Federated FaaS for Flexible Scientific Computing

Kyle Chard

Joint work with Ryan Chard, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard, Ben Blaiszik, Ben Galewsky, Josh Bryan, Daniel S. Katz, and Ian Foster

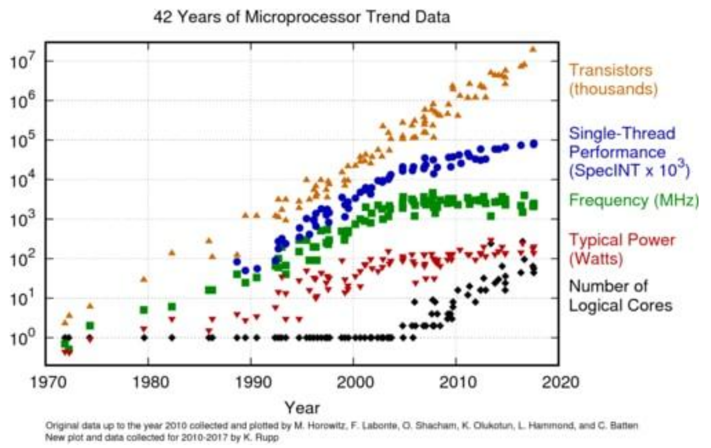
THE UNIVERSITY OF CHICAGO

Argonne NATIONAL LABORATORY

The scientific computing ecosystem is rapidly evolving

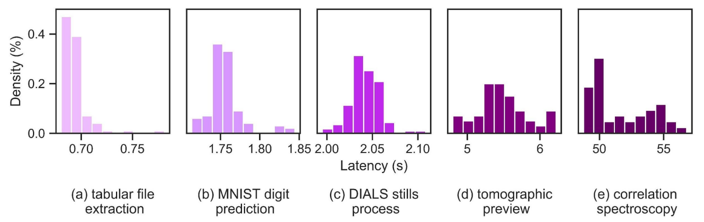
Resources

- Hardware specialization (e.g., architectures, accelerators)
- Specialization leads to distribution



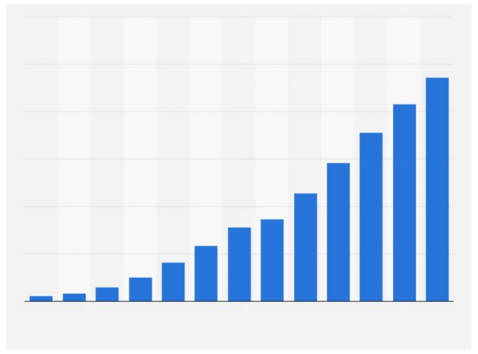
Workloads

- Interactive, real-time workloads
- Machine learning training and inference
- Components may best be executed in different places

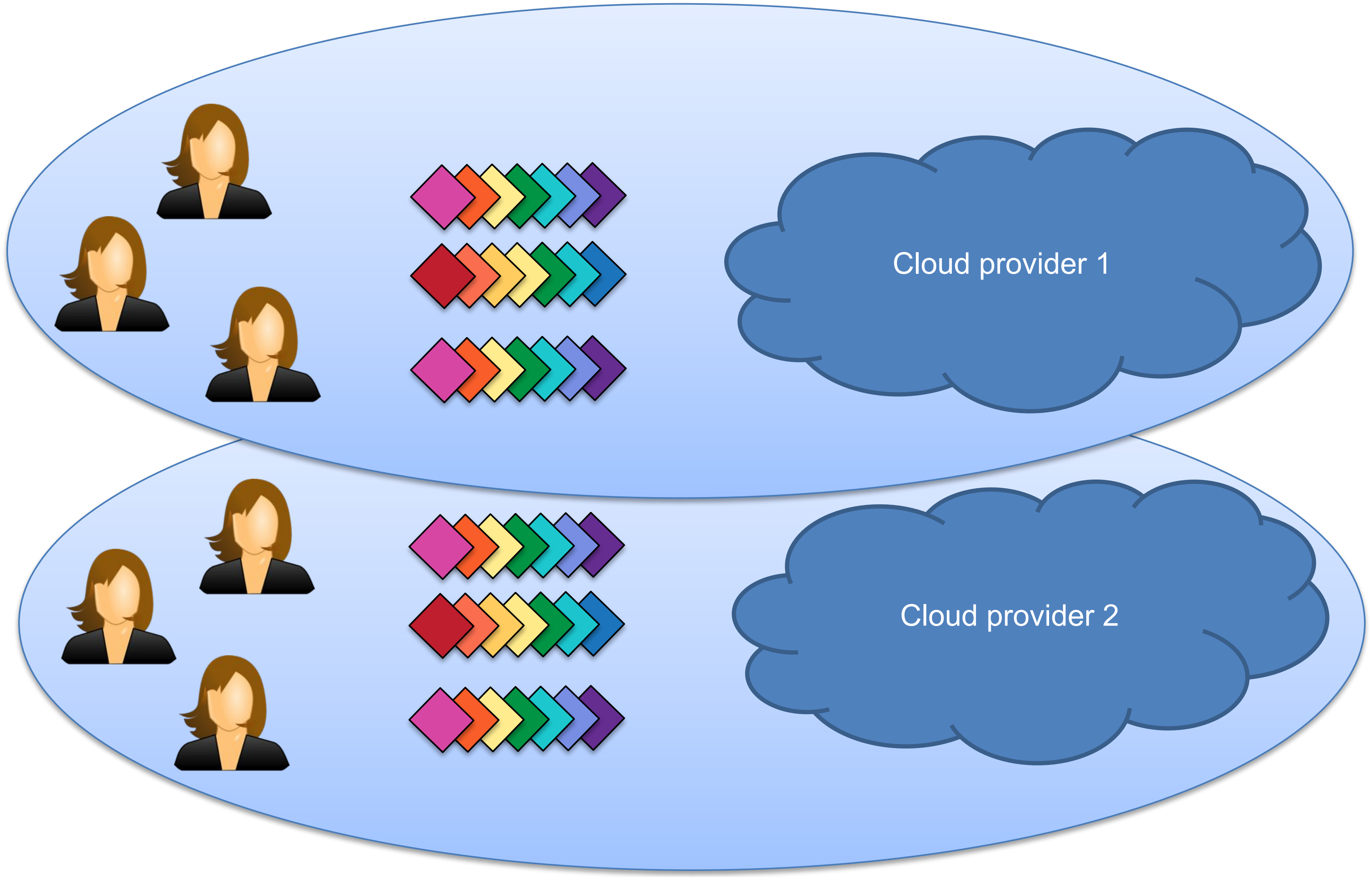


Users

- Diverse backgrounds and expertise
- Different user interfaces (e.g., notebooks)

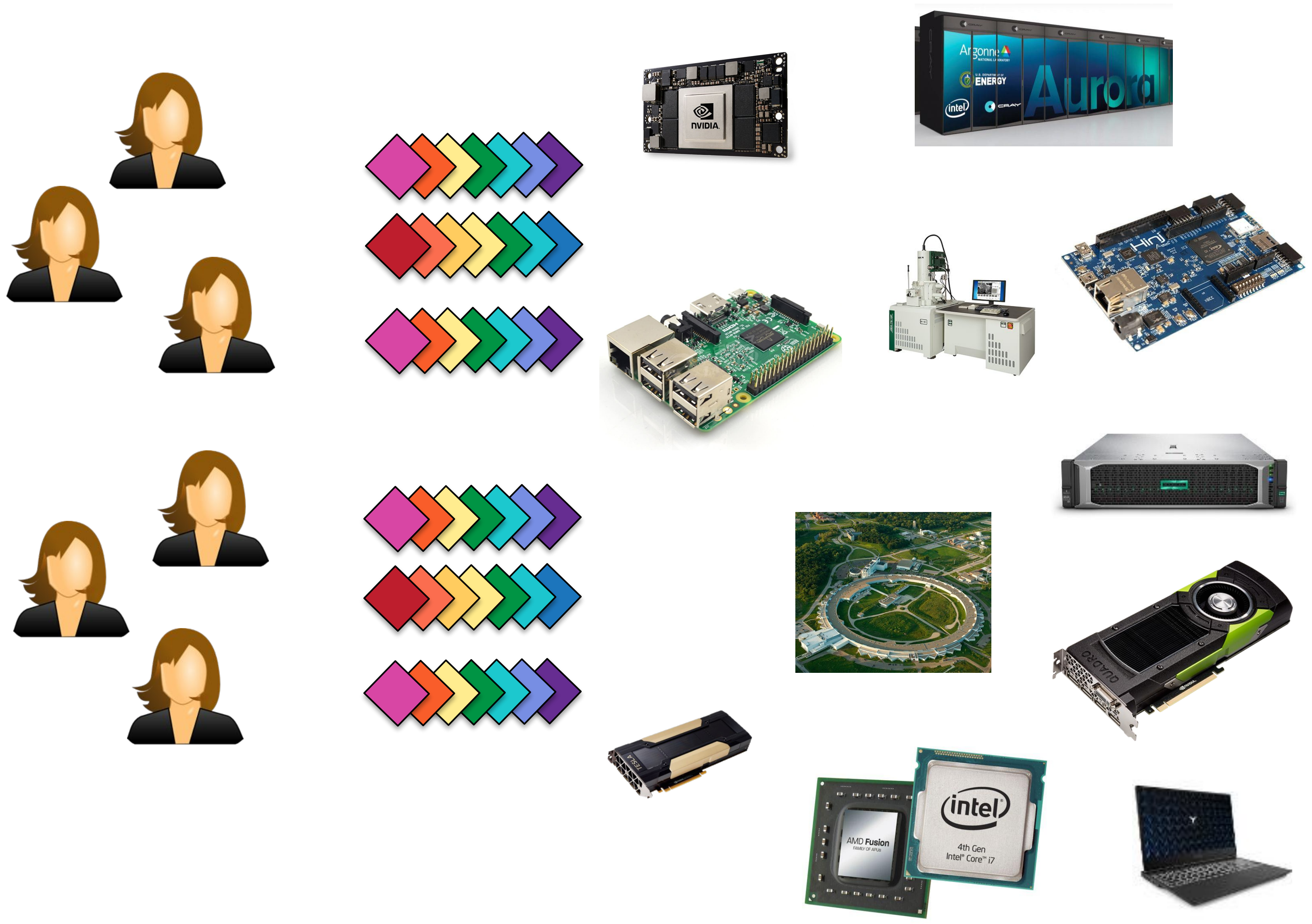


FaaS as offered by cloud providers



- Single provider, single location to submit and manage tasks
- Homogenous execution environment
- Transparent and elastic execution (of even very small tasks)
- Integrated with cloud provider data management

FaaS as an interface to the scientific computing ecosystem?



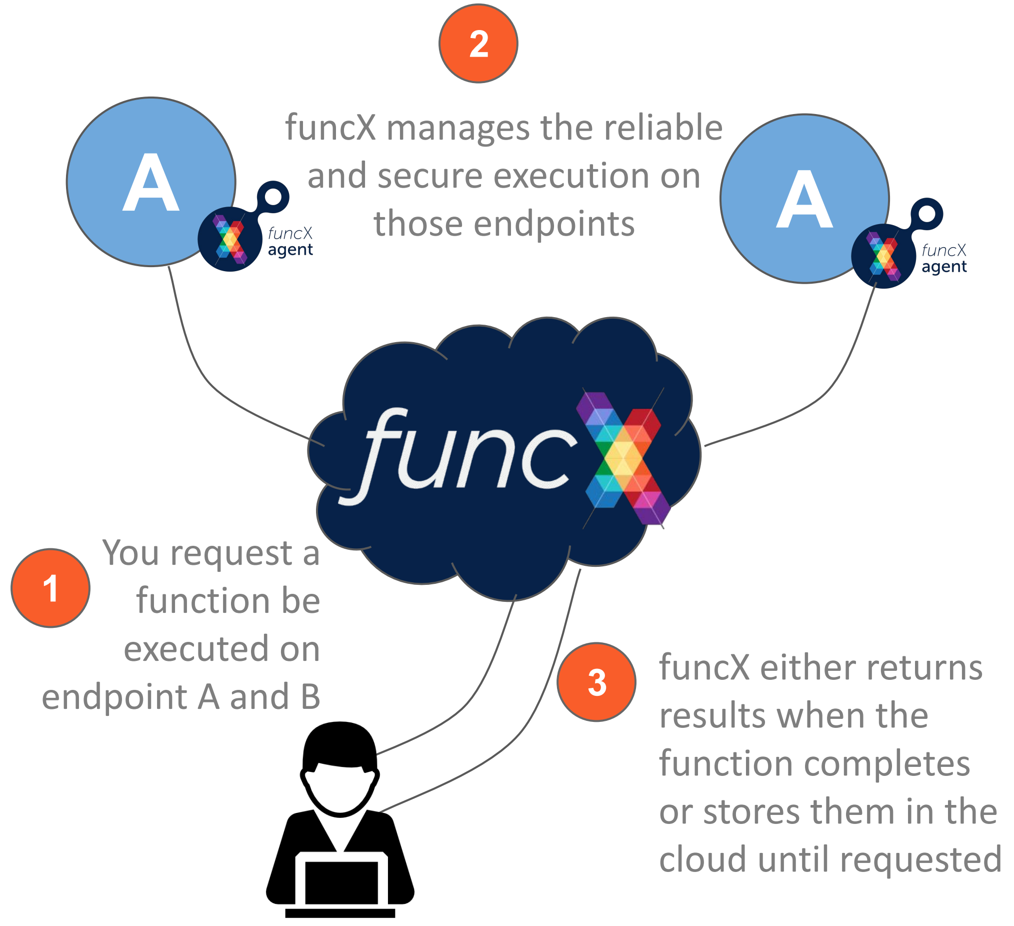
We still want

- Single interface
- Homogenous execution environment
- Transparent and elastic execution
- Integrated with data management

funcX: managed and federated FaaS

- Cloud-hosted service for managing compute
- Register and share *FaaS compute endpoints*
- Register and share Python functions
- Reliable, scalable, secure function execution

Try funcx on Binder


2
funcX manages the reliable and secure execution on those endpoints
A
funcX agent
A
funcX agent
funcX
1
You request a function be executed on endpoint A and B
3
funcX either returns results when the function completes or stores them in the cloud until requested

funcX https://funcx.org

FuncX: a federated function serving ecosystem for research

Endpoints:

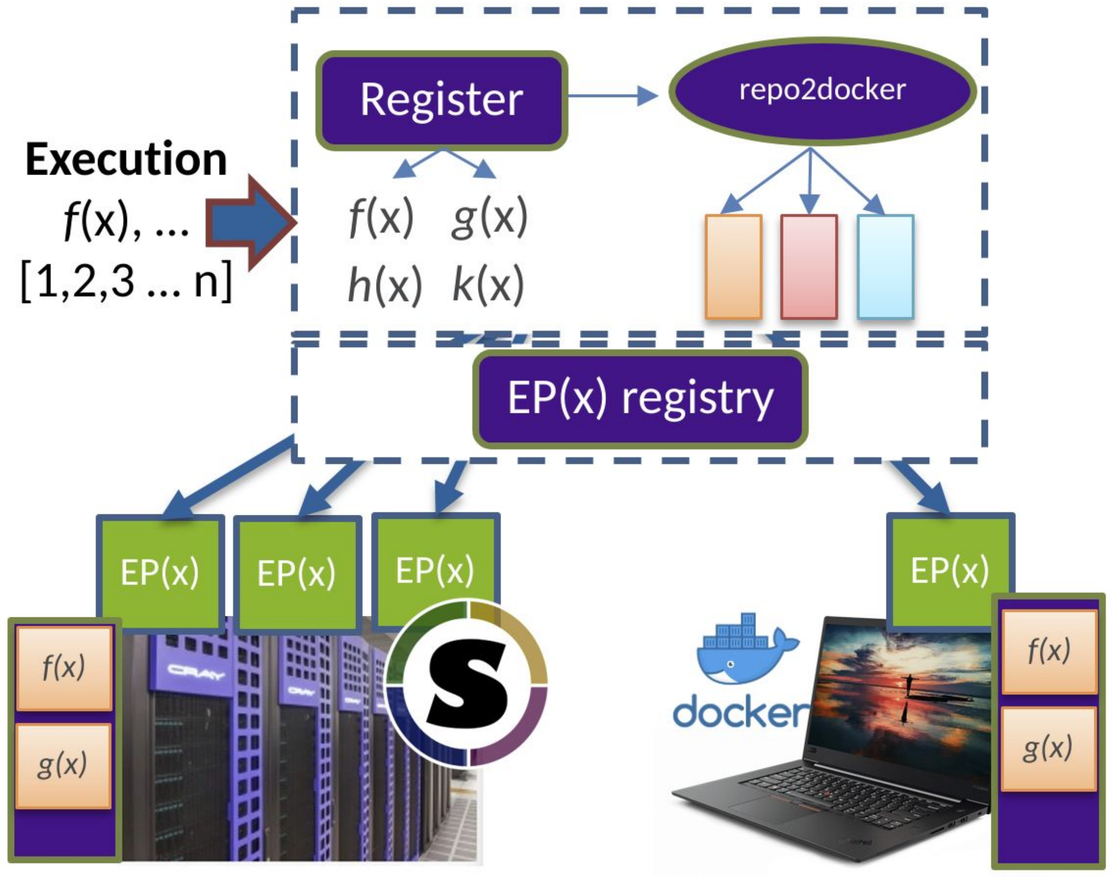
- User-deployed and managed
- Dynamically provision resources, deploy containers, and execute functions
- Exploit local architecture/accelerators

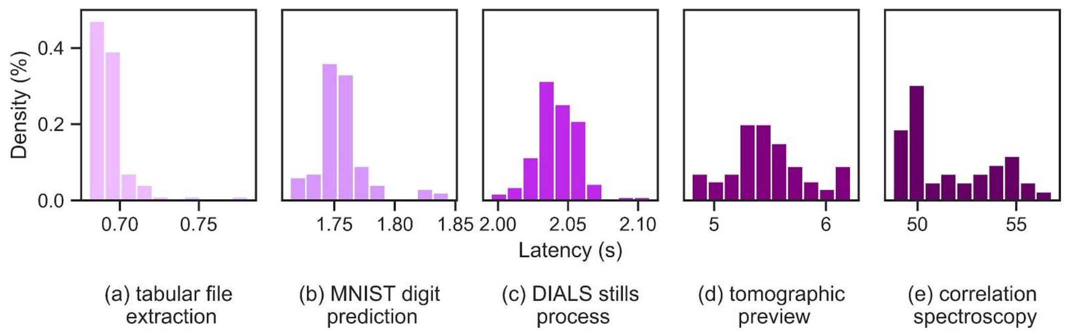
funcX Service:

- Single reliable cloud interface
- Register and share endpoints
- Register, share, run functions

Choose where to execute functions

- Closest, cheapest, fastest, accelerators …





Common use case 1: fire-and-forget execution

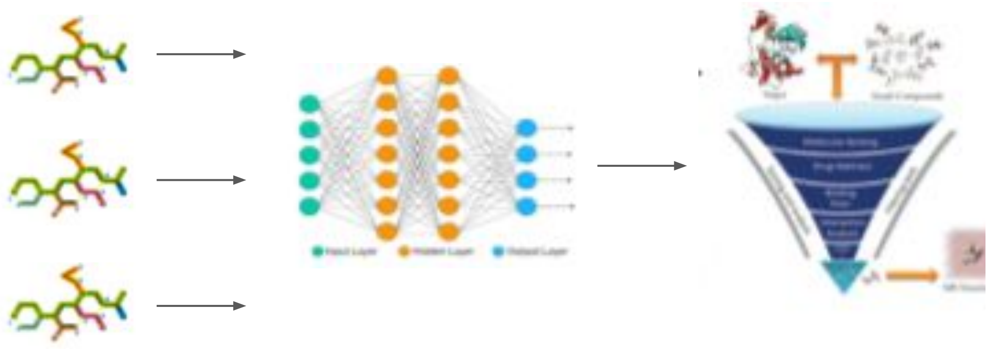
Executing a bag of tasks (e.g., running simulations with different parameters, executing ML inferences) on one or more remote computers directly from your environment (e.g., Jupyter on your laptop)

Advantages:

- Fire-and-forget execution managed by funcX (tasks/results cached until endpoint/client online)
- Portability across different systems (optionally making use of specialized hardware)
- Elastic scaling to provision resources as needed (from HPC and cloud systems)

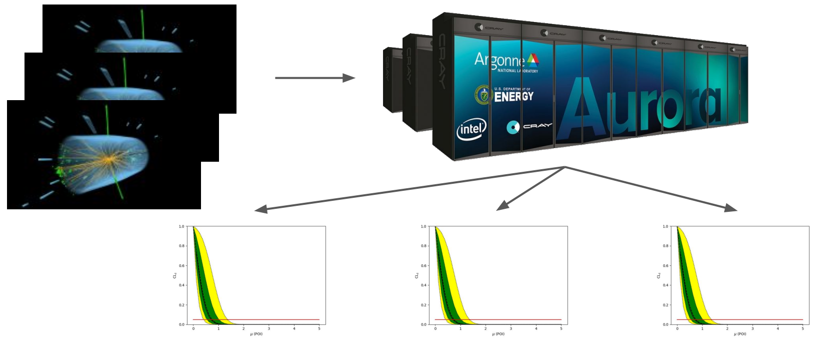
Examples:

ML-based drug screening



Screening billions of molecules to identify potential COVID-19 therapeutics. Computing molecule features, running ML inference, selecting top results.
(National Virtual Biotechnology Laboratory, arXiv:2006.02431)

Distributed statistical inference for HEP



Wrapping a C-based statistical inference tool as a function so scientists can easily fit multiple different hypotheses for new physics signatures (signals).
(Feickert et al., arXiv:2103.02182)

Common use case 2: automated analysis of data

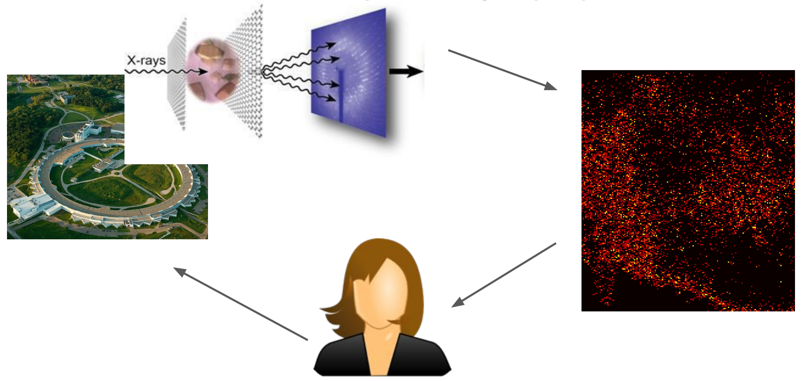
Constructing and running automated analysis pipelines that include data processing steps that need to be executed in different locations (e.g., near an instrument, in a data center, on specialized hardware)

Advantages:

- Automatically processes data as they are acquired (event- and workflow-based)
- Integrates with data movement and other actions (both human and machine)
- Execute functions across the computing continuum (close to data, on specialized hardware, etc.)

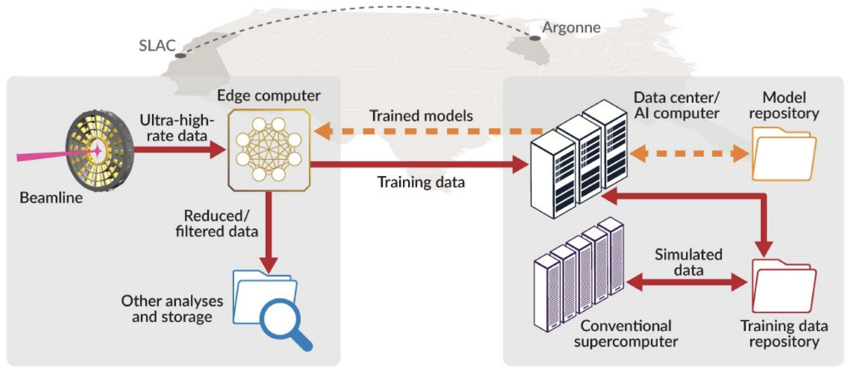
Examples:

Serial Crystallography



Near-real-time analysis of data acquired from the Advanced Photon Source to solve protein structures at room temperature.
(Joachimiak et al., https://doi.org/10.1073/pnas.2100170118)

Remote training of DNNs



Using DNNs to estimate probability density function by training DNN with real-time data (e.g., on Cerebras, DGX, SambaNova) and inference at the edge
(Liu, Thayar, et al.)

Common use case 3: funcX as a platform

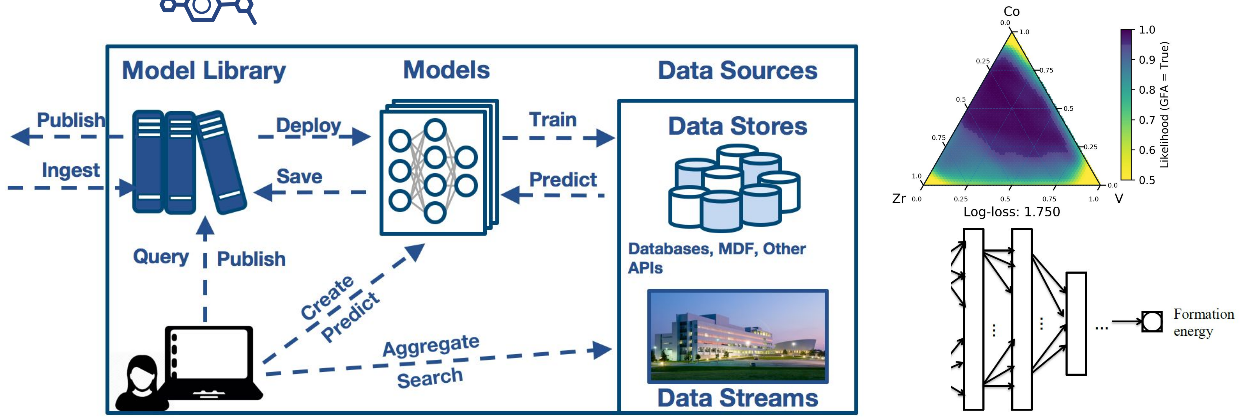
Building new applications and services that seamlessly execute application components or user workloads on remote resources

Advantages:

- Robust, secure, and scalable platform for managing parallel and distributed execution across a federated ecosystem of computing endpoints
- Simple cloud-based API and Python SDK for integration

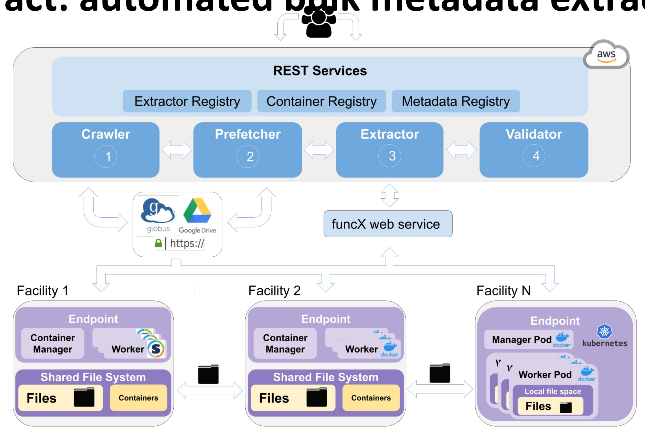
Examples:

The Data and Learning Hub for Science (DLHub)



A hosted service that enables researchers to find, share, publish, and run machine learning models and discover training data for science. funcX enables remote inference on specialized resources.
(Chard et al. https://arxiv.org/pdf/1811.11213)

Xtract: automated bulk metadata extraction



An automated and scalable system for bulk metadata extraction from large, distributed research data repositories. Xtract orchestrates the application of metadata extractors to groups of files, using funcX to dispatch extractors to data.
(Skluzacek et al. https://doi.org/10.1145/3431379.3460636)

funcX usage is growing rapidly

231,000
registered functions

17.2 million
function invocations

335
users

3683
registered endpoints

121s
average function runtime

funcX Documentation Support Publications News Blog

Federated function as a service

Use funcX to execute functions across a federated ecosystem of funcX endpoints.

https://funcx.org

https://funcx.org/binder