Enabling Science with Functions-as-a-Service: New Features and Usage of the Abaco Platform

Joe Stubbs, Christian R. Garcia, Julia Looney
Anagha Jamthe, Mike Packard and Matthew Vaughn.
Texas Advanced Computing Center
University of Texas at Austin
Austin, TX 78758
Email: [jstubbs, cgarcia,jlooney, ajamthe, mpackard, vaughn]@tacc.utexas.edu

***Abstract*—The Abaco Platform is an NSF funded project enabling researchers and developers to run containerized functions on infrastructure at the Texas Advanced Computing Center by making simple API calls over HTTP. Sometimes referred to as a "serverless" platform, Abaco reduces the administrative cost of research projects by eliminating the need to maintain infrastructure to run these workloads. Since its initial production release in January 2018, several major new features driven by feedback from the community have been released, including: an autoscaler capability for automatically managing the pool of workers for an actor; actor aliases, events and additional features designed to support building complex actor networks; synchronous executions for very low latency actors; and a global search capability. During this time, a number of new projects across a wide range of domains of science and engineering have adopted Abaco in interesting ways. In this paper, we describe the major new capabilities of Abaco since its initial production release and we discuss some of the innovative ways projects have been leveraging them to enable research.**

***Keywords*—Docker, containers, functions-as-a-service, actors, REST API, autoscaling, cloud computing.**

I. Introduction

Abaco (Actor Based Containers) is an open-source, distributed computing platform and web-based Application Programming Interface (API) hosted at the Texas Advanced Computing Center at The University of Texas, where clients execute atomic, independent workloads, or *functions*, on cloud infrastructure. In Abaco, such functions are referred to as *actors*, and clients define actors by making an API request to Abaco that includes a reference to a publicly available Docker image. Once an actor is defined, a client can send the actor a message by making an API request to the URI assigned to the actor. Abaco queues such messages on an internal message queue assigned to the actor, and, for each message, Abaco launches a container from the actor's Docker image. The system injects the original message into the container — as an environment variable in the case of a text message, and over a Unix Domain Socket in the case of a binary message — before starting the container's primary executable. In addition to performing basic computations, the actor executable can take advantage of a number of special aspects of the Abaco runtime environment including a full authentication context with which it can persist state, create new actors, and send messages to existing actors. Actors can also read and write to high-performance attached storage and exploit other kinds of specialized hardware, including nodes with large memory, many-core, and even GPUs. While supervising the container execution, Abaco collects any results registered by the actor as well as the associated container logs, resource utilization, and other data, and exposes this information to the end user through various endpoints. As such, Abaco provides a unique functions-as-a-service platform combining Linux container technology, HTTP web API architecture, and the Actor Model of Concurrent Computation, a theoretical model of computation pioneered by Carl Hewitt in the 1970s [1].

For science gateways specifically, Abaco supports virtually any and all asynchronous (i.e., long-running) task execution that might be required; for example: account initialization tasks when new users sign up, indexing file collections on large servers when new data arrives, analyzing web server log data for usage patterns, compiling the latest version on an HPC code, etc. Some projects have even built data pre-processing and analysis pipelines by networking multiple actors together. By running workloads on Abaco, a gateway project frees itself of administering the servers and other infrastructure where the tasks themselves run.

The National Science Foundation funded Abaco as a three year project beginning in September of 2017, and the initial production software release appeared in January of 2018. Since that time, usage of Abaco has grown substantially and a number of major new features have been released in response to community feedback and demand. First, Abaco added an autoscaler feature to simplify user management and administration of an actor's worker execution pool while simultaneously improving resource utilization dramatically. The autoscaler considers the system's available compute resources and the total number of pending messages for each actor to allow it to assign resources where they are needed most. While most actors run asynchronously, Abaco introduced synchronous executions for actors with very fast response times. To reduce latency, synchronous actors can be tagged with the "sync" hint which instructs Abaco to alter its standard autoscaler algorithm for improved performance.

Next, a set of features were developed to support actor networks: sets of actors that coordinate to perform a larger task. Abaco's event subsystem allows actors to automatically

execute in response to certain events on other actors, and its aliases feature provides a mechanism for insulating an actor from changes to other actors it sends messages to. Abaco nonces ease integration with third-party systems with different authentication systems, and are additionally supported at the alias level. Finally, a powerful, global search feature was recently added to Abaco allowing for full-text exact and fuzzy search across all primary collections and object attributes including actors, workers, executions, and logs.

Usage and adoption by new projects has also increased across this time. A sophisticated Extract, Transform, Load (ETL) pipeline referred to as RoundTrip [2] has been developed as part of the Synergistic Discovery and Design project to automate experimental design and analysis in grand challenge problems in computational biology [3]. A significant subsystem of RoundTrip consists of a complex network of roughly 30 Abaco actors. GenApp [4], another NSF funded project providing a tool for developers to rapidly build web portals for computational science, now supports building and running containerized applications on the Abaco cloud [5]. A number of prominent science gateway projects are utilizing Abaco for a variety of asynchronous tasks such as new user account initialization, resource monitoring, email and alert delivery, etc. A NASA JPL project is starting to use Abaco as part of a data pipeline geared at allowing scientists to measure the mass of exoplanets in an effort to determine a planet's habitability [6]. Finally, Abaco was recently leveraged by the UT Austin Covid-19 Modelling consortitum for asynchronous task execution [7].

We provide additional details regarding the new features and usage introduced above, as well as a look at our future development plans for Abaco in the subsequent sections.

II. New Features

In this section we provide details on the major new features of Abaco recently developed and released.

A. Autoscaler

Abaco makes use of an internal agent referred to as a *worker* to facilitate actor executions. When a worker is started, it is assigned exactly one actor to manage, and the worker listens to the actor's message queue for new messages for the actor. When Abaco puts a new message on the actor's message queue, the worker receives it, performs some light pre-processing, and then launches a new container from the Docker image defined for the actor. The worker monitors the execution to completion before returning to the actor's message queue for another message.

As a result, the number of messages an actor can process concurrently is equal to the number of workers assigned to the actor. Abaco provides a `/workers` endpoint in its API that allows users to manually start and stop workers for actors they own, but manually scaling the worker pool is challenging technically and the endpoint comes with a number of restrictions to prevent abuse and provide stability across the cluster.

To address this issue, Abaco developed an autoscaler capability to automatically scale the pool of workers assigned to a given actor. The autoscaler is a separate Abaco component that tracks the sizes of the actor message queues as well as the number of workers assigned compared to the total number of workers supported by a given installation as shown in Figure 1. It starts new workers for actors that have additional messages pending and it stops workers for actors that do not. The autoscaler thereby eliminates the burden of worker management on end users. Additionally, by shutting down idle workers, the autoscaler reclaims resources from the cluster, making the overall system more efficient. Based on anecdotal feedback from users, the autoscaler has been one of the most helpful and popular new features.

The scalability of the autoscaler and the Abaco system in general was examined extensively in a previous study where it was established that Abaco scaled correctly to 100 JetStream "m1.medium" instances running a variety of different workloads, including matrix multiplication for various matrix sizes and SHA256 hashing ([8]). The autoscaler compared very well to "manual scaling", and Abaco in general compared well to running the code directly on JetStream. A sample of these data are shown in Figure 2, where Abaco achieves over 19 TFLOPs running a matrix multiplication code on 100 nodes.

B. Synchronous Executions and Actor Hints

Synchronous executions are related to another feature of Abaco referred to as actor hints. When registering an actor, a user can supply a list of strings representing "tags" or metadata about the actor. Additionally, Abaco recognizes "official" hints that control configurable aspects of the actor runtime. For example, the "sync" hint tells Abaco that the user expects the actor to respond to synchronous messages. For such actors, Abaco adjusts its autoscaler algorithm to not reduce the worker pool all the way to 0 as quickly. Keeping a single "warm" worker ready for actors responding to synchronous messages prevents the actor from experiencing latency spikes associated with the performance penalty of starting the initial worker.

When a user sends a message to an actor, Abaco puts the message in the actor's queue and then, in the typical case, responds to the user immediately with an identifier for the execution associated with the message. The user can then make subsequent requests to the API to check the status of the execution and retrieve additional information such as the execution results, logs, resource utilization, etc. However, Abaco has added support for synchronous executions: in this case, when a user sends an actor a message, Abaco will block until the execution completes, and then respond with the execution result. This mode works very well for actors implementing lightweight microservices or other low-latency functions.

C. Actor Networks

The Actor Model implemented in Abaco allows users to build up complex networks of actors coordinating on a larger task. While powerful, engineering such an actor network

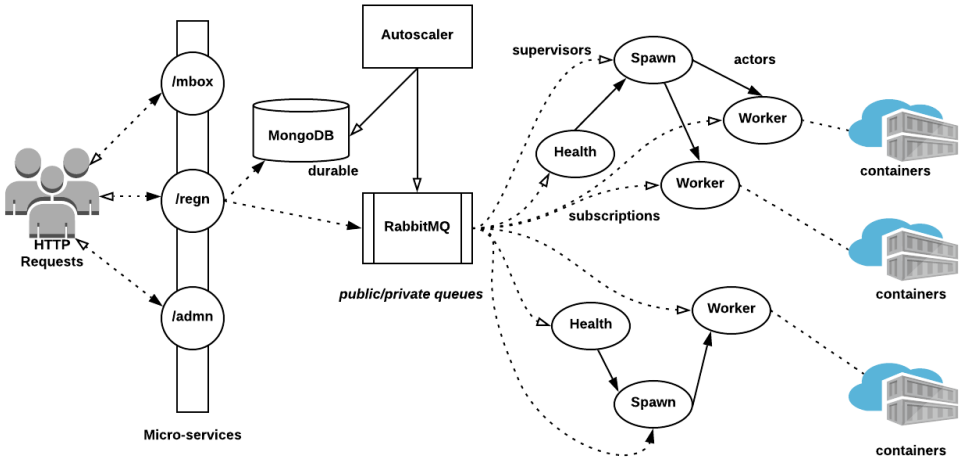

Fig. 1. Abaco architecture

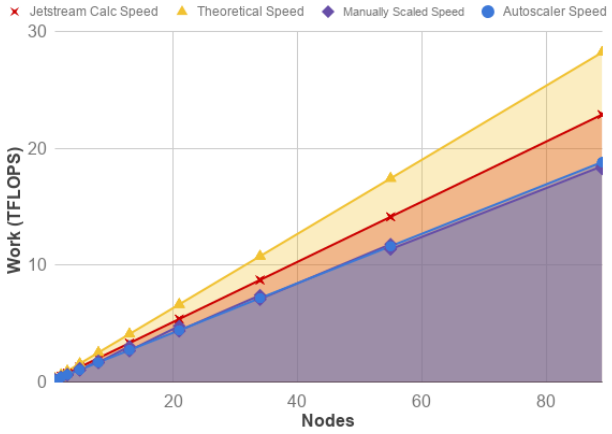

Fig. 2. Abaco performance: a comparison of speeds achieved using Abaco manual scaling and autoscaling to the speed achieved by running the code directly on JetStream as well as the theoretical speed of the hardware.

can be challenging to develop and maintain. Abaco provides three primary sets of features to aid in these efforts. First, Abaco includes a rich event system which tracks all primary state changes in the platform, including changes in status of an actor, such as to the `READY` state, the `ERROR` state, etc., and changes in status of an execution (`SUBMITTED`, `RUNNING`, `COMPLETE`). Users can register URLs to receive webhooks — HTTP POST requests with event data included in the message payload — for events as they occur. Alternatively, users can create actors to manage the events of other actors using the actor *link* property. When a user links actor A to actor B, Abaco will automatically send a message to actor B whenever an event on actor A occurs. This creates a loose coupling between the actors and allows for interesting patterns such as the supervisor pattern, popular in systems such as Erlang [9].

Actor aliases provide another helpful feature for networks. Put simply, an alias is a user-provided identifier that maps to a specific actor; however, the key point is that users control the mapping and can update it at any time. To see how this is useful, suppose actor A needs to send messages to actor B as part of an actor network. Instead of "hard coding" the identifier of actor B, actor A instead sends messages to an alias that maps to B. If in the future, changes are needed and messages from actor A need to be routed to a new actor, C, this can be accomplished automatically by updating the alias. No code changes in actor A are required.

Actor *config* objects are another useful feature when developing complex actor networks. Config objects in Abaco are JSON objects that are managed independently of any given actor but that are shared with one or more actors. Config objects contain valuable configuration metadata for an actor, such as the URL, port and credentials for a database service. When a worker starts an execution for an actor that has been shared a config object, the worker injects the entire JSON object into an environment variable for the actor code to use. Thus, instead of hard-coding such configuration into the definitions of all the actors that need it, Abaco config objects centralize the data in one place. Additionally, if the definition of a config object is updated, all actor executions immediately get the updated data — there is no need to redeploy the actor. In this way, users can update the configuration of all actors in a network simultaneously.

D. Global Search

As part of the most recent 1.6 release, Abaco provides a powerful global search capability has been built on top of the MongoDB aggregation system allowing users to search based on any attribute associated with resources that they have permission to view. All objects in the primary Abaco collections are retrievable by search, including actors, executions, workers and logs. The Abaco search uses a formal grammar comprised of attributes, operations and values, and includes

TABLE I
TOTAL ABACO USAGE SINCE JAN, 2018

Metric	Total
Total Number of Actors	$43,784$
Total Executions	$729,327$
Total Runtime (seconds)	$20,766,431$
Total CPU (jiffies)	$6.21x10^{18}$
Total Network IO (bytes)	$5.85x10^{14}$

the ability to perform full-text exact-match and fuzzy-match searches. Retrieving all executions with status `SUBMITTED` or finding all logs containing a specific error message across all executions for all actors the user has access to are just two simple examples of searches that can now be performed with a single API call.

III. USE CASES AND USAGE

Since its initial production release in January of 2018, the Abaco platform has seen significant usage and adoption. Almost 44,000 actors have been registered and Abaco has performed nearly 730,000 actor executions that have collectively run for over 20 million seconds (346,000 hours) consuming nearly 100 TB of network I/O. Precise usage statistics are given in table I.

In the remainder of this section, we highlight some of the more interesting, advanced, and exemplar projects making use of the Abaco platform.

A. RoundTrip in the Synergistic Discovery and Design Project

The Synergistic Discovery and Design (SD2) project tackles grand challenge problems in synthetic design across domains such as computational biology and chemistry in which quality, formal mathematical models are unavailable [3]. In lieu of models, experimental data and machine learning techniques are combined with automated design. An elaborate Extract Transform Load (ETL) pipeline, referred to as RoundTrip, has been developed for SD2 to automate the experimental design, execution and analysis of efforts to build digital circuit components such as AND and OR gates in microorganisms such as yeast and bacteria. RoundTrip must interface with semi-structured experimental request objects, cloud-laboratory experiments, and machine learning analyses. A significant subsystem of RoundTrip is implemented as a network of nearly 30 Abaco actors, making it perhaps the most sophisticated such actor network. RoundTrip is capable of handling a number of different Experimental Request (ER) types. In April of 2020, RoundTrip succeeded in reducing the overall time to process ERs from three weeks to approximately four hours, reducing laboratory idle time from several weeks to a few days.

B. Containerized Codes in GenApp

The NSF funded project GenApp has adopted Abaco as a compute backend for containerized applications [4]. GenApp allows developers to rapidly build and deploy full-featured, end user applications with graphical user interfaces for executing research codes on a variety of HPC and high-throughput resources. Genapp can automate the process of containerizing an existing code and executing it on the Abaco platform.

C. Asynchronous Task Execution in Science Gateways

A number of science gateways, including an analysis portal launched very recently by the UT Austin Covid-19 Modelling Consortium, use Abaco for asynchronous task execution [7]. For example, to finalize account creation when a new user initially logs in, a number of tasks get queued with Abaco actors such as creating a UID and GID for the user and generating and storing SSH key pairs for use on HPC storage and execution resources, adding the user to project allocations, etc. Actors are used to monitor the health of complex infrastructure such as data transfer nodes with large network mounts to high-performance storage. Another set of actors manage external messaging and alerts by delivering emails, Slack notifications, etc. In one case, the third-party Mailgun service is used for email delivery. The "mailgun actor" owns credentials for sending email through the Mailgun service, and no other part of the system requires the credentials. This is a common and successful pattern where an actor becomes a very light-weight microservice running on scalable infrastructure that the web development team does not need to manage.

D. NASA JPL NEID Automated Data Pipeline

In February of 2020, TACC started working with NASA JPL to build a data analysis pipeline for the NEID project. The NEID, a specialized spectrometer installed at the top of Arizona's Kitt Peak, collects data which can be used to compute the mass and density of exoplanets, planets in other solar systems, a first step towards determining habitability. Large sets of raw data are transmitted from the telescope in Arizona to the California Institute of Technology and then to the Texas Advanced Computing Center via a Globus transfer multiple times per day [10]. A series of sophisticated algorithms developed at Penn State and CalTech reduce the raw to a set of level 0, level 1 and level 2 data products. These products are then transferred back to CalTech. The team is developing a mix of Abaco actors and codes running on traditional HPC systems at TACC to automate the end-to-end data pipeline, removing any human from the loop.

IV. FUTURE WORK

A number of new features are planned for Abaco over the coming months and years. First, the Abaco platform will become a first-class API in the new Tapis v3 system being developed. Tapis is a hosted web framework for reproducible, distributed computational research and will provide support for data management and analysis of streaming data. Among other capabilities, Abaco functions will integrate directly with the Tapis Streams API to enable users to execute actors in response to streaming data events.

The Abaco project will add support for executing actor containers on a Kubernetes cluster for additional scalability, and, by request, will architect support for remote deployments to allow institutions to provide compute power to TACC's

Abaco instance for running their own actors locally. (Institutions are already free to deploy the entire open-source Abaco platform locally, as has recently been done at the Centers for Disease Control). The development team is working on a prototype to add support for exposing actors over gRPC which would open a number of possibilities including bi-directional streaming between client and actor or between two actors. Another research and development effort is attempting to add support for conflict-free replicated data types (CRDTs) to the actor state API, to enable actors that process multiple messages in parallel and persist state safely without having to deal with coordination.

Several additional utility features are planned such as time-scheduled actor executions analogous to cron, integration with an on-premise container image registry to allow users to register actors with images containing security sensitive and/or proprietary software, and many more.

V. Related Work

Abaco draws comparison to a number of commercial and open-source Functions-as-a-Service (FaaS) platforms. In the commercial space, the most popular offerings include AWS Lambda [11], Azure Functions [12] and Google Cloud Functions [13], but other commercial cloud vendors also provide FaaS platforms (e.g., IBM Cloud Functions based on Apache OpenWhisk [14]). Open-source projects include Apache OpenWhisk [15], Fn [16], IronFunctions [17], Knative [18], Kubeless [19], OpenFaas [20], etc. To our knowledge, Abaco is the only FaaS platform built on top of the Actor Model, allowing actors to save state across executions, create actors, etc., and Abaco is the only hosted platform with direct integration into the nation cyberinfrastructure ecosystem of advanced storage and computing resources. Additionally, Abaco does not impose resource or runtime limits for actor container executions and is freely available for use by the research community.

Acknowledgment

This material is based upon work supported by the National Science Foundation Office of Advanced CyberInfrastructure, award number 1740288.

References

[1] G. Agha, *Actors: A Model of Concurrent Computation in Distributed Systems*. Cambridge, MA, USA: MIT Press, 1986.
[2] D. Bryce *et al.*, "Round-trip: An automated pipeline for experimental design, execution, and analysis," in *International Workshop on Bio-Design Automation*. submitted, 2020.
[3] (2020) Sd2e. [Online]. Available: https://www.tacc.utexas.edu/research-development/tacc-projects/sd2e
[4] (2020) Genapp. [Online]. Available: https://genapp.rocks
[5] E. Brookes and J. Stubbs, "Genapp, containers and Abaco," in *Proceedings of the Practice and Experience on Advanced Research Computing*, 2019, pp. 1–8.
[6] (2020) NASA's NEID: A new tool for 'weighing' unseen planets. [Online]. Available: https://www.jpl.nasa.gov/news/news.php?feature=7571
[7] (2020) Ut-covid-19. [Online]. Available: https://covid-19.tacc.utexas.edu
[8] C. Garcia *et al.*, "The Abaco platform: A performance and scalability study on the Jetstream cloud," in *The 16th International Conference on Grid, Cloud, and Cluster Computing (GCC'20)*. Springer Nature, 2020.
[9] (2020) Erlang. [Online]. Available: https://erlang.org/doc/man/supervisor.html
[10] W. Allcock *et al.*, "Secure, efficient data transport and replica management for high-performance data- intensive computing." Proceedings of the IEEE Mass Storage Conference, pp. 13-28 April 2001.
[11] (2020) Aws lambda. [Online]. Available: https://aws.amazon.com/lambda/
[12] (2020) Microsoft azure functions. [Online]. Available: https://azure.microsoft.com/en-us/services/functions
[13] (2020) Google cloud functions. [Online]. Available: https://cloud.google.com/functions
[14] (2020) Ibm functions. [Online]. Available: https://www.ibm.com/cloud/functions
[15] (2020) Apache openwhisk. [Online]. Available: https://openwhisk.apache.org
[16] (2020) Fn. [Online]. Available: https://fnproject.io/
[17] (2020) Iron io. [Online]. Available: https://github.com/iron-io/functions
[18] (2020) Knative. [Online]. Available: https://knative.dev/
[19] (2020) Kubeless. [Online]. Available: https://kubeless.io/
[20] (2020) Openfaas. [Online]. Available: https://www.openfaas.com