Fast Paxos Made Easy:
Theory and Implementation

Wenbing Zhao, Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, OH, USA

ABSTRACT

Distributed consensus is one of the most important building blocks for distributed systems. Fast Paxos is one of the latest variants of the Paxos algorithm for distributed consensus. Fast Paxos allows an acceptor to cast a vote for a value of its choice unilaterally in a fast round, thereby eliminating a communication step for reaching consensus. As a tradeoff, the coordinator must build a quorum that is bigger than the simple majority used in Classic Paxos. This article presents the theory, implementation, and a comprehensive performance evaluation of the Fast Paxos algorithm. The theory is described in an easier-to-understand way compared with the original article by Lamport. In particular, an easy-to-implement value selection rule for the coordinator is derived. In the implementation of Fast Paxos for state-machine replication, a number of additional mechanisms are developed to cope with practical scenarios. Furthermore, the experiments reveal that Fast Paxos is most appropriate for use in a single-client configuration. The presence of two or more concurrent clients even in a local area network would incur frequent collisions, which would reduce the system throughput and increase the mean response time as experienced by clients. Due to frequent collisions, Fast Paxos actually performs worse than Classic Paxos in the presence of moderate to large number of concurrent clients.

Keywords: *Distributed Consensus, End-to-End Latency, Fault Tolerance, Probability Density Function, Quorum Requirements, State-Machine Replication, System Throughput*

INTRODUCTION

Distributed consensus is one of the most important building blocks for distributed systems (Zhao, 2014). For example, it is impossible to build a highly available cloud service without using some distributed consensus algorithm to ensure that all replicas remain consistent (Camargos, Madeira, & Pedone 2006; Camargos, Schmidt, & Pedone, 2008; Zhao, Melliar-Smith, & Moser, 2010; Zhao, 2010). Fast Paxos (Lamport, 2006) is one of the latest variants of the original Paxos algorithm (Lamport, 2001) (referred to as Classic Paxos) for distributed consensus. Classic Paxos is a good fit for state-machine replication and it has been used in a number of practical fault tolerant systems (Bolosky et al., 2011; Burrows, 2006; Hunt et al., 2010; Mao et al., 2008; Rao, Shekita, & Tata, 2011). Fast Paxos aims to further reduce the latency for reaching consensus by using a larger quorum size. Similar to Classic Paxos, Fast Paxos operates in rounds and there are two phases in each round. If a consensus is not reached within a round, a new round will be launched for liveness. In Fast Paxos, there can

DOI: 10.4018/ijdst.2015010102

be two different types of rounds: fast rounds and classic rounds. A classic round would operate the same way as a round in Classic Paxos except that the value selection rule at the coordinator is different, as to be explained in later sections. In the original article published by Lamport (Lamport, 2006), the quorum requirement as well as the value selection rule depend on the evaluation of the following observation known as O4(v) in (Lamport, 2006):

A value has been or might yet be chosen in round k only if there exists a k-quorum R such that vr(a) = k and vv(a) = v for every acceptor a in RTQ. (Lamport, 2006)

Here Q refers to the quorum formed for the current round, k is the most recent round number in which an acceptor a has casted a vote, k-quorum means the quorum used in round k, vr(a) refers to the round number in which the acceptor a has casted a vote, and vv(a) refers to the value contained in that vote. O4(v) is true if and only if the above observation is true for some round k for the value v.

As we can see, to evaluate this observation, one must examine every previous round k, and determine whether or not a k-quorum exists for round k that satisfies the specific constraint on v. This implies that for the coordinator to evaluate whether or not a value v satisfies O4(v), it must collect votes from every acceptor of the system in every round, which is simply not practical in asynchronous environment.

In this article, we introduce a more implementation-friendly value selection rule for the coordinator, and provide a more intuitive reasoning on the quorum requirements, both without the need to evaluate O4(v). To demonstrate the practicality of the proposed value selection rule, we present an implementation of Fast Paxos for state-machine replication. We show that many additional mechanisms are needed to cope with practical scenarios. Furthermore, we have conducted a comprehensive evaluation of Fast Paxos using our research prototype. Our experiments reveal that Fast Paxos is most appropriate for use in a single-client configuration. The presence of two or more concurrent clients even in a local area network would incur frequent collisions, which would reduce the system throughput and increase the mean response time as experienced by clients. Due to frequent collisions, Fast Paxos actually performs worse than Classic Paxos in the presence of moderate to large number of concurrent clients.

The remaining of the article is organized as follows. Section 2 describes the system model used in Fast Paxos as well as Classic Paxos and their variants. Section 3 defines the safety and liveness requirements for distributed consensus solutions. Sections 4 and 5 introduce Classic Paxos and its application in state-machine replication (referred to as Multi-Paxos) as the foundation for Fast Paxos. In Section 6, we describe Fast Paxos and our theoretical contributions. In Section 7 and Section 8, we report the details of our implementation and performance evaluation of Fast Paxos. We conclude the article with the final two sections on related work and concluding remarks.

SYSTEM MODEL

We consider a distributed system with a number of processes, and any one of them may propose a value. A process may participate in a distributed consensus algorithm (such as Classic Paxos and Fast Paxos) in one of three roles: (1) proposer, (2) acceptor, or (3) learner. A proposer is one that proposes values to be chosen and learned by others. An acceptor participates in agreement negotiation on the values proposed. A learner is one that learns the value that has been chosen. Note that the roles are logical and a process can assume multiple roles, e.g., a process may act both as a proposer and an acceptor.

In a client-server system, the clients send their requests to the server for processing and expect to receive the corresponding replies. When state-machine replication of the server is used, it is essential to ensure that all replicas deliver and execute the requests in the same total order. For each request, its total order