Data Pallets For Traceable Data

Jay Lofstead, Joshua Baker, Andrew J. Younge
Sandia National Laboratories

***Abstract*—The ability to trace any given output of scientific HPC workloads back to the originating application and input deck is a "holy grail" for reproducible computational science. Provenance and workflow systems have tried for decades to achieve this goal, but have produced fragile and/or partial solutions limiting adoption.**

The widespread adoption of container technology, along with a few modifications collectively we call *Data Pallets*, can achieve this traceability invisibly to the application and end user and allow immediate identification of production artifacts through simple data inspection.

I. Introduction

The application of the scientific method mandates the reproduction of any scholarly data to provide an expectation of explainability for any results, insights, and knowledge generated. The peer review process evaluates these artifacts in part to validate the claims and if a reasonable process was followed. Historically, achieving this standard with computational results has proven difficult. Bespoke hardware and software stacks are rarely available or replicable for others. As a first significant step towards solving this gap between practice and a standard scientific process, we present *Data Pallets*.

The adoption of container technology is largely due to the low overhead for isolated program execution on a platform and the ease for redeploying instances of said programs. However, the key container design aspects that matter are the unique identification hash code and the encapsulation. Through incorporating system support to ensure generated data is encapsulated into a new container we call a *data pallet*, and tagging it with the IDs for all mounted containers that were running when it was created, we can automatically generate provenance information that is inseparable from the data itself, while not perturbing the data contents or accessibility.

II. Design

The basic design extends the container system to automatically generate a new, writeable container each time the code writes to storage. As an initial proof of concept, we incorporate FUSE [1] to intercept basic filesystem calls, such as `mkdir`, and mount a newly created container (*data pallet*) at that location. Each data pallet has an associated hash which is accounted for and mantained as metadata within the container image mechanism.

III. Implementation

To simplify testing this idea, the Singularity [2] writeable container technology is leveraged. While the base-line Singularity writeable container must be created prior to starting Singularity to run the application, we have modified the environment to incorporate FUSE to intercept the `mkdir` command and mount the container into the Linux namespace at runtime. This accomplishes creating the container that will hold all of the generated output.

The provenance information relies on the new Singularity image format's ability to house multiple partitions. We use the base partition to store the data and add a second one, containing a JSON file of container IDs for the environment in use when this container was created.

IV. Initial Results

The initial results examine the space overheads of this approach. The basic container image imposes a 700 KB overhead using the ext3 format, as is required for the writeable format. An additional 1.1 MB is taken for the attribute partition. While this overhead is not minimal, compared to today's data sizes of 10s of GB per node, the resulting overhead is $< 0.1\%$.

We have also demonstrated using the Sandia Analysis Workbench workflow tool to deploy such a set of containers generating the desired output into a data pallet.

V. Implications and Future Work

There are some underlying base assumptions for this approach. First, we must maintain the containers for all applications for which we want to trace back data. Second, and more importantly, all input and configuration files for a simulation run should each be placed in their own container allowing unique identification. While this is not free, we believe that a few GBs stored with PBs of generated data is a negligible burden given the resulting benefits.

There are numerous challenges and issues that are still to be addressed. For example, one file per process workloads, such as Sandia's Sierra codes, the annotations (artifact IDs) must be maintained while merging and splitting data. There are a whole host of additional data management issues for the container proliferation as well. But as a first step, this offers a true "scientific" environment.

Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This work is funded through the LDRD program and ASC CSSE.

References

[1] M. Szeredi, "Fuse: Filesystem in userspace. 2005," *URL http://fuse. sourceforge. net*, 2005.

[2] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PloS one*, vol. 12, no. 5.