Erasure Coding for production in the EOS Open Storage system

Andreas-Joachim Peters[1,*], *Michal Kamil* Simon[1,**], and *Elvin Alin* Sindrilaru[1,***]

[1]CERN, Esplanade des Particules 1, 1217 Meyrin, Geneva, Switzerland

Abstract. The storage group of CERN IT operates more than 20 individual EOS[1] storage services with a raw data storage volume of more than 340 PB. Storage space is a major cost factor in HEP computing and the planned future LHC Run 3 and 4 increase storage space demands by at least an order of magnitude.
A cost effective storage model providing durability is Erasure Coding (EC) [2]. The decommissioning of CERN's remote computer center (Wigner/Budapest) allows a reconsideration of the currently configured dual-replica strategy where EOS provides one replica in each computer center.
EOS allows one to configure EC on a per file bases and exposes four different redundancy levels with single, dual, triple and fourfold parity to select different quality of service and variable costs.
This paper will highlight tests which have been performed to migrate files on a production instance from dual-replica to various EC profiles. It will discuss performance and operational impact, and highlight various policy scenarios to select the best file layout with respect to IO patterns, file age and file size.
We will conclude with the current status and future optimizations, an evaluation of cost savings and discuss an erasure encoded EOS setup as a possible tape storage replacement.

1 Introduction

The storage group of CERN IT operates more than 20 individual EOS [4] storage services reaching a raw data storage volume of 340 PB in 2020. Storage space is a major cost factor in HEP computing and the planned future LHC Run 3 and 4 increase storage space demands by at least an order of magnitude.

A cost effective storage model providing durability is Erasure Coding (EC). Until the end of 2019 the CERN EOS deployment was using a distributed storage model. Each file was stored with one replica at the CERN and one replica at the Wigner computer center in Budapest. This model allowed batch jobs in both centers to profit from low latency access to local file replicas. Erasure Coding would have been suboptimal in this deployment scenario - low latency access is possible only for the locally stored EC fragments of each data file (see details in section 2.2). The decommissioning of the Wigner computer center allows to

*e-mail: andreas.joachim.peters@cern.ch
**e-mail: michal.simon@cern.ch
***e-mail: elvin.alin.sindrilaru@cern.ch

reconsider the currently configured dual-replica strategy. In the near future all computing nodes will be co-located in the same center close to storage nodes.

EOS allows to configure EC on a per file bases and exposes four different redundancy levels with single, dual, triple and fourfold parity to select different quality of service and variable costs.

In April 2019 first tests have been performed to migrate files on a production instance from dual-replica to various EC profiles to measure performance and operational impact and investigate various policy scenarios to select the best file layout with respect to IO patterns, file age and file size.

2 Erasure Coding

2.1 Implementation

Erasure Coding is a method of data protection in which data is broken into fragments, expanded and encoded with redundant data pieces and stored across a set of different locations of storage media. An example of a widely used algorithm is Reed Solomon forward error correction (RS FEC) [11].

Figure 1 illustrates the application of erasure coding in a storage system.

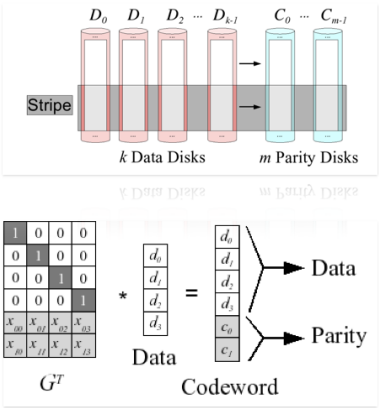

Figure 1. Parity computation in erasure coded storage systems

There are two widely used open source libraries implementing erasure encoding:

- JERASURE [6]
- Intel ISA-L [8]

Using AVX support on modern CPUs both libraries reach on a Xeon CPU (2.2.GHz) encoding performances of over 5 GB/s/core. Another key performance indicator for an EC algorithm is the data loss probability p. One formula to compute the data loss probability is shown in Figure 2. Parameters entering the equation are the mean time to repair (MTTR), the number of disks used and the annual failure rate of a disk (AFR). This results for a typical sized storage system of 2,000 disks, 2% AFR and 24h MTTR in a data loss probability order of

Data loss probability $p = e^{-\lambda}\frac{\lambda^k}{k!}$

where $\lambda = \frac{AFR \times (Number\ of\ Disks)}{365 \times 24 \div MTTR}$

MTTR = Mean Time To Repair
AFR = Annualized Failure Rate

Figure 2. Data loss probability in erasure coded storage systems [9]

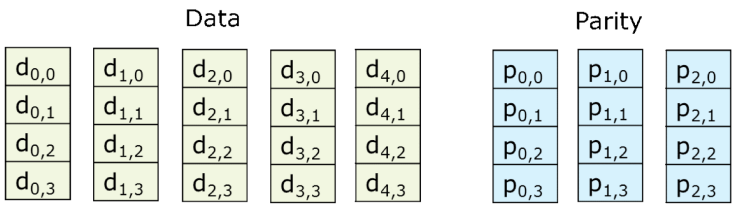

Figure 3. Block layout in erasure coded storage systems

10^{-9} to 10^{-8} - a very low risk. To apply EC in a storage system files are split into equal size blocks. This is illustrated in Figure 3.

An EC Reed-Solomon algorithm is normally described as $RS(m + k, k)$ where k is the number of data and m is the number of parity blocks.

2.2 Usability at CERN

Until the end of 2019 the CERN computer center was split into two sites, one in Meyrin and one in Wigner but operated as a single storage setup with 22 ms latency and 3x100 Gbps connectivity between both sites. This geographical setup favored the usage of a simple replication model, where one replica is stored in each computer center to allow low latency access from batch nodes located in both sites. When using EC in such a setup one can either co-locate all EC fragments of a single file in a single computer center or distribute over both. In either case for some computing nodes some or all storage fragments have to be read over the 22 ms link. By moving all computing and storage nodes to the Meyrin site this problem has been eliminated and allows CERN to use EC. The advantages of EC are:

- configurable data resilience and high availability
- increased IOPS and bandwidth per file
- high cost saving potential of several million CHF in a hundreds of petabytes storage system

In comparison to a dual replica model EC allows to configure the equivalent of manifold replication using less disk space. The IOPS and access bandwidth per file scales with the number of data disks. In a replicated configuration these are given by the performance of a single hard disk. The disadvantages of EC are:

- read amplification during reconstruction for non sequential access
- write amplification for updates when writes are not aligned to EC blocks
- higher CPU usage when reconstruction is required

For pure sequential access there is neither read or write amplification if some data disk is degraded. This is not true anymore for random access patterns. For random and sequential access the missing data pieces have to be reconstructed using additionally CPU resources.

2.3 Native EC implementation in EOS

The native implementation of erasure coding in EOS supports the following parity configurations [5]:

- single parity **RAID5**
- double parity **RAID6**
- triple parity **ARCHIVE**
- fourfold parity **QRAID**

The EC algorithm is implemented using the RS variant of the JERASURE library. EOS supports block sizes of 4 kb, 64 kb, 128 kb, 512 kb, 1 Mb, 4 Mb and 64 Mb. The block size describes the size of a data block before chunking. To identify data corruption in chunks, EOS stores for every 4 kb data stripe a block checksum in a separate memory mapped file. The algorithm is configurable to be one of many supported checksum types[1] of EOS. The natural choice is to use a hardware accelerated crc32c checksum[7]. The IO path of erasure

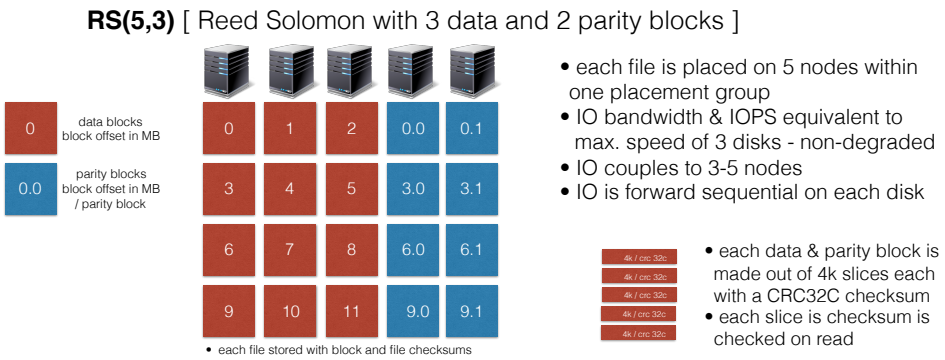

Figure 4. Illustration of the $RS(5,3)$ EC layout

coding is using by default a **gateway model**. In the gateway model a client sends/retrieves

[1] adler32, md5, crc32, crc32c, sha1, sha256, xxhash

all data to/from one storage server (FST[2]). This gateway FST is responsible for erasure en-/decoding of data and fan-out to other storage servers. The advantage of a gateway model is that clients don't require a new plug-in to support erasure coding. File checksums can still be computed in the regular way because the IO of one file flows through a single storage node and computation can be done on the fly. The disadvantage of this model is increased traffic and latency for read and write operations. The read/write amplification for a large number of data disks k asymptotically reaches a factor of two when erasure coding is applied.

In contrast to write operations, a standard read operation can be served using only data chunks. If one of the data chunks is unavailable or unreadable due to a checksum error, the missing data block is reconstructed using parity chunks. An example of an $RS(5,3)$ erasure coding layout is shown Figure 4. EOS provides a specialized file copy command **eoscp** which drops the model of an FST gateway and allows to read all required data chunks directly from storage nodes. In this model there is no read amplification. EOS provides also a XRootD client plug-in[3] with the same IO path, which is not yet in use mainly due to compatibility requirements of plug-in and XRootD versions used in physics application. The IO models are shown in Figure 5.

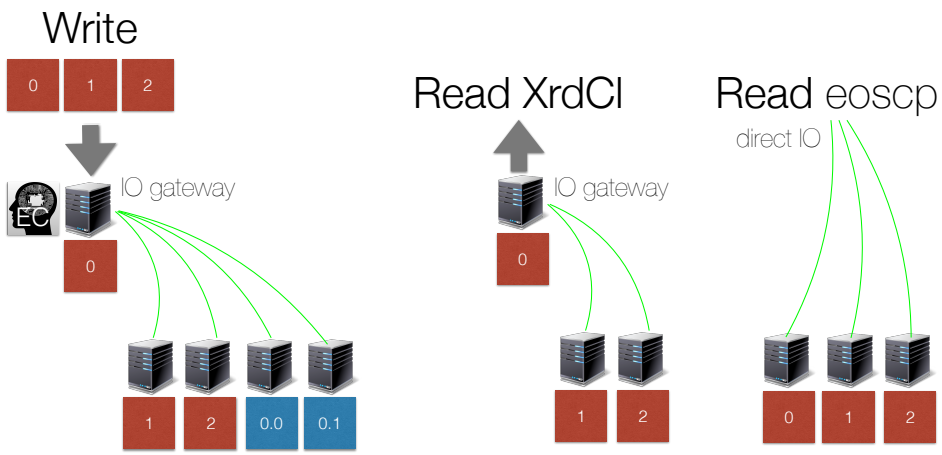

Figure 5. IO path of read and write access for EC file layouts

Figure 6 shows the traffic amplification factor for various EC configurations. A particular case is the traffic created by a drain operation. Draining a disk results in reconstruction of all blocks stored on the draining disk. This amplification factor is proportional to the number of data disks in the EC set.

[2]File Storage Server

amplification factor	RS(2,1)	RS(3,2)	RS(6,4)	RS(9,6)	RS(18,14)
Write gateway	2	1.66	1.75	1.83	1.92
Read gateway	1	1.66	1.75	1.83	1.92
Read direct	1	1	1	1	1
Drain*	x1	x2	x4	x6	x14

Figure 6. Traffic amplification for various EC layouts

2.4 Storage Volume Increase

The crucial factor for cost savings is the storage volume increase due to re-encoding data from two-fold replication to erasure coding layouts. Figure 7 shows the potential storage space increase applying $RS(10,8)$ which results in over 100 PB more physical space without any additional hardware for EOS at CERN.

3 Testing EC conversions

EOS provides an embedded namespace service called **Converter** which allows one to schedule file layout transformations e.g. rewrite $RS(2,1)$ (replicated) files into $RS(10,8)$ using internal third party transfers. The limiting factors are the number of conversions/day and extra IO traffic. As an example $RS(10,8)$ conversions requires 14 meta-data transactions (write-lock+commit). The encoding of 1 GB of data requires 1 GB of disk/network reads and 1.25 GB of disk/network writes. A goal in a conversion exercise is to have maximum impact on space savings with minimal cost. Therefore the target of a conversion exercise are the largest files available. In April 2019 we conducted a test where we converted 2.4 PB of 2018 ALICE raw data from $RS(2,1)$ to $RS(12,10)$ layout during 84 hours. The conversion freed up 2 PB of storage space. The data integrity has been verified using the known MD5 checksums of the data. During the test we fixed a few issues which were triggered under certain error conditions such as timeouts. In particular a bug in error handling led to loss of EC stripes. The CPU requirements for encoding and decoding of data was negligible. The progress of the test is shown in Figure 8, which shows on the top plot the linear increase of free space and the observed total IO rates in the storage system on the bottom plot.

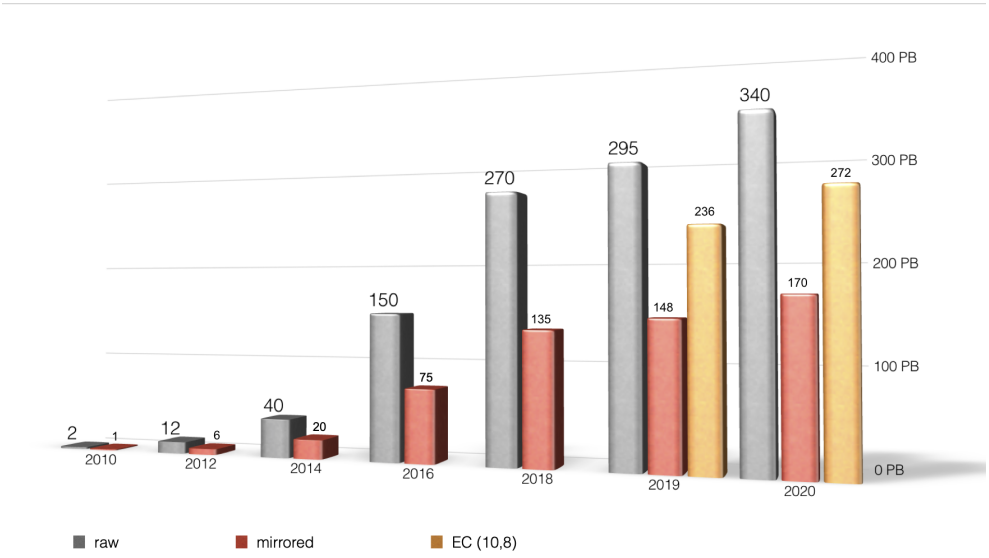

Figure 7. Comparison of available storage space of raw, replicated and EC encoded files. The available space for replicated files is 50% of the raw space. The available space for RS(10,8) encoded files is 80% of the raw space.

3.1 Conversion Policies

EOS supports conversion policies in combination with the **LRU** engine [10]. Policies can be defined via extended attributes on directories and they provide a combination of time, size and name based selections for eligible conversion. As an example the policies used in listing 1 describe the conversion of all files older than a week and bigger than one gigabyte to $RS(6,4)$. The layout $RS(6,4)$ is expressed in an internally used hexadecimal layout format (20640524).

Listing 1. Policy Setting

```
eos attr set sys.lru.convert.match=*:1w:>1G <dir>
eos attr set sys.conversion.*=20640542 <dir>
```

Considering a future configuration, we can define a minimum file size where EC should be applied. For small files it is favorable to use replication because the open/creation time for a file increases with the number of stripes and can dominate the total IO time. Statistics from various EOS instances at CERN suggests to set this limit between 200 MB and 1 GB. Analysis of file popularity suggests that most accesses are happening during the first 3-6 month after creation. Therefore a second criteria for conversion could be a minimum age of 3 month. In this way we guarantee nearly identical performance for present analysis use cases. Further tests are needed to conclude if the time based criteria is really required since an initial creation of files in an EC layout avoids additional conversion traffic.

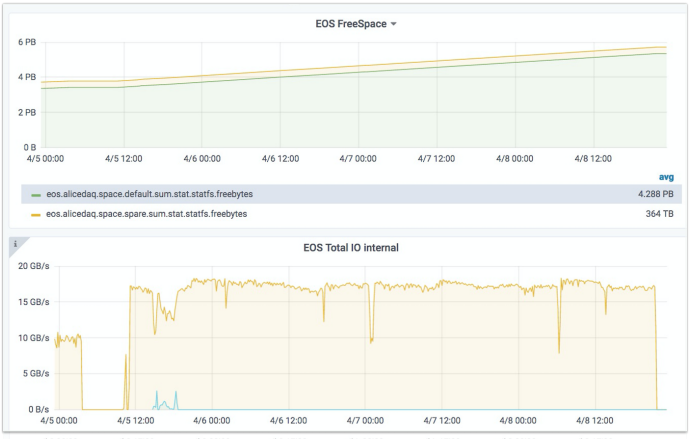

Figure 8. Space and bandwidth observed in an EC conversion test of 2.4 PB of data

4 Conversion Time Extrapolation

The described conversion test was running on a storage setup with 69 10GE connected disk servers mostly with two trays and 24 disks each. Using the observed rates of approx. 7 GB/s reading/writing, one can encode 600 TB per day. If we extrapolate this figure to a 1,000 disk server setup we can expect to encode 8 PB per day using approximately 100 GB/s read bandwidth. In production we observe peak rates on all storage pools of up to 150 GB/s. Since we cannot use a majority of the storage IO capacity for conversion, we assume to use only 10% of the maximum encoding performance which results in the conversion of 800 TB per day using $RS(10,8)$. In one month we can provide 24 PB of additional free space dedicating only 10% of the available IO performance in a 1000 disk server installation to file conversions.

5 EC Evolution - Direct Object Storage

The introduced gateway approach is the best trade-off to serve clients in WAN and LAN environments. For high performance use cases like online storage, the traffic amplification due to erasure encoding represents a strong limitation. To reduce the network requirements to the minimum the erasure coding algorithm can be implemented on client side. In this Direct Object Storage model (DOS), where clients can access fragments of data and parity chunks directly, there is no traffic amplification in non-degraded or degraded read mode. The traffic amplification for writing is only due to the additional parity volume. First tests of a prototype implementation look very promising.

6 Summary

The conducted tests corroborate that EC is beneficial and feasible (after the decommissioning of the Wigner computer center). Assuming an $RS(10,8)$ code, EC allows incremental storage space increase, without additional costs, in the order of 100 PB. Dedicating 10% of the

available IO bandwidth, 0.8 PB can be added per day converting replicated files to $RS(10,8)$ files. The network traffic increase is in the order of two. This should not be problematic since the network at CERN is performance overcommissioned by a factor 4 to 10. Using expected hard drive failure rates, we can have additional (non critical) background traffic of 220 MB/s for disk draining activities in combination with $RS(10,8)$. As mentioned EC implies a read latency and traffic volume amplification which might have a negative impact on very selective analysis use-cases (using sparse reading). These adverse effects have to be quantified in the following months. XCache [12] - a block-based file proxy cache - could be a suitable technology to optimize analysis use cases with erasure encoded back-end storage.

7 Outlook

We expect a slow introduction of erasure coding during 2020 at CERN. We are still considering additional tests, the upgrade of operational procedures and studies of analysis job efficiencies before running a large-scale conversion campaign. In particular in the context of online storage evaluation for ALICE (LHCb) Run-3, all aspects of erasure coding will be tested in detail with 100GE technology. During 2020 an EC storage system at KISTI in South Korea will be put into production with the aim to replace a tape-based storage system. It will consist of nine storage servers providing 13.5 PB of storage using a $RS(16,12)$ layout [13]. This new system will be another milestone to demonstrate $k >> m$ EC codes with the EOS Open Storage System in production.

References

[1] Exabyte Scale Storage at CERN, Andreas J Peters and Lukasz Janyst 2011 J. Phys.: Conf. Ser. 331 052015

[2] Erasure Coding, https://en.wikipedia.org/wiki/Erasure_code

[3] XRootd EOS EC Client Plugin, https://gitlab.cern.ch/dss/eos/-/tree/master/fst/xrdcl_plugins

[4] EOS Open Storage, https://eos.web.cern.ch

[5] EC Layouts, http://eos-docs.web.cern.ch/eos-docs/using/rain.html

[6] Jerasure: Erasure Coding Library, https://jerasure.org

[7] The iSCSI CRC32C Digest and the Simultaneous Multiply and Divide Algorithm, Luben Tuikov, Vicente Cavanna, 2002, Computer Science

[8] Intelligent Storage Acceleration Library, https://github.com/intel/isa-l

[9] Durability and Availability of Erasure-Coded Systems with Concurrent Maintenance, SS. Arslan, http://www.suaybarslan.com/Reliability_Systems_14.pdf

[10] EOS LRU Engine, http://eos-docs.web.cern.ch/eos-docs/configuration/lru.html

[11] Reed, Irving S.; Solomon, Gustave (1960), "Polynomial Codes over Certain Finite Fields", Journal of the Society for Industrial and Applied Mathematics, 8 (2): 300–304

[12] XRootd, disk-based, caching proxy for optimization of data access, data placement and data replication, L. Bauerdick et al, 2014, Journal of Physics: Conference Series

[13] Alternative to tape based archive storage project at KISTI, https://indico.cern.ch/event/862873/contributions/3724465/