Federated Function as a Service for eScience

Yadu Babuji*, Josh Bryan*, Ryan Chard*, Kyle Chard*,
Ian Foster*, Ben Galewsky+, Daniel S. Katz+, Zhuozhao Li*
*University of Chicago, +University of Illinois at Urbana-Champaign

***Abstract*—The function as a service paradigm aims to abstract the complexities of managing computing infrastructure for users. While adoption in industry has been swift, we have yet to see widespread adoption in academia. This is in part due to barriers such as the need to access large research data, diverse hardware requirements, monolithic code bases, and existing systems available to researchers. We describe funcX, a federated function-as-a-service platform that addresses important requirements for use of FaaS in research computing. We outline how funcX has been used in early science deployments.**

***Index Terms*—Serverless computing, FaaS, funcX**

I. Introduction

Function as a service (FaaS) allows for applications to be decomposed into functions that are executed remotely and elastically without considering the resources on which functions are deployed. Cloud FaaS platforms abstract the underlying computing infrastructure almost entirely and offer duration-based pricing models in which customers are charged for every millisecond of execution. While FaaS has been widely adopted in industry it is yet to be broadly adopted in research computing. There are several key reasons: the mismatch with current research computing workloads, cost of transferring and storing large research data in the cloud, lack of support for specialized computing hardware, and the availability of research computing systems that are pre-configured with common libraries.

To address the specialized needs of research computing we are developing funcX [1], a federated FaaS platform. funcX transforms the FaaS paradigm by allowing function invocations to be routed to endpoints hosted on arbitrary remote computer systems, from the edge to supercomputing facilities. In this poster we describe funcX and highlight how it is being used in various research scenarios.

II. funcX

funcX is implemented as two components: (1) a cloud-hosted service and (2) user-deployable agent software.

The cloud service provides a single available point of contact for users to register and manage endpoints, register and share functions, and invoke functions on accessible endpoints. Interactions with the service are via a Python SDK.

The funcX agent can be installed on *any* computer, transforming it into a function-serving endpoint. The agent is a Python application that is responsible for managing computing resources and executing functions on behalf of users. The agent uses Parsl [2] to interact with diverse computing systems, from clouds to high performance clusters with various batch scheduler interfaces (e.g., SLURM, PBS). The agent dynamically provisions resources and executes functions in defined execution environments (optionally using containers).

To users, funcX looks like any other FaaS platform. Users register Python functions with the funcX service and optionally specify dependencies or a container, set access permissions to share functions with other users, and provide descriptive metadata. After registration, authorized users may invoke the function via the funcX service by specifying the necessary input arguments. Unlike traditional FaaS systems, funcX also allows users to specify the endpoint on which they wish to execute the function.

Users are thus able to route function invocations based on requirements. For example, they may execute functions where data may be located, where specialized hardware or software may be deployed, where resources are available at a particular time, or where costs may be lowest.

III. Deployments

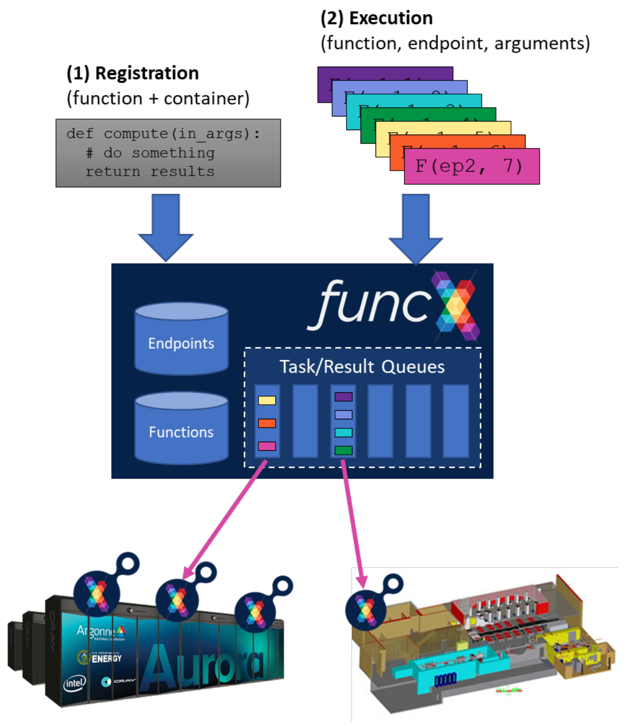
funcX is currently being used in three main ways: to execute arbitrary analysis and simulation workloads, to provide interfaces to research cyberinfrastructure, and as a platform for building other services and applications. We briefly describe representative case studies for each type.

A. funcX for analysis

High-energy physicists face the challenge of applying statistical models to large amounts of data to derive physics information. In the past, statistical fitting was traditionally implemented in C++ and installation represented a significant hurdle for potential users. Feickert et al. recently implemented a funcX-based approach using the pyhf pure-python fitting/limit-setting/interval estimation package and exposing it as a funcX function [3]. This function has been used to process large datasets on various remote systems, including SDSC Expanse, BlueWaters, and an OSG cluster.

Biologists using Argonne National Laboratory's Advanced Photon Source (APS) face the challenge of rapidly computing results as experiments are running, for example to check quality, view reconstructions, and see preliminary results all while the sample is in the beamline. Rapid analysis allows them to steer the experiment, reorient samples, restart experiments, and move to new samples when necessary. The computing

This work is supported by NSF 2004894/2004932 and an Argonne Lab Directed Research and Development award.



```
def compute(in_args):
  # do something
  return results
```

Fig. 1. funcX architecture showing funcX agents deployed on the Aurora supercomputer and a beamline at SLAC. The funcX service (middle) manages a catalog of endpoints and functions, and is responsible for routing function invocations to accessible endpoints for execution. Users interact with funcX by (1) registering functions, and (2) invoking functions with necessary input arguments and target endpoint.

tasks here are varied: some can be run on modest compute resources, while others require HPC systems, for example to refine structures. APS researchers have deployed funcX on local compute resources and supercomputers at Argonne's Leadership Computing Facility, and they use it as part of their real-time experiment pipelines. funcX has been adopted at multiple beamlines to process hundreds of thousands of datasets, including as part of an effort to solve structures of the SARS-CoV-2 at the Structural Biology Center [4].

Molecular scientists are building machine learning models for predicting the 3D geometries of small organic molecules. Their approach relies on training data generated by running molecular dynamics (MD) simulations for tens of thousands of molecules. The researchers use funcX to port MD simulations across various computing resources, including a local GPU cluster as well as a Kubernetes cluster. funcX allows them to easily route simulations to resources when they are available, to scale analyses based on needs, and to leverage containers to abstract system differences.

COVID researchers have developed a simulation to explore contagion via agent-based modeling, as well as potential effectiveness of various measures. They developed a model that predicts spread based on various input parameters. Using the model requires executing an ensemble of scenarios and exploring the effects of various features. To do this, the researchers registered the model as a funcX function. They use funcX to run large ensembles on campus and national resources by invoking the function with various input parameters.

B. *funcX as an interface*

The National Center for Supercomputing Applications' Hardware-Accelerated Learning (HAL) cluster uses funcX to provide straightforward access to various groups of researchers. In one example, HAL's administrators configured a funcX endpoint and registered a set of astrophysics codes for detecting gravitational waves as a funcX function. They then shared access to the endpoint and function with a select group of users, enabling them to easily execute the code using HAL's FPGAs.

C. *funcX as a platform*

Xtract [5] is a metadata extraction system that can process scientific data repositories and derive metadata from files stored in those repositories. Xtract applies metadata extraction pipelines that are able to learn about individual files and apply suitable metadata extractors to those files. For example, a PDF file may be processed with a text, table, and image extractor; and the image, if identified as a map, may then be processed by a map extractor. Xtract uses funcX for all computation. Each extractor is registered as a funcX function with a container that wraps necessary dependencies. When processing a target repository, Xtract uses funcX to invoke the extractor function "near" the data by sending it to a co-located endpoint.

The **Data and Learning Hub for science for science (DLHub)** [6] is a multi-tenant system that provides both a model repository and model serving capabilities with a focus on science applications. DLHub uses funcX to provide scalable and low-latency serving engine that can leverage remote and heterogeneous computing resources to democratize access to published models through a simple web interface.

IV. Summary

funcX is a federated function as a service platform that is designed to address the unique requirements of research use cases. It offers a single cloud-hosted interface via which users may manage not only functions but also the endpoints on which those functions are executed. funcX is being used in various domains from biology to high energy physics, as an interface to heterogeneous computing systems, and as a platform for developing new applications and services.

References

[1] R. Chard *et al.*, "Funcx: A federated function serving fabric for science," in *29th Intl. Symp. on High-Perf. Par. & Dist. Comp. (HPDC)*, 2020, p. 65–76.

[2] Y. Babuji *et al.*, "Parsl: Pervasive parallel programming in Python," in *28th Intl. Symp. on High-Perf. Par. & Dist. Comp. (HPDC)*, 2019.

[3] M. Feickert *et al.*, "Distributed statistical inference with pyhf enabled through funcX," *arXiv*, vol. 2103.02182, 2021.

[4] M. Wilamowski *et al.*, "2'-o methylation of RNA cap in SARS-CoV-2 captured by serial crystallography," *PNAS*, vol. 118, no. 21, 2021.

[5] T. J. Skluzacek *et al.*, "A serverless framework for distributed bulk metadata extraction," in *30th Intl. Symp. on High-Perf. Par. & Dist. Comp. (HPDC)*, 2020, p. 7–18.

[6] R. Chard *et al.*, "DLHub: Model and data serving for science," in *IEEE Intl. Par. & Dist. Proc. Symp. (IPDPS)*, 2019.