International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

Special Issue Paper

The role of machine learning in scientific workflows

The International Journal of High Performance Computing Applications
1–12

DOI: 10.1177/1094342019852127
journals.sagepub.com/home/hpc
SAGE

Ewa Deelman[1], Anirban Mandal[2], Ming Jiang[3] and Rizos Sakellariou[4]

Abstract

Machine learning (ML) is being applied in a number of everyday contexts from image recognition, to natural language processing, to autonomous vehicles, to product recommendation. In the science realm, ML is being used for medical diagnosis, new materials development, smart agriculture, DNA classification, and many others. In this article, we describe the opportunities of using ML in the area of scientific workflow management. Scientific workflows are key to today's computational science, enabling the definition and execution of complex applications in heterogeneous and often distributed environments. We describe the challenges of composing and executing scientific workflows and identify opportunities for applying ML techniques to meet these challenges by enhancing the current workflow management system capabilities. We foresee that as the ML field progresses, the automation provided by workflow management systems will greatly increase and result in significant improvements in scientific productivity.

Keywords

Scientific workflows, machine learning, workflow systems, anomaly detection, workflow composition

1. Introduction

Scientific workflows are being used in a number of scientific domains (Altintas et al., 2004; Deelman et al., 2017b; Li et al., 2008; Poehlman et al., 2016; Usman et al., 2015). They allow researchers to organize their computations, describing them as directed, often acyclic graphs. These graphs indicate the computational tasks that need to be performed and their order as well as the data that flows through these tasks (Deelman et al., 2017b; Jones, 2007; Liu et al., 2015). Large-scale distributed computing infrastructures are typically used for the execution of scientific workflows. A number of different workflow management systems have been designed over the last 20 years to deal with various aspects of workflow management (Abramson et al., 2008; Chase et al., n.d.; Deelman et al., 2015; Fahringer et al., 2005; Goderis et al., 2007; Wolstencroft et al., 2013). The workflow management functions can be defined as a cycle of steps or phases: workflow composition, resource provisioning, and workflow execution with potential workflow adaptation. Workflow composition deals with workflow component and data selection as well as the logical sequencing of tasks. Resource provisioning selects and provisions resources needed to execute the workflow tasks. Based on the results of the workflow execution, the workflow may be modified resulting in a new entity, which needs to be executed and managed.

As scientific workflows play an increasing role in significant advances in science and engineering (most notably the detection of gravitational waves by the laser interferometer gravitational-wave observatory (LIGO) experiment (Weitzel et al., 2017)) and workflow management becomes more demanding, there is a need for more sophisticated techniques to help with each phase. Artificial intelligence (AI) techniques were examined as early as 2004 (Gil et al., 2004) for their ability to help with the process of workflow composition, execution, fault detection, and workflow repair in a distributed environment—at that time the Grid. The heart of the problem of these workflow management functions was the lack of detailed knowledge about the application components or the execution environment; gaining up-to-date information about workflow execution was difficult and in some cases impossible. The approach

[1] University of Southern California Information Sciences Institute, Marina Del Rey, CA, USA
[2] Renaissance Computing Institute, Chapel Hill, NC, USA
[3] Lawrence Livermore National Laboratory, Livermore, CA, USA
[4] The University of Manchester, Manchester, UK

Corresponding author:
Ewa Deelman, University of Southern California Information Sciences Institute, Marina Del Rey, CA 90292, USA.
Email: deelman@isi.edu

was thus to envision a set of knowledge bases that capture implicit knowledge about the state of the application and of the system. Reasoners would then invoke the appropriate knowledge source to refine the user request to a specific workflow, then reason about particular resources to use, and classify failures. Unfortunately, such knowledge was hard to obtain and consequently the reasoners for workflow composition and management were very limited.

A number of different techniques and heuristics have been developed to manage the workflow life cycle, which include workflow composition (Gil et al., 2011; Goderis et al., 2007; Goecks et al., 2010; Taylor et al., 2007), resource provisioning (Byun et al., 2008; Malawski et al., 2015b; Zhou et al., 2015), and various aspects of workflow execution such as job scheduling (Durillo et al., 2012; Lee et al., 2015.; Pietri and Sakellariou, 2014; Ramakrishnan et al., 2007) and fault detection (Duan et al., 2005; Plankensteiner et al., 2009; Samak et al. 2011b). With the advances being made in machine learning, new opportunities are presenting themselves, providing potentially more sophisticated and scalable methods that can generate better performing workflows and possibly learn to create workflows based on prior knowledge. In this article, we explore the opportunities that ML techniques can offer in the area of scientific workflows. We discuss the current challenges in each phase of the workflow life cycle (i.e. composition, provisioning, execution) and describe how ML can be used to address them (e.g. unexpected and anomalous behavior). Finally, the article gives examples of how ML techniques are used today in managing scientific workflows and hypothesizes potential uses of these techniques to improve the state of the art in workflow management.

2. Current challenges in scientific workflows

In this section, we discuss the current challenges in managing scientific workflows in distributed systems and how ML techniques can be used to address these challenges. Our focus is on the three main phases of the workflow life cycle: workflow composition, resource provisioning, and workflow execution.

2.1. Workflow composition

There are a number of ways to compose scientific workflows. In some cases, the workflows are composed graphically (Goderis et al., 2007; Missier et al., 2010; Taylor et al., 2007), in some cases using a variety of well-known languages (Deelman et al., 2015) or with the aid of new languages (Albrecht et al., 2012; Kotliar et al., 2018; Qin and Fahringer, 2012). However, the main problem of designing or reusing an existing workflow still remains: the challenge is in picking the right analysis for the data at hand or for the desired result. In addition to selecting the right analysis, the workflow component parameters may have to be tuned for a particular problem. If no such analysis exists, then a new workflow needs to be designed.

One approach for finding suitable workflows is to reuse workflows from a repository, such as MyExperiment (David De and Goble, 2009; DeRoure et al., 2007). The workflows can be searched based on tags, contributors, and systems they are written in among others. It is up to the user to decide which workflows to select for use or reuse. However, one can imagine that ML can be used to learn what workflows are relevant to the users based on their previous searches or the type of data the user wants to analyze, or the results that the user is looking for.

Taking this further, ML could be used to select a set of "similar" workflows and then suggest new workflow components that could be used to augment the workflow to obtain the desired results. Today, there are systems such as Wings (Gil et al., 2007, 2011), which enable users to compose workflow templates. A Wings template represents a skeleton of a workflow indicating the types of components and data needed but not the exact data sets of component implementations. Wings defines semantic constraints about data sets and workflow components, which can be used in component selection (filling out the template) as well as in workflow validation. The semantic information is also propagated to the results by providing the metadata for the data sets generated by the workflow.

Finally, ML techniques could be used to compose an entire workflow from scratch. This would require performing experiments to explore different workflow component combinations that can execute successfully and work well for particular data sets. However, this level of automation may not be fully desirable. In previous work, AI planning techniques for workflow composition were explored: the workflow goals were the desired data products and the operators were the application components (Blythe et al., 2003; Gil et al., 2004). The planners also received the current state of the distributed system. Although these techniques were able to produce valid workflows, the target scientists did not like the fact that the processes of component selection and workflow composition were all automated. Scientists wanted to be able to reason about the workflow and how to compose it themselves, often through an exploratory process. However, ML techniques can help fill in the details for high-level workflow structures. For example, ML techniques can learn from previous workflow executions to infer which parameters used by the workflow components worked the best to obtain successful executions or desired results. At a more fine-grained level, ML techniques could be used to reason about the right shape of the workflow to enable a smaller workflow data footprint (Singh et al., 2007).

2.2. Resource provisioning

After the workflow is composed, a user or a workflow management system needs to decide what storage, network, and computational resources are needed for the successful

workflow execution. This decision involves figuring out the types of resources (what types of operating system and other software) are needed to support the execution of a workflow component. Other resource characteristics such as the amount of memory or CPU speed to use also need to be taken into account (Pietri and Sakellariou, 2019). Finally, the amount of each type of resources, storage, network bandwidth needs to be decided, often requiring a balance between different requirements and cost (Malawski et al., 2015a, 2015b).

Various approaches have been taken to performance modeling and prediction, which can guide resource selection decisions. Some involve developing analytical models of applications executing on particular platforms, such as is the case with Aspen (Spafford and Vetter, 2012). This is particularly useful when the performance model needs to provide quick results, or when the prediction about the application performance is made in the context of a system that is not available (yet to be designed or deployed). In some cases, analytical models are not sufficient to provide desired accuracy, for example, in cases where there is contention for system resources, such as networks or I/O systems. In these cases, simulations are often used to model the system and potentially the application in more detail. An example of such a simulator is ROSS (Carothers et al., 2002), which can simulate complex high-performance interconnects and I/O. Simulations may take longer to run than analytical models, so their usefulness for online predictions is often limited.

When it is possible to run the applications on the target platforms, performance information can be collected and analyzed using simple statistical methods (calculating averages, standard deviations, etc.) (Deelman et al., 2017a; Krol et al., 2016) or in combination with simple analytical models (Pietri et al., 2014). However, in order to use these metrics, one often needs to determine which factors influence application performance and what effect they have. In some cases, trial-and-error approaches are employed. In Tovar et al. (2018), the authors provision resources based on previously observed job needs, being conservative in their approach to limit resource wastage. When the resources turn out to be insufficient, the job is given a larger amount of resources. The process is repeated until the job successfully completes.

ML techniques promise to be able to learn patterns of application and system behavior that can be more accurate in predicting application performance (Jain et al., 2013; Matsunaga and Fortes, 2010; Nemirovsky et al., 2017). Such methodologies can be used to analyze past and current workflow performance to identify the important parameters that affect workflow behavior from a particular point of view—for example, from the point of view of data source selection for a task. In that case, one can learn which data sources have high availability, what are the best parameter settings to use for data transfers over particular networks, and so on. Some parameters can be time dependent, for example, the load at the data sources may be particularly high during some time periods.

To discover parameters that determine workflow performance, clustering type algorithms can be employed. One can, for example, analyze class membership for both "normal" and "faulty" clusters to understand the patterns in the workflow executions. By analyzing the members of the "normal" clusters, we can determine which parameters to use for a task (e.g. degree of concurrency to use, parameters to set, amount of memory needed, and so on) and for jobs such as how many processors, which resources to use and which to avoid, how to configure data transfers and more. By analyzing members of the "faulty" class, we can learn which resource or combination of resources to avoid, for example, avoiding using a data transfer path that is experiencing packet loss or corruption, or an endpoint experiencing disk failure. Hence, ML algorithms can provide recommendations for suitable or unsuitable resources for a workflow, how to configure them, and so on.

ML techniques could also be used to guide other forms of resource selection where multiple subsystems are involved, which would require careful considerations of subtle bottlenecks and interferences. For example, the scheduling of data transfer between tasks can too often create bottlenecks between computation and communication phases, and manual optimizations are often complex (Huang et al., 2019). We can train ML models to classify the workflow phases to optimize data movements, to orchestrate I/O (Meng et al., 2014; Wang et al., 2015), and to manage hierarchical storage (Dong et al., 2016) and data staging (Subedi et al., 2018). Also, as in-situ execution becomes more prevalent (Huang et al., 2019; Kwan-Liu, 2009; Subedi et al., 2018), ML can play an important role in automating the placement of tasks to automatically find an optimal trade-off.

2.3. Workflow execution

Workflow execution involves scheduling the tasks in a workflow in the order they are supposed to execute on the resources provisioned for the tasks. In some cases, the workflow tasks may be scheduled just-in-time without an additional prior provisioning step (Deelman et al., 2006). Once the jobs are scheduled, they need to be monitored for success or failure, resource consumption, or any sort of anomalous behavior, which indicates some departure from what is expected. Figure 1 shows how an example workflow that traverses several stages (left), and how different types of cyberinfrastructure facilities provide resources for the execution of the workflow (right).

As the workflow is executing, the analysis of the performance data being generated needs to encompass two aspects: (1) identification of workflow anomalies and task performance bottlenecks by using the metrics and performance data relevant to workflows and component applications, and (2) detection and localization of faults in the multi-domain, distributed infrastructure by leveraging

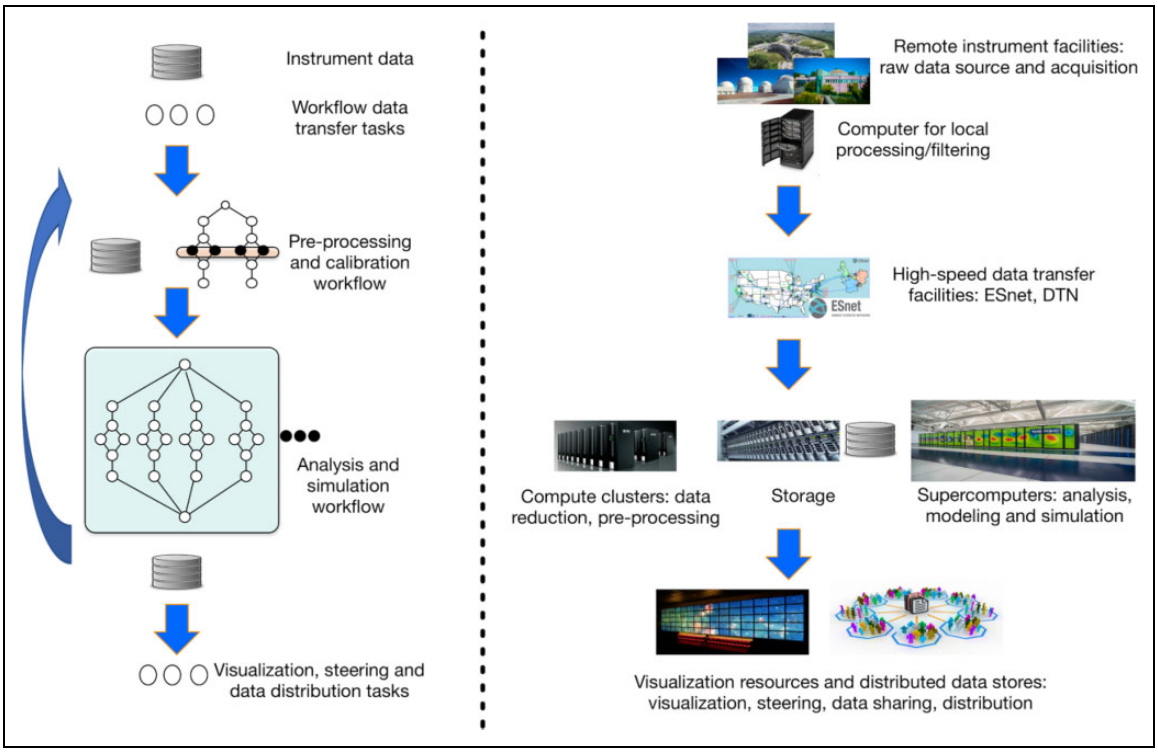

Figure 1. On the left, a sample workflow that processes instrumental data. On the right, possible resources to be used for the execution of the workflow tasks.

infrastructure-centric metrics. One can then build correlated failure models across workflow- and infrastructure-level metrics for troubleshooting and pinpointing sources of failures (Gunter et al., 2011; Samak et al., 2011b, 2013).

If undesired or anomalous behavior is detected, the workflow management system may decide to reschedule jobs onto a different resource, select a different data source or target storage system, decide to provision additional resources, or, in the most drastic case, decide to compose a different workflow altogether. Today, these decisions are made in a simplistic way, mostly through trial-and-error or basic heuristics. A first step to improve the quality of adaptation decisions is to be able to detect anomalies in the first place (Mandal et al., 2016; Prathamesh et al., 2016).

In today's workflows, 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. 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 and simple statistics (e.g. moving averages) (Jinka and Schwartz, 2015) that can: (1) fail to understand longitudinal patterns, (2) miss opportunities for anomaly detection, and (3) seldom be used for identifying the root cause of the anomalies. Existing statistical techniques can make assumptions about underlying distributions for the metric values, which might not hold in large-scale execution environments. Being univariate in nature, these statistical models cannot capture interactions between features. Hence, multivariate techniques, in particular ML algorithms, are envisioned as an appropriate approach for building failure models and for detecting and diagnosing failures in large-scale workflow executions on complex systems.

One potential approach is to apply ML algorithms in a top–down approach beginning with workflow-level analysis. This analysis can use high-level, aggregate workflow performance metrics, such as the number of failed/completed tasks versus the total number of tasks in the workflow, to predict the overall behavior of a running workflow by clustering statistically similar workflows into classes. When the aggregate analysis of workflow-level metrics reveals membership to an anomalous class, job-level analysis can be triggered. Job-level analysis can then detect faults and bottlenecks using detailed job information such as resource usages, data sizes, resource parameters, and application-specific job parameters.

3. ML approaches in scientific workflows

While the previous sections touched upon the challenges faced in managing scientific workflows in distributed systems, here we describe ML techniques that are being used today to analyze the behavior of workflows at various levels of abstraction (workflow, task, and infrastructure) using different processing modalities (online and off-line). We

also describe potential issues with using ML, such as collecting the training data.

3.1. Workflow-level analysis

Workflow-level analysis explores the coarse-grained behavior of jobs in the workflow: their runtime, success, failure, and so on. Unsupervised clustering, in which the classes are formed without prior information ("supervision"), to classify similar workflow runs and associate them with behavioral classes is a potential ML approach that can be employed for workflow-level analysis. Features selected for each workflow can be either static or real time. The static features are independent of a given execution, and thus can be computed directly from the workflow descriptions. Static features can include the average degree of job dependencies or fan out, the average level of parallelism, and the average degree of data dependencies. Real-time features are dependent on a given execution and include the percentage of jobs/tasks that executed successfully, the percentage of jobs/tasks that failed, the average duration of successful jobs/tasks, and the average duration of failed jobs/tasks (before termination). Any metric that is available both in the historical and real-time data can be a feature.

Researchers have used the efficient k-means clustering algorithm (Amer and Goldstein, 2012; He et al., 2003; MacQueen, 1967; Samak et al., 2011b; Wang et al., 2014), with randomized initial centers and the (standard) Euclidean distance metric to cluster workflows. More sophisticated clustering algorithms should be investigated and compared with prior approaches (Duda et al., 2012). In particular, fuzzy c-means clustering (Hathaway and Bezdek, 1986; Pal et al., 1996) can be used to exploit some class overlaps that have been found with the initial clustering. Fuzzy clustering has the ability to capture non-separable classes, without the expensive pre-processing of features.

Clustering algorithms produce a set of classes, with associated numerical models, which can be used for future prediction. Using online processing, feature vectors can be computed, and the cluster model can be used to compute a degree of membership for each cluster. For example, a workflow can be classified in real time to belong in a High Failure Workflow (HFW) class with 90% membership, while another workflow can be 60% HFW. Feature vectors can be also computed at different points in the workflow lifetime, generating a classification result. By tracking a workflow's degree of membership over time, triggers can be generated when the membership in a given class crosses a threshold. These triggers can be used by workflow management systems to signal the need to adapt the workflow or the resources. The quality of clustering algorithms can be evaluated by measuring inter-cluster homogeneity (minimum overall distance between objects from the same class) and intra-cluster separation (maximum distance between different clusters).

3.2. Task-level analysis

To better understand the source of the anomaly, task-level analysis should be triggered when a possibly anomalous workflow is found. This analysis should identify possible causes of task failures and help identify performance bottlenecks. The analysis can be aided by task performance data and job-relevant metadata from a workflow performance data repository. The workflow management system collects task-level data as the workflow is executing. Hence, the system can perform accurate labeling of the data when task failures are observed. Based on the status of the job containing the task, the workflow management system can label a feature vector consisting of task-specific metrics as "Failed" or "Successful." As a result, one can assume to have a significant portion of tasks' performance data being labeled. This allows the building of supervised learning classifiers that distill task failures from historical labeled training data, and those models can be used for classifying task failures at runtime for fast detection.

Naive Bayes classifier has been shown to accurately predict the failure probability of tasks for scientific workflows on the cloud using task performance data (Samak et al., 2013). Others (Bala and Chana, 2015) have compared logistic regression, artificial neural nets (ANN), Random Forest and Naive Bayes for failure prediction of workflow tasks in the cloud and concluded that the Naive Bayes' approach provided the maximum accuracy. In Buneci and Reed (2008), the authors have used a k-nearest neighbors (k-NN) classifier to classify workflow tasks into "Expected" and "Unexpected" categories using feature vectors constructed from temporal signatures of task performance data. In addition to applying the Naive Bayes classifier, further research should be conducted to explore a spectrum of classifiers for task performance data, which can include k-NN, ANNs, logistic regression, and Support Vector Machines (SVM) (Lorena et al., 2011). The accuracy of the classification algorithms should be then evaluated using paired criteria like *precision* and *recall*, and combined criteria like the *balanced classification rate*, which takes into account both true negative and true positive rates, and *F-measure*, which is the harmonic mean between *precision* and *recall*.

Sometimes, performance bottlenecks can also be detected using metrics that are gathered from the task metadata and provenance information, some of which might be categorical in nature. In these cases, decision tree-based classifiers can be used because they are suitable for fast online inference of the "tree path" that led to the anomaly. The features can include task information such as task type, input size, parameters to the executable, as well as system related information such as user name, site name, host IP, job delay, and job exit code. The feature vectors can then be fed to a learning classifier as the training set to generate a model for predicting behaviors of interest. In previous work (Samak et al., 2011a), regression trees were used as the learning classifier, giving both prediction ability and fault

isolation results. Other partitioning algorithms should be investigated for constructing decision trees (Breiman, 2017). These algorithms are able to build a tree, where internal nodes are feature descriptions and tree leaves are task states. Traversal of the tree from root to leaf for a given input task can help identify bottlenecks.

3.3. Infrastructure-level analysis

The distributed, heterogeneous nature of the end-to-end platform with multiple resource providers makes it harder to collect labeled data systematically about anomalies and failures in the infrastructure used to execute workflows. In addition, the dynamic characteristics of the platform-induced anomalies often manifest themselves in an unknown manner. This makes it difficult to define anomaly classes a priori. Hence, various unsupervised learning (UL) techniques should be more suitable for infrastructure-level data analysis.

Choosing an appropriate feature space needs to be the first step in the process. Metrics like *uptime*, number of jobs per machine, system throughput and latency, available storage and network capacity, and so on play a critical role to ensure workflows reach completion, and hence should be present in relevant feature vectors. The next step is the selection of appropriate UL strategies. Several researchers have studied the use of UL techniques for anomaly detection in computing systems (Ibidunmoye et al., 2015). Three broad UL strategies, and combinations thereof, have been shown to work well in different scenarios: (1) nearest neighbor-based techniques like those based on local outlier factor (LOF), k-NN (Amer and Goldstein, 2012; Bhaduri et al., 2011; Elomaa et al., 2002; He et al., 2003; Wang et al., 2014), (2) clustering-based techniques using k-means clustering combined with outlier factor (Amer and Goldstein, 2012; He et al., 2003; Wang et al., 2014), and (3) self-organizing maps-based techniques (Dean et al., 2012; Kohonen, 2001).

Selecting the appropriate UL technique for workflow use cases is a non-trivial problem. While nearest neighbor-based approaches produce more accurate models, they are computationally more expensive than clustering-based approaches (Goldstein and Uchida, 2016). Some techniques work well to detect global anomalies (e.g. k-NN) but fail to identify local anomalies. The strategy should be to systematically explore this spectrum of UL techniques, with particular emphasis on online techniques that combine incremental clustering with dynamic LOF calculations, which potentially balances accuracy and detection time.

3.4. Cross-level analysis

Most research on performance anomaly detection using ML techniques has dealt with either the system or the application (Chandola et al., 2009; Ibidunmoye et al., 2015). Correlating these two types of anomalies in a unified framework is a relatively unexplored problem, albeit a very important one. Correlations allow users to identify the source of anomalies and performance bottlenecks by convolving failing or poorly performing workflow tasks with infrastructure elements potentially responsible for the anomalies. One approach can be to extend the UL techniques developed above to cluster feature vectors and identify outliers. Then the anomalous samples can be investigated to troubleshoot the sources of anomalies.

In addition to the infrastructure-level metrics, one will need to include the relevant workflow- or task-level metrics in the feature vectors. This will have an effect on how the UL algorithms will scale because this would significantly increase the number of metrics. In choosing the UL algorithm, several variants of clustering-based algorithms should be explored since they tend to scale better with larger feature spaces and sample sizes. Reducing the dimensionality of data using techniques such as principal component analysis, factor analysis, and similarity identification (Jolliffe and Cadima, 2016; Fu, 2011; Steuer et al., 2002) will help with scalability as well.

After detecting anomalous instances, the anomalous metrics should be located for analyzing root causes of performance deviations or failures. Since the feature vector contains both kinds of metrics, one approach can be to use a simple but effective method based on the Student's *t*-test (Weiss and Weiss, 2012) statistical method. This method exploits the underlying property that normal data instances occur in high probability regions in a stochastic model, while anomalies occur in low probability regions. If comparison with prior samples of a metric results in significant differences, it can signify the presence of an anomaly. For every metric in the anomalous feature vector, one can calculate the *t*-transfer to fit the *t*-distribution to calculate the metric anomaly value (MAV) (Wang et al., 2014), and then sort the MAVs to locate the suspicious metrics. Correlated anomalies will manifest as high MAV values for multiple metrics, which can help troubleshoot the sources of the anomaly.

3.5. Online/off-line analysis

The ML-based methods will need to analyze both online and off-line performance and provenance data, and they can be integrated following the principles of the "Lambda architecture" (LA) (Kiran et al., 2015). LA is a generic, linearly scalable, and fault-tolerant data processing architecture that is able to serve a wide range of queries and computations on both fast-moving (streaming) data and historical data (batch). Large volumes of performance and provenance data can be analyzed with both batch- and stream-processing techniques. The stream processing component, the "speed layer," can encapsulate the ML approaches for online analysis, while batch-processing can be leveraged for the heavyweight off-line ML techniques at the "batch layer" analyzing data across multiple workflows.

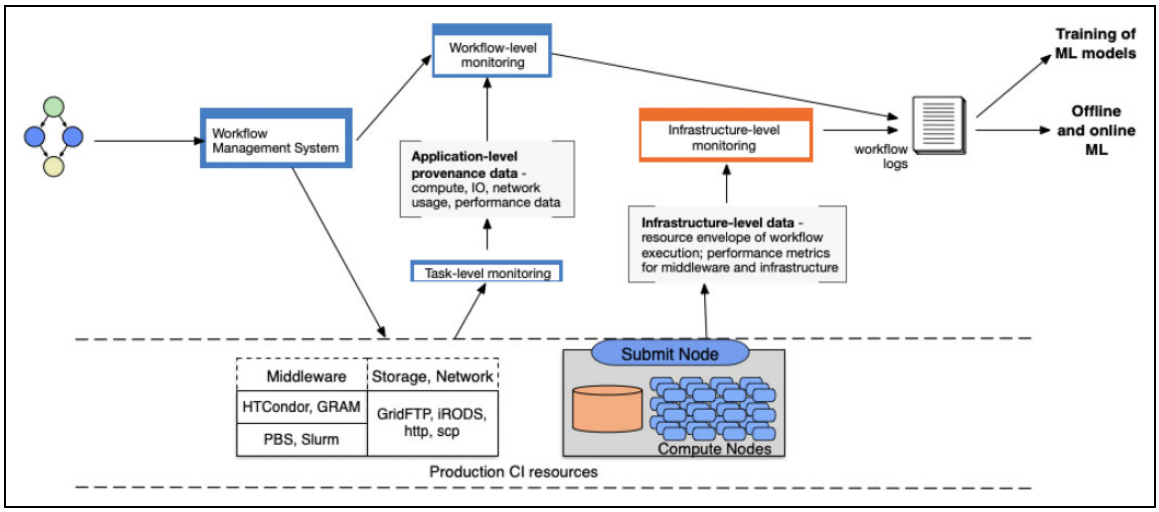

Figure 2. Overview of the workflow management, monitoring, and analysis systems.

For comprehensive introspection and analysis of performance and provenance data, *off-line ML-based approaches* need to be developed to enable longitudinal analysis across multiple workflow executions. Such a type of analysis can be computationally intensive since it integrates and correlates data from multiple workflows, thereby enabling discovery of patterns across workflows. Several supervised and semi-supervised learning approaches (SVM, Random Forest, Bayesian classifiers, etc.) have been shown to be effective in identifying system and application anomalies. All these techniques rely on the availability of high-quality labeled data to train the ML models to be used later for classification of test cases. Hence, one should use carefully labeled performance and provenance data obtained from workflow executions on isolated and controlled environments to train the ML-based analysis algorithms.

A significant challenge for *online analysis* is to develop algorithms that are lightweight yet provide accurate detection of anomalies when operating on high-volume, real-time, streaming data. A two-pronged approach can be employed for online analysis of a single workflow execution. Deploying the ML models developed using off-line techniques on production cyberinfrastructure is one option. One can then evaluate the accuracy of those models in production for predicting faults and detecting anomalies. The other option can be to explore the application of low-overhead, streaming versions of ML techniques like streaming mini-batch k-means and logistic regression directly on streaming data. We can leverage the state-of-the-art data stream processing engines and libraries including Apache Spark Streaming (Spark, 2014) and Apache MOA/SAMOA (Kourtellis et al., 2019) and the best off-line ML models for online analysis. While off-line ML models can be used to guide online analysis, the results of the online analysis, capturing specific characteristics of workflow ensemble run, can also be used to tune and update the off-line ML models. Such feedback loops between off-line and online approaches will be essential to improve the accuracy of each.

3.6. Training data collection

In order to train robust ML models that can be used for scientific workflows, one needs to collect a large and diverse set of data from workflows, not just individual users but from large collaborations as well. One will need to develop new capabilities to instrument the scientific workflow to automatically collect and store various metrics for the end-to-end workflow that can be used for ML training data, including information about (a) the input, intermediate, and output data products; (b) application codes that constitute the workflow; and (c) the resource envelope or the infrastructure the workflow is executing in.

Novel architectures are needed for triaging and collating data from a variety of tools responsible for workflow and infrastructure performance monitoring and the collection of various metrics that can be used for ML training. Figure 2 provides a conceptual framework for such a data collection architecture. The expanded set of metrics collected from multiple sources of data can be triaged at a message bus that can be used for off-line and online analysis as described in Section 3.5. The data collected can also be used as an audit trail for the workflow execution and can help in capturing provenance of both the application processes and the infrastructure used for processing.

Collecting provenance information for workflows is critical, as provenance data provide an audit trail by which the integrity of the data can be judged (Simmhan et al., 2005, 2006; Zhang et al., 2011). Data provenance collection in a cloud or another multi-provider environment, common for workflow execution today, is challenging due to the need for cross-layer correlation of data from multiple layers and sources (Muniswamy-Reddy et al., 2009, 2010). Part of provenance data collection includes data about specific virtual and physical resources used for the execution

of workflow tasks as well as storage and transfer of initial, intermediate, and final workflow data products, all of which can be used as learning features to train ML models. This information can be used to identify data sets that were produced by hardware that became faulty or infrastructure that was in some way compromised and whose outputs therefore cannot be trusted. Collecting this information requires specific mechanisms and trust structures by which the data can be acquired, attributed and stored.

4. Conclusions

The community is just at the beginning of exploring ML techniques in the scientific workflow space. There are many opportunities that are outlined in this article. If we successfully leverage and potentially develop new ML techniques for workflows, doing science will become as easy as using a smartphone app. As a result, scientific productivity will increase and the population of scientists that use computational methods for their work will grow as well. New workflow systems will be able to understand the user's previous requests, discover the related data and structure the computations needed to deliver the desired results. However, providing this level of automation may make the introspection of the processes used to obtain the results more difficult. It would also potentially make reproducibility more difficult. Nevertheless, it can potentially provide a means of comparison of different scientific methods and their similarities and differences to other approaches.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was performed in part under the auspices of the US Department of Energy (DOE) by Lawrence Livermore National Laboratory (LLNL) under Contract DE-AC52-07NA27344. This work was also funded by DOE under grant #DESC0012636, "Panorama–Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows," by NSF under grant #1839900: CICI: SSC: Integrity Introspection for Scientific Workflows (IRIS), and by the Laboratory Directed Research and Development Program at LLNL under project 16-ERD-036 (LLNL-JRNL-765200).

ORCID iD

Ewa Deelman https://orcid.org/0000-0001-5106-503X

References

Abramson D, Enticott C and Altinas I (2008) Nimrod/K: towards massively parallel dynamic grid workflows. In: *SC '08: Proceedings of the 2008 ACM/IEEE conference on supercomputing*, Austin, TX, USA, 15–21 November 2008.

Albrecht M, Donnelly P, Bui P, et al. (2012) Makeflow: a portable abstraction for data intensive computing on clusters, clouds, and grids. In: *1st workshop on scalable workflow execution engines and technologies, 2012, SWEET '12*, Scottsdale, Arizona, USA, 20 May 2012.

Altintas I, Berkley C, Jaeger E, et al. (2004) Kepler: an extensible system for design and execution of scientific workflows. In: *Proceedings of the 16th international conference on scientific and statistical database management*, Santorini Island, Greece, 23 June 2004, p. 423. Washington, DC: IEEE Computer Society.

Amer M and Goldstein M (2012) Nearest-neighbor and clustering based anomaly detection algorithms for RapidMiner. In: *Proceeding of the 3rd RapidMiner community meeting and conference (RCOMM 2012)*, Budapest, Hungary, 28–31 August 2012, pp. 1–12.

Bala A and Chana I (2015) Intelligent failure prediction models for scientific workflows. *Expert Systems with Applications* 42(3): 980–989.

Bhaduri K, Das K and Matthews BL (2011) Detecting abnormal machine characteristics in cloud infrastructures. In: *2011 IEEE 11th international conference on data mining workshops*, Vancouver, Canada, 11–14 December 2011, pp. 137–144.

Blythe J, Deelman E, Gil Y, et al. (2003) The role of planning in grid computing. In: *ICAPS, 2003*. Available at: https://www.globus.org/sites/default/files/ICAPS03-016.pdf (accessed 4 January 2019).

Breiman L (2017) *Classification and Regression Trees*. Abingdon: Routledge.

Buneci ES and Reed DA (2008) Analysis of application heartbeats: learning structural and temporal features in time series data for identification of performance problems. In: *SC '08: Proceedings of the 2008 ACM/IEEE conference on supercomputing*, Austin, Texas, 15–21 November 2008, pp. 1–12.

Byun E-K, Kim J-S, Kee Y-S, et al. (2008) Efficient resource capacity estimate of workflow applications for provisioning resources. In: *4th IEEE international conference on e-Science (eScience)*, Indianapolis, IN, USA, 7–12 December 2008.

Carothers CD, Bauer D and Pearce S (2002) ROSS: a high-performance, low-memory, modular Time Warp system. *Journal of Parallel and Distributed Computing* 62(11): 1648–1669.

Chandola V, Banerjee A and Kumar V (2009) Anomaly detection: a survey. *ACM Computing Surveys* 41(3): 15:1–15:58. New York, NY, USA: ACM.

Chase J, Gorton I, Sivaramakrishnan C, et al. (n.d.) Kepler + MeDICi service-oriented scientific workflow applications. In: *2009 world conference on services—I*, Los Angeles, CA, 6–10 July 2009, pp. 275–282.

David De R and Goble C (2009) Lessons from myExperiment: research objects for data intensive research. In: *Microsoft e-Science Workshop*, Pittsburgh, PA, 15–17 October 2009.

Dean DJ, Nguyen H and Gu X (2012) Ubl: unsupervised behavior learning for predicting performance anomalies in virtualized

cloud systems. *Proceeding of the 9th international conference on autonomic computing*. Available at: https://dl.acm.org/citation.cfm?id=2371572 (accessed 4 January 2019).

Deelman E, Carothers C, Mandal A, et al. (2017a) PANORAMA: an approach to performance modeling and diagnosis of extreme scale workflows. *The International Journal of High Performance Computing Applications* 31(1): 4–18.

Deelman E, Kosar T, Kesselman C, et al. (2006) What makes workflows work in an opportunistic environment? *Concurrency and Computation: Practice & Experience* 18(10): 1187–1199.

Deelman E, Peterka T, Altintas I, et al. (2017b) The future of scientific workflows. *The International Journal of High Performance Computing Applications* 32(1): 159–175.

Deelman E, Vahi K, Juve G, et al. (2015) Pegasus, a workflow management system for science automation. *Future Generations Computer Systems: FGCS* 46(0): 17–35.

DeRoure D, Goble CA and Stevens R (2007) Designing the myExperiment virtual research environment for the social sharing of workflows. In: *IEEE international conference on e-Science and grid computing (e-Science)*, Bangalore, India, 10–13 November 2007.

Dong B, Byna S, Wu K, et al. (2016) Data elevator: low-contention data movement in hierarchical storage system. In: *2016 IEEE 23rd international conference on high performance computing (HiPC)*, Hyderabad, India, 19–22 December 2016, pp. 152–161.

Duan R, Prodan R and Fahringer T (2005) DEE: a distributed fault tolerant workflow enactment engine for grid computing. In: Yang LT, Rana OF, Di Martino B, et al. (eds) *High Performance Computing and Communications. Lecture Notes in Computer Science*. Berlin: Springer Berlin Heidelberg, pp. 704–716.

Duda RO, Hart PE and Stork DG (2012) *Pattern Classification*. Hoboken: John Wiley & Sons.

Durillo JJ, Prodan R and Fard HM (2012) MOHEFT: a multi-objective list-based method for workflow scheduling. In: *Proceedings of the 2012 IEEE 4th international conference on cloud computing technology and science (CloudCom)*, CLOUDCOM '12, Taipei, Taiwan, 3–6 December 2012, pp. 185–192. Washington, DC: IEEE Computer Society.

Elomaa T, Mannila H and Toivonen H (2002) *Principles of data mining and knowledge discovery*. In: *Proceedings 6th European conference, PKDD 2002*, Helsinki, Finland, 19–23 August 2002. Berlin, Heidelberg: Springer Berlin Heidelberg.

Fahringer T, Prodan R, Duan R, et al. (2005) ASKALON: a grid application development and computing environment. In: *Proceedings of the 6th IEEE/ACM international workshop on grid computing*, Seattle, WA, USA, 13 November 2005. Washington, DC: IEEE.

Fu S (2011) Performance metric selection for autonomic anomaly detection on cloud computing systems. In: *2011 IEEE global telecommunications conference—GLOBECOM 2011*, Kathmandu, Nepal, 5–9 December 2011, pp. 1–5. Washington, DC: IEEE.

Gil Y, Deelman E, Blythe J, et al. (2004) Artificial intelligence and grids: workflow planning and beyond. *IEEE Intelligent Systems*. Available at: https://scitech.isi.edu/wordpress/wp-content/papercite-data/pdf/gil2004ai.pdf (accessed 4 January 2019).

Gil Y, Ratnakar V, Deelman E, et al. (2007) Wings for Pegasus: creating large-scale scientific applications using semantic representations of computational workflows. In: *Proceedings of the national conference on artificial intelligence*, Vancouver, British Columbia, Canada, 22–26 July 2007, p. 1767. Menlo Park, CA; Cambridge, MA; London: AAAI Press; MIT Press.

Gil Y, Ratnakar V, Kim J, et al. (2011) Wings: intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems* 26(1): 62–72.

Goderis A, Brooks C, Altintas I, et al. (2007) Composing different models of computation in Kepler and Ptolemy II. In: Shi Y, van Albada GD, Dongarra J, et al. (eds) *2nd international workshop on workflow systems in e-Science*, Beijing, China, 27–30 May 2007.

Goecks J, Nekrutenko A and Taylor J (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* 11(8): R86.

Goldstein M and Uchida S (2016) A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS One* 11(4): e0152173.

Gunter D, Deelman E, Samak T, et al. (2011) Online workflow management and performance analysis with Stampede. In: *2011 7th international conference on network and service management (CNSM)*, Paris, France, 24–28 October 2011, pp. 1–10.

Hathaway RJ and Bezdek JC (1986) Local convergence of the fuzzy c-means algorithms. *Pattern Recognition* 19(6): 477–480.

He Z, Xu X and Deng S (2003) Discovering cluster-based local outliers. *Pattern Recognition Letters* 24(9): 1641–1650.

Huang D, Liu Q, Klasky S, et al. (2019) Harnessing data movement in virtual clusters for in-situ execution. *IEEE Transactions on Parallel and Distributed Systems* 30(3): 615–629.

Ibidunmoye O, Hernández-Rodriguez F and Elmroth E (2015) Performance anomaly detection and bottleneck identification. *ACM Computing Surveys* 48(1): 4:1–4:35.

Jain N, Bhatele A, Robson MP, et al. (2013) Predicting application performance using supervised learning on communication features. In: SC '13: *ACM/IEEE international conference for high performance computing, networking, storage and analysis*, Denver, CO, USA, 17–22 November 2013. Washington, DC: IEEE Computer Society.

Jinka P and Schwartz B (2015) *Anomaly Detection for Monitoring: A Statistical Approach to Time Series Anomaly Detection*. Newton: O'Reilly Media.

Jolliffe IT and Cadima J (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374(2065): 20150202.

Jones AC (2007) Workflow and biodiversity e-Science. In: Taylor IJ, Deelman E, Gannon DB, et al. (eds) *Workflows for*

e-Science: Scientific Workflows for Grids. London: Springer London, pp. 80–90.

Kiran M, Murphy P, Monga I, et al. (2015) Lambda architecture for cost-effective batch and speed big data processing. In: *2015 IEEE international conference on big data (big data)*, Santa Clara, CA, USA, 29 October–1 November 2015, pp. 2785–2792. Washington, DC: IEEE.

Kohonen T, Schroeder MR and Huang TS (2001) *Self-Organizing Maps*. Berlin, Heidelberg: Springer-Verlag. ISBN: 3540679219.

Kotliar M, Kartashov A and Barski A (2018) CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language. *bioRxiv* 2018: 249243. DOI: 10.1101/249243.

Kourtellis N, De Francisci Morales G and Bifet A (2019) Large-scale learning from data streams with apache SAMOA. In: Sayed-Mouchaweh M (ed) *Learning from Data Streams in Evolving Environments: Methods and Applications*. Cham: Springer International Publishing, pp. 177–207.

Krol D, Ferreira da Silva R, Deelman E, et al. (2016) Workflow performance profiles: development and analysis. In: *Euro-Par 2016: parallel processing workshops*, Grenoble, France, 24–26 August 2016, pp. 108–120.

Kwan-Liu M (2009) In situ visualization at extreme scale: challenges and opportunities. *IEEE Computer Graphics and Applications* 29(6): 14–19.

Lee YC, Han H, Zomaya AY, et al. (2015) Resource-efficient workflow scheduling in clouds. *Knowledge-Based Systems 80:* 153–162. *25th anniversary of Knowledge-Based Systems.*

Li P, Castrillo JI, Velarde G, et al. (2008) Performing statistical analyses on quantitative data in Taverna workflows: an example using R and maxdBrowse to identify differentially-expressed genes from microarray data. *BMC Bioinformatics* 9: 334.

Liu J, Pacitti E, Valduriez P, et al. (2015) A survey of data-intensive scientific workflow management. *Journal of Grid Computing* 13(4): 457–493.

Lorena AC, Jacintho LFO, Siqueira MF, et al. (2011) Comparing machine learning classifiers in potential distribution modelling. *Expert Systems with Applications* 38(5): 5268–5275.

Mandal A, Ruth P, Baldin I, et al. (2016) Toward an end-to-end framework for modeling, monitoring, and anomaly detection for scientific workflows. In: *Workshop on large-scale parallel processing (LSPP 2016)*, Chicago, IL, 23–27 May 2016, pp. 1370–1379.

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, 1967, pp. 281–297.

Matsunaga A and Fortes JAB (2010) On the use of machine learning to predict the time and resources consumed by applications. In: *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing,* CCGRID '10, Melbourne, Victoria, Australia, 17–20 May 2010, pp. 495–504. Washington, DC: IEEE Computer Society.

Malawski M, Figiela K, Bubak M, et al. (2015a) Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. *Scientific Programming* 2015(Article 5).

Malawski M, Juve G, Deelman E, et al. (2015b) Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. *Future Generations Computer Systems: FGCS* 48: 1–18.

Meng F, Zhou L, Ma X, et al. (2014) vCacheShare: automated server flash cache space management in a virtualization environment. In: *2014 USENIX annual technical conference (USENIX ATC '14)*, Philadelphia, PA, 19–20 June 2014, pp. 133–144.

Missier P, Soiland-Reyes S, Owen S, et al. (2010) Taverna, reloaded. In: Gertz M and Ludäscher B (eds) *Scientific and Statistical Database Management. Lecture Notes in Computer Science*. Berlin: Springer Berlin Heidelberg, pp. 471–481.

Muniswamy-Reddy K-K, Braun U, Holland DA, et al. (2009) Layering in provenance systems. In: *Proceedings of the 2009 conference on USENIX annual technical conference,* USENIX '09, San Diego, CA, USA, 14–19 June 2009, pp. 10–10. Berkeley, CA: USENIX Association.

Muniswamy-Reddy K-K, Macko P and Seltzer MI (2010) Provenance for the Cloud. *FAST* 10: 15–14.

Nemirovsky D, Arkose T, Markovic N, et al. (2017) A machine learning approach for performance prediction and scheduling on heterogeneous CPUs. In: *2017 29th international symposium on computer architecture and high performance computing (SBAC-PAD)*, Campinas, Brazil, 17–20 October 2017, pp. 121–128.

Pal NR, Bezdek JC and Hathaway RJ (1996) Sequential competitive learning and the fuzzy c-means clustering algorithms. *Neural Networks: The Official Journal of the International Neural Network Society* 9(5): 787–796.

Pietri I and Sakellariou R (2014) Energy-aware workflow scheduling using frequency scaling. In: *3rd international workshop on power-aware algorithms, systems, and architectures*, Minneapolis, MN, 10 September 2014.

Pietri I and Sakellariou R (2019) A Pareto-based approach for CPU provisioning of scientific workflows on clouds. *Future Generations Computer Systems: FGCS* 94: 479–487.

Pietri I, Juve G, Deelman E, et al. (2014) A performance model to estimate execution time of scientific workflows on the cloud. In: *Proceedings of the 9th workshop on workflows in support of large-scale science*, WORKS '14, Piscataway, NJ, USA, 16 November 2014, pp. 11–19. Washington, DC: IEEE Press.

Plankensteiner K, Prodan R, Fahringer T, et al. (2009) Fault detection, prevention and recovery in current grid workflow systems. In: Yahyapour R, Talia R and Meyer N (eds) *Grid and Services Evolution*. Boston, MA: Springer US, pp. 1–13.

Poehlman WL, Rynge M, Branton C, et al. (2016) OSG-GEM: gene expression matrix construction using the open science grid. *Bioinformatics and Biology Insights* 10: 133–141.

Prathamesh G, Mandal A, Ruth P, et al. (2016) Anomaly detection for scientific workflow applications on networked clouds. In: *IEEE 2016 international conference on high performance computing & simulation (HPCS 2016)*, Innsbruck, Austria, 18–22 July 2016.

Qin J and Fahringer T (2012) Abstract workflow description language. *Scientific Workflows: Programming, Optimization, and*

Synthesis with ASKALON and AWDL. Berlin, Heidelberg: Springer-Verlag, pp. 31–61.

Ramakrishnan A, Singh G, Zhao H, et al. (2007) Scheduling data-intensive workflows onto storage-constrained distributed resources. In: *Seventh IEEE international symposium on cluster computing and the grid—CCGrid 2007*, Rio De Janeiro, Brazil, 14–17 May 2007.

Samak T, Gunter D, Goode M, et al. (2011a) Failure prediction and localization in large scientific workflows. In: *Proceedings of the 6th workshop on workflows in support of large-scale science*, Seattle, WA, 14 November 2011, pp. 107–116. New York, NY: ACM.

Samak T, Gunter D, Goode M, et al. (2011b) Online fault and anomaly detection for large-scale scientific workflows. In: *2011 IEEE international conference on high performance computing and communications*, Banff, Canada, 2–4 September 2011, pp. 373–381.

Samak T, Gunter D, Goode M, et al. (2013) Failure analysis of distributed scientific workflows executing in the cloud. In: *Proceedings of the 8th international conference on network and service management,* CNSM '12, Laxenburg, Austria, 22–26 October 2013, pp. 46–54. Laxenburg, Austria: International Federation for Information Processing.

Simmhan YL, Plale B and Gannon D (2005) A survey of data provenance in e-Science. *ACM SIGMOD Record* 34(3): 31–36.

Simmhan YL, Plale B, Gannon D, et al. (2006) Performance evaluation of the karma provenance framework for scientific workflows. In: Moreau L and Foster I (eds) *Provenance and Annotation of Data. Lecture Notes in Computer Science*. Berlin: Springer Berlin Heidelberg, pp. 222–236.

Singh G, Vahi K, Ramakrishnan A, et al. (2007) Optimizing workflow data footprint. *Scientific Programming* 15(4): 249–268.

Spafford KL and Vetter JS (2012) Aspen: a domain specific language for performance modeling. In: *Proceedings of the international conference on high performance computing, networking, storage and analysis*, SC '12, Los Alamitos, CA, USA, 10–16 November 2012, pp. 84:1–84:11. Washington, DC: IEEE Computer Society Press.

Spark A (2014) Apache spark streaming. *Spark streaming programming guide*. Available at: https://spark.apache.org/streaming/ (accessed 4 January 2019).

Steuer R, Kurths J, Daub CO, et al. (2002) The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics* 18(Suppl 2): S231–S240.

Subedi P, Davis P, Duan S, et al. (2018) Stacker: an autonomic data movement engine for extreme-scale data staging-based in-situ workflows. In: *Proceedings of the international conference for high performance computing, networking, storage, and analysis,* SC '18, Piscataway, NJ, USA, 11–16 November 2018, pp. 73:1–73:11. Washington, DC: IEEE Press.

Taylor I, Shields M, Wang I, et al. (2007) The triana workflow environment: architecture and applications. In: Taylor I, Deelman E and Gannon D, et al. (eds) *Workflows for E-Science*. London: Springer, pp. 320–339.

Tovar B, da Silva RF, Juve G, et al. (2018) A job sizing strategy for high-throughput scientific workflows. *IEEE Transactions on Parallel and Distributed Systems* 29(2): 240–253.

Usman SA, Kehl MS, Nitz AH, et al. (2015) An improved pipeline to search for gravitational waves from compact binary coalescence. arXiv:1508.02357 [astro-ph, physics: gr-qc]. Available at: http://arxiv.org/abs/1508.02357 (accessed 4 January 2019).

Wang T, Oral S, Pritchard M, et al. (2015) TRIO: Burst Buffer Based I/O Orchestration. In: *2015 IEEE International conference on cluster computing*, Chicago, IL, USA, 8–11 September 2015, pp. 194–203. Washington, DC: IEEE.

Wang T, Wei J, Zhang W, et al. (2014) Workload-aware anomaly detection for Web applications. *The Journal of systems and software* 89: 19–32.

Weiss NA and Weiss CA (2012) *Introductory Statistics*. London: Pearson Education.

Weitzel D, Bockelman B, Brown DA, et al. (2017) Data Access for LIGO on the OSG. In: *Proceedings of the practice and experience in advanced research computing 2017 on sustainability, success and impact*, New Orleans, LA, 9 July 2017, p. 24. New York, NY: ACM.

Wolstencroft K, Haines R, Fellows D, et al. (2013) The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research* 41: W557–W561.

Zhang OQ, Kirchberg M, Ko RKL, et al. (2011) How to track your data: the case for cloud computing provenance. In: *2011 IEEE third international conference on cloud computing technology and science*, Athens, Greece, 29 November–1 December 2011, pp. 446–453.

Zhou AC, He B, Cheng X, et al. (2015) A declarative optimization engine for resource provisioning of scientific workflows in IaaS clouds. In: *Proceedings of the 24th international symposium on high-performance parallel and distributed computing,* HPDC '15, Portland, OR, 15–19 June 2015, pp. 223–234. New York, NY: ACM.

Author biographies

Ewa Deelman received her PhD in Computer Science from the Rensselaer Polytechnic Institute in 1998 in the area of parallel computing. Following a postdoc at the UCLA Computer Science Department, she joined the University of Southern California's Information Sciences Institute (ISI) in 2000, where she is serving as a Research Director and is leading the Science Automation Technologies group. Her research focuses on distributed systems with a particular emphasis on workflow technologies. She is also a Research Professor at the USC Computer Science Department and an IEEE Fellow.

Anirban Mandal serves as a Research Scientist at RENCI, UNC-Chapel Hill and leads several projects in the network research and infrastructure group. His research interests include resource provisioning, scheduling, performance analysis and measurements for distributed computing systems, cloud computing, and scientific workflows. Prior to

joining RENCI, he earned his PhD degree in Computer Science from Rice University in 2006.

Ming Jiang received his PhD in Computer Science and Engineering from The Ohio State University in 2005 in the area of scientific visualization and data mining. He joined Lawrence Livermore National Laboratory (LLNL) as a postdoc in 2005 and worked on large-scale image and video processing. He is currently a computer scientist at the Center for Applied Scientific Computing at LLNL. His current research is focused on exploiting machine learning and Big Data analytics for improving large-scale HPC simulations.

Rizos Sakellariou obtained his PhD from the University of Manchester in 1997. Following brief spells with Rice University and the University of Cyprus, since 2000 he has been with the University of Manchester where he is currently Professor of Computer Science leading a laboratory that carries out research on a range of parallel and distributed systems topics with an emphasis on efficient management of resources.