Erasure Code Replication Revisited

W. K. Lin, D. M. Chiu, Y. B. Lee
Department of Information Engineering, The Chinese University of Hong Kong
{wklin3, dmchiu, yblee}@ie.cuhk.edu.hk

Abstract

Erasure coding is a technique for achieving high availability and reliability in storage and communication systems. In this paper, we revisit the analysis of erasure code replication and point out some situations when whole-file replication is preferred. The switchover point (from preferring whole-file replication to erasure code replication) is studied, and characterized using asymptotic analysis. We also discuss the additional considerations in building erasure code replication systems.

1 Introduction

The tremendous growth in compute power, storage and communication bandwidth in personal computers and devices have led to proposals for building "serverless" systems to more economically provide traditional services. For example, [1] proposed a serverless file system; [2] and [3] proposed a serverless video streaming system; [4] proposed a distributed secure information dispersal system. Such systems may have varying degree of decentralization in management, thus can be considered either as clusters or peer-to-peer systems depending on where they situate in the spectrum.

At some level of abstraction, all serverless systems offer a storage service. Therefore, a very important issue is data availability. Since no component is 100% reliable, we cannot have 100% availability; but we can achieve very high availability by replication. Normally, our notion of replication is redundancy by creating extra copies. It is a simple trade-off of storage overhead and availability. If you create **S** copies of a file, you increase the storage overhead by **S**, but reduce the probability that none of the copies are available (which you can calculate based on some assumptions on the component failure model). We will refer to this as whole-file replication.

It has been known for sometime that erasure coding can be used to achieve significantly higher availability [5]. In this case, the file is divided into **b** (equal size) blocks. Erasure coding is then applied to the **b** blocks[1], producing **c** > **b** blocks (of same size as before). We can then recover the original file from any **b** out of the **c** encoded blocks. The storage overhead in this case is **c**/**b**. The file availability can again be computed based on a suitable model for component reliability [6], [7], [8]. The analyses show that file availability can be significantly higher. We refer to this as erasure code replication, or sometimes called block replication.

In this paper, we report some additional analyses on erasure code replication. First, we note that erasure code replication is not always preferable to whole-file replication. This situation occurs when the component availability is low relative to some threshold (determined by the storage overhead). This result is relevant, particularly for some peer-to-peer systems where the average peer availability may be low. Secondly, we note that once the threshold is crossed so that we prefer erasure code replication, the optimal way is to do erasure code replication using as many blocks as possible. In other words, there is a very sharp transition from preferring to whole-file replication to preferring to replicate with many blocks. This sharp transition is characterized analytically, using asymptotic analysis. Lastly, we discuss how to decide whether to use whole-file or erasure code replication in practice, and if erasure code replication, how to decide the number of blocks (**b**) to use. We argue that there is always some cost associated with using erasure code replication, and this cost increases more than linearly with **b**. At some point, this cost becomes overwhelming in comparison to the gain in availability. So erasure code replication with large **b** is unlikely to be profitable. Furthermore, if the peer availability is not accurately known and can be below certain threshold, then the expected gain in file availability may completely disappear.

The paper is organized as follows. In section 2, we review the availability analysis of whole-file replication versus erasure code replication. In section 3, we discuss how the effectiveness of erasure code replication varies with different parameters and under what situations whole-file replication is preferable. We discuss the cost of erasure code replication and how to choose the right number (of blocks) in section 4. Finally we conclude the paper and discuss future works in section 5.

[1] For the purpose of this discussion, we do not need to know exactly how erasure coding is done and why it works. Many papers describe the details, e.g. [5].

2 Review Availability Analysis

When a file is replicated by either whole-file replication or erasure code replication, we create replicas of the original data and place them on different components (peers). In a RAID system [5] or a computer cluster, the availability of each component is modeled by its reliability and the time to repair and replace faulty components. For a peer-to-peer system, the availability of a peer depends on how often the peer is on-line versus off-line. In either case, we will characterize the availability of a peer by a simple parameter, **μ,** known as peer availability. For convenience and tractability, we are assuming all peers have the same peer availabilities; their availabilities are independent of each other; and the single parameter peer availability incorporates possibly multiple sub-components, such as storage and communication components.

Another key parameter is the storage overhead **S**, sometimes referred as the stretch factor. For whole-file replication, this is simply the number of copies. For erasure code replication, this is the ratio of number of erasure-coded blocks to the original number of blocks, **c**/**b**.

TABLE I: Parameters used in erasure code replication

Parameter	Description
μ	Peer availability
A, A_b, A_w	File availability
b	Number of blocks a file is divided into
S	Storage overhead or stretch factor
c = S*b	Number of blocks after erasure coding

2.1 File availability using whole-file replication

Assume that a file is replicated **S** copies and placed at **S** peers. Since the peers are independent to each other, and any 1 out of the **S** peers is enough to recover the whole file, the resulting file availability $\mathbf{A_w}$ is:

$$A_w(S) = \binom{S}{1}\mu^1(1-\mu)^{S-1} + \binom{S}{2}\mu^2(1-\mu)^{S-2} + \ldots + \binom{S}{S}\mu^S(1-\mu)^{S-S}$$

$$A_w(S) = \sum_{i=1}^{S}\binom{S}{i}\mu^i(1-\mu)^{S-i} \qquad (1)$$

2.2 Erasure code replication

If a file is divided into **b** blocks, we need to have **b** blocks to completely recover the whole file. In erasure code replication with storage overhead of **S**, each file block is replicated **S** times. Therefore we have **S*b** number of blocks in the system.

Erasure code makes use of the dependencies between the file blocks to enhance the availability. These **S*b** blocks are dependent of each other, and we need any **b** out of these **S*b** blocks to recover the original file. Therefore, the availability of a file $\mathbf{A_b}$ using erasure code replication is [7]:

$$A_b(b) = \sum_{i=b}^{Sb}\binom{Sb}{i}\mu^i(1-\mu)^{Sb-i} \qquad (2)$$

Notice that when **b = 1**, $\mathbf{A_b = A_w}$, i.e. whole-file replication. Therefore unless otherwise specified, we denote file availability as simply by **A**. Moreover, we assume that the number of peers in the system is large compared with the number of erasure-coded blocks **S*b**. With this assumption, each block is allocated to one peer and therefore each block is independent to each other.

3 Properties of erasure code replication

Based on Equations 1 and 2, it is straightforward to compare whole-file replication and erasure code replication with the same storage overhead. For example, plugging **S** = 2, **μ** = 0.8 in equation 1 gives **A** = 0.96. Using equation 2 with **b** = 2 gives **A** = 0.9728. Therefore erasure code replication performs better.

From equation 2, we see that erasure code replication benefits (in comparison to whole-file replication) from the *combinatorial effects*. For the same storage cost, whole-file replication requires 1 out of **S** peers while erasure code requires **b** out of **S*b** peers. By examining the corresponding combinatorial term for the two cases, we see $_{Sb}C_b$ is much larger than $_SC_1$ as **b** increases. In other words, it is easier to have **b** of **S*b** peers available than **1** out of **S** peers.

However, again from equation 2, we see another term, $\mu^b(1-\mu)^{Sb-b}$, that works against erasure code replication, because it multiplies together a larger number of quantities smaller than 1. The smaller the value of peer availability, the more erasure code replication is penalized. We call this the *peer availability effect*. Therefore, the benefit of erasure code, to a large extent, depends on which of the above two effects is more dominant – the *combinatorial effects*, or the *peer availabilities effect*.

Figure 1 shows a plot of two factors, the combinatorial factor $_{Sb}C_b$ and the peer availability factor $\mu^b(1-\mu)^{Sb-b}$. From the plot, we see that the two factors are running in opposite directions, and therefore the resultant which is the product of the two, $_{Sb}C_b\mu^b(1-\mu)^{Sb-b}$, depends on which factor is more dominant.

In particular, when peer availability is low, it seems that factor can be so dominant that erasure code replication would loose out to whole-file replication. Even though erasure code replication involves more summation terms, we expect that there are cases that erasure code replication perform worse than whole-file replication.

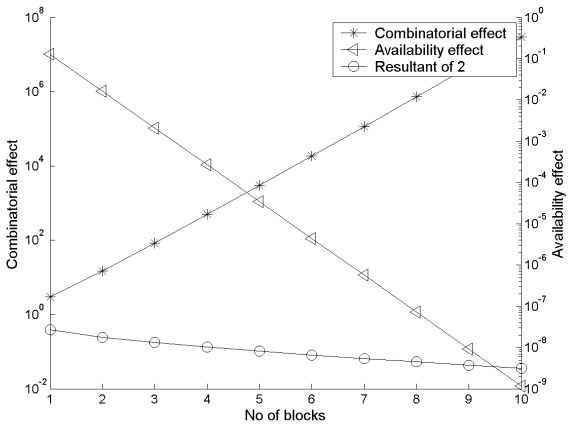

Figure 1: A qualitative analysis of erasure code replication.

3.1 Effect of peer availability

Figure 2 is a plot of file availability **A** against the number of blocks **b** in the system, with different peer availabilities using equation 2. The storage overhead **S** is 2 for all cases.

As notices in section 2.2, the replication strategy is whole-file replication when **b** = 1, and erasure code replication when **b** ≠ 1. From the result we see that when the peer availabilities are low (about 0.2 – 0.5), indeed, whole-file replication can be better than erasure code replication. This supports our earlier observation when we considered the two factors that contribute to the value of file availability. In fact, the advantage of erasure code becomes more apparent only when the peer availabilities are reasonably high (greater than 0.6). At these levels, the overall file availability approach to 1 as **b** increases.

From Fig 2, we also note that **A** may not be always monotonic in **b**. For example, when peer availability is 0.6, file availability (**A)** first decreases and then increases again as **b** increases. This implies that even erasure code replication beats whole-file replication, for certain values of **b** this may not be true. However, as **b** increase, file availability (**A**) seems to become monotonically increasing eventually[2].

[2] We will later prove this to be true.

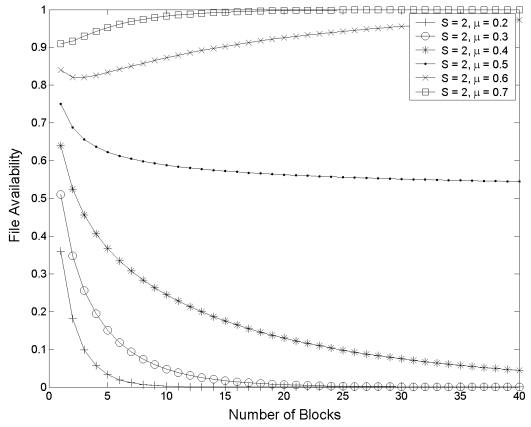

Figure 2: Effect of changing μ on A

3.2 Effect of storage overhead

The storage overhead **S**, which represents the storage constraint of the system, also affects the overall file availability.

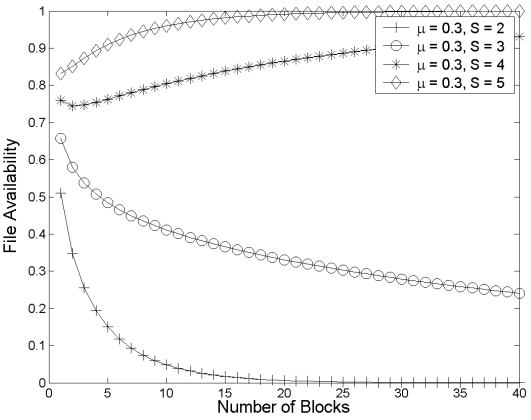

Figure 3: Effect of changing S on A

Similar to the result in section 3.1, the performance of erasure code replication depends on the value of **b**. For example, in Fig 3, when the storage overhead **S** is 4, erasure code replication performs worse than whole file replication when **b** is small, although the situation reverses when **b** increases.

Based on the discussion so far, an interesting question is — what are the optimal values of **b** for different system parameters?

3.3 Optimal value of b

From Fig 2 and 3, we observe that **A** is either monotonically increasing or monotonically decreasing for large values of **b**. This leads us to postulate that the optimal value of **b** (when optimizing file availability **A**) is

either 1 or infinity (or **b** as large as possible to exhaust all the peers in the system). The optimal **b** would equal to 1 when peer availability is small relative to the storage overhead **S**; it would equal to infinity if peer availability is large relative to the storage overhead.

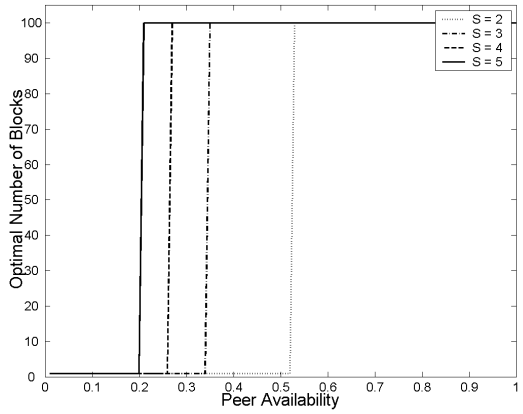

Figure 4: Optimal value of b to achieve highest file availability

Fig 4 plots the optimal value of **b** against **μ** to achieve highest file availability with different values of storage overhead **S**. We fixed the maximum value of **b** to be 100 in this plot.

From Fig 4, we observe that there is a sharp *threshold* **μ′** for each storage overhead **S**. When **μ** is greater than **μ′**, we use erasure code replication with maximum number of blocks (**b**=100) allowed. When **μ** is smaller than **μ′**, we use whole-file replication (**b**=1). For example, when **S** is equal to 2, **μ′** is about 0.5. When **S** increases, this *threshold* becomes a smaller value.

3.4 Analytical derivation

We can compute this threshold by brute force. That is, for each value of **S**, we try different values of **μ** to see at what value of **μ′** the optimal **b** transitions from 1 to 100 (in our example). Figure 5 plots the threshold **μ′** for different values of **S**. The maximum number of blocks **b** for the file is 100. When **S** = 1 (no replication), we find that we should always use whole-file replication for all peer availability levels (notice that **μ′** =1). When **S** increases, **μ′** decreases, and it is more likely to prefer erasure code replication. In general, for values of (**μ**, **S**) in the region above the curve, erasure code replication is preferred; while for values of (**μ**, **S**) below the curve whole-file replication is preferred.

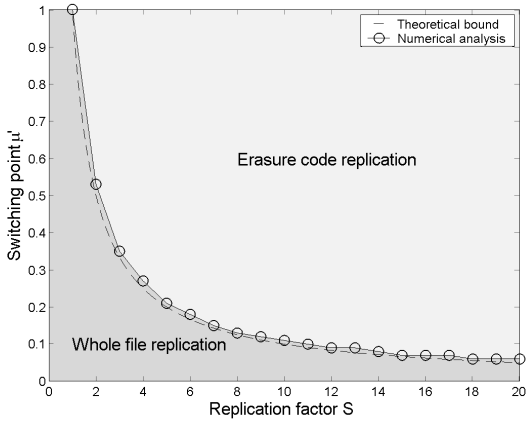

Figure 5: Switching point μ' for different values of S.

In fact, we can analytically derive this dividing curve between where whole-file replication is preferred and erasure code replication is preferred. Indeed, it is the function:

$$\mu > \frac{1}{S} \qquad (3)$$

In [9], the authors proved an asymptotic result in a related problem. They considered the use of erasure codes for maximizing the reliable transmission of data across a large number of (lossy) communication channels in parallel. They showed the following (rephrased using our notations) by using Chebysbev's inequality:

***Proposition*:** Assume **μ** is the probability of packet loss, **b** is the number of blocks and **S** is the storage overhead of using erasure encoding. If **μ** > 1/**S**, then the probability of successful transmission tends to 1 as **b** tends to infinity.

We reproduce the proof in the Appendix for both the asymptotic result when **μ** > 1/**S** as well as the inverse result when **μ** < 1/**S**. In our case, "successful transmission" is replaced by "successful file retrieval"

Note, this is an asymptotic result that states what happens when **b** is large. When **b** is small, the file availability curve may not be monotonic, as shown in Fig 2 and 3. Figure 5 shows the asymptotic theoretical bound, which is very close to the numerical analysis.

This proposition actually points out the power of erasure coding. Namely, if we use enough redundancy so that the expected amount of retrievable data is no smaller than the size of the original data (**S*μ**>1), then we can achieve close to perfect availability by using a large number of blocks, **b**. When **S*μ**=1, we can only achieve file availability **A**=0.5, asymptotically for large **b** [9]. When **S*μ**<1, erasure coding becomes counter-productive for large **b**, since it asymptotically leads to zero file availability.

3.5 Discussion

First, we comment on the relevance of our result. Normally, we build systems with high availability components. This would certainly be the case for RAID and computing clusters. In a decentralized peer-to-peer system, however, it is not entirely unrealistic for the average peer availability to be low. We give some plausible numerical examples below.

RAID: Assume that the backup hard disks are independent to each other, each with a servicing availability of 0.7. By formula 1, using whole-file replication with storage overhead of 3 yields an availability of 0.9730. By formula 2, using erasure code replication with the same storage overhead and **b** = 5 yields a file availability of 0.9993. Thus for a highly reliable system, erasure code helps improve system availability.

P2P: Assume that each peer is only occasionally online, and each has availability of 0.1. By formula 1, if we have 10 peers to do whole-file replication, then file availability is 0.6513. Using erasure code replication with **b** = 10 with the same storage overhead yields 100 blocks in the system. This yields a file availability of 0.5487. In this case, whole-file replication is better.

Secondly, we observe that the result in this section is telling us something rather unsatisfactory. Since there is a sharp transition from preferring to whole-file replication to preferring to replicate with many blocks, the decision for using erasure code replication system is sensitive to system parameters. In practice, however, various cost factors would cause us to consider smaller values of **b** for erasure code replication, or whole file replication, as we argue in the next section.

4 Additional considerations in choosing erasure code replication

4.1 Cost of erasure code replication

Systems gain from erasure code replication because the combinatorial effect. From section 3, we see erasure code replication will achieve near 100% file availability when the number of blocks **b** is large enough. However, after dividing a file into blocks, cost is involved in reassembly the file. Moreover, if we are downloading real time video data, this reassembly may require real-time scheduling of multiple incoming streams of data. Authors in [10] discuss real time decoding cost when using erasure code. It is therefore natural to associate a cost function that is monotonically increasing with the number of blocks **b**.

Let us define a function **C(b)** as the cost function for the overhead of using erasure code replication. We assume the difficulty of scheduling the re-assembly increases more than linearly with the number of blocks **b**. When **b** = 1, the replication scheme is whole-file replication, and the cost is minimal. Based on these assumptions, a simple cost function for **C(b)** is:

$$C(b) \propto (b-1)^2 = \alpha(b-1)^2 \qquad \text{for some } \alpha$$

Given the cost function, we have two considerations when selecting a value for **b**. The problem is how to maximize the first objective function – file availability and minimize the second objective function – the cost function.

Figure 6 is the tradeoff curve for file availability **A** with the above cost function **C(b)** letting $\alpha=10^{-2}$. The number of blocks **b** is bounded to a maximum value of 100, with storage overhead **S** = 3. From section 3, we know that erasure code replication is preferred when **S*μ**>1. We plot the curves with different values of **S*μ** satisfying this criterion.

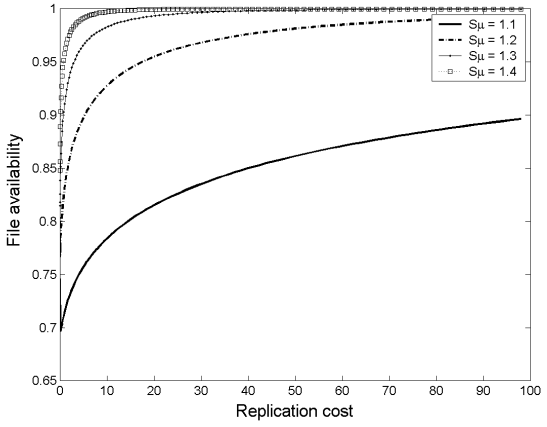

Figure 6: Tradeoff curve for file availability versus erasure code replication cost

The tradeoff curve defines the *pareto optimal* points of file availability with erasure code replication cost. At these pareto optimal points, the system cannot achieve higher file availability without increasing the erasure code replication cost. From the figure, we observe that as we increase the value of **b**, the incremental improvement in file availability decreases while the incremental increase in cost accelerates. For example, when **S*μ** = 1.2, file availability **A** grows faster than the replication cost **C(b)** when the file availability is less than 0.95. When the file availability **A** exceeds this level, the gain in file availability cannot follow the increase in replication cost. This phenomenon is more apparent when **S*μ** is larger.

This means that when it is profitable to do erasure code replication, it is impractical to achieve maximum possible availability gain due to the associated costs.

4.2 Sensitivity to the system parameters

The average peer availability **μ** and storage overhead **S**, may be hard to measure accurately[3]. How sensitive is the selection for **b** to variations of these system parameters?

If it is virtually certain that **S*μ** > **1**, then the choice of **b** can be based on the tradeoff between file availability and cost as discussed in the last subsection. Inaccurate estimation of the parameters **μ** and **S** would result in slightly different tradeoff points between these two metrics (all for **S*μ** > **1)**, which would not be a problem.

On the other hand, if **S*μ** could either be greater than 1 or smaller than 1 due to small variations of **S** and **μ**, then the choice of **b** can become very sensitive to where the value of **S*μ** falls. If we selected a large value for **b**, trying to maximize file availability without knowing **S*μ** is actually less than 1, this could be quite counter-productive. Imagine a system running with peer availability **μ** = 0.35 and storage overhead **S** = 3. Since **S*μ** > 1, we should use erasure code replication with as many blocks as possible (as the cost function allows).

Now suppose **μ** can only be measured with ±10% accuracy, then **μ** can be anywhere in a range [μ_U, μ_L] with μ_U = 0.385 and μ_L = 0.315. Plugging μ_U, μ_L into equation 2, we have the corresponding file availability curves, as shown in Fig 7. From the figure, we find that the difference between two curves (**Δ**), increases with **b**. Furthermore, for most plausible distribution of **μ,** the expected value of file availability would decrease with **b**, starting from **b**=1! This line of argument suggests that even if **S*μ >** 1 (for expected values of **S** and **μ**), the *right* decision may still be to select **b**=1 (whole-file replication) because this choice is more robust against measurement errors. This may be a plausible explanation for why erasure code replication has rarely been adopted by peer-to-peer systems [11], [12] (which tend to have lower and unknown peer availability value than a computer or storage cluster).

[3] This is especially the case in a peer-to-peer system where such parameters may be the aggregate of highly decentralized individual decisions by peers.

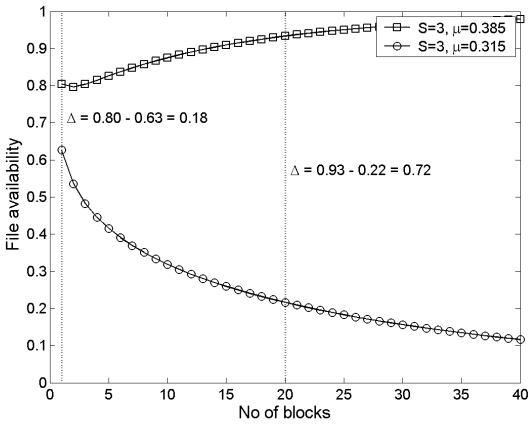

Figure 7: Difference in file availability due to measurement uncertainty

5 Conclusion and future works

In this paper, we revisit the analysis for erasure code replication under different scenarios. Two key parameters that differentiate these different scenarios are: the peer availability level **μ** and the storage overhead **S**. Existing research all implicitly or explicitly assume that the system has high availability level, and therefore the use of erasure code is automatic. However, in this paper we showed that the benefit of erasure code replication actually depends on the peer availability level (relative to the storage overhead). If the peer availability level is low, whole-file replication might perform better and have less cost.

When erasure code replication is used, we also discuss the problem of selecting the optimal **b.** We point out that while theoretically higher values of **b** achieve higher availability, in practice smaller values of **b** are chosen due to re-assembly and scheduling costs. When systems parameters (**μ** and **S**) cannot be accurately determined, the conservative choice of using whole file replication is often the right decision.

There are several interesting issues left for further studies. The analysis in this paper assumed that all the peers have the same availability level **μ**. The study of replication strategies and availability analyses when peers have different availability levels is an interesting direction. As pointed out by many others, the peer availability may be correlated to each other, or to time of day [13]. This is another direction for further studies. From a practical point of view, there are many system level issues in building a peer-to-peer system, in particular how to deal with continuous joining and departure of peers, how to estimate peer availability, choose peers to do replication, and locate replicas, and are there incentives for peers to cooperate to achieve common system level goals. These are all interesting issues for further study.

Acknowledgements:
We thank Professor Soung Liew for an interesting discussion that led us to his work on a related by different problem. We also acknowledge the support from the Areas of Excellence scheme established under the University Grant Committee of the Hong Kong Special Administrative Region, China (Project Number AoE/E-01/99).

Appendix – Proof of asymptotic theoretical bound

Case I: **μ** > 1/**S**

From [9], define the loss probability of the file $\mathbf{L_b}$ as:

$$L_b = \sum_{i=0}^{b-1} \binom{Sb}{i} \mu^i (1-\mu)^{Sb-i}$$

where,

$$A_b + L_b = 1$$

Let X be a binomial random variable having mean $\mu' = \mathbf{Sb\mu}$ and variance $\sigma^2 = \mathbf{Sb\mu(1-\mu)}$. Then $\mathbf{L_b}$ is the sum probabilities of the random variable X with values *0* to **b-1**. Similarly, $\mathbf{A_b}$ is the sum of probabilities of random variable X with values **b** to **Sb**.

$$\begin{aligned} L_b &= \sum_{i=0}^{b-1} \binom{Sb}{i} \mu^i (1-\mu)^{Sb-i} \\ &= P(X=0) + P(X=1) + \ldots + P(X=b-1) \\ &= P(X \le b) \end{aligned}$$

Since :

$$\mathrm{P}(\mathrm{X} \le b) = P(X \le \mu' - (\mu' - b))$$

If $\mu' - b > 0 \Rightarrow Sb\mu - b > 0 \Rightarrow \mu > \frac{1}{S}$

By Chebysbev Inequality,

$$\begin{aligned} P(X \le \mu' - (\mu' - b)) &\le \frac{\sigma^2}{\sigma^2 + (\mu' - b)^2} \\ &= \frac{Sb\mu(1-\mu)}{Sb\mu(1-\mu) + (Sb\mu - b)^2} \\ &= \frac{\mu(1-\mu)}{\mu(1-\mu) + \frac{1}{Sb}(Sb\mu - b)^2} \\ &= \frac{\mu(1-\mu)}{\mu(1-\mu) + Sb(\mu - \frac{1}{S})^2} \\ &\to 0 \quad \text{as } b \to \infty \end{aligned}$$

Therefore, if $\mu > \frac{1}{S}$

$A_b \to 1$ as $b \to \infty$

Case II: **μ** < 1/**S**

From I, $\mathbf{A_b}$ converge to 1 as **μ** > 1/**S**. We are going to prove $\mathbf{A_b}$ converge to 0 as **μ** < 1/**S**.

Similarly,

$$\begin{aligned} A_b &= \sum_{i=b}^{Sb} \binom{Sb}{i} \mu^i (1-\mu)^{Sb-i} \\ &= P(X=b) + P(X=b+1) + \ldots + P(X=Sb) \\ &= P(X \ge b) \end{aligned}$$

Since :

$$\mathrm{P}(\mathrm{X} \ge b) = P(X \ge \mu' + (b - \mu'))$$

If $b - \mu' > 0 \Rightarrow b - Sb\mu > 0 \Rightarrow \frac{1}{S} > \mu$

By Chebysbev Inequality,

$$\begin{aligned} P(X \ge \mu' + (b - \mu')) &\le \frac{\sigma^2}{\sigma^2 + (b - \mu')^2} \\ &= \frac{Sb\mu(1-\mu)}{Sb\mu(1-\mu) + (b - Sb\mu)^2} \\ &= \frac{\mu(1-\mu)}{\mu(1-\mu) + \frac{1}{Sb}(b - Sb\mu)^2} \\ &= \frac{\mu(1-\mu)}{\mu(1-\mu) + Sb(\frac{1}{S} - \mu)^2} \\ &\to 0 \quad \text{as } b \to \infty \end{aligned}$$

Therefore, if $\mu < \frac{1}{S}$

$A_b \to 0$ as $b \to \infty$

References

[1] Bolosky et al "Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs", *Proceedings of Sigmetrics, 2000.*

[2] Jack Y. B. Lee and W. T. Leng, "Study of a Server-less Architecture for Video-on-Demand Applications," *In Proccedings of IEEE International Conference on Multimedia and Expo., Lausanne, Switzerland, 26-29 August 2002, pp.233-236*

[3] V. N. Padmanabhan, H. J. Wang, and P. A. Chou "Resilient Peer-to-Peer Streaming", *IEEE ICNP 2003*

[4] Michael O. Rabin. "Efficient dispersal of information for security, load balancing and fault tolerance". *Journal of the Association for Computing Machinery, 36(2):335--348, April 1989*

[5] Patterson, D., Gibson, G., and Katz, R. “The Case for RAID: Redundant Arrays of Inexpensive Disks”, *ACM SIGMOD, May 1988*

[6] H. Weatherspoon and J. Kubiatowicz. “Erasure coding vs. replication: A quantitative comparison”. *In Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002).*

[7] R. Bhagwan, D. Moore, S. Savage, and G. M. Voelker. “Replication strategies for highly available peer-to-peer storage”. *In Proceedings of FuDiCo: Future directions in Distributed Computing, June 2002*

[8] Charles Blake, Rodrigo Rodrigues, "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two", *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS '03), Lihue (Kauai), Hawaii, May 2003*

[9] Tony T. Lee, Soung C. Liew, “Parallel Communications for ATM Network Control and Management”, *IEEE Globecom 1993*

[10] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", *Computer Communication Review, 27(2):24--36, April 1997*

[11] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System., *In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000*

[12] V. N. Padmanabhan and K. Sripanidkulchai, "The Case for Cooperative Networking", *In Proceedings of the first International Workshop on Peer-to-Peer Systems (IPTPS 02), March 2002*

[13] Ranjita Bhagwan, Stefan Savage, Geoffrey M. Voelker: “Understanding Availability”, *In Proceedings of the second International Workshop on Peer-to-Peer Systems (IPTPS 03) , Feb 2003*