An Empirical Evaluation of MapReduce under Interruptions

Hui Jin, Xi Yang, Xian-He Sun, Ioan Raicu
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
{hjin6, xyang34}@iit.edu, {sun, iraicu}@cs.iit.edu

***Abstract*—The presence of interruptions is an unwanted but inevitable fact that all large-scale distributed computing systems have to face. The interruptions are more prevailed for MapReduce applications, as often MapReduce runs on the top of the commodity hardware based clusters, which are more vulnerable than traditional HEC systems. The problem is further exaggerated when running MapReduce applications in distributed non-dedicated computing environment, where the host applications have the privilege to take back the computing power at random and interrupt MapReduce applications. This study intends to evaluate the resilience of MapReduce applications through an empirical evaluation. In particular, we set up a MapReduce system, inject interruptions with different patterns, and study their impact on the performance of the TeraSort benchmark. We simulate both cluster and distributed non-dedicated computing environment to observe the impact of these interruptions. Both the data locality and benchmark execution time have been measured. We also vary the number of replicas to observe its impact on the application performance. The experimental results show that interruptions have a significant impact on the performance of MapReduce applications. MapReduce in distributed computing environment is more vulnerable to interruptions due to the high data migration cost. Finally, we show that extra data replicas help to mitigate the impact of interruptions.**

I. Introduction

The presence of interruptions is inevitable for large-scale parallel applications. The performance analysis and optimization under failures of traditional HPC applications such as MPI is currently an active research area and has been well studied in the literature. However, little work has been done to investigate the impact of interruptions on MapReduce applications. Interruptions are more prevailed for MapReduce applications since they typically operate on commodity hardware based clusters. The problem will be exaggerated when running distributed MapReduce applications under non-dedicated distributed computing environment, where the interruptions from the host applications occur arbitrarily to suspend the MapReduce applications [1].

The performance of MapReduce is a coordinated impact of multiple factors. From the perspective of hardware, computing power directly decides the duration of each map/reduce task and has a substantial impact on the MapReduce performance. In addition, extra data migration is triggered during the job execution to balance the workloads of different nodes, which depends on the network bandwidth. From the perspective of MapReduce framework, data placement decides the locality of the tasks and affects the data migration during job execution. The job scheduler is also a key factor in deciding the MapReduce performance. The MapReduce application performance is also impacted by other parameters, such as the number of replicas and the failure pattern, among many other parameters.

In this study, we aim at building an experimental framework to evaluate the resilience of MapReduce applications. In particular, we set up Hadoop [2], an open-source MapReduce framework in a cluster, inject interruptions to the Hadoop benchmark, and observe the impact. The experimental study differentiates the impact of interruption patterns, the number of interrupted nodes, and the number of replicas. We also modify the Hadoop source code to emulate distributed non-dedicated computing environment and observe its performance under interruptions.

The rest of this paper is organized as follows. We introduce the experimental frame work in section II. Section III presents the experimental results and the observations. Section IV concludes this work.

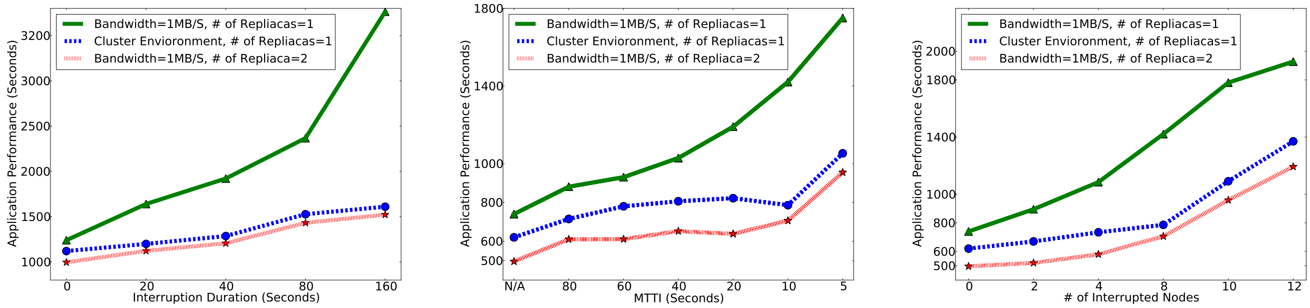
II. Experimental Framework

Our experiments were conducted on a cluster of 17 Sun Fire Linux-based compute nodes. Each node is equipped with dual 2.7 GHz Opteron quad-core processors, 8 GB memory and 250GB SATA hard drive.

Hadoop 0.20.2 has been installed on the cluster to evaluate the MapReduce performance. One node works as the dedicated namenode and job tracker and other 16 nodes are configured as the datanodes and task trackers. Each node is able to run one map task at a time. The Hadoop filesystem (HDFS) directory of each datanode resides in the local disk.

We use TeraSort benchmark for the performance evaluation. We first use TeraGen to generate 400M rows of data, which is used as the input of TeraSort. The input of the TeraSort is a set of 596 blocks, organized in two files. The size of each block is 64MB.

The experiments are conducted on both the cluster environment and the distributed computing environment. To simulate the distributed computing environment, we have modified the Hadoop source code such that the data transfer between two distinct nodes are delayed based on a bandwidth of 1MB/s. We



(a) Interruption Duration (b) MTTI (c) # of Interrupted Nodes

Fig. 1: MapReduce Application Performance

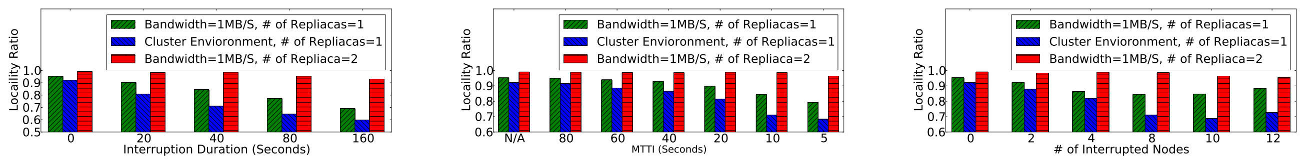


Fig. 2: Locality Ratio

have also varied the number of replica from 1 to 2 to observe its impact on the MapReduce performance.

We stop and resume the task tracker process of a node to simulate the arrival and termination of an interruption.

Currently we are interested in the reliability of the map phase. We measure the map phase duration of each node and use the one with maximum duration as the application performance. We have also measured the locality ratio of the map tasks to observe its relationship with interruptions and the application performance.

III. Preliminary Experimental Result

In Fig. 1a we plot the application performance with different interruption durations. The application performance is more vulnerable to interruptions for the first scenario, which simulates the distributed environment and the number of replicas is set as 1. Increasing the number of replicas helps to improve the performance. When the number of replicas is 2, the application performance under distributed environment has a performance better than the cluster environment, due to its high locality ratio, which can be observed from Fig. 2a. Scenario 2 of cluster environment has more data migration and the lowest locality ratio in Fig. 2a, but still leads to an optimal performance in Fig. 1a. More data migration occurs in this scenario due to its high network bandwidth, and helps to improve the performance.

In Fig. 1b we vary the Mean-Time-To-Interruption (MTTI) from 80 to 5 seconds. The first group of data with N/A reflects the interruption-free performance. The third scenario is still least sensitive to interruptions. From Fig. 2b we observe that its locality ratio is close to 1. The extra replica reduces the data migration while keeping the locality of the map tasks, which helps to maintain its optimal performance.

Fig. 1c and 2c demonstrate the MapReduce performance with different number of interrupted Nodes. An interesting observation is that the locality of scenario 1 initially goes down, and goes up when the number of interrupted noes is 8 or higher. Similar observation can be found for scenario 2. The amount of migrations is essentially decided by the reliability variance among different nodes. The reliability variance is at its maximum when 8 nodes are interrupted.

IV. Conclusions and Future Work

In this study, we quantify the impact of interruptions on the MapReduce applications by an experimental approach. The experimental results show that MapReduce under distributed computing environment is significantly impacted by interruptions. The extra replica helps to mitigate the impact of interruptions. Data locality has a critical impact on the application performance.

In the future, we plan to extend the experiments to large computing environments (e.g. ANL Magellan [3], FutureGrid [4]). We will also evaluate the performance by injecting interruptions into other MapReduce component such as datanode.

References

[1] H. Lin, X. Ma, J. S. Archuleta, W. chun Feng, M. K. Gardner, and Z. Zhang, "MOON: MapReduce On Opportunistic eNvironments," in *HPDC 2010*.
[2] "The Hadoop Project Website," *http://hadoop.apache.org/*.
[3] "Magellan Website," *http://magellan.alcf.anl.gov/*.
[4] "FutureGrid Website," *https://portal.futuregrid.org/*.