MJRET
Open Access
Multidisciplinary Journal of Research in Engineering and Technology
Journal homepage: www.mjret.in
ISSN:2348 - 6953

DYNAMIC LOAD BALANCING SCHEME FOR ITERATIVE APPLICATIONS

Rahul S. Wankhade, Darshan M. Marathe, Girish P. Nikam, Milind R. Jawale
Department of Computer Engineering, JSPM's Imperial college of Engineering and Research, Wagholi, Savitribai Phule Pune University,
Pune, India
rahulpoly2009@gmail.com,darshanmrth5@gmail.com,girishp.nkm@gmail.com
jawalemili@gmail.com

Abstract: *Load balancing is a very important issue in parallel and distributed systems to ensure fast processing and optimum utilization of computing resources. Dynamic load balancing scheme is use for minimizing the execution time of single application running in parallel on multicomputer system. Dynamic load balancing is good for efficient use of highly parallel system. In a large distributed computing environment, multiple processes or task can be submitted at any node and the random arrival of tasks in such an environment can cause some nodes to be high loaded while others are idle or low loaded. For some many applications, computation load varies over time. Such applications require dynamic load balancing to increase performance. Some of the load balancing scheme or systems is already existed like centralize load balancing, hierarchical load balancing but some of the limitations in this existing system. The Centralized load balancing schemes, which perform the load balancing decisions at a central location, are not scalable that's why we concentrate on dynamic load balancing. This paper present introduction of our system and there modules like task creation, task scheduling, task migration and resource allocation. This paper include the result of our proposed system time after load balancing and result of task migration if any of performer is busy.*

Keywords: *Load balancing, Distributed dynamic load balancing, Task Migration, Task Scheduling, Resource Allocation.*

1. INTRODUCTION

Dynamic load balancing in iterative application of distributed system is most important part of the parallel system because when number of request is come on any system it may goes down or it goes to high loaded and so we required to reduce the load of that particular system. To improve the utilization of the processor parallel computation required that process be distributed to processor in such a way that the computational load is spread

among the processor. To understand Load balancing, it is necessary to understand load. Load may be define as number of tasks are running in queue, CPU utilization, load average, I/O utilization, amount of free CPU time/memory, etc. The dynamic load balancer can be apply to restored balance. The dynamic load balancer is useful for the distributed operating system where no of nodes are connected to each other. The system is scalable while increase in the no of nodes, it gives same performance as previous.

Dynamic **L**oad **B**alancing (DLB), are based on the redistribution of tasks among the available processors during execution time. This redistribution is performed by transferring tasks from the heavily loaded processors to the lightly loaded processors (during runtime) with the aim of improving the performance of the application typical DLB scheme is generally defined by their four inherent policies and the policies are: Transfer policy, Selection policy, Location policy and Information policy.

Our aim to develop a system for iterative application in distributed system which reduce load and schedule load parallel to the performer in our purposed system. High performance of iterative application can be achieved when task is completed in less time than single system can complete. This is achieved by partitioning the workload parallel to the performer in distributed system. Dynamic load balancing for distributed system is based on the four main module it include task creation, task scheduling, task migration. **Task creation** is nothing but the request which coming from the client. The request is nothing but the task and it consider as load by scheduler. We consider as **task scheduling** is assigning a performer to each task. Task scheduling is perform by scheduler that dividing a task and assign the performer. **Task migration** is the transfer of process from one node to another node in network of workstations or nodes. It is very useful mechanism for balancing the load on distributed system. Load balancing in a distributed system can be done through transferring a process form heavily loaded node to lightly loaded node.

The proposed strategy is more focused towards the distributed dynamic load balancing which is based on Task Creation, Task Scheduling and Task Migration. Task creation is nothing but the request which coming from the client. The request is nothing but the task and it consider as load by scheduler. There are the numbers of task scheduling scheme for example , GA (Genetic Algorithm), Min-Min, Max-Min .Genetic algorithm is a method of scheduling in which the tasks are assigned resources according to individual solutions which tells about which resource is to be assigned to which task[11]. The main disadvantage of this technique is it takes a long for the task scheduling. Y. Zhang evaluated the impact of migration as an additional feature in job scheduling mechanisms for distributed systems. Typical job scheduling for distributed systems uses a static assignment of tasks to nodes. With migration the additional ability to move some or all tasks of a job to different nodes during execution of the job. This flexibility facilitates filling holes in the schedule that would otherwise remain empty. The mechanism for migration we consider is checkpoint/restart, in which tasks have to be first vacated from one set of nodes and then re-instantiated in the target set. There are some limitations it good for static load balancing and job is stored in waiting queue of performer.

The primary contributions of this paper are:

- In Background details contain load balancing scheme that are existed such as centralized , hierarchical , static load balancing .

- Propose system introduce the idea behind the dynamic load balancing for iterative application. It also contain the result of our system for task scheduling. And gives more about task migration.

2. RELATED WORK

Load characteristics in dynamic applications can change over time. Therefore, such applications require periodic load balancing to maintain good system utilization. To enable load balancing, a popular approach is over decomposition. The application writer exposes parallelism by over decomposing the computation into tasks or objects. The problem is decomposed into communicating objects and the run-time system can assign these objects to processors and perform rebalancing. The load balancing problem in our context can be summarized as: given a distributed collection of work units, each with a load estimate, decide which work units should be moved to which processors, to reduce the load imbalance. The load balancer needs information about the loads presented by each work-unit. The load balancing strategy we describe can be used with either model-based or persistence-based load predictions. In persistence-based load balancer, the statistics about the load of each task on a processor is collected at that processor. The database containing the task information is used by the load balancers to produce a new mapping. The run-time system then migrates the tasks based on this mapping.

Dynamic load balancing schemes for iterative system can be broadly classified as centralized, distributed and hierarchical. Centralized strategies tend to yield good load balance but exhibit poor scalability. Alternatively, several distributed schemes have been proposed in which processors autonomously make load balancing decisions based on localized workload information. Popular nearest neighbor schemes are dimension-exchange and the diffusion methods. Dimension-exchange method is performed in an iterative fashion and is described in terms of a hypercube architecture. A processor performs load balancing with its neighbor in each dimension of the hypercube. Diffusion based load balancing schemes were rest proposed by Cybenko [8] and independently by Boillat [7]. This scheme suffers from slow convergence to the balanced state.

To overcome the disadvantages of centralized and distributed, hierarchical [2, 4, 1] strategies have been proposed. It is another type of scheme which provides good performance and scaling. In our proposed scheme, global information is spread using a variant of gossip protocol [10]. Probabilistic gossip based protocols have been used as robust and scalable methods for information dissemination. Demers et al. use a gossip-based protocol to resolve inconsistencies among the Clearinghouse database servers [6]. Birman et al. [1] employ gossip-based scheme for bi-modal multicast which they show to be reliable and scalable. Apart from these, gossip based protocols have been adapted to implement failure detection, garbage collection, aggregate computation etc. But hierarchical strategies suffer from excessive data collection at the lowest level of the hierarchy and work being done at multiple levels.

3. PROPOSED SYSTEM

Our proposed system is Dynamic Load balancing for distributed system is based on four main module which include task creation, task scheduling, task migration and resource allocation. This system is good for the distributed system where the number of nodes is connected to each other and overcome the disadvantage of centralized system which is suffer by slow convergence to the balanced state and limit of the number of connected to the central system. Because our proposed system distributed and if number of nodes increases it not goes down but it improve its performance. It also overcome the disadvantage of static load balancing by dynamic load balancing so by dynamic performance it required less time to perform any task. In our proposed system the load is parallely distributed to performers that's why if number of no increases is it also increase it performance. Using our system client can assume or create one node as the scheduler and other nodes are consider as the performer. Here scheduler is one node that have the more facility than other node it can schedule the task to the performers which are connected to the scheduler.

Main architecture of our system is mainly divided into two main part are Scheduler and performers. Scheduler is a main node who performs task distribution when scheduler is to be loaded. And performers are performing the task which is come from the scheduler.System Architecture shows the details of system.

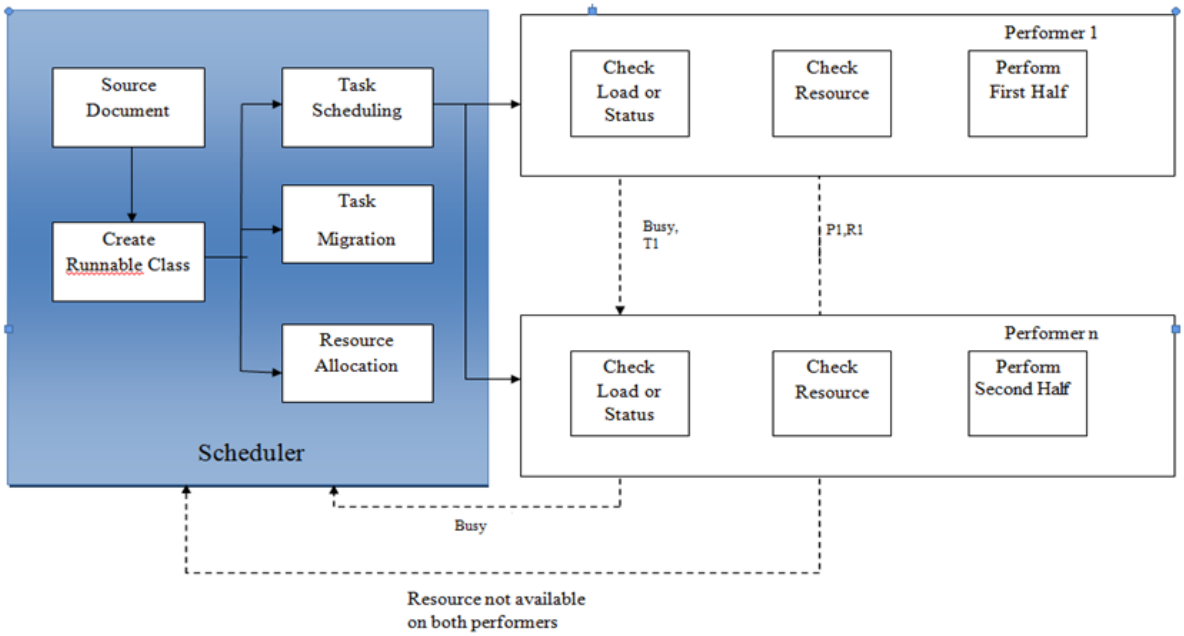

Fig.1: System Architecture

The main for module of our system are as follows.

1. **Task Creation:** Task Creation is the task or job which provided by client or user to scheduler. When task is created it consider as the load so this process is creating the load. Task is created on the scheduler because of using that it is to schedule the task to the performer.
2. **Task Scheduling:** Task Scheduling is main and very important module in our system it can be defined as assigning the performer to the divided task. In task scheduling the scheduler take a job as task and the divided into the number of task or multiple

task in this we can say that no divided task are equal to the number of performer are connected to the scheduler.

3. **Task Migration:** Task migration is another one important module of our system in the name of module it understand that is related to the migration of the task. In other load balancing this mechanism is static but in our system it dynamic. In load balancing if any node is loaded then it is more important to migrate that load to the unloaded node and if the migration is dynamic then it improve the performance of any system. In our proposed system migration is perform dynamically it mean that if any performer is busy then it migrate their task to the other performer and by coincidently other node is also busy then that performer to the next one until task not got the unloaded node. In coincidently all node are loaded then task is store in the queue of scheduler. After that when scheduler got any unloaded node then task is assign to that performer or node.
4. **Resource Allocation:** Resource allocation is one module which is might be not added in other load balancing scheme. To perform any task some resources or files are required so if any file or resource which is required by the performer the that resource is allocated from other performer or scheduler directly.

4. RESULT DISCUSSION

Program Name	Time required without Load balancing for iterative application	Time required using Load balancing for iterative application
Print 1 to 100 Number	10920	5468
Print 8 table	1092	546
Print A to Z alphabet	2839	1249
Print 1 to 100 even	10920	5468
Print 1 to 10 Number in String	1092	546

Table 1: Result of Task Scheduling

5.CONCLUSIONS

We conclude that the distributed dynamic load balancing for iterative application is reliable for large network and it takes a minimum time to scheduling a task. It also conclude that balance the load on the working servers the task is been migrating in run time to maintain the balance work flow in distributed scenario. It gives the better performance than other load balancing. And also concluding that resource allocation is perform dynamically.

ACKNOWLEDGMENT

We are always thankful to our guide Prof. D. P. Gadekar for his valuable discussion and constructive suggestions, which greatly contributed to our Paper. We sincerely thank our Head of Department (Comp.) Prof.S.R.Todmal for his reassuring encouragement throughout the preparation of our Paper. We are also grateful to our respected Principal Dr. Sachin Admane sir for his co-operation despite his busy schedule. Last but not the least we thank all the staff members of Computer Department for providing an excellent environment to undergo this Paper.

REFERENCES

[1]. K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. ACM Transactions on Computer Systems (TOCS), 1999.

[2]. G. Zheng, A. Bhatele, E. Meneses, and L. V. Kale. Periodic Hierarchical Load Balancing for Large Supercomputers. IJHPCA, March 2011.

[3]. Harshitha Menon, Laxmikant Kalé A Distributed Dynamic Load Balancer for Iterative Applications

[4]. J. Li_ander, S. Krishnamoorthy, and L. V. Kale. Work stealing and persistence-based load balancers for iterative overde composed applications.

[5]. I. Ahmad and A. Ghafoor. A semi distributed task allocation strategy for large hypercube supercomputers.

[6]. A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry.Epidemic algorithms for replicated database maintenance.

[7]. J. E. Boillat. Load balancing and poisson equation in a graph.

[8]. G. Cybenko. Dynamic load balancing for distributed memory multiprocessors.

[9]. Load Balancing In Distributed Computing.

[10]. Vatsal Shah, Viral Kapadia ;Load Balancing by Process Migration in Distributed Operating System.

[11]. Rafiqul Zaman Khan, Javed Ali Classification of Task Partitioning and Load Balancing Strategies in Distributed Parallel Computing Systems.

[12]. An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Support in Nodes by Exploiting the Interrupt Service.

[13]. Rahul S. Wankhade, Darshan M. Marathe, Girish P. Nikam, Milind R. Jawale Dynamic Load Balancing Scheme for Parallel Distributed System

[14]. Laercio L. Pilla A Hierarchical Approach for Load Balancing on Parallel Multi-core Systems.