Vol. 7, Issue 4, January 2018 ISSN 2249-894X

REVIEW OF RESEARCH

An International Multidisciplinary Peer Reviewed & Refereed Journal

Impact Factor: 5.2331

UGC Approved Journal No. 48514

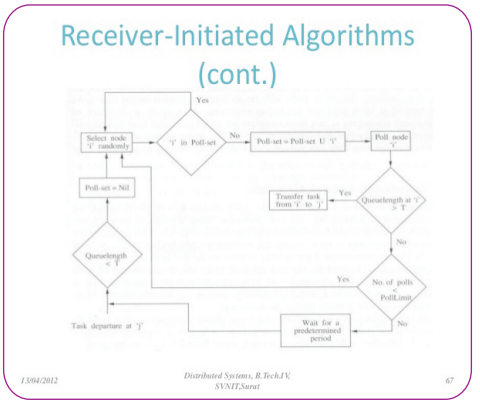
REVIEW OF RESEARCH
ISSN: 2249-894X
IMPACT FACTOR : 5.2331(UIF)
VOLUME - 7 | ISSUE - 4 | JANUARY - 2018

PERFORMANCE IMPROVEMENT IN DISTRIBUTED SYSTEMS USING RECEIVER INITIATED ALGORITHMS

Dr. Putti SrinivasaRao
Professor & HOD in CSE & Dean of PG Studies ,
JBIET Telangana ,Hyderabad, India .

ABSTRACT: -

System performance can be improved by using proper load balancing. if some processors are less loaded or idle and others are heavily loaded, the system performance will be reduced drastically. This paper explains how the performance can be improved using Receiver initiated algorithm in distributed system.



KEYWORDS: *distributed systems , Receiver Initiated algorithms.*

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. The most important aspect of a distributed computing environment is effective coordination of the resources available across several nodes. The algorithm chosen to distribute the tasks should apply the task distribution logic in such a manner that the task executions are completed in most optimum manner, keeping the cost to a minimum and ensuring the resources available are utilised effectively .

II DISTRIBUTED JOB PROCESSING SYSTEMS

Distributed System consists of several systems connected by a network . The nodes may not be similar in configuration, having completely different capabilities The basic requirements of a Job Processing system are:

1. Accepting jobs from different channels
2. Scheduling the jobs
3. Dispatching to the appropriate processor
4. Ensuring the job is completed
5. Managing different job priorities
6. Handling job dependencies
7. Notification mechanisms
8. Monitoring and Control

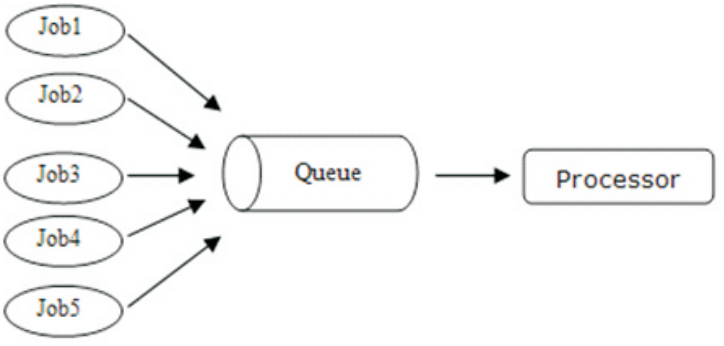
However, there are many challenges in this type of a system Some of the important challenges are given below.

1. The input channels to be supported and the technique therein.
2. Ensuring that once a job is accepted, the system processes the job, completes the job and the job reaches a final state.

3. In case of crashes, it should be possible to re-start the job and it should continue from where it had left earlier.
4. For a long running job, it should be possible to track the progress of the job processing.

In this work, various approaches are introduced and feasible and practical implementations of an effective job processing system.

A Job can be defined as a set of computational tasks, defined in a pre-determined manner, to be performed on the identified data set and the result is generated. Thus, once submitted, the job is expected to complete its execution and submit the result in the form that is already defined.



Job Processing Model (Single Processor)

III SCHEDULING IN DISTRIBUTED SYSTEMS

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 .

RECEIVER INITIATED ALGORITHM

In Receiver Initiated Algorithm Less loaded node gets the job/task from heavily loaded node. In this algorithm if length of the queue is falls below the threshold value then the receiver initiated algorithm is activated and it receives the load from the heavily loaded processor.

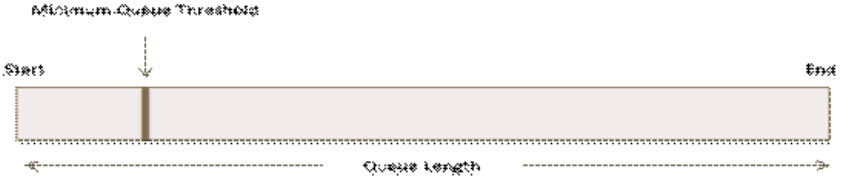


Figure Receiver Initiated Algorithm

It also maintains the policies like sender initiated algorithm. It is not good in case of heterogeneously loaded nodes. Most tasks are pre-emptive. Poor performances because of increased resource under-utilization .Un stability happen with Receiver Initiated algorithms if the system load is low and most of the nodes are receiver.

It consists of different policies like Transfer policy Selection policy Location policy Information policy

Transfer Policy tells when a node transfers the job/task. Selection policy decides which job/task to be

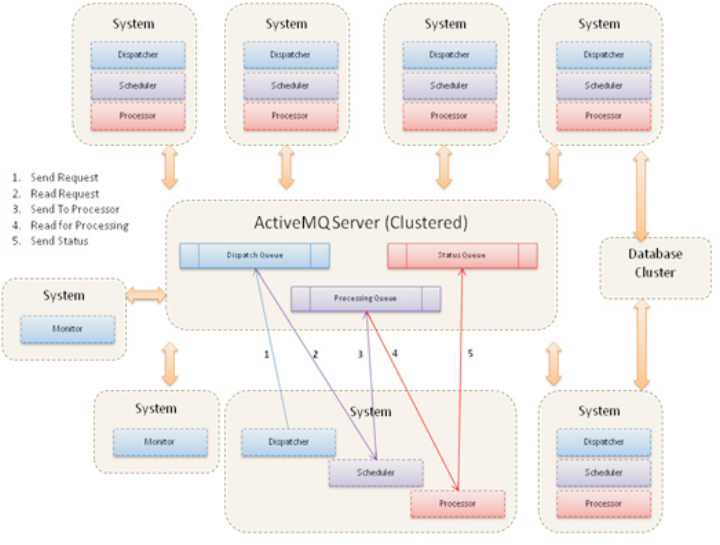
transferred. Location policy decides partner node for job/task migration. Information policy contains information about all the nodes.

IV PROPOSED MODEL OF RECEIVER INITIATED ALGORITHM

The fundamental assumption in this design is the distributed nature of the nodes. The nodes may be present in any physical location, with any type of connectivity to a central message bus. As it will be clear later, the design has built-in load balancing option. The Scheduler has the responsibility of identifying a processor and dispatching a job. The processor will only execute the job.

The various components of the system are 1.Job Dispatcher 2.Job Scheduler 3.Job Processor 4.Job Monitor 5.Dispatch Queue 6.Progress / Status Queue 7.Database / Persistence 8.Processor Affinity

Fig:High Level Architecture



The architecture and design presented here is such that each of the components are self-contained and do not depend on each other. Any number of instances of any component type may be started, without any specific order, yet each instance seamlessly participates in the Job Processing network.

The proposed model can leverage the computing power available with the old and inexpensive hardware as well as future hardware, including the Cloud Computing frameworks.

In this model ,job processing system will have 1.Heterogeneous nodes 2. An Enterprise Service Bus (ESB).3.A central database.4.One or more monitors. 5.One or more processors. 6.One or more dispatchers.7.Jobs can be submitted from any node in the net-work.8.Processors as well as Jobs can be monitored periodically.9.One or more Monitors can perform the task of monitoring messages and update the database.

In order to maintain the flexibility in performing various experiments, the system is designed with various configurable parameters. The dispatcher has also been designed to read a configuration file that may contain a batch of jobs. The system is also designed with a view that newer processing components can be added any time through configuration, without having to modify all the nodes.

V. IMPLEMENTATION

The processing logic is given below

PROC findProcessorWithLeastCost

```
BEGIN
  SET INDEX = 0
  SET COST = 9999
  FOR Each Processor
    IF COST > Processor Cost THEN
      INDEX = Processor Index
      COST = Processor Cost
    END IF
  NEXT
  RETURN Processor[INDEX]
END
```

For experiments, the following technologies / components are used.

i.Java SDK 1.6
ii.MySQL Community Edition Server
iii.Active (A JMS compliant Enterprise Message System)
iv.Net Beans IDE

VI. RESULTS & COMPARISON

In this experiment a Java based job processing system is implemented with normal priority and the Job processing time is allowed to take whatever time it takes to compute.

Table :Comparison of Algorithms with different Parameters

Scheduling Algorithm	Average Wait Time(ms)	Average Processing Time(ms)	Average Total Time (ms)
Receiver Initiated	104824	6081207	6186031
Sender Initiated	204202	7278612	7482814

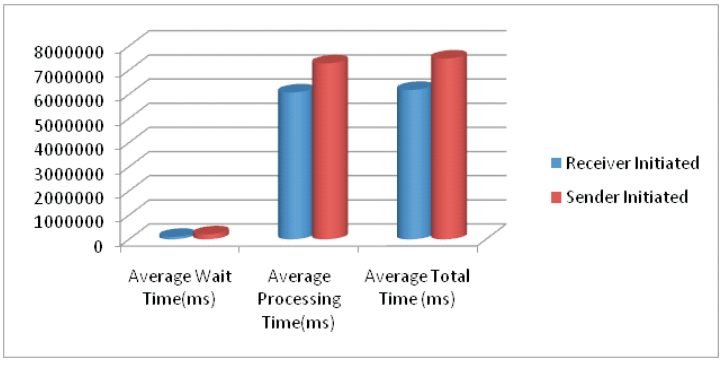


Fig::Comparisons with Algorithms

The results show that at-least 15% improvement in the total processing time in the proposed Receiver Initiated approach than the Sender initiated with normal priority jobs .

VII .CONCLUSION :

This paper presented a Receiver Initiated Scheduling algorithm Technique in distributed systems. This paper also explained how the performance can be improved and shows the comparison of results with the existing algorithms i.e. sender initiated algorithms.

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] T.L. Casavant and J.G. Kuhl. "A taxonomy of scheduling in general-purpose distributed computing systems". Software Engineering, IEEE Transactions on,14(2):141–154, 1988.

[10] K.W. Ross and D.D. Yao, "Optimal Load Balancing and Scheduling in a Distributed Computer System," J. ACM, vol. 38, no. 3, pp.676-690, July 1991.

[11] K.G. Shin and Y. Chang, "A Coordinated Location Policy for Load Sharing in Hypercube-Connected Multicomputer," IEEE Trans. Computers, vol. 44, no. 5, pp. 669-682, May 1995.

[12] A.N. Tantawi and D. Towsley, "A General Model for Optimal Static Load Balancing in Star Network Configurations," Proc.PERFORMANCE '84, E. Gelenbe, ed., pp. 277-291. North-Holland:Elsevier Science Publishers B.V., 1985.

[13] C.Z. Xu and F.C.M. Lau. "Iterative dynamic load balancing in multi computers". Journal of the Operational Research Society", pages 786–796, 1994.

[14] A.N. Tantawi and D. Towsley, "Optimal Static Load Balancing in Distributed Computer Systems," J. ACM, vol. 32, no. 2, pp. 445-465, Apr. 1985.

[15] Michael Boyer, Kevin Skadron, Shuai Che, and Nuwan Jayasena. "Load balancing in a changing world: dealing with heterogeneity and performance variability". In Conf. Computing Frontiers", page 21, 2013.