Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290

Distributed Computing: An Overview

Md. Firoj Ali
Department of Computer Science, Aligarh Muslim University, Aligarh-02
Email: firojali.mca@gmail.com
Rafiqul Zaman Khan
Department of Computer Science, Aligarh Muslim University, Aligarh-02
Email: rzk32@yahoo.co.in

ABSTRACT

Decrease in hardware costs and advances in computer networking technologies have led to increased interest in the use of large-scale parallel and distributed computing systems. Distributed computing systems offer the potential for improved performance and resource sharing. In this paper we have made an overview on distributed computing. In this paper we studied the difference between parallel and distributed computing, terminologies used in distributed computing, task allocation in distributed computing and performance parameters in distributed computing system, parallel distributed algorithm models, and advantages of distributed computing and scope of distributed computing.

Keywords – **Distributed computing, execution time, heterogeneity, shared memory, throughput.**

Date of Submission: April 18, 2015 Date of Acceptance: June 08, 2015

1. Introduction

Distributed computing refers to two or more computers networked together sharing the same computing work. The objective of distributed computing is to sharing the job between multiple computers. Distributed network is mainly heterogeneous in nature in the sense that the processing nodes, network topology, communication medium, operating system etc. may be different in different network which are widely distributed over the globe [1, 2]. Presently several hundred computers are connected to build the distributed computing system [3, 4, 5, 6, 7, 8]. In order to get the maximum efficiency of a system the overall work load has to be distributed among the nodes over the network. So the issue of load balancing became popular due to the existence of distributed memory multiprocessor computing systems [3, 9]. In the network there will be some fast computing nodes and slow computing nodes. If we do not account the processing speed and communication speed (bandwidth), the performance of the overall system will be restricted by the slowest running node in the network [2, 3, 4, 5, 6, 7]. Thus load balancing strategies balance the loads across the nodes by preventing the nodes to be idle and the other nodes to be overwhelmed. Furthermore, load balancing strategies removes the idleness of any node at run time.

A distributed system can be categorized as a group of mostly autonomous nodes communicating over a communication network and having the following features [10]:

1.1 No Common Physical Clock

This plays an important role to introduce the element of "distribution" in a system and takes the responsibility to provide inherent asynchrony amongst the processors. In distributed network the nodes do not share common physical clock [10].

1.2 No Shared Memory

This is an important aspect of for message-passing communication among the nodes present in a network. There is no common physical clock concept in this memory architecture. But it is still possible to provide the abstraction of a common address space via the distributed shared memory abstraction [10, 11].

1.3 Geographical Separation

In distributed computing system the processors are geographically distributed even over the globe. However, it is not essential for the processors to be present on a wide-area network (WAN). It is possible to make a network/cluster of workstations (NOW/COW) present on a LAN can be considered as a small distributed system [10, 12]. Due to the low-cost high-speed off-the-shelf processor's availability NOW configuration becomes popular. The Google search engine is built on the NOW architecture.

1.4 Autonomy and Heterogeneity

The processors are autonomous in nature because they have independent memories, different configurations and are usually not part of a dedicated system connected through any network, but cooperate with one another by offering services or solving a problem together[10, 12].

2. Differences between Parallel and Distributed Computing

There are many similarities between parallel and distributed computing but there are some differences also exist that are very important in respect of computing, cost and time. Parallel computing actually subdivides an application into small enough tasks that can be executed at the concurrently while distributed computing divides an application into tasks that can be executed at different sites using the available networks connected together. In parallel computing multiple processing elements exist within one machine in which every processing element being dedicated to the overall system at the same time. But in distributed computing a group of separate nodes

Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290

possibly different in nature that each one contributes processing cycles to the overall system over a network.

Parallel computing needs expensive parallel hardware to coordinate many processors within the same machine but distributed computing uses already available individual machines which are cheap enough in today's market.

3. Terminologies Used in Distributed Computing

There are some basic terms used in distributed computing and ideas that will be defined first to understand the concept of distributed computing.

3.1 Job

A job is defined as the overall computing entity that's need to be executed to solve the problem at hand [11]. There are different types of jobs depending upon the nature of computation or algorithm itself. Some jobs are completely parallel in nature and some are partially parallel. Completely parallel jobs are known as embarrassingly parallel problem. In embarrassingly parallel problem communication among different entities is minimum but in case of partially parallel problem communication becomes high due to the communication among different processes running on different nodes to finish the job.

3.2 Granularity

Simply the size of tasks is expressed as the granularity of parallelism. The grain size of a parallel instruction is a measure of how much work each processor does compared to an elementary instruction execution time [11]. It is equal to the number of serial instructions done within a task by one processor. There are mainly three types of grain size exists: fine, medium and coarse grain.

3.3 Node

A node is an entity that is capable of executing the computing tasks. In traditional parallel system this refers mostly to a physical processor unit within the computer system. But in distributed computing a computer is generally considered as a computing node in a network [11]. But in reality trends have been changed. A computer may have more than one core like dual core or multi core processors. Both the terms node and processor have been used interchangeably in this literature.

3.4 Task

A task is a logically discrete part of the overall processing job. Each task is distributed over different processors or nodes connected through a network to work on each task to complete the job at the aim of minimized task idle time. In the literature, tasks are sometimes referred to as jobs and vice-versa [11].

3.5 Topology

The way of arranging the nodes in a network or the geometrical structure of a network is known as topology. Network topology is the most important part of the distributed computing. Actually topology defines how the nodes will contribute their computational power towards the tasks [11, 15].

3.6 Overheads

Overheads measure the frequency of communication among processors during execution. During the execution, processors communicate to each other for the completion of the job as early as possible, so obviously communication overheads take place. There are three types of overheads mainly bandwidth, latency and response time [11]. First two are mostly influenced by the network underlying the distributed computer system and the last one is the administrative time taken for the system to respond.

3.7 Bandwidth

It measures the amount of data that can be transferred over a communication channel in a finite period of time [11].It always plays a critical role for the system efficiency. Bandwidth is a crucial factor especially in case of fine grain problem where more communication takes place. The bandwidth is often far more critical than the speed of the processing nodes. The slow data rate obviously will restrict the speed of the processor and ultimately will cause poor performance efficiency.

3.8 Latency

It refers to the interval between an action being initiated and the action actually having some effect [11]. Latency specifies different meanings in different situations. Latency is the time between the data being sent and the data actually being received in case of underlying network called network latency. In case of task, latency is the time between a task being submitted to a node and the node actually begins the execution of the task called response time. Network latency is closely related with the bandwidth of the underlying network and both are critical to the performance of a distributed computing system. Response time and the network latency together are often called parallel overhead.

4. Performance Parameters in Distributed Computing

There are many performance parameters which are mostly used for measuring parallel computing performance. Some of them are listed as follows:

4.1 Execution Time

Execution time is defined as the time taken to complete an application after submission to a machine till finish. When the application is submitted to a serial computer, the execution time is called serial execution time and denoted by T_S and when application is submitted to a parallel computer, the execution time is called parallel execution time and denoted by T_P.

4.2 Throughput

It is defined as the number of jobs completed per unit time [11]. Throughput depends on the size of jobs. Throughput may be one process per hour for large process

Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290

while it may be twenty processes per seconds for small processes. It is fully dependent on the underlying architecture and the size of the running processes on that architecture.

4.3 Speed Up

Speed up of a parallel algorithm is the ratio of execution time when the algorithm is executed sequentially to the execution time when the same algorithm is executed by more than one processor in parallel. Speed up [11, 14] can be mathematically represented as: $S_p=T_s/T_p$, where T_s is the sequential execution time, T_p is the parallel execution time. In ideal situation, the speed up is equal to the number of processor in parallel but it is always less than the ideal one because the other important factors in a cluster like communication delay, memory access delay reduces the speed up.

4.4 Efficiency

It is the measure of the contribution by the processors to an algorithm in parallel. Efficiency [11, 14] can be measured as $E_p= S_p/p$ $(0>E_p<1)$ where S_p is the speed up and p is the number of processors in parallel. The Value of E_p is closure to 1 indicates an efficient algorithm.

4.5 System Utilization

This is a very important parameter. System utilization measures the involvement of resources present in a system. It may fluctuate between zero to 100 percent [11, 14].

4.6 Turnaround Time

It is defined as the time elapsed by the job from its submission to completion. Turnaround time is the summation of the time to get into memory, waiting in ready queue, executing on the processor and spending time for input/output operations [11, 14].

4.7 Waiting Time

I tis the total time spent by a processor waiting in ready queue for getting a resource. In other words, waiting time is the duration waited by a process to get the resource attention. Waiting time depends upon the parameters similar as turnaround time [11, 14, 15].

4.8 Response Time

Time between submission of requests and first response to the request is known as response time. This time can be restricted by the output devices of computing system [11, 14, 15].

4.9 Overheads

The overheads offered by a parallel program are expressed by a single function known as overhead function [11, 14, 15]. We denote the overhead function of a parallel system by the symbol *To*. The total over heads in solving a problem summed over all processing elements is pT_P. Therefore, the overhead function (To) is given by (1) where T_S time is free from overhead.

$$:T_o = pT_P - T_S \quad (1)$$

4.10 Reliability

Reliability ensures operations without fail under any specified conditions for a definite period of time [11, 14, 15].

5. Parallel Distributed Algorithm Models

In this section we have stated parallel distributed algorithm models. An algorithm model is classically a method of forming a parallel algorithm by picking proper decomposition and mapping technique and applying the appropriate strategy to minimize overheads [11, 14, 15].

5.1 The Data-Parallel Model

The data-parallel model is shown in Fig. 1 [11]. This is a simplest algorithm model. In this model, the tasks are generally statically mapped onto computing elements and each task does the similar operations on different data [16]. Data parallelism occurs as the processors operate similar operations but the operations may be executed in phases having different data. Uniform partitioning technique and static mapping are followed for load balancing as the processors operate on same [11, 14, 15]. Data-parallel algorithms [17, 18] follow either shared-address-space or message passing paradigms technique. However, message passing offers better performance for partitioned address space memory structure. Overheads can be minimized in the data-parallel model by selecting a locality preserving [17, 18]. The most attracting feature of data-parallel problems is that the degree of data parallelism grows with the size of the problem and can be effectively solve by adding more number of processors.

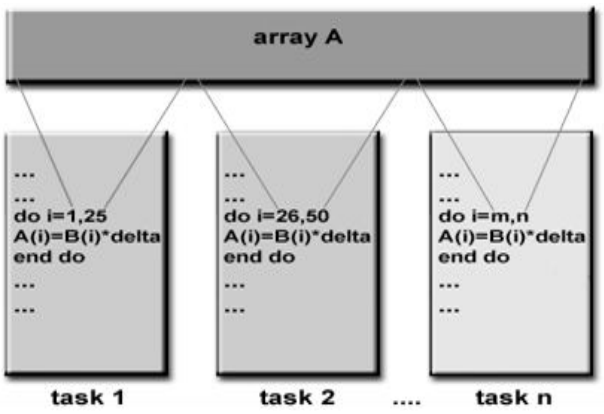

Figure 1 data parallel model

5.2 The Task Graph Model

Task dependency graph is an important way of representing the computations in any parallel algorithm [11, 14, 15].The task-dependency graph has two varieties: trivial and nontrivial. However, task dependency graph is also used in mapping of tasks on to the processors. This model is useful for solving problems which has the volume of data associated with the tasks islarge in comparison to the amount of computation associated with them. Generally, static mapping technique is used to

Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290

optimize the cost of data movement among tasks. Even a decentralized dynamic mapping uses the information about the task-dependency graph structure for minimizing interaction overhead [11, 14, 15].

5.3 The Work Pool Model

In this model the tasks may be assigned to any processor by a dynamic mapping technique for load balancing either by centralized or decentralized fashion [11, 14, 15].This model does not follow any pre-mapping scheme. The work already may be statically available before computation orcan be created dynamically. Whatever the process available or generated will be added to the global (possibly distributed) work pool. It is necessary to use termination detection algorithm for notifying the other processes to understand the completion of entire work when dynamic and decentralized mapping is used so that the processor can stop finding more jobs [11, 14, 15].

5.4 The Master-Slave Model

In this model generally one node is specially designated called master node and other nodes are called worker or slave nodes [15, 19, 20]. Simply master node generates the work and distributes the works to the worker nodes. This model does not have any absolute way of mapping and whatever work has been assigned to any worker will have to complete. The worker nodes do the necessary computations and the master node collects that result. The master node may allocate tasks to the worker nodes depending on the priori information about the worker nodes or on random basis which is more preferred approach. If the master node takes more time to generate works, the worker nodes can work in phases so that the next phase may start after the completion of previous phase [15, 19, 20].This model resembles to the hierarchical model in which the root nodes acts as master nodes and the leaf nodes acts as slave nodes. Both shared-address-space and message-passing paradigms are suitable for this model [15, 19, 20].

A large number of communications over heads generated at the master node may crash the whole system [15, 19, 20].Thus it is necessary to choose such granularity of tasks so that the system may have more dominance on computation rather than communication.

5.5 The Pipeline or Producer-Consumer Model

In this model, the data is passed through pipeline which has several stages and each stage (process) does some work on the data and passed to the next stage. This concurrent execution on a data stream by different programs is called stream parallelism[15, 19, 20, 21]. The pipelines may be in the form of linear or multidimensional arrays, trees or general graphs. A pipeline is a chain of producers and consumers because in this model each process generates result for next process. In general, static mapping is used in this model.

The larger granularity may take longer time to fill up the pipeline and the first process may take longer time to pass the data to the next step so the next process may have to wait longer and too fine granularity may cause more over heads so this model uses overlapping interaction with computation to reduce the overheads [11].

5.6 Hybrid Models

Sometimes, one or two models are combined to form hybrid model shown in Fig. 2 to solve the current problem in hand [11]. Many times, an algorithm design may need features of more than one algorithm model. For example, pipeline model is combined with a task dependency graph in which data passed through the pipeline model lead by the dependency graph [15, 19, 20].

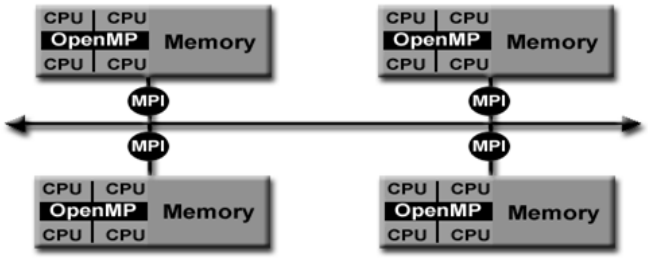

Figure 2 hybrid model

6. Advantages of Distributed Computing

Followings are the main advantages of distributed computing:

6.1 Inherently Distributed Computations

The applications which are distributed over the globe like money transfer in banking, reservations in flight journey which involves consensus among parties are inherently distributed in nature [10, 11, 15].

6.2 Resource Sharing

As the replication of resources at all the sites is neither cost-effective nor practical for performance improvement, the resources are distributed across the system. It is also impractical to place all the resources at a single site as it can degrade significant performance [10, 11, 15].For quick access as well as higher reliability distributed database like DB2 partition the data sets across a number of servers along with replication at a few sites [15].

6.3 Access to Geographically Remote Data and Resources

In many instances data cannot be replicated at each site due to its heavy size and it also may be risky to keep the vital data in each site [10, 11, 15]. For example, banking system's data cannot be replicated everywhere due to its sensitivity. So it is rather stored in central server which can be accessed by the branch offices through remote log in. Advances in mobile communication through which the central server can be accessed which needs distributed protocols and middleware [15].

Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290

6.4 Enhanced Reliability

Enhanced reliability is provided by the distributed system as it has inherent potential by replicating resources [10, 11, 15]. Further, in general the distributed resources do not crash or malfunction. Reliability involves several points:

6.4.1 Availability

The resources are always available and can be accessed any time.

6.4.2 Integrity

The resources or the data should always be in correct state as the data or resources are accessed concurrently by multiple processors.

6.4.3 Fault-Tolerance

Distributed system is fully fault tolerant because it works properly even some of its resources stop to work [11, 22].

6.5 Increased Performance/Cost Ratio

The performance/cost ratio is improved by resource sharing and accessing geographically remote data and resources [10, 11, 15].In fact, any job can be partitioned and can be distributed over numbers of computer in a distributed system rather than to allocate whole job to the parallel machines.

6.6 Scalability

More numbers of nodes may be connected to the wide-area network which does not directly affect in communication performance [10, 11, 15].

6.7 Modularity and Incremental Expandability

Heterogeneous processors running the same middleware algorithm may be simply included into the system without altering the performance and the existing nodes can be easily replaced by other nodes [10, 11, 15].

7. Scope of Distributed Computing

Distributed computing has changed the scenario of computation. Distributed computing is involved in almost every field of computation. The cost benefit analysis of distributed computing is always higher than the other dedicated computing [11]. Distributed computing is being highly applied in the fields such as engineering and design, scientific applications, commercial applications and applications in computer systems.

Distributing computing is widely being used to design applications like airfoils, internal combustion engines ,high-speed circuits, micro-electromechanical and nano-electromechanical systems in engineering and design. These types of applications need mainly optimization. Algorithms like Genetic programming for discrete optimization Branch-and-bound, Simplex, Interior Point Method for linear optimization which are being mostly used in optimization are parallelized and computed by distributed computing [10, 11, 15].

High performance computing is being highly used in scientific applications like sequencing of the human genome, examining biological sequences to develop new medicines and treatments for diseases, analyzing extremely large dataset in bioinformatics and astrophysics, understanding quantum phenomena and macromolecular structures in computational physics and chemistry [10, 11, 15].

Distributed computing is extensively being used in commercial applications. As the applications frequently used web and database servers, it is indispensible to make optimization for queering and taking quick decisions for better business processes. The huge volume of data and geographically distributed nature of this data need the use of effective parallel and distributed algorithms for the issues like classification, time-series analysis, association rule mining and clustering [10, 11, 15].

Since the computer systems become widespread in every field of computer science applications itself, the parallel distributed computing embedded in a diverse field of computer applications like computer security analysis, network intrusion detection, cryptography analysis, computations in ad-hoc mobile etc. [15].

8. Conclusion

This paper focuses on distributed computing. In this paper we studied the difference between parallel and distributed computing, terminologies used in distributed computing, task allocation in distributed computing and performance parameters in distributed computing system, parallel distributed algorithm models and advantages of distributed computing and scope of distributed computing.

References

[1] A Chhabra, G Singh, S S Waraich, B Sidhu and G Kumar, Qualitative Parametric Comparison of Load Balancing Algorithms in parallel and Distributed Computing Environment, Word Academy of Science, Engineering and Technology, 2006, 39-42.

[2] D Z Gu, L Yang and L R Welch, A Predictive, Decentralized Load Balancing Approach, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, Colorado, April 2005, 04-08.

[3] M F Ali and R Z Khan, The Study on Load Balancing Strategies in Distributed Computing System, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012.

[4] R Z Khan and M F Ali, An Efficient Diffusion Load Balancing Algorithm in Distributed System, I.J. Information Technology and Computer Science, Vol. 08, July 2014, 65-71.

[5] R Z Khan and M F Ali, An Efficient Local Hierarchical Load Balancing Algorithm (ELHLBA) in Distributed Computing, IJCSET, Vol 3, Issue 11, , November, 2013, 427-430.

Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290

[6] R Z Khan and M F Ali, An Improved Local Hierarchical Load Balancing Algorithm (ILHLBA) in Distributed Computing, International Journal of Advance Research in Science and Engineering (IJARSE), Vol. No.2, Issue No.11, November, 2013.
[7] M F Ali and R Z Khan, A New Distributed Load Balancing Algorithm, International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), vol: 2, Issue: 9, September 2014, 2556 – 2559.
[8] D L Eager, E D Lazowska and J Zahorjan, A Comparison of Receiver Initiated and Sender Initiated Adaptive Load Sharing, Performance Evaluation, Vol. 6, 1986, 53-68.
[9] S P Dandamudi and K C M Lo, Hierarchical Load Sharing Policies for Distributed Systems, Technical Report TR- 96-22, Proc. Int. Conf. Parallel and Distributed Computing Systems, 1996.
[10] A D Kshemkalyani and M. Singhal, Distributed Computing: Principles, Algorithms and Systems, Cambridge University Press, 2008.
[11] B Barney, Introduction to Parallel Computing, Retrieved from Lawrence Livermore National Laboratory: http://computing .IInl.govt/utorials/parallel comp/,2010.
[12] www.cs.uic.edu/~ajayk/chapter1.pdf
[13] M Nelson, Distributed Systems Topologies: Part 1, http://openp2p.com, 2001.
[14] I Ahmad, A Gafoor and G C Fox, Hierarchical Scheduling of Dynamic Parallel Computations on Hypercube Multicomputers, Journal of Parallel and Distributed Computing, 20, 19943, 17-329.
[15] A Grama, A Gupta, G Karypis and V Kumar, Introduction to Parallel Computing, Publisher: Addison Wesley, uJanuary 2003.
[16] K Cristoph and K Jorg, Modles for Paralel Computing: Review and Perspectives, PARS-Miteilungen 24, Dec. 2007, 13-29.
[17] P J Hatcher and M J Quinn, Data-Parallel Programming on MIMD Computers, MIT Press, Cambridge, MA, 1991.
[18] W D Hillis and G Steele, Data Parallel Algorithms, Communications of the ACM, Vol. 29, 1986.
[19] A Clamatis and A Corana, Performance Analysis of Task based Algorithms on Heterogeneous systems with message passing, In Proceedings Recent Advances in Parallel Virtual Machine and Message Passing Interface, 5th European PVM/MPI User's Group Meeting, Sept 1998.
[20] D Gelernter, M R Jourdenais and Kaminsky, Piranha Scheduling: Strategies and Their Implementation, International Journal of Parallel Programming, Feb 1995, 23(1): 5-33.
[21] S H Bokhari, Partitioning Problems in Parallel, Pipelined, and Distributed Computing, IEEE Transactions on Computers, January 1988, 37:8-57.
[22] www.cse.iitk.ac.in/report-repository, 2004.

***Dr. Rafiqul Zaman Khan*:** Dr. Rafiqul Zaman Khan is presently working as an Associate Professor in the Department of Computer Science in Aligarh Muslim University (A.M.U), Aligarh, India. He received his B.Sc. Degree from M.J.P Rohilkhand University, Bareilly, M.Sc and M.C.A from A.M.U. and PhD (Computer Science) from Jamia Hamdard University, New Delhi, India. He has 19 years of Teaching Experience of various reputed International and National Universities viz King Fahad University of Petroleum & Minerals (KFUPM), K.S.A, Ittihad University, U.A.E, Pune University, Jamia Hamdard University and AMU, Aligarh. He worked as a Head of the Department of Computer Science at Poona College, University of Pune. He also worked as a Chairman of the Department of Computer Science, AMU, Aligarh. His Research Interest includes Parallel & Distributed Computing, Gesture Recognition, Expert Systems and Artificial Intelligence.

Mr. Md Firoj Ali: Mr. Md Firoj Ali is presently working as an Assistant Engineer in WBSEDCL, India. He received his B.Sc. and MCA Degree from A.M.U. He has been awarded Senior Research Fellowship by UGC, India and also cleared National Eligibility Test conducted by UGC, 2012 and State Eligibility Test conducted by WBCSC, 2013. His Research Interest includes Load balancing in Distributed Computing System. He has published eleven research papers in International Journals/Conferences in the field of parallel and distributed computing.