Journal of Engineering, Science and Mathematics

Volume 01, Number 01, Pages 1-6, Year 2020
https://jesm.in/archives/

JESM

Review Paper

Utilization of Resources on Cloud Hosting Network in Secure Environment

Manik Rakhra[1], Neha Verma[2]

Department of Computer Science and Engineering, Lovely Professional University Phagwara, India

Correspondence should be addressed to Manik Rakhra; rakhramanik786@gmail.com

Received 14-11-2020; Accepted 01-12-2020; Published 02-12-2020

Handling Editor: Ashutosh Sharma

In this recommended framework that strengthened the adaptive hierarchical scheduling objectives to build such a program that can perform N amount of jobs on processors that can work much better and operate more effectively than current systems. In this paper, suggest a hybrid method for scheduling multi-computers in large-scale distributed memory, named dynamic scheduling, which illustrates the possible advantages of memory slicing and duration filtering while concurrently addressing their disadvantages. The effects of such a mix through comprehensive simulated studies where two workload models test the relative efficiency of the tree strategy along with appropriate space and time slicing techniques.

Keywords: *Hybrid, CPU, Distributed System*

1. Introduction

1.1 Load Sharing Policies

Transfer load sharing policy is used in load sharing. The transfer policy seems in the job line at the node when a new job comes at the node. The job is filled into the job transfer line if the job is fit for remote execution. The job is eligible for execution if the job is put in the queue [1]. Transferred job is enclosed in destination when it arrives. A hybrid policy is used by both the sender and the receiver. In this, we select a random node for execution. Each node has the information from where the job comes and also of the target node. The round robin is used to process the jobs that are placed in the queue. This method is employed to the distributions database and the centralized databases [2]. Many

databases applications spend their time waiting for the input output operation and execute only simple transactions. Concurrent processing of many of the jobs arises a situation of the bottle neck in main memory. If the operating system just has to create a process for every concurrent transaction the most of the process will be swapped so we apply specialized techniques of the database applications. In today's society various widely dispersed capital. For cloud services systems, optimization is the most important field. Resource management and time preparation is an environment that needs to be taken into account particularly if performance development is our motive for change. I/O are an exciting opportunity for several works. The majority of grid details would be electronic network. For efficient output, the distribution of the processor and I/O resources must be synchronized. For this function, a tree scheduling scheme is introduced to handle concurrent workers in order to control the background workload [3].

1.2 System and Work Load Model

In this proposed policy, a distributed system is used which comprises of nodes just like the processing in a grid computing system. In this distributed environment, our focus is on the efficiency that is the number of messages which are sending to different sites [4]. The homogenous nodes are considered because of the distributed environment. In this process, all the aspects of the execution of jobs along the nodes are interrelated to each other because the cost of the messages is not only derived from the transmission cost but also the set of the rules which the CPU has to execute. The data transfer can be more important than the messages which are controlled. Transfer of the jobs depends upon the application and the reduction of control messages to a bigger extent depend upon the mechanism provided by the system. A batch mechanism has been implemented in this scenario to evaluate the confidence interval based upon the comparison of the optimization of the performance of the global tree policy performed better than the local policy. The performance of the hierarchical policy is much better than the adaptive policy. The atomicity, durability, serializability and isolation properties of the transaction are employed in this to remove the deadlock condition and also to maintain the integrity. The availability of the system is maximized to a greater extent. The cost of the main memory, CPU, and their response time are minimized to obtain a correct response time is more important for the distributed application than the local application due to the delay which arises when communication is done with various sites. Another aspect which is considered in the distributed database is the availability of resources of the whole system. Failure of one of the sites in the distributed system does not stop the execution at the other site. The recovery is maintained in the failure by the transparency in the distributed database. If multiple transactions are executed in the same order by a different processor this process is called as a concurrency control. The efficient, reliable, concurrent execution of the transaction, these three goals are strongly interconnected with each other [5]. A session is made between the processes in which we want to communicate.

1.3 Terminologies in Time and Space Sharing Strategies

1) The users specify the ideas to the operating system directly by using and input device and wait for the output device to response. The reply time should be less than 1 second.
2) Multi-programming is logical extension of time sharing. In the time distribution system, the CPU run many jobs by merging jobs in between them, but merging of job occur so fast that the users can interact with every program when it is running.
3) As the system switches very fast to one user to another, each user is given the idea that the entire computer is

dedicated to his job only even though the computer is used by many users.

4) The CPU go to next job that can execute whenever the current job goes in waiting state or current job take a certain amount of time [6].
5) We get a view that CPU is running multiple jobs at the same time.
6) A time-shared operating system basically performs multiprogramming and scheduling of CPU to provide each user with a portion of time-shared computer.
7) Multiprogramming and time sharing require multiple jobs to be kept in memory at the same time. As the main memory is too small to handle all jobs the jobs are placed in the jobs pool or jobs queue.
8) The concept of paging is done i.e. swap in or swap out. This is how memory is shared between many user programs. This process is called as virtual memory. It means we dynamically allocate the memory.
9) Virtual memory is method that allows the execution of process that is not completely in memory but the dynamic memory. The main feature is that is allowing users to execute the program that are bigger than actual physical memory.
10) If multiple jobs are ready to be taken into the memory, and there is not enough memory than the system choose jobs from it. This is called as job scheduling.
11) The jobs loaded into the memory for execution when the operating system selects a job from the queue.
12) In process several jobs are ready to execute simultaneously. The system must select from them this is called CPU scheduling.
13) The large main memory separating a logical memory is viewed by the user as physical memory. Time sharing system provides a file system. The place of the file system is on the collection of disks.
14) To make execution in a serial order the system uses a method called as job synchronization. This is done so that deadlock does not occur. The sharing of resources is handling by the operating system so it should never be blocked from executing.

1.4 Problem in Job Scheduling

In computer science and operations job scheduling is an optimization problem. In job scheduling is an optimization problem. In job scheduling a term make span is used, makes span means when all the jobs have finished processing. Various job scheduling problem are

1.5 How to Assign the Task in Distributed System

A distributed server system model is used in which host execute the job in parallel. All the scenario is taken where jobs are executed by the machines upon arrival, and where jobs are placed in central waiting list till the job is requested by the host machine [7]. A Technique is used which is introduced as a busy period transaction. The long jobs are separated from the short job in long job queue if it is idle this process is called as cycle stealing. With the help of cycle stealing we can minimize the response time for short jobs but not for the long job which is an issue in distributed system.

1.6 Dynamic Work Load Execution in Distributed System

In many scenarios of the research done the sector of analogous execution of job presume that the number of assignments per job is described by a particular distribution whether it is a constant or standard. The time between the consecutive arrival in queuing problem of jobs and the task execution are also defined by particular distribution. However, in practical system the execution of the jobs in parallel varies depending upon on which particular time interval the jobs execute. The issue addressed in this problem for jobs is on time

and how to do the scheduling of task in parallel. The results are executed at different execution time and are compared with different scheduling pattern by simulation. The most important scheduling policies addressed in this paper are analysed. The first come first serve, the shortest task first, the limited shortest task are used. Jobs are examined as independent so that we can perform it on any processor. Each task is dispatched to any processor in a random manner. The task waits for the other task of the same job to be executed. So homologous in between the task is required for the completion of execution. Jobs with a smaller number of tasks have a prolonged execution time.

1.7 Various Features of Distributed System

1) Organizational and Economic Reasons: As many organizations still are decentralized and a distributed approach is followed and suited for the structure of the organizations. In the distributed database the idea of the centralized control is less preferred. In the distributed database a hierarchal control is used based on the global database administrator who has a full control the whole database. We also use a local database that has a control on the local database. A high degree of autonomy is there in local databases as compared to the global databases.
2) Existing Database Interconnection: In a distributed database several databases are interconnected to teach other and the applications are accessed globally. A bottom up approach is followed and a distributed system is made from the already existing databases. The reconstruction of the load is done and the effort required for the reconstruction of load in distributed database is less as compared to the centralized database because the allocation of the jobs in centralized system leads to complication.
3) A Process of Incremental Growth: A distributed database approach supports a smooth incremental growth with less impact on the already existing units. New branches and new ware houses are easy to add in the distributed system then in a centralized system. In a centralized system the care for the future enhancement of the application is done on the basis of the initial dimensions [8]. In a distributed system we can easily scale in and scale out the resources depending upon the usage of the resources
4) Availability and Reliability of Resources: The distributed database approach is used in order to obtain the high reliability and availability of data. The data is redundant at various sites from where the data is being accessed to improve the availability of data. The redundancy of data is done because in a distributed database the fact is that data is not placed at one site it is geographically dispersed. There are some common properties which tie them together so that we can distinguish between a geographical database and local databases that are resided at the different sites of the network.
5) Complex Physical Structure and Efficient Access: Complex accessing of the structures like the indexes, interfile chains, are a major aspect of the traditional databases. The support for these structures is the most important part of database management system. The main purpose for the efficient access of the data is to obtain the efficient access of the data. Efficient access is the issue in the distributed databases, but physical structures are not technological issue.
6) Integrity, Recovery and Concurrency Control: The issue of the consistency recovery and concurrency control are strongly intercreative. The only solution of these issues is to provide transactions. A transaction is an atomic unit of execution. We can also call it as

a sequence of operation which is either all committed or all aborted. Atomic transactions are the method to insure the consistency and integrity in database. Because they assure that either all the action which convert the database from one consistent state to other consistent state are performed or the state of initial consistent state is left untouched.

1.8 Deadlock Detection and Concurrency Control for Distributed System

The main purpose of locking the data item whenever we access a transaction is that a data item which is joined by another job must wait until the lock get released by the data item of another transaction. A transaction is locked in shared mode if we only want to read the data and the transaction is locked in the exclusive mode if we want to write the data item. A transaction is well performed if always lock a job in shared mode before reading the data, locked the data in a mode where a lock is applied on the whole before making any updating on transaction. The results of the locking mechanism are based on the fact that all the transaction is performed in an order. Conflict arises between the two transactions if we lock the same data with two opposite modes. Read-write and write –write conflict arises in transaction. We can also refer it as a granularity of the transaction. Nested transaction is accessed. The ACID properties in a transaction ensure the consistency in geographical distributed system in the presence of failure and concurrent operations. The transactions are accessed in a serializable fashion. Multiple transactions are executed concurrently and the result must be same and it appears as if the transactions are executed in some order. A transaction may commit or abort. This is shown in Figure 1.1.

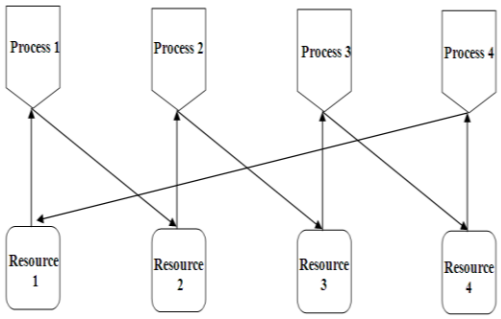

Figure 1.1: Deadlock scenario

2. Conclusion

In this paper, we have proposed a policy that overcomes the issues that arise in the distributed environment during the execution of the transaction. A hybrid approach mechanism is used in large scale distributed memory multicomputer called as the hierarchical scheduling that put the focus on the potential benefit of space sharing and time sharing by overcoming their drawbacks at the same time. A new idea which inspired to add and replace the number of the traditional mechanism used to maintain the atomicity, consistency and integrity of the database. In the proposed work, we have simulated many ideas of integrity which emphasize the importance of accuracy in distributed environment. By the space sharing techniques, the jobs are spotted on different processors in distributed environment. If one processor is not able to fulfill the requirement of the jobs than the jobs will be spotted on different processors which make the job executed in less time and also maintain the integrity in distributed database

3. Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] Dandamudi, Sivarama P, Cheng, Philip S P, “A Hierarchical Task Queue Organization for Shared-Memory Multiprocessor Systems”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, pp. 1-16, 1995.

[2] Abawajy, J H Ã, “Adaptive hierarchical scheduling policy for enterprise grid computing systems”, *Journal of Network and Computer Applications*, Vol. 32, pp. 770–779, 2009.

[3] Dandamudi, Sivarama P, “A Comparative Study of Adaptive and Hierarchical Load Sharing Policies for Distributed Systems”, *In Proc. Int. Conf. on Computers and Their Applications, Honolulu, Hawaii*, 1998.

[4] Young-Chul Shim, “Performance evaluation of scheduling schemes for NOW with heterogeneous computing power”, *Future Generation Computer Systems*, Vol. 20, pp. 229–236, 2004.

[5] Abawajy, J H, “An efficient adaptive scheduling policy for high-performance computing”, *Future Generation Computer Systems*, Vol. 21, pp. 220-227, 2005.

[6] Michael Lo and Sivarama P. Dandamudi School, Performance of Hierarchical Load Sharing in Heterogeneous Distributed Systems”, *In Proc. Int. Conf. Parallel and Distributed Computing Systems*, pp. 1-8, 1996.

[7] Dandamudi, Sivarama P., Zhou, Zhengao, “Performance of adaptive space-sharing policies in dedicated heterogeneous cluster systems”, *Future Generation Computer Systems*, Vol. 20, pp. 895–906, 2004.

[8] Wu, Ming, Member, Student, Sun, Xian-he, “A General Self-adaptive Task Scheduling System for Non-dedicated Heterogeneous Computing Ming”, *IEEE*, pp. 1-11, 2006.