Analysis of the Repair Time in Distributed Storage Systems

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

Analysis of the Repair Time in Distributed Storage Systems

Frédéric Giroire — Sandeep Kumar Gupta — Remigiusz Modrzejewski — Julian Monteiro — Stéphane Pérennes

N° 7538

February 2011

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7538--FR+ENG

Analysis of the Repair Time in Distributed Storage Systems

Frédéric Giroire[*] , Sandeep Kumar Gupta[†] , Remigiusz Modrzejewski[*] , Julian Monteiro[*] , Stéphane Pérennes[*]

Thème COM — Systèmes communicants
Projets Mascotte

Rapport de recherche n° 7538 — February 2011 — 28 pages

Abstract: Distributed or peer-to-peer storage systems introduce redundancy to preserve the data in case of peer failures or departures. To ensure long-term fault tolerance, the storage system must have a self-repair service that continuously reconstructs lost fragments of redundancy. The speed of this reconstruction process is crucial for the data survival. This speed is mainly determined by available bandwidth, a critical resource of such systems. We propose a new analytical framework that takes into account the correlation of concurrent repairs when estimating the repair time and the probability of data loss. Mainly, we introduce queuing models in which reconstructions are served by peers at a rate that depends on the available bandwidth. The models and schemes proposed are validated by mathematical analysis, extensive set of simulations, and experimentation using the GRID'5000 test-bed platform.

Key-words: P2P storage systems, data lifetime, queuing model, regenerating codes performance evaluation

This work has been partially supported by ANR SPREADS and ANR DIMAGREEN.

[*] MASCOTTE, I3S(CNRS,UNSA), INRIA Sophia Antipolis, France – Email: `firstname.lastname@sophia.inria.fr`
[†] IIT Delhi, New Delhi, India

Analyse du temps de reconstruction de données dans le système de stockage pair-a-pair

Résumé : Dans les systèmes de stockage distribués ou pair à pair, redondance des données doit être rajoutée afin de garantir l'intégrité du contenu en cas de panne ou de départ d'un pair. Afin d'assurer au système une résistance aux pannes sur le long terme, un processus interne doit continuellement reconstruire les fragments de redondance perdus. La vitesse de reconstruction de ces fragments des données est cruciale pour garantir l'intégrité du contenu. La bande passante disponible au sein du système déterminant en grande partie la vitesse de reconstruction.

Une nouvelle méthode d'analyse est proposée prenant en compte la corrélation entre réparation simultanées lors de l'estimation du temps total de réparation et la probabilité de perte de données. Notre contribution principale est une modélisation basée sur le modèle des files d'attente dans laquelle les reconstructions sont effectuées par les pairs à un débit dépendant de la bande passante disponible. Ce modèle montre que pour la plupart des systèmes actuels, un temps de reconstruction exponentiel est inadéquate. Les modèles et schémas proposés ont été validés par analyse mathématique ainsi que par un grand nombre de simulations et expérimentations en utilisant la plateforme GRID'5000.

Mots-clés : Système d'stockage pair-à-pair, durée de vie des données, modèles de files d'attente, regenerating codes, évaluation de performances,

1 Introduction

Distributed storage systems are foreseen as a cheap and scalable way to backup data. These systems exploit the already available resources of users in terms of bandwidth, disk space, and processing, dismissing the need of building costly dedicated infrastructure. The highly distributed nature of such systems raises questions about durability, availability, security, and routing of the data.

These systems are subject to peer failures or departures. Thus, redundancy data is introduced to ensure long term data survival. To introduce redundancy, most of the proposed storage systems use either the simple replication or the space efficient erasure codes [21], such as the Reed-Solomon coding. When using these codes, a file (or more generally, a *block of data*), is divided into s fragments, from which the coding scheme generates r fragments of redundancy. Then, the system sends the $n = s+r$ fragments of the data block into different peers of the network. The data block can be recovered if any s among the $s+r$ fragments are present in the system.

This redundancy needs to be maintained during the system lifetime by a self-repairing process. The duration of this repairing process is crucial to determine the system reliability. That is, repairs that last long increase the probability of losing data exponentially. The speed of this repair process is mainly determined by how much bandwidth is available. This bandwidth is usually limited by the peers' upload link capacity, which is arguably one of the most scarce resource of such systems (i.e., when compared to the processing capacity or the storage space).

Imagine a scenario where users are connected using a typical home connection via an Asymmetric Digital Subscriber Line (ADSL) with upload capacity of 1Mbps. We expect that only part of this bandwidth is allocated to the storage system, let us say 128kbps. The average amount of data per peer is 100 gigabytes. When a peer fails, if 100 peers participate to the repairing process at an optimal rate of 128kbps, then the system would need theoretically 17 hours to recovery the contents of the failed disk. By our models, if we consider that peers have an expected lifetime of 1 year, this repair time lasts around 22 hours, which gives a probability of data loss per year (PDLPY) of 10^{-8} (we set $s = 7$ and $r = 7$).

However, in this paper we show that, due to several factors, in practice the repair time is in fact much greater than this optimal time. For instance, the *imbalance* on the amount of data per peer negatively impacts the efficiency of the bandwidth utilization. Continuing with the same example, for the same average amount of data per disk of 100 gigabytes, if the system have disks with heterogeneous capacity (limited to 3 times the average amount), then the repair time of a disk reaches 9 days. Which gives a PDLPY of 0.2. Many orders of magnitude more than the previous case! Hence, the importance of having models that estimate correctly the repairing time for limited bandwidth scenarios.

Our contribution

We propose a *new analytical model that precisely estimates the repair time and the probability of losing data* of storage systems based on erasure codes. This model takes into account the bandwidth constraints of peers when processing reconstructions.

We show that *is crucial to take into account the peer imbalance* to estimate the system efficiency. Indeed, we show that the traffic load is not well distributed among peers: young peers inherently store less data than the old ones, thus they contribute asymmetrically to the reconstruction process. Hence, we propose to introduce biases in the protocol to correct this imbalance, and show that it improves the efficiency of the system.

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 address *scheduling and control issues.* Indeed, each peer is involved in many reconstructions, thus we need to schedule their execution. We compare several policies, and *propose a simple greedy-like policy* that allows the system to reduce the reconstruction time.

We show a somewhat counterintuitive result that we *can reduce the reconstruction time by using a less bandwidth efficient* Regenerating Code. This is due to the *degree of freedom* given by erasure codes to choose which peers participate to 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.*

Related Work

Several works related to P2P storage systems have been done, and a large number of systems have been proposed [4, 3, 2, 12]. But few theoretical studies exist. Most studies are Markov chain models that assume in fact a poissonian reconstruction process (i.e., with independent reconstruction time). Furthermore, in these models, only the average analysis are studied and the impact of congestion is not taken into account.

In [17, 1, 7] the authors use a Markov chain model to derive the lifetime of the system. In these works, the reconstruction time follows 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.

Dandoush et al. in [6] 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, they assume that reconstruction events are independent, which means that they do not take into account their correlation when a disk fails.

Similarly to the present paper, other works also study the impacts of competition for the bandwidth. Ramabhadran et Pasquale in [18] address resource allocation problems in replicated systems. They study different schemes to optimize the average file availability.

Picconi et al. in [15] 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 more complex case of erasure coding.

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 queueing 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 Description

In this section we give details about the studied storage system and the modeling assumptions.

Peer bandwidth. In a peer-to-peer system, peers are typically connected to the network via an ADSL (Asymmetric Digital Subscriber Line) link. Thus, we model here asymmetric capacities as they are the configurations most often encountered in practice: each peer has a maximum upload and download bandwidth, resp. BW_{up} and BW_{down}; we set $BW_{down} = 10BW_{up}$ (in real systems, this value is often between 4 and 10). The bottleneck of the system is considered to be the peer links and not the network internal links.

Peer availability and peer failures. Peers can be highly available (as servers that are kept in a controlled environment), or conversely, be barely available with a low presence interval. Since we consider the case of backup storage systems, the peers are expected to stay connected at least few hours per day.

Following the work by Dimakis [8] on network coding, we use similar values of availability and failure rate from the PlanetLab [16] and Microsoft PCs traces [3]. To distinguish from transient failures, a peer is considered as failed if it leaves the network for more than a timeout, which was set to 24 hours. In that case, all data is considered lost. The Mean Time To Failure (MTTF) in the Microsoft PCs and the PlanetLab scenarios are respectively 30 and 60 days. For given values of $s = 7$ and $r = 7$, we achieve a block availability of 5 nines in the Microsoft PCs scenario. The peer failures are then considered as independent 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 $\alpha = 1/MTTF$.

Redundancy Scheme and Repair. In this study we use Regenerating Codes (RC) [8] to introduce redundancy, as they are foreseen as the most efficient codes in terms of bandwidth usage. Similarly to the Reed-Solomon erasure codes, a data block is divided into s fragments, to which are added r fragments of redundancy. Then, the $n = s + r$ fragments are spread into n peers in such a way that the block can be regenerated by retrieving any s fragments. However, when employing the Regenerating Codes, the repairing process can be done by creating a new fragment to replace the missing one, instead of regenerating the whole block as required by Reed-Solomon codes. Hence, a lost fragment can be repaired efficiently by contacting d peers, with $s \leq d < n$ (d is called the repair degree of the block). Each one of the d peers needs to send a small sub-fragment to the *reconstructor* peer, which in turn will store the repaired fragment. This reconstructor peer which is in charge of the repair is chosen uniformly at random. To achieve this repair efficiency, these codes introduce an overhead on the fragment size (that is, how much the original fragment must be increased in size to achieve the regenerating code property). δ_{MBR} is the overhead factor of the *Minimum-Bandwidth Regenerating Codes* [8]. It is defined as follows:

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

The most efficient case is when d is the maximum, $d = n - 1$. Hereafter we note $L_r = \delta_{MBR}(n-1)L_f$, as the amount of information transferred to reconstruct one fragment when $d = n - 1$, where L_f is the original size of the fragment.

Nevertheless, the model presented in this work can be adapted to systems using different codes to introduces redundancy, e.g., Replication, Reed-Solomon, Hierarchical codes [9], or Hybrid coding. Basically, the main change would be to replace the bandwidth efficiency of RC by the bandwidth efficiency of the other code.

Monitoring the data and network size. We consider distributed systems of any size, e.g., thousands of peers. However, for practical reasons and maintainability, the fragments of blocks are often stored on small logical subset of peers that are self-structured. This subset is inherited from the DHT terminology of P2P architectures, where they are often called *leafset* or *neighborhood*. In the following examples, we consider sizes of neighborhood of 100 to 200 peers. Hereafter in this chapter, we use the terms *peer* and *disk* interchangeably.

Table 1 shows a summary of the notations used in this chapter.

3 Preliminary: Impact of Disk Asymmetry

In this section we start by showing that the efficiency of the system is affected by the imbalanced distribution of data among peers. We then estimate analytically this imbalance and its impact. After this preliminary study, the definition of the queuing model is given in Section 4.

Factor of efficiency. When a peer fails, it is replaced by a new peer with an *empty* disk. Since disks fill up during the system life, a recently replaced disk is empty, while an old disk contains many fragments. Hence, at any given time *disks with very heterogeneous number of fragments* are present in the system. This heterogeneity has a strong impact on the reconstruction process: (1) when a disk dies, the number of block reconstructions that start depends on the number of fragments present in this disk. A lot of fragments are lost if the disk was full, but much less for a young disk. (2) during the repair, the peers have to send fragments to the reconstructors that rebuild the missing fragments. A peer storing more fragments has to send a lot more fragments during this phase than a peer with fewer fragments. Hence, such peers become a bottleneck of the system, when on the contrary the less loaded peers stay idle during some part of the time.

To estimate the impact of this imbalance on the system, we introduce a *factor of efficiency* ρ when the system is under load, defined as

$$\rho(load) = \frac{work}{\min(load, bandwidth)}$$

where *load* is the sum over all peers of the number of fragments in their queues at the beginning of the time step; *bandwidth* is the total bandwidth of the system ($BW_{up} \cdot N\tau$) accounted in time steps of size τ; and *work* is the number of fragments that were effectively uploaded by the peers during the time step. When $\rho = 1$, the system works at its maximum speed, meaning that no peer was idle while another one could not finish its work. Note that ρ greatly depends of the load. If the load is very large compared to the bandwidth of the system, every peer works at almost full capacity and the efficiency is close to one. Similarly, when the load is small, everybody has few fragments to upload and all the work is done. But, between these two cases, the imbalance between the peers causes a range of inefficiencies.

Estimation of the Imbalance The disk size has in fact a very strong effect on the general imbalance of the system. Figure 1 shows a histogram with the number of fragments in failed disks. These results are obtained by simulation of $N = 200$ peers with $MTTF = 60$ days (1440 hours). The amount of data per peer is $14GB$. We set $s = r = 7$, and the fragment size $l_r = 2$ MB. Hence we have a total of $F = 7{\cdot}10^5$ fragments in the system. Then, the average number of fragments per peer is $\bar{D} = 7000$.

We denote the disk capacity of peers as C (number of fragments). Hence, $x = C/\overline{D}$ is the size factor of disks, i.e., how big is the disk when compared to the average amount of fragments per disk in the system. When the size factor $x = 3$ (that is, disk capacity

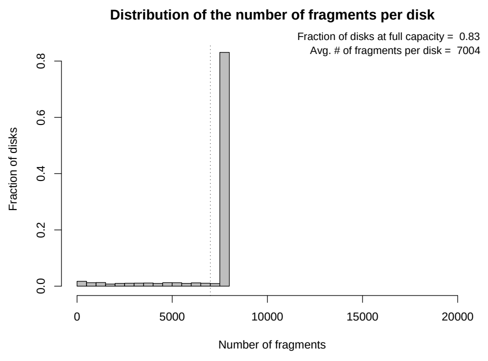
Table 1: Summary of the main notations.

N	Total number of peers
s	Number of initial fragments of a block
r	Number of redundancy fragments of a block
n	Number of fragments of a block, $n = s + r$
d	Repair degree of the Regenerating Code, by default $d = n - 1$
δ_{MBR}	Efficiency of the Regenerating Codes
L_f	Size of a fragment, in bytes
L_r	Amount of data to repair a fragment
B	Total number of blocks in the system
F	Total number of fragments in the system
α	Peer failure rate ($\alpha = 1/MTTF$)
N_F	Number of peers with full disks
φ	Ratio of full disks, N_F/N
C	Capacity of a disk (number of fragments)
$\bar{D}$	Average number of fragments per disk
x	Disk size factor, $x = C/\overline{D}$
BW_{up}	Peer upload bandwidth (kbit/s)
v	Rate at which a disk fills up (fragments per cycle)
$T_{\max}$	Number of time steps to fill up a disk, $T_{\max} = C/v$

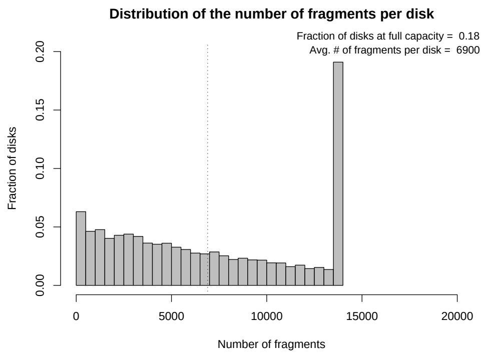
$C = 21,000$ fragments), the imbalance is very large. At the opposite, when $x = 1.1$, the disk size is close to the average number of pieces per disk in the system. Hence, most of the disk fillings become full, 83% in our example. The disks that are not full (17%) have an almost uniform distribution. In the following, we give a method to calculate that imbalance analytically.

Disk age and disk size distributions can be precisely approximated for systems with a large number of blocks. The block fragments are reconstructed by peers that have free space in their disks (i.e., there are $N - N_F$ such peers, where N_F is the number of peers with full disks). Since these peers are chosen at random to reconstruct the blocks, at each time step the distribution of the rebuilt fragments among peers follows a multinomial distribution with parameters: the number of rebuilt fragments and $1/(N - N_F)$. As the multinomial distribution is very concentrated around its mean, the *filling up process can be approximated by an affine process of its age*, in which, at each time step, each disk gets the number of reconstructed fragments divided by the number of non-full peers, roughly

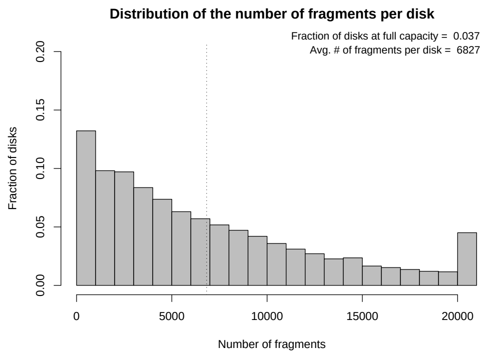
$$v = \frac{\alpha F}{N - N_F}$$



(a) $x = 1.1$



(b) $x = 2.0$



(c) $x = 3.0$

Figure 1: Distribution of fragments per failed disk for different disk size factor x of 1.1, 2, and 3. The number of full disks in each scenario is respectively 83%, 18%, and 4%. (y-scales are different)

where α is the peer failure rate. This filling process stops when the disk is full. That is after a number of time steps $T_{\max}$ such that $C = \alpha T_{\max} F/(N - N_F)$, where C is the peer disk capacity (maximum number of fragments per disk). The number of fragments of a disk thus depends on the age of the disk.

At each time step a disk has a probability α to experience a failure. Hence, the dead age of a disk follows a geometric law of parameter α. That is, $\Pr[\text{dead age} = T] = (1-\alpha)^{T-1}\alpha$. Hence the distribution of the number of fragments in a disk follows a truncated geometric distribution, that is, for $1 \leq T < T_{\max}$

$$\begin{aligned} Pr[D = vT] &= (1-\alpha)^{T-1}\alpha, \text{ and} \\ Pr[D = C] &= 1 - (1-\alpha)^{T_{\max}}. \end{aligned} \tag{1}$$

Note that here v, N_F, and $T_{\max}$ are unknown for the moment. The value of v depends on the number of full disks N_F, and of $T_{\max}$ depends directly of the filling rate v. To find the value of these variables, we use the fact that we know the expectation of the geometric distribution which is just the average number of fragments inside the system. This number is F/N (we neglect here the fragments that are in reconstruction, first order approximation for small α). Hence, we get $\mathbb{E}[D] = \overline{D} := F/N$. By definition, the expectation is also given by

$$\mathbb{E}[D] = \sum_{i=1}^{T_{\max}-1} vi(1-\alpha)^{i-1}\alpha + C(1-(1-\alpha)^{T_{\max}}).$$

To obtain $T_{\max}$, we now have to solve the equation:

$$\frac{1}{x} = \frac{1-\alpha-(1-\alpha)^{T_{\max}+1}}{\alpha T_{\max}},$$

obtained by identifying the two expressions for the expectation, by dividing by v, and because $C = x\overline{D}$. By solving that equation using the Maple software, we obtain that

$$T_{\max} = \frac{\alpha W(\frac{1}{\alpha}\ln(1-\alpha)x(1-\alpha)^{\frac{x+\alpha-x\alpha}{\alpha}}) - \ln(1-\alpha)x(1-\alpha)}{\ln(1-\alpha)\alpha},$$

where W is the Lambert W function. For example, when $MTTF = 1440$ hours ($\alpha = 1/1440$), the number of full disks and the number of time steps to fill up a disk are, for different disk capacities:

x	N_F (in %)	$T_{\max}(hours)$
1.1	83	278
1.5	42	1257
2	20	2293
3	6	4060

We verify that these values are very close to the ones obtained by simulation (Figure 1).

Effects of the Imbalance on the Bandwidth Efficiency Since some peers store less fragments, their load during the reconstruction process is also smaller. Thus, the overall bandwidth of the system is not fully utilized.

In a system using Regenerating Codes encoding, to repair a fragment, $d = n - 1$ small sub-fragments have to be sent to the peer in charge of the reconstruction. Simulations show that the speed of the reconstruction is given by the time that the *most loaded peer* takes to send the fragment. This time is in turn given by the number of fragments stored by this peer. We get this number from the distribution of the number of fragments per peer previously derived. For a majority of data blocks, *the most loaded peer storing one of its* *fragment is in fact a full disk.* This claim is valid for most systems in practice, that is, for the parameters usually found in the literature.

Indeed, recall that N_F denotes the number of full disks (and $\varphi = N_F/N$ the fraction of full disks). We compute the probability for a block that one of its fragment is on a full peer (with $n-1$ available fragments when it is being repaired). Recall also that a full disk stores x times the average number of fragments per disk in the system. Then, the fraction of fragments stored on full disks is φx. The probability of the block to have at least one fragment on a full disk is then

$$P_{full} = 1 - (1 - x\varphi)^{n-1}.$$

For a system with $n = 14$ (the value of N_F for different values of x is given above), the probability for different disk capacities is

x	1.1	1.5	2	3
φx	0.91	0.63	0.4	0.18
P_{full}	$1-10^{-14}$	$1-10^{-5}$	$1-10^{-3}$	0.92

We see that for most practical systems, each block has a fragment on a full disk. Hence, it is enough to consider the work done by the most loaded peers to obtain the reconstrution times. These peers have a load greater than the average load by a factor of $\frac{1}{x}$.

Factor of efficiency. An other way to phrase it: the factor of efficiency ρ of the system is approximately

$$\rho \approx \frac{1}{x}$$

where x is the fraction between disk capacity and the average number of fragments per disk.

More complex models for large disk capacities. We consider that in practice, for fairness issues, the storage system sets a limit of disk capacity not too far from the average amount of data stored. A factor x between 1.1 and 3 seems reasonable. For systems with a very large disk capacity (for example $x = 10$), ρ has to be estimated in a different way. As a matter of fact, a large number of blocks store no fragments on full disks. It is thus not enough to only consider the load of the full disks. This difficulty can be addressed by using a *multi-queue model.* The peers are partitioned into a number C of classes depending on the number of data they store. The model has one queue per class. When a disk fails, we estimate the number of fragments that each class has to upload, that is how much work they do, and in this way derive the factor of efficiency ρ. The analysis of this model is beyond the scope of our study here.

4 The Queueing Model

We introduce here a *Markovian Model* that allows us to estimate the reconstruction time under bandwidth constraints. The model makes an important assumption:

1. *The limiting resource is always the upload bandwidth.*

Assumption 1 is reasonable as download and upload bandwidths are strongly asymmetric in common installations. Using this assumption, we model the storage system with a *queue storing 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. The peers 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 it. The upload bandwidth of the system, $BW_{up}N$, is the limiting resource. Then, the *service* provided by the server is

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

which corresponds to the number of fragments that can be reconstructed at each time step τ. 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 is caused by peer failures. When a failure occurs, all the fragments stored in that peer 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 explained 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 (see Equation (1) for the probability distribution function of β). Recall that v is the speed at which empty disks get filled, and that $T_{max} = C/v$ is the elapsed time to fill a disk. Further on, β/v is the elapsed 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

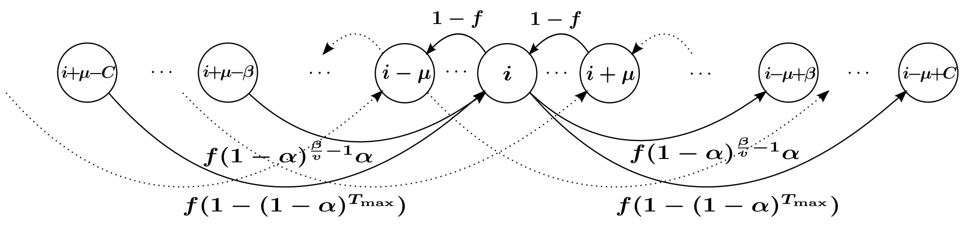

$1-f$
$1-f$
$i{+}\mu{-}C$
$i{+}\mu{-}\beta$
$i-\mu$
i
$i+\mu$
$i{-}\mu{+}\beta$
$i{-}\mu{+}C$
$f(1-\alpha)^{\frac{\beta}{v}-1}\alpha$
$f(1-\alpha)^{\frac{\beta}{v}-1}\alpha$
$f(1-(1-\alpha)^{T_{\max}})$
$f(1-(1-\alpha)^{T_{\max}})$

Figure 2: Transition around state i of the Markovian queuing model.

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 $\forall i \geq \mu$,

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

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 $i-\mu$:

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

Figure 2 presents the transitions for a state i. The following table summarizes the notation introduced in this section.

$Q(t)$	Number of fragments to be repaired
f	Batch arrival rate, $f = \alpha N$
β	Number of fragments on a failed disk (i.e., batch size)
ρ	Factor of efficiency, $\rho \approx \frac{1}{x}$
μ	Service rate, $\mu = \rho BW_{up} N\tau / L_r$ (fragments per time step)

4.2 Analysis

Here, we give the expressions to estimate the values of two important system metrics: the distribution of the block reconstruction time and the probability of data loss. These expressions are derived from the stationary distribution of the Markovian model, as presented in the following.

A Normalized Model. The queuing model has a service of μ and an input process of average $f\beta$. To simplify the presentation of the analysis, we introduce then a *normalized model* with service of 1, hence an input of mean $\beta' = \beta/\mu$.

4.2.1 Stationary Distribution

We analyze here the stationary state of this normalized queuing model. As the chain is irreducible and aperiodic, it exists when the service rate is larger than the load. Let P be the probability generating function of the Markovian model, that is P is defined as:

$$P(z) = \sum_i P_i z^i,$$

where P_i is the probability that the system is in state i, that is, i fragments have to be repaired.

The system reconstructs one block per time step (unless of course, no block is in the queue). It is translated in the generating function language into a division by z. The effect of a peer failure is translated by a multiplication by the probability generating function of the input I, defined as

$$I(z) = \sum_{j=0}^{\infty} I_j z^j,$$

with I_j the probability that the batch is of size j. Hence, we obtain the functional equation

$$\left(\frac{P(z) - P_0}{z} + P_0\right) I(z) = P(z).$$

It gives

$$P(z) = \frac{(z-1)P_0}{\frac{z}{I(z)} - 1}.$$

As $P(1) = 1$, $I(z) - z$ admits 1 as a root and thus can be written as $I(z) - z = (z-1)Q(z)$. We have

$$P(z) = \frac{P_0 I(z)}{Q(z)}. \tag{2}$$

As we have seen in Section 3, the size of the input follows a truncated geometric distribution of parameter α. A batch is of size vj with probability $(1-\alpha)^{j-1}\alpha$, for $j \in [0, 1, ..., T_{\max}]$. It gives

$$I(z) = (1-f) + f \sum_{j=1}^{T_{\max}-1} (1-\alpha)^{j-1}\alpha z^{vj} + f(1-\alpha)^{T_{\max}-1} z^{vT_{\max}}.$$

It can be rewritten as

$$I(z) = 1 + \frac{f(z^v - 1)(z^{T_{\max}}(1-\alpha)^{T_{\max}} - 1)}{(1-\alpha)z^v - 1}.$$

We factorize $I(z) - z$ by $(z-1)$. We get

$$\begin{aligned} Q(z) &= I(z) - z \\ &= (z-1)(-1 + \tfrac{f(\sum_{j=1}^{v} z^i)(z^{vT_{\max}}(1-\alpha)^{T_{\max}}-1)}{(1-\alpha)z^v - 1}). \end{aligned}$$

The value of P_0 is obtained by the normalization $\sum_{i=0}^{\infty} P_i = 1$ which implies $P(1) = 1$.

$$P_0 = \frac{Q(1)}{I(1)} = 1 - \frac{1}{\alpha}(fv((1-\alpha)^{T_{\max}} - 1)).$$

We now have an expression of the three terms of Equation 2 and we get a close form of the probability generating function $P(z)$.

4.2.2 Distribution of the Waiting Time

The distribution of the block reconstruction time is given by the stationary distribution P of the model calculated above. As we have Markovian (batch) arrivals, the probability for a batch to arrive when there are n blocks in the queue is exactly P_n (for the difference of distribution for an arriving customer and an outside observer, see for example [5]). If there are Q fragments in the queue when a batch of size $\beta' = jv$ arrives, the arriving fragments have waiting times of $Q+1$, $Q+2$, $Q+\beta'$. We define the probability generating function J as

$$J(z) = \sum_{j=1}^{T_{\max}} \Big((1-\alpha)^{j-1}\alpha \sum_{i=1}^{jv} z^i\Big).$$

The probability generating function W of the waiting times then is just

$$W(z) = P(z)J(z).$$

The distribution of the waiting times can then be directly obtained from the generating function by extracting its coefficients

$$\Pr(W = k) = [z^k]W(z) = \left.\frac{d^k W(z)}{k!(dz)^k}\right|_{z=0}. \qquad (3)$$

The first coefficients can be computed numerically and then a singularity analysis gives the asymptotic behavior, see for example [10]. Hence, the value of $\Pr(W = k)$ can be computed analytically. However, in the following, we also use another method and calculate them numerically by iterating the queuing model.

4.2.3 Number of Dead Blocks

The expected number of dead blocks is indirectly given by the model by computing the waiting time in the queue of a block that has to be reconstructed.

As a matter of fact, a block dies if it loses, before the end of the reconstruction, the $r-1$ fragments of redundancy that it has left when the repair starts, plus an additional fragment. The probability for a peer to still be alive after a period of time of θ time step is $(1-\alpha)^{\theta}$, where α is the probability for a disk to die during a time step, that is

$$\alpha = \frac{\tau}{MTBF}.$$

Hence a good approximation of the probability $\Pr[die]$ to die during a reconstruction lasting a time θ is given by

$$\Pr[die|W=\theta] = \sum_{i=r}^{s+r} \binom{s+r}{i} (1-(1-\alpha)^{\theta})^{i}((1-\alpha)^{\theta})^{s+r-i}.$$

For practical systems, the ratio $\theta/MTTF$ is small as the probability to of data loss should be very low. Hence $\Pr[die]$ is well approximated by

$$\Pr[die|W=\theta] \approx \binom{s+r}{r} (1-(1-\alpha)^{\theta})^{r}((1-\alpha)^{\theta})^{s-1}.$$

From this and from the distribution of the waiting time, we get the probability to die during a reconstruction, P_D, with

$$P_D = \sum_{i=0}^{\infty} \Pr[\text{die}|W=i] \Pr[W=i].$$

The number of dead blocks during a time T, D_T, is then obtained by the number of reconstructions during T, R_T:

$$D_T = P_D R_T. \tag{4}$$

4.2.4 Bandwidth Usage

The bandwidth usage is directly given by the distribution of the number of reconstructions being processed by the system, which comes from the stationary distribution of the queuing model.

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 peer. 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 two important practical implementation points: (1) when choosing the parameters of the Regenerating Code, it is important to give to the peer in charge of the repair a choice between several peers to retrieve the data; (2) we show the strong impact of different scheduling options on the data loss rate.

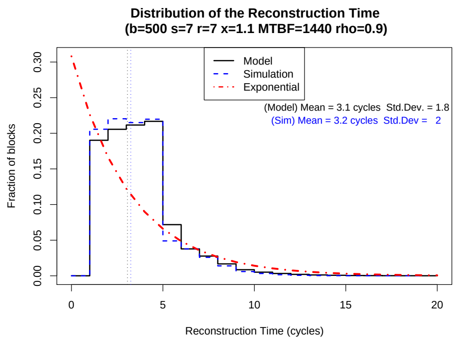
5.1 Distribution of Reconstruction Time

Figure 3 shows the distribution of the reconstruction time and the impact of the peer 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 peer).

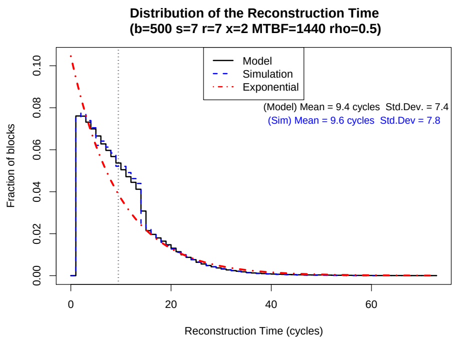
First, we see that the model (dark solid line) closely matches the simulations (blue dashed line). For example, when $x = 1.1$ (top plot), the curves are almost merged. The average reconstruction times are 3.1 cycles vs 3.2 for the model. We see that there is a small gap when $x = 3$. As a matter of fact, we saw in Section 3 that simulating the queue of the full disks is an approximation in this case, as only 92% of the blocks have a fragment on a full disk.

Second, we confirm the strong impact of the disk capacity. We see that for the three values of x considered, the shape 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 replace by a trapezoidal shape explained by the large range of disk fillings.

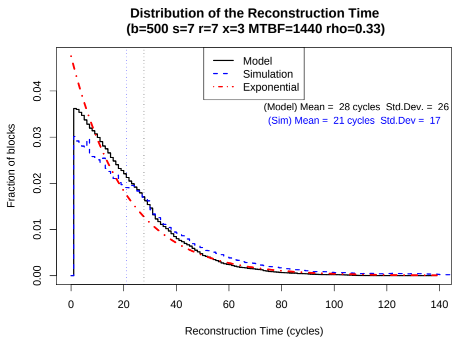
Third, we compare the distributions obtained with the exponential distribution that is classically used in the literature. We see that the distributions are far from the exponential when $x = 1.1$ and $x = 2$, but get closer for $x = 3$. Hence, as we will confirm, the exponential distribution is only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



(a) $x = 1.1$



(b) $x = 2.0$



(c) $x = 3.0$

Figure 3: 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).

Figure 4 presents the distribution of a distributed storage system experiencing three different rates of failures: MTTF of 90, 180 and 360 days. We clearly see the evolution of the shape of the distribution due to the larger probability to experience failures when the peer 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 2 a small subset of these experiments.

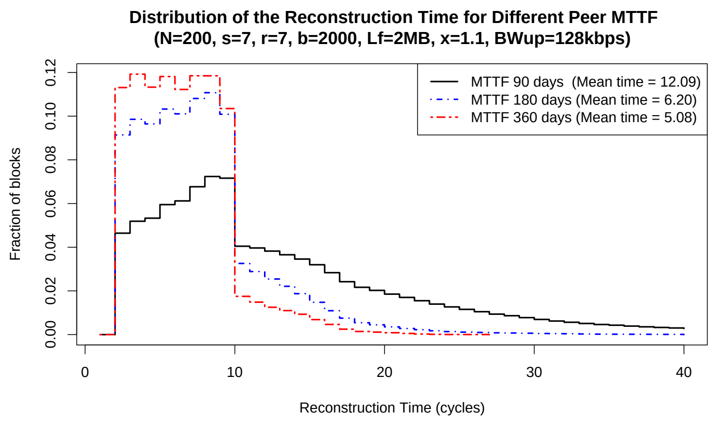


Figure 4: Distribution of reconstruction time for different MTBF. Different shapes for different values.

Table 2: 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 (MTBF).

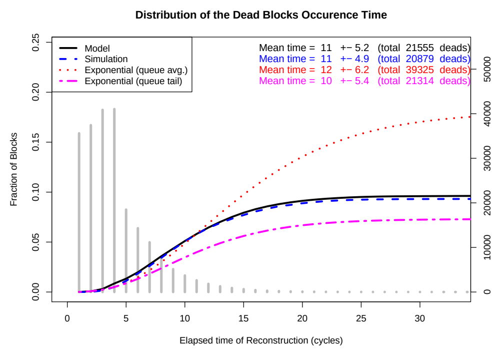
$MTBF$	60	120	180	365
T_{sim}	3.26	2.90	2.75	2.65
T_{model}	2.68	2.60	2.49	2.46

(c) Peer Upload Bandwidth (kbps).

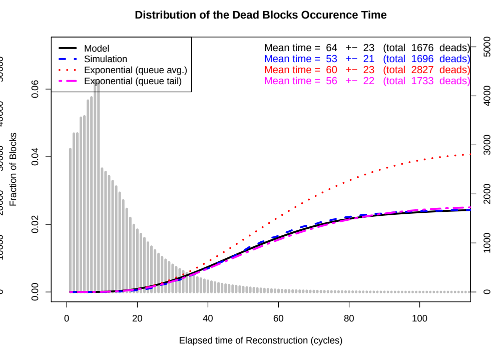
$upBW$	64	128	256	512
T_{sim}	8.9	3.30	1.70	1.07
T_{model}	8.3	3.10	1.61	1.03

5.2 From Where the Deads Come From?

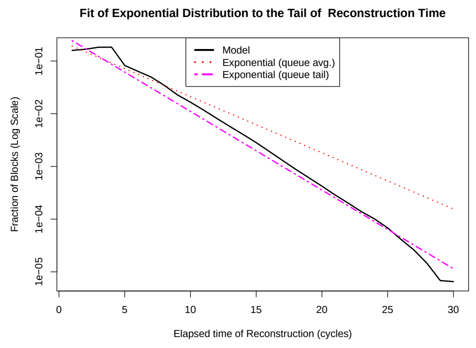
In this section, we discuss in which circumstances the system has more chances to lose some data. First a preliminary remark: backup systems are conceived to experience basically no data loss. Thus, for realistic sets of parameters, it would be necessary to simulate the systems for a prohibitive time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



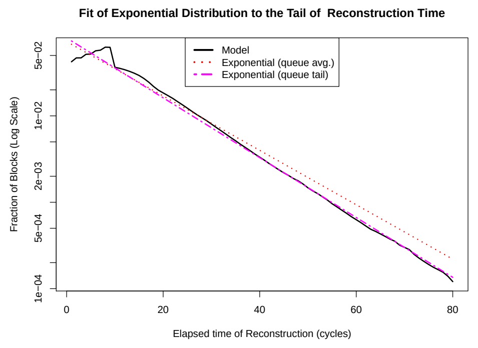
(a) Scenario A



(b) Scenario B



(c) Scenario A (fitting)



(d) Scenario B (fitting)

Figure 5: (Top): 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. (Bottom): Fitting of exponential distribution with the tail of queueing model (axis scales are different).

We plot in Figure 5 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 queueing 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 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 5b and 5d). 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 Discussing the Implementation of Regenerating Codes

As presented in Section 2, when the redundancy is added using regenerating codes, $n = s+r$ peers store a fragment of the block when s are enough to retrieve the block. When a fragment is lost, $s \leq d \leq n-1$ peers are in charge of repairing the fragments. The larger d is, the smaller is the bandwidth needed for the repair. Figures 6 and 7 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 the peers that send a sub-fragment to the peer that will store the repaired fragment. Hence, the system is able to lower the load of the more loaded disks and to *balance more evenly the load between peers.*

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 peers uploads the sub-fragments. The full disks store a proportion φx of the fragments of the system, with φ the fraction of full disks. We simply look at the how much work we *must* do on the

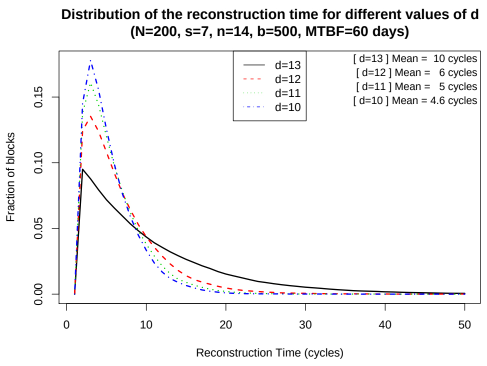


Figure 6: Distribution of reconstruction time for different values of degree d.

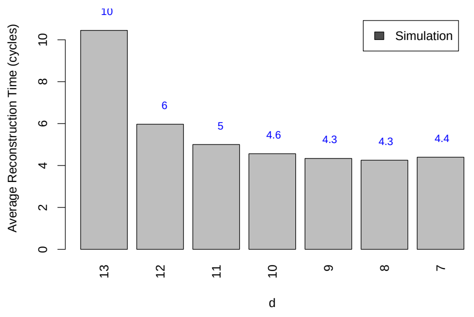


Figure 7: Average Reconstruction Time for different values of degree d. Smaller d implies more data transfers, but may mean smaller reconstruction times!

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 sends $i-\delta$ units of work the full disks (whenever $i \geq \delta$). So the load of the full disks is

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

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

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.

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

5.4 Scheduling

As peers have a large number of repairs to carry out but very limited bandwidth, the question of which repairs to do first is crucial. In this section, we study three different scheduling choices: FIFO, Random, and Most-Damaged data block first.

The FIFO is the default scheduling in the simulator, as discussed in Section 2, the blocks are processed in the order of arrival. In the Random scheduling, the simulator processes

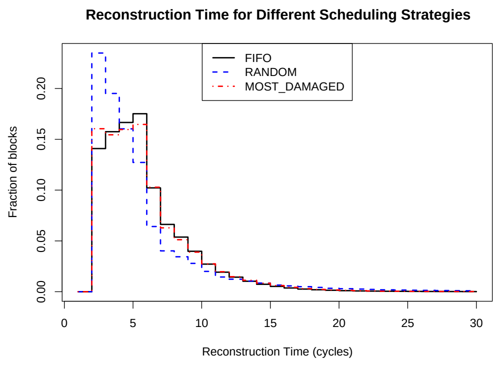


Figure 8: Reconstruction time for different scheduling strategies. The average reconstruction time is almost the same (4.4 cycles), but the distribution changes.

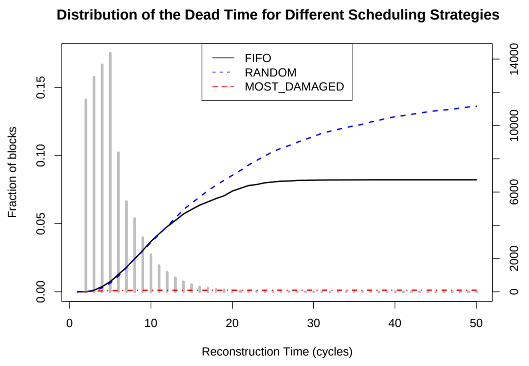


Figure 9: Cumulative number of dead blocks for different scheduling strategies. Processing the most damaged first is the best strategy.

blocks in a random order (at each time step the list of blocks to be reconstructed is shuffled). In the Most-Damaged scheduling the blocks are ordered by the level of redundancy (i.e., blocks with less fragments available come first). In case of tied values, then the FIFO order is assumed.

Figure 8 presents the reconstruction time of these three schedulings. All strategies give almost the same average reconstruction time, 4.40, 4.43, 4.43 respectively for FIFO, Random and Most-Damaged. We see that their distribution changes slightly. In the Random order the shape has the form of a geometric distribution, with many blocks finishing the reconstruction "early". However, as depicted in Figure 8, the differences in the number of dead blocks are enormous. When using the Random scheduling, the dead increases considerably, as expected.

Most-Damaged has a reconstruction time very close to the others but the number of losses is much lower. Hence, this is the strategy of choice when implementing such systems.

6 Experimentation

Aiming at validating the simulation and the model results, we performed a batch of real experimentation using the Grid'5000 platform [11]. We used a prototype of storage system implemented by a private company (Ubistorage [20]).

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 [19]. The lower layer is in charge of managing the logical topology: finding peers, routing, alerting of peer 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 [14] to introduce redundancy. Each data block has a peer responsible of monitoring it. This peer keeps a list of the peers storing a fragment of the block. The fragments of the blocks are stored locally on the Pastry leafset of the peer in charge [13].

Monitoring the system. The storage system uses the information given by the lower level to discover peer failures. In Pastry, a peer checks periodically if the members of its leafset are still up and running. When the upper layer receives a message that a peer left, the peer 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 peers can be adjusted.

6.2 The Grid'5000 Infrastructure

Grid'5000 is an infrastructure dedicated to the study of large scale parallel and distributed systems. It provides a highly reconfigurable, controllable and monitorable experimental platform to scientists. The platform contains 1582 machines accounting for 3184 processors and 5860 cores. The machines are geographically distributed on 9 different hosting sites in France (two additional sites in Luxemburg and Porto Alegre, Brazil are being added). These site are connected to RENATER Education and Research Network with a 10Gb/s link.

6.3 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 when doing experimentation this is not the case. We need to plan our experiments to last 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 gigabytes to 100 megabytes, a reduction of $100\times$), inducing a much smaller time to repair

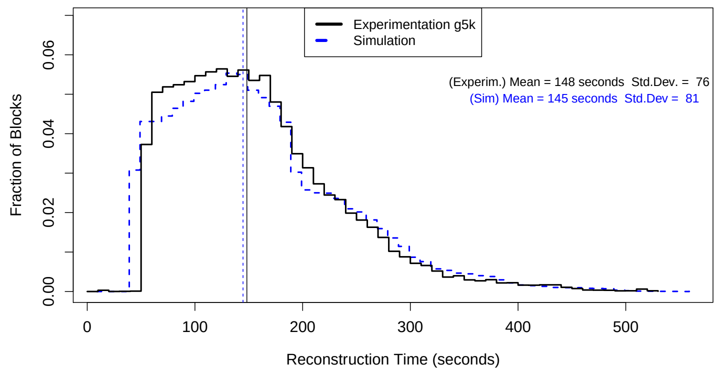


Figure 10: Distribution of reconstruction time on a experimentation with 64 nodes during 4 hours compared to simulations.

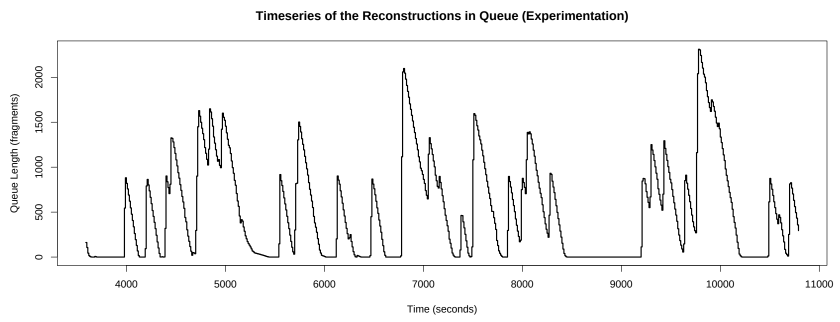
a failed disk. Then, the peer failure rate is increased (from months to a few hours) to keep the ratio between disk failures and repair time proportional. The bandwidth limit value, however, is kept close to the one of a "real" system. The idea is to avoid inducing strange behaviors due to very small packets being transmitted in the network.

Figure 10 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 from some seconds.

Moreover, to have an intuition of the system dynamics over time, in Figure 11 we present a timeseries 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 determinist 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 Conclusion

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 the correlation between data repairs that compete for the same bandwidth. We bring to light the impact



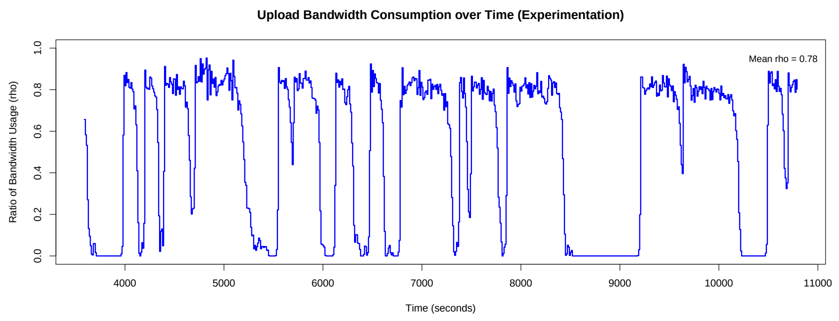


Figure 11: Timeseries of the queue size during time (top) and the upload bandwidth ratio (bottom).

of peer heterogeneity on the system efficiency. The model is validated by simulation and by real experiments on the GRID'5000 PLATFORM.

We show that the exponential distribution classically taken to model the reconstruction time is valid for certain sets of parameters, but that different shapes of distribution appear for other parameters. 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 system 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.

In addition, we confirm the strong impact of scheduling on the system loss rate.

References

[1] S. Alouf, A. Dandoush, and P. Nain. Performance analysis of peer-to-peer storage systems. *Proceedings of the 20th International Teletraffic Congress (ITC)*, LNCS 4516:642–653, 2007.

[2] 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*, pages 337–350, 2004.

[3] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. *ACM SIGMETRICS Perf. Eval. Review*, 28:34–43, 2000.

[4] 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*, pages 45–58, 2006.

[5] R. B. Cooper. *Introduction to Queuing Theory.* North Holland New York, 1981.

[6] 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)*, pages 1–8, France, 2009.

[7] 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)*, pages 133–144, 2006.

[8] A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Proc. of IEEE INFOCOM*, pages 2000–2008, May 2007.

[9] A. Duminuco and E. Biersack. Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems. In *Proc. of IEEE Intl. Conf. on Peer-to-Peer Computing (P2P)*, pages 89–98, 2008.

[10] P. Flajolet and R. Sedgewick. *Analytic combinatorics.* Cambridge University Press, 2008.

[11] Grid5000Platform. https://www.grid5000.fr/.

[12] 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*, 28(5):190–201, 2000.

[13] S. Legtchenko, S. Monnet, P. Sens, and G. Muller. Churn-resilient replication strategy for peer-to-peer distributed hash-tables. In *Proceedings of SSS*, volume LNCS 5873, pages 485–499, 2009.

[14] 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*, pages 150–159, 1997.

[15] 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)*, volume 2, pages 532–538, Oct. 2007.

[16] Planetlab.

[17] S. Ramabhadran and J. Pasquale. Analysis of long-running replicated systems. In *Proc. of IEEE INFOCOM*, pages 1–9, Spain, 2006.

[18] S. Ramabhadran and J. Pasquale. A resource allocation problem in replicated peer-to-peer storage systems. In *Proc. of IEEE Intl. Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–8, 2007.

[19] 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)*, volume LNCS 2218, pages 329–350, 2001.

[20] Ubistorage. http://www.ubistorage.com/.

[21] 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)*, volume LNCS 2429, pages 328–337, 2002.

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399