Extended Abstract: Reactive, Interactive, and High Throughput Computation in the Agave Platform

Rion Dooley
Texas Advanced Computing Center
University of Texas, Austin
Austin, TX 78758-4497
Email: dooley@tacc.utexas.edu

Joe Stubbs
Texas Advanced Computing Center
University of Texas, Austin
Austin, TX 78758-4497
Email: jstubbs@tacc.utexas.edu

***Abstract*—The Agave Platform is an open Science-as-a-Service platform that empowers users to run code, manage data, collaborate meaningfully, and integrate easily with the world around them. In this extended abstract, we examine how new use cases have helped evolve the ways in which Agave supports code execution. We start out by identifying three new code execution paradigms identified in the past 2 years: Interactive, Reactive, and High Throughput computing. We then briefly describe the primary use cases driving each paradigm and how the platform evolved in response to them. Finally, we highlight areas of future work in the platform related to code execution.**

I. Introduction

The Agave Platform is an open Science-as-a-Service platform that empowers users to run code, manage data, collaborate meaningfully, and integrate easily with the world around them [1]. Since its launch in 2011, Agave has orchestrated hundreds of thousands of traditional HPC jobs on systems spanning the globe. In the last two years, the Platform has evolved to support three alternative execution paradigms which are poised to overtake traditional batch execution in cores used, hours burned, and total executions.

In this extended abstract, we highlight some of the real use cases that helped evolve Agaves code execution options.

II. Interactive Computing

Interactive computing involves live, ongoing interaction between a user and a running code. The nature of the code may vary from a Desktop application running on a remote visualization node, to a headless code responding to steering commands given through an external management process. The following use cases helped shape our addition of Interactive computing support in Agave.

A. Use case: Dynamic VNC sessions

The TACC Visualization Portal provides a single point of access to TACCs visualization resources [2]. One of the tools it provides is remote, interactive, web-based visualization over VNC. The challenge in creating VNC connection on TACCs HPC systems is that connectivity information is not known in advance. Interactive visualization nodes are allocated through the batch scheduler. Thus, the location of the VNC server is not known until the jobs starts. A second challenge is that the visualization nodes are not addressable from outside the HPC system itself. A tunnel must be established through a publicly addressable host to connect to the VNC server on the visualization node. This use case presented a challenge, as there was no natively supported way for applications to communicate runtime data through the Platform.

B. Use case: Application Steering

The iAnimal project constructed a home grown variant detection pipeline using the Agave Command Line Interface (CLI) [3] [4]. Individual runs of the pipeline were challenging to optimize due to the unpredictable runtimes on individual datasets. Additionally, Agave did not have a practical, secure way to communicate runtime information, such as a quality metric indicating success or failure, from within a run, so additional cleanup and analysis was needed to parse and interpret the output data after the job lifecycle completed.

C. Evolutionary impact: Runtime job events, internal and websocket notifications

These use cases led to the addition of two new features in the Platform. First, formal support was added for data pipelining where the output of one job could be referenced relative to the job URL and used as an input in another job request. Second support for a new template variable was added to the Jobs API to allow for user-defined runtime events to be thrown from a job. This enabled users to receive information from a running job through existing pubsub mechanisms native to the Platform.

III. Reactive Computing

Reactive computing is the process of defining an overall computational system with the context of one or more actions that should be performed in response to the occurrence of an event. This is an increasingly popular design pattern used to defer the complexity of building distributed systems and build more reliable, scalable systems. The following use cases helped shape our addition of Reactive computing support in Agave.

A. Use case: Event-based automation

The DesignSafe project provides a cloud storage solution for the natural hazards research community with the ability

Presented at Gateways 2016, San Diego Supercomputer Center, La Jolla, CA, November 2-3, 2016.
https://gateways2016.figshare.com/

to sync data between a user's Box [5] account and their DesignSafe storage account [6]. The challenge in doing so is that data changes must be synced from three sources: Agave, Box, or a third-party API. Regardless of where the change originated, custom automation must be triggered in response to any change.

Agave's notification system was not verbose or scalable enough to keep up with the activity of a community the size of DesignSafe. Additional raw capacity was needed to handle thousands of new events each hour. Further, the concept of a batch job did not fit well with the short-lived, functional model needed to run small bits of code in response to each event.

B. Use case: ETL automation

Another challenge of the DesignSafe project was the need to migrate all their historical data into the new cloud storage system. Automated metadata extraction, indexing, creation and translation of ACL, UUID assignment, directory transformation, deduplication, and basic sanitization were needed for every file and folder transferred. The migration process was a classic export, transform, load (ETL) problem common in data warehousing, but with 50 million items was sufficiently large to require a non-traditional solution.

C. Evolutionary impact: delegated auth tokens, abaco, expanded events, internal and websocket notifications

These use cases led to the realtime, reactive design features recently added into the platform. A new Actor-based Containers (abaco) technology was developed to allow for fast, reliable execution of user-supplied code in response to events within the platform [7]. New primary and cascading events were added to the Platform to provide coverage over every action taken in the platform. A new websocket interface was added to the Platform allowing users to create custom notification channels and receive realtime updates directly from the platform. Finally, the concept of delegated credentials was added to the platform and used to inject short-lived, contextual auth tokens into each container and job context.

IV. High Throughput Computing

High throughput computing (HTC) is the sustained use of many compute resources to over time to accomplish a computational task. The focus in HTC is reliability and robustness over potentially long periods of time. While the popular HTC system, HTCondor [8], has been a supported execution option in the Platform for several years, the following use cases illustrate scenarios where users needed to run thousands of small tasks, taking full advantage of Agave's provenance features, across heterogeneous environments where HTCondor was not supported.

A. Use case: Elastic scaleover to commercial cloud

The DNA subway project [**?**] needed a way to elastically scale their computational needs in response to unpredictable job throughput running on HPC systems at Cold Spring Harbor Lab, TACC, and the University of Arizona when conducting classroom trainings. Agave's concept of an App was bound to a single execution system, so no native load balancing was possible. In response, the DNA Subway team moved their runtime application configuration into the Metadata API and used Agave to provision an elastic HPC cluster in AWS. This approach enabled them to migrate the entire execution environment by updating a single metadata item. As a result, when a class comes online, they are able to move all traffic from classroom users to the AWS cluster, which provides excellent throughput.

B. Use case: Task farming with Launcher

The Center for Reproducible Neuroscience needed an efficient way to run dozens of memory and compute intensive analysis pipelines across hundreds of OpenFMRI samples [9]. Agave enabled them to scale horizontally by running their pipelines in Docker containers, but Agave's public Docker compute system did not have enough capacity at their launch date to meet their production demands. The solution was to build Singularity images (which run on systems like Stampede) from the Docker images, and leverage Launcher to run ensemble HPC jobs [10]. Launcher coordinates the individual execution of each pipeline as a Singularity container.

C. Evolutionary impact: atomic metadata operations, custom apps, dynamic systems

In order to support these use cases, we updated the Metadata API to support atomic, partial updates. This opened up the use of the Metadata API for highly concurrent access and allowed it to be used to track progress, monitor subtask success, keep running counters, and manage runtime configurations. We also expanded the Notifications API to automatically preauthenticate webhook notifications sent to the Science APIs. This allowed both projects to publish progress events that could update metadata about the job and create a touch point for further automation.

D. Summary and Future Work

The use cases driving the introduction of interactive, reactive, and high throughput computing paradigms into the Agave Platform have evolved the platform and allowed it to support increasingly complex and impactful usage scenarios. Going forward, we are introducing new APIs to formally support reactive computing use cases, a managed EndOfDay service to formalize workflow and pipelining of tasks in HTC scenarios [11], and expanded websocket support to allow tighter communication with realtime applications.

Acknowledgment

This material is based upon work supported by the National Science Foundation Plant Cyberinfrastructure Program (DBI-0735191), the National Science Foundation Plant Genome Research Program (IOS-1237931 and IOS-1237931), the National Science Foundation Division of Biological Infrastructure (DBI-1262414), the National Science Foundation Division of Advanced CyberInfrastructure (1127210), and the National Institute of Allergy and Infectious Diseases (1R01A1097403).

REFERENCES

[1] R. Dooley, M. Vaughn, S. Terry, D. Stanzione, E. Skidmore, and N. Merchant, "Your Data, Your Way," Salt Lake City, UT, 2012.

[2] G. P. Johnson, S. A. Mock, B. M. Westing, and G. S. Johnson, "EnVision: A Web-Based Tool for Scientific Visualization." IEEE, 2009, pp. 603–608. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5071929

[3] Eric Fritz, "iAnimal variant discovery pipeline." [Online]. Available: https://github.com/ianimal/pipelines

[4] (2016) Agave Command Line Interface. Last access: 2016-09-02. [Online]. Available: https://bitbucket.org/agaveapi/cli

[5] (2016) Box.com. Last access: 2016-09-02. [Online]. Available: https://box.com/

[6] (2016) designsafe-ci. Last access: 2016-09-02. [Online]. Available: http://designsafe-ci.org/

[7] (2014) Abaco. Last access: 2015-11-11. [Online]. Available: https://github.com/TACC/abaco

[8] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[9] R. A. Poldrack and K. J. Gorgolewski, "OpenfMRI: Open sharing of task fMRI data," *NeuroImage*, Jun. 2015. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1053811915004632

[10] L. A. Wilson and J. M. Fonner, "Launcher: A Shell-based Framework for Rapid Development of Parallel Parametric Studies," in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, ser. XSEDE '14. New York, NY, USA: ACM, 2014, pp. 40:1–40:8. [Online]. Available: http://doi.acm.org/10.1145/2616498.2616534

[11] (2014) endofday. Last access: 2015-11-11. [Online]. Available: https://github.com/jstubbs/endofday