Towards Quantifiable Eventual Consistency

Francisco Maia, Miguel Matos and Fábio Coelho
INESC TEC & U. Minho, Braga, Portugal

Keywords: Large Scale, Data Stores, Epidemic Protocols.

Abstract: In the pursuit of highly available systems, storage systems began offering eventually consistent data models. These models are suitable for a number of applications but not applicable for all. In this paper we discuss a system that can offer a eventually consistent data model but can also, when needed, offer a strong consistent one.

1 INTRODUCTION

In recent years, extensive research work has been focusing on large scale data storage (ex. (Chang et al., 2006; Lakshman and Malik, 2010)). Large scale systems, composed by several thousand of machines, raise several interesting challenges predominantly related with their instability. In fact, an increase in system scale is necessarily accompanied by the increase in the number and type of failures. Strikingly, failures can actually become the rule, not the exception (Schroeder and Gibson, 2007). The impact of failures in the design of a data storage system can be significant and designing fault tolerant data storages is a non trivial task.

Traditional data storage systems were designed to provide four well-defined properties: atomicity, consistency, isolation and durability. There are well known approaches to provide these properties in a centralized system. In this scenario the failure of the machine necessarily means unavailability, even if the above properties are always guaranteed. Moreover, centralized systems can only grow their capacity to answer increased demand by adding physical resources to the existing machine. In other words, they cannot scale out. In order to provide better availability and scalability, the natural answer are distributed systems where demand is balanced across several machines and where an increase in demand is handled by the addition of more machines, i.e. scale out. The failure of a number of machines can be tolerated by assigning their tasks to the remaining ones. However, distributed systems require intricate coordination protocols in order to guarantee the four properties described and these protocols exhibit performance limitations when deployed in large scale scenarios. The are know to struggle with deployments of more than a few tens of nodes.

In the pursuit of available, scalable and usable storage systems, compromises between availability and the guarantees offered have been proposed. In particular, the idea of eventual consistency has been the subject of intense research work (Vogels, 2009). Eventual consistency considers that data is allowed to be temporarily inconsistent but eventually converges to a consistent state. This relaxation allows data storage systems to avoid costly coordination protocols and offer continuous availability even in the presence of failures. Even so, the concrete implementations of the notion of eventual consistency are several and distinct. There is no clear and consensual definition of the term and of the programming model it implies. Different implementations originate different models and there is no easy way to quantify the guarantees provided by each model or even to compare them. As a consequence, using and reasoning about an eventual consistent system becomes very complex.

In this paper, we propose a new approach to data storage. We leverage previous work on epidemic large scale data storage and on a disruptive epidemic total order protocol. We discuss a system that can effortlessly be configured to provide strong consistency or a weaker consistency model.

2 DataFlasks - LARGE SCALE STORAGE

DataFlasks (Maia et al., 2014) is a data store aimed at very large scale deployments. Entirely built on top of epidemic protocols, this system is able to guarantee data persistence even in the presence of high levels of failures.

In DataFlasks, nodes are organized into groups. Each group is responsible for a subset of the data and groups do not overlap. A client application can write key-value objects to DataFlasks by issuing a *put* operation and later retrieve them via a *get* operation. Objects are carry a version and the triple (key,version,value) is considered unique by the storage system. However, DataFlasks does not enforce any kind of data consistency. As a consequence a client application is responsible for explicitly manage data versioning in order to provide consistency.

We leverage the work on DataFlasks in order to take advantage of its resilience properties. Our proposal is to use DataFlasks as a persistence layer.

3 EpTO- STRONG CONSISTENCY WITH HIGH PROBABILITY

EpTO(Matos et al., 2015) is a scalable and robust total order protocol. While validity, integrity and total order properties are deterministic, the agreement property of classic total order is relaxed to be probabilistic and implemented at the expense of epidemic dissemination protocols, know precisely for their scalability and robustness. This allows EpTO to scale to thousands of nodes, at least an order of magnitude larger than previous proposals, which enables building very large systems with strong (consistency) semantics.

Combining DataFlasks with EPTO, allows us to offer total order on data writes to the store and, as a consequence, a strong consistency model. DataFlasks group construction mechanism and the fact that each group dataset is disjoint (Guerraoui and Schiper, 1997) allows us to use the EPTO protocol only on a restricted subset of the system nodes allowing the system to scale.

4 RELAXED CONSISTENCY

With DataFlasks and the EPTO protocol we are able to provide a storage system with strong consistency with high probabiility. Moreover, we are able to achieve this even for a deployment of several thousand of nodes. Naturally, in order to achieve such level of consistency a latency cost must be paid.

In DataFlasks, every node can receive requests. When a write request is received, in order to guarantee strong consistency with high probability, nodes must follow the EPTO protocol to ensure they assign the correct version to that write operation. This may result in increased request latency.

Our proposal is offering a weaker consistency model where there is a small probability of temporarily considering an incorrect version for write operations. It works as follows. Let us consider a system component that gives nodes an estimate of the time it takes a message to reach all nodes in their DataFlasks group. Recall that each group is responsible for a certain subset of the data. This time estimate is associated with a probability of being correct. When a node receives a write request automatically becomes the coordinator for that write. It looks at its current state and assigns the write a version it thinks is the correct one based only on local knowledge. It disseminates to all the other nodes in the system the write operation and the version. Next, it waits for an amount of time equal to that given by the estimation. If no write is received for that object in such time, it stores the object with the assigned version. All the other nodes, when receiving such object and version go through the same procedure. Each time a node receives a conflicting request the one that was proposed by the node with smaller identification wins.

This simple model allows the user to explicitly tune the desired level of consistency by configuring the time estimation component. When the time estimation component is configured with a probability of 1 of being correct, the system automatically discards this algorithm and uses the EPTO protocol. For every value smaller than 1, the system will relax consistency guarantees and become faster. This way, the same system architecture is able to provide a stronger or a weaker consistency model according to the priority given to consistency and performance.

5 CHALLENGES

The weaker consistent model we propose shares similarities with the unconscious model presented in (Baldoni et al., 2006). In it, processes are not aware - i.e. are unconscious - of when consistency has been reached. Our proposal allows for consciousness in the sense that processes may know with probability 1 that a consistency state has been reached while also allowing for unconscious operation. We believe exposing and quantifying these notions to the application is an interesting research path, and in particularly its interplay with the reliability guarantees of the gossip mutation and the freshness of the membership provided by DataFlasks group construction protocols. Besides, the consistency constraints imposed by operations affecting multiple DataFlasks groups need to be studied (Guerraoui and Schiper, 1997).

We aim at quantifying these trade-offs and constraints such that one can achieve a better understanding of the consistency models underlying modern distributed applications and in particular, studying how the relaxations proposed compare with the stronger consistency models.

ACKNOWLEDGMENT

This work was part-funded by project CoherentPaaS: A Coherent and Rich PaaS with a Common Programming Model (FP7-611068).

REFERENCES

Baldoni, R., Guerraoui, R., Levy, R. R., Quéma, V., and Piergiovanni, S. T. (2006). Unconscious eventual consistency with gossips. In *Stabilization, Safety, and Security of Distributed Systems*, pages 65–81. Springer.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2006). Bigtable: a distributed storage system for structured data. In *The Symposium on Operating Systems Design and Implementation*. USENIX.

Guerraoui, R. and Schiper, A. (1997). Total order multicast to multiple groups. In *Distributed Computing Systems, 1997., Proceedings of the 17th International Conference on*, pages 578–585. IEEE.

Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. In *ACM SIGOPS Operating Systems Review*. ACM.

Maia, F., Matos, M., Vilaça, R., Pereira, J., Oliveira, R., and Riviere, E. (2014). Dataflasks: epidemic store for massive scale systems. In *2014 IEEE 33rd International Symposium on Reliable Distributed Systems (SRDS)*, pages 79–88. IEEE.

Matos, M., Mercier, H., Felber, P., Oliveira, R., and Pereira, J. (2015). Epto: An epidemic total order algorithm for large-scale distributed systems. In *Proceedings of the 16th Annual Middleware Conference*, Middleware '15, pages 100–111, New York, NY, USA. ACM.

Schroeder, B. and Gibson, G. A. (2007). Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of the 5th USENIX Conference on File and Storage Technologies*. USENIX.

Vogels, W. (2009). Eventually consistent. *Communications of the ACM*, 52(1):40–44.