EVALUATION OF DATA OPTIMAL STORAGE SYSTEM IN HYBRID CLOUD

Miho Imazaki[1,3], Satoshi Kaneko[1], Norihisa Komoda[2] and Takenao Ohkawa[3]
[1]*Research & Development Group, Hitachi, Ltd., Yokohama, Japan*
[2]*Corporate Adviser, Code Solution K.K,.Osaka, Japan*
[3]*Graduate School of System Informatics, Kobe University, Kobe, Japan*

ABSTRACT

A hybrid cloud is a mix of compute systems, storage systems and services consisting of on-premises systems, private clouds and public clouds for satisfying requirements such as security and performance. In the case of hybrid cloud configuration where data are in on-premises storage systems and compute systems is in public clouds, data should be transferred to the public cloud when a read/write command is issued by compute systems. Thus, long data transfer time is a problem since the on-premises systems and public clouds are connected by the internet. One possible solution is to create an additional cache area inside the compute system; however, it is not clear how effective it is. In this paper, we verify the effect of the data cache by creating the hybrid cloud environment with/without the cache area and measuring input/output latency. We created the measurement environment in the hybrid cloud with dm-cache technology and set up the assumed workload of TPC Benchmark^{TM} E with the write-through mode. The average latency with the cache area reduces by up to 79% compared to without the cache area, and we confirm the effect of installing cache areas in this environment.

KEYWORDS

Hybrid Cloud, Storage System, Public Cloud, Data Cache

1. INTRODUCTION

The demand for IT infrastructure flexibility increases the number of companies using the public cloud which is defended as computing services. The public cloud infrastructures are managed by a third-party provider and shared with multiple users over the internet. However, some users require high performance and have sensitive data whose they want to keep in their own data centers to manage where are stored. This environment type is called a on-premises system. And there is another case that some users need a dedicated system environment for a higher level of security with flexible operations like a public cloud's ones. For these users, a private cloud is offered, and it is built in on-premises systems or public clouds by users, therefore users should be managed private cloud by themselves. As a mentioned above, each system has its own characteristics, and one of the popular IT environments is using a mix of on-premises system and public clouds, which is called a hybrid cloud (Weinman, 2016) (Shibambu, 2022). A development and test (dev/test) are one of popular use cases for the hybrid cloud. This is because users can keep the data in the production environment of secure and high-performance on-premises systems while create and delete the dev/test environment easily with public cloud services. For the demand of using modern applications/computing services with the public cloud in the dev/test however the transferring only the minimum required data to the public cloud securely, one of hybrid cloud architectures is that a storage system volume in the physical data center's on-premises system is attached to a virtual machine (VM) created in the public cloud.

In this dev/test environment, the data in the physical data center should be transferred to the public cloud as needed. The physical data center is connected to the public cloud through the internet. Public cloud vendors offer high-speed internet services; however, this has limitations in reducing latency compared to internal networks. Thus, the problem in this environment is that it takes too long time to complete the data transfer before starting the dev/test.

To solve this problem, reducing the amount of data transfer and appropriate data placement in storage systems are important. There are some methods to do this, a data cache and a data tiering (Arteaga et al, 2016)

(Zhang et al, 2010). The data cache method is that creating a data cache area in the public cloud and storing accessed data in it. And the data tiering method is that setting up a device in the public cloud and storing the high frequency access data in it. However, it isn't clear how much each method can reduce the amount of data transfer. There are related studies: creating and measuring a cache area in a computing system on a public cloud (Park et al, 2020); creating cache servers among cloud block storage devices (Zhang et al, 2020). However, they assume only public cloud configurations and does not consider hybrid cloud configurations. Therefore, we choose the data cache method and measure and evaluate the improvement of data caching technology in the hybrid cloud.

The rest of this paper organized as follows. Section 2 describes the data cache method in a public cloud. We describe the measurement environment to show the effect of the data cache method in Section 3, and the measurement result in Section 4. Finally, Section 5 concludes the paper.

2. DATA CACHE METHOD IN PUBLIC CLOUD

We consider the dev/test environment that the data are in the physical data center and the VM as compute is in the public cloud. In this environment, there are 2 major issues: it takes a long time to transfer the data from the physical data center before starting a dev/test; the cost of public cloud storage system is too expensive to store the all dev/test data. Therefore, the key point to solve them is reducing the amount of data transfer between the on-premises system and the public cloud. To reduce it, the dev/test application in the public cloud directly need to access to the dev/test data in the on-premises system. However, this causes high access latency from the dev/test application to the dev/test data.

One of the solutions to this is that the part of dev/test data, which are frequently accessed from the dev/test application, are stored in the VM's data cache area. The compute system where the dev/test application runs already has a virtual dynamic rando-access memory (DRAM) as a data cache area. However, the size of DRAM in a public cloud is not large enough and its cost is relatively high if you want to read/write large amount of data.

To solve this, you can use a virtual volatile storage volume in a public cloud as a data cache area. Figure 1 shows the overview of this solution. In this case, the dev/test data are not necessary to copy to public cloud before setting up a dev/test environment. When a dev/test application starts to run on the VM, it accesses to dev/test data in the on-premises, and they are stored in the cache area on VM. After a certain period, only dev/test data with high access frequency are stored in the cache area, and dev/test data transferring between the on-premises and the public cloud hardly occurs if the cache area size is appropriate.

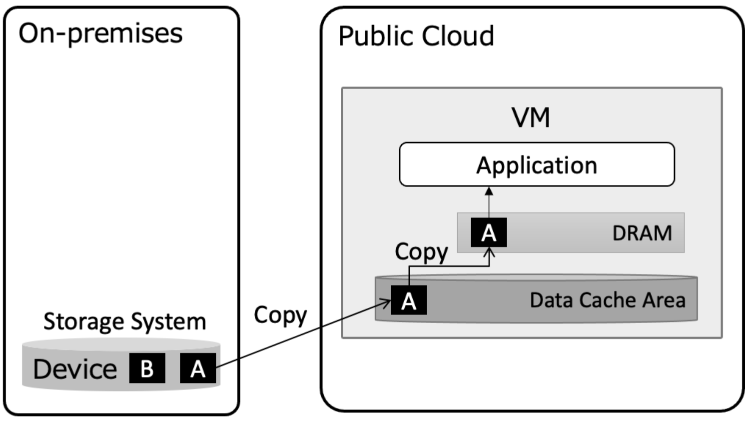

Figure 1. Caching data in a public cloud

We describe read and write processes in this configuration. A data cache program checks if there is the target data in the VM's DRAM when the application issues a read data request. If the answer is no, the program checks if the data is stored in the data cache area. If the data are not stored in it, the program commands the data copy from the on-premises storage system to the data cache area in the VM, then copies it to the DRAM. And the application reads the target data from the DRAM. After that, it can be read from the DRAM or the data cache area through the public cloud internal network when the same data are read from the application, resulting in a shorter response to the application. While about the write process, there are 2 modes: a write-through mode and a write-back mode (Jouppi, 1993). The write-though mode is that the target data are written into the DRAM first then written into both the data cache area and the on-premises storage system at the same time. While the write-back mode is that the target data are written into the DRAM first then write into the data cache area. And the data are written to the on-premises storage system asynchronously with the process described above. In this mode, the data cache program can send a write completion message to the application immediately after writing to the data cache area. Thus, the write process response is shortened. However, there is a possibility to lose track of how much data were written if an error occurs in the VM before the data is transferred to the on-premises storage system. While in the write-through mode, the possibility of data loss is lower than in the write-back mode since the application receives the write completion message is complete. However, the write process response in the write-through mode is longer than in the write-back mode.

There is the case of no free space in the data cache area due to the read/write process, since the capacity of the data cache area is limited. In this case, if data have not yet been written to the on-premises storage system, the data are transferred to it and then deleted from the data cache area. Alternatively, read data with low access frequency is deleted from the data cache area.

This data cache method is theoretically valid to reduce the data transfer between on-premises system and public clouds. However, it's not clear that how effective this method is.

3. MEASUREMENT OF DATA CACHE EFFECT

3.1 Evaluated Environment

To be clear the effectiveness of the data cache method, we consider the verification environment which is shown in Fig 2. We prepare Hitachi Virtual Storage Platform G370 as the on-premises storage system which is a block storage and stores the data in the data center, and the Amazon Web Service Elastic Computing (AWS EC2) with an instance store volume as the compute in the public cloud (Hitachi Vantara, 2018) (Amazon Web Service, 2022a). An instance store provides temporary block-level storage which is a volatile memory (Amazon Web Service, 2022a). The storage system and the compute instance relate to AWS Direct Connect 10 Gbps, which is the AWS high-speed private network service (Amazon Web Service, 2022b). The protocol is iSCSI. We use the instance store volume as the cache area and set a data cache function with a dm-cache (Thornber, 2015). The dm-cache is a component of the Linux kernel's device mapper and improves performance of a block device by dynamically migrating some of its data to a faster and smaller device. When the AWS EC2 needs to access to the data in the on-premises storage system, the data is copied to the instance store volume.

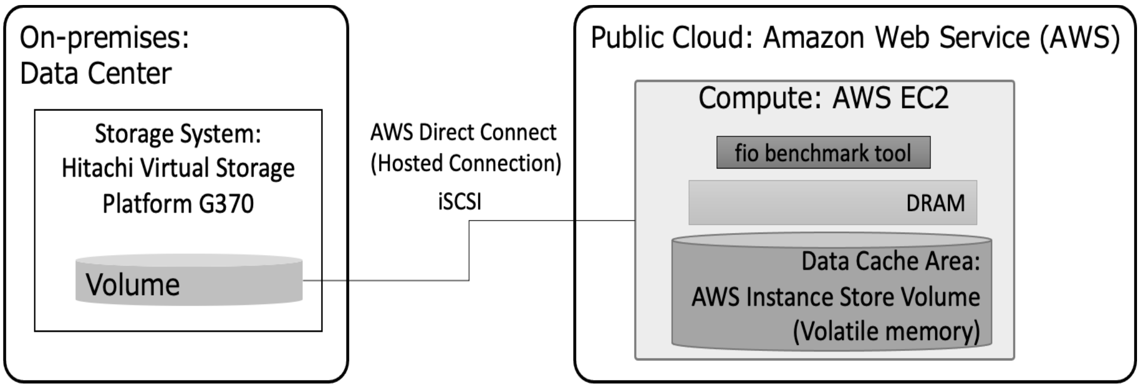

Figure 2. Caching data in public cloud

Figure 3 shows a detail of the cache architecture. The cache area is managed in units of chunks which are data aggregates. The cache pool is consisted with the cache area and the cache area meta data.

When the read command is committed from the AWS EC2 to the on-premises storage system, the dm-cache algorithm judges whether the target data are stored in the cache pool. In the case of the data are stored in it, the data are read from the cache pool through the DRAM and the AWS EC2 doesn't need to access to the on-premises storage system. Other than that, the data should be read from the on-premises storage system to the cache area with the chunk included with the target data and the only target data are read from the cache area through the DRAM.

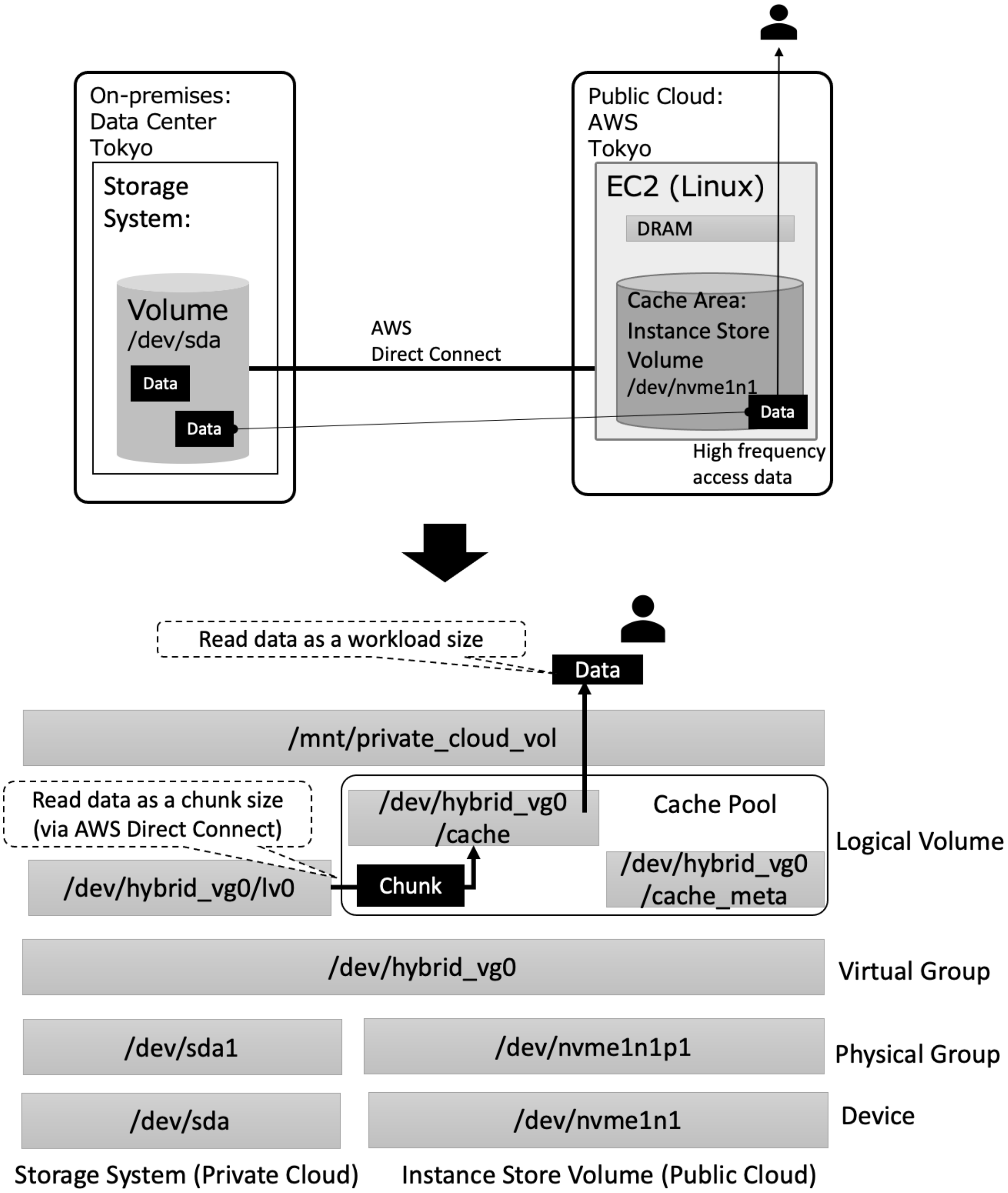

Figure 3. Cache area architecture

In this dm-cache environment, we set up the write-through mode which is that data is written to the on-premises storage system without the cache area when a write command is issued. The reason for choosing it is that the data with the write back mode may be broken when the system has an error before the data are transferred to the on-premises storage system.

3.2 Workload and Measurement Condition

The workload in this environment is based on TPC Benchmark^{TM} E (Transaction Processing Performance Council, 2015). TPC Benchmark^{TM} E is an on-line transaction processing benchmark. It models the activity of brokerage firm that must manage customer accounts, execute customer trade orders, and be responsible for the interactions of customers with financial markets. And The TPC Benchmark^{TM} E Input/Output (IO) request breakdown for data devices is clear in the related research (Chen, 2011). Therefore, in this study, we set up the issuing IO set as shown in Table 1 and this IO is issued to the device using fio benchmark which is the tool spawn a number of threads or processes doing a particular type of IO action as specified by a user (Axboe, 2022).

Table 1. I/O patterns

I/O pattern	Data size [kByte]	Event probability [%]
Random Read	8.0	90.0
Random Write	8.0	10.0

Table 2 shows other measuring conditions in this study. We select 3 types of cache chunk size to confirm the effect of different chunk sizes. The number of chunks decreases as the chunk size increases since the cache area size is fixed.

Table 2. Measurement

Item	Detail
I/O access range	5 GBytes
Measurement time	6 minutes
AWS EC2 type	r5d.xlarge (Instance Store: 150 GBytes)
Cache area size	150 GBytes
Cache chunk size	32 kBytes, 128 kBytes, 160 kBytes
Volume size in the on-premises storage system	170 GBytes

4. MEASUREMENT RESULTS

We measured IO latencies in this cache environment and compared the differences between the ones with/without cache area.

Figure 4 shows the result of single I/O average latency in the random read/write mix. The latency with the cache area and 160 kBytes chunk reduced by maximum 79% compared to without the cache area. Following this result, we confirmed that there is an effect of data caching in this workload. However, in the case with the cache area, a chunk read occurred from the storage system in the on-premises system for the first I/O. Therefore, the latency reduction effect with the cache area at the first time was low.

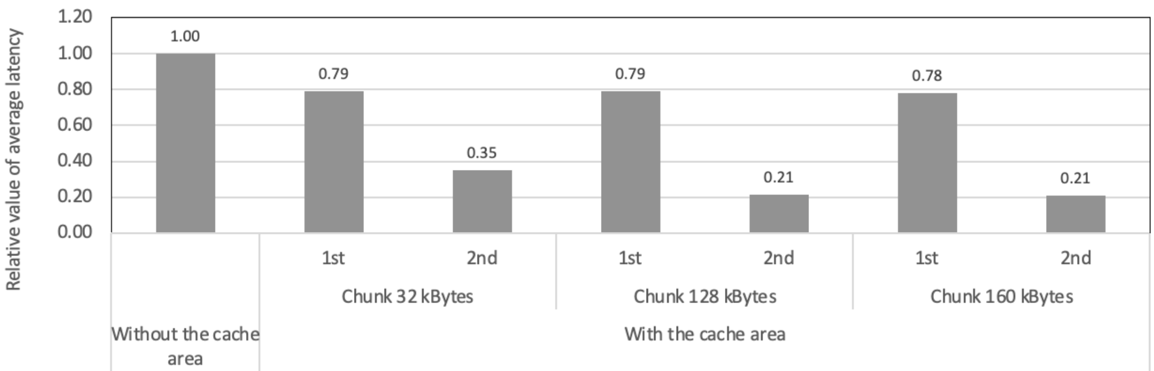

Figure 4. Single I/O latency: random read / write mix

We evaluated the latency reduction effect for each random read and random write in the measurement result Figure 4 shows. Figure 5 shows the result of single I/O average latency in the random read. In the case with the cache area, a chunk read always occurred in the first I/O access. Therefore, the maximum latency, especially for large chunk sizes, tended to be higher than without the cache area. We thought that the larger the chunk size is, the higher the maximum latency is. However, the first measurement showed that the 160 kBytes chunk had a lower maximum latency than the 128 kBytes chunk in this workload. We consider that this was because the data to be accessed was already stored in the cache area in more cases with the 160 kBytes chunk than with the 128 kBytes chunk due to the access locality of this workload.

We also thought that the second measurement would have less latency than the first one, since the most of chunk copies were already done. However, the second maximum latency was 17% higher than the first maximum latency in the case of 32 kBytes chunk. We believe that this is because of the access locality of this workload.

Even with the cache area, the average latency was reduced by more than 26% in the first measurement and by more than 79% in the second measurement compared to the case without the cache area. This means large latency didn't occur frequently. Therefore, the effect of using the cache area is confirmed.

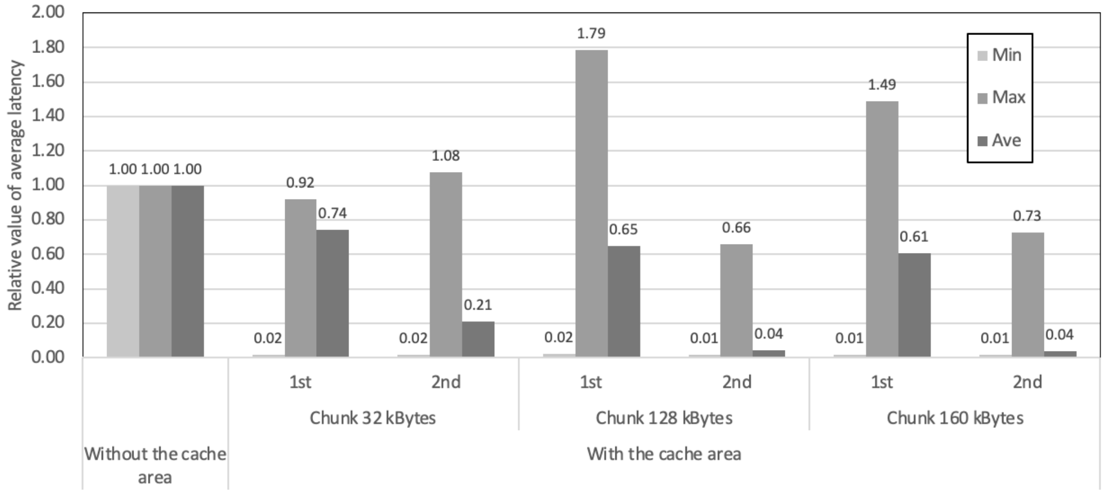

Figure 5. Single IO latency: random read

Figure 6 shows the result of single I/O average latency in the random write. We measured them with the write though mode. However, the larger the chunk size is, the higher the maximum latency is. We think this is because some write operations are disturbed by reading chunks in the case with the cache area. In the second measurement, the average latency was the same with and without the cache area since the number of reading chunks was reduced. Thus, we can confirm the effect of the cache area in this workload. However, there may be a risk that it leads to particularly poor write performance if the chunk size is not set up appropriately for a workload and operation time.

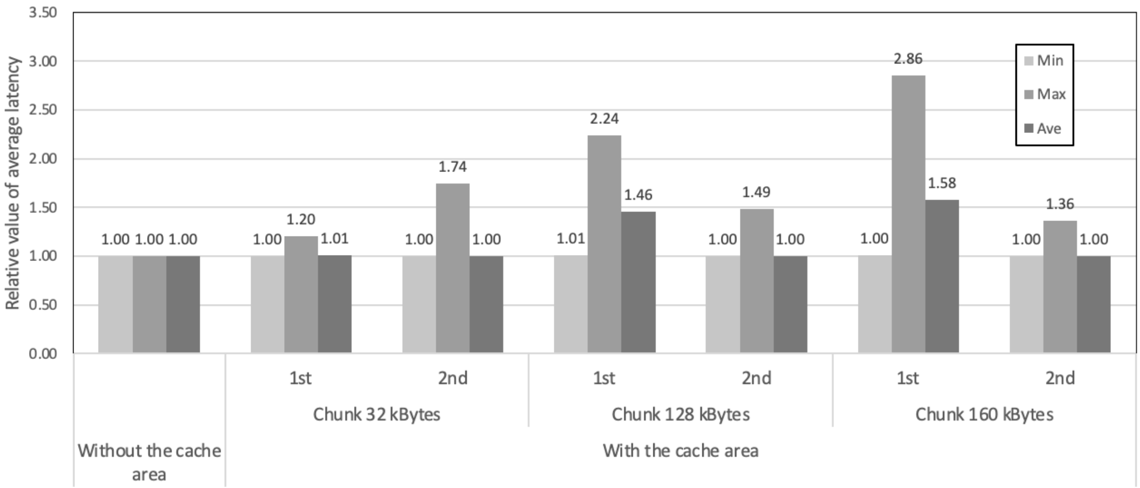

Figure 6. Single IO latency: random write

5. CONCLUSION

We verified the effect of data caching, which is one of the methods for reducing the amount of data transfer between the on-premises system and the public cloud, in the hybrid cloud environment where the compute system and the storage system were located separately. We created the measurement environment in the hybrid cloud with the dm-cache technology and set up the assumed workload of TPC Benchmark^{TM} E with the write-through mode. And it is confirmed that the average latency with the cache area reduced by maximum 79% compared to without the cache area. The latency increased when transferring chunks occurred, thus having the cache area increased the difference between maximum and minimum latencies. However, when comparing the average latencies with/without the cache area, it is considered that setting up the cache area is effective for the workload verified this evaluation. In addition, we found that setting up an appropriate chunk size is important especially in write-through mode to prevent write latency degradation, as data access characteristics change as workload change.

REFERENCES

Amazon Web Service, Inc., (2022a). *Amazon Elastic Compute Cloud User Guide for Linux Instance*, Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html (Accessed: 2 September 2022).

Amazon Web Service, Inc., (2022b). *AWS Direct Connect User Guide*, Available at: https://docs.aws.amazon.com/directconnect/latest/UserGuide/Welcome.html (Accessed: 2 September 2022).

Arteaga, D. et al, (2016). CloudCache: On-demand Flash Cache Management for Cloud Computing. *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST '16)*, Santa Clara, USA, pp. 355-369.

Axboe, J., (2022). fio-Flexible I/O tester rev.3.30, Available at: https://fio.readthedocs.io/en/latest/fio_doc.html (Accessed: 2 September 2022).

Chen, S., (2011). TPC-E vs. TPC-C: Characterizing the new TPC-E benchmark via an I/O comparison study. *ACM Sigmod Record*, 39(3), pp. 5-10.

Hitachi Vantara Corporation, (2018). *VSP G350 and G370 Hardware – Hitachi Vantara Knowledge*, Available at: https://knowledge.hitachivantara.com/Documents/Storage/VSP_G130_GF350_GF370_GF700_GF900/88-07-0x/About_Your_System/VSP_G350_G370_Hardware (Accessed: 2 September 2022).

Jouppi, N.P., (1993). Cache write policies and performance. *ACM SIGARCH Computer Architecture News*, 21(2), pp. 191-201.

Park, H. et al, (2020). More IOPS for Less Exploiting Burstable Storage in Public Clouds. *Proceedings of 12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*, online.

Shibambu, A., (2022). Migration of government records from on-premises to cloud computing storage in South Africa. *South African Journal of Libraries and Information Science*, 88(1), pp.1-11.

Thornber, J. et al, (2015). Linux kernel documentation: Documentation/device-mapper/cache.txt, Available at: https://www.kernel.org/doc/Documentation/device-mapper/cache.txt (Accessed: 2 September 2022)

Transaction Processing Performance Council, (2015). *TPC BENCHMARKTM E Standard Specification version 1.14.0*, Available at: https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-e_v1.14.0.pdf (Accessed: 2 September 2022).

Weinman, J., (2016). Hybrid cloud economics. *IEEE Cloud Computing, 3(1)*, pp. 18-22.

Zhang, G. et al, (2010). Adaptive Data Migration in Multi-tiered Storage Based Cloud Environment. *Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing*, Miami, USA, pp. 148-155.

Zhang, Y. et al, (2020). OSCA: An Online-Model Based Cache Allocation Scheme in Cloud Block Storage Systems. *Proceedings of 2022 USENIX Annual Technical Conference (USENIX ATC 20)*, online, pp. 785-798.