2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Understanding Unsuccessful Executions in Big-Data Systems

Andrea Rosà
Università della Svizzera italiana (USI)
Faculty of Informatics
Lugano, Switzerland
Email: andrea.rosa@usi.ch

Lydia Y. Chen
IBM Research Lab Zurich
Cloud Server Technologies Group
Rüschlikon, Switzerland
Email: yic@zurich.ibm.com

Walter Binder
Università della Svizzera italiana (USI)
Faculty of Informatics
Lugano, Switzerland
Email: walter.binder@usi.ch

***Abstract*—Big-data applications are being increasingly used in today's large-scale datacenters for a large variety of purposes, such as solving scientific problems, running enterprise services, and computing data-intensive tasks. Due to the growing scale of these systems and the complexity of running applications, jobs running in big-data systems experience unsuccessful terminations of different nature. While a large body of existing studies sheds light on failures occurred in large-scale datacenters, the current literature overlooks the characteristics and the performance impairment of a broader class of *unsuccessful executions* which can arise due to application failures, dependency violations, machine constraints, job kills, and task preemption. Nonetheless, deepening our understanding in this field is of paramount importance, as unsuccessful executions can lower user satisfaction, impair reliability, and lead to a high resource waste. In this paper, we describe the problem of unsuccessful executions in big-data systems, and highlight the critical importance of improving our knowledge on this subject. We review the existing literature on this field, discuss its limitations, and present our own contributions to the problem, along with our research plan for the future.**

I. Introduction

In today's multi-tenancy and multi-purpose datacenters, "big-data" is becoming the key application, featuring high fanout jobs and performance dependency on data locality. Workloads in these *big-data systems* are greatly diversified, showing a high degree of heterogeneity and dynamicity [1]. On the one hand, short-running interactive applications are often employed by data analysts to formulate complex queries on large data sets [2]. On the other hand, long-running computational tasks execute on these systems continuously, to run large-scale simulations [3], solve challenging problems in scientific fields [4], support various services like web search and indexing, video streaming, and enterprise applications [5], or process large data streams collected from sensor networks.

Due to their large-scale, heterogeneity, and complexity, applications running in big-data systems experience unsuccessful terminations of different nature which potentially turn into a critical performance impediment. Apart from application *failures* which can naturally arise due to programmers' errors, system failures, software bugs or unforeseen events, jobs or tasks may be specifically terminated for several reasons. First of all, complex intra-jobs and inter-jobs dependences may lead to forced terminations due to *dependency violation*, e.g., a job can be terminated due to the death of another job on which it was dependent. Similarly, users may impose *machine constraints* on their tasks, requiring them to run on a specific subset of machines with a given configuration, either in terms of software, hardware, or architecture. Unavailability of these machines causes the unsuccessful termination of constrained tasks. Job inter-dependencies and machine constraints are frequent in today's applications [6], causing job and task terminations repeatedly. Users may also deliberately *kill* their applications for a number of reasons [6]–[8], further exacerbating the problem of dependency violation and leading to wasted computations [9]. Finally, the multitude of jobs co-executing in big-data systems [1] and the impressive amount of resources needed for their computations [10] push administrators to include *task preemption* in scheduling policies for their systems, either in the form of hard deadlines [2] or task priorities [6], [7]. As a result, applications may be evicted to release resources for higher-priority tasks. We refer to all these terminations as *unsuccessful executions* which do not contribute to the completion of jobs or tasks.

Although a large body of related work analyzes failures in large datacenters, most notably in terms of hardware [4], [11], software [7], [8], network components [5], and virtual machines [12], little work has been done [13], [14] in studying the broader class of unsuccessful executions in big-data systems. Nevertheless, deepening our knowledge in this field is of paramount importance, as unsuccessful executions can result in degradation of Quality of Service (QoS), reliability and energy waste that can ultimately lead to a high resource waste and performance impairment. As a first step toward this direction, it is necessary to understand the impact of unsuccessful executions on existing large-scale systems, by measuring and quantifying their performance drawbacks on system resources and running applications. Secondly, a deep investigation of their root causes and their correlations on job attributes and system parameters would allow an optimal design of both applications and systems, with the final goal of maximizing their performance and reliability. Finally, as motivated by several prior studies on system failures [15], [16], modeling unsuccessful executions could uncover patterns, dependencies and characteristics that can benefit the development of new scheduling policies, e.g., based on detection and prediction that can minimize the occurrences of unsuccessful executions in big-data systems. Understanding unsuccessful executions is a fundamental step towards improving datacenters' robustness and reliability.

The challenges of the proposed studies stem from the large-scale and complex nature of such big-data systems, where computing nodes are highly heterogeneous, applications show intricate dependencies among jobs and on the underlying hardware, tasks are subjected to multiple resubmissions with different outcomes, and job scheduling gets complicated by task preemption, dependency violation, and machine constraints. Moreover, jobs may be composed of an enormous amount of tasks [6] which can have disparate resource demands and be executed on dissimilar hardware. As a result, understanding unsuccessful executions in big-data systems is an essential and challenging research problem.

II. Related Work

Unsuccessful executions in big-data systems have been rarely studied. On the one hand, a large body of related work focuses on workload characterization [7]–[9] in large-scale datacenters, paying little attention to the dependability issues of these systems. On the other hand, despite the presence of several studies on the reliability of big-data systems, many of them consider only hardware [4], [11] or software failures [7], [8], without taking into account unsuccessful executions that could arise due to scheduling policies, task/machine dependencies, or users' job killing decisions. As a result, little work has been done in understanding unsuccessful executions in big-data systems.

One of the few existing studies in this direction is the one conducted by Chen et al. [13] which presents a general statistical summary on job failures in a Google large-scale datacenter [6]. The authors attempt to correlate job failures with operations on machines, users attributes, and machine constraints. Their main findings are that: 1) resource consumption and running time of successful jobs are lower than those of killed and failed jobs, and 2) updating machines repeatedly increases their reliability. They also highlight how low-priority jobs contend for resources with high-priority jobs, leading high-priority jobs to fail with a high probability. However, this observation is in contrast with the inner principles of task preemption that evict low-priority tasks without harming high-priority ones. Generally speaking, their analysis pays little attention to task evictions, although the presence of rich field data on priority preemption in the traces of this system [6].

Using the same data set of the previous work, Garraghan et al. [14] study the statistical properties of failure and repair times for tasks and servers. Their main goal is to provide realistic parameters for simulations in big-data systems. Consequently, they especially focus on fitting empirical distributions to several theoretical ones. Although it is very important to provide simulation parameters for big-data machines and tasks, this work does not consider other failure characteristics that can benefit modeling studies in this field, such as dependencies between failures, and correlations between jobs, tasks and machines. Moreover, the authors assimilate task evictions and kills, without deepening the knowledge on these two classes of tasks.

Still, on the same dataset, Liu et al. [9] distinguish resources used by a task depending on its final outcome. They highlight that only 60% of used CPU time is actually consumed by successful tasks, while 15% and 25% are used by failed and killed tasks, respectively. However, they conduct their analysis on a very limited spatial and temporal domain, i.e., they only analyze a single day of workload running on a single machine. Moreover, they only consider CPU, overlooking other resources like memory and disk, whose data is included in the original trace [6]. Hence, their results may suffer from the limited portion of analyzed data which may not represent every aspect of the complex and dynamic workload in big-data systems [1].

Ren et al. [7] provide a characterization study on successful, failed and killed jobs in a MapReduce environment where jobs are frequently evicted by high-priority applications. They highlight that failed jobs consume non-negligible computing resources, and that most task failures are mainly caused by out-of-memory exceptions. Their work points out that it is necessary to implement more efficient memory allocation strategies in order to reduce job failures. Analyzing MapReduce data that Yahoo! made available to selected universities, Kavulya et al. [8] observe the presence of a high latency between the first aborted task of a job and its job failure, with a maximum latency of 4.3 days. The authors use this finding to pinpoint the need of better diagnosis and recovery methods in order to reduce error latencies in computing tasks. They also show that most failures occur in the map phase, and that node failures are more frequent in killed jobs. However, as workload in these clusters is limited to MapReduce applications, it is difficult to generalize the findings of these two works to generic cloud and big-data workloads.

Overall, although aforementioned studies analyze different aspects of failures thoroughly, they often simplify the complex relationships between systems and applications, overlook their dependences, and fall short in providing a comprehensive analysis on unsuccessful executions, their modeling characteristics, their root causes, and their impact on the underlying systems and applications.

III. Research Directions

Understanding unsuccessful executions in big-data systems is an interesting and challenging open problem. In the following, we present our contribution on this subject along two different research directions.

A. Improve Understanding of Unsuccessful Executions

As a first step towards understanding unsuccessful executions, it is imperative to analyze field data collected from existing big-data systems. Such an analysis can shed light on the characteristics of unsuccessful executions, their performance impact, their dependences, and their root causes. In this direction, we are studying [17], [18] three types of

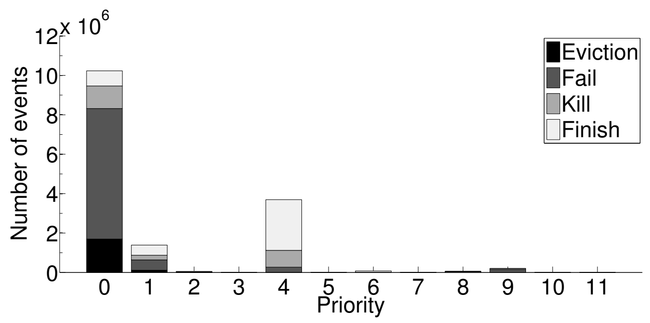


Fig. 1. Occurrences of ending events, breakdown by task priority. Higher values of task priority represent important tasks. The figure is taken from our prior work [17].

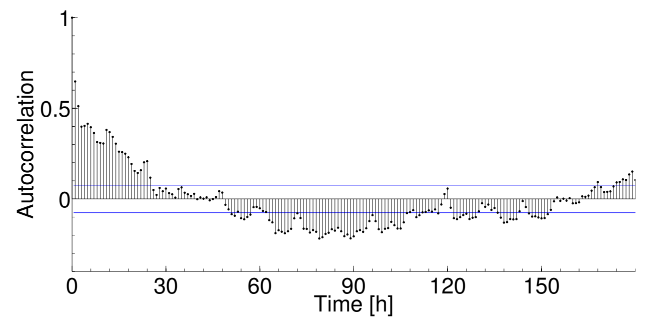


Fig. 2. Autocorrelation function (ACF) of eviction events. Horizontal lines indicate 95% confidence bounds. We show only time lags ranging from 0 to 180 hours. The figure is taken from our prior work [17].

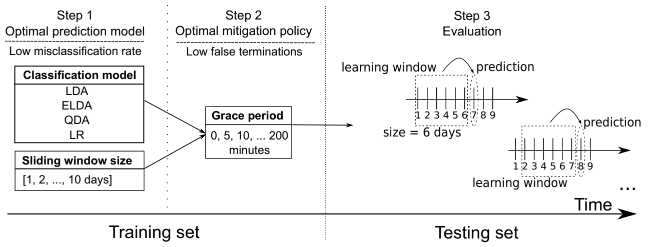


Fig. 3. Summary of the proposed predictive model for failed jobs. The figure is taken from our prior work [19].

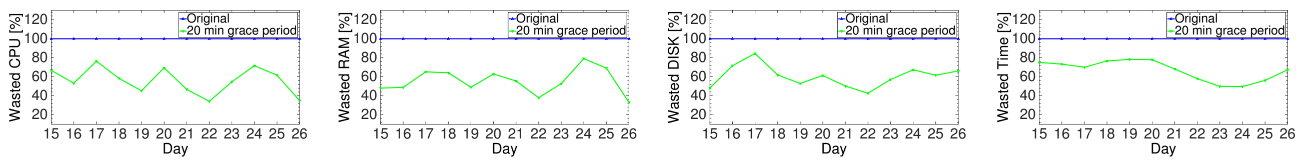
unsuccessful executions, i.e., task preemption (eviction), task kills (kill), and application failures (fail), in traces of a large-scale Google datacenter [6], where each *job* is composed of multiple *tasks*, which in turn experience multiple *events*. As a motivation of the urgency of investigating unsuccessful executions, we present the distribution of different types of events in Figure 1. To our surprise, less than 26% of task schedulings are completed successfully, and a non-negligible percentage of scheduled tasks is terminated due to fail and eviction. Unsuccessful executions are non negligible especially at low priorities, as shown by the high number of kill, eviction, and fail events occurring to tasks of priority 0 and 1. Both jobs and tasks have high probability to be executed unsuccessfully, causing non negligible resource waste and slowdown of the application performance.

Motivated by the high dependency of task preemption on the scheduling policy and the increasing adoption of priority scheduling in big-data systems, we particularly focus on eviction events and conduct an in-depth analysis on their performance impact on the underlying system, their temporal dependencies, and the underlying mechanisms of the eviction process. As an example of our current results, we present the autocorrelation function (ACF) of eviction based on the number of events per hour in Figure 2. Our goal is to find out whether an eviction event increases or decreases the chances of experiencing a subsequent eviction in the near future. From the figure, we can observe that there is a strong time dependency in the first few hours, indicating that eviction events tend to happen repeatedly in the system. Moreover, judging from the shape of the ACF, we propose that Moving Average models are appropriate to describe the time dependency of eviction.

B. Predict Unsuccessful Executions

As applications may be terminated upon experiencing unsuccessful executions, resources consumed by them might actually be wasted, as they lead to no useful computation. A possible way to reduce such a waste of resources is identifying and predicting jobs failed due to unsuccessful executions. Execution of these jobs can be denied or halted, effectively reducing resource waste and task slowdown. In this direction, we are developing [19] an on-line prediction model that can predict expected job outcomes, i.e., successful vs. failed, upon job arrival. In addition to the model, we propose a delay-based mitigation policy that proactively terminates predicted-to-fail jobs. The main objectives of the proposed methodology are (1) to accurately predict job failures in highly heterogenous and time-varying big-data systems, (2) to reduce false terminations of jobs that could have completed their execution successfully, and (3) to minimize the amount of resources wasted by predicted-to-fail jobs. We evaluate the proposed model by predicting failed jobs in traces of a large-scale Google datacenter [6].

Figure 3 shows the skeleton of our model. Following conventional statistical learning approaches, we partition the data traces into a training set and a testing set. We first determine the optimal choice of the prediction model and mitigation policy in the training set during the training phase. The selection criteria are the low misclassification rate as well as the low number of false terminations of successful jobs. Afterward, we apply the prediction model and mitigation policy that result from the training phase on the testing set. Our proposed model classifies job outcomes based on historical data in a sliding window fashion, as shown in the figure. We consider several well-known classification models in the training phase, i.e., 1) Linear Discriminant Analysis (LDA), 2) Linear Discriminant Analysis on expanded basis (ELDA), 3) Quadratic Discriminant Analysis (QDA), and 4) Logistic Regression (LR). The input of the model is a set of features which describes key job and system attributes collected upon



(a) CPU. (b) RAM. (c) DISK. (d) Computational time.

Fig. 4. Comparison of daily resource waste with no mitigation policy and mitigation policy with 20 minutes grace period. The figure is taken from our prior work [19] and is partially adapted.

job arrival. When jobs are predicted to fail, we first grant their execution, then we proactively terminate them after a grace period, if they are still running. In this way, we give chance to predicted-to-fail jobs to complete their execution, with the objective of striking a good tradeoff between resource waste and false terminations of successful jobs.

As a result from the evaluation phase, we determine ELDA to be the optimal classification model, with a corresponding sliding window size of 10 days and a grace period equal to 20 minutes. Figure 4 shows the percentage of resource wasted by the optimal model, normalized by the resource waste of the original traces. To ease the comparison, we plot an additional straight line of 100%, which refers to the original resource waste in the traces. In terms of average over the entire testing set, our model achieves a reduction of resource waste equal to 46.7%, 47.1%, 40.5%, and 33.2% for CPU, RAM, DISK, and computational time, respectively, compared to the original resource waste. Moreover, the correspondent false negative rate is very low, i.e., below 1%. These results confirm that our predictive model, combined with the proposed mitigation policy, is able to effectively reduce resource waste in big datacenters, where the workload is highly dynamic and diversified and system load changes frequently.

IV. Conclusion and Future Work

In this paper, we pinpointed the critical importance of understanding unsuccessful executions in big-data systems, and discussed our contribution to the problem. In the future, we plan to extend our work as follows. Currently, our characterization study on unsuccessful executions is limited to a single datacenter, due to the difficulty of finding field data from large-scale big-data systems. If traces from other datacenters will be released to the public, it would be interesting to extend this analysis on those systems, in order to validate our findings and find out common behaviors among different datacenters. Moreover, we plan to uncover in-depth performance modeling of task preemption, either via field simulation or analytical studies, in order to improve current knowledge about priority scheduling and the resulting impact on the system. Finally, we plan to extend our prediction model for failed jobs by including also dynamic job features, i.e., attributes that can be collected only after job completion, such as used resources, execution time, and patterns of non-fatal unsuccessful executions experienced. Such an addition can lead to a better prediction, resulting into a more accurate reduction of resource waste, at the cost of increased model complexity.

Acknowledgment

This work has been supported by the Swiss National Science Foundation (project 200021_141002) and EU commission under FP7 GENiC project (608826).

References

[1] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis," in *ACM SoCC*, 2012, pp. 7:1–7:13.

[2] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz, "Energy Efficiency for Large-scale MapReduce Workloads with Significant Interactive Analysis," in *ACM EuroSys*, 2012, pp. 43–56.

[3] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how HPC systems fail," in *IEEE/IFIP DSN*, 2013, pp. 1–12.

[4] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam, and R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models," in *IEEE/IFIP DSN*, 2006, pp. 425–434.

[5] R. Potharaju and N. Jain, "When the Network Crumbles: An Empirical Study of Cloud Network Failures and Their Impact on Services," in *ACM SoCC*, 2013, pp. 15:1–15:17.

[6] J. Wilkes, "More Google cluster data," Google research blog. https://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1, Nov 2011.

[7] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou, "Workload Characterization on a Production Hadoop Cluster: A Case Study on Taobao," in *IEEE IISWC*, 2012, pp. 3–13.

[8] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An Analysis of Traces from a Production MapReduce Cluster," in *IEEE/ACM CCGrid*, 2010, pp. 94–103.

[9] Z. Liu and S. Cho, "Characterizing Machines and Workloads on a Google Cluster," in *SRMPDS*, 2012, pp. 397–403.

[10] D. Çavdar, A. Rosà, L. Y. Chen, W. Binder, and F. Alagöz, "Quantifying the Brown Side of Priority Schedulers: Lessons from Big Clusters," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 3, pp. 76–81, Dec. 2014.

[11] K. V. Vishwanath and N. Nagappan, "Characterizing Cloud Computing Hardware Reliability," in *ACM SoCC*, 2010, pp. 193–204.

[12] R. Birke, I. Giurgiu, L. Y. Chen, D. Wiessmann, and T. Engbersen, "Failures Analysis of Virtual and Physical Machines: Patterns, Causes and Characteristics," in *IEEE/IFIP DSN*, 2014, pp. 1–12.

[13] X. Chen, C.-D. Lu, and K. Pattabiraman, "Failure Analysis of Jobs in Compute Clouds: A Google Cluster Case Study," in *IEEE ISSRE*, 2014, pp. 167–177.

[14] P. Garraghan, P. Townend, and J. Xu, "An Empirical Failure-Analysis of a Large-Scale Cloud Computing Environment," in *IEEE HASE*, 2014, pp. 113–120.

[15] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in the Wild: A Large-scale Field Study," in *ACM SIGMETRICS*, 2009, pp. 193–204.

[16] R. Sahoo, M. Squillante, A. Sivasubramaniam, and Y. Zhang, "Failure Data Analysis of a Large-Scale Heterogeneous Server Environment," in *IEEE/IFIP DSN*, 2004, pp. 772–781.

[17] A. Rosà, L. Y. Chen, R. Birke, and W. Binder, "Demystifying Casualties of Evictions in Big Data Priority Scheduling," *SIGMETRICS Perform. Eval. Rev.*, Mar. 2015.

[18] A. Rosà, L. Y. Chen, and W. Binder, "Understanding the Dark Side of Big Data Clusters: an Analysis beyond Failures," in *IEEE/IFIP DSN*, 2015.

[19] ——, "Predicting and Mitigating Jobs Failures in Big Data Clusters," in *IEEE/ACM CCGrid*, 2015.