Identifying Execution Anomalies for Data Intensive Workflows Using Lightweight ML Techniques

Cong Wang*, George Papadimitriou†, Mariam Kiran‡, Anirban Mandal*, Ewa Deelman†

*RENCI, University of North Carolina at Chapel Hill, Chapel Hill, NC
†Information Sciences Institute, University of Southern California, Marina Del Rey, CA
‡Lawrence Berkeley National Laboratory, Berkeley, CA

***Abstract*—Today's computational science applications are increasingly dependent on many complex, data-intensive operations on distributed datasets that originate from a variety of scientific instruments and repositories. To manage this complexity, science workflows are created to automate the execution of these computational and data transfer tasks, which significantly improves scientific productivity. As the scale of workflows rapidly increases, detecting anomalous behaviors in workflow executions has become critical to ensure timely and accurate science products. In this paper, we present a set of lightweight machine learning-based techniques, including both supervised and unsupervised algorithms, to identify anomalous workflow behaviors. We perform anomaly analysis on both workflow-level and task-level datasets collected from real workflow executions on a distributed cloud testbed. Results show that the workflow-level analysis employing k-means clustering can accurately cluster anomalous, i.e. failure-prone and poorly performing workflows into statistically similar classes with a reasonable quality of clustering, achieving over 0.7 for Normalized Mutual Information and Completeness scores. These results affirm the selection of the workflow-level features for workflow anomaly analysis. For task-level analysis, the Decision Tree classifier achieves >80% accuracy, while other tested classifiers can achieve >50% accuracy in most cases. We believe that these promising results can be a foundation for future research on anomaly detection and failure prediction for scientific workflows running in production environments.**

I. Introduction

Today's computational science applications are increasingly dependent on many complex, data-intensive operations on distributed datasets that originate from a variety of scientific instruments and repositories. To manage this complexity, science workflows are created to automate the execution of these computation and data transfer tasks, which significantly improves the productivity of large scale computational sciences. Scientific workflow management systems, such as the Pegasus Workflow Management System (WMS) [8], are critical automation components that enable efficient workflow execution across heterogeneous HPC infrastructures. Although such systems provide some mechanisms to move past errors when possible (e.g. retries), it remains challenging to perform a thorough analysis of performance anomalies and detection and diagnosis of errors in workflow executions. The problem of unexpected or anomalous behavior during workflow execution is exacerbated by the use of complex distributed cyberinfrastructure that often encounters both performance problems and faults/errors that potentially span all levels of the system—applications, middleware, and the underlying execution platform. As the scale of science workflows rapidly grows, it becomes urgent to find appropriate mechanisms to detect and pinpoint the source of bottlenecks that impair normal workflow executions [7].

The performance and reliability of complex science workflows can be affected by many factors, which may lead workflows to fail. Workflow configuration errors, network congestion and packet loss, compute node I/O, disk and RAM slow down, all negatively impact the performance of a workflow run. While end-to-end monitoring of workflow applications and systems is an essential building block to detect such problems, current techniques for anomaly detection are often based on thresholds, moving averages, or univariate statistical analysis [13], [9] that can't capture interactions between performance features, or using rule-based classification methods, e.g., identify workflow failures using execution time, file transfer throughput, etc [20]. With ever-increasing complexity, today's large scale workflows have brought significant challenges to such traditional failure detection mechanisms that are univariate or are based on simple rules. Hence, multivariate techniques, in particular machine learning (ML) algorithms, are needed for building failure models and for detecting and diagnosing failures in large-scale workflow executions on complex systems.

In this paper, we seek to characterize the performance of science workflows, and to explore ML methods to find early signs of anomalous behaviors that may cause the workflows to fail. Our work looks into workflow anomaly detection using a multi-level approach: overall workflow level and sub-workflow (task) level analysis. The workflow-level performance analysis uses high-level, aggregate workflow performance metrics, such as the proportion of failed tasks in the workflow, to predict the overall behavior of a running workflow by clustering statistically similar workflows into classes. The task-level analysis detects faults and bottlenecks using classification approaches that leverage detailed task-level metrics such as resource usages and data sizes, to predict task execution failures.

Specifically, in this paper, we make following contributions:

- We characterize the performance features of scientific workflow execution aided by an end-to-end, streamlined

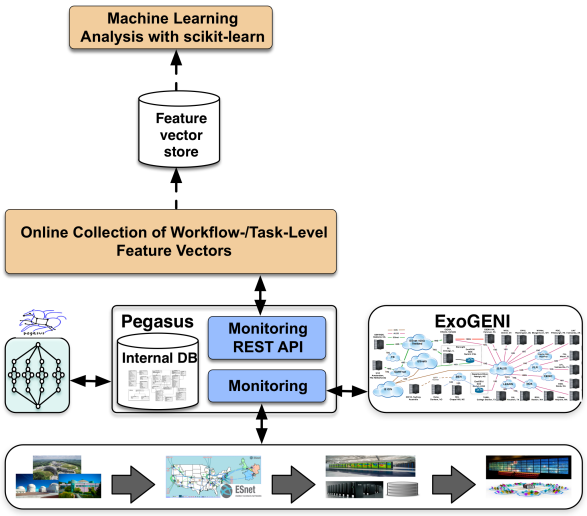


Fig. 1: The Panorama 360 workflow analysis framework.

data collection infrastructure [3] integrated with Pegasus; it provides the dataset for multi-level analysis.

- We develop and test multiple machine learning-based methods, including K-Means, Naive Bayes, Decision Tree, and Isolation Forest, to study workflow-level and task-level characteristics and anomalous behaviors.
- We make innovative use of an experimental testbed to train and develop our machine learning methods through systematic injection of faults and performance anomalies.
- We test anomaly detection methods using data collected from a real science workflow — the 1000 Genome workflow, which produces a reference for human gene variation, having reconstructed the genomes of 2,504 individuals across 26 different populations [4].
- We perform an in-depth analysis of findings and issues from the machine learning algorithms, laying the foundation for further research on anomaly detection and failure prediction for scientific workflows.

II. Methodologies

Our goal in this work is to capture anomalous workflow behavior in existing workflow runs and use such data as training data for future anomaly detection and failure prediction. This section describes the methodologies for collecting the sample data, as well as machine learning methods for extracting underlying features.

A. The Panorama 360 workflow analysis framework

We use the Panorama 360 workflow performance analysis framework [3] for collecting the workflow execution data. Figure 1 depicts the general data collection architecture. The Panorama 360 framework uses Pegasus to automate the workflow execution on both production HPC environments (such as Summit [17]), as well as on testbed environments (such as ExoGENI testbed [6]). The workflow performance data is collected by Pegasus online monitoring services and stored in an internal database. At 30-second intervals, an online module collects data using the Pegasus REST API and stores the required features for the machine learning analysis. Finally, we perform clustering and classification on the captured performance data for multi-level workflow performance analysis.

Metric	Description
J_s	jobs_succeeded/jobs_completed
J_f	jobs_failed/jobs_completed
t_s	Sum(duration(jobs_succeeded))/jobs_succeeded
t_f	Sum(duration(jobs_failed))/jobs_failed
o_j_s	jobs_succeeded/total_workflow_jobs

TABLE I: Workflow level metrics and descriptions.

Metric	Description
Utime	User level CPU usage
Bwrite & Bread	Bytes written to/read from disk
IOwait	CPU idle time waiting for I/O
Wchar & Rchar	Characters read/write

TABLE II: Task level metrics and descriptions.

B. Workflow performance analysis

Figure 2 summarizes the overall architecture of our machine learning analysis, which contains two aspects: 1) workflow-level analysis that predicts overall behavior of running workflow by clustering statistically similar workflows, and 2) task-level analysis for detecting faults and bottlenecks using task-level metrics to predict task-level execution failures. Both analysis methods use the monitoring data collected by Pegasus, to be used in the respective machine learning models.

1) Workflow-level analysis: Table I summarizes the overall workflow execution statistics, i.e. the relevant features used for workflow-level ML analysis, which are similar to the ones used in [20]. These features are calculated from different values stored in the internal Stampede database and are indicative of the overall progress of the workflow toward a failed or successful state. For example, the *J_s* and *J_f* features are measures of the current proportions of workflow jobs that have succeeded or failed. The *t_s* and *t_f* features are measures of average duration of successful and failed jobs. We hypothesize that using these features will help uncover patterns in successful vs. failure-prone workflows, and part of this work is to verify this hypothesis and to understand whether these features should indeed be selected for workflow failure prediction.

Clustering analysis. In the workflow-level analysis, we make use of the workflow-level features to predict the overall behavior of running workflow by clustering the statistically similar workflows. We use the K-means [23] clustering algorithm, which is a commonly used unsupervised clustering algorithm, a fast iterative algorithm for partitioning the input feature vectors into *k* clusters. The K-means algorithm clusters data by trying to separate samples in groups, such that the group members are close to each other for the nearest mean that minimizes within-cluster variances (squared Euclidean distances). K-means requires proper parameter selection in regards to the number of clusters to be partitioned. It also has good performance when processing a large number of

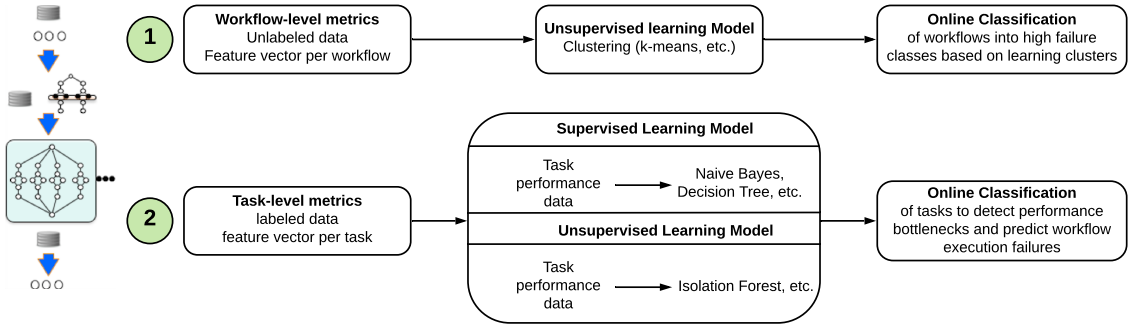


Fig. 2: Workflow- and task-level anomaly detection.

samples, which makes it useful for a variety of applications in many different fields.

In our case, the input to the clustering algorithm was a set of feature vectors of the form: $\{J\_s, J\_f, t\_s, t\_f, o\_j\_s\}$. Each feature vector corresponds to one execution of a given workflow, thereby characterizing the overall performance of that workflow at a very coarse level of granularity. The output of the clustering algorithm partitions the set of these feature vectors into multiple clusters with statistically similar feature vectors belonging to the same cluster.

2) Task-level performance analysis: The task-level analysis uses real-time performance data to characterize the performance of individual workflow tasks, which is in contrast to the coarse performance characterization of the entire workflow as described in the last section. The task-level analysis captures early signs of anomalous behavior and ultimately predicts workflow failures. We make use of three machine learning classifiers for the failure prediction: Naive-Bayes, Decision Tree, and Isolation Forest. The input to the classifiers was a set of feature vectors, summarized in Table II: $\{$*Utime, Bwrite, Bread, IOwait, Wchar, Rchar*$\}$. These features were collected from low-level Linux statistics and processed through the above Pegasus data collection framework.

Naive Bayes. Naive Bayes classifiers [21] are a family of supervised, probabilistic classifiers, for predicting the likelihood of failures. This classifier uses Bayes's theorem and assumes that all variables are independent considering the value of the class variable. This algorithm can efficiently learn a variety of controlled classification problems with high accuracy.

Decision Tree. Decision Tree [19] is a supervised learning classification algorithm that recursively partitions the data sets into a rule space, and makes decisions based on rules learned from the labeled training dataset. The objective is to partition the samples based on the data into the purest samples possible. This measurement of purity is determined by the Gini index or entropy function.

Isolation Forest. Isolation Forest [14] is an unsupervised classification algorithm that is commonly used for outlier detection in high-dimensional data sets. Isolation Forest is based on the fact that anomalies are data points that are usually fewer in number and have different distribution patterns from the normal data. Usually, random partitioning produces shorter paths for anomalies. Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies.

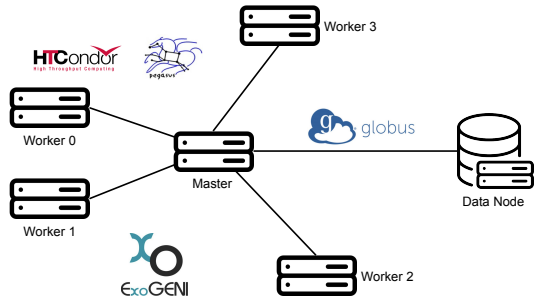


Fig. 3: Experimental setup on the ExoGENI testbed.

III. Experiment Setup

Testbed infrastructure. We deployed the workflows on the ExoGENI testbed [6], a cloud testbed with sites located across the US and connected via research networks such as Internet2 [2] and ESNet [1], through their programmable exchange points. By using an isolated, controlled testbed environment, we ensure minimal system interference and generate synthetic anomalies for evaluation purposes.

Figure 3 shows the experiment set up. Our setup consists of one data node, one master node, and four compute (worker) nodes. Each node has four 2.2 GHz vCPUs and 10 GB RAM. A shared file system (NFS) is hosted on the master node, to speed up the data sharing among the master node and the compute nodes.

Science workflow. We use the 1000 Genome workflow for performance evaluations. The 1000 Genomes project provides a reference for human variation, having reconstructed genomes of 2,504 individuals across 26 different populations [4]. The 1000 Genome workflow used in this paper is composed of 52 tasks, identifying mutational overlaps in data from the 1000 genomes project, for statistical evaluation of potential disease-related mutations.

Dataset. We perform clean runs to mimic normal workflow executions, and use synthetically injected anomalies to mimic real workflow failures and performance degradation. The synthetic anomalies are generated by (1) randomly terminating some of the tasks to mimic workflow failures caused by improper configurations or other factors, and (2) using Linux

Experiment	Total Samples
Clean	30
Failure injection (level high)	50
Failure injection (level low)	50
Stress on CPU, RAM, HDD	10 each
High stress (with CPU, RAM and HDD)	10

TABLE III: Workflow-level Samples Collected.

Experiment	Total Samples
Clean	1352
Stress on CPU (on 1, 2, 3 workers)	2496
Stress on I/O (on 1, 2, 3 workers)	2496
Stress on HDD (on 1, 2, 3 workers)	2496

TABLE IV: Task-level Samples Collected.

stress [25] tool to slow down the workflow execution and therefore cause timeout failures. Stressing includes CPU, I/O, disk, and memory to mimic interference in real workflow execution in shared infrastructure.

We conducted 170 workflow runs for each of the workflow-level and task-level analysis. For workflow-level analysis, we collect data, summarized in Table III, from 30 clean runs, 100 runs with synthetic task failures at two failure levels (low and high, 50 runs each), and 40 runs with stress at two levels (10 runs of high stress and 30 runs of low stress of CPU, memory and disk with 10 runs for each resource type).

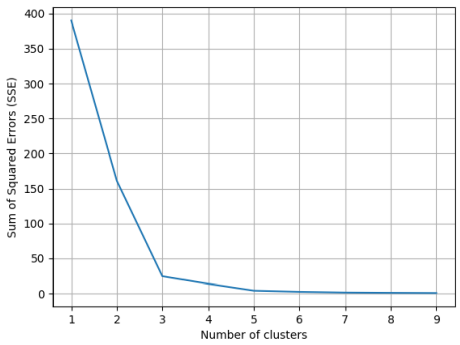
Since synthetically terminated tasks pose little logical relationship with the actual task failure, for task-level failure prediction, we introduce anomalies by stressing the systems. In this case, we collected data, summarized in Table IV, from 26 clean runs, and 144 runs with stress under 9 different anomaly scenarios, where each workflow contains 52 tasks. In total, the task level analysis dataset captured 8840 tasks. For task-level classifiers, we use the clean run data as the *training* data, and the workflow runs with stress anomalies as *testing* data for performance evaluations. The model is trained individually for each type of anomalies. In this case, training data contains 1352 samples, each type of anomaly (testing case) contains 832 samples.

IV. Results and Analysis

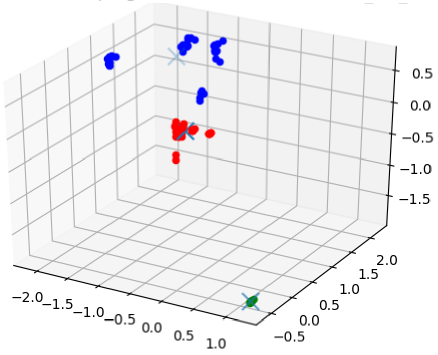
This section presents detailed evaluation of the workflow- and task-level anomaly and failure identification. We herein present the results and analysis of the experiments using the setup described in the previous section.

A. Workflow-level analysis

Workflows with failure injection. For workflow-level performance analysis, we use k-means clustering, aiming to find common features by clustering statistically similar workflows. The input was the set of feature vectors corresponding to clean runs and runs with synthetically injected failures. We selected *J_s, t_s* and *t_f* as the features used for this clustering analysis. For k-means, the parameter k is known to be challenging to choose when not given by external constraints. Therefore, our first step was to find the optimal number of clusters. Figure 4a shows the optimal number of clusters for this set of



(a) Finding optimal number of clusters.



(b) K-means clustering.

Fig. 4: Clustering for workflows with failure injection.

feature vectors by comparing the sum of squared errors (SSE) among the different numbers of clusters, i.e., k. We look for the elbow where we get diminishing returns by increasing k. We can observe that the sum of squared errors decreases significantly until near-optimal in the case with 3 clusters. Figure 4b shows the actual k-means clustering with 3 clusters. The x, y, and z axes represent value ranges for scaled features selected in this analysis. The green dots represent the clean workflow runs. The red and blue dots represent the workflow runs with failure injection, corresponding to low and high failures. Figure 4b shows that all clean workflow runs can be easily distinguished from workflows with synthetic injected failures, justifying the k value selection from Figure 4a.

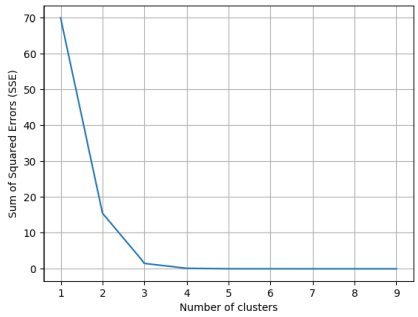
We evaluated the classifier performance by comparing it with information from domain knowledge. In this case, since we injected synthetic failures, we knew which feature vectors corresponded to clean runs and which ones corresponded to the two levels of failure. Mapping the feature vectors to clusters, we calculated three metrics to measure the quality of clustering algorithms, namely *Normalized Mutual Information (NMI) score* [15], *Completeness score* [16], and *Calinski Harabaz (CH) score*. For all three scores, a higher value represents better clustering performance. The first two scores have values between 0 and 1. An NMI or Completeness score of 1 indicates that the clustering of the feature vectors as obtained by the clustering algorithm completely matches the manual mapping. The first column in Table V shows our scores for workflows with injected

Metric	clustering with failure injection	clustering with stress
Normalized mutual info score [0, 1]	0.727	0.697
Completeness score [0, 1]	0.739	0.714
Calinski Harabaz score	951.57	1513.60

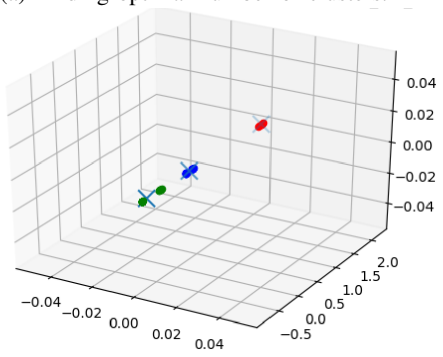
TABLE V: Workflow-level clustering performance summary.

failures. An NMI/Completeness score of 0.73 indicates that most of the workflows were clustered into the "right" cluster.

From these results, we can infer that the workflow-level features chosen for the clustering analysis were appropriate to differentiate failure-prone workflows from clean workflows, and could reasonably differentiate high failure ones from low failure ones. Hence, these features should be extracted for failure detection for workflows using other machine learning-based methods.



(a) Finding optimal number of clusters.



(b) K-means clustering.

Fig. 5: K-means clustering for workflows with stress.

Workflows with stress. Similar observations hold for workflow runs with stress, where we introduce synthetic loads for CPU, disk, and memory to emulate the interference in real shared infrastructures. The input was a set of feature vectors corresponding to clean runs and runs with synthetically injected stress functions. Figure 5a shows that the elbow k happens when $k = 3$, which produces near-optimal SSE. Figure 5b shows the clustering results, where green dots represent a cluster of clean runs, and the rest represents runs with different levels of stress. As in the previous analysis, the various scores of the accuracy of the clustering algorithms are shown in the second column of Table V. In this case, with an NMI/Completeness score of 0.7, we observe that most of the feature vectors were mapped to the "right" cluster. The reason that the score was not 1 was that the clustering algorithm mapped one set of feature vectors corresponding to low-stress levels into the normal cluster. These results also reaffirm our choice of the workflow-level features to distinguish anomalous workflows from normal ones and that they should be used for anomaly detection algorithms.

B. Task-level analysis

For the task-level analysis, we compare the performance between Naive Bayes, Decision Tree and Isolation Forest, where the first two are supervised learning, and Isolation Forest is an unsupervised learning algorithm. We introduce various CPU, I/O and hard-disk anomalies on 1, 2, and 3 worker nodes, respectively, aiming to introduce synthetic task failures due to extended run time errors.

Figure 6 summarizes the overall task execution status, where x-axis shows different stressing scenarios and y-axis shows the percentage of failed, succeeded, and unsubmitted tasks. We disabled the auto-retry feature from the workflow, for evaluation purposes. Therefore, the remaining tasks will not be submitted for execution if their dependency tasks have failed. Stressing on CPU produces a similar number of task failures, with stress on different workers. The success rate of I/O stress decreases when more workers are being stressed. Finally, stressing on hard-disk produces the most number of failures, especially for the case with stress on 3 workers. This is because 1000Genome is a data-intensive workflow; slow hard-disk significantly affects workflow performance.

We evaluate the performance of classifiers using two commonly used metrics: *precision* and *balanced accuracy*. The precision (PPV) is evaluated by:

$$PPV = \frac{TruePositive}{TruePositive + FalsePositive}. \quad (1)$$

The balanced accuracy (BACC) is evaluated by:

$$BACC = \frac{TruePositive}{Positive} + \frac{TrueNegative}{Negative}. \quad (2)$$

Figure 7 shows the performance of the three classifiers, Naive Bayes (NB), Decision Tree (DT), and Isolation Forest (IF), across different anomaly scenarios. The Decision Tree classifier has the best precision, over 90%, in most of the cases. The Naive Bayes classifier also performs well, with over 70%, for scenarios with CPU and I/O anomalies. However, NB cannot produce correct prediction when hard-disk anomalies were introduced. Isolation Forest provided moderate precision (>50%) for all the cases.

Similarly, we observe that, for balanced accuracy evaluation shown in Figure 8, the Decision Tree performs best compared to others. Naive Bayes performs better in the cases with hard-disk anomalies, compared to precision evaluation. This is because Naive Bayes predicts more True Negatives, which is included in this metric. We also observe that Isolation Forest has decreased balanced accuracy when anomalies were introduced on more worker nodes.

In summary, for task-level analysis, we see a better overall performance from Decision Tree and moderate performance

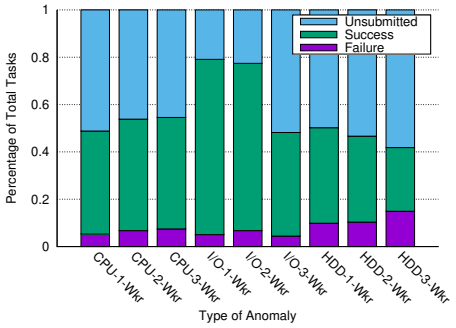


Fig. 6: Task execution summary.

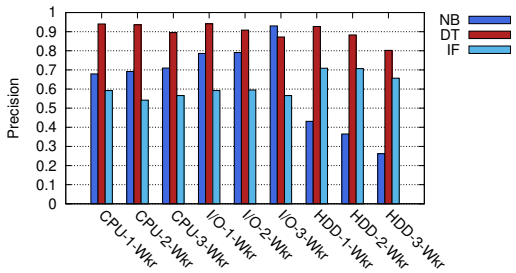


Fig. 7: Classifier precision.

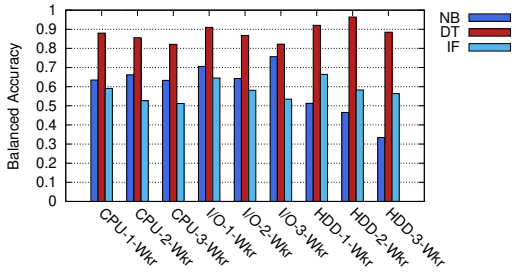


Fig. 8: Classifier balanced accuracy.

from Isolation Forest. Note that Isolation Forest is the only unsupervised classifier we evaluated in this paper, and unsupervised algorithms can be especially useful in real studies when we lack the ground truth labels in the training data.

V. Related Work

Science workflows often leverage a collection of many elements including compute, storage and networks. It has been brought to increasing attention to use machine learning to characterize the workflow behavior, in many aspects, such as execution timeline, data transfer performance etc. In a recent paper, Singh et al. [22] presented a machine learning framework that predicts workflow performance and forecasts workflow behavior. This work presented results on the XSEDE SDSC Comet cluster, showing that independent workflows still show the dependency of multiple components which are beneficial for predicting overall workflow performance. On the other hand, Herbst et al. [12], [11] presented the acquisition of workflow models and their adaptation to changing requirements. They proposed two machine learning methods based on the induction of hidden Markov models for workflow management systems.

Although there has been previous work on clustering workloads and workflows to identify failure patterns [5], [10], [20], [24], our work makes innovative use of testbed infrastructure for fault and stress injections to enable deterministic attribution of domain mapping and hence a more thorough clustering evaluation with realistic workflow use case. Gaikwad et al. [9] presented an anomaly detection method based on auto-regression and time-series prediction to understand how workflow anomalies can be detected on cloud infrastructures.

Feature extraction can also be used to investigate user behavior with traffic models [18]. Zhang et al. [26] described anomaly detection as a significant step for network intrusion detection systems to work, but highlighted that these systems are based on known or supervised data sets. The authors recognize that a more robust unsupervised feature extraction method is needed, with features learned from real network data sets, to make the systems more reliable.

In this paper, we focus on using both supervised and unsupervised machine learning methods, based on domain knowledge, to extract certain characteristics from known workflow-level and task-level datasets. The collected feature filters can be leveraged for any future machine learning methods. With the goal of understanding data intensive workflow performance, our results show positive results with the selected feature extraction methods.

VI. Conclusions

Machine learning is a powerful technique to identify behavioral characteristics of scientific workflows on distributed infrastructures.In this paper, we presented a set of lightweight machine learning techniques, including both supervised and unsupervised algorithms, to identify anomalous workflow executions. We performed anomaly detection analysis on both workflow-level, using k-means clustering, and sub-workflow (task) level, using Naive Bayes, Decision Tree, and Isolation Forest. We evaluated the proposed techniques on datasets that are collected from real science workflow executions on a testbed environment.

Results show that the workflow-level analysis employing k-means clustering can accurately cluster anomalous workflows into statistically similar classes with a reasonable quality of clustering, achieving over 0.7 for NMI and Completeness scores, affirming our workflow-level feature selection. For task-level analysis, in most of the cases, the tested classifiers can achieve $>50\%$ accuracy, especially Decision Tree classifier can achieve $>80\%$ accuracy. We believe that the promising results can be a foundation for future research on science workflow anomaly detection and failure prediction for workflows running in production environments.

For future work, we will extend the classifiers to explore more hyperparameter tuning and deep learning neural networks to help identify feature relationships in scientific workflow executions. We also plan to apply the proposed classifiers to the workflow runs on real infrastructures.

ACKNOWLEDGMENT

This work was funded by DOE contract number #DE-SC0012636M, "Panorama 360: Performance Data Capture and Analysis for End-to-end Scientific Workflows". Any opinions or findings in this article are of the authors and do not necessarily reflect views of the sponsor.

REFERENCES

[1] Energy Sciences Network (ESnet). https://www.es.net/.

[2] Internet2. https://www.internet2.edu/.

[3] Panorama 360 project. https://panorama360.github.io/.

[4] 1000 GENOMES PROJECT CONSORTIUM. A global reference for human genetic variation. *Nature 526*, 7571 (2012), 68–74.

[5] AMER, M., AND GOLDSTEIN, M. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proceedings of the 3rd RapidMiner Community Meeting and Conferernce (RCOMM 2012). RapidMiner Community Meeting and Conference (RCOMM-2012), August 28-31, Budapest, Hungary* (8 2012), S. Fischer and I. Mierswa, Eds., Shaker Verlag GmbH, pp. 1–12.

[6] BALDINE, I., XIN, Y., MANDAL, A., RUTH, P., HEERMAN, C., AND CHASE, J. Exogeni: A multi-domain infrastructure-as-a-service testbed. In *Testbeds and Research Infrastructure. Development of Networks and Communities* (Berlin, Heidelberg, 2012).

[7] DEELMAN, E., MANDAL, A., JIANG, M., AND SAKELLARIOU, R. The role of machine learning in scientific workflows. *The International Journal of High Performance Computing Applications 33*, 6 (2019), 1128–1139.

[8] DEELMAN, E., VAHI, K., JUVE, G., RYNGE, M., CALLAGHAN, S., MAECHLING, P. J., MAYANI, R., CHEN, W., FERREIRA DA SILVA, R., LIVNY, M., AND WENGER, K. Pegasus: a workflow management system for science automation. *Future Generation Computer Systems 46* (2015), 17–35.

[9] GAIKWAD, P., MANDAL, A., RUTH, P., JUVE, G., KROL, D., AND DEELMAN, E. Anomaly detection for scientific workflow applications on networked clouds. *International Conference on High Performance Computing & Simulation* (2016).

[10] HE, Z., XU, X., AND DENG, S. Discovering cluster-based local outliers. *Pattern Recogn. Lett. 24*, 9–10 (June 2003), 1641–1650.

[11] HERBST, J. A machine learning approach to workflow management. In *Machine Learning: ECML 2000* (Berlin, Heidelberg, 2000), Springer Berlin Heidelberg, pp. 183–194.

[12] HERBST, J., AND KARAGIANNIS, D. Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. *Intelligent Systems in Accounting, Finance and Management 9*, 2 (2000), 67–92.

[13] JINKA, P., AND SCHWARTZ, BARON, A. *Anomaly detection for monitoring : a statistical approach to time series anomaly detection*, first edition ed. Sebastopol, CA : O'Reilly Media, 2015. Includes bibliographical references.

[14] LIU, F., MING, K. T., AND ZHOU, Z.-H. Isolation forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining* (2008), ICDM, pp. 413–422.

[15] MCDAID, A. F., GREENE, D., AND HURLEY, N. Normalized mutual information to evaluate overlapping community finding algorithms, 2011.

[16] NAUMANN, F., FREYTAG, J.-C., AND LESER, U. Completeness of integrated information sources. *Information Systems 29*, 7 (2004), 583 – 615. Data Quality in Cooperative Information Systems.

[17] OF ENERGY, U. D. Summit - oak ridge leadership computing facility. https://www.olcf.ornl.gov/summit/.

[18] ROSSI, D., MELLIA, M., AND CASETTI, C. User patience and the web: a hands-on investigation. *Global Telecommunications Conference* (2003).

[19] SAFAVIAN, S. R., AND LANDGREBE, D. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics 21*, 3 (May 1991), 660–674.

[20] SAMAK, T., GUNTER, D., GOODE, M., DEELMAN, E., JUVE, G., MEHTA, G., SILVA, F., AND VAHI, K. Online fault and anomaly detection for large-scale scientific workflows. In *2011 IEEE International Conference on High Performance Computing and Communications* (Sep. 2011), pp. 373–381.

[21] SARITAS, M., AND YASAR, A. Performance analysis of ann and naive bayes classification algorithm for data classification. *International Journal of Intelligent Systems and Applications in Engineering 7*, 2 (Jun. 2019), 88–91.

[22] SINGH, A., RAO, A., PURAWAT, S., AND ALTINTAS, I. A machine learning approach for modular workflow performance prediction. In *Proceedings of the 12th Workshop on Workflows in Support of Large-Scale Science* (2017), WORKS, pp. 7:1–7:11.

[23] WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHRÖDL, S. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning* (San Francisco, CA, USA, 2001), ICML '01, Morgan Kaufmann Publishers Inc., p. 577–584.

[24] WANG, T., WEI, J., ZHANG, W., ZHONG, H., AND HUANG, T. Workload-aware anomaly detection for web applications. *Journal of Systems and Software 89* (2014), 19 – 32.

[25] WATERLAND, A. stress, POSIX workload generator, 2013.

[26] ZHANG, J., AND ZULKERNINE, M. Anomaly based network intrusion detection with unsupervised outlier detection. *IEEE Communications* (2006).