Introducing Permuted States for Analyzing Conflict Rates in Optimistic Replication

An-I A. Wang, Florida State University
Peter Reiher, University of California, Los Angeles
Geoff Kuenning, Universität Karlsruhe and Harvey Mudd College

Categories and Subject Descriptors
D.4.3 [**File Systems Management**]: Distributed file systems. D.4.8 [**Performance**]: Modeling and prediction; simulation.

General Terms
Performance and measurement

Keywords
Optimistic replication, analytical modeling, simulation, permuted states, conflict rates

1. INTRODUCTION
Optimistic replication is a technique that allows read/write access to any available replica of a data item, even during network outages. In optimistic systems, data synchronization, or ***reconciliation***, guarantees convergence and the correctness of data in the case of improper concurrent modifications, or ***conflicts*** [3, 4]. Typically, reconciliation involves two replicas, with bidirectional data propagation. Conflicts are detected only at reconciliation time, when both replicas have been updated since the previous reconciliation. The *conflict rate* is a very important metric when evaluating an optimistic replication system, since it reflects the level of data consistency and the efforts involved in achieving such consistency.

Although optimistic replication is widely deployed [1, 3, 4, 5], an analytical understanding of conflicts in these systems is limited to two replicas, due to the exponential growth of the state space. Even for 3 replicas, we need 64 states to track all pairwise, update relationships. The base case of an important class of internal system conflicts, ***identical conflicts***, requires 4 replicas for analysis, or 4,096 states. Therefore, the current understanding of conflicts largely relies on simulations, validated with only two replicas. Without validation, simulation errors can easily go undetected for higher replication factors.

2. PERMUTED STATES
To summarize the state space efficiently, we apply combinatorics. We use an event-based model in which time is measured in terms of updates and reconciliations.

Figure 1 illustrates the two-replica case, with λ as the probability of having an update at either replica, and μ as the probability of having a pair-wise reconciliation process, respectively, as the next system event. We use a Poisson interarrival model. At each state, the outbound update and reconciliation probabilities sum to λ and μ, respectively. The sum of outbound λ and μ at each state is 1.

This analysis assumes uniform λ and μ across all replicas, to make the analysis tractable. However, the resulting model can be used to validate simulations that explore non-uniform transition probabilities.

SIGMETRICS'05, June 6–10, 2005, Banff, Alberta, Canada.
ACM 1-59593-022-1/05/0006.

Each replica is represented by a dot. In the starting state (shaded), two replicas are identical, represented via a horizontal line connecting the two. If reconciliation occurs, the replicas remain identical, so the starting state transitions back to itself.

If one of the replicas is updated, we move to the middle state, where the update-receiving replica ***dominates*** the ***subordinate*** one. This relationship is represented by a non-horizontal line, where the upper replica dominates the lower one. Regardless of which replica is updated, we can only transition from the starting state to the middle state. By decoupling the system state from the labeling of individual replicas, each state captures all *isomorphic* states resulted from permuting the replica identifications. We call this representation ***permuted states***.

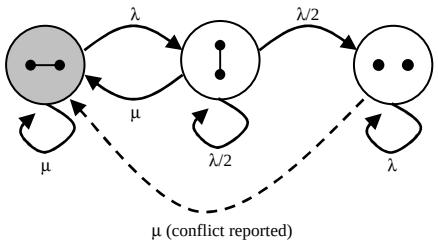


Figure 1: The state-transition diagram for two replicas.

If a dominating replica reconciles with its subordinate, it will replace the content of the latter with its own, and then both will be marked as identical (back to the starting state). An update to the dominating replica will not change its dominance over the subordinate. An update to the subordinate replica breaks the dominance relationship, and the system enters the rightmost state (conflict).

Conflicting replicas (dots) are not connected by lines. An update to either conflicting replica leaves both in conflict. However, a reconciliation between the two leads to identical replicas (the starting state or the convergence state), with a reported conflict.

A system can be in a state with conflicting replicas without reporting conflicts, since conflicts are detected at reconciliation time. Therefore, the conflict rate, or the probability of having conflicts due to either update or reconciliation, is computed by obtaining the equilibrium probability of a state that contains replicas in conflict, multiplied by the probability of traversing its conflict-resolving transition.

To compute the equilibrium probability based on Figure 1, we can construct a system of linear equations by equating the outbound transition flow at each state with the inbound flow. The sum of probabilities at each state should be 1. We omit the equations and their solutions here; the reader is referred to [7] for details.

3. GENERALIZATION & VALIDATIONS

By applying permuted states, we can reduce the original 64 states for three replicas down to 8 states; 4,096 states for four replicas reduce to 27 permuted states.

For validation, we compare analytical results with those obtained from a simulation [6]. Briefly, each replica keeps a local "version vector" V of update counters for all replicas. A replica i increments its local V_i whenever it performs an update. At reconciliation time, if $\forall\, i,\ V_{replica\_Xi} \geq V_{replica\_Yi}$, X dominates Y. If X dominates Y, and if Y dominates X, X and Y are equal. Otherwise, we have a conflict. A subordinate replica copies the version vector from the dominant one. To merge conflicting version vectors, $\forall\, i,\ V_{replica\_Xi} = V_{replica\_Yi} = max(V_{replica\_Xi}, V_{replica\_Yi})$. The counter of the conflict-resolving replica is incremented by one, indicating that a new version was generated as a result of resolving the conflict. The simulation includes only one replicated item. We follow the methodology presented in [6].

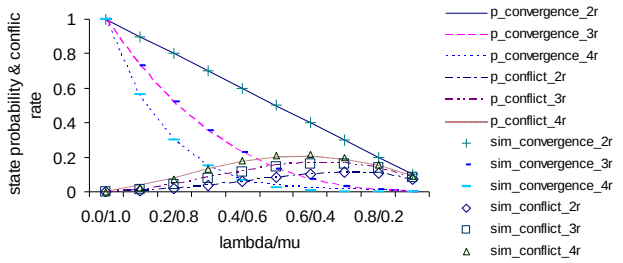
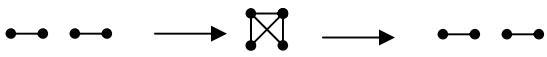


Figure 2: Permuted state models cross-validated with simulations between two and four replicas. Since the 90% confidence intervals are <1% of the mean, we have removed them for clarity of presentation.

Figure 2 shows that simulation models match well with the models based on permuted states, for the exhaustive range of λ/μ. Also, conflict rate can actually be bounded up to 22% for 4 replicas, a valuable insight for systems like OceanStore [2], where the core writable replicas have only a replication factor of four.

4. IDENTICAL CONFLICTS



Event	Replica 1		Replica 2		Replica 3		Replica 4	
	Content	Ver	Content	Ver	Content	Ver	Content	Ver
	X	1000	X	1000	Y	0010	Y	0010
Setup sequence								
Recon(1, 4)	**XY**	**2010**	X	1000	Y	0010	**XY**	**2010**
Recon(2, 3)	XY	2010	**XY**	**1110**	**XY**	**1110**	XY	2010
Identical conflict formation								
Recon(1, 2)	**XY**	**3110**	**XY**	**3110**	XY	2010	XY	1100

Figure 3: Identical conflict formations. The states being reconciled are in boldface.

With permuted states, for the first time, we can enumerate how identical conflicts are formed. The formation of an identical conflict goes through a setup sequence (Figure 3). The system first enters the state with two pairs of identical replicas, replicas 1 and 2, and replicas 3 and 4. A reconciliation process between one replica from each pair (e.g. replicas 1 and 4) will result in a new pair of identical replicas, dominating the replicas 2 and 3, which remain in conflict with each other. When replicas 2 and 3 reconcile, they will create a new version of the data during reconciliation and break away from the dominating replicas 1 and 4.

With two sets of identical pairs after the setup sequence, if replicas 1 and 2 reconcile, we will have an identical conflict based on their version information; however, the content is the same.

5. AUTOMATING PERMUTED STATES

To validate simulations with higher replication factors, we need an automated way to enumerate permuted states. We represent two identical replicas 1 and 2 with $\binom{=,=}{=,=}$. The first row belongs to replica 1, where '=' indicates that replica 1 is identical to itself and replica 2. The second row belongs to replica 2, showing that replica 2 is identical to replica 1 and itself. Two conflicting replicas are represented by $\binom{=,*}{*,=}$, where '*' show that replica 1 is in conflict with replica 2 (vector 1), and replica 2 is in conflict with replica 1 (vector 2). If replica 1 dominates 2, we have the state $\binom{=,>}{<,=}$, where '>' shows that replica 1 dominates 2 (vector 1), and '<' shows that replica 2 is subordinate to 1 (vector 2). By removing isomorphic graphs, we can eliminate redundant states and generate the state transition diagram. We have successfully validated simulations up to 10 replicas with only 10^5 permuted states, as opposed to 10^{33} original states. The ten-replica case covers most common replication deployment scenarios and gives confidence in the accuracy of even larger simulations.

6. CONCLUSIONS

We have described methods to represent and automate permuted states, which have enabled us to use analytical methods to explore the 4-replica base case of identical conflicts, and automate the analytical investigation up to 10 replicas. All results have been confirmed by an independent simulation based on version vectors.

The analysis of problems with exponential state spaces is always challenging. Permuted states are a new technique that makes the analysis of complex replicated systems tractable. As a result, we can characterize and quantify important system behaviors that have previously been unrecognized or poorly understood.

REFERENCES

[1] Daniels D et al. Oracle's Symmetric Replication Technology and Implications for Application Design. *SIGMOD*, 1994.

[2] Kubiatowicz J et al. OceanStore: An Architecture for Global-Scale Persistent Storage, *ASPLOS 2000*, November 2000.

[3] Reiher P et al. Resolving File Conflicts in the Ficus File System. *USENIX*, 1994.

[4] Satyanarayanan M. Coda: A Highly Available File System for a Disconnected Workstation Environment. *2nd Workshop on Workstation Operating Systems*, 1989.

[5] Terry DB et al. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. *SOSP*, 1995.

[6] Wang AIA. A Simulation Evaluation for Optimistically Replicated Filing Environments. Master's Thesis. Computer Science Department, UCLA, 1998.

[7] Wang AIA et al. Using Permuted States and Validated Simulation to Analyze Conflict Rates in Optimistic Replication, submitted to *SPECTS*, 2005.