Introduction to Cloud Computing

Distributed Systems

15-319, spring 2010

11th Lecture, Feb 16th

Majd F. Sakr

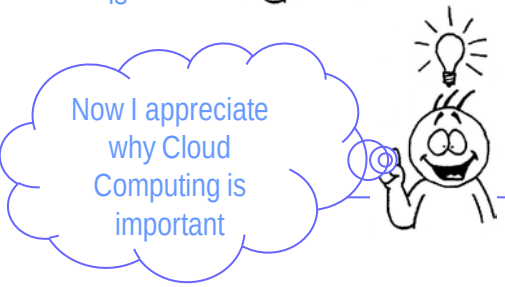
Lecture Motivation

- Understand Distributed Systems Concepts
- Understand the concepts / ideas and techniques from Distributed Systems which have made way to Cloud Computing

Lecture Outline

- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

Do you … ?



End of week two

Parallel Processing

End of week five

Distributed Systems

Time

What is a Distributed System?

- **Distributed Computing:** a CS field that studies distributed systems
- **Distributed System:** a group of independent/autonomous computers that
  - are networked together
  - appear to the user as a one computer
  - Work together to achieve a common goal

What is a Distributed System?

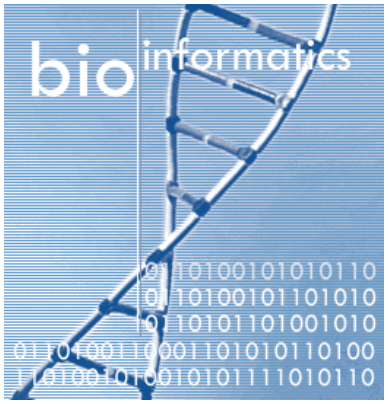
- **Distributed Computing:** a CS field that studies ideas around designing and building distributed systems and infrastructure to enable such systems
- **Distributed System:** a group of independent/autonomous computers that
  - are networked together
  - appear to the user as a one computer
  - Work together to achieve a common goal

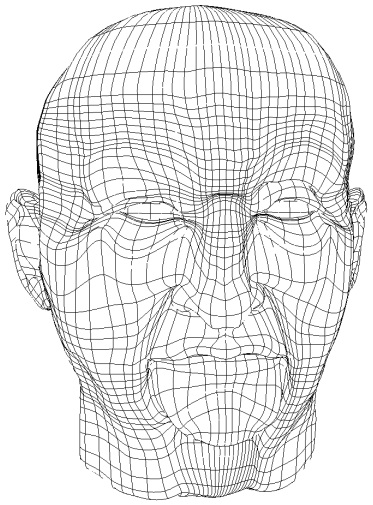
History

- Problems that are larger than what a single machine can handle
- Computer Networks, Message passing were invented to facilitate distributed systems
- ARPANET eventually became Internet

Where are they used?

- Strategic Systems (Defense / Intelligence)
- Bioinformatics
- Visualization and Graphics
- Economics and Finance
- Scientific Computing



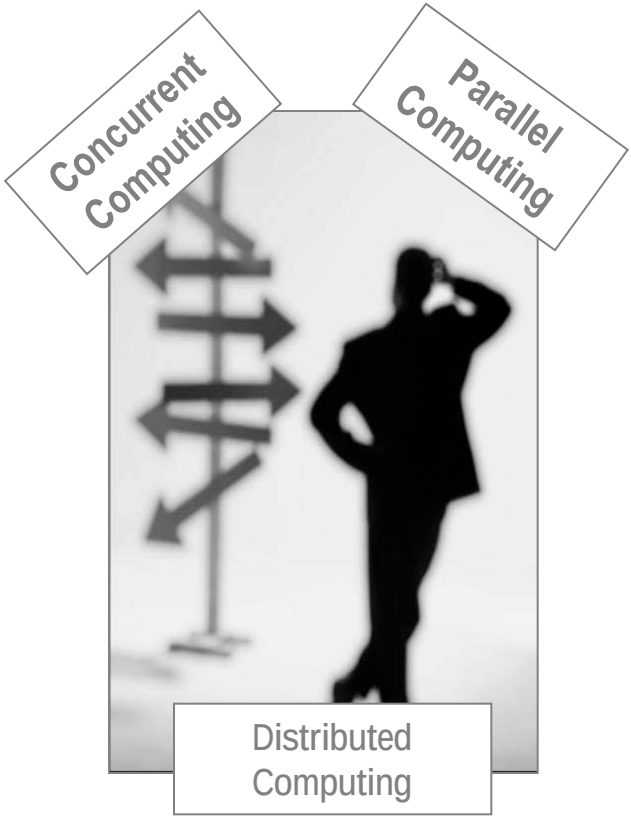


Lecture Outline

- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

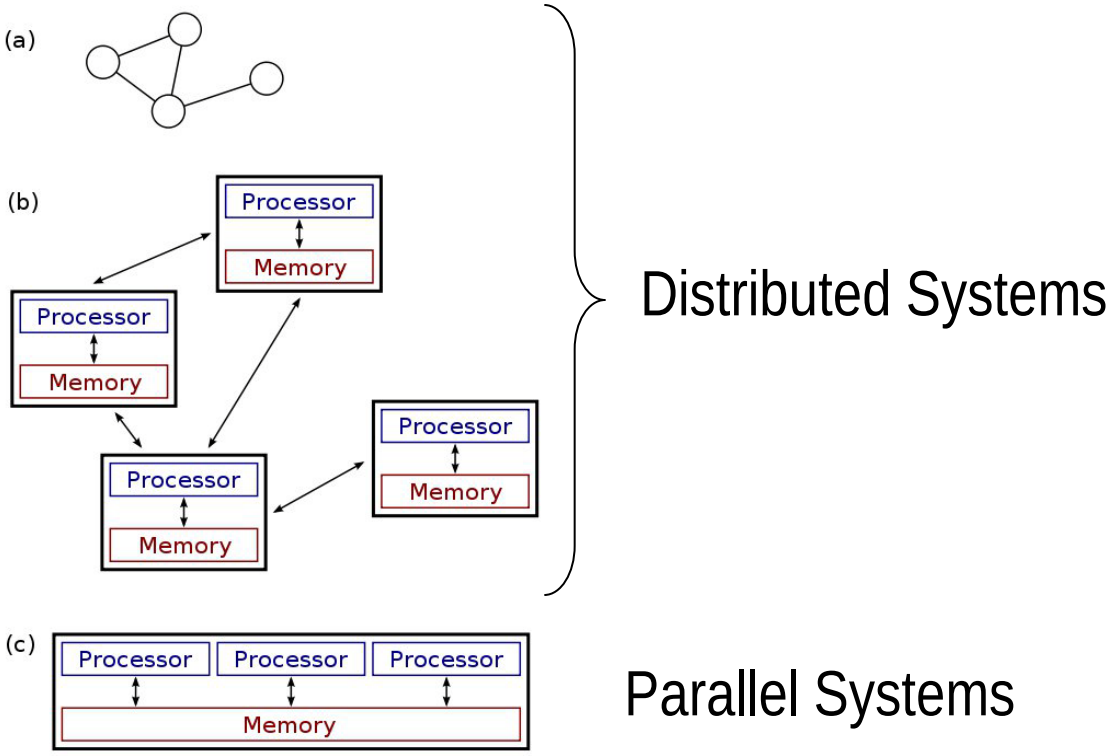
Parallel vs. Distributed Systems

- **A concurrent system could be Parallel or Distributed:**
  - Two possible Views to make the distinction
  - View 1:
    - *Parallel System :* a **particular tightly-coupled** form of distributed computing
    - *Distributed System:* a **loosely-coupled form** of parallel computing
  - View 2:
    - *Parallel System:* processors access a **shared memory** to exchange information
    - *Distributed System:* uses a "**distributed memory**". **Massage passing** is used to exchange information between the processors as each one has its own private memory.



Further Distinctions

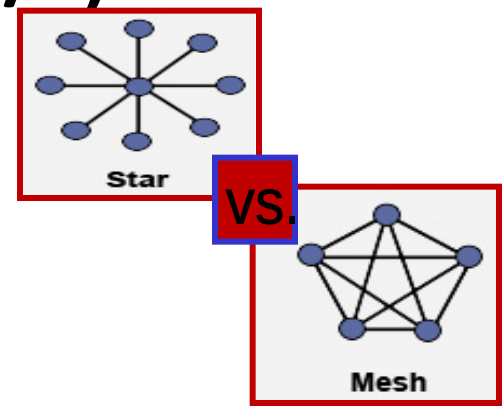
- **Granularity**
  - Parallel Systems are typically finer-grained Distributed Systems
  - Distributed Systems are typically the most coarse-grained.



Lecture Outline

- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

Advantages of Distributed Systems (1/2)



- **…Over Centralized Systems**
  - **Economics:**
    - lower (price/performance) ratio
  - **Speed:**
    - May have a more total computing power than a centralized system
    - Enhanced performance through load distributing.
  - **Inherent Distribution:**
    - Some applications are inherently distributed
  - **Availability and Reliability:** No single point of failure.
    - The system survives even if a small number of machines crash
  - **Incremental Growth:**
    - Can add computing power on to your existing infrastructure

Advantages (2/2)

- **…Over Independent PCs**
  - **Computation:** can be shared over multiple machines
  - **Shared management of system:** backups & maintenance…
  - **Data Sharing:** many users can access the same common database
  - **Resources Sharing:** can share expensive peripherals
  - **Flexibility:** Spreading workload over the system CPUs.

Disadvantages

- **Software:** Developing a distributed system software **is hard**
  - Creating OSs / languages that support distributed systems concerns
- **Network:** When network is overloaded/messages lost, rerouting/rewiring the network is costly/difficult
- **Security :** more sharing leads to less security especially in the issues of confidentiality & integrity
- **Incremental growth is hard in practice due to changing of hardware and software**

Lecture Outline

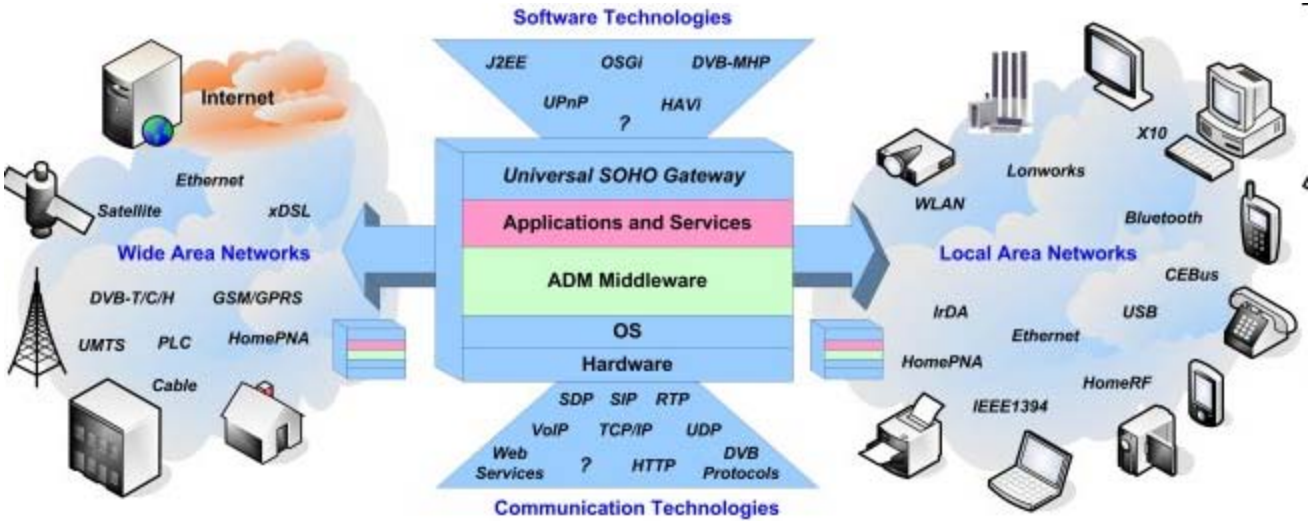
- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

Design Goals/ Characteristics

- **Distributed Transparency**
  - Location
  - Migration
  - Replication
  - Concurrency
- **Openness**
- **Scalability**
- **Fault-tolerance**
- **High availability**
- **Recoverability**
- **Performance Predictability**
- **Security**

One more characteristic: Heterogeneity

- **Distributed Systems are heterogeneous in terms of:**
  - **Hardware:** PCs, mainframes, servers, …
  - **Software:** different operating systems (MS Windows, UNIX, …)
  - **Unconventional devices:** telephone switches, robots, …
  - **Networking:** different protocols and networks (Ethernet, TCP/IP, FDDI, …)



http://anso.vtt.fi/graphics/ANSO_innovations.jpg

One more characteristic: Heterogeneity

- **Distributed Systems are heterogeneous in terms of:**
  - **Hardware:** PCs, mainframes, servers, …
  - **Software:** different operating systems (MS Windows, UNIX, …)
  - **Unconventional devices:** telephone switches, robots, …
  - **Networking:** different protocols and networks (Ethernet, TCP/IP, FDDI, …)
- 

  Middleware masks this heterogeneity
  - » An additional software layer

Lecture Outline

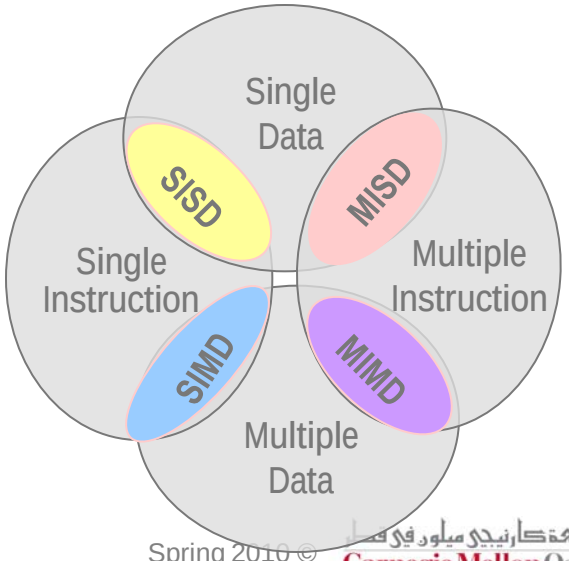
- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- **Distributed System Types**
- **Performance of Distributed Systems**
- **Programming Distributed Systems**

Distributed Systems Hardware

- MIMD
- Memory differentiated
  - Multicomputers
  - Multiprocessors
- Interconnection Network
  - Bus System
  - Switched System
- Coupling
  - Tightly coupled hardware
  - Loosely coupled hardware

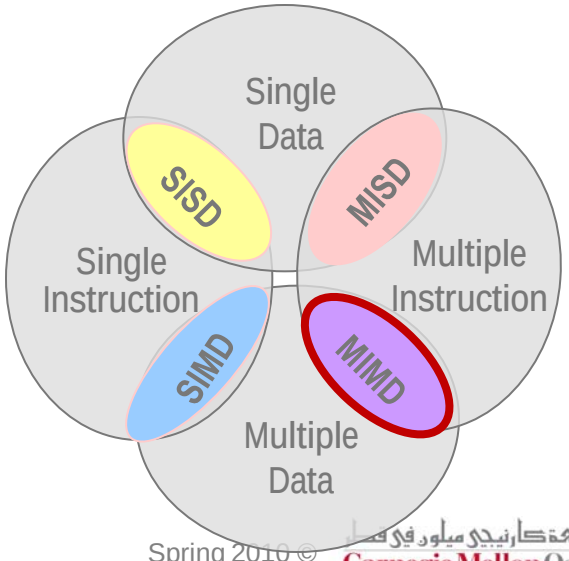
Hardware: MIMD

- **Remember Flynn's Taxonomy?**
- **The four possible combinations:**
  - **SISD:** in traditional uniprocessor computers
  - **MISD:** Multiple concurrent instructions operating on the same data element. Not useful!
  - **SIMD:** Single instruction operates on multiple data elements in parallel
  - **MIMD:** Covers parallel & distributed systems and machines that contain multiple computers



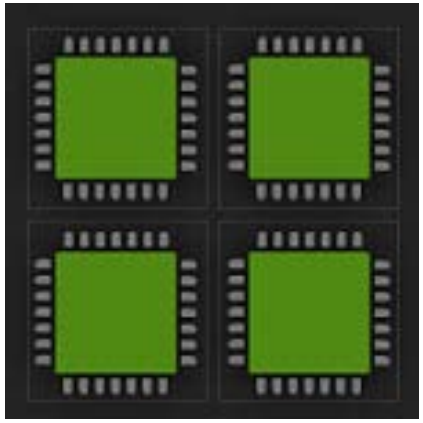
Hardware: MIMD

- **Remember Flynn's Taxonomy?**
- **The four possible combinations:**
  - **SISD:** in traditional uniprocessor computers
  - **MISD:** Multiple concurrent instructions operating on the same data element. Not useful!
  - **SIMD:** Single instruction operates on multiple data elements in parallel
  - **MIMD:** Covers parallel & distributed systems and machines that contain multiple computers
- **Distributed systems are MIMD**



Hardware: Memory Differentiated

- **Multicomputers:** machines without shared memory
  - Each machine has its special memory and address space
  - Example: multiple PCs connected by a network
- **Multiprocessors:** machines with shared memory
  - Single virtual address space
  - Access/modify same memory locations
  - SMP (Symmetric Multiprocessor): all processors are of the same type
  - **Bus-based multiprocessor**
    - One memory ←→several processors
    - Bus Overloaded → lower performance
      - Cache memory allows adding more CPUs before bus gets overloaded



Hardware: Interconnection Network

- **Bus System:**
  - Has single network medium connecting all processors
  - Medium could be: bus, backplane, cable, …
  - Examples:
    - Multiprocessors: Sequent, Encore SGI
    - Multicomputers: Workstations on a LAN
- **Switched System**
  - Has individual wires between machines
  - Messages sent along the wires
  - Routing/switching decisions made in step-by-step manner along the route
  - Examples:
    - Multiprocessors: Ultracomputer, RP3, …
    - Multicomputers: Hypercube, Transputer, …

Hardware: Coupling

- **Tightly coupled hardware**
  - Small delay in its network
  - Fast data transfer rate
  - Common in parallel systems
  - Multiprocessors are usually tightly coupled
- **Loosely coupled hardware**
  - Longer delay in sending messages between machines
  - Slower data transfer rate
  - Common in distributed systems
  - Multicomputers are usually tightly coupled

Lecture Outline

- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

Distributed Systems Software

- **Network OS**
  - OS containing components that facilitate network resource access / sharing
  - All modern OSes have network eatures
- **Integrated Distributed System**
- **Multiprocessor timesharing system**
  - Multics and UNIX

Distributed Systems Software

Hardware		Software	
		Loosely-coupled	**Tightly-coupled**
	Loosely-coupled	■ **Software Type: Network OS** ■ **Multicomputer** ■ **Each machine running its OS. OSes may differ.**	■ **Software Type: Integrated Distributed System** ■ **Group of Shared machines work like one computer but do not have shared memory**
	Tightly-coupled	■ **Doesn't make sense**	■ **Software Type: Multiprocessor Timesharing System** ■ **E.g. UNIX machine with several processors and several terminals**

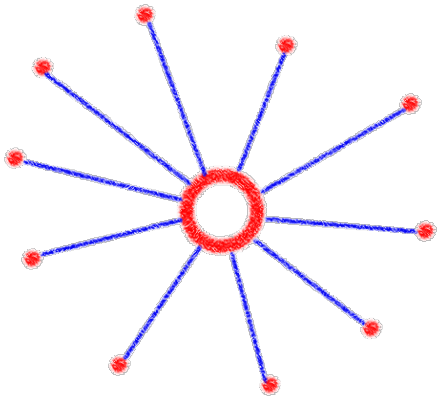
Lecture Outline

- What are Distributed Systems?
- Advantages and Disadvantages
- Design Issues
- Distributed Systems Hardware
- Distributed Systems Software
- Service Models
- Types of Distributed Systems
- Performance
- Programming Distributed Systems

Service Models

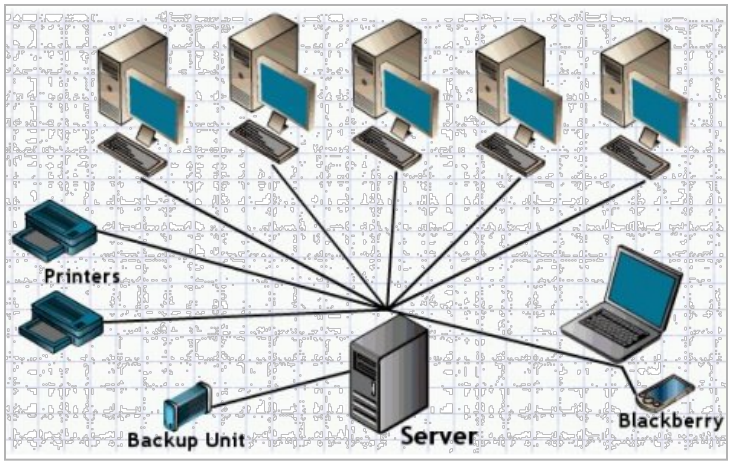
- Centralized model
- Client-server model
- Peer-to-peer model
- Thin and thick clients
- Multi-tier client-server architectures
- Processor-pool model

Service Models: Centralized



- **Application is hosted on one machine and user machines connect to it**
- **Example: Mainframes to which clients connect via a terminal**
- **Problems:**
  - Scaling is not easy.
    - There is limit on the number of CPUs in a system. Eventually it needs to be replaced
  - Multiple entities competing for the same resources

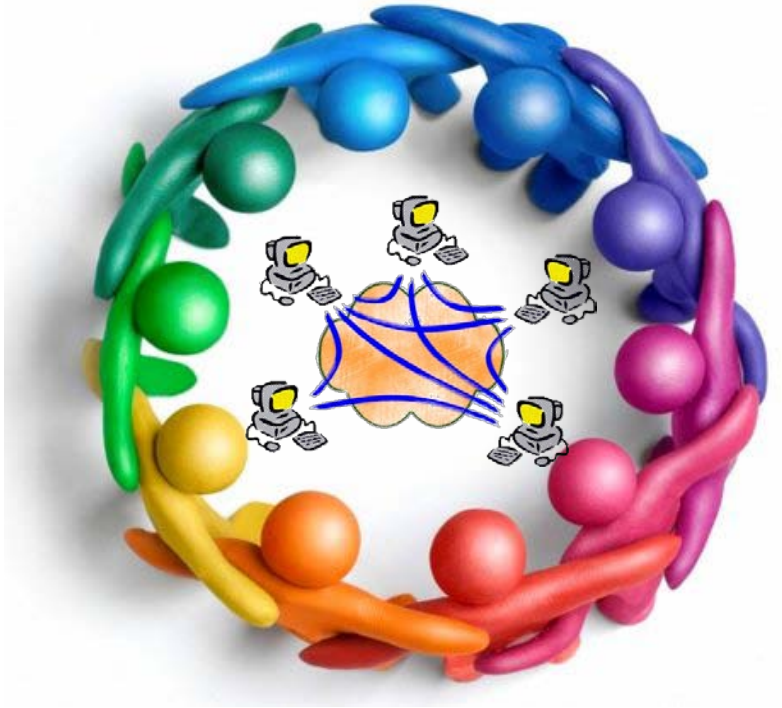
Service Models: Client-server



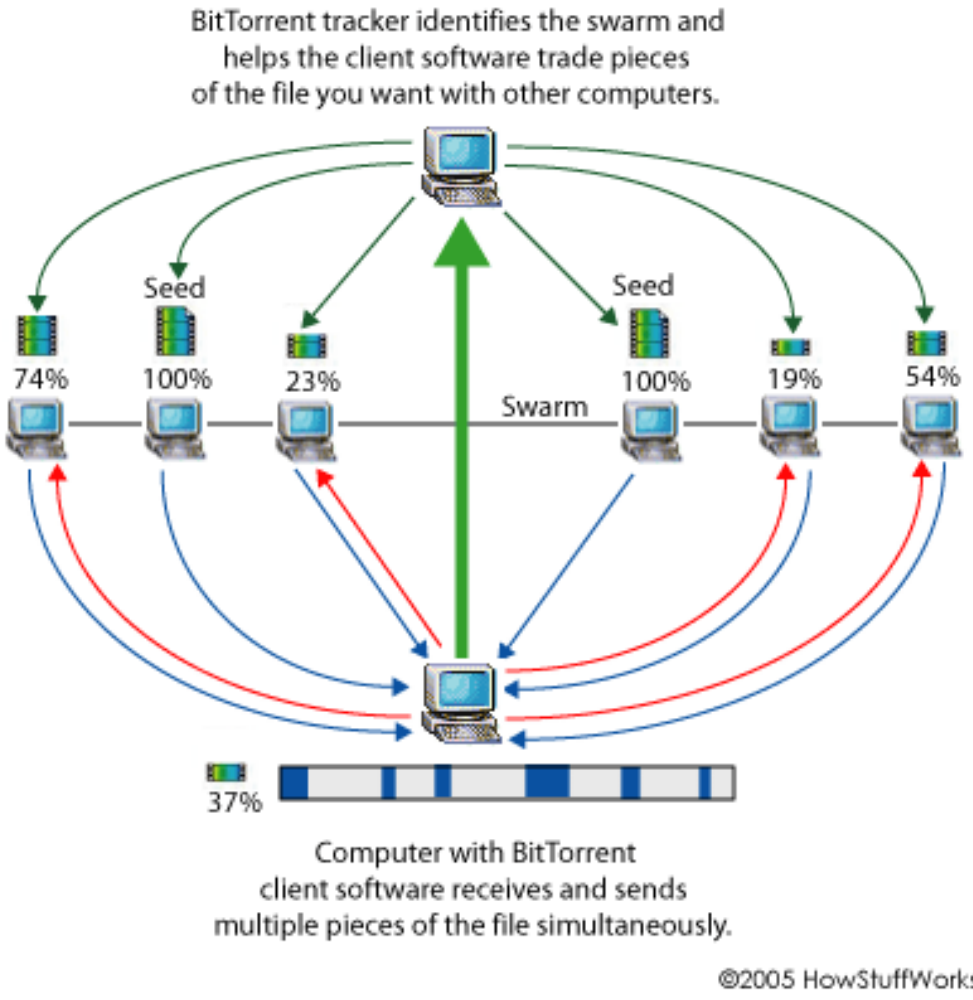
- **3 Components:**
  - **Service**
    - Per service, there exists:
    - **Server:** the service hosting machine
    - **Client:** requests the service
      - Could be a server for another service
- **Assumption:** certain machines are better suited for providing certain services
  - Eg. A file server would have a large amount of disk space and backup facilities
- **Example:** Workstation Model, Internet, SaaS etc.

Service Models: Peer-to-peer

- **Assumption:** machines have equivalent capabilities.
- No one machine is dedicated to provide special services for others
- **Example:**
  - File-Sharing among PCs in a LAN
  - **BitTorrent**, Napster
- **Active Research Area**



BitTorrent Example

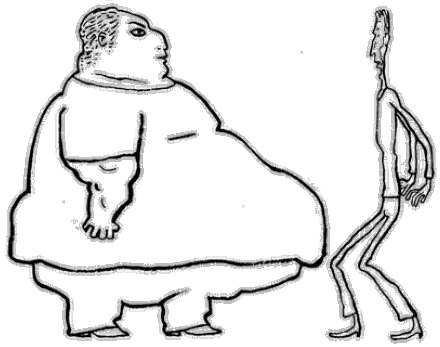


Service Models: Peer-to-peer

- **WWW**
  - A distributed system (of information).
  - An evolving system to publish and access resources and services on the Internet
  - CORBA is a good tool for creating a distributed system of programming objects

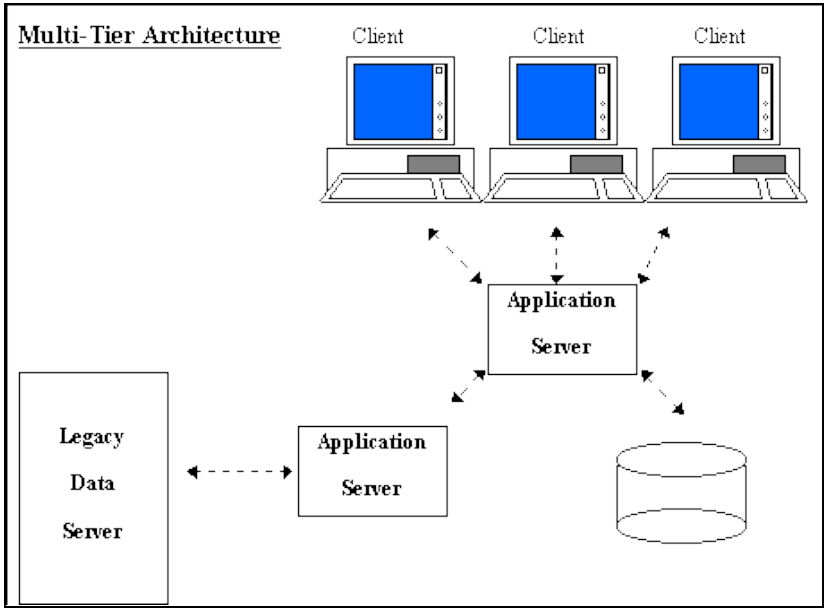
Service Models: Thin and thick clients

- **How is software partitioned between clients and server?**
- **What are the client's responsibilities?**
  - **Thin Client**
    - Client: small amount of client software
    - Servers: bulk of processing
    - No need for much administration, expansion slots, CDs, or even disks.
    - Like *Informed Appliance:* only needs connectivity to resource-rich networking
  - **Thick Client**
    - Client: bulk of data processing
    - Servers: services like web services, file storage
    - Needs: faster processors, high capacity storage devices and a lot of system configuration and administration.



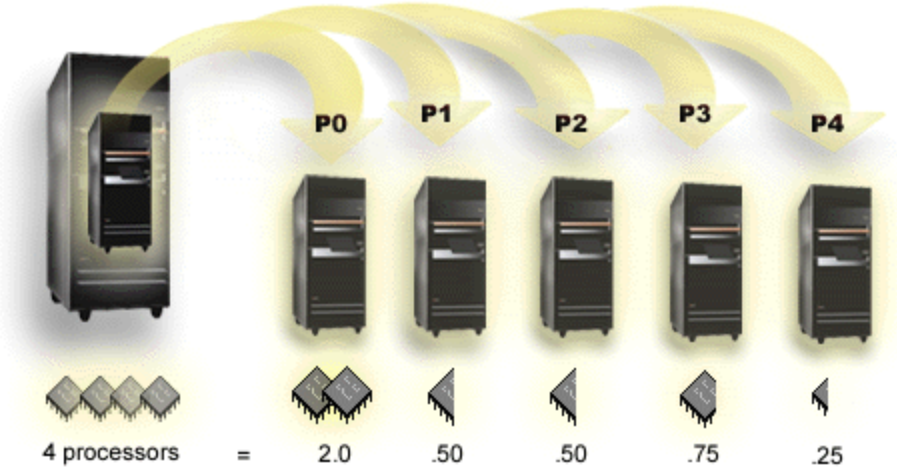
Service Models: Multi-tier client-server architectures

- There is hierarchy in connectivity.
- A server would contact other servers for services to accomplish it tasks.
- Two-tier architecture
- Three-tier architecture
- Example: Internet DNS



Service Models: Processor-pool

- **How about idle computing resources?**
  - Either you ignore …
  - Or you try to utilize using all of them to run jobs …
- **Processor-pool model: CPUs are dynamically assigned to processes on demand**
- **Example : Hadoop's JobTracker System distributes map and reduce tasks among a pool of processors**



Lecture Outline

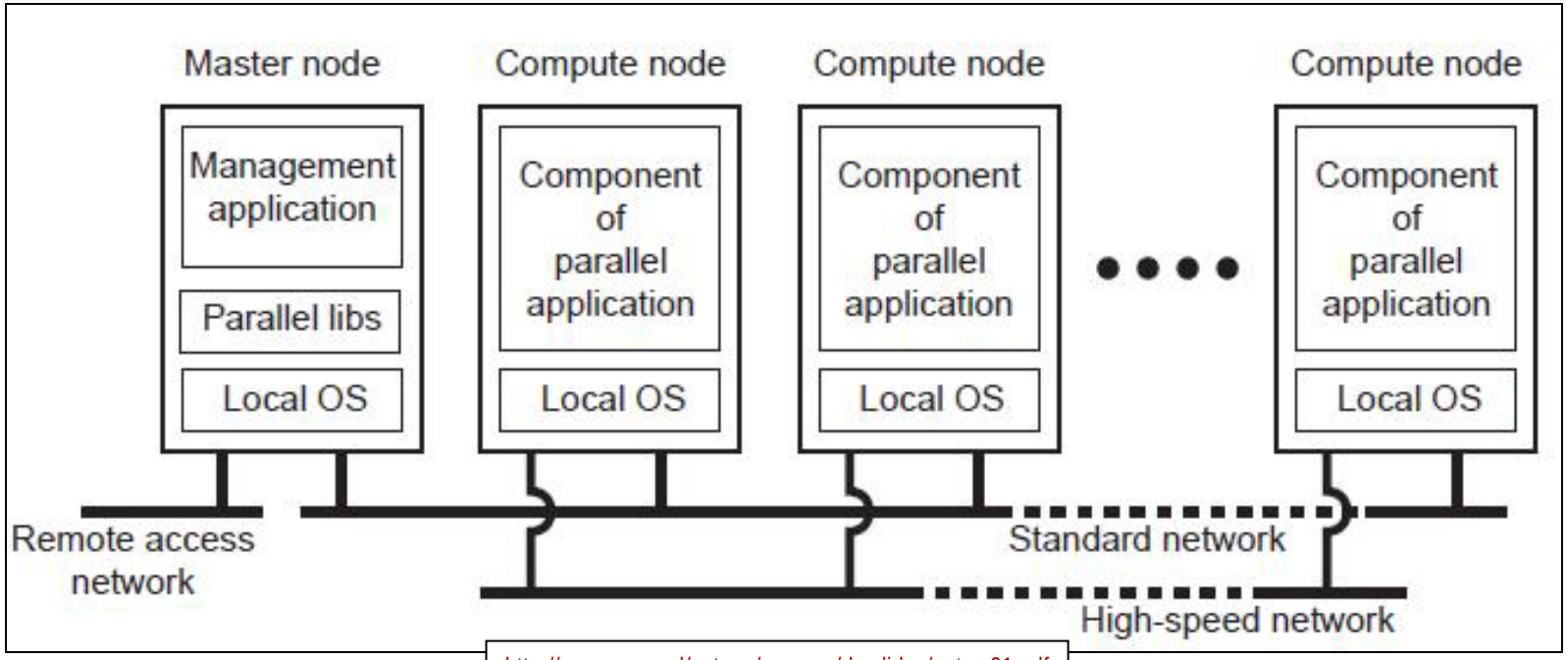
- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

Distributed System Types

- **Distributed Computing Systems**
  - Cluster Computing
  - Grid Computing
- **Distributed Information Systems**
  - Transaction Processing Systems
- **Distributed Pervasive Systems**
  - Eg: Smart Homes, Sensor-Networks etc.

Types: Distributed Computing Systems (1/3)

- **Configured for High-performance computing**



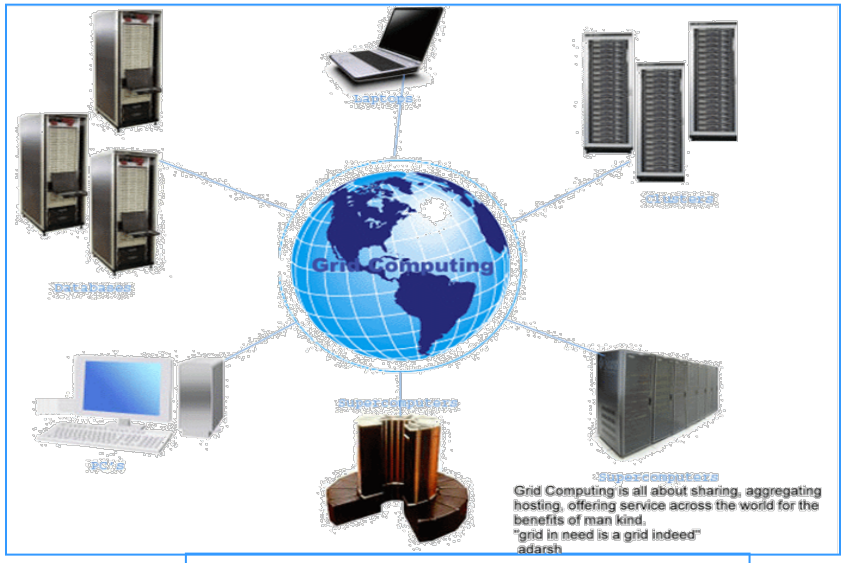
http://www.cs.vu.nl/~steen/courses/ds-slides/notes.01.pdf

Types: Distributed Computing Systems (2/3)

- **Cluster Computing**
  - **Collection of high-end computers (workstations/PCs) usually closely connected through a LAN**
  - **Homogeneous: Same OS, and similar Hardware**
  - **Brought to work together in a problem like a single computer**
    - More cost-effective than single computers with same speed and capabilities

Types: Distributed Computing Systems (3/3)

- **Grid Computing**
  - Clusters may be combined to form a "Grid“ of a massive computing power
  - Heterogeneous: systems differ in hardware/software/ administrative domains and deployed network technologies
  - Can easily span a WAN
  - For collaborations, grids use **virtual organizations**.

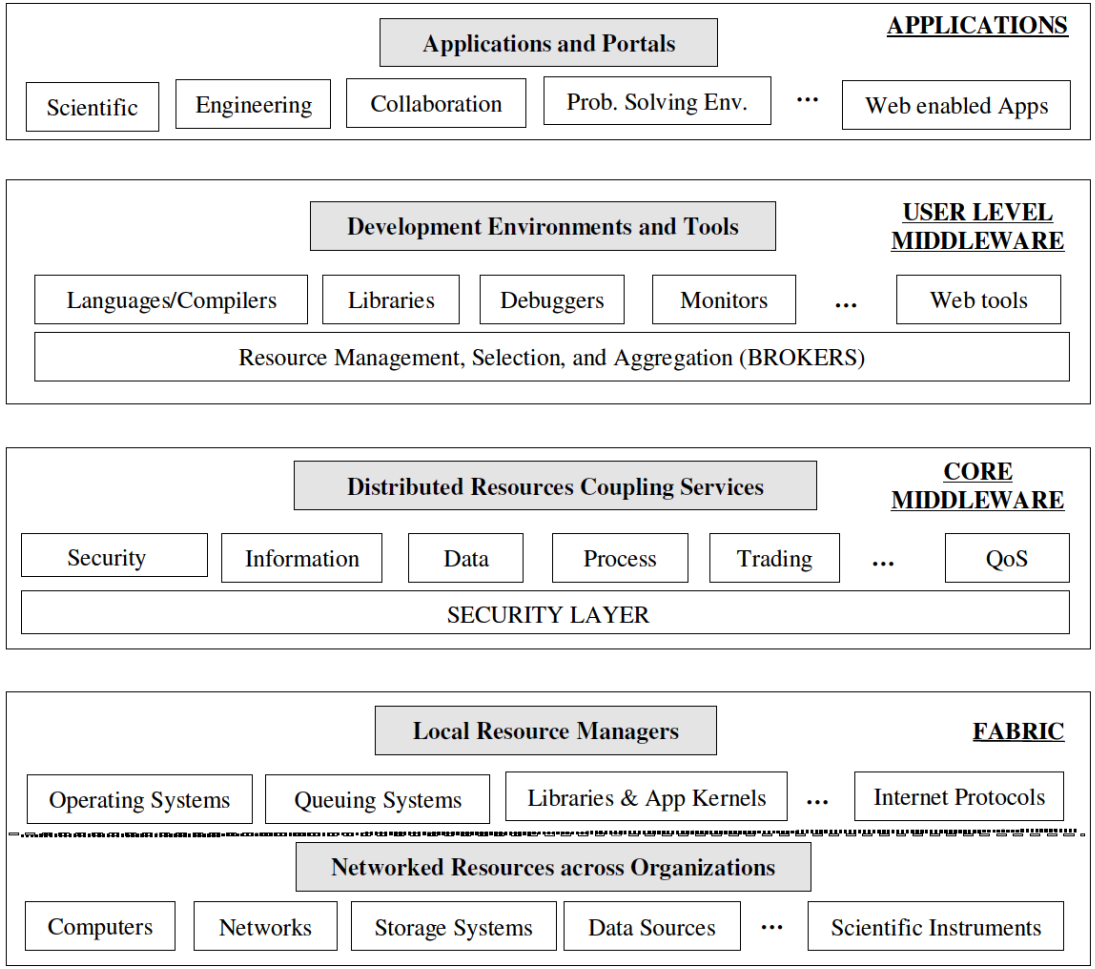


http://www.adarshpatil.com/grid/gridcomputing.gif

Grid Computing in Detail

- **Types of Grids**
  - **Computational Grid** – Shared Compute Resources
  - **Data Grid** – Access to Large amounts of Data spread across various sites
  - **Collaboration Grid** - multiple collaboration systems for collaborating on a common issue.
- **Grid Computing is an enabling technology and inspiration for Cloud Computing**
- **Applications / Domains**
  - Scientific Computing
  - Manufacturing
  - Financial services
  - Government.

Grid Components



Types: Distributed Information Systems (1/4)

- **Most distributed systems used today are forms of traditional Information systems**
- **Example: Transaction Processing Systems**
  - Data is processed using **transactions**
  - Fault-Tolerance, Data Integrity and Security is of extreme importance
  - Such systems are usually turnkey and cannot be used for general-purpose computing
  - Eg: Financial Processing, Airline Reservation systems.

Performance

- **Measures**
  - Response time
  - Jobs/hour
  - Quality of services
  - Balancing computer loads
  - System utilization
  - Fraction of network capacity
- **Messaging Delays affect:**
  - Transmission time
  - Protocol handling
- **Grain-size of computation**

Lecture Outline

- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

Performance: Measures

- **Responsiveness**
  - fast and consistent response to interactive applications
- **Throughput or Jobs/Hour**
  - It is the rate at which computational work is done
- **Quality of services**
  - The ability to meet qualities of users needs:
    - Meet the deadlines.
    - Provide different priority needs to different applications/ users/ data flows
    - guarantee a certain a performance level to a user/ application/ data flow

Performance: Measures

- **Balancing computer loads**
  - load balancing techniques:
  - Example: moving partially-completed work as the loads on hosts changes
- **System utilization (in %)**
- **Network-related Metrics**
  - Messaging Delay
  - Protocol Overhead
  - Round-Trip Time

Lecture Outline

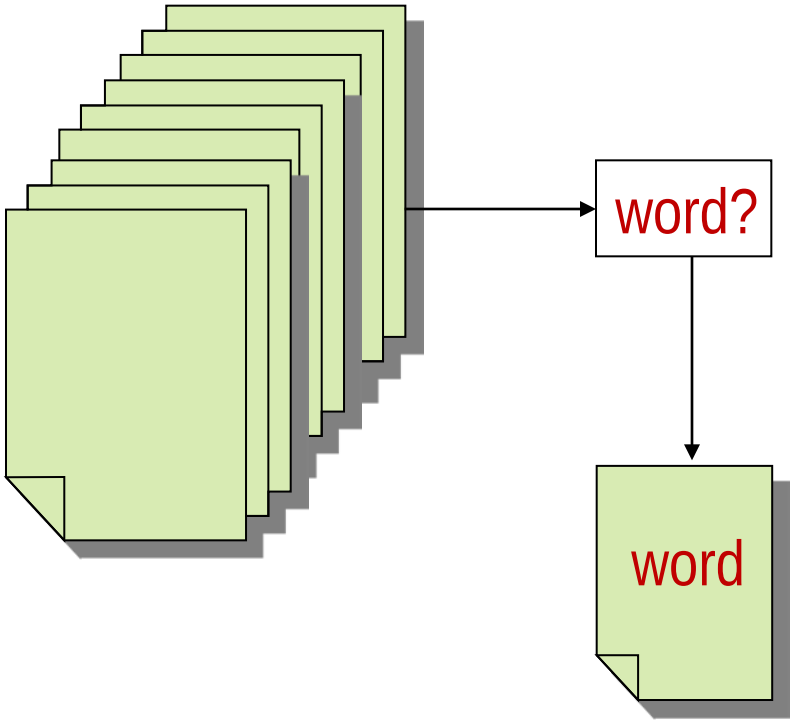
- What are Distributed Systems?
- Distributed vs. Parallel Systems
- Advantages and Disadvantages
- Distributed System Design
  - Hardware
  - Software
  - Service Models
- Distributed System Types
- Performance of Distributed Systems
- Programming Distributed Systems

Programming Distributed Systems

- Distributed systems tend to have distributed memory
- By far, the most common programming model is Message Passing
- The processes on a distributed system communicate by passing messages which can be either control or data messages.
- Message Passing Interface (MPI) is a standardized protocol and API for this type of model
- Popular programming interfaces are Fortran, C
- Implementations: MPI-Ch2, OpenMPI

Example with Message Passing

- Let's start with Parallel WordCount on Documents

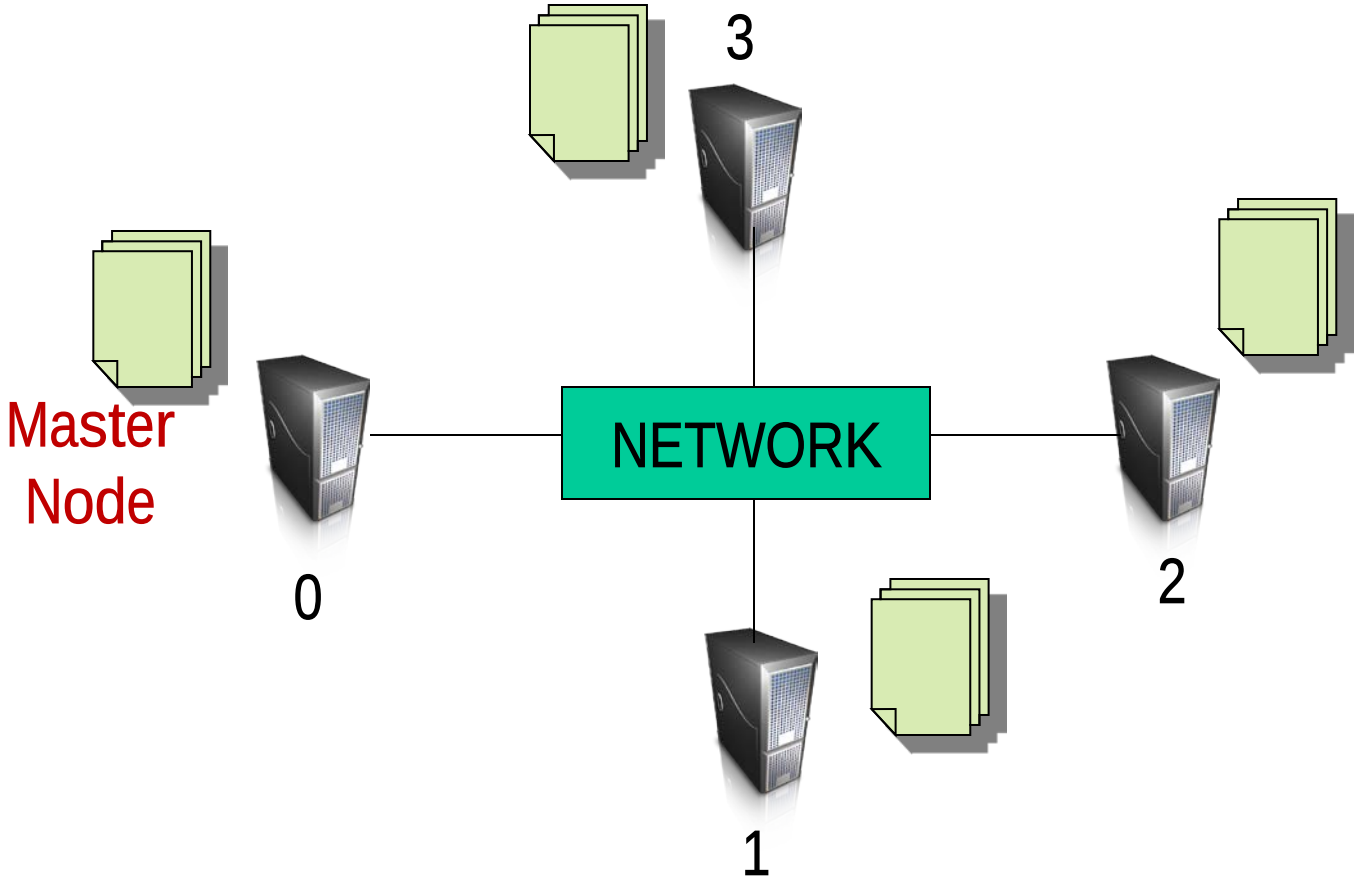


- We are looking for a word or pattern in a set of documents
- We could distribute the work among each processor in a cluster

Embarrasingly Parallel!

The Hardware:

- You are given the following cluster with 4 nodes:



- Distribute the documents over each node

Writing an MPI program

- The same program runs on each node
- We use conditional statements to figure out what the individual node has to do
- First we initialize the MPI and figure out the “rank” of each node in the cluster
- Then we perform operations associated with the rank.

Message Passing Pseudo code

Get my node rank (id)

Get total number of nodes (p)

```
For (i=id; I < total number of docs; i+=p)
    scan document
    if (word is present in document)
        count++;
```

Node 0 does doc 0, 4, 8,12 …

Node 1 does doc 1, 5, 9, 13 …

Node 2 does doc 2, 6, 10, 14 …

Node 3 does doc 3, 7, 11, 15 …

Collect the "count" values from all the worker nodes and sum it up on the master node.

MPI Code (wordcount.c)

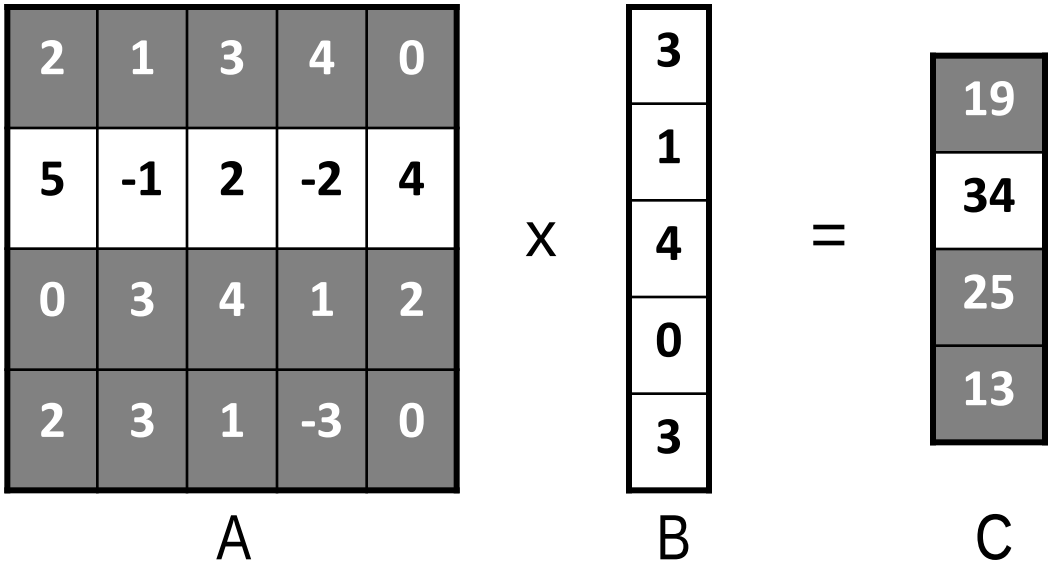
```
#include <mpi.h>

int main (int argc, char *argv[]) {
   int i,id,p, count, global_count;
   int check_doc (char[], char[]); //Function Declaration

   char filename[] = get_files(); //List of Files
   char pattern = “word”;
   count = 0;

   MPI_Init (&argc, &argv); //Initialize MPI Cluster
   MPI_Comm_rank (MPI_COMM_WORLD, &id); //Get my rank
   MPI_Comm_size (MPI_COMM_WORLD, &p);  //Get total number of nodes

   for (i = id; i < TOTAL_NUM_DOCS; i += p)
      count += check_doc (filename[i], pattern); //Function to check input

   //Reduce all the counts to global count on master node
   MPI_Reduce (&count, &global_count, 1, MPI_INT, MPI_SUM, 0,
               MPI_COMM_WORLD);

   if (id==0) printf (“The pattern appears %d times\n", global_count);

   MPI_Finalize();
   return 0;
}
```

Running the MPI Code


```
% mpirun -np 3 wordcount

The pattern appears 456 times
```

Another Example: Matrix-Vector Multiply

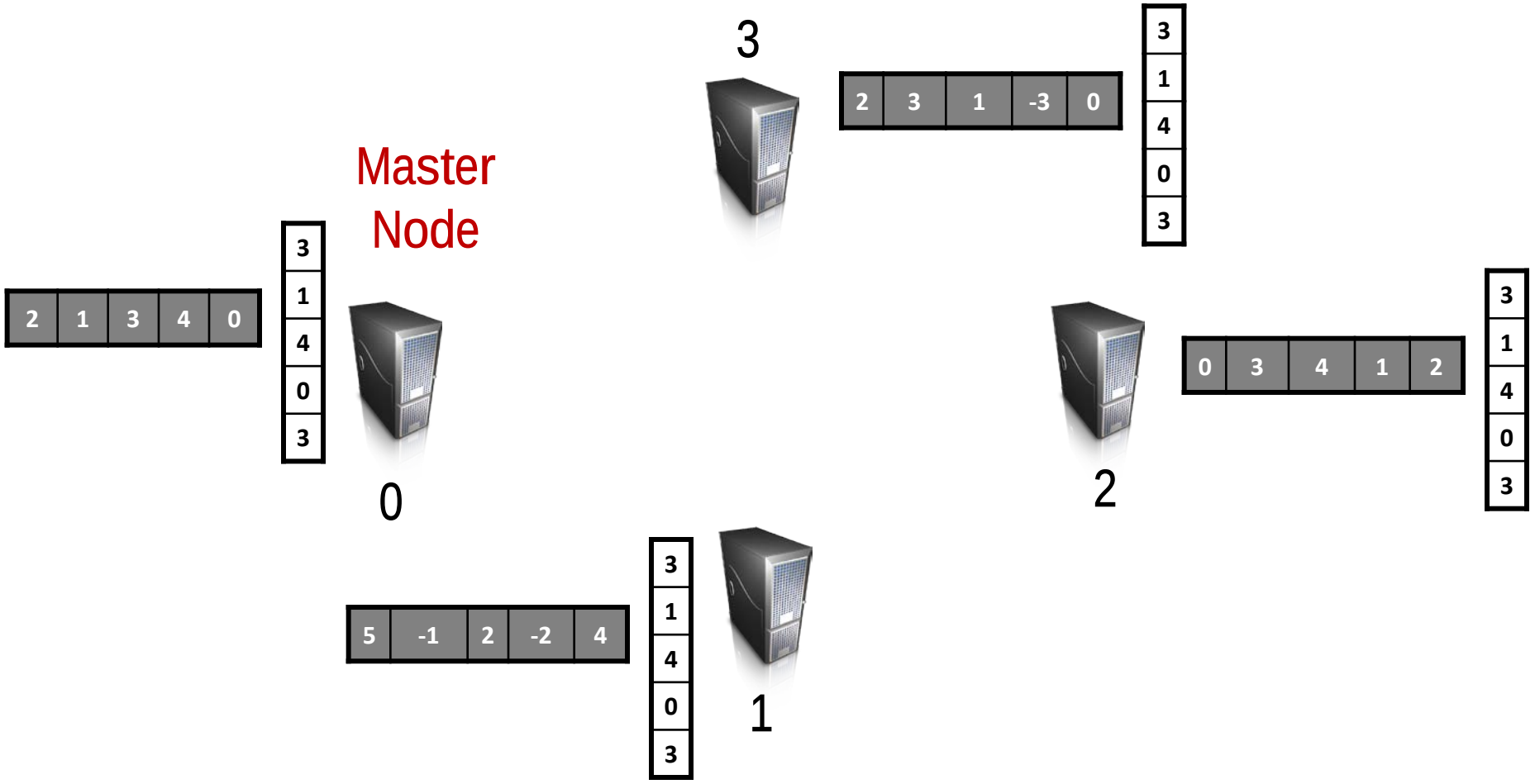
- Multiply a x*y Matrix with vector sized x



- Each row of the resultant vector can be computed independently and in parallel

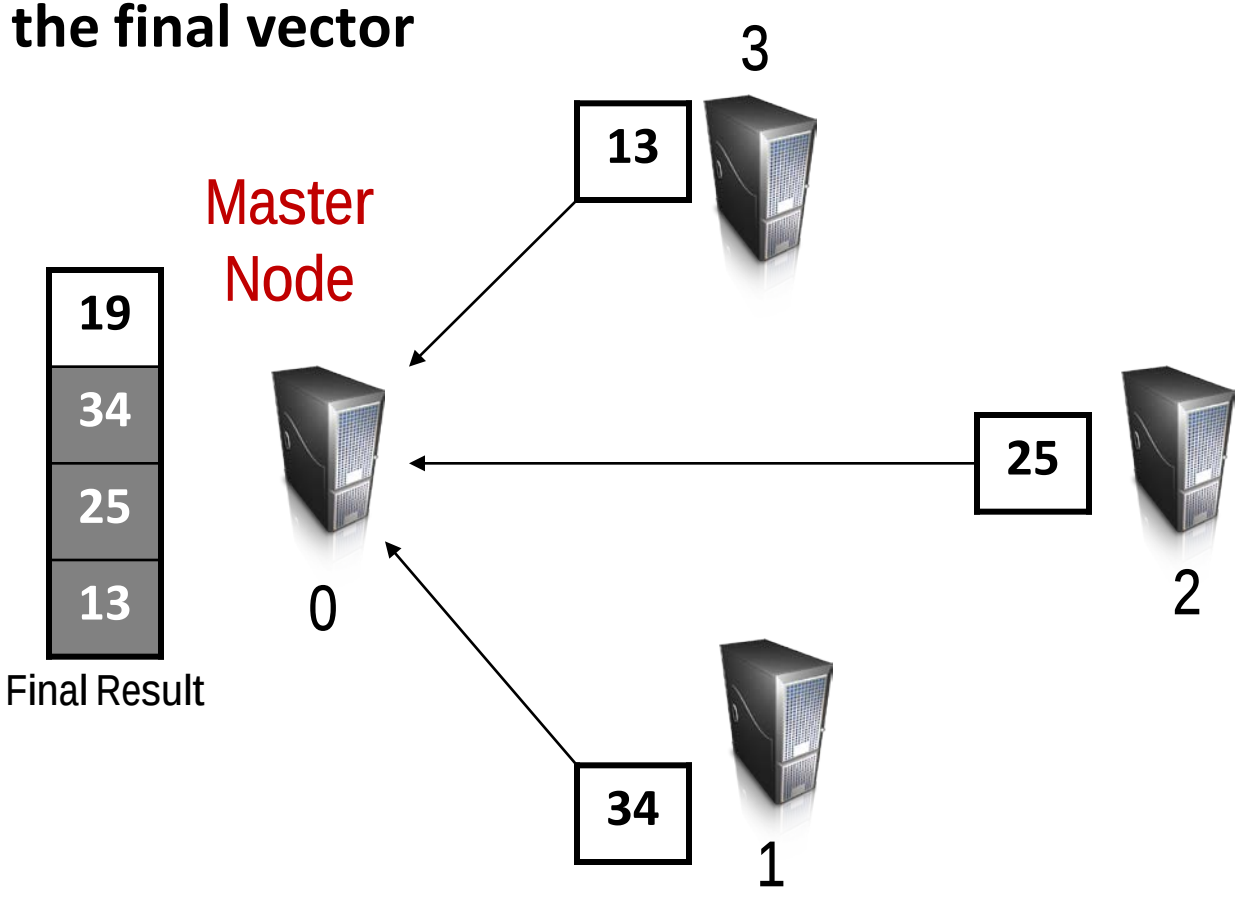
Decomposition

- Let's give each row and the entire vector to each machine



Aggregating the result

- Each node then passes it's result to the master node for the final vector



Program Pseudocode

Get my node rank (id)

Get total number of nodes (p)

If (master) send every node one row and vector

If (worker) Collect row array A and vector B

For (i=0; I < row_size; i++)

    C[i] += A[i] * B[i]

If (worker) Send results to Master

If (master) Collect results from each worker, output result

MPI Code (m_v_multiply.c)

```
#include <stdio.h>
#include <mpi.h>
#include "../MyMPI.h"

int main (int argc, char *argv[])
{
    double **a;        /* First factor, a matrix */
    double *b;         /* Second factor, a vector */
    double *c_block;   /* Partial product vector */
    double *c;         /* Replicated product vector */
    double *storage;   /* Matrix elements stored here */
    int    i, j;      /* Loop indices */
    int    id;        /* Process ID number */
    int    m;         /* Rows in matrix */
    int    n;         /* Columns in matrix */
    int    nprime;    /* Elements in vector */
    int    p;         /* Number of processes */
    int    rows;      /* Number of rows on this process */
```

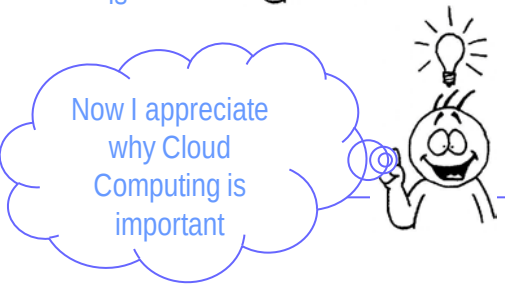
MPI Code (m_v_multiply.c)

```
/* Initialize MPI Environment */
MPI_Init (&argc, &argv);
MPI_Comm_rank (MPI_COMM_WORLD, &id);
MPI_Comm_size (MPI_COMM_WORLD, &p);

/*Read and Distribute Matrix and Vector from Command Line*/
read_row_striped_matrix (argv[1], (void *) &a,(void *) &storage,
                         MPI_DOUBLE, &m, &n, MPI_COMM_WORLD);
rows = BLOCK_SIZE(id,p,m);
read_replicated_vector (argv[2], (void *) &b, MPI_DOUBLE, &nprime,
                         MPI_COMM_WORLD);

/*Read and Distribute Matrix and Vector from Command Line*/
c_block = (double *) malloc (rows * sizeof(double));
c = (double *) malloc (n * sizeof(double));

for (i = 0; i < rows; i++) /*Compute Local Product*/
{
   c_block[i] = 0.0;
   for (j = 0; j < n; j++)
      c_block[i] += a[i][j] * b[j];
}
/*Collect Results*/
replicate_block_vector (c_block, n, (void *) c, MPI_DOUBLE,
                         MPI_COMM_WORLD);
MPI_Finalize();
}
```

Closing Notes

- **Link between Distributed Systems and Cloud Computing**
- **Lessons Learned in Building Large-Scale, Distributed Systems for Cloud**
  - Similar Goals, Requirements
- **Cloud Computing combines the best of these technologies and techniques**

Do you … ?



End of week two

Parallel Processing

End of week five

Distributed Systems

Time

References

- *www.cis.upenn.edu/~lee/00cse380/lectures/ln13-ds.ppt*
- http://code.google.com/edu/parallel/dsd-tutorial.html
- http://www.ida.liu.se/~TDDD25/lecture-notes/lect1.frm.pdf
- http://www.cs.rutgers.edu/~pxk/rutgers/notes/content/01-intro.pdf
- http://www.computing.dcu.ie/~kpodesta/distributed/
- http://www2.cs.uregina.ca/~hamilton/courses/430/notes/notes1.htm
- http://www.cs.vu.nl/~steen/courses/ds-slides/notes.01.pdf
- http://en.wikipedia.org/wiki/Middleware#Types_of_middleware
- *www.idi.ntnu.no/~conradi/dif8914/p1a-coulouris-ch1-2.ppt*