Replica Placement Strategies for Wide-Area Storage Systems

Byung-Gon Chun and Hakim Weatherspoon
Computer Science Division, University of California, Berkeley
bgchun@cs.berkeley.edu and hweather@cs.berkeley.edu

Abstract

Wide-area durable storage systems trigger data recovery to maintain target data availability levels. Data recovery needs to be triggered when storage nodes permanently fail, i.e., data is lost. Transient failures, where nodes return from failure with data, add noise to determining that a node has failed permanently. Replica placement strategies maintain data availability while minimizing cost, wide-area bandwidth. Correlated failures can compromise data availability. We demonstrate that correlated downtimes exist and that simple replica placement strategies can overcome the correlation and maintain target data availability levels.

1 Introduction

Abstractly, wide-area storage systems are as simple as their local-area counterparts: they replicate data to reduce risk of data loss and replace lost redundancy when they have detected permanent failures. Permanent failure is loss of data on a node. Transient failure is when a node becomes temporarily unavailable and comes back with its data intact. Wide-area storage systems are required to continuously maintain data availability\(^1\) as nodes fail (permanent or transient). Data maintenance\(^2\) is the process of copying replicas from one node to another to ensure data availability.

\(^1\)Continuously maintaining data availability is not necessary but is sufficient to maintain data durability.
\(^2\)We use data maintenance and data recovery interchangeably.

A fundamental issue for any wide-area storage system is replica placement. Replica placement is the process in which nodes are selected to store data replicas. The goal of replica placement is to efficiently maintain a target data availability. Efficiency means using minimal wide-area bandwidth. There are two categories of replica placement: random and selective. Random placement assumes node failures are independent or independent enough. If node failures are correlated, the end result could lead to a reduced data availability below the target. In contrast, selective placement chooses nodes that are specifically chosen to have low correlation and long expected remaining lifetime.

In this paper, we analyze a wide-area system for correlated failures. We compare different replica placement strategies to best place replicas that will simultaneously maintain data availability, avoid correlated nodes, and reduce cost (we focus on wide-area bandwidth). The contribution of this work is to combine the analysis of correlated failures, replica placement, and the cost of maintaining data availability.

The rest of the paper is organized as follows. In Section 2, we discuss, at a high level, how a wide-area storage system maintains data. We analyze correlated failures in a wide-area system in Section 3. Based on the correlation analysis, we suggest some replica placement strategies for comparison in Section 4. In Section 5 and 6, we evaluate and compare the proposed replica placement strategies. Finally, we conclude in Section 7.
2 Wide-Area Storage System Overview

In this section, we present an overview of the requirements and nomenclature of a wide-area storage system.

We assume that data is maintained on nodes, in the wide area, and in well maintained sites. Sites contribute resources such as node (storage, cpu, ram) and network bandwidth. Nodes collectively maintain data availability. That is they are self-organizing, self-maintaining, and adaptive to changes. In particular as nodes fail, data recovery is triggered, the process of copying data replicas to new nodes to maintain data availability.

There are five variables of concern when maintaining data in the wide-area. First, the redundancy type $m$. $m$ is the required number of components necessary to reconstruct the original data item. For example, $m = 1$ for replication. The minimum data availability threshold $th$ is the minimum number of replicas required to maintain the target data availability. That is, when the number of available replicas decrease below $th$, new replicas are created. $e$ is the extra replicas above the threshold. $e + 1$ replicas have to simultaneously be down for data recovery to be triggered. $n = th + e$ is the total number of replicas. When data recovery is triggered, enough new replicas are created to refresh the total number back to $n$. Finally, the timeout $to$ is used to determine when a node has failed.

For example, Figure 1 shows one object replicated ($m = 1$) into a wide area storage system. The minimum threshold number of replicas, $th = 7$, and there is one extra replica, $e = 1$. The total number of replicas is $n = 7 + 1 = 8$. Data recovery is triggered when 2 replicas ($e + 1$) are simultaneously down. In this example a replica in Georgia has permanently failed and a replica in Washington has transiently failed since a heartbeat has been lost. Data recovery is triggered because the number of remaining replicas is less then the threshold; that is, two new nodes are chosen and replicas are copied to the new nodes. Notice that the node in Washington could return from failure, in which case, recovery work would have been wasted creating a new replica. The problem is that it is not possible to determine a node failure as lost heartbeat (e.g. lossy link due to congestion, bad router configuration, etc), a node being transiently down (reboot), or a permanent failure (data on node removed permanently from the network). A timeout $to$ is the only tool available to a wide-area system to determine failure.

3 Correlated Failure Analysis

In this section we present our methodology of characterizing a wide-area storage system. First, we describe the wide-area environment that we analyzed. Second, we analyze the correlated failures in the wide-area environment.

3.1 PlanetLab Wide-Area Environment

We used the all-pairs ping data set [6] to characterize PlanetLab. The data set was collected over a period of 20 months, between Feb 16, 2003 to Oct 6, 2004 and included a total of 512 nodes in that time period (Table 1). We used the data set to characterize the sessiontime, downtime, and lifetime distributions. We begin by describing how the all-pairs ping data set was collected and how we used the data set to interpret a node as being available or not.

All-pairs ping data collects minimum, average, and maximum ping times (over 10 attempts) between all pairs of nodes in PlanetLab. Measurements were...
Issues and Limitations

The all-pairs ping data collection technique is vulnerable to experimental and measurement bias. First, there were instances where the all-pairs ping program had a bug and the process failed on almost all nodes simultaneously. Second, there were related instances where a majority of nodes were unusable due to high load. In both cases the all-pairs ping program did not return data for most nodes. We did not use these all-pairs ping instances since they were experimental error.

All-pairs ping does not measure permanent failure. Instead, all-pairs ping measures the availability of a node name (i.e. availability of an ip address) and not the availability of data on a node. As a result, all-pairs ping was used to produce an estimated upper bound on node lifetimes. In addition, we computed an estimated lower bound on the availability of data on a node by supplementing the trace with a disk failure distributions obtained from [5]. We used a technique described by Bolosky et al. [3] to estimate the expected node lifetime based on node attrition where the rate of attrition is constant. In particular, we counted the the number of remaining nodes that started before December 5, 2003 and permanently failed before July 1, 2004. The expected node lifetime of a PlanetLab node is 878 days (Table 2.(f)). Figure 2.(d) shows the node attrition.

No all-pairs ping data exist between December 17, 2003 and January 20, 2004 due to a near simultaneous compromise and upgrade of PlanetLab. In particular, Figure 2.(a) shows that 150 nodes existed on December 17, 2003 and 200 existed on January 20, 2004, but no ping data was collected in between the dates.

3.2 Correlated Failures

In this section, we test the PlanetLab data for the possibility of nodes with correlated failures. Figure 3 shows the different views of correlation between PlanetLab nodes. Figure 3.(a) shows the (total and unavailable) number of nodes per day vs time. Figure 3 demonstrates that during any daily interval, a number of failures (transient or permanent) occur. Later graphs show that many failures are correlated although most seem to be uncorrelated.

As far as we know, most studies that produce a correlation metric (i.e. probability that y is down given that x is down), only use the one-dimension;
that is, nodes that are chronically down can increase the number of correlated nodes. To capture node correlation not influenced by long downtimes, we use a two-dimensional space of conditional downtime probabilities, both $p(x \text{ is down} \mid y \text{ is down})$ and $p(y \text{ is down} \mid x \text{ is down})$. Figure 3.(b) and (c) show the two-dimensional conditional downtime probability, Figure 3.(c) is the upper right quadrant for Figure 3.(b). Both Figures 3.(b) and (c) highlight nodes that are in the same site with light open circles. The fraction of correlated nodes in both figures are highlighted in Table 3.(g). Table 3.(g) shows that 22% of the time that a node goes down, there is at least a 50% chance another node in the same site will go down as well. Alternatively, Table 3.(g) shows that the two-dimension conditional downtime probability of nodes in different sites is insignificant.

Next we look at the effects of nodes with long downtimes. Figure 3.(d) shows the total downtime for each node and orders the nodes from most to least total downtime. Of the 512 total PlanetLab nodes, 188 nodes have total downtimes greater than 1000 hours. In Figure 3.(e), we again show the two-dimension conditional downtime probability, but with nodes with total downtime longer than 14 days factored out. Figure 3.(e) shows more density along the diagonal, meaning that $p(x \text{ is down} \mid y \text{ is down})$ is more symmetric with $p(y \text{ is down} \mid x \text{ is down})$; that is, the nodes are not asymmetrically influenced by long downtimes. Similar to Table 3.(g), Table 3.(h) shows that 33% of the time that a node goes down, whose total downtime is less than 14 days, there is at least a 50% chance another node in the same site will go down. However, the inter-site two-dimension conditional downtime probability is still insignificant.

Finally, Figure 3.(f) shows the joint probability that $k$ nodes, picked randomly, simultaneously fail; that is, the probability of a catastrophic failure if there are $k$ replicas. Figure 3.(f) shows that the probability of a catastrophic failure decreases exponentially as $k$ increases. In particular, $k = 5$ is enough to make the joint probability that $k$ failures simultaneously occur 0.00001 (1 in 10,000). The key is
that as long as all replicas do not simultaneously fail, then the lost replicas can be reproduced from the remaining and data will not be lost.

4 Replica Placement Strategies

The analysis of correlated failures from Section 3.2 suggests that random, with replacement of duplicate site nodes, and a “blacklist” of nodes showing long downtimes, may be a good replica placement strategy. In order to validate this hypothesis, we compare four different replica placement strategies including random and four different “oracle” placement strategies.

The four replica placement strategies that we compare are Random, RandomBlacklist, RandomSite, and RandomSiteBlacklist. Random placement picks $n$ unique nodes at random to store replicas. RandomBlacklist placement is the same as Random but avoids the use of nodes that show long downtimes. The blacklist is comprised of the top $k$ nodes with the longest total downtimes. RandomSite avoids placing multiple replicas in the same site. RandomSite picks $n$ unique nodes at random and avoids using nodes in the same site. We identify a site by the 2B IP address prefix. The other criteria can be geography or domains. Finally, RandomSiteBlacklist placement is the combination of RandomSite and RandomBlacklist.

The “oracle” placement strategies use future knowledge of node lifetimes, downtimes, and availability to place replicas. In all, we define four different oracle algorithms. First, the Min-Max-TTR oracle places replicas on nodes that exhibited the minimum maximum future downtimes. Similarly, the second and third oracle, the Min-Mean-TTR and Min-Sum-TTR oracle, place replicas on nodes that exhibit the minimum mean and sum downtimes, respectively. The fourth oracle, Max-Lifetime-Avail, places replicas on nodes that permanently fail furthest in the future and exhibit the highest availability, which is the ratio of the total uptime and lifetime.

5 Evaluation Methodology

In this section we present our methodology of using a trace-driven simulation to compare and evaluate our proposed replica placement strategies.

The trace-driven simulation runs the entire 20 month trace of PlanetLab (Figure 2.(a)). The simulator added a node at the time a node was available in the trace and removed a node that was not available. That is, nodes that are not available in the trace are not available in the simulator (and visa versa). We implemented timeouts/heartbeats in the simulator to detect node failures. Data recovery was triggered when the extra replicas plus one, $e + 1$, were simultaneously considered down (description in Section 2). Recall that $e = n - th (n \geq th)$. The timeout used to detect failure was $to = 1hr$.

We set that target data availability to 6 9’s using equations described in [1, 2]. Data availability equals $1 - \epsilon = 1 - (1 - a)^{th}$, where $a$ is the average node availability, $th$ is the minimum data availability threshold, and $\epsilon$ is the probability that no replicas are available. The equality assumes that all nodes are independent. As a result, we calculated the minimum data availability threshold $th$ to be 9 for replication $m = 1$ and an average PlanetLab node availability $a = 0.822$.

The amount of data that we simulated maintaining was 10,000 unique objects or 2TB if each object was 200MB. The replication factor, $k = \frac{n}{m}$ increased the size of the repository. For example, with $n = 10$ and $m = 1$ the total size of the repository was 100,000 replicas and 20TB of redundant data.

We use the trace supplemented with the disk failure distribution to compare the rate of data recovery: We measured the total number of repairs triggered vs time and the bandwidth used per node vs time. Additionally, we measured the average bandwidth per node as we vary the timeout and extra replicas. Finally, we measured the average number of replicas available vs time, but we omit this graph since the number of replicas was actually at least the threshold.

6 Evaluation

In this section, we present results using replication $m = 1$ with the trace modified with disk failures.
Table 2: Comparison of Replica Placement Strategies

![Figure 4: Cost Tradeoffs for Maintaining Minimum Data Availability](image)

Figure 4: Cost Tradeoffs for Maintaining Minimum Data Availability for 10,000 200MB objects. (a) Number of Repairs vs. Total Number of replicas (greater than threshold \( th = 9 \)). (b) and (c) Average bandwidth per node vs Total Number of Replicas. (a) and (b) varies the replication placement strategy and fixes the timeout \( to = 1hr \). And (c) fixes the replication placement strategy to Random and varies the timeout.

We did produce results for different erasure code schemes (e.g. \( m = 4 \)), but we omit the graphs for space since the trends of using extra redundancy with erasure codes were the same.

### 6.1 Evaluation of Replica Placement Strategies

First, we compare different replication placement strategies. Table 2 shows the number of repairs and percentage of improvement over Random for the four different random placement and four different oracle placement strategies. Table 2.(a) and (b) use \( n = 10 \) and \( n = 15 \) respectively.

For \( n = 10 \) (Table 2.(a)), more sophisticated placement algorithms have marginal improvement over Random; for example, RandomSiteBlacklist shows a 3.37% improvement. The observation that three of the four oracles perform worse than Random indicate that more is happening beyond just placement. In the next section, we will investigate further by varying the number of extra replicas and timeouts. We also separate repairs triggered due to permanent failure (loss of data on a node) and transient failure (node returns from failure with data).

For \( n = 15 \) (Table 2.(b)), more sophisticated placement algorithms exhibit noticeable increase in performance; that is, fewer repairs triggered compared to Random. For example, the RandomSiteBlacklist placement shows a 7.25% improvement over Random, which is about the same as the sum of the parts 3.70% and 2.96% for RandomBlacklist and RandomSite respectively. The oracle placement algorithms exhibit a 14.89% to 17.75% improvement for the oracles that do not consider lifetime and 30.92% improvement for the oracle that considers lifetime.

The fact that number of absolute repairs decreases from \( n = 10 \) to \( n = 15 \) suggests that the system is responding to transient failures instead of placement and permanent failures. The improvement increases when \( n \) increases, and then it decreases as \( n \) further
increases. We will investigate further in the next section.

### 6.2 Evaluation of Extra Redundancy

In this section, we vary the extra redundancy (i.e. total replicas subtracted by the threshold $e = n - th$) to investigate the effects that transient failures influence triggering data recovery. Also, we continuously add data to the repository increasing the size of the repository over time. We use two different write rates 10Kbps and 1Kbps per node. To make the write rates concrete, we used the measurement that an average workstation creates 35MB/hr (or 10Kbps) [3] of total data and 3.5MB/hr of permanent data (or 1Kbps). Finally, 10Kbps and 1Kbps per node correspond to 40GB and 4GB of unique data per node per year added to the repository, respectively. With an average of 300 nodes, the system increases in size at a rate of 30TB and 3TB, respectively.

Figure 4 show the effects of varying the number of extra replicas and the timeout. Figure 4.(a) and (b) fix the timeout $to = 1$hr and vary the replica placement strategy; where as, Figure 4.(c) varies the timeout and fixes the replica placement strategy to Random.

Figure 4.(a) shows that the number of repairs triggered decreases exponentially as the number of extra replicas increases linearly. Correspondingly, the bandwidth required to maintain the minimum data availability threshold and extra replicas decreases as seen in Figure 4.(b). Figure 4.(c) shows that large timeout values exponentially decrease the cost of data recovery but potentially compromise data availability. Similarly, linear increase in extra replicas exponentially decreases the cost of data recovery without sacrificing data availability.

Figure 5 shows the breakdown in bandwidth cost for maintaining a minimum data availability. Figure 5 fixes both the timeout $to = 1$hr and replica placement strategy to Random. Figure 5.(a) and (b) used a per node unique write rate of 1Kbps and 10Kbps, respectively. Both Figures 5.(a) and (b) illustrate that the cost of maintaining data due to transient failures dominates the total cost. The total cost is dominated by unnecessary work. As the number of extra replicas ($e = n - th$), which are required to be simultaneously down in order to trigger data recovery, increases, the cost due to transient failures decreases, thus the cost due to actual permanent failures, which is a system fundamental characteristic, dominates. The difference between Figure 5.(a) and (b) is that the cost due to permanent failures dominates in (a) and the cost due to new writes dominates in (b). Finally, the cost due to probing each node once an hour is insignificant.

In summary, Figure 4 demonstrate that extra redundancy reduces the rate of triggering data recovery, thus reduces the cost of maintaining data. In particular, wide-area storage systems can tradeoff increased storage for decreased bandwidth consum-
tion and maintain the same minimum data availability threshold.

7 Conclusion

In this paper, we analyze the machine failure characteristics in wide area systems and we examine replica placement strategies for such systems. We found that there does exist correlated downtimes, especially significant correlated downtimes between nodes in the same site. As a placement strategy, random seems to be good enough to maintain data availability efficiently. In particular, the random placement with extra replicas and with constraints to avoid using same sites and unreliable nodes is sufficient to reduce the rate of triggering data recovery due to node failures and most observed correlation.

References


