Optimal Inter-Object Correlation When Replicating for Availability

Haifeng Yu
National University of Singapore
haifeng@comp.nus.edu.sg

Phillip B. Gibbons
Intel Research Pittsburgh
phillip.b.gibbons@intel.com

ABSTRACT

Data replication is a key technique for ensuring data availability. Traditionally, researchers have focused on the availability of individual objects, even though user-level tasks (called *operations*) typically request multiple objects. Our recent experimental study has shown that the *assignment* of object replicas to machines results in subtle yet dramatic effects on the availability of these operations, even though the availability of individual objects remains the same.

This paper is the first to approach the assignment problem from a theoretical perspective, and obtains a series of results regarding assignments that provide the best and the worst availability for user-level operations. We use a range of techniques to obtain our results, from standard combinatorial techniques and hill climbing methods to Janson's inequality (a strong probabilistic tool). Some of the results demonstrate that even quite simple versions of the assignment problem can have surprising answers.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*

General Terms

Algorithms, Design, Reliability, Theory

Keywords

Multi-object operation, inter-object correlation, availability, data replication, object assignment

1. INTRODUCTION

Masking failures is a key goal in distributed computing, and data replication is a well-known and widely-used technique to ensure data availability in the presence of failures. Traditionally, researchers typically focus on the availability of individual data objects (e.g., individual file blocks [4] or individual variable-sized objects [7, 15]). On the other hand, a user-level task often needs to request multiple data objects; we refer to this as a *multi-object operation*. For example, in order to compile a project, all source files need to be available. Similarly, a database query usually touches multiple database objects. Our recent experimental study [26] shows that the *assignment* of object replicas to machines has a subtle yet critical effect on the availability of such multi-object operations, even though the availability of individual objects remains the same.

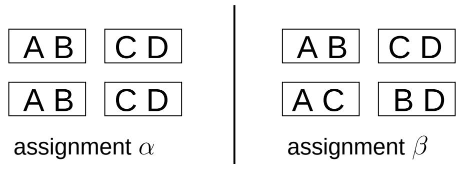

Figure 1: Two possible assignments of four objects, A, B, C, and D, to four machines. Each box represents a machine.

A simple yet subtle example. Consider the example in Figure 1 with 4 objects: A, B, C, and D. Each object has exactly 2 replicas[1]. We have 4 identical machines to hold these 8 object replicas, and each machine holds exactly 2 objects. Each machine may fail (crash) independently with the same probability p, causing all its data to become unavailable. An object is unavailable if and only if both its replicas are unavailable. Clearly, there are many ways to assign the object replicas to the machines. Figure 1 gives two possible assignments.

Imagine that the four objects are source files of a project and the user is trying to compile the project. Here, if any source file is unavailable, the multi-object operation (i.e., the compilation process) will fail. Which assignments in Figure 1 gives us a better probability that all four files are available so that the operation succeeds? A calculation will show that assignment α provides better availability than β.[2]

Assume now that, instead, the four objects are database objects that have numerical values. Our multi-object operation intends to compute the average of the values. Suppose we are willing to tolerate some error in the average, and the operation is considered successful as long as we can retrieve three or more objects. Under this assumption, assignment β now provides better availability than α.[3]

There are several important observations from this example. First, individual objects have the same availability (i.e., $1-p^2$) in α as in β. Also, the 8 object replicas occupy the same number of machines

[1]The same assignment problem also arises [26] when using erasure coding for the objects, but that is beyond the scope of this paper.
[2]The failure probabilities are $FP(\alpha) = p^4 + 4p^3(1-p) + 2p^2(1-p)^2$ and $FP(\beta) = p^4 + 4p^3(1-p) + 4p^2(1-p)^2$.
[3]The failure probabilities (i.e., the probabilities that fewer than three objects are available) are $FP(\alpha) = p^4 + 4p^3(1-p) + 2p^2(1-p)^2$ and $FP(\beta) = p^4 + 4p^3(1-p)$.

in α as in β, thus the difference does not result from "concentrating" or "spreading" the objects.

Second, all machines are completely identical, so for any *individual* object, it does not matter to which two machines the object is assigned. This clearly distinguishes the assignment problem from classic replica placement problems [3, 5, 27]. The difference between α and β arises purely from *inter-object correlation*: Because each machine has multiple objects, object failures are correlated even when machine failures are independent. For example, A is fully correlated with one object (i.e., B) in α, while A is partially correlated with two objects (i.e., B and C) in β. However, because neither the correlation in α nor the correlation in β is strictly stronger than the other, comparing the two assignments is subtle and no single numerical value can be used to summarize/rank such correlations in the general case.

Practical importance. The object assignment problem has significant practical relevance, and applies to almost all replication systems. A long list of previous replication systems and protocols (such as CAN [16], CFS [4], Chord [20], Coda [13], FARSITE [1], GFS [6], GHT [17], Glacier [7], Pastry [18], R-CHash [12], and RIO [19]) use different object assignments, which can yield dramatically different availability for multi-object operations. For example, it has been shown [26] that under practical settings, the failure probability of the TPC benchmark [23] can vary by four orders of magnitude under different assignments used in these previous system. Thus, a thorough understanding of the subject is critical to guide system design.

Previous results. We recently first identified [26] the availability effects of object assignments and inter-object correlation on multi-object operations. This earlier work used simulation to compare several specific assignments, including the PTN and RAND assignments. PTN is the assignment where we partition the objects into sets and mirror each set across multiple machines (as in assignment α in Figure 1). RAND is obtained by randomly assigning object replicas to machines. The simulation results [26] show that: (1) previously proposed assignments can result in dramatically different availability; (2) if the multi-object operation cannot tolerate any missing objects, then PTN and RAND provide the best and the worst availability among the set of assignments simulated, respectively; and (3) in contrast, if the multi-object operation can tolerate a sufficient number of missing objects, then PTN and RAND provide the worst and the best availability among the set of assignments simulated, respectively. Our earlier work [26] further proposed designs to approximate PTN and RAND for dynamic contexts, and performed a thorough evaluation of their implementation on the PlanetLab. We are not aware of any other previous work on this topic.

Our results. To the best of our knowledge, this paper is the first to study this problem from a theoretical perspective. Experimental methods as in [26] have the following fundamental limitations: Given the exponential number of possible assignments, it is infeasible to experiment with them all. Are there better assignments that were overlooked? Also, experimental methods can cover only specific parameter values (e.g., specific p values)—will the same conclusions hold under other parameter values? The theoretical results in this paper not only provide a deep understanding of the problem, but also help to ultimately confirm what was observed experimentally.

Our goal in this paper is to find the best and the worst[4] assignments, among all possible assignments, in terms of the availability provided to multi-object operations. Achieving our goal, however, is challenging for several reasons. First, human intuition often fails here, even when the problem appears quite simple on the surface. The example in Figure 1 already gave us some flavor of this, and later in Section 7.2 we will show that the problem can become considerably more intriguing. Second, natural approaches to solve the problem (such as constrained optimization and hill-climbing methods) do not work out well. To address these challenges, this paper leverages Janson's inequality [2, 9, 11] to obtain results for the most general cases. Combinatorial techniques and hill climbing are then used to prove stronger results under more restricted settings. Our final results are clean and simple:

- Calculating the availability of an arbitrary given assignment is #P-hard.
- If the multi-object operation cannot tolerate any missing objects, then PTN and RAND provide the best and the worst (within small constants) availability among all possible assignments, respectively.
- In contrast, if the multi-object operation can tolerate a sufficient number of missing objects, then PTN and RAND provide the worst and the best (within small constants) availability among all possible assignments, respectively.
- Under some restricted settings, we are able to construct the best and worst assignments and remove all constants.
- It is impossible for any single assignment strategy to achieve the best of both PTN and RAND.

2. RELATED WORK

On the surface, object assignment is related to the classic replica placement problem. Replica placement has been extensively studied for both performance and availability targets. Replica placement research for availability [3, 5, 27] typically considers the availability of individual objects rather than multi-object operations. Such results cannot be easily extended to our context because the two problems are fundamentally different: Replica placement problems [3, 5, 27] stem from the heterogeneity of machines (e.g., different failure probabilities). In contrast, because of inter-object correlation, object assignment affects availability even when all machines are identical (as shown in Figure 1). Chain replication [25] investigates the availability of *individual* objects where the system creates additional replicas to compensate for lost data. Here, if the repair bandwidth is limited, different placements will result in different repair times and thus different availabilities for *individual* objects. Such effects are orthogonal to the inter-object correlation effect observed by multi-object operations.

3. FORMAL MODEL AND ASSUMPTIONS

There are N data *objects* in the system, where an object can be a file block, a file, a database tuple, a group of database tuples, an image, etc. (See Table 1.) Each object has k replicas, and the object is considered *available* as long as any of the k replicas is available. There are s machines in the system, each of which may independently experience crash (benign) failures with probability p. This paper assumes $p \leq 0.5$, because this is the common case in practice. For the purpose of load balancing, each machine holds

[4]We are interested in the worst assignment as well because (i) previous simulation results [26] suggest the best and the worst may flip when the "tolerance" level of the operation changes, (ii) knowing the worst helps guide us to avoid it, and (iii) it is also of theoretical interest.

N	number of objects in the system
k	number of replicas per object
l	number of objects on each machine
s	number of machines in the system ($= Nk/l$)
p	failure probability of each machine
n	number of objects requested by an operation
t	number of objects needed for the operation to succeed (out of the n objects)

Table 1: Notation used in this paper, where $N > l \geq k \geq 2$.

the same number of $l = kN/s$ objects. To rule out trivial and uninteresting scenarios, we assume that $N > l \geq k \geq 2$. If a machine fails, all l object replicas on it become unavailable.

A *multi-object operation* (or *operation* in short) requests (for reading and/or writing) a specific subset of n objects (out of the N objects) in order to perform a certain user-level task. In some application scenarios such as image databases [8, 22], the number of objects requested can reach thousands or more. If not all n objects requested by the operation are available, the operation may or may not be considered successful, depending on its tolerance for missing objects. This paper studies the *threshold* criteria: an operation is *successful* if and only if at least t out of the n objects are available. Here t is a value from 1 to n depending on application semantics.

An *assignment* is a mapping from the kN objects (or more precisely, object replicas) to the s machines, where no machine holds multiple replicas of the same object. Assigning multiple replicas of the same object to the same machine obviously waste resources. Thus we do not consider those scenarios (it is also easy to avoid in practice).

The PTN assignment is obtained by partitioning the N objects into N/l groups of size l, and then mirroring each group onto k machines. Obviously there are many ways to do this partitioning, but they all result in the same availability when $n = N$, because the operation needs *any* t available objects to succeed. For $n < N$, we will refine the definition for PTN in Section 8. The RAND assignment is an assignment drawn uniformly randomly from all possible assignments. Thus strictly speaking, RAND is a distribution of assignments. Similarly, the definition for RAND will be refined later for $n < N$.

For a given operation, we define the *availability* of an assignment α to be the probability that the operation is successful under α. The complement of availability is called the *failure probability*, denoted by $FP(\alpha)$. Our goal is to find α such that $FP(\alpha)$ is either minimized or maximized. Doing so for all t values in $[1, n]$ is challenging, and this paper focuses on cases where t takes the two extremes.

The next section explains the challenges. Section 5 through Section 7 prove the best/worst assignment for the two extremal t values when $n = N$. Finally, Section 8 explains why the results for $n = N$ easily generalize to $n < N$.

4. CHALLENGES

To find the best/worst assignment, a brute force enumeration of all assignments is obviously infeasible. Thus the first natural attempt would be to cast the problem into a constrained optimization problem. To do this, we need a closed-form expression for the failure probability $FP()$ of any given assignment (the assignment itself, of course, also needs to be expressed as constraints). Unfortunately, our first theorem shows that even calculating $FP(\alpha)$ is difficult:

THEOREM 1. *Calculating $FP(\alpha)$ for an arbitrary α is #-P hard. This is true even if every object has only two replicas.*

Proof: We reduce the #-P hard MONOTONE 2-SAT problem [24], to calculating $FP(\alpha)$. The MONOTONE 2-SAT problem asks for the number of possible ways to satisfy a monotone boolean formula in two-conjunctive normal form. For a given instance of MONOTONE 2-SAT with n two-conjunctive tuples, we consider each tuple as an object and each literal in the problem as a machine. Each object has exactly two replicas, assigned to the two literals in the tuple. We add some dummy objects to the problem so that all machines have the same load, so that the assignment is valid. Let s denote the total number of literals (machines). A *global state* describes exactly which of the s machines are available. Clearly we have 2^s possible global states. If we set $p = 0.5$, then every global state occurs with the same probability of 0.5^s. Now consider an operation that requests all the n objects except the dummy objects, with $t = n$. One can see that the operation is available under a global state if and only if that global state satisfies the original boolean formula. Thus we have $1 - FP(\alpha) = 0.5^s \times$ *(# of ways to satisfy the boolean formula)*. $\square$

A second natural attempt to find the best/worst assignment is to use hill-climbing methods. For example, we can hope to adjust any given assignment step by step, where each step always increases (or decreases if we want to find the worst) the availability. Unfortunately, it is difficult to design these steps in general. Later in some significantly simplified scenarios in Section 7.2, we will apply such an approach.

Instead of using constrained optimization or hill climbing, in the following, we first leverage a strong probabilistic tool to prove upper and lower bounds on $FP(\alpha)$. Then we show that some specific assignments are small constants away from the respective bounds.

5. BEST AND WORST ASSIGNMENTS FOR $t = n = N$

We start from the simple (and also perhaps most interesting) case of $n = N$. It is most interesting because when $n = N$, the objects requested fully occupy all the machines (as in Figure 1). Thus there is no difference resulting from "concentrating" or "spreading" the objects, and inter-object correlation is the sole cause of the availability difference. This section considers the largest t, i.e., when $t = n$.

5.1 Upper and Lower Bounds

We derive the upper and lower bounds using a probabilistic tool called Janson's inequality [2, 9, 11].[5] Janson's inequality is mainly used to study the property of random graphs, and it provides a tail approximation for the sum of a set of *dependent* Bernoulli random variables. We briefly describe Janson's inequality for completeness.

Let J_z, $1 \leq z \leq s$ be independent indicator random variables denoting the failure of machine z in our context. In other words, J_z is 1 if machine z fails. For $1 \leq i \leq n$, let $I_i = \prod_{z \in Q_i} J_z$, where Q_i are arbitrary subsets of $\{1, 2, ..., s\}$. In our context, Q_i is the set of machines holding object i, and I_i indicates the loss of object i. We write $i \not\sim j$ if $i \neq j$ and $Q_i \cap Q_j \neq \emptyset$. Define $\mu = \sum_{i=1}^{n} Pr[I_i]$ and $\Delta = \sum_{(i,j):i \not\sim j} Pr[I_i \wedge I_j]$. Notice that the sum is done over ordered pairs of i and j. (The sum over unordered pairs is $\Delta/2$.) Then we have:

[5]Not to be confused with the well-known *Jensen's inequality*.

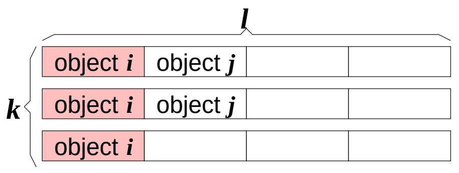

Figure 2: Illustrating the Δ term in Janson's inequality, for $k = 3$ and $l = 4$. Each row is a machine.

THEOREM 2. **Janson's inequality [2, 9, 11]**

$$Pr[I_1 = I_2 = ... = I_n = 0] \geq \prod_{i=1}^{n} Pr[I_i = 0] \quad (1)$$

$$Pr[I_1 = I_2 = ... = I_n = 0] \leq e^{-\mu^2/(\Delta+\mu)} \quad (2)$$

Obviously, when $t = n$, we have $FP(\alpha) = 1 - Pr[I_1 = I_2 = ... = I_n = 0]$. An upper bound for $FP(\alpha)$ will then immediately follow from Inequality 1. We will use Inequality 2 to obtain the lower bound. For any assignment α, the term μ is always np^k. So to obtain a lower bound on $FP(\alpha)$, all we need is to upper bound Δ. The term Δ can be written as $\sum_{i=1}^{n}(\sum_{j:j\not\sim i} Pr[I_i \wedge I_j])$. Now consider a given object i as in Figure 2, whose k replicas reside on k machines. Each machine holds $l-1$ additional objects, and these k machines have a total of $k(l-1)$ empty slots for other objects. If $j \not\sim i$, j must occupy x of these slots where $1 \leq x \leq k$. For such j, we trivially have $Pr[I_i \wedge I_j] = p^{2k-x}$.

There are many different ways of "filling up" the $k(l-1)$ slots. One extreme is to use $l-1$ objects, where each object takes k slots. The other extreme is to use $k(l-1)$ objects, where each object takes 1 slot. In the first extreme, the total number of terms in the summation of $\sum_{j:j\not\sim i} Pr[I_i \wedge I_j]$ is minimized ($l-1$ terms), but each individual term is maximized (each is p^k). In the second extreme, the total number of terms is maximized ($k(l-1)$), while each term is minimized (p^{2k-1}). Because $p \leq 0.5$, however, one would suspect that the term p^{2k-x} quickly decreases as x decreases. Thus, the number of terms becomes less important, and the magnitude of individual terms dominates. In other words, intuitively, the first extreme above will likely maximize Δ, while the second extreme will likely minimize it.

It is interesting to note that the first extreme above exactly corresponds to the PTN assignment[6], while the second extreme is closer to the RAND assignment. In other words, the Δ term actually gives us an intuition of why, when $t = n = N$, PTN is the best while RAND is the worst (within constants).

THEOREM 3. *When $t = n = N$, for any assignment α, we have:*

$$FP(\alpha) \leq 1 - (1 - p^k)^n \quad (3)$$

$$FP(\alpha) \geq 1 - e^{-(n/l)p^k} \quad (4)$$

Proof: From Inequality 1, we trivially have $1 - FP(\alpha) \geq (1 - p^k)^n$. To obtain the lower bound, consider a particular object i. Suppose there are altogether u j's such that $j \not\sim i$. For each such j, we define $x = |Q_i \cap Q_j|$. Let these u x's be $x_1, x_2, \ldots, x_u$. Clearly $1 \leq x_1, x_2, \ldots, x_u \leq k$ and $x_1 + x_2 + \ldots + x_u = (l-1)k$ (Figure 2). For the given i, we have $\sum_{j:j\not\sim i} Pr[I_i \wedge I_j] = p^{2k-x_1} + p^{2k-x_2} + ... + p^{2k-x_u}$. Lemma 15 in the appendix will prove that this summation is upper bounded by $(l-1)p^k$. When we consider all possible i's, we have $\Delta = \sum_{(i,j):i\not\sim j} Pr[I_i \wedge I_j] \leq n(l-1)p^k$. Finally, from Inequality 2 and with $\mu = np^k$, we have:

$$1 - FP(\alpha) \leq e^{-\mu^2/(\Delta+\mu)} \leq e^{-(n/l)p^k} \qquad \square$$

[6]This, of course, does not necessarily mean that PTN is optimal. All it says is that the failure probability of PTN is lower bounded by the lowest lower bound among all assignments.

5.2 Approaching Upper and Lower Bounds

To show that PTN is near the lower bound, we trivially have $FP(\texttt{PTN}) = 1 - (1 - p^k)^{n/l}$, which is not far from the lower bound of $1 - e^{-(n/l)p^k}$:

THEOREM 4. *When $t = n = N$, for any assignment α and any constant $\epsilon > 0$:*

1. *$FP(\texttt{PTN}) < 1.14FP(\alpha)$.*
2. *When either p is sufficiently small or when n is sufficiently large[7], $FP(\texttt{PTN}) < (1+\epsilon)FP(\alpha)$.*

Proof: Theorem 3 tells us that $FP(\alpha) \geq 1 - e^{-(n/l)p^k}$. Let $x = n/l$ where $x \geq 1$, $y = p^k$ where $0 < y \leq 0.5^2 = 0.25$, and $f(x,y) = (1-(1-y)^x)/(1-e^{-xy})$. It can be shown that for any constant y, $f(x,y)$ is a monotonically decreasing function of x. Next define $g(y) = f(1,y)$. We can show that $g(y)' \geq 0$ for any y. Thus $f(1,y)$ is a monotonically increasing function of y, and we have $f(x,y) \leq f(1,y) \leq f(1,0.25) < 1.14$. Furthermore, $y \to 0$ as $p \to 0$, $x \to \infty$ as $n \to \infty$, $\lim_{y\to 0} f(x,y) \to 1$, and $\lim_{x\to\infty} f(x,y) \to 1$. $\square$

Different from performance measures, because $FP(\alpha)$ is usually a close-to-zero value in practice, having a multiplicative constant is more desirable than an additive constant.

Next we intend to show that RAND is close to the worst. Remember that RAND is actually a distribution of assignments. Given the #-P hardness of calculating failure probability, it is unlikely that we can enumerate the failure probability of all assignments in the distribution. Instead, we use Janson's inequality to approximate the failure probability of the assignments in the distribution. By carefully upper bounding Δ, we can show that with high probability, an assignment drawn according to the distribution is close to the worst.

Recall from Figure 2 and the intuition in Theorem 3 that bounding Δ is all about bounding the summation $\sum_{j:j\not\sim i} Pr[I_i \wedge I_j]$ for any given i. Earlier we explained that, for $p \leq 0.5$, the magnitude of individual terms in the summation for i is more important than the number of terms. For an object j that occupies x of the $k(l-1)$ slots in Figure 2, $Pr[I_i \wedge I_j] = p^{2k-x}$, which can vary between p^{2k-1} to p^k. We will show that in RAND, with high probability, any j will occupy at most roughly $k/2$ slots (when $n \geq 2l$). This is easy to imagine since with $k(l-1)$ object replicas, it is unlikely that we end up with too many replicas from the same object. On the other hand, this will upper bound $Pr[I_i \wedge I_j]$ within roughly $p^{1.5k}$, which is sufficient to prove the result.

THEOREM 5. *When $t = n = N$, $k \geq 3$, $n \geq 2l$ and $2lp^{\lfloor k/2 \rfloor} \leq 1$, for any assignment α and any constant $\epsilon > 0$, with probability at least $1 - O(1/n)$:*

1. *$FP(\texttt{RAND}) > 0.46FP(\alpha)$.*
2. *When either p is sufficiently small or n is sufficiently large, $FP(\texttt{RAND}) > (1-\epsilon)FP(\alpha)$.*

[7]The assignment problem requires that $s = Nk/l = nk/l$ and n is not a "free" variable. In this paper, whenever we consider "sufficiently large" n, we make the natural assumption that k and l are fixed, while s changes with n as $s = nk/l$. This follows the practical meaning of the problem: Namely, when the number of objects increases, we will use more machines to hold them.

Proof: We know that for any assignment $\mu = np^k$. Let $q = 4^k kn(l/n)^{\lceil k/2 \rceil + 1}$. We will show that for RAND, $Pr[\Delta > 2nlp^{k+\lfloor k/2 \rfloor}] \le nq$. To study the distribution of Δ, consider a particular object and its corresponding indicator variable I_i as in Janson's inequality. As in the proof of Theorem 3, we suppose there are altogether u j's such that $j \not\sim i$ and for each such j, we define $x = |Q_i \cap Q_j|$. Let these u x's be $x_1, x_2, \ldots, x_u$. We have $1 \le x_1, x_2, \ldots, x_u \le k$ and $x_1 + x_2 + \ldots + x_u = (l-1)k$. Let $z = max(x_1, x_2, \ldots, x_u) \le k$. For the given i, Lemma 15 tells us that $\sum_{j \not\sim i} Pr[I_i \wedge I_j] = p^{2k-x_1} + p^{2k-x_2} + \ldots + p^{2k-x_u} \le \lceil \frac{(l-1)k}{z} \rceil p^{2k-z} \le \frac{kl}{z} \cdot p^{2k-z}$. Define $h(z) = \frac{kl}{z} \cdot p^{2k-z}$ and we have $\sum_{j \not\sim i} Pr[I_i \wedge I_j] \le h(z)$.

On the other hand, the definition of z is exactly the same as in Lemma 16, which tells us:

$$Pr[z > \lceil k/2 \rceil] = Pr[z \ge \lceil k/2 \rceil + 1] \le q$$

Because $p \le 0.5$, one can show that $h(1) \le h(2) < h(3) < h(4) < \ldots$. This means that if $\sum_{j \not\sim i} Pr[I_i \wedge I_j] > h(\lceil k/2 \rceil)$, we must have $h(z) > h(\lceil k/2 \rceil)$ and $z > \lceil k/2 \rceil$ (since $\lceil k/2 \rceil \ge 2$). Thus for given i:

$$\begin{aligned} q &\ge Pr[z \ge \lceil k/2 \rceil + 1] = Pr[z > \lceil k/2 \rceil] \\ &\ge Pr[\sum_{j \not\sim i} Pr[I_i \wedge I_j] > \frac{kl}{\lceil k/2 \rceil} \cdot p^{2k - \lceil k/2 \rceil}] \\ &\ge Pr[\sum_{j \not\sim i} Pr[I_i \wedge I_j] > 2lp^{k+\lfloor k/2 \rfloor}] \end{aligned}$$

When considering all possible i's, we have:

$$Pr[\Delta > 2nlp^{k+\lfloor k/2 \rfloor}] \le nq$$

Thus Inequality 2 tells us that with at least $1 - nq$ probability:

$$\begin{aligned} FP(\texttt{RAND}) &\ge 1 - e^{(-n^2 p^{2k})/(np^k + 2nlp^{k+\lfloor k/2 \rfloor})} \\ &= 1 - e^{-np^k/(1+2lp^{\lfloor k/2 \rfloor})} \ge 1 - e^{-0.5np^k} \end{aligned}$$

From Theorem 3, we know that $FP(\alpha) \le 1 - (1-p^k)^n$. Let $x = n$ where $1 \le x$, $y = p^k$ where $0 < y \le 0.5^2 = 0.25$, and $f(x,y) = (1 - e^{-0.5xy})/(1-(1-y)^x)$. It can be shown that for any constant y, $f(x,y)$ is a monotonically increasing function of x. Next define $g(y) = f(1,y)$. One can show that $g'(y) \le 0$ for any y. Thus $f(1,y)$ is a monotonically decreasing function of y, and we have $f(x,y) \ge f(1,y) \ge f(1, 0.25) > 0.46$. Furthermore, we also have $\lim_{x\to\infty} f(x,y) \to 1$ and $\lim_{y \to 0} f(x,y) \to 1$. □

A similar proof can be constructed for $k = 2$, which we omit for brevity. The theorem also extends to some other parameter regions where the condition of $2lp^{\lfloor k/2 \rfloor} \le 1$ is not met – we do not tediously enumerate all such regions here because this is about the worst assignment, which is less important than the best assignment.

Discussion on inter-object correlation in PTN and RAND. We have shown that PTN and RAND are the best and worst assignments (within constants and under the conditions in the theorems). For $t = n$, if the n objects were all independent, then the success probability of the operation would be $(1-p^k)^n$. Inter-object correlation helps us to improve such probability: Conditioned upon one object being available, other objects that reside on the same machines as that object will have an availability larger than $(1-p^k)$. However, notice that the availability of RAND approaches $(1-p^k)^n$, meaning that the inter-object correlation in RAND is weak and the availability is almost as if the n objects were independent. On the other hand, PTN has a strong correlation and the availability $(1-p^k)^{n/l}$ is as if we only had n/l independent objects. When p^k is small, the difference between $FP(\texttt{RAND})$ and $FP(\texttt{PTN})$ is about a factor of l.

Because randomly assigning objects seems to be a good way to minimize the correlation among objects, it may appear obvious that RAND should be close to the worst. But such intuition overlooks the subtlety in defining a single quantitative measure for inter-object correlation. For example, for $t = n$, Janson's inequality uses a single quantity Δ to "summarize" inter-object correlation. We indeed showed that RAND tends to give us a small Δ for $p \le 0.5$, which led to Theorem 5. On the other hand, when $p \to 1$, each term in the summation of $\Delta = \sum_{(i,j):i \not\sim j} Pr[I_i \wedge I_j]$ approaches 1. As a result, because RAND tends to maximize the number of terms in the summation, RAND will actually give us a large Δ (much larger than the Δ in PTN) when $p \to 1$. This, however, does not mean that the correlation in RAND becomes larger now, since obviously the correlation level is an inherent property of the assignment and should not depend on p. Thus the only explanation is that Δ now fails to accurately capture such correlation. In fact, one can prove that PTN is still the best assignment when $p \to 1$.

6. BEST AND WORST ASSIGNMENTS FOR $t = l+1 < n = N$

Having discussed the largest t in the previous section, we now turn to the smallest t in this section. When $t \in [1, l]$ the assignment problem reduces to a trivial one, because all assignments have the same availability. Thus the smallest t we consider is $l + 1$.

We first use basic combinatorial arguments to prove that PTN is the worst assignment when $t = l + 1$. The intuition is that we need one machine to be available to give us l objects, and some other machines to be available such that at least one of these machines provides at least one other object. This does not happen only when all the available machines have exactly the same set of objects. A counting argument will show that the number of such scenarios is maximized under PTN.

To formalize such arguments, we introduce some definitions that are used only in this section. A *configuration* G is the subset of the available machines out of the s machines. A configuration is *unavailable* (under a given assignment) if the number of available objects in that configuration is smaller than t.

THEOREM 6. *When $t = l + 1 < n = N$, $FP(\texttt{PTN}) \ge FP(\alpha)$ for any assignment α.*

Proof: For a given assignment and any $1 \le i \le s$ and $1 \le v \le s$, we define $U_{i,v}$ to be:

$$\{G | i \in G,\ |G| = v, \text{ and } G \text{ is an unavailable configuration}\}$$

The failure probability of the assignment is then:

$$p^s + \sum_{v=1}^{s} \sum_{i=1}^{s} [|U_{i,v}|/v \times (1-p)^v p^{s-v}]$$

In the expression, the term $|U_{i,v}|/v$ is the total number of size v configurations that are unavailable (each such configuration appears in v different $U_{i,v}$'s).

To prove $FP(\texttt{PTN}) \ge FP(\alpha)$, it suffices to show that for any i and v, $|U_{i,v}|$ is maximized under PTN. In any configuration $G \in U_{i,v}$, machine i must be available. Thus at least l objects are already available. As G is not available under the threshold of $l + 1$, all other machines in G must hold exactly the same set of objects as machine i. With each object having k replicas, we can have at most $k - 1$ such machines. Thus $|U_{i,v}| \le \binom{k-1}{v-1}$ for $1 \le v \le k$ and $|U_{i,v}| = 0$ for $k + 1 \le v \le s$. On the other hand, in PTN, we have exactly $|U_{i,v}| = \binom{k-1}{v-1}$ for $1 \le v \le k$. Thus $|U_{i,v}|$ is maximized under PTN. □

Next we want to find the best assignment for $t = l+1$. This is not difficult, because all we need is that no two machines hold exactly the same set of objects. It is simple to construct such an assignment using a sliding-window approach. We number all objects from 1 to n and place them sequentially on a ring. Imagine that there is a window of size l. The first machine holds object 1 through object l. Then we slide the "window" to the right by l/k, and have the second machine hold object $(l/k+1)$ through $(l/k+l)$. Similarly, the third machine will hold object $(2l/k+1)$ through $(2l/k+l)$, and so on.

In the following, however, we intend to derive a more interesting result—we will prove that RAND is close to optimal when $t = l+1$. This means that just randomly picking an assignment can be almost as good as carefully constructing one. The intuition is simple: if we randomly assign object replicas, then it is unlikely that two machines will have exactly the same set of objects.

THEOREM 7. *When $t = l+1 < n = N$, for any assignment α and any positive constant ϵ, we have $E[FP(\texttt{RAND})] \le (1+\epsilon)FP(\alpha)$ when n is sufficiently large.*

Proof: For any assignment, if less than two machines are available, there must be less than $l+1$ objects. Thus when $t = l+1$, for any assignment α, we have the trivial lower bound of $FP(\alpha) \ge p^s + s(1-p)p^{s-1}$.

For RAND, we define random variable U_i to be the set of unavailable configurations whose size is i. From the linearity of expectation, we have $E[FP(\texttt{RAND})]$ as:

$$p^s + s(1-p)p^{s-1} + \sum_{i=2}^{k} E[|U_i|](1-p)^i p^{s-i}$$

It is important to notice that s is not a constant and when $n \to \infty$, we also have $s = nk/l \to \infty$. Thus the lower bound of $p^s + s(1-p)p^{s-1}$ will approach zero when $n \to \infty$. To prove the theorem, showing $E[|U_i|] \to 0$ is not sufficient – we need to prove that it approaches zero at a faster rate than the lower bound, as following.

Let C_i to be the set of all configurations whose size is i, and we have $|C_i| = \binom{s}{i}$. Notice that C_i is not a random variable. Define indicator random variable $x_{i,j}$, where $x_{i,j} = 1$ if and only if the jth configuration in C_i belongs to U_i. Obviously, we have $E[|U_i|] = E[\sum_j x_{i,j}] = \binom{s}{i}E[x_{i,1}] \le s^i E[x_{i,1}]$. Notice that $E[x_{i,1}]$ is the probability of having exactly l distinct objects on a given set of i machines in RAND. Thus $E[x_{i,1}]$ decreases with i and we have $E[|U_i|] \le s^k E[x_{2,1}]$.

Determining the probability (i.e., $E[x_{2,1}]$) of having exactly l distinct objects on a given set of 2 machines is actually not trivial because a machine is not allowed to hold multiple replicas of the same object. As a result, the number of ways to assign object replicas to the remaining $n-2$ machines is dependent on how we assign the object replicas to the 2 machines. Lemma 17 in the appendix proves that $E[x_{2,1}] < (l/n)^l \le (l/n)^k$. Thus we have $E[FP(\texttt{RAND})]$ as:

$$\begin{aligned}
& p^s + s(1-p)p^{s-1} + \sum_{i=2}^{k} E[|U_i|](1-p)^i p^{s-i} \\
< \quad & p^s + s(1-p)p^{s-1} + k \cdot s^k \cdot (l/n)^k \cdot (1-p) \cdot p^{s-k} \\
\le \quad & p^s + s(1-p)p^{s-1} + s(1-p)p^{s-1} \cdot \left(\frac{1}{p^{k-1}} \cdot \frac{k^k l}{n}\right) \\
\le \quad & (1+\epsilon)FP(\alpha) \text{ when } n \text{ sufficiently large}
\end{aligned}$$

□

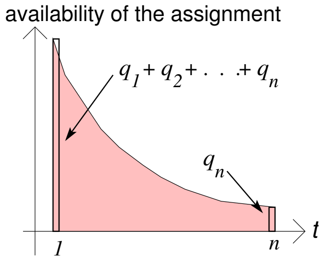

Figure 3: Illustrating the area bounded by the availability curve. The curve is actually a step function, so the area bounded by the curve exactly equals the sum of the rectangular areas.

COROLLARY 8. *For any assignment α and any positive constants ϵ_1 and ϵ_2, when $t = l+1 < n = N$ and n is sufficiently large, $Pr[FP(\texttt{RAND}) \ge (1+\epsilon_1)FP(\alpha)] \le \epsilon_2$.*

7. A DEEPER LOOK

7.1 Impossibility of Remaining Optimal Across All Values of t

Our results so far show that when t decreases from n to $l+1$, the best assignment (PTN) becomes the worst, while the worst assignment (RAND) becomes the best. Ideally, we would prefer an assignment that is optimal for all t values. However, the following shows that no such assignment exists.

Our proof is based on the area bounded by the availability curve (Figure 3) for any assignment, where the x-axis is t and the y-axis is the availability of the given assignment. We will prove that the area is a constant independent of the assignment. If we want to raise one part of the curve, some other part must necessarily drop to keep the area constant. It is impossible for a single assignment to be optimal under all t values because otherwise the area will no longer remain constant. Also because the difference between PTN and RAND is usually large, it is not even possible for a single assignment to be near to the optimal under all t value.

Specifically, for any assignment, let q_i be the probability that exactly i objects are available, for $0 \le i \le n$. The area bounded by the availability curve is:

$$(q_1 + \cdots + q_n) + (q_2 + \cdots + q_n) + \cdots + (q_n) \;=\; \sum_{i=1}^{n} iq_i$$

On the other hand, the summation $\sum_{i=1}^{n} iq_i$ is exactly the expected number of available objects in the system. In any assignment, each object is available with probability $1-p^k$. Linearity of expectation tells us that $\sum_{i=1}^{n} iq_i = n(1-p^k)$, which is independent of the assignment.

7.2 Removing Constant Factors When $t = n = N$ and $k = l = 2$

For $t = l+1$, we already obtained the best and the worst assignments without any constant factors. While for $t = n$, the best and the worst assignments are within constant factors (Theorem 4 and Theorem 5). In particular, the worst "assignment" RAND is actually a distribution, which does not shed much light onto the structure of the worst assignment. Thus this section aims to find the best/worst assignment (without constants) for $t = n$, under some significantly restricted scenarios.

The scenarios we consider are when every object has 2 replicas and each machine holds 2 objects (as in Figure 1). We will see

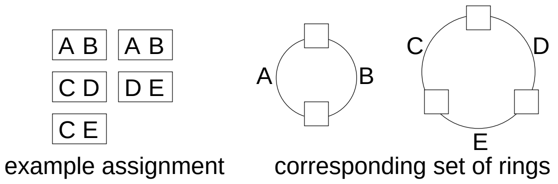

Figure 4: When $k = l = 2$, each assignment can be uniquely represented as a set of rings. Each box represents a machine.

that even under such significantly restricted and perhaps impractical parameters, the problem is still far from trivial. To simplify discussion, the rest of this section assumes that $n|2$ and $n|3$.

When $k = l = 2$, any assignment can be uniquely represented as a set of rings (Figure 4). Each ring edge represents an object, and each ring node corresponds to a machine holding the two adjacent edges (objects). The size of a ring can range from 2 to n. Obviously, if two adjacent nodes on any ring fail, then we lose an object and the assignment becomes unavailable under $t = n$. Notice that because all objects are equivalent, it is not important which object corresponds to which ring edge – only the ring sizes matter. Define $f(x)$ to be the probability of not having any two adjacent nodes failing on a ring of size x for $2 \le x \le n$. If assignment α corresponds to rings of size $x, y, z, \ldots$, then $FP(\alpha) = 1 - f(x)f(y)f(z)\ldots$.

To find the best (worst) assignment, we use hill-climbing and adjust an assignment repeatedly so that at each step $FP()$ is decreased (increased). For any assignment, the sum of the sizes of all rings is always n. To keep this invariant, an adjustment step can either split a big ring of size $x + y$ into two smaller ones of size x and y, or merge two smaller rings of size x and y into one big ring of size $x + y$. Any assignment can be transformed into any other assignment via a sequence of these adjustment steps. The crux is to understand how availability changes in these steps, or more precisely, which of $f(x)f(y)$ and $f(x+y)$ is larger. Interestingly, the comparison outcome is uniquely determined by the parity of the smaller of x and y:

LEMMA 9.

$$\begin{aligned} f(x) &= z_1^x + z_2^x, \text{ where: } q = \sqrt{(3p+1)(1-p)}, \\ z_1 &= (1-p+q)/2, \; z_2 = (1-p-q)/2 \end{aligned}$$

We also have $z_1 > 0$, $z_2 < 0$, and $0 < |z_2| < |z_1| < 1$.

Proof: Consider a string of n nodes (i.e., a broken ring). Define $g(x)$ to be the probability of not having any two adjacent nodes failing on a string of size x for $x \ge 1$. Define $g(0) = 1$. One can show the following linear recurrence for $x \ge 2$:

$$g(x) = (1-p)g(x-1) + p(1-p)g(x-2) \tag{5}$$

Using standard techniques to solve the recurrence, we have for $x \ge 0$:

$$\begin{aligned} g(x) &= az_1^x + bz_2^x, \text{ where:} \\ a &= (q+1+p)/(2q), \; b = (q-1-p)/(2q) \end{aligned}$$

z_1, z_2 and q are as defined in the Lemma

For $x \ge 3$, we have:

$$f(x) = (1-p)^2 g(x-2) + 2p(1-p)^2 g(x-3) \tag{6}$$

One can easily verify that $f(x) = z_1^x + z_2^x$ satisfies the above equation. It is trivial to show that $f(2) = z_1^2 + z_2^2$. □

LEMMA 10. *Let x and y be integers where $x \ge y \ge 2$:*

- *$f(x)f(y) > f(x+y)$ if y is even.*
- *$f(x+y) > f(x)f(y)$ if y is odd.*

Proof:

$$\begin{aligned} & f(x)f(y) - f(x+y) \\ = \; & (z_1^x + z_2^x)(z_1^y + z_2^y) - (z_1^{x+y} + z_2^{x+y}) \\ = \; & z_1^x z_2^y + z_1^y z_2^x = z_1^y z_2^y (z_1^{x-y} + z_2^{x-y}) \end{aligned}$$

Because $(x-y) \ge 0$, $z_1 > 0$, and $|z_1| > |z_2|$, the term $(z_1^{x-y} + z_2^{x-y})$ is positive. Thus the sign of the above expression must be the same as the sign of z_2^y. □

We are now ready to prove the two main theorems of this section, which say that $n/2$ size-2 rings are the best while $n/3$ size-3 rings are the worst. This is somewhat surprising because it is tempting to conjecture that a single ring of size n is the worst.

THEOREM 11. *When $t = n = N$ and $k = l = 2$, the assignment corresponding to $n/3$ size-3 rings has the highest failure probability.*

Proof: Consider the set of rings corresponding to any given assignment α. We want to adjust the rings to ultimately obtain $n/3$ size-3 rings. To always decrease availability in the adjustment steps, Lemma 10 allows two kinds of adjustments:

- Merge two rings of size x and y into one ring of size $(x+y)$ where $x \ge y$ and y is even.
- Split a ring of size $(x+y)$ into two rings of size x and y where $x \ge y$ and y is odd.

Now we adjust the set of rings corresponding to assignment α. First, for any odd size ring whose size is at least 7, we split it into two smaller rings where one of them is of size 3. We will end up with three kinds of rings: size-3 rings, size-5 rings, and even size rings. We merge all even size rings into a single big one, and then keep ripping size-3 rings out of the big ring until the size of the big ring is either 3, 4, or 5. If we do have a size-4 ring, then there must be at least one size-5 ring (since $n|3$). We merge the size-4 ring and the size-5 ring into a size-9 ring. This ring can then be split into 3 size-3 rings. Now we have only size-3 rings and size-5 rings. One can easily show that $f^3(5) > f^5(3)$, which enables us to adjust all size-5 rings into size-3 rings. At this point, we obtain $n/3$ size-3 rings. □

Note that the ring structure of RAND significantly differs from $n/3$ size-3 rings. For $k = l = 2$, RAND can be roughly viewed as selecting a random permutation of $1..n$. In a random permutation, the expected number of cycles of size $m \le n$ is $1/m$ and the average cycle size is $\Theta(n/\log n)$ [14], which is far from $n/3$.

THEOREM 12. *When $t = n = N$ and $k = l = 2$, the assignment corresponding to $n/2$ size-2 rings (i.e., the PTN assignment) has the lowest failure probability.*

Proof: To always increase availability in the adjustment, Lemma 10 allows two kinds of adjustments:

- Merge two rings of size x and y into one ring of size $(x+y)$ where $x \ge y$ and y is odd.
- Split a ring of size $(x+y)$ into two rings of size x and y where $x \ge y$ and y is even.

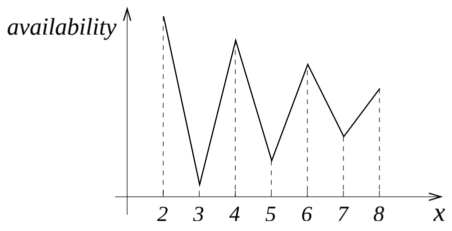

Figure 5: Illustrating the availability of the assignment corresponding to n/x rings of size x.

Now we adjust the set of rings corresponding to assignment α. We first break all even size rings whose size is at least 4 into size-2 rings. For all odd size rings whose size is at least 5, we keep ripping out size-2 rings until all odd size rings become size 3. We now have only size-2 rings and size-3 rings. We merge all rings of size 3 into one big ring, and then split the big ring into size-2 rings. □

As we mentioned, it was tempting to conjecture that a single size-n ring has the worst availability. Given that is not true, what availability does a single size-n ring provide? Is it close to the best or close to the worst? What about rings of other sizes? To shed light onto these question, we consider assignments that correspond to rings of homogeneous sizes (i.e., (n/x)[8] rings of size x). We will prove that:

- If x is odd, then the larger x is, the better the availability. For example, 3 rings of size 25 is better than 5 rings of size 15.
- If x is even, then the larger x is, the worse the availability. For example, 4 rings of size 12 is worse than 6 rings of size 8.
- If x is even and y is odd, then (n/x) size-x rings is always better than (n/y) size-y rings.

Figure 5 illustrates these results, where as we increase x, the availability oscillates with a decreasing oscillation magnitude. Notice that these results, however, do not necessarily imply Theorem 11 and 12, which also apply to rings of heterogeneous sizes.

THEOREM 13.

- *For any odd integers x and y where $x > y \geq 3$, we have $(f(x))^{n/x} > (f(y))^{n/y}$.*
- *For any even integers x and y where $x > y \geq 2$, we have $(f(x))^{n/x} < (f(y))^{n/y}$.*
- *For any even integer $x \geq 2$ and any odd integer $y \geq 3$, we have $(f(x))^{n/x} > (f(y))^{n/y}$.*

Proof: For the first case, define $z_3 = -z_2 > 0$. We have from Lemma 9 that $f(x) = z_1^x - z_3^x$ and $f(y) = z_1^y - z_3^y$:

$$\begin{aligned} & (f(x))^{n/x} > (f(y))^{n/y} \Leftrightarrow (z_1^x - z_3^x)^y > (z_1^y - z_3^y)^x \\ \Leftrightarrow \quad & (1 - (z_3/z_1)^x)^y > (1 - (z_3/z_1)^y)^x \end{aligned}$$

On the other hand, because $0 < z_3/z_1 < 1$, we have $(1 - (z_3/z_1)^x)^y > (1 - (z_3/z_1)^x)^x > (1 - (z_3/z_1)^y)^x$. The other two cases are similar. □

[8]To simplify discussion, wherever we use the notation n/x below, we assume $n|x$.

8. GENERALIZING TO $n < N$

So far we have considered only $n = N$. When $n < N$, the nk replicas of the n requested objects may not be evenly distributed among the s machines. Each machine may hold any number (ranging from 0 to l) of the replicas for these n objects. This provides us with another degree of freedom, and PTN and RAND as defined in Section 3 are no longer well-defined. We refine these definitions as follows. The PTN assignment is obtained by partitioning the N objects into N/l groups of size l where the n objects requested by the operation belong to exactly n/l groups, and then mirroring each group onto k machines. In other words, the n objects "concentrate" and occupy as few machines as possible. The RAND assignment is a random assignment drawn uniformly randomly from all assignments where each machine holds exactly nk/s object replicas of the n objects requested by the operation. In other words, the n objects "spread" and occupy (evenly) as many machines as possible.

Now regarding the best/worst assignments for $t = n < N$, observe that Theorem 3 can be trivially modified to apply to the case of $n < N$, while Theorem 4 applies without modification. Thus PTN is optimal within a constant of 1.14. Similarly, it is trivial to show that Theorem 5 applies for $t = n < N$, which means that RAND is the worst within a constant of 0.46. As for the best and worst assignments for small t, notice that the smallest interesting t value is now 1 (instead of $l + 1$). For $t = 1$, we only need any machine holding any of the nk object replicas to be available for the operation to be successful. PTN uses the minimal number of machines for the n objects, while RAND uses the maximum number of machines. Thus trivially, they are the best and worst assignment, respectively.

In practice, the application may have multiple multi-object operations, and each multi-object operation may access a different subset of n objects. Mathematically modelling these operations is challenging because the access pattern can be complex and simply assuming that each operation accesses a uniformly random subset will be far from reality. Our previous experimental study [26] has investigated these scenarios for some specific applications.

9. CONCLUSION

This paper has proved a series of strong results regarding the best and the worst assignments (from object replicas to machines) in terms of the availability they provide to multi-object operations. Quite different from classic research on replica placement, here the availability difference arises from inter-object correlation (even when machine failures are independent and identical). As mentioned in Section 1, the same assignment problem arises under erasure coding. We have also obtained similar initial results by using a later version [10] of Suen's inequality [21]. Obtaining a complete set of results for erasure coding is part of our future work.

10. ACKNOWLEDGMENTS

We thank Suman Nath for various discussions related to this paper. We also thank Avishai Wool and the anonymous reviewers for many helpful comments on the paper. This work is partly supported by NUS grant R-252-050-284-101 and R-252-050-284-133.

11. REFERENCES

[1] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. In *USENIX OSDI*, 2002.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley & Sons, 2000.

[3] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. In *ACM SIGMETRICS*, 2000.

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area Cooperative Storage with CFS. In *ACM SOSP*, 2001.

[5] J. R. Douceur and R. P. Wattenhofer. Competitive Hill-Climbing Strategies for Replica Placement in a Distributed File System. In *DISC*, 2001.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google File System. In *ACM SOSP*, 2003.

[7] A. Haeberlen, A. Mislove, and P. Druschel. Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures. In *USENIX NSDI*, 2005.

[8] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. Ganger, E. Riedel, and A. Ailamaki. Diamond: A Storage Architecture for Early Discard in Interactive Search. In *USENIX FAST*, 2004.

[9] S. Janson. Poisson Approximations for Large Deviations. *Random Structures and Algorithms*, 1:221–230, 1990.

[10] S. Janson. New Versions of Suen's Correlation Inequality. *Random Structures and Algorithms*, 13(3-4):467–483, 1998.

[11] S. Janson, T. Luczak, and A. Rucinski. An Exponential Bound for the Probability of Nonexistence of a Specified Subgraph in a Random Graph. In *Random Graphs'87 (M. Karo'nski and J. Jaworski and A. Ruci'nski, Eds.)*. John Wiley & Sons, 1990.

[12] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *ACM STOC*, 1997.

[13] J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Trans. Comput. Syst.*, 10(1):3–25, 1992.

[14] D. E. Knuth. *The Art of Computer Programming, Volume 1*. Addison Wesley, 1997.

[15] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-scale Persistent Storage. In *ACM ASPLOS*, 2000.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-addressable Network. In *ACM SIGCOMM*, 2001.

[17] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-Centric Storage in Sensornets with GHT, A Geographic Hash Table. *Mobile Networks and Applications*, 8(4), 2003.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems. In *ACM Middleware*, 2001.

[19] J. Santos, R. Muntz, and B. Ribeiro-Neto. Comparing Random Data Allocation and Data Striping in Multimedia Serviers. In *ACM SIGMETRICS*, 2000.

[20] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM*, 2001.

[21] S. Suen. A Correlation Inequality and a Poisson Limit Theorem for Nonoverlapping Balanced Subgraphs of a Random Graph. *Random Structures and Algorithms*, 1(2):231–242, 1990.

[22] A. Szalay, P. Kunszt, A. Thakar, J. Gray, and D. Slutz. Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey. In *ACM SIGMOD*, 2000.

[23] TPC Benchmark. `http://www.tpc.org/`.

[24] L. G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. on Computing*, 8(3):410–421, 1979.

[25] R. van Renesse and F. B. Schneider. Chain Replication for Supporting High Throughput and Availability. In *USENIX OSDI*, 2004.

[26] H. Yu, P. B. Gibbons, and S. Nath. Availability of Multi-Object Operations. In *USENIX NSDI*, 2006.

[27] H. Yu and A. Vahdat. Minimal Replication Cost for Availability. In *ACM PODC*, 2002.

APPENDIX

LEMMA 14. *For $1 \le x_1, x_2 \le z$, we have:*

- *If $p \le 0.5$, then $p^{-x_1} + p^{-x_2} \le p^{-(x_1+x_2)}$.*
- *If $x_1 + x_2 \ge z + 1$, then $p^{-x_1} + p^{-x_2} \le p^{-z} + p^{-x_1-x_2+z}$*

Proof:

- Without loss of generality, assume $x_1 \le x_2$. Let $q = 1/p \ge 2$. We have $q^{x_1-x_2} \le 1 \Rightarrow q^{x_1-x_2} + 1 \le 2 \le q^{x_1} \Rightarrow q^{x_1} + q^{x_2} \le q^{x_1+x_2}$.
- Let $d = x_1 + x_2$ and define $f(x) = p^{-x} + p^{-(d-x)}$ for $d - z \le x \le z$. We only need to prove that $f(x_1) \le f(z)$. We have $f'(x) = (\ln p)(p^{x-d} - p^{-x})$ and $f''(x) = (\ln p)^2(p^{-x} + p^{x-d}) > 0$. Thus the maximum of $f(x)$ must occur at the boundary. On the other hand, $f(d-z) = f(z) = p^{-z} + p^{-(d-z)}$. So we have $f(x_1) \le f(z)$.

□

LEMMA 15. *Consider $1 \le x_1, x_2, ..., x_u \le z$ where $\sum_{i=1}^{u} x_i = (l-1)k$. We have $p^{2k-x_1} + p^{2k-x_2} + ... + p^{2k-x_u} \le \lceil \frac{(l-1)k}{z} \rceil p^{2k-z}$ when $p \le 0.5$.*

Proof: It suffices to show that $p^{-x_1} + p^{-x_2} + ... + p^{-x_u} \le \lceil \frac{(l-1)k}{z} \rceil p^{-z}$. We combine the terms on the left-hand side step by step. If $x_i + x_j \le z$, we combine the two terms of p^{-x_i} and p^{-x_j} into one term $p^{-x_i-x_j}$. Otherwise we convert the two terms to $p^{-z} + p^{-x_1-x_2+z}$. From Lemma 14, we know that the summation is never decreased at any of these steps. Thus $p^{-x_1} + p^{-x_2} + ... + p^{-x_u} \le \lceil \frac{(l-1)k}{z} \rceil p^{-z}$. □

LEMMA 16. *Consider the* RAND *distribution and a given object A. Without loss of generality assume that machine $1, 2, ..., k$ each holds a replica of A. Let random variable z ($1 \le z \le k$) denote the maximum number of replicas that these k machines hold for any other object. Then $Pr[z \ge a] \le 4^k kn(l/n)^a$ for any a, $1 \le a \le k$.*

Proof: After A is already been assigned to the machine 1 through k, we define a *remainder assignment* to be the assignment from the remaining $n-1$ objects (each with k replicas) to the remaining space on the s machines. Among these s machines, k will each take $l-1$ additional object replicas, while $s-k$ will each take l additional object replicas.

We will bound the probability of the first k machines holding exactly i ($1 \le i \le k$) replicas of a given object B. Because the first k machines are all identical, we have:

$$\begin{aligned}
& Pr[\text{first } k \text{ machines hold exactly } i \text{ replicas of } B] / \binom{k}{i} \\
= \; & Pr[\text{first } i \text{ machines hold } i \text{ replicas of } B \text{ and} \\
& \quad \text{the next } k-i \text{ machines do not hold } B] \\
< \; & Pr[\text{machine 1 holds } B] \times \\
& Pr[\text{machine 2 holds } B \mid \text{machine 1 holds } B] \times ... \times \\
& Pr[\text{machine i holds } B \mid \text{machine } 1..(i-1) \text{ hold } B] \qquad (7)
\end{aligned}$$

To simplify the notation, we define the event "machine 0 holds B" to be an event that always happens. We will prove that for any $1 \le j \le i$, $Pr[\text{machine } j \text{ holds } B \mid \text{ machine } 0..(j-1) \text{ hold } B] < 1/(s-j+1)$.

For any given j, define $p_r = Pr[\text{machine } r \text{ holds } B \mid \text{machine } 0..(j-1) \text{ hold } B]$ for $j \le r \le s$. By symmetry, we know that $p_j = p_{j+1} = ... = p_k$ and $p_{k+1} = ... = p_s$. Also, we have $p_j + p_{j+1} + ... + p_k + p_{k+1} + ... + p_s = 1$. Thus to prove that $p_j < 1/(s-j+1)$, it suffices to show that $p_j < p_s$, which we prove in the following.

Let Γ be the set of remainder assignments where machine 1 through j holds object B, and let Λ be the set of remainder assignments where machine 1 through $j-1$ and also machine s holds object B. To prove that $p_j < p_s$, it suffices to show that $|\Gamma| < |\Lambda|$. Define $\Gamma' = \Gamma - (\Gamma \cap \Lambda)$ and $\Lambda' = \Lambda - (\Gamma \cap \Lambda)$. In turn, it suffices to prove that $|\Gamma'| < |\Lambda'|$. We define Γ'_w for $0 \le w \le l-2$ to be the subset of Γ' where machine j and machine s have exactly w objects in common. Obviously, Γ'_w's are all disjoint and $\Gamma' = \cup_{w=0}^{l-2} \Gamma'_w$. Similarly, define Λ'_w for $0 \le w \le l-1$ to be the subset of Λ' where machine j and machine s have exactly w objects in common. We again have $\Lambda' = \cup_{w=0}^{l-1} \Lambda'_w$. To prove that $|\Gamma'| < |\Lambda'|$, it suffices to show that $|\Gamma'_w| < |\Lambda'_w|$ for any $0 \le w \le l-2$.

For the purpose of counting, we define a many-to-many mapping between Γ'_w and Λ'_w. Consider any remainder assignment $\gamma \in \Gamma'_w$. We can obtain another remainder assignment $\lambda \in \Lambda'_w$ by swapping B on machine j with some other object C on machine s as long as C is not already on machine j. There are exactly $l-w$ choices for C. Similarly, for any remainder assignment $\lambda \in \Lambda'_w$, we can obtain another remainder assignment $\gamma \in \Gamma'_w$ by swapping B on machine s with some other object C on machine j as long as C is not already on machine s. There are exactly $l-w-1$ choices for C. We say that γ is the pre-image of λ and λ is the after-image of γ. Each γ has exactly $l-w$ after-images, while each λ has exactly $l-w-1$ pre-images. Thus it must be the case that $|\Gamma'_w| < |\Lambda'_w|$.

We just proved that $p_j < p_s$, which implies that for any $1 \le j \le i$, $Pr[\text{machine } j \text{ holds } B \mid \text{machine } 0..(j-1) \text{ hold } B] < 1/(s-j+1)$. Together with Inequality 7, this means that:

$$\begin{aligned} & Pr[\text{first } k \text{ machines hold exactly } i \text{ replicas of } B] \\ < \quad & \binom{k}{i} \cdot \frac{1}{s} \cdot \frac{1}{s-1} \cdot ... \cdot \frac{1}{s-(i-1)} < \left(\frac{k}{s-k}\right)^i \\ = \quad & \left(\frac{l}{n-l}\right)^i \le \left(\frac{4l}{n}\right)^i \end{aligned}$$

The last step follows from $n \ge 2l$. Finally we take a summation for i from a to k, and also consider all possible B's:

$$Pr[z \ge a] \quad < \quad n \sum_{i=a}^{k} \left(\frac{4l}{n}\right)^i < 4^k kn(l/n)^a$$

□

LEMMA 17. *For any given 2 machines in the* RAND *assignment, the probability that they hold exactly the same set of objects is less than* $(l/n)^l$.

Proof: We consider two scenarios where the first scenario requires that the 2 machines hold exactly the same set of objects while the second scenario does not have any such requirement. Recall that a machine is never allowed to hold multiple replicas of the same object. In the first scenario, the number of ways to assign object replicas to the 2 machines is $\binom{n}{l}$. For each such way, let x be the number of possible *remainder assignments*, where a *remainder assignment* is the assignment of the remaining object replicas to the remaining $n-2$ machines. In the second scenario, the number of ways to assign the objects to the 2 machines is $\binom{n}{l}\binom{n}{l}$. For each such way, let y be the number of possible remainder assignments.

We would like to prove that x is always smaller than or equal to y. In the first scenario, after we assign objects to the 2 machines, we let $k-2 \le a_1 \le a_2 \le ... \le a_n \le k$ denote the number of remaining replicas for the n objects. Obviously, $a_1 = ... = a_l = k-2$ and $a_{l+1} = ... = a_n = k$. The value of x does not depend on the identities of the objects, and is uniquely determined by the sequence $a_1 a_2 ... a_n$. We define $k-2 \le b_1 \le b_2 \le ... \le b_n \le k$ similarly in the second scenario. We know that $a_1 \le b_1, ..., a_l \le b_l$ and $a_{l+1} \ge b_{l+1}, ..., a_n \ge b_n$. Again, the value of y is uniquely determined by the sequence $b_1 b_2 ... b_n$.

The sequence $b_1 b_2 ... b_n$ contains at most three distinct values: $k-2$, $k-1$, and k. There must be an even number of b_i's with a value of $k-1$. We associate the first $k-1$ with the last $k-1$ and call that a *pair*. We then consider the remaining values and form a second pair, and so on. We will prove that $x \le y$ via an induction on the number of pairs.

For the induction base, if the number of pairs is 0, then the two sequences $a_1 a_2 ... a_n$ and $b_1 b_2 ... b_n$ are the same. Since the identities of the objects do not affect the accounting, we have $x = y$.

Now assume that $x \le y$ if the number of pairs is d, and we need to prove that $x \le y$ if the number of pairs is $d+1$. For the sequence $b_1 b_2 ... b_n$, let (b_i, b_j) be the first pair formed where $i < j$. Without loss of generality, we number the objects such that b_i corresponds to the number of remaining replicas for object i. We construct a third sequence $c_1 c_2 ... c_n$ from $b_1 b_2 ... b_n$ by changing b_i from $k-1$ to $k-2$ and b_j from $k-1$ to k. Let z be the number of possible remainder assignments for sequence $c_1 c_2 ... c_n$. Because $c_1 c_2 ... c_n$ has d pairs, based on our inductive assumption, we know that $x \le z$. In the next, we will prove that $z \le y$, thus finishing the inductive step.

For any machine (except the given 2 machines) holding objects i and/or j, there are three possibilities: the machine holds i only, the machine holds j only, the machine holds both i and j. For any remainder assignment for sequence $c_1 c_2 .. c_n$, let u_i be the number of machines (except the given two machines) holding only object i. Similarly define u_j. Because $c_i = k-2$ and $c_j = k$, we must have that $0 \le u_i = u_j - 2 < u_j \le k$. For any remainder assignment for sequence $b_1 b_2 ... b_n$, let v_i be the number of machines (except the given two machines) holding only object i. Similarly define v_j. Because $b_i = b_j = k-1$, we must have $0 \le v_i = v_j \le k-1$.

For any given integer w where $2 \le w \le k$, we consider the set Γ of all remainder assignments for sequence $c_1 c_2 ... c_n$ where $u_j = w$ and the set Λ of all remainder assignments for sequence $b_1 b_2 ... b_n$ where $v_j = w-1$. For the purpose of counting, we construct a many-to-many mapping from Γ to Λ. An assignment $\gamma \in \Gamma$ is mapped to another assignment $\lambda \in \Lambda$ if we can obtain λ by picking one of the u_j machines in γ that hold object j only and substituting object j with object i on that machine. We say that γ is the pre-image of λ and λ is the after-image of γ. Each γ has exactly $u_j = w$ after-images, while each λ has exactly $v_i = v_j = w-1$ pre-images. Thus it must be the case that $|\Gamma| < |\Lambda|$. Since this is true for all w, we know that $z \le y$.

Finally, the probability that the 2 machines hold exactly the same set of objects is thus smaller than $(\binom{n}{l} \cdot x)/(\binom{n}{l}\binom{n}{l} \cdot y) < (l/n)^l$.
□