Evaluating the Fault Tolerance Performance of HDFS and Ceph

Yehia Arafa
Klipsch School of ECE
New Mexico State University
Las Cruces, NM
yarafa@nmsu.edu

Atanu Barai
Klipsch School of ECE
New Mexico State University
Las Cruces, NM
atanu@nmsu.edu

Mai Zheng
Computer Science Department
New Mexico State University
Las Cruces, NM
zheng@nmsu.edu

Abdel-Hameed A. Badawy
Klipsch School of ECE
New Mexico State University
Las Cruces, NM
and Los Alamos National Laboratory
Los Alamos, NM
badawy@nmsu.edu

ABSTRACT

Large-scale distributed systems are a collection of loosely coupled computers interconnected by a communication network. They are now an integral part of everyday life with the development of large web applications, social networks, peer-to-peer systems, wireless sensor networks and many more. Because each disk by itself is prone to failure, one key challenge in designing such systems is their ability to tolerate faults. Hence, fault tolerance mechanisms such as replication are widely used to provide data availability at all times. On the other hand, many systems now are increasingly supporting new mechanism called erasure coding (EC), claiming that using EC provides high reliability at lower storage cost than replication. However, this comes at the cost of performance. Our goal in this paper is to compare the performance and storage requirements of these two data reliability techniques for two open source systems: HDFS and Ceph especially that the Apache Software Foundation had released a new version of Hadoop, Apache Hadoop 3.0.0, which now supports EC. In addition, with the Firefly release (May 2014) Ceph added support for EC as well. We tested replication vs. EC in both systems using several benchmarks shipped with these systems. Results show that there are trade-offs between replication and EC in terms of performance and storage requirements.

CCS CONCEPTS

• **Computer systems organization → Reliability**; *Availability*; *Redundancy*;

KEYWORDS

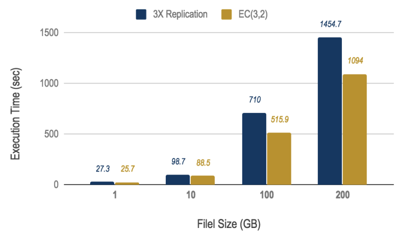
Fault Tolerance, Performance Evaluation, HDFS, Ceph, Distributed Storage Systems

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6446-1/18/07.
https://doi.org/10.1145/3219104.3229269

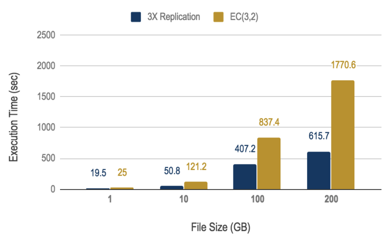
ACM Reference Format:
Yehia Arafa, Atanu Barai, Mai Zheng, and Abdel-Hameed A. Badawy. 2018. Evaluating the Fault Tolerance Performance of HDFS and Ceph. In *PEARC '18: Practice and Experience in Advanced Research Computing, July 22–26, 2018, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3219104.3229269

1 INTRODUCTION

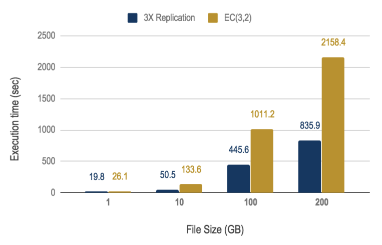
Due to data explosion in the recent years the key challenge is to design a cost efficient storage system to cope with this massive increase in data. A distributed storage system which is formed by networking together a large number of inexpensive storage devices provides one such alternative to store such a massive amount of data with high reliability and ubiquitous availability. Furthermore, in order for these huge amount of data to be managed distributed file systems are used providing the clients support for operation on the data. And since that as the system size increases the node failures becomes the norm rather than the exception. These file systems uses fault tolerance techniques to ensure data availability to the user at all times. Replication as in RAID-1 is the simplest way of such fault tolerance techniques where the same data is copied multiple times on different machines. Consequently, the system will have high fault tolerance and durability against losing its data. However, system architects must increase the number of replicas to achieve high durability for large systems and this will lead to increase in the system bandwidth and storage requirements. As a result, most large-scale distributed storage systems are transitioning to the use of erasure codes [3, 4, 7] which they claim it will provide higher reliability at significantly lower storage costs. The main idea behind EC is that the data is divided into k chunks and add m chunks of codes, those chunks are then distributed among the devices, this is called encoding. If one or more disks that hold the data fails, it can then be reconstructed from the remaining data disks + the coding disks, this is called decoding. A system with m coding chunks can tolerate failures up to m disks. One common and current use case for erasure coding is large scale storage systems. Hence, Knowing which fault tolerance technique to use under different conditions can help us understand their limitations and fields of improvements. Our goal here is to compare



(a) Writing Performance

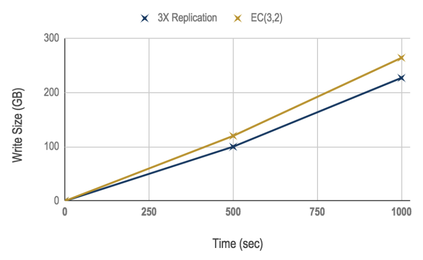


(b) Reading Performance, No Nodes Down

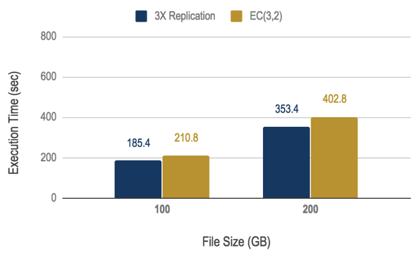


(c) Reading Performance, 2 Nodes Down

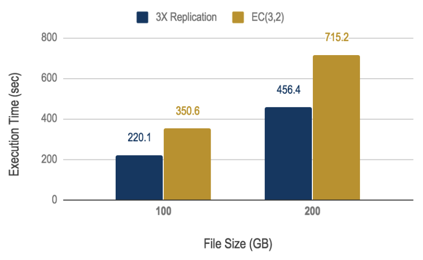
Figure 1: HDFS TESDFSIO Performance



(a) Writing Performance



(b) Reading Performance, No Nodes Down



(c) Reading Performance, 2 Nodes Down

Figure 2: Ceph Rados Bench Performance

the performance of these data reliability techniques for two open source system; HDFS[6] and Ceph[8]. We choose HDFS and Ceph because they are both open-source systems which uses commodity hardware to store large quantities of data. In addition, one can easily grow either systems so that it can handle more data simply by adding nodes. And most importantly, data are protected against hardware failure as they support both replication and EC techniques. Furthermore, both have a very good and growing community. In this paper we will show the results of running benchmarks on both of these systems to test the performance and storage requirements for fault tolerance techniques. We are concerned here with how fast the systems can recover from a failure and what are the storage overhead needed in order to do that. Experiments were performed on CloudLab [5], a free service for research and educational use. We tested replication vs. erasure coding in both systems using several benchmarks shipped with these systems. And in order to simulate device failures, we simulated hardware node failure(s) by implementing a fault injector mechanism that kills the target node process itself. For replication we used a replication factor of 3x, meaning that the same data is copied 3 times on different devices. On the other hand, EC, Reed Solomon technique was used with (3,2) setup, meaning that the data is divided into 3 chunks and 2 coding chunks are added. we ran the benchmarks in two testing setups for each technique: no hardware nodes failures, and with 2 hardware nodes failures.

2 EVALUATION

Experiments were performed on CloudLab [5]. We used the cluster at the University of Wisconsin where we used 7 different nodes

Table 1: Total Storage used in HDFS cluster

Write Size	Storage Used in 3X Replication	Storage Used in RS(3,2)
1 GB	3 GB	1.66 GB
10 GB	30.26 GB	16.82 GB
100 GB	303 GB	168 GB

Table 2: Total Storage used in Ceph cluster for the 1000 sec writing window

Storage Used in 3X Replication	Storage Used in RS(3,2)
690 GB	456 GB

for both systems, HDFS 3.0 and Ceph, each equipped with two Intel E5-2660 v3 10-core CPUs at 2.60 GHz (Haswell EP), 160GB ECC memory (10x 16 GB DDR4 2133 MHz dual rank RDIMMs), and 1.2 TB 10K RPM 6G SAS SFF HDDs. We tested replication vs. EC in both systems using several benchmarks shipped with these systems. We simulated node failure(s) by implementing a fault injector which kills the target node process itself. We were concerned with the total execution time and the overall throughout of running the benchmarks in two testing setups for each technique: no data nodes killed, which correspond to no hardware failure, and 2 nodes killed. Each node in the cluster ran Ubuntu 16.04 with ext4 file system. And for Ceph each OSD has XFS local file system.

2.1 HDFS

We performed our tests on HDFS using the TestDFSIO benchmark [2]. TestDFSIO is a benchmark used for read and write I/O test for HDFS. It is helpful for tasks such as stress testing HDFS, to discover performance bottlenecks, and give you a impression of how fast the cluster is in terms of I/O. We performed the same test once enabling 3X replication. And another enabling EC(3,2).

2.1.1 Writing performance. TestDFSIO wrote different file sizes and calculated the total execution time the system took to perform writing. Figure 1-a shows the written file sizes vs the total execution time each size took. It can be seen that with 3X replication HDFS requires more time to write data than EC(3,2). This is because the data is written to the disks in a serial manner. Then we were concerned about the total space each technique took. We found that the total space needed for EC(3,2) is nearly half of that 3x replication as shown in table 1.

2.1.2 Reading performance. Read tests were performed with two setups, first with all the data available and again after applying our technique which kills the target node and simulate node/device failure. We simulated 2 node failures (two data nodes down) while performing the read operation. Figure 1-b shows the reading performance with no nodes down while figure 1-c shows reading performance with 2 nodes down. In the case of read 3X replication, it needs less time to read the data which means much better performance than EC. This is due to the overhead of decoding and encoding data that EC apply.

2.2 Ceph

Rados Bench [1] was used to measure the performance of a Ceph object store. By Writing/reading files for a specific time and give back the findings. The test was performed by writing/reading files for a 1000 second for both 3X replication and EC(3,2).

2.2.1 Writing Performance. Figure 2-a shows that EC(3,2) shows better performance than 3X replication in writing, it wrote more data than 3x replication in the 1000 sec time window. Also for storage requirement EC needed only half storage than of that 3x replication as seen in table 2.

2.2.2 Reading Performance. The results of reading test are shown in figure2. Figure 2-b shows the reading performance while no nodes down, while figure 2-c shows the reading performance with 2 nodes down. It can be seen that 3x replication always have better performance in reading than EC(3,2), as in HDFS case this is due to the overhead of encoding and decoding found in EC.

3 CONCLUSIONS

Preliminary results shows that there is a trade-off between replication and erasure coding in HDFS and Ceph. Regarding Storage requirements, erasure coding is better as it needs nearly half the storage than that needed for replication of replication. The extra storage requirement needed in EC is for storing the m coding chunks. For the write performance, EC is slightly better than replication as the data is divided and then written in a parallel way but for replication the data is written serially which takes more time. Reading performance in replication is better than EC, since EC consumes more time in decoding of reconsecrating the data. Also EC needs to access all the surviving disks to reconsecrate the original data. On the other hand, replication needs only to access one disk which usually the nearest surviving replica to the user.

4 FUTURE WORK

Further research is needed to evaluate the performance of EC and replication in HDFS and Ceph. We plan to compare the performance of each technique in the two systems by running the same benchmark on both systems. This could help users to not only choose which fault tolerance technique to use but rather which system to to use and when to use it.

ACKNOWLEDGMENTS

The authors would like to thank the support we got from CloudLab to do our experiments without that support we would not have been able to conduct this study.

REFERENCES

[1] Rados Bench. [n. d.]. *The Ceph Benchmarking Tool.* https://github.com/ceph/cbt

[2] Michael G. Noll. 2011. *TestDFSIO Benchmark.* http://www.michael-noll.com/blog

[3] J. S. Plank. 2013. Erasure Codes for Storage Systems: A Brief Primer. *;login: the Usenix magazine* 38, 6 (December 2013).

[4] K.V. Rashmi, Nihar B. Shah, Dikang Gu, Hairong Kuang, Dhruba Borthakur, and Kannan Ramchandran. 2014. A "Hitchhiker's" Guide to Fast and Efficient Data Reconstruction in Erasure-coded Data Centers. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14).* ACM, New York, NY, USA, 331–342. https://doi.org/10.1145/2619239.2626325

[5] Robert Ricci, Eric Eide, and The CloudLab Team. 2014. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *USENIX ;login:* 39, 6 (Dec. 2014). https://www.usenix.org/publications/login/dec14/ricci

[6] K. Shvachko, Hairong Kuang, S. Radia, and R. Chansler. 2010. The Hadoop Distributed File System. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on.* 1 –10. https://doi.org/10.1109/MSST.2010.5496972

[7] Hakim Weatherspoon and John Kubiatowicz. 2002. Erasure Coding Vs. Replication: A Quantitative Comparison. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS '01).* Springer-Verlag, London, UK, UK, 328–338. http://dl.acm.org/citation.cfm?id=646334.687814

[8] Sage Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. 2006. Ceph: A Scalable, High-Performance Distributed File System. In *Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI '06).*