Containerized Environment for Reproducibility and Traceability of Scientific Workflows

Student: Paula Olaya* **Mentors:** Jay Lofstead [+], Michela Taufer *

*University of Tennessee, Knoxville, [+]Sandia National Laboratory

Problem Overview

- **Scientific** domains **use simulations** to understand and predict natural phenomena
- **Trusting the output** of these simulations **is vital** for the scientists
- Trust **requires reproducibility, replicability, transparency and traceability** of the simulation process and its results

  Hypothesis: **Annotating the workflow execution at system-level** provides a way to ensure reproducibility, replicability, transparency and traceability of simulations

- When we **annotate** the execution at **system-level**, we are able **to build the record trail of data** moving across the workflow
- This **record trail of data includes** every **workflow component** used to generate the new data (e.g., input datasets, applications, and parameter values)

Supported Workflows

We implement our prototype to support four workflows with two base applications:
- Visualization applications
- ML applications

Supported workflow taxonomy / Tested workflow applications
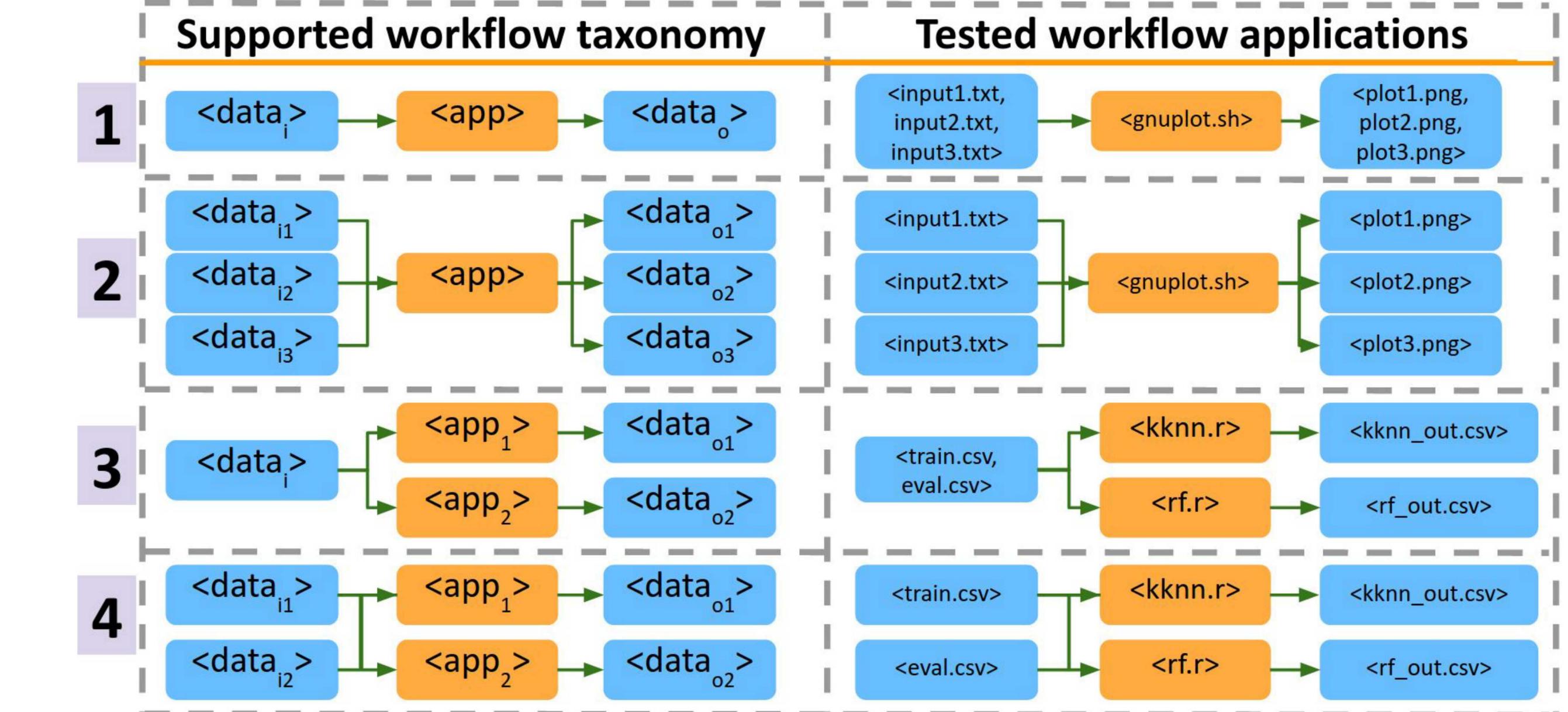
Our Solution: a Containerized Environment

Leverage cutting edge **container technologies** to address **metadata from the OS level** to build the record trail and ensure the reproduction and traceability of scientific workflows
- Create an **application-agnostic containerized environment**
- Capture **workflow record trails** at runtime

Why container technology?
- Portability: Immutable applications machine-agnostic
- Isolation: namespaces
- Encapsulation: sif format
- Unique identification: UUID

Why Singularity?
- Reproducible software stacks
- Mobility of compute
- Compatibility with complex architectures
- Security model

Data Container: Compressed data added as a single and independent partition

App Container: System with specific software stack built from recipe and added as a single and independent partition

For more information, please visit the following link:

Data container — $Data_i$ — **Metadata** — Dynamic → Static
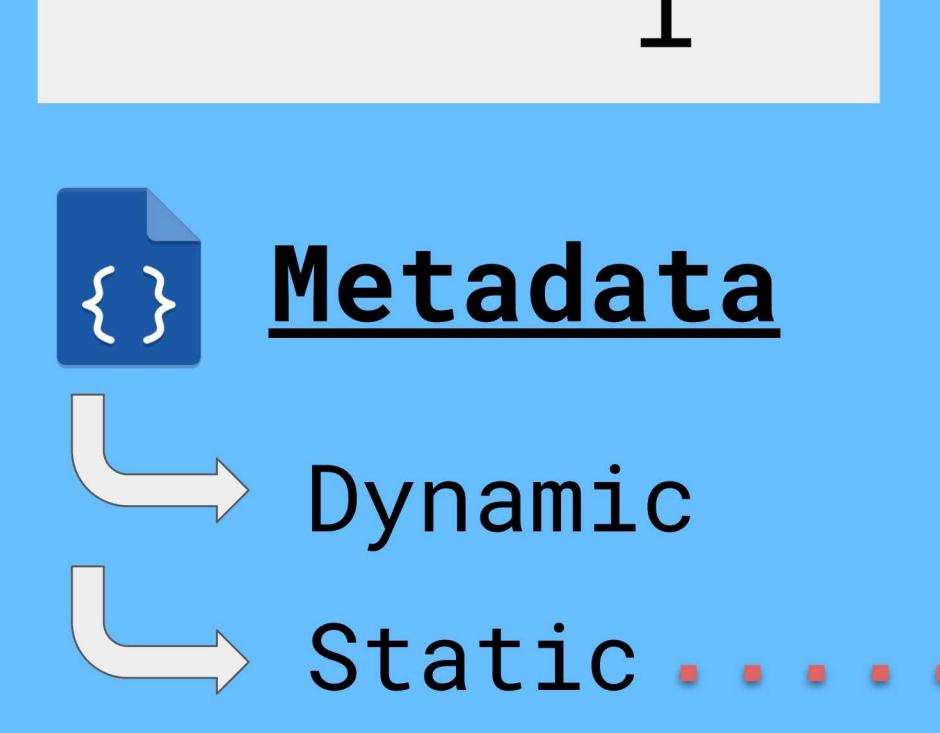
zero-copy →

App container — App — **Metadata** — Dynamic, Static

zero-copy →

Data container — $Data_o$ — **Metadata** — Dynamic → Static

Singularity Plugin

Static Metadata Input
Static Metadata App
Static Metadata Output
**Record trail**
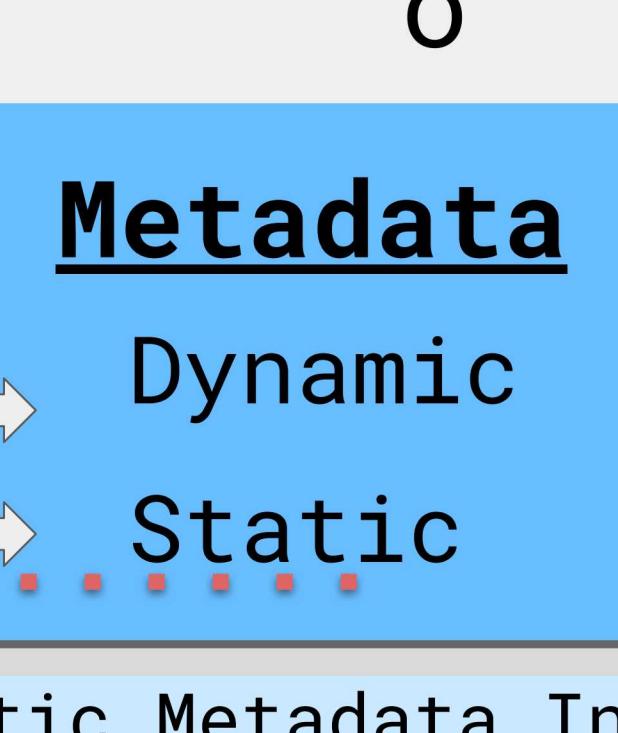
Zero-copy data transfer: It allows to bind mount direct paths inside containers a transfer data avoiding the third party transfer through the host

Singularity Plugin: Package dynamically loaded that interacts with all the workflow components to extract the metadata. It builds and allocates the record trail

Record Trail: Format: json Info: file name, UUIDs, creation + modification date

Costs: Time and Memory

Execution Time: Measure average wall-clock over 500 executions for workflow **4**

With a more complex and larger application like the RF model the **wall-clock time is tolerable** (0.7% overhead)

**Space Overhead:** Size of the workflow components (e.g., data and applications)
- Analyze overhead introduced by data and application containers size

Data container includes:
- Data files (i.e., Inputs, outputrf.csv, and outputkknn.csv)
- Workflow metadata (i.e. metadata.json)

	Original Workflow		Containerized Workflow	
	File	Size [KB]	File	Size [KB]
3	Inputs	12,072	inputs.sif	14,368
	outputkknn.csv	2,264	kknn_output.sif	4,132
	outputrf.csv	2,264	rf_output.sif	4,132

The **space overhead** for **data containers** is ~2 MB, **caused by the filesystem** used
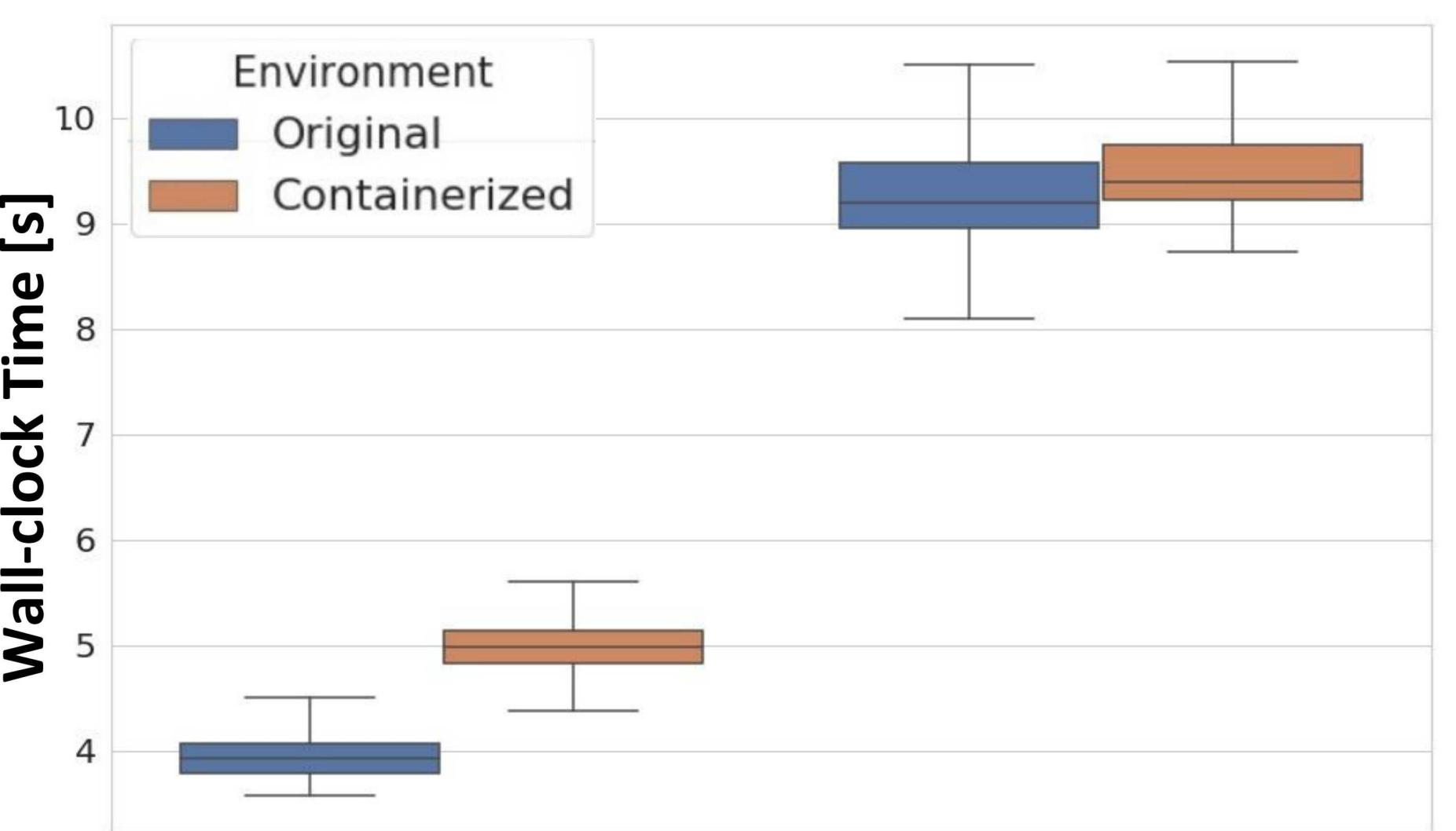
Application container includes:
- Application executable (i.e., gnuplotScript.sh)
- SW package includes system tools, system libraries, and settings (i.e., Ubuntu, gnuplot)

	Original Workflow		Containerized Workflow	
	File	Size [KB]	File	Size [KB]
1	gnuplotScript.sh	4	gnuPlotScript.sh	4
	gnuplot	139	app.sif gnuplot	139
	Ubuntu 16.04	>40000	Ubuntu 16.04	153,000

The **app container space overhead** is **driven by** the **software stack and OS**

Conclusions

Our containerized environment supports:
- No modification of the applications
- Zero-copy transfer of data between containers
- Record trail of different scenarios for workflow metadata
- Metadata attached to dataflow
- Tolerable overhead as workflow complexity increases

Future Work

- Leverage our containerized environment for a broader range of workflows
- Expand containers with automatic set-up, retrieved, and enabled reproducibility of the workflow

THE UNIVERSITY OF TENNESSEE KNOXVILLE