Analysis of Durability in Replicated Distributed Storage Systems

Sriram Ramabhadran
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093-0404, USA
sriram@cs.ucsd.edu

Joseph Pasquale
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093-0404, USA
pasquale@cs.ucsd.edu

***Abstract*—In this paper, we investigate the roles of replication vs. repair to achieve durability in large-scale distributed storage systems. Specifically, we address the fundamental questions: How does the lifetime of an object depend on the degree of replication and rate of repair, and how is lifetime maximized when there is a constraint on resources? In addition, in real systems, when a node becomes unavailable, there is uncertainty whether this is temporary or permanent; we analyze the use of timeouts as a mechanism to make this determination. Finally, we explore the importance of memory in repair mechanisms, and show that under certain cost conditions, memoryless systems, which are inherently less complex, perform just as well.**

I. Introduction

Replication is a cornerstone of reliable distributed system design. By replicating functionality (which may involve data, computation, communication or any combination of these), at multiple locations in the system, a distributed application can achieve the redundancy necessary to tolerate individual failures. Reliability has many aspects in this context; one well-studied metric is *availability*, defined as the fraction of time the system is able to provide access to the replicated entity. The entity is available when any one of its replicas is functioning; conversely, it is unavailable when all of its replicas are not.

Another metric of reliability, which is the focus of this paper, is *durability*, defined as the duration of time the system is able to provide access to this entity. This is significant when the replicas in the system are susceptible to failures of a more permanent nature; if such attrition is ignored, the system will eventually lose all replicas of the entity in question. Thus, while availability deals with temporary inaccessibility when all replicas are non-operational, durability deals with more permanent loss, when the system no longer has even a single replica.

Our work is motivated by several efforts [8], [13], [9], [3], [1] that use replication to build distributed wide-area storage systems in a peer-to-peer environment. Engineering reliability in this environment is a complex and challenging endeavor; storage is provided by autonomous peer nodes (connected by a wide-area network such as the Internet) whose participation in the system is generally entirely voluntary. The lack of control over peer participation implies that the underlying storage substrate in such a system may be highly unreliable. A peer node may be offline for significant periods of time, resulting in the temporary unavailability of any data stored on it. In addition, it may cease its participation in the system at any time, resulting in the permanent loss of that data. Given that these failures are significantly more frequent than, say, disk failures in a conventional storage system, achieving high reliability is far from trivial.

Replication can be used by the system to provide high availability in a statistical sense [1]. To significantly extend the durability of an object for periods exceeding individual node lifetimes, the system must also implement a *repair* mechanism that compensates for lost replicas by creating new ones. The repair process must be automated; the system monitors the set of replicas, and creates a new one when necessary. The decision to trigger a repair is necessarily based on incomplete information, as in general, the system cannot determine exactly when a replica is lost. It may have to infer this based on, for example, if a particular node is unresponsive for a significant period of time. Clearly, durability is a function of both replication and repair, as in addition to the number of replicas, it also depends on how aggressively the system responds to the potential loss of a replica [2].

In this paper, our objective is to investigate durability of objects in distributed storage systems as characterized by the most basic of parameters: degree of replication, replication rate, and failure rate. The key features of the problem we deal with are that (1) applications are expected to significantly outlive the nodes on which they run; (2) replication and repair are used to achieve durability, and both must be automated; (3) there is an inherent uncertainty in the repair process in that the system cannot perfectly distinguish between components that are temporarily down

[1]Under the assumption of independence, if the availability of an individual node, and therefore replica, in the system is p, the availability of an object with r replicas (each on a different node, and assuming nodes fail independently) is given by $1-(1-p)^r$.

[2]In abstract terms, the repair process may be associated with a *rate* parameter [10] that quantifies how aggressive or lazy the system is in carrying out a repair. In practice, this depends on the exact repair mechanism used.

vs. those that have permanently failed. The last point raises interesting questions, such as "Is repair actually needed (or will just waiting solve the problem)?", and "If a mistake is made (a repair is made that is actually not needed, or no repair occurs when it actually was needed), what should the system do, if anything, to compensate?".

II. System Model

The *system* consists of and controls some population of storage *nodes* on which it can replicate objects. By the term system, we mean the distributed logic whose function is to ensure the durability of the objects it stores. This objective would be trivial if the storage substrate were perfectly reliable; unfortunately, it is not. A node may "leave," i.e., no longer be part of, the system at any time, at which point the system cannot retrieve any objects stored on that particular node. Nodes that leave the system are replaced by new ones; thus, we assume that the system is never limited by the inability to find a node on which to replicate. However, it must take measures to ensure that objects persist over a constantly changing population of nodes.

A node participates in the system for some duration of time called its *lifetime*. This is the time that elapses between the instant a node initially "joins," i.e., becomes part of, the system, and the instant the node permanently leaves it. The lifetime of a node consists of alternating periods of availability and unavailability. Node *uptime* refers to periods when the node is online (or available), while *downtime* refers to periods when the node is offline (or unavailable). When a node is offline, it is temporarily unable to participate in the system; thus, any objects stored on that particular node are inaccessible until the node becomes online again.

A. Node model

Our model for node behavior is based on simple statistical assumptions of node lifetime, uptime and downtime. In particular, we assume that node lifetimes are exponentially distributed with mean T. In addition, we assume that node uptimes and downtimes are also exponentially distributed with means t and $\bar{t}$ respectively, where $T \gg t, \bar{t}$. We assume that nodes are homogeneous, i.e., the behavior of each node is governed by an i.i.d. random process.

Exponentially distributed uptimes and downtimes imply that the underlying random process is Markovian; consequently, we model node behavior by a continuous Markov chain with three states. A node may be either *online* (state 1), *offline* (state 2) or *dead* (state 3). A node always begins its existence online (in state 1), and alternates between online (state 1) and offline (state 2), before it is finally goes dead (absorbed in state 3). More precisely, the only valid transitions are online to offline ($1 \rightarrow 2$), offline to online ($2 \rightarrow 1$), and online to dead ($1 \rightarrow 3$). [3]

It remains to ensure that the lifetime, the time between beginning in state 1 and ending in state 3, is exponentially distributed. We must also obtain the appropriate values for the transition rates λ_{12}, λ_{13} and λ_{21}, in terms of t, $\bar{t}$ and T. Uptimes X_i are identically distributed as the sojourn time of the chain in state 1, denoted by the random variable $X \sim$ Exponential $(\lambda_{12} + \lambda_{13})$. Equating $E[X]$ to t,

$$(\lambda_{12} + \lambda_{13})^{-1} = t \tag{1}$$

Similarly, downtimes $\bar{X}_i$ are identically distributed as the sojourn time of the chain in state 2, denoted by the random variable $\bar{X} \sim$ Exponential (λ_{21}). Equating $E\left[\bar{X}\right]$ to $\bar{t}$,

$$\lambda_{21}^{-1} = \bar{t} \tag{2}$$

Finally, the lifetime is the absorption time of the chain in state 3, denoted by the random variable Y, having begun in state 1. If the chain visits state 2 N times ($N \geq 0$) before being absorbed in state 3, Y is given by

$$Y = \sum_{i=1}^{N} \left(X_i + \bar{X}_i\right) + X_{N+1} \tag{3}$$

Let $p_{13} = \frac{\lambda_{13}}{\lambda_{12}+\lambda_{13}}$ be the transition probability from state 1 to state 3. With probability p_{13}, $N = 0$, and with probability $1 - p_{13}$, $N \sim$ Geometric (p_{13}), which implies

$$\begin{aligned} E[N] &= \frac{1 - p_{13}}{p_{13}} \\ &= \frac{\lambda_{12}}{\lambda_{13}} \end{aligned} \tag{4}$$

Therefore

$$\begin{aligned} E[Y] &= E\left[\sum_{i=1}^{N} \left(X_i + \bar{X}_i\right) + X_{N+1}\right] \\ &= E[N]\, E\left[X + \bar{X}\right] + E[X] \\ &= \frac{\lambda_{12}}{\lambda_{13}} \left(t + \bar{t}\right) + t \end{aligned}$$

Equating $E[Y]$ to T,

$$\frac{\lambda_{12}}{\lambda_{13}} \left(t + \bar{t}\right) + t = T \tag{5}$$

[3]Despite the symmetry it adds to the model, we do not consider the transition $2 \rightarrow 3$ because it does not correspond to any directly observable behavior by the node. It is certainly conceivable that a node does not return to the system while offline, and thus becomes dead. But since the system cannot determine the point at which such a transition occurs (it may not even be well defined), it is simply considered dead from the point it became offline.

Solving Equations 1,2 and 5, and using $T \gg \bar{t}$ to simplify yields [4]

$$\begin{aligned}
\lambda_{12} &= \frac{1}{t} - \frac{1}{pT} \\
\lambda_{13} &= \frac{1}{pT} \\
\lambda_{21} &= \frac{1}{\bar{t}}
\end{aligned} \tag{6}$$

where

$$p = \frac{t}{t + \bar{t}} \tag{7}$$

where p is the node availability, i.e., the fraction of time a node is online. Finally, verifying that lifetime Y is exponentially distributed, we show in the appendix that if $T \gg \bar{t}$, $Y \sim \text{Exponential}\left(\frac{1}{T}\right)$.

B. Replication and repair

We now focus our attention on how the system can implement durable storage. As noted earlier, the system must use a combination of replication and repair for this purpose. The process of *replication* consists of making copies of an object; the system is said to create a *replica* when it copies the object onto some node in the system. For simplicity, we will assume that a node stores at most one replica, allowing us to characterize the state of a replica based on the state of its underlying node. Thus, a replica may be either *online*, *offline* or *dead*, depending on the node that stores it. The degree of replication is defined by the number of replicas of the object initially created by the system; we denote this system parameter by r.

Given a degree of replication r, the system starts by creating r replicas of the object [5]. Subsequently, the system attempts to maintain r replicas of the object at all times[6] through a process of *repair*. Repair is necessary to replace replicas that are dead; otherwise, node departures will eventually result in the permanent loss of the object when all its replicas are dead. To guard against this, the system must effect a repair, i.e., create a new replica of the object whenever it detects that an existing replica is dead. Ideally, a repair should be made only when a replica is dead; however, as we will now discuss, the temporary unavailability of an offline replica has significant implications for the repair process.

First, the system cannot distinguish between a replica that is dead, and one that is merely offline (in both cases, its node is unresponsive). Thus, the system cannot reliably detect when a replica is dead; each time a replica transitions out of the online state, the system is dealing with the potential loss of a replica, and must decide whether to make a repair or not. We will call a replica that is not online, i.e., is in either the offline or dead state, as *not-online*. A natural mechanism to decide when to trigger a repair is to wait some period of time for the not-online replica to return to the online state, i.e., to use a *timeout* [7]. If the replica returns before the timeout occurs, no repair is necessary; on the other hand, if it does not, the system assumes that it is dead, and triggers a repair. Of course, this mechanism is not perfect; the system will occasionally time out a replica that is not dead, but is merely offline for a long time.

Second, when a timeout occurs, the system attempts to create a new replica; however, it may not be able to do so immediately. If all replicas of the object are not-online, the repair must be delayed until at least one of the replicas is online so that the object is available to be copied onto a new node. This delay depends on node availability; in general, higher availability implies it is less likely the system will have to wait to make a repair. We revisit this relationship between availability and repair in Section III-B2.

Choosing a timeout period depends on how aggressively the system wants to repair. Intuitively, it would appear that there is a balance between causing frequent, and perhaps unnecessary, repairs when the system prematurely times out a replica (i.e., one that would return if the system waited longer), and delaying replacing a replica when it actually is dead. To investigate this tradeoff, we assume the timeout is given by $\alpha\,\bar{t}$, where α is a system parameter. It is natural to express the timeout period in terms of mean node downtime: on average, a node is down, and hence a replica is offline, for a period of $\bar{t}$. In addition, if downtimes are exponentially distributed, we can compute the probability that an offline replica will be prematurely timed out simply as $\Pr\{\bar{X} > \alpha\,\bar{t}\} = e^{-\alpha}$, i.e., the probability that downtime exceeds the timeout period. Thus, the repair parameter α allows the system to trade off how aggressively it responds to the potential loss of a replica versus how many unnecessary repairs are made.

The degree of replication r and the repair parameter α represent the space of possible strategies for system to achieve durability. There is one additional factor that we now consider, e.g., the role of *memory*. A replica that has been timed out will subsequently "return" if it was simply offline but not dead. There are two broad approaches to how the system can deal with this.

Repair without memory Repair is said to be *without-memory* if the system simply discards such a replica, i.e., it does not reintegrate this replica into the set of remaining ones. Rather, it assumes that the replica is lost (in effect, treating its node as though it were joining the system for the first time). The advantage of the without-memory approach

[4]First solve and simplify for λ_{13}, and then use Eq. 1 to solve for λ_{12}

[5]By object we mean a file; however it could also be a smaller unit, such as a file block, or a large unit, such as a group of files belonging to the same user.

[6]Note that not all r replicas need be online simultaneously.

[7]While a timeout is certainly not the only repair mechanism that the system could use, its simplicity makes it an attractive candidate for actual implementation.

is its simplicity, not just in terms of modeling but also in actual system implementation. In addition to the system not having to "remember" replicas that time out, its resource usage is predictable as the number of replicas of an object never grows beyond r. The disadvantage of the without-memory approach is that it is potentially inefficient by not making use of replicas that were previously considered lost. A key question is, how significant is this inefficiency, and can it be tolerated to enjoy the benefits of design and implementation simplicity? We address this in Section III.

The alternative to without-memory repair is *with-memory* repair, in which the system remembers the timed-out replica, and may decide to readmit it to the set of existing replicas. This enables the system to take advantage of what is essentially a "free" repair. Depending on the specific conditions under which the system readmits a timed-out replica, there could be many possible with-memory repair strategies. In this study, we adopt the following strategy: the system readmits a timed-out replica only if the number of replicas is currently below the target number r. Since the system always attempts to effect a repair as soon as possible after a timeout, this scenario would occur when the system was unable to complete the repair due to all other replicas being not-online. We choose this strategy because it is both reasonable and fairly comparable to the without-memory repair strategy. It enables the system to take advantage of memory by readmitting a replica when necessary, while retaining the property that the number of replicas does not exceed the target of r, just like the without-memory strategy.

C. Metrics

The main measure of durability that we will use in our evaluation is *object lifetime* [8]. This is defined as the time that elapses between the instant the system starts by creating r replicas of the object, and the instant the system no longer has any replica of the object. Object lifetime is a variable that depends on the random processes governing node behavior; therefore, in most cases, we will use its expected value, denoted by L, as our measurement standard.

Durability cannot be considered in isolation; we need metrics for resources consumed by replication (storage) and repair (communication), respectively. For simplicity and generality, we characterize these in a way that is independent of replica size. Thus, the number of replicas r itself is a measure of the storage consumed by replication. To model communication incurred by repair, we define the *cost* of repair, denoted by C, to be the long-term rate of replica regeneration. We normalize this quantity in terms of the mean node lifetime T (for reasons that will soon be apparent); therefore, the cost C is simply the number of copies of the object that the repair process makes, on average, over a time period of T. Clearly, both L and C depend on node parameters T, t and $\bar{t}$, system parameters r and α, and finally, the nature of repair (without-memory versus with-memory). In Section III, we characterize this dependence.

We conclude this section by considering what range of costs constitutes a reasonable cost regime. Adopting the approach in [3], we consider a hypothetical scenario in which the system has access to an oracle that can precisely predict the behavior of each node; in particular, it can tell when a node is about to leave the system permanently [9]. In this case, the system can achieve an infinite object lifetime with just one replica by making a repair just before that replica is about to leave. The long-term rate of replica regeneration for this process is one replica every mean node lifetime T, resulting in a normalized cost of $C = 1$. While the existence of such an oracle is obviously unrealistic, it serves as a guideline for what a reasonable cost regime might be (within the same order of magnitude, for example).

III. Analysis

Although our node model is Markov, the use of a fixed timeout complicates a completely analytic characterization of L and C. Our evaluation is therefore a mixture of analytical (Section III-A) and simulation results (Section III-B).

A. Analytic results

Our analytic results in this section consist of deriving an expression for the expected amount of time that elapses before a replica is timed out by the system (Equation 11). Bounds on the cost of repair C (Equation 12) follow directly.

1) Frequency of timeouts: Let Y_α denote the time that elapses between the instant a replica is created, and the instant it leaves the online state before being timed out (i.e., it is either dead or offline for a period exceeding the timeout). Note that this does not include the timeout period itself; the "time to timeout" is Y_α and the timeout value itself, i.e., $Y_\alpha + \alpha\bar{t}$. Clearly, this quantity depends on α; the smaller α is, the more likely a replica will be timed out when it is offline. Similar to Equation 3, Y_α is given by

$$Y_\alpha \quad = \quad \sum_{i=1}^{N_\alpha} \left(X_i + \bar{X}_{\alpha,i}\right) + X_{N_\alpha+1} \qquad (8)$$

where the random variable $\bar{X}_\alpha$ denotes replica downtime given a timeout period of $\alpha\,\bar{t}$. We note that uptime remains unchanged due to a timeout; however, downtime, the time during which a replica is offline, must be conditioned on the fact that it is less than the timeout period [10]. A simple

[8]Note that this is different from *node* lifetime.

[9]Note that we do not assume that the oracle can predict the lifetime of a node in advance; rather, it can only inform the system about an event just before it happens.

[10]This is the reason why the system is no longer Markov.

application of Bayes' theorem gives the p.d.f. $f_{\bar{X}_\alpha}$ of $\bar{X}_\alpha$ as

$$f_{\bar{X}_\alpha}(x) = \left\{ \begin{array}{cc} \frac{1}{(1-e^{-\alpha})}\, e^{-\frac{x}{\bar{t}}} & \text{if } x < \alpha\bar{t} \\ \\ 0 & \text{if } x \geq \alpha\bar{t} \end{array} \right\}$$

Therefore

$$\begin{aligned} E\left[\bar{X}_\alpha\right] &= \int_0^\infty x\, f_{\bar{X}_\alpha}(x)\; dx \\ &= \bar{t}\left(1 - \frac{\alpha\, e^{-\alpha}}{1-e^{-\alpha}}\right) \end{aligned} \qquad (9)$$

The random variable $N_\alpha (N_\alpha \geq 0)$ denotes the number of times the replica goes offline without being timed out. In state 1, the replica goes to state 3 with probability p_{13}, where a timeout will occur. With probability $1 - p_{13}$, the replica goes to state 2, where either a timeout occurs with probability $e^{-\alpha}$, or the replica goes back to state 1 with probability $(1 - e^{-\alpha})$. Recall from Section II that the probability that the system will prematurely time out an offline replica is $e^{-\alpha}$. Thus, with probability p_{13}, $N_\alpha = 0$, and with probability $(1-p_{13})(1-e^{-\alpha})$, $N_\alpha \sim$ Geometric $(p_{13} + (1-p_{13})\, e^{-\alpha})$, which implies

$$E\left[N_\alpha\right] = \frac{(1-p_{13})(1-e^{-\alpha})}{p_{13} + (1-p_{13})\, e^{-\alpha}} \qquad (10)$$

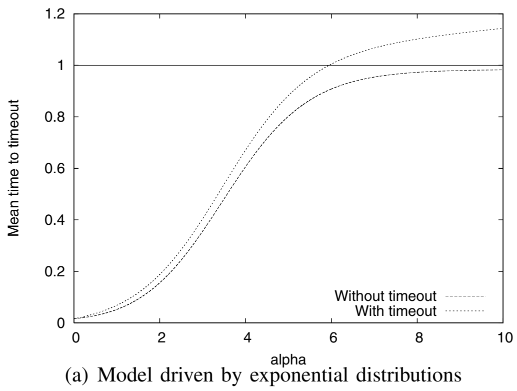
Equations 9 and 10 yield

$$\begin{aligned} E\left[Y_\alpha\right] &= E\left[N_\alpha\right]\left(E\left[X + \bar{X}_\alpha\right]\right) + E\left[X\right] \\ &= \frac{(1-p_{13})(1-e^{-\alpha})\left[t + \bar{t}\left(1 - \frac{\alpha\, e^{-\alpha}}{1-e^{-\alpha}}\right)\right]}{p_{13} + (1-p_{13})\, e^{-\alpha}} \\ &\quad + t \end{aligned} \qquad (11)$$

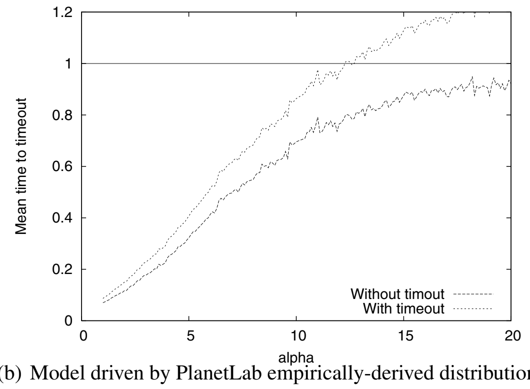
For $\alpha = 0$, i.e., a replica is timed out as soon as it leaves the online state, Equation 11 reduces to $E\left[Y_0\right] = t$, the mean node uptime. For $\alpha = \infty$, i.e., a replica is never timed out, Equation 11 yields $E\left[Y_\infty\right] = \frac{1-p_{13}}{p_{13}}\left(t + \bar{t}\right) \approx T$, the mean node lifetime.

Figure 1(a) shows curves for both $E\left[Y_\alpha\right]$ and $E\left[Y_\alpha\right] + \alpha\bar{t}$ (mean time to timeout) as a function of α. The values of T, t and $\bar{t}$ are 1 month, 12 hours, and 12 hours respectively. For small α, a replica is more likely to be timed out in state 2; this happens relatively quickly. On the other hand, for large α, the replica is less likely to be timed out in state 2, and times out in state 3 after, on average, a duration close to T. In between, there is a transitional behavior during which $E\left[Y_\alpha\right]$ increases rapidly [11], before tapering off. This "knee" is the point where the tradeoff of increasing α to avoid prematurely timing out an offline replica and decreasing α to avoid delaying a necessary repair is balanced. Figure 1(b) shows the same graphs except that the node uptimes and downtimes are drawn from the PlanetLab empirical distributions (based on tests that we carried out in [10]) rather then exponential. We note that the same trends are observed, except that the knee in the curve occurs at larger values of the timeout, a fact that can be attributed to the heavier tail of the empirical distributions [12]. Finally, we note that in both cases the point where $E\left[Y_\alpha\right] + \alpha\bar{t} = T$, i.e., the point where the upper curve intersects 1, is the point where the repair process generates exactly one new replica every node lifetime. Intuitively, this seems like a desirable operating point for the system, a fact we will demonstrate in Section III-B4.

2) Bounds on cost: Equation 11 can be used to derive bounds on the cost C of repair as follows. Consider any one of the r replicas; it times out after a duration given by the random variable $Y_\alpha + \alpha\bar{t}$. In response, the system creates a new replica immediately if possible; if not, it may have to wait until at least one other replica is online. Therefore, the expected time that elapses between the instant a replica is created and the instant it is replaced is at least



(a) Model driven by exponential distributions



(b) Model driven by PlanetLab empirically-derived distributions

Figure 1. $E\left[Y_\alpha\right]$ and $E\left[Y_\alpha\right] + \alpha\bar{t}$ as a function of α, as given by Equation 11. The Y-axis is scaled by the mean node lifetime T. Note the similarity in the shapes of the curves derived from both exponential and PlanetLab empirically-derived distributions.

[11]It is straightforward to verify that the probabilities that a replica is timed out in states 2 and 3 are $\frac{(1-p_{13})e^{-\alpha}}{p_{13}+(1-p_{13})e^{-\alpha}}$ and $\frac{p_{13}}{p_{13}+(1-p_{13})e^{-\alpha}}$ respectively. Therefore the replica is more likely to timeout in state 3 when $e^{-\alpha} < \frac{p_{13}}{1-p_{13}} \approx \frac{t+\bar{t}}{T}$

[12]Since there is a larger chance that a replica will be offline for a longer time, the system must wait a correspondingly longer time before increasing the timeout starts producing diminishing returns

$E[Y_\alpha] + \alpha\bar{t}$. The long-term rate of replica regeneration is therefore at most $\frac{1}{E[Y_\alpha]+\alpha\bar{t}}$. Normalizing for node lifetime T, and accounting for r replicas, we get

$$C \quad \leq \quad \frac{r\,T}{E[Y_\alpha] + \alpha\bar{t}} \tag{12}$$

We note that this bound applies to both without-memory and with-memory repair. In the case of without-memory repair, it is possible to derive a lower bound on cost as well. After a replica times out, if the system cannot make a repair immediately, it must do so within another timeout period $\alpha\bar{t}$. If this were not the case, all the other replicas would be timed out as well, and in the case of without-memory repair, the object would no longer be available. Therefore, the expected duration after which a replica is replaced is at most $E[Y_\alpha] + 2\alpha\bar{t}$, and therefore,

$$C \quad > \quad \frac{r\,T}{E[Y_\alpha] + 2\alpha\bar{t}} \tag{13}$$

Figure 2 shows the bound on cost for $r = 1$, i.e., the cost per replica.

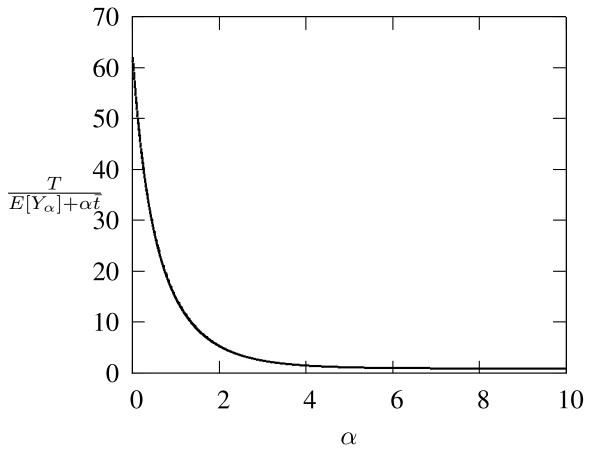
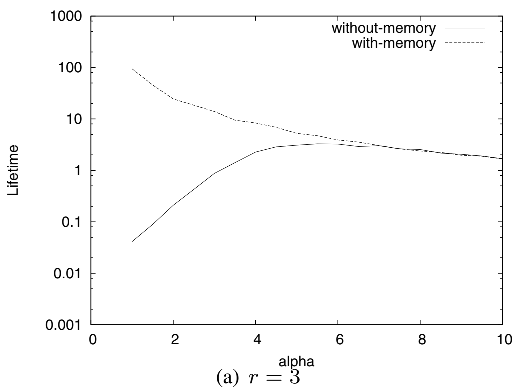


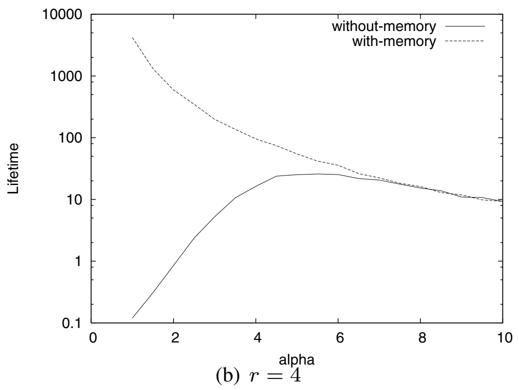
Figure 2. Bounds on the cost of repair. When α is small, many repairs occur that are unnecessary, as the likelihood of timing out a replica that is simply offline (rather than dead) is high. This leads to high cost, as the replication rate is high. Cost then decreases sharply as α increases beyond small values, as an offline replica will have had a chance to return (which is "free"), rather than requiring a repair.

B. Simulation results

This section presents the results of our simulation study. For each (r, α) pair, we simulate the behavior of the system over a large number number of runs (typically 1000). For each run, we record the lifetime and the number of repairs that were made. We estimate L by computing the mean lifetime over all runs. We estimate C by dividing the total number of repairs made over all runs by the total lifetime. Unless otherwise mentioned, the values of T, t and $\bar{t}$ are set to 1 month, 12 hours and 12 hours respectively.



(a) $r = 3$



(b) $r = 4$

Figure 3. Lifetime L as a function of α for $r = 3$ and $r = 4$. For small α, repair with memory produces superior lifetimes than without-memory repair. Note the difference in lifetimes when $r = 3$ vs. $r = 4$: a single additional replica results in an order-of-magnitude improvement in lifetime (regardless of strategy).

1) Basic behavior: Figure 3 shows the lifetime L as a function of α for $r = 3$ and $r = 4$, for both without-memory and with-memory repair. Intuitively, one might expect more aggressive repair, i.e., smaller α, to result in higher lifetime. This is certainly true in the case of with-memory repair; however, without-memory repair unexpectedly shows a peak in lifetime for α between 5 and 6.

This striking difference in behavior can be explained by a fundamental inefficiency in without-memory repair. When a replica that was prematurely timed out by the system subsequently returns, the system, if using memory, has the option of reusing this replica if necessary. Thus, the system can afford to be aggressive about repair, i.e., use a smaller α, without suffering any penalty despite that it inevitably makes more mistakes. Without using memory, the system does not have this option; it must discard this replica, even though retaining it would clearly be useful [13]. Beyond a point, the penalty associated with prematurely discarding replicas that are only offline outweighs the ability to quickly replace a replica that is actually dead. Thus, in the case

[13]This would be the case, for example, if all existing replicas were not-online.

of without-memory repair, using a smaller α is actually counter-productive.

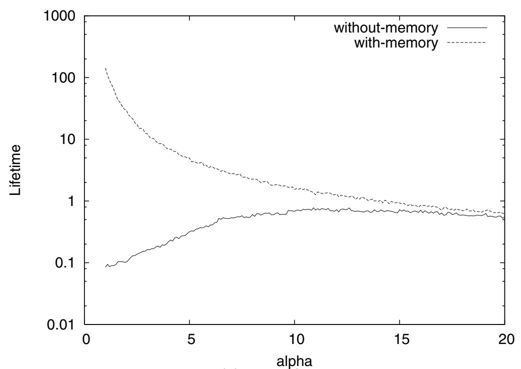
To further understand this behavior, we observe that in the case of without-memory repair, the critical condition that results in the loss of the object is if all r replicas are timed out within a period of $\alpha\bar{t}$. Note that since the system is forced to discard replicas that were prematurely timed out, it is irrelevant whether this timeout occurred because the replica was in state 3 (dead), or state 2 (offline). On the other hand, in the case of with-memory repair, the corresponding condition is if all r replicas transition to state 3 (dead) within a period of $\alpha\bar{t}$. Note that a timeout in state 2 cannot affect lifetime, as the system has the option to "correct" its mistake when the timed-out replica eventually returns. The probability that a replica goes to state 3 from state 1 is p_{13}, which is independent of α. Therefore, in the case of with-memory repair, L is influenced only by increasing the timeout period, which increases the chance of concurrent timeouts, and therefore decreases lifetime.

In the case of without-memory repair, the probability of a timeout in either state 2 or state 3 is $p_{13}+(1-p_{13})e^{-\alpha}$. Thus, when α is small, even though the window for concurrent timeouts is small, what dominates is that the probability of a replica timing out is high, resulting in smaller lifetime. As α increases, this probability drops and lifetime increases. Finally, when α is large enough, i.e., $(1-p_{13})e^{-\alpha}$ is small compared to p_{13}, lifetime decreases, and without-memory and with-memory repair show practically identical behavior. Not coincidentally, the peak in the without-memory lifetime curve corresponds to the knee of the curve in Figure 1(a). *This is significant, as we conjecture that it allows prediction of the timeout value that maximizes the lifetime of a without-memory multi-node system by simply analyzing the behavior of a single node.*

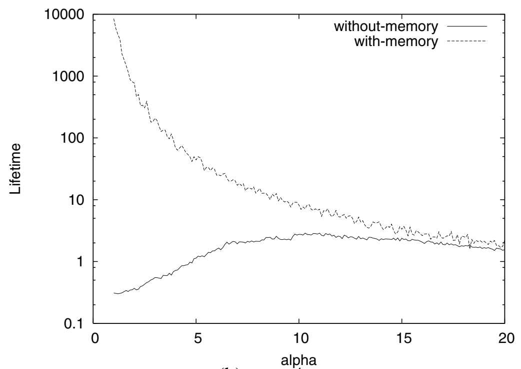
When using the PlanetLab empirically-derived distributions to drive the model, the corresponding lifetime curves for without-memory and with-memory repair are shown in Figure 4. As can be observed, the respective with-memory and without-memory curves have similar shapes to those in Figure 3. What is different again are the time scales; in Figure 4, the peak in lifetime for without-memory repair occurs between 10 and 12. The difference is a result of the heavier tails in the empirically-derived distributions. Despite this difference, what is important is that the peak is similarly predicted by the knee of the curve in Figure 1(b).

In terms of lifetime, using memory is superior to repair without memory. However, this difference is significant only for small α; for larger α, the lifetimes are roughly comparable. The question of whether it is worthwhile to use memory can be answered by looking at cost as well.

Figure 5 shows cost as a function of α for both without-memory and with-memory repair. In addition, the simulated cost is compared with the bound given by Equation 12. Our first observation is that, for a given (r, α), cost is roughly the



(a) $r = 3$



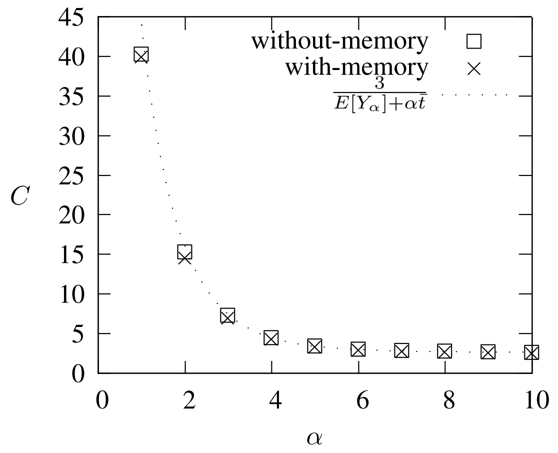
(b) $r = 4$

Figure 4. Lifetimes resulting from PlanetLab empirically-derived distributions. Note the similarity in the shapes of these curves to those of Figure 3.

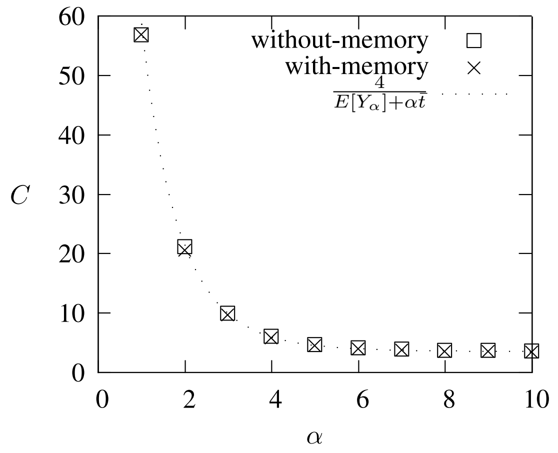
same whether the system uses memory or not. The cost is also close to the bound, indicating that the bound is indeed tight. For small α, cost may be very high; for example, for $r = 3$ and $\alpha = 2$, the system makes approximately 15 copies of the object every node lifetime, compared to the optimal value of 1. For such small α, using memory is significantly better in terms of lifetime. For larger α, cost is more reasonable; for example, when $\alpha = 6$, the system makes only 3 copies every node lifetime. However, for this value of α, the benefits of using memory are not great, the lifetimes being roughly comparable. Thus, although using memory is indeed superior, the benefits are realized only at the price of higher cost.

Figure 6 shows lifetime as a function of r for $\alpha = 6$. Since the Y-axis is logscale, this indicates an exponential dependence of lifetime on the number of replicas. This is expected; the probability of r timeout events occurring within the same window of time goes down exponentially with r. Figure 6 has strong predictive value; since the graph is linear, it is easy to predict the lifetime for a larger value of r when simulation becomes infeasible.

2) Impact of availability: In this section, we study the impact of node availability on object lifetime. Recall that node availability is the fraction of time a node is up, given by $p = \frac{t}{t+\bar{t}}$. We simulate system behavior for three different



(a) $r = 3$



(b) $r = 4$

Figure 5. Cost C as a function of α for r=3 and r=4. Cost varies with α in the same way for both without-memory and with-memory repair strategies. Furthermore, they closely follow the bound given by Equation 12. For small α, cost is high for both strategies. As α increases, C quickly decays and then stabilizes to a region of α where Figure 3 shows there is little difference in resulting lifetimes between without-memory and with-memory strategies.

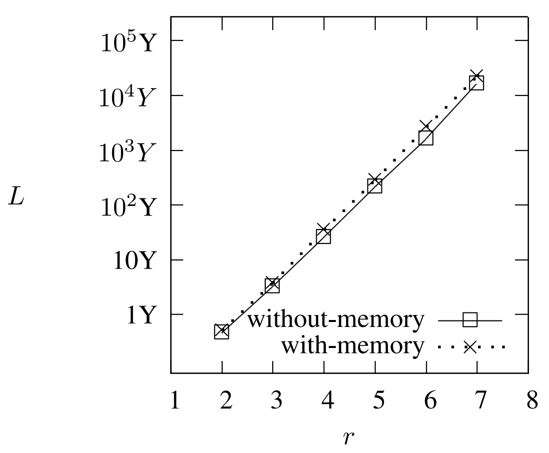


Figure 6. Lifetime as a function of r for $\alpha = 6$. Since the Y-axis is logscale, lifetime is exponential with increasing number of replicas.

values of availability, keeping both node lifetime T and cycle time $t + \bar{t}$ fixed.

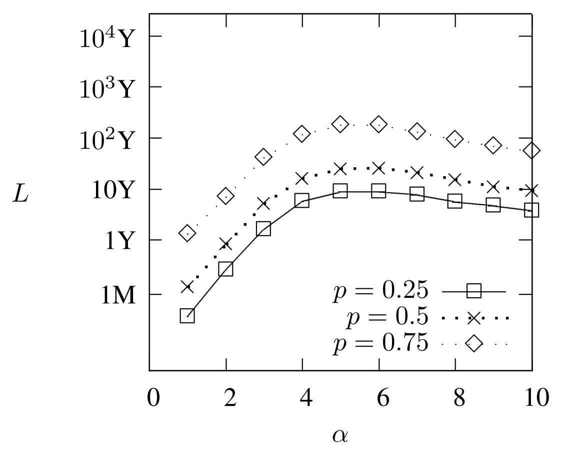
Figure 7 shows lifetime as a function of α for $r = 4$ in the case of both without-memory and with-memory repair. Our first observation is that node availability does not significantly change the fundamental shape of the lifetime curves. As can be seen from Figure 7, the shapes of the curves are similar for all three availabilities. However, availability does have a dramatic effect on lifetime. At $p = 0.75$, lifetimes are about an order of magnitude greater than at $p = 0.5$ for all α.

The effect of availability on lifetime can be explained as follows. When a node has poor availability, its average downtime $\bar{t}$ is larger. For a fixed α, this implies that the timeout value is correspondingly larger for a node with lower availability (because the timeout value equals $\alpha\bar{t}$). A longer timeout value increases the window of time in which concurrent timeouts have to occur to cause loss of the object, thereby decreasing lifetime. As can be seen, increasing the node availability from $p = 0.5$ to 0.75 results in a larger increase in lifetime than increasing node availability from $p = 0.25$ to 0.5. This is because, as the "window size" gets bigger (i.e., as availability decreases), the rate of increase of the probability of concurrent failures gets smaller.

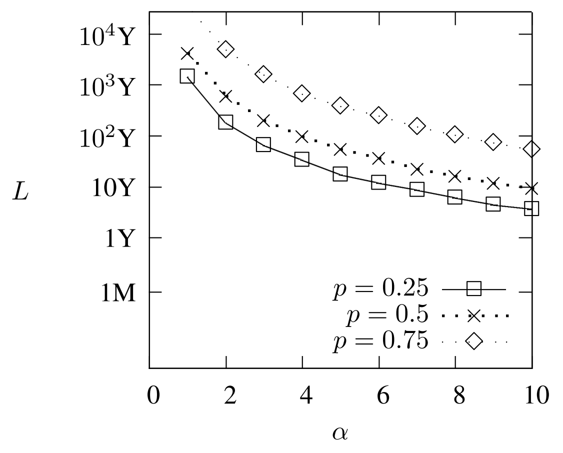
In contrast to the effect of availability on lifetime, cost seems fairly insensitive to availability, as shown in Figure 8. Cost is identical (and varies identically with α) for all three chosen values of p.

The effect of availability on lifetime and cost has implications for the design of such replicated systems. Clearly, given a choice, the system should choose to replicate on nodes with higher availability, as this results in higher lifetime at no additional cost. In fact, choosing nodes with higher availability may even be preferable to increasing the number of replicas r. Both result in higher lifetimes, but increasing r increases the cost of repair as well.

3) Distribution of lifetime: Up to this point, we have only considered the expected value of object lifetime L; we now consider how it is distributed. Figure 9 shows the cumulative distribution function for lifetime for $r = 4$ replicas and $\alpha = 6$. As can be seen, for both without-memory and with-memory repair, the distribution of lifetime is quite skewed. For example, even though the mean lifetime in the case of without-memory repair is greater than 25 years, there is a 19% chance that the object will be permanently lost in less than 5 years, and a 4.5% chance it will be lost in less than 1 year. If the system uses memory, these numbers are 13.4% and 2.6% respectively. Thus, if the system is required to guarantee lifetime with several nines of certainty, it is

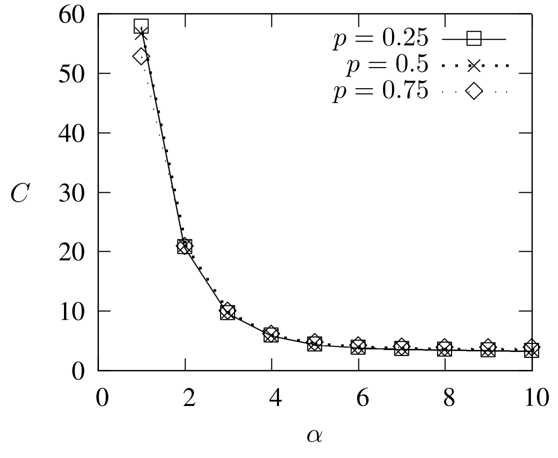


(a) Without-memory

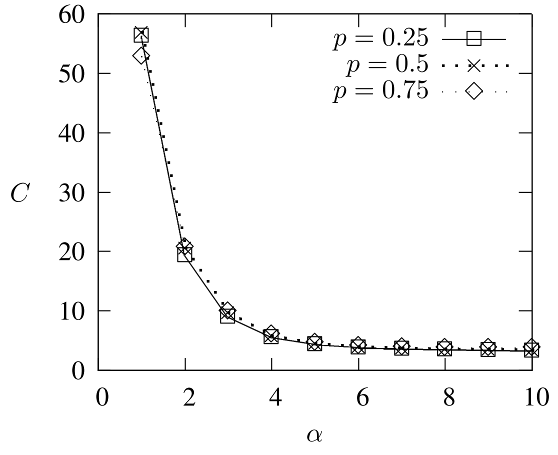


(b) With-memory

Figure 7. Object lifetime as a function of α for $r = 4$ and different node availabilities.



(a) Without-memory



(b) With-memory

Figure 8. Cost as a function of α for $r = 4$ and different node availabilities.

necessary to choose a replication strategy that results in an expected lifetime that may be many orders of magnitude larger.

Providing precise probabilistic guarantees on lifetime is problematic; it requires us to know the exact distribution of lifetime. As far as we know, this is analytically intractable; however, we have empirically determined that lifetime can be approximated by an exponential distribution with the same mean. For example, in the above case of $r = 4$ and $\alpha = 6$, we carried out a Chi-Square goodness of fit test to evaluate the hypothesis that the without-memory and with-memory data sets fit exponential distributions with means 25.4 and 35.8 (years), respectively. The computed chi-square statistics are 13.9 and 10.4, respectively, both of which are below the critical value of 15.5 (0.05 significance level, 8 degrees of freedom) to accept the hypothesis.

Modeling the lifetime by an exponential random variable allows us to make probabilistic guarantees on lifetime, given its expected value. The expected value itself can be obtained either by simulation for small values of r, or by extrapolation (using Figure 6 for example) for larger values of r.

4) Choosing r and α: In the previous sections, we have studied lifetime and cost as a function of the system parameters r and α. In this section, we will consider the question of how can the system maximize lifetime when given a certain amount of resource. Specifically, given a constraint on repair cost C, what is the optimal choice of r and α the maximizes lifetime L?

To obtain an answer to this question, we plot lifetime as a function of r for a constant value of cost. Given a cost, and the number of replicas r, we determine the value of α that produces the required cost using Equation 12 [14]. Naturally, the greater the number of replicas r, the larger

[14]This procedure relies on the fact that Equation 12 appears to be tight. We subsequently verify that the cost obtained by simulation is close to the target cost.

α is required to be to produce the same cost. We then obtain the corresponding lifetime for this (r, α) pair through simulation. The value of cost we use ranges from 3 to 6; as noted earlier in the paper, the desired cost regime is a small multiple of the optimal cost of 1. The results are shown in Figure 10.

Figure 10 indicates that there is an optimal value for r, (and therefore a correspondingly optimal value of α). For both without-memory and with-memory repair, the optimal value of r is in fact C, where C is the constraint on cost. For example, when the cost is limited to 4 replicas per node lifetime, lifetime peaks at $r = 4$. [15] The optimal value of r is precisely the number of replicas the system is budgeted to create every node lifetime. In other words, there is no benefit in using either more replicas (and less aggressive repair) or less replicas (and more aggressive repair) than the system can afford to create every node lifetime. Therefore, the corresponding optimal value of α is approximately given by $\frac{T}{E[Y_\alpha]+\alpha \bar{t}} = 1$, i.e., the per-replica repair cost (Figure 2) is 1. In other words, the timeout value that causes the mean time to timeout to be 1 for a single node is the timeout value that all nodes should use in a multi-node system to achieve a peak lifetime given a constraint on cost.

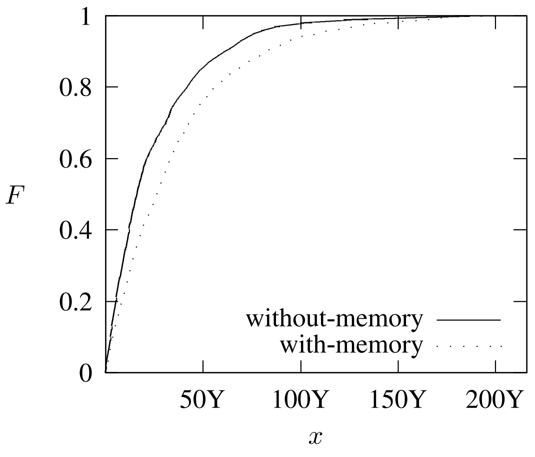


Figure 9. Cumulative distribution function of lifetime for $r = 4$ and $\alpha = 6$.

IV. Related Work

Our work is closely related to two distinct bodies of research: reliability research in the distributed systems community, and system reliability theory in the performance and modeling community. In recent years, the peer-to-peer model has emerged as a paradigm for building large-scale self-managing distributed systems. This has led to several efforts to build highly available distributed storage systems based on this paradigm; prominent among these are CFS [8], PAST [13], TRFS [2], [3], Oceanstore [9], [12], and Farsite [1], [5]. These systems use replication and/or erasure coding to primarily ensure high availability; more recent work such such as Carbonite [7] and Antiquity [15] considers durability, which is the focus of this paper.

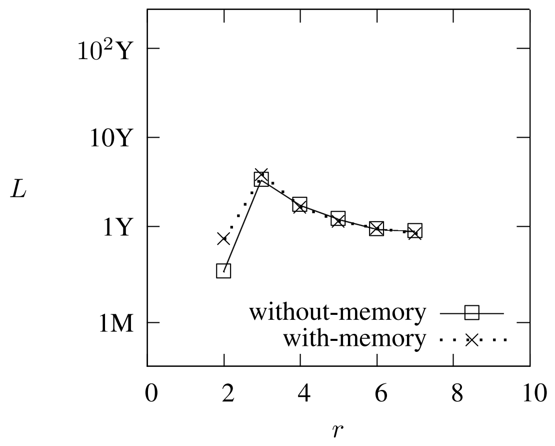
[15]Note that in the case C is not an integer, the optimal number of replicas (which is an integer) may be determined by inspecting whether lifetime is maximized at the nearest integer below C or the nearest above.

The use of a timeout to mask transient failures is proposed in [4], in which the authors argue that cross-system bandwidth limitations make it impossible to implement reliable storage over a unreliable peer-to-peer system. Their focus was on guaranteeing high availability, which requires a much higher level of replication than durability, as also observed in [7]. In [14], the authors assess the impact of churn on object maintainance strategies. In [7], the authors propose a replica maintainance protocol called Carbonite. Based on their experience, they conclude that re-integrating replicas that return after transient failures is key to ensuring low overhead. This is in apparent conflict with our observation that at low cost levels, using memory does not have significant advantages, in terms of cost or repair, over without-memory repair. The key observation is that unlike Carbonite, in our model, we did not allow the number of replicas to go above r, even in the case of with-memory repair. When we considered alternate strategies that made more aggressive use of memory, we did observe significantly higher lifetimes and lower costs; however, these are likely the result of having more replicas in the system, rather than the effect of memory, per se. Finally in [10] we evaluate expected object lifetimes in a distributed storage system for more general notions of repair; a similar model is also considered in [15].

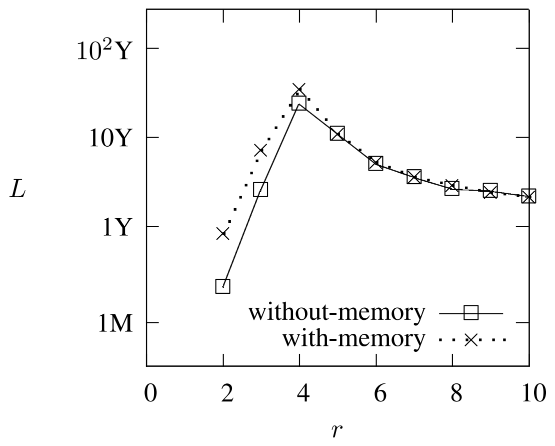
This work is also related to work in system reliability theory [11], which attempts to model the failure properties of multi-component systems. For example, the seminal work of Brown [6] deals with analyzing the time-to-failure of a parallel system of components with repair. However, the key difference in this paper is that our model incorporates several features that are unique to the domain of distributed storage systems, and distinguish it from existing models of repairable systems. For example, the model described in Section II takes into account not only the long-term participation of a node in the system, but also its limited availability. As a result, we are able to incorporate the notion of *uncertainity* in the repair process, wherein the system cannot determine precisely when a repair is necessary. Rather, it can only decide when a repair is likely to be required. Furthermore, unlike other repair models, the process of repair is not performed by an external agent; rather, repair is tied to the availability of other replicas in the system.

V. Conclusion

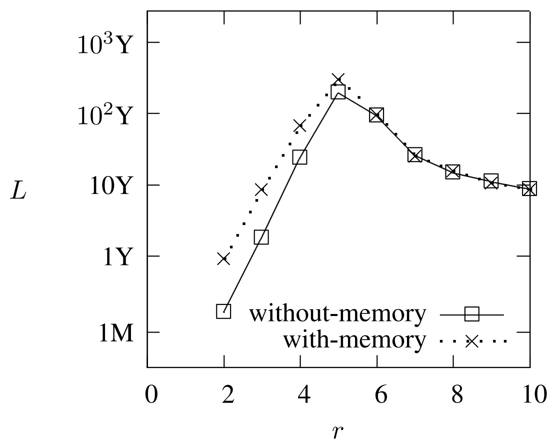
In this paper, we characterized data durability in distributed storage systems based on replication and repair. Using a stochastic model, we determined how object lifetime depends on the degree of replication and rate of repair, and



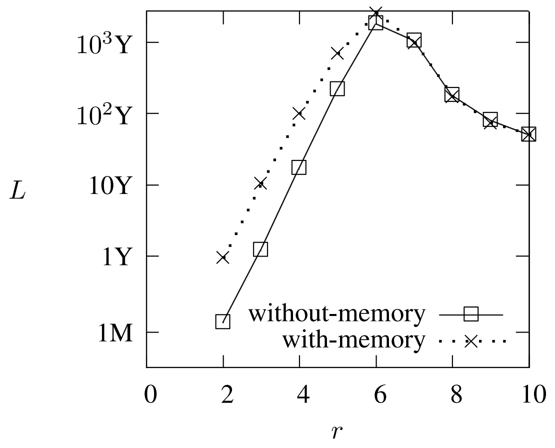
(a) $C = 3$



(b) $C = 4$



(c) $C = 5$



(d) $C = 6$

Figure 10. Lifetime as a function of r keeping C constant. For both without-memory and with-memory repair, there is clearly an optimal value for r that maximizes lifetime (and therefore a correspondingly optimal value of α), which in fact coincides with the value of cost C.

how lifetime is maximized when there is a constraint on resources. Our model uses a timeout mechanism to explicitly account for the inherent uncertainty a system has in whether "failures" are temporary (a replica is simply offline, but will eventually return) or permanent.

Our main conclusions are as follows:

- There is a threshold for cost below which without-memory repair is nearly as good as with-memory repair. On the other hand, if cost is not an issue, then using memory is superior to without-memory repair.
- When there is a constraint on cost, there is an optimal number of replicas – more or less will only degrade lifetime – and this is exactly the number of replicas that the system can create per node lifetime without violating the cost budget. For the correspondingly optimal α, with-memory repair is not signficantly better than without-memory repair.
- Availability is strongly correlated with lifetime. As availability goes up, the corresponding increase in lifetime is dramatic. However, higher availability does not affect the long-term cost of repair.

References

[1] Adya, A., Bolosky, W., Castro, M., Cermak, G., Chaiken, R., Douceur, J., Howell, J., Lorch, J., Theimer, M., and Wattenhofer, R. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. *In Proceedings of USENIX OSDI* (2002).

[2] Bhagwan, R., Savage, S., and Voelker, G. Understanding availability. *In Proceedings of IPTPS* (2003).

[3] Bhagwan, R., Tati, K., Cheng, Y., Savage, S., and Voelker, G. Total recall: System support for automated availability management. *In Proceedings of USENIX NSDI* (2004).

[4] Blake, C., and Rodrigues, R. High availability, scalable storage, dynamic peer networks: Pick two. *In Proceedings of HotOS IX* (2003).

[5] BOLOSKY, W., DOUCEUR, J., ELY, D., AND THEIMER, M. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. *In Proceedings of ACM SIGMETRICS* (2000).

[6] BROWN, M. The first passage time distribution for a parallel exponential system with repair. *Reliability and Fault Tree Analysis* (1974).

[7] CHUN, B., DABEK, F., HAEBERLEN, A., SIT, E., WEATHERSPOON, H., KAASHOEK, F., KUBIATOWICZ, J., AND MORRIS, R. Efficient replica maintenance for distributed storage systems. *In Proceedings of USENIX NSDI* (2006).

[8] DABEK, F., KAASHOEK, F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with cfs. *In Proceedings of ACM SOSP* (2001).

[9] KUBIATOWICZ, J., BINDEL, D., CHEN, Y., CZERWINSK, S., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., WELLS, C., AND ZHAO, B. Oceanstore: An architecture for global-scale persistent storage. *In Proceedings of ACM ASPLOS* (2000).

[10] RAMABHADRAN, S., AND PASQUALE, J. Analysis of long running replicated systems. *In Proceedings of IEEE INFOCOM* (2006).

[11] RAUSAND, M., AND HOYLAND, A. *System Reliability Theory: Models, Statistical Methods, and Applications*. John Wiley & Sons, 2004.

[12] RHEA, S., EATON, P., GEELS, D., WEATHERSPOON, H., ZHAO, B., AND KUBIATOWICZ, J. Pond: the oceanstore prototype. *In Proceedings of USENIX FAST* (2003).

[13] ROWSTRON, A., AND DRUSCHEL, P. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. *In Proceedings of ACM SOSP* (2001).

[14] TATI, K., AND VOELKER, G. On object maintenance in peer-to-peer systems. *In Proceedings of IPTPS* (2006).

[15] WEATHERSPOON, H. Design and evaluation of distributed wide-area on-line archival storage systems. *University of California Berkeley Ph.D. Dissertation* (2006).

Appendix

If $T \gg \bar{t}$, $Y \sim$ Exponential $\left(\frac{1}{T}\right)$.

Proof: Equation 3 implies that the p.d.f. f_Y of Y is the convolution of $f_{X+\bar{X}}$ N times and f_X, where $N \sim$ Geometric (p_{13}). Therefore the Laplace transform [16] of f_Y is given by

$$\begin{aligned}
\mathcal{L}\{f_Y\} &= \frac{p_{13}}{1-(1-p_{13})\,\mathcal{L}\{f_{X+\bar{X}}\}}\,\mathcal{L}\{f_X\} \\
&= \frac{p_{13}}{1-(1-p_{13})\,\frac{1}{1+st}\,\frac{1}{1+s\bar{t}}}\,\frac{1}{1+st} \\
&= \frac{p_{13}\,(1+s\bar{t})}{(1+st)\,(1+s\bar{t})-(1-p_{13})} \\
&= \frac{p_{13}\,(1+s\bar{t})}{s^2 t\bar{t}+s\,(t+\bar{t})+p_{13}}
\end{aligned}$$

Substituting $p_{13} = \frac{\lambda_{13}}{\lambda_{12}+\lambda_{13}} = \frac{t}{pT}$ gives

$$\mathcal{L}(f_Y) = \frac{1+s\bar{t}}{s^2 p\bar{t}T+sT+1}$$

Splitting into partial fractions,

$$\mathcal{L}(f_Y) = \frac{c_1}{1+s\,\tau_1}+\frac{c_2}{1+s\,\tau_2}$$

and inverting the transform gives

$$f_Y(x) = c_1.\frac{1}{\tau_1}\,e^{-\frac{x}{\tau_1}}+c_2.\frac{1}{\tau_2}\,e^{-\frac{x}{\tau_2}} \qquad (14)$$

Thus, f_Y is a mixture of two exponential distributions with rates $\frac{1}{\tau_1}$ and $\frac{1}{\tau_2}$ respectively. Now

$$\begin{aligned}
\tau_1+\tau_2 &= T \\
\tau_1\,\tau_2 &= p\,\bar{t}\,T
\end{aligned}$$

which implies that τ_1 and τ_2 are roots of the quadratic equation equation $\tau^2 - T\tau + p\,\bar{t}\,T = 0$, given by $\frac{T}{2}\left(1 \pm \sqrt{1-\frac{4p\bar{t}}{T}}\right)$. If $T \gg \bar{t}$,

$$\begin{aligned}
\tau_1 &= \frac{T}{2}\left(1+\sqrt{1-\frac{4p\bar{t}}{T}}\right) \\
&\approx T \\
\tau_2 &= \frac{T}{2}\left(1-\sqrt{1-\frac{4p\bar{t}}{T}}\right) \\
&\approx \frac{T}{2}\left(1-1+\frac{2p\bar{t}}{T}\right) \\
&= p\,\bar{t}
\end{aligned}$$

In addition,

$$\begin{aligned}
c_1+c_2 &= 1 \\
\tau_2\,c_1+\tau_1\,c_2 &= \bar{t}
\end{aligned}$$

Solving for c_1 and c_2, and substituting for τ_1 and τ_2,

$$\begin{aligned}
c_1 &= \frac{1-\frac{\bar{t}}{T}}{1-\frac{p\bar{t}}{T}} \\
&\approx 1
\end{aligned}$$

Thus, $f_Y(x) \approx \frac{1}{T}e^{-\frac{x}{T}}$, i.e., $Y \sim$ Exponential $\left(\frac{1}{T}\right)$.

[16] The Laplace transform $\mathcal{L}\{f\}$ of a function $f(x)$ is defined as $F(s) = E\left[e^{-sx}\right] = \int_0^{\infty} e^{-sx}\,f(x)\,dx$.