Repair Time in Distributed Storage Systems

Frédéric Giroire, Sandeep Kumar Gupta, Remigiusz Modrzejewski, Julian Monteiro, Stéphane Perennes

Repair Time in Distributed Storage Systems *

Frédéric Giroire[1], Sandeep K. Gupta[2], Remigiusz Modrzejewski[1], Julian Monteiro[3] and Stéphane Perennes[1]

[1]Project MASCOTTE, I3S (CNRS/Univ. of Nice)/INRIA, Sophia Antipolis, France
[2]IIT Delhi, New Delhi, India
[3]Department of Computer Science, IME, University of São Paulo, Brazil

Abstract. In this paper, we analyze a highly distributed backup storage system realized by means of nano datacenters (NaDa). NaDa have been recently proposed as a way to mitigate the growing energy, bandwidth and device costs of traditional data centers, following the popularity of cloud computing. These service provider-controlled peer-to-peer systems take advantage of resources already committed to always-on set top boxes, the fact they do not generate heat dissipation costs and their proximity to users.

In this kind of systems redundancy is introduced to preserve the data in case of peer failures or departures. To ensure long-term fault tolerance, the storage system must have a self-repairing service that continuously reconstructs the fragments of redundancy that are lost. The speed of this reconstruction process is crucial for the data survival. This speed is mainly determined by how much bandwidth, which is a critical resource of such systems, is available. In the literature, the reconstruction times are modeled as independent (e.g., poissonian, deterministic, or more generally following any distribution). In practice, however, numerous reconstructions start at the same time (when the system detects that a peer has failed). Consequently, they are correlated to each other because concurrent reconstructions do compete for the same bandwidth. This correlation negatively impacts the efficiency of the bandwidth utilization and henceforth the repair time.

We propose a new analytical framework that takes into account this correlation when estimating the repair time and the probability of data loss. Mainly, we introduce a queuing model in which reconstructions are served by peers at a rate that depends on the available bandwidth. We show that the load is unbalanced among peers (young peers inherently store less data than the old ones). This leads us to introduce a correcting factor on the repair rate of the system. The models and schemes proposed are validated by mathematical analysis, extensive set of simulations, and experimentation using the GRID5000 test-bed platform. This new model allows system designers to operate a more accurate choice of system parameters in function of their targeted data durability.

* The research leading to these results has received funding from the European Project FP7 EULER, ANR CEDRE, ANR AGAPE, Associated Team AlDyNet, project ECOS-Sud Chile and région PACA.

1 Introduction

Nano datacenters (NaDa) are highly distributed systems owned and controlled by the service provider. This alleviates the need of incentives and mitigates the risk of malicious users, but otherwise they face the same challenges as peer-to-peer systems. The set-top boxes realizing them are connected using consumer links, which can be relatively slow, unreliable and congested. The devices themselves, compared to servers in a traditional datacenter, are prone to failures and temporary disconnections, e.g. if the user cuts the power supply when not in home. When originally proposed in [1], they were assumed to be available no more than 85% of the time, with values as low as 7% possible.

In this paper we concentrate on application of NaDa, or any similar *peer-to-peer* system, for backup storage. In this application, users want to store massive amounts of data indefinitely, accessing them very rarely, i.e. only when original copies are lost. Due to risk of peer failures or departures, redundancy data is introduced to ensure long term data survival. To this end, most of the proposed storage systems use either the simple replication or the space efficient erasure codes [2], such as the Reed-Solomon or Regenerating Codes [3]. The redundancy needs to be maintained by a self-repair process. Its speed is crucial to determine the system reliability, as long repairs exponentially increase the probability of losing data. The limiting factor, in this setting, is the upload link capacity.

Imagine a scenario where the system is realized using home connections, out of which an average 128kbps are allocated to the backup application. Furthermore, each device is limited to 300GB, while average data stored is 100GB, redundancy is double, 100 devices take part in each repair and the algorithms are as described in the following sections. A naive back-of-envelope computation gives that the time needed to repair contents of a failed device is 17 hours ($= 100 \cdot 8 \cdot 10^6\text{kb}/(100 \cdot 128\text{kbps})$). This translates, by our model, to a probability of data loss per year (PDLPY) of 10^{-8}. But, taking into account all findings presented in this work, the actual time can reach 9 days. This gives a PDLPY of 0.2, many orders of magnitude more than the naive computation. Hence, it is important to have models that estimate accurately the repair time for limited bandwidth.

Our contribution

We propose a new analytical model that precisely estimates the repair time and the probability of losing data in distributed storage systems. This model takes into account the bandwidth constraints and inherent workload imbalance (young peers inherently store less data than the old ones, thus they contribute asymmetrically to the reconstruction process) effect on the efficiency. It allows system designers to obtain an accurate choice of system parameters to obtain a desired data durability.

We discuss how far the distribution of the reconstruction time given by the model is from the exponential, classically used in the literature. We exhibit the different possible shapes of this distribution in function of the system parameters.

This distribution impacts the durability of the system. We also show a somewhat counter-intuitive result that we can reduce the reconstruction time by using a less bandwidth efficient Regenerating Code. This is due to a degree of freedom given by erasure codes to choose which peers participate in the repair process.

To the best of our knowledge, this is the first detailed model proposed to estimate the distribution of the reconstruction time under limited bandwidth constraints. We validate our model by an extensive set of simulations and by test-bed experimentation using the Grid'5000 platform, see [4] for its description.

Related Work

Several works related to highly distributed storage systems have been done, and a large number of systems have been proposed [5–8], but few theoretical studies exist. In [9–11] the authors use a Markov chain model to derive the lifetime of the system. In these works, the reconstruction times are independent for each fragment. They follow an exponential or geometric distribution, which is a tunable parameter of the models. However, in practice, a large number of repairs start at the same time when a disk is lost, corresponding to tens or hundreds of GBs of data. Hence, the reconstructions are not independent of each other. Furthermore, in these models, only the average analysis are studied and the impact of congestion is not taken into account.

Dandoush et al. in [12] perform a simulation study of the download and the repairing process. They use the NS2 simulator to measure the distribution of the repair time. They state that a hypo-exponential distribution is a good fit for the block reconstruction time. However, again, concurrent reconstructions are not considered. Picconi et al. in [13] study the durability of storage systems. Using simulations they characterize a function to express the repair rate of systems based on replication. However, they do not study the distribution of the reconstruction time and the case of erasure coding. Venkatesan et al. in [14] study placement strategies for replicated data, deriving a simple approximation for mean time to data loss by studying the expected behaviour of most damaged data block. The closest to our work is [15] by Ford et al., where authors study reliability of distributed storage in Google, what constitutes a datacenter setting. However, they do not look into load imbalance, their model tracks only one *representative* data fragment and is not concerned by competition for bandwidth.

Organization

The remainder of this paper is organized as follows: in the next section we give some details about the studied system, then in Section 3 we discuss the impact of load imbalance. The queuing model is presented in the Section 4, followed by its mathematical analysis. The estimations are then validated via an extensive set of simulations in Section 5. Lastly, in Section 6, we compare the results of the simulations to the ones obtained by experimentation.

2 System Description

This section outlines the mechanisms of the studied system and our modelling assumptions.

Storage. In this work we assume usage of the Regenerating Codes, as described in [3], due to their high storage and bandwidth efficiency. More discussion of them follows later in this section. All data stored in the system is divided into *blocks* of uniform size. Each block is further subdivided into s *fragments* of size L_f, with r additional fragments of redundancy. All these $n = s + r$ fragments are distributed among random devices. We assume that in practice this distribution is performed with a Distributed Hash Table overlay like Pastry [16]. This, due to practical reasons, divides devices into subsets called *neighbourhoods* or *leaf sets*.

Our model does not assume ownership of data. The device originally introducing a block into the system is not responsible for its storage or maintenance. We simply deal with a total number of B blocks of data, which results in $F = n \cdot B$ fragments stored in N cooperating devices. As a measure of fairness, or *load balancing*, each device can store up to the same amount of data equal to C fragments. Note that C can not be less than average number of fragments per device $\bar{D} = {}^{F}/_{N}$.

In the following we treat a device and its disk as synonyms.

Bandwidth. Devices of NaDa are connected using consumer connections. These, in practice, tend to be asymmetric with relatively low upload rates. Furthermore, as the backup application occasionally uploads at maximum throughput for prolonged times, while the consumer expects the application to not interfere with his network usage, we assume it is allocated only a fraction of the actual link capacity. Each device has a maximum upload and download bandwidth, respectively BW_{up} and BW_{down}. We set $BW_{down} = 10BW_{up}$ (in real offerings, this value is often between 4 and 20). The bottleneck of the system is considered to be the access links (e.g. between a DSLAM and an ADSL modem) and not the network internal links.

Availability and failures. Mirroring requirements of practical systems, we assume devices to stay connected at least a few hours per day. Following the work by Dimakis [3] on network coding, we use values of availability and failure rate from the PlanetLab [17] and Microsoft PCs traces [6]. To distinguish transient unavailability, which for some consumers is expected on a daily basis, from permanent failures, a timeout is introduced. Hence, a device is considered as failed if it leaves the network for more than 24 hours. In that case, all data stored by it is assumed to be lost.

The Mean Time To Failure (MTTF) in the Microsoft PCs and the PlanetLab scenarios are respectively 30 and 60 days. The device failures are then considered as independent, like in [9], and Poissonian with mean value given by the traces explained above. We consider a discrete time in the following and the probability to fail at any given time step is denoted as $\alpha = {}^{1}/_{MTTF}$.

Repair process. When a failure is detected, neighbours of the failed device start a reconstruction process, to maintain desired redundancy level. For each fragment stored at the failed disk, a random device from the neighbourhood is chosen to be the *reconstructor*. It is responsible for downloading necessary data from remaining fragments of the block, reconstructing and storing the fragment.

Redundancy schemes. Minimum Bandwidth Regenerating Codes, assumed in this paper, are very efficient due to not reconstructing the exact same lost fragment, but creating a new one instead, in the spirit of Network Coding. The reconstructor downloads, combines and stores small *subfragments* from d devices having other fragments of the repaired block. We call d the repair degree, $s \leq d \leq n$. Construction of the code requires some additional redundancy for each fragment. In other words L_r, the total amount of data transferred for a repair of a fragment, is greater than L_f by some overhead factor. This factor, the efficiency of the code, has been given for MBR in [3] as:

$$\delta_{MBR}(d) = \frac{2d}{2d - s + 1}.$$

The most bandwidth efficient case is clearly when $d = n - 1$. However, as we will show in following sections, it may be beneficial to set it to a lower value to give the reconstruction an additional degree of freedom.

The model presented in this work was also successfully applied to other redundancy schemes. Minimum Storage Regenerating Codes, also given in [3], are more space efficient at the cost of additional transfer overhead. Reed-Solomon codes, more popular in practice, are reconstructed by recreating the input data and then coding again the lost fragment. In both cases the only difference for the model are different values of L_r. In practical systems, it may be interesting for RS-based systems to reconstruct at one device, but store the new fragment on some other one. This is especially true for *saddle*-based systems, where we wait until a few fragments of a block are lost, to repair them all at once. The model gives good results also for these more complicated cases. We omit them due to lack of space, and because this only brings slightly longer analysis with little new insight.

3 Preliminary: Impact of Disk Asymmetry

Disk occupancy follows a truncated geometric distribution. Denote by x the average disk size divided by the average amount of data stored per device. Let ρ be the *factor of efficiency*: the average bandwidth actually used during a repair process divided by the total bandwidth available to all devices taking part in it. It has been observed in simulations that $\rho \approx 1/x$. This has been further confirmed both by experiments and by theoretical analysis, which has to be omitted here due to lack of space, but can be found in the research report [18]. What follows is a brief intuition of the analysis.

First, notice that a new device joins the system empty and is gradually filled throughout its lifetime. Thus, we have disks with heterogeneous occupancy. For

$x < 2$ almost all disks in the system are full. Most blocks have a fragment on a full disk. Even for $x = 3$, when only 6% of disks are full, probability of a block having a fragment on a full disk is 92%. Thus the average repair time depends on the time the full disks take, which is in turn x times the average disks take. This shows that load balancing is crucial and for practical systems x should be kept below two.

4 The Queuing Model

We introduce here a *Markov Chain Model* that allows us to estimate the reconstruction time under bandwidth constraints. The model makes an important assumption: the limiting resource is always the upload bandwidth. It is reasonable because download and upload bandwidths are strongly asymmetric in systems built on consumer connections. Using this assumption, we model the storage system with a queue tracking the upload load of the global system.

4.1 Model Definition

We model the storage system with a Markovian queuing model storing the upload needs of the global system. The model has one server, Poissonian batch arrivals and deterministic time service ($M^\beta/D/1$, where β is the batch size function). We use a discrete time model, all values are accounted in time steps. The devices in charge of repairs process blocks in a FIFO order.

Chain States. The state of the chain at a time t is the current number of fragments in reconstruction, denoted by $Q(t)$.

Transitions. At each time step, the system reconstructs blocks as fast as its bandwidth allows. The upload bandwidth of the system, $BW_{up}N$, is the limiting resource. Then, the *service* provided by the server is

$$\mu = \rho \frac{BW_{up}N}{L_r},$$

which corresponds to the number of fragments that can be reconstructed at each time step. The factor ρ is the bandwidth efficiency as calculated in the previous section, and L_r is the number of bytes transferred to repair one fragment. Hence, the number of fragments repaired during a time step t is $\mu(t) = \min(\mu, Q(t))$.

The *arrival process* of the model corresponds to device failures. When a failure occurs, all the fragments stored in the failed device are lost. Hence, a large number of block repairs start at the same time. We model this with batch inputs (sometimes also called *bulk arrival* in the literature). The size of an arrival is given by the number of fragments that were stored on the disk. As stated in Section 3, it follows a truncated geometric distribution.

We define β as a random variable taking values $\beta \in \{0, v, 2v, \ldots, T_{max}v\}$, which represents the number of fragments inside a failed disk Recall that v is the speed at which empty disks get filled, and that $T_{max} = C/v$ is the expected

time to fill a disk. Further on, β/v is the expected time to have a disk with β fragments.

The arrival process of the model is Poissonian. A batch arrives during a time step with probability f, with $f \approx \alpha N$. For the simplicity of the exposition, we consider here that only one failure can happen during a time step (note that to ensure this, it is sufficient to choose a small enough time step). Formally, the transitions of the chain are, for $\forall i \geq \mu$,

$$\begin{array}{ll} Q_i \to Q_{i-\mu} & \text{with prob. } 1-f \\ Q_i \to Q_{i-\mu+\beta}, \forall \beta & \text{with prob. } f(1-\alpha)^{\frac{\beta}{v}-1}\alpha \\ Q_i \to Q_{i-\mu+C} & \text{with prob. } f(1-(1-\alpha)^{T_{\max}}) \end{array}$$

When $0 \leq i < \mu$, the i blocks in the queue at the beginning of the time step are reconstructed at the end. Hence, we have transitions without the term $i - \mu$:

$$\begin{array}{ll} Q_i \to Q_0 & \text{with prob. } 1-f \\ Q_i \to Q_\beta, \forall \beta & \text{with prob. } f(1-\alpha)^{\frac{\beta}{v}-1}\alpha \\ Q_i \to Q_C & \text{with prob. } f(1-(1-\alpha)^{T_{\max}}) \end{array}$$

Figure 1 presents the transitions for a state i.

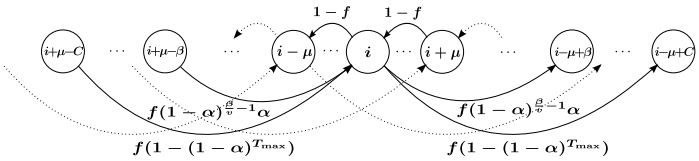


Fig. 1: Transition around state i of the Markovian queuing model.

4.2 Analysis

Expressions to estimate the values of the bandwidth usage, the distribution of block reconstruction time and the probability of data loss can be derived from the stationary distribution of the Markovian model. We omit here the analysis due to lack of space, but it can be found in the research report [18].

5 Results

To validate our model, we compare its results with the ones produced by simulations, and test-bed experimentation. We use a custom cycle-based simulator. The simulator models the evolution of the states of blocks during time (number of available fragments and where they are stored) and the reconstructions being processed. When a disk failure occurs, the simulator updates the state of all

blocks that have lost a fragment, and starts the reconstruction if necessary. The bandwidth is implemented as a queue for each device, respecting both BW_{up} and BW_{down} constraints. The reconstructions are processed in FIFO order.

We study the distribution of the reconstruction time and compare it with the exponential distribution, which is often used in the literature. We then discuss the cause of the data losses. Finally, we present an important practical implementation point: when choosing the parameters of the Regenerating Code, it is important to *give to the device in charge of the repair a choice between several devices to retrieve the data.*

5.1 Distribution of Reconstruction Time

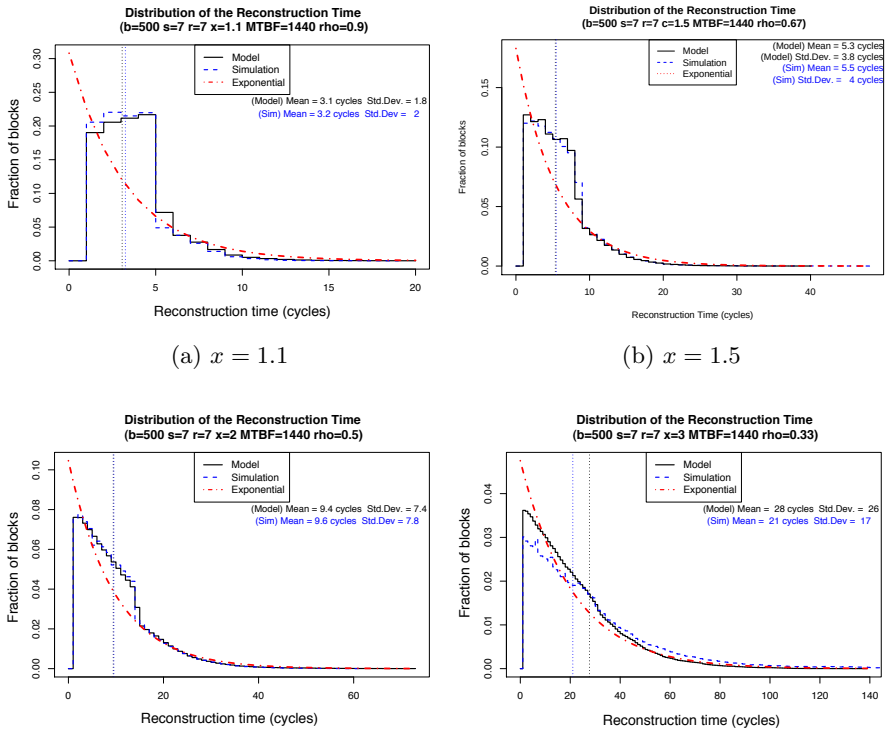
Figure 2 shows the distribution of the reconstruction time and the impact of device asymmetry on the reconstruction time for the following scenario: N = 100, s = 7, r = 7, L_r=2 MB, B = 50000, MTTF = 60 days, BW_{up} = 128 kpbs. All parameters are kept constant, except the disk size factor x (recall that x is the ratio of the maximum capacity over the average amount of data per device).

First, we see that the model (dark solid line) closely matches the simulations (blue dashed line). For example, when $x = 1.1$ (top plot), the curves are almost merged. Their shape is explained in the next paragraph. The average reconstruction times are 3.1 time steps for the model vs 3.2 for the simulation. We see that there is a small gap when $x = 3$. As a matter of fact, as we saw in Section 3, model's assumptions do not hold in this case: only 92% of the blocks have a fragment on a full disk. Recall that, for the sake of load balancing, higher values of x are impractical.

Second, we confirm the strong impact of the disk capacity. We see that for the four considered values of x, the shape of distributions of the reconstruction times are very different. When the disk capacity is close to the average number of fragments stored per disk (values of x close to 1), almost all disks store the same number of fragments (83% of full disks). Hence, each time there is a disk failure in the system, the reconstruction times span between 1 and C/μ, explaining the rectangle shape. The tail is explained by multiple failures happening when the queue is not empty. When x is larger, disks also are larger, explaining that it takes a longer time to reconstruct when there is a disk failure (the average reconstruction time raises from 3.2 to 9.6 and 21 when x goes from 1.1 to 2 and 3). As the number of fragments per disk follows a truncated geometric distribution, we see the rectangle shape is replaced by a trapezoidal shape, explained by the large range of disk fillings for big values of x.

Third, we compare the distributions obtained with the exponential distribution that is classically used in the literature. We see that the distributions are far from the exponential when $1.1 \leq x \leq 2$, but get closer for $x = 3$. Hence, as we will confirm in the next section, the exponential distribution is only a good choice for some given sets of parameters. Note that the tails of the distributions are close to exponential.

Figure 3 presents the distribution of a distributed storage system experiencing three different rates of failures: MTTF of 90, 180 and 360 days. We clearly



(a) $x = 1.1$ (b) $x = 1.5$

(c) $x = 2.0$ (d) $x = 3.0$

Fig. 2: Distribution of reconstruction time for different disk capacities x of 1.1, 2, and 3 times the average amount. The average reconstruction times of simulations are respectively 3.2, 9.6, and 21 hours (Note that some axis scales are different).

see the evolution of the shape of the distribution due to the larger probability to experience failures when the device queues are still loaded. The average reconstruction time increases from 5 hours when the MTTF is 360 days to 12 hours when the MTTF is 90 days.

We ran simulations for different sets of parameters. We present in Table 1 a small subset of these experiments.

5.2 Where the Dead Come From?

In this section, we discuss in which circumstances the system has more probability to lose some data. First a preliminary remark: backup systems are conceived to experience basically no data loss. Thus, for realistic sets of parameters, it would be necessary to simulate the system for a prohibitive time to see any data loss. We hence present here results for scenarios where the redundancy of the data is lowered ($r = 3$ and $r = 5$).

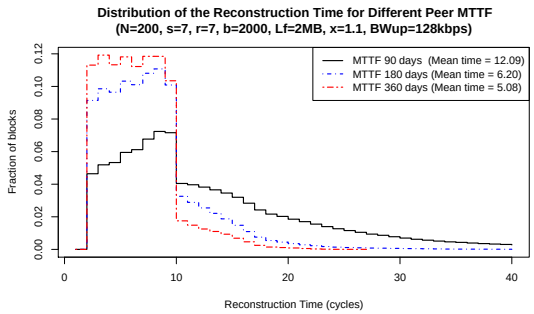


Fig. 3: Distribution of reconstruction time for different MTTF. Different shapes for different values.

Table 1: Reconstruction time T (in hours) for different system parameters

(a) Disk capacity c.

c	1.1	1.5	2.0	3.0
T_{sim}	3.26	5.50	9.63	21.12
T_{model}	3.06	5.34	9.41	21

(b) Peer Lifetime (MTTF).

$MTTF$	60	120	180	365
T_{sim}	3.26	2.90	2.75	2.65
T_{model}	2.68	2.60	2.49	2.46

(c) Peer Upload Bandwidth (kbps).

$upBW$	64	128	256	512
T_{sim}	8.9	3.30	1.70	1.07
T_{model}	8.3	3.10	1.61	1.03

In Figure 4 we plot the cumulative number of dead blocks that the system experiences for different reconstruction times. We give this fraction in function of the time the block spent in the system before dying. For the queuing model, we derive the expected number of blocks that died at time T from the distribution of the reconstruction time. A block dies at time T if its reconstruction process lasts a time $\theta \geq T$ and that it loses r fragments during time T with at least one exactly at time T. This can be expressed as

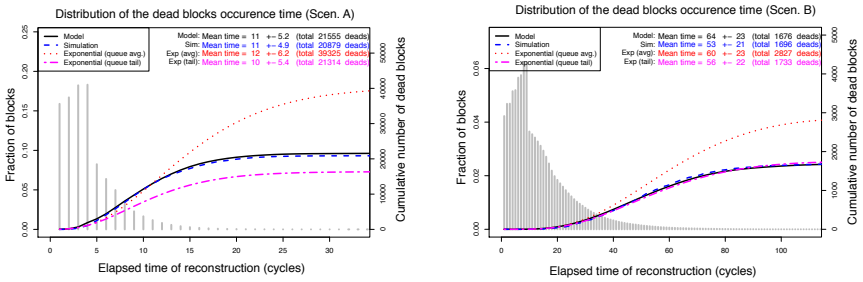
$$N[\text{die at time } T] = \Pr[\text{die at time } T] \sum_{\theta \geq T} NP[W = \theta]$$

with

$$\Pr[\text{die at time } T] = \binom{s+r-1}{r-1}(1-(1-\alpha)^T)^r((1-\alpha)^T)^{s-1} - \binom{s+r-1}{r-1}(1-(1-\alpha)^{T-1})^r((1-\alpha)^T)^{s-1}.$$

We give the distribution of the reconstruction times as a reference (vertical lines). The model (black solid line) and the simulation results (blue dashed line) are compared for two scenarios with different number of blocks: there is twice more data in Scenario B.

The first observation is that the queuing models predict well the number of dead experienced in the simulation, for example, in the scenario A the values are 21,555 versus 20,879. The results for an exponential reconstruction time with



(a) Scenario A (b) Scenario B

Fig. 4: Distribution of dead blocks reconstruction time for two different scenarios. Scenario A: $N = 200, s = 8, r = 3, b = 1000, MTTF = 60$ days. Scenario B: $N = 200, s = 8, r = 5, b = 2000, MTTF = 90$ days.

the same mean value are also plotted (queue avg.). We see that this model is not close to the simulation for both scenarios (almost the double for Scenario A). We also test a second exponential model (queue tail): we choose it so that its tail is as close as possible to the tail than the queuing model (see Figures 4b). We see that it gives a perfect estimation of the dead for Scenario B, but not for Scenario A.

In fact, two different phenomena appear in these two scenarios. In Scenario B (higher redundancy), the *lost blocks are mainly coming from long reconstructions*, from 41 to 87 cycles (tail of the gray histogram). Hence, a good exponential model can be found by fitting the parameters to the tail of the queuing model. On the contrary, in Scenario A (lower redundancy), the *data loss comes from the majority of short reconstructions*, from 5.8 to 16.2 cycles (the right side of the rectangular shape). Hence, in Scenario A, having a good estimate of the tail of the distribution is not at all sufficient to be able to predict the failure rate of the system. It is necessary to have a good model of the complete distribution!

5.3 Discussion of Parameters of Regenerating Codes

As presented in Section 2, when the redundancy is added using regenerating codes, $n = s + r$ devices store a fragment of the block, while just s are enough to retrieve the block. When a fragment is lost d devices, where $s \leq d \leq n - 1$, cooperate to restore it. The larger d is, the smaller is the bandwidth needed for the repair. Figures 5 and 6 show the reconstruction time for different values of the degree d. We observe an interesting phenomena: at the opposite of the common intuition, the average reconstruction time decreases when the degree decreases: 10 cycles for $d = 13$, and only 6 cycles for $d = 12$. The bandwidth usage increases though (because the δ_{MBR} is higher when d is smaller). The explanation is that the decrease of the degree *introduces a degree of freedom* in the choice of devices that send a sub-fragment to the device that will store the repaired fragment. Hence, the system is able to decrease the load of the more loaded disks and to *balance more evenly the load between devices*.

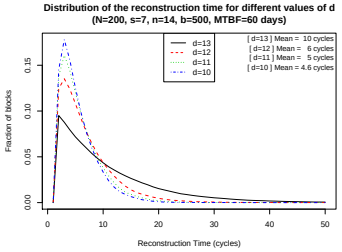


Fig. 5: Distribution of reconstruction time for different values of degree d.

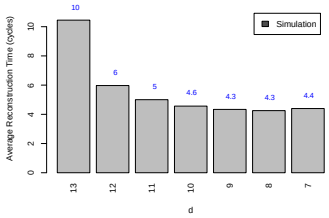


Fig. 6: Average Reconstruction Time for different values of degree d. Smaller d implies more data transfers, but may mean smaller reconstruction times!

In fact, we can estimate for which degree of freedom, the reconstruction time is minimum. It happens when the load of the full disks is the same as the load of the other disks. We define $\delta = n - 1 - d$ the allowed degree of freedom for the choice of which devices upload the sub-fragments. The full disks store a proportion φx of the fragments of the system, with φ being the fraction of full disks. We simply look at how much work we *must* do on the full disks. The probability to have i fragments (among the $n-1$ fragments) on full disks is $\binom{n-1}{i}(\varphi x)^i(1-\varphi x)^{n-1-i}$. Those blocks send $i - \delta$ fragments (whenever $i \geq \delta$). So the load of the full disks is:

$$\sum_{i=\delta}^{n-1}(i-\delta)\binom{n-1}{i}(\varphi x)^i(1-\varphi x)^{n-1-i}.$$

When the load of the full disks becomes equal to the load of the other disks ($\sum_{i=\delta}^{n-1}(d-i+\delta)\binom{n-1}{i}(\varphi x)^i(1-\varphi x)^{n-1-i}$), it is no more useful to decrease d. We see that the average reconstruction time increases when d is too small, as the increased usage of bandwidth is no more compensated by a better balance of the load.

We presented here a cut argument for only two classes of devices (full disks and non full disks). This argument can be generalized to any number of device classes.

Note that this phenomena exists for other codes like Reed Solomon where the device in charge of the reconstruction has to retrieve s fragments among the $s + r - 1$ remaining fragments.

6 Experimentation

Aiming at validating the simulation and the model results, we performed a batch of real experimentation using the Grid'5000 platform [4]. It is an experimental platform for the study of large scale distributed systems. It provides over 5000 computing cores in multiple sites in France, Luxembourg and Brazil. We

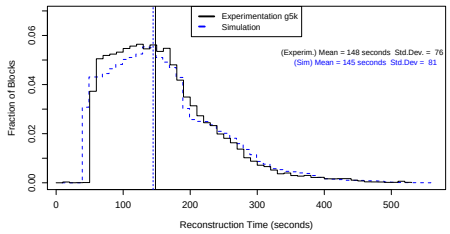


Fig. 7: Distribution of reconstruction time in an 64 nodes during 4 hours experiment compared to simulation.

used a prototype of storage system implemented by a private company (Ubistorage [19]).

Our goal is to validate the main behavior of the reconstruction time in a real environment with shared and constrained bandwidth, and measure how close they are to our results.

6.1 Storage System Description

In few words, the system is made of a storage layer (upper layer) built on top of the DHT layer (lower layer) running Pastry [16]. The lower layer is in charge of managing the logical topology: finding devices, routing, alerting of device arrivals or departures. The upper layer is in charge of storing and monitoring the data.

Storing the data. The system uses Reed-Solomon erasure codes [20] to introduce redundancy. Each data block has a device responsible of monitoring it. This device keeps a list of the devices storing a fragment of the block. The fragments of the blocks are stored locally on the Pastry leafset of the device in charge [21].

Monitoring the system. The storage system uses the information given by the lower level to discover device failures. In Pastry, a device checks periodically if the members of its leafset are still up and running. When the upper layer receives a message that a device left, the device in charge updates its block status.

Monitored metrics. The application monitors and keep statistics on the amount of data stored on its disks, the number of performed reconstructions along with their duration, the number of dead blocks that cannot be reconstructed. The upload and download bandwidth of devices can be adjusted.

6.2 Results

There exist a lot of different storage systems with different parameters and different reconstruction processes. The goal of the paper is not to precisely tune a model to a specific one, but to provide a general analytical framework to be able to predict any storage system behavior. Hence, we are more interested here by the global behavior of the metrics than by their absolute values.

Studied Scenario. By using simulations we can easily evaluate several years of a system, however it is not the case for experimentation. Time available for a

simple experiment is constrained to a few hours. Hence, we define an *acceleration factor*, as the ratio between experiment duration and the time of real system we want to imitate. Our goal is to check the bandwidth congestion in a real environment. Thus, we decided to shrink the disk size (e.g., from 10 GB to 100 MB, a reduction of 100×), inducing a much smaller time to repair a failed disk. Then, the device failure rate is increased (from months to a few hours) to keep the ratio between disk failures and repair time proportional. The bandwidth limit value, however, is kept close to the one of a "real" system. The idea is to avoid inducing strange behaviors due to very small packets being transmitted in the network.

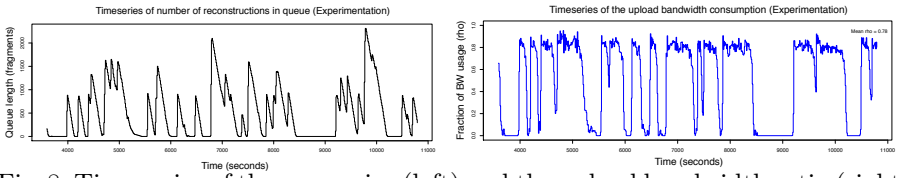


Fig. 8: Time series of the queue size (left) and the upload bandwidth ratio (right).

Figure 7 presents the distribution of the reconstruction times for two different experimentation involving 64 nodes on 2 different sites of Grid'5000. The amount of data per node is 100 MB (disk capacity 120MB), the upload bandwidth 128 KBps, $s = 4$, $r = 4$, $L_F = 128$ KB. We confirm that the simulator gives results very close to the one obtained by experimentation. The average value of reconstruction time differs by a few seconds.

Moreover, to have an intuition of the system dynamics over time, in Figure 8 we present a time series of the number of blocks in the queues (top plot) and the total upload bandwidth consumption (bottom plot). We note that the rate of reconstructions (the descending lines on the top plot) follows an almost linear shape. Comforting our claim that a deterministic processing time of blocks could be assumed. In these experiments the disk size factor is $x = 1.2$, which gives a theoretical efficiency of 0.83. We can observe that in practice, the factor of bandwidth utilization, ρ, is very close to this value (value of $\rho = 0.78$ in the bottom plot).

7 Conclusions and take-aways

In this paper, we propose and analyze a new Markovian analytical model to model the repair process of distributed storage systems. This model takes into account competition for bandwidth between correlated failures. We bring to light the impact of device heterogeneity on the system efficiency. The model is validated by simulation and by real experiments on the Grid'5000 platform.

We show that load balancing in storage is crucial for reconstruction time. We introduce a simple linear factor of efficiency, where throughput of the system is divided by the ratio of maximum allowed disk size to the average occupancy.

We show that the exponential distribution, classically taken to model the reconstruction time, is valid for certain sets of parameters, but introduction of load balancing causes different shapes to appear. We show that it is not enough to be able to estimate the tail of the repair time distribution to obtain a good estimate of the data loss rate.

The results provided are for systems using Regenerating Codes that are the best codes known for bandwidth efficiency, but the model is general and can be adapted to other codes. We exhibit an interesting phenomena to keep in mind when choosing the code parameter: it is useful to keep a degree of freedom on the choice of the users participating in the repair process so that loaded or deficient users do not slow down the repair process, even if it means less efficient codes.

References

1. V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 37–48.
2. H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Revised papers from the 1st Intl. Workshop on Peer-to-Peer Systems (IPTPS)*, vol. LNCS 2429, 2002, pp. 328–337.
3. A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. of IEEE INFOCOM*, May 2007, pp. 2000–2008.
4. Grid5000Platform, "https://www.grid5000.fr/."
5. B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," in *Proc. of USENIX NSDI*, 2006, pp. 45–58.
6. W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs," *ACM SIGMETRICS Perf. Eval. Review*, vol. 28, pp. 34–43, 2000.
7. R. Bhagwan, K. Tati, Y. chung Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," in *Proc. of the USENIX NSDI*, 2004, pp. 337–350.
8. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells *et al.*, "OceanStore: an architecture for global-scale persistent storage," *ACM SIGARCH Computer Architecture News*, vol. 28, no. 5, pp. 190–201, 2000.
9. S. Ramabhadran and J. Pasquale, "Analysis of long-running replicated systems," in *Proc. of IEEE INFOCOM*, Spain, 2006, pp. 1–9.
10. S. Alouf, A. Dandoush, and P. Nain, "Performance analysis of peer-to-peer storage systems," *Proceedings of the 20th International Teletraffic Congress (ITC)*, vol. LNCS 4516, pp. 642–653, 2007.
11. A. Datta and K. Aberer, "Internet-scale storage systems under churn – a study of the steady-state using markov models," in *Procedings of the IEEE Intl. Conf. on Peer-to-Peer Computing (P2P)*, 2006, pp. 133–144.
12. A. Dandoush, S. Alouf, and P. Nain, "Simulation analysis of download and recovery processes in P2P storage systems," in *Proc. of the Intl. Teletraffic Congress (ITC)*, France, 2009, pp. 1–8.

13. F. Picconi, B. Baynat, and P. Sens, "Predicting durability in dhts using markov chains," in *Proceedings of the 2nd Intl. Conference on Digital Information Management (ICDIM)*, vol. 2, Oct. 2007, pp. 532–538.
14. V. Venkatesan, I. Iliadis, and R. Haas, "Reliability of data storage systems under network rebuild bandwidth constraints," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*. IEEE, 2012, pp. 189–197.
15. D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association, 2010, pp. 1–7.
16. A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware)*, vol. LNCS 2218, 2001, pp. 329–350.
17. "Planetlab." [Online]. Available: http://www.planet-lab.org/
18. F. Giroire, S. Gupta, R. Modrzejewski, J. Monteiro, and S. Perennes, "Analysis of the repair time in distributed storage systems," INRIA, Tech. Rep. 7538, Feb. 2011.
19. "Ubistorage," http://www.ubistorage.com/.
20. M. Luby, M. Mitzenmacher, M. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the 29th annual ACM symposium on Theory of computing*, 1997, pp. 150–159.
21. S. Legtchenko, S. Monnet, P. Sens, and G. Muller, "Churn-resilient replication strategy for peer-to-peer distributed hash-tables," in *Proceedings of SSS*, vol. LNCS 5873, 2009, pp. 485–499.