

Predicting Scheduling Failures in the Cloud: A Case Study with Google Clusters and Hadoop on Amazon EMR

Mbarka Soualhia¹ Foutse Khomh² Sofiène Tahar¹
¹Concordia University, ²Polytechnique Montréal, Montréal, Quebec, Canada
{soualhia,tahar}@ece.concordia.ca , foutse.khomh@polymtl.ca

Abstract—Cloud Computing has emerged as a key technology to deliver and manage computing, platform, and software services over the Internet. Task scheduling algorithms play an important role in the efficiency of cloud computing services as they aim to reduce the turnaround time of tasks and improve resource utilization. Several task scheduling algorithms have been proposed in the literature for cloud computing systems, the majority relying on the computational complexity of tasks and the distribution of resources. However, several tasks scheduled following these algorithms still fail because of unforeseen changes in the cloud environments. In this paper, using tasks execution and resource utilization data extracted from the execution traces of real world applications at Google, we explore the possibility of predicting the scheduling outcome of a task using statistical models. If we can successfully predict tasks failures, we may be able to reduce the execution time of jobs by rescheduling failed tasks earlier (*i.e.*, before their actual failing time). Our results show that statistical models can predict task failures with a precision up to 97.4%, and a recall up to 96.2%. We simulate the potential benefits of such predictions using the tool kit CloudSim and found that they can improve the number of finished tasks by up to 40%. We also perform a case study using the Hadoop framework of Amazon Elastic MapReduce (EMR) and the jobs of a gene expression correlations analysis study from breast cancer research. We find that when extending the scheduler of Hadoop with our predictive models, the percentage of failed jobs can be reduced by up to 45%, with an overhead of less than 5 minutes.

Keywords-Failure Prediction, Tasks Scheduling, Cloud, Google Clusters, Hadoop, Amazon Elastic MapReduce.

I. INTRODUCTION

Cloud Computing has emerged as a key technology that delivers and manages services over the Internet. Customers can lease services provided by cloud computing systems, ramping up or down the capacity as they need and paying only for what they use. Nowadays, cloud computing services are used for several applications such as Internet of Things, Image Processing, Data Mining, and Web Analytics [1] [2]. Task scheduling problems are of paramount importance in cloud environments. Indeed, an efficient scheduling of tasks and jobs across the various heterogeneous virtual clusters that constitute a cloud is critical to ensure good computation time and resource utilisation. Although several task scheduling algorithms have been proposed in the literature for cloud computing systems, cloud schedulers still experience many failures due to unforeseen events such as unpredicted demands of services or hardware outages. We believe that an efficient scheduling of tasks requires a proactive response to changes in cloud environments. If we can predict changes in cloud environments accurately, we may be able to adjust scheduling

decisions accordingly and reduce the amount of task scheduling failures. Recently, Chen et al [3], [4] examined tasks failures in compute clouds and suggest that predicted failed tasks be killed immediately without processing them, in order to avoid wasting resources. However, although killing these predicted failed tasks may reduce resources wastage, it does not guarantee a good level of QoS (Quality of Service), since the killed tasks are likely to affect the overall performance of a cloud application. A better decision would be to reschedule the tasks quickly on appropriate clusters with adequate resources in order to ensure their timely and successful completion.

In this paper, we explore the possibility of predicting the scheduling outcome of a task using statistical models and historical information about the execution of previously scheduled tasks. Our goal is to achieve early rescheduling of potential failed tasks in order to improve tasks and jobs execution time and resources utilisation. We use statistical modelling to establish and inspect dependencies between tasks and jobs characteristics such as execution time, scheduling time, resources usage, machines workload, scheduling constraints, and tasks scheduling outcomes. Using tasks execution and resource utilization data from Google applications, collected over a period of one month in 2011 [5], we address the following three research questions:

RQ1) How often does a scheduled task or job is Failed, Evicted, Lost, or Killed?

We observed that 42% of the jobs and 40% of the tasks from the Google dataset were not finished successfully. We also found that unfinished (*i.e.*, evicted, failed or killed) jobs and tasks are characterized by long waiting times and execution times. Moreover, we noticed that a job often fails because of the failures of some of its tasks, and tasks also fail because of the failure of dependent tasks.

RQ2) Can we predict the outcome of scheduling events based on cluster log files?

First, we determined the variables that affect directly the scheduling outcome of task or job. Then, we applied Decision Tree, Boost, GLM, CTree, Random Forest and Neural Network algorithms to predict whether or not a scheduled task will fail. Our best prediction model is obtained with Random Forest. This model achieves a precision up to 97.4%, and a recall up to 96.2%. Cloud service providers could make use of such prediction models to improve the performance of their scheduling algorithms.

RQ3) Which benefits can be achieved by predicting the outcome of scheduling events?

We evaluate the potential benefits of our prediction models using the tool kit GloudSim which was built to simulate the original workload of Google applications [6]. We examine whether our models can identify and predict failure events when scheduling tasks and enable better scheduling decisions. Results show that prediction models can help reduce the execution time of the jobs and tasks. Also, the early failure predictions reduce the number of failed tasks by up to 40%.

To demonstrate the practicality of our prediction models in a real world setting, we implement and deploy the obtained prediction models on Amazon EC2, extending the scheduler of the Hadoop framework of Amazon EMR [6]. We reproduce and execute a series of jobs from a gene expression correlations analysis study in breast cancer research [7]. Results show that the extended version of Hadoop’s scheduler generates better scheduling policies, *i.e.*, the percentage of failed jobs is reduced by 45%. This improvement of the performance of the scheduler is achieved at a minimum cost of less than 5 minutes over a total execution time of 30 minutes.

The remainder of this paper is organized as follows: Section II describes the case study design, our proposed methodology to process the data from Google Traces. Section III describes the results of RQ1 and RQ2. The simulation of the benefits of our proposed prediction models with GloudSim (*i.e.*, RQ3) is presented in Section IV. Section V presents the results of the case study with Hadoop. Section VI discusses threats to the validity of our work, Section VII summarizes the related literature and Section VIII presents the conclusion and discusses future works.

II. METHODOLOGY

In this section, we will describe the design of our study, the studied system and our data extraction and analysis approaches to answer the following research questions:

- RQ1) How often does a scheduled task or job is Failed, Evicted, Lost, or Killed?
- RQ2) Can we predict the outcome of scheduling events based on cluster log files?
- RQ3) Which benefits can be achieved by predicting the outcome of scheduling events?

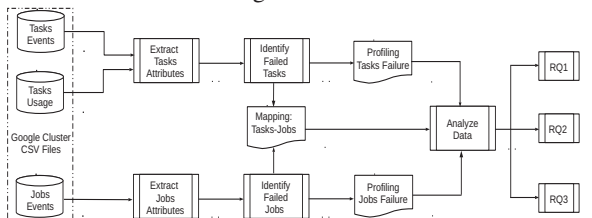


Fig. 1: Overview of Our Proposed Methodology

A. Case Study Design

In order to answer our three research questions, we performed an empirical study using large-scale data (*i.e.*, 158

GB) collected from Google clusters. A Google cluster is a set of different machines that are inter-connected with high-bandwidth network dedicated to large and distributed clusters. The machines of one cluster are sharing the same scheduling resources allocation and management systems. The schedulers of these machines receive and schedule a large number of jobs (*i.e.*, users’ applications). A job is composed of one or multiple tasks [5]. Jobs are classified into four categories : *single-task*, *sequential-tasks*, *parallel-tasks* and *mix-mode-tasks*. Each task is a Linux program involving one or multiple processes. Every task or job has its own resources requirements in terms of *CPU*, *RAM* and *Disk Space*, and its own scheduling priority and constraints. The Google dataset contains six tables in CSV (Comma-Separated Values) format: *Task Event*, *Job Event*, *Machine Attribute*, *Task Constraint*, and *Task Usage* [5]. More details about the Google traces are described in [8].

B. Data Extraction and Processing

Figure 1 describes our proposed methodology to extract and analyse Google traces files. First, we parse the CSV files containing scheduling events and the resources usage of tasks and jobs. Then we extract the attributes describing the tasks/jobs. Next, we map the failure events of tasks to the failure events of jobs, to identify correlations between them. The remainder of this section elaborates more on each of these steps.

1) *Extraction of Tasks/Jobs Attributes*: We implemented a Java program to parse task (respectively job) events and usage files and extract useful attributes. For each task, we extract the following metrics: job ID; task ID; waiting time; service time; scheduling class; priority; requested and used *CPU*, *RAM* and *Disk Space*; number of previous dependent tasks that were finished, killed, failed, evicted, lost or unscheduled; number of times the task was rescheduled after being failed and the final status of the task. For each job, we extract the: job ID; waiting time; service time; scheduling class; number of finished, killed, failed, evicted, lost or unscheduled tasks within this job; total number of tasks composing the job and the final status of the job. Description of these attributes and our rationale for selecting them can be found in [8].

2) *Identification/Profiling of the Failed Tasks/Jobs*: To identify failed tasks/Jobs we look for one of the following status : failed, killed, lost, evicted and unscheduled. Tasks/jobs with a dependent task/job that was failed is consider to be failed. If there are some missing information in the files about a task/job’s final status, we consider that the tasks/job is lost.

3) *Mapping between Failed Tasks and Jobs*: Since jobs are composed of one or multiple tasks. We extracted the distribution of tasks within each job according to their final status finished, failed, killed, evicted, lost and unscheduled to analyse the correlation between tasks scheduling outcomes and job scheduling outcome.

III. CASE STUDY RESULTS

This section presents and discusses the results of our first two research questions:

RQ1: How often does a scheduled task or job is Failed, Evicted, Lost, or Killed?

Motivation: This question is preliminary to the others (*i.e.*, **RQ2** and **RQ3**). It aims to examine the proportion of failed, killed, evicted, lost, and unscheduled jobs that occurred in Google clusters over a period of one month. If these events are very frequent, then they are worth studying in more details. We also examine the waiting and service times of jobs (respectively tasks) in each category to evaluate the impact of task and job failures on processing costs.

Approach: We address this question by extracting information about unfinished tasks and jobs from our data set following the method described in Section II-B2. We used all jobs files. However, we experienced very long processing times (*i.e.*, lasting multiple days) when analysing the task files. We decided to reduce the amount of data to process by randomly sampling 2% (10 files out of 500) of the tasks files, in order to speed up our analysis. However, we verified the relevance of our sample by re-sampling the data-set multiple times (*i.e.* 5 times) and comparing the results of our analysis.

Findings: Only 58.47% of the submitted jobs were finished successfully and the rest were killed, failed, unscheduled or evicted as shown in Table I. We also observed that few jobs (*i.e.*, 0.8%) were not scheduled. Also, the number of killed task is very important (almost 40%) compared to the finished ones. Furthermore, we also found that failed and killed

TABLE I: Distribution of Jobs across Google Traces Files

Job Status	Nbr Jobs	%
Finished	379586	58.47%
Killed	255280	39.33%
Failed	9080	1.4%
Evicted	14	0.0%
Lost	0	0.0%
Unscheduled	5169	0.8%
Total	649129	100%

jobs are characterized by long waiting times, as described in Figure 2a. Meaning that a reduction of the amount of failed and killed jobs can help reduce processing times on clusters, which would result in energy and resources savings. We also noticed that killed and failed jobs have longer service time compared to other finished jobs, as shown in Figure 2b. Therefore, it is very important to identify the main reasons that lead to jobs failure in order to reduce their processing cost (in terms of service and waiting times) and consequently improve the cluster performance.

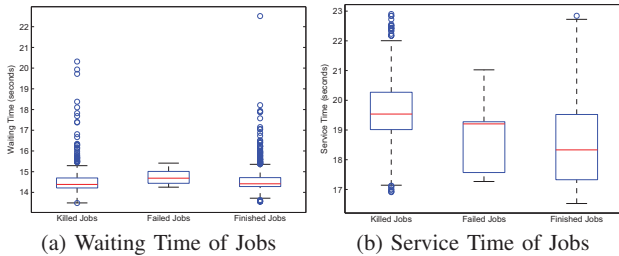


Fig. 2: Waiting and Service Time of Jobs (log scale)

TABLE II: Distribution of Task across 10 Google Traces Files

Task Status	Nbr Tasks	%
Finished	33020	52.22%
Killed	8473	13.40%
Failed	7044	11.14%
Evicted	12798	20.24%
Lost	0	0.0%
Unscheduled	1897	3%
Total	63234	100%

More than 24% of tasks were failed or killed and only 52% of tasks were finished successfully. However, 97% of tasks were scheduled successfully, *i.e.*, only 3% of tasks were not scheduled and were resubmitted for scheduling (some tasks were resubmitted up to 182 times). We also observed a high percentage of evicted tasks (*i.e.*, 20%) as shown in Table II. Evicted tasks had lower priority compared to production and monitoring tasks. In addition, we also observed that evicted tasks have long waiting time and service time compared to other tasks (see Figure 3a and Figure 3b). Also, failed and killed tasks are characterized by long waiting and execution time. Therefore, it is crucial to reduce the amount of failed and evicted tasks if we want to optimize jobs and tasks processing times (jobs are composed of multiple tasks). We repeated the analysis on all the other samples collected from the Google trace data and obtain similar results; suggesting that the observed high rates of job and task failures are not specific to the studied sample but rather likely representative of the general situation of jobs and tasks scheduling issues in Google clusters.

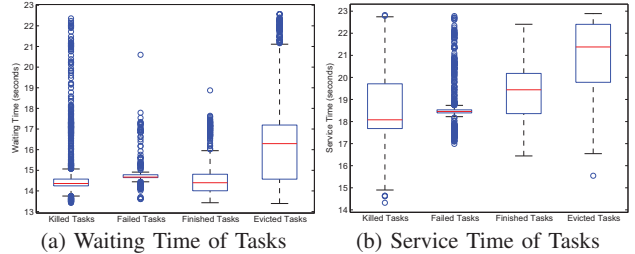


Fig. 3: Waiting and Service Time of Tasks (log scale)

RQ2: Can we predict the outcome of scheduling events based on cluster log files?

Motivation: In **RQ1**, we observed that scheduled tasks and jobs experience high failure rates. In this research question, we examine the correlation between the characteristics of tasks (respectively jobs) and scheduling outcomes, in order to predict tasks failures and eventually prevent their occurrence. Specifically, our goal is to determine whether the scheduling outcome of a task can be predicted early on before it actually happens. Such predictions can be used to reschedule potential failing tasks quickly on appropriate clusters with adequate resources in order to ensure their timely and successful completion.

Approach: We extract attributes of the jobs and tasks as described in Section II-B. We use the Spearman rank correlation [9] to test the association between these metrics

and task scheduling outcomes. We also perform a Variance Inflation Factor (VIF) analysis to examine multi-collinearity between the metrics. We use a threshold of 5 to decide on the multi-collinearity of the metrics, *i.e.*, metrics with VIF result greater than 5 are considered as correlated. We choose several regression and classification algorithms in R [10] to build models: GLM (General Linear Model), Random Forest, Neural Network, Boost, Tree and CTree (Conditional Tree). More details about these algorithm are presented in [8]. We use different training and testing data sets for both jobs and tasks. We apply 10-fold random cross validation to measure the accuracy, the precision, and recall of the prediction models [11]. In the cross validation, each data set is randomly split into ten folds. Nine folds are used as the training set, and the remaining one fold is used as the testing set.

Findings:

a) Job Level: We analysed the correlation between job attributes and jobs scheduling outcomes. We observed multi-collinearity between the following attributes: total number of tasks, service and waiting time and number of unscheduled/lost task (*i.e.*, VIF results were over 5). We found a strong correlation between the number of finished, failed, killed and evicted tasks within a job (with VIF values of respectively 1.87, 4.17, 3.65 and 2.42) and the final status of the job. Therefore we conclude that, when there are dependencies between the tasks composing a job, the scheduling outcome of the job is impacted by the scheduling outcome of its contained tasks. **Random Forest achieves the best precision and recall when predicting the outcome of job scheduling. It can achieve an average accuracy of 85.6%, a precision of 94.2% and a recall of 85.9%.** Table III summarises the performance of the six models. Only 10 job files mapped to the tasks files were used in the construction of the models because of our processing resources limitation. In fact, the 10 files contained 2, 594 jobs and 6, 3234 tasks which processing took 100 minutes (processing 10 cross-validations on one file took only 9.5 minutes).

TABLE III: Accuracy, Precision, Recall (In %)

# File	Algo.	Acc.	Pre.	Rec.
10	Tree	66.7	83.7	66.7
	Boost	75	89.1	75
	Glm	68.8	90.7	61.7
	CTree	61.8	89.7	54.9
	Random Forest	85.6	94.2	85.9
	Neural Network	56	67.6	66.6

b) Tasks Level: We analysed the relation between task attributes and scheduling outcomes and obtained a strong correlation between the number of previously finished, killed, failed and evicted tasks, priority and the scheduling outcome of tasks (having respectively these VIF values: 1.14, 1.02, 1.07, 1.03, 1.06). We observed multi-collinearity between the number of rescheduled tasks, service time, waiting time, and the amount of requested/used resources (*CPU*, *RAM*, *Disk*). In addition, we noticed that the resources assigned to each task were higher than the requested resources (which can be explained by the overbooking strategy followed by Google [12]).

Overall, tasks characterized by dependent tasks that failed in the past have a high probability to fail in the future. Also, tasks with low priority values have a high probability to be evicted [13]. When predicting failed tasks, Random Forest can achieve an average accuracy of **95.8%**, a precision of **97.4%** and a recall of **96.2%** (see Table IV). From Table IV, we also observe that results obtained with 1 file are quite similar to the results obtained with 10 files. We explain this result by the fact that the distribution of failure events in these files are very similar (as shown in **RQ1**).

TABLE IV: Accuracy, Precision, Recall (In %)

Algo.	Acc.		Pre.		Rec.	
	1 F.	10 F.	1 F.	10 F.	1 F.	10 F.
Tree	74	66.2	84.8	77	74.3	66.7
Boost	88.6	89.3	99.5	99.6	80.8	81.4
Glm	70.8	74.5	97	99.6	52.6	55.2
CTree	87.4	92.5	94.9	98	85.6	98.2
Random Forest	95.8	97.3	97.4	98.1	96.2	97.7
Neural Network	50	50	56.4	50	50	50

* F. = File

IV. RESULTS SCHEDULING OUTCOMES PREDICTION: GOOGLE CLUSTER

RQ3: Which benefits can be achieved by predicting the outcome of scheduling events?

Motivation: Results from **RQ2** show that a Random Forest model can predict task failure events with high precision (*i.e.*, 97.4%) and recall (*i.e.*, 96.2%). Therefore, rather than waiting for a scheduled task to fail, a scheduler equipped with such predictions can reschedule the tasks quickly on appropriate clusters with adequate resources. For example by restarting on a different node a task predicted to fail on its current node because of insufficient resources. To quantify the benefits that can be achieved by predicting the scheduling outcome of tasks early, we measure the execution time and numbers of finished tasks and jobs of a scheduler equipped with a Random Forest prediction model.

Approach: We used the simulation toolkit *GloudSim* to reproduce the execution traces from the dataset. Indeed, the *GloudSim* toolkit was developed to simulate the original workload of Google applications in order to support academic research [6]. We deployed *GloudSim* using 8 virtual machines (VMs) managed by a *XEN* hypervisor. Each VM had a one *Core(TM)2 Quad CPU* (*i.e.*, 2.66GHz) and 1024 MB of memory. We implemented a script to collect the following data about submitted tasks : priority, scheduling class, number of previous failure events and requested resources. We trained the random Forest model from **RQ2** using historical data generated by *GloudSim* about scheduled tasks and used it to predict the scheduling outcome of each new task submitted for scheduling. We used a real-time learning algorithm to update the scheduling policies at fixed time intervals (*i.e.*, every 10 minutes). Also, our proposed prediction algorithm can be used off-line to add the new learned scheduling rules periodically. We extend the scheduler implemented in *GloudSim* to integrate the Random Forest prediction model of tasks. If a scheduled task was predicted to fail, the new

scheduler would resubmit the task directly in the scheduling queue without executing it. Only tasks that were predicted to succeed would be processed on the scheduler. On the new scheduler, tasks that are predicted to fail are enqueued until they get a prediction of success. Consequently, if many tasks in a submitted job are predicted to fail, the execution of the job can take a long time and the job may even fail since the tasks will be rescheduled until they are predicted to succeed (which may not occur). We compare the scheduling performance of the new scheduler and the original scheduler implemented in GloudSim, by executing between 100 and 800 tasks and between 100 and 400 jobs. We considered three types of tasks and jobs during the comparison: single (100 tasks-100 jobs), batch (800 tasks-110 jobs) and mix (600 tasks-400 jobs). The performance of the schedulers were measured in terms of execution times and numbers of finished tasks and jobs. We choose these measures because the execution time of jobs and tasks are two important metrics that capture resource utilisation in the cluster. The number of failure events is a good measure of the quality of a scheduler.

Findings:

a) Job Level: **Overall, the number of finished jobs is increased and the number of failed jobs decreased when extending the GloudSim scheduler with our Random Forest prediction model. In addition, the execution times of the jobs were optimized (the number of rescheduling of failed jobs dropped, reducing the total execution time of the jobs).** The improvement is larger for batch jobs as shown on Figure 4a and Figure 4c compared to mix jobs as described in Figure 4b and Figure 4d. For single jobs, the number of finished and failed jobs is almost the same with and without prediction of tasks failure. We explain this result by the fact that our prediction model performs better when a job is composed of multiple dependant tasks; the number of failed dependant tasks and the number of killed dependant tasks are two main characteristics of job failure. In general, we conclude that prediction models of tasks can help reduce jobs failure rates because the job scheduling outcome is impacted by the scheduling outcome of its tasks. Moreover, the execution time of jobs was optimized for the batch and mix jobs which can be explained by the reduction of the number of failure events within these jobs, as shown in Figure 7c and Figure 7e. However, for single jobs, the execution time is the same with or without prediction (see Figure 7a). We attribute this outcome to the fact that the distribution of failure events is the same in the two configurations (*i.e.*, with and without prediction). Overall, reducing the number of failed tasks can help to avoid the starvation problem of tasks waiting on the queue until the successful processing of their dependent tasks, and long scheduling delays in cluster scheduler.

b) Task Level: **At task level, we also obtained a reduction of the number of failures and an increase of the number of successful execution when extending the GloudSim scheduler with our Random Forest prediction model.** Similar to jobs, batch tasks show the larger improve-

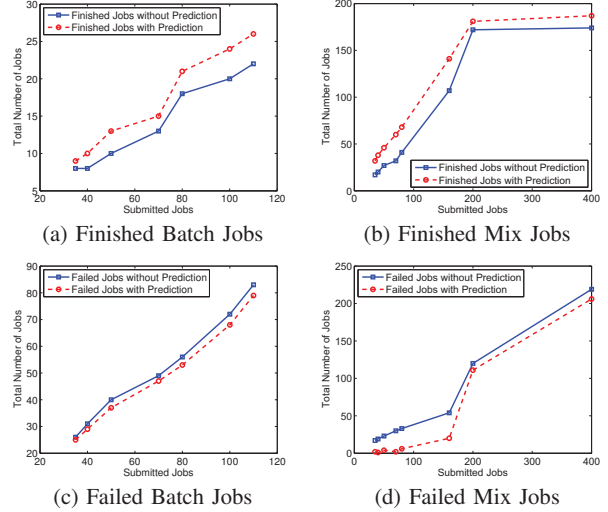


Fig. 4: Distribution of Finished and Failed Jobs

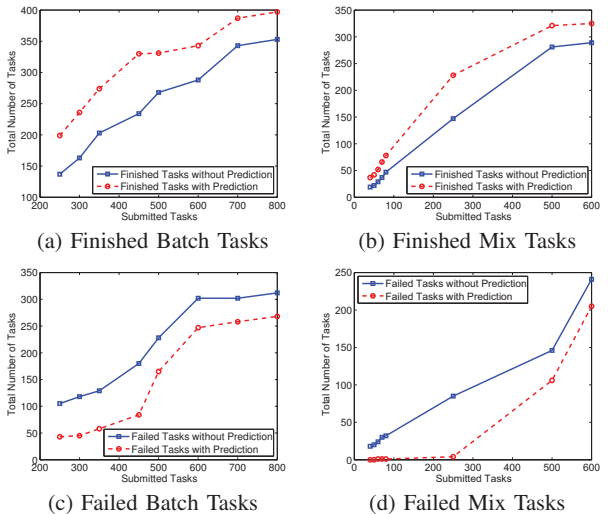


Fig. 5: Distribution of Finished and Failed Tasks

ments (see Figure 5a and Figure 5c in comparison to Figure 5b and Figure 5d). Single tasks show no improvements. Moreover, we noticed that the number of scheduled tasks was improved. This was expected since the prediction model enables the quick rescheduling of tasks that are predicted to fail. However, we noticed that the number of task failures is still high compared to the number of finished tasks. This is due to the fact that these tasks were failing because of other scheduling constraints (resources, task constraints, etc). Our rescheduling scheme was mainly based on dependencies between tasks but it can be extended to include those other constraints if they are reflected in training data. These failed tasks that we could not predict their failure affect the final scheduling outcome of the jobs. Furthermore, we observed that the execution time was optimized for batch and mix tasks as shown in Figure 7d and Figure 7f since the submitted tasks were processed and

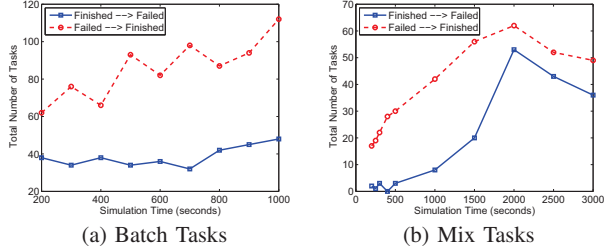


Fig. 6: Improvement of the Predictive Model: Task Level

finished without waiting for other submitted or queued tasks to be finished. We explain this improvement by the fact that the scheduler knew in advance which tasks should be scheduled first to ensure the successful processing of the tasks. The execution time of single tasks remained the same as presented Figure 7b. This is probably due to the fact that our model achieves good prediction, when information about previous killed, evicted and failed dependant tasks are available, which is not the case for single tasks.

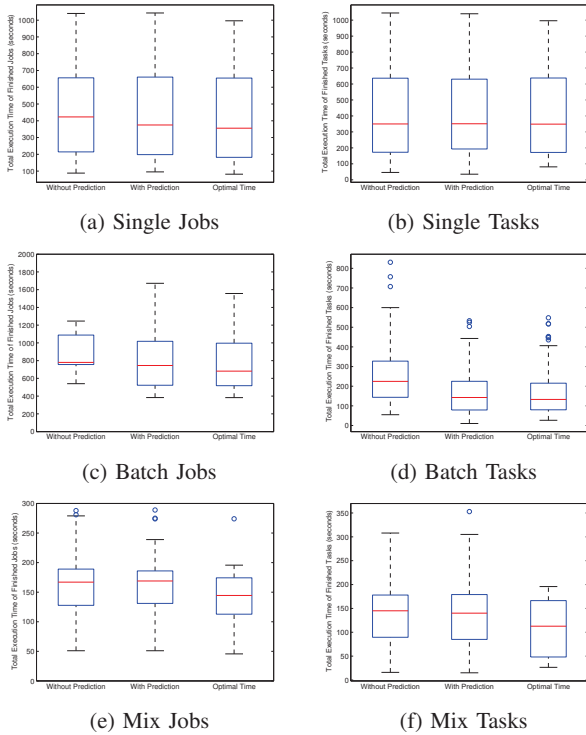


Fig. 7: Total Execution Time of Finished Jobs and Tasks

Moreover, we evaluated the improvement of our new scheduler (*i.e.*, the scheduler extended with our Random Forest model) by computing the number of tasks that were failed without prediction but succeeded (*i.e.*, their execution finished successfully) when scheduled with the new scheduler. We also computed the number of tasks that succeeded without prediction but failed when scheduled with the new scheduler. **Overall, the number of finished tasks was improved by**

40% when scheduling was done with the new scheduler as shown in Figure 6a and Figure 6b. 2% of tasks that succeeded without prediction failed when scheduled with the new scheduler. These failures are due to the false positive predictions of our prediction model. The model is not totally accurate.

V. APPLICATION: HADOOP ON AMAZON EMR

We implemented and deployed the prediction models from Section II on Amazon EC2, extending the standard scheduler of the Hadoop framework on Amazon EMR. To evaluate the performance of these extended schedulers, we selected an application used for gene expression correlations analysis in the context of Breast cancer research. The application is used to uncover factors causing breast cancer by identifying differential gene expressions between different conditions (*e.g.*, cancerous versus normal cells). This application was recently parallelized by Tzu-Hao *et al* [7], using the MapReduce programming model and deployed on Hadoop [14]. Since the application performs sensitive analysis, any job failure on Hadoop may lead to inaccurate information and wrong conclusions about the disease. The application is composed of 9 jobs [7] shown in Figure 8. These jobs are dependent and any failure occurrence may propagate inaccurate results and incur delays that will affect the total execution time of the application. We emulated the behaviour of this application by running the same flow (presented on Figure 8) using the *wordcount* example provided by Apache with Hadoop as job unit; linking the output of these dependent jobs together as on Figure 8, to obtain the final output.

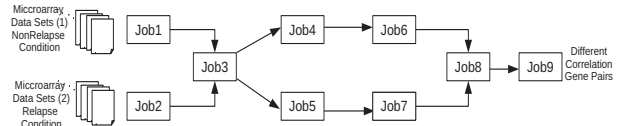


Fig. 8: Flow of Gene Expression Correlation Analysis

We ran the analysis on Amazon EMR using 4 machine instance of type *m3.large* ($ECPU=6.5$, $VCPU=2$, $MEM=7.5$ GB, $Instance\ Storage=32$, $Network\ Performance=Moderate$). The first machine was the master node submitting the jobs to two other machines considered as the workers. The last machine was the secondary master node. We ran different simulations on these machine instances to collect log files that we parsed to extract attributes described in Section II and train the prediction models. We compared the performance of the six models described in Section II and found that the best results are achieved with Neural Networks, *i.e.*, accuracy (72.8%), precision (97.2%) and recall (72.7%). We used different training and testing data sets when assessing the performance of the models. Table V summarizes the results achieved by the six models.

Using Hadoop's scheduler extended with the Neural Network prediction model, refreshing its scheduling policies every

TABLE V: Accuracy, Precision, Recall (In %)

Algo.	Acc.	Pre.	Rec.
Tree	41.9	84.1	42.7
Boost	38.1	72.8	39.2
Glm	26.5	77.8	26.0
C'Tree	61.8	89.7	54.9
Random Forest	24.0	69.9	23.7
Neural Network	72.8	97.2	72.7

5 minutes, we performed different simulations by submitting jobs to worker nodes and injecting an early failure on Job1, a late failure on Job8 and 2 mixed failures (late and early) on Job1 and Job8. For each of these simulations, we measured the total execution time and the total number of failed jobs and compared the obtained results with those of the default Hadoop scheduler. We also measured the execution time of the jobs for different numbers of learning iterations.

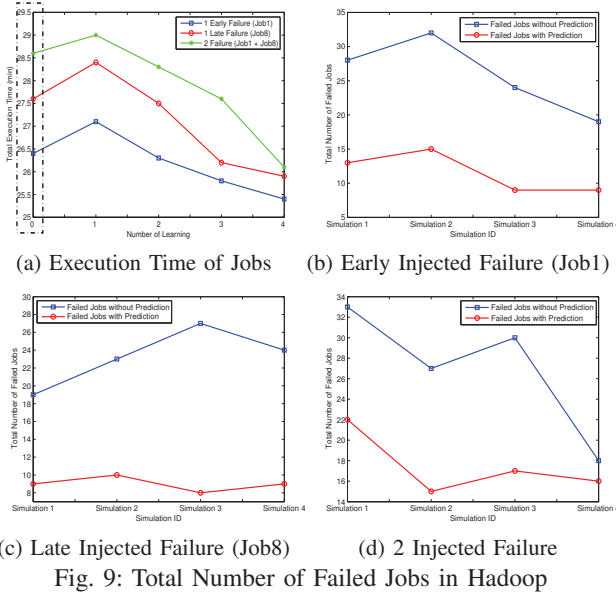


Fig. 9: Total Number of Failed Jobs in Hadoop

The Hadoop’s scheduler extended with the Neural Network prediction model can reduce the number of job failures by up to 45% (see Figure 9b, Figure 9c and Figure 9d). In our proposed scheduling scheme, if the job satisfies its dependency requirement, it will be submitted to the worker to be executed or it will be rescheduled and restarted from the beginning to ensure its successful completion. However, we noticed that there are still some jobs which are still failing although they were predicted to be completely finished. This is due to the failure related to cluster environment (errors occur while executing the job, insufficient resources, long running tasks, etc). Moreover, the execution time of jobs is significantly reduced with more learning (see Figure 9a). We explain this results by the fact that more learnings (*i.e.*, more trainings of the model on larger historical data) improve the performance of the prediction model, enabling it to reschedule earlier multiple jobs that would have failed.

VI. THREATS TO VALIDITY

This section discusses the threats to validity of our study following the guidelines for case study research [15].

Construct validity threats concern the relation between theory and observation. Our modelling approach assumes that tasks and jobs characteristics alone can explain scheduling outcomes, when in reality, this may not be the case. It is possible that other factors such as scheduling class or resources allocation strategy also play a role in scheduling decisions. However, in our data set we found a low correlation between scheduling class and scheduling outcomes. According to [5], this low correlation is due to the fact that the scheduling class which represents the latency-sensitivity of a task/job mostly affects local machine policies for accessing cluster resources. They are therefore more likely to affect resource usages than scheduling outcomes.

Internal validity threats concern our selection of subject systems, tools, and analysis method. Although the Google data set used in this study may not contain all the different kinds of task and jobs used in the industry, it represents the execution of real applications from a major company (*i.e.*, Google). The GloudSim scheduler used in **RQ3** does not represent of all existing schedulers in the industry. However, it is designed to reproduce the scheduler used in some Google clusters.

External validity threats concern the possibility to generalize our results. Our study is based on large-scale data (*i.e.*, 158 GB) collected from Google clusters. Nevertheless, further validation on larger and diverse sets of tasks and jobs is desirable.

VII. RELATED WORK

There is a large body of research that aimed to characterize the task and jobs contained in the Google cluster traces used in this paper. We classify these works into the following categories.

A. Scheduling Characterization

The characterization of scheduling events has been the focus of many workload analysis studies. Recently, [16] addressed the batch jobs scheduling in distributed data centres and proposed GreFar to optimize the energy cost and fairness across different clusters which are characterized by scheduling delays constraints. Zhang et al. used the Google cluster data to propose Harmony, a heterogeneity-aware framework that can minimize scheduling delays and the total energy consumption by controlling the number of machines that are provisioned [13] [17]. The performance of Harmony was found to be better than GreFar [16]. In [18], Sharma et al. showed that task placement have a large impact on scheduling delays: task waiting time can be increased by a factor of 2 to 6 due to the cluster and task constraints. They proposed a methodology that takes into account resources requirement and task placement.

B. Failure Analysis and Prediction

Failure analysis and prediction have become popular in researches on distributed systems, since they allow for early identification of failure and can improve the performance of the cluster. Fadishei et al. [19] used the Grid Workload Archive project to analyse the correlation between job failures and resources attributes (e.g., resources utilisation and scheduler characteristics). They found that scheduler load, execution hour of day and CPU-intensity are among the most factors that can affect failure rates. Ganesha was proposed by Pan et al [20] as a black-box tool to identify failures between faulty and normal nodes in MapReduce. Xin Chen et al. used Google trace files to identify and predict jobs failure in batch applications. They used Recurrent Neural Networks to perform their predictions. This model was able to reduce resources utilisation by between 6% and 10% [3] [4]. They recommend that predicted failed tasks be killed immediately without processing, in order to avoid wasting resources. However, killing predicted failed tasks is likely to affect the overall performance of a cloud application. A better decision would be to reschedule the tasks quickly on appropriate clusters with adequate resources. To the best of our knowledge, our work is the first that proposes an approach to predict and reschedule failed tasks in order to improve the performance of cloud systems. In addition, our work evaluates the performance of many statistical models in predicting failed tasks. Also, we show that Random Forest achieves better results compared to Neural Networks, both in terms of precision and recall.

VIII. CONCLUSION

Task scheduling is an important issue that greatly impacts the performance of cloud computing systems. In this paper, we examined task failures in Google clusters data and found that 42% of the jobs and 40% of the tasks were not finished successfully. We noticed that a job often fails because of the failures of some of its tasks, and tasks also fail because of the failure of dependent tasks. We investigated the possibility of predicting the scheduling outcome of a task using statistical models and historical information about the execution of previously scheduled tasks and found that Random Forest models can achieve a precision up to 97.4%, and a recall up to 96.2%. We also extended the schedulers implemented in GloudSim and Hadoop to incorporate task failure predictions; the goal being to achieve early rescheduling of potential failed tasks (*i.e.*, early on before their actual failing time). We compared the scheduling performance of the new scheduler and the original scheduler implemented in GloudSim, in terms of execution times and numbers of finished tasks and jobs, and found that the number of finished tasks (respectively jobs) can be increased by up to 40 % (respectively 20 %) and the execution time reduced by the new scheduler. In the case of Hadoop, the new scheduler can reduce the number of job failures by up to 70% with an overhead time of less than 5 minutes. Cloud service providers could improve the performance of their task scheduling algorithms by extending them with our proposed failure prediction models. Since the extra

layer of prediction can have an impact on the performance of cloud applications (*i.e.*, training and applying the proposed prediction model can cause delays in scheduling decisions), although we found it to be less than 5 minutes in our case study on Hadoop, in future work, we plan to examine in details the trade-offs between precision and execution time when selecting a prediction model, as well as the frequency at which the predictions should be performed in order to ensure optimal scheduling response times.

REFERENCES

- [1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Communications ACM*, 51(1):107–113, (2008).
- [2] W. Zhao, Y. Peng, F. Xie, and Z. Dai, "Modeling and simulation of cloud computing: A review," in *IEEE Asia Pacific Cloud Computing Congress (APCloudCC)*, pages 20–24, (2012).
- [3] X. Chen, C.-D. Lu, and K. Pattabiraman, "Failure analysis of jobs in compute clouds: A google cluster case study," in *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*, pages 167–177, (2014).
- [4] —, "Failure prediction of jobs in compute clouds: A google cluster case study," in *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 341–346, (2014).
- [5] Traces of google workloads. [Online]. Available: <http://code.google.com/p/googleclusterdata/>, (Last Access April,2015).
- [6] F. C. Sheng Di, "Gloudsim: Google trace based cloud simulator with virtual machines," *Journal of Software Practice and Experience*, (2014).
- [7] T.-H. C., S.-L. W., W.-J. W., J.-T. H., and C.-W. Ch., "A novel approach for discovering condition-specific correlations of gene expressions within biological pathways by using cloud computing technology," in *BioMed Research International*, (2014).
- [8] M. Sualhia, F. Khomh, and S. Tahar, "Predicting scheduling failures in the cloud," Department of Electrical and Computer Engineering, Concordia University, Tech. Rep., 2015. [Online]. Available: <http://arxiv.org/abs/1507.03562>
- [9] M. Cortina-Borja, "Handbook of parametric and nonparametric statistical procedures," (2012).
- [10] The r project for statistical computing. [Online]. Available: <http://www.r-project.org/>, (Last Access April,2015)
- [11] B. Efron, "Estimating the error rate of a prediction rule: improvement on cross-validation," *Journal of the American Statistical Association*, 78(382):316–331 (1983).
- [12] F. Caglar and A. Gokhale, "ioverbook: intelligent resource-overbooking to support soft real-time applications in the cloud," in *7th IEEE International Conference on Cloud Computing (IEEE CLOUD)*, (2014).
- [13] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein, "Harmony: Dynamic heterogeneity-aware resource provisioning in the cloud," in *IEEE 33rd International Conference on Distributed Computing Systems (ICDCS)*, pages 510–519 (2013).
- [14] Amazon elastic mapreduce (amazon emr). [Online]. Available: <http://aws.amazon.com/elasticmapreduce/>, (Last Access April,2015).
- [15] R. K. Yin, *Case Study Research: Design and Methods - Third Edition*, 3rd ed. SAGE Publication, (2002).
- [16] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*, pages 22–31, (2012).
- [17] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Transactions on Cloud Computing*, 2(1):14–28, (2014).
- [18] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das, "Modeling and synthesizing task placement constraints in google compute clusters," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, pages 1–14, (2011).
- [19] H. Fadishei, H. Saadatfar, and H. Deldari, "Job failure prediction in grid environment based on workload characteristics," in *14th International CSI Computer Conference CSICC*, pages 329–334, (2009).
- [20] X. Pan, J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan, "Ganesha: Blackbox diagnosis of mapreduce systems," *SIGMETRICS Perform. Eval. Rev.*, 37(3):8–13, (2010).