Replica Divergence in Data-Centric Consistency Models

Leticia Pascual-Miret, Francesc D. Muñoz-Escó

Instituto Universitario Mixto Tecnológico de Informática
Universitat Politècnica de València
46022 Valencia (SPAIN)

{let,fmunyoz}@iti.upv.es

Technical Report TR-IUMTI-SIDI-2016/001
Replica Divergence in Data-Centric Consistency Models

Leticia Pascual-Miret, Francesc D. Muñoz-Escó
Instituto Universitario Mixto Tecnológico de Informática
Universitat Politècnica de València
46022 Valencia (SPAIN)

Technical Report TR-IUMTI-SIDI-2016/001

e-mail: \{let, fmunyoz\}@iti.upv.es

March 29, 2016

Abstract

Eventual consistency is demanded nowadays in highly scalable and available geo-replicated services. According to the CAP theorem, when network partitions may arise a distributed service should choose between being strongly consistent or being highly available. Since scalable services must be available, relaxed consistency is the regular choice. Eventual consistency is not a regular data-centric consistency model, but only a state convergence property to be added to a relaxed consistency model. This paper discusses which data-centric consistency models are not implicitly convergent and, because of this, provide an adequate basis for building eventually consistent services.

1 Introduction

Eventual consistency [25] has received a lot of attention due to the emergence of highly scalable distributed services in the cloud computing area. Those services need to be both scalable and adaptive, ensuring good levels of functionality, performance and responsiveness (i.e., QoS) – combined with a low cost – to their users, and of economical profit to their providers. In order to reach those levels of performance and responsiveness when the incoming workload being supported is high, the consistency among server replicas should be relaxed and this explains why eventual consistency has become so popular.

Scalable services are commonly deployed onto multiple datacentres and they may use thousands of computers. In those environments network partitions may arise. According to the CAP theorem [8, 10], when a network partition happens, there is a tradeoff between strong consistency and service availability. Since elastic services must guarantee availability in order to comply with their QoS requirements, consistency needs to be relaxed in those situations. This is another reason for the success of eventually consistent services.

There have been several recent surveys or tutorials on eventual consistency [22, 4, 5]. In spite of this, up to our knowledge, no recent work has identified the borderline between inherently convergent consistency models and potentially divergent models. The goal of this report is to settle that frontier.

2 System Model

We assume a system composed of many processes. All those processes share a set of replicated variables. When one of the processes writes a new value on any of the variables, a consistency protocol propagates that new value to every other process. That consistency protocol is able to implement a given consistency model using that propagation-based principle. Although there are other implementation approaches (e.g.,
using an invalidation approach [1]), the propagation-based technique is general enough for supporting all consistency models to be discussed in this report.

Some executions will be presented in the following sections. Those executions consist in a sequence of read or write actions. The first character in each action states the type of action (“R” for reads and “W” for writes), the second character is the process identifier. The argument of the action is the variable on which the action is applied. Finally, the last character expresses the value being written in a write action or returned in a read action. So, W1(x)2 means that process P1 has written value 2 onto variable “x”. Read actions express the point at which a given process has received the effect of a previous write action generated at another process. For instance, R2(x)2 means that P2 has received value 2 for variable “x” (i.e., the result of the write action we had described). Therefore, the executions being presented in the next sections are only concerned on the write propagation events being used by the consistency protocol. Once a value is received by a target process, that process may locally read it as many times as needed until a new value is locally written on the same variable or a subsequent propagation of another write on that variable is received by that same process. However, those local read actions are unimportant for our discussion.

3 Setting the Borderline

Distributed shared memory (DSM) consistency models [9, 19] regularly assume that multiple processors (and processes) share a part of the main memory. Those classical memory models are known nowadays as data-centric or server-centric consistency models [24].

Some recent research works (e.g., [20, 7]) consider that state convergence is only a liveness property. Because of this, the requirements stated in many definitions of eventual consistency [22, 25] could be achieved adding a convergence property to a relaxed consistency model. Unfortunately, there is no agreement on where to place the borderline between strong and relaxed consistency models. Depending on the problem being considered, that frontier may be varied.

In the scope of eventual consistency characterisation, we believe that two frontiers exist, separating these three sets:

- **Strong models**: A strong model is one where state convergence is reached among replicas on each write action. This means that no read old-new inversion will be tolerated. According to [3] the linearisable [12] (or atomic [16]) model is in this strong group.

- **Convergent models**: We consider that a model is inherently convergent when it will be able to ensure that once the effects of each write have been propagated to the remaining processes, if no new write action is received in a sufficiently long interval, then the consistency model will ensure (without any other external mechanism) that all replicas have the same state.

The sequential [15] consistency model is convergent but not strong. For instance, the following execution complies with the requirements of the sequential model, but different readers have been able to read different values between two consecutive writes:

\[ W2(x)3, W1(x)2, W1(y)5, R3(x)3, R3(x)2, R3(y)5, R2(x)2, R2(y)5 \]  

(1)

The execution is sequentially consistent, since all processes have seen the same sequence of actions (W(x)3, W(x)2 and W(y)5) and such sequence is consistent with the writing order on each writer (in this case, P1 wrote value 2 on “x” before writing 5 on “y”). However, each process has seen the events of that sequence at different times. If we assume that the initial values of both variables are 0, then after the fifth event of that execution, each process holds the following values:

- P1: x=2, y=5.
- P2: x=3, y=0.
- P3: x=2, y=0.
So, their states do not converge yet. Indeed, they do not converge again until all the events in the execution have been considered. At that moment, all processes have got \( x=2 \) and \( y=5 \).

Additionally, if we considered that P1, P2, and P3 are replicas of a given service, it might happen that a given client has read (between the second and third actions) variable “x” from P1, receiving value 2 and later on (e.g., between the third and sixth actions) would read again “x” but from P2, obtaining at that moment value 3, that is older than 2. So, sequential consistency allows read old-new inversions. Therefore, this example shows that the sequential model is convergent but not strong (i.e., not enough strong in the sense stated in this section).

- **Relaxed models**: A model is relaxed if it does not ensure convergence when all its consistency requirements are respected. PRAM (also known as FIFO) consistency is an example of relaxed model since it only requires that the writes of each process are applied in writing order on the other replicas, allowing any interleaving of the writes made by different processes. Because of this, different receivers may see different values on a given set of variables when they have applied all their incoming updates.

For instance, this other FIFO-consistent execution considers the same three write actions and processes shown in execution 1. However, now the states of those processes do not converge:

\[
W_2(x)3, W_1(x)2, W_1(y)5, R_3(x)2, R_3(y)5, R_2(x)2, R_2(y)5, R_3(x)3, R_1(x)3
\]  

Note that the final state is \( x=2, y=5 \) in P2 but \( x=3, y=5 \) in P1 and P3.

In order to set the borderline between relaxed and convergent models, we need to revise the consistency properties of all non-strong models. Steinke and Nutt [23] provide an appropriate composable specification of those properties. They are informally summarised as follows:

- **GPO (Global Process Order)**: There is a global agreement on the order of writes at each processor. Writes from different processors may be freely interleaved by each reader.

- **GDO (Global Data Order)**: There is global agreement on the order of writes on each variable.

- **GWO (Global Write-read-write Order)**: There is a global agreement on the order of potentially causally-related writes; i.e., write A globally precedes write B when the value written in A had been read by process \( p \) before it wrote B.

- **GAO (Global Anti Order)**: There is a global agreement on the order of any two writes when a process can prove that it read one before the other.


GDO is convergent. It states that all processes agree on the order of writes onto each variable. Therefore, when all the updates have been propagated to every replica, the same value on each variable should be seen in every replica. Note that all those processes agree on a single order of writes on each variable and all write actions have been propagated to every other node. Therefore, all they have the same value per variable. If GDO is convergent, then GAO is also convergent, since GAO is stronger than GDO.

On the other hand, GPO is not convergent. Execution 2 has provided a counter-example for convergence in FIFO executions and GPO is equivalent to FIFO consistency. Additionally, GWO is neither convergent. Execution 2 trivially complies with GWO and it is not convergent.

Therefore, to be convergent only depends on complying with the GDO property. This means that non-strong consistency models may be classified as follows:

- **Convergent**: sequential (GPO+GWO+GAO), processor (GPO+GDO) and cache (GDO).

- **Relaxed**: causal (GPO+GWO) and FIFO (GPO).
Figure 1: Convergent and relaxed models.

Figure 1 graphically shows this classification. Each box maintains a combination of properties. Those properties are shown in the top half of the box, while the bottom half shows the name of the resulting consistency model, if that model is named. The basic properties are in the bottom of the figure. Combinations of properties are shown in layers above those taken as their base. The topmost layer corresponds to the sequential consistency model, which is the strongest one that may be built using those consistency properties.

With this, implementations of eventual consistency should choose between two alternatives: (1) to implement a replication protocol supporting a convergent consistency model, or (2) to support a relaxed model and extend its protocol with some data convergence mechanism when data divergences arise.

In the regular case, eventual consistency has been implemented using multi-master replication protocols with lazy update propagation [13, 14] and, in most cases, eventually consistent services are able to tolerate network partitions remaining available. With that kind of replication, convergent models cannot be supported. Note that in a multi-master algorithm multiple processes may concurrently write different values on the same variable (perhaps, in disjoint subgroups of a partition), propagating such values lazily to the other processes. It is impossible to reach an agreement on a common order for the writes applied on each variable (as GDO requires), since every process, besides writing, is also reading the values from other variables. In the end, any possible agreed write order would have been violated by those concurrent reads. Therefore, it is mandatory to take as a base a relaxed model and complement it with some mechanism that fixes the data divergences that might occur. Additionally, we should also consider that other papers [18, 21] have proved that the causal model is the strongest one to be supported for sharing in a consistent and available way a general data resource in a partitionable network.

Thus, depending on the regular consistency requirements of the service to be implemented following an eventual convergence principle, two real alternatives exist for data-centric consistency: (1) to use PRAM replica consistency when optimal throughput is the main goal, or (2) to use causal replica consistency when such a causal behaviour is expected, sacrificing performance in this case.

4 Summary

Eventual consistency is a liveness property (data convergence) to be added to relaxed memory models. Due to this, it cannot have a formal specification similar to other regular consistency conditions since none of
them considers time. However, there is a recently proposed formalisation [6] that carefully characterises safety (programme correctness) and liveness (eventual state convergence, even when there are no quiescent intervals in a trace) correctness conditions for eventually consistent services.

Using the consistency model specification framework provided in [23], the border between inherently convergent and relaxed models has been set. Those relaxed models (e.g., slow, PRAM and causal) may be taken as a basis for implementing eventually consistent services. This shows that eventual consistency is quite a relaxed condition.

References


