Key Techniques and Supporting Tools of Consistency Checking in Distributed System \*

Firstname Surname†  
 Dept. Computer Science and Technology  
 Nanjing University  
 Nanjing Jiangsu China

ABSTRACT

Nowadays more and more Internet-scale systems replicate data in distributed data centers for better performance. However, replicas bring us both convenience and the issue of conflict resolution. In most scenarios, distributed system designers have to sacrifice consistency for availability and partition tolerance. Therefore, some consistency models have been proposed for tradeoffs between consistency and performance. In our research, we investigate the problem of checking whether a given execution trace of a distributed data-store system adheres to a certain consistency model. And we design a scalable platform which can give the traces produced by different distributed systems an efficient check upon different consistency models. Moreover, we devise some techniques for the optimization of checking and a suite of supporting tools.

KEYWORDS

consistency, testing, distributed system

1 Introduction

Today more and more Internet-scale systems replicate data in distributed data centers for large throughputs, low latency and high fault-tolerance. Just as there is no free lunch in the world, replicas bring us both the improvement of performance and the issue of conflict resolution which is a rather thorny problem.

In 2000, Eric Brewer introduced the idea that there is a fundamental trade-off between consistency, availability and partition tolerance, which has become known as the *CAP Theorem* [1]. Since it is impossible to achieve both consistency and availability in an unreliable system, it is necessary to sacrifice one of these desired properties [2]. Partition tolerance is indispensable in the practical scene because the system must continues to operate despite an arbitrary number of messages being dropped or delayed by the network between nodes. Also we cannot sacrifice availability considering the user experience, which every request receives a non-error response in a certain time in spite of no guarantee that it contains the most recent write. Hence, programmers have no choice but to use weak consistency to provide specifications instead of strong consistency.

However, there is a huge gap between totally weak and just a bit weaker that strong consistency. So many consistency models such as quiescent consistency, basic eventual consistency, causal consistency, sequential consistency, have been proposed to help specify and verify distributed systems especially distributed data storage. Different consistency models have different guarantees, different performance and different difficulty of implement. Given the potentially-huge amount of system that relies on these distributed data storage systems, it is important to maintain precise specifications and ensure that implementations adhere to their consistency specifications.

Testing a distributed data storage system raises two issues. First, it is hard to derive a suitable set of testing scenarios, e.g., faults to inject into the system and the set of workloads to be executed. Second, it is hard to check the given execution satisfies the consistency model efficiently. The Jepson framework have been design to solve the first issue by using randomization. However, the second issue is ignored in some sense.There are some consistency models which specify the weak-consistent distributed data storage, but we find few validation tools can check real executions upon various consistency models due to the complexity of checking correctness.

In this work, we aim to develop a platform that checks the executions from different systems upon different consistency specification under the visibility-arbitration framework in order to get a precise specification of a system [3]. And we use pruning technique to reduce the searching space of checking, which makes it possible to check large-scale execution traces in an acceptable time. With these tools, we can obtain the relatively precise consistency specification of the system and find the subtle consistency difference between implementations of the same algorithm thus we can measure the consistency of distributed data storage systems.

We plan to provide an experimental evaluation of our platform on the executions of Riak, which claims to implement a set of conflict-free replica data types. We also want to check the executions of Cassandra, which can change its consistency through the setting of quorum.

2 Related Work

Burckhardt gives us the specification methodology which uses visibility relations and a formalization of consistency citerion [3]. The visibility relation represent the fact that an operation observes the effects of another operations. Emmi and Enea develop a simple annotation language for specifying weak-consistent operations in Java concurrent objects via visibility relaxation, which also naturally capture consistency mechanisms in the distributed systems and also develop a validation methodology for specifying software whose operations satisfy multiple distinct consistency levels [4]. Their relaxed-visibility specification is more expressive than Burckhardt's along a few different axes, which can specify the weak consistency on replica and message-passing.

Biswas and Enea have done a great work on checking transactional consistency which inspires us a lot [5]. However, this work focuses on consistency models for transaction of modern databases while our research focuses on consistency models for replica data storage such as conflict-free replica data type. In other words, Biswas and Enea aim to check consistency models like read committed, read atomic, and causal consistency while we aim to check consistency models like basic eventual consistency and weak consistency via visibility relaxation.

Chao Wang and Enea address the problem of specifying and verifying CRDTs by introducing a new correctness criterion called Replication-Aware Linearizability which is inspired by Linearizability, but they do not use it to verify client applications of CRDTs [6].

Michael Emmi have developed the first completely-automatic algorithm for checking weak consistency of concurrent object implementations and identified an optimization to weak-consistency checking [7]. Since this algorithm is designed to check java concurrent objects, we want to use its framework to implement a platform to check distributed systems.

3 Problem Formulation

4 Research Plan

5 Preliminary Results

6 Discussions

7 Conclusion

1.1 Heading Level 2

In the below paragraph, it is explained how alt-txt value is placed in **MS Word 2010**. To add alternative text to a picture in Word 2010, follow these steps:

1. In a Word 2010 document, insert a picture.
2. Right click on the inserted picture and select the **Format Picture** option.
3. Select the **Alt Txt** option from the left-side panel options.
4. In the "Title:" and "Description:" text boxes, type the text you want to represent the picture, and then click "Close".

Below are steps to place alt-txt value in **MS Word 2013/2016**. To add alternative text to a picture in Word 2013/2016, follow these steps:

1. In a Word 2013/2016 document, insert a picture.
2. Right click on the inserted picture and select the **Format Picture** option.
3. In the settings at the right side of the window, click on the "Layout & Properties" icon (3rd option).
4. Expand **Alt Txt** option.
5. In the "Title:" and "Description:" text boxes, type the text you want to represent the picture, and then click "Close".

*1.1.1 Heading Level 3.* Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

*1.1.1.1 Heading Level 4.*Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

ACKNOWLEDGMENTS

Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

REFERENCES

[1] Brewer, E. (2010, July). A certain freedom: thoughts on the CAP theorem. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing* (pp. 335-335).

[2] Gilbert, S., & Lynch, N. (2012). Perspectives on the CAP Theorem. *Computer*, *45*(2), 30-36.

[3] Burckhardt, Sebastian. "Principles of eventual consistency." (2014).

[4] Emmi, Michael, and Constantin Enea. "Weak-consistency specification via visibility relaxation." *Proceedings of the ACM on Programming Languages* 3.POPL (2019): 1-28.

[5] Biswas, Ranadeep, and Constantin Enea. "On the complexity of checking transactional consistency." Proceedings of the ACM on Programming Languages 3.OOPSLA (2019): 1-28.

[6] Wang, Chao, et al. "Replication-aware linearizability." Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2019.

[7] Emmi, Michael, and Constantin Enea. "Monitoring weak consistency." International Conference on Computer Aided Verification. Springer, Cham, 2018.Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00