Distrib. Comput. (2009) 21:367–384
DOI 10.1007/s00446-008-0072-2

Optimal inter-object correlation when replicating for availability

Haifeng Yu · Phillip B. Gibbons

Received: 23 August 2007 / Accepted: 9 October 2008 / Published online: 11 November 2008

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.

Keywords Multi-object operation · Inter-object correlation · Availability · Data replication · Object assignment

H. Yu (✉)
Department of Computer Science, School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543, Singapore
e-mail: haifeng@comp.nus.edu.sg

P. B. Gibbons
Intel Research Pittsburgh, 4720 Forbes Avenue, Suite 410, Pittsburgh, PA 15213, USA
e-mail: phillip.b.gibbons@intel.com

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 [7,13]). 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 [24] 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.

A simple yet subtle example Consider the example in Fig. 1 with four objects: A, B, C, and D. Each object has exactly two replicas.[1] We have four identical machines to hold these eight object replicas, and each machine holds exactly two 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 Fig. 1

[1] The same assignment problem also arises [24] when using erasure coding for the objects, but that is beyond the scope of this paper.

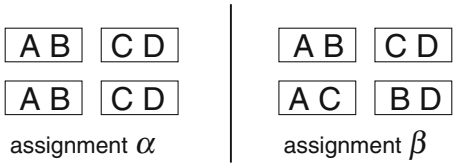

Fig. 1 Two possible assignments of four objects, *A*, *B*, *C*, and *D*, to four machines. Each *box* represents a machine

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 eight object replicas occupy the same number of machines (i.e., four machines) in α as in β, thus the difference does not result from "concentrating" or "spreading" the objects. (For example, if we spread the eight object replicas onto eight machines with one object replica per machine, then it will be clear that the probability of having all objects available will be lower than the probability in α.)

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 [3,5,25]. 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 [15], CFS [4], Chord [20], Coda [11], FARSITE [1], GFS [6], GHT [16], Glacier [7], Pastry [18], R-CHash [10], and RIO [19]) use different object assignments, which can yield dramatically different availability for multi-object operations. For example, it has been shown [24] that under practical settings, the failure probability of the TPC benchmark [22] can vary by four orders of magnitude under different assignments used in these previous systems. Thus, a thorough understanding of the subject is critical to guide system design.

Previous results We recently first identified [24] 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 Fig. 1). `RAND` is obtained by randomly assigning object replicas to machines. The simulation results [24] 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 [24] 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 [24] 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

[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 $FP(\beta) = p^4 + 4p^3(1-p)$.

[4] We are interested in the worst assignment as well because (i) previous simulation results [24] 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.

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 Fig. 1 already gave us some flavor of this, and later in Sect. 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 [2,8,9] 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 [3,5,25] 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 [3,5,25] 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 Fig. 1). Chain replication [17] 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.

To improve availability, some file systems [1,14] try to place related file blocks (or meta-data) on a small number of machines instead of spreading them over a large number of machines. The availability effect of such "concentration" or "spreading" is orthogonal to the effect of inter-object correlation in data replication, which is the focus of this paper. However, if the data on the small number of machines are in turn mirrored onto more machines, then the resulting assignment will be similar as our PTN assignment.

3 Formal model and assumptions

There are N (numbered) 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 identical replicas, and the object is considered *available* as long as any of the k replicas is available. There are s (numbered) 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 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 astrophysical image databases [21], 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. For example, most aggregation queries (e.g., "compute the average brightness of all galaxies") are likely to be able to tolerate some limited fraction of missing objects. On the other hand, a query of "check whether any of the images in the database contains the face of a terrorist" would likely require cheching all objects in the database, and thus cannot tolerate any missing object. See our earlier work [24] for a more detailed discussion on applications. This paper studies

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

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)

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 one-to-k mapping from the N objects to the s machines, where each object is assigned to k distinct machines and each machine holds exactly l objects. Notice that according to this definition, no machine can hold 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 Sect. 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 Sect. 7 prove the best/worst assignment for the two extremal t values when $n = N$. Section 8 explains why the results for $n = N$ easily extend to $n < N$. Finally, Sect. 9 generalizes our basic model (described above) to scenarios where different objects may have different number of replicas, and then proves the best (within constants) assignment for $t = n$.

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 $FP(\cdot)$ 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 [23], 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).* □

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 Sect. 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 case of $n = N$. This is also perhaps the most interesting case because when $n = N$, the objects requested fully occupy all the machines (as in Fig. 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 [2,8,9].[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.

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

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 $\{1, 2, \ldots, s\}$. 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:

$$\mu = \sum_{i=1}^{n} Pr[I_i]$$

$$\Delta = \sum_{(i,j):i \not\sim j} Pr[I_i \wedge I_j]$$

Notice that the sum in Δ is done over ordered pairs of i and j and the sum over unordered pairs would be $\Delta/2$.

Theorem 2 Janson's inequality [2,8,9]

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

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

Obviously, when $t = n$, we have $FP(\alpha) = 1 - Pr[I_1 = I_2 = \cdots = I_n = 0]$. 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 Fig. 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.

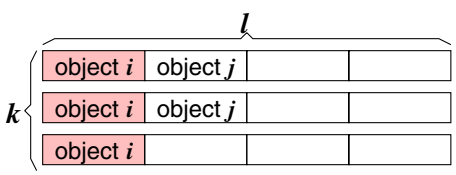

Fig. 2 Illustrating the Δ term in Janson's inequality, for $k = 3$ and $l = 4$. Each *row* is a machine

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).

Lemma 1 *For $1 \leq x_1, x_2 \leq z$, we have*:

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

Proof

- Without loss of generality, assume $x_1 \leq x_2$. Let $q = 1/p \geq 2$. We have $q^{x_1-x_2} \leq 1 \Rightarrow q^{x_1-x_2} + 1 \leq 2 \leq q^{x_1} \Rightarrow q^{x_1} + q^{x_2} \leq q^{x_1+x_2}$.
- Let $d = x_1 + x_2$ and define $f(x) = p^{-x} + p^{-(d-x)}$ for $d - z \leq x \leq z$. We only need to prove that $f(x_1) \leq f(z)$. 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 $f(x_1) \leq f(z)$. □

Lemma 2 *Consider $1 \leq x_1, x_2, \ldots, x_u \leq z$ where $\sum_{i=1}^{u} x_i = y$. We have $p^{2k-x_1} + p^{2k-x_2} + \cdots + p^{2k-x_u} \leq \lceil \frac{y}{z} \rceil p^{2k-z}$ when $p \leq 0.5$.*

Proof It suffices to show that $p^{-x_1} + p^{-x_2} + \cdots + p^{-x_u} \leq \lceil \frac{y}{z} \rceil p^{-z}$. We combine the terms on the left-hand side step by step. If $x_i + x_j \leq z$, 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 1, we know that the summation is never decreased at any of these steps. Thus $p^{-x_1} + p^{-x_2} + \cdots + p^{-x_u} \leq \lceil \frac{y}{z} \rceil p^{-z}$. □

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

[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.

x's be $x_1, x_2, \ldots, x_u$. Clearly $1 \le x_1, x_2, \ldots, x_u \le k$ and let $y = x_1 + x_2 + \cdots + x_u \le (l-1)k$ (Fig. 2). For the given i, we have $\sum_{j:j \not\sim i} Pr[I_i \wedge I_j] = p^{2k-x_1} + p^{2k-x_2} + \cdots + p^{2k-x_u}$. Lemma 2 shows that this summation is upper bounded by $\lceil \frac{y}{k} \rceil p^k \le (l-1)p^k$. When we consider all possible i's, we have:

$$\Delta = \sum_{(i,j):i \not\sim j} Pr[I_i \wedge I_j] \le n(l-1)p^k$$

Finally, from Inequality (2) and with $\mu = np^k$, we have:

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

□

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 $\varepsilon > 0$:*

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

Proof Theorem 3 tells us that $FP(\alpha) \ge 1 - e^{-(n/l)p^k}$. Let $x = n/l$ where $x \ge 1$, $y = p^k$ where $0 < y \le 0.5^2 = 0.25$, 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 $g(y)' \ge 0$ for any y. Thus $f(1, y)$ is a monotonically increasing function of y, and we have $f(x, y) \le f(1, y) \le f(1, 0.25) < 1.14$. 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 $\lim_{x \to \infty} f(x, y) \to 1$. □

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 Fig. 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 $p \le 0.5$, 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 Fig. 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. 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 $t = n = N$, $k \ge 3$, and $2lp^{\lfloor k/2 \rfloor} \le 1$, for any assignment α and any constant $\varepsilon > 0$, with probability of at least $1 - O(1/n)$:*

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

Proof We know that for any assignment $\mu = np^k$. Let

$$q = n \sum_{i=\lceil k/2 \rceil + 1}^{k} (kl/n)^i$$

Notice that when $k \ge 3$, $q = O(n \cdot (1/n)^{\lceil k/2 \rceil + 1}) = O(1/n^2)$. We will show that for RAND:

$$Pr[\Delta > 2nlp^{k+\lfloor k/2 \rfloor}] \le nq = O(1/n)$$

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, imagine that 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 $x_1 + x_2 + \cdots + x_u = (l-1)k$. Let $z = max(x_1, x_2, \ldots, x_u) \le k$. For the given i, Lemma 2 tells us:

$$\begin{aligned} \sum_{j \not\sim i} Pr[I_i \wedge I_j] &= p^{2k-x_1} + p^{2k-x_2} + \cdots + p^{2k-x_u} \\ &\le \lceil \frac{(l-1)k}{z} \rceil p^{2k-z} \\ &\le \frac{kl}{z} \cdot p^{2k-z} \end{aligned}$$

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 7 (in the Appendix), which tells us:

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

[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.

Because $p \leq 0.5$, one can show that $h(1) \leq h(2) < h(3) < h(4) < \ldots$. 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 $\lceil k/2 \rceil \geq 2$). Thus for given i:

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

When considering all possible i's, we have:

$$Pr[\Delta > 2nlp^{k+\lfloor k/2 \rfloor}] \leq nq = O(1/n)$$

Thus Inequality (2) tells us that with at least $1 - O(1/n)$ probability:

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

From Theorem 3, we know that $FP(\alpha) \leq 1 - (1-p^k)^n$. Let $x = n$ where $1 \leq x$, $y = p^k$ where $0 < y \leq 0.5^2 = 0.25$, 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 $g'(y) \leq 0$ for any y. Thus $f(1, y)$ is a monotonically decreasing function of y, and we have $f(x, y) \geq f(1, y) \geq f(1, 0.25) > 0.46$. Furthermore, we also have $\lim_{x \to \infty} f(x, y) \to 1$ and $\lim_{y \to 0} f(x, y) \to 1$. □

We construct a customized proof for $k = 2$:

Theorem 6 *When $t = n = N$, $k = 2$, and $4lp \leq 1$, for any assignment α and any constant $\varepsilon > 0$, with probability of 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, $FP(\texttt{RAND}) > (1-\varepsilon)FP(\alpha)$.*

Proof We know that for any assignment $\mu = np^2$. 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, imagine that 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 $1 \leq x_1, x_2, \ldots, x_u \leq 2$ and $x_1 + x_2 + \cdots + x_u = 2(l-1)$. By definition, we have:

$$\sum_{j \not\sim i} Pr[I_i \wedge I_j] = p^{2k-x_1} + p^{2k-x_2} + \cdots + p^{2k-x_u}$$

Different from the proof of Theorem 5, here it is not sufficient to simply reason about the (small) probability of having a large x_j in the sequence. Rather, we need to be more precise and reason about the number of large x_j's. Fortunately, $k = 2$ simplifies such reasoning. For the given i, let y_i be the number of x_j's that are 2. Lemma 6 (in the Appendix) tells us that the expectation of y_i is smaller than $n \cdot (k^2/s)^2 = 4l^2/n$. Now consider all y_i's for $1 \leq i \leq n$ and define $y = \sum_{i=1}^{n} y_i$. We have $E[y] < 4l^2$. Applying a Markov inequality will show that $Pr[y \geq 0.5n] \leq 8l^2/n = O(1/n)$.

When $k = 2$, Δ is a summation of terms in the form of p^{4-x_j} where x_j is either 1 or 2. In other words, the terms appearing in the summation are either p^2 or p^3. The sum of all x_j's, across all j and all i, is exactly $2(l-1)n$. Thus the total number of p^3 terms in the summation is at most $2nl$. The number of p^2 terms is exactly y. Thus with $1 - O(1/n)$ probability, we have:

$$\Delta \leq 2nl \cdot p^3 + 0.5n \cdot p^2,$$

and also:

$$\begin{aligned} FP(\texttt{RAND}) &\geq 1 - e^{(-n^2p^4)/(np^2 + 2nlp^3 + 0.5np^2)} \\ &= 1 - e^{-np^2/(1.5+2lp)} \geq 1 - e^{-0.5np^2} \end{aligned}$$

The rest of the proof is the same as the second half of the proof for Theorem 5. □

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 $p \leq 0.5$, 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.

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

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

$\{G | i \in G,\ |G| = v,$ and G 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 $FP(\texttt{PTN}) \geq FP(\alpha)$, 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 $|U_{i,v}| \leq \binom{k-1}{v-1}$ for $1 \leq v \leq k$ and $|U_{i,v}| = 0$ for $k + 1 \leq v \leq s$. On the other hand, in PTN, we have exactly $|U_{i,v}| = \binom{k-1}{v-1}$ for $1 \leq v \leq k$. 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 8 *When $t = l + 1 < n = N$, for any assignment α and any positive constant ε, we have $E[FP(\texttt{RAND})] \leq (1 + \varepsilon)FP(\alpha)$ 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 $FP(\alpha) \geq p^s + s(1-p)p^{s-1}$.

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 $E[|U_i|] = E[\sum_j x_{i,j}] = \binom{s}{i} E[x_{i,1}] \leq s^i E[x_{i,1}]$. 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 $E[|U_i|] \leq s^k E[x_{2,1}]$.

Determining the probability (i.e., $E[x_{2,1}]$) of having exactly l distinct objects on a given set of two 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 8 in the Appendix proves that $E[x_{2,1}] < (l/n)^l \leq (l/n)^k$. Thus, we have $E[FP(\texttt{RAND})]$ as:

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

□

Corollary 1 *For any assignment α and any positive constants ε and δ, when $t = l + 1 < n = N$ and n is sufficiently large, $Pr[FP(\texttt{RAND}) \geq (1+\varepsilon)FP(\alpha)] \leq \delta$.*

Proof To prove this corollary, we need to translate the expectation guarantee from Theorem 8 to a high probability guarantee. We will apply a Markov inequality, in a careful way. Let β be the assignment with the smallest failure probability. Obviously such β exists, since there are only finite number of possible assignments. By definition of β, we always have $FP(\texttt{RAND})/FP(\beta) \geq 1$.

To prove the corollary, we only need to show that $Pr[(FP(\texttt{RAND})/FP(\beta)) \geq (1+\varepsilon)] \leq \delta$. We prove by contradiction and assume that $Pr[(FP(\texttt{RAND})/FP(\beta)) \geq (1+\varepsilon)] > \delta$. This means that $E[FP(\texttt{RAND})/FP(\beta)] > 1 \cdot (1-\delta) + (1+\varepsilon) \cdot \delta = 1 + \varepsilon\delta$. On the other hand, Theorem 8 tells us that $E[FP(\texttt{RAND})] \leq (1+\varepsilon\delta)FP(\beta)$ when n is sufficiently large. Contradiction. □

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 (Fig. 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.

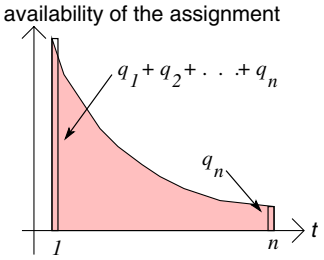

Fig. 3 Illustrating the area bounded by the availability curve. The curve is actually a step function, so the area bounded by the curve exactly equals the sum of the rectangular areas

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

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

On the other hand, the summation $\sum_{i=1}^{n} i q_i$ 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 $\sum_{i=1}^{n} i q_i = n(1-p^k)$, which is independent of the assignment.

7.2 Removing constant factors when $t = n = N$ and $k = l = 2$

For $t = l+1$, we have already obtained the best and the worst assignments without any constant factor. While for $t = n$, the best and the worst assignments are within constant factors (Theorems 4 and 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 two replicas and each machine holds two objects (as in Fig. 1). We will see 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 (Fig. 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

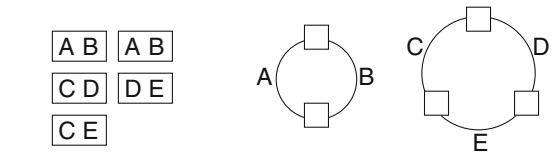

Fig. 4 When $k = l = 2$, each assignment can be uniquely represented as a set of rings. Each *box* represents a machine

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 $FP(\cdot)$ 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 3

$f(x) = z_1^x + z_2^x$, *where*: $q = \sqrt{(3p+1)(1-p)}$,
$z_1 = (1 - p + q)/2,\ z_2 = (1 - p - q)/2$

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 $x \ge 0$:

$g(x) = az_1^x + bz_2^x$, where:
$a = (q + 1 + p)/(2q),\ b = (q - 1 - p)/(2q)$,
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 4 *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 9 *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 4 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 $1 \ldots n$. In a random permutation, the expected number of cycles of size

$m \leq n$ is $1/m$ and the average cycle size is $\Theta(n/\log n)$ [12]. Such structure is obviously quite different from $n/3$ size-3 rings.

Theorem 10 *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 4 allows two kinds of adjustments:

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

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, three rings of size 25 is better than five rings of size 15.
- If x is even, then the larger x is, the worse the availability. For example, four rings of size 12 is worse than six 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, whereas we increase x, the availability oscillates with a decreasing oscillation magnitude. Notice that these results, however, do not necessarily imply Theorems 9 and 10, which also apply to rings of heterogeneous sizes.

Theorem 11

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

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

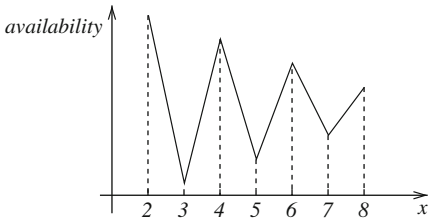

Fig. 5 Illustrating the availability of the assignment corresponding to n/x rings of size x

- *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 3 that $f(x) = z_1^x - z_3^x$ and $f(y) = z_1^y - z_3^y$:

$$(f(x))^{n/x} > (f(y))^{n/y} \Leftrightarrow (z_1^x - z_3^x)^y > (z_1^y - z_3^y)^x$$
$$\Leftrightarrow (1 - (z_3/z_1)^x)^y > (1 - (z_3/z_1)^y)^x$$

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 Extending the previous results 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 Sect. 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 we are ready to explore the best/worst assignments for $t = n < N$. Both Theorems 3 and 4 (together with the exact proofs) apply to $t = n < N$ without modification. Thus PTN is optimal within a constant of 1.14. Theorems 5 and 6 (together with the proofs) apply to $t = n < N$ once we substitute l with nk/s in the theorems and proofs. This 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 they are obviously 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 modeling 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 [24] has investigated these scenarios for some specific applications.

9 Heterogeneous replication degrees

So far we have obtained a complete set of results under the model from Sect. 3, where all objects have the same number (k) of replicas. One can easily imagine that in some applications, different objects may have different importance. For example, in a file system, some files may be deemed as more important than others, and metadata blocks (containing directory information) are often considered to be more critical than data blocks. Naturally, systems designers will use more replicas for important objects to protect them. In other words, the objects in the system will have different number of replicas or heterogeneous *replication degrees*.

This section thus aims to study the object assignment problem under this generalized model with heterogeneous replication degrees. We first define the generalized model. The system has N objects, from object 1 to object N. Object i ($1 \le i \le N$) has k_i ($k_i \ge 1$) replicas and k_i is called the *replication degree* of object i . The value of k_i is determined by the system designer based on the importance of object i. The s machines each hold exactly l objects, with $s \cdot l = \sum_{i=1}^{N} k_i$.

Our results under heterogeneous replication degrees are not as general as for homogeneous replication degrees, due to the difficulty of the problem. Instead of finding out the best and worst assignment for $t=n$ and $t=l+1$ (i.e., four cases), here we have results only regarding the best assignment for $t=n$. This, however, is perhaps the most important case in practice among the four cases.

At a high level, we will show that the failure probability of the PTN assignment, when properly generalized to heterogeneous replication degrees, is at most $\bar{k}$ multiplicative factor from the optimal. Here $\bar{k}$ is the average replication degree across all n objects requested by the operation. Replication degrees (especially the average replication degree) in a real replication system tend to be small (e.g., below 5). Furthermore in practice, because failure probability or availability can span a wide range, it is usually discussed in log-scale. For example, system designers often use "number of nines" as the measure for availability, where x number of nines means that the availability is $1-0.1^x$. A multiplicative factor of 5 in failure probability would then translate to less than one nine's difference. Thus we do not expect this $\bar{k}$ factor to be significant in practice.

It is worth noting that the proving technique we use to obtain this result is different from before. Earlier in Sect. 5, we used Janson's inequality to obtain a lower bound on $FP(\alpha)$, and then some straightforward calculation showed that $FP(\text{PTN})$ is close to the lower bound. With heterogeneous replication degrees, Janson's inequality can no longer provide a good enough lower bound. Instead, we obtain a lower bound via elementary combinatorial techniques. We then carefully show that the properly generalized PTN assignment can approach the lower bound within $\bar{k}$ factor.

In the next, without loss of generality, we assume that the n objects requested by the operation are objects 1 through n. Also without loss of generality, we further assume that $1 \le k_1 \le k_2 \le \cdots \le k_n$. We let $\bar{k} = \sum_{i=1}^{n} k_i/n$. We do not (in fact we cannot) assume that $k_n \le k_{n+1} \le \cdots \le k_N$.

9.1 Lower bound

We will first obtain a lower bound on $FP(\alpha)$ for any α, by finding out a set of mutually independent objects (out of the n objects requested). In other words, no machine holds more than one object from that set. For the assignment to be available under $t=n$, all objects in that set must be available. This then yields an upper bound on availability and a lower bound on $FP(\,)$.

Obviously, the lower bound depends on the set of objects we find and the lower bound is stronger if (i) the number of objects in the set is larger, and (ii) the replication degree (and thus the availability) of individual objects in the set is smaller. Further remember that we are trying to obtain a lower bound for any α, so we cannot leverage any special structure of α. To obtain a strong lower bound, we first include object 1 in the set, since its replication degree is the smallest. Object 1 has k_1 replicas on k_1 different machines. The number of distinct objects (including object 1 itself) that reside on these k_1 machines is at most $z_1 = (l-1)k_1 + 1$. Namely, there can only be at most z_1 objects that are correlated with object 1. We then exclude these z_1 objects, and pick (out of the remaining $n - z_1$ objects) a second object with the smallest replication degree to include in the set. Obviously, this second object must have a replication degree of no larger than k_{z_1+1}. We can continue such process and keep adding objects to the set until we run out of objects. We will eventually reach the following theorem with an elementary proof. The only slightly tricky part in the proof is to appropriately avoid circular arguments.

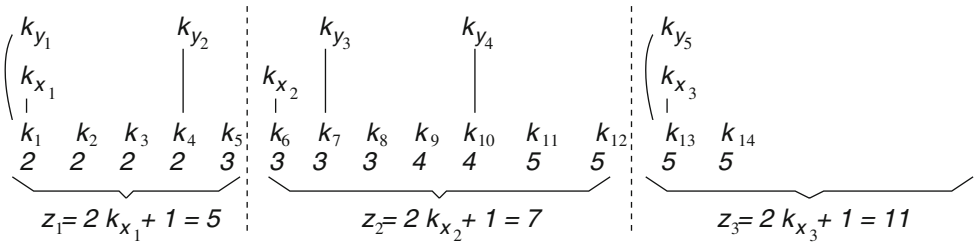

Fig. 6 Illustrating the definitions of x_i, z_i, and y_i in Theorems 12 and 13. We assume $l = 3$ and $N = n = 14$ in the example

Theorem 12 *When $t = n$, for any assignment α, we have*:

$$FP(\alpha) \geq 1 - (1 - p^{k_{x_1}})(1 - p^{k_{x_2}}) \ldots (1 - p^{k_{x_u}}),$$

where x_i is recursively defined as following and u is the maximum index such that $x_u \leq n$:

$$\begin{cases} x_1 = 1 \\ z_i = (l-1)k_{x_i} + 1 & \text{for } i \geq 1 \\ x_{i+1} = x_i + z_i & \text{for } i \geq 1 \end{cases}$$

Proof Figure 6 illustrates the definitions for x_i and z_i. We find integers $x'_1, x'_2, \ldots, x'_{u'}$ (where $1 = x'_1 < x'_2 < \cdots < x'_{u'} \leq n$) in the following way. We start from the sequence of objects "1, 2, ... n". We pick the first object in the sequence (i.e., object 1) and let x'_1 be the name of that object (i.e., let $x'_1 = 1$). We then delete from the sequence all objects that are correlated with object x'_1 in the given assignment α (including object 1 itself). We next pick the first object in the remaining sequence and let x'_2 be the name of that object. We then again delete from the remaining sequence all objects that are correlated with object x'_2 in α. The process is repeated until the sequence becomes empty. Let $x'_{u'}$ be the name of the last object picked.

Obviously, all these objects $x'_1, x'_2, \ldots, x'_{u'}$ are mutually independent. Thus the probability that they are all available is exactly $(1 - p^{k_{x'_1}})(1 - p^{k_{x'_2}}) \ldots (1 - p^{k_{x'_{u'}}})$. In order for assignment α to be available under $t = n$, all these objects must be available. So we immediately have:

$$FP(\alpha) \geq 1 - (1 - p^{k_{x'_1}})(1 - p^{k_{x'_2}}) \ldots (1 - p^{k_{x'_{u'}}})$$

To prove the theorem, it suffices to show that $u' \geq u$ and $x_i \geq x'_i$ for $1 \leq i \leq u$, which would imply:

$$\begin{aligned} FP(\alpha) &\geq 1 - (1 - p^{k_{x'_1}})(1 - p^{k_{x'_2}}) \ldots (1 - p^{k_{x'_{u'}}}) \\ &\geq 1 - (1 - p^{k_{x_1}})(1 - p^{k_{x_2}}) \ldots (1 - p^{k_{x_u}}) \end{aligned}$$

Define z'_i to be the number of objects deleted from the object sequence after we find object x'_i, for $1 \leq i \leq u'$. Let $w = \min(u, u')$. We prove via induction the following for $1 \leq i \leq w$:

$$\begin{cases} x'_i \leq x_i \\ z'_i \leq z_i \end{cases} \tag{7}$$

For the induction base, we trivially have $x'_1 = 1 \leq 1 = x_1$ and $z'_1 \leq (l-1)k_1 + 1 = z_1$. Now assume the above inequalities hold up to i and we consider $i + 1$. We already have $z'_j \leq z_j$ for all $1 \leq j \leq i$. This means that

$$\sum_{j=1}^{i} z'_j \leq \sum_{j=1}^{i} z_j = x_{i+1} - 1$$

Object x'_{i+1} is the first object in the sequence of "1, 2, ..., n", after deleting from the sequence some (unknown) set of $\sum_{j=1}^{i} z'_j$ objects. Thus we obviously have $x'_{i+1} \leq \sum_{j=1}^{i} z'_j + 1 = x_{i+1}$. In any assignment, there can be at most $(l-1)k_{x'_{i+1}} + 1$ objects that are correlated with object x'_{i+1}. So we immediately have $z'_{i+1} \leq (l-1)k_{x'_{i+1}} + 1$. Now because $x'_{i+1} \leq x_{i+1}$ and also because k_j is monotonically increasing with j, we have $z'_{i+1} \leq (l-1)k_{x'_{i+1}} + 1 \leq (l-1)k_{x_{i+1}} + 1 = z_{i+1}$. This completes the inductive step.

We have proved that Inequality (7) holds. We prove via contradiction that it is impossible for $u' < u$. Notice that even if $u' < u$, we still have $\sum_{j=1}^{u'} z'_j \leq \sum_{j=1}^{u'} z_i = x_{u'+1} - 1$ from Inequality (7). Since $u' + 1 \leq u$ and $x_{u'+1} \leq x_u \leq n$, we must have $\sum_{j=1}^{u'} z'_j \leq n - 1$. This means that after picking $x'_{u'}$, we will have deleted at most $n - 1$ objects from the original sequence. Thus there is at least one object and we should be able to pick $x'_{u'+1}$. Contradiction. □

The lower bound in Theorem 12 of course, applies to homogeneous replication degree as well. In such case, the lower bound will be $1 - (1 - p^k)^{\lceil \frac{n}{(l-1)k+1} \rceil}$. When p is small, this is roughly $\frac{n}{l \cdot k} p^k$ and is about k times smaller/weaker than the lower bound of $1 - e^{-(n/l)p^k} \approx \frac{n}{l} p^k$ from Theorem 3.

9.2 Approaching the lower bound using generalized PTN

We would like to show that some appropriate generalization of the PTN assignment will approach the lower bound on $FP(\,)$ within some factor. Figure 7 provides an example for how we generalize PTN. We use k_1 machines to hold objects 1 through l, where each machine holds exactly l objects and each object exactly has k_1 replicas on these machines. Since $k_1 \leq k_2 \leq \cdots \leq k_l$, It is possible that the replication degree k_i of some object i $(2 \leq i \leq l)$ is larger than k_1, but we

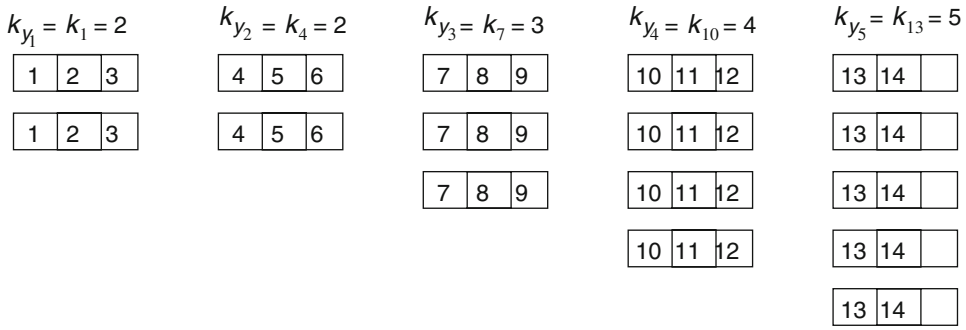

Fig. 7 Illustrating the definition of PTN under heterogeneous replication degrees. We assume the same N, n, l, and k_i's as in Fig. 6. Each *box* corresponds to a machine, which holds exactly three objects. Notice that objects 5, 6, 9, 11, and 12 each need one additional replica, according to their replication degrees. These five additional replicas can be place in an arbitrary way into the five empty "slots" in figure

will not assign the additional $k_i - k_1$ replicas of that object for now. Next we use k_{l+1} machines to hold objects $(l+1)$ through $2l$. Again, some objects may need to have more than k_{l+1} replicas. We continue such process until we are done with object n. Finally, remember that some objects (between object 2 and object n) may need to have more replicas, and we also need to assign objects $n+1$ through N. We will simply assign all these additional replicas to the remaining "slots" on the machines in an arbitrary way. Because $l \cdot s = \sum_{i=1}^{N} k_i$, we are guaranteed to have enough "slots" to hold all these left-over replicas. It is easy to see that when $k_1 = k_2 = \cdots = k_n = k$, our new PTN definition is exactly the same as the old PTN definition, which means that it is indeed a generalization.

Notice that strictly speaking, according to the above definition, PTN is not a single assignment any more. Rather, it is a set of assignments where each assignment corresponds to a specific way of assigning the left-over replicas to the remaining "slots". To simplify notation, when we write PTN below, we mean any assignment in the above set. We will show that all assignments in the set actually have the same failure probability and are near-optimal. In other words, how we assign the left-over replicas of objects 2 through n or how many left-over replicas they have does not affect failure probability. Intuitively, this is because objects 1, $(l+1)$, $(2l+1)$, ... (with replication degree of $k_1, k_{l+1}, k_{2l+1}, \ldots$) are the "bottleneck" for availability.

Theorem 13 *When $t = n$, we have*:

$$FP(\texttt{PTN}) = 1 - (1 - p^{k_{y_1}})(1 - p^{k_{y_2}}) \ldots (1 - p^{k_{y_v}}),$$

where y_i is recursively defined as following and v is the maximum index such that $y_v \leq n$:

$$\begin{cases} y_1 = 1 \\ y_{i+1} = y_i + l \quad \text{for } i \geq 1 \end{cases}$$

Proof Figure 6 illustrates the definition for y_i and Fig. 7 illustrates the corresponding PTN assignment. In PTN, if object 1 is available, then all objects from object 1 thorough object l must be available. Similarly, if object $(l+1)$ is available, then all objects $(l+1)$ through $2l$ must be available. Thus, if objects $y_1, y_2, \ldots, y_v$ are all available, then all the n objects must be available. The availability of the assignment is thus at least $(1-p^{k_{y_1}})(1-p^{k_{y_2}}) \ldots (1-p^{k_{y_v}})$. On the other hand, for the assignment to be available, object 1, $(l+1)$, $(2l+1)$, ..., must all be available. Further because they are mutually independent, we know that the availability of the assignment is at most $(1 - p^{k_{y_1}})(1 - p^{k_{y_2}}) \ldots (1 - p^{k_{y_v}})$. □

We would like to draw a clean connection between the lower bound from Theorem 12 and $FP(\texttt{PTN})$ from Theorem 13. The connection is somewhat obscure. After several unsuccessful attempts, we conjectured that the two values are roughly $\bar{k}$ apart. We formalize and prove such claim in the next. We will only consider sufficiently small p, where the lower bound from Theorem 12 approaches $\sum_{i=1}^{u} p^{k_{x_i}}$ and the $FP(\texttt{PTN})$ from Theorem 13 approaches $\sum_{j=1}^{v} p^{k_{y_j}}$.

Figure 6 helps to give an intuitive comparison between the two summations. Namely, there are more y_i's than x_i's, and both sequences tend to spread out (though in different ways) over the region of $[1, n]$. To compare the two summations, we would like to somehow convert y_j's into x_i's. For any $1 \leq i \leq u$, we define the interval *covered* by x_i to be the interval of $[x_i, x_{i+1}-1]$. If a certain y_j falls within the interval covered by x_i, we say that y_j is *covered* by x_i. We group all y_j's into groups where each group is covered by a distinct x_i. For example, in Fig. 6, y_1 and y_2 are covered by x_1, y_3 and y_4 are covered by x_2, while y_5 is covered by x_3. Obviously, all y_j's covered by an x_i are at least as large as that x_i. When upper bounding $FP(\texttt{PTN})$, we can treat all these y_j's as x_i. We will also show that the number of y_j's covered by x_i is at most k_{x_i}. All these will lead to the following useful transformation:

Lemma 5

$$\sum_{j=1}^{v} p^{k_{y_j}} \leq \sum_{i=1}^{u} (k_{x_i} \cdot p^{k_{x_i}}),$$

where x_i and y_j are as defined in Theorems 12 and 13.

Proof The interval covered by x_i is of size $x_{i+1} - x_i = (l-1)k_{x_i}+1$. Because consecutive y_i's are l apart, the number of y_j's that fall within an interval of size $(l-1)k_{x_i}+1$ is at most

$$1 + \left\lfloor \frac{(l-1)k_{x_i}}{l} \right\rfloor = 1 + \left\lfloor k_{x_i} - \frac{k_{x_i}}{l} \right\rfloor \leq 1 + k_{x_i} - 1 = k_{x_i}$$

Any y_j covered by x_i is at least as large as x_i, thus for any given i, we have:

$$\sum_{j\,:\,x_i \leq y_j < x_{i+1}} p^{k_{y_j}} \leq k_{x_i} \cdot p^{k_{x_i}}$$

Finally we have:

$$\sum_{j=1}^{v} p^{k_{y_j}} = \sum_{i=1}^{u} \left(\sum_{j\,:\,x_i \leq y_j < x_{i+1}} p^{k_{y_j}} \right) \leq \sum_{i=1}^{u} (k_{x_i} \cdot p^{k_{x_i}})$$

□

Besides Lemma 5, our proof next will also use a well-known elementary inequality, the Chebyshev sum inequality. For completeness, we include this inequality below:

Theorem 14 Chebyshev Sum Inequality *For any two sequences where* $0 \leq a_1 \leq a_2 \leq \cdots \leq a_n$ *and* $0 \leq b_1 \leq b_2 \leq \cdots \leq b_n$, *we have*:

$$\frac{a_1 b_1 + \cdots + a_n b_n}{n} \geq \frac{a_1 + \cdots + a_n}{n} \cdot \frac{b_1 + \cdots + b_n}{n}. \quad (8)$$

Similarly, for any two sequences where $0 \leq a_1 \leq a_2 \leq \cdots \leq a_n$ *and* $b_1 \geq b_2 \geq \cdots \geq b_n \geq 0$, *we have:*

$$\frac{a_1 b_1 + \cdots + a_n b_n}{n} \leq \frac{a_1 + \cdots + a_n}{n} \cdot \frac{b_1 + \cdots + b_n}{n}. \quad (9)$$

Theorem 15 *For any assignment* α *and any constant positive constant* ε, *when* $t = n$ *and when* p *is sufficiently small, we have:*

$$FP(\texttt{PTN}) \leq (1+\varepsilon) \cdot \frac{\sum_{i=1}^{u} k_{x_i}}{u} \cdot FP(\alpha),$$

where x_i*'s are as defined in Theorem 12.*

Proof We only need to show that $\lim_{p\to 0} FP(\texttt{PTN})/FP(\alpha) \leq \sum_{i=1}^{u} k_{x_i}/u$. We have:

$$\begin{aligned}
\lim_{p\to 0} \frac{FP(\texttt{PTN})}{FP(\alpha)} &\leq \frac{\sum_{j=1}^{v} p^{k_{y_j}}}{\sum_{i=1}^{u} p^{k_{x_i}}} \text{ (from Theorems 12 and 13)} \\
&\leq \frac{\sum_{i=1}^{u} (k_{x_i} \cdot p^{k_{x_i}})}{\sum_{i=1}^{u} p^{k_{x_i}}} \text{ (from Lemma 5)} \\
&\leq \frac{\sum_{i=1}^{u} k_{x_i}}{u} \text{ [from Inequality (9)]}
\end{aligned}$$

□

Corollary 2 *For any assignment* α *and any constant positive constant* ε, *when* $t = n$ *and when* p *is sufficiently small, we have* $FP(PTN) \leq (1+\varepsilon) \cdot k_n \cdot FP(\alpha)$.

Proof Directly from Theorem 15. □

The quantity of $\frac{\sum_{i=1}^{u} k_{x_i}}{u}$ is the average replication degree of those u objects we picked. Because these u objects are spread across all n objects, in many cases, $\frac{\sum_{i=1}^{u} k_{x_i}}{u}$ is related to $\overline{k}$:

Corollary 3 *For any assignment* α *and any constant positive constant* ε, *when* $t = n$, $x_{u+1} = n+1$, *and* p *is sufficiently small, we have* $FP(\texttt{PTN}) \leq (1+\varepsilon) \cdot \overline{k} \cdot FP(\alpha)$.

Proof Given Theorem 15, we only need to prove that $\frac{\sum_{i=1}^{u} k_{x_i}}{u} \leq \overline{k}$. Notice that if $x_{u+1} = n+1$, we must have $n = \sum_{i=1}^{u} z_i$. From Eq. (8), we have:

$$\frac{\sum_{i=1}^{u} k_{x_i}}{u} \leq \frac{\sum_{i=1}^{u} k_{x_i} z_i}{\sum_{i=1}^{u} z_i} = \frac{\sum_{i=1}^{u} k_{x_i} z_i}{n} \leq \frac{\sum_{i=1}^{n} k_i}{n} = \overline{k}$$

□

Figure 6 helps to provide a better intuition into the role of the condition $x_{u+1} = n+1$. In Fig. 6, $x_u = x_3 = 13$, $x_{u+1} = x_4 = 24$, and $n = 14$. Thus the (last) interval covered by x_u (i.e., [13, 23]) is not fully "populated" with 11 k_i's. Rather, the interval only has k_{13} and k_{14}. Notice that if we did have k_{15} through k_{23}, they would all be at least as large as k_{14}. Thus the non-existence of k_{15} through k_{23} brings down the average and thus invalidates Corollary 3.

A closer examination however, shows that the requirement of $x_{u+1} = n+1$ is overly restrictive. After all, only the very last interval is impacted if $x_{u+1} > n+1$. When we have a lot of objects (and machines), the number of intervals will be large as well. The effect of the last interval should then be diminished:

Corollary 4 *For any assignment* α *and any constant positive constant* ε, *when* $t = n$, n *is sufficiently large,*[9] *and* p *is sufficiently small, we have* $FP(\texttt{PTN}) \leq (1+\varepsilon) \cdot \overline{k} \cdot FP(\alpha)$.

Proof Given Theorem 15, it suffices to show that $\frac{\sum_{i=1}^{u} k_{x_i}}{u} \leq \overline{k}$ when $n \to \infty$. The basic idea is to extend the current k_i sequence so that the extended sequence can satisfy the condition in Corollary 3. Then we will show that such extension does not affect $\overline{k}$ for $n \to \infty$.

Define $n' = x_{u+1} - 1$. We have:

$$\begin{aligned}
\lim_{n\to\infty} \frac{n'-n}{n} &= \lim_{n\to\infty} \frac{x_u + z_u - 1 - n}{n} \\
&\leq \lim_{n\to\infty} \frac{n + z_u - 1 - n}{n} \\
&< \lim_{n\to\infty} \frac{z_u}{n} \leq \lim_{n\to\infty} \frac{(l-1)k_n + 1}{n} = 0
\end{aligned}$$

[9] Similarly as in Sect. 5, for "sufficiently large" n, we assume that l is fixed and k_i is upper bounded by some constant for all i, while s changes with n (and N). As explained earlier, this follows the practical meaning of the problem.

We consider a second sequence of k'_i's for $1 \le i \le n'$, where $k'_i = k_i$ for $1 \le i \le n$ and $k'_i = k_n$ for $n < i \le n'$. Obviously, this second sequence satisfies the condition in Corollary 3 and we have:

$$\frac{\sum_{i=1}^{u} k'_{x_i}}{u} \le \frac{\sum_{i=1}^{n'} k'_i}{n'}$$

We also have:

$$\begin{aligned}\lim_{n\to\infty} \frac{\sum_{i=1}^{n'} k'_i}{n'} &= \lim_{n\to\infty} \frac{\sum_{i=1}^{n} k'_i + \sum_{i=n+1}^{n'} k'_i}{n + (n' - n)} \\ &= \lim_{n\to\infty} \frac{\frac{\sum_{i=1}^{n} k_i}{n} + \frac{(n'-n)k_n}{n}}{1 + \frac{n'-n}{n}} \\ &= \lim_{n\to\infty} \frac{\sum_{i=1}^{n} k_i}{n} = \overline{k}\end{aligned}$$

Finally, we have for $n \to \infty$:

$$\frac{\sum_{i=1}^{u} k_{x_i}}{u} = \frac{\sum_{i=1}^{u} k'_{x_i}}{u} \le \frac{\sum_{i=1}^{n'} k'_i}{n'} = \overline{k}$$

□

It may appear that many inequalities we use so far (such as Lemma 5, Theorem 14, and other transformations) are quite loose and ad hoc. A natural question is whether we can strengthen the $\overline{k}$ factor in Corollary 4 by using tighter inequalities. That is unfortunately impossible, as all inequalities become equalities when $k_1 = k_2 = \cdots = k_n$. Thus to improve the factor of $\overline{k}$, it is necessary to improve the lower bound in Theorem 12.

10 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). There are many ways to further generalize our model (e.g., consider all possible t values or using erasure coding). Obtaining results under these generalizations is part of our future work.

Acknowledgments We thank Suman Nath for various discussions related to this paper. We also thank Avishai Wool and the anonymous reviewers of PODC'07 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.

Appendix

Lemma 6 *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. For any given object B (B ≠ A) and any i (1 ≤ i ≤ k), we have*:

$$Pr[\textit{machines } 1 \textit{ through } k \textit{ hold exactly } i \textit{ replicas of } B] < \left(\frac{k^2}{s}\right)^i$$

Proof By symmetry, we have:

$$\begin{aligned}&Pr[\text{machines 1 through } k \text{ hold exactly } i \\ &\text{replicas of } B]/\binom{k}{i} \\ &= Pr[\text{first } i \text{ machines hold } i \text{ replicas of } B \text{ and} \\ &\text{the next } k-i \text{ machines do not hold } B] \\ &< Pr[\text{machine 1 holds } B] \\ &\times Pr[\text{machine 2 holds } B \mid \text{machine 1 holds } B] \\ &\times \cdots \times Pr[\text{machine i holds } B \mid \text{machine} \\ &1 \ldots (i-1) \text{ hold } B] \end{aligned} \tag{10}$$

To simplify notation, we define the event "machine 0 holds B" to be an event that always happens. We will prove that for any $1 \le j \le i$:

$$\begin{aligned}&Pr[\text{machine } j \text{ holds } B | \text{machine } 0 \ldots (j-1) \text{ hold } B] \\ &< \frac{k-j+1}{s-j+1}\end{aligned} \tag{11}$$

It is important to understand that Inequality (11) is not trivial. To see why, consider an example where we have three objects A, B, C, where each object has two replicas. There are three (numbered) machines where each machine holds two objects. Imagine that we already know that machines 1 and 2 hold A, and machine 1 additionally holds a replica of B. There are three empty "slots" remaining, one "slot" from machine 2 and two "slots" from machine 3. The tricky part is that the second replica of B does not go into the three "slots" with equal probability. In fact, the second replica of B can never reside on machine 2, since that would cause machine 3 to hold two copies of C. Thus here we actually have $Pr[\text{machine 2 holds } B | \text{machine } 0 \ldots 1 \text{ hold } B] = 0$.

After A is already assigned to machines 1 through k, we define a *remainder assignment* to be a one-to-k mapping from the remaining $n-1$ (numbered) objects to the s (numbered) machines. Among these s machines, machines 1 through k will each take $l-1$ objects (except object A), while the rest of the machines will each take l objects. To prove Inequality (11), for a given j and any $j \le r \le s$, we define x_r to be the number of remainder assignments where machines 0 through $(j-1)$ hold B and machine r also holds B. Let x be the number of remainder assignments where machines 0 through $(j-1)$ hold B. For each remainder assignments where machines 0 through $(j-1)$ hold B, B has $k-(j-1) = k-j+1$ additional replicas besides those

replicas on machines 0 through $(j-1)$. Thus that remainder assignment is counted in $(k-j+1)$ different x_i's. This means $\sum_{r=j}^{s} x_r = (k-j+1)x$. For $j \le r \le s$, define:

$$p_r = Pr[\text{machine } r \text{ holds } B | \text{machine } 0 \ldots (j-1) \text{ hold } B] = \frac{x_i}{x}$$

By symmetry, we know that $p_j = p_{j+1} = \cdots = p_k$ and $p_{k+1} = \cdots = p_s$. Also, we have $p_j + p_{j+1} + \cdots + p_k + p_{k+1} + \cdots + p_s = k-j+1$. Thus to prove Inequality (11) or equivalently $p_j < (k-j+1)/(s-j+1)$, it suffices to show that $p_j < p_s$, which we prove in the following.

Let Γ be the set of remainder assignments where machines 1 through j hold object B, and let Λ be the set of remainder assignments where machines 1 through $j-1$ and also machine s hold 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 Inequality (11) holds. This means:

$$p_j < \frac{k-j+1}{s-j+1} \le \frac{k}{s} \quad (\text{because } 1 \le j \le k \le s)$$

Together with Inequality (10), we have:

$$Pr[\text{machines 1 through } k \text{ hold exactly } i \text{ replicas of } B] < \binom{k}{i} \cdot \left(\frac{k}{s}\right)^i < \left(\frac{k^2}{s}\right)^i$$

□

Lemma 7 *Consider the* RAND *distribution and a given object A. Without loss of generality assume that machines 1 through k each hold 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 for any a where $1 \le a \le k$. we have $Pr[z \ge a] < n \sum_{i=a}^{k} (kl/n)^i$.*

Proof Lemma 6 shows that for any given object B ($B \ne A$), the probability of these k machines holds exactly i replicas of B is at most $(k^2/s)^i$. Take a summation for i from a to k, and also apply a union bound across all possible B's:

$$Pr[z \ge a] < n \sum_{i=a}^{k} (k^2/s)^i = n \sum_{i=a}^{k} (kl/n)^i$$

□

Lemma 8 *For any given two 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 two machines hold exactly the same set of objects while the second scenario does not have 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 two 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 two machines is $\binom{n}{l}\binom{n}{l}$ (remember that $k \ge 2$). 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 two machines, we let $k-2 \le a_1 \le a_2 \le \cdots \le a_n \le k$ denote the number of remaining replicas for the n objects. Obviously, $a_1 = \cdots = a_l = k-2$ and $a_{l+1} = \cdots = a_n = k$. The value of x does not depend on the identities of the objects, and is uniquely determined by the sequence $a_1 a_2 \ldots a_n$. We define $k-2 \le b_1 \le b_2 \le \cdots \le b_n \le k$ similarly in the second scenario. We know that $a_1 \le b_1, \ldots, a_l \le b_l$ and $a_{l+1} \ge b_{l+1}, \ldots, a_n \ge b_n$. Again, the value of y is uniquely determined by the sequence $b_1 b_2 \ldots b_n$.

The sequence $b_1 b_2 \ldots b_n$ 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 $a_1 a_2 \ldots a_n$ and $b_1 b_2 \ldots b_n$ 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 $b_1 b_2 \ldots b_n$, 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 $c_1 c_2 \ldots c_n$ from $b_1 b_2 \ldots b_n$ 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 $c_1 c_2 \ldots c_n$. Because $c_1 c_2 \ldots c_n$ has d pairs, based on our inductive assumption, we know that $x \leq z$. In the next, we will prove that $z \leq y$, thus finishing the inductive step.

For any machine (except the given two 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 $c_1 c_2 \ldots c_n$, 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 $0 \leq u_i = u_j - 2 < u_j \leq k$. For any remainder assignment for sequence $b_1 b_2 \ldots b_n$, 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 $0 \leq v_i = v_j \leq k-1$.

For any given integer w where $2 \leq w \leq k$, we consider the set Γ of all remainder assignments for sequence $c_1 c_2 \ldots c_n$ where $u_j = w$ and the set Λ of all remainder assignments for sequence $b_1 b_2 \ldots b_n$ 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 $z \leq y$. This finishes the proof for $x \leq y$.

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

References

1. Adya, A., Bolosky, W.J., Castro, M., Cermak, G., Chaiken, R., Douceur, J.R., Howell, J., Lorch, J.R., Theimer, M., Wattenhofer, R.P.: FARSITE: federated, available, and reliable storage for an incompletely trusted environment. In: USENIX OSDI (2002)
2. Alon, N., Spencer, J.H.: The probabilistic method. Wiley, New York (2000)
3. Bolosky, W.J., Douceur, J.R., Ely, D., Theimer, M.: Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In: ACM SIGMETRICS (2000)
4. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: ACM SOSP (2001)
5. Douceur, J.R., Wattenhofer, R.P.: Competitive hill-climbing strategies for replica placement in a distributed file system. In: DISC (2001)
6. Ghemawat, S., Gobioff, H., Leung, S.T.: The google file system. In: ACM SOSP (2003)
7. Haeberlen, A., Mislove, A., Druschel, P.: Glacier: highly durable, decentralized storage despite massive correlated failures. In: USENIX NSDI (2005)
8. Janson, S.: Poisson approximations for large deviations. Random Struct Algorithm **1**, 221–230 (1990)
9. Janson, S., Luczak, T., Rucinski, A.: An exponential bound for the probability of nonexistence of a specified subgraph in a random graph. In: Karo'nski, M., Jaworski, J., Ruci'nski, A. (eds.) Random Graphs'87. Wiley, New York (1990)
10. Karger, D., Lehman, E., Leighton, T., Levine, M., Lewin, D., Panigrahy, R.: Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In: ACM STOC (1997)
11. Kistler, J., Satyanarayanan, M.: Disconnected operation in the coda file system. ACM Trans. Comput. Syst. **10**(1), 3–25 (1992)
12. Knuth, D.E.: The Art of Computer Programming, vol 1. Addison Wesley, Reading (1997)
13. Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: OceanStore: an architecture for global-scale persistent storage. In: ACM ASPLOS (2000)
14. Pang, J., Gibbons, P.B., Kaminsky, M., Seshan, S., Yu, H.: Defragmenting DHT-based distributed file systems. In: Proceedings of International Conference on Distributed Computing Systems (2007)
15. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: ACM SIGCOMM (2001)
16. Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., Yu, F.: Data-centric storage in sensornets with GHT: a geographic hash table. Mobile Netw. Appl. **8**(4), 427–442 (2003)
17. van Renesse, R., Schneider, F.B.: Chain replication for supporting high throughput and availability. In: USENIX OSDI (2004)
18. Rowstron, A., Druschel, P.: Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: ACM Middleware (2001)
19. Santos, J., Muntz, R., Ribeiro-Neto, B.: Comparing random data allocation and data striping in multimedia serviers. In: ACM SIGMETRICS (2000)
20. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. In: ACM SIGCOMM (2001)
21. Szalay, A., Kunszt, P., Thakar, A., Gray, J., Slutz, D.: Designing and mining multi-terabyte astronomy archives: the sloan digital sky survey. In: ACM SIGMOD (2000)
22. TPC Benchmark. http://www.tpc.org/
23. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. **8**(3), 410–421 (1979)
24. Yu, H., Gibbons, P.B., Nath, S.: Availability of multi-object operations. In: USENIX NSDI (2006)
25. Yu, H., Vahdat, A.: Minimal replication cost for availability. In: ACM PODC (2002)