Distributed Computing (CS 515)

Dr. Ihsan Ullah

Department of Computer Science & IT
University of Balochistan, Quetta
Pakistan

July 5, 2013
Course objectives

- Getting an overview of distributed computing
- Understanding of hot research areas
- Current research challenges
- Future research trends
Marks distribution

- 30% Midterm
- 10% Assignments, Quizes
- 20% Case study
- 40% Terminal
Distributed Computing (CS 515)

Reference Book:


2. Research papers that will be provided and self-searched
Outline

1. Introduction
2. Technologies for network-based systems
3. System models
4. Software environments for distributed systems
5. Parallel and distributed programming models
6. Performance and scalability analysis
Outline

1. Introduction
2. Technologies for network-based systems
3. System models
4. Software environments for distributed systems
5. Parallel and distributed programming models
6. Performance and scalability analysis
Centralized computing

- Computer resources (processors, memory, and storage) are fully shared and tightly coupled within one integrated OS
- All computer resources are centralized in one physical system
- Scalability and cost issues under heavy load of user requests at global scale
Internet usage

- Users: **16 million** in Dec. 1995, **2749 million** in Mar. 2013
- Population: **0.40%** in Dec. 1995, **38.80%** in Mar. 2013
Parallel computing

- All processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory
- Also called parallel processing
- Interprocessor communication is accomplished through shared memory or via message passing
- A computer system capable of parallel computing is commonly known as a parallel computer
- Programs running in a parallel computer are called parallel programs
Distributed computing

- A field of computer science/engineering that studies distributed systems
- A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network
- Information exchange in a distributed system is accomplished through message passing
Definitions

- A distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved [KS08]
- A collection of independent computers that appears to the users of the system as a single coherent computer [TS03]
- A term that describes a wide range of computers, from weakly coupled systems such as wide-area networks, to strongly coupled systems such as local area networks, to very strongly coupled systems such as multiprocessor systems [Gos91]
Concurrent computing/programming

- Refer to the union of parallel computing and distributing computing
- Biased practitioners may interpret them differently
- **Ubiquitous computing** refers to computing with pervasive devices at any place and time using wired or wireless communication
- **The Internet of Things (IoT)** is a networked connection of everyday objects including computers, sensors, humans, ...
- The IoT is supported by Internet clouds to achieve ubiquitous computing with any object at any place and time
- The term **Internet computing** is even broader and covers all computing paradigms over the Internet
Outline

1. Introduction
2. Technologies for network-based systems
3. System models
4. Software environments for distributed systems
5. Parallel and distributed programming models
6. Performance and scalability analysis
Advances in CPU technology

- Multicore architecture with dual, quad, six, or more processing cores
- Parallelism at ILP and TLP levels
- Intel Core i7 990x reports 159,000 MIPS execution rate
Advances in Ethernet bandwidth

- 10 Mbps in 1979 – 1 Gbps in 1999, and 40/100 GE in 2011
- High-bandwidth networking increases the capability of building massively distributed systems
Advances in memory technology

- Rapid growth of flash memory and solid-state drives (SSDs)
- Tape units are dead, disks are tape units, flashes are disks, and memory are caches now [Jim Gray]
Virtual Machines

- A single OS offers a rigid architecture that tightly couples application software to a specific hardware platform.
- Some softwares that run well on one machine with an OS may not run on another machine with a different OS.
- **Virtual machines (VMs)** offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.
- To build large clusters, grids, and clouds, we need to access large amounts of computing, storage, and networking resources in a virtualized manner.
- Aggregation of resources to offer a single system image.
- A cloud of provisioned resources must rely on virtualization of processors, memory, and I/O facilities dynamically.
Virtual Machines

- Virtual Machine Monitor (VMM) is a middleware between VMs and the host platform.
- A VM can be implemented in privileged, non-privileged and dual mode.
- Offers hardware independence of the OS and applications.
VM Primitive Operations

- VMs can be multiplexed between hardware machines – (a)
- A VM can be suspended and stored in stable storage – (b)
- A suspended VM can be resumed or provisioned to a new hardware platform – (c)
- A VM can be migrated from one hardware platform to another – (d)
Outline

1. Introduction
2. Technologies for network-based systems
3. System models
4. Software environments for distributed systems
5. Parallel and distributed programming models
6. Performance and scalability analysis
Massive systems

Highly scalable, and can reach web-scale connectivity, either physically or logically

- Clusters
- Computing grids
- P2P networks
- Internet clouds
Clusters

- Interconnected stand-alone computers which work cooperatively as a single integrated computing resource
- Can handle heavy workloads with large data sets
- A cluster can be as simple as a Storage Area Network (SAN)
Clusters

- For a larger cluster with more nodes, the interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches.
- Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes.
- The cluster is connected to the Internet via a virtual private network (VPN) gateway.
- The gateway IP address locates the cluster.
- Most clusters have loosely coupled nodes each of them managed by its own OS.
Clusters

- An ideal cluster should merge multiple system images into a single-system image (SSI) [Pfi98]
- Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes
- SSI makes the cluster appear like a single machine to the user
- A cluster-wide OS for complete resource sharing is not available yet
- Middleware or OS extensions were developed at the user space to achieve SSI at selected functional levels
Computing grids

- A computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together [KS12]
- Often constructed across LAN, WAN, or Internet backbone networks at regional, national, or global scales
- The computers used in a grid are primarily workstations, servers, clusters, and supercomputers
- Integrates the computing, communication, contents, and transactions as rented services
- SETI (Search for Extraterrestrial Intelligence) @Home project
Grid vs. cluster

- A cluster is homogenous while grids are heterogeneous
- The computers that are part of a grid can run different operating systems and have different hardware whereas in a cluster, all computers have the same hardware and OS
- A grid can make use of spare computing power on a desktop computer while the machines in a cluster are dedicated to work as a single unit and nothing else
- Grid is Loosely coupled systems (diverse and decentralized) with distributed job management & scheduling
- Cluster is tightly coupled systems with a Single system image and centralized job management & scheduling
P2P systems

- End-hosts (peers) interconnect to form an overlay network
- Every node acts as both a client and server, providing part of the system resources
- Nodes are autonomous and can join and leave freely
- No peer machine has a global view of the entire P2P system
- The system is self-organizing with distributed control
P2P applications

- **File sharing**
  - Peers replicate copies of files and distribute them
  - BitTorrent, eMule

- **Text/video chatting**
  - Peers exchange text messages, voice, and videos
  - Skype

- **Streaming broadcast**
  - Live video streaming such as P2P IPTV and Video-on-Demand (VoD)
  - SopCast, PPStream
Cloud computing

- A cloud is a pool of virtualized computer resources
- Can host a variety of different workloads, including batch-style back-end jobs and interactive, user-facing applications
- A cloud allows workloads to be deployed and scaled out quickly through rapid provisioning of virtual or physical machines
- The idea of cloud is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers
Cloud service models

- **Infrastructure as a Service (IaaS)**
  - Puts together infrastructures demanded by users
  - Servers, storage, networks, and the data center fabric

- **Platform as a Service (PaaS)**
  - Enables the user to deploy user-built applications onto a virtualized cloud platform
  - Middleware, databases, development tools, and some runtime support such as Web 2.0 and Java

- **Software as a Service (SaaS)**
  - Refers to browser-initiated application softwares over thousands of paid cloud customers
  - Business processes, industry applications, enterprise resources planning (ERP), ...
Cloud deployment modes

- **Public clouds**
  - The cloud infrastructure is accessible to general public and shared in a pay as you go model of payment

- **Private clouds**
  - The cloud resources are not shared by unknown third parties
  - Resources may be located within the client organization premises or offsite

- **Hybrid clouds**
  - Combines different clouds for example the private and public clouds
  - The combined clouds retain their identities but are bound together by standardized technology

- **Community clouds**
  - The cloud infrastructure is shared by multiple organizations or institutions that have a shared concern or interest
Outline

1. Introduction
2. Technologies for network-based systems
3. System models
4. Software environments for distributed systems
5. Parallel and distributed programming models
6. Performance and scalability analysis
Service Oriented Architecture (SOA)

- A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains and implemented using various technology stacks
- A set of components which can be invoked and whose interface descriptions can be published and discovered (W3C)
- SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents
- A service is a unit of work done by a service provider to achieve desired end results for a service consumer
- Both provider and consumer are roles played by software agents on behalf of their owners [Pet06]
Distributed operating systems

- The computers in most distributed systems are loosely coupled.
- Thus, a distributed system inherently has multiple system images.
- This is mainly due to the fact that all node machines run with an independent operating system.
- To promote resource sharing and fast communication among node machines, it is best to have a distributed OS that manages all resources coherently and efficiently.
- Such a system is most likely to be a closed system, and it will likely rely on message passing and RPCs for internode communications.
MOSIX2 for Linux Clusters

- Runs with a virtualization layer in the Linux environment
- This layer provides a partial single-system image to user applications
- This is mainly due to the fact that all node machines run with an independent operating system
- Supports both sequential and parallel applications, and discovers resources and migrates software processes among Linux nodes
- Can manage a Linux cluster or a grid of multiple clusters
Outline

1. Introduction
2. Technologies for network-based systems
3. System models
4. Software environments for distributed systems
5. Parallel and distributed programming models
6. Performance and scalability analysis
Message-Passing Interface (MPI)

- This is the primary programming standard used to develop parallel and concurrent programs to run on a distributed system.
- MPI is essentially a library of subprograms that can be called from C or FORTRAN to write parallel programs running on a distributed system.
- The idea is to embody clusters, grid systems, and P2P systems with upgraded web services and utility computing applications.
MapReduce

- This is a web programming model for scalable data processing on large clusters over large data sets.
- The model is applied mainly in web-scale search and cloud computing applications.
- The master node specifies a Map function to divide the input into subproblems.
- Applies a Reduce function to merge all intermediate values with the same intermediate key.
- Highly scalable to explore high degrees of parallelism at different job levels.
- A typical MapReduce computation process can handle terabytes of data on tens of thousands or more client machines.
- Thousands of MapReduce jobs are executed on Google’s clusters every day.
Hadoop Library

- Offers a software platform that was originally developed by a Yahoo! group
- The package enables users to write and run applications over vast amounts of distributed data
- Users can easily scale Hadoop to store and process petabytes of data in the web space
- **Economical:** Comes with an open source version of MapReduce that minimizes overhead in task spawning and massive data communication
- **Efficient:** Processes data with a high degree of parallelism across a large number of commodity nodes
- **Reliable:** Automatically keeps multiple data copies to facilitate redeployment of computing tasks upon unexpected system failures
Outline

1. Introduction
2. Technologies for network-based systems
3. System models
4. Software environments for distributed systems
5. Parallel and distributed programming models
6. Performance and scalability analysis
Performance metrics

- **CPU speed**: Million Instructions Per Second (MIPS)
- **Network bandwidth**: Megabits Per Second (Mbps)
- **Distributed systems throughput**
  - Tera floating-point operations per second (Tflops)
  - Transactions per second (TPS)
  - Job response time and network latency
- **Low latency and high bandwidth interconnection is preferred**
- **System overhead** is often attributed to OS boot time, compile time, I/O data rate, and the runtime support system
- **QoS for Internet and web services**, system availability and dependability, security
Dimensions of Scalability

- The ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth [wikipedia]

- **Size scalability:** Achieving higher performance by increasing the machine size (processors, cache, memory, storage, ...)

- **Software scalability:** Upgrades in the OS or compilers, adding mathematical and engineering libraries, porting new application software, and installing more user-friendly programming environments

- **Application scalability:** Refers to matching problem size scalability with machine size scalability

- **Technology scalability:** This refers to a system that can adapt to changes in building technologies
Technology scalability

- When scaling a system design with new technology one must consider three aspects
  - **Time scalability**: Refers to the impact of new generation technology such as a processor on motherboard, power supply, ...
  - **Space scalability**: Space is related to packaging and energy concerns. Technology scalability demands harmony and portability among suppliers
  - **Heterogeneity**: Refers to the use of hardware components or software packages from different vendors. Heterogeneity may limit the scalability.
Amdahl’s Law

- Let us suppose a uniprocessor workstation executes a given program in time $T$ minutes.
- Now the same program is partitioned for parallel execution on a cluster of many nodes.
- We assume that a fraction $\alpha$ of the code must be executed sequentially. Therefore, $(1 - \alpha)$ of the code can be compiled for parallel execution by $n$ processors.
- The total execution time of the program is calculated by:

$$\alpha T + (1 - \alpha) \frac{T}{n}$$

where, the first term is the sequential execution time on a single processor and the second term is the parallel execution time on $n$ processing nodes.
Amdahl’s Law: Speedup factor

The speedup factor of using the n-processor system over the use of a single processor is expressed by:

\[
\text{Speedup} = S = \frac{T}{\alpha T + (1 - \alpha) T/n}
\]

\[
= \frac{1}{\alpha + (1 - \alpha)/n}
\]

- The maximum speedup of \( n \) is achieved only if the code is fully parallelizable with \( \alpha = 0 \)
- As the cluster becomes sufficiently large, that is, \( n \rightarrow \infty \), \( S \) approaches \( 1/\alpha \), an upper bound on the speedup \( S \)
- The sequential bottleneck is the portion of the code that cannot be parallelized
- If \( \alpha = 0.25 \rightarrow 1 - \alpha = 0.75 \), Max. speedup=4
Fixed workload system efficiency

- Amdahl’s law assumes the same amount of workload for both sequential and parallel execution of the program with a fixed problem size or data set (fixed-workload speedup).
- To execute a fixed workload on n processors, parallel processing may lead to a system efficiency:

\[ E = \frac{S}{n} = \frac{1}{\alpha n + 1 - \alpha} \]

- Low efficiency with large cluster size.
- For \( \alpha = 0.25 \) and \( n = 256 \), \( E = 1.5\% \).
- Low usage of resources: Majority of the processors remain idle.
Gustafson’s Law

- Scaling the problem size to match the cluster capability (scaled-workload speedup)
- Let $W$ be the workload in a given program. When using an $n$-processor system, the user scales the workload to

$$W' = \alpha W + (1 - \alpha) n W$$

- The parallel execution time of a scaled workload $W'$ on $n$ processors is defined by scaled-workload speedup

$$S' = \frac{W'}{W} = \frac{[\alpha W + (1 - \alpha) n W]}{W} = \alpha + (1 - \alpha) n$$

- Thus efficiency is $E' = S'/n = \alpha/n + (1 -\alpha)$
- For $\alpha = 0.25$ and $n = 256$, $E = 75\%$
Availability

- A system is highly available if it has a long mean time to failure (MTTF) and a short mean time to repair (MTTR)

\[
\text{System Availability} = \frac{MTTF}{(MTTF + MTTR)}
\]

- Failure may occur in hardware, software or network component

- Any failure that will pull down the operation of the entire system is called a single point of failure

- A reliable computing system must be designed with no single point of failure

- In general, as a distributed system increases in size, availability decreases due to a higher chance of failure and a difficulty in isolating the failures
References

A. Goscinski.
*Distributed Operating Systems: The Logical Design, Reading.*
Addison-Wesley, 1991.

Ajay D. Kshemkalyani and Mukesh Singhal.

Brijender Kahanwal and Tejinder Pal Singh.
The distributed computing paradigms: P2p, grid, cluster, cloud, and jungle.

Helmut Petritsch.
Service-oriented architecture (SOA) vs. component based architecture.

Gregory F. Pfister.
*In search of clusters (2nd ed.).*

A. Tanenbaum and M. Van Steen.
*Distributed Systems: Principles and Paradigms.*