Contextual linking between workflow provenance and system performance logs

Elias el Khaldi Ahanach, Spiros Koulouzis, Zhiming Zhao
elias.el.khaldi@gmail.com, {S.Koulouzis|Z.Zhao}@uva.nl
University of Amsterdam,
Amsterdam, The Netherlands

Index Terms—**scientific workflow, provenance, system logs**

***Abstract*—When executing scientific workflows, anomalies of the workflow behavior are often caused by different issues such as resource failures at the underlying infrastructure. The provenance information collected by workflow management systems only captures the transformation of data at the workflow level. Analyzing provenance information and apposite system metrics requires expertise and manual effort. Moreover, it is often time-consuming to aggregate this information and correlate events occurring at different levels of the infrastructure. In this paper, we propose an architecture to automate the integration among workflow provenance information and performance information from the infrastructure level. Our architecture enables workflow developers or domain scientists to effectively browse workflow execution information together with the system metrics, and analyze contextual information for possible anomalies.**

I. Introduction

A complex scientific workflow often consists of many services, which are deployed on distributed infrastructures [1]. The runtime behavior of the workflow, e.g., monitored by the underlying infrastructure, is important for analyzing the workflow's provenance, in particular when the workflow has an unexpected performance issue or failure. On the one hand provenance (PROV)[1] is often used in scientific workflows to capture the transformation of data and therefore provide reproducibility. On the other hand monitoring systems provide metrics about the usage of resources, e.g., CPU usage or memory consumption. Those metrics can be useful for workflow developers to investigate the workflow behavior at the low-level resources[2]. However, the provenance and system metrics are provided by different sources, which makes the integrated analysis difficult and time-consuming.

However, it is very challenging to analyze the workflow performance, due to difficulty in gathering and analyzing performance metrics across distributed infrastructures. Moreover, the workflow provenance and the system logs are provided by Workflow Management System (WFMS) and the underlying infrastructure, and they contain different information.

In this poster, we propose a context-aware information integration and exploration framework for users to effectively investigate possible workflow execution anomalies or bottlenecks by combining provenance with available system metrics.

[1]https://www.w3.org/TR/prov-overview/

II. Related Work

The steps in scientific workflows are often implemented as web services and hosted distributed infrastructures like Cloud[3]. It is often hard to trace execution bottlenecks which are caused by underlying infrastructure. Cloud infrastructures nowadays can provide sophisticated monitoring tools for virtual environments; however, those monitoring information are often at scope of virtual machines, containers and network, which are difficult for application developers to link with the context of high-level workflow logic and to seamlessly analyse casualty between them.

Prodan et al., propose a model for estimating the ideal lowest execution time of a workflow and then compare it with the workflow's measured execution time [4]. The solution relies on a Grid resource scheduler GRAM to collect the state of each workflow task [5]. Ferreira et al, classifies the state of each task in a workflow and apply the appropriate rule to mitigate failures[6]; however, resource scaling were not addressed since Grid environments assign tasks in a static resource. Madougou et al., analysed task failures in an e-Infrastructure, based on history of workflow activity from the e-BioInfra platform[7]; however, the analysis only focuses on underlying resource usage and did not connect with the higher level workflow deception tasks. The work presented in [8] proposes an online mechanism for detecting anomalies while executing scientific workflows on clouds. The authors use a framework to collect online monitoring time-series data from workflow tasks and the infrastructure. This approach is tightly coupled with the Pegasus WFMS which depends on specific worker nodes to execute workflows tasks.

An effective solution is thus needed to enable workflow users to 1)analyse the execution time of service-based scientific workflows, 2)detect bottlenecks that cause workflow performance degradation and 3) intuitively perform the analysis.

III. Architecture

We propose a Cross-context Workflow Execution Analyzer (CWEA), made the components shown in Fig. 1. **The Workflow Context Data Retriever (WCDR)** extracts the name, start-time, and end-time of each web service call described in the workflow from the provenance data. **The Resource Context Data Retriever (RCDR)** queries the corresponding hosts to retrieve available performance data within the web service's call time-ranges, using the service endpoints obtained

by the WCDR. Therefore, these two components perform the crucial task of combining provenance and performance data. **Workflow Execution Analyzer (WFEA)** identifies the most time-consuming web services within the context of a workflow execution. A **GUI** allows users to visualize the workflow execution with the performance metrics of the underlying resources. The CWEA may be integrated with any WFMS that is compatible with the PROV specification.

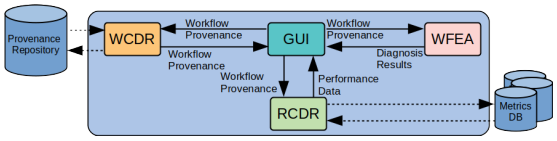

Fig. 1. The overarchitecture of CWEA

IV. Prototype and Usability

To demonstrate the usability of CWEA, we implemented a simple REST service with three methods: 1) a lightweight only using very little resources named *LWx*, 2) a CPU intensive intended to exhaust the system's CPU resources called *CPUx* and 3) a memory intensive that makes heavy use the system's of memory named *Memx* . We deployed that service on three Virtual Machines (VMs) (labeled A, B and C) together with cAdvisor [9], a performance metrics collector and Prometheus[10], a time-series database used to store performance metrics.

On a separate VM we deployed the CWEA and its components to gather metrics from different hosts and to perform the visualization.

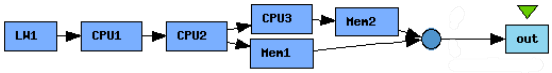

Fig. 2. Example workflow. VM A hosted tasks LW1, CPU2 and Mem2. VM B tasks CPU1 and Mem1. VM C task CPU3.

Using the workflow in Fig. 2, the CWEA creates a number of outputs. Fig. 3 presents the duration and execution time of each task. We see the execution timeline of the workflow and the time required to execute each task. In this timeline, each task is presented by a different color bar which is also highlighted in the resource usage graphs below. Fig. 4 visualises the CPU used by each task with the color that corresponds to each task. Fig. 5 present the memory used by each task. From the results presented, we see that the most time-consuming task is the mem_intensive. With this information we can better adapt the underling infrastructure and assign that task to a more suitable VM.

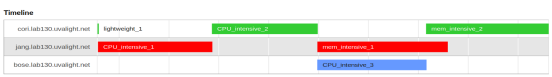

Fig. 3. Workflow execution timeline.

V. Conclusions and Future Work

We presented a CWEA that allows developers to analyze the execution of service-based workflows and visualize possible bottlenecks related with the infrastructure. We demonstrated the usage via a test workflow. However, it is important to test larger scale workflows. The CWEA relies on the WFMS to collect provenance data. Having performance data from many workflow executions will enable us to make use of statistical and AI algorithms to detect and predict possible workflow execution failures due to errors in the resource infrastructure.

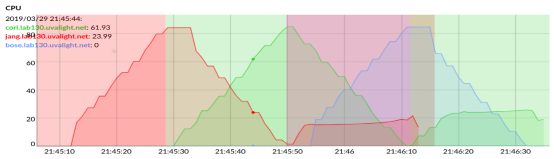

Fig. 4. CPU usage.

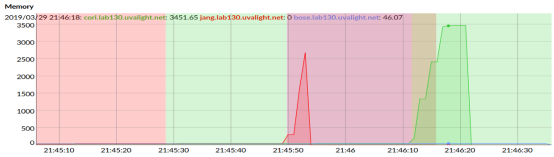

Fig. 5. Memory usage.

Acknowledgment

This work was supported by the EU's Horizon 2020 research and innovation programme under grant agreements No. 824068 (ENVRI-FAIR), 654182 (ENVRIPLUS) and 825134(ARTICONF).

References

[1] Z. Zhao, A. Belloum, C. de Laat, P. Adriaans, and B. Hertzberger, "Distributed execution of aggregated multi domain workflows using an agent framework," in *2007 IEEE Congress on Services (Services 2007)*, pp. 183–190, IEEE.

[2] S. Koulouzis, A. S. Belloum, M. T. Bubak, Z. Zhao, M. ivkovi?, and C. T. de Laat, "SDN-aware federation of distributed data," vol. 56, pp. 64–76.

[3] J. Wang, A. Taal, P. Martin, Y. Hu, H. Zhou, J. Pang, C. de Laat, and Z. Zhao, "Planning virtual infrastructures for time critical applications with multiple deadline constraints," vol. 75, pp. 365–375.

[4] R. Prodan and T. Fahringer, "Overhead analysis of scientific workflows in grid environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 378–393, March 2008.

[5] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," *Journal of computer science and technology*, vol. 21, no. 4, p. 513, 2006.

[6] R. Ferreira da Silva, T. Glatard, and F. Desprez, "Self-healing of operational workflow incidents on distributed computing infrastructures," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 318–325, May 2012.

[7] S. Madougou, S. Shahand, M. Santcroos, B. van Schaik, A. Benabdelkader, A. van Kampen, and S. Olabarriaga, "Characterizing workflow-based activity on a production e-infrastructure using provenance data," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 1931 – 1942, 2013. Including Special sections: Advanced Cloud Monitoring Systems & The fourth IEEE International Conference on e-Science 2011 e-Science Applications and Tools & Cluster, Grid, and Cloud Computing.

[8] P. Gaikwad, A. Mandal, P. Ruth, G. Juve, D. Krl, and E. Deelman, "Anomaly detection for scientific workflow applications on networked clouds," in *2016 International Conference on High Performance Computing Simulation (HPCS)*, pp. 645–652, July 2016.

[9] "cadvisor (container advisor), official github page." https://github.com/google/cadvisor. Accessed: 2019-03-28.

[10] "Prometheus, an open-source systems monitoring and alerting toolkit.." https://prometheus.io/docs/introduction/overview/. Accessed: 2019-03-28.