An Implementation Model for Cloud Storage Systems

Vinh Tao
Scality and Inria-LIP6
vinh.tao@lip6.fr

Vianney Rancurel
Scality
vianney.rancurel@scality.com

João Neto
KTH and Inria-LIP6
joaon@kth.se

Abstract

Cloud storage systems, which essentially are eventually consistent distributed file systems, and their automatic resolution for conflict updates have been well studied [2]. However, it remains challenging to make the resolution to work correctly with more than two replicas in real-world distributed file systems. In this research, we target the implementation model for such file systems.

File System We model a file system as a partially ordered set (poset) in which an element could be a directory, a file, or an inode. These elements are identified by their unique absolute path for directories and files or by their unique inode number for inodes. The binary relation between a directory and its children is *one-to-many* while the binary relation between a directory or a file and their inode is *one-to-one* or *many-to-one*, respectively. Because of the *one-to-one* property of their binary relation, we can consider a pair of a directory and its inode as a single element in the system model without impacting the correctness of the model.

Divergence and Reconciliation Different replicas of a file system may differ in their poset structures and/or in the content of file inodes due to their local updates. All of the updated elements of a replica in a period of time form a so-called *delta poset* which is a subset of the state of that replica at that moment. A pair of diverged replicas is reconciled by exchanging their delta posets. A replica would merge its own delta poset with that of the other replica to compute a merged delta poset that when applying on a replica would converge a replica's state with the other. This process is defined as:

$$\begin{cases} \delta = \delta_A \sqcup \delta_B \\ \delta + S_A = \delta + S_B \end{cases} \quad (1)$$

where S_A and S_B are the diverged states of replicas A and B respectively, δ_A, δ_B are their delta posets, $\sqcup$ is the merge operation that computes the least-upper-bound of a pair of posets, and $+$ is the poset ordered sum operation.

Merging a pair of delta posets is done by simply computing their union. Merging elements appear in either poset overwrites the other; merging those appear in both posets may cause conflict that requires conflict resolution, which in turn may create new unique elements. A full description of our conflict resolution can be found in another work [2]. Consider the new elements created by the conflict resolution also exist with the state of *deleted* in the delta poset of each replica, merging these delta posets becomes computing their union—operation that satisfies the requirement of computing the least-upper-bound of the delta posets by definition.

How do the replicas converge? Consider all replicas start with the same file system represented by a poset S, each of the replicas then updates its local file system to generate an updated file system S'_i that $S'_i = \delta_i + S$. Merging any pair of replicas results in $\delta = \bigsqcup \delta_i$ following (1). Then applying δ to each local state is:

$$S''_i = \delta + S'_i = \delta + \delta_i + S. \quad (2)$$

with S''_i is the final state of a replica i. Because $\delta_i \subseteq \delta$ thus $\delta + \delta_i = \delta$. Applying to (2) we have $S''_i = \delta + S$ for any i, which means the replicas converge after exchanging updates.

Reconciling Multiple Diverged Replicas Merging multiple replicas is a pairwise process in which each pair of replicas are merged together; the results from pairwise merging are then combined together to produce the final outcome.

The pairwise merging process has some important properties in converging diverged replicas. First, this process is applied on all combinations of the replicas, it is thus able to identify all cases of conflict between them. And second, as merging uses the union operation, the merge function is *idempotent*, *commutative* and *associative*—these are the properties for the merge function to converge replicas toward their least-upper-bound poset. Systems with the merging function that has these properties are also known in literature as instances of CRDT (Conflict-Free Replicated Data Type) [1], data types which ensure the eventual consistency of their replicas without coordination.

References

[1] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. In *Stabilization, Safety, and Security of Distributed Systems*, pages 386–400. Springer, 2011.

[2] V. Tao, M. Shapiro, and V. Rancurel. Merging Semantics for Conflict Updates in Geo-Distributed File Systems. In *Proceedings of the 8th ACM International Systems and Storage Conference*, Systor '15, New York, NY, USA, 2015. ACM.