Software Defined Cyberinfrastructure for Data Management

Ryan Chard[1], Kyle Chard[2], Steven Tuecke[2], and Ian Foster[1,2]

[1]Computing, Environment, and Life Sciences, Argonne National Laboratory
[2]Computation Institute, University of Chicago and Argonne National Laboratory

***Abstract*—Scientific research is data-centric, relying on the acquisition, management, movement, analysis, and sharing of data. Proficiently managing the end-to-end lifecycle of scientific data is non-trivial and comprises many time consuming and mundane tasks. While individual tasks are not prohibitive, when done repeatedly and frequently they represent a significant strain on researchers. We posit that a better approach is to automate these tasks through a Software Defined Cyberinfrastructure. We have developed RIPPLE to provide such capabilities by automating research data management activities via a programmable and event-based cyber-environment. Users specify high-level management policies, such as data movement and metadata extraction, using intuitive *If-Trigger-Then-Action* rules. These rules are then autonomously, and reliably, executed and managed by RIPPLE.**

I. INTRODUCTION

The data-driven nature of modern research has inundated researchers with data. These data require complex processing pipelines, bridge multiple computing environments, and are touched by many researchers. The increasing complexity of data management presents new challenges to researchers as they must, often repeatedly, perform mundane and time consuming data management tasks, such as transferring, purging, cataloging, and sharing data—distracting from discovery.

A programmable cyber-environment can alleviate many of these challenges and enhance researcher productivity. Software Defined Cyberinfrastructure (SDCI) [1] leverages concepts from software defined networking to animate simple storage devices and create a dynamic fabric to facilitate automation. SDCI abstracts the underlying resources and empowers users to implement high-level control policies which are performed autonomously across arbitrary compute infrastructures.

This poster describes RIPPLE [2]—a realization of SDCI. A light-weight RIPPLE agent allows a device to integrate itself into the SDCI environment. This agent can be programmed (via distributed rules) to monitor a storage device, report relevant data events, and perform actions on behalf of users. A serverless cloud Web service processes reported events and manages the execution of actions across the environment, ranging from an end user's laptop to supercomputing facilities.

II. SOFTWARE DEFINED CYBERINFRASTRUCTURE

SDCI builds upon concepts used in Software Defined Networking (SDN). For example, SDCI separates the data and control planes, allowing the definition of abstract rules which are distributed across storage and compute devices in a similar fashion to SDN's distribution of policies to routers.

SDCI can be achieved by instrumenting existing cyberinfrastructure with programmable agents that can be configured to detect events, respond to these events, and perform various actions. A control plane manages these dynamic agents and facilitates the definition and distribution of new policies. Such a system creates an environment in which rules govern the movement and processing of data.

III. RIPPLE

RIPPLE establishes a distributed cyber-environment that can be controlled via simple *If-Trigger-Then-Action* rules. Researchers can encode best practice data management policies to be automatically carried out on their behalf. Once a rule is defined and enabled, a RIPPLE agent monitors a filesystem for relevant events and performs actions as required.

Architecture: Fig. 1 summarizes RIPPLE's architecture. A Python-based agent monitors filesystems and a cloud service processes events and manages the execution of actions. RIPPLE leverages a hybrid computing model in which events are filtered locally before being evaluated by the cloud service.

RIPPLE uses the Python Watchdog module to detect events. Watchdog provides an API and utilities to monitor many common filesystems via inotify, kqueue, FSEvents, and an OS-independent polling mechanism. We are actively extending these capabilities with new monitoring solutions for large scale filesystems, such as Lustre.

The RIPPLE cloud service provides a scalable platform to process events, manage agents, evaluate rules, and coordinate the execution of actions. The service exposes a REST API, supported by serverless Lambda functions. As events are reported to the service they are placed in a reliable AWS Simple Queue Service queue. Events trigger Lambda functions which process the event, evaluate rules, and finally instantiate appropriate actions to be executed, and, once completed, remove the event from the queue.

Actions are performed by the *job runners* supported by an agent, or by a cloud-based runner for performing actions via external services. The job runner API provides a common execution interface across arbitrary compute resources. Multiple job runners have been implemented, including those for: Docker, Singularity, SLURM, and raw subprocess commands. In addition, cloud-based runners have been developed for various external services, such as Globus (perform transfers and set permissions) and Amazon services (send emails and invoke Lambda functions).

Rules: RIPPLE's intuitive If-Trigger-Then-Action programming model, used to express rules, enables even non-technical users to create powerful data management workflows. Users can author rules by selecting from a list of predefined triggers and actions. They can optionally configure triggers and actions with custom conditions and parameters. For example, a trigger

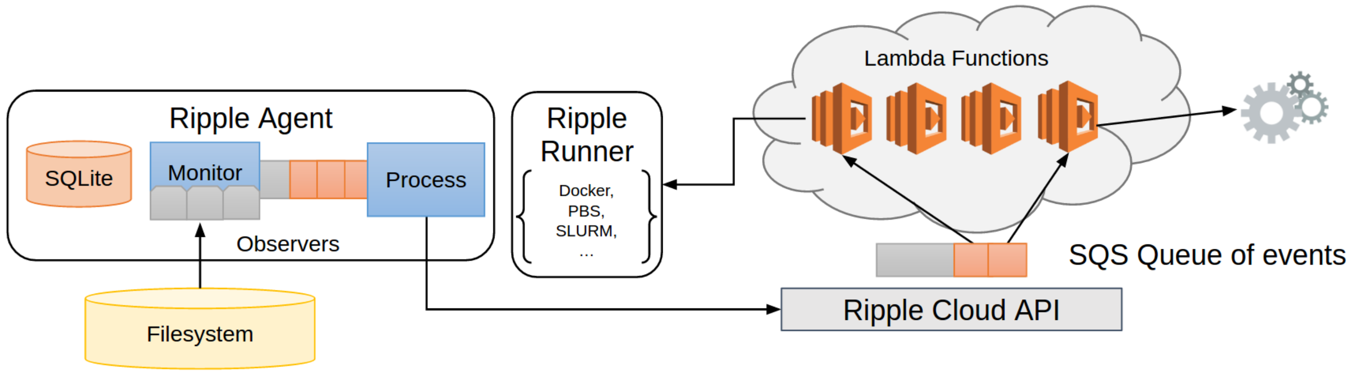

Fig. 1: RIPPLE architecture. Filesystem events are captured and evaluated by the local agent then reported to the cloud service, where Lambda functions evaluate rules and orchestrate actions.

may be configured to fire when a file with a *.csv* extension is created in a specific directory on a specific server. The action specifies the agent, or service, to perform the job, the type of job runner to use (for example a subprocess command), and the parameters to pass to the runner (e.g., Globus endpoints, email address, or the command to be executed). Triggers and actions can be set, or performed, on any of the enabled devices to which a user has access.

Use Cases: We have used RIPPLE to implement a variety of data management processes and pipelines. Many common management processes can be achieved with an individual RIPPLE rule. For example, one could define a rule to: modify a file, report results, extract metadata, execute an analysis script, or publish data to a catalog. More complex pipelines can be constructed by daisy-chaining rules, such that outputs trigger subsequent rules and establish cascading flows of data. These pipelines are capable of accomplishing complex, multi-step, tasks and can easily manage data across facilities or perform transformations using a range of resources and services.

As one example of RIPPLE's use, we have implemented a pipeline to perform tomographic reconstructions for Light Source science experiments. RIPPLE agents are deployed on both a computer near a beamline and at the NERSC computing facility. The workflow is as follows: The RIPPLE agent detects the creation of files as the beamline generates data and writes it to the local device. This initiates a Globus transfer [3] of the data to NERSC. A cloud-hosted monitor detects the completion state for the transfer and reports the event. A second rule is triggered, causing the NERSC-based RIPPLE agent to perform metadata extraction on the data and to create a batch submission file. A third rule then triggers (due to creation of the batch file) and results in submission of batch script to the Edison supercomputer's SLURM queue. Once processed, the resulting images are transferred back to the beamline's machine where a final rule notifies collaborators via email and publishes the results as a Globus Shared Endpoint.

IV. RELATED WORK

Existing automated data management services, such as Lustre's Robinhood [4] and the Integrated Rule-Oriented Data System (iRODS) [5], are rarely used by end-users. Rather, they are designed for administrators to configure infrastructure-wide policies. RIPPLE is user-oriented and designed to expose such functionality to even non-technical users. AbdelBaky et al. [6] proposed a programmable infrastructure service composition approach which leverages software-defined concepts to control service composition. This work complements SCDI and provides insight into the federation of diverse services.

V. SUMMARY

As research processes become increasingly data-driven and data lifecycles become yet more complex, researchers are faced with a myriad of new challenges. SDCI provides the means to address these challenges by enabling researchers to programmatically instrument storage devices with simple rules that can automate mundane and cumbersome tasks. RIPPLE employs local agents to collect filesystem events and a cloud service to evaluate rules and mange the execution of actions. We leverage a simple programming model to empower technical and non-technical users to instrument their devices with sophisticated functionality.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy under contract DE-AC02-06CH11357. We also acknowledge generous research credits provided by Amazon Web Services.

REFERENCES

[1] I. Foster *et al.*, "Software Defined Cyberinfrastructure," in *The 37th IEEE International Conference on Distributed Computing Systems*, 2017.
[2] R. Chard *et al.*, "RIPPLE: Home Automation for Research Data Management," in *The 37th IEEE International Conference on Distributed Computing Systems*, 2017.
[3] K. Chard *et al.*, "Efficient and secure transfer, synchronization, and sharing of big data," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 46–55, 2014.
[4] T. Leibovici, "Taking back control of HPC file systems with Robinhood Policy Engine," *arXiv preprint arXiv:1505.01448*, 2015.
[5] A. Rajasekar *et al.*, "iRODS Primer: Integrated rule-oriented data system," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 2, no. 1, pp. 1–143, 2010.
[6] M. AbdelBaky *et al.*, "Software-defined environments for science and engineering," *The International Journal of High Performance Computing Applications*, p. 1094342017710706, 2017.