Volume - 5 , Issue - 6 , January - 2018 ISSN 2347-2723

Reviews of Literature

An International Multidisciplinary Peer Reviewed & Refereed Journal

Impact Factor: 3.3754

UGC Approved Journal No. 48385

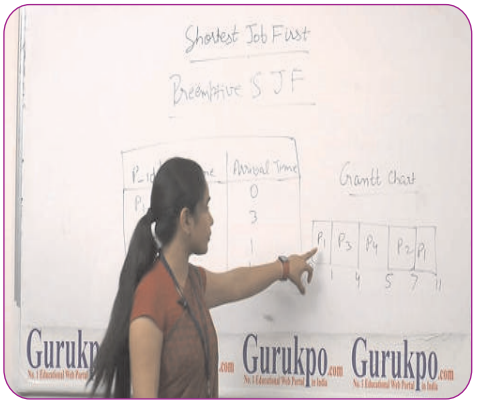
Reviews of Literature
ISSN: 2347-2723
Impact Factor : 3.3754(UIF)
Volume - 5 | Issue - 6 | january - 2018

DISTRIBUTED JOB SCHEDULING WITH PRIORITY TASKS

Dr. Putti SrinivasaRao
Associate Professor and Dean of PG Studies ,
JBIET Hyderabad Telanagana India.

ABSTRACT: -

This paper presents a priority based job processing for Scalable distributed System to distribute the load equally to all the processors. The system identify the best processor to assign the task and the scheduler takes in to account several cost factors like processing cost, storage cost network cost and job complexity.



KEYWORDS: *job processing ,Scalable distributed System ,heterogeneous processors.*

I.INTRODUCTION :

A distributed system consists of many heterogeneous processors with different processing power and all processors are interconnected with a communication channel. In such a system, if some processors are less loaded or idle and others are heavily loaded, the system performance will be reduced drastically. System performance can be improved by using proper load balancing

To start with, consider the basic requirement of a job processing system. The capabilities should include the following:

i.Submit a Job from anywhere.
ii.Assign a priority to a Job.
iii.Monitor the overall state of the system as well as the progress of Jobs.
iv.Processors should be able to handle priority.
v.Processors should be able to report their availability.
vi.Processors should be able to report their current load.
vii.Every component (Dispatcher, Processor etc.) that handles a Job should be able to report the status of the Job.
viii.The status should be updated and be available at a central location.

II. SCHEDULING AND LOAD BALANCING IN DISTRIBUTED SYSTEMS

Scheduling : Scheduling is most important and pays a crucial role in improving the systems performance in distributed systems . From the research point of view the distributed scheduling algorithms can be classified in three categories, i.e. 1-Sender Initiated Algorithms 2-Receiver Initiated Algorithms and 3-Symmetric Algorithms. The basic idea of distributed scheduling is to improve the performance. The challenge in scheduling of distributed systems is how to schedule the tasks in order to be processed effectively .

III .DESIGNING OF PRIORITY SYSTEM MODEL

Distributed Scheduling Mechanism of the Proposed Model

The dispatcher reads the Job definition, identifies its target processor and then submits it for scheduling.

Scheduler then picks up this job and checks for availability and load of the target processor. The target processor can be any processor or a specific processor. Having identified, it checks if the target processor is available and what is its current load. If the Job can be handled by multiple processors, the scheduler finds the processor that is least loaded. This information is available in the database at a central location. After the potential processor is identified, the scheduler sends the request to the target processor through the Processor Queue. Any number of schedulers may be running without having any conflicts. When a job is submitted, a unique identifier is assigned to it and the information is logged into the database.

The Job Processor waits on the Processing queue for new jobs. The processor also has two thread pools for processing jobs. The two pools are Low Priority pool and High Priority pool. The design is flexible enough to have any number of pools for any number of priority levels. Once a Job is available, the processor checks its priority and puts the job into the corresponding internal processing pool. The threads pool manager then takes the job and starts processing. Therefore, for load balancing, the architecture has two important pieces of components.

APPROACH

Consider that all the processors are capable of handling any job and of any priority. In such a scenario, the system will have the following features:

I. Dispatcher can dispatch a job to the request queue, without bothering about the priority.
ii. The scheduler schedules the Jobs. It will have the capability to identify a processor that is least loaded.
iii. The processor is capable of handling jobs of any priority.
iv. The processor internally, maintains independent thread-pools for different priority jobs.
v. Based on the priority, the processor assigns the job to appropriate pool.
vi. The threads in a given pool have pre-defined priority, i.e. they are allocated CPU time based on the priority number assigned to them.

This solution appears simple and feasible.

The constraints are:

i.Each Processor and Monitor must be given a unique ID in the system.
ii.The information about Processor must be added to JOB_PROCESSOR table for validation and dispatch.
iii.A Job's definition must be available in JOB_DEFINITION table before it is submitted. (Note that a Job may be defined only once, but submitted several times. Each such submission will have a unique

IV. COMPUTING LEAST COST TO IDENTIFY THE BEST NODE

In the process of identifying the best processor, the scheduler takes into account several cost factors, such as:

- Processing Cost
- Storage Cost
- Network Cost
- Job Complexity

For every processor, the Processing Cost, Storage Cost and Network Cost are defined. The Job complexity is defined for a Job based on how the Job definition is. The Processing cost is defined based on the computational load on the processor. The Storage Cost is defined based on the storage space needed for the computation of the Job. Network Cost is defined based on the network latency and network load for processing of the Job [16, 45].

The load factor is defined in the range of 0% to 100% scale where 0 is the minimum and 100 is the maximum. The scale can be a certain absolute value (e.g. 0 to 10). However, the same needs to be applied uniformly to all Jobs.

Considering a Job is received at the scheduler, the scheduler checks for all available processors in the central Data store and computes the total cost for the Job for each processor. In the process, it also takes into account the number of pending Jobs for the processor in the queue. The following logic is used for cost computation.

Cp →Processing Cost
Cs →Storage Cost
Cn →Network Cost
Cc →Job Complexity
Cl →Current Load Factor of Processor

Thus, the total cost of computation for the processor would be:

$C_t = C_p + C_s + C_n + C_c + C_l$;

Having computed the total cost of processors, the processor is identified based on the total cost.

$C_m = \text{Min}\,(C_t^1 + C_t^2 + C_t^3 + \ldots + C_t^n)$

The processor corresponding to Cm is then considered as the target processor for the Job under consideration

The final scheduler algorithm is given below (implementation is given in Appendix B).

```
PROC forwardToProcessor
BEGIN
  IF Target Processor Specified
    Foreward Request To Target Processor
  ELSE IF Least Cost Algorithm Used
    CALL PROC findProcessorWithLeastCost
    Forward Request To Processor With Least Cost
  ELSE IF Least Cost Algorithm Used
    CALL findProcessorWithLeastLoad
    Forward Request To Processor With Least Load
  ELSE
    Forward Request To Default Processor
  ENDIF
END
```

V .Priority Job Processing With No Time Limit

Java based job processing system has been designed with various options. As part of this experiment, a Job that compute 200000 prime numbers has been used. Thus, the Job processing time is allowed to take whatever time it takes to compute.

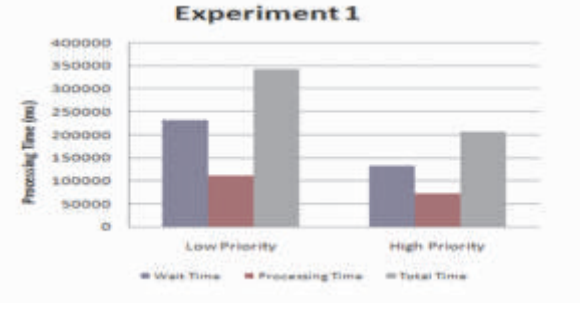


Figure 5.1 Result of Experiment 1

i.The Job could take the priority as an attribute. The priority could be Low or High.
ii.The dispatcher is able to dispatch the job to the alternate queues (i.e. first to Low priority queue, second to high priority queue, third to low priority queue and so on).
iii.The processor is designed to have two independent thread pools, one for Low priority and the other is for high priority.
iv.Each thread pool had the capacity to process 10 jobs concurrently, beyond which, jobs will wait in the queue.
v.When the Job processing is delegated to the appropriate thread, the thread priority is set to either low or high based on the Job's priority.
vi.The job is configured to compute 200000 prime numbers.
vii.A total of 40 Jobs are dispatched, 20 with low priority and 20 with high priority.
viii.The wait time, processing time is measured for each job.
ix.Finally, the average values are plotted as a graph as shown in Figure 5.1

Here is the data collected.

Table 5.1 Data collected from Experiment 1

Priority	Wait Time (ms)	Processing Time (ms)	Total Time (ms)
Low	231224	112135	343359
High	133137	72944	206081

As can be observed, the total time for Low priority jobs are very high as compared to that for the high priority jobs when the number prime number computations are kept same at 2000000.

VI .CONCLUSION

This Paper provided a Priority bases job scheduling in distributed system .it shows how this systems will help to execute the high priority jobs first and improve the performance in distributed systems

REFERENCES

[1] Zeng Zeng, and Bharadwaj Veeravalli "Design and Performance Evaluation of Queue-and-Rate-Adjustment Dynamic Load Balancing Policies for Distributed Networks", IEEE Transactions on computers, VOL. 55, NO. 11, NOVEMBER 2006.
[2] Abhijit , Rajguruand and S.S. Apte "A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters", International Journal of Recent Technology and Engineering (IJRTE) , ISSN: 2277-3878 Volume-1, Issue-3, August 2012.
[3] Iman Sadoop, Sandeep Palur and Ioan Raicu "Achieving Effiecient Distributed Scheduling with Mesage Queue in Cloud Computing for Many-Task Computing and High –Performance Computing".
[4] Andrei Radulescu, Arjan J.C. and van Gemund "Low-Cost Task Scheduling for Distributed-Memory Machines" , IEEE Transactions on Parallel and Distributed Systems VOL. 13, NO. 6, JUNE 2002.
[5] K.Q. Yan, S.C. Wang, C.P. Chang and L.Y. Tseng "The Anatomy Study of Load Balancing in Distributed System", Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06). 2006.
[6] "Kevin Barker, Andrey Chernikov, Nikos Chrisochoides, and Keshav Pingali "A Load Balancing Framework for Adaptive and Asynchronous Applications", IEEE Transactions on Parallel and Distributed Systems, VOL. 15, NO. 2, FEBRUARY 2004.
[7] Jorge E. Pezoa "Decentralized Load Balancing for Improving Reliability in Heterogeneous Distributed Systems ", 2009 International Conference on Parallel Processing Workshops.

[8] Jorge E. Pezoa and Majeed M. Hayat. "Reliability of Heterogeneous Distributed Computing Systems in the Presence of Correlated Failures" . IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 4, APRIL 2014.
[9] Young Joon Lee, Geon Yong Park, Ho Kuen Song, and Hee Yong Youn "A Load Balancing Scheme for Distributed Simulation Based on Multi-Agent System".2012 IEEE 36th International Conference on Computer Software and Applications Workshops.
[10] Deng Huafeng, Zhong Linhui, and Ye Maosheng "An Efficient Load Balancing Algorithm in Distributed Systems". 2010 International Forum on Information Technology and Applications.