IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-3 E-ISSN: 2347-2693

Mapreduce- A Fabric Clustered Approach to Equilibrate the Load

Deepti Sharma[1*] and Vijay B. Aggarwal[2]

[1*] Dept. of Information Technology, Jagan Institute of Management Studies, GGSIPU, Rohini, Delhi, India

[2.] DIT, JIMS, Rohini, Delhi, India

www.ijcseonline.org

Received: Feb/26/2016 Revised: Mar/04/2016 Accepted: Mar/19/2016 Published: Mar/31/2016

Abstract- **In recent years, load balancing is the challenging task which affects the performance in allotting the resources on homogeneous and heterogeneous cluster computing environment. This research proposes an enhancement in ACCS (Adaptively Circulates job among all servers by taking account of both Client activity and System load) policies by incorporating Map Reduce to overcome the problem in balancing the workload for resources. This technique provides simplicity and flexibility for data partitioning, localization and processing jobs as indicated by their present sizes and ranks the servers based on their loads by giving high priority to the smaller jobs. Map Reduce emphasizes more on high throughput of data on low-latency of job execution in a cluster to accomplish huge execution advantages. Trace driven simulations demonstrate the viability and robustness of Map Reduce under numerous different situations.**

Keywords: **Load Balancing, Map Reduce, Web Server Clusters, AdaptLoad, ACCS**

I. INTRODUCTION

Cluster computing is the next generation computing in the field of distributed and parallel processing. Various load balancing approaches such as Join Shortest Queue (JSQ) policy in [1] and Locality Aware Request Distribution (LARD) [2] are broadly executed in real system because of their simplicity and effectiveness. The execution advantages of these strategies are reduced when workloads are highly variable and transiently related. The illustrations of these frameworks incorporate High Performance Computing (HPC) that endeavors data frameworks, cloud computing and large scale cluster servers[3].

The best fit descending algorithm in ordering the agents is used to avoid the backing problems but it lags in balancing the load when the resource leakage increases [4]. In such cases multi-server cluster uses a front-end dispatcher for dispatching the incoming jobs to the back-end servers. Such a front-end dispatcher works using the First-Come First-Serve (FCFS) queuing order [5] and [6]. Join Shortest Queue (JSQ) has been an optimal solution for a cluster with homogeneous servers when there is no prior knowledge about the job size. JSQ also gives optimal result when the job sizes are exponentially distributed [7]. The optimality of JSQ rapidly vanishes when the job service times is highly variable and substantially tailed.

The proposed load balancing technique inculcated in ACCS (Adaptively Circulates the job among all servers by taking account of Client activity and System load) is using Map Reduce. Map Reduce is a programming model proposed to process a large number of datasets in a cluster. To achieve simplicity and effectiveness in load balancing, Map Reduce handles all parallel and distributed computing issues in ACCS to overcome the computational overhead.

Section 2 gives a review of load balancing in web server clusters framework. Section 3 presents the proposed load balancing technique Map Reduced ACCS which aims at parallelizing the jobs distribution among all servers by taking account of both client activity and system load. Section 4 depicts the trace driven simulations to evaluate the performance of the proposed strategy. Conclusions and future work are summarized in Section 5.

II. REVIEW ON LOAD BALANCING

The size based polices previously used to balance load in a web cluster framework accomplishes the goal of minimum job response time and job slow down [8] and [9]. The AdaptLoad approach created in enhancing average job response time and average job slow down for distributing similar sized jobs to the same server [10]. The principle target of load balancing is Maximizing Utilization of Resources (Like CPU), Minimizing Response Time, Minimize Inter Processor Communication Overhead and keeps the load balanced. Load balancing is the methodology of reassigning the aggregate loads to the individual hubs of the collective framework to make the best response time and also great

use of the resources are detailed in [11]. In [12] framed on-line load balancing using Greedy strikes back. It concentrated on the on-line load balancing issue on account of temporary tasks with limited task and no pre-emption. The load sharing issue among heterogeneous cluster frameworks considers time-offering, and the PCs in these cluster have distinctive CPU powers and memory limits.

Dynamic load balancing in [14] have the capability of performing better than static techniques, they are unavoidably more intricate. Load balancing was found to decrease essentially the mean and standard deviation of job reaction times, particularly under overwhelming on unequal workload. This methodology decreased the waiting time by significant measure of time.In [15] proposed the queuing management and load balancing for the on demand connectivity sharing. Their primary objective was to model the effect of the visitor client's vicinity on the home- user for distinctive extents of traffic demand. In light of that, they restricted the guest user's packet arriving rate (k_2) to a value that has irrelevant effect on the home-user.

III. PROPOSED METHODOLOGY

In the proposed methodology a new technique Map Reduce is implemented in addition to the above load balancing mechanism, aiming to inherit the effectiveness of ACCS and to overcome the limitations of both adapt load and ACCS policies. The main significance of Map Reduce is that it gives equal priority to both the short and long jobs.

A. Map Reduce Framework

The processing takes a set of data key and value sets combined and produces a set of output key and value sets. The calculation includes two fundamental operations: Map and Reduce. The Map operation takes the input pair which was composed by the client and produces a set of intermediate key and value sets. The Map Reduce library cluster combines it altogether which are connected with the same intermediate key and pass the data's to the reduce function. The reduce function acknowledges with an intermediate key along with set of qualities for that key. It also consolidates the result with the qualities to structure a conceivably smaller set of qualities. Typically just an output value of 0 or 1 is delivered in one reduce invocation. The intermediate values are supplied to the client's through an iterator based on reduce capacity (a protest that permits a developer to navigate through all the components of an accumulation paying little mind to its particular usage). This permits the client to handle arrangements of values substantial and it would be impossible to fit in the memory. The frame work of the Map Reduce function is shown in Figure 1.

B. Map Reduce Algorithm

The Description of the Map Reduce algorithm in load balancing is discussed below:

1. The web clusters can be divided into n number of chunks depending upon the amount of data and processing capacity of individual unit.
2. Next, it is passed to the mapper functions. Map (key = K_1, value = V_1); for each $K_i \in$ P do; foreachV_i = (V_1 , V_2 , …, aV_n), Where Kp , ar∈Vi ∧ar ≠ as∀ 1 ≤ r, s ≤ Kc. Here, P = partition of work; kc = minimum number of common keys
3. Once all the clusters are iteratedthe output is processed simultaneously at the same time, which embraces the parallel processing of data.
4. After shuffling the output, Kb is a kc-combination from V_i and then sorted with the aggregate values.
5. Finally, reducers combine them all to get a consolidated output. Reduce (key = K_i , value = V_i); for each key Kb do; foreach v in the list of valueslogic.

This algorithm embraces scalability as depending on the size of the input data, it keeps on increasing the number of the parallel processing units.

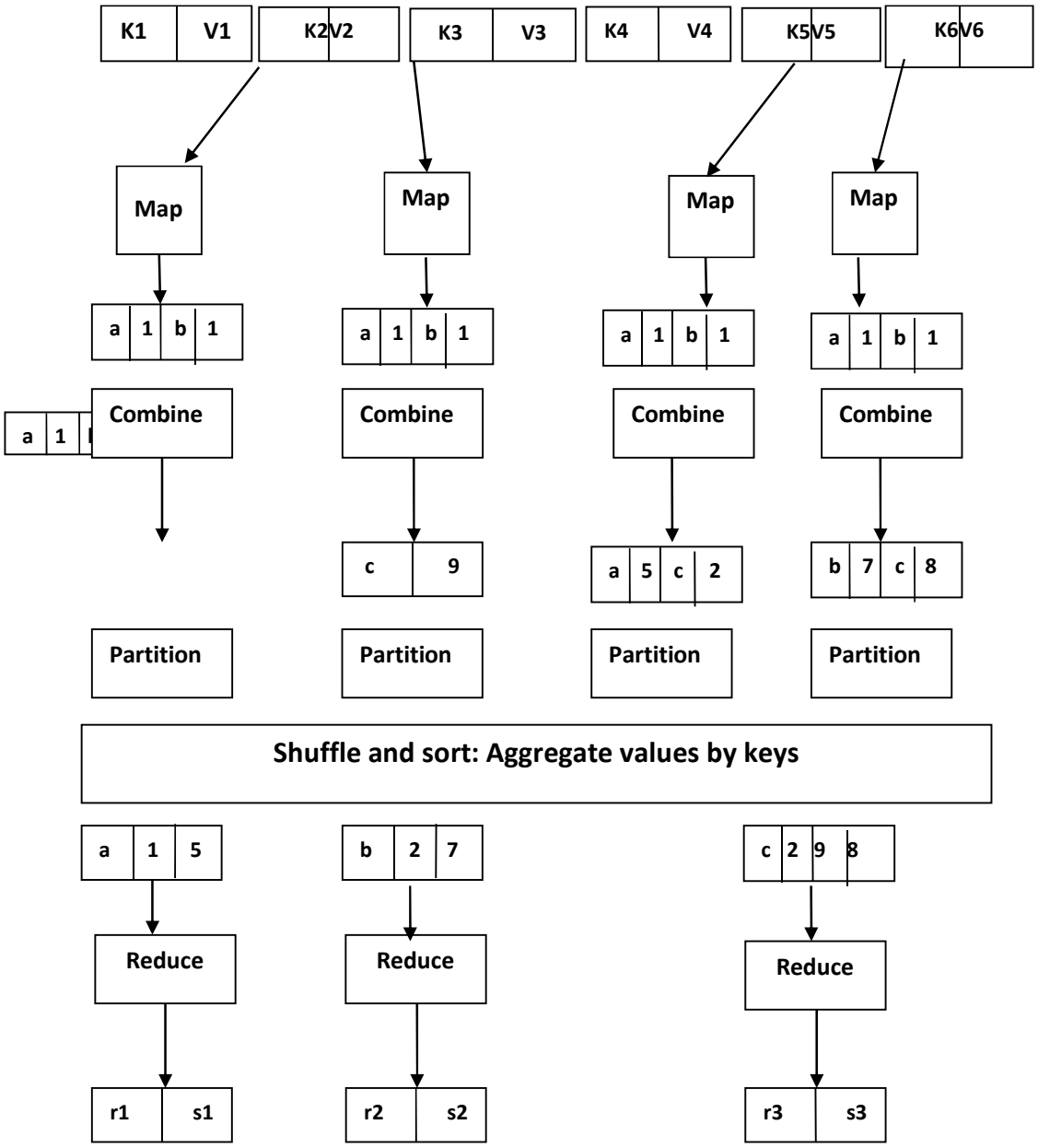

Figure-1 Map Reduce Framework

C. Map Reduce on ACCS Policies

ACCS occasionally positions all servers which are focused around their present system loads, e.g., server use or weighted line lengths and continues sending the approaching jobs of comparable sizes to a server with the same ranking rather than the same server which may be over-burden by the past arrived jobs. The working procedure of the Map Reduce algorithm is shown below

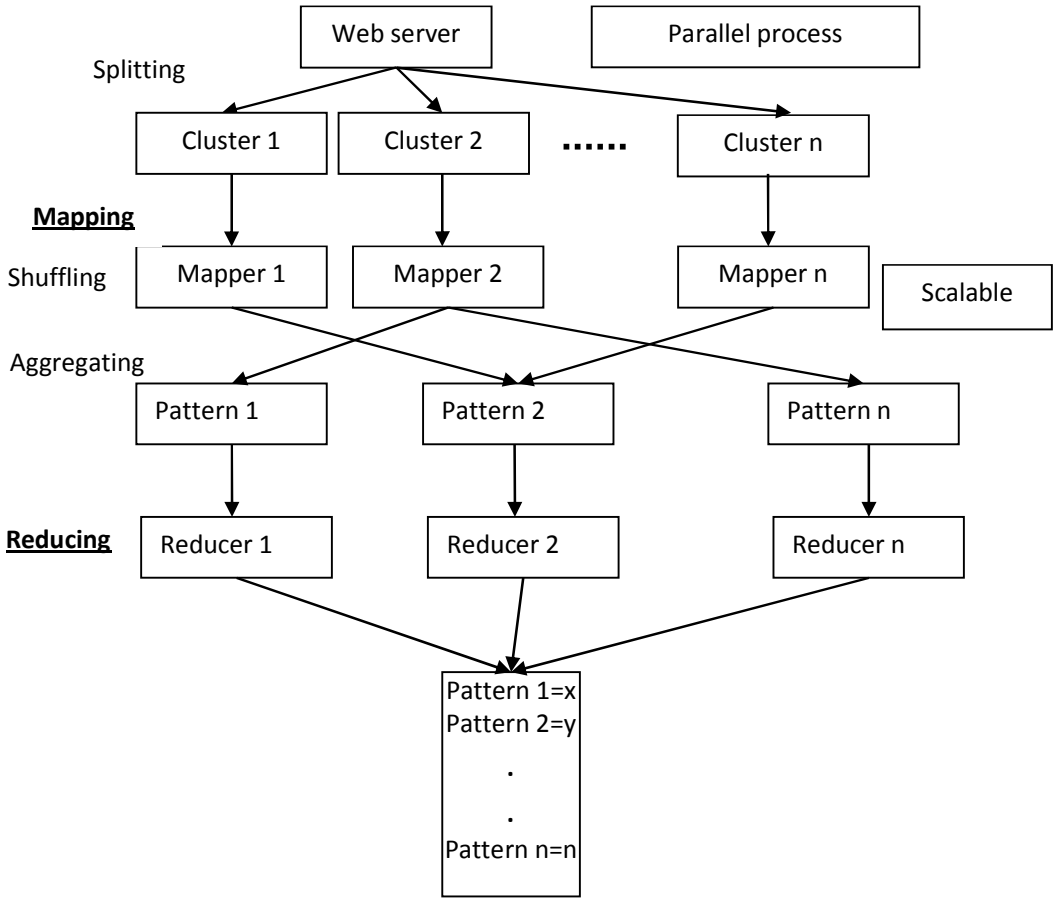

Figure-2 ACCS load balancing using Map Reduce

1. Initially the web server splits the process with clusters. The priority list of web servers, W'={W_1' ..... W_N'}; and the clusters of $c = c_1, c_{2...}c_{n-1.}$

2. Cluster sizzle boundaries: $[(0,c_1),[c_1,c_2)\ldots..c_{n-1,}\ \alpha]$; clusters the job; then Sort all N servers in increasing order and update priority list W';

3. Cluster boundaries are updated such that work is equally distributed into N areas;

4. Map, separate out clusters to be processed to balance load. Then the reducer analyzes the cluster data.

5. For each arriving job the job size $\in [c_{i-1}, ci)$ direct this job to the server $W_{i.}$

The instruction progressively changes the cluster sizzle the boundaries C which is processed by using Map Reduce technique to solve the load balancing problem. So now the ACCS will circulates the job equally to all its serverswith reference to the account of both client activity and system load, Map Reduce helps in enhancing the performance by overcoming critical and computational overhead and also in balancing the load in the web server clusters.

D. The Execution Framework

The Map Reduce functions are distributed across web server clusters automatically by splitting the input data into a set of mapper M. The inputs are processed in parallel on different machines. Now the Reduce functions are distributed by partitioning the intermediate key/values into R pieces using a partitioning function (e.g hash (key) mod R). The number of partitions R and the partition functions are specified by the client. The working procedure of Map Reduce operation used for implementation is shown in Figure 3.

1. The Map Reduce library in the web cluster splits the input files initially into M pieces generally 16 megabytes

to 64 megabytes (MB) per piece. Then it starts up many copies of the program on the cluster.

2. Among the copies a single copy is considered as the master and the rest are considered as workers. The work was assigned by the master to the workers. There are M map tasks and R reduce tasks that are ready to assign. The master picks idle workers and assigns a map task or a reduce task to that worker.

3. The loaded worker assigned for a map task reads the contents of the corresponding input split. It parses the key/value pairs out of the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are stored in the memory.

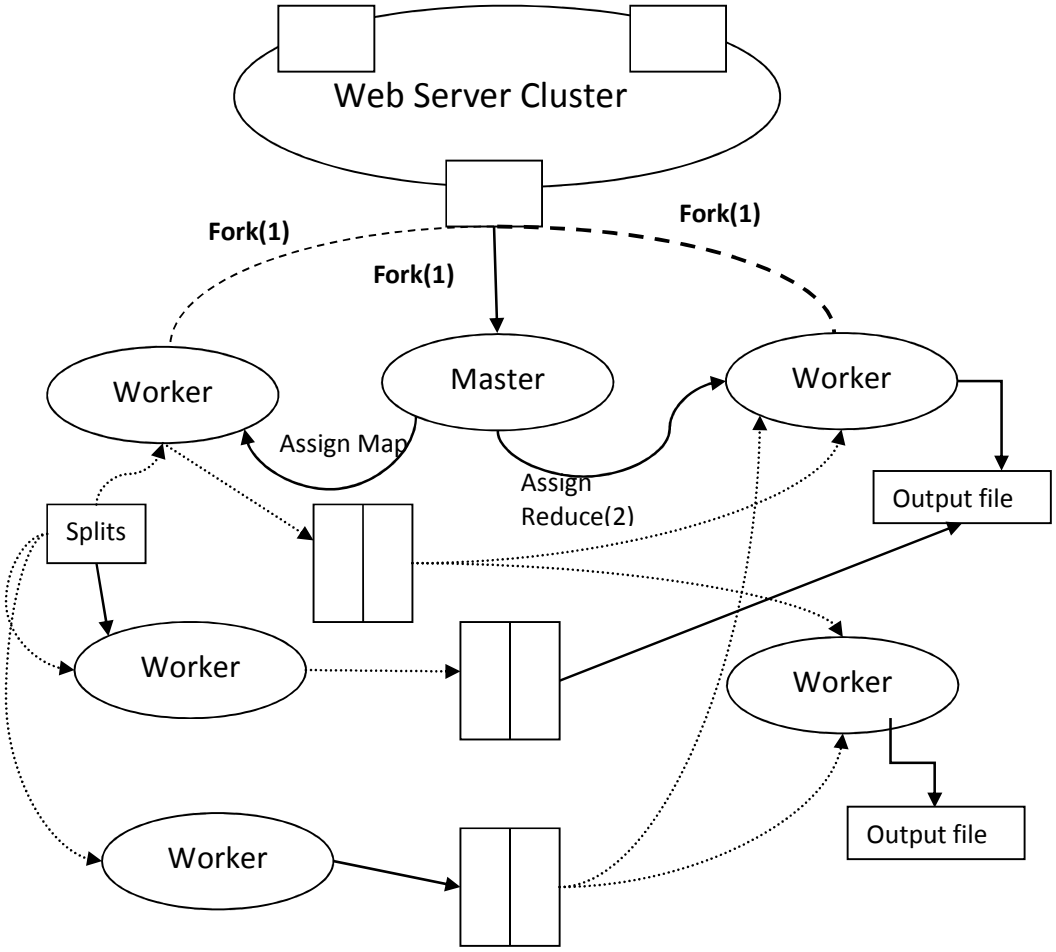

Figure-3 Execution framework of Map Reduce

4. Periodically, the buffered pairs are written on the localdisk which, partitioned into R regions by the partitioningfunction. The locations of these buffered pairs onthe local disk are forwarded back to the responsible location of the master, in order to reduce the workers.

5. When a reduced worker is notified by the master along with its locations, it uses Remote Procedure Calls (RPC) to read the buffered data from the local disks of the map workers. When a reduce worker has read all its intermediate data, it sorts it by the intermediate key to all occurrences of the same key can be grouped together. The sorting is needed because typically many different keys map to the same reduce task. If the amount of cluster data is too large to fit in memory, an external sort is used.

6. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's Reduce function. The

output of the Reduce function is appended to a final output file for these reduce partition.

7. When all map task and reduce task are completed, the master wakes up the web cluster. At this point, the Map Reduce call in the user program returns back to the user code. After successful completion, the output of the Map Reduce execution is available in the R output files. Typically, it is able to deal with input that is partitioned into multiple files.

IV. RESULTS AND DISCUSSIONS

The viability and robustness of ACCS using Map Reduce technique is discussed in this section. Different load balancing calculations are executed by a load dispatcher which is in charge of distributing the entries to one of N servers. For all simulations, the details of job incorporate job entry times and job sizes, which are created, focused around the defined appropriations. The used processing rates of the N servers is $\mu_{p=}1$.

A. *Performance improvement of Map Reduce*

The job inter-arrival times are exponentially distributed with mean λ^{-1} =0.7.The job sizes (i.e., the service process) are drawn from a MMPP Markov-Modulated Poisson Process (full form) (2) with the mean equal to μ^{-1}=1, Squared Coefficient of Variation equal to SCV =20, and Autocorrelation Function (ACF) at lag 1 equal to 0.5. Therefore, high variability and temporal dependence are injected into the workload, i.e., the service process. Also, consider N = 4 for homogeneous server nodes in the cluster such that the average utilization levels at each server node are ρ= 50%. The mean job response times (i.e., the summation of waiting times in the queue and service times) and the mean job slow down are measured the individual performance, where T= μ^{-1}(1+2CV) is used for the job size classification.

Let us classify the total number of jobs based on 4 categories like Small jobs (50 – 150 jobs), Medium small (150 - 250), Medium large (250 - 350) and Large jobs (350 - 500). Let N be the number of jobs. The performance is calculated based on [16] four measures like response time, workload, delay and number of jobs. The Map Reduce ACCS allocates the resources based on job sizes and it gives equal priority to the smallest jobs. The response time for small jobs is high when compared with the larger jobs. The system response time for map reduce ACCS outperforms with 58% and 66% is slow compared with the performance of ACCS and Adapt Load. The observation under Map Reduce in ACCS is that about 99.4% of jobs experience faster response times and maximum jobs have smallest slowdowns. Such performance for system response time is shown in Figure 4.

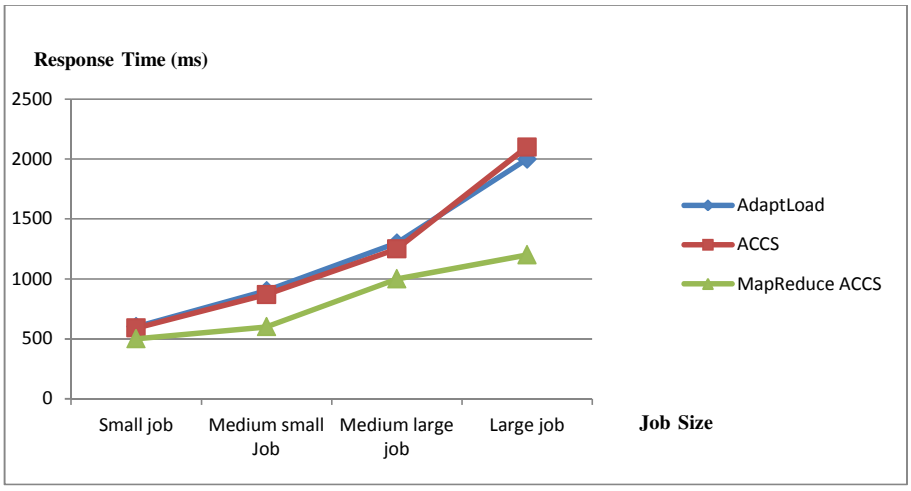

Figure-4 System Response Time

B. *Sensitivity to Job slow down*

The job slowdown of data taken is calculated in successive 60 seconds interval. Map Reduce ACCS has less slowdown performance under various network sizes.The number of servers in a cluster are N =4, 16 and 32. Also, keep other parameters same such as the job arrival rates, in order to maintain the 50% utilization level on each server. As a result, Map Reduce in the ACCS is the best policy with a clear improvement under

three different network sizes. The robustness of Map Reduce under different experimental parameter settings is evaluated using job slowdown in Figure5. Map Reduce achieves the best performance (i.e., fastest response times and smallest slowdown).ACCS improves the performance by around 50%. As the load increases to 80%, the performance improvement under ACCS becomes more visible and significant. Furthermore, such a long tail becomes shorter and finally disappears as the system becomes heavy loaded.

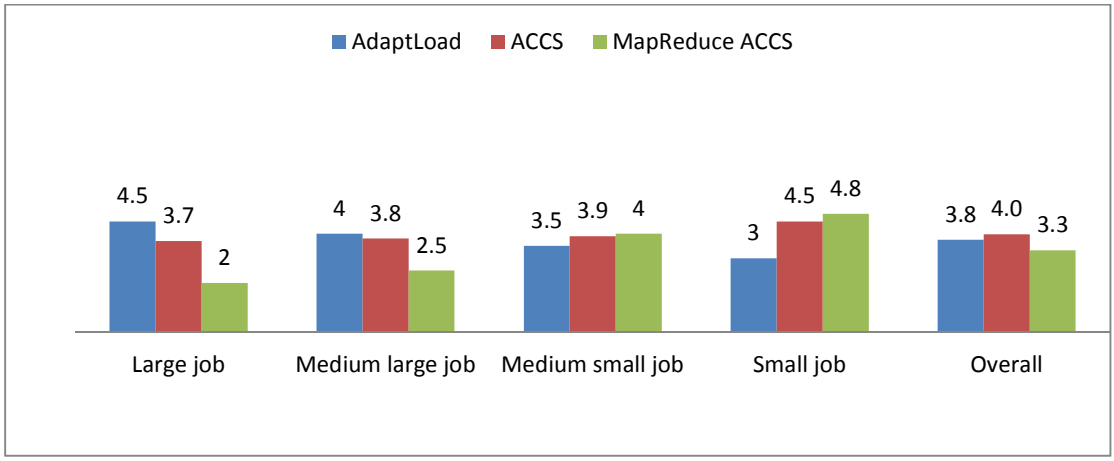

Figure-5 Job slowdown

V. CONCLUSION

The proposed load balancing technique, ACCS Policies on Map Reduce outperforms in dispersing the work in the framework by making note of both client activity and system load. It improves general framework execution by acquiring the viability of ACCS. Utilizing trace driven simulations on engineered and genuine acquires the adequacy of ACCS and size-based strategies and then overcome their impediments which brings about critical execution and computational overhead. Map Reduce can rapidly adjust to the workload changes by checking client activity and system load. Extensive experimental results show that Map Reduce essentially enhances system performance, e.g., job response time and job slowdowns, under heavy tailed and temporal dependent workloads additionally overcoming computational overhead. Later on, refine new load adjusting calculation such that it can accommodate toward oneself its parameters (e.g., the window size) to transient workload conditions. In future the proposed framework on heterogeneous servers is expected that the usage of ACCS along with Map Reduce will give a straightforward yet successful methodology for asset designation in extensive web server cluster process.

REFERENCES

[1] Gupta, V., Balter, M. H., Sigman, K., & Whitt, W. (2007). *Analysis of join-the-shortest-queue routing for web server farms*. *Performance Evaluation*,*64*(9), 1062-1081.

[2] Pai. V. S., Aron, M., Banga, G., Svendsen, M., Druschel, P., Zwaenepoel, W., & Nahum, E. (1998, October). *Locality-aware request distribution in cluster-based network servers.*In*ACM Sigplan Notices* (Vol. 33, No. 11, pp. 205-216).ACM.

[3]Teo, Y. M., &Ayani, R. (2001). Comparison of load balancing strategies on cluster-based web servers. *Simulation*, *77*(5-6), 185-195.

[4]Alonso-Calvo, R., Crespo, J., Garc'ia-Remesal, M., Anguita, A., &Maojo, V. (2010). On distributing load in cloud computing: A real application for very-large image datasets. *Procedia Computer Science*, *1*(1), 2669-2677.

[5]Feng, H., Misra, V., & Rubenstein, D. (2005). Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems. *Performance evaluation*,*62*(1), 475-492.

[6]Harchol-Balter, M., & Downey, A. B. (1997). Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems (TOCS)*, *15*(3), 253-285.

[7]Winston, W. (1977). Optimality of the shortest line discipline. *Journal of Applied Probability*, 181-189.

[8]Bonomi, F. (1990). On job assignment for a parallel system of processor sharing queues. *Computers, IEEE Transactions on*, *39*(7), 858-869.

[9]Bachmat, E., &Sarfati, H. (2010). Analysis of SITA policies. *Performance Evaluation*, *67*(2), 102-120.

[10]Riska, A., Sun, W., Smirni, E., &Ciardo, G. (2002). ADAPTLOAD: effective balancing in clustered web servers under transient load conditions. In*Distributed Computing Systems, 2002.Proceedings. 22nd International Conference on* (pp. 104-111). IEEE.

[11]Luis, A., &Azer, B. (2000). Load balancing a cluster of web servers. In*Proceedings of IEEE International Performance, Computing, and Communications Conference (IPCCC" 00), ISBN: 0-7803-5979-*

[12]Crescenzi, P., Gambosi, G., Nicosia, G., Penna, P., & Unger, W. (2007). On-line load balancing made simple: Greedy strikes back. *Journal of Discrete Algorithms*, *5*(1), 162-175.

[13]Niu, Y., Chen, H., Hsu, F., Wang, Y. M., & Ma, M. (2007, February). A Quantitative Study of Forum Spamming Using Context-based Analysis.In*NDSS*.

[14]Garg, A. (2015). A Framework to Optimize Load Balancing to Improve the Performance of Distributed Systems. *International Journal of Computer Applications*, *122*(15).

[15]Psaras, I., &Mamatas, L. (2011). On demand connectivity sharing: Queuing management and load balancing for user-provided networks. *Computer Networks*, *55*(2), 399-414.

[16] Gupta, R. K., & Ahmad, J. (2014). Dynamic Load Balancing By Scheduling In Computational Grid System. *Computer Engineering and Intelligent Systems*, *5*(6), 39-45.

[17] Ungureanu, V., Melamed, B., &Katehakis, M. (2008). Effective load balancing for cluster-based servers employing job preemption. *Performance Evaluation*, *65*(8), 606-622.

Author Profile

Ms. Deepti Sharma is an Asst. Professor in Department of Computer Science at Jagan Institute of Management Studies, Rohini, Delhi. She is MPhil, MCA and pursuing her PhD in Computer Science from IGNOU. She has more than 12 years of teaching experience. Her research areas include "Load Balancing in Heterogeneous Web Server Clusters", Big Data Analytics, Distributed Systems and Mobile Banking on which papers have been published in National and International conferences and journals. Various seminars, workshops and AICTE sponsored FDP have been attended.

Dr. V.B. Aggarwal was awarded Ph.D Degree by University of Illinois in USA in 1973 for his research work in the areas of Super Computers, Array Processors, Cray XMP and Data Base Management Systems. He has been faculty member of Computer Science Deptt at Colorado State University and University of Vermont in USA. Dr. V.B. Aggarwal has been Head & Professor of Computer Science at University of Delhi and Professor at Dept of Electrical Engg and Computer Science at University of Oklahoma, USA. Currently he is Dean (Infotech), DIT, JIMS, Rohini, Delhi. In 2001 Dr. V.B. Aggarwal was elected to the prestigious office of Chairman, Delhi Chapter, Computer Society of India. He has been associated as a computer subject Expert with NCERT, CBSE, AICTE and Sikkim Govt Technical Education Department. Presently he has been nominated as Computer Subject Expert in Academic Council of Guru Govind Singh Indraprastha University in Delhi. Prof. V.B. Aggarwal has authored more than 20 Computer Publications which are very popular among the students of schools, Colleges and Institutes.

