Clustering Based Dynamic Load Balancing Algorithm

Deepti Sharma, Asst. Professor (IT)
Department of Computer Science, Jagan Institute of Management Studies, Affiliated to GGSIPU, Rohini, Delhi

Abstract: *The growth of Internet requires speedy web servers so that reaction time can be reduced and better services can be given to clients. The multiple servers are interconnected to reduce the workload and partitioning the request. There is a vital need of a load balancing technique to distribute this load among web servers by achieving performance and minimizing response time. In this paper, we proposed a framework for load balancing in heterogeneous web server clusters. Load is distributed on the basis of memory and processing requirements. Framework is explained with pseudo code and experimental results are also shown.*

Keywords: *Web Server Clusters, Load Balancing, Response time, Throughput*

I. INTRODUCTION

In a distributed system, jobs arrive at various hosts independently, causing uneven loads on the hosts. To balance the load effectively, there are various factors that need to be considered. Load balancing means to transfer jobs among different processors in a distributed system. Its aim is to keep the performance of application by keeping processoridle time and interprocessor communication as low as possible. To balance load effectively, there are various factors that need to be considered like performance of system after doing load balancing, the algorithm to be used, load index value and impact of load balancing on system behavior [9].

In case, jobs are coming with constant workloads, static load balancing can be used. In such case, there may be one central processor that can act as a pre-processorto the computations. For other applications with inconsistent workloads dynamic load balancers can be used. Several strategies for static and dynamicload balancing have been developed.

Load balancing in distributed systems is a challenge due to the independence of the processors and the IPC overhead incurred. A parallel and distributed systemwhere large task is divided among multiple hosts is the finest option for computing parallel applications.In such systems, there are chances that one node becomes overloaded and others under loaded. The algorithms which help to transfer jobs from heavily loaded nodes to lightly loaded nodes are known as load balancing algorithms.

II. FORMAL DESCRIPTION OF THE PROPOSED ALGORITHM

There is a need for various load balancing algorithms to improve the performance of nodes in distributed systems. In this research paper, we propose a new load balancing algorithm to distribute the load among nodes so that all nodes will be well balanced.
In this algorithm, we consider the following things:

I) Jobs allocated on the basis of memory requirement and processing speed.
II) Clusters are defined on the basis of memory partitions.

Following are the various terms defined:

A) About Job Requirements:

J_i: Incoming Job ID
J_{im}: Incoming Job Memory Requirement
J_{ip}: Incoming Job Processing Speed Requirement
J_{is}: Job Status
where i = 0 to n (n is total number of jobs)

B) About Servers (Total Number of Servers Say k=6)

S_k: Server ID
S_{ck}: Server ID in cluster 'C'
S_{ckm}: Server having maximum memory
S_{ckp}: Server having maximum processing speed
S_{ckl}: Memory left after job being allocated
S_{ckj}: Job Assigned to server

C) About Clusters (Total number of clusters say L=2)

C_L: Cluster ID

C_{Lrs}: Intial value of Range
C_{Lri}: Last value of Range

Initial Information:

A) Total number of incoming Jobs
0 to n (n=7 and 0 means no job in the queue)
B) Total number of servers available is k = 6
C) Defining Range of Clusters (L =2) on the basis of memory partitions are as follows:

Cluster 1	***Cluster 2***
C_{1RS} = 0	C_{2RS}=1001
C_{1RL} = 1000	C_{2RL}=2000

D) For current server:
j=1 to count
Count is same as k

Next step is to allocate the jobs on the basis of memory and processing speed requirements. Firstly, job's memory requirement is compared with cluster's range and then job is assigned to particular server under that cluster in round robin manner.

Step 1: Assigning Jobs to servers in the clusters:

Initially, following is the status:

J_i = 1 to n (where n = 7)
T_i = 1 to count (where count = 6)
S_k = 1 to k (where k = 6)
C_L = 1 to l (where L = 2)
J_{is} = "Unallocated"
Now for allocating jobs, first Cluster is selected:

Condition 1: (Selecting Cluster)

if (J_{im}> C_{LRS}) AND (J_{im}< = C_{LRL})
Then
Select Cluster C_L
Else
C_L =+1 (move to next cluster)
End if
Now select Server under selected Cluster.

Condition 2: (Selecting Web Server)

if (J_{im}< = S_{ckm}) AND (J_{ip}< = S_{ckp})
then
i) Select server :S_{ck}
ii) Calculate memory left over:
S_{ckl}=S_{ckm}-J_{im}
iii) Find status of the server :
if (S_{ckl}< 0)
then
S_{cks} = "Uneven"
else
S_{cks}="Even"
iv) Job is assigned: S_{ckj}=J_i
v) Change job status: J_{is}="Allocated"
vi) Else (from iii)
J=+1 AND k=+1 AND check with condition i) and ii)
Until satisfied

Finally, when current job is allocated to any server, i = +1 (move to next job). If current job do not get the server, it is queued and checked once again later on.

Step 2: Load Balancing (LB)

Once jobs are allocated, some web servers might be unbalanced. Thus next step is to balance the load among web servers. For doing LB, following things will be considered:

i) Status of the server i.e. uneven
ii) Servers with no load

Basic Logic:

Step 1: Check server's status as 'Uneven'

Step 2: To make status even, LB is done

a) Within the cluster
b) Between Clusters

Step 3: If at any point of time, status cannot be changed from UNEVEN to EVEN, then

a) Job would be waited in Queue
b) When the respected server would be free, re-introduce the job.

Initially,

a) S_k=1 to k // Server ID, k = 6
b) i = 0 to n // Jobs, n = 7
c) C_L = 1 to L // L = 2
d) J = 1 to count // count is jobs allocated to server

ALGORITHM:

Step 1: a) J_i = 0 to n // n = 7 for jobs
J = 1
S_k = 1 to k

C_L = 1 to L
b)IfS_{cks} = 'UNEVEN'
Then
J = S_{ckj}// j contains all the allocated jobs of servers

Step 2: a) Calculate left over memory after removing jobs for 'j'

J_i = j
$S_{ckl} = S_{ckm} + J_{im}$

b) If (S_{ckl}> 0)
Then
S_{cks} = "EVEN"
J_{is} = "UNALLOCATED"
Goto step 3(a)
k = +1
goto step 1(b)
else if (S_{ckL}< 0)
then
j = +1
repeat step 2(a)
else if (S_{ckL} == 0)
k = +1
goto step 1(b)

Step 3: a) k = +1 // checking for other servers

b) if (k <=max.server in a cluster)
then
if ([J_{im}<= S_{ckm}] AND [J_{in}<=S_{ckp}])
then
goto step 3(c)
else
goto step 3 (a)
else
C_l= +1 // move to next clusters
K = 1
endif
Goto step 3(b)

c) If (S_{cks} == "UNEVEN")
Then
Goto step 3(a)
Else
$S_{ckl}=S_{ckm} – J_{im}$
If (S_{ckl}< 0)
Then
$S_{ckl} = S_{ckm} + J_{im}$
Goto step 3(a)
Else
$S_{ckj} = + J_i$ // Adding jobs to allocated list of servers
J_{is} = "ALLOCATED"

III. EXPERIMENTAL RESULTS AND ANALYSIS

The results of the experiments are shown below. Figure 1 presents how the average execution time varied with the total load of the system. Thus it is shown that with the increase in total number of jobs, execution time is also increased at consistent rate. Even if we have more number of jobs, then also execution time rate would be same.

TABLE 1
EXECUTION TIME W.R.T TOTAL NUMBER OF JOBS

Total Number of Jobs	Execution Time (in Secs)
3	3.73
5	7.36
7	9.72
10	11.09

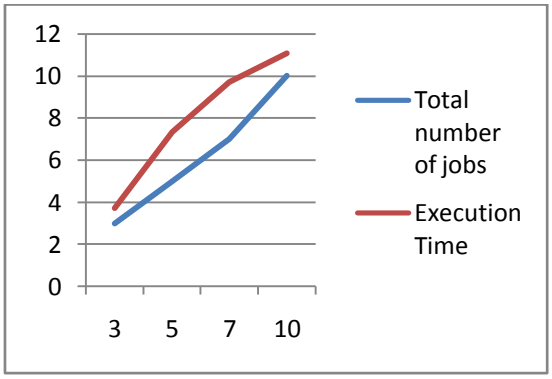

Fig 1 Performance of Load Balancing Approach

IV. CONCLUSION AND FUTURE WORK

We proposed a framework for load balancing in heterogeneous web server clusters. Load is distributed on the basis of memory and processing requirements. Preliminary evaluation reveals that use of this algorithm is necessary to improve the performance of web servers by proper resource utilization and reducing the mean response time by distributing the workload evenly among the servers in the cluster.
In future, the proposed algorithm will be compared with others to prove that it is better than all existing algos.

REFERENCES

[1] KanungoPriyesh (2013), Scheduling Algorithms in Web Server Clusters, InternationalJournal of Computer Science and Mobile Computing ISSN 2320–088X, IJCSMC, Vol. 2, Issue. 10, October 2013, pg.78 – 85

[2] KanungoPriyesh (2013), Study of Server Load Balancing Techniques, International Journal of Computer Science & Engineering Technology (IJCSET), ISSN : 2229-3345 Vol. 4 No. 11 Nov 2013

[3] Yousofi Ahmad, Banitabamostafa and YazdanpanahSaeed (2011), A Novel Method for Achieving Load Balancing in Web Clusters Based on Congestion Control and Cost Reduction, IEEE Symopsium on Computers & Informatics, IEEE 2011

[4] MahmoodAmjad and Rashid Irfan (2011), Comparison of Load Balancing Algorithms for Clustered Web Servers, Proceedings of the 5th International Conference on IT & Multimedia at UNITEN (ICIMU 2011) Malaysia, 14-16 November 2011

[5] Ho Lai Kuen et al (2004), Improving web server Performance by a clustering-based Dynamic Load Balancing Algorithm, Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04), IEEE 2004

[6] Ling Yibei, Chen Shigang, Lin Xiaola (2003), Towards Better Peformance Measurement of Web Servers, ICICS-PCM, December 2003

[7] Choi Min et al (2003), Improving Performance of Load Balancing System by Using Number of Effective Tasks, Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'03)

[8] KrawczykHenryk, UrbaniakArkadiusz (2002), Allocation Strategies of User Requests in Web Server Clusters, Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC'02)

[9] SongnianZlon, "Performance Studies of Dynamic Load Balancing in Distributed Systems", Dissertation, 1987.