ISSN 2348 – 9928
IJAICT Volume 2, Issue 5, May 2015 Doi:01.0401/ijaict.2014.02.18 Published Online 05 (05) 2015

DYNAMIC SCHEDULING AND RESCHEDULING WITH FAULT TOLERANCE STRATEGY IN GRID COMPUTING

Ms. P. Kiruthika
Computer Science & Engineering,
SNS College of Engineering,
Coimbatore, Tamilnadu, India.

Prof. S. Gokul Dev
Computer Science & Engineering,
SNS College of Engineering,
Coimbatore, Tamilnadu, India.

Abstract — ***Grid Computing is a form of distributed systems that enables the sharing of resources from various administrative domains to solve a particular problem. Grid resource management is defined as the process of identifying requirements, matching resources to applications, allocating those resources, and scheduling and monitoring grid resources over time in order to run the various grid applications. Job scheduling is used to schedule the applications requested by the user to allocate the jobs with minimum time. The process of monitoring refers to systematically gather information about the status of all resources. The data stored in the Distributed Hash Table replicates the request given by the user. The proposed system focused on Fault Identification and Dynamic Scheduling. Since there is a probability of resource overloading due to dynamic scheduling of multiple jobs, and there may be chances of fault in the resources. The monitoring component gives the information about the resource overloading. In this, a novel algorithm is proposed to address this issue through implementing a strategy for dynamic rescheduling of jobs in the grid environment.***

Keywords— Job scheduling, load balancing, fault tolerance.

I. INTRODUCTION

A grid is a collection of resources, sometimes referred to as nodes, machines, members, donors, clients, hosts and engines. They all contribute any combination of resources to the grid. Some resources may be used by all users of the grid while others may have specific restrictions. A grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of resources distributed across multiple administrative domains on their availability, capacity, performance, cost and users quality of service requirements. Grid computing is a model of distributed computing that uses geographically and administratively disparate resources.

It is used for solving large-scale computational and data intensive problems in science, engineering, and commerce thus creating virtual organizations and enterprises that come together to share resources and skills, core competencies or resources in order to better respond to business opportunities. Grid computing is an emerging field and can be visualized as an enhanced form of distributed computing. A new type of computing then came into existence, known as distributed computing. In distributed computing, the problem is divided into numerous tasks that are then distributed to various computers for processing. They are generally connected through a Local Area Network (LAN) and the problem is also divided into modules that can be executed in parallel to each other. The collections of clusters (computing resources) form the grid. Sharing in a grid is not just a simple sharing of files but of hardware and software, data and other resources.

Grid can be classified into three types. They are Computational grid, Data grid and Service grid. A computational grid that has the processing power as the main computing resource shared among its nodes. In computational grid, the main resource that is being managed by the resource management system is the processors. The data grid deals with the controlled sharing and management of large amount of distributed data and also provides an infrastructure for synthesizing new information from data repositories such as digital libraries or data warehouses that are distributed in wide area networks. . Service grid is a system that provides function, program license, resource and dynamic creating, running, maintaining and cancelling of application.

1.1 Background

Load balancing must perform a critical role in enhancing the utilization of each resource and shortening the expenditure of time. Load-balancing algorithms can be roughly categorized as centralized, decentralized, or hierarchical in terms of location where the load-balancing decisions are made. Load-balancing algorithms can be static or dynamic. In a static algorithm, the scheduling is carried out according to a predefined approach.
On the other hand, a dynamic algorithm adapts its decision to the current state of the system. Independent job scheduling aims at computing efficient and optimal mapping of jobs to the grid

resources. Heuristic and Meta heuristic approaches are the most feasible methods of scheduling in grids due to their ability to deliver high-quality solutions in reasonable computing time [1]. Scheduling algorithms can be separated into two types. They are batch mode and online mode. In batch mode, jobs are queued and collected into a set. The scheduling algorithm will start after a fixed time period. Batch mode heuristic algorithms are more appropriate for environments utilizing the same resource. In online mode, jobs are scheduled when they arrive.
Since the grid environment is heterogeneous and speed of each processor varies quickly, online mode heuristic scheduling is more appropriate for grid environment [2]. Resource failures (processors/links) may frequently occur in grid systems and have an adverse effect on applications. Consequently, there is an increasing need for developing techniques to achieve fault tolerance [3]. In multiprocessor systems, fault tolerance can be provided by scheduling replicas of jobs on different processors.

1.2 Motivation

There has been a great effort in recent years in developing fault-tolerant scheduling and load-balancing algorithms, capable of improving the reliability and flexibility of computer systems with minimum execution time. In this research we focused on providing fault free resources with minimum response time. There are many types of fault for which tolerant methods are implemented. This made us to implement the fault tolerance dynamic scheduling method with minimum execution time. The major contribution is to identify the fault when the processor gets overloaded. Minimizing the response time when the processor gets rescheduled.

1.3 Our Contribution

Our Contribution in this paper is multifold. We have proposed dynamic scheduling and rescheduling when fault is identified. The type of fault occurred here is overloaded fault. This fault occurs when the processor gets overloaded. To identify this type of fault the novel algorithm named Dual Space Search Algorithm (DSSA) is used proposed a dynamic, adaptive, and decentralized performance driven fault-tolerant load-balancing algorithm for computational grid environment. One of the key strengths of our algorithm is in estimating the system parameters and in proactive job migration. The contribution is to provide the fault free processor when this processor gets overloaded. The overloading of resources are monitored and processed by Distributed Hash Table which provides the key value and load of the processor.

II. RELATED WORK

Although load balancing, job scheduling, and fault tolerance are active areas of research in grid environments, these areas have largely been and continue to be developed independent of one another each focusing on different aspects of computing. The failure detection approach using Dynamic Heartbeat Grouping Algorithm within a Grid requires the use of a robust and efficient grouping algorithm that incorporates new members into the failure detection system, relocates members from a group that has too few members and spawns new groups when group membership exceeds the ceiling[4].

In the Consensus algorithm [5], all functioning processes must propose and unanimously agree on a value, in spite of the fact that any of the participating processes may fail during the execution of the algorithm. Consensus has interesting practical uses as it can be used to implement essential fault-tolerant functions such as leader election and voting. A fundamental result from distributed systems is that Consensus cannot be implemented in a distributed system subject to crash failures if the system is asynchronous, i.e., if no timing assumptions can be made, such as the amount of time it takes for two processes to communicate or the amount of time it takes for an operation to complete. One solution to this problem is to augment the distributed system with additional information, such as knowledge about which system components have failed.

Check pointing Algorithm refers to the operation of saving the execution state of a running computation. In order to save a consistent snapshot of any group of communicating objects, it is necessary not to modify any object while the check pointing of the group is in progress. This can be ensured in two ways: blocking and unblocking. In the blocking method, all threads suspend themselves during the first part. In the non-blocking method, all threads execute in this phase, but they are responsible for not modifying any objects that are not check pointed yet [6]. Fault Tolerance Strategy Is Used to save the generated checkpoint is to install checkpoint server and connect them to the resources through a high-speed network. As the grid size increases, the dedicated server can easily become a bottleneck. The stored checkpoint data on a server will be unavailable when requested to restart a failed application. The solution for this problem is to store multiple replicas of checkpoint data, so that the stored data can be recovered even when checkpoint server is unavailable. The last usable checkpoint was saved so the computation time is elapsed.

III. GRID SCHEDULING ISSUES

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [7]. A resource in the computational grid is something that is required to carry out an operation, for example, a processor used for data processing. The resource management of computational grid is responsible for resource discovery and allocation of a task to a particular resource. Usually it is easy to get the information about the ability to process data of the available resource. In this paper, we will discuss the problem that *n* tasks work on *m* computing resources with an objective of minimizing the completion time and utilizing the resources effectively.

If the number of tasks is less than the number of resources in grid environment, the tasks can be allocated on the resources according to the first-come-first-serve rule. If the number of task is more than the number of resources, the allocation of tasks is to be made by some scheduling schemes. Considering the number of tasks is more than the computing resources in this paper, so one task cannot be assigned to different resource, implying that the task is not allowed to be migrated between resources [8].

IV. IMPLEMENTATION

4.1 Scheduling

Job is an atomic unit of work or a collection. The jo scheduling system is responsible to select best suitable machines in a grid for user jobs. The management and scheduling system generates job schedules for each machine in the grid by taking static restriction and dynamic parameters of jobs and machines into consideration. It is very cumbersome in large grids and it determines efficiency of grid. The first come first serve scheduler starts the jobs in the order of their submission. If not enough resources are currently available, the scheduler waits until the job can be started. The processes are assigned the CPU in the order they requested. It completes the job in the order of arrival. The ant colony algorithm aims to search for an optimal path in the graph. The main idea behind this algorithm is that the self organizing principle which allow the highly coordinated behavior of ants can be exploited to coordinate populations of artificial agents that collaborate to solve problems

4.2 Fault Identification

Load balancing is a technique to spread work between two or more grid resources, in order to get optimal resource utilization, maximize throughput, and minimize response time. The load of the peer will be calculated before processing the request. The load of the individual peer CPU will be calculated first. If the individual load is greater than its target load then that processor will be considered as overloaded. Else it will be considered as normally loaded. If the processor is normally loaded the load will be handled by itself. Else if it is overloaded the number of processors in the region will be counted.

If there is only one processor the load will be shared to that machine. If the number of processors is even, the processors will be split as pairs of processors and the load is shared to the pair which is having low load. If the number of processors is odd then it will be split as pair of processors and there will be one odder processor. The load will be shared to either pair or to the odd one, which is having fewer loads. The response may be like decompressed file, compressed file, document, zipped folder or unzipped files. The response will be saved in the processor who sends the request. Let T_j denote the target load of physical node j, C_i denote the current load on GIS i, I denote the set of all GISs, and J denote the set of nodes in the system. A grid system is defined to be balanced if the sum of the load S_j a physical node j is smaller than or equal to the target load of the node for every node j ∈ J in the system. That is,

$$S_j = \sum_{i \in I} C_i x_{ij} \leq T_j, \ \forall j \in J \ (I) \qquad (1)$$

In (1), the binary variable $x_{ij} = 1$ indicates that GIS i is assigned to node j. A physical node j is called heavy if its combined load exceeds its target load, $Sj > Tj$; otherwise, the node is called light. When the system is imbalanced, the goal of a load balancing algorithm is to find a way to move GISs from heavy nodes to light nodes in a way that minimizes the total load moved.

In such case, Calls between two points are typically routed through a number of intermediate switching stations, or nodes in a large network, there are many possible routes for each such call. It is thus possible to relieve actual or potential local congestion by routing calls via parts of the network which have spare capacity. Load balancing is essentially the construction of call routing schemes which successfully distribute the changing load over the system and minimize lost calls. Of course it is possible to determine the shortest routes from every node to every other node of the network. In this way the average utilization of nodes will be minimized, but this is not necessarily the ideal way to avoid node congestion, as this has to do with how the traffic on the network is distributed.

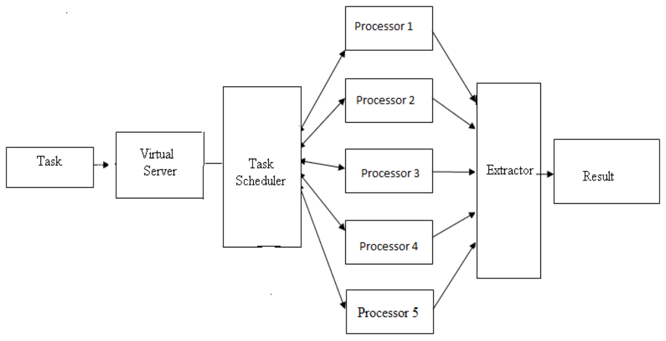

Fig 1 : System Architecture Design

4.3 Distributed Hash Table

A Distributed Hash Table (DHT) is a class of a distributed system that provides a lookup service similar to a hash table (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures. The physical machines in the network are called nodes and the data items are called objects. A typical name for a node in Internet is its IP address. An identifier refers to the unique digital name of a node within a certain integer name space. In grid system, it can be gotten from hashing the name of a node, such as identifier P = hash (IP). Because we apply distinct integers to mark each node's name here, name and identifier have the same meanings if there is no special claiming.

A key is the unique identifier of a data item and can be gained from hashing the name of a data item K = hash (V).In DHT-based grid systems, files are associated to keys (produced by hashing the filename); each node in the system handles a portion of the hash space and is responsible for storing a certain range of keys. After a lookup for a certain key, the system will return the identity (e.g., the IP address) of the node storing the object with that key. The DHT functionality allows nodes to put and get files based on their key, and has been proved to be a useful substrate for large distributed systems and a number of projects are proposing to build Internet-scale facilities layered above DHTs. In DHTs, each node handles a portion of the hash space and is responsible for a certain key range. Routing is location deterministic distributed lookup. The most important enhancements are deterministic locating and load balance.

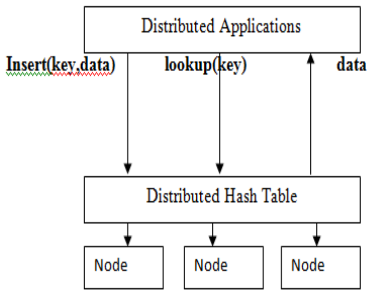

Fig 2 : Distributed Hash Table

4.4 Ant System Heuristics

ASH is an instance of the Ant Colony Optimization approach. In computer science and operations research, the Ant Colony Optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. The first algorithm was aiming to search for an optimal path in a graph, based on the behavior of ants seeking a path between their colony and a source of food. In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food. The original idea comes from observing the exploitation of food resources among ants, in which ants' individually limited cognitive abilities have collectively been able to find the shortest path between a food source and the nest.

V. CONCLUSION

Grid computing is a tendency to solve high demanding applications and all kinds of grid problems. The main objective of grid computing is to achieve high performance computing by optimal usage of geographically distributed and heterogeneous resources. Load balancing is transferring work from heavily loaded to lightly loaded resources. This helps to improve the application performance. Task scheduling is a fundamental issue in achieving high performance in grid computing systems. The reason is, large numbers of tasks are computed on the geographically distributed resources, a reasonable scheduling algorithm is adopted in order to minimize job completion time with uniform load distribution. Fault tolerance techniques are most important for grid systems. Moreover, fault is identified when the processor gets overloaded.

In the Proposed work, the search component that finds the local optima solution with the neighborhood using Ant System Heuristics. The future work is to monitor the identified fault and reschedule the process in the grid environment.

References

[1] J. Kolodziej and F. Xhafa, “Enhancing the Genetic Based Scheduling in Computational Grids by a Structural Hierarchical Population,”Future Generation Computer Systems, vol. 27, pp. 1035- 1046, 2011.

[2] Y.-H. Lee, S. Leu, and R.-S. Chang, “Improving Job Scheduling Algorithms in a Grid Environment,” Future Generation Computer Systems, vol. 27, pp. 991-998, 2011.

[3] A. Benoit, M. Hakem, and Y. Robert, “Multi-Criteria Scheduling of Precedence Task Graphs on Heterogeneous Platforms,” The Computer J., vol. 53, no. 6, pp. 772-785, 2010.

[4] Amit Jain and R.K. Shyamasundar, “Failure Detection and Membership Management in Grid Environments” Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04) pp. 44-52

[5] Fischer M J, Lynch N A, and Paterson M S., “Impossibility of distributed consensus with one faulty process”. Journal of the ACM, 32(2), Apr. 1982

[6] Mangesh Kasbekar Chandramouli Narayanan Chita R Das, “Selective Checkpointing and Rollbacks in Multithreaded Object-oriented Environment” 6th IEEE International On-Line Testing Workshop (IOLTW)

[7] Foster and C. Kesselman (editors), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufman Publishers, USA, 1999.

[8] Lin Jian Ning and Wu Hui Zhong ,Scheduling in Grid Computing Environment Based on Genetic Algorithm, Journal of Computer Research and Development, pp.2195-2199,Vol. 4 ,No.12,Dec 2004.