Electronic Preprint for Journal of Information Processing Vol.26

Technical Note

Identifier-first Multi-datacenter Synchronization for In-Memory State Management

Arif Herusetyo Wicaksono[1,a)] Kazuhide Aikoh[1,b)]

Received: April 21, 2017, Accepted: September 5, 2017

Abstract: With the needs of consistency between multiple datacenters for state management in transaction processing system, multi-datacenter synchronization is required, both for reducing response time and for providing fail-over in the event of disaster. However, in synchronization, prioritizing consistency causes latency degradation, while preserving response times may cause data inconsistencies, both of which are unacceptable. Therefore, we propose a synchronization method with two characteristics: prioritization of identifier transmission along with transaction identifier, and replication status verification prior to access. By applying the proposal to prototype in-memory KVS, we verified the response time degradation can be reduced while eliminating potential of conflicts.

Keywords: in-memory KVS, disaster recovery, multi-datacenter synchronization

1. Introduction

Along with improvement in sensors and networking technologies which enables various IoT applications, latency and capacity requirements of transaction processing system which handles the resulting data are increasing as well. State management in such transaction processing system often adopt in-memory Key-Value Store (KVS) due to its benefit of providing low-latency response by managing data in main memory, while also horizontally scalable to multiple servers for adding new capacity.

However, transaction processing system often involves several datacenters located in major metropolitan areas to reduce response time by placing datacenters close to data source. Furthermore, having multiple datacenters also has the advantage of ensuring business continuity in the event of a large scale disaster, such as earthquake. Meanwhile, in applications where IoT data is utilized in critical system, for example using sensors to improve efficiency, ensuring data consistency is a must. Therefore, various in-memory KVS adopt distributed consensus algorithm, such as Paxos [1] to obtain strong consistency [2], ensuring correct results is returned to client even when processing involves multiple servers.

To satisfy latency, and consistency requirements, multi-datacenter synchronization is required. This paper is organized as follows. In Section 2, we discuss multi-datacenter synchronization utilized by existing in-memory KVS. Then, we describe our proposal in Section 3, and discuss evaluation in Section 4. In Section 5, we discuss the recovery automation approach. Finally, we conclude our discussion in Section 5.

[1] Hitachi, Ltd. R&D Group, Yokohama, Kanagawa 244–0817, Japan
[a)] arif.wicaksono.ou@hitachi.com
[b)] kazuhide.aikoh.qr@hitachi.com

2. Related Works

There are two approaches to perform multi-datacenter synchronization, preserving latency via asynchronous method or prioritizing consistency via synchronous method.

Synchronous [3] replication satisfies consistency requirements, by ensuring replication is finished. Main datacenter operation waits for response from backup datacenter before responding to client. By confirming data is successfully written in both main and backup datacenters, conflict may never occur. However, a main datacenter operation grows proportionally to data size, due to round-trip network communication to backup datacenter.

Asynchronous [3] replication satisfies latency requirements by ensuring low-latency response. By replicating data after main datacenter operation is finished, low-latency response of the system is preserved. However, if disruption occurs prior to completion of the data transmission to the backup datacenter, there is potential of conflicts.

In-memory KVS which operate with strong consistency by default exist, such as Riak [4] and Kudu [5]. While Riak support asynchronous replication which sacrifice the strong consistency for multi-datacenter deployment, Kudu does not support multi-datacenter yet. Performing synchronization by adopting distributed consensus across multi-datacenter is also unacceptable, because of the response time degradation

We aim to develop a semi-synchronous replication which can maintain both response time and data consistency.

3. Identifier-first Multi-datacenter Synchronization

3.1 Synchronization Design

We want to satisfy both latency and consistency requirements to ensure critical data are consistent.

To satisfy latency requirement, client response latency has to be reduced. In synchronous replication, response time degrades significantly because of round-trip network communication between main and backup datacenters required to transmit the data. Therefore, we decided to send only a small part of the data to reduce the transmission time between main and backup datacenters.

To satisfy consistency requirement, any potential of conflicts has to be prevented. In existing method, preventing conflicts is done by ensuring replication is completed. We think only error detection is enough to prevent conflicts. Error detection is possible if we know that changes have occurred in the main datacenter. Therefore, we decided to focus on enabling error detection instead of finishing replication.

Therefore, our proposal for multi-datacenter synchronization has two main components, separate key-value replication for reducing synchronization time, and replication status verification at disaster occurrence. In this section, we will describe each component and discuss our design decision.

3.2 Identifier-first

Instead of transmitting complete data, we satisfy latency requirements by sending only a small part to backup datacenter.

Therefore, we separated the key from its value, and prioritized it for transmission. The key size is small, thus it can be transmitted quickly. On the other hand, value size is big, and majority of synchronization time is the value replication time.

We introduce a new variable called transaction identifier. Transaction identifier consists of distinct data:

Sequence information utilized to recombine key back to its value in backup datacenter, and maintain the order of transaction as it happens in the main datacenter. The log number of distributed consensus algorithms for ensuring strong consistency can be utilized as sequence information.

The key is transmitted along with the transaction identifier first. When backup datacenter has acknowledged the transmission, main datacenter respond to client.

Meanwhile the corresponding value is transmitted with the key and its transaction identifier at times independent of the key transmission. Due to its bigger size, the value transmission takes longer to finish compared to the key packet. It is also possible to send values in batches.

Once replication of the value for a key is finished, then the key and its corresponding value is matched by transaction identifier and moved to key-value store in the backup datacenter. **Figure 1** provides the flowchart of replication in our method due to a put request from a client.

We would like to point out that due to the small size of key and transaction identifier, it is feasible to transmit them by a single packet. If the wired network connection between main and backup datacenters is unavailable, it is also possible to transfer them via wireless networking, for example satellite or cellular connection. If response time degradation is completely unacceptable, while consistency requirements can be relaxed, it is also possible to send the key packet after responding to client, and our proposal still require less time to achieve consistency.

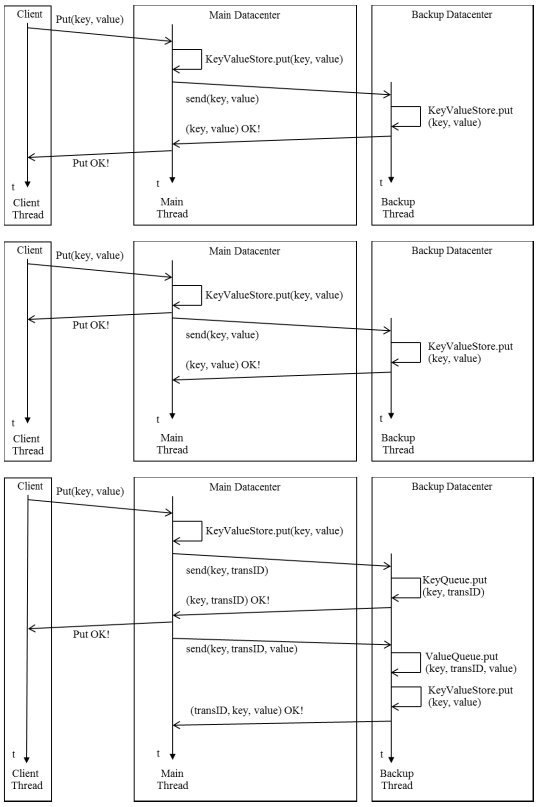


Fig. 1 Sequence chart of replication after put request by a client during normal operation for synchronous, asynchronous and our proposal, respectively.

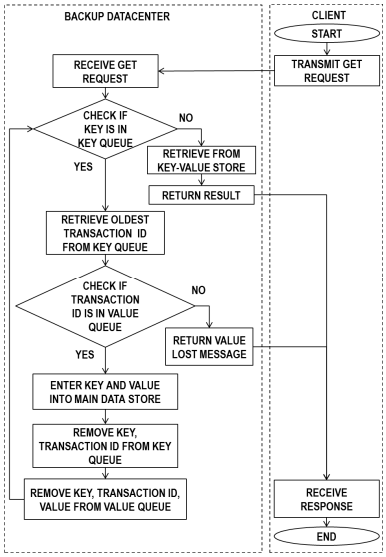


Fig. 2 Flowchart of a get request by client to backup datacenter in the event of fail-over.

3.3 Replication Status Verification

In our datacenter configuration, as long as main datacenter is operational, backup datacenter does not respond to read request from client. Client prioritizes access to main datacenter. If main datacenter is not available, client access backup datacenter.

In the event of disaster, client may access backup datacenter because main datacenter is not available. In such cases, a read request is received by the backup datacenter for a key.

When backup datacenter received the request, before checking the key-value store, we added a new step. We routed the request

to check for unfinished replication managed in a queue.

Backup datacenter checks for the existence of the key in the unfinished key queue. If the key does not exist, there is not any unfinished replication for the request, which means it can be forwarded to the key-value store and processed normally.

However, if requested key exists in the unfinished key queue, the read request is deferred until the replication is completed. If the replication cannot be completed, then missing data is detected, and further access is prevented to maintain consistency. **Figure 2** illustrated the flow of replication status confirmation.

We would like to point out that consistency protection via error detection is not the same as consistency protection via completed replication. While in existing approaches, data can be processed normally after fail-over, our proposal prevented access to the missing data. Thus, we sacrifice complete replication, and instead provide logic to deal with missing data.

4. Evaluation

To evaluate the effectiveness of our proposed method where key is separated from its corresponding value, we compared its latency performance to existing approaches where key is transmitted simultaneously with its value. We developed two prototypes of KVS, one for our proposed key-value separated method and one for simultaneous key-value transmission method. We implemented a new prototype for KVS, instead of utilizing existing one because we want to ensure objective comparison, where we can control all evaluation parameter.

We performed the evaluation on bare metal servers with specifications as listed in **Table 1**. The servers are connected via IPv4 network. To simulate distance between datacenters, we introduced latency of 5 milliseconds by utilizing tc utility.

4.1 Response-time Evaluation

We evaluated the response-time performance of our proposed method by measuring the time required to achieve consistency protection and prevent conflicts.

In this evaluation, we compared our proposal against simultaneous synchronous replication to understand how response time can be reduced. Since our proposal requires only identifier data to be transmitted, the effect on response time should be significantly lower.

Therefore, we measured three points in main datacenter:

1. Time of arrival for read request.
2. Time of attainment for consistency protection.
3. Time of completion for finishing synchronization.

In our proposed method, time of attainment for consistency protection corresponds to the time where the key and transaction identifier reaches backup datacenter. In simultaneous method, time of attainment for consistency protection corresponds to the time when the replication is finished. Figure 1 illustrate the difference in a sequence chart.

Table 1 Evaluation environment.

Parameter		Value
Hardware	# of Host	2xQuantaPlex
	CPU	Intel Xeon® E5-2640v3 (Haswell) x 2
	Memory	DDR4-1866 64GB
	NIC	NIC Intel X540-AT2 10GbE
Software	OS	CentOS 7 x86_64
	Kernel	3.10.0-229 x86_64
	Java	Java SE 8u77

In our proposed method, consistency protection time and synchronization finish time are two distinct measurement points. Meanwhile, in simultaneous method, they are the same measurement point.

In our evaluation, we focused our evaluation on the latency, because backend storage such as our KVS is not limited by the database layer response to application layer, but by the application layer response to the client. Significant processing is performed by the application, while storing the data to the in-memory KVS is finished in comparatively shorter time.

4.2 Data Consistency Evaluation

We evaluated the consistency protection capability of our proposed method by simulating the disaster and understanding how our proposed method deals with conflicts.

In this evaluation, we compared our proposal against simultaneous asynchronous replication to understand how much data can be protected. Since our proposal provide error detection capability by verification of replication status prior to access, the data protection capability should be higher.

In simultaneous asynchronous replication, two states exist:

Complete A key-value pair is written in both main and backup datacenters. If fail-over occurs, backup datacenter can serve the same key.

Data loss A key-value pair is written in main datacenter, but backup datacenter does not know of its existence. If backup datacenter overwrite the same key with different value, consistency is lost and conflict occurs. Meanwhile, in our proposed method, the "data loss" state is replaced because we provide verification prior to data access.

Error detect A key-value pair is written in main datacenter, but only the key is available in backup datacenter. If fail-over occurs, backup datacenter block access to the key to prevent conflicts and maintain consistency

In consistency evaluation, we operated both method at maximum throughput, and disabled network connection via scripts. Then, we compared the number of key-value pairs with different states.

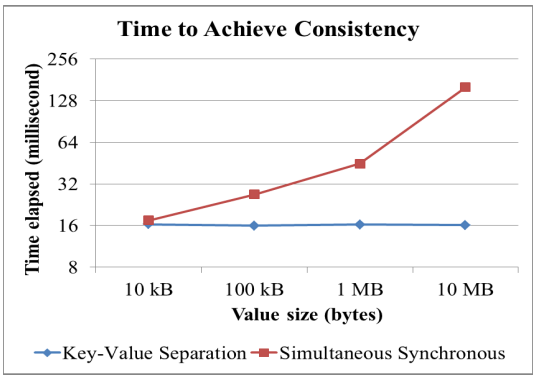
4.3 Evaluation Results and Discussion

In this section, we discussed the evaluation results for both response-time and data consistency evaluation.

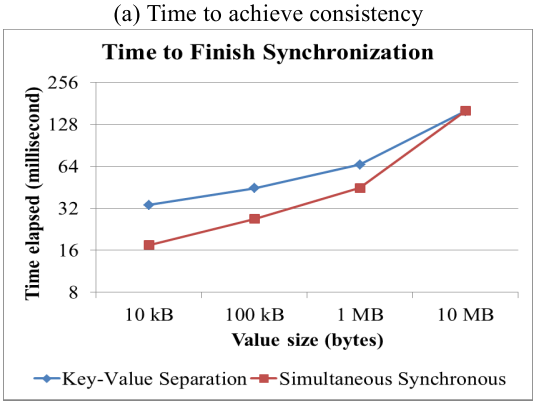
Figure 3 (a) depicts the comparison between our proposed key-value separation method and simultaneous synchronous replication method for the time to achieve consistency protection.

When value size is less than 10 kB, our proposed method did not provide significant advantage compared to simultaneous transmission approach. The difference of attainment time for consistency protection is negligible.

However, when the value size grows bigger, our proposed method provided faster time to achieve consistency protection. In simultaneous approach, the time to achieve consistency protection grows proportionally to value size, which results in longer time to achieve consistency protection. In our proposed method, consistency protection is achieved when the key reaches backup



(a) Time to achieve consistency



(b) Time to finish synchronization

Fig. 3 Response-time evaluation results.

site. Therefore, time to achieve consistency protection in our proposed method does not grow with value size, and depends solely on inter-datacenter network latency.

Time to achieve consistency protection grows proportionally to the simulated latency, because latency corresponds to the required time to finish round-trip communication from main to backup server.

Figure 3 (b) depicts the total synchronization time for our proposed key-value separation method and simultaneous transmission method. As the trade-off for faster consistency protection time, the total time for synchronization is longer. The reason for this behavior is our proposed method requires two network transmissions for finishing synchronization. On a side note, we believe parallel processing can improve the performance of our proposed method to alleviate this performance trade-off.

Similar to the time to achieve consistency evaluation, time to finish synchronization grows proportionally to simulated latency. Due to the multiple round-trip communications required to finish the synchronization, our proposed method requires longer to finish the synchronization.

For value size of 100 kB, 1 MB, and 10 MB, our proposal reduces the time to achieve consistency protection by 39%, 64%, and 90% respectively.

Figure 4 depicts the evaluation results for data consistency evaluation for our proposed method against simultaneous asynchronous method.

In value size of 10 kB, our proposed method could complete less replication compared to simultaneous asynchronous replication. Due to our design of prioritizing transmission of key, there are some key-value pairs whose keys exist without the corresponding value.

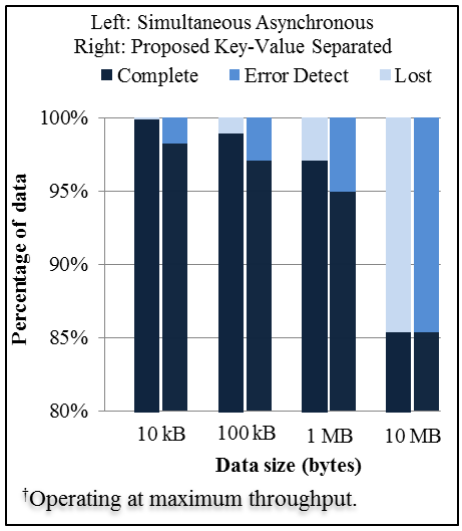


Fig. 4 Data consistency evaluation results.

For value bigger than 10 kB our proposed method is effective in the form of additional consistency protection. In simultaneous asynchronous method, there are increasing amount of data loss proportional to data size. Meanwhile our proposal managed to provide consistency protection by verifying status of replication before accessing the key-value store.

As value size grows bigger due to application improvement, we believe our proposal will provide significant advantage over approach where key and value are transmitted simultaneously.

5. Conclusion

We proposed a multi-datacenter synchronization method for in-memory KVS to improve consistency protection while maintaining response time.

Our proposed method separates key from its value, prioritizes the key for transmission to the backup datacenter, and utilizes key along with transaction identifier to prevent inconsistencies in the event of large-scale disasters. We demonstrated that our method provided faster synchronization time compared to simultaneous synchronous replication, while also manage to prevent conflicts unlike simultaneous asynchronous replication. Therefore, we believe our method could provide benefit for processing big data.

References

[1] Lamport, L.: Paxos made simple, *ACM Sigact News*, Vol.32, No.4, pp.18–25 (2001).
[2] Sakr, S. and Gaber, M. (Eds.): *Large Scale and Big Data: Processing and Management*, CRC Press (2014).
[3] Zavoral, F., Yaghob, J., Pichappan, P. and El-Qawasmeh, E. (Eds.): *Networked Digital Technologies, Part II*, Springer (2010).
[4] Basho Riak Replication Model (online), available from ⟨http://docs.basho.com/riak/kv/2.2.1/developing/app-guide/strong-consistency/⟩ (accessed 2017-03-15).
[5] Cloudera Kudu (online), available from ⟨http://www.cloudera.com/documentation/betas/kudu/0-5-0/topics/kudu.html⟩ (accessed 2017-03-15).

Arif Herusetyo Wicaksono is a research staff working on data analytics for Research and Development Group of Hitachi, Ltd.

Kazuhide Aikoh is as a researcher working on data management platform for Research and Development Group of Hitachi, Ltd.