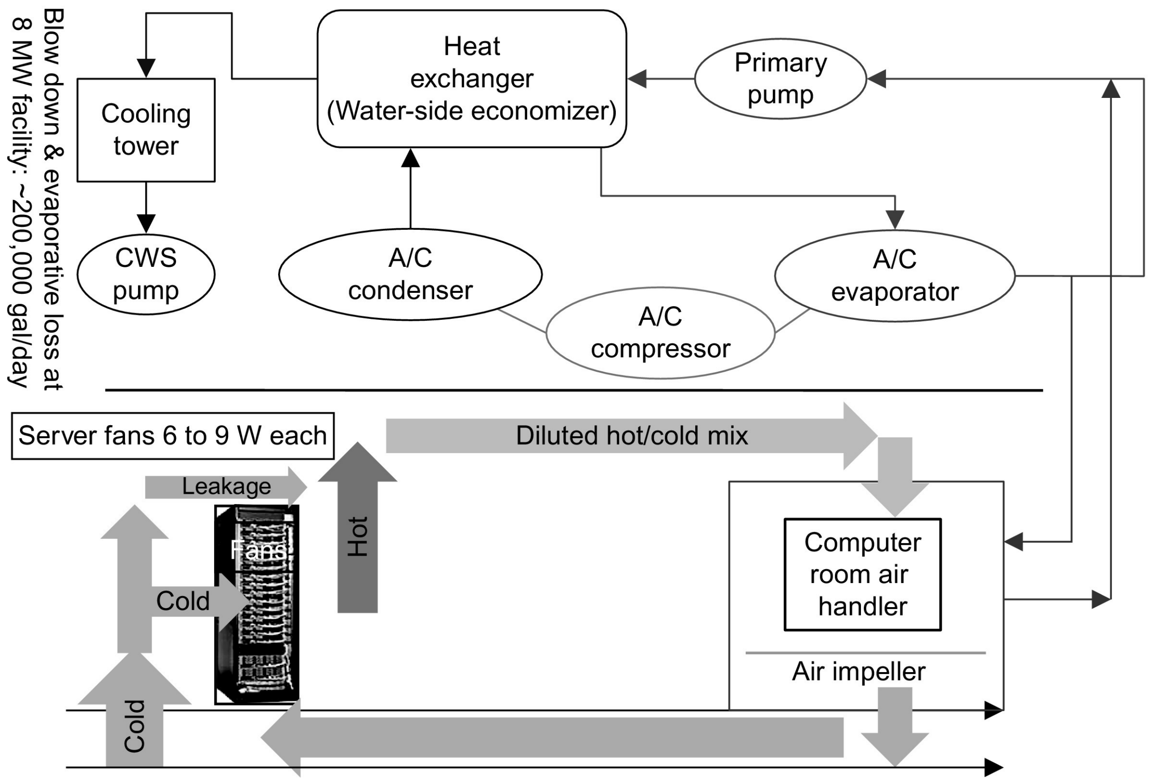
- WSC memory hierarchy

	Local	Rack	Array
DRAM latency (μs)	0.1	300	500
Flash latency (μs)	100	400	600
Disk latency (μs)	10,000	11,000	12,000
DRAM bandwidth (MB/s)	20,000	100	10
Flash bandwidth (MB/s)	1000	100	10
Disk bandwidth (MB/s)	200	100	10
DRAM capacity (GB)	16	1024	31,200
Flash capacity (GB)	128	20,000	600,000
Disk capacity (GB)	2000	160,000	4,800,000

Infrastructure of Warehouse-Scale Computer (cont.)

- Cooling
  - Air conditioning used to cool server room
  - Cooling system also uses water
    - E.g., 70,000 to 200,000 gallons per day for an 8 MW facility

- Typical power usage by component:
  - Processors: 42%
  - DRAM: 12%
  - Disks: 14%
  - Networking: 5%
  - Cooling: 15%
  - Power overhead: 8%
  - Miscellaneous: 4%



Cost of a WSC

- Capital expenditures (CAPEX)
  - Cost to build a WSC

- Operational expenditures (OPEX)
  - Cost to operate a WSC

Cloud Computing: The Return of Utility Computing

- A large-scale computing system that
  - Focus on hosting data and applications for users by utilizing service-oriented architecture, virtualization and storage techniques
- Driven by
  - Industry, economies of scale, pay as you go model – attractive for small/medium-scale businesses
  - Virtualization, dynamically-scalable resources
  - Delivered on demand

Google Cloud

Microsoft Azure

amazon web services

Cloud Computing: The Return of Utility Computing (cont.)

In 2017 Google had 15 sites. In the Americas: Berkeley County, South Carolina; Council Bluffs, Iowa; Douglas County, Georgia; Jackson County, Alabama; Lenoir, North Carolina; Mayes County, Oklahoma; Montgomery County, Tennessee; Quilicura, Chile; and The Dalles, Oregon. In Asia: Changhua County, Taiwan; Singapore. In Europe: Dublin, Ireland; Eemshaven, Netherlands; Hamina, Finland; St. Ghislain, Belgium. https://www.google.com/about/datacenters/inside/locations/.

Readings

- Chapter 6, 6.2-6.5
- Message Passing Interface (MPI) tutorial, by Blaise Barney, Lawrence Livermore National Laboratory: https://hpc-tutorials.llnl.gov/mpi/
- "MapReduce: Simplified Data Processing on Large Clusters", by Jeffrey Dean and Sanjay Ghemawat, Google, Inc., OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA (2004), pp. 137-150