Enabling Data Processing under Erasure Coding in the Fog

Jad Darrous
Inria, ENS de Lyon, LIP
Lyon, France
jad.darrous@inria.fr

Shadi Ibrahim
Inria, IMT Atlantique, LS2N
Nantes, France
shadi.ibrahim@inria.fr

ABSTRACT

In recent years, Fog computing has been widely adopted as an extension to clouds by placing cloud services close to the users. While Fog promises to provide low-latency by enabling computation close to the data sources, it imposes several challenges for Big Data analytics including limited storage capacity, and computation and network heterogeneity. In this work, we argue that erasure coding (EC) can be applied as an alternative to replication to store data in Fog environments. Moreover, by making map task scheduling network-aware, we might achieve similar performance as replication but with half storage cost.

1 INTRODUCTION

The current explosion of devices connected to the Internet will generate a huge amount of data that becomes unsustainable to move to the cloud for processing. As a response, Fog computing paradigm has emerged by proposing to leverage computation resources along the path between end users and centralized cloud data centers. These computation resources can be micro data centers [6], Cloudlets [15], etc or in a more general way, can be called a Fog site. A variety of applications benefit from these nearby resources, for example, the accuracy of video processing and the latency of smart city applications can be improved using Fog [8, 13]. However, Fog poses many challenges: in contrary to the centralized clouds where storage and computation resources can be seen as virtually unlimited, Fog sites have limited storage and computation capacities. Moreover, these sites are connected with a heterogeneous network. These constraints raise the critical importance of efficient resource management in Fog.

Big Data analytics in Fog environments. Nevertheless, these sites generate an unprecedented amount of data that should be processed collectively. For example, analyzing logs to monitor global health of a system, or analyzing browsing habits to propose adequate products to the customers. Fog computing has been successfully used to improve the performance of a wide range of data processing applications including stream data applications [13], smart city applications [10], etc. Moreover, most efforts have focused on addressing network heterogeneity when transferring intermediate data in-between successive operators [13], or by improving the locality of tasks in MR applications as in Nebula framework [9].

Big Data analytics under EC. Erasure coding (EC) is a redundancy technique to achieve fault tolerance guarantee but with lower storage overhead compared to replication [14]. Recently, EC has been explored in many storage systems as an alternative of replication [11], also, it is integrated in the last major release of Hadoop Distributed File System (*i.e.*, HDFS 3.0.0) which is the primary storage back-end for Big Data analytics frameworks (*e.g.*, Hadoop [3], Spark [4], etc.). Many research efforts have been dedicated to adopt erasure coding in data-intensive clusters. HDFS-RAID [2] and DiskReduce [7] extend HDFS to encode replicated data offline. On the other hand, Zhang et al. [17] implement EC on the top of HDFS on the critical path (online encoding). They show that the execution times of MR applications can be reduced when the data are encoded compared to 3-ways replication.

EC for Big Data analytics in the Fog. Given the limited storage capacities of Fog nodes, the storage reduction brought by EC, and the important progress which has been made in reducing the CPU overhead of EC operations (and thus integrating EC operations on the critical path of data accesses [11]), EC is an ideal candidate for data processing in the Fog. However, EC brings important "high" network overhead. In contrary to replication where the majority of tasks can run locally, all the tasks under EC have to read most of their input data remotely: HDFS implemented EC with striped layout where an HDFS block is represented by "n" original chunks and "k" parity chunks, distributed on "$n+k$" data nodes. Even worse, the cost of data transfer when reading input data and its impact on the performance of data analytic jobs will be amplified in Fog environment due to network heterogeneity.

Contributions. Accordingly, as a first step towards realizing EC for data processing in Fog , we empirically demonstrate the impact of network heterogeneity on the execution time of MR applications when running in the Fog. We found that the map tasks under EC suffer obvious performance degradation (the maximum map task runtime is 3.3x longer compared to the mean) when reading input data from remote nodes. Therefore, we argue that by making map task scheduling network-aware we might achieve similar performance as replication but with half the storage cost.

2 BACKGROUND

Erasure Codes. Erasure coding (EC) is an encoding technique which can provide the same fault tolerance guarantee as replication [14] while reducing storage cost. Reed-Solomon codes (*RS*) [12] are the most deployed codes in current storage systems [2, 16]. $RS(n, k)$ splits the block of the data to be encoded into (n) smaller blocks called data chunks and then computes (k) parity chunks from these data chunks. Any (n) chunks out of the ($n + k$) chunks are sufficient to rebuild the original data block. RS codes present a trade-off between higher fault tolerance and lower storage overhead depending on the parameters (n) and (k). $RS(6, 3)$ and $RS(10, 4)$ are among the most widely used configurations. Compared to replication, $RS(6, 3)$ has 50% storage overhead and can tolerate 3 simultaneous failures, while 3-way replication has 200% storage overhead and can only tolerate 2 simultaneous failures.

EC Block layout in HDFS. EC is implemented with *striped* block layout in HDFS, *i.e.*, chunks of each block is physically distributed on multiple nodes. Compared to *contiguous* block layout,

striped layout is more efficient for handling small files and has less memory overhead when encoding and decoding.

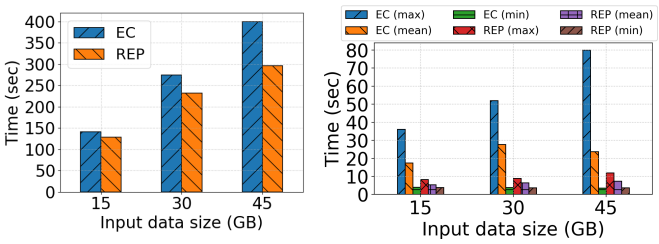
3 THE IMPORTANCE OF NETWORK-AWARE MAP TASK SCHEDULING

In this section, we show through experiments how jobs under EC behave compared to replication (REP) in a Fog environment.

Methodology. To demonstrate the feasibility of EC, we take a Fog infrastructure of 10 sites, each site is represented by a single physical machine. We suppose that these sites have the same storage capacities, but heterogeneous compute and network resources. The heterogeneity of computation is emulated by controlling the number of active cores (between 2 to 10 active cores per site). The network heterogeneity is emulated with the Linux Traffic-Control tool [1] (from 500 Mbps to 5000 Mbps). We deploy Hadoop workers on these 10 machines, with one extra machine to host the master processes. We set HDFS block size to 256 MB and the replication factor to 3. For EC, we use the default EC policy in HDFS, *i.e.*, $RS(6, 3)$ scheme with a cell size of 1 *MB*. Also, we disable the overlapping shuffle to study the impact of each phase separately. We use the Sort application as an example of a data-intensive job. The experiments have been performed on top of Grid'5000 [5]. Each machine is equipped with two Intel Xeon E5-2660 8-cores processors, 64 GB of main memory, and a 1 TB HDD. The machines are connected by 10 Gbps Ethernet network, and run 64-bit Debian stretch Linux with Java 8 and Hadoop 3.0.0 installed. We store the data at the workers in the main memory to eliminate the impact of the disks.

Experimental Results. Figure 1a depicts the job execution time of Sort application while increasing the input size. We can notice that REP outperforms EC by up to 35%. For example, for 30 GB input size, job execution time under EC is 274*s*, thus 18% higher than that under REP (232*s*). This difference can be explained by the time taken by the map and reduce phases, knowing that these two phases are not overlapping. Again, for 30 GB input size, reduce phase takes almost the same time under both EC and REP. However, map phase finishes in 21*s* under REP while it takes 61*s* under EC. The reason behind this is that map tasks exhibit high variation under EC (60%) compared to REP (19%). Figure 1b shows the minimum, the average, and the maximum runtimes of map tasks. We can clearly see that the runtimes of some map tasks are 2x to 3.3x longer compared to the mean under EC. Hence, by reducing the variation of map tasks, the performance of MR applications under EC can be improved.

Discussion. The previous results show that the performance of EC during the map phase is greatly outperformed by replication. While under REP, some tasks are non-local, under EC all the task read most of their data non locally. As the network is heterogeneous, tasks under EC should wait for the last chunk to be able to process the current piece of data, thus they experience huge variation in their runtimes. The late binding method used in other storage systems [11] to mitigate stragglers might not be efficient due to extra network traffic. Hence, a network-aware solution that takes network heterogeneity into account could be more suitable. In essence, an original data block could be recovered from any n chunks out of $n + k$. As decoding operation has negligible CPU overhead, leveraging parity chunks could reduce the retrieval time. For example, instead of requesting just the data chunks (as currently done by HDFS) where some of them reside on highly congested nodes, other parity blocks could be requested instead from faster nodes. However, to achieve the best job level performance, the scheduler should consider all the map tasks at once. The problem of minimizing the makespan of map tasks under EC is what we are going to investigate in more details in our future work.



(a) Job execution time **(b) Min and Max map runtime**

Figure 1: Sort application under EC and REP.

4 CONCLUSION

With the proliferation of Fog computing and the need to store and process more data, erasure codes manifest as an alternative to replication. In this work, we spot the inefficiency of EC when reading input data and we argue that incorporating network-aware chunks scheduler can potentially reduce the variation of map tasks, and thus, achieving lower job execution time.

ACKNOWLEDGMENTS

This work is supported by the Stack/Apollo connect talent project, Inria Project Lab program Discovery (see beyondtheclouds.github.io), and the ANR KerStream project (ANR-16-CE25-0014-01). The experiments presented in this paper were carried out using the Grid'5000/ALADDIN-G5K experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see www.grid5000.fr for details).

REFERENCES

[1] 2006. Linux Traffic Control. http://tldp.org/HOWTO/Traffic-Control-HOWTO.
[2] 2011. HDFS-RAID wiki. https://wiki.apache.org/hadoop/HDFS-RAID.
[3] 2019. Apache Hadoop. http://hadoop.apache.org.
[4] 2019. Apache Spark. https://spark.apache.org.
[5] 2019. Grid'5000. http://www.grid5000.fr.
[6] V. Bahl. 2015. Emergence of micro datacenter (cloudlets/edges) for mobile computing. https://www.microsoft.com/en-us/research/publication/emergence-of-micro-datacenter-cloudlets-edges-for-mobile-computing/.
[7] B. Fan, W. Tantisiriroj, L. Xiao, et al. 2009. DiskReduce: RAID for Data-intensive Scalable Computing. In *Proceedings of the 4th Workshop on Petascale Data Storage.*
[8] C.-C. Hung, G. Ananthanarayanan, P. Bodik, et al. 2018. Videoedge: Processing camera streams using hierarchical clusters. In *SEC'18.*
[9] A. Jonathan, M. Ryden, K. Oh, A. Chandra, et al. 2017. Nebula: Distributed Edge Cloud for Data Intensive Computing. *IEEE Transactions on Parallel and Distributed Systems* 28, 11 (Nov 2017).
[10] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, et al. 2017. SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services. *IEEE Access* 5 (2017).
[11] K. V. Rashmi, M. Chowdhury, J. Kosaian, I. Stoica, et al. 2016. EC-Cache: Load-Balanced, Low-Latency Cluster Caching with Online Erasure Coding. In *OSDI'16.*
[12] I. Reed et al. 1960. Polynomial codes over certain finite fields. *Journal of the Society of Industrial and Applied Mathematics* 8, 2 (06/1960 1960), 300–304.
[13] E. G. Renart, J. Diaz-Montes, et al. 2017. Data-Driven Stream Processing at the Edge. In *ICFEC'17.*
[14] R. Rodrigues et al. 2005. High Availability in DHTs: Erasure Coding vs. Replication. In *Peer-to-Peer Systems IV.*
[15] M. Satyanarayanan, P. Bahl, R. Caceres, et al. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4 (Oct 2009).
[16] K. Shvachko, H. Kuang, S. Radia, et al. 2010. The Hadoop Distributed File System. In *MSST'10.*
[17] Z. Zhang, A. Deshpande, X. Ma, E. Thereska, et al. 2010. *Does erasure coding have a role to play in my data center?* Technical Report. Microsoft research.