Novateur Publication's
International Journal of Innovation in Engineering, Research and Technology [IJIERT]
ICITDCEME'15 Conference Proceedings
ISSN No - 2394-3696

THEORY OF DISTRIBUTED COMPUTING AND PARALLEL PROCESSING WITH ITS APPLICATIONS, ADVANTAGES AND DISADVANTAGES.

Virendra Dilip Thoke.
Department of Computer Technology, Adarsh Institute of Technology (Polytechnic)
Vita. Sangli, Maharashtra, India.

ABSTRACT
Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs .There are several autonomous computational entities, each of which has its own local memory.

- The entities communicate with each other by message passing.
- The system has to tolerate failures in individual computers.
- The structure of the system is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.

Parallel processing is simultaneous use of more than one CPU or processor core to execute a program or multiple computational threads. In practice, it is often difficult to divide a program in such a way that separate CPUs or cores can execute different portions without interfering with each other. Most computers have just one CPU, but some models have several, and multi-core processor chips are becoming the norm. There are even computers with thousands of CPUs.With single-CPU, single-core computers, it is possible to perform parallel processing by connecting the computers in a network. However, this type of parallel processing requires very sophisticated software called distributed processing software.
In parallel computing, all processors may have access to a shared memory to exchange information between processors.

- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.
- Distributed computing has to be less bandwidth intensive. Therefore, the tasks have to be fewer co-dependents.
- Parallel processing is faster and has higher bandwidth between nodes, but is harder to scale.

KEYWORDS: Distributed Computing, Parallel Computing, CPU

INTRODUCTION

In the past decade, the world has experienced one of the most exciting periods in computer development. Computer performance improvements have been dramatic - a trend that promises to continue for the next several years. One reason for the improved performance is the rapid advancement in microprocessor technology. Microprocessors have become smaller, denser and more powerful. The result is that microprocessor based supercomputing is rapidly becoming the technology of preference in attacking some of the most important problems of science and engineering. To exploit microprocessor technology, vendors have developed high parallel computers .Highly parallel systems of the enormous computational power needed for solving some of the most challenging computational problems such as circuit simulation incorporating various acts. Unfortunately, software development has not kept pace with hardware advances. New programming paradigms languages, scheduling and partitioning techniques, and algorithms are needed to fully exploit the power of these highly parallel machines.
A major new trend for scientific c problem solving is distributed computing. In distributed computing, computers are connected by a network are used collectively to solve a single larger problem. Many scientists are

discovering that their computational requirements are best served not by a single, monolithic computer but by a variety of distributed computing resources, linked by high speed networks. By parallel computing, we mean a set of processes that are able to work together to solve a computational problem. There are a few things that are worthwhile to point out. First, the use of parallel processing and the techniques that exploit them are now everywhere; from the personal computer to the fastest computer available. Second, parallel processing doesn't necessary imply high performance computing.

TRADITIONAL COMPUTERS AND THEIR LIMITATION

The traditional computer or conventional approach to computer design involves a single instruction stream. Instructions are processed sequentially and the result is movement of data from memory to functional unit and back to memory. As demands for faster performance increased, modifications were made to improve the design of the computers. It became evident that a number of factors were limiting potential speed: the switching speed of the devices, packaging and interconnection delays, and compromises in the design to account for realistic tolerances of parameters in the timing of individual components. Even if a dramatic improvement could be made in any of these areas, one factor still limits performance: the speed of light. Today's supercomputers have a cycle time on the order of nanoseconds. One nanosecond translates into the time it takes light to move about a foot (in practice, the speed of pulses through the wiring of a computer ranges from 0.3 to 0.9 feet per nanosecond). Faced by this fundamental limitation, computer designers have begun moving in the direction of parallelism.

PARALLEL PROCESSING AND DISTRIBUTEDCOMPUTING

PARALLEL PROCESSING

Traditionally software has been written for serial computation. Parallel computing is the simultaneous use of multiple compute resources to solve a computational problem.

CONCEPTS AND TERMINOLOGY:
WHY USE PARALLEL COMPUTING?

Saves time – wall clock time
Cost savings
overcoming memory constraints
it's the future of computing

FLYNN'S CLASSICAL TAXONOMY

Distinguishes multi-processor architecture by instruction and data
SISD – Single Instruction, Single Data
SIMD – Single Instruction, Multiple Data
MISD – Multiple Instruction, Single Data
MIMD – Multiple Instruction, Multiple Data
SIMD-All processing units execute the same instruction at any given clock cycle.Eachprocessing unit operates on a different data element.
MISD-Different instructions operated on a single data element. Very few practical usesfor this type of classification. Example: Multiple cryptography algorithms attempting to crack a single coded message.
MIMD-Can execute different instructions on different data elements. Most common type of parallel computer.

GENERAL TERMINOLOGY

Task – A logically discrete section of computational work
Parallel Task – Task that can be executed by multiple processors safely
Communications – Data exchange between parallel tasks
Synchronization – The coordination of parallel tasks in real time
Concepts and Terminology:
More Terminology
Granularity – The ratio of computation to communication

Coarse – High computation, low communication
Fine – Low computation, high communication
Parallel Overhead
Synchronizations
Data Communications
Overhead imposed by compilers, libraries, tools, operating systems, etc.

PARALLEL COMPUTER MEMORY ARCHITECTURES:
SHARED MEMORY ARCHITECTURE

All processors access all memory as a single global address space.
Data sharing is fast.
Lack of scalability between memory and CPUs

PARALLEL COMPUTER MEMORY ARCHITECTURES:
DISTRIBUTED MEMORY

Each processor has its own memory.
Scalable, no overhead for cache coherency.
Programmer is responsible for many details of communication between processors.

PARALLEL PROGRAMMING MODELS

Exist as an abstraction above hardware and memory architectures
Examples:
Shared Memory
Threads
Messaging Passing
Data Parallel

PARALLEL PROGRAMMING MODELS:
SHARED MEMORY MODEL

Appears to the user as a single shared memory, despite hardware implementations. Locks and semaphores may be used to control shared memory access. Program development can be simplified since there is no need to explicitly specify communication between tasks.

PARALLEL PROGRAMMING MODELS:
THREADS MODEL

A single process may have multiple, concurrent execution paths.
Typically used with a shared memory architecture.
Programmer is responsible for determining all parallelism.

PARALLEL PROGRAMMING MODELS:
MESSAGE PASSING MODEL

Tasks exchange data by sending and receiving messages.
Typically used with distributed memory architectures .
Data transfer requires cooperative operations to be performed by each process. Ex.- a send operation must have a receive operation.
MPI (Message Passing Interface) is the interface standard for message passing.

PARALLEL PROGRAMMING MODELS:
DATA PARALLEL MODEL

Tasks performing the same operations on a set of data.Each task working on a separate piece of the set.
Works well with either shared memory or distributed memory architectures.

DESIGNING PARALLEL PROGRAMS:
AUTOMATIC PARALLELIZATION

Automatic
Compiler analyzes code and identifies opportunities for parallelism
Analysis includes attempting to compute whether or not the parallelism actually improves performance.
Loops are the most frequent target for automatic parallelism.

DESIGNING PARALLEL PROGRAMS:
MANUAL PARALLELIZATION

Understand the problem
A Parallelizable Problem:
Calculate the potential energy for each of several thousand independent conformations of a molecule.
When done find the minimum energy conformation.
A Non-Parallelizable Problem:
The Fibonacci Series
All calculations are dependent

DESIGNING PARALLEL PROGRAMS:
DOMAIN DECOMPOSITION

Each task handles a portion of the data set.
Designing Parallel Programs:
Functional Decomposition
Each task performs a function of the overall work

DISTRIBUTED SYSTEM

Computer architectures consisting of interconnected, multiple processors are basically of two types:

- Tightly coupled system
- Loosely coupled system
- Tightly coupled systems

In these systems, there is a single system wide primary memory that is shared by all the processors . Usually tightly coupled systems are referred to as parallel processing systems. Loosely coupled systems In these systems, the processors do not share memory, and each processor has its own local memory .Loosely coupled systems are referred to as distributed computing systems, or simply distributed systems

DEFINITION

A distributed computing system is basically a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals, and the communication between any two processors of the system takes place by message passing over the communication network Models Used In Distributed Computing System

- Minicomputer Model:
- Workstation model:
- Workstation-Server Model:
- Processor-Pool model :
- Hybrid Model:

ADVANTAGES OF DISTRIBUTED COMPUTING SYSTEM

INHERENTLY DISTRIBUTED APPLICATIONS:
Several applications are inherently distributed in nature and require distributed computing system for their realization

INFORMATION SHARING AMONG DISTRIBUTED USERS:

In a distributed computing system, information generated by one of the users can be easily and efficiently shared by the users working at other nodes of the system .The use of distributed computing systems by a group of users to work cooperatively is known as computer-supported cooperative working (CSCW), or groupware

RESOURCE SHARING:

Information is not the only thing that can be shared in a distributed computing system. Sharing of software resources such as software libraries and databases as well as hardware resources such as printers, hard disks, and plotters can also be done in a very effective way among all the computers and the users of a single distributed computing system.

EXTENSIBILITY AND INCREMENTAL GROWTH:

It is possible to gradually extend the power and functionality of a distributed computing system by simply adding additional resources (both hardware and software) to the system as and when the need arises . Incremental growth is very attractive feature because for most existing and proposed applications it is practically impossible to predict future demands of the system. Extensibility is also easier in a distributed computing system because addition of new resources to an existing system can be performed without significant disruption of the normal functioning of the system.

SHORTER RESPONSE TIMES AND HIGHER THROUGHPUT:

The multiple processors of the distributed computing system can be utilized properly for providing shorter response times and higher throughput than a single processor centralized system. Another method often used in distributed computing systems for achieving better overall performance is to distribute the load more evenly among the multiple processors by moving the jobs from currently overloaded processors to lightly loaded ones

HIGHER RELIABILITY:

Reliability refers to the degree of tolerance against errors and component failures in a system. A reliable system prevents loss of information even in the event of component failures An important aspect of reliability is availability, which refers to the fraction of time for which a system is available for use.

BETTER FLEXIBILITY IN MEETING USER'S NEEDS:

Performing different types of computers are usually more suitable for different types of computation. A distributed computing system may have a pool of different types of computers, in which case the most appropriate one can be selected for processing a user's job depending on the nature of the job.

BETTER PRICE-PERFORMANCE RATIO:

With the rapidly increasing power and reduction in prize of the microprocessors, combined with the increasing speed of communication network, distributed computing systems potentially have a much better price performance ratio than a single large centralized system. More cost-effective then the centralized system is that they facilitate resource sharing among multiple computers.

PROPERTIES OF DISTRIBUTED SYSTEMS

So far the focus has been on designing a distributed system that solves a given problem. A complementary research problem is studying the properties of a given distributed system.

The halting problem is an analogous example from the field of centralized computation: we are given a computer program and the task is to decide whether it halts or runs forever. The halting problem is undesirable in the general case, and naturally understanding the behavior of a computer network is at least as hard as understanding the behavior of one computer.

However, there are many interesting special cases that are decidable. In particular, it is possible to reason about the behavior of a network of finite-state machines. One example is telling whether a given network of interacting (asynchronous and non-deterministic) finite-state machines can reach a deadlock. This problem is PSPACE-complete,[i.e., it is decidable, but it is not likely that there is an efficient (centralized, parallel or distributed) algorithm that solves the problem in the case of large networks.

DISTRIBUTED SYSTEM: - The word distributed in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area.[4] The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing.[5] While there is no single definition of a distributed system,[6] the following defining properties are commonly used:

- There are several autonomous computational entities, each of which has its own local memory.
- The entities communicate with each other by message passing.
- In this article, the computational entities are called computers or nodes.

A distributed system may have a common goal, such as solving a large computational problem.[9] Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.

PARALLEL PROCESSING: - Parallel processing is the ability to carry out multiple operations or tasks simultaneously. The term is used in the contexts of both human cognition, particularly in the ability of the brain to simultaneously process incoming stimuli, and in parallel computing by machines

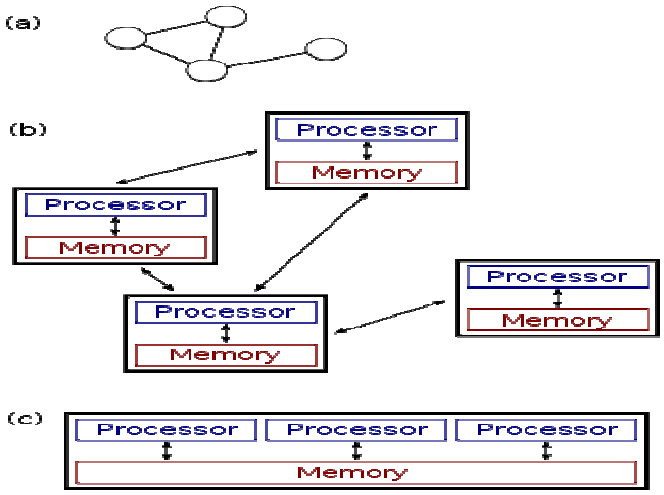

Figure 1: (a)–(b) A distributed system & (c) A parallel system.

Many tasks that we would like to automate by using a computer are of question–answer type: we would like to ask a question and the computer should produce an answer. In theoretical computer science, such tasks are called computational problems. Formally, a computational problem consists of instances together with a solution for each instance. Instances are questions that we can ask, and solutions are desired answers to these questions.

The field of concurrent and distributed computing studies similar questions in the case of either multiple computers, or a computer that executes a network of interacting processes: which computational problems can be solved in such a network and how efficiently? However, it is not at all obvious what is meant by "solving a problem" in the case of a concurrent or distributed system: for example, what is the task of the algorithm designer, and what is the concurrent or distributed equivalent of a sequential general-purpose computer.

The discussion below focuses on the case of multiple computers, although many of the issues are the same for concurrent processes running on a single computer.

Three viewpoints are commonly used:

PARALLEL ALGORITHMS IN SHARED-MEMORY MODEL

- All computers have access to a shared memory. The algorithm designer chooses the program executed by each computer.

- One theoretical model is the parallel random access machines (PRAM) that are used. However, the classical PRAM model assumes synchronous access to the shared memory.
- A model that is closer to the behavior of real-world multiprocessor machines and takes into account the use of machine instructions, such as Compare-and-swap (CAS), is that of asynchronous shared memory. There is a wide body of work on this model, a summary of which can be found in the literature.

PARALLEL ALGORITHMS IN MESSAGE-PASSING MODEL

- The algorithm designer chooses the structure of the network, as well as the program executed by each computer.
- Models such as Boolean circuits and sorting networks are used.[A Boolean circuit can be seen as a computer network: each gate is a computer that runs an extremely simple computer program. Similarly, a sorting network can be seen as a computer network: each comparator is a computer.

DISTRIBUTED ALGORITHMS IN MESSAGE-PASSING MODEL

- The algorithm designer only chooses the computer program. All computers run the same program. The system must work correctly regardless of the structure of the network.
- A commonly used model is a graph with one finite-state machine per node.
- In the case of distributed algorithms, computational problems are typically related to graphs. Often the graph that describes the structure of the computer network is the problem instance. This is illustrated in the following example.

COMPLEXITY MEASURES

In parallel algorithms, yet another resource in addition to time and space is the number of computers. Indeed, often there is a trade-off between the running time and the number of computers: the problem can be solved faster if there are more computers running in parallel (see speedup). If a decision problem can be solved in time by using a polynomial number of processors, then the problem is said to be in the class NC.[30] The class NC can be defined equally well by using the PRAM formalism or Boolean circuits – PRAM machines can simulate Boolean circuits efficiently and vice versa.

In the analysis of distributed algorithms, more attention is usually paid on communication operations than computational steps. Perhaps the simplest model of distributed computing is a synchronous system where all nodes operate in a lockstep fashion. During each communication round, all nodes in parallel

(1) Receive the latest messages from their neighbors,

(2) Perform arbitrary local computation, and

(3) Send new messages to their neighbors. In such systems, a central complexity measure is the number of synchronous communication rounds required to complete the task.

This complexity measure is closely related to the diameter of the network. Let D be the diameter of the network. On the one hand, any computable problem can be solved trivially in a synchronous distributed system in approximately 2D communication rounds: simply gather all information in one location (D rounds), solve the problem, and inform each node about the solution (D rounds).

On the other hand, if the running time of the algorithm is much smaller than D communication rounds, then the nodes in the network must produce their output without having the possibility to obtain information about distant parts of the network. In other words, the nodes must make globally consistent decisions based on information that is available in their local neighborhoods. Many distributed algorithms are known with the running time much smaller than D rounds, and understanding which problems can be solved by such algorithms is one of the central research questions of the field.

Other commonly used measures are the total number of bits transmitted in the network (cf. communication complexity).

OTHER PROBLEMS

Traditional computational problems take the perspective that we ask a question, a computer (or a distributed system) processes the question for a while, and then produces an answer and stops. However, there are also problems where we do not want the system to ever stop. Examples of such problems include the dining philosophers problem and

other similar mutual exclusion problems. In these problems, the distributed system is supposed to continuously coordinate the use of shared resources so that no conflicts or deadlocks occur.
There are also fundamental challenges that are unique to distributed computing. The first example is challenges that are related to fault-tolerance. Examples of related problems include consensus problems, Byzantine fault tolerance, and self-stabilization.

A lot of research is also focused on understanding the asynchronous nature of distributed systems:

- Synchronizers can be used to run synchronous algorithms in asynchronous systems.
- Logical clocks provide a causal happened-before ordering of events.
- Clock synchronization algorithms provide globally consistent physical time stamps.

DIFFERENCE & ADVANTAGES-DISADVANTAGES OF PARALLEL AND DISTRIBUTED SYSTEM

DIFFERENCES:

Distributed computing is a subset of parallel computing. Distributed computing is when you use more than one memory address space. In this situation variables are not shared between different threads of a program. The different threads often running on totally septate computers must send messages to each other in order to communicate
In the most simple form = Parallel Computing is a method where several individual (autonomous) systems (CPU's) work in tandem to resolve a common computing workload.
Distributed Computing is where several dis-associated systems are working seperatly to resolve a multi-faceted computing workload.
An example of Parallel computing would be two servers that share the workload of routing mail, managing connections to an accounting system or database, solving a mathematical problem, ect...
Distributed Computing would be more like the SETI Program, where each client works a separate "chunk" of information, and returns the completed package to a centralized resource that's responsible for managing the overall workload.
If you think of ten men pulling on a rope to lift a load, that is parallel computing . If ten men have ten ropes and are lifting ten different loads from one place to consolidate at another place, that would be distributed computing.
In Parallel Computing all processors have access to a shared memory. In distributed computing, each processor has its own private memory.

ADVANTAGES OF PARALLEL
SAVE TIME AND/OR MONEY: In theory, throwing more resources at a task will shorten its time to completion, with potential cost savings. Parallel clusters can be built from cheap, commodity components.
PROVIDE CONCURRENCY: A single compute resource can only do one thing at a time. Multiple computing resources can be doing many things simultaneously
USE OF NON-LOCAL RESOURCES: Using compute resources on a wide area network, or even the Internet when local compute resources are scarce.
LIMITS TO SERIAL COMPUTING: Both physical and practical reasons pose significant constraints to simply building ever faster serial computers:

- Transmission speeds - the speed of a serial computer is directly dependent upon how fast data can move through hardware. Absolute limits are the speed of light (30 cm/nanosecond) and the transmission limit of copper wire (9 cm/nanosecond). Increasing speeds necessitate increasing proximity of processing elements.
- Limits to miniaturization - processor technology is allowing an increasing number of transistors to be placed on a chip. However, even with molecular or atomic-level components, a limit will be reached on how small components can be.
- Economic limitations - it is increasingly expensive to make a single processor faster. Using a larger number of moderately fast commodity processors to achieve the same (or better) performance is less expensive.

DISADVANTAGE OF PARALLEL:

1. Costly
2. Replication
3. Complex

ADVANTAGES OF DISTRIBUTED COMPUTING

RELIABILITY
The important advantage of distributed computing system is reliability. It is more reliable than a single system. If one machine from system crashes, the rest of the computers remain unaffected and the system can survive as a whole.

INCREMENTAL GROWTH
In distributed computing the computer power can be added in small increments i.e. new machines can be added incrementally as per requirements on processing power grow.

SHARING OF RESOURCES
Shared data is required to many applications such as banking, reservation system and computer-supported cooperative work. As data or resources are shared in distributed system, it is essential for various applications.

FLEXIBILITY
As the system is very flexible, it is very easy to install, implement and debug new services. Each service is equally accessible to every client remote or local.

SPEED
A distributed computing system can have more computing power than a mainframe. Its speed makes it different than other systems.

OPEN SYSTEM
As it is open system, it can communicate with other systems at anytime. Because of an open system it has an advantage over self-contained system as well as closed system.

PERFORMANCE
It is yet another advantage of distributed computing system. The collection of processors in the system can provide higher performance than a centralized computer.

DISADVANTAGES OF DISTRIBUTED COMPUTING

TROUBLESHOOTING
Troubleshooting and diagnosing problems are the most important disadvantages of distributed computing system. The analysis may require connecting to remote nodes or checking communication between nodes.

SOFTWARE
Less software support is the main disadvantage of distributed computing system. Because of more software components that comprise a system there is a chance of error occurring.

NETWORKING
The underlying network in distributed computing system can cause several problems such as transmission problem, overloading, loss of messages. Hence, the problems created by network infrastructure are the disadvantages of distributed computing.

SECURITY

The easy distributed access in distributed computing system which increases the risk of security. The sharing of data creates the problem of data security.

APPLICATIONS

There are two main reasons for using distributed systems and distributed computing. First, the very nature of the application may require the use of a communication network that connects several computers. For example, data is produced in one physical location and it is needed in another location.

Second, there are many cases in which the use of a single computer would be possible in principle, but the use of a distributed system is beneficial for practical reasons. For example, it may be more cost-efficient to obtain the desired level of performance by using a cluster of several low-end computers, in comparison with a single high-end computer. A distributed system can be more reliable than a non-distributed system, as there is no single point of failure. Moreover, a distributed system may be easier to expand and manage than a monolithic uniprocessor system.

Examples of distributed systems and applications of distributed computing include the following:

- **TELECOMMUNICATION NETWORKS:**
  - Telephone networks and cellular networks.
  - Computer networks such as the Internet.
  - Wireless sensor networks.
  - Routing algorithms.

- **NETWORK APPLICATIONS**:
  - World Wide Web and peer-to-peer networks.
  - Massively multiplayer online games and virtual reality communities.
  - Distributed databases and distributed database management systems.
  - Network files systems.
  - Distributed information processing systems such as banking systems and airline reservation systems.

- **REAL-TIME PROCESS CONTROL:**
  - Aircraft control systems.
  - Industrial control systems.

- **PARALLEL COMPUTATION:**
  - Scientific computing, including cluster computing and grid computing and various volunteer computing projects; see the list of distributed computing projects.
  - Distributed rendering in computer graphics.

CONCLUSION

Parallel computing is fast .There are many different approaches and models of parallel computing. Parallel computing is the future of computing. And Distributed computing deals with hardware and software systems containing more than one processing element or storage element, concurrent processes, or multiple programs, running under a loosely or tightly controlled regime.

In distributed computing a program is split up into parts that run simultaneously on multiple computers communicating over a network. Distributed computing is a form of parallel computing, but parallel computing is most commonly used to describe program parts running simultaneously on multiple processors in the same computer. Both types of processing require dividing a program into parts that can run simultaneously, but distributed programs often must deal with heterogeneous environments, network links of varying latencies, and unpredictable failures in the network or the computers.

REFERENCES

[1]"Application-Specific Resource Provisioning for Wide-Area Distributed Computing" Xin Liu, Brookhaven National Laboratory ChunmingQiao, SUNY BuffaloDantong Yu, Brookhaven National Laboratory Tao Jiang, Huazhong University of Science and Technology
[2] "A Data Throughput Prediction and Optimization Service for Widely Distributed Many-Task Computing" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 6, JUNE 2011
[3]"Toward Efficient and Simplified Distributed Data Intensive Computing "YunhongGu and Robert Grossman IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 6, JUNE 2011
[4] "Energy Conscious Scheduling for Distributed Computing Systems under Different
Operating Conditions "Young Choon Lee, Member, IEEE, and Albert Y. Zomaya, Fellow, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 8, AUGUST 2011
[5]"Performance and Reliability of Non-Markovian Heterogeneous Distributed Computing Systems " Jorge E. Pezoa, Member, IEEE, and Majeed M. Hayat, Senior Member, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 23, NO. 7, JULY 2012
[6]"Utility Accrual Dynamic Routing in Real-Time Parallel Systems"Mehdi Kargahi, Member, IEEE, and Ali Movaghar, Senior Member, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 21, NO. 12, DECEMBER 2010
[7] Reference Book – "Distributed System Principles and paradigms" by Andrew S. Tanenbaum and Maarten Van Steen