Using Permuted States and Validated Simulation to Analyze Conflict Rates in Optimistic Replication

An-I Andy Wang
Computer Science Department
Florida State University, Tallahassee, FL, USA
awang@cs.fsu.edu

Geoff Kuenning
Computer Science Department
Harvey Mudd College, Claremont, CA, USA

Peter Reiher
Computer Science Department
University of California, Los Angeles, CA, USA

Optimistic replication provides high data availability in the presence of network outages. Although widely deployed, this relaxed consistency model introduces concurrent updates, whose behavior is poorly understood due to the vast state space.

This paper introduces the notion of permuted states to eliminate system states that are redundant and unreachable, which can constitute the majority of states (4069 out of 4096 for four replicas). With the aid of permuted states, we are for the first time able to construct analytical models beyond the two-replica case. By examining the analysis for 2 to 4 replicas, we can demystify the process of forming identical conflicts—the most common conflict type at high replication factors. Additionally, we have automated and optimized the generation of permuted states, which allows us to explore higher replication factors (up to 10 replicas) using hybrid techniques. It also allows us to validate our results with existing simulations based on actual replication mechanisms, which previously were analytically validated with only one pair of replicas.

Finally, we have discovered that update locality and bimodal access patterns are the primary factors contributing to the formation of identical conflicts.

Keywords: permuted states, optimistic replication, conflict rates

1. Introduction

Optimistic replication is a tool to provide high data availability in the presence of network outages. Multiple users can edit distant copies of the same data simultaneously, even without network connectivity. Data synchronization is achieved through a relaxed consistency model, which guarantees convergence and the correctness of data in the case of improper concurrent modifications or conflicts. Common applications of optimistic replication include document sharing, banking and reservation systems. Coda [1], Lotus Notes [2], Ficus [3], Oracle 7 [4], Bayou [5], Ingres, Microsoft Briefcase, and the Concurrent Version System are well-known research and commercial systems that use optimistic replication.

Although widely deployed, this relaxed consistency model introduces conflicts, whose behavior is not well understood. Empirical and simulation experience has shown evidence that conflicts occur infrequently at the level of aggregate statistics [3, 6, 7]. However, a theoretical result in the database literature suggests that the proliferation of conflicts will prevent optimistic replication from scaling

SIMULATION, Vol. 83, Issue 8, August 2007 551–570
© 2007 The Society for Modeling and Simulation International
DOI: 10.1177/0037549707085073
Figures 4, 6, 12–16 appear in color online: http://sim.sagepub.com

[8]. To our knowledge, the current paper presents the first analytical modeling and understanding of conflict rates beyond two replicas, with analytical results validated by simulations built with actual optimistic mechanisms.

We have overcome a number of challenges to be able to characterize conflict rates analytically, including: (1) capturing the conflict rate in the analytical model; (2) leveraging symmetries, permutations and reachability to reduce the state space; (3) defining appropriate representations for automation and optimization of the state reduction process; and (4) studying the dominant conflict type (identical conflicts) via both analytical methods and simulation.

The following summarizes our major findings. (1) By exploiting the redundancy of states via permutation and removing unreachable states, we can reduce the state space by two to six orders of magnitude for as few as six replicas, greatly simplifying analysis. (2) Conflicts are not directly captured by the system states. Since conflicts are detected when two replicas synchronize, a conflict occurs during a transition between two system states. Therefore, it is entirely possible for a system to be in a state with many conflicting data versions and without conflicts, as long as the system does not synchronize. (3) The temporal and spatial locality of updates interacts with optimistic mechanisms, resulting in significant changes in the number and types of conflicts.

In terms of methodology, our contributions include: (1) a compact system-state representation that eliminates unimportant variation; (2) transition rules that can be used to automate analytical modeling at high replication factors; and (3) a hash-table heuristic for finding isomorphic states in this constrained problem domain.

2. Background

In distributed environments, where component failure is the norm, replication provides high data availability by avoiding centralized, single-point failures. *Optimistic replication* further allows immediate access to any available replica of a data item, even during network outages. The tradeoff is permitting concurrent updates.

In many scenarios, this tradeoff is justifiable. First, for many applications, the majority of concurrent data modifications can proceed in parallel. With proper handling, the modifications can be later merged automatically or manually without data loss. Directories are an important example of this case. Independent file creations can be applied to two replicas of a directory and merged without causing problems [9]. Second, many applications (e.g. library database systems) can still provide meaningful service without immediate propagation of new updates.

Diverging data content requires a *reconciliation* process to bring replicas into synchronization at some convenient time (e.g. when portable computers are temporarily connected to the network). Typically, reconciliation takes place between two replicas. Updates are tracked using either logging [1] or scanning [10].

Conflicts occur when different replicas of the same file are updated after the most recent reconciliation. Optimistic systems often provide extensible application-specific libraries to resolve the majority of conflicting updates automatically [3, 11]. The remaining conflicts require user intervention.

2.1 Definition of Conflicts

Common definitions of *conflicts* fall into three categories. The first is an update that conflicts with existing updates at any replica. This definition assumes oracle knowledge, which is not practical to measure in real systems.

The second definition is oriented toward the log-based reconciliation approach. At reconciliation time, both replicas replay logs of all updates since the last reconciliation between the same replica pair. Whenever two updates for different replicas of the same file are seen in the logs, a conflict is indicated.

The third definition is related to the scanning approach, in which a reconciliation time scan detects updates and resolves conflicts. The difference from the second definition is that multiple updates are collapsed into one and will thus result in the report of only a single conflict. (In practice, most log-based systems optimize out multiple updates to save storage, which also causes conflicts to collapse. Thus, most real systems use the third definition.)

For the remainder of this paper, we will use the third definition. Without loss of generality in our results, we will also assume bidirectional propagation of data at reconciliation time and deterministic resolution of conflicts.

2.2 An Example of Optimistic Replication

Consider document sharing as a simple example. A document can be optimistically replicated into replicas 1 and 2, and two users can concurrently make updates to local replicas. Given that both replicas begin with an identical content version X, four scenarios may occur at the time of the first reconciliation. (1) Neither replica has received an update. The reconciliation process will do nothing. (2) Only replica 1 has been updated and turns the version X into Y. Since replica 2's version X is merely an older version of Y, the reconciliation process will declare that version Y *dominates* X (or version X is *subordinate* to Y), and copy the content of Y over X. (3) Only replica 2 has been updated. This situation is symmetric to the second scenario. (4) Both replicas have received updates. Replica 1 has turned version X into Y, and replica 2 has turned version X into Z. A reconciliation process will declare the two replicas to be in conflict. Versions Y and Z will be merged to create a new version YZ that dominates both version Y and version Z. After the reconciliation, both replicas will have the version YZ.

In this example, a reconciliation process can be trivially automated whenever two users update different parts of a document. If the updates of the two users overlap, the reconciliation process can either follow certain rules (e.g. one user's update is more important than the other) or seek human interventions. Clearly, the usefulness of optimistic replication is dependent on how frequently conflicts occur and what fraction of these conflicts can be resolved automatically.

2.3 Challenges of Analytical Modeling

Analytical modeling is important to understand the behavior of replication systems. In particular, a concise analytical form that can predict whether conflicts can be bounded under the worst scenario is invaluable for resource provisioning. An analytical model is also preferable for validating the correctness of simulations. Otherwise, even a trivial error in a simulation based on a single-point validation (i.e. two replicas) can go undetected and lead to very misleading conclusions for distributed systems.

Analytical modeling of the conflict rate is difficult for three reasons: First, the state space is exponential. By *state*, we mean the global system state. A state captures the relationship between any replica pair so we can determine whether two replicas are the same or in conflict, and whether one replica has a more recent update or an older version of the data. Since conflicts are defined pairwise, two conflicting replicas might not be in conflict with a third. Therefore, each replica needs to track its update and conflict status relative to all other replicas. If each replica needs two states to indicate whether or not it is modified, a pairwise relationship needs four states. For R peer-to-peer replicas with $\frac{R(R-1)}{2}$ pairwise relationships, we need $4^{\left(\frac{R(R-1)}{2}\right)}$ states. Even for three replicas we need 64 states, which is prohibitive to track without automation.

Second, conflict resolution itself may lead to further conflicts, or *metaconflicts*. To illustrate, suppose that initially we have many replicas of three conflicting data versions: X, Y and Z, with each version in conflict with the other two versions. However, pairwise reconciliations may generate the intermediate 'meta-versions', XY, YZ and XZ, resulting in each version being in conflict with three other versions (e.g. XY is in conflict with Z, YZ, and XZ). Effectively, the original three-way conflict has evolved into a four-way conflict due to conflict resolutions. Therefore, the final conflict count is dependent on how data are propagated and resolved, in addition to the initial number of conflicting updates.

Third, a prior simulation study [12] suggests that a single class of metaconflicts – *identical conflicts* – accounts for the majority of conflicts at large replication factors. Unfortunately, the base case for identical conflicts involves four replicas, or 4096 states, making it difficult to characterize their causes using existing analytical methods. To illustrate, suppose users K and L independently

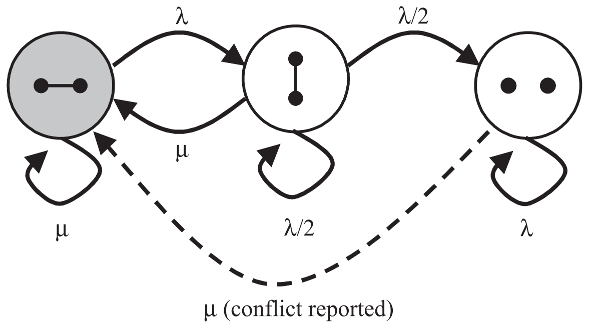


Figure 1. The state-transition diagram for two replicas. The shaded circle is the starting state, where replicas are identical. The rightmost state is the conflict state, where replicas are in conflict. The solid lines mark various transitions among states, and the dashed line marks a conflict-reporting transition. Note that a conflict is not reported until the system transitions back to the starting state as the result of reconciling between two conflicting replicas

update separate replicas. K propagates her updates to user M; L propagates updates to N. When K and L reconcile, they detect a conflict and resolve it by generating an update to create version KL. Similarly, M and N detect a conflict and create MN. Now, when KL and MN reconcile, we have an identical conflict, since the content of KL and MN is the same.

3. Permuted States

How do we visualize the problem so that the number of states is tractable, at least for the four-replica case? Although four replicas may sound small, the analysis for the four-replica case contains three orders of magnitude more states than the two-replica case, which is a giant leap in complexity. As we will see, the four-replica case also introduces critically important behaviors (such as conflict-resolution loops) that are characteristic of much larger systems, and do not appear in smaller cases.

Our solution is to transform the problem and analyze it in the domain of combinatorics. We use an event-based model in which time is measured in terms of 'interesting' system events (updates and reconciliations).

Figure 1 illustrates the system states for two replicas, with λ as the probability of having an update at either replica and μ as the probability of having a pairwise reconciliation process as the next system event. We use a Poisson interarrival model. At each state, the outbound update probabilities sum to λ, and the outbound reconciliation probabilities sum to μ. Finally, the sum of outbound λ and μ at each state is 1.

This analysis assumes uniform update and reconciliation probabilities across all replicas, which is necessary to make the mathematics tractable. However, the resulting model can be used to cross-validate simulations at moderate replication factors. (Previously, simulations were analytically validated for only two replicas.) The validated simulations can then be reconfigured to account for non-uniform access patterns to explore higher replication factors.

Each replica is represented by a dot. In the starting state (shaded), two replicas are identical, represented by 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. Note that regardless of which replica is updated, we are guaranteed to transition from the starting state to the middle state. By decoupling the state of the system from the labeling of individual replicas, each state effectively captures all *isomorphic* system states resulting from permuting the replica identifications. We refer to this type of state representation as *permuted states*.

In the case of reconciliation between a dominating replica and its subordinate, the content of the former will replace that of the latter and then both replicas will be marked as identical (transition back to the starting state). An update to the dominating replica will not change its dominance over the subordinate. However, an update to the subordinate replica breaks its subordinate relationship to its dominating replica, and the system enters the rightmost state (conflict).

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

Note that a system can be in a state with conflicting replicas without reporting conflicts, since conflicts are detected only at reconciliation time. Therefore, the *conflict rate* used in this paper, or the probability of having conflicts due to a system event (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.

3.1 Analysis for Two Replicas: The Base Case

With the state diagram in Figure 1, we can assign probability variables p_0 (leftmost) to p_2 (rightmost) to each state. When the system is in equilibrium, the outbound transition flow at each state should be equal to the inbound flow, resulting in a system of linear equations (1), (2) and (3). Also, the sum of probabilities at each state should be 1 (equation (4)).

$$\lambda p_0 = \mu p_1 + \mu p_2, \tag{1}$$

$$\left(\frac{\lambda}{2} + \mu\right) p_1 = \lambda p_0, \tag{2}$$

$$\mu p_2 = \frac{\lambda}{2} p_1, \tag{3}$$

$$p_0 + p_1 + p_2 = 1, \tag{4}$$

$$p_0 = \frac{\lambda\mu + 2\mu^2}{(\lambda + 2\mu)(\lambda + \mu)}, \tag{5}$$

$$p_1 = \frac{2\lambda\mu}{(\lambda + 2\mu)(\lambda + \mu)}, \tag{6}$$

$$p_2 = \frac{\lambda^2}{(\lambda + 2\mu)(\lambda + \mu)}, \tag{7}$$

$$p_{\text{conflict}} = \mu p_2 = \frac{\lambda^2\mu}{(\lambda + 2\mu)(\lambda + \mu)}. \tag{8}$$

After solving for the probability of the conflicting state (equation (7)), the probability of reporting a conflict $p_{conflict}$ (equation (8)) can be computed by multiplying equation (7) by μ, the probability of taking the transition that resolves conflicting replicas.

As expected, the conflict rate depends on both the update arrival rate and the reconciliation rate, even for this simple two-replica case. Figure 2 shows the percentage contribution of each state as a function of λ/μ, superimposed on the conflict-rate curve. (We will discuss the simulation validation later.) As λ/μ approaches 0, the probability of convergence (p_0) approaches 1 and the probability of being in the conflict state p_2 approaches 0. As λ/μ increases asymptotically, the probability of convergence p_0 approaches 0 and the probability of being in the conflict state p_2 approaches 1.

For the conflict rate (equation (8)), as μ approaches 1 (λ/μ approaches 0), the high frequency of reconciliation will bring the conflict rate $p_{conflict}$ to 0. Intriguingly, as μ approaches 0, the lack of opportunities to report conflicts with reconciliation processes will also bring the conflict rate down to 0. This finding is consistent with prior findings [13]. Thus, a system can spend most of its time updating two conflicting replicas, but only one conflict is reported per reconciliation process. Also, through this exhaustive range of ratios between the update and reconciliation rates, we can see that under this system setting the conflict rate can actually be bounded as a fraction of the total update and reconciliation events (11%) by solving equation (8) for its maximum. This is an important insight for capacity planning under optimistic replication.

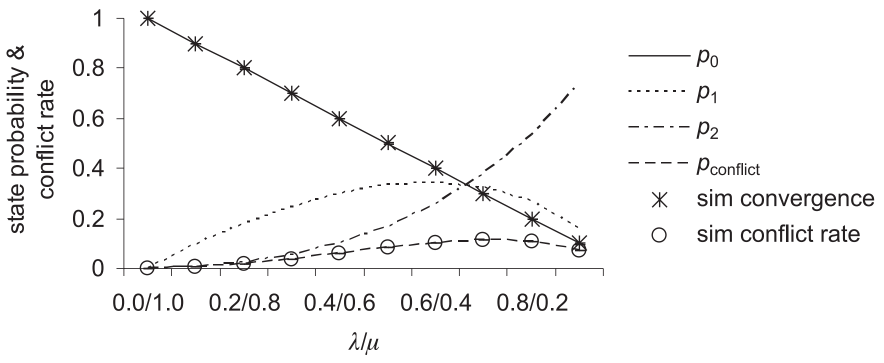


Figure 2. Percentage contributions of states for two replicas, superimposed on analytical conflict-rate curves and simulation validation data points. The simulation was repeated five times with different random seeds, each with a total of 100 000 update and reconciliation events. Confidence intervals are less than 1% of the mean and have been omitted for clarity

3.2 Validation for Two Replicas

To validate the analytical results based on the use of permuted states, we compare them with results obtained from a version-vector-based simulation of optimistic replication, similar to that used in a prior study [12]. Note that version vectors are widely used [1, 3, 5, 7, 10]; our findings are therefore applicable to systems where the semantics of conflict resolution are based on pairwise synchronization. For N replicas, a replica i keeps a 'version vector' V_i[*N update counters*]. Each counter in V_i represents the ith replica's current knowledge of updates made by other replicas. Whenever replica i performs an update, it increments its local counter $V_i[i]$.

A reconciliation process between replicas X and Y modifies the version vector according to the following rules.

- If $V_X[i] \geq V_Y[i]\ \forall\ i$, then X dominates Y. Y copies the version vector from X.
- If $V_Y[i] \geq V_X[i]\ \forall\ i$, then Y dominates X. X copies the version vector from Y.
- If X dominates Y and Y dominates X, then X and Y are equal.
- Otherwise, we have a conflict. To merge conflicting version vectors, $\forall\ i$, $V_X[i] = V_Y[i] = max(V_X[i], V_Y[i])$. 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.

For example, replicas 1 and 2 can start with an identical version, with $V_1 = (0, 0)$ and $V_2 = (0, 0)$. If replica 1 makes an update, V_1 changes from (0, 0) to (1, 0) since the first counter is the local counter of replica 1. V_2 stays at (0, 0), since replica 2 does not know about replica 1's update until the next reconciliation. At this point, reconciliation between the two replicas will detect that replica 1 dominates replica 2, since all version counters of V_1 are greater than or equal to those of V_2. Thus, the reconciliation process will copy the content of replica 1 over that of replica 2 and V_1 over V_2, so that both version vectors are set to (1, 0) with both replicas agreeing that an update has been made by replica 1.

Now, if replica 1 receives an update, V_1 will change from (1, 0) to (2, 0). If replica 2 also receives an update, V_2 will change from (1, 0) to (1, 1). However, if the two replicas reconcile, neither can dominate the other, since replica 1 is unaware of replica 2's latest update, and vice versa. Both version vectors will first be set to (2, 1), reflecting an update made by each replica.

If replica 1 is responsible for determining the outcome of the merged content, both resulting version vectors are set to (3, 1), indicating replica 1's role in the conflict-resolution process. Likewise, if replica 2 is responsible, both resulting version vectors are set to (2, 2). This counter increment is necessary, since the merged content cannot always be deterministically reproduced (e.g. two bank deposits can occur at the same time, and the exact order of transactions after the merge may not matter). Note that either (3, 1) or (2, 2) dominates both version vectors from before the conflict resolution.

Our simulation includes only one replicated item. We follow the methodology presented by Wang [12].

All simulation results are presented at the 90% confidence level. We assume that all updates and reconciliations take place instantly.

Figure 2 also shows the validation results based on simulation. The results are a good match with the model based on permuted states. To our knowledge, this is the first time that simulation results for optimistic replication have been cross-validated with an analytical model, although with only two replicas.

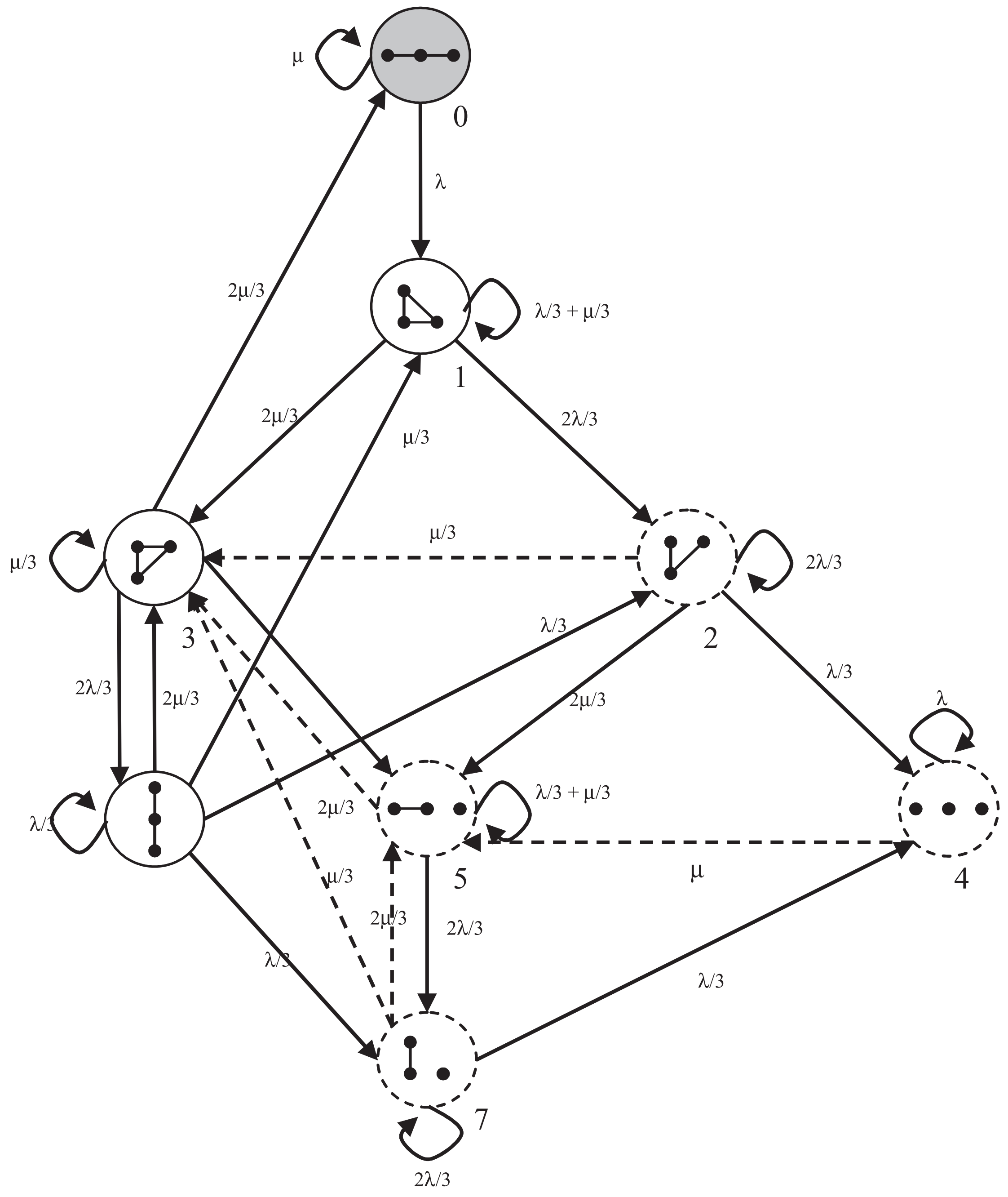


Figure 3. The state-transition diagram for three replicas. The shaded state is the starting state, where replicas are identical. States with dashed outlines contain conflicting replicas. The solid lines mark various transitions among states, and the dashed lines mark conflict-reporting transitions

3.3 Generalization to Three Replicas

The three-replica case tests whether we can reduce the original 64 states down to a more manageable number, and whether the notation and manipulation rules invented for the two-replica case can be generalized.

3.3.1 State-Transition Diagram for Three Replicas

Figure 3 shows the transition diagram for three replicas. Surprisingly, the three-replica case can be completely captured with only eight permuted states. The starting state (state 0), which is also the convergence state, is easily generalized from the two-replica case. However, since an update can be applied to any one of the replicas, the update-receiving replica dominates the two remaining identical replicas (state 1). At this point, an update to one of the two identical subordinate replicas will first break off the update-receiving replica from the replica that dominates it, and will then make the update-receiving replica dominate its original identical partner (state 2).

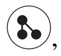
At state 2, it is interesting to note that the two dominating replicas are in conflict while dominating the same replica. A reconciliation between the two dominating replicas will lead to the report of a conflict and transition to state 3. An update to the subordinate replica will break the update-receiving replica from all its dominating replicas and reach state 4.

At state 3, reconciling between any dominating version and the subordinate version will lead to the convergence state (state 0). An update to any dominating version will lead to state 6. An update to the subordinate replica will break off the update-receiving replica from its dominating replicas and reach state 5. At state 4, an update to any replica will leave all three replicas in conflict. Reconciling any pair of replicas will lead to the report of a conflict and a transition to state 5.

At state 5, an update to one of the identical replicas will lead to state 7; reconciling between one of the identical replicas and the replica in conflict will lead to the report of a conflict and the generation of a new version that dominates the replica not involved in reconciliation (state 3).

At state 6, the dominance relationship is transitive. An update to the top dominating replica results in a self-transition. An update to the middle dominating replica breaks its relationship with the top dominating replica and leads to state 2 while preserving its dominating relationship to the subordinate replica. An update to the subordinate replica will break its relationship to both dominating replicas and lead to state 7. Reconciling the top dominating replica with either of the other replicas will result in state 3. Reconciling between the bottom two replicas will result in state 1.

At state 7, an update to the subordinate replica will lead to state 4. Reconciling between the dominating and subordinate replicas will result in state 5. Reconciling the dominating replica with the conflicting replica will result in state 3. Reconciling the subordinate replica with the conflicting replica is equivalent to making an update to the subordinate replica (state 4) and then reconciling it with the conflicting replica. The result is state 5.

Overall, the three-replica case demonstrates the variety of behaviors in optimistic replication. Interestingly, not all conceivable states are possible. For example, it is not possible to have , which represents one replica dominating two replicas that are themselves in conflict. Using permuted states for analysis eliminates both isomorphic states and unreachable states.

3.3.2 Analyses and Validation for Three Replicas

Similar to the two-replica analysis, we set up a system of equations based on the state-transition diagram in Figure 3. The exact equations used are listed in Appendix A. Note that the conflict rate is the sum of the product of the probability of each conflict-originating state with the outbound transition probability of its conflict-resolving edges:

$$p_{\text{conflict}} = \frac{\mu}{3} p_2 + \mu p_4 + \frac{2\mu}{3} p_5 + \frac{2\mu}{3} p_7. \tag{9}$$

After solving the system of equations with MathCAD, the following equations highlight our findings:

$$p_0 = \frac{4\mu^3}{(3\lambda + 2\mu)(\lambda + 2\mu)(\lambda + \mu)}, \tag{10}$$

$$p_4 = \frac{2\lambda^3}{(2\lambda + 3\mu)(\lambda + 2\mu)(\lambda + \mu)}, \tag{11}$$

$$p_{\text{conflict}} = \frac{2\lambda^2 \mu \left(3\lambda^2 + 11\lambda\mu + 9\mu^2\right)}{(2\lambda + 3\mu)(3\lambda + 2\mu)(\lambda + 2\mu)(\lambda + \mu)}. \tag{12}$$

One immediate surprise from these resulting equations is that the order of complexity is smaller than expected. For nine equations and eight unknowns, we would expect the resulting equations to have exponents of seven to eight; here we have four to five, suggesting that our permuted-state representation can be further compacted. For example, a state with a single inbound transition can be merged with the state that makes the inbound transition. Based on the equations listed in Appendix A, states p_0 and p_6 can be directly replaced with state p_3 (appropriately weighted), and the overall system can be characterized with six states.

Figure 4 shows the probability contributions of each state across an exhaustive range of λ/μ ratios. Similar to the two-replica case, as λ/μ approaches 0, the probability of convergence p_0 approaches 1. As λ/μ increases asymptotically, the probability of being in the fully divergent state p_4, where all replicas are conflicting versions, approaches 1. Also, in the case of a 0.9/0.1 λ/μ ratio, the most probable states (p_4, p_5 and p_7) contain conflicting versions.

Figure 5 shows the percentage contribution for the all-identical and all-conflict states and the conflict-rate curve. The remaining states have been removed for clarity. Compared to the two-replica case, the additional replica has increased the conflict-rate potential from 11% to 17%. Also, the peak of the curve shifts left, meaning that fewer updates are needed to cause more conflicts, given a fixed number of reconciliation processes.

In terms of the simulation validation, the data points once again confirm the validity of modeling optimistic replication via permuted states.

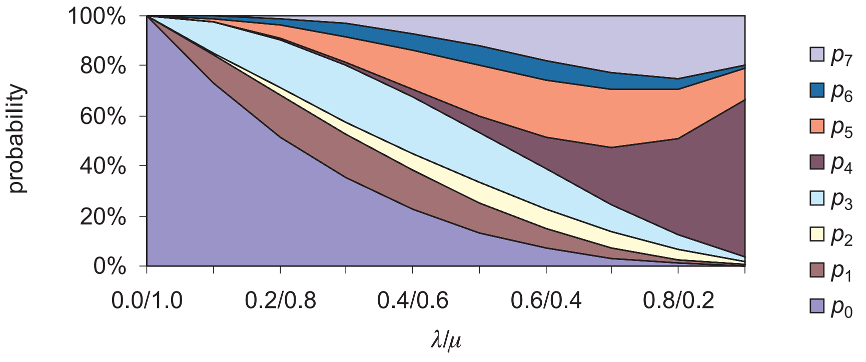


Figure 4. Percentage contribution of states in the three-replica case

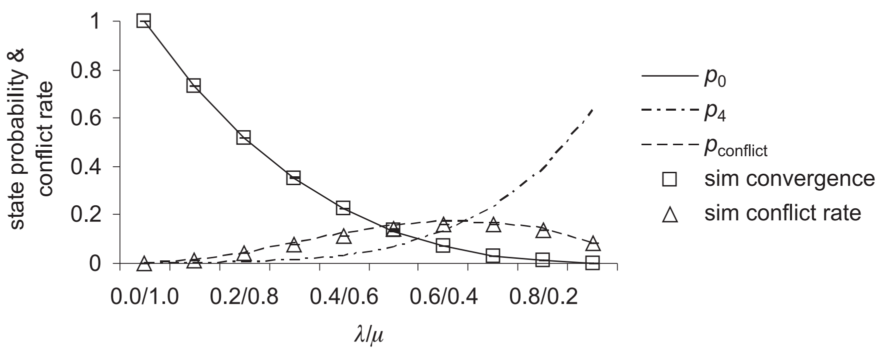


Figure 5. Percentage contributions of states for three replicas, superimposed on analytical conflict-rate curves and simulation validation data points. The simulation was repeated five times with different random seeds, each with 100 000 update and reconciliation events (i.e. 100K events in total). Confidence intervals are less than 1% of the mean and have been omitted for clarity

3.4 The Base Case for Identical Conflicts

The four-replica case is the base case for identical conflicts. This exploration helps us to gain insights on how identical conflicts are formed and how to predict their proliferation. Rather than repeating the same analyses as for two and three replicas, this section will only highlight the results of interest.

Surprisingly, the four-replica case contains only 27 permuted states (Appendix B), which is valuable for analyzing systems such as Oceanstore [14], where the core writable replicas have a replication factor of only four. Five of the states have only a single inbound transition, so the 27 states can potentially be compacted down to 22. Unfortunately, MathCAD did not find a closed form for these equations, so we used the Microsoft Excel solver to generate Figures 6 and 7, with 10 000 iterations to approximate each data point. The simulation results provide a good match for the curves derived from analytical equations. (Note that the peak conflict rate has increased and is shifted slightly to the left.)

As in the three-replica case (Figure 4), a few states have dominating contributions when the system operates with extreme λ/μ ratios, namely those close to the convergence state p_0 (states with many lines interconnecting dots that represent replicas), and those close to the divergent state p_8 (states with few lines interconnecting the dots that represent replicas). This finding prompts the question of whether it is possible to trim the state space for any number of replicas down to a characteristic subset of states. We will leave the answer as future work.

With the aid of permuted states, for the first time we can understand and enumerate the cases where identical conflicts are generated. The formation of an identical conflict goes through a setup sequence, as shown in Figure 8. Incidentally, this sequence is in the example we used in Section 2.3. During the setup phase, the system first enters the state with two pairs of identical replicas, replicas 1 and 2, and replicas 3 and 4. Based on the version vectors, we can tell that content X was originated by replica 1 and propagated to replica 2, with version vectors of (1, 0, 0, 0). Likewise, content Y was originated by replica 3 and propagated to replica 4, with version vectors of (0, 0, 1, 0). A reconciliation process between one replica from each pair (e.g. replicas 1 and 4) will result in a new pair of identical replicas with content XY, dom-

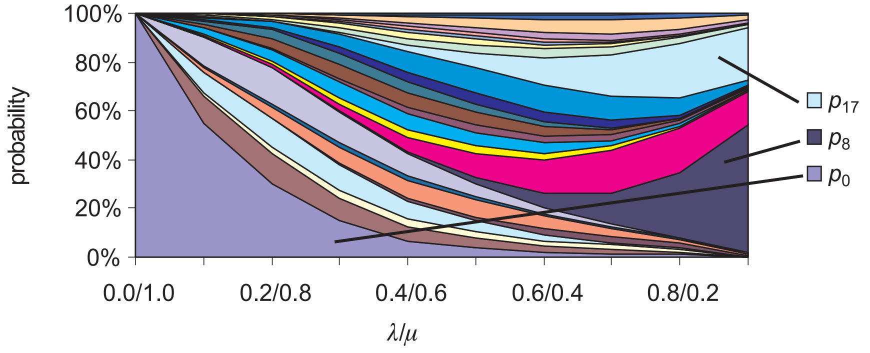


Figure 6. Percentage contribution of states in the four-replica case

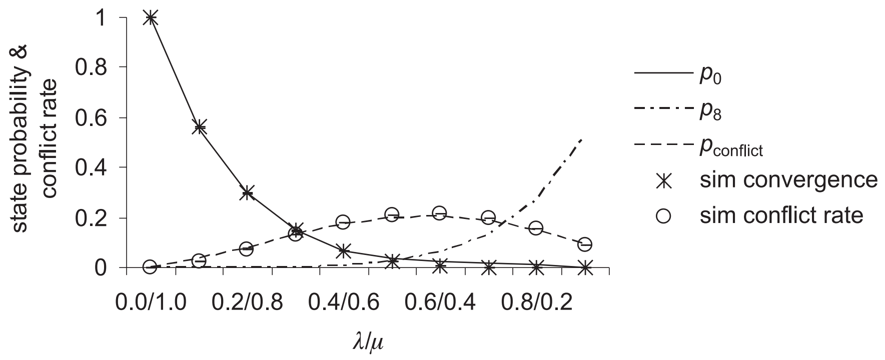


Figure 7. Percentage contributions of states for four replicas, superimposed on analytical conflict-rate curves and simulation validation data points. The analytical numbers were approximated by the Microsoft Excel solver with 10 000 iterations per data point. p_0 is the convergence state and p_8 is the fully divergent state. The simulation was repeated five times with different random seeds, each with a total of 100 000 update and reconciliation events. Confidence intervals are less than 1% of the mean and have been omitted for clarity

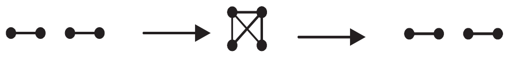
inating the remaining two replicas (e.g. replicas 2 and 3) which remain in conflict with each other. At the version-vector level, if replicas 1 and 4 reconcile and replica 1 is responsible for the merged version, the resulting version vector will be (2, 0, 1, 0) which dominates both replica 2's version vector (1, 0, 0, 0) and replica 3's version vector (0, 0 1, 0). When replicas 2 and 3 reconcile and merge their content, they will create a new version vector, (1, 1, 1, 0), during the conflict reconciliation. This vector will no longer be dominated by the version vectors of replicas 1 and 4.

Figure 9 shows the states involved in forming identical conflicts. For simplicity, not all outbound transitions are shown. Basically, after the setup sequence, as long as one replica from each version pair remains, they can reconcile and form identical conflicts. The other two replicas can be in a number of states resulting from updates and reconciliations.

Consider the first right-looping branch from Figure 9 as an illustrative example. Figure 10 begins with two sets of identical pairs, continued from Figure 8. Based on version vectors, replicas 1 and 4 are identical, as are replicas 2 and 3. If replicas 1 and 2 reconcile, we will have an identical conflict based on their version information; however, the content is the same. Replicas 3 and 4 can reconcile and form another identical conflict.

3.5 Comments on Identical Conflicts

During the process of analyzing the base case for identical conflicts, we made two disturbing observations. First, for the four-replica scenario, we can see that identical conflicts can potentially be self-inducing due to looping behavior. Based on a prior simulation study of large-scale optimistic replication systems [12], identical conflicts constitute most of the conflicts and this looping behavior may be a contributing factor. Second, identical conflicts are defined as a function of data content, in addition to the system states. In Figure 9, the same states can generate either regular or identical conflicts. However, tracking the data content and system states is analytically prohibitive



	Replica 1		Replica 2		Replica 3		Replica 4	
Event	Content	Version	Content	Version	Content	Version	Content	Version
	X	1000	*X*	1000	*Y*	0010	*Y*	0010
Reconcile (1,4)	***XY***	**2010**	*X*	1000	*Y*	0010	***XY***	**2010**
Reconcile (2,3)	*XY*	2010	***XY***	**1110**	***XY***	**1110**	*XY*	2010

Figure 8. A setup sequence of events for identical conflicts in the four-replica case, corresponding to Figure 8. The states being reconciled are in boldface

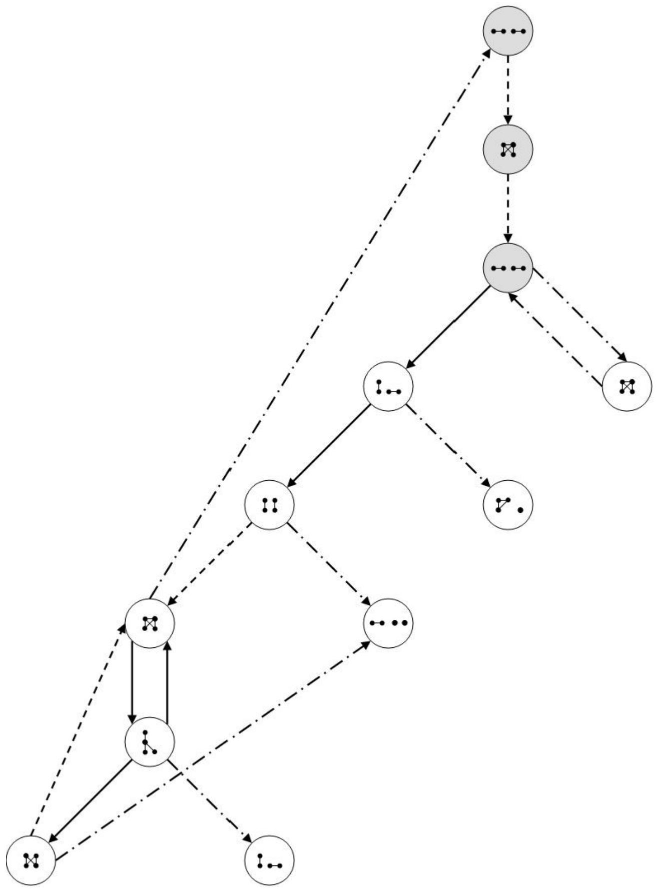
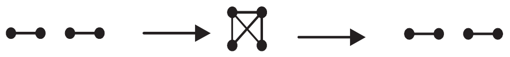


Figure 9. A subset of four-replica states and transitions illustrating methods of forming identical conflicts. The shaded states are the setup sequence. The solid lines mark various transitions among states, the dashed lines mark conflict transitions and the dotted dashed lines mark identical-conflict transitions

for even a few replicas since we can no longer compact states effectively.

4. Automatic Generation of Permuted States

Although permuted states enable a significant step forward in understanding replicated systems, there is still a



	Replica 1		Replica 2		Replica 3		Replica 4	
Event	Content	Version	Content	Version	Content	Version	Content	Version
	XY	2010	*XY*	1110	*XY*	1110	*XY*	2010
Reconcile (1, 2)	***XY***	**3110**	***XY***	**3110**	*XY*	2010	*XY*	1100
Reconcile (3, 4)	*XY*	3110	*XY*	3110	***XY***	**2120**	***XY***	**2120**

Figure 10. The first right-looping branch corresponding to Figure 9. The states being reconciled are in boldface

limit to the size of the analysis. In exploring higher replication factors, we thus turn from pure analysis to an analytically validated simulation. However, such validation requires an automated way to generate analytical solutions for higher replication factors. Although the number of states still grows rapidly, the ability to validate a simulation of up to ten replicas can cover common replication deployment scenarios and give confidence in the accuracy of even larger simulations. Even without closed-form solutions to equations, using random traversals of the states with appropriate probabilities for updates and reconciliations can be a good sanity check for simulation results.

Four steps are involved in automating state generation, including: (1) map the diagram state representation into a data structure; (2) define rules to transition among states; (3) remove isomorphic states resulting from permuted labeling of replicas; and (4) traverse states with the given transition probabilities.

4.1 Data Structure Representation

The first simplification in our automation is an assumption of global knowledge, which allows us to directly translate diagram states into graph-based representations. To illustrate, identical replicas 1 and 2 have the state

$$\begin{bmatrix} = & = \\ = & = \end{bmatrix}.$$

The first row belongs to replica 1. The equality symbols indicate that replica 1 is identical to itself and to replica 2. The second row belongs to replica 2, showing that replica 2 is identical to replica 1 and to itself.

If replicas 1 and 2 are in conflict, we have state

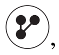
$$\begin{bmatrix} = & * \\ * & = \end{bmatrix}.$$

The asterisks 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 replica 2, we have the state

$$\begin{bmatrix} = & > \\ < & = \end{bmatrix}.$$

The greater than sign shows that replica 1 dominates replica 2 (vector 1), and the less than sign shows that replica 2 is subordinate to replica 1 (vector 2).

In this simple two-replica scenario, vectors 1 and 2 appear to contain redundant information. However, as the number of replicas increases, we need each replica to track its relationship to others to capture the full complexity of system states. For example, with , replicas 1 (top left), 2 (top right), and 3 (the bottom) can be represented with the state

$$\begin{bmatrix} = & * & > \\ * & = & > \\ < & < & = \end{bmatrix}.$$

4.2 State Transition Rules

The state transition rules are also translated from the state diagram. An update to a dominating replica does not change the dominance of the replica. An update to a subordinate replica breaks its relationship with its dominating replicas. An update to a number of identical replicas makes one of the replicas dominating over all other replicas.

Reconciliation rules fall into one of the following three categories. (1) Reconciling two identical replicas: no actions are needed. (2) Reconciling a dominant and a subordinate replica: the subordinate replica first receives an update from the dominating replica (with update rules applied, meaning that if a subordinate is dominated by two replicas, it has to break off from both dominating replicas first) and then copies over the vector from the dominating replica. (3) Reconciling conflicting replicas: each replica first receives an update to form a new data version (with similar update rules applied). For each vector element, if one of the replicas dominates a third replica not involved in reconciliation, both replicas are set to dominate the third. If one replica is identical to a third, both reconciling replicas are set to be identical to each other, dominating the third. Note that since conflict resolution is equivalent to first applying chosen updates to the conflicting replicas in order to make them equal and then reconciling them, they cannot be subordinate to any other replica after reconciliation.

4.3 Isomorphic State Reduction

As we analyze the system at the level of permuted states, the labeling of replicas becomes irrelevant. For example, in the two-replica case, we make no distinctions between replica 1 dominating over 2 and replica 2 dominating over 1 i.e.

$$\begin{bmatrix} = & > \\ < & = \end{bmatrix} \text{ cf. } \begin{bmatrix} = & < \\ > & = \end{bmatrix}.$$

In the three-replica case of , we make no distinctions between

$$\begin{bmatrix} = & * & > \\ * & = & > \\ < & < & = \end{bmatrix}, \quad \begin{bmatrix} = & > & * \\ < & = & < \\ * & > & = \end{bmatrix}$$

$$\text{and} \quad \begin{bmatrix} = & < & < \\ > & = & * \\ > & * & = \end{bmatrix}.$$

This problem of compacting isomorphic states into a permuted state is similar to the problem of finding isomorphic graphs. Unlike subgraph isomorphism, the isomorphic graph problem is neither NP-complete, nor a P-problem [15]. However, known algorithms for constant-bounded vertex degree are $O(n^4)$ [16], which is still too computationally intensive to scale well.

An intriguing observation is that one state can be turned into another by swapping two corresponding rows and columns, with the invariant that the diagonal entries are always '='. For example, swapping the first and third rows of

$$\begin{bmatrix} = & * & > \\ * & = & > \\ < & < & = \end{bmatrix}$$

leads to

$$\begin{bmatrix} < & < & = \\ * & = & > \\ = & * & > \end{bmatrix}.$$

The first and third columns are then swapped to obtain the isomorphic state:

$$\begin{bmatrix} = & < & < \\ > & = & * \\ > & * & = \end{bmatrix}.$$

This raises the possibility that we might find a reduced representation of the state that captures these variations due to permutations. If so, we can look up the isomorphic states in a dictionary. We currently use a hash table for this purpose.

To construct the reduced representation, we must consider the following constraints. (1) We need a function that is commutative for row elements and column elements, so that any swapping between two rows and columns results in the same value. (2) We need to break the diagonal symmetry of matrices by using two different functions for rows and columns, or our reduction will not be as effective in eliminating unintended collisions (false isomorphisms). False compaction of states will confound the probability of reaching distinct states and increase the error of our analyses. (3) We need to account for all matrix elements.

The '=', '>', '<', and '*' symbols are first mapped to four 32-bit numbers (chosen randomly at design time). Currently, we use a hash function H of a permuted state M that is the sum of values of three functions: F_1, F_2 and F_3 (integer overflow is ignored), defined:

$$H(M) = F_1(M) + F_2(M) + F_3(M), \quad (13)$$

$$F_1(M) = \prod_i \left(\sum_j M_{ij} \right), \quad (14)$$

$$F_2(M) = \sum_i \left(\prod_j M_{ij} \right), \quad (15)$$

$$F_3(M) = \sum_{ij} M_{ij}. \quad (16)$$

The observed false compaction rate is around 0.6% for six replicas, found by comparing the number of obtained states with that obtained by brute-force permutation of matrix rows and columns. The false compaction rate is expected to become lower as the number of replicas increases, because two valid matrices first have to follow the transitivity constraints (if replica A dominates B, and if replica B dominates C, then replica A dominates C) and be hashed to the same value.

Figure 11 compares the effectiveness of the permuted state approach to the number of states obtained by theoretical limits (four states for each replica pair) and brute-force enumeration of reachable states. Note that the y axis uses a logarithmic scale. Using permuted states for as few as six replicas can reduce the state space by 2–6 orders of magnitude.

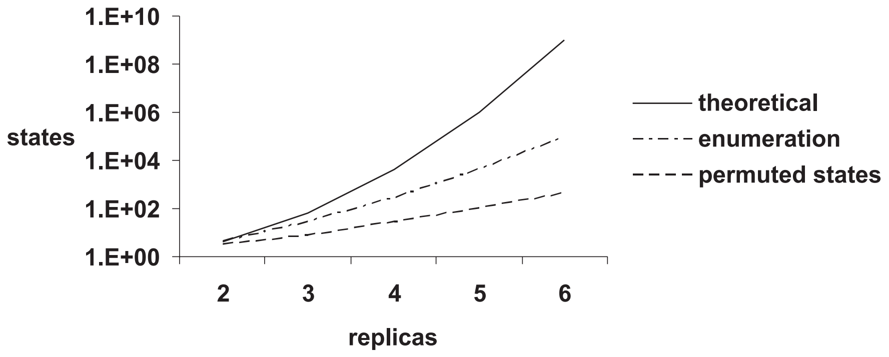


Figure 11. Comparison of number of system states obtained by theoretical limits (four states for each replica pair), brute-force enumeration of reachable states and permuted states

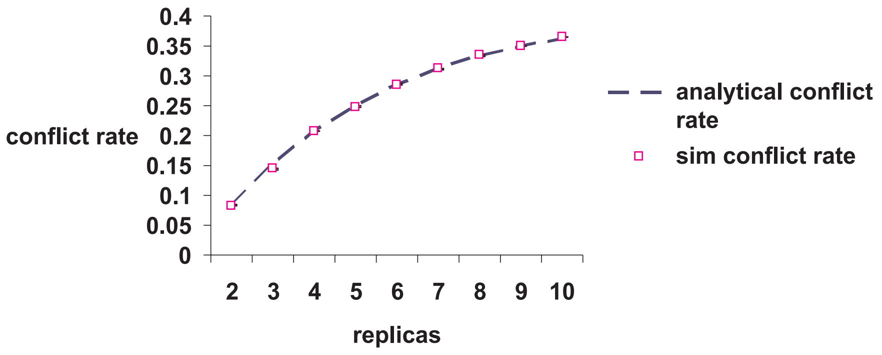


Figure 12. Comparison between automated analytical modeling based on permuted states and simulation with counter-based version vectors. The analytical numbers were obtained with 100 000 random transitions with $\lambda = \mu$. The simulation was repeated five times with different random seeds, each with a total of 100 000 update and reconciliation events. Confidence intervals are less than 1% of the mean so have been omitted for clarity

4.4 Validation

After building the permuted states, all we need to do is construct the state diagram through systematic state enumeration and randomly traverse the graph with specified update and reconciliation arrival rates. For $\lambda = \mu$, we validated our automated analytical method (which contains 488 013 permuted states) against a simulation with up to 10 replicas (Figure 12). Intuitively, λ should be much greater than μ in real systems. However, based on trace analyses [12], due to write-back caching and the work cycle (2-day weekends and 8-hour working days), the average λ and μ are not that far apart.

In the past, simulations of optimistic replication have only been validated for two replicas. Now we have two independent implementations of optimistic models that cross-validate well even at ten replicas. We are thus more confident in using simulation to explore optimistic systems at higher replication factors.

5. Identical Conflicts Revisited

A prior trace-based simulation study has shown that identical conflicts account for a significant fraction of conflicts at high replication factors (around 50 replicas) [12]. However, the simulation validated by our analytical model shows that identical conflicts are relatively rare events compared to the overall conflicts (Figure 13). (Note that although the non-identical conflict rate appears to be high, most of them can be automatically resolved. The analyses of identical and non-identical conflicts mostly contribute to the understanding of the system dynamics of optimistic replication and determining whether resources devoted to conflict resolution can be bound and predicted.)

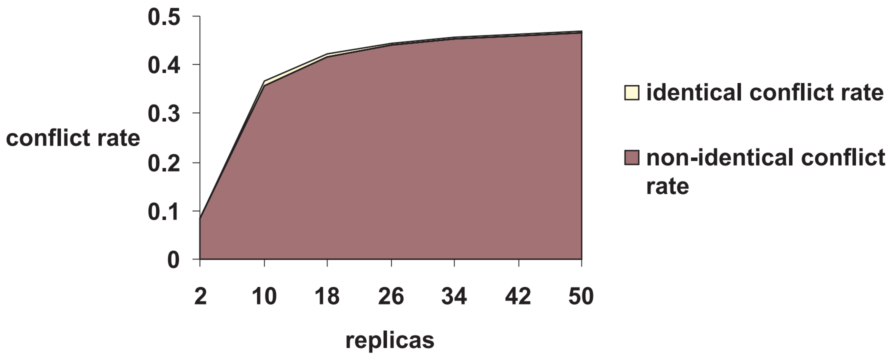


Figure 13. A stack graph of non-identical and identical conflict rates at different replication factors. The simulation was repeated five times with different random seeds, each with a total of 100 000 update and reconciliation events, with $\lambda = \mu$. The identical-conflict curve is just above the non-identical one. The difference is too small to be seen

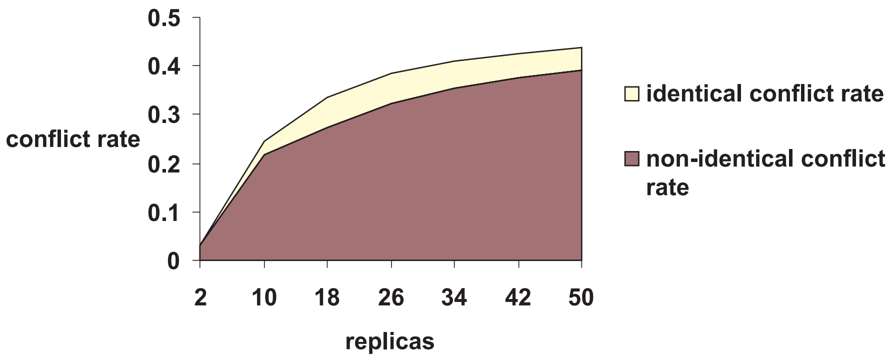


Figure 14. A stack graph of non-identical and identical conflict rates at different replication factors, with 90% of the updates going to 10% of the replicas. The simulation was repeated five times with different random seeds, each with a total of 100 000 update and reconciliation events, with $\lambda = \mu$

The previous study also suggested that identical conflicts are caused by access locality; most updates are applied to a subset of replicas. To test this possibility, we adjusted our simulation to have 90% of the updates take place at only 10% of the replicas.

Figure 14 shows that the impact of access locality is clearly visible, but the effect is within 10% of the total. To make sure that we were within the plausible range of parameter settings, we examined conflict and identical conflict rates as a function of λ/μ, with 90% of the updates going to 10% of the replicas.

Figure 15 shows the decomposition of identical and non-identical conflicts with a wide range of λ/μ ratios. The non-identical conflict rate is low for both high and low λ/μ ratios, due to either the lack of updates to create diverging versions, or the lack of reconciliation to detect conflicts. When $\lambda = \mu$, the identical conflict rate is expected to be low. However, as the λ/μ ratio decreases, the identical conflict rate accounts for most conflicts. It is tempting to conclude that optimistic replication systems should operate with only high λ/μ ratios, but such a system will suffer from slow propagation of updates. While not addressed in this paper, another important performance metric for optimistic replication systems is how often stale data is read because a replica has not yet received the latest update; high λ/μ ratio systems will perform poorly by this metric.

From the viewpoint of traces, updates occur more frequently than reconciliations on average, to amortize the cost of reconciliation over time. Therefore, we hypothesize that a high average λ/μ ratio with accompanying high identical conflict rate is induced by a bimodal traffic pattern, where updates arrive in bursts to maintain an upper range of λ/μ, while the system is running a lengthy back-

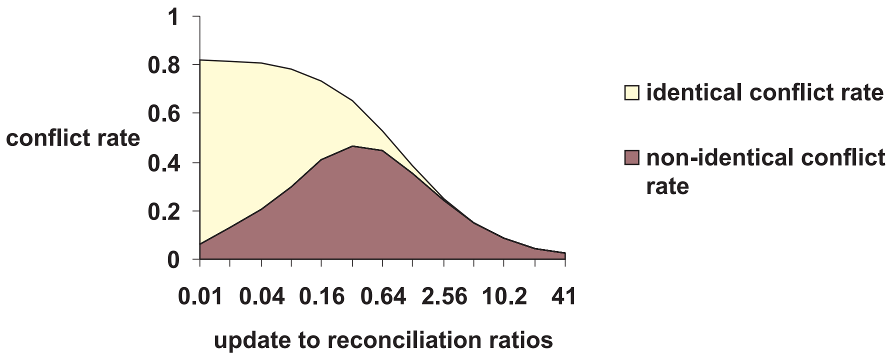


Figure 15. A stack graph of non-identical and identical conflict rates for 50 replicas versus various λ/μ ratios (in log scale) with 90% of updates going to 10% of all replicas. The simulation was repeated five times with different random seeds, each with a total of 100 000 update and reconciliation events

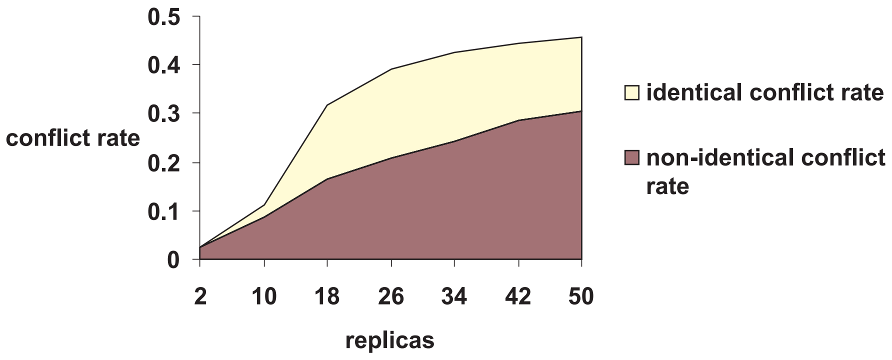


Figure 16. A stack graph of non-identical and identical conflict rates under different replication factors, with 90% of updates going to 10% of all replicas. A bimodal traffic pattern, where the λ/μ ratio is 8 for one-third of the time reflecting 8-hour working days and is 0.08 to reflect non-working hours, is in place. The simulation was repeated five times with different random seeds, each with a total of 100 000 update and reconciliation events

ground reconciliation most of the time. This temporal locality of updates is also sensible from the viewpoint of a single replicated file, which is likely to be updated intensively over short durations with periods of no updates but many reconciliations.

We model this behavior of update bursts with an irregular square wave, to reflect weekly activities. The function parameters are extracted from the same trace used in [12]. The function consists of five 8 hour working sessions with $\lambda/\mu = 8$, each followed by a 16 hour 'off' session dominated by reconciliation with $\lambda/\mu = 0.08$. The five eight-hour days are followed by two 24 hour reconciliation periods. A working hour has an average of three updates, and the remaining hours have an average of one reconciliation per hour. The overall average value of λ/μ is about 0.86, based on the aggregate number of updates and reconciliations. Access locality still applies.

Figure 16 shows a much more drastic decomposition of conflict rate compared to Figure 14. Clearly, bimodal access patterns have a large impact on identical conflicts. Intriguingly, based on Figure 15, an average λ/μ of 0.86 should produce relatively few identical conflicts. However, beyond ten replicas, identical conflicts can account for up to 48% of the total. This suggests that load generators based on aggregate mean arrival rates of updates and reconciliations are not suitable for studying optimistic replication. Also, given that our square function is a crude approximation of a trace, a direct trace-based simulation would be expected to have more identical conflicts, which is consistent with prior findings [12].

Interestingly, the introduction of access and temporal localities to the workload affects the overall conflict statistics (beyond ten replicas) very little, but the internal composition changes dramatically.

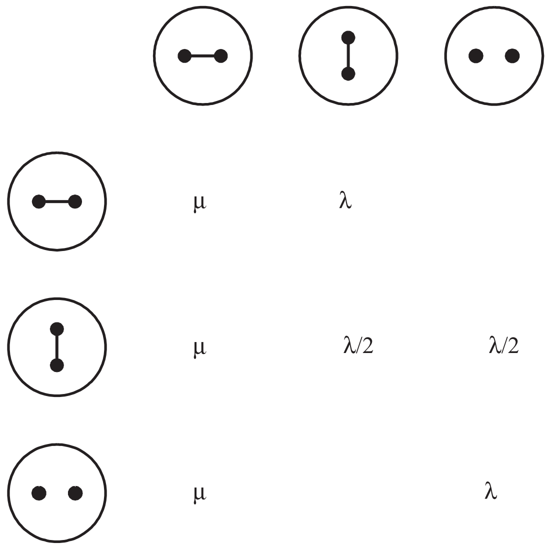


Figure 17. The state-transition table for two replicas. The rows represent the *from* states, and the columns the *to* states

6. Related Work

The permuted-state approach suggests that it might be profitable to revisit existing approaches to evaluating optimistic replication. An early approach by Golding [17] was to measure the mean time to converge with R conflicting replicas. No updates were allowed during the convergence process.

Golding's approach is equivalent to making λ/μ zero. Based on Figures 4 and 6, for three and four replicas optimistic replication under this setting is dominated by only a few states. Many states can be overlooked because they are only reachable through a mixture of update and reconciliation events. On the other hand, we have also observed that an optimistic replication system typically operates with alternating extremes of λ/μ. Therefore, Golding's approach does reflect the case where reconciliation events dominate the system. With only the state transitions, one might conclude that Golding's approach misses 33% of system states for two replicas (Figure 17), since it starts with the fully divergent states where all replicas are in conflict and transitions directly back to the starting state. The intermediate state is not exercised at all. For three replicas, Golding's approach misses 38% of states; 56% for four replicas. In practice, however, many missed states are not heavily exercised, since alternating between extreme λ/μ ratios is the norm.

Gray et al. [8] studied replication under a database workload, with relatively uniform access patterns to all replicated items. Their results suggest that the conflict rate grows at a rate that is prohibitive for scaling of optimistic replication. However, Gray's analytical model assumes an access pattern that is not applicable in environments where update locality is the norm. Also, due to the strong correlation between the usage model and the working day, our traffic pattern is bimodal. Our model cycles through extreme update-to-reconciliation ratios.

Kistler and Satyanarayanan [6] have conducted an empirical study of disconnected operation in the Coda file system, showing a low likelihood of concurrent updates [11]. A study of the Ficus file system [3] showed that optimistic replication used in an office environment achieved an extremely low conflict rate after the automation of conflict resolution for many applications and after removing identical conflicts. The study reported many identical conflicts, but their relationships to the system parameters and their implications on scaling were not explained. Neither the Coda nor the Ficus experience has examined the relationship between the update-to-reconciliation ratio and the formation of identical conflicts, which constitute the majority of conflicts.

There are relatively few studies that use both simulation and analytical methods to investigate the causes of conflicts and identical conflicts. Through a trace-driven simulation, one paper observed the inverse relationship between update locality and conflict and identical conflict rates [12]. However, our study has further investigated the effect of a bimodal traffic pattern on optimistic replicated systems. Another paper analytically characterized the conflict rate, but the results were limited to two replicas and not applicable in the general case [13]. Our use of permuted states can capture the combinatorial growth of states at a small scale, which is representative of most replication scenarios.

There have also been other studies that examined the service quality of optimistic replication [18, 19]. However, the behavior of the conflict-rate curve was not deeply explored in these studies.

7. Future Work

Through this exploration of optimistic replication with both analytical and simulation approaches, we have begun to gain more mature intuition about the behavior of replicated systems. Although the state space of optimistic replication is large, we believe that a system can be reasonably characterized with fewer than 200 states in order to capture all major aggregate statistics. We intend to design, implement, analyze and automate algorithms to extract the top contributing states. The aggregate statistics obtained from a *trimmed* state diagram will be compared with the full state diagram for verification. If successful, we can use this simplified model to provide system feedback, prediction and tuning at runtime.

Since traffic characteristics can significantly influence the fraction of conflicts that are identical, a fruitful area for future research would be to construct traffic filters that can shape the decomposition of conflicts. Ideally, we want no conflicts. If that is not possible, we want most conflicts to be identical for easy resolution. (Note that for modeling purposes, an automated resolver such as those described in [3, 11] would cause non-identical conflicts to behave as if they were identical.)

8. Lessons and Conclusion

The results presented here capture several iterations of experimentation with analytical methods, and many findings are not obvious in retrospect. We originally made naïve attempts to cluster states with conflicts into small sets of superstates to simplify the computation. However, the results were similar to variable substitutions in complex equations. Although the resulting state-transition diagram had fewer states, the complexity of the equations remained unchanged.

Upon realizing that conflicts occur as transitions, not states, we tried to insert *probing* states into each conflict-generating transition. To be specific, a state can be inserted into a transition, where its inbound probability is identical to the original probability of transition, but its outbound probability is 1. The hope was that the equilibrium probability of the probing state would capture the conflict probability. Unfortunately, the probing states significantly distorted the results, making it difficult to compute conflict probabilities.

We have described methods to represent, automate and optimize permuted states, which have enabled us to use analytical methods to explore the four-replica base case of identical conflicts and to automate the analytical investigation up to ten replicas. All results have been independently confirmed by a simulation based on version vectors. As a consequence, we discovered that update locality and bimodal access patterns are the primary factors that influence the fraction of identical conflicts.

The analysis of problems with exponential state spaces is always challenging. By introducing the concept of permuted states, we have developed a new technique that makes the base-case analysis of complex replicated systems tractable. As a result, we have been able to characterize and quantify important system behaviors that have previously been unrecognized or poorly understood.

Appendix A: State-Equilibrium Equations for Three Replicas

$$
\begin{aligned}
\lambda p_0 &= \frac{2\mu}{3} p_3 \\
\frac{2(\lambda+\mu)}{3} p_1 &= \lambda p_0 + \frac{\mu}{3} p_6 \\
\left(\frac{\lambda}{3} + \mu\right) p_2 &= \frac{2\lambda}{3} p_1 + \frac{\lambda}{3} p_6 \\
\left(\lambda + \frac{2\mu}{3}\right) p_3 &= \frac{2\mu}{3} p_1 + \frac{\mu}{3} p_2 + \frac{2\mu}{3} p_5 + \frac{2\mu}{3} p_6 \\
&+ \frac{\mu}{3} p_7 \\
\mu p_4 &= \frac{\lambda}{3} p_2 + \frac{\lambda}{3} p_7 \\
\frac{2(\lambda+\mu)}{3} p_5 &= \frac{2\mu}{3} p_2 + \frac{\lambda}{3} p_3 + \mu p_4 + \frac{2\mu}{3} p_7 \\
\left(\frac{2\lambda}{3} + \mu\right) p_6 &= \frac{2\lambda}{3} p_3 \\
\left(\frac{\lambda}{3} + \mu\right) p_7 &= \frac{2\lambda}{3} p_5 + \frac{\lambda}{3} p_6 \\
& p_0 + p_1 + p_2 + p_3 + p_4 \\
&+ p_5 + p_6 + p_7 = 1
\end{aligned}
$$

Appendix B: State-Equilibrium Equations for Four Replicas

$$
\begin{aligned}
\lambda p_0 &= \frac{\mu}{2} p_7 \\
\left(\frac{3\lambda}{4} + \frac{\mu}{2}\right) p_1 &= \lambda p_0 + \frac{\mu}{3} p_{14} \\
\left(\frac{\lambda}{2} + \frac{5\mu}{6}\right) p_2 &= \frac{3\lambda}{4} p_1 + \frac{\lambda}{4} p_6 + \frac{\mu}{6} p_{21} \\
\left(\lambda + \frac{2\mu}{3}\right) p_3 &= \frac{\mu}{2} p_1 + \frac{\mu}{6} p_2 + \frac{\mu}{6} p_6 + \frac{\mu}{2} p_{11} \\
&+ \frac{\mu}{6} p_{13} + \frac{\mu}{6} p_{14} + \frac{\mu}{6} p_{15} \\
\left(\frac{\lambda}{4} + \mu\right) p_4 &= \frac{\lambda}{2} p_2 + \frac{\lambda}{4} p_{12} + \frac{\lambda}{4} p_{21} \\
\left(\frac{3\lambda}{4} + \frac{5\mu}{6}\right) p_5 &= \frac{2\mu}{3} p_2 + \frac{\lambda}{2} p_3 + \frac{\mu}{2} p_4 + \frac{\mu}{6} p_{12} \\
&+ \frac{\lambda}{4} p_{13} + \frac{\mu}{3} p_{15} + \frac{2\mu}{3} p_{21} + \frac{\mu}{3} p_{22} \\
&+ \frac{\mu}{6} p_{23} + \frac{2\mu}{3} p_{26}
\end{aligned}
$$

$$\left(\frac{3\lambda}{4}+\frac{5\mu}{6}\right)p_6 = \frac{\lambda}{2}p_3+\frac{\mu}{6}p_{20}$$

$$\left(\lambda+\frac{\mu}{2}\right)p_7 = \frac{2\mu}{3}p_3+\frac{2\mu}{3}p_{13}+\frac{2\mu}{3}p_{19}$$

$$\mu p_8 = \frac{\lambda}{4}p_4+\frac{\lambda}{4}p_{17}+\frac{\lambda}{4}p_{22}$$

$$\begin{aligned}\left(\frac{\lambda}{2}+\frac{5\mu}{6}\right)p_9 &= \frac{\mu}{2}p_4+\frac{\lambda}{4}p_5+\mu p_8+\frac{\lambda}{4}p_{16}\\ &+ \frac{\mu}{2}p_{17}+\frac{\lambda}{4}p_{18}+\frac{\mu}{2}p_{22}\\ &+ \frac{\mu}{6}p_{24}+\frac{\mu}{6}p_{26}\end{aligned}$$

$$\begin{aligned}\left(\lambda+\frac{2\mu}{3}\right)p_{10} &= \frac{\mu}{6}p_5+\frac{\mu}{6}p_9+\frac{\mu}{6}p_{16}+\frac{\mu}{6}p_{18}\\ &+ \frac{\mu}{6}p_{19}\end{aligned}$$

$$\left(\frac{3\lambda}{4}+\frac{\mu}{2}\right)p_{11} = \frac{\mu}{3}p_5+\frac{\lambda}{4}p_7+\frac{\mu}{3}p_{16}$$

$$\left(\frac{\lambda}{2}+\mu\right)p_{12} = \frac{\lambda}{2}p_5+\frac{\lambda}{2}p_6+\frac{\lambda}{4}p_{20}$$

$$\begin{aligned}\left(\lambda+\frac{5\mu}{6}\right)p_{13} &= \frac{\mu}{3}p_5+\frac{\mu}{3}p_6+\frac{\mu}{6}p_{12}+\frac{\mu}{3}p_{14}\\ &+ \frac{\mu}{3}p_{16}+\frac{\mu}{2}p_{20}+\frac{\mu}{6}p_{21}\\ &+ \frac{\mu}{6}p_{23}+\frac{\mu}{3}p_{25}\end{aligned}$$

$$\left(\frac{3\lambda}{4}+\frac{5\mu}{6}\right)p_{14} = \frac{\mu}{3}p_6+\frac{3\lambda}{4}p_7+\frac{\mu}{3}p_{20}+\frac{\mu}{3}p_{25}$$

$$\left(\frac{\lambda}{2}+\frac{5\mu}{6}\right)p_{15} = \frac{3\lambda}{4}p_{11}+\frac{\mu}{6}p_{12}+\frac{\lambda}{4}p_{14}+\frac{\mu}{6}p_{23}$$

$$\begin{aligned}\left(\frac{3\lambda}{4}+\frac{5\mu}{6}\right)p_{16} &= \frac{2\mu}{3}p_9+\frac{\mu}{3}p_{12}+\frac{\lambda}{4}p_{13}+\frac{\mu}{3}p_{15}\\ &+ \frac{\mu}{3}p_{17}+\frac{\mu}{3}p_{18}+\frac{\lambda}{2}p_{19}+\frac{\mu}{6}p_{22}\\ &+ \frac{\mu}{3}p_{23}+\frac{\mu}{3}p_{24}\end{aligned}$$

$$\left(\frac{\lambda}{4}+\mu\right)p_{17} = \frac{\lambda}{2}p_9+\frac{\lambda}{4}p_{12}+\frac{\lambda}{4}p_{23}+\frac{\lambda}{2}p_{24}$$

$$\begin{aligned}\left(\frac{3\lambda}{4}+\frac{5\mu}{6}\right)p_{18} &= \lambda p_{10}+\frac{\mu}{6}p_{12}+\frac{\mu}{6}p_{17}+\frac{\mu}{6}p_{23}\\ &+ \frac{\mu}{3}p_{24}+\frac{\mu}{6}p_{25}\end{aligned}$$

$$\begin{aligned}\left(\lambda+\frac{5\mu}{6}\right)p_{19} &= \frac{2\mu}{3}p_{10}+\frac{\mu}{3}p_{18}+\frac{\mu}{6}p_{24}+\frac{\mu}{6}p_{25}\\ &+ \frac{\mu}{6}p_{26}\end{aligned}$$

$$\left(\frac{3\lambda}{4}+\mu\right)p_{20} = \frac{\lambda}{2}p_{13}$$

$$\left(\frac{\lambda}{2}+\mu\right)p_{21} = \frac{\lambda}{2}p_{14}+\frac{\lambda}{4}p_{20}$$

$$\left(\frac{3\lambda}{4}+\mu\right)p_{22} = \frac{\lambda}{2}p_{15}+\frac{\lambda}{4}p_{21}+\frac{\lambda}{4}p_{23}+\frac{\lambda}{2}p_{26}$$

$$\left(\frac{\lambda}{2}+\mu\right)p_{23} = \frac{\lambda}{2}p_{16}+\frac{\lambda}{4}p_{20}+\frac{\lambda}{2}p_{25}$$

$$\left(\frac{\lambda}{2}+\mu\right)p_{24} = \frac{\lambda}{2}p_{18}$$

$$\left(\frac{3\lambda}{4}+\mu\right)p_{25} = \frac{\lambda}{2}p_{19}$$

$$\left(\frac{\lambda}{2}+\mu\right)p_{26} = \frac{\lambda}{4}p_{25}$$

$$\sum_{i=0}^{26}p_i = 1$$

$$\begin{aligned}p_{\text{conflict}} &= \frac{\mu}{6}p_2+\frac{\mu}{2}p_4+\frac{\mu}{3}p_5+\mu p_8+\frac{5\mu}{6}p_9\\ &+ \frac{2\mu}{3}p_{10}+\frac{\mu}{2}p_{11}+\frac{\mu}{3}p_{12}+\frac{\mu}{2}p15+\frac{\mu}{2}p_{16}\\ &+ \frac{5\mu}{6}p_{17}+\frac{2\mu}{3}p_{18}+\frac{\mu}{6}p_{19}+\frac{\mu}{6}p_{21}\\ &+ \frac{2\mu}{3}p_{22}+\frac{\mu}{2}p_{23}+\frac{2\mu}{3}p_{24}+\frac{\mu}{6}p_{25}+\frac{\mu}{3}p_{26}\end{aligned}$$

References

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

[2] Kawell, L.J., S. Beckhardt, T. Halvorsen, R. Ozzie, and I. Greif. 1992. Replicated Document Management in a Group Communication System. *Groupware: Software for Computer-Supported Cooperative Work*, IEEE Computer Society Press.

[3] Reiher, P., J. Heidemann, D. Ratner, G. Skinner, and G. Popek. 1994. Resolving File Conflicts in the Ficus File System. In *Proceedings of USENIX Conference*.

[4] Daniels, D., L.B. Doo, A. Downing, C. Elsbernd, G. Hallmark, S. Jain, B. Jenkins, P. Lim, G. Smith, B. Souder, and J. Stamos. 1994. Oracle's Symmetric Replication Technology and Implications for Application Design. In *Proceedings of SIGMOD Conference*.

[5] Terry, D.B., M.M. Theimer, K. Petersen, A.J. Demers, M.J. Spreitzer, and C.H. Hauser. 1995. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of the 15th ACM Symposium on Operating Systems Principle*.

[6] Kistler, J.J. and M. Satyanarayanan. 1992. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems* 10(1).

[7] Wang, A.I.A., P. Reiher, and R. Bagrodia. 1999. A Simulation Evaluation of Optimistically Replicated Filing in Mobile Environments. In *Proceedings of the 18th IEEE International Performance, Computing, and Communication Conference* (*IPCCC*).

[8] Gray, J., P. Helland, P. O'Neil, and D. Shasha. 1996. The Dangers of Replication and a Solution. In *Proceedings of the 1996 ACM SIGMOD Conference*, pp. 173–182.

[9] Guy, R., G. Popek, and T.W. Page. 1993. Consistency Algorithms for Optimistic Replication. In *Proceedings of the 1st International Conference on Network Protocols*, *IEEE*.

[10] Page, T., R. Guy, J. Heidemann, D. Ratner, P. Reiher, A. Goel, G. Kuenning, and G. Popek. 1997. Perspectives on Optimistically Replicated, Peer-to-Peer Filing. *Software—Practice and Experience*, 28(2): 155–180.

[11] Kumar, P. and M. Satyanarayanan. 1995. Flexible and Safe Resolution of File Conflicts. In *Proceedings of the 1995 USENIX Technical Conference*.

[12] Wang, A.I.A. 1998. *A Simulation Evaluation for Optimistically Replicated Filing Environments*. M.Sc. Thesis. Computer Science Department, University of California, Los Angeles.

[13] Wang, A.I.A., P. Reiher, R. Bagrodia, and G. Kuenning. 2002. Understanding the Behavior of the Conflict-Rate Metric in Optimistic Peer Replication. In *Proceedings of the 5th IEEE International Workshop on Mobility in Databases and Distributed Systems* (*MDDS*), Aix-en-Provence, France.

[14] Kubiatowicz, J., D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. 2000. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems* (*ASPLOS 2000*).

[15] Skiena, S. 1990. Graph Isomorphism. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Reading, Massachusetts: Addison-Wesley; 181–187.

[16] Foggia, P., C. Sansone, and M. Vento. 2001. A Performance Comparison of Five Algorithms for Graph Isomorphism. In *Proceedings of the 3rd Workshop on Graph-based Representations in Pattern Recognition*.

[17] Golding, R.A. 1992. *Weak-Consistency Group Communication and Membership*. Ph.D. Thesis, Department of Computer Science, University of California, Santa Cruz.

[18] Rowstron, A.I.T., N. Lawrence, and C.M. Bishop. 2001. Probabilistic Modeling of Replica Divergence. In *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems*.

[19] Yu, H. and A. Vahdat. 2000. Design and Evaluation of a Continuous Consistency Model for Replicated Servers. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation*.

***An-I A. Wang** is an assistant professor of computer science at Florida State University. He received his Ph.D. and M.Sc. in computer science from UCLA in 2003 and 1998. His research interests include file systems, optimistic peer-to-peer replication, performance evaluation, ad hoc network routing, operating systems and distributed systems.*

***Geoff Kuenning** is an associate professor of computer science at Harvey Mudd College. He received his Ph.D. in computer science from UCLA in 1997, and his M.Sc. in computer science from Michigan State University in 1974. From 1974 to 1989, he worked in the areas of operating systems and embedded systems. His research interests include file systems, performance analysis and computer system security.*

***Peter Reiher** is an adjunct associate professor of computer science at UCLA. He received his Ph.D. and his M.Sc. in computer science from UCLA in 1987 and 1984, respectively. Dr. Reiher's research interests include active networks, advanced operating systems, parallel discrete event simulation and security for distributed systems.*