HIOBS: A Block Storage Scheduling Approach to Reduce Performance Fragmentation in Heterogeneous Cloud Environments

Denis M. Cavalcante, Flávio R. C. Sousa, Manoel Rui P. Paula,
Eduardo Rodrigues, José S. Costa Filho, Javam C. Machado and Neuman Souza
LSBD, Departament of Computer Science, Federal University of Ceara, Fortaleza, Brazil

Keywords: Cloud Resource Fragmentation, Cloud Block Storage Scheduling, Service Level Agreements (SLAs), Heterogeneous Storage Resources, I/O throughput (IOPS).

Abstract: Cloud computing is a highly successful paradigm of service-oriented computing and has revolutionized the usage of computing infrastructure. In the cloud storage, the service user has requirements regarding availability and performance. Once the cloud resources are shared among multi-tenants, a service level agreement (SLA) is defined by the service provider and the user. The multi-tenant characteristic of the cloud implies a heterogeneity of SLA requirements from users. At the same time, cloud providers should upgrade their infrastructure with modern storage nodes to attend data-driven applications, resulting in a heterogeneous cloud environment. The heterogeneity of both SLA requirements and storage resources makes the volume scheduling problem complex to guarantee SLAs. This paper presents HIOBS, an SLA-aware approach for block storage scheduling in heterogeneous cloud environments to reduce the performance fragmentation of the available storage resources, thus increasing the chances of new SLA requirements to be met. We demonstrate through experiments that our method improves more than 40% the rate of SLA violations while using fewer storage nodes.

1 INTRODUCTION

Cloud computing is a highly successful paradigm of service-oriented computing and has revolutionized the way computing infrastructure is abstracted and used. Scalability, elasticity, pay-per-use pricing, and economies of scale are the primary reasons for the successful and widespread adoption of Infrastructure as a Service (IaaS) (Agrawal et al., 2011). In IaaS model, cloud providers usually offer services of computing, raw (block) storage and networking to service users (Manvi and Shyam, 2014). In that case, a consolidated cloud environment has to handle loads of heterogeneous applications and requests as well as heterogeneous equipment classes, representing different options in configuring computing power, networking and storage (Reiss et al., 2012).

Users of cloud block storage service (CBS-Service) often have at least two possibilities of Service Level Agreements (SLA) requirements regarding availability and performance (Frey et al., 2013) (Yao et al., 2014). The performance aspect, for example, may have heterogeneous performance demands because data analytics workloads are often CPU-intensive but other times are I/O-intensive (Lee and Katz, 2011). Those heterogeneous resource demands led cloud providers to support performance customization not only for CPU resources, but for disk I/O throughput resources as well. Popular cloud storage services on the market such as Amazon Web Services (AWS) platform and Google Cloud Platform support the determination of performance characteristics in terms of SLA-IOPS (Amazon, 2017) (Google, 2017), in which the Input/Output Operations Per Second (IOPS) is defined by a Service Level Objective (SLO). SLOs are specific measurable characteristics of the SLA such as availability, throughput, frequency, response time, or latency (Yao et al., 2015). In cloud storage service, I/O throughput SLA is defined as the I/O throughput of users volume is higher or equal to a specific number of input/output operations per second in at least 99.9% of time.

Figure 1 shows the basic flow of a CBS-Service with heterogeneous SLA-IOPS support. First, the service user requests the provisioning of a new volume with custom SLA-IOPS requirements. Next,

M. Cavalcante, D., R. C. Sousa, F., Paula, M., Rodrigues, E., S. Costa Filho, J., Machado, J. and Souza, N.
HIOBS: A Block Storage Scheduling Approach to Reduce Performance Fragmentation in Heterogeneous Cloud Environments.
DOI: 10.5220/0006686800900099
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 90-99
ISBN: 978-989-758-295-0

the CBS-Service sends the cloud block storage provisioning request (CBS-Request) to the cloud block storage scheduler (CBS-scheduler) which takes the volume placement decision. A failed provisioning request prevent any usage of the demanded volume. A succeeded provisioning request enables the service user to perform data access into the new storage volume where the cloud provider promises to be able to serve data access into the new volume according to the SLA-IOPS.

The volume placement decision is taken according to the following three points: (a) the storage and performance (SLA-IOPS) requirements of the CBS-Request, (b) the storage and performance available capacities of the cloud block storage resource Pool (CBS-Pool), in which the available capacities of the CBS-Pool is often the aggregation of all the storage nodes (SNs); and finally (c), the CBS-scheduler algorithm, which aims to select the best storage node (SN) to host the new volume within a short period of time.

A popular and open source on market platform for CBS-Service (OpenStack, 2017a), Cinder, is part of a larger project of cloud computing, OpenStack (OpenStack, 2017b). OpenStack is also composed of other projects where each manages a different cloud resource, like computing, networking, object storage and so on. OpenStack-Cinder is a very scalable implementation of a cloud block storage service. Several CBS-Scheduler algorithms are available in Cinder source code, but they do not consider performance metrics such as IOPS.

The works (Yao et al., 2014) and (Yao et al., 2015) addressed lacking of support for performance requirements in block storage scheduling algorithms. They both modeled their solutions considering IOPS to schedule new volumes onto storage resources. However, both works did not evaluate their solutions for heterogeneous cloud environments regarding IOPS capacity. Scheduling algorithms to heterogeneous environments have been proposed in (Lee, 2012), but it focused its solutions to the management of computing resources.

It is already known that honoring a variety of SLA-IOPS and yet guarantying a certain level of Quality of Service (QoS) imposes a robust constraint and increases the complexity of a scheduling decision (Wang and Veeravalli, 2017). The heterogeneous aspect of a cloud environment implies in additional importance to the subject because a naive design of a CBS-Scheduler may lead to cloud IOPS-resource fragmentation (Sridharan et al., 2011). Cloud IOPS-resource fragmentation happens when the CBS-Scheduler takes poor scheduling decisions that cause inefficient utilization of storage performance. Hence, in order to meet QoS, extra resources are utilized which increases service provider expenditures. A detailed example of IOPS-resource fragmentation is explained in section 3.

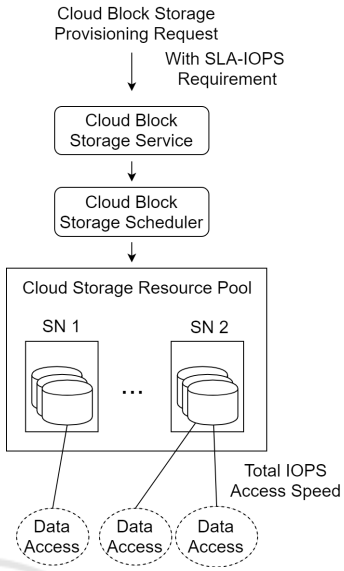


Figure 1: Cloud Block Storage Service.

The lacking of suitable algorithms may lead cloud providers to add extra storage resources to reach the desired level of QoS regarding performance requirements. That occurs because they do not efficiently provision volumes increasing the matching between a CBS-Request and a storage node. To the best of our knowledge, this is the first paper to formulate and explore the problem of how to schedule block storage volumes with the goal of reducing IOPS-resource fragmentation for heterogeneous cloud environment with support to heterogeneous SLA-IOPS.

The major contributions of this paper are as follows:

- The presention of HIOBS, an SLA-aware approach for block storage scheduling in heterogeneous cloud environments. While other approaches have been proposed and evaluated their solutions for homogeneous SLA IOPS requirements and cloud storage resources, our approach is designed and assessed to address the complexity of heterogeneity of CBS-Requests in terms of performance(SLA IOPS) and storage resources.
- The implemention and performance of a set of experiments to evaluate our approach against the default OpenStack-Cinder approach as well as the best approach proposed by the related work (Yao et al., 2014). We demonstrate through experiments that our method not only minimizes SLA violations but also uses fewer storage nodes.

Organization: This paper is organized as follows: Section 2 discusses related work. Section 3 explains HIOBS's theoretical aspects and the implementation of the solution. The evaluation of HIOBS is presented in Section 4. Section 5 presents the conclusions.

2 RELATED WORK

OpenStack-Cinder (OpenStack, 2017a) implements cloud block storage as a service by managing physical storage resources. The core functionality of cinder is to support the creation and deletion of volumes as well as the attachment of volumes to virtual machines by a self-service Web API. These volumes are allocated to storage nodes. Theoretically, while the maximum storage capacity of a storage node is not reached, more volumes can be assigned to that storage node. Cinder scheduling algorithms can reduce infrastructure costs by increasing the sharing of physical storage resources, but they lack support for any performance guarantee demanded by SLA requirements.

The paper (Tremblay et al., 2016) categorizes four research challenges: workload characterization, storage node characterization, storage node selection and online storage node tuning. Our proposed algorithm may be classified as a storage node selection research. In the paper, the necessity of better algorithms for software-defined storage (SDS) solutions is raised. It is proposed a workload Aware Storage Platform (WASP), which advantage on deployment automation and the deployability of the SDS solutions, allowing applications to specify the expected workload, determining storage node solutions to serve the workload, and performing timely reconfiguration based on the changing behavior of the workload over time. The paper only suggests directions, not applying any experimentation of its ideas.

In the work (Yao et al., 2014), the authors use the OpenStack Cinder as a study case for volume scheduling in the cloud. They proposed a volume scheduling algorithm with the goal of enabling the cloud management based on IOPS ratio performance to minimize SLA IOPS violations. The authors considered the available storage and IOPS capacities from storage nodes for allocating each new volume request. Precisely, they explored many heuristics at combining IOPS ratio and storage capacities of the storage nodes. Although the paper presented efficiency at reducing the SLA IOPS violations at considering the available IOPS ratio capacity to select the most suitable storage node, the paper did not evaluate heterogeneous IOPS ratio for SLA IOPS requests and storage nodes.

The work (Yao et al., 2015) aimed to solve the volume scheduling problem as well as in (Yao et al., 2014). However, in (Yao et al., 2015), the problem was modeled as a NP-complete problem (Yao et al., 2015) and treated as a Vector Bin Package Problem (Gary and Johnson, 1979) with d dimensions. Where each dimension d represents the capability of a system resource like CPU, memory, IOPS ratio, storage capacity and so on. Then they proposed an objective function to minimize the number of storage nodes used to meet the volume requests' requirements. For this reason, the authors proposed a heuristic based on *Standard Best Fit (BF)* heuristic named Modified Vector BFD (MVBFD) to process a set of volume requests arrived for each short period. This proposed algorithm was also not evaluated for heterogeneous SLAs requests and storage nodes. That work also did not analyze the size of the time window and its impact on processing volume requests.

3 HIOBS APPROACH

HIOBS is an SLA-aware approach for block storage scheduling in heterogeneous cloud environments to improve the QoS of the SLA IOPS requirements. SLA IOPS requirements make the cloud block storage scheduling problem more challenging because they are directly impacted by the IOPS capacity of the storage nodes. The main difficulty to provide QoS to SLA IOPS requirements in heterogeneous environments is to handle the performance fragmentation of the available IOPS of the storage nodes.

3.1 Cloud Block Storage Heterogeneous Problem

Consider that a user of the cloud block storage service needs two new volumes to attach to its virtual machines. Those two volumes will be used by data-driven applications with different SLA IOPS requirements, i.e. performance requirements, for working properly. The first application needs a volume capable of at least 150 IOPS whereas the second application needs at least 300 IOPS. Also, consider that a cloud block storage provider has available three storage nodes with heterogeneous IOPS capacity. The IOPS capacities of the A, B and C storage nodes are 150, 250 and 350 respectively. Consider now, in the Figure 2, that the user firstly requested Volume 1 with SLA IOPS of 150. A scheduling policy as in (Yao et al., 2014) that takes into account the available IOPS of storage node may choose the storage node C because it has the largest available IOPS capacity to host the

Volume 1. When the user requests the second volume with SLA IOPS of 300, the cloud provider has no longer any storage node available to provide the requested IOPS to the Volume 2. In this work, we define this problem as a performance fragmentation of the total available IOPS of the storage nodes. As consequence, the problem described may lead to poor performance of cloud user applications.

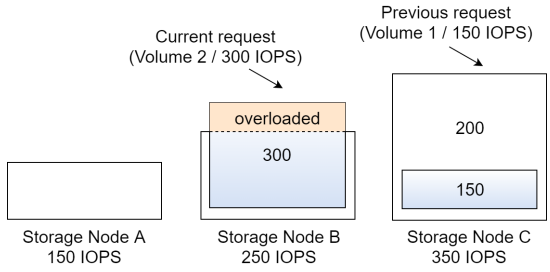


Figure 2: Performance Fragmentation Problem.

3.2 HIOBS Cloud Block Storage Scheduling

When a cloud user demands a new volume with SLA requirements, a cloud block storage scheduling algorithm should analyze the CBS-Request to select the proper storage node to host the new volume. This process should be properly handled by any service of cloud block storage. From our understanding of the OpenStack-Cinder, we extended its cloud block storage scheduling framework with more phases where it is possible to insert, change, append or remove behaviors from each phase. As shown in Figure 3, the four phases of this framework are: (1) retrieval of the meta-data of each storage node, (2) filtering storage nodes by removing those ones non-capable according to some criteria, (3) evaluation of the preference of each storage node according to HIOBS function cost and (4) selection of the storage nod to host the new volume. Below, it is described how HIOBS uses this framework to handle volumes CBS-Requests with SLA requirements for cloud heterogeneous environments.

3.2.1 HIOBS Retrieval Storage Nodes Meta-data Phase

From each storage node available to the cloud, HIOBS maintains four meta-data where two of them are dynamic. The static meta-data are the maximum storage capacity and the maximum IOPS capacity. The dynamic meta-data are the total used storage and the total used IOPS. The dynamic meta-data are evaluated every time a new volume CBS-Request is accepted. In the scenario where a storage node has

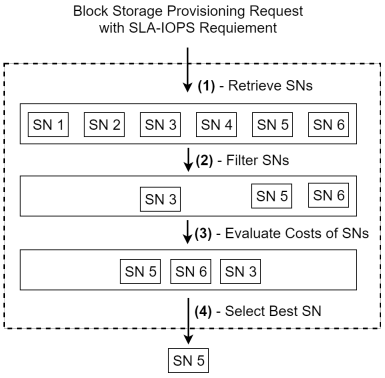


Figure 3: Cloud Block Storage Scheduling.

100 GB and 400 IOPS for maximum storage capacity and maximum IOPS capacity respectively, if two volumes requesting 10GB and 200 IOPS are allocated to that storage node, the total used storage and the total used IOPS are evaluated as 20 GB and 400 IOPS. From these 4 meta-data, it is possible to evaluate the available storage and IOPS capacities meta-data.

3.2.2 HIOBS Filtering Phase

HIOBS uses available storage capacity as the criterion to remove non-capable storage nodes. HIOBS decision avoids problems related to insufficient storage by discarding storage nodes with no available storage capacity to attend the volume CBS-Request.

3.2.3 HIOBS Cost Evaluation Phase

The intuition behind HIOBS cost evaluation phase is fundamentally based on avoiding the performance fragmentation problem mentioned at Section 3.1. This mission seems difficult because volume requests arrive one at a time in unknown order as stated by (Yao et al., 2014). Once the cloud block storage framework schedules volumes immediately, HIOBS also does not know beforehand the IOPS demanded by next volume CBS-Request. By this means, HIOBS evaluates the fragmentation cost of every storage node according to the Equation 1 where b is the available IOPS capacity of a storage node and v is the SLA IOPS requirement of the new volume request.

$$FragmentationCost = b - v; \quad (1)$$

3.2.4 HIOBS Selection Phase

After fragmentation cost evaluation of every storage node in the previous phase, HIOBS selects the storage node with the smallest fragmentation cost to host the new volume CBS-Request. This decision reduces the

performance fragmentation problem presented in Figure 2. The same scenario is presented below, but now using HIOBS approach to exemplify how the performance fragmentation is reduced. In Figure 4, the volume 1 requested 150 IOPS and was scheduled to the storage node A which supports 150 IOPS. The volume 2 requested 300 IOPS and was scheduled to the storage node C which supports until 350 IOPS. This way, by using our approach, the volume requirement was met.

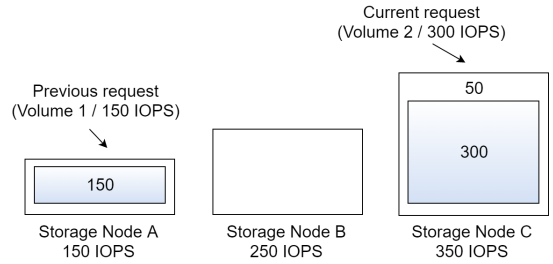


Figure 4: HIOBS Block Storage Scheduling Example.

3.3 HIOBS Algorithm

In Algorithm 1 we show the pseudo-code of the implemented algorithm. The worst case complexity of the algorithm is $O(n \log n)$, where n is the number of storage nodes. This algorithm has two inputs: the available storage nodes list *snl* and the current volume CBS-Request *vr*. As output, the algorithm returns the best storage node to meet the volume requirement. From the Line 2 to 7, it is verified which storage nodes have available storage capacity *sn.availableStorage* to attend the volume storage *vr.STORAGE*. Only the volumes that meet the storage capacity filtering are added to the $filteredStorageNodeList$ as described at line 4.

At Line 9, HIOBS evaluates the available IOPS of each storage node of the filtered storage nodes $filteredStorageNodeList$. Next, the available IOPS *b.availableIOPS* and the SLA IOPS volume requirement *vr.SLA_IOPS* are processed to evaluate the fragmentation cost of each storage node as described at Line 10. A storage node with zero value for performance fragmentation cost *FragmentationCost* means that its available IOPS will not be fragmented at all whether the current volume CBS-Request be scheduled to that storage node. The greater the positive $fragmentationCost$ value, the higher the storage node performance fragmentation is whereas negative values mean, a storage node does not have available IOPS anymore. In our implementation, HIOBS puts a storage node with negative $fragmentationCost$ value as the last option to host a volume using the *BIG_CONSTANT* value as shown at Line 11. As the final step at Line 16, HIOBS picks the storage node with smallest fragmentation cost to host the new volume.

```
   Input: vr (volumeRequest), snl (storageNodeList)
   Output: selectedStorageNode
 1 filteredStorageNodeList.clear()
 2 foreach Storage Node sn in the snl do
 3 |   sn.availableStorage = sn.TOTAL_STORAGE
   |     − sn.usedStorage
 4 |   if sn.availableStorage >= vr.STORAGE then
 5 |   |   filteredStorageNodeList.append(sn)
 6 |   end
 7 end
 8 foreach Storage Node sn in the
   |   filteredStorageNodeList do
 9 |   sn.availableIOPS = sn.TOTAL_IOPS −
   |     sn.usedIOPS
10 |   sn.fragmentationCost = sn.availableIOPS −
   |     vr.SLA_IOPS
11 |   if sn.fragmentationCost < 0 then
12 |   |   sn.fragmentationCost =
   |   |     BIG_CONSTANT
13 |   end
14 end
15 sortByFragmentationCost(filteredStorageNodeList)
16 return min(filteredStorageNodeList)
```

Algorithm 1: HIOBS Performance Fragmentation Cost Algorithm.

4 EXPERIMENTAL EVALUATION

4.1 Compared Approaches

The authors of the work (Yao et al., 2014) described and evaluated several approaches including the ones implemented by the OpenStack-Cinder as well as their proposals based on IOPS. Among those, we chose the default OpenStack-Cinder approach based on available storage capacity and the best approach (SLA-Aware to homogeneous cloud environments) proposed by (Yao et al., 2014) to compare against our (HIOBS). Based on the phases of the Cloud Block Storage Scheduling framework mentioned at Section 3.2, we summarized the compared approaches at Table 1.

We decided not to compare HIOBS with the method presented in (Yao et al., 2015) because their block scheduling algorithm considers more than one CBS-Request at a time. Meanwhile, all approaches in Table 1 have the similarity to make a scheduling decision at the moment that a new volume CBS-Request arrives.

4.2 Block Storage Simulator

To prove the effectiveness of the HIOBS and evaluate our approach against the other two previously descri-

Table 1: Approaches in comparison.

Phase	Cinder	SLA-aware	**HIOBS**
1	Avail. Stor.	Avail. Stor.	Avail. Stor.
2	Avail. Stor.	Avail. Stor.	Avail. Stor.
3	Avail. Stor.	Avail. IOPS	IOPS Matching
4	Avail. Stor.	Avail. IOPS	IOPS Matching

bed in Table 1, we performed a set of experiments on a simulator of a cloud block storage schedule.

The original simulator was designed to evaluate a cloud block storage using different scheduling policies in which SLA IOPS requirements were homogeneous as well as the cloud environment[1]. Homogeneous SLA IOPS means that all volume CBS-Requests demand the same IOPS. Homogeneous cloud environment means that all storage nodes have the same IOPS capacity. On the original simulator design, the IOPS consumption of each volume CBS-Request is performed according to the Equation 2. In other words, the total IOPS capacity of a storage node is split evenly between its allocated volumes.

$$IOPS = MaxStorageNodeIOPS/TotalAllocatedVolumes; \quad (2)$$

The original design of the simulator has poor management for heterogeneous SLA IOPS requirements and storage nodes. The Equation is not able to split the total IOPS capacity of a storage node between its allocated volumes accurately because the volumes with higher IOPS demands will suffer more SLA IOPS penalties. For example, if two volume CBS-Requests, A and B, with SLA IOPS requirements of 400 and 500 respectively, are allocated to a storage node with 900 IOPS capacity. The original volume IOPS consumption gives 450 IOPS to each volume, which means that SLA IOPS of B suffers unnecessary SLA violation.

We extended the simulator with a new IOPS control mechanism to better control the volume IOPS consumption for heterogeneous SLA IOPS and storage nodes[2]. The IOPS control mechanism limits the maximum IOPS use of each volume to the SLA IOPS requirement, i.e., each volume can consume only the amount of IOPS it requested. Considering the previous example, but with the new version of the simulator, both volumes, A and B, would have their SLA IOPS attended without any SLA violation. The extra IOPS demanded above a storage node capacity is split evenly as SLA IOPS penalty to its allocated volumes. This mechanism is exemplified at the Figure 5, where at t_1 a storage node with 2000 IOPS capacity hosts volume 1, 2, 3 with 800, 500 and 500 SLA IOPS respectively. At t_2, the volume 4 with 600 SLA IOPS is also allocated to the same storage node. At this time, the storage node is at an overloaded state. At t_3, the IOPS control handles the extra 400 IOPS above the storage node IOPS capacity by limiting the maximum IOPS of each volume evenly, i.e., the IOPS of the four volumes will by reduced by 100 IOPS.

[1]https://github.com/ipapapa/OpenStackCinderSimulator
[2]https://github.com/ResearchLSBD/HIOBS

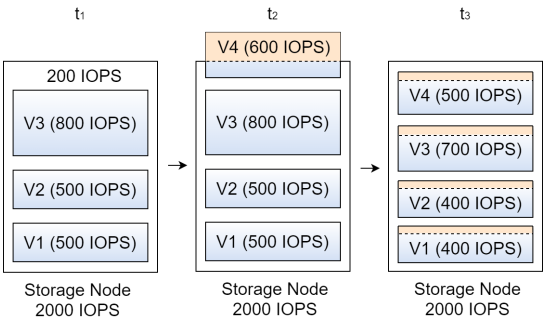


Figure 5: IOPS consumption Control.

Table 2: Experiment Setup.

Volume Size (GB)	100, 500, 1000
Number of Iterations	50
Seconds per Iteration	120000
Mean Volume Duration (s)	600 (Poisson)
Mean Arrival Time (s)	20 (Poisson)
Total Volume CBS-Requests	5000
Number of Storage Nodes	2-20

4.3 Environment Description

Table 2 describes our experiment parameters based on the setup used by the work (Yao et al., 2014). Our experiment ran 50 iterations of 120000 minutes. At every iteration, a new workload was created randomly based on the following parameters: size of volumes 100GB, 500GB or 1 TB; arrival time and the duration of each volume sampled from a Poisson distribution with a mean of 20 and 600 respectively. The number of volumes requested during one iteration was limited to 5000 CBS-Requests or the maximum of requests made during the experiment execution.

4.4 Simulation Setup

We have proposed different scenarios for experimentation as shown in table 3. The second column shows the possible values for the SLA IOPS requirement of each volume CBS-Request. The third column shows possible values of the maximum I/O throughput of a storage node.

The first scenario is the same homogeneous one evaluated by (Yao et al., 2014), where each volume request demands an SLA IOPS of 450 and each storage node of the cloud block storage has an IOPS capacity

of 1948.

To assess our algorithm with heterogeneous values for SLA IOPS and storage node IOPS capacity, we considered the Scenarios 2 and 3, where each volume requests an SLA IOPS of 200, 300 or 850. These SLA IOPS were chosen to simulate an environment where hte majority of applications do not demand high SLA IOPS, whereas the minority have critical and high demands for SLA IOPS.

The storage node IOPS capacities for the Scenario 2 were chosen to simulate an environment where 25% of the IOPS capacity is low, 50% of IOPS capacity is average, and 25% of the IOPS capacity is high. The storage node IOPS capabilities for the Scenario 3 were chosen to simulate an environment where the minority of storage nodes, i.e., 40%, can provide a high IOPS capacity of 4000, whereas the majority, i.e., 60%, can provide only low IOPS capacities of 500 and 700.

Table 3: Scenarios Setup.

Scenario	SLA (IOPS)	S.N. (IOPS)
1	450	1948
2	200,300,850	974,1948,2922
3	200,300,850	500,700, 4000

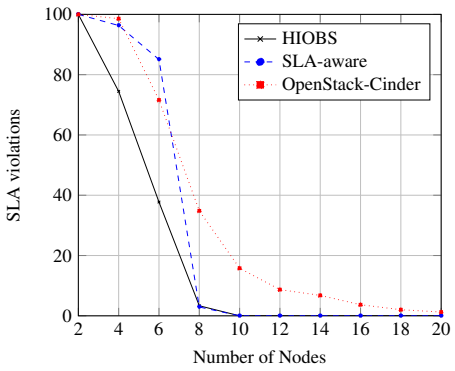
4.5 Experiment Results

For each scenario described at Table 3, we present three metrics to explain the results, comparing the approaches described at the Table 1:

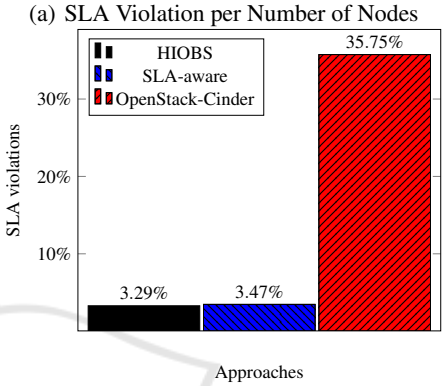
- **SLA Violation per Number of Nodes** describes the behavior the rate of SLA violations for different numbers of storage nodes until any approach reaches zero SLA violation.
- **SLA Violations per Approach** was chosen to compare the results before of the moment of reaching zero SLA violations (the minimum number of nodes to have zero SLA violations), once that condition is too strict and not easily reachable by cloud providers.
- **Volume Speed per Approach** shows the distribution of IOPS consumption obtained by each approach.

4.5.1 Scenario 1 (Homogeneous)

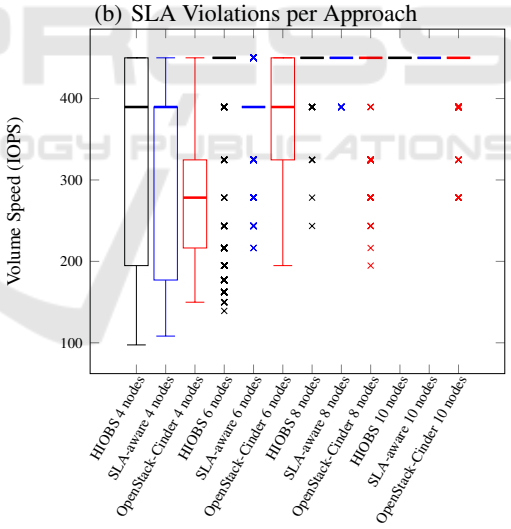
Evaluating the number SLA violations per number of storage nodes for the Scenario 1 in Figure 6(a), we note that our approach had less SLA violations than the SLA-aware approach and the OpenStack-Cinder default approach. While Cinder default approach requires more than twenty nodes to reach zero SLA violations, HIOBS and the SLA-Aware baseline require



(a) SLA Violation per Number of Nodes



(b) SLA Violations per Approach



(c) Volume Speed per Approach

Figure 6: Scenario 1.

only ten storage nodes to reach zero SLA violations. Unlike the other approaches, HIOBS notably reduced the SLA violations even when the minimal number of storage nodes was not properly set up to support the SLA IOPS of volume requests. Considering the moment before reaching zero SLA violations, i.e. eight nodes, the default OpenStack-Cinder approach faces the worst performance with 35.75% of SLA violati-

ons. HIOBS and the SLA-aware (Yao et al., 2014) approaches face similar SLA violations around 3.3% as shown at the Figure 6(b).

Analyzing the results of volume speed per approach as illustrated at the Figure 6(c), the volume requests under the SLA-aware and our approach obtained the IOPS consumption closer to the claimed than the default OpenStack Cinder capacity approach. Comparing ours and SLA-aware approaches, SLA-Aware has slightly better volume speed as shown at around six and eight nodes. This happened because the SLA-aware approach does not seek greater resource sharing, i.e, it always seeks to allocate new volumes to the storage nodes that have the highest available IOPS.

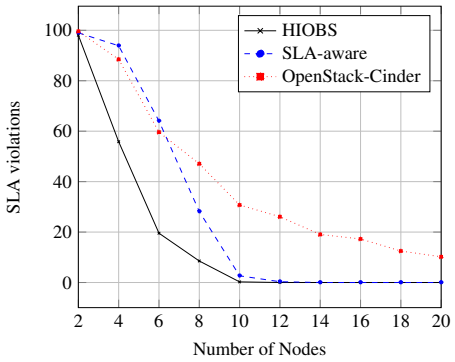
4.5.2 Scenario 2

Evaluating the number of SLA violations per number of storage nodes for the Scenario 2 in Figure 7(a), we can easily see that our approach had much fewer SLA violations than the others approaches from four nodes until ten nodes. Using the minimum number of nodes before reaching zero SLA violations as shown in Figure 7(b), our approach faced only 8.53% SLA violations, while SLA-aware and Cinder default approaches faced 28.16% and 47.09% SLA violations respectively. This improvement happened because our approach has complied better the SLA IOPS requirements, even reducing significantly the fragmentation of the available IOPS of the storage node. In other words, once the requirements are being met without allocating the current volumes to storage nodes with the most available IOPS, new volume requests have more chances to be met even with big requirements.

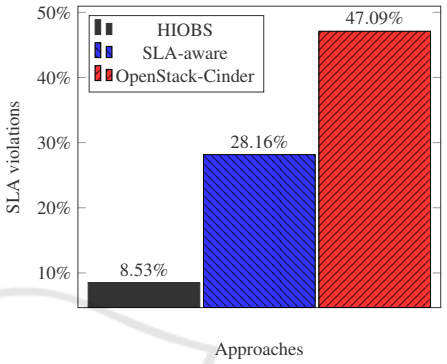
Our approach also required fewer storage nodes than the SLA-aware baseline to reach zero SLA violations. On the other hand, Cinder default method was not able to reach zero SLA violations with less than twenty nodes. Analyzing the volume speed per approach in Figure 7(c), the HIOBS and SLA-aware approaches had better performance than Cinder default approach, which faced some IOPS consumption below the smallest SLA IOPS requirements.

4.5.3 Scenario 3

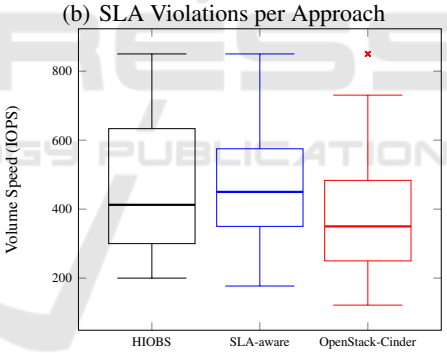
Studying the number of SLA violations per number of storage nodes for the Scenario 3 in Figure 8(a), our approach achieved similar results to the Scenario 2 where it was needed two less nodes than the SLA-aware approach to reach zero SLA violations. From four nodes until before reaching zero SLA violations, it is again possible to verify that our approach had less SLA violations than the other approaches. While



(a) SLA Violation per Number of Nodes



(b) SLA Violations per Approach



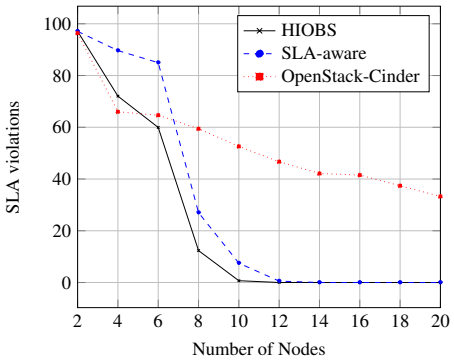
(c) Volume Speed per Approach

Figure 7: Scenario 2.

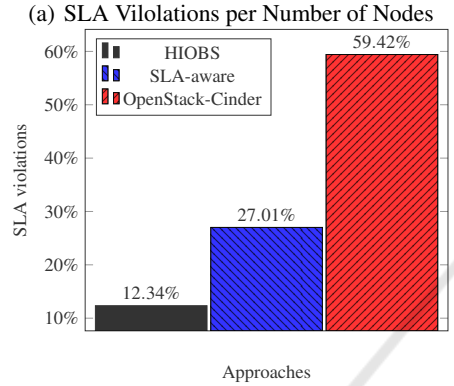
our approach faced 12.34% for SLA violations, SLA-aware and Cinder Default faced 27.01% and 59.45% respectively as shown in Figure 8(b). The volume speed per approach also had similar proportional results to the Scenario 2 as illustrated in Figure 8(a).

5 CONCLUSION AND FUTURE WORK

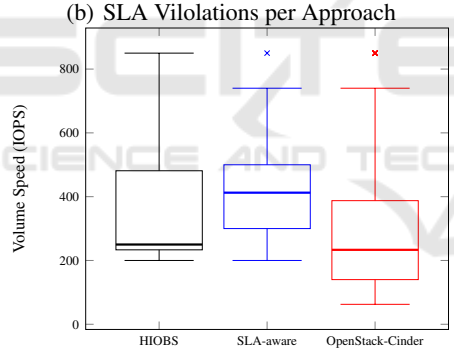
This work presented HIOBS, an SLA-aware approach for block storage scheduling in heterogeneous cloud environments to improve QoS of the SLA IOPS re-



(a) SLA Vilolations per Number of Nodes



(b) SLA Vilolations per Approach



(c) Volume Speed per Approach

Figure 8: Scenario 3.

quirements. SLA IOPS requirements makes the cloud block storage scheduling problem more challenging because they are directly impacted by the IOPS capacity of the storage nodes. We evaluated our approach by considering the homogeneous environment scenario proposed by (Yao et al., 2014) as well as two different heterogeneous scenarios.

As seen in the results, our approach significantly improved QoS, i.e., it presented less SLA violations in comparison to the other methods since it reduced the performance fragmentation of the available storage nodes, thus increasing the chances of new SLA IOPS requirements to be properly attended. Our strategy also manages volume requests to use fewer storage nodes once it maximizes the resource sharing at allocating as much possible volumes into one storage node. On the other hand, SLA-aware approach requires more nodes to meet the requirements, since it considers the available IOPS at the cost evaluation phase, thus giving preference to nodes not used yet. Our approach is not the fastest at providing IOPS consumption as observed in the homogeneous scenario, due to the high resource competition from maximizing resource sharing, thus delaying HIOBS to reach the volume speed near to the requested.

As future work, we intend to implement HIOBS in a production-like environment and validate our volume scheduling solution conducting a set of experiments in the cloud block storage OpenStack Cinder. As new directions to improve HIOBS, we plan to test new scenarios for heterogeneous SLA requirements and cloud storage resources as well as for new distributions to arrival and duration time of SLA requests. Also, we will explore techniques such as volume migration and forecasting models to predict future SLA requirements, thus making the volume scheduling based on HIOBS more robust.

ACKNOWLEDGEMENTS

This work was partially funded by Lenovo, as part of its R&D investment under Brazil's Informatics Law, by grant from Capes/Brazil and also by LSBD/UFC.

REFERENCES

Agrawal, D., Das, S., and El Abbadi, A. (2011). Big data and cloud computing: Current state and future opportunities. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 530–533, New York, NY, USA. ACM.

Amazon (2017). *Elastic Block Store (EBS)*. https://aws.amazon.com/ebs/.

Frey, S., Reich, C., and Lüthje, C. (2013). Key performance indicators for cloud computing slas. In *The Fifth International Conference on Emerging Network Intelligence, EMERGING*, pages 60–64.

Gary, M. R. and Johnson, D. S. (1979). Computers and intractability: A guide to the theory of np-completeness.

Google (2017). *Optimizing Persistent Disk and Local SSD Performance*. https://cloud.google.com/compute/docs/disks/performance.

Lee, G. (2012). *Resource allocation and scheduling in heterogeneous cloud environments*. PhD thesis, UNIVERSITY OF CALIFORNIA, BERKELEY.

Lee, G. and Katz, R. H. (2011). Heterogeneity-aware resource allocation and scheduling in the cloud. In *HotCloud*.

Manvi, S. S. and Shyam, G. K. (2014). Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of Network and Computer Applications*, 41:424–440.

OpenStack (2017a). *Cinder*. https://wiki.openstack.org/wiki/Cinder.

OpenStack (2017b). *Open Source Cloud Computing Software*. https://www.openstack.org.

Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7. ACM.

Sridharan, M., Calyam, P., Venkataraman, A., and Berryman, A. (2011). Defragmentation of resources in virtual desktop clouds for cost-aware utility-optimal allocation. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 253–260. IEEE.

Tremblay, B., Kozubal, K., Li, W., and Padala, C. (2016). A workload aware storage platform for large scale computing environments: Challenges and proposed directions. In *Proceedings of the ACM 7th Workshop on Scientific Cloud Computing*, pages 27–33. ACM.

Wang, X. and Veeravalli, B. (2017). A genetic algorithm based efficient static load distribution strategy for handling large-scale workloads on sustainable computing systems. In *Intelligent Decision Support Systems for Sustainable Computing*, pages 7–31. Springer.

Yao, Z., Papapanagiotou, I., and Callaway, R. D. (2014). Sla-aware resource scheduling for cloud storage. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 14–19.

Yao, Z., Papapanagiotou, I., and Callaway, R. D. (2015). Multi-dimensional scheduling in cloud storage systems. In *2015 IEEE International Conference on Communications (ICC)*, pages 395–400. IEEE.