Dynamic Load Balancing Algorithm for Cloud Data Centers

Fauziya Anjum[1], Prof. Nitinkumar Chaudhary[2], Dr. Jayant Karanjekar[3]

[1]Student of Computer Science Engineering Department, Wainganga College of Engineering & Management, Nagpur, India.
[2]Asst.Prof. of Computer Science Engineering Department, Wainganga College of Engineering & Management, Nagpur, India
[3]Prof. of Computer Science Engineering Department, Wainganga College of Engineering & Management, Nagpur, India

---***---

Abstract - *Storage device technologies are rapidly adapting to meet the ever-increasing demands by the users. Nowadays, users need high speed and low-cost storage options even for home applications. To cater this demand, researchers have come up with a hybrid storage system, where the frequently accessed data like operating system files, user's program files, and others are stored on solid state drives (SSDs), while the non-frequently used data, like videos, documents, music and everything else is stored on normal hard disk drives (HDDs). In order to obtain better efficiency in terms of load balancing and task scheduling, researchers have proposed Dynamic and Integrated resource scheduling (DAIRS) algorithm for non-hybrid storage systems. In this work, we propose to implement the DAIRS algorithm for hybrid storage systems, and consider the nuances like intra-differences in read and write cycle times for different storage types, caching high frequency data from for low speed storage, and others. This work aims to improve the efficiency of the existing load balancers and task schedulers for hybrid systems, and provide a system which is robust and can be used for real time hybrid systems.*

***Key Words*: Hybrid, storage, load balancing, DAIRS, task, scheduling**

1. INTRODUCTION

A Cloud datacenter can be a distributed network in structure, which is composed of many compute nodes (such as servers), storage nodes, and network devices. Each node is formed by a series of resources such as CPU, memory and so on. Each resource has its corresponding properties. There are many different types of resources for Cloud providers. This paper focuses on the Infrastructure as a Service (IaaS) level. The definition and model defined in this paper aim to be general enough to be applicable to a variety of Cloud providers. In a traditional data center, applications are tied to specific physical servers that are often overprovisioned to deal with workload surges and unexpected failures [2]. Such configuration rigidity makes data centers expensive to maintain with wasted energy and floor space, low resource utilization and significant management overheads. With virtualization technology, today's Cloud datacenters become more flexible, secure and ondemand allocating. With virtualization, Cloud datacenters should have ability to migrate an application from one set of resources to another in a non-disruptive manner. Such agility becomes a key in modern cloud computing infrastructures that aim to efficiently share and manage extremely large data centers. One key technology plays an important role in Cloud datacenters is resource scheduling. There are quite many research conducted in the area of scheduling algorithms. Most of them focus on load balancing for traditional Web servers or server farms. One of the challenging scheduling problems in Cloud datacenters is to take into consideration both the allocation and migration of reconfigurable virtual machines, and the integrated features of hosting physical machines. Unlike traditional load-balancing scheduling algorithms which consider only physical servers with one factor such as CPU load, DAIRS treats CPU, memory and network bandwidth integrated for both physical machines (PMs) and virtual machines (VMs). We develop integrated measurement for the total imbalance level of a Cloud datacenter as well as the average imbalance level of each server. Buyya et al. [1] introduced a way to model and simulate Cloud computing environments, in which a few simple scheduling algorithms such as time-shared and space-shared were discussed and compared. [5] introduced three general scheduling algorithms for Cloud computing and some results were provided. Wood et. al. [6] introduced techniques for virtual machine migration and proposed some migration algorithms. Zhang [7] compared major load-balance scheduling algorithms for traditional Web servers. Singh et al. [2] proposed a novel load balancing algorithm called VectorDot for handling the hierarchical and multi-dimensional resource constraints by considering both servers and storage in a Cloud. Tian et al. [3] provided a comparative study of major existing scheduling strategies and algorithms for Cloud datacenters. In this work, we introduce a dynamic and integrated resource scheduling algorithm (DAIRS), which treats CPU, memory and network bandwidth integrated for both physical machines and virtual machines. We develop integrated measurement of the total imbalance level of a Cloud datacenter as well as the average imbalance level of each server for performance evaluation.

2. PROPOSED ALGORITHM

2.1 Related work

Hybrid storage systems are gaining popularity due to their many fold performance related advantages. These advantages include but are not limited to, low cost of storage, at par performance when compared to SSD systems, higher retention than SSDs, etc. The main reason why hybrid storage systems are so successful is due to the fact that the research and development in this area has been tremendous. Researchers from various fields have deployed techniques which utilize the advantages of the hybrid combination, and improve the overall QoS of the system. In the work done by Xiaojian Wu and A. L. Narasimha Reddy in [2], concurrency techniques are used in order to achieve low latency and high throughput for hybrid storage systems. They have used initial block allocation along with task migration in order to get to a state called as "Wardrop equilibrium". The algorithm proposes different techniques for both read and write requests, and updates the cache on each request completion. The system's performance with compared with simple HDDs, only SSDs, SSD+HDD with stripping policies and SSD+HDD with the concurrency policies. They claim to have improved the overall throughput by more than 200%, which should be re-evaluated by researchers before implementing it. They also claim to have reduced the latency by around 2 to 3% than the SSD implementation, which looks to justifiable due to the fact that the proposed technique uses mathematical corollaries in order to evaluate the read and write cycle times. The developed protocol also improves the number of Input Output Operations Per Second (IOPS) by about 10%, when compared on dbench workloads. While the throughput on NetApp workload is almost identical to other implementations, but they claim to have improved it in theory. Other tests on dbench and NetApp workloads show similar result trends for latency, throughput and IOPS.

Researchers Yu Hua, Bin Xiao, Xue Liu and Dan Feng propose a locality aware algorithm in [3], where they have implemented a system called as NEST, inspired by how cuckoos keep things-of-interest in a nests, and finds them as and when needed. Their work uses SSDs, Dynamic Random-Access Memories (DRAMs) and HDDs, & they propose to have used the division of labor mechanism for getting a proper response from the hybrid system. They have analyzed the proposed NEST system on large cloud datasets, and made the implementation open source for everyone to test and try. They have compared their performance with LSB tree, Locality-sensitive hashing (LSH) and the baseline models. For the LANL workload set, the lookup latency is observed to have reduced by more than 60%, than the best Buffer Hash technique, which might be an oversell, due to the fact that the system doesn't use any additional hardware resources for latency minimization. Similar claims have been made read latency (a.k.a. lookup latency), and write latency (a.k.a. insert latency). These observations are done w.r.t. Buffer Hash Algorithm, Buffer Bloom Filters, Chunk Stash and Berkeley DB techniques, and the researchers must perform due diligence before trying to implement this on their real time working environments.

2.2 Process Diagram

The algorithmic flow of the existing DAIRS algorithm which is primarily meant for non-hybrid storage systems can be shown in the following figure,

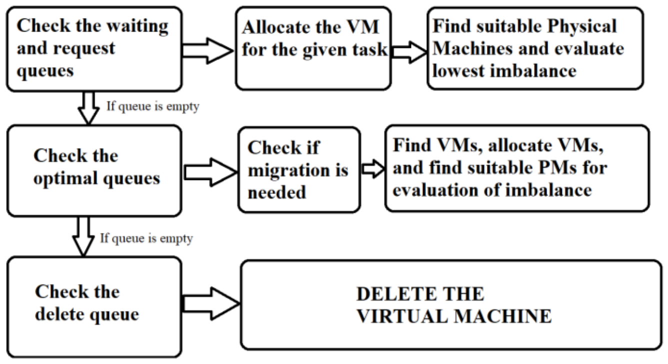
Check the waiting and request queues
Allocate the VM for the given task
Find suitable Physical Machines and evaluate lowest imbalance
If queue is empty
Check the optimal queues
Check if migration is needed
Find VMs, allocate VMs, and find suitable PMs for evaluation of imbalance
If queue is empty
Check the delete queue
DELETE THE VIRTUAL MACHINE

Figure 1. Algorithmic flow of the DAIRS implementation in non-hybrid storage systems

- In the DAIRS algorithm specified above, there is no mention about any kind of memory structure, which is a major research gap in this system. Because, due to incompatible memory types in the system, the following issues arise,
- Inconsistent read and write cycles, faster memory systems like solid state drives (SSDs) have faster read and write cycles than hard disk drives (HDDs), due to which the overall speed of the system takes a hit
- Cache invalidation problem, due to the difference in storage technology, the memory caches are invalidated at different time instances, and thus the cached memory data available with the system is not consistent throughout the lifecycle of the system
- Storage freezing issues due to processing cycles, this is a very common issue with large capacity SSDs when combined with low capacity SSDs or any capacity HDDs. Due to this issue, the processing freezes momentarily, as one of the storage devices might not be ready to accept data. For example, if a high capacity SSD is combined with a small capacity SSD, and a processing operation is in process, then due to difference in capacity, the search time for lower capacity SSD is lower than higher capacity SSD, thus the processing unit waits for the higher capacity SSD to finish the search/read/write operations. This is an inherent issue when combining multiple storage types for processing
- To resolve these issues, we propose a novel DAIRS algorithm which is based on hybrid storage system, and takes into consideration all the issues which are described. The algorithm tries to solve these issues by making technological improvements to the overall algorithm, and ensure that these limitations are part of the running process for the system.

2.3 Proposed Work

The flow diagram for the improved DAIRS algorithm which can be used in hybrid storage is given in figure 2. The proposed hybrid DAIRS (hDAIRS) system, works in the following steps,

1. Assume that there are 'N' different storage systems connected to the cloud (the value of N can be 2 for simple cases, where only 1 SSD and 1 HDD is connected, but for complex systems, the value of N can change). Arrange the storage system numbers from fastest to slowest. Meaning, storage system 1 is the fastest, and storage system N is the slowest in terms of per unit task execution time
2. Initiate dummy requests for all the 'N' storage systems, and capture the following responses,
    a. Read time for all systems Tr1, Tr2, … , TrN
    b. Write time for all systems Tw1, Tw2, … , TwN
    c. Processing time for all systems Tp1, Tp2, … , TpN

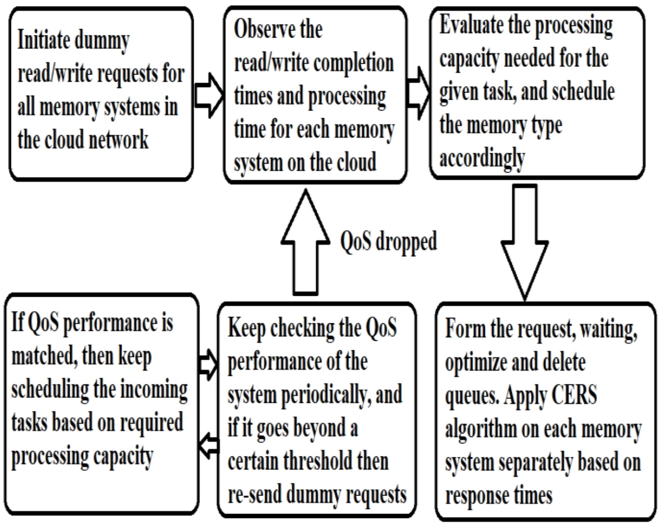
Initiate dummy read/write requests for all memory systems in the cloud network
Observe the read/write completion times and processing time for each memory system on the cloud
Evaluate the processing capacity needed for the given task, and schedule the memory type accordingly
QoS dropped
If QoS performance is matched, then keep scheduling the incoming tasks based on required processing capacity
Keep checking the QoS performance of the system periodically, and if it goes beyond a certain threshold then re-send dummy requests
Form the request, waiting, optimize and delete queues. Apply CERS algorithm on each memory system separately based on response times

Figure 2. Proposed CERS method on hybrid storage system

1. Find the total delay for all systems
2. Find the minimum delay needed for execution of one unit of the given task on each of the storage systems, let the delays for each system be, D1, D2, D3, … DN
3. Let the total number of cycles needed for execution of the given task on the given storage system be Tni
4. For each of the tasks, evaluate the value of Tni, and establish the relationship table
5. The relationship table is made for all of the 'k' tasks and 'N' memory types
6. Evaluate the 2D mean of the table, and evaluate a progressive threshold
7. Evaluate the task scheduling interval factor (Tsif)
8. Evaluate the column wise mean of the relationship table, to find the mean task execution time. Let this time be MT1, MT2, …, MTk
9. Sort the tasks in descending order of MT values, let the new sorted task order be Ts1, Ts2, ..., Tsk

Using the above identities, the issues of hybrid storage system will be resolved, and a proper system with optimum task allocation speed will be developed using the DAIRS algorithm. The mentioned algorithm will allow larger and time-consuming task units to be executed on higher speed devices, and simpler task units to be executed on slower speed devices, due to this, the overall QoS of task and resource scheduling should improve. We plan to implement this system on a cloudsim+ based environment with multiple datasets in order to check its validity, and compare the algorithm with normal DAIRS system in the presence of hybrid storage.

3. EXPERIMENT AND RESULT

An algorithm is proposed based on a differential evolution algorithm in which new strategies for its mutation and crossover operations are proposed and used. It has faster convergence based on our experimental results. The results show that it can obtain shorter completion time and more balanced load ratio than three other methods do. It also achieves higher user-provider satisfaction degree. Because in a resource allocation process, the time to compute a plan with traditional and proposed methods is far less than the completion time of a task, we do not discuss it. Quality of service (QoS) and trust are important for resource allocation. Task deadline and energy consumption of resources are two additional considerations. Their incorporation into the present model deserves more study.

4. CONCLUSION

The The developed algorithm must show good results in terms of the overall QoS of the system, but in case there are issues, then researchers can opt for machine learning optimizations like Q-learning, and implement a reward mechanism. This will help to further improve the cloud's performance by imbibing intelligence in itself, which will select proper storage systems for most suited tasks.

REFERENCES

[1] Armbrust, M., Fox, A., Griffith, R., et al.: 'Above the clouds: a Berkeley view of cloud computing'. Technical Report No. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009

[2] Vaquero, L.M., Merino, L.R., Moran, D.: 'Locking the sky: a survey on IaaS cloud security', Computing, 2011, **91**, (1), pp. 93–118

[3] Xiao, Z., Xiao, Y.: 'Security and privacy in cloud computing', IEEE Commun.
Surv. Tutorials, 2013, **15**, (2), pp. 843–859

[4] Majumder, A., Namasudra, S., Nath, S.: 'Taxonomy and classification of access control models for cloud

environments'. in Mahmood, Z. (Ed.): 'Continued rise of the cloud' (Springer, 2014), pp. 23–53

[5] Namasudra, S., Roy, P.: 'A new secure authentication scheme for cloud computing environment', Concurr. Comput., 2016, doi: 10.1002/cpe.3864

[6] Namasudra, S., Roy, P.: 'Secure and efficient data access control in cloud computing environment: a survey', Multiagent Grid Syst., 2016, **12**, (2), pp. 69–90

[7] Schuster, F., Costa, M., Fournet, C., et al.: 'VC3: trustworthy data analytics in the cloud using SGX'. Proc. of the IEEE Symp. on Security and Privacy, San Jose, CA, May 2015, pp. 38–54

[8] Li, J., Lin, D., Squicciarini, A., et al.: 'Towards privacy-preserving storage and retrieval in multiple clouds', IEEE Trans. Cloud Comput., 2015, doi: 10.1109/TCC.2015.2485214

[9] Namasudra, S., Roy, P.: 'Size based access control model in cloud computing'.Proc.oftheInt.Conf.onElectrical,Electronics,Signals, Communication and Optimization, Visakhapatnam, India, 2015, pp. 1–4

[10] Zhu, Y., Hu, H., Ahn, G.J., et al.: 'Towards temporal access control in cloud computing'. Proc. of the IEEE INFOCOM, Orlando, FL, March 2012, pp. 2576–2580

[11] Danwei, C., Xiuli, H., Xunyi, R.: 'Access control of cloud service based on UCON'. Proc. of the 1st Int. Conf. on Cloud Computing, Beijing, China, 2009, pp. 559–564

[12] Gao, X., Jiang, Z., Jiang, R.: 'A novel data access scheme in cloud computing'. Proc. of the 2nd Int. Conf. on Computer and Information Applications, Taiyuan, China, 2012, pp.124–127

[13] Yu, S., Wang, C., Ren, K., et al.: 'Achieving secure, scalable, and fine-grained data access control in cloud computing'. Proc. of the IEEE INFOCOM, San Diego, USA, 2010, pp.1–9

[14] Wu, Y., Suhendra, V., Guo, H.: 'A gateway-based access control scheme for collaborative clouds'. Proc. of the 7th Int. Conf. on Internet Monitoring and Protection, Stuttgart, Germany, May 2012, pp.54–60

[15] Sun,L.,Wang,H.:'Apurposebasedusageaccesscontrolmodel',Int.J. Comput. Inf. Eng., 2010, **4**, (1), pp. 44–51

[16] Pazzani, M.J.: 'A framework for collaborative, content-based and demographic filtering', Artif. Intell. Rev., 1999, **13**, (5–6), pp.393–408

[17] Calheiros, R.N., Ranjan, R., Beloglazov, A.: 'CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms', Softw. Pract. Exp., 2011, **41**, (1), pp.23–50

[18] http://commons.apache.org/proper/commons-math/download_math.cgi, accessed 11 March2016

[19] http://java.com/en/download/index.jsp, accessed 09 March2016