Università della Svizzera italiana

Faculty of Informatics

Research Prospectus

May 26, 2014

Composable Data Consistency Policies

Nosheen Zaza

Abstract

The importance of relaxed consistency models has become evident, with an increasing variety of systems trading off strong consistency for availability and performance, such as eventually consistent replicated key-value stores, or algorithms exhibiting benign data races. Developing such systems is a challenging, because consistency requirements may vary across system components or for different system clients; even when uniform consistency is required, the underlying environment often provide weaker or stronger consistency than desired. Thus, developers have to encode complex interaction patterns among the environment and various system components to obtain the desired consistency properties for each component, without compromising consistency, performance or availability of other components.

While ongoing research has produced several methodologies, tools, and programming models that assist developers reason about, express and check consistency policies, most work is polarized either around relaxed consistency models, as employed in distributed systems, or around strict models described in shared-memory concurrency literature. Hence, we do not have a unified theory to describe diverse, commonly used consistency policies on different scales, neither we have comprehensive semantics to describe how various consistency models interact or compose. In this research prospectus, we overview classic and recent work on managing consistency, and motivate the need for further research. We believe that studying consistency as a single, universal property of applications will pave the way to creating elegant, generic frameworks and programming languages to express a wide variety of consistency management patterns, in both distributed systems and multicore applications. We also believe that enabling programmers to accurately and precisely define consistency policies can lead to better safety and performance properties.

Research Advisor
Prof. Nate Nystrom

Academic Advisor
Prof. Nate Nystrom

Review Committee
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