# SLA-aware Resource Scheduling for Cloud Storage

Zhihao Yao
Computer and Information Technology
Purdue University
West Lafayette, Indiana 47906
Email: yao86@purdue.edu

Ioannis Papapanagiotou
Computer and Information Technology
Purdue University
West Lafayette, Indiana 47906
Email: ipapapan@purdue.edu

Robert D. Callaway
Cloud Solutions Group
NetApp, Inc.
RTP, North Carolina 27709
Email: bob.callaway@netapp.com

***Abstract*—As most on-line services are now hosted on the cloud, customers are requesting Service Level Agreements (SLAs) in order to use cloud services with acceptable Quality of Service. Nonetheless, the cloud is based on provisioning resources on demand (known as cloud elasticity). Hence, it is of primary importance to design multi-tenant cloud storage solutions that can provide storage services with guarantees equivalent or close to bare-metal deployments.**

**In this paper, we address the problem of scheduling volume create requests to backend hosts. We design and implement SLA-aware scheduling policies based on the distributed OpenStack scheduling model. We compare and contrast the existing scheduling storage policies by performing a simulation experiment. We demonstrate that a new SLA-aware scheduling policy that takes into account both the available capacity but also the I/O throughput of the backend nodes is needed to offer quality storage services. Our SLA-aware scheduling policy is able to achieve more than 20% improvement in the rate of SLA violations. Furthermore, it requires fewer storage nodes (hence lower capital expenses) and can provide higher volume I/O throughput performance compared to the default policies.**

***Keywords—Cloud storage; Service Level Agreement (SLA); Resource Scheduling; Infrastructure as a Service.***

## I. Introduction

Cloud computing has gained significant attention in both academia and industry. Most major IT companies are either investing resources into the cloud computing area or buying cloud services from major cloud service providers. Cloud infrastructure, or Infrastructure as a Service (IaaS), challenges the traditional methods of high performance computing and datacenter management. It offers high flexibility, scalability and cost-effectiveness and a growing number of customers are moving their IT services to cloud platforms [1].

With the popularity of cloud computing, a variety of challenges have appeared, such as performance prediction and management [2]. More specifically, in today's cloud storage field, many IaaS platforms such as OpenStack [3] and Eucalyptus [4] lack an effective performance-oriented resource scheduling policy to control the system performance. On the other hand, SLAs are one of the major considerations for every buyer of cloud computing services. The question often asked is how many nines of availability or what is the maximum delay a given provider will guarantee. For some sensitive applications, a minimum of three nines (99.9%) of availability is required, whereas for others, low-latency services are more important. The inherently dynamic nature of a cloud offering, combined with the fact that resources change dynamically, makes it difficult to define a meaningful SLA for various cloud computing services. As a consequence, only a few of the cloud storage providers are offering service SLAs on I/O throughput performance. Amazon Elastic Block Store is the only commercial cloud provider who provides a specific level of I/O performance by creating a provisioned I/O per second (IOPS) volume [5]. This service is able to deliver no less than 90% of expected IOPS performance in 99.9% of the time. In order to increase the market share in the highly competitive cloud market, cloud providers must offer higher and differentiated performance SLAs in terms of availability, capacity and I/O throughput.

From the research perspective, only a few studies have looked in the issue of SLA-aware virtual resource management on the cloud [6] [7] [8], and from the perspective of allocating VMs. Nonetheless, due to the essential difference in operation mechanisms, these research achievements are not directly applicable to the storage volume allocation problem. More recently, some studies looked into an energy-aware resource scheduling algorithm [9] [10]. In the context of SLA-based or SLA-aware resource scheduling, Goudarzi et al. [11] presented an SLA-based multi-dimension resource allocation algorithm for multi-tier applications in cloud computing environments. Wang et al. [12] proposed an automatic optimization schema that utilizes data chunking, placement and replication to achieve better I/O performance. However, to our knowledge, there has not been any work that tackled the volume request allocation problem in cloud block storage systems using more than the available capacity as an input to the allocation algorithm.

The problem addressed in this paper is the design of a scheduling policy to effectively utilize storage resources and manage the performance in a cloud infrastructure platform so that cloud storage services can be offered with guaranteed capacity and I/O throughput. In cloud storage, there are three main categories on how the data can be stored: (a) *ephemeral storage*: data are stored on the provisioned cloud VMs and are lost after reboot, or the release of that VM; (b) *block storage*: data is stored in the persistent storage that is attached, and it is accessible and can be deleted from the VM; (c) *object storage*: data that is accessed from a large number of clients that demands scalable access, generally through a REST API.

In this paper, we propose an SLA-aware resource scheduling policy for cloud block storage. The key idea is based on the fact that I/O throughput should be taken into account when the scheduler is making a volume request

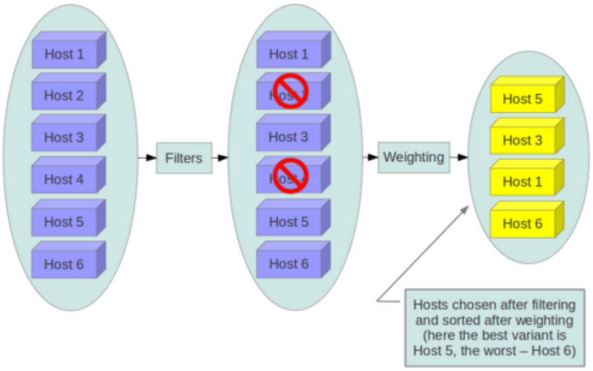
Host 1
Host 2
Host 3
Host 4
Host 5
Host 6
Filters
Host 1
Host 3
Host 5
Host 6
Weighting
Host 5
Host 3
Host 1
Host 6
Hosts chosen after filtering
and sorted after weighting
(here the best variant is
Host 5, the worst – Host 6)

Figure 1. Workflow of Filter Scheduler

placement decision. As the beneficiaries of such an approach, the cloud storage providers will be able to provide higher I/O throughput performance SLA to customers with fewer storage nodes.

The reminder of our paper is organized as follows: In Section II, we briefly describe the current scheduling algorithm in the OpenStack block storage service and its corresponding weaknesses. In Section III, we describe the proposed scheduling algorithm design. In Section IV, we present our experimental methodology including evaluation metric and simulator design. Finally, in Section V we conclude our work and present potential extensions.

## II. OpenStack Cinder Scheduling

OpenStack is a global collaboration of developers and cloud computing technologists producing a ubiquitous open source cloud computing platform for public and private clouds. The open source community and more than 200 companies have joined the development of OpenStack. OpenStack consists of multiple components focusing on different cloud services so that user is able to customize the infrastructure based on their individual needs. The core services in OpenStack are identity (Keystone), compute (Nova), network (Neutron), Glance for a repository of VM images, and storage (Cinder for block storage, Swift for object storage).

Our work focuses on the Cinder service [13], the block storage service of OpenStack. Cinder provides an infrastructure for managing block storage devices (i.e. volume) and provides end users with a self-service API to request and consume those resources. Various storage backends can be connected to Cinder to support storage virtualization. Cinder includes three components: *cinder-api*, *cinder-volume* and *cinder-scheduler* to enable management of volumes and snapshots. The key component is the *cinder-scheduler*, which encompasses several scheduling policies. When Cinder receives a volume request such as create, the *cinder-scheduler* selects a storage node as the best candidate to serve the request based on each node's status and request parameters. The workflow of the filter scheduler is shown in Fig.1 and consists of two steps:

i. *Filtering*. Scheduler maintains a list of storage hosts with metadata. During this step, some hosts which are unable to provide enough capability to meet the need of a volume request are filtered out from the list. For example, if a 10GB volume is requested, then all hosts who have less than 10GB available space will be excluded from the candidate list.

ii. *Weighting*. After the filtering step, all hosts on the list are capable to serve the request. The scheduler calculates the cost, i.e. weight, of each host by comparing the hosts characteristics and the request characteristics. Then hosts on the list are sorted based on their weight. The host which has the least weight will be chosen as the best candidate.

Several weighting policies have been implemented for the filter scheduler in the Havana release of OpenStack in October 2013. The default weighing policy is the *available capacity* policy, in which hosts are sorted based on their available capacity; the host with the largest available capacity will be the chosen one to provision. *Allocated space* policy is an optional policy in which hosts are selected based on the lowest allocated space. If the total capacity of each host is the same, then the allocated space policy will behave as the capacity policy. Finally, the *Chance* policy randomly picks one host from list.

Based on the workflow model, the performance of the scheduler is heavily dependent upon the weighing step. After a new volume is placed on a storage node, all operations on that volume will affect the state of the host node and even changes the state of whole cloud storage system. Therefore, the decision made by the scheduler will prominently influence the performance of the storage system. However, not many system designers and cloud providers have realized the importance of proper resource scheduling in cloud storage systems. Most of them have focused on the scheduling optimization on computing resources, such as the virtual machine placement problem. This is also obvious in the OpenStack's code base: 30 scheduling filtering policies have been implemented for the Nova compute platform, but only 6 policies exist in the Cinder block storage service. Furthermore, there are a number of limitations on the current scheduling strategies in Cinder. The key disadvantage is that none of the existing policies is I/O throughput aware. In other words, the scheduler may not select hosts that offer higher I/O throughput, even if they have the same available capacities. This may result in poor performance for a cloud storage service, unpredictability of the currently allocated volumes, and the extra allocation of resources to satisfy the end-to-end SLAs. In this paper, the I/O throughput performance of current policies is investigated and our results show that none of them is able to provide satisfied I/O performance management ability.

## III. SLA-aware Scheduling

The main objectives of our work are (i) to enable cloud storage systems with I/O performance management (ii) to minimize the I/O throughput SLA violations using effective scheduling policies. I/O throughput SLA in cloud storage service is defined as the I/O throughput of user's volume is higher or equal to a specific number of IO operations per second (IOPS) in at least 99.9% of time. The core mechanism of our strategies takes into account the I/O throughput property, when the scheduler makes a new volume placement decision.

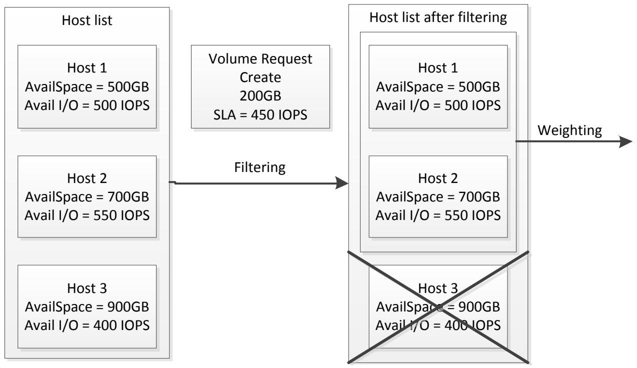

Figure 2. I/O throughput filtering

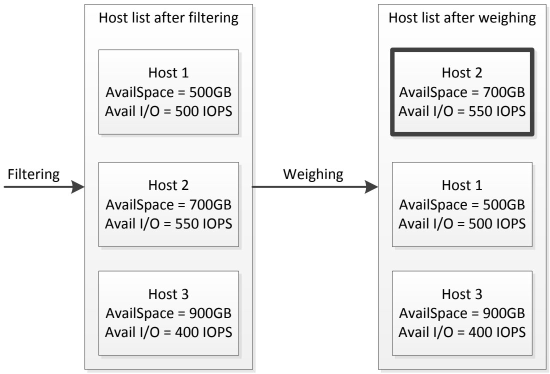

Figure 3. I/O throughput weighing

The scheduler continuously monitors the state of all storage hosts, including available capacity and allocated capacity.

In the OpenStack's scheduling implementation, the following properties are added (1) *host I/O throughput*, (2) *volume I/O throughput* and (3) *available volume I/O throughput*. Available volume I/O throughput is defined as the I/O throughput a new volume will get after it has been created on a specific host. By taking advantage of these properties, volume placement decision is not just based on the storage capacity, but also depends on the I/O throughput of the both the volume in service and the volumes that are going to be allocated to the backends. Hence, the scheduler gains the ability to manage the I/O performance and the cloud provider is able to offer an I/O throughput SLA to the customer. We study the performance of the OpenStack Cinder filter scheduling by allocating the I/O throughput properties at the filtering and the weighting steps. More specifically:

- *I/O throughput filtering.* After filtering out hosts without sufficient capacity to serve volume request, the scheduler also filters out the hosts whose available volume I/O throughput is lower than the SLA IOPS.
- *I/O throughput weighing.* After sorting the eligible hosts according to their available volume I/O throughput, the host with the highest I/O throughput will be chosen as best candidate to place the volume request.

### *A. I/O throughput filtering*

The purpose of taking into account the I/O throughput at the filtering stage is to minimize the I/O SLA violations because no matter what weighting policy will be applied on the filtered list, the final decision is SLA guaranteed. In Fig. 2, an example is depicted. Assuming a request to create a 200GB volume with 450 IOPS SLA, the scheduler first checks the available capacity of each host. The host state shows that the available space on all hosts is large enough to serve this request. However, host #3 has to be filtered out because it can only provide 400 IOPS for this new volume. Host #1 and #2 pass the filtering step and ready for weighing since their available I/O speed is higher than the SLA requirement.

If the I/O throughput filter returns an empty host list, which means no host can serve the volume request with the SLA guaranteed, the scheduler will try a second filtering round using available capacity filtering policy. If some hosts have sufficient capacity, then the volume request will be served according to the result of available capacity filtering algorithm. The reason we design this way is because serving a request is always better than rejecting it, although I/O is not fully guaranteed. However, an SLA violation will happen as long as I/O throughput filter return an empty list.

### *B. I/O throughput weighing*

*I/O throughput* weighing policy utilizes the default available capacity filter, but considers the I/O throughput property at the weighing step. Fig. 3 shows the weighing procedure. The host with the highest available volume speed will be chosen as the best fit. Assuming the same volume request, as in the previous example, all hosts will pass the capacity filter because there is enough available storage space in all nodes. Then, the I/O throughput weighing algorithm sorts hosts in descending order by their available volume I/O throughput. In the end, host #2 with 550 IOPS available volume I/O throughput wins the game although it does not have the largest available capacity. Host #3 which is a potential SLA violation choice is sorted to the end of list. In general, the top host is able to guarantee the I/O throughput SLA. If the available volume I/O throughput of the top host is lower than SLA requirement, then all active volumes are violating the I/O throughput SLA. In this case, the cloud provider may need to add an extra storage node since the current hardware setup is unable to support the pre-defined SLA.

In addition to the above, we are also proposing two modified policies based on the I/O throughput weighting policies. Both of them are taking into account the available capacity and I/O throughput at the weighing step. We call the first policy *ThrThenCap,* which stands for I/O throughput then capacity. The scheduler performs two weighing steps before a decision is made. After the first I/O throughput weighing step, several hosts on the sorted list may have the same available volume I/O throughput. These hosts are sorted again based on the available capacity. The winner of this method will have the highest available volume I/O throughput as well as the largest available capacity.

The second policy we introduce is the *ThrAndCap,* which stands for I/O throughput and capacity. It is a weighted sum of the available I/O and the available capacity of the host. The equation of calculating the host weight ($W$) is as follows:

$$W = \left(\frac{Available\ I/O}{Host\ bandwidth} + \frac{Available\ space}{Host\ total\ capacity}\right) * 100 \qquad (1)$$

The host with the highest weight will be selected to create a new volume. Only one weighing step is performed on this algorithm. However, this approach may not always return the most ideal candidate when considering I/O throughput. The reason is that when the available volume I/O throughput of all hosts are very close, and there is a host whose available space is significantly higher than other hosts, then this host will have the highest weight according to equation (1). In this case, the available capacity property is given a higher priority.

## IV. Performance Evaluation

In this section, we present the performance results obtained from the evaluation of OpenStack's Cinder scheduler policies. Due to the complexity of the cloud storage system and the fact that it is hard to perform multiple iterations of our experiments in our Cinder storage deployment, we chose to use a simulation environment to achieve repeatability of our experiments. We developed a simulator in Java based on the Cinder scheduling model and implemented the corresponding scheduling policies. The combination of different filtering and weighing scheduling policies tested in this evaluation are shown in Table 1.

TABLE I. Policy Combination

| Policy Number | Filtering Policy | Weighing Policy |
|---|---|---|
| 1 | Capacity | Available Capacity |
| 2 | Capacity | Chance |
| 3 | Capacity | I/O throughput |
| 4 | Capacity | ThrThenCap |
| 5 | Capacity | ThrAndCap |
| 6 | I/O throughput + Capacity | Available Capacity |
| 7 | I/O throughput + Capacity | Chance |
| 8 | I/O throughput + Capacity | I/O throughput |
| 9 | I/O throughput + Capacity | ThrThenCap |
| 10 | I/O throughput + Capacity | ThrAndCap |

### A. *Performance Metrics*

In order to compare the efficiency of different scheduling policies and evaluate their corresponding performance, we use two metrics. The first metric is the *SLA violation percentage*. When a volume performs I/O operations and the I/O throughput is lower than the SLA threshold, an SLA violation occurs. The SLA violation percentage is defined as the total number of SLA violation events relatively to the total number of sampled simulation time. The second metric is the *volume I/O throughput*. On the agreement of complying with the SLA performance, customers always expect higher volume speed than they will have. An intelligent scheduling algorithm should be able to demonstrate outstanding management ability in terms of both these metrics.

### B. *Experiment Design*

We choose the virtual Cinder block storage deployment proposed by Rackspace [14]. There are 8 storage nodes in the system, each node is comprised of 1 CPU core, 8 GB memory and 12 * 600GB 15K Serial Attached SCSI (SAS) in a Redundant Array of Independent Disk (RAID) 1+0. A single hard disk is able to achieve 190 IOPS throughput with 8k block size when I/O operation is 70% read and 30% write. This load ratio is the most common ratio in cloud storage systems [15] [16]. We use a RAID IOPS calculator [16] to compute the maximum I/O bandwidth of a storage node to be 1948 IOPS. The test workload consists of 5000 volume requests. Each volume request contains parameters such as arrival time, expiration time, volume size and I/O throughput SLA. Arrival time and expiration time is modeled according to a Poisson distribution with the mean value of 20 and 600 minutes, respectively. We choose Poisson distribution for simplicity, as other distributions do not affect the core outcome of our results. The simulation simulates 120000 minutes operation and the data are sampled between 1000 to 9000 minutes to achieve steady state performance. Volume sizes are randomly selected from 100GB, 500GB or 1TB. Each experiment has been run 10 times.

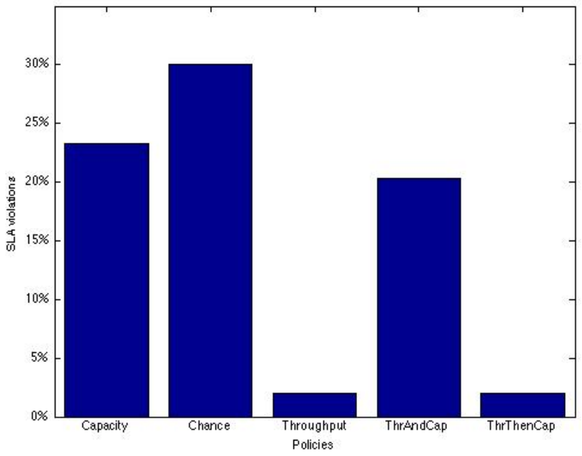

Figure 4. Percentage of SLA violation for different weighing policies with available capacity filtering policy

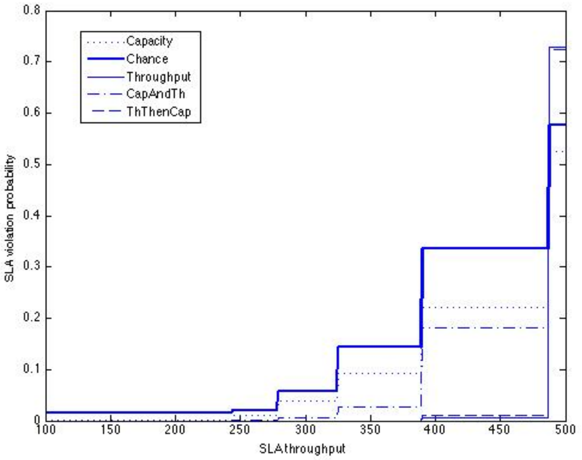

Figure 5. Percentage of SLA violation for different weighing policies with different I/O throughput SLA

### C. *Simulation Results*

Fig. 4 presents the number of SLA violations of the policies #1-#5 over 10 simulation runs. For our first

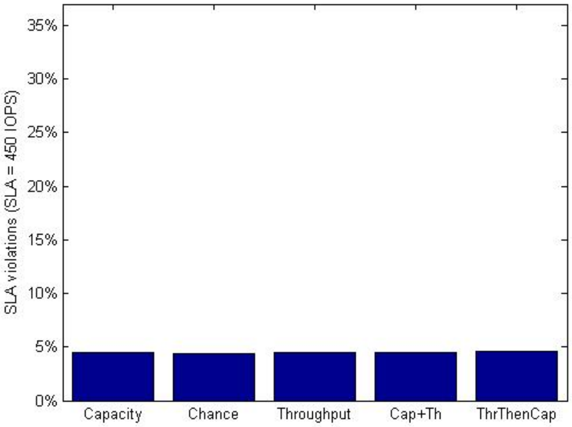

Figure 6. Percentage of SLA violations for different weighing policies with capacity and I/O throughput filtering policy

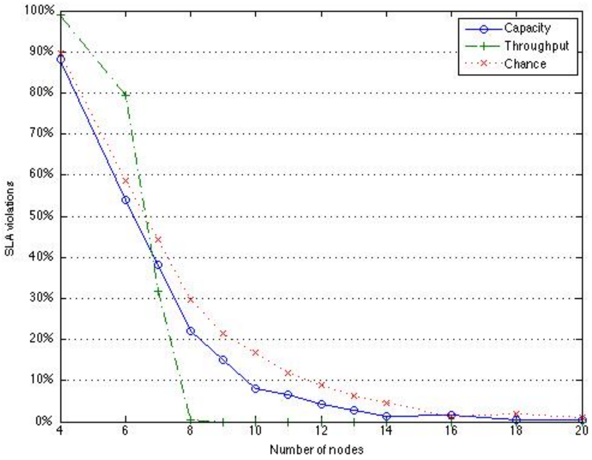

Figure 7. Percentage of SLA violations with different number of nodes

experiment, we set the I/O throughput SLA violation threshold to 450 IOPS. Note that the first two policies are the ones currently implemented in Cinder scheduler. Evidently the Capacity and Chance weighting policies provide the worst performance. The first one may result in 23% of SLA violations, whereas the second one in 30%. On the contrary, our I/O throughput weighting policy (#3 in Table 1) provides only 2.01% of SLA violations. In other words, nearly 98% of the time the volumes will receive more than 450 IOPS by the block storage hosts. Due to the weakness we discussed in Section III-B, available capacity plus I/O throughput strategy still have 20.29% violation rate. The mean and the confidence intervals of Fig.4 are also shown in Table II. However, the two stage weighing policy has SLA violations close to the I/O throughput policy (around 2%).

TABLE II. THE MEAN OF SLA VIOLATIONS

| Policy Number | Mean | 95% CI |
|---|---|---|
| 1 | 23.29% | (22.14% 24.44%) |
| 2 | 30% | (29.10% 30.90%) |
| 3 | 2.01% | (1.33% 2.69%) |
| 4 | 20.29% | (19.06% 21.52%) |
| 5 | 2% | (0.93% 3.07%) |

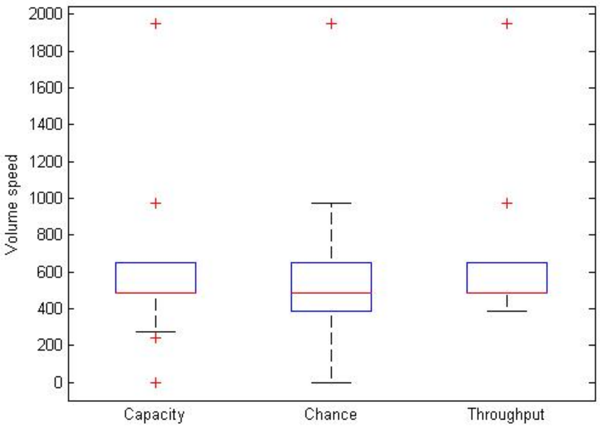

Figure 8. Volume speed performance of different policies

In Fig.5, we performed the same experiment with a variable number of SLA violation threshold. The threshold is varied from 100-550. We did not vary for higher numbers because changes to the simulated hardware setup would have been required to achieve predictable SLAs. One may observe that the I/O Throughput weighting policy can provide better performance across the whole range of the IOPS values. Hence, we conclude that to provide the best SLA performance, I/O throughput policy should be applied at the weighing step.

We then proceed with a different strategy. We now modify the filtering policies. Fig.6 presents policies #6 - #10 from Table 1. This graph reveals that such a change would not affect the number of SLA violations, which would be constant and around 5%. However, none of these combinations can achieve lower SLA violations rate than policy #3 (2%). Hence, taking into account I/O throughput at the weighting step is necessary to improve the SLA performance instead of the I/O throughput filtering.

We also modify the simulated hardware setup. We increase the number of nodes, such that the load is distributed across more storage nodes. We set again the limit for the SLA equal to 450 IOPS, and investigate how the number of nodes affect the SLA violations. In Fig.7, we observe that the throughput policy that we proposed requires eight storage nodes to decrease the SLA violation to less than 0.5%. On the other hand, both the default scheduling policies in Cinder require more than 20 nodes to achieve such a performance. This shows that the proposed scheduling policy can decrease the capital expenses for storage nodes to almost half of what it would be if the default policies of Cinder were used.

Finally, we also look into the volume speed performance. Fig. 8 shows a box plot of the volume speed. All policies have the same median value. Not surprisingly, the chance weighing policy has the largest variance because the decision is made randomly. The distribution of the available capacity and throughput policy is very close. Nevertheless, I/O throughput still achieves higher minimum value and there is no outlier. We may conclude that I/O throughput weighing policy have better volume speed performance than the default available capacity policy.

## V. Conclusion

In this paper, we focused on volume request scheduling policies that can reduce the I/O throughput SLA violations in cloud storage systems. To achieve this goal, we proposed multiple SLA-aware scheduling policies based on the OpenStack Cinder scheduler model. Our simulation results indicate that the combination of capacity filtering and I/O throughput weighing policy reduces the I/O throughput SLA violation rate. Furthermore, the volume speed increases and the number of nodes to satisfy the SLA are decreased. Overall, the proposed policies can significantly reduce the capital expenditures for the cloud storage providers.

As a future work, we plan to contribute the cloud storage scheduling algorithms code base to the OpenStack community. At the same time, we are deploying our own private cloud in order to perform experiments on bare metal servers using Commercial Off-The-Shelf (COTS) hardware.

## References

[1] R. Ghosh, I. Papapanagiotou and K. Boloor, “A Survey on Research Initiatives for Healthcare Clouds,” in Cloud Computing Applications for Quality Health Care Delivery, 2014, pp. 1-18.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I. Stoica, “A view of cloud computing,” Communications of the ACM, vol. 53, pp. 50-58, 2010.

[3] “Openstack,” [Online]. Available: http://www.openstack.org/.

[4] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, “The eucalyptus open-source cloud-computing system,” in Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on, 2009.

[5] “Amazon Elastic Block Store,” [Online]. Available: http://aws.amazon.com/ebs/details/.

[6] A. Gupta, L. V. Kale, D. Milojicic, P. Faraboschi and S. M. Balle, “ HPC-Aware VM Placement in Infrastructure Clouds,” in Cloud Engineering (IC2E), 2013 IEEE International Conference on, 2013.

[7] H. N. Van, F. D. Tran and J.-M. Menaud, “SLA-aware virtual resource management for cloud infrastructures,” in Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on, 2009.

[8] L. Wu, S. K. Garg and R. Buyya, “Sla-based resource allocation for software as a service provider (saas) in cloud computing environments, ” in Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on, 2011.

[9] A. Beloglazov, J. Abawajy and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” Future Generation Computer Systems, vol. 28, issue 5, pp. 755-768, 2012.

[10] C.-C. Lin, P. Liu and J.-J. Wu, “Energy-aware virtual machine dynamic provision and scheduling for cloud computing,” in Cloud Computing (CLOUD), 2011 IEEE International Conference on, 2011.

[11] H. Goudarzi and M. Pedram, “Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems, ” in Cloud Computing (CLOUD), 2011 IEEE International Conference on, 2011.

[12] J. Wang, P. Varman and C. Xie, “Avoiding performance fluctuation in cloud storage,” in High Performance Computing (HiPC), 2010 International Conference on, 2010.

[13] “OpenStack Cinder,” [Online]. Available: https://wiki.openstack.org/wiki/Cinder.

[14] K. Hui, “Laying Cinder Block (Volumes) In OpenStack, Part 2: Solutions Design,” 2014. [Online]. Available: http://www.rackspace.com/blog/laying-cinder-block-volumes-in-openstack-part-2-solutions-design/.

[15] A. W. Leung, S. Pasupathy, G. R. Goodson and E. L. Miller, “ Measurement and Analysis of Large-Scale Network File System Workloads.,” in USENIX Annual Technical Conference, 2008.

[16] S. Liu, X. Huang, H. Fu and G. Yang, “Understanding Data Characteristics and Access Patterns in a Cloud Storage System,” in Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, 2013.

[17] “RAID Performance Calculator,” [Online]. Available: http://wintelguy.com/raidperf.pl.