A Consensus Quorum Algorithm for Replicated NoSQL Data

Tadeusz Pankowski

Institute of Control and Information Engineering, Poznań University of Technology, Poland

tadeusz.pankowski@put.poznan.pl

Abstract. We propose an algorithm, called Lorq, for managing NoSQL data replication. Lorq is based on consensus quorum approach and is focused on replicating logs storing update operations. Read operations can be performed on different levels of consistency (from strong to eventual consistency), realizing so-called service level agreements (SLA). In this way the trade-off among availability/latency, partition tolerance and consistency is considered. We discuss correctness of Lorq and its importance in developing modern information systems based on geo-replication and cloud computing.

1 Introduction

New classes of information systems such as Web services, Web search [8,7], e-commerce [9], and social networks [15], demand data stores using paradigms different from those of centralized conventional databases. They need more scalable distributed solutions for managing huge data repositories based on NoSQL key-value data models [6,4], and guaranteeing very high availability and low latency [9,15,19]. An important way in which such data stores differ from conventional databases is that they have to deal with the trade-off between consistency, availability/latency and partition tolerance. This issue was formulated by so-called CAP theorem [5]. The theorem states that these three features cannot be achieved simultaneously, and a lot of research have been done to propose solutions considering this trade-off [1,12,14].

In this paper, we propose a new data replication algorithm called Lorq (Log Replication based on Quorum consensus). The algorithm is based on a consensus quorum approach [11,16], and on eventual consistency [21,20,13], which is a weak variant of strong consistency. In Lorq, the replication is realized through replication of logs (storing update operations) instead of data, like in Raft [18] managing data in a replicated state machine. To solve the trade-off between consistency and high availability, read operations in Lorq may be performed using different levels of guaranteed consistency – from strong consistency to one of four kinds of eventual consistency.

The outline of this paper is as follows. The next sections reviews ideas of consensus quorum algorithms. Section 3 presents the Lorq algorithm and some ways for achieving strong and eventual consistency in Lorq. The correctness of Lorq is discussed in section 4. Finally, section 5 concludes the paper.
2 Basic Idea of Quorum Consensus Algorithms

In a quorum-based data replication, it is required that an execution of an operation (i.e., a propagation of an update operation or a read operation) is committed if and only if a sufficiently large number of servers acknowledge the successful termination of this operation [11]. Let us denote: \( N \) – a number of servers storing copies of data (replicas); \( R \) – an integer called read quorum, meaning that at least \( R \) copies were successfully read; \( W \) – an integer called write quorum, meaning that propagations to at least \( W \) servers have been successfully terminated. The following relationships hold between \( N \), \( R \) and \( W \):

\[
W > N/2, \quad (1) \\
R + W > N. \quad (2)
\]

To commit a read operation, a server must collect the read quorum, and to commit a write operation must collect the write quorum. Condition (1) guarantees that the majority of copies is updated, and (2) that among read copies at least one is up-to-date.

The aim of consensus algorithms is to allow a collection of servers to process users’ commands (updates and reads) as long as the number of active servers is not less than \( \max\{W, R\} \). It means that the system is able to survive failures of some of its servers.

3 Lorq – Log Replication Based on Quorum Consensus Algorithm

During last decade, the research on consensus algorithms is dominated by Paxos algorithms [10,16,17]. Lately, a variant of Paxos, named Raft [18], was presented as a consensus algorithm for managing a replicated log. Lorq is based on ideas underlying Paxos and Raft, and includes such steps as: (1) leader election; (2) log replication, execution and commitment of update operations; (3) realization of read operations on different consistency levels.

3.1 Architecture

The architecture of Lorq (Fig. 1), like in the case of Raft [18], is organized having in mind: operations, clients, and servers occurring in the system managing data replication.

Operations. We distinguish three update operations: set, insert, and delete, and one read operation. In order to informally define syntax and semantics of operations, we assume that there is a NoSQL database \( DB = \{(x, \{ A : a\})\} \) storing a key-value data object \( (x, \{ A : a\}) \). Additionally, any data object in \( DB \) has a timestamp of the last operation updating this object. Operations are specified as follows: