On the Consistency of Heterogeneous Composite Objects

Alysson Bessani Ricardo Mendes Tiago Oliveira
Faculdade de Ciências/LaSIGE, Universidade de Lisboa – Portugal
anbessani@ciencias.ulisboa.pt , {rmendes, toliveria}@lasige.di.fc.ul.pt

1. Introduction

Several recent cloud-backed storage systems advocates the composition of a number of cloud services for improving performance and fault tolerance (e.g., [1, 3, 4]). An interesting aspect of these compositions is that the consistency guarantees they provide depend on the consistency of such base services, which are normally different.

In this short paper we discuss two ways in which these services can be composed and the implications in terms of the consistency of the composed object. Although these techniques were devised (or observed) when solving practical problems in dealing with the eventual consistency guarantees of current cloud storage services (e.g., Amazon S3 [6], Windows Azure Blob Storage [7]), we believe they might be of general interest, and deserve the attention of the community. In particular, we want to discuss some initial ideas about the theoretical underpinnings of object compositions in which base objects provide different consistency guarantees.

2. Model

We consider an asynchronous distributed system composed by a number of processes interacting through a number of shared memory objects (e.g., rw-registers, key-value stores). Furthermore, we consider two types of objects: *base objects* and *composite objects*. The latter are built on top of the former, by aggregating them. For the sake of simplicity, we assume all operations provided by the objects are wait-free. However, the objects may provide different consistency guarantees.

In our discussion we consider an hierarchy of consistency models. More specifically, let $c_1, c_2, ..., c_n$ be different consistency models, we assume $c_1 \subset c_2 \subset ... \subset c_n$, meaning that the guarantees provided by c_i are also provided by c_{i+1}, i.e., c_{i+1} provides the same guarantees of c_i plus some others. Notice that not all consistency models can be organized as an hierarchy, however, several important models employed in practical system do, e.g., eventual $\subset$ FIFO $\subset$ causal $\subset$ sequential consistency.

PaPoC'15, April 21-24, 2015, Bordeaux, France.

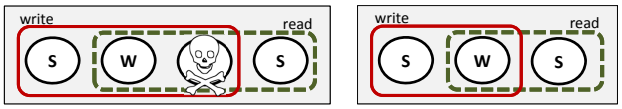
http://dx.doi.org/10.1145/2745947.2746687

3. Consistency proportionality

First, we consider the composition of a set of fail-prone base objects for implementing a fault-tolerant one. The simplest case is the use of several fail-prone base registers to implement a t-tolerant register. There are many works that study such compositions (e.g., [1, 2]). In this scenario, we consider the following question: assuming that base objects provide different consistency guarantees, what will be the consistency provided by the composite object?

To the best of our knowledge, the only work that considers such *consistency diversity* is DepSky [1]. This work defines the notion of *consistency proportionality* in the following sense: if the underlying base objects support at least a consistency model $\mathscr{C}$, the composite object provides a consistency model $\mathscr{C}$. This means that if the underlying base objects are heterogeneous in terms of consistency guarantees, the composite object provides the weakest consistency among those provided. As shown in the paper, this holds for several consistency models.

Figure 1 illustrates the notion of consistency proportionality for two popular quorum systems.



(a) Dissemination Byzantine Quorum (b) Majority quorum

Figure 1. Composite object write and read quorums for both Byzantine and crash fault-tolerant systems.

Figure 1(a) represents the write and read quorums of a dissemination Byzantine quorum system [5] that tests the integrity and authenticity of the data through the use of authenticated data (e.g., by using signatures). In the figure we consider $n = 4$ base objects, tolerating one fault ($f = 1$).

As can be seen, these systems use read and write quorums with at least $f+1$ base objects in their intersections. In such scenario, in the worst case where f objects fail and f up-to-date objects are not accessed in the read, the read value will depend on the single correct object of the intersection, which may be the one providing the weakest consistency in the system. Consequently, the consistency model of the composite object will be defined by this base object. An analogous approach happens in majority quorum systems (Figure 1(b)), where in the worst case f up-to-date objects do not respond to a read (because they fail crashing).

4. Consistency anchors

Cloud storage services such as Amazon S3 [6] are broadly used nowadays due to their (almost) infinite scalability and pay-as-you-go model. However, most of these systems are constrained by a key-value REST interface and only ensure eventual consistency. Recently, the SCFS cloud-backed file-system introduced the notion of *consistency anchor* to increase the consistency guarantees of weakly-consistent cloud storage services [3]. A similar idea was independently devised for the Hybris key-value store [4].

The technique works by composing a strongly-consistent base object (with limited storage capacity) with one or more weakly-consistent base objects, making the consistency of the composed object similar to the strongly-consistent one, called consistency anchor. In SCFS, a strongly-consistent metadata service is composed with cloud storage services to ensure Linearizability [3].

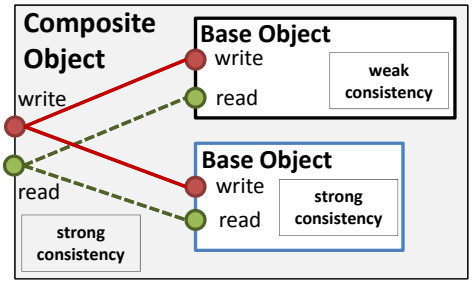


Figure 2. Strong consistent composite object resulting of a consistency anchor and a weak consistent base object.

Figure 2 depicts a composite object integrating two base objects, one providing strong semantics and another with weak consistency. The main objective is to implement the operations of the composite object in such a way that it can take advantage of the base object with strong consistency semantics to increase the consistency guarantees provided by the composite object itself. In this way, besides the weak semantics provided by one of its objects, the composite object is able to provide strong consistency semantics.

This idea become even more interesting when we encapsulate a weakly-consistent composition of objects (that uses, for instance, several fail-prone objects to create a fault-tolerant one) as the weakly-consistent base object of this technique.

5. Open Questions

This paper describe some initial ideas that have been applied to practical cloud-backed storage systems for dealing with the consistency-heterogeneity of cloud services. We believe these ideas open a number of interesting questions that require further investigation, namely:

- What would be the consistency model satisfied by a composite object based on a set of objects satisfying consistency models that cannot be organized in an hierarchy?
- We have already some preliminary results for two popular quorum systems (dissemination Byzantine quorums and majority quorums), but what would be the consistency provided by other types of quorum systems using heterogeneous objects?
- Would it be possible to define more efficient consistency-proportional algorithms for implementing read/write-registers objects? What about other object types?
- Can the notion of consistency anchor be used for increasing the consistency of other object types, beside read/write registers?

Acknowledgments

This work was supported by FCT through the LaSIGE Research Unit, ref. UID/CEC/00408/2013 and by EU H2020 Program, through the SUPERCLOUD project (643964).

References

[1] Alysson Bessani, Miguel Correia, Bruno Quaresma, Fernando Andre, and Paulo Sousa. DepSky: Dependable and secure storage in cloud-of-clouds. *ACM Transactions on Storage*, 9(4), 2013.

[2] Ittai Abraham, Gregory Chockler, Idit Keidar, and Dahlia Malkhi. Byzantine disk paxos: optimal resilience with byzantine shared memory. *Distributed Computing*, 18(5), 2006.

[3] Alysson Bessani, Ricardo Mendes, Tiago Oliveira, Nuno Neves, Miguel Correia, Marcelo Pasin, and Paulo Verissimo. SCFS: a shared cloud-backed file system. *Proc. of the 2014 USENIX Annual Technical Conference (ATC'14)*, 2014.

[4] Dan Dobre, Paolo Viotti, and Marko Vukolic. Hybris: Robust hybrid cloud storage. *Proc. of the 5th ACM Symposium on Cloud Computing (SoCC'14)*, 2014.

[5] Dahlia Malkhi and Michael Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4), 1998.

[6] Amazon S3. *http://aws.amazon.com/s3/*

[7] Microsoft Azure Storage. *http://azure.microsoft.com/en-us/services/storage/*