EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

SLA-aware resource scheduling algorithm for cloud storage

Yong Wang[1,3†], Xiaoling Tao[1,2,3*†], Feng Zhao[1], Bo Tian[1,3] and Akshita Maradapu Vera Venkata Sai[4]

Abstract
Cloud computing is a novel computing paradigm, which connects plenty of computing resources and storage resources via Internet. Cloud computing provides a number of high-quality services, such as cloud storage, outsourcing computing, and on-demand self-service, which have been widely accepted by the public. In cloud computing, by submitting their tasks to cloud, plenty of applications share huge computation and storage resources. However, how to schedule resource efficiently is a big challenge in cloud computing.
In this paper, we propose a SLA-aware resource algorithm to enable cloud storage more efficiently. In our scheme, we take advantage of the back-end node space utilization and I/O throughput comprehensively simultaneously. We compare and contrast the existing scheduling storage policies by implementing those algorithms. The extensive tests show that our algorithm achieves a considerable improvement in terms of violation rate and the number of used hosts.

Keywords: Cloud storage, SLA, Scheduling algorithm, Weight algorithm, OpenStack

1 Introduction

Cloud computing is the development and fusion of the grid computing, parallel processing, and distributed computing, and it uses Internet to connect lots of computing resources and storage resources [1]. As a novel computing paradigm, cloud computing can provide quite cheap and dynamically scalable storage and computing services through Internet [2–4]. With the attractive advantages, more and more clients, including individual, companies and government employ cloud services, especially cloud storage service. By using cloud storage service, the resource-constraint clients can enjoy unstinted storage spaces.

In spite of tremendous advantages of cloud storage service, it also suffers many challenges at the same time. The first challenge is the security of the outsourced data in cloud [5]. Because the outsourced data might contain some private information, which should be kept secret in the data owner's view. The traditional encryption is only a partial solution to this problem because it is hard to perform meaningful operations over the ciphertext. The other one is the cloud computing resource scheduling problem. In cloud computing, the resource which client needs is constantly hanging and unpredictable. Besides, plenty of clients' applications share hardware and storage devices. Although the cloud service providers can shield the underlying hardware complexity and increase the flexibility, it still cannot control continuous change. Therefore, cloud computing resource scheduling has been a very rigorous challenge for cloud service providers.

Recently, some researchers have realized the problems for virtual (service level agreement-aware, SLA-aware for short) resource management in cloud computing [6–13]. To address the tackle SLA-aware service issue, Wada et al. [14] designed a multi-objective optimization model in 2008, which is based on the heuristic genetic algorithm. Their model takes into account the multiple SLAs, and can provide a series of solutions for equivalent quality simultaneously. However, The performance of their model could not be predicted because their proposed model is based on the provided heuristic genetic algorithm. In 2009, to solve the problem of resource provision optimization, Chaisiri et al. [15] presented an optimal virtual machine placement algorithm. In their algorithm, they used the stochastic integer programming to minimize the total

*Correspondence: txl@guet.edu.cn
†Equal contributor
†Yong Wang and Xiaoling Tao contributed equally to this work.
[1]Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, Guilin, China
[2]State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China
Full list of author information is available at the end of the article

overhead of resource provision. However, their algorithm did not take into account the notion of SLA.

In 2010, Wang et al. [16] paid attention to solving the problem of I/O performance fluctuation in cloud storage, and then, they presented an automatic optimization scheme (AOSC). In their scheme, to make the performance more stable and predictable, they considered data chunking, placement, and master-slave replication. In 2011, Goudarzi et al. [17] presented an algorithm based on force-directed search to the problem of SLA-based resource allocation in cloud computing. Besides, Lin et al. [18] and Beloglazov et al. [19] focus on energy-aware resource scheduling algorithm respectively.

In 2013, Gupta et al. [20] proposed an HPC-aware scheduler to address application-aware allocation. And then they implemented it on top of OpenStack Compute (Nova); to be specific, they deployed *n* instances to physical hosts from a single pool successfully and efficiently. In 2015, Singh and Chana [21] and Xiong and Chen [22] also studied and improved the resource management and placement for cloud computing. In 2016, Zheng et al. [23] proposed a novel scheduling policy—remaining time-based maximal (RTBM) scheduling policy. RTBM pushes forward link scheduling from a throughput-optimal and QoS concerning problem to a QoE-based and application-aware one. However, all of them only focus on the point of cloud hosting allocation, and their schemes cannot be applied to the storage volume allocation directly. However, to the best of our knowledge, less attention was paid to the issue of volume request distribution in cloud storage, of which the algorithm takes more available space or less allocated space as it inputs. Therefore, it is essential to develop and design more reasonable resource scheduling scheme to manage cloud resources and enhance system performance.

Our contributions: In this paper, to solve the problem of resources scheduling in cloud computing, we propose a novel SLA-aware resource scheduling algorithm for block storage. The main idea of our proposal is that when the scheduler requests a volume to put decisions, we take I/O throughput as one of the decision-making factors. And then, we make cloud storage services provide guaranteed space capacity and I/O throughput. The main contributions of this paper are as follows:

- We present a weight algorithm, and we integrate it with the default space filter to design an OpenStack scheduling module-based SLA aware scheduling algorithm. Compared with the previous protocols, our scheme can decrease the SLA violation rate more effectively. Besides, our scheme needs less nodes under the same violation rate, which can save the cost of hardware for cloud providers.
- We consider both of the storage space and the I/O throughput as the factors of decision-making; therefore, our scheme can provide better I/O rate for volume requests. Furthermore, it can also minimize the number of active hosts and save energy consumption.

1.1 Related work

The problem of aware resource scheduling for cloud storage is highly important. For the past decades, a lot of researchers have paid attention to this problem and proposed a series of solutions [24–30].

In cloud computing, to solve the problem of resource scheduling based on SLA, Li and Guo [31] presented a new method by using stochastic integer programming in 2010. In their scheme, they extend Minimised Geometric Buchberger Algorithm(MGBA), and combined the Grobner bases theory to address the stochastic integer programming firstly. Then, they presented a method for the optimal model of SLA-based resource schedule issue. In 2010, Prasad et al. [6] wanted to use resources efficiently and presented a new method for resource allocation. In 2014, a dynamic resource management scheme was presented by Gao et al. [32]. They aimed to solve the problem of performance management and combined power. They used the merits of server consolidation and dynamic voltage to improve the efficiency in cloud data centers.

In 2014, Yao et al. [33] aimed to solve scheduling volume create requests to back-end hosts problem. They paid attention to the volume request scheduling policies, and they presented an OpenStack Cinder scheduler model-based multiple SLA-aware scheduling policies. They reduce the I/O throughput SLA violations by combining I/O throughput weighing policy and capacity filtering. Besides, their scheme can offer higher volume I/O throughput performance under fewer storage nodes. Subsequently, Yao et al. [34] proposed a Modified Vector Best Fit Decreasing algorithm (MVBFD) to address the volume allocation problem for cloud storage systems in 2015. They chose the proper storage node according to multiple resources, the volume requests and so on. Their scheme can perform better and be suitable for most cloud platforms. Besides, they presented a trace-driven research methodology [35].

In 2016, Babak et al. [36] studied two candidate algorithms, which based on two-phase learning and feedback learning, respectively. And then they presented a self-learning algorithm, which applied machine learning to help make scheduling decisions. Without knowing any knowledge of the underlying hardware, their algorithm can dynamically adapt according to the workload and offer scheduling decisions efficiently. Besides, their algorithm can increase the cloud storage resource utilization and reduce the cost of infrastructure. In 2017, Babak et al. [37] proposed a block software-defined storage (BSDS) for cloud block storage, which can be seen as a step of designing software defined storage (SDS) system. Their scheme was based on a self-learning black box scheduler as introduced in [36]. Their scheme can provide quality of service under knowing nothing about the underlying storage hardware. Their scheme can also integrate with OpenStack Cinder easily.

1.2 Organization

The rest of the paper is organized as follows: some preliminaries are described in Section 2, and we will introduce our scheme in detail in Section 3. The analysis of our scheme is discussed in Section 4, including standards for evaluating and testing, experimental design, and simulation results. Finally, we give a brief conclusion of prior art in Section 5.

2 Preliminaries

2.1 OpenStack block storage scheduling

OpenStack is a very popular open-source cloud platform and has attracted lots of attention from industry and academia. Nowadays, over 200 companies join in developing the open-source community of the OpenStack. With the development of the OpenStack platform, it has provided plenty of different services. In this paper, we focus on Cinder which is used for block storage. It provides an infrastructure service for managing block storage devices (e.g., volume). Besides, it also provides an API, which can be used for requesting and customizing these storage resources for end clients. Various back-end storage providers connect to Cinder and provide virtual storage.

Three components are involved in Cinder: cinder-api, cinder-volume, and cinder-scheduler. The purpose of these components is to manage volumes and snapshots. Among these three components, the major critical component is cinder-scheduler, which includes series of scheduling strategies. When Cinder receives a volume request, such as creating a volume, according to the state of each storage node and the parameter of the request, Cinder-scheduler selects the most suitable storage node to response this request. Filter scheduling is always used to manage the storage, the main process two parts as shown in Fig. 1.

- *Filtering.* The filtering scheduler maintains a list of storage hosts. After receiving the volume request, a suitable host should be selected to provide enough available storage space for the request.
- *Weight sorting.* After the filtering step, the selected host can satisfy the volume request. Therefore, the

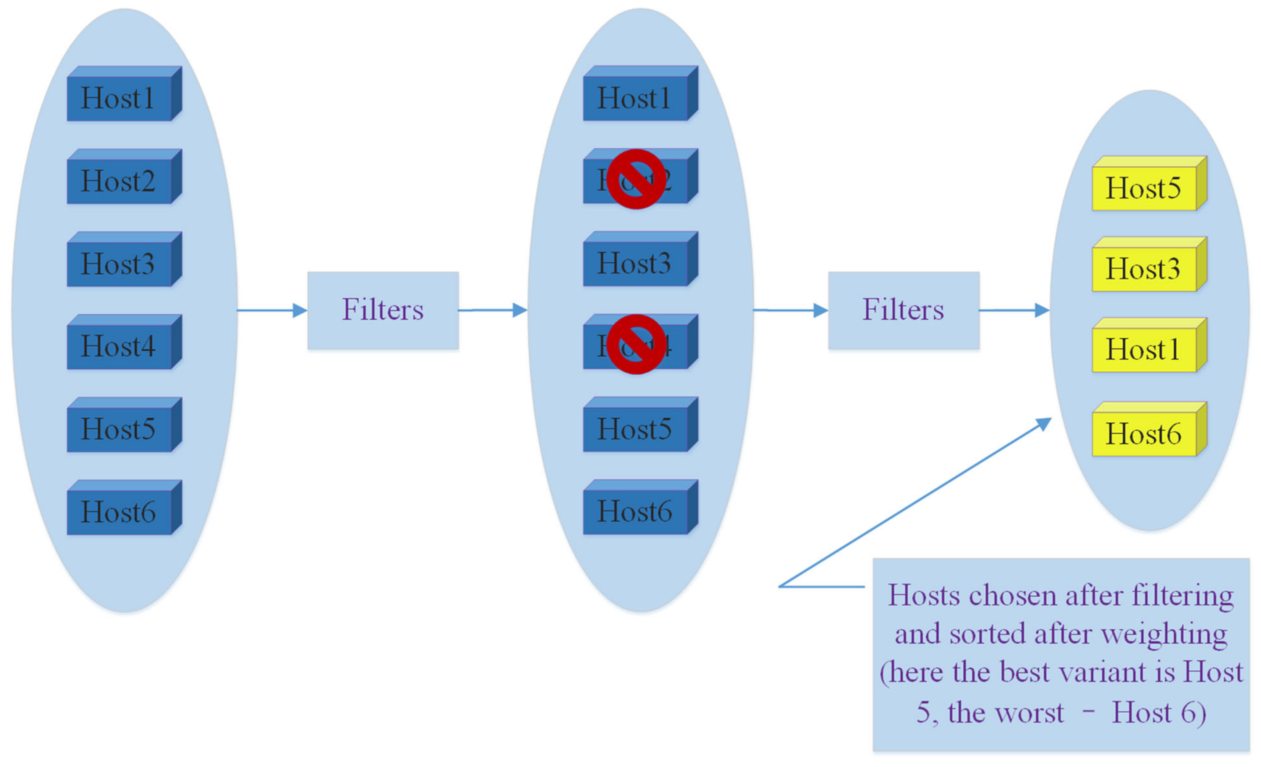
Host1
Host2
Host3
Host4
Host5
Host6
Filters
Host1
Host2
Host3
Host4
Host5
Host6
Filters
Host5
Host3
Host1
Host6
Hosts chosen after filtering and sorted after weighting (here the best variant is Host 5, the worst – Host 6)

Fig. 1 The work process of filter scheduling

scheduler calculates the corresponding weight firstly, then compares and sorts the weight on the basis of the parameter of volume request and the state of storage host.

In the released version OpenStack, a series of scheduling strategies have been applied in it. The default weight strategies are as follows: (1) the available space strategy, the essence of it is to select the host with maximum available space as the best storage node; (2) the allocated space strategy, its essence is to select the host with minimum allocated space as the best storage node; and (3) the random matching strategy, which means to be matched at random from the list of candidate hosts.

Based on the aforementioned process, the scheduler process largely depends on the weight sorting. After a new volume is created and allocated to a storage node, all operations of the volume can affect the storage node and the state of the whole cloud storage system. Thus, the decision made by the scheduler affects the performance of the whole cloud storage system significantly. However, not all of the system architects and the cloud computing providers can realize the importance of proper resource scheduling algorithm in the cloud storage system. Most of them focus on the computing resource scheduling optimization, such as the virtual machine scheduling algorithm. This problem is more obvious in OpenStack.

To the best of our knowledge, although there are about 30 kinds of scheduling strategies used to the Nova computing project, only 6 kinds of strategies are applied to the Cinder project. Therefore, there are still many defects in the existing Cinder scheduling algorithm. One of the main defects is that the I/O throughput has not been paid attention in most strategies. This defect could cause a poor cloud storage service performance; to be specific, it cannot predict allocated volumes and whether its additional attributes meet end-to-end SLAs. Therefore, we add I/O throughput to schedule strategies in our algorithm, and then, we improve the management performance of I/O throughput.

3 SLA-aware scheduling algorithm

We aim to reach two goals in our work: the first one is to make the cloud storage system implement the I/O throughput management, and the other one is to use the improved scheduling to maximize I/O throughput SLA utilization of each storage node. In terms of cloud storage services, I/O throughput SLA refers to the I/O throughput of user volume, which is greater than or equal to a specified IOPS at least 99.9% time(Input/output operands per second). The core mechanism of our algorithm is that the IOPS parameter should be considered when the scheduler decides to place a new volume. The default scheduling algorithm continuously monitors the states of the hosts, including the states of available space and allocated space. The improved scheduling algorithm also monitors the states of available IOPS and allocated IOPS.

In the modified OpenStack, additional attributes should be added into the Cinder scheduling algorithm, such as the host IOPS, the host available IOPS, the volume IOPS, and the volume available IOPS. The volume available IOPS means that a new volume can obtain IOPS as being assigned to a storage node. By adding the IOPS attribute of I/O, the volume allocation algorithm is no longer confined to storage space and also depends on the volume and the volume IOPS, where the volume has been allocated to the back-end storage for services, and the volume IOPS will be allocated to the back-end storage with no service. Therefore, the improved scheduling algorithm can gain the capability of I/O performance management, so that cloud providers can provide a better I/O throughput SLA for users. In the step of weight sorting, we bind the IOPS attribute and investigate the performance of Cinder filter scheduler, as described below.

- Available space filtering: selecting a suitable storage node which can provide enough available space for a volume request.
- I/O throughput weight sorting: according to the qualified host list of available space, selecting a suitable available IOPS to satisfy the volume request. Besides, the selected storage node with maximum space utilization is the best node.

3.1 Available space filtering

Based on the existing space filtering algorithm, we select the suitable storage node from the host list, and to guarantee the SLA, the weight sorting algorithm is applied to the host list after being filtered. We give an example of the storage node filtering in Fig. 2, which contains three storage nodes:

- Host 1. The host 1 with available space of 600 GB, available I/O throughput of 600 IOPS.
- Host 2. The host 2 with available space of 800 GB, available I/O throughput of 400 IOPS
- Host 3. The host 3 with available space of 500 GB, available I/O throughput of 550 IOPS.

Then when a volume request requires to create a 300-GB volume with 500 IOPS, the list still includes the hosts after being filtered since the hosts in the experiments 1, 2, and 3 all have enough available space. If the list is empty, it means that a SLA violation occurs, and then, the cloud storage providers should add new storage devices or storage nodes.

3.2 I/O throughput weight sorting

The I/O throughput weight sorting algorithm employs the default filtering algorithm(available space scheduling

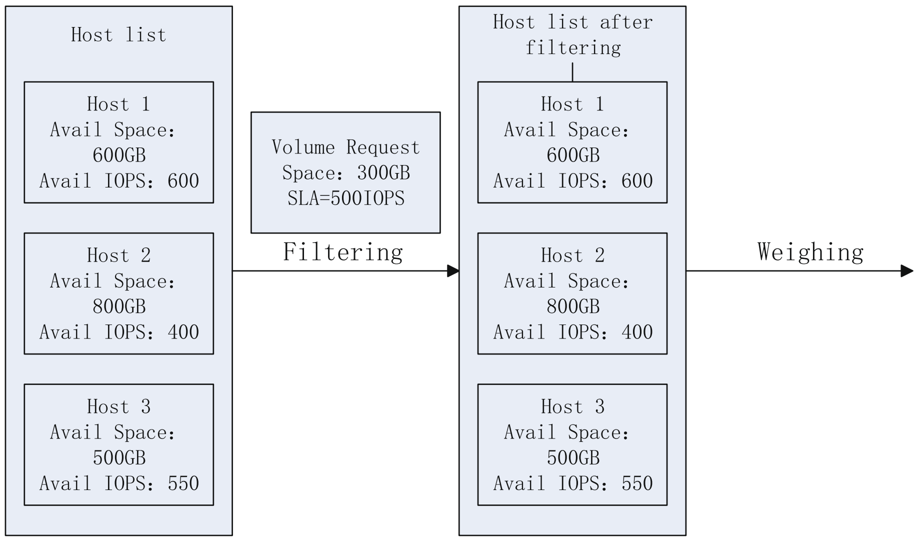

Fig. 2 Available space filtering

algorithm). Besides, the I/O throughput attribute is considered in the step of weight sorting. Figure 3 shows the process of the weight sorting in our algorithm. We assume that the examples in Figs. 2 and 3 are the same because the weight coding algorithm is improved on the basis of the classic $0-1$ Knapsack Problem. The weight sorting algorithm designed in our scheme as below:

$$max\ \frac{S_v \cdot V_k}{S_k - A_{sk}} \cdot 100, \forall k \in K$$
$$V_k \le T_k - A_{tk}, k \in K, V_k \in \{0,1\}$$

where the K represents a collection of storage nodes, and V is a new volume request, namely, $V_k \in \{0,1\}$ means that if the newly created volume is allocated to the storage node K, $V_k = 1$ or else $V_k = 0$. S represents the space size, S_k is the total size of the storage node K, and S_v is the storage space size of volume request. T represents the IOPS throughput, T_k is the total IOPS throughput of the storage node K, and T_v is the IOPS throughput of volume request. A_{sk} is the allocated space size of the storage node K, and A_{tk} is the allocated IOPS throughput of the storage node K.

According to the above weight sorting algorithm, the storage nodes 1# and 3# conform to the two constrained conditions. The host 3# is the best node, followed by host 1#, because we need to select the host with maximum space utilization. That can not only make the available

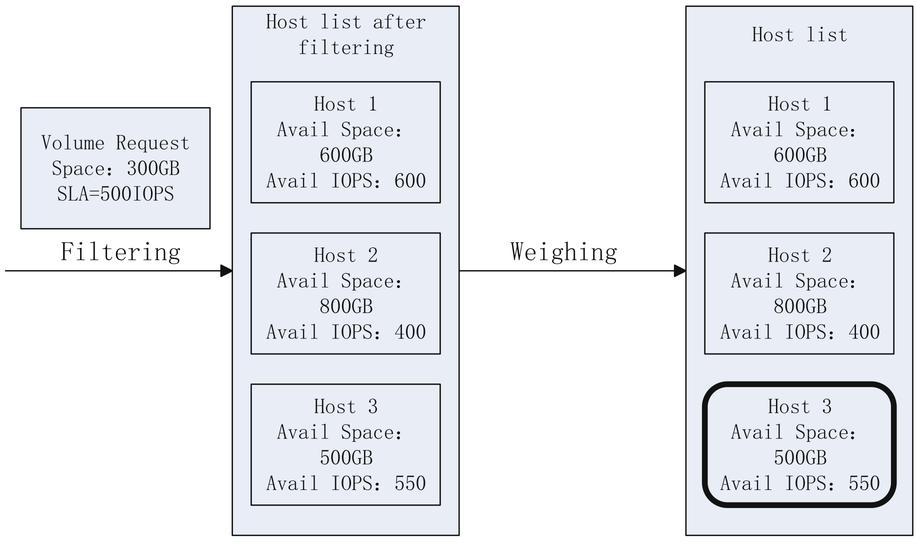

Fig. 3 I/O throughput weight sorting

IOPS of volume satisfies SLA requirements, but also make the space utilization of storage node reach its maximum value. After the available space is filtered, if the host list is not empty, and the above constrained conditions are unsatisfied, then the volume request waits until there are enough resources for its use. And it also means that a SLA violation occurs. In this case, it needs cloud storage providers to add new storage devices or storage hosts.

4 Performance evaluation

This part is mainly to evaluate the simulation results of the cinder scheduling algorithm in OpenStack platform. Firstly, we deploy a cinder storage system. Because of the complexity of the cloud storage system, it is rather difficult to repeat experiments on our cinder storage system. As a result, we implement our repeated experiments in a simulation environment. Specifically, we use JAVA to develop a simulator, which is based on the filtering algorithm of cinder and weighting algorithm model in OpenStack platform. Then, we repeat the experiments on it.

We mainly implement two test evaluations. In the first part, the mainly evaluation results which are of the combination between filtering algorithm and weighting algorithm are shown in Table 1. In the second part, the main evaluation results of the combination between filtering algorithm and weighting algorithm are shown in Table 2, where the largest available IOPS sorting algorithm is a kind of SLA awareness scheduling algorithm in [33].

4.1 Standards for evaluating and testing

In order to compare different weight algorithms and their performances effectively, we use SLA violation rate as the test and evaluation standards. The host list is empty after the strategies are filtered, or when the available IOPS of volume is less than the requested. At this moment, the SLA violation occurs. The SLA violation is defined as dividing the total of SLA violations in the experimental sampling by the total of experimental sampling [33].

4.2 Experimental design

To deploy solutions, we select the virtual Cinder block storage [38], which is recommended by Rackspace. There are 8 storage nodes in the system, each of which contains a CPU core, 8 GB memory, and a RAID disk array. The disk array consists of 12 hard disks (600 GB each 1500 rpm).

Table 1 Combination of filtering algorithm and weighting algorithm

Algorithm no.	Filtering algorithm	Weight sequencing
1	Available capacity	Available capacity
2	Available capacity	Random selection
3	Available capacity	I/O throughput
4	Available capacity	Allocated capacity

Table 2 Combination of filtering algorithm and weighting algorithm

Algorithm no.	Filtering algorithm	Weight sequencing
5	Available capacity	Available capacity
6	Available capacity	Max available IOPS
7	Available capacity	I/O Throughput

Every single disk can reach 175 IPOS when the file block size is 8 kB with 70% reading and 30% writing. According to the calculation formula of RAID IOPS, we can conclude that the maximum IOPS of each storage node can be up to 1948. Our work is mainly to deal with 9000 volume requests on the basis. Among them, each volume request includes arrival time, deadline, volume size, and IOPS parameters set by SLA. To perform the experimental operation more simply, we select Poisson distribution to state the two incidents of arrival and deadline. The arrival time and the deadline of volume requests should conform to the Poisson distributions with $\lambda = 20\ min$ and $\lambda = 600\ min$ respectively. The volume size is selected randomly from 100, 500, to 1000 GB. To improve the veracity, we repeat each experiment more than 30 trials.

4.3 Simulation results

We set the specified value of SLA to 450 IOPS and operate the strategies from 1# to 4# at a SLA violation rate for more than 30 times, and the results are shown in Fig. 4. It is obvious that the random selection sorting performs worst, the available space sorting and the allocated space sorting perform better, and I/O throughput behaves best. Specifically, the SLA violation rates of the available space sorting and the allocated space sorting both are close to 24%, and the random selection sorting is about 45%. However, the SLA violation rate of the presented I/O throughput sorting is about 2%. It means that about 98% of volume requests can use IOPS, which is greater than or equal to 450 IOPS set by SLA.

Figure 5 shows different violation rates of weight algorithm under the same conditions with IOPS from 200 to 600. When the IOPS is between 200 and 500, the SLA violation rate of the I/O throughput sorting is much better than those of other weight sorting strategies. When the number of the IOPS is between 500 and 600, although its violation rate increases, something has to do with the realization process. To satisfy the volume request of IOPS, the volume request will wait for all the time until there is appropriate host providing enough available IOPS. The SLA violation rate will increase if no new storage nodes added. Therefore, in weight sorting strategies used, the I/O weight sorting algorithm can provide better management performance.

To load volume allocated requests with more storage nodes, we can change the simulation hardware by increas-

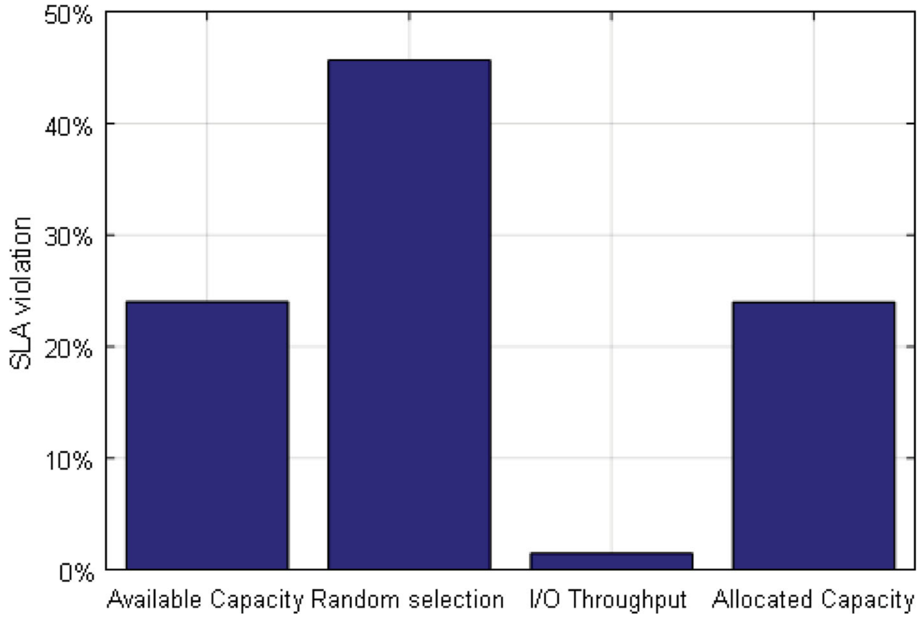

Fig. 4 Different violation rates in available space filtering strategies

ing the storage nodes. We set the SLA volume IOPS as 450 and test the results of different counts of nodes on the SLA violation rate. As shown in Fig. 6, when the SLA violation rate is about 2%, the I/O throughput weight sorting algorithm's storage nodes is 8; however, the Cinder available and allocated space scheduling algorithm's storage nodes reach to 20. Besides, when the random selection sorting scheduling algorithm's storage nodes reach to 24, the SLA violation rate only reaches 3.5%. In other words, our algorithm needs less storage nodes under the same I/O performance management.

We combine the algorithms of 5# − 7#, and the number of average of used hosts shown in Fig. 7. From the Fig. 7, we can know that the maximum number of hosts is 8, and the number of used hosts is increasing with the volume requests. Besides, when the number of request

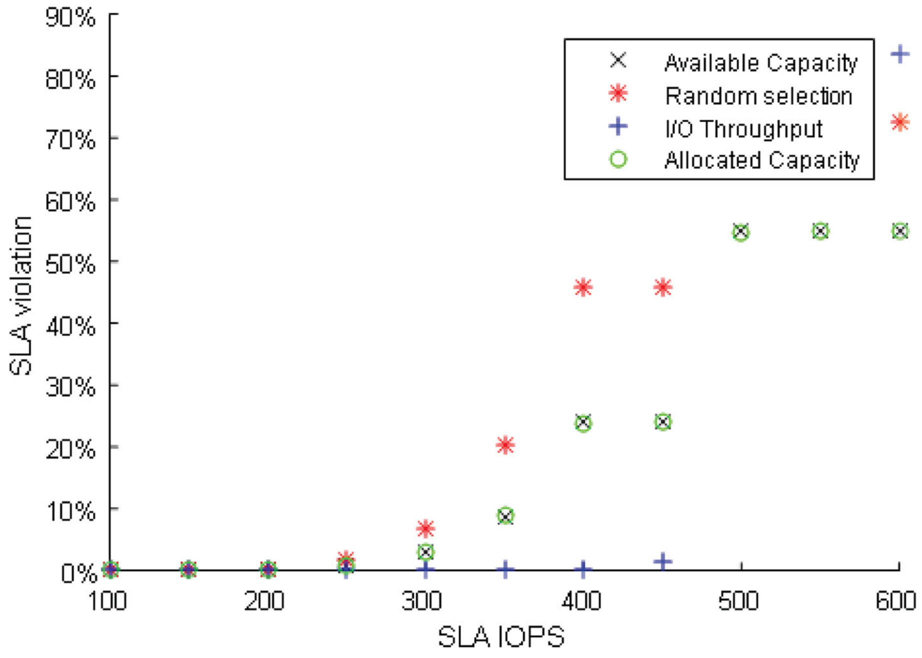

Fig. 5 Different SLA violation rates of different weight strategies with different values

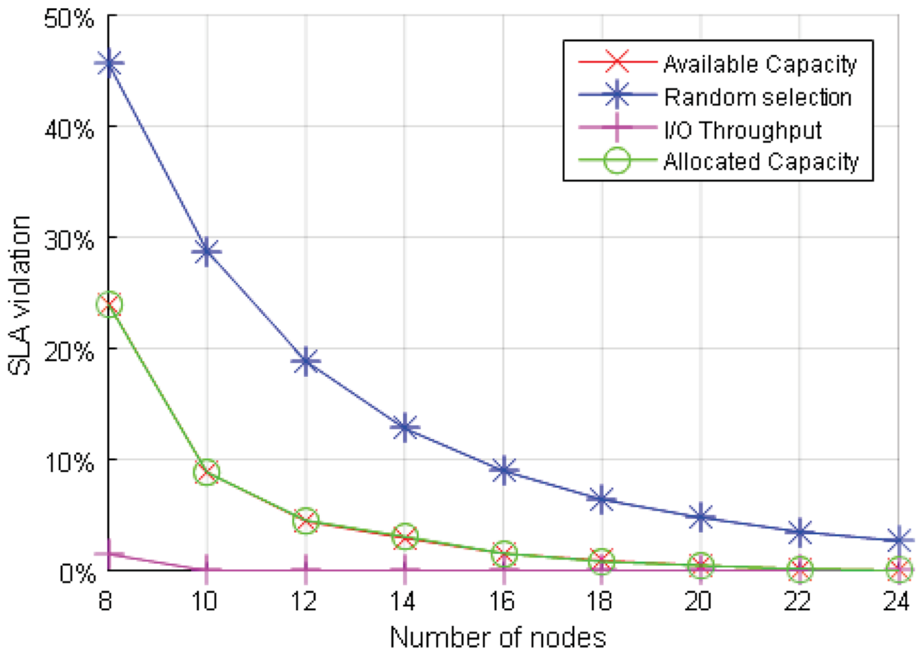

Fig. 6 Violation rates of different weight sorting strategies with different counts of nodes

reaches 5, the number of hosts used in our algorithm is less than that in the maximum available IOPS sorting algorithm and the default maximum space available sorting algorithms which are raised in [33]. Furthermore, when the number of volume requests comes to 30, 8 hosts used in our presented algorithm, which is same with the other two algorithms. That is, the number of volume requests have gradually reached the maximum which the physical host can handle. Through the contrast experiment, we can get that our proposed algorithm is better than the default scheduling algorithm in OpenStack and the algorithm presented in [33] in the using hosts.

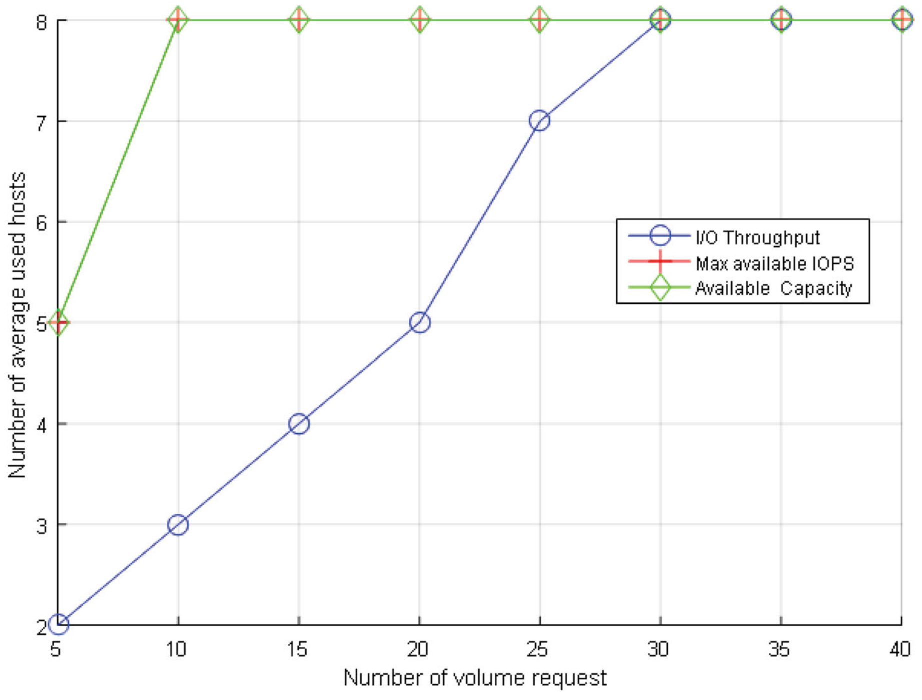

Fig. 7 Number of average used hosts of different weight sorting algorithm with different counts of nodes

5 Conclusion

In this paper, we investigate the Cinder weight algorithm in OpenStack, which is a SLA-aware scheduling algorithm based on the OpenStack Cinder scheduling module. The implementations show that the combination of the weight algorithm with the default space filter used in OpenStack, as presented in this paper, can effectively decrease the SLA violation rate. That means, under the condition of the same violation rate, less nodes are used in our presented algorithm. The expenses of cloud storage providers can also be reduced. However, a drawback of our weight algorithm is that it cannot adjust the number of cloud host node dynamically. Therefore, in the future work, we plan to design a dynamically scalable weight algorithm, which dynamically adjusts the number of cloud storage nodes according to demands.

Abbreviations

AOSC: automatic optimization scheme; API: Application Programming Interface; A_{sk}: the allocated space size of the storage node K; A_{tk}: the allocated IOPS throughput of the storage node K; BSDS: block software defined storage; HPC-aware: High Performance Computing-aware ; IOPS: Input/Output Operations Per Second; K: a collection of storage nodes; MGBA: Minimised Geometric Buchberger Algorithm; MVBFD: Modified Vector Best Fit Decreasing; QoE: Quality of Experience; QoS: Quality of Service; RAID: Redundant Arrays of Independent Drives; RTBM: remaining time based maximal; S: the space size; SDS: software defined storage ; S_k: the total size of the storage node K; SLA-aware: service level agreement-aware; S_v: the storage space size of volume request; T: the IOPS throughput; T_k: the total IOPS throughput of the storage node K; T_v: the IOPS throughput of volume request; V: a new volume request; V_k: means that if the newly created volume is allocated to the storage node K, $V_k = 1$ or else $V_k = 0$

Authors' contributions

YW, XT, and FZ contributed to the conception and algorithm design of the study. YW, XT, and FZ contributed to the design of experiment scheme. YW, XT, and AM contributed to the analysis of experimental data and approved the final manuscript. All authors read and approved the final manuscript.

Funding

This work was supported by the National Natural Science Foundation of China (No. 61363006) and the Open Projects of State Key Laboratory of Integrated Service Networks (ISN) of Xidian University (No.ISN19-13) and the National Natural Science Foundation of Guangxi (No. 2016GXNSFAA380098) and the Science and Technology Program of Guangxi (No. AB17195045).

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, Guilin, China. [2]State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China. [3]Guangxi Cooperative Innovation Center of cloud computing and Big Data, Guilin University of Electronic Technology, Guilin, China. [4]Department of Computer Science Georgia State University, Atlanta, USA.

Received: 27 August 2018 Accepted: 15 November 2019
Published online: 03 January 2020

References

1. C. Yang, X. Chen, Y. Xiang, Blockchain-based publicly verifiable data deletion scheme for cloud storage. J. Netw. Comput. Appl. **103**, 185–193 (2018)
2. X. Chen, J. Li, J. Weng, J. Ma, W. Lou, Verifiable computation over large database with incremental updates. IEEE Trans. Comput. **65**(10), 3184–3195 (2016)
3. X. Chen, J. Li, J. Ma, Q. Tang, W. Lou, New algorithms for secure outsourcing of modular exponentiations. IEEE Trans. Parallel Distrib. Syst. **25**(9), 2386–2396 (2014)
4. C. Yang, J. Ye, Secure and efficient fine-grained data access control scheme in cloud computing1. J. High Speed Netw. **21**(4), 259–271 (2015)
5. X. Chen, J. Li, X. Huang, J. Ma, W. Lou, New publicly verifiable databases with efficient updates. IEEE Trans. Dependable Secure Comput. **12**(5), 546–556 (2015)
6. K. H. Prasad, T. A. Faruquie, L. V. Subramaniam, M. Mohania, G. Venkatachaliah, in *2010 IEEE International Conference on Services Computing (SCC)*. Resource allocation and sla determination for large data processing services over cloud, (Miami, 2010), pp. 522–529
7. D. Breitgand, A. Epstein, in *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2011*. Sla-aware placement of multi-virtual machine elastic services in compute clouds (IEEE, Dublin, 2011), pp. 161–168
8. X. Qiu, M. Hedwig, D. Neumann, in *2012 workshop on e-life: web-enabled convergence of commerce, work, and social life, 2011*. Sla based dynamic provisioning of cloud resource in oltp systems (Springer, Berlin, 2011), pp. 302–310
9. Q. Huang, K. Shuang, P. Xu, J. Li, X. Liu, S. Su, Prediction-based dynamic resource scheduling for virtualized cloud systems. J. Netw. **9**(2), 375–383 (2014)
10. S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, R. Buyya, Sla-based virtual machine management for heterogeneous workloads in a cloud datacenter. J. Netw. Comput. Appl. **45**, 108–120 (2014)
11. C. C. Li, K. Wang, in *2014 International Conference on Information Networking (ICOIN), 2014*. An sla-aware load balancing scheme for cloud datacenters (IEEE, Phuket, 2014), pp. 58–63
12. S. Singh, I. Chana, Qrsf: Qos-aware resource scheduling framework in cloud computing. J. Supercomput. **71**(1), 241–292 (2015)
13. Z. Wang, X. Su, Dynamically hierarchical resource-allocation algorithm in cloud computing environment. 71. **7**, 2748–2766 (2015)
14. H. Wada, P. Champrasert, J. Suzuki, K. Oba, in *IEEE Congress on Services-Part I, 2008*. Multiobjective optimization of sla-aware service composition (IEEE, Honolulu, 2008), pp. 368–375
15. S. Chaisiri, B. S. Lee, D. Niyato, in *IEEE Asia-Paciffic Services Computing Conference (APSCC), 2009*. Optimal virtual machine placement across multiple cloud providers (IEEE, Singapore, 2009), pp. 103–110
16. J. Wang, P. Varman, C. Xie, in *2010 International Conference on High Performance Computing (HiPC), 2010*. Avoiding performance ?uctuation in cloud storage (IEEE, Dona Paula, 2010), pp. 1–9
17. H. Goudarzi, M. Pedram, in *2011 IEEE International Conference on Cloud Computing (CLOUD), 2011*. Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems (IEEE, Washington, DC, 2011), pp. 324–331
18. C. C. Lin, P. Liu, J. J. Wu, in *2011 IEEE International Conference on Cloud Computing (CLOUD), 2011*. Energy-aware virtual machine dynamic provision and scheduling for cloud computing (IEEE, Washington, DC, 2011), pp. 736–737
19. A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for e?cient management of data centers for cloud computing. Futur. Gener. Comput. Syst. **28**(5), 755–768 (2012)
20. A. Gupta, L. V. Kale, D. Milojicic, P. Faraboschi, S. M. Balle, in *2013 IEEE International Conference on Cloud Engineering (IC2E), 2013*. Hpc-awarevm placement in infrastructure clouds (IEEE, Redwood City, 2013), pp. 11–20
21. S. Singh, I. Chana, Qos-aware autonomic resource management in cloud computing: a systematic review. ACM Comput. Surv. (CSUR). **48**(3), 42 (2016)
22. K. Xiong, X. Chen, in *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2015*. Ensuring cloud service guarantees via service level agreement (sla)-based resource allocation (IEEE, Columbus, 2015), pp. 35–41
23. X. Zheng, Z. Cai, H. Gao, A Study on Application-Aware Scheduling in Wireless Networks. IEEE Trans. Mob. Comput. **16**(7), 1787–1801 (2017)
24. P. Leitner, W. Hummer, B. Satzger, C. Inzinger, S. Dustdar, in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD), 2012*. Cost-efficient and application sla-aware client side request scheduling in an infrastructure as-a-service cloud (IEEE, Honolulu, 2012), pp. 213–220
25. S. Yoo, S. Kim, in *The international multiconference of engineers and computer scientists, 2013*. Sla-aware adaptive provisioning method for

hybrid work load application on cloud computing platform (IMECS, Hong Kong, 2013), p. 1
26. Z. Xiao, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1107–1117 (2013)
27. M. Alrokayan, A. V. Dastjerdi, R. Buyya, in *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2014*. Sla-aware provisioning and scheduling of cloud resources for big data analytics (IEEE, Bangalore, 2014), pp. 1–8
28. S. Ayesha, Sla-aware and cost-aware provisioning and scheduling of cloud resources across multiple data centres. Int. J. Adv. Res. Comput. Commun. Eng. **4**(5), 238–435 (2015)
29. Z. Jianrong, L. Jing, Z. Yi, Utility-based virtual cloud resource allocation model and algorithm in cloud computing. Int. J. Grid Distrib. Comput. **8**(2), 177–190 (2015)
30. X. Cai, F. Li, P. Li, L. Ju, Z. Jia, Sla-aware energy-e?cient scheduling scheme for hadoop yarn. J. Supercomput. **73**(8), 3526–3546 (2017)
31. Q. Li, Y. Guo, in *2010 12th International Symposium on Symbolic and Numeric Algorithms for Scienti?c Computing (SYNASC), 2010*. Optimization of resource scheduling in cloud computing (IEEE, Timisoara, 2010), pp. 315–320
32. Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, L. Liu, Service level agree ment based energy-efficient resource management in cloud data centers. Comput. Electr. Eng. **40**(5), 1621–1633 (2014)
33. Z. Yao, I. Papapanagiotou, R. D. Callaway, in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), 2014*. Sla-aware resource scheduling for cloud storage, (2014), pp. 14–19
34. Z. Yao, I. Papapanagiotou, R. D. Callaway, in *2015 IEEE International Conference on Communications (ICC), 2015*. Multi-dimensional scheduling in cloud storage systems (IEEE, London, 2015), pp. 395–400
35. Z. Yao, I. Papapanagiotou, in *2017 IEEE International Conference on Communications (ICC), 2017*. A trace-driven evaluation of cloud computing schedulers for iaas (IEEE, Paris, 2017), pp. 1–6
36. B. Ravandi, I. Papapanagiotou, B. Yang, in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD), 2016*. A black-box self-learning scheduler for cloud block storage systems (IEEE, San Francisco, 2016), pp. 820–825
37. B. Ravandi, I. Papapanagiotou, in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD), 2017*. A self-learning scheduling in cloud software defined block storage (IEEE, Honolulu, 2017), pp. 415–422
38. Laying cinder block (volumes) in openstack, part 2. Solutions design. http://www.rackspace.com/. Accessed 3 Jan 2018

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.