Paving the way towards a practical integration of CRDTs in IPFS

Quentin Acher[1] Claudia-Lavinia Ignat[1] Shadi Ibrahim [2]
quentin.acher@inria.fr claudia.ignat@inria.fr shadi.ibrahim@inria.fr

[1]Inria, Université de Lorraine, CNRS, LORIA, France
[2]Inria, Univ. Rennes, CNRS, IRISA, France

May 1, 2024

The continuous growth in data volume increases the interest in using peer-to-peer systems not only to store immutable data, but also to store and share mutable data which are updated and modified by multiple users. Usually, data are replicated across different machines/peers to ensure high data availability and avoid data loss, especially in the presence of failures and churns. However, this raises an important question: how to keep the data consistent across all the replicas.

InterPlanetary File System (IPFS) is an open-source content-addressable peer-to-peer system that provides large scale distributed data storage and delivery. However, IPFS is designed to share immutable data, thus modifying data results in a new copy. This may lead to a high cost in terms of storage, data transfer, and latency. For example, in collaborative applications like text editing, a new version of the whole document will be generated and replicated when one single letter is written or deleted.

Conflict-free Replicated Data Types (CRDTs) are specific data types built in a way that mutable data can be managed without the need for consensus-based concurrency control. They are a promising solution for merging modifications on mutable data in IPFS. Previous efforts have explored how to implement such CRDTs in IPFS and discussed the limitations and several potential optimizations, but they have not been implemented and evaluated in real IPFS deployment.

In this work-in-progress, we conduct experiments to show the deficiencies of IPFS when dealing with mutable data; and evaluate the convergence time, traffic usage and scalability of our implementations of CRDT in IPFS for two simple data types (i.e., Counter and Set). Our implementations are inspired by previous work and written in Go. They utilize libP2P as a communication system and use a rendez-vous point to connect peers together.