International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

Providing Enhanced Resource Management Framework for Cloud Storage

S. Magesh Kumar, S. Ashokkumar, A. Balasundaram

Abstract: ***Data centers are progressively being re-intended for workload combination with a specific end goal to receive the rewards of better resource usage, control cost, and physical space investment cost. Among the strengths driving costs are server and storage virtualization innovations. A key understanding is that there is a more noteworthy cooperative energy between the two layers of storage and server virtualization to be application piece sharing data than was beforehand thought conceivable. In this segment, we display ERMF, a platform that is intended to have MapReduce applications in virtualized cost. ERMF gives a bunch file framework that backings a uniform record framework name-space over the group by coordinating the discrete nearby storage of the individual hubs. Our paper proposes ERMF accommodates the two data and VM resource assignment with contending requirements, for example, storage usage, changing CPU load and system connect limits. ERMF utilizes a stream arrange based calculation that can improve MapReduce performance under the predetermined limitations by starting situation, as well as by straightening out through VM and data relocation also. Moreover, ERMF uncovered, generally shrouded, bring down level topology data to the MapReduce work scheduler with the goal that it makes close ideal task scheduling.***

Keywords: Virtual Machine, Enhanced Resource Management Framework, De-Duplication

I. INTRODUCTION

Cloud resource management [1] filling in as the center empowering innovation for distributed computing organizes resources from a substantial number of PCs and presents a uniform view to clients and applications. It support functionalities, for example, supporting a remote and secure interface for making, decimating, designing and checking virtual resources, progressively managing resources, giving configurable resource designation strategies, flexibly provisioning resources in view of organization' needs [2]. In any case, fabricating a versatile cloud resource management framework while meeting the prerequisites of adaptability, execution detachment, productivity is challenging. Cloud management frameworks[14], for example, Amazon EC2 [3] have low machine use utilization because of the low combination proportion by statically mapping one VM to a deployed number of physical CPUs and memories. Also, cloud suppliers enable clients to control resource request for through parameter design.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
S. Magesh Kumar*, Associate Professor, Department of CSE, Saveetha School of Engineering, mmce6450@gmail.com
S. Ashokkumar, Assistant Professor, Department of CSE, Saveetha School of Engineering, sabariashok2016@gmail.com
A. Balasundaram, Assistant Professor, Department of CSE, Saveetha School of Engineering, balasundaram2682@gmail.com

It is imperative to help clients in effortlessly issuing exact resource demands without additional endeavors. At long last, as the size of bunch builds, the present resource management procedures can scarcely get up to speed with the scale speed. The unified storage framework as a rule restrains the quantity of VMs conveyed in the cloud. Along these lines, it is basic for cloud serviceprovider to configuration cloud management frameworks and devices that offer versatility, superior, unwavering quality, accessibility and security for fast and wide reception of cloud managements. The resource management of MapReduce in the cloud requires the coordinated effort amongst clients and MapReduce serviceprovider. From the suppliers' viewpoint, they ought to have the capacity to put the information and VM viably to avoid unnecessary system movement and assurance proficient resource use; from the clients' viewpoint, they have to give precise resource demands which normally rely upon application attributes.
The rest of the chapter as follows: Section 2 summarizes the related work for cloud computing resource management. In section 3, ERMF framework has proposed. Section 4demonstrates results and discussion of proposed work and Conclusion is presented with Section 5.

II. RELATED WORK

2.1 Challenges of Predicting Application IaaS

Exact forecast of application nature of management would spare tremendous measures of cash for cloud clients, however is hard to fulfill. Issues, for example, multi-occupancy and delicate resource limits entangle the challenge of producing usable data when running on an IaaS cloud. Customary models of application execution profiling, concentrate on one metric of programming execution, for example, add up to runtime Tp, as figured utilizing the recurrence of programming execution Hi and the runtime of individual programming pieces Ti:
Relative Resource Limits: Cloud IaaS suppliers, for example, Amazon customarily utilize relative, or delicate, resource constrains that permit a VM to devour extra resources past it's base stipend if those resources are unused by different inhabitants. As the Cloud IaaS supplier can't reapplication unused calculation time, the basic practice is to permit unused calculation time to be devoured by the inhabitants executing on a host. This is specifically relatable to frameworks, for example, the Linux Completely Fair Scheduler (CFS), which will share any unused cycles between forms presently anticipating calculation. This makes a generous weight for execution profiling, as it winds up noticeably vague if benchmark comes about incorporate any additional process time that was unused by different inhabitants.

In addition, this additional instability expands the difficulty of distinguishing distinctive classes of blunder introduce while benchmarking more than once, along these lines making it hard to detach and dispose of reasons for benchmarking clamor.

For instance, it has been demonstrated that indistinguishably indicated VMs as of now have different wellsprings of changeability [4].

Splitting Processing Capacity: A new challenge of LXC IaaS is the need to reasonably share a host PC among various occupants likely surpassing the quantity of accessible equipment strings. Customary instruments for genuinely sharing accessible processor limit in IaaS concentrated on varieties of CPU sticking (otherwise called processor partiality), whereby a procedure or gathering of procedures is executed exclusively on assigned equipment processors. CPU sticking is exceptionally powerful on the grounds that 1) processor reserves are more compelling when forms execute on a similar CPU, and 2) CPU sticking does not require worldwide process utilization counters and in this manner requires no coordination between various processors. Be that as it may, the unit of division utilized as a part of CPU sticking is dependably a solitary equipment processor, for example, a center or equipment string, and in this manner CPU sticking can't ensure measure up to sharing of numerous procedures executing on a similar equipment processor.

Benchmark-based Performance Models: Multi-inhabitant have frameworks are characteristically testing to gather benchmark data from, as there is an unmistakable potential for obstruction. Amazon EC2 right now imparts each physical machine to up to eight visitor VMs. While resource limits exist to decently subdivide the host machine, earlier work has demonstrated that these resource limits are not upheld with supreme strictness, which can bring about more awful benchmark comes about if a neighbor is being uproarious.

Cooperative Caching: Cooperative storing systems concentrate on outlining effective expulsion calculations and meta-data [14] files to total appropriated customer reserves as a brought together store and to encourage quick query. N-chance sending [5] allots more weight to singlets that have just a single duplicate of data in the reserve by sending singlets to irregular associates. LAC [6] advances the ousted data square to peers in view of data reapplication separation and dynamic customer synchronization. In[2] plans an area mindful conveyed list to empower customers to find adjacent duplicates of data. These strategies are orthogonal to SeaCache, as SeaCache brings the de-duplication idea into the agreeable reserve and concentrates on how agreeable storing can lessen I/O transfer speed utilization. In our SMIO framework, we offer a substitute answer for relieving the I/O request to storage server [13] other than utilize agreeable store. Additionally, for the situation where a gathering of VMs have comparative I/Os for a moderately lengthy timespan period, our arrangement will kill the related system activity (at the cost of one-time relocation), which helpful store can not.

2.2 Build Effective Prediction Models

Point by point models of ensured application execution at different resource distribution levels would empower fast and exact sending of programming onto hosts. we initially built up a strategy for producing precise execution profiles for HTTP web servers running inside Linux Containers. We at that point utilized an extensive store of application of web server usage to build execution profiles for an extensive variety of web server executions, fluctuating in programming dialect, web server structure, database ORMs, and different components. The web application engineers input their coveted nature of management and either a portrayal of the application or the application itself. The technique will report the LXC dispatch alternatives the engineer should application to accomplish the coveted QoS measurements. The basic novel segment is hard-constrain benchmarking, which gives answers for different talked about difficulties.

To address the challenge of relative resource limits, and in addition the challenge of sharing CPUs at high granularity depicted, we look at the transfer speed control highlight of the Linux Completely Fair Scheduler (CFS) [7]. Generally this component is utilized for assigning relative offers of CPU that are then used to figure level of permitted CPU time. In any case, it is additionally conceivable to characterize hard resource roof parameters that enable constraining of a procedure to a particular measure of CPU time. For instance, CFS data transfer storage cutoff points can be utilized to characterize a quantity that a procedure is permitted to application. CFS transfer speed control effectively keeps away from the issue of relative transmission storage impediments causing blunder in benchmark data, as it empowers limiting a Linux Container to the more terrible case situation where no extra resource are accessible. Also, this CFS hard utmost can apparently be utilized as a part of continuous close by other LXC IaaS occupants that are utilizing the more typical relative restricting, as the main distinction in complete framework state will be that the CFS-constrained holder never again devours CPU from the pool of unused figure time. In any case, there are various handy concerns encompassing the utilization of CFS limits. For instance, CFS requires coordination between processors to decide how much calculation time a procedure has gotten, and the overhead of this coordination is indistinct right now. Turner et al. over a short examination of CFS points of confinement and CPU sticking however leave various issues.

2.3 Building Performance Profiles For Software

To address the essential challenge of foreseeing application performance, we benchmarked more noteworthy than 100different web server usage spreading over 21 distinctive programming dialects. For every usage we benchmarked a scope of CPU and RAM conditions, and utilized a straight spline interjection to decipher these settled data for applications into a direct surface. While every server usage we utilized is at first completely determined, we discovered mean outcomes for a few general classes and made unpleasant models of performance in view of insignificant data about the application.

While these unpleasant models have extreme loss of precision, they likewise offer generously better performance appraises rather than having no earlier data. Utilizing abnormal state data about their application, for example, programming dialect, the engineer would then be able to counsel our database to discover most
pessimistic scenario performance models for their application. As the designer indicates more data about their application, for example, programming dialect and web structure utilized, we can supply more precise performance models.

III. ENHANCEDRESOURCE MANAGEMENT FRAMEWORK(ERMF) FOR CLOUD

3.1 Storage-Level De-Duplication

We at that point talk about the methodologies we conceived to enhance the storage adaptability and productivity for VDEs in this and next sections. Data centers are progressively being re-intended for workload combination with a specific end goal to receive the rewards of better resource usage, control cost, and physical space investment cost. Among the strengths driving costs are server and storage virtualization innovations. As more combined workloads are focused on physical machines e.g., the virtual thickness is now high in virtual desktop situations, and will be headed to phenomenal levels with the quickly developing high-center tallies of physical servers the common storage layer must react with virtualization advancements of its own, for example, de-duplication and thin provisioning. A key understanding is that there is a more noteworthy cooperative energy between the two layers of storage and server virtualization to be application piece sharing data than was beforehand thought conceivable. In this Chapter, We uncover the cooperative energy by means of building up a methodical cache framework, Sea-Cache, to investigate the storage and virtualization servers connections. We additionally quantitatively assess the I/O data transmission and latency decrease that is conceivable between virtual machine hosts and storage servers utilizing true follow driven reproduction. In addition, we display a proof of idea NFS performance that consolidates our methods to evaluate their I/O inertness benefits.

A customary way to deal with I/O data transfer storage sparing is to not adjust the host and storage server programming stack, rather to present devoted hubs for de-duplication, i.e., de-dup boxes [8], both at the host and the storage server. These containers monitor the data obstructs through a content sharing Index (CSI) database of the considerable number of pieces they have been sent and got, and cooperate to abstain from composing various duplicates of data to the disk. A CSI section typically comprises of a piece identifier and the comparing hash esteem computed utilizing impact safe hash capacities, for example, SHA-1 [9]. Also, SeaCache expect that hash crash from SHA-1 is lower than memory bit flip blunders application of inestimable beams for all handy purposes. In the de-dup box approach, the container is a different content and it doesn't know about the host-side or storage server-side reserve content. Along these lines, read demands from various customers will dependably be sent to the storage server regardless of the possibility that the data as of now exists in the host reserve. Besides, the storage server side piece de-duplication data is not utilized, and accordingly, this data is kept up independently at the de-dup boxes and at the storage server. By incorporating host-side reserve with server-side de-duplication, we can investigate the chance to construct an agreeable I/O de-duplication arrangement amongst host and storage server.

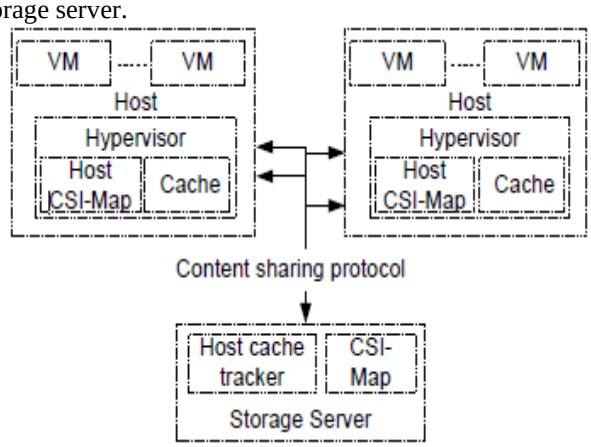

Figure 3.1 ERMF Architecture

Figure 3.1 demonstrates the general architecture of ERMF. The objective condition includes have physical machines with different customers (VMs), which collaborate with a storage server for relentless datastorage. The software components incorporate a particular page cache supervisor on the host, a de-duplication framework on the storage server, a storage server cache tracker that monitors the host store content, and a convention for sharing CSI between the hosts and the common storage server When data is composed to the storage server, it is de-copied as takes after. The contentare hashed at the granularity of a square, and the hash data is spared in the CSI data structure. The CSI is then contrasted with and, if not effectively present, put away in a CSI database. Every section of the CSI database is a tuple comprising of the intelligent piece number (LBN), and the square's hash esteem. On the off chance that an intelligent piece's CSI matches one as of now in the CSI database, it demonstrates that the consistent square is a copy and its content are not composed to the disk. Therefore, a physical square could conceivably guide to numerous sensible pieces. The data about mapping of consistent pieces to a physical square is kept up by the storage server in a mapping structure, CSI-Map, which has a passage for each physical square (being used). In this segment, we talk about both a challenge system and a proof-of-idea performance of ERMF, and the workloads we have utilized for experimentation

3.2 Least Cost flow Based Resource Management

In this segment, we display ERMF, a platform that is intended to have MapReduce applications in virtualized cost. ERMF gives a bunch file framework that backings a uniform record framework name-space over the group by coordinating the discrete nearby storage of the individual hubs appeared in figure 3.2. The sharedfile framework empowers a VM to be put on any bunch hub or in this way moved as necessary.

ERMF stays away from the position irregularities with an inventive resource scheduler for the cloud, particularly intended for enhancing the performance of MapReduce employments. ERMF accommodates the two data and VM resource assignment with contending requirements,
for example, storageusage, changing CPU load and system connect limits. ERMF utilizes a stream arrange based calculation that can improve MapReduce performance under the predetermined limitations by starting situation, as well as by straightening out through VM and data relocation also. Moreover, ERMF uncovered, generally shrouded, bring down level topology data to the MapReduce work scheduler with the goal that it makes close ideal taskscheduling.

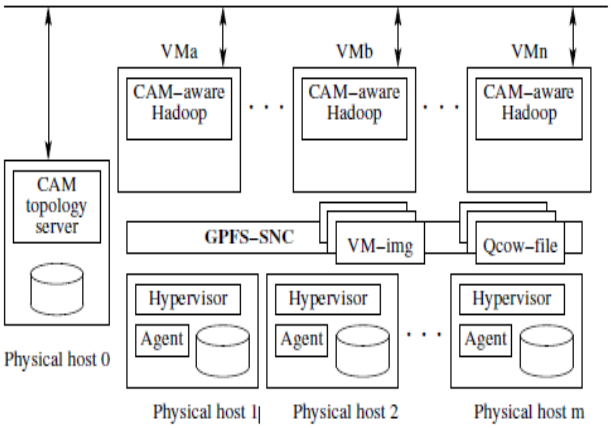

Figure 3.2 ERMF Components

ERMF is outlined as an expansion to IBM ISAAC item [93]. ISAAC executes key cloud capacities, for example, creating and deleting VMs and their determined volumes, setting the VMs in view of load and limit, keeping up accessibility of cloud benefits through grouping and bomb over components. The design parts of ERMF are executed as augmentations to related partners in ISAAC. Specifically, we have incorporated ISAAC with the GPFS-SNC [10]file framework to give an appropriate bunch record framework required by ERMF, and have stretched out ISAAC to help data and VM position in light of strategies we portray in Figure 3.2 delineates the parts of ERMF and their cooperation when conveyed in a cloud domain. The physical resources supporting the cloud comprises of a bunch of hypervisor (physical) hubs with neighborhood storage specifically joined to the individual hubs.

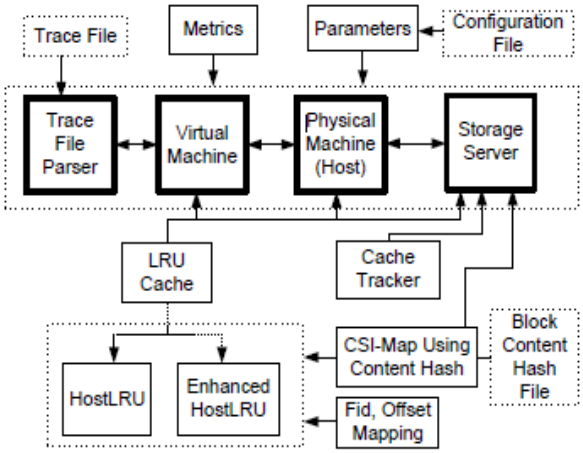

Figure 3.3 Simulation Framework

ERMF utilizes GPFS-SNC [10] to give its storage layer. GPFS-SNC is composed as a distributed storage platform, which support convenient and resource productive organization of VMs. GPFSSNC deals with the neighborhood disks specifically connected to a bunch of item physical machines. All the more particularly, it has various remarkable components that make it a cloud-accommodating storage framework. To begin with, GPFS-SNC support co-finding all squares of a file at one area, as opposed to stripping the record crosswise over machines. This empowers a VM I/O ask for to be overhauled locally from the put away area rather than remotely from physical has over the system. ERMF application this element to guarantee that co-found VM pictures are put away at one area and can be gotten to productively. Second, GPFS-SNC support an effective piece level pipelined replication plot, which ensures quick dispersed recuperation and high I/O throughput through quick parallel applications. This component is helpful for ERMF for accomplishing proficient disappointment recuperation. At long last, GPFS-SNC determines a client level API that can be utilized to inquiry the physical area of files. ERMF utilizes this API to decide genuine piece area, and utilizations this data to induce storagesimilarity for data and VM position.
MapReduce task scheduler utilizes the topology data of the bunch hubs to choose taskscheduling. The data is provided by the client as a piece of the employment design file when the occupation is submitted. Notwithstanding, trying to digest equipment level points of interest and present a basic interface to the client, existing cloud performances don't uncover the data about the topology of the bunch or the genuine arrangement of VMs to the MapReduce scheduler [11]. Besides, the underlying setup gave by the client may wind up plainly stale when the VMs are moved later. ERMF tends to these issues with three necessary parts that together give topology mindfulness as appeared in Figure 3.3. To start with, the ERMF topology server gives the extra topology data required to empower the MapReduce scheduler to put the task ideally. The topology server is a necessary segment of the ISAAC cloud benefit framework and gives a REST interface, which the scheduler conjures. The data uncovered by the topology server comprises of system and storage topologies, and other dynamic hub level data, for example, CPU stack. Second, an arrangement of operators running on the physical hubs of the bunch occasionally gather and pass on to the topology server, bits of data about the separate hub, for example, application of outbound/inbound system data transmission, IO usage and CPU/memory/storage load. The topology server merges the dynamic data it gets from the services and serves it alongside topology data about each employment running in the group. The topology data is gotten from existing VM position setup. Third, another MapReduce task scheduler interfaces with the topology server to acquire exact and current topology data. The scheduler rearranges assignment position appropriately at whatever point an adjustment in the setup is watched.

Note that ERMF needs to give an alternate scheduler in light of the fact that the standard MapReduce scheduler is intended to settle on the position choices just in view of a static design record and just toward the start of the employment [12].

While the MapReduce taskscheduler is changed to application the topology and physical host resource application data in ERMF, the MapReduce applications can keep running with no alteration. The system topology data is spoken to by a separation grid that encodes the separation between each match of VMs as cross-rack, cross-hub, or cross-VM. Current MapReduce assignment schedulers consider rack and hub territories however do not have the idea of VM region.

At the point when two VMs are set on a similar hub, they are associated through a virtual system association actualized as a piece of the hypervisor. By prudence of the way that the VMs share a similar hub equipment, the virtual system gives a rapid medium that is necessarily speedier than the between hub or between rack joins. The system activity between the VMs on a similar hub does not need to go through the outside equipment connect. The system virtual gadget essentially advances the activity in-memory through profoundly improved ring buffers. ERMF stretches out the MapReduce scheduler to consider this fine-grain area data to settle on ideal position decisions for the tasks.

The system topology data is spoken to by a separation lattice that encodes the separation between each match of VMs as cross-rack, cross-hub, or cross-VM. Current MapReduce assignment schedulers consider rack and hub territories yet do not have the idea of VM territory. At the point when two VMs are set on a similar hub, they are associated through a virtual system association actualized as a piece of the hypervisor. By uprightness of the way that the VMs share a similar hub equipment, the virtual system gives a rapid medium that is altogether speedier than the between hub or between rack joins. The system movement between the VMs on a similar hub does not need to go through the outer equipment connect. The system virtual gadget essentially advances the activity in-memory through exceedingly upgraded ring supports. ERMF stretches out the MapReduce scheduler to consider this fine-grain territory data to settle on ideal position decisions for the task. The storage[15] topology data is given as a mapping between each virtual gadget containing the dataset and the VM to which the gadget is neighborhood. In the local equipment setting, a SATA circle appended to a hub can be straightforwardly gotten to through the PCI transport. In the cloud, be that as it may, the physical pieces having a place with a VM picture appended to a VM could be situated on an alternate hub. Despite the fact that a virtual gadget may give off an impression of being specifically associated with the VM, the picture file backing the gadget could be over the system, and conceivably nearer to another VM in the bunch than the one it is straightforwardly joined to in the virtual setup. The topology server questions the physical picture area through the GPFS API and presents the data to the MapReduce scheduler.

IV. RESULTS

In our model, we accept that it is workable for the cloud supplier to profile a job and estimate its attributes, for example, work type and data, yield and middle of the road data sizes. For our present performance, we depend on client gave or foreordained sets of responsibilities to recognize ajobs type. In any case, the framework can be effortlessly reached out to decide the measure of time an application spends in various platforms (Map, Reduce), and utilize this data to decide an occupation's type. For instance, work that spends over 30% of the time in Map can be considered as Map-escalated.

We express the issue of ideally putting data in a given cloud clusterarchitecture as an example of the notable min-cost stream issue [9]. To accomplish this, we separate the arrangement issue into three sub-issues, specifically ensuring VM similarity, staying away from hotspots, and adjusting physical storage usage as indicated by various occupation types. We catch the three limitations by means of also named factors in our model. VM similarity communicates how close data ought to be set to VMs with the goal that the system activity between the relating VMs is limited. Hotspot factor communicates the normal load on a machine, and distinguishes machines that don't have enough computational resource to help the VM(s) allotted to them. To dodge a hotspot, data should be put on the minimum stacked machine. This can be dictated by measuring the current computational resource heap of the machine and adding it to the normal computational prerequisites of the VMs that will work with the data to be put on the machine. Storageapplication communicates the level of aggregate physical machine storage room that is being used.

For workloads that are both Map and Reduce serious, related data ought to be set near one another and on the minimum stacked machine. For Map concentrated workloads, the data ought to be set on the minimum stacked machine, yet does not really should be put near one another necessarily need to rearrange movement in such workloads. For Reduce escalated workloads, the main concern is the storageapplication of the machine on which the VM is to be set. For a wide range of workloads, it is attractive to put data equitably crosswise over racks to limit the need to adjust data after some time for supporting relocating VMs. We utilize these variables in building a min-cost stream diagram that encodes the components. At that point we utilize a stretched out solver to limit the worldwide cost of the diagram, along these lines taking care of the first issue of deciding how data ought to be set in the virtualized cloud.

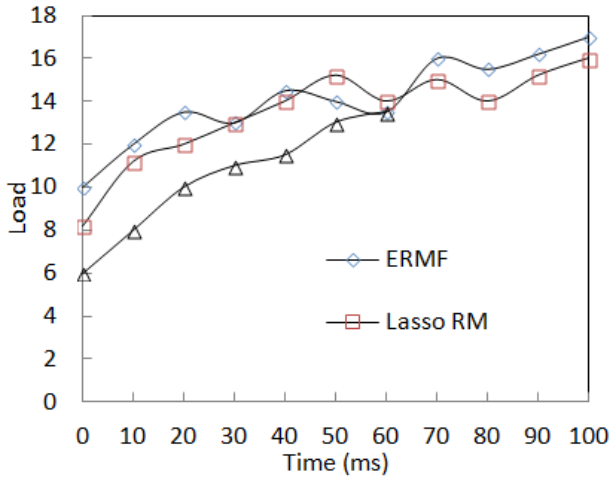

Figure 4.4 Time Vs Load

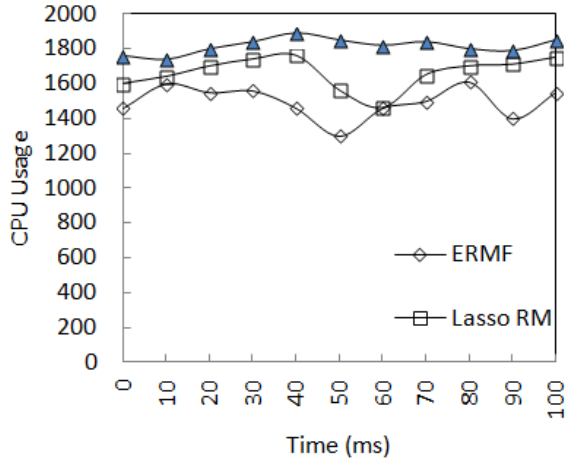

Figure 4.5 Time vs CPU Usage

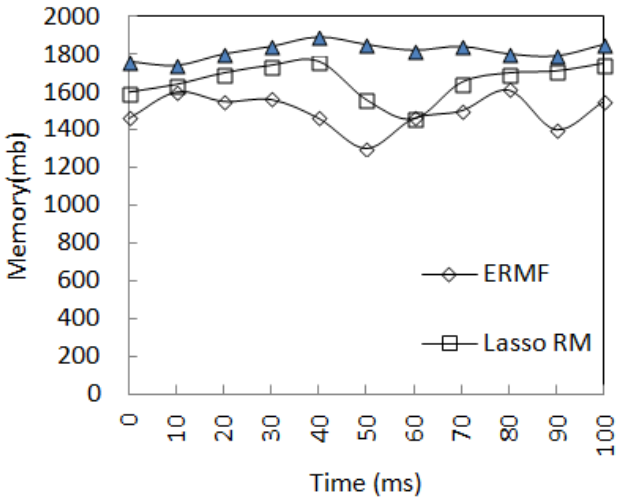

Figure 4.6 Time Vs Memory

Figure 4.4, Figure 4.5, Figure 4.6 shows the performance comparison with similar other model Lasso RM, Without ERMF with various aspects like Load, CPU Usage and Memory Utilization. The figure shows that ERMF has better efficiency than other model with the various performance metrics.

V. CONCLUSION

Our productive stream based flow based resource manager oversees data/VM position for MapReduce in multi-inhabitant virtualized cost. EMRF depends on a three-level way to deal with maintain a strategic distance from the arrangement inconsistencies as a result of overlaid topology and wasteful resource portion. All the more uniquely, ERMF uncovered process, storage and system topologies to MapReduce work scheduler, places data as per organize activity of relating occupations, expected machine load and storage usage and spots VMs with an objective of boosting worldwide data area and employment throughput.

REFERENCES

1. GrzegorzMalewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, NatyLeiser, and GrzegorzCzajkowski. Pregel: A system for large-scale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.
2. Ali Ghodsi, MateiZaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: fair allocation of multiple resource types. In USENIX NSDI, 2011.
3. Xuhui Li, Ashraf Aboulnaga, Kenneth Salem, AamerSachedina, and Shaobo Gao. Second-tier cache management using write hints. In FAST'05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies, pages 9–9, Berkeley, CA, USA, 2005. USENIX Association
4. Apache Software Foundation. Hadoop, May 2007. http://hadoop.apache.org/core
5. Song Jiang, FabrizioPetrini, Xiaoning Ding, and Xiaodong Zhang. A locality-aware cooperative cache management protocol to improve network file system performance. In Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, ICDCS '06, pages 42–, Washington, DC, USA, 2006. IEEE Computer Society.
6. A. Thusoo, J.S. Sarma, N. Jain, Zheng Shao, P. Chakka, Ning Zhang, S. Antony, Hao Liu, and R. Murthy. Hive - a petabyte scale data warehouse using hadoop. In Data Engineering (ICDE), 2010 IEEE 26th International Conference on, pages 996 –1005, march 2010.
7. NirajToila, Michael Kozuch, MahadevSatyanarayanan, Brad Karp, Bressoud Thomas, and Adrian Perrig. Opportunistic use of content addressable storage for distributed file sytems. In ATC 2003: Proceedings of USENIX Annual Technical Conference, 2003.
8. N. Attrapadung and B. Libert, "Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation," PKC 2010, vol. 6056, pp. 384-402, 2010.
9. Apache Software Foundation. Hadoop, May 2007. http://hadoop.apache.org/core/. [93]E.-J. Goh, "Secure indexes, " CryptologyePrint Archive on October 7th, pp. 1-18, 2003.
10. Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," ACNS 2005, vol. 3531, pp. 442-455, 2005.
11. Lombardi F, Di Pietro R. Secure virtualization for cloud computing. Journal of Network Computer Applications (2010), doi:10.1016/j.jnca.2010.06.008.
12. B. H. Bloom, "Space/time trade-off's in hash coding with allowable errors," Communications of the ACM, vol. 13, pp. 422-426, 1970.
13. C. Nalini, S. MageshKumar, S. Sivasubramanian, Efficient Data Access Through Secure Searching Techniques in Cloud Storage, International Journal of Pure and Applied Mathematics, Volume 116 No. 8 2017, 335-340.
14. S. Magesh Kumar, S SivaSubramanian , E. Anbalagan, Nature-Inspired Metaheuristic Algorithm for Cloud computing Disk optimization Using Bridge Framework, International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.20 (2015).
15. Magesh Kumar S, Sathish Kumar P.J, Parthipan, Optimized Data Storage And Security Over Transmission In Cloud Computing, International Journal of Pharmacy & Technology Vol. 8 | Issue No.4 | 23660-23668, Page 23660.