Advanced Research in Electrical and Electronic Engineering
p-ISSN: 2349-5804; e-ISSN: 2349-5812 Volume 3, Issue 3, April-June, 2016, pp. 208-212
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/Publication.html

Analysis of Load Balancing Technique by using Different Discrete Distributions of Load

Varun Krishna[1] and Sarita Singh Bhadauriya[2]

[1,2]Electronics Engineering Dept. Madhav Institute of Technology & Science, Gwalior, M.P. India
E-mail: [1]varun.krish28@gmail.com, [2]saritamits61@yahoo.co.in

Abstract—*The primary goal in parallel computing systems is the base execution time, so circulation of the workload in both the classes viz. heterogeneous and homogeneous is a complicated issue. For a proficient load balancing, analytical methodology has been employed. In this paper, a centralized Load Balancing Strategy has been proposed utilizing adaptive threshold approach for the parallel and distributed systems. The central scheduler screens the load and yields the choice to disperse the workload to various handling components. In this several distributions on load has been assumed and threshold values are considered as versatile which are changed by load of the system.*

1. INTRODUCTION

In load balancing, arrangement of the jobs is a vital undertaking to circulate the load productively on various processing elements or nodes, so that in minimum time more work is finished [1]. Thus the turnaround time is reduced. This move is made by the scheduler which dispatch the jobs for execution to various handling components and screens their running. The circulation of loads on different nodes is uneven, so some are over-burden and rest are underutilized [2].

Thus, for efficient working of such grid systems, task scheduling and resource management are crucial functions [3], where issues of task allotment and load balancing represent a typical issue for most grid systems. Load balancing systems expects to equally spread the load on each computing node, maximizing their utilization and minimizing the undertaking execution time [4]. In general, load- balancing algorithms can be generally categorized as centralized or decentralized in terms of the location where the load balancing decisions are made[5].In centralized scheduling, a central machine acts as a resource manager to schedule jobs of all the surrounding nodes that are a part of the grid environment[6].

Jobs are first submitted to the central scheduler, who then dispatches the jobs to the various nodes where as in decentralized scheduling, jobs are randomly distributed to various nodes. Load balancing is also categorized as static and dynamic. In static load balancing, the number of jobs are fixed during execution time and in dynamic load balancing, the number of jobs are added during execution time [4,5].

In this paper the distribution of load on processing elements are Uniform, Binomial, Gaussian and Poisson distributed each with their mean, variance and standard deviation. These parameters are used in calculation of execution time of job. The service rate is normal distributed provided by the central scheduler to various nodes.

This paper is organized as follows. Section 2, outline the architecture of this system. In Section 3, an illustrative example is described, experimental results are presented in Section 4 and finally Section 5 concludes the paper.

2. ARCHITECTURE

The proposed model introduces a centralized dynamic load balancing procedure which persistently monitors the load on the nodes utilizing threshold with the point of minimizing aggregate execution time and the turnaround time of the jobs submitted for execution. In this model, central scheduler distributes the jobs to different computing elements. The distribution of load on handling components is done by Uniform, Binomial, Gaussian and Poisson discrete distributions. Every node has a queue where the designated jobs are lined up and are taken up for execution. The scheduler design has been illustrated in Fig. 1. In this approach, one node serves as central scheduler and others are utilized for job completion. These jobs are dispersed and monitored by central scheduler among various nodes. It places the choice for load balancing utilizing threshold values. Further, new jobs can be incorporated reliably while the older ones can keep finishing the jobs. This process is known as dynamic job scenario with an evolving load[1,7].

Whenever, the model encounters an uneven distribution of load as per various distribution discussed, a central scheduler readjust the load on nodes using threshold for a balanced state. The load on the nodes is evaluated by using two threshold values viz. Lower Threshold (T_L) and Upper Threshold (T_U) values which are adaptive by nature. The allotted workload on nodes, enters in its job execution queue. The global queues are maintained by the central scheduler only and are realized as maximum priority queue for heavily loaded nodes and minimum priority queue for lightly loaded nodes [8]. As the load varies thresholds are followed to suite the changing load on the system making the threshold selection versatile i.e. the threshold values increases with increasing load and vice versa. The load balancing process is instantaneous and iterative until complete load balancing is achieved. However, as the heavily loaded nodes and lightly loaded nodes are reported, the central scheduler starts load balancing.

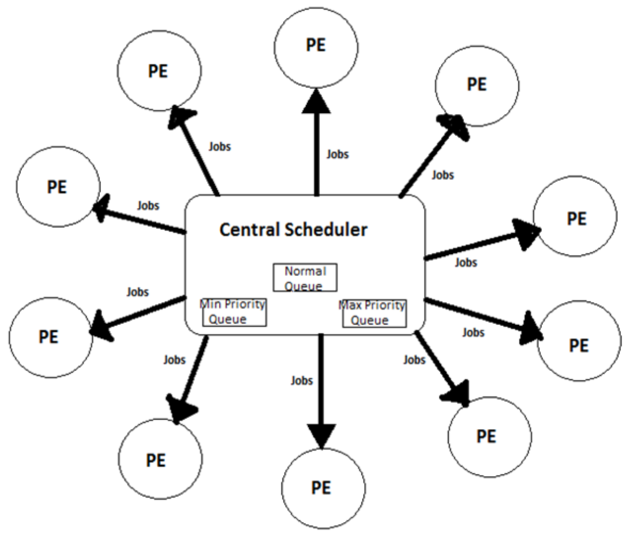

Fig. 1: Scheduler Architecture

The various parameters used in the model are presented in Table 1 along with their description.

Table I: Parameter Used in the Model

Parameters	Description
N	Number of nodes
K	Number of jobs
K_l	Job identifier where $1 \le l \le K$
B_l	Node identifier where $0 \le l \le N$-1
W_l	Workload on each node B_l
T_L	Lower threshold
T_U	Upper threshold
LM	Lower half mean of W_l for the nodes sorted in ascending order
UM	Upper half mean of W_l for the nodes sorted in ascending order
M	Mean of W_l for the nodes sorted in ascending order
B_{min}	Min priority queue containing node identifier for nodes having load W_l below T_L
B_{max}	Max priority queue containing node identifier for nodes having load W_l above T_U
B_{mid}	Queue for nodes having load between T_L& T_U

Since the scheduler load adjusts the workload utilizing thresholds limits, these values for under loaded nodes and overloaded nodes are considered as T_L and $T_{U,}$ respectively which can be calculated with the help of Lower Half Mean (*LM*), Upper Half Mean (*UM*), and Mean *M* values. To calculate *LM*, *UM* and *M*, firstly the nodes are sorted in ascending order of their workloads with the condition $W_l \geq W_{l\text{-}1}$ and then calculated using equations (i) - (iii) as

$$LM = \frac{1}{(N-1)/2} \sum_{l=1}^{\frac{N-1}{2}} W_l \text{ -----} \tag{1}$$

$$UM = \frac{1}{(N-1)/2} \sum_{\frac{N-1}{2}+1}^{N-1} W_l \text{ ----------} \tag{2}$$

$$M = \frac{1}{(N-1)/2} \sum_{l=1}^{N-1} W_l \text{ -------} \tag{3}$$

Using equations (1) - (3), T_L and T_U can be calculated as equations (4)-(5),

$$T_L = \begin{cases} LM : LM \geq 0.9M \\ 0.9\,M : LM < 0.9M \quad \text{------} \\ 1 : LM < 1, 0.9M < 1 \end{cases} \tag{4}$$

$$T_U = \begin{cases} UM : UM \geq 1.1M \\ 1.1\,M : UM < 1.1M \quad \text{--------} \\ 2 : UM < 2, 1.1M < 2 \end{cases} \tag{5}$$

In proposed model the scheduler works with the expectation of bringing that condition of the system in which both *LM* and *UM* (and hence T_L and T_U) ranges between ± 10% of the mean *M* achieving a load balanced state. If *LM* and *UM* are outside this range, T_L and T_U are set to be 90% and 110% of *M*, respectively. In this way, the scheduler balances the load of the system to bring the average workload between ± 10% of the mean *M*. Initially the values of thresholds T_L and T_U are taken as 1 and 2, respectively and are continuously adjusted using the node workload sorted in the ascending order. Nodes having a place with B_{min}, B_{max} and B_{mid} can be chosen using equation (6), (7) and (8) respectively.

$$B_l \text{ € } B_{min} \text{ if } W_l < T_L \text{----------------------} \tag{6}$$

$$B_l \text{ € } B_{max} \text{ if } W_l > T_U \text{------------------------------------} \tag{7}$$

$$B_l \text{ € } B_{mid} \text{ if } W_l \geq T_L \text{ and } W_l \leq T_U \text{---------------------} \tag{8}$$

As the estimation of *LM* and *UM* encroaches *M* the system reaches the balanced state with uniform distribution of number of jobs. The procedure of load redistribution proceeds for remaining number of nodes in B_{min} and B_{max}, outlining lightly loaded and heavily loaded status until either of the queue B_{min} or B_{max} gets vacant. Simultaneously, the threshold values are also modified with the varying values of queue B_{min}, B_{max} and B_{mid}. In this way, no node is empty if additional load is there on any node in a system by utilizing work offloading as basic load redistribution strategy.

The relocation of jobs from a heavily loaded node to the lightly loaded node is governed using equation (9).

Number of jobs to be transferred= $(W_l \text{ € } B_{max} \text{- } W_l \text{€ } B_{min})/2$ (9)

The following parameters are calculated by using these relations:

The execution time can be calculated as, the summation of the ratio of service rate (μ) to the number of jobs at each node.

$T=\sum\frac{\mu}{Ni}$------------------ -(10)

The Sequential time is calculated as:

T_{seq}=Execution time*No of Jobs -- -(11)

The total turnaround time (*TAT*) can be evaluated as:

TAT= Max number of jobs executed on any node *Execution Time of a Single Job --- (12)

The speedup is the ratio of the time taken by the job when completed successively on a node T_{seq} to the time taken for parallel execution T_{par} or *TAT*.

Speedup '*S*'= T_{seq} / *TAT*------------------ ---------(13)

The efficiency can be expressed as,

q =Speedup/Number of nodes --------------------(14)

3. DESCRIPTIVE SAMPLE

In this example, the centralized scheduling is performed that implies no job enters amid execution and arrival rate of job is more noteworthy than service rate so that no job is expelled until allotment is finished.

Assume, total number of accessible nodes as 11. As system is static, B_0 acts as the central node and B_1 to B_{10} act as the handling elements for job completion. The distribution of load on these nodes are uniform, binomial, Gaussian and Poisson, respectively and load W_1 is distributed on each node B_l. Prior, T_L and T_U are suppose to be 1 and 2 respectively. The total number of jobs is 171 which are distributed arbitrary on different nodes. The service rate of various nodes is normal distributed of mean 0.1 and variance 0.9. The table1 depicts the allocation of jobs to various nodes.

Table 1: Initial Allocation of Load

B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
3	15	20	14	5	18	47	23	25	1

3.1 Procedure for load balancing

The following steps are followed for balancing the load.

Step1: Nodes are arranged in ascending order according to the workload as appeared in table 2.

Table 2: Sorting in ascending order

B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
1	3	5	14	15	18	20	23	25	47

Step2: Now assess *LM*, *UM* and *M* by utilizing lower half, upper half and total nodes, separately using the equation(1),(2)and (3).

Step3: Observe the values of T_L and T_U in the range of ±10%. As it does not lie in the extent, therefore new T_L and T_U are assessed by following equation,

T_L = max (max (*LM*, 0.9*M*), 1)

T_U = max (min (*UM*, l.1*M*), 2)

Step4: With the new estimations of T_L and T_U, B_{min}, B_{max} and B_{mid} are organized by (6), (7) and (8), individually.

Step5: Now lower nodes B_{min} and upper nodes B_{max} are adjusted by taking mean of both node values and so on.

Step6: And, the values of B_{mid} stay steady.

Step7: Now the modified values of nodes are set in a frame.

Table 3: Load on Nodes after Balancing

B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
24	14	14	17	15	18	17	14	14	24

Again follow the same steps, until threshold values are in the specified range.

Table 4-Load Redistribution of Nodes of Table 3

B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
24	14	14	17	15	18	17	14	14	24

Table 5- Load Redistribution of Nodes of Table 4

B_8	B_9	B_5	B_4	B_7	B_6	B_1	B_2	B_3	B_{10}
16	16	17	17	17	18	17	17	17	19

Table 6- Load Redistribution of Nodes of Table 5

B_8	B_9	B_1	B_4	B_2	B_3	B_4	B_5	B_6	B_{10}
17	17	17	17	17	17	17	17	17	18

The new estimations of *LM*, *UM* and *M* are now computed as 17, 17.2 and 17.1 respectively which are approximately equal. This is the driving condition which portrays the even distribution of load. The values of T_L and T_U are 17 and 17.2 respectively. Hence, no node is observed to be under B_{min} while nodes that come under B_{max} are B_{10}. The load is readjusted only when B_{min} and B_{max} are both non void. Since B_{min} has become void, no load balancing is required any further. Load status after execution of 17 jobs is shown in Table 7.

Table 7: Nodes after 13 Jobs Execution

B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
0	0	0	0	0	0	0	0	0	1

Again, the estimations of *LM*, *UM* and *M* are 0, 0.2 and 0.1 respectively. So the values of T_L and T_U are 1 and 2 respectively. All nodes exist in B_{min} and there is no node in B_{max}. This state introduces the other extreme state in which B_{min} is non empty and B_{max} is empty, shows the balance condition of frame. In this manner, the nodes carry on execution till either of the queues (B_{min} or B_{max}) gets empty. Table 8 depicts the nodes with exact number of jobs distributed and consequently executed. It is observed that this value is close to 17.1 which is the mean M of the workload.

Table 8: Nodes with Number of Jobs Allotted / Executed

Node No	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
Allotted	1	3	5	14	15	18	20	23	25	47
Executed	17	17	17	17	17	17	17	17	17	18

4. EXPERIMENTAL RESULTS

The experiment is performed on MATLAB by allocating the number of jobs using different distributions, the number of jobs varied from 50 to 500. Execution time is recorded to analyze or break down the feasibility of the calculations. The 11 nodes is utilized as a part of this model in which one is central scheduler and 10 nodes are processing elements on which number of jobs are distributed. Load balancing is achieved by means of threshold calculation iteratively. The performance is assessed by means of turnaround time, speed up and efficiency. The system is bringing about an efficiency almost of 90% on each distribution which can be dealt with as fairly great. The different graphs are plotted on each distribution of various parameters as number of iterations, efficiency and execution time as shown in Fig. 3, 4 and 5 respectively.

Table 13: Mean, Variance and Standard Deviation of Different Distribution

Distribution	Uniform	Gaussian	Binomial	Poisson
Mean	27.5	17.1	25.4	49.1
Variance	249.611	179.877	10.0441	71.2111
Standard Deviation	0.0154	0.0179	0.0220	0.0108

Table 14: Resulting Various Parameters of Different Distribution

Distribution	Uniform	Gaussian	Binomial	Poisson
Number Of Jobs	275	171	254	491
Execution Time(s)	0.8807	0.5255	1.5239	1.4270
TAT(sec)	24.6598	9.4597	39.6202	72.774
T_{seq}(sec)	242.1942	89.8673	387.0589	700.635
Speedup	9.8214	9.5	9.7692	9.6275
Efficiency(%)	98.214	95	97.69	96.275
No Of Iterations	3	5	3	2

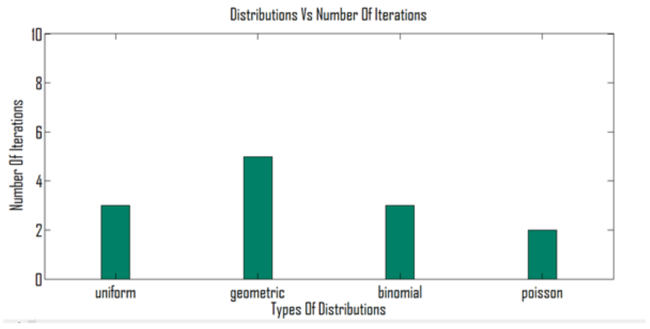

Fig. 3: Graph plot between various distribution vs number of iterations.

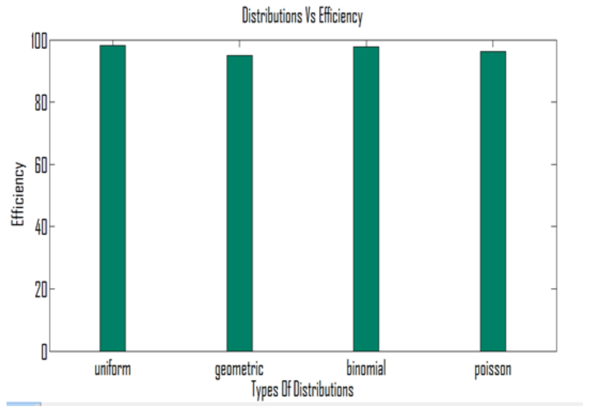

Fig. 4: Graph plot between various distributions vs efficiency.

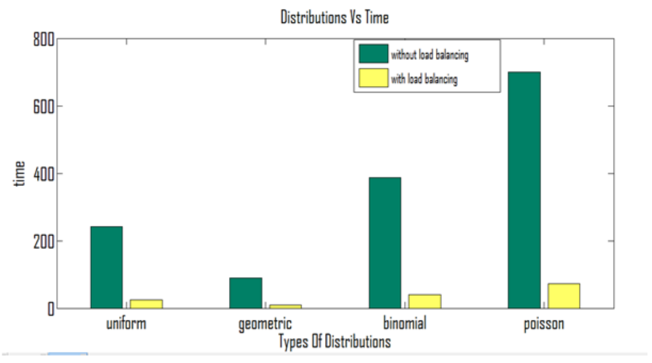

Fig. 5: Graph plot between various distribution vs time

In terms of efficiency, uniform distribution is more effective as compare to other distributions. On the other hand, less number of iterations is required for poisson distribution to balance the load.

By load balancing strategy, all distributions require less execution time but poisson has vast difference between both the execution time (with or without load balance).

5. CONCLUSION

This paper introduces the load balancing techniques using adaptive threshold approach for the parallel and distributed systems. Various distributions on the load is assumed and service rate is normal distributed with 0.1 mean and 0.9 variance. It is observed that, the load balancing is accomplished by following factors that is number of iterations, efficiency and the execution time (with or without load balancing). The outcome exhibit that uniform distribution is more efficient and poisson requires less number of iterations yet the distinction between the efficiency of both the distribution is little. Therefore, poisson distribution of load is superior to other distributions.

REFERENCES

[1] Taj Alam and Zahid Raza, "A Dynamic Load Balancing Strategy with Adaptive Threshold Based Approach" *2nd IEEE International Conference on Parallel, Distributed and Grid Computing*, 2012 pp-927-932

[2] Syed Nasir Mehmood Shah, Ahmad Kamil Bin Mahmood and Alan Oxley, "Hybrid Resource Allocation Method for Grid Computing", *Second International Conference on Computer Research and Development,* 2010 IEEE, pp-426-431

[3] Yajun Li, Yuhang Yang, Maode Ma, Liang Zhou, "A hybrid load balancing strategy of sequential tasks for grid computing Environments" *Future Generation Computer Systems,* 2009 Elsevier, pp-819-828

[4] Shafi Jindal, RK Bansal, Savina Bansal, "Efficient Load Balancing Scheduling Technique on Grid Computing" *International Conference on Communication, Information & Computing Technology (ICCICT),* 2015 IEEE pp-1-6.

[5] Amrik Singh, Lalit K. Awasthi, "Performance Comparisons and Scheduling of Load Balancing Strategy in Grid Computing", IEEE 2011, pp-438-443

[6] Riky Subrata, Albert Y. Zomaya, Bjorn Landfeldt, "Artificial life techniques for load balancing in computational grids", *Journal of Computer and System Sciences*, 73 2007 pp-1176–1190

[7] Wantao Liu, Rajkumar Kettimuthu, Bo Li, Ian Foster, "An Adaptive Strategy for Scheduling Data-Intensive Applications in Grid Environments", *17th International Conference on Telecommunications,* 2010 IEEE pp-642-649

[8] Rupak Biswas, Sajal K. Das, Daniel J. Harvey, Leonid Oliker, "Parallel dynamic load balancing strategies for adaptive irregular applications", *Applied Mathematical Modelling,* 2000 *Elsevier,* pp-109-122

[9] Kun-Ming Yu, Cheng-Kwan Chen, "An Evolution-based Dynamic Scheduling Algorithm in Grid Computing Environment" *Eighth International Conference on Intelligent Systems Design and Applications,* 2008 IEEE pp-450-455

[10] Vishnu Kant Soni, Raksha Sharma, Manoj Kumar Mishra, "An Analysis of Various Job Scheduling Strategies in Grid Computing", *2nd International Conference on Signal Processing Systems*, 2010 IEEE, vol2, pp-162-166.

[11] Belabbas Yagoubi and YahyaSlimani, "Dynamic Load Balancing Strategy for Grid Computing*", International Journal of Computer, Electrical, Automation, Control and Information Engineering,* Vol:2, No:7, 2008, world Academy of Science, Engineering and Technology, pp-2424-2429