P2P Storage Systems: How Much Locality Can They Tolerate?

Frédéric Giroire, Julian Monteiro, Stéphane Pérennes

P2P Storage Systems: How Much Locality Can They Tolerate?

Frédéric Giroire and Julian Monteiro and Stéphane Pérennes
MASCOTTE, joint project INRIA - I3S - CNRS - Univ. of Nice-Sophia, France
Email: {firstname.lastname}@sophia.inria.fr

***Abstract*—Large scale peer-to-peer systems are foreseen as a way to provide highly reliable data storage at low cost. To achieve high durability, such P2P systems encode the user data in a set of redundant fragments and distribute them among the peers. In this paper, we study the impact of different data placement strategies on the system performance when using erasure codes redundancy schemes. We compare three policies: two of them *local*, in which the data are stored in logical neighbors, and the other one *global*, in which the data are spread randomly in the whole system. We focus on the study of the probability to lose a data block and the bandwidth consumption to maintain enough redundancy. We use simulations to show that, without resource constraints, the average values are the same no matter which placement policy is used. However, the variations in the use of bandwidth are much more bursty under the *local* policies. When the bandwidth is limited, these bursty variations induce longer maintenance time and henceforth a higher risk of data loss. Finally, we propose a new *external reconstruction* strategy and a suitable degree of locality that could be introduced in order to combine the efficiency of the global policy with the practical advantages of a local placement.**

I. Introduction and System Description

The key concept of Peer-to-Peer storage systems is to distribute redundant data among peers to achieve high reliability and fault tolerance at low cost. The addition of redundant data could be done by trivial *Replication* [1], in which identical copies of data are sent to different nodes in the system; or be based on *Erasure Codes* [2], such as Reed Solomon and Tornado, as used by some RAID schemes [3]. Hereafter, we summarize the main parts usually found in such a system:

Data Redundancy. When using Erasure Codes, the original user data (e.g. files, raw data, etc.) is cut into data-blocks that are divided into s initial *fragments* (or pieces). The encoding scheme produces $s + r$ fragments that can tolerate r failures. In other words, the original data-block can be recovered from any s of the $s+r$ encoded fragments. In a P2P storage system, these fragments are then placed on $s+r$ different peers of the network according to a placement policy (which is the main subject of our study).

Control Mechanism. To ensure a durable long-term storage despite disk failures, the system must be capable to maintain a minimum number of fragments available in the network. This means that the system continuously monitors the number of fragments of each data-block. This control is done in a distributed way by the means of a Distributed Hash Table (DHT) [4]. If this number of fragments drops to a threshold value r_0, the block need to be reconstructed.

Reconstruction Strategy. Setting a low value for r_0 decreases the number of reconstructions (as the reconstruction starts only after that $r-r_0$ pieces are lost), but increases the probability to lose a block. After the reconstruction, the regenerated missing pieces are spread among different nodes.

Data Placement Policies. It has been shown that fragment (or replica) placement has a strong impact on the system performance [5]. We study here three different strategies of data placement as explained in the following and depicted in Figure 1:

- *Global & Random Policy:* The $s+r$ fragments of a block are sent to $s+r$ peers chosen uniformly at random *among all the N peers* present in the system (see [6]).
- *Chain Policy:* In this policy the fragments of blocks are stored on the $s + r$ closed set of "logically" (consecutive) neighbors peers. This policy corresponds to what is done in most distributed systems implementing a DHT (see [7]).
- *Buddy (or RAID) Policy:* This is an extreme case of a *local* policy, in which the system is composed of small independent subsystems with $s + r$ peers each. It could be seen as a collection of local RAID like storage. In this situation, each peer in the group stores exact the same set of pieces.

The use of the Global strategy allows to distribute more uniformly the load among peers, leading to a faster reconstruction and a smoother operation of the system [6]. However, the use of *local* strategies brings practical advantages [8]. For instance, the DHT update mechanisms of the leafset can be used to simplify the management of the system (e.g. to know the states of the blocks stored locally). Also, the management traffic and the amount of meta-information to be stored by the nodes are kept low.

Our goal is to investigate the amount of resource (bandwidth and storage space) required to maintain redundancy and to ensure a given level of reliability. In this paper[1], we study the impacts of placement policies for two different scenarios. In Section III-A, we study a *provisioning scenario*, where peers do not have bandwidth constraints. It allows to estimate the bandwidth use for different sets of parameters. Then, in Section III-B, we study a scenario where peers have resource constraints, that corresponds to the operation of practical systems. Finally, in Section IV we propose some improvements of the placement and reconstruction architectures.

[1]An extended version of this work can be found in the INRIA Research Report RR-7006, http://hal.inria.fr/inria-00408078

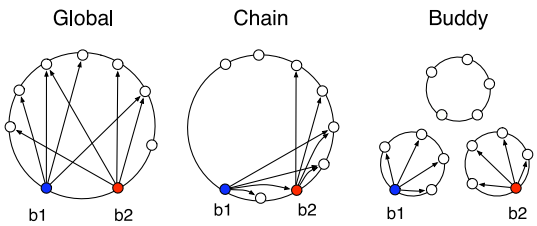

Fig. 1. Placement of two blocks $b1$ and $b2$ in the system. Global: $s+r$ fragments are placed at random among all peers; Chain: fragments are placed on $s+r$ neighboring peers; Buddy: many small subsystems of size $s+r$, in this case all peers inside each small group contain the same data.

Related Work

The majority of existing or proposed systems, e.g., Intermemory, CFS, Farsite, PAST, TotalRecall, Glacier, use a local placement policy. Chun et al. in [6] also discuss the impacts of placement strategies, but they do not address the case of Erasure Codes. In [5] the authors study the impact of data placement on the Mean Time to Data Loss (MTTDL) metric. But they do not discuss other very important metrics: the probability to lose a block and the bandwidth usage.

II. Simulations

To evaluate such a system, we developed a *custom cycle-based simulator* that implements all the characteristics described in Section I. The simulator models a detailed view of the system, as it monitors the state and the localisation of each fragment individually.

Modeling Failures. It is assumed that the nodes stay connected almost all the time into the system. So, we model the case of *peer failures*, mainly caused by a disk crash or by a peer that definitively leaves the system. In both cases, it is assumed that all the data on the peer's disk are lost. Following most works on P2P storage systems [5], peers fails independently according to a memoryless Poisson process. To avoid the problem of transient failures and deal with churn, a peer is just considered lost if it has left the system for a period longer than a given timeout [9] (set to $\theta = 12$ hours in our simulations).

Monitored metrics. The simulator keeps detailed traces of different performance metrics. To be sure that we are studying a system in a steady state, the first part of the simulation traces is thrown away. We focus our analysis on three main metrics:

Bandwidth: Average bandwidth consumption per peer, i.e., estimated from the number of pieces transmitted and received per hour due to the reconstruction process;

FDLPY: Fraction of Data Loss Per Year, which gives the probability to lose a data-block per year;

MTTDL: Mean Time To Data Loss, i.e., the period of time between two occurrences of data loss in the system.

Simulation parameters. We did a large number of simulations for different sets of the parameters. Otherwise explicit indicated, the main ones are $N = 1005$ the number of peers, $F = 1.5 \cdot 10^6$ the total number of fragments. The fragment size is 400KB, thus, with $s = 9$ and $r = 6$, the system block size is 6MB. The reconstruction threshold $r_0 = 2$. The amount of stored data and the number of peers are kept constant during the simulation, this means that dead blocks are re-injected in the system. Crashed disks reappear empty. The time-step of the simulator is $\tau = 1$ hour and the simulated time is 10 years.

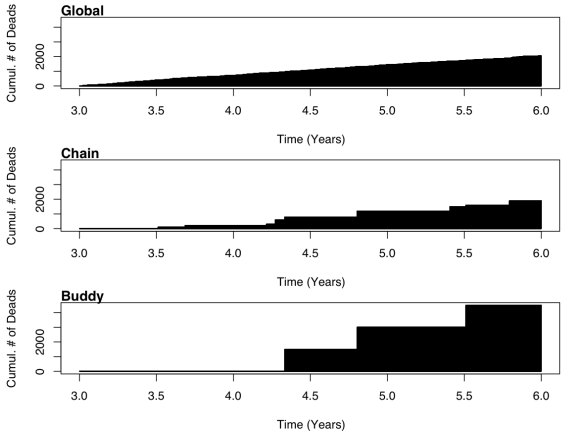

Fig. 2. Illustrative example of the cumulative number of dead blocks for a period of three years.

Peer bandwidth. Each peer has a maximum upload and download bandwidth, resp. BW_{up} and BW_{down}. We assume asymmetric capacities, e.g., ADSL lines (in our experiments $BW_{down} = 5BW_{up}$). So the limiting resource is the upload bandwidth and it is the one presented in our results. When the peer's bandwidth is limited, not all blocks can be reconstructed at the same time. To model a peer's bandwidth, we implemented a *non blocking FIFO queue with one server*: when there is a peer failure, the blocks to be reconstructed are put in the queues of the peers in charge of the reconstruction.

Remark 1 (Size of the simulated system) *In practice, peers have huge disks of tens of Gigabytes, each one containing tens of thousands of blocks. As we want to be able to simulate a storage system for several years in a reasonable time, we chose a disk size and bandwidth limits around 100 times smaller than the one expected in practice.*

Remark 2 (Measuring block losses) *As it is difficult to simulate in a reasonable time events of very low probability, for example, a probability of lose data of* 10^{-20}*, we chose* non realistic values *for some parameters (in particular, the reconstruction threshold* $r_0 = 2$ *and the disk* $MTBF = 90$ *days are set very low). In this way, we experience data loss in our simulations.*

III. Simulation Results

A. Without Resource Constraints

First, we study the provisioning scenario (*unlimited bandwidth*), which is important to measure the required bandwidth to maintain the system. Briefly, the results shown here are: (1) the three placement strategies have the same value of average bandwidth demand; (2) however *local* policies exhibit strong variations in resource usage across peers; (3) they have the same probability to lose a data-block, (4) but the MTTDLs of the Buddy and the Chain policies are longer.

TABLE I
SUMMARY OF RESULTS (WITHOUT BANDWIDTH CONSTRAINTS).

Policy	Bandwidth (kbit/s)		FDLPY (blocks)		MTTDL (years)	
Global	1.99	($\pm$ 1.34)	$4.1 \cdot 10^{-4}$	($\pm$ $0.6 \cdot 10^{-4}$)	0.02	($\pm$ 0.02)
Chain	1.99	($\pm$ 12.83)	$4.1 \cdot 10^{-4}$	($\pm$ $8.6 \cdot 10^{-4}$)	4.0	($\pm$ 3.0)
Buddy	1.99	($\pm$ 15.92)	$4.4 \cdot 10^{-4}$	($\pm$ $25.4 \cdot 10^{-4}$)	25.8	($\pm$ 21.7)

1) **Bandwidth Usage:***:* The left column of Table I shows the average value of upload bandwidth usage across peers during time, (i.e., at each time step we measure the average number of fragments transmitted by each peer), along with the experimental standard deviation (in parenthesis). First, as expected, the average bandwidth use across peers is roughly the same for all policies. The reason is that the different placement policies do not change the number of pieces that have to be reconstructed, but they change the repartition of these pieces among peers.

However, the variations are not the same. The Chain policy and Buddy policy variations are significantly higher. Analysing the bandwidth usage per user at a typical instant of time we see that the load is around 2 kbit/s for all the users and all strategies. However, we see that the distributions of the bandwidth are not the same at all. In the case of Global policy, the reconstruction load is evenly distributed among all the peers. On the opposite scenario, Buddy policy, when a failure happens, only the immediate neighbors possess the remaining pieces of the blocks, which induces a high bandwidth demand on the group of $s+r$ peers. The situation for the Chain policy is similar to the Buddy, but less correlated, as a peer failure induces reconstructions on blocks in a distance of $2(s+r)-1$.

2) **Probability to Lose a Block***:* The probability to lose a block in the three different policies are shown in the middle column of the Table I, normalized as the Fraction of Data Loss Per Year (FDLPY). When there is no bandwidth limit, the *expected number of dead blocks is the same for the three policies*. As a matter of fact, the probability for a block to die does not depend on where its fragments are placed. It can be easily calculated using a Markov Chain Model, see for example [10]. But, we note that the *deviations* during time of the number of dead blocks is higher for *local* policies. To explain that, we look at the MTTDL.

3) **Mean Time To Data Loss (MTTDL)***:* The measure of the time between two occurrences of data loss shows that the three policies have very distinct behaviors, as depicted in the right column of Table I. However, as we have seen before, the three policies have in average the same number of dead blocks per year. In other words, the average quantity of data loss per year is the same, but the distribution across time of these losses is very different.

The Figure 2 illustrates an example of the cumulative number of dead blocks for a period of 3 years for the three placement policies under the same fault scenario. We see that the loss occurs regularly for the Global policy. Conversely, they occur very rarely for the Buddy placement, but, when they occur, they affect a large batch of data. Basically, all the blocks of a small buddy subsystem of size $s+r$ peers lose all their blocks at the same time. The behavior of the Chain policy is somewhere in the middle of both.

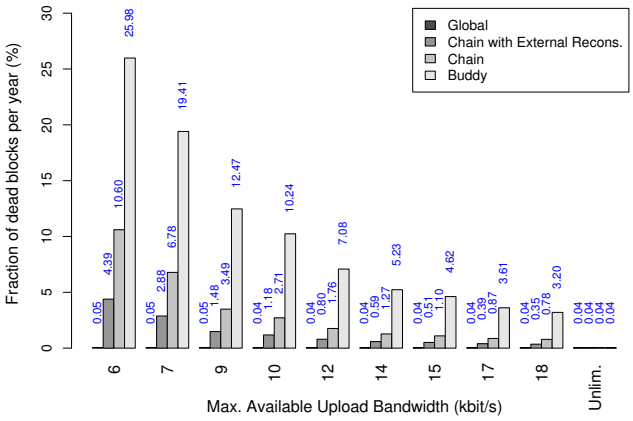

Fig. 3. *Fraction of block losses per year* (see Remark 2) for different bandwidth limits. Note that, for this set of parameters, the results obtained with the chain policy using external reconstruction shows an improvement of about 50% to the original chain policy.

B. Results under Resource Constraints

In this section, we study the behavior of the system with bandwidth limitation per peer (each peer having a maximum upload and download bandwidth). In this context we show that, using similar available resources, the amount of data loss is no more the same for the three data placement policies. The Global policy behaves considerably better in comparison to the Chain and Buddy policy. Furthermore, the *local* policies now experience more loss events (smaller MTTDL).

1) **Reconstruction Time versus Bandwidth***:* Experiments shows that limiting the available bandwidth the average reconstruction time is a lot longer for the Chain policy and even more for the Buddy policy when compared to the Global one. As an example, for a maintenance bandwidth of 6 kbit/s, the reconstruction time is around 49 hours for the Chain policy and 82 hours for the Buddy, but only 2 hours for the Global policy (Figure is not shown for space reasons). This bandwidth limit corresponds to three times the average bandwidth usage of the system (as computed without resource constraints). Hence, we see that the irregularity of the reconstruction load among peers has a very strong impact on the reconstruction time, even if each policy has the same average bandwidth demand. Thus, under resource constraints, the big local events constituted by peer failures induce longer reconstruction time and henceforth an increase of data loss when using the *local* policies, as shown in the following.

2) **FDLPY versus Bandwidth***:* A critical performance measure of a P2P storage architecture is the probability to lose a block for a given amount of bandwidth. Figure 3 compares the trade-offs of the three policies for different values of BW_{up}. We see that the Global policy behaves a lot better for any bandwidth limit than the Chain policy, which itself is more efficient than the Buddy policy. For example, for a bandwidth limit of 18 kbit/s (which represents 9 times the average bandwidth need of the system), the Global experiences 0.04% of data loss per year, to compare with 0.78% and 3.2% for the Chain and the Buddy, respectively.

3) **MTTDL versus Bandwidth***:* Opposed to what was showed without bandwidth constraints, the Global policy behaves better than the others with low bandwidth limitations. For instance, without resource constraints, the time between data loss were 0.02, 4.0, and 25.8 years respectively for the Global, Chain and Buddy. Conversely, with an available bandwidth of 6 kbit/s, these values are 166, 53, and 75 (in hours), many orders of magnitude less. These results show that the impact of the bandwidth limits per peer needs to be taken into account when analysing such systems.

IV. Proposition for P2P Storage System Architectures

In this section, we propose a new reconstruction architecture for the Chain policy, namely *external reconstruction*, and show that it can lower the duration of the sending phase of reconstructions, and thus improve the probability to lose data.

The idea is to use peers outside the Chain group to carry out the reconstruction process. In this way, the bandwidth usage is more uniformly spread among peers. More precisely, only the upload bandwidth of the retrieval phase of the reconstruction is needed locally, while the bandwidth for the sending phase is provided by all the peers of the system. Hence, the *External Reconstruction* has two main advantages: a local control for discovering failed peers and updating the data-blocks' states; a more uniform distribution of resources among peers, which lowers the reconstruction time. However, a small cost is paid: the external peer in charge of the reconstruction does not contain any previous piece of the reconstructing block.

A rough estimate of the gain in terms of reconstruction time can be given. In the internal reconstruction, local peers have to support $s+r-r_0$ uploads of pieces. However, when using the external reconstruction, they only have to support s uploads of pieces. As the local peers basically are the bottleneck of the reconstruction, the gains in terms of bandwidth and hence of reconstruction time are roughly $1-s/(s-1+r-r_0)$.

Note that the gains in terms of data loss will be significantly higher. Figure 3 also compares the internal and external policies. It gives the trade-off between the average number of dead blocks per year and the available bandwidth. For the same bandwidth, the fraction of data loss decreases by a factor between 0.5 and 0.6 for this set of parameters.

Exponential relation between the probability to die and the reconstruction time. During a reconstruction, a block dies if it loses r_0+1 fragments before it finishes. The probability for a peer to be alive after a time T is $e^{-\lambda T}$, where λ is the peer failure rate. Hence a good approximation of the probability to die during a reconstruction lasting a time T is given by

$$\Pr[die|Rtime=T]=\binom{s+r_0}{r_0+1}(1-e^{-\lambda T})^{r_0+1}(e^{-\lambda T})^{s-1}.$$

Hence we have an exponential relationship between the number of block losses and the neighborhood size. The neighborhood size should mainly be chosen in function of two parameters: the disk size and the peer bandwidth. Note that a size of $\frac{D}{(r-r_0)BW_{up}}$ allows to reconstruct the blocks in one time step and is sufficient to get the benefits of Global (with D the number of fragments per disk, BW_{up} expressed in blocks/time step and $1/(r-r_0)$ the fraction of blocks of the lost disk that go beyond the threshold).

Concluding, to implement a local policy, the neighborhood should at least be a little bit larger than $s+r$, as the marginal utility of increasing the neighborhood size is tremendous for very small sizes. In addition, the neighborhood size should be chosen in function of the disk size: Larger the number of fragments per disk, the larger the neighborhood should be.

V. Conclusion

In this paper, we showed that placement policies strongly impact the performance of P2P storage systems. We studied three different policies, a Global and two *local*, and showed that, under resource constraints the Global policy behaves better in terms of probability to lose data and MTTDL than the *local* policies. We showed that, by using a new reconstruction strategy, namely *external reconstruction*, and by increasing the size of the neighborhood, local policies can have performances almost equivalent to the ones of the Global, while keeping their practical advantages.

Acknowledgment

This work was partially funded by the European project IST/FET Aeolus and the ANR projects Spreads and Dimagreen.

References

[1] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," in *Proc. of ACM SOSP*, 2001.

[2] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. of IPTPS*, vol. 2, 2002, pp. 328–338.

[3] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proc. of ACM SIGMOD*, 1988.

[4] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the evolution of peer-to-peer systems," in *Proc. of PODC*, 2002.

[5] Q. Lian, W. Chen, and Z. Zhang, "On the impact of replica placement to the reliability of distributed brick storage systems," in *Proc. of ICDCS'05*, vol. 0, 2005, pp. 187–196.

[6] 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 NSDI*, 2006, pp. 45–58.

[7] J. R. Douceur and R. P. Wattenhofer, "Large-scale simulation of replica placement algorithms for a serverless distributed file system," in *Proc. of MASCOTS*, 2001, pp. 311–319.

[8] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, "Designing a DHT for low latency and high throughput," in *Proc. NSDI*, San Francisco, California, 2004, pp. 85–98.

[9] R. Rodrigues and B. Liskov, "High availability in dhts: Erasure coding vs. replication," in *Peer-to-Peer Systems IV*. LNCS, 2005, pp. 226–239.

[10] O. Dalle, F. Giroire, J. Monteiro, and S. Pérennes, "Analysis of failure correlation impact on peer-to-peer storage systems," in *Proc. of the 9th Intl. Conf. on Peer-to-Peer Computing (P2P'09)*, 2009, to Appear.