Cost and Performance Comparison of Replication and Erasure Coding

Hitachi white paper 2014

Sifat Ferdousi

January 27, 2017

UCDAVIS

Introduction

- To protect data from device or system failures, storage systems employ mechanisms for data protection. Storing redundant information on distributed servers can increase reliability for storage systems since users can retrieve duplicated pieces of data in case of disk, node, or site failures.

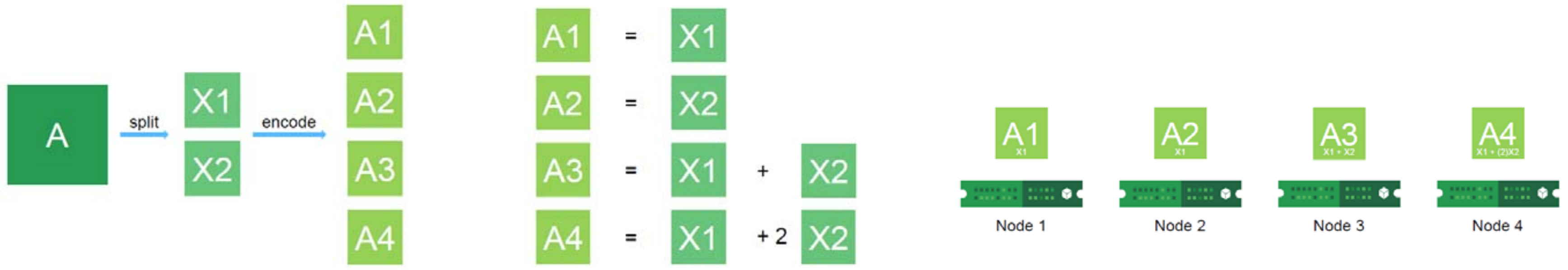
- Data protection results in consumption of additional storage (s*torage overhead)* and a *storage efficient* protection mechanism uses less overhead.

- Two protection mechanism variants: *replication* and *erasure coding* (referred to as Reed Solomon coding).

UCDAVIS
UNIVERSITY OF CALIFORNIA

Replication vs. Erasure coding

- Replication is a process where a whole object is replicated across different locations, thus providing protection if a copy of an object is lost or unavailable.
- Erasure coding is a parity based protection technique. Data/objects are broken into fragments and encoded.



Comparison and trade-off

- Erasure coding has been widely studied for distributed storage systems and used by companies like Facebook and Google since it provides space-optimal data redundancy (storage efficiency) to protect against data loss. There are, however, critical factors that affects performance, and issue of data availability.

- EC is advantageous for “cold” data, data that is unlikely to be accessed or for which read performance is not an issue.
- Replication is superior for “hot” data, content that is accessed frequently or for which performance is important.
-
- In particular, the more emphasis one places on availability and read performance, the greater the advantage of replication; the more emphasis one places on storage efficiency, the greater the advantage of erasure coding.

Data availability

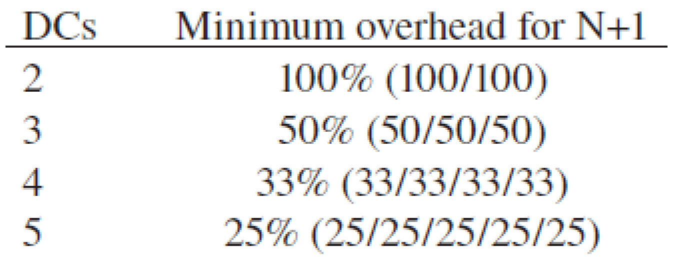
- Enterprise-class data storage systems are designed to protect against data loss (DL) and data unavailability (DU).
  - Data loss refers to permanent loss, data that cannot be retrieved by any means.
  - Data unavailability refers to data that is temporarily unavailable but that can be retrieved after some undetermined delay.
- For data to be *highly available* in a multi-datacenter environment means that the data must be distributed amongst the various data centers (DCs) in such a way that all data can still be read in the event that any one DC is unavailable.

Data availability

- Data availability calculations are sometimes incorrectly conflated with data loss calculations. While the two can be identical for a single site, they diverge for multisite.

  - Assume we have two DCs and want our data to be highly available, what is the *minimum* amount of overhead required to protect against DU? - 100% (when one of the data centers is unavailable, all of the data must be available from the only data center that is reachable). - Same to protect against DL.

- The table below illustrates how data must be spread across DCs to provide full data availability (full data access in the event of the failure of any one DC).



DCs	Minimum overhead for N+1
2	100% (100/100)
3	50% (50/50/50)
4	33% (33/33/33/33)
5	25% (25/25/25/25/25)

Data protection

- Data loss can occur when a disk fails. Let p_L be the probability that a single disk is permanently unreadable. An estimate of p_L depends on the mean time between failures as well as the time required to replace the disk.
  - For example, suppose a disk performs reliably on average for 1,000 days, roughly three years. If a hot spare is available to replace the disk upon failure and it takes one day to format the disk and copy data to it, $p_L = 0.001$. If there is not a hot spare and it takes a couple days to order and install a replacement disk and a day to fill it, $p_L = 0.003$.

- Similarly let p_U be the probability that a disk is unavailable. A disk may be unavailable, yet still alive (temporarily disconnected from the network). Because disks can be temporarily unavailable without failing, say due to a network outage, p_L is always smaller than p_U.

Data protection in Replication

- In a replicated system with k replicas of each object, the probability of data loss is $p^k{}_L$, *assuming disk failures are independent*.

- Given a tolerable probability of data loss ε, we can solve for the number of disks k needed to keep the probability of loss below this level:

$$k = \frac{\log \varepsilon}{\log p_L}$$

- The same calculation applies to data unavailability if we replace p_L with p_U.

UCDAVIS
UNIVERSITY OF CALIFORNIA

Data protection in Erasure Coding

- In an *m* + *n* erasure-encoded system, each object is divided into *m* equal-sized fragments. In addition, *n* parity fragments are created, each the size of one of the data fragments. The data can be read if any *m* out of the *m* + *n* fragments are available. Data will be lost if more than *n* fragments are simultaneously lost.

- Since we need *m* out of *m* + *n* disks to reconstruct an object, the probability of data loss is the probability of more than *n* disks being falied simultaneously:

$$\sum_{i=n+1}^{m+n} \binom{m+n}{i} p_L^i (1-p_L)^{m+n-i}$$

- Given a number of data fragments *m* and an acceptable probability of unavailability ε, we can solve for the smallest value of *n* such that

$$\sum_{i=n+1}^{m+n} \binom{m+n}{i} p_L^i (1-p_L)^{m+n-i} < \varepsilon$$

UCDAVIS
UNIVERSITY OF CALIFORNIA

Data protection in Erasure Coding

- For example, suppose you would like to build a system with probability of data recovery 0.999999 (six nines) using disks that have probability 0.995 of being alive. Triple replication would have a probability of DL equal to $0.005^3 = 1.25 \times 10^{-7}$.

- Suppose you want to use erasure coding with 8 data disks. An $8 + n$ system would require $n = 3$ to keep the probability of DL below 10^{-6}. In fact, an 8 + 3 system has a probability of DL 1.99×10^{-7}.

- A 1-Gbyte video stored in the triple replicated system would require 3 Gbyte of storage. In an 8+3, the same object would be stored in 8 data fragments and 3 parity fragments, each 125 Mbyte in size, for a total of 1.375 Gbyte. In short, the erasure-coded system would use about half as much disk space and offer the same level of data protection.

Data protection in Erasure Coding

Choosing the number of data disks

- Given a value of *m* and the individual disk reliability, we can choose *n* to achieve the desired level of protection. But how do you choose *m*?
- $\boldsymbol{k_c = (m + n)/m}$ - redundancy factor for $m + n$ erasure coding, analogous to *k* for replication (Replication is a special case of erasure coding with $m = 1$).
- Increasing *n* with *m* fixed increases reliability. Increasing *m* with *n* fixed decreases reliability. But in a sense, we gain more reliability by increasing *n* than we lose by increasing *m*. We can increase reliability by increasing *m* and *n* proportionately, keeping the redundancy factor k_c constant.
  - For example, a 4 + 2 system will be more reliable than a 2 + 1 system even though both have the same redundancy k_c =1.5. So why not make *m* and *n* larger and larger, obtaining more and more reliability for free?
- Of course the increase in m and n is not free. They potentially increase latency and reconstruction costs.

UCDAVIS
UNIVERSITY OF CALIFORNIA

Data protection in Erasure Coding

- Increasing *m* and *n* also increases the total number of data fragments *m* + *n* to manage. In practice, erasure-encoded systems use values of *m* on the order of 10, not on the order of 100. For example, 6 + 3 systems or 12 + 4 systems, but not 100 + 50 systems.

- Aside from the memory required to keep an inventory of data fragments, there is also the time required to find and assemble the fragments (depends greatly on how EC is implemented). Such overhead associated with EC is not required with replication.

- This explains why an EC system can be slower than a replicated system, even when all fragments are in the same data center. Finally, we note that the more data fragments there are to manage, the more work that is required to rebuild the fragment inventory database when failures occur.

Disk allocation to DCs

- How disks are allocated to data centers impacts EC more than replication, and DU more than DL.
- It is far more likely that an entire data center would be unavailable than that an entire data center would be destroyed. Data centers are occasionally inaccessible due to network outages. This causes all disks inside to be simultaneously unavailable. However, it is very unlikely that all disks in a data center would fail simultaneously. This means that DU is more correlated than DL.
- In replicated systems, it is common for one replica of each object to reside in each data center. If we assume network failures to data centers are independent, then the same probability calculations apply to data loss and data unavailability. If there are *d* data centers, the probabilities of an object being lost or unavailable are p^d_L and p^d_U respectively.

Disk allocation to DCs

- However, in EC systems, the number of fragments per object is typically larger than the number of data centers. A mid-sized company often has two or three data centers (at most 4). And yet EC systems using a scheme such as 8+4 are not uncommon. With fewer than 12 data centers, some of these fragments would have to be co-located.

- If every fragment in an EC system were stored in a separate data center, the unavailability calculations would be analogous to the data loss calculations, as they are for replicated systems. But because data fragments are inevitably co-located, these fragments have correlated probabilities of being unavailable and so the unavailability probability for the system goes up.

Probability Assumptions

- Reliability calculations, whether for replicated systems or erasure-encoded systems, depend critically on the assumption of independence.
- Disk failures could be correlated for any number of reasons: disks coming from the same manufacturing lot, disks operating in the same physical environment, and so forth.

- The assumption of independence is more accurate for disks in separate data centers. And so, for replicated systems with each replica in a separate data center, independence is a reasonable assumption.
- But for EC systems with multiple fragments in each data center, the assumption of independence is less justified for data loss, and unjustified for data unavailability.

Efficiency Considerations

- In a replicated system, the operating system directly locates objects. In the event of a failure, requests are redirected to another server, but in the usual case objects are accessed directly.
- With erasure coding, fragments of objects must be cataloged. Since the bits necessary to reconstruct an object exist on multiple disks, a system must keep track of where each of these fragments are located.
- When an object is requested from an $m + n$ system, a server looks up the location of at least m fragments. (A server could, randomly choose m fragments or try to determine the m closest fragments.)
- If the requests succeed, these m fragments are transferred to a server to be re-assembled into the requested object. Since assembly cannot begin until the last fragment is available, the process would be slowed down if one of the fragments were coming from a location farther away than the others.

Costs Of Disk Failures

- For a fixed level of reliability, erasure coding requires less disk space than replication. If storage cost were the only consideration, erasure coding would have a clear advantage over replication.

- While erasure coding can lower the probability of DL, it increases the probability of DU.
    - We compared triple replication to 8 + 3 erasure coding. Both systems had roughly the same probability of data loss. Replication used 3 disks per object while erasure coding used 11, and so the erasure-coded system is nearly 4 times as likely to experience a single, recoverable disk failure.

UCDAVIS
UNIVERSITY OF CALIFORNIA

Latency

- If all replicas and all erasure-encoded fragments are in the same data center, latency is not as much of an issue as when replicas and encoded fragments are geographically separated.

- In a replicated system, requests could be routed to the nearest available replica (nearest in terms of latency). If the nearest replica is not available, the request would fail over to the next nearest replica and so on until a replica is available or the request fails.

-
    - Suppose a replicated system maintains k copies of an object, each in a different data center, and that requesting data from these centers has latency $L1 < L2 < \ldots < Lk$ for a given user. Suppose each replica has a probability p of being unavailable. With probability $1 - p$ the latency in the object request will be $L1$. With probability $p(1 - p)$ the latency will be $L2$. The expected latency will be

$$\sum_{i=1}^{k} p^{i-1}(1-p)L_i$$

    - If p is fairly small, the terms involving higher power of p will be negligible and the sum above will be approximately $(1 - p)L_1 + pL_2$

UCDAVIS
UNIVERSITY OF CALIFORNIA

Latency

- In an $m+n$ erasure-encoded system, an object request could be filled by reconstructing the object from the m nearest fragments. Since object requests more often encounter (recoverable) disk failures in erasure-encoded systems, they will more often involve an increase in latency.
  - If latency is a concern, an $m + n$ system would store at least m fragments in a single location so that only local reads would be necessary under normal circumstances. If m fragments are stored in one location, the probability of one local disk failure is mp. This means the probability of a single local failure, and the necessity of transferring data from a more distant data center, is m times greater for an erasure-coded system compared to a replicated system. The expected latency increases from approximately $(1 - p)L_1 + pL_2$ in a replicated system to approximately $(1 - p)L_1 + mpL_2$ in an erasure-encoded system.
- For active data, objects that are accessed frequently, latency is a major concern and the latency advantage of replication should be considered. For inactive data, objects are archived and seldom accessed, latency may be less of a concern.

Reconstruction

- With replication, the content of a failed disk is simply copied, either from within a data center or from a remote data center, depending on how replicas are distributed.
- With $m + n$ erasure coding, the content of m disks must be brought to one location. For 6 + 3 encoding, if three fragments are stored in each of three data centers and one disk fails, the content of four disks must be transferred from a remote data center to the site of the failed disk in order to have enough data for reconstruction.

UCDAVIS
UNIVERSITY OF CALIFORNIA

Local Reconstruction

- *Local reconstruction codes* is a variation on erasure codes to mitigate this problem.
- With this approach, two data centers would each contain three data disks and a local parity disk. A third data center would contain two global parity disks computed from all six data disks. They call this approach 6+2+2. Any single failure could be repaired locally. Since single failures are most likely, this reduces the average reconstruction time. The 6 + 2 + 2 scheme offers a level of data protection intermediate between 6 + 3 and 6 + 4 Reed-Solomon codes. The 6 + 2 + 2 system can recover from any combination of three-disk failures, and from 86% of four-disk failures.
- The cost of reconstructing a disk is lowest with replication, highest with traditional Reed-Solomon erasure coding, and intermediate with local reconstruction codes.
- The time necessary for reconstruction feeds back into the data protection calculations. If failed disks can be reconstructed faster, availability improves, increasing the availability of the system.

Trade-offs

- A system with mostly inactive data, archival objects accessed rarely, and primarily concerned with storage costs, would be better off using erasure coding. Such a system might also want to use a moderately large value of *m* in its *m* + *n* encoding.

- A hybrid approach, used by Microsoft Azure, is to use both replication and erasure coding. Objects are replicated on entering the system. Later they are erasure-encoded and the replicas are deleted. Thus, active data is served by replication and inactive data is served by erasure coding.

- In the Azure implementation, the cost of encoding fragments is not a significant concern. Because objects are replicated immediately, encoding can be done out of band. A system that encodes objects as they enter might be more concerned about encoding costs. One could also use a hybrid approach of using *k* replications of *m*+*n* erasure-encoded fragments. In such a case, *k* and *n* might be very small.

UCDAVIS
UNIVERSITY OF CALIFORNIA

Summary

- Replication use case: primary data storage (high availability and high recoverability)

  - Pros: Less CPU intensive = faster write, simple restores = faster rebuild
  - Cons: Requires 2x or more the original storage space

- Erasure coding use case – Latency tolerant archival storage (long-term high volume data storage in 1.5x disk space). Rebuilds are slower but capacity savings outweigh latency.

  - Pros: Consumes less storage than replication – storage efficient, allows for two or more failures Cons: Parity calculation is CPU-intensive, increased latency – slow production writes and rebuilds

UCDAVIS
UNIVERSITY OF CALIFORNIA