Containerized Environment for Reproducibility and Traceability of Scientific Workflows

Paula Olaya
University of Tennessee, Knoxville
polaya@vols.utk.edu

Jay Lofstead (mentor)
Sandia National Laboratory
gflofst@sandia.gov

Michela Taufer (advisor)
University of Tennessee, Knoxville
mtaufer.utk.edu

***Abstract*—Scientists rely on simulations to study natural phenomena. Trusting the simulation results is vital to develop sciences in any field. One approach to build trust is to ensure the reproducibility and traceability of the simulations through the annotation of executions at the system-level; by the generation of record trails of data moving through the simulation workflow. In this work, we present a system-level solution that leverages the intrinsic characteristics of containers (i.e., portability, isolation, encapsulation, and unique identifiers). Our solution consists of a containerized environment capable to annotate workflows, capture provenance metadata, and build record trails. We assess our environment on four different workflows and measure containerization costs in terms of time and space. Our solution, built with a tolerable time and space overhead, enables transparent and automatic provenance metadata collection and access, an easy-to-read record trail, and tight connections between data and metadata.**

1. Introduction

As computational simulations are becoming more influential in science, trusting the simulation results becomes vital to the integrity of the scientific discovery. Scientists can trust results when those results can be reproduced and traced. One general approach to meet both requirements is to annotate the simulations' executions and build record trails of data moving across a simulation workflow. Past efforts such as PASS [1] have advocated for the need to build record trails at the system-level because such an approach is general across applications and offers full visibility into system-wide behavior. A key hindrance to any system-level solution has been the technology limits for building and managing record trails. The evolution from virtual machines to containers has opened new opportunities for system-level solutions.

In this work, we build upon the PASS vision and leverage cutting edge container technologies to collect provenance metadata at the system level. Specifically, we build a prototype of an application-agnostic containerized environment that encapsulates each component of a scientific workflow (i.e., data and applications) in individual containers and collects record trailers of the application's workflows. Our prototype implementation features zero-copy data transfer between containers, requires no modification of the underlying applications, and automatically links system metadata to workflow components. The information captured in a record trail includes input, application, and output documentation as well as command line after each execution; the combination of this metadata is essential for the simulation reproduction. We measure the containerization costs of our environment in terms of time and space.

The contributions of this work are as follows:

- The novel solution to collect provenance metadata at the system-level by leveraging container technology.
- The prototype of an application-agnostic containerized environment that enables transparent and automatic metadata collection, access, easy-to-read record trails, and tight connections between data and metadata.
- The demonstration of how our environment can deliver provenance metadata that support workflow reproducibilty.

2. Building Containerized Environments

Our solution to create a general application-agnostic containerized environment for reproducibilty and traceability of scientific workflows consists of six steps.

1. Leverage container technologies to host workflow executions: Leveraging container technologies raises two key questions: (a) what are the benefits of container technologies and (b) which specific container technology best fits our needs in high performance computing. For the first question we consider the properties that container technology offers including ***portability***, ***isolation***, ***encapsulation***, and ***unique identifications.*** For the second question we select Singularity [2] because it does not require administrative privileges, making Singularity usable across HPC platforms for scientists.

2. Decouple a workflow into components: The taxonomy of the workflow targeted in this work consists of data (i.e., input and output) and applications (non-distributed). Data is the compilation of individual units of information. Applications are programs that execute transformations on data. We consider data and applications as individual components of a workflow.

3. Encapsulate a single component in a single container: Our environment supports two types of containers: data container and application container. We encapsulate

each workflow component into one of the two container types.

4. Connect the containerized components to rebuild the original workflow: The original Singularity implementation allows *two-copy data transfer* between two containers through the host storage. We augment Singularity to support direct transfer copy or *zero-copy data transfer*. Our extension allows our environment to bind mount direct paths inside containers through their namespaces to transfer data, avoiding third party transfer through the host.

5. Capture containers' metadata: We design a Singularity plugin to capture information across the workflow components during execution. The plugin creates a provenance metadata file. With the file information we can build the record trail backwards and determine the communication patterns of the considered execution.

6. Allocation of the record trail: After the record trail is created, we attach its parts to the workflow. Our Singularity plugin allocates the metadata as a new partition of the output data container, offering consistency and relation between data and metadata.

We assess the effectiveness of our prototype for four increasingly complex workflows, ranging from simple visualization applications such as gnuplot to machine learning applications, in particular weighted k-Nearest Neighbors (kKNN) and random forest (RF). We show that our solution can build workflow record trails at the system-level for four scenarios in an automatic, easy-to-read way, allowing a tight connection between data and metadata.

Figure 1 shows an example of containerized environment for a workflow with a single application (gnuplot benchmark), single input (text data files), and single output (plots).

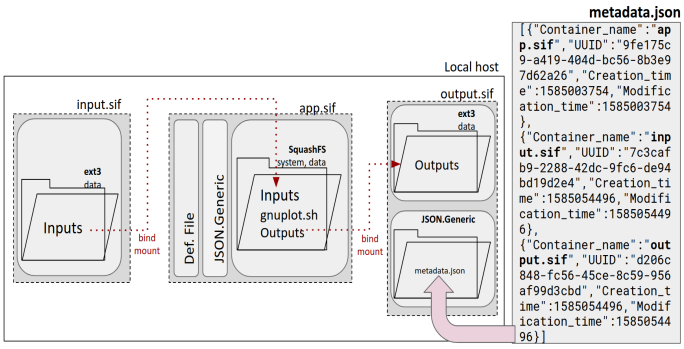

Figure 1. Containerized environment for a simple workflow comprising a single application (gnuplot benchmark), single input (text data files), and single output (plots).

3. Time and Memory Costs

We measure the cost in time and space to deploy our containerized environment for four increasingly complex workflows: two workflows based on a simple visualization applications such as gnuplot, and two workflows based on machine learning applications using kKNN and RF respectively. We measure the wall clock time over 500 executions for each workflow, before and after containerization; we observe that with more complex and larger workflows (e.g., RF), the wall clock time overhead becomes negligible (0.7%). We also compare the size of the workflow components (data and application) before and after containerization; we observe that our environment identifies and packages the software stack and its dependencies, guaranteeing to have a reproducible and transparent software system no matter on which platform the workflow is executed. We observe how the space overhead of the data and application containers is linked to the selection of OS, software stack and file system.

4. Conclusion and Future Work

We present a prototype of an application-agnostic containerized environment. Our environment supports (a) **reproducibility** by automatic collecting provenance metadata for each workflow component and encapsulation of the software stack in the application container; (b) **traceability** of executions by capturing data and their provenance metadata, assembling record trails of data movement, and uniquely identifying each workflow component. The new knowledge collected with our environment comes at a manageable cost in time and space for large workflows. Future work includes generalizing the metadata specifications to target a broader range of workflows as well as adding more detailed provenance metadata for each workflow component.

Acknowledgments

This research was supported by NSF awards CCF 1900888 and by Sandia National Laboratories DE-NA0003525. The authors acknowledge the Singularity team, specially Cedric and Ian Kaneshiro, for the support.

References

[1] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ATEC '06, (USA), p. 4, USENIX Association, 2006.

[2] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PLOS ONE*, vol. 12, pp. 1–20, 05 2017.