Exploring the consistency problem space

Nishith Krishna, Marc Shapiro, Karthikeyan Bhargavan

Brief Announcement: Exploring the Consistency Problem Space

Nishith Krishna
Courant Institute of Mathematical Sciences
New York University
New York, USA
nishith@cs.nyu.edu

Marc Shapiro
INRIA Rocquencourt, France,
LIP6 Paris, France, and
Microsoft Research
Cambridge, UK
marc.shapiro@acm.org

Karthikeyan Bhargavan
Microsoft Research
Cambridge, UK
karthb@microsoft.com

ABSTRACT

We study formally the consistency problem, for replicated shared data, in the Action-Constraint framework (ACF). ACF can describe a large range of application semantics and replication protocols, including optimistic and/or partial replication. ACF is used to decompose the consistency problem into simpler sub-problems. Each is easily understood. Existing algorithms from the literature can be explained as combinations of concrete sub-problem implementations. Using ACF, we design a new serialisation algorithm that does not cause aborts and only needs pairwise agreement (not global consensus).

Categories and Subject Descriptors: C.2 Computer-Communication Networks

General Terms: Algorithms, Theory.

Keywords: Replicated data, consistency, strong consistency, weak consistency, semantic consistency, pessimistic replication, optimistic replication, partial replication.

1. ACTION-CONSTRAINT FRAMEWORK

Replication protocols incorporate complex trade-offs affecting semantic correctness, complexity, message costs, lost updates, partial replication, etc. We study them formally in *Action-Constraint* framework (henceforth ACF) [**?**]. ACF enables a formal study of the full range of replication protocols, whether pessimistic optimistic protocols, and whether using full or partial replication.

The primitives of ACF are opaque application *actions* related by *constraints* i.e., scheduling invariants. Constraint types are $\rightarrow$ (Before), implication $\lhd$ (MustHave) and non-commutativity ν (NonCommuting). For instance $\alpha \rightarrow \beta \wedge \alpha \lhd \beta$ means that β depends causally on α. A site's local knowledge of actions and constraints is called its *multilog*; sites exchange multilogs with asynchronous messages. Given sufficient constraints, an action becomes *guaranteed* (executes in all schedules) or *dead* (executes in no schedules), *serialised* (ordered w.r.t. all non-dead actions that do not commute with it), *decided* (either dead, or both guaranteed and serialised). An action is *stable* if, either it is dead, or it is both guaranteed and serialised and all its $\rightarrow$ predecessors are stable.

The correctness conditions for consistency are: Mergeability (a safety condition), i.e., in any arbitrary distributed snapshot, no action is both dead and guaranteed, and Eventual Decision (liveness), i.e., every action is eventually decided.

PODC'05, 17–20 July 2005, Las Vegas, Nevada, USA.
ACM 1-58113-994-2/05/0007.

2. DECOMPOSING CONSISTENCY

Given the above specification, we subdivide the consistency problem into three simple graph sub-problems. *Conflict Breaking* considers the $\rightarrow$ sub-graph. Any algorithm that deletes (makes dead) enough nodes to ensure this sub-graph is acyclic is correct. Computing an optimal solution (minimising dead nodes) is NP-hard. Replication protocols from the literature use a spectrum of heuristics. In the full paper, we compare some by simulation. Generally, the cheap, decentralised, asynchronous heuristics are poor compared to the optimal; high-quality solutions are expensive and/or synchronous.

Agreement propagates dead status along the $\lhd$ sub-graph.

Serialisation considers the combined $\rightarrow/\nu$ graph. Any algorithm that orders all ν edges is correct. Note that serialisation might create new $\rightarrow$ cycles. Protocols from the literature vary across a wide spectrum. The full paper compares a number of combinations of conflict-breaking and serialisation algorithms. Again, cheap, decentralised, asynchronous heuristics have poor quality. The best algorithms do not introduce new cycles, but these are either completely local and deterministic (hence inflexible), or are consensus-based.

In a distributed system, sub-algorithms may execute in parallel, and parallel instances of each may exist. However a distributed termination protocol is necessary.

By examining the structure of the ACF graph, we propose a new serialisation algorithm that does not introduce cycles. The algorithm waits for the predecessors of an action to become stable; then, adding an outgoing $\rightarrow$ edge cannot create a cycle. To avoid races, serialising a single ν edge is atomic. Interestingly, there is no need for a global order, only pair-wise agreement between the two ν actions. This serialisation algorithm never creates $\rightarrow$ cycles, hence never causes an action to become dead.

3. REFERENCES

[1] Marc Shapiro, Karthikeyan Bhargavan, and Nishith Krishna. A constraint-based formalism for consistency in replicated systems. In *Proc. 8th Int. Conf. on Principles of Dist. Sys. (OPODIS)*, Grenoble, France, December 2004.