Composable Data Consistency Policies

Nosheen Zaza

Abstract

Nothing here yet

Research Advisor
Prof. Nate Nystrom

Academic Advisor	Review Committee
Prof. Nate Nystrom	Prof. Committee Member1, Prof. Committee Member2

Research Advisor's approval (Prof. Nate Nystrom):   Date: ....

PhD Director's approval (Prof. Stefan Wolf):   Date: ....

1 Introduction and Problem Domain

1.1 Replication and Partitioning

Data replication and partitioning appear in hardware and software systems at every scale. Consider, on a small scale, the replication of data among caches and main memory, or partitioning program data among objects in objects oriented systems, to be later composed and used as logically meaningful modules. Replicating data across web servers, or partitioning data across tables in a database are some larger scale examples. Both replication and partitioning introduce several challenges to system designers and programmers. A major challenge is the problem of data consistency: that an update to a replica or partition will be propagated correctly to clients of other replicas or partitions.

1.2 Correctness of Concurrent Operations

The notion of correctness means different things in different contexts, depending on different requirements. Traditionally (mostly before the internet era), correctness meant that systems behave as if all operations were performed serially at one replica or across all partitions, meaning that operations form a total order. examples of consistency models that possess this quality are one-copy serializability [Bernstein et al. 1987]. sequential consistency[Leslie some year] or linearizability[wing some year]. Enforcing a total order on operations occurring within a large distributed system, with many replicas can prove expensive, as well as unnecessary for many applications. In this era of massive, distributed online applications, Weaker models, such as eventual consistency [Sheth et al. 1991] or session consistency[Ref], are the norm not the exception. In fact, common issues in distributed systems, such as loss of data during communication or inavailability of processing nodes makes it challenging to enforce even those weaker models. Weak consistency models can be found on a much smaller scale, in the absence of replication as well. Class ConcurrentSkipListMap[ref], a part of Java standard library, contains execution paths that do not correspond to any sequential execution, because some carefully characterized operations that lost races are not retired, the consistency of instances is enforced by the semantics of the class.

1.3 Enforcing and Strengthening Consistency

Enforcing data consistency requires coordinating update requests, and determining when those updates become visible to clients. There are several mechanisms for achieving this, depending on the required consistency and performance properties such as hardware level CAS operations, explicit synchronization through locking, transactionsâĂę etc. I want to say here that all these mechanisms must determine what to do on failure of operation, specify the retry policy