ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

An Approch for Share Resource Allocation For Load Sharing

Monika Gurbhele, Chandu Vaidya, Hemant Turkar

Scholar Student, Dept. of CSE, Rajiv Gandhi College of Engineering and Research, Nagpur, India

Assistant Professor, Dept. of CSE, Rajiv Gandhi College of Engineering and Research, Nagpur, India

Assistant Professor, Dept. of CSE, Rajiv Gandhi College of Engineering and Research, Nagpur, India

ABSTRACT: In multiprocessor system it is very important to handle problem of resource sharing because it play an important role in overall system performance. Load sharing is the process of redistributing the task among slaves of the multiprocessor system to improve the resource utilization and response time .It also avoiding a situation where some nodes are heavily loaded while others are idle. For effective performance master node splits the jobs into smaller units and submits those tasks to the slave nodes to complete it .The slave nodes itself imposed the spitted task and send the results to the master node. The master node collects and aggregates all the results from the slave nodes. In this way the given job is completed. In our proposed approach slave need to perform the assign task as per their local decision or it can take helps from their neighbour. If it wants to take the help from neighbour for that it need to take a call to migrate the load on the basis of availability of the neighbour and collects the executed task and submit it to the master. It will give good performance and response time**.** .

KEYWORDS: Load sharing; Task migration; Scheduling

I. INTRODUCTION

In multiprocessor system it is very important to handle problem of resource sharing because it play an important role in overall system performance. Load sharing is the process of redistributing the task among slaves of the multiprocessor system to improve the resource utilization and response time .It also avoiding a situation where some nodes are heavily loaded while others are idle.

In master slave architecture there is master act as server and slaves node. For effective performance master node splits the jobs into smaller units and submits those tasks to the slave nodes to complete it .The slave nodes imposed the spitted task and send the results to the master node. The master node collects and aggregates all the results from the slave nodes.

Proposed system will works in which master will divide the task into number of chunks and assigned the task as per the capacity of slave node. The master will collects the statistical information from slave node at regular interval & then take decision for the transfer. The one which is lightly loaded will get more number of chunks & the one which is highly loaded will get the low load. Then slave need to performed the assign task as per their local decision or it can take helps from their neighbour. If it wants to take the help from neighbour for that it need to take a call to migrate the load on the basis of availability of the neighbour & collects the executed task and submit it to the master. The technique of transferring the task from overload node to lightly loaded node called as task migration. The identification of idle node is done with the help of threshold like CPU, memory, Queue parameters value.

In the existing master slave mechanism server transfers the task to the slaves for the completion, then slaves have to perform as per availability. But in propose approach client will take local decision to execute the task with the help of neighbour by transferring the assigned task to the side computer. This will improve the system performance and give less execution time.

PARALLEL COMPUTING SYSTEM

Parallel Processing Systems are designed to speed up the execution of programs by dividing the program into multiple fragments and processing these fragments simultaneously. Such systems are multiprocessor systems also known as tightly coupled systems. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.

Parallel computing is an evolution of serial computing where the jobs are broken into discrete parts that can be executed concurrently. Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs.

LOAD BALANCING

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., or any combination of the above indicators. Load balancing can be done among interconnected workstations in a network or among individual processors in a parallel machine. Load balancing is nothing but the allocation of tasks or jobs to processors to increase overall processor utilization and throughput [3].

Actually load balancing is done by process migration. But to balance the load it is necessary to measure the load of individual node in network or in a distributed environment. For calculating node above mentioned factor in a definition of load are calculated. After calculating the node of individual, mark the under loaded/free and overloaded/busy node[13,14].

Now to balancing load transfer the process from overloaded node to under loaded node. In this way load can be balance in network of work station or in a distributed environment.

A. Three types of Algorithm [13,14]
1) Sender-Initiated Algorithm
2) Receiver-initiated Algorithm
3) Symmetrically Initiated Algorithm

B. Components of Algorithm [13,14]:
Typically, a scheduler has four components:

- A *transfer policy* [5] that determines whether a node is in a suitable state to participate in as process transfer
- A *selection policy* [5] that determines which process should be transferred
- A *location policy* [5] that determines to which node a process selected for transfer should be sent
- An *information policy* [5], which is responsible for triggering the collection of system state information.

LOAD BALANCING TAXONOMY

A load-balancing algorithm can be characterized as *static* or *dynamic* depending upon its requirement and the nature of the strategy applied [11]. These categories can be sub-classified into various schemes as illustrated in Fig. 1.

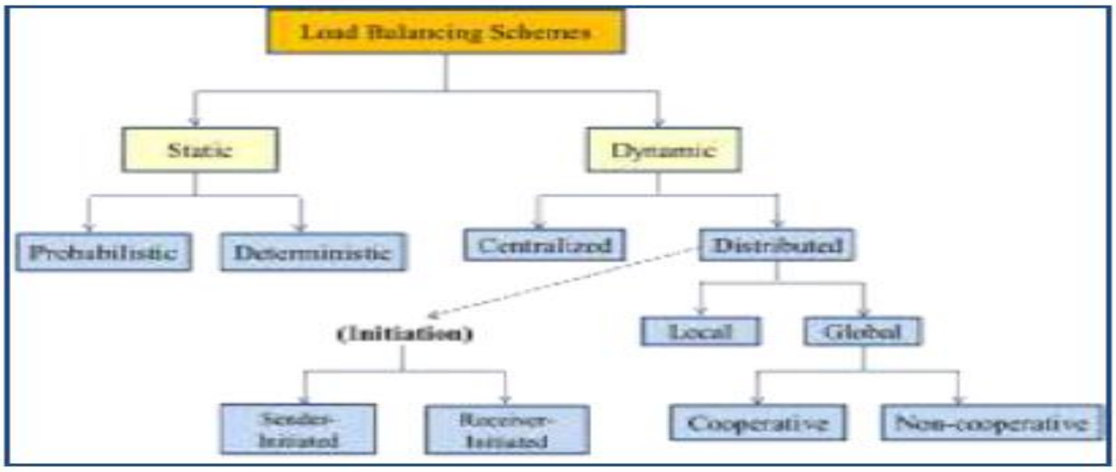

Fig.1. Load Balancing Taxonomy

1. Static Load Balancing

Static Load Balancing policies base their decision on statistical information about the system. They do not take into consideration the current state of the system. Static Load Balancing is performed when the system load and number of processes is fixed and known at compile time. All parameters are fixed for the system. Static Load Balancer makes balancing decision on the basis of average workload of thesystem. Hence the Static Load Balancing takes less time to execute and is simpler. But it is not suitable for the environments with changing workloads. Hence, a dynamic approach is required [7].

2. Dynamic Load Balancing

Dynamic policies base their decision on the current state of the system. They are more complex than static policies. Dynamic Load Balancing is performed when the system load and Number of processes is likely to change at run time. In this case, there is a need of consistently monitoring the system load. This increases the overhead and makes the system more complex. Dynamic Load Balancer makes balancing decision on the basis of current state of the system [11].

TASK SCHEDULING

Task scheduling can be defined as allocating processes to processor so that total execution time will be minimized, utilization of processors will be optimized. Load balancing is the process of improving the performance of system through is distribution of load among processor [10].

Open MP

Defacto standard API for writing shared memory parallel applications in C, C++, and FORTRAN. OpenMP API consists of:

1. Compiler Directives
2. Runtime subroutines/functions
3. Environment variables

OpenMP Programming Model

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

1. Fork and Join Model

   a. Master thread only for all serial regions.
   b. Master thread forks new threads at the beginning of parallel regions.
   c. Multiple threads share work in parallel.
   d. Threads join at the end of the parallel regions.

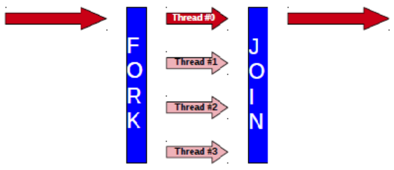

Fig. 2. Fork and Join Model

2. Each thread works on global shared and its own private variables.
3. Threads synchronize implicitly by reading and writing shared variables.

II. RELATED WORK

In the [1] author Introduce a approach based on Genetic Algorithms and Fuzzy Logic for load balancing in parallel multiprocessor systems that call GAF algorithm. Simulation results indicate that algorithm has maximum utilization and it reduce total response time of system. In paper [2] author make the operating systems adjusts the time quantum according to the burst time of the set of waiting processes in the ready queue. Simulation shows that the new proposed algorithm solves the fixed time quantum problem and increases the performance of Round Robin. Paper [3] used three clustering algorithms to determine the efficiency of an algorithm. As per result it is found that SNN algorithm has better execution time and accuracy as compare to k-means clustering algorithm and CURE algorithm. In[4] author designed a new Round Robin Scheduling. It gives better result compare to Round Robin (RR), Improvent Round Robin (IRR), Enhanced Round Robin (ERR), Self-Adjustment Round Robin (SARR), FCFS and some other scheduling algorithm. In [5] author uses process migration technique to balance the load. Algorithm is tested for both type of process non preemptive as well as preemptive. To measure the load of the system it take two parameters in consideration cpu_utilization and memory_utilization. In [6] author present main methods and techniques of scheduling in brief. In [7] author suggest a method to dynamically load balance using service queue wherein every server computes its load value by summing the load parameters like memory utilization, CPU utilization, and network utilization and exchange load value with central node in a certain cyclic period. In [8] author studies the performance of system under different type of load like I/O as well as CPU, MEMORY based on IOCM dynamic load balancing algorithm in heterogeneous computing system.

III. PROPOSED ALGORITHM

a. Design Considerations

Proposed system will works in which master will divide the task into number of chunks and assigned the task as per the capacity of slave node. The master will collects the statistical information from slave node at regular interval & then take decision for the transfer. The one which is lightly loaded will get more number of chunks & the one which is highly loaded will get the low load. Then slave need to performed the assign task as per their local decision or it can take helps from their neighbor. If it wants to take the help from neighbor for that it need to take a call to migrate the load on the basis of availability of the neighbor& collects the executed task and submit it to the master. This will increase system performance.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

b. *Flow chart:*

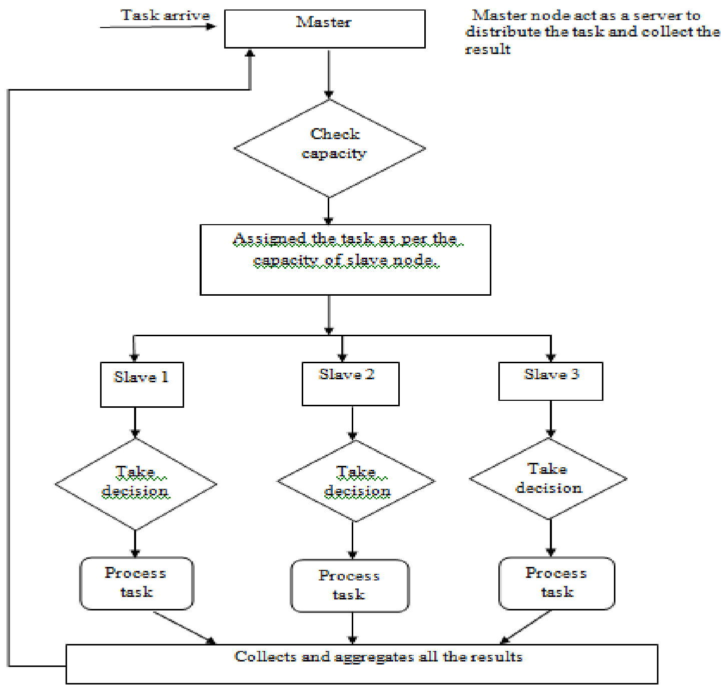
Task arrive
Master
Master node act as a server to distribute the task and collect the result
Check capacity
Assigned the task as per the capacity of slave node.
Slave 1
Slave 2
Slave 3
Take decision
Take decision
Take decision
Process task
Process task
Process task
Collects and aggregates all the results

Fig 3.Flow chart for proposed approach

IV. PSEUDO CODE

```
Step 1: Generate all cluster.
Step 2: Select one node as master which acts as server.
Step 3: Task arrived at master node.
Step 4: Master node splits the jobs into smaller units and submits those tasks to the slave nodes according to their capacity.
Step 5:  Slave need to perform the assign task as per their local decision or it can take helps from their neighbour.
         The identification of idle node is done with the help of threshold like CPU, memory, Queue parameters value

        if ( Idle node available )

                Migrate the load on the basis of availability of the neighbour.
         else
                Itself perform the task by the client

Step 6: The master node collects and aggregates all the results from the slave nodes.
Step 7: End.
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

V. RESULTS

It is expected that the proposed approach increase the throughput of the system by using master slave architecture and decrease the total execution time by migrating the load.

VI. CONCLUSION AND FUTURE WORK

In master slave architecture there is master act as server and slaves node. For effective performance master node splits the jobs into smaller units and submits those tasks to the slave nodes to complete it .The slave nodes imposed the spitted task and send the results to the master node. The master node collects and aggregates all the results from the slave nodes. If slave want to take decision on its level to process task by itself or take the help of neighbour is not possible. So our aim of research work is to design the system which helps to parallel execution of task on tightly coupled platform, with the help of task distribution from the highly loaded node to the lightly loaded node. Future work is to save the total execution time of very large task, with the help of master slave mechanism.

REFERENCES

1. Roya Nourzadeh, Mehdi Effatparvar,'A Genetic-Fuzzy Algorithm for Load Balancing in Multiprocessor Systems', International Journal of Computer Applications (0975 – 8887) Volume 101– No.10, September 2014.
2. Abbas Noon, Ali Kalakech, SeifedineKadry 'A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average' IJCSI Vol. 8, Issue 3, No. 1, May 2011.
3. Bharati Patil, Prof. V. S. Wadne' A Novel Method for Client Server Assignment in Distributed System Using Clustering' Algorithm IJARCSSE Volume 5, Issue 1, January 2015.
4. Radhe Shyam Sunil Kumar Nandal, 'Improved Mean Round Robin with Shortest Job First Scheduling'IJARCSSE Volume 4, Issue 7, July 2014.
5. Jayna Donga, Rahul Goradia, Vatsal Shah,Kanu Patel,' Process Division and Migration based Load Balancing for Distributed Operating System', IJARCSSE Volume 3, Issue 12, December 2013.
6. Masoud Nosrati, Ronak Karimi, Mehdi Hariri ,'Task Scheduling Algorithms Introduction',World Applied Programming, Vol (2), Issue (6), 394-398, June 2012
7. Gowtham Kanagaraj, Naveen Shanmuga sundaram And Sathish Prakash.'Adaptive Load BalancingAlgorithm Using Service Queue',ICCSIT 2012.
8. Sandeep Agarwalla,'Optimal Load Balancing Algorithm in Distributed Systems', JARCSSE Volume 5, Issue 3, March 2015 ISSN: 2277 128X.
9. Mr. Sunil Kumar Pandey, Prof. Rajesh Tiwari,'The Efficient load balancing in the parallel Computer', International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 4, 2013.
10. Abhijit A. Rajguru, S.S. Apte,'A Performance Analysis of Task Scheduling Algorithms using Qualitative Parameters',International Journal of Computer Applications (0975 – 8887) Volume 74– No.19, 2013
11. Shweta Rajani, Niharika Garg, 'A Clustered Approach for Load Balancing in Distributed Systems' SSRG International Journal of Mobile Computing & Application (SSRG-IJMCA) – volume 2 ,Issue 1 ,2015.
12. Jiajing Zhuo , Chen Meng ,'A Task Scheduling Algorithm of Single Processor Parallel Test System ',2007 IEEE.
13. Vatsal Shah and Kanu Patel, "Load balancing algorithm by process migration in distributed operating system" in IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249- 9555 Vol. 2, No.6, 2012 .
14. M. Kacer, P. Tvrdik, "Load Balancing by Remote Execution of Short Processes on Linux Clusters", Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID, 2002).

BIOGRAPHY

Chandu Vaidya Assistant Professor in the CSE Department, Rajiv Gandhi College of Engineering and Research, Nagpur, India. He has received Master of Technology (Mtech.) degree in 2012 RTMNU ,India. His research interests are Parrellel Processing ,Networking and Distributed System etc.

Hemant Turkar is a Research Assistant in the CSE Department, Rajiv Gandhi College of Engineering and Research, Nagpur, India. He is pursuing Phd. His research interests is Image Processing.