Federated Function-as-a-Service to Power Distributed Computing Pipelines

Ian Foster

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

Presentation at ROSS'22, Dallas TX, November 13, 2022

funcX

Federated function as a service

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

Check out the videos from our online funcX/ParslFest 2021!

Try funcX

Use Binder to run funcX tutorials in hosted Jupyter notebooks. No installation required!

Try now »

Install funcX

Install the funcX SDK to register, share, and execute functions.

Quickstart »

Deploy an endpoint

Deploy a funcX endpoint on a laptop, cloud, or cluster.

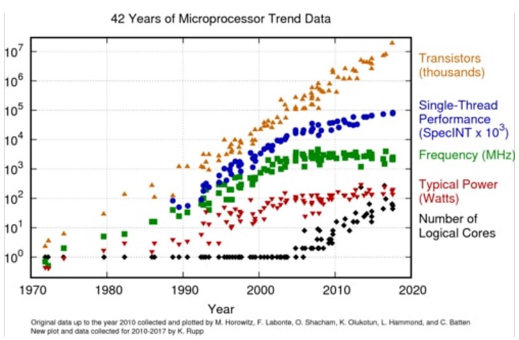
Learn more »

funcx.org

The scientific computing ecosystem is evolving rapidly

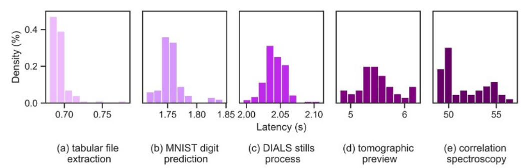
Resources

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



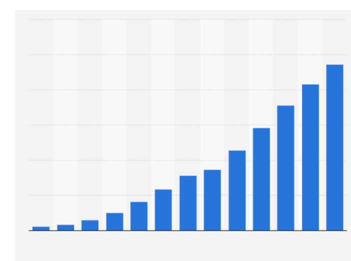
Workloads

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

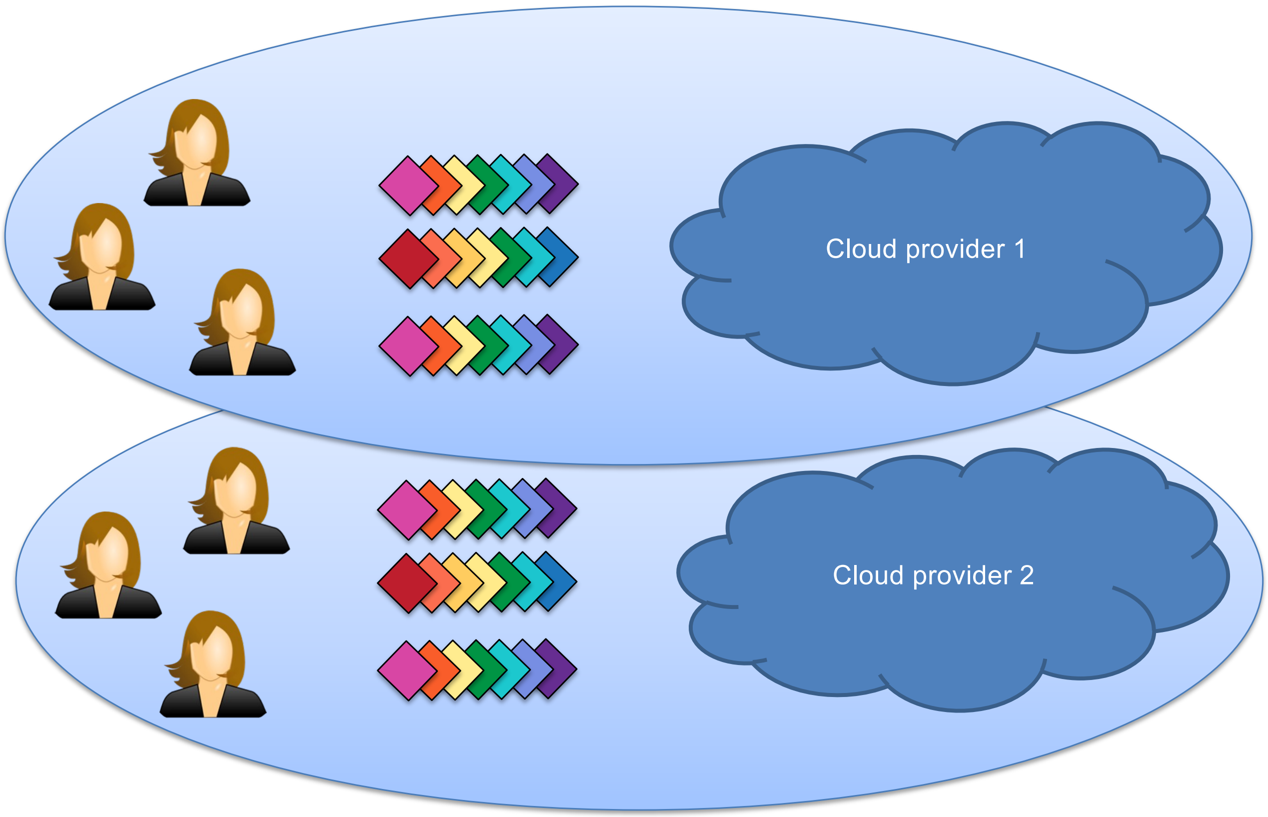


Users

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

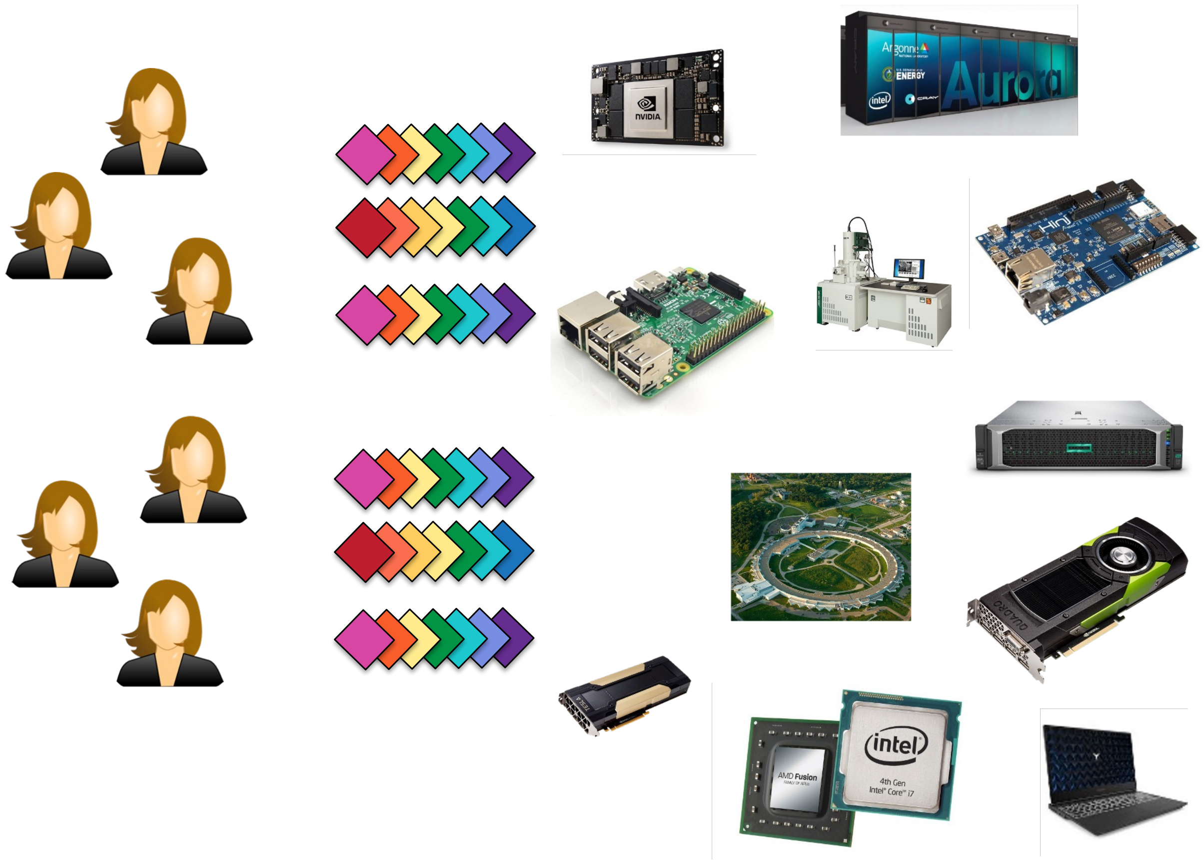


FaaS as offered by cloud providers



- 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?

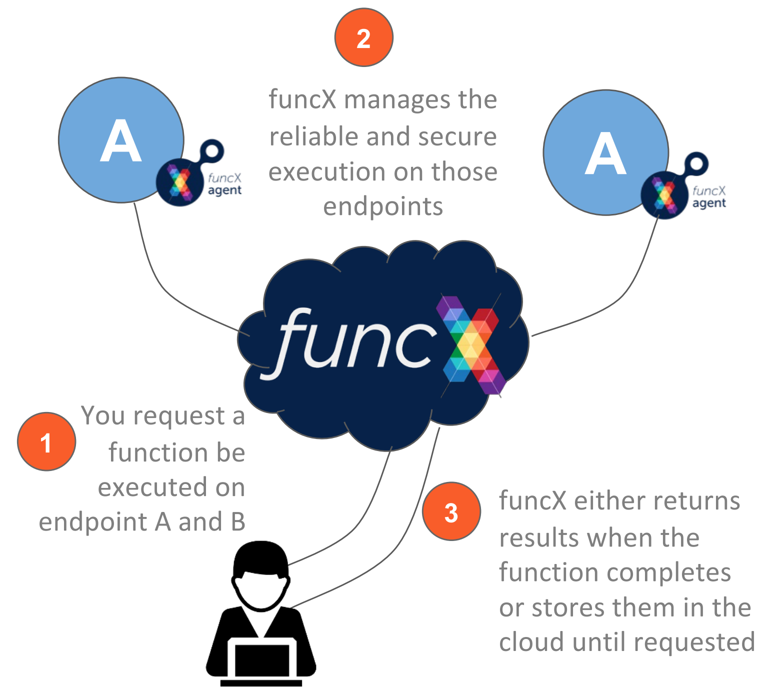


We still want:

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

funcX: managed and federated FaaS

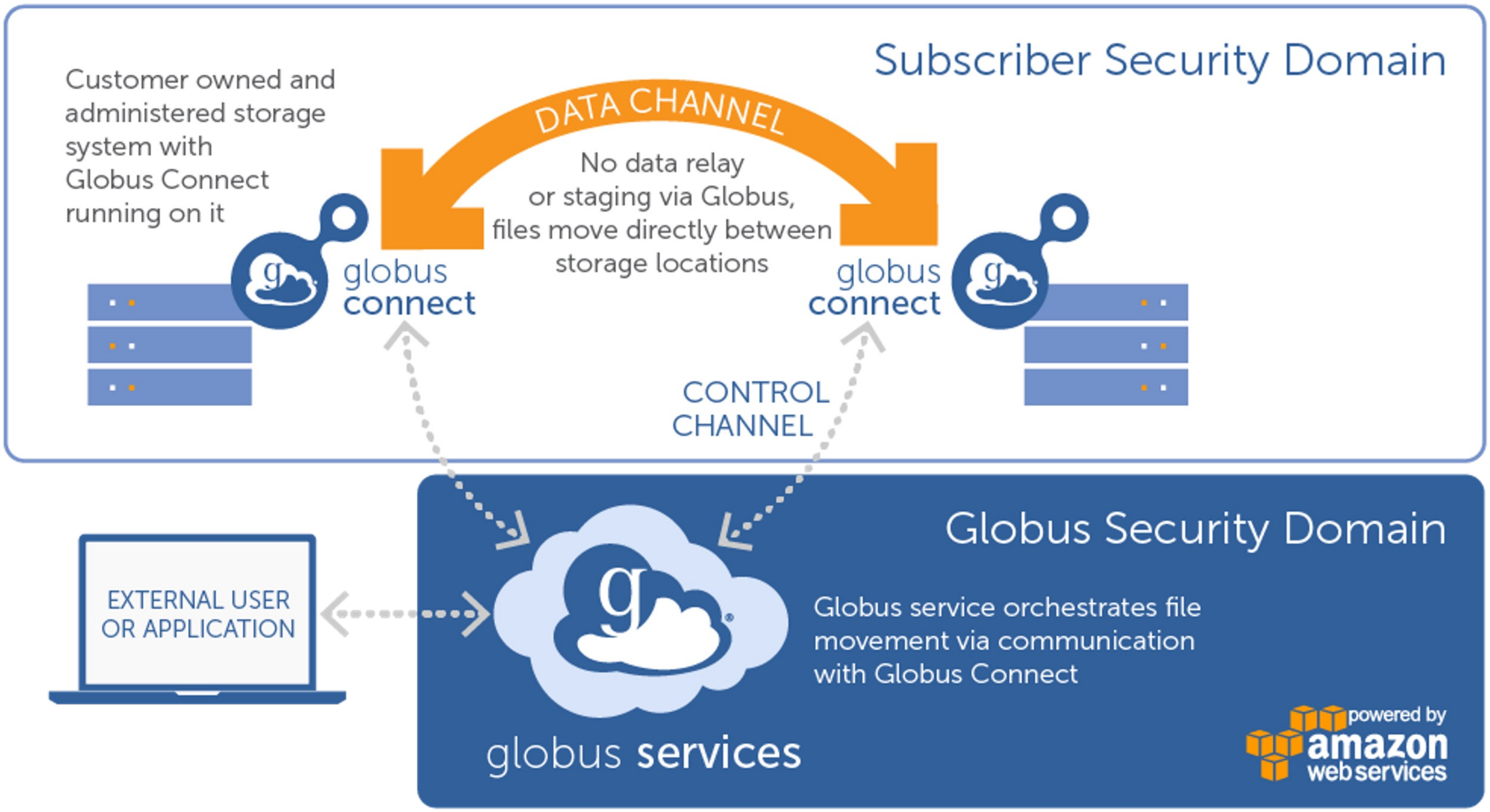
- Cloud-hosted service for managing compute
- Register and share *FaaS compute endpoints*
- Register and share Python functions
- Reliable, scalable, secure function execution

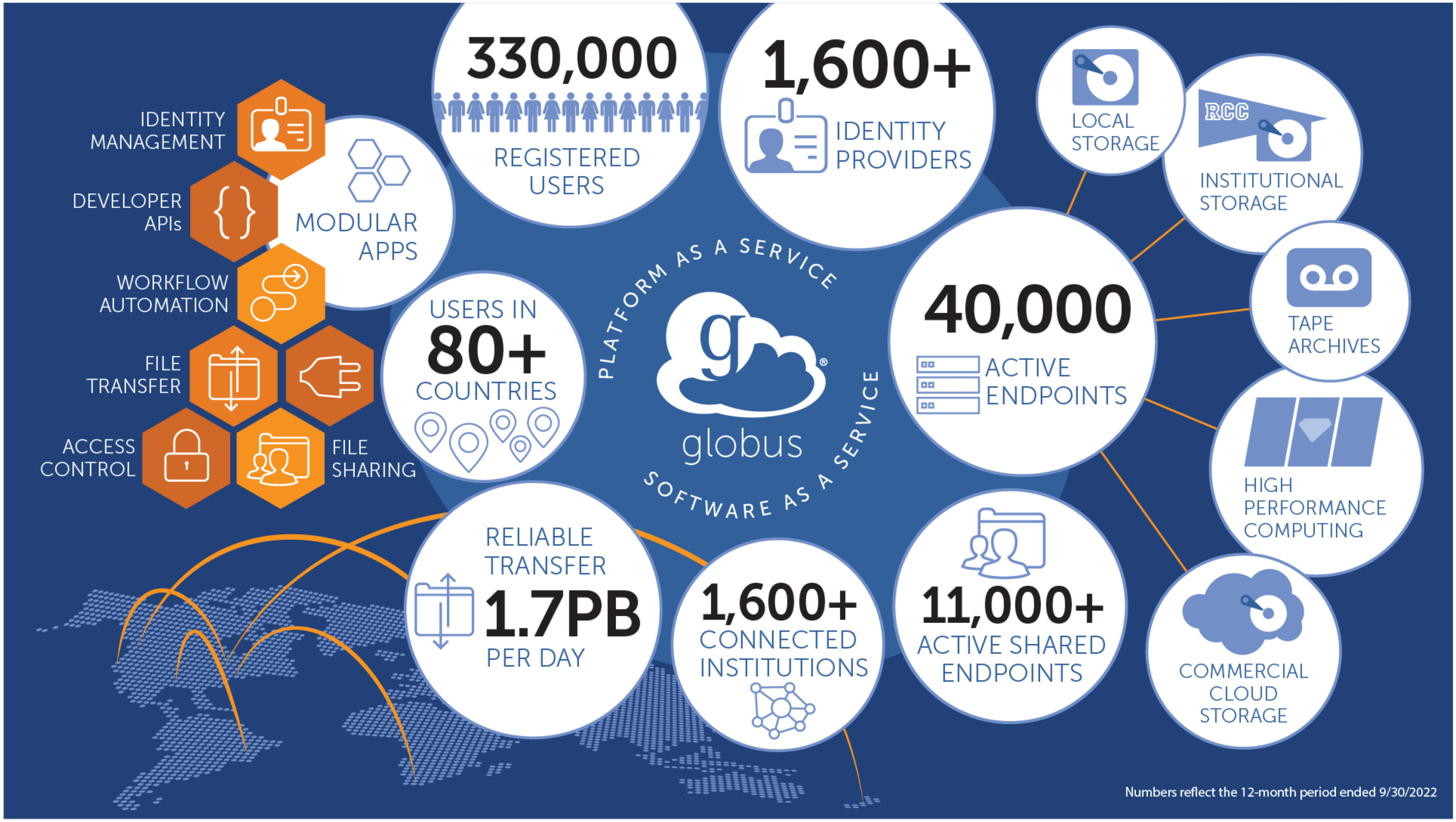


Try funcx on Binder

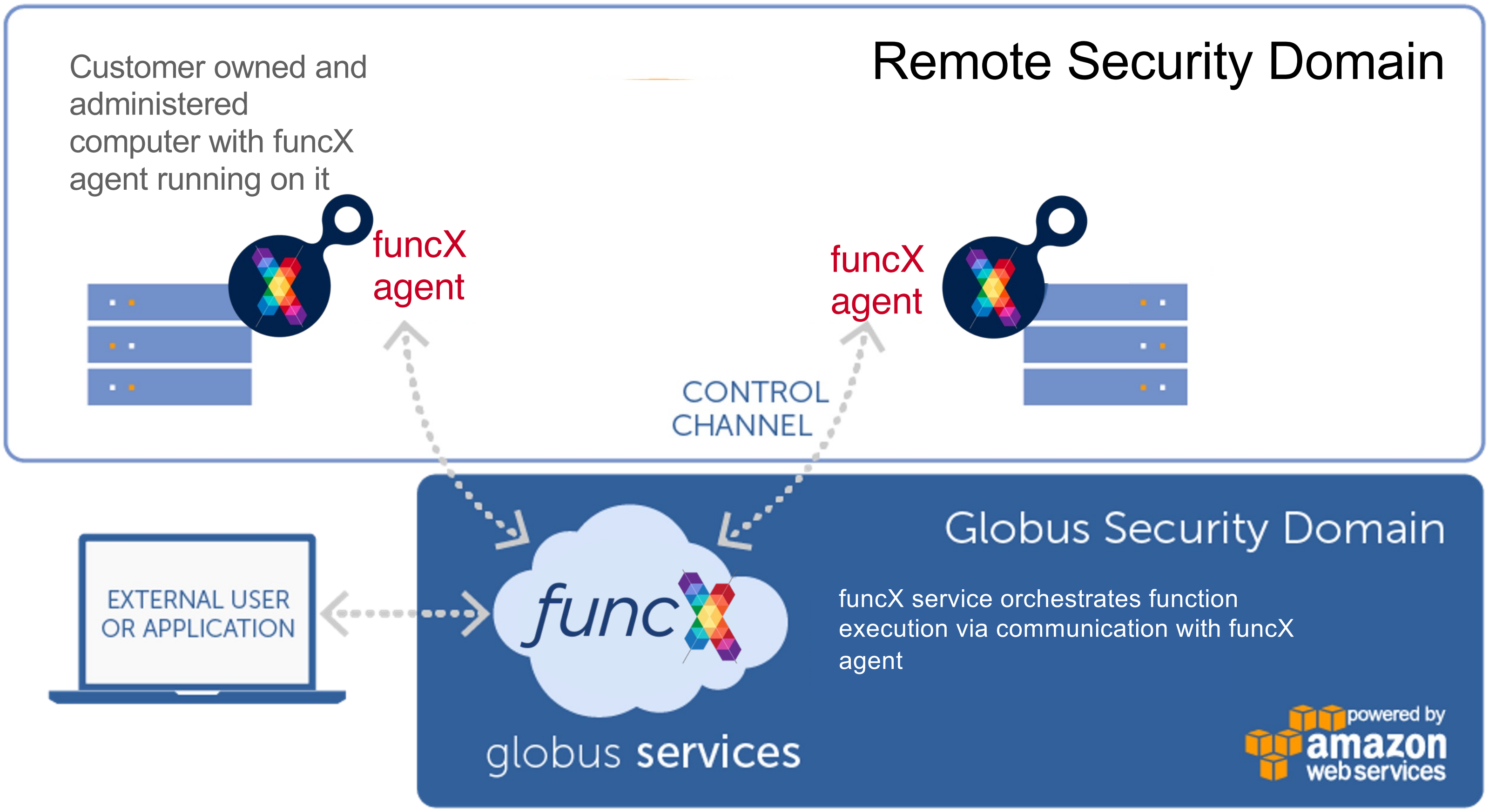
funcX https://funcx.org

Globus hybrid “SaaS” model: Global auth and data fabric




IDENTITY MANAGEMENT
DEVELOPER APIs
WORKFLOW AUTOMATION
FILE TRANSFER
ACCESS CONTROL
FILE SHARING
MODULAR APPS
330,000
REGISTERED USERS
1,600+
IDENTITY PROVIDERS
USERS IN
80+
COUNTRIES
PLATFORM AS A SERVICE
globus
SOFTWARE AS A SERVICE
40,000
ACTIVE ENDPOINTS
LOCAL STORAGE
RCC
INSTITUTIONAL STORAGE
TAPE ARCHIVES
HIGH PERFORMANCE COMPUTING
COMMERCIAL CLOUD STORAGE
RELIABLE TRANSFER
1.7PB
PER DAY
1,600+
CONNECTED INSTITUTIONS
11,000+
ACTIVE SHARED ENDPOINTS
Numbers reflect the 12-month period ended 9/30/2022

FuncX hybrid “SaaS”: Global compute fabric



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
- Fire-and-forget execution: outsource complexity of remote execution to funcX
- OAuth-based security model to access and share functions and endpoints

Choose where to execute functions

- Closest, cheapest, fastest, accelerators …

Common use case 1: Fire-and-forget execution

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

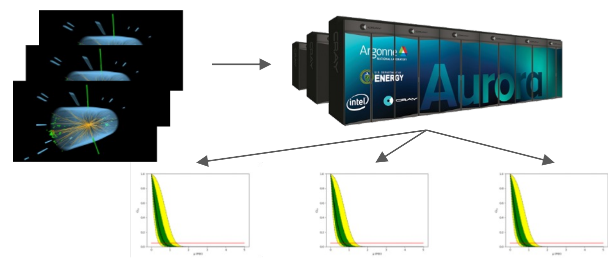
Advantages:

- 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

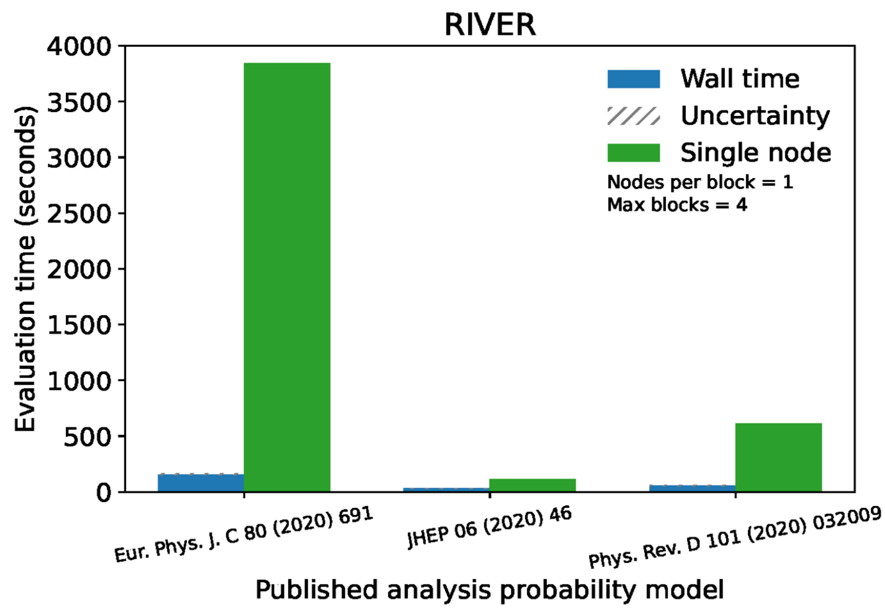


Wrapping a C-based statistical inference tool as a function so scientists can easily fit multiple different hypotheses for new physics signatures (signals).
(Feickert et al., arXiv:2103.02182)

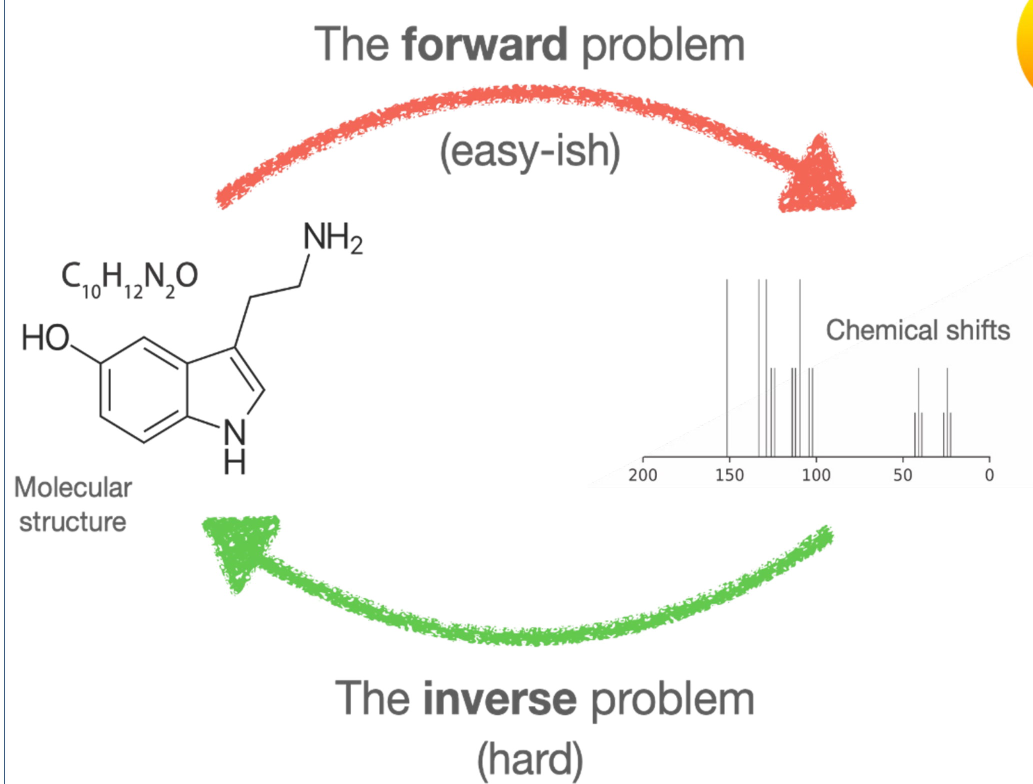
Application: Fitting-as-a-Service

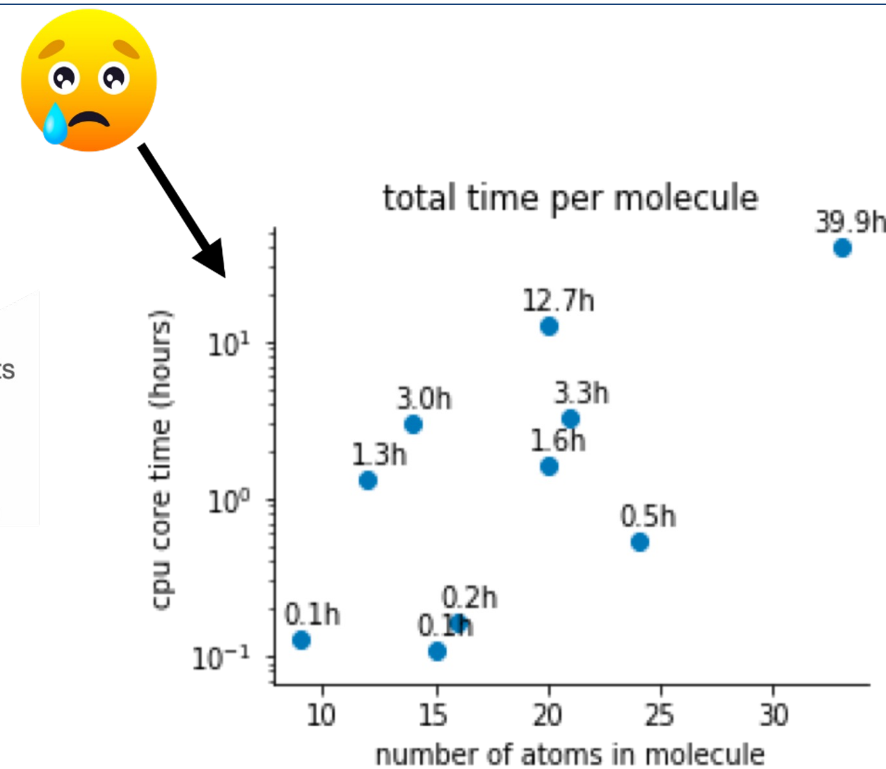
Scaling of Statistical Inference

- Fitting all 125 models from pyhf pallet for published ATLAS SUSY 1Lbb analysis
- Using University of Chicago River cluster: 2 minutes 30 seconds



Application: Inverse Spectroscopy





Courtesy Eric Jonas, University of Chicago

Use Case: Inverse Spectroscopy

- Typical run involves 100,000 tasks
- Average of 40 core-hours per task
- Would take 7 years on a modern workstation
- Able to complete analysis in one month at TACC
- Fire and forget: Launch 100,000 tasks

***"funcX lets us all spend more time on science and less on infrastructure!"* Eric Jonas**

Common use case 2: Automated data analysis

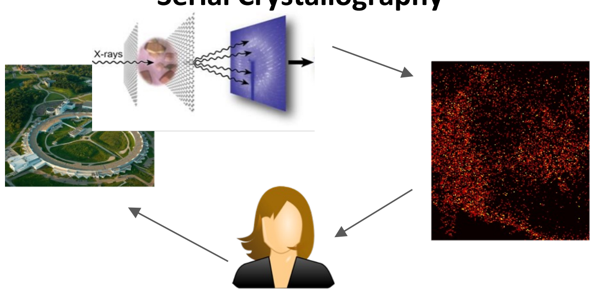
Construct and run automated analysis pipelines that include steps that need to execute in different locations (e.g., near instrument, in data center, on specialized hardware)

Advantages:

- Automatically process data as acquired (event- and workflow-based)
- Integrate with data movement and other actions (both human and machine)
- Execute functions across the computing continuum (close to data, on accelerators, …)

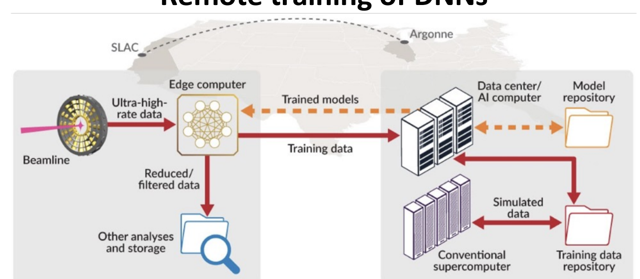
Examples:

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.)

Use case: Research Automation

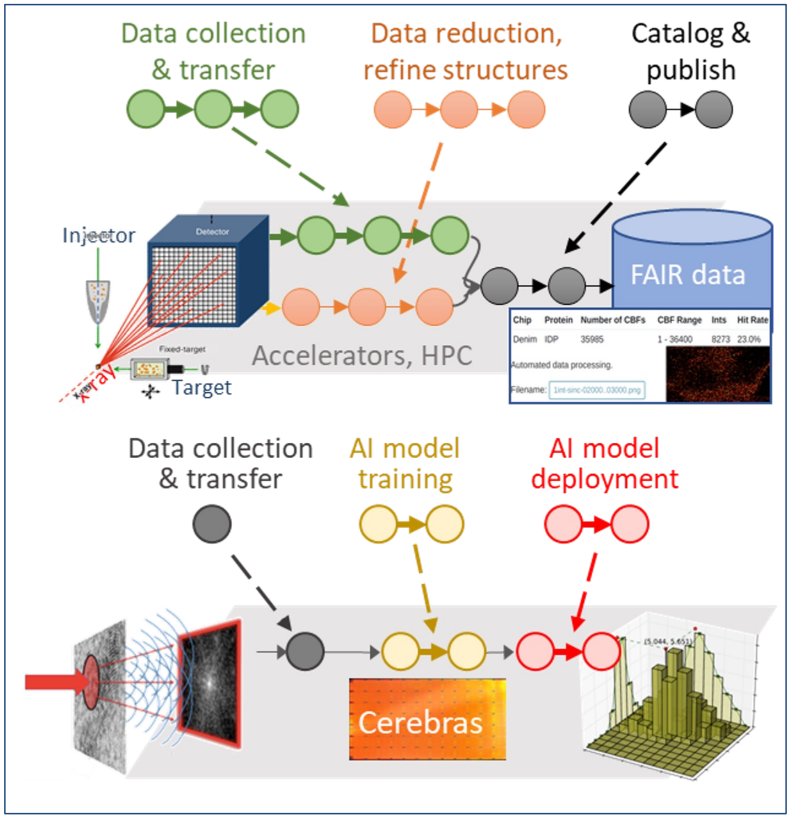
Light source experiments process samples with bright, high-energy x-rays

- XPCS: studying materials dynamics
- SSX: solving crystal structures
- HEDM: studying microstructure evolution

Automation allows researchers to catalog data automatically, process samples faster, perform real-time control, etc.

Most flows require computation

- Quality control, reconstruction, analysis, machine learning training, transformation, inference, plotting, visualization, metadata extraction, aggregation

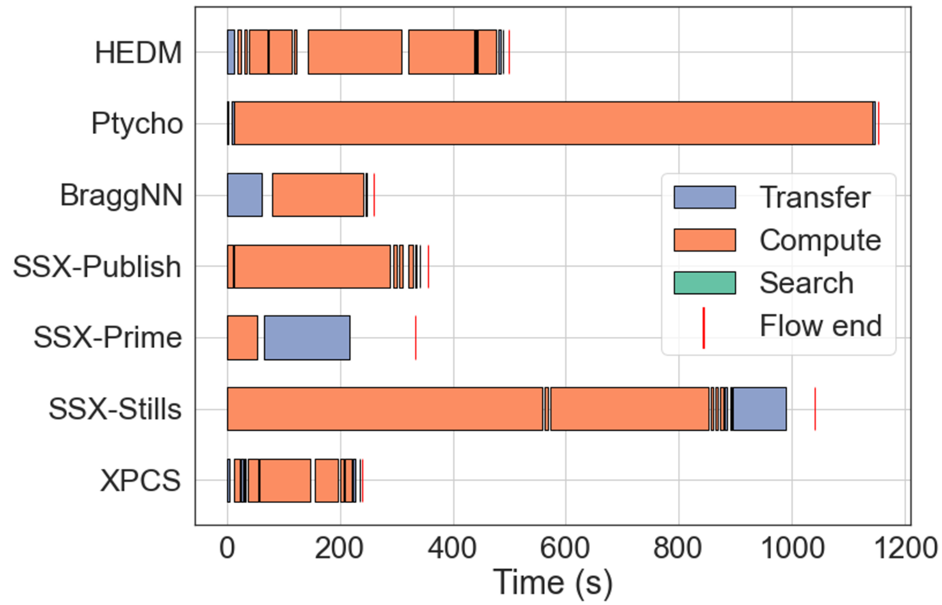


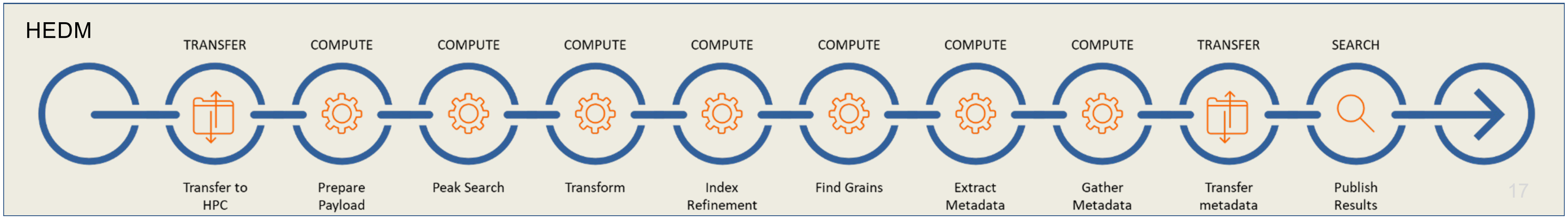
Linking Scientific Instruments and HPC: Patterns, Technologies, Experiences **https://arxiv.org/abs/2204.05128**

funcX action provider enables seamless integration in flows

Globus Flows can invoke arbitrary functions via the funcX action provider

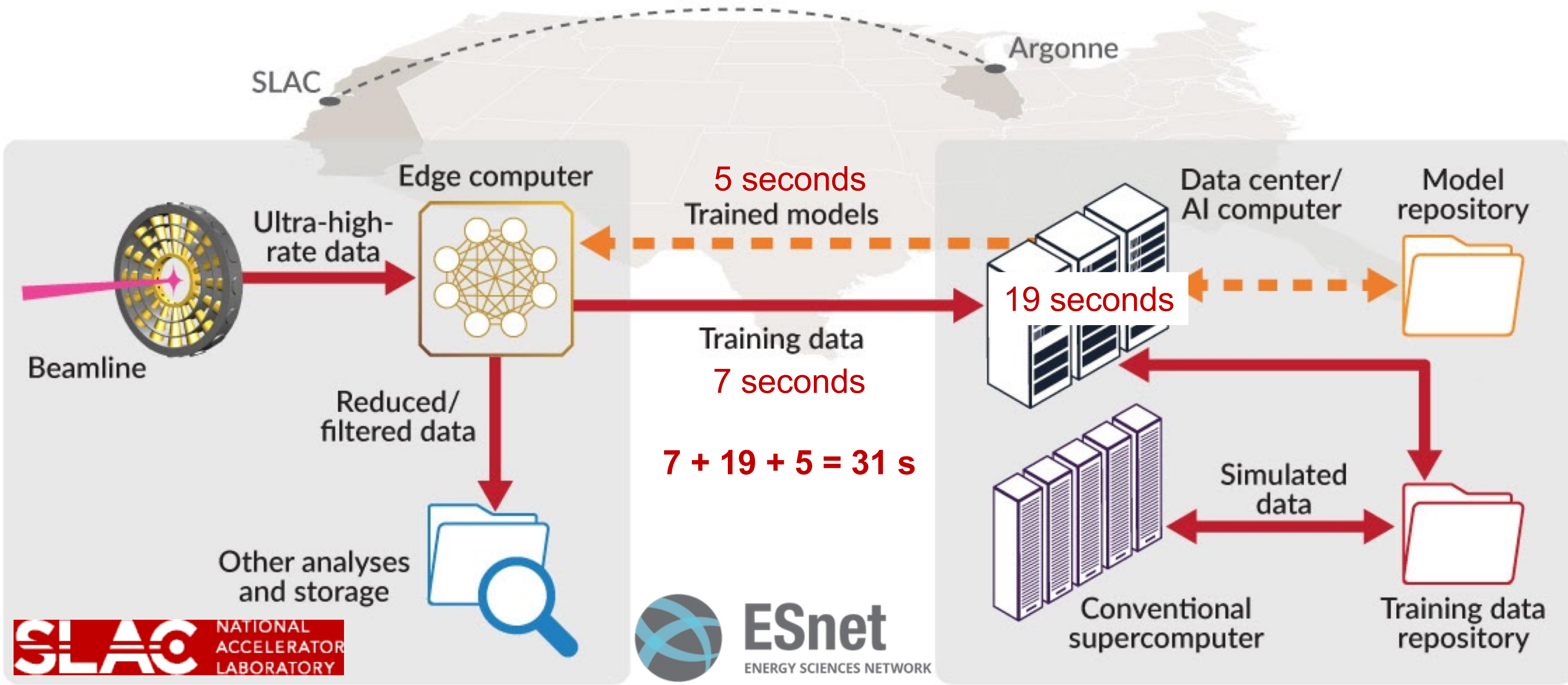
Functions may be executed in various locations: at the beamline, local server, cluster, cloud





CryoEM automation

High energy diffraction microscopy



https://doi.org/10.48550/arXiv.2105.13967

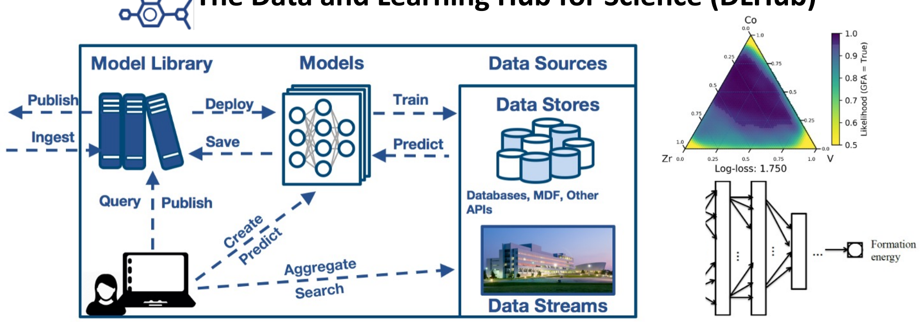
Common use case 3: funcX as a platform

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

Advantages:

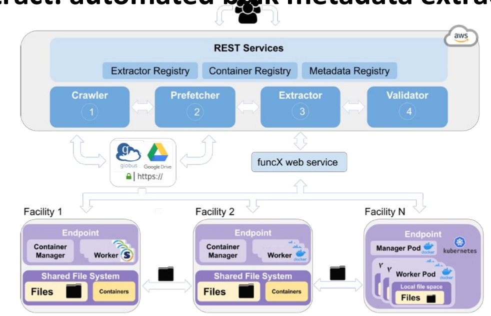
- Robust, secure, and scalable platform for managing parallel and distributed execution across a federated ecosystem of computing endpoints
- Simple cloud-based API and Python SDK for integration

The Data and Learning Hub for Science (DLHub)



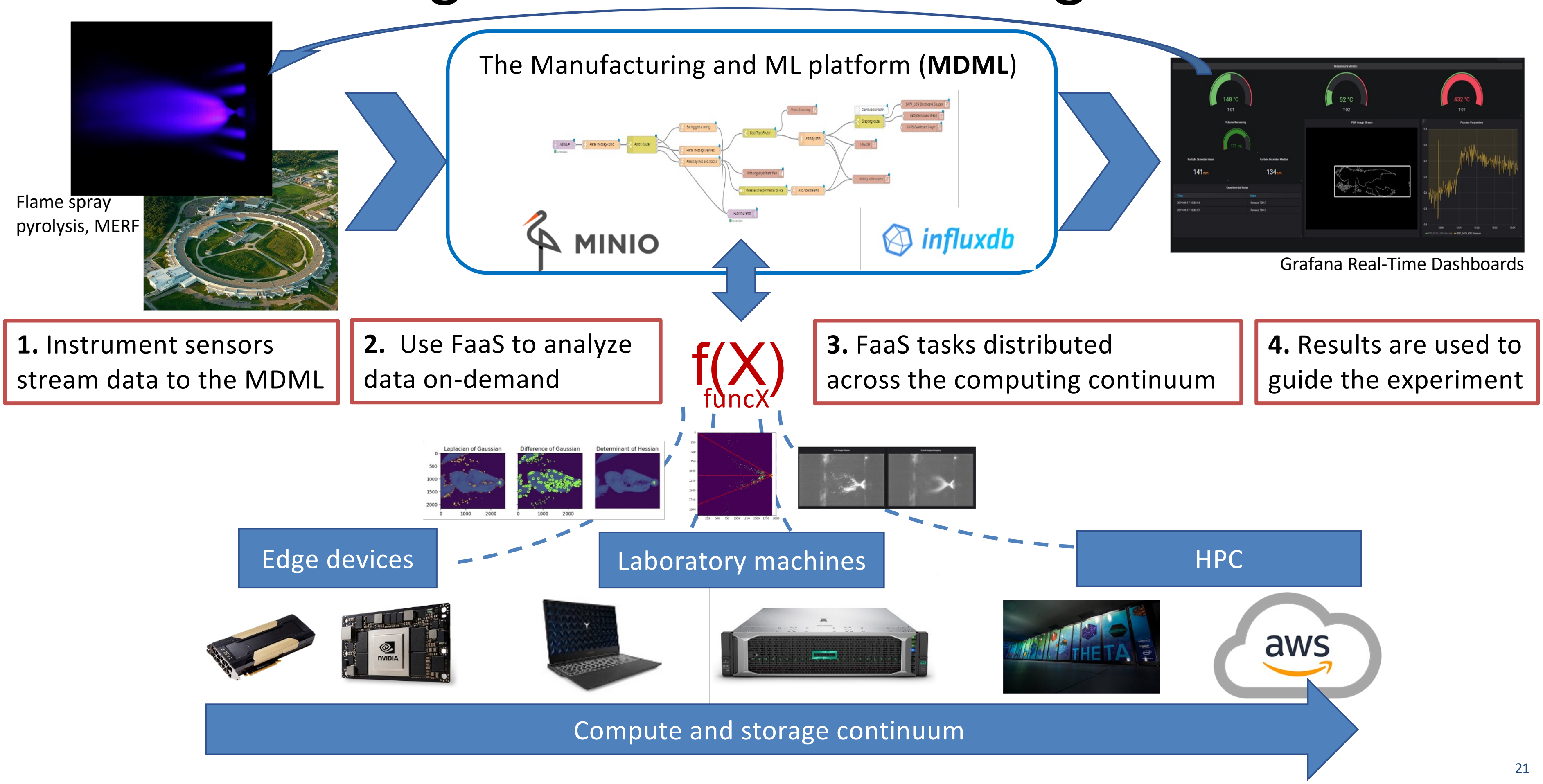
A hosted service that enables researchers to find, share, publish, and run machine learning models and discover training data for science. funcX enables remote inference on specialized resources.
(Chard et al. https://arxiv.org/pdf/1811.11213)

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)

Manufacturing and machine learning



funcX usage is growing rapidly

231,000
registered functions

17.2 million
function invocations

335
users

3683
registered endpoints

121s
average function runtime

Transform laptops, clusters, clouds into function serving endpoints

- Python-based agent and pip installable locally or in Conda
- Elastically provisions resources from local, cluster, or cloud system
  - Using Parsl library
- Manages concurrent execution on provisioned resources
- Optionally manages execution in Docker, Singularity, Shifter containers
- Share endpoints with collaborators

```
$ pip install funcx-endpoint

$ funcx-endpoint configure myep

$ funcx-endpoint start myep
```

Register and share functions

Create funcX client (and authenticate)

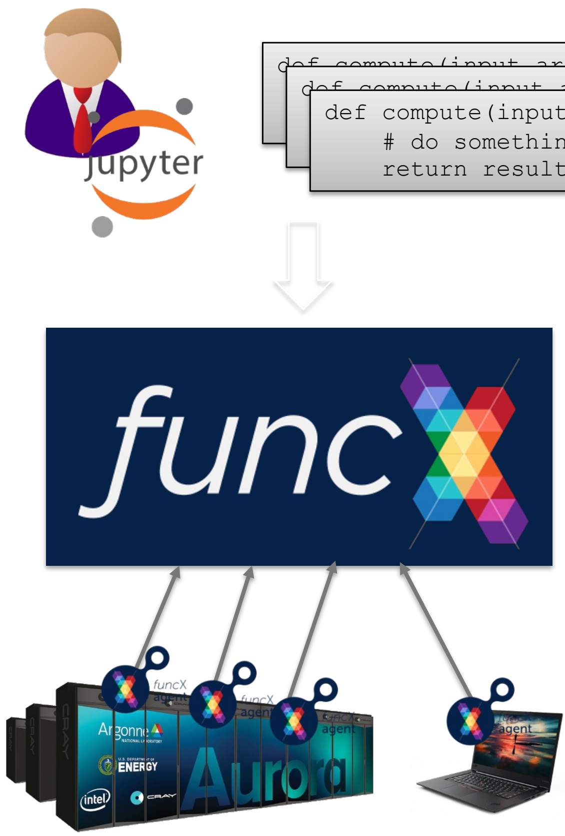
```
from funcx.sdk.client import FuncXClient

fxc = FuncXClient()
```

Define and register Python function

```
def hello_world():
    return "Hello World!"

func_uuid = fxc.register_function(hello_world)
print(func_uuid)
```



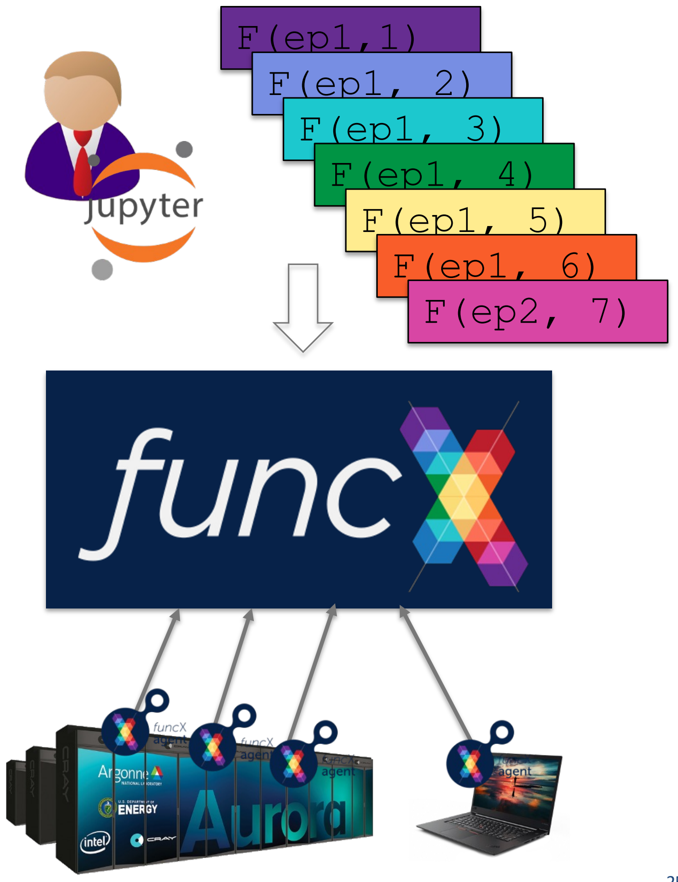
Execute tasks on any accessible endpoint

Select: function ID, endpoint ID, and input arguments

```
tutorial_endpoint = '4b116d3c-1703-4f8f-9f6f-39921e5864df'
res = fxc.run(endpoint_id=tutorial_endpoint,
              function_id=func_uuid,
              arg1, arg2, arg3)
```

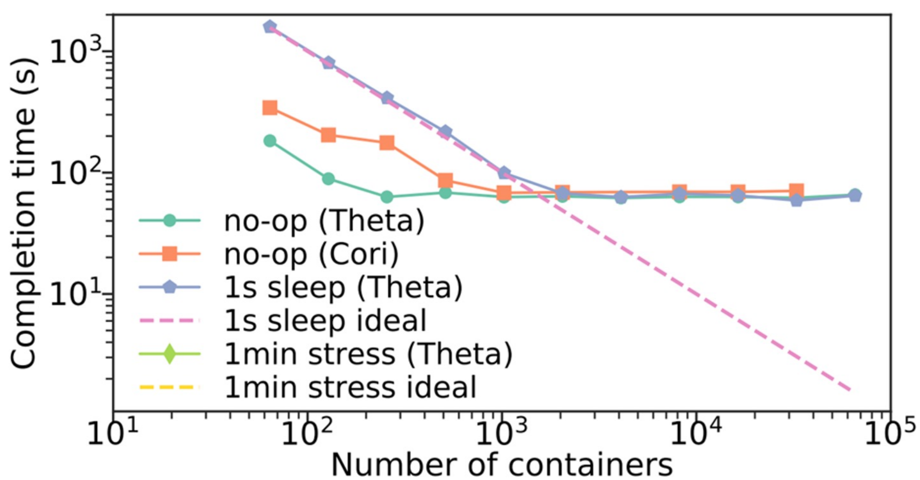
Retrieve results asynchronously (funcX stores results in the cloud)

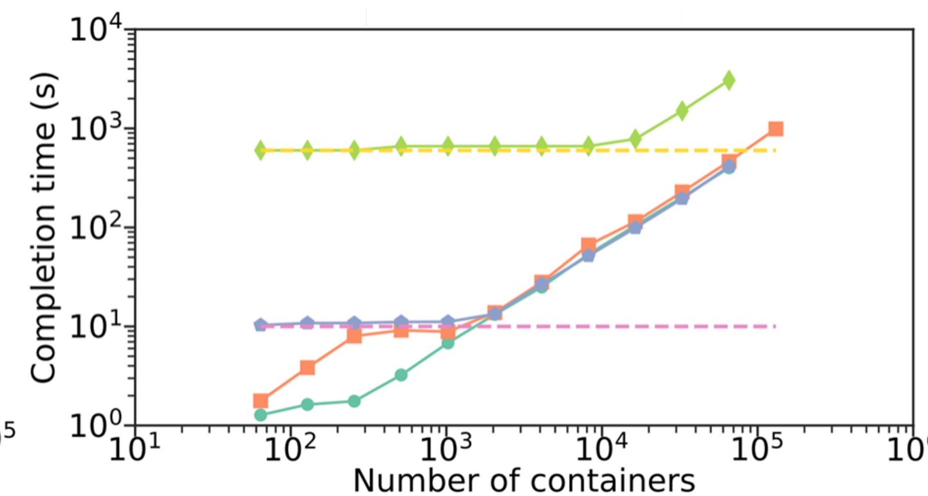
```
print(fxc.get_result(res))
```



funcX scales to 100K+ workers

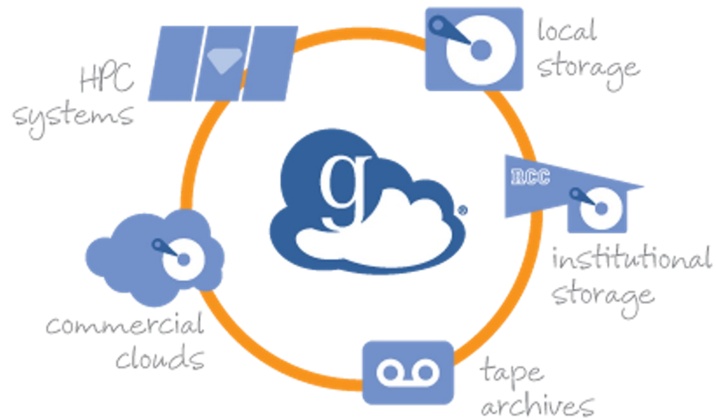
- funcX endpoints deployed on ALCF Theta and NERSC Cori
- Strong scaling (100K concurrent functions) shows good scaling up to 2K containers even with short no-op/sleep tasks
- Weak scaling (10 tasks per container) scales to 131K concurrent containers (1.3M tasks)

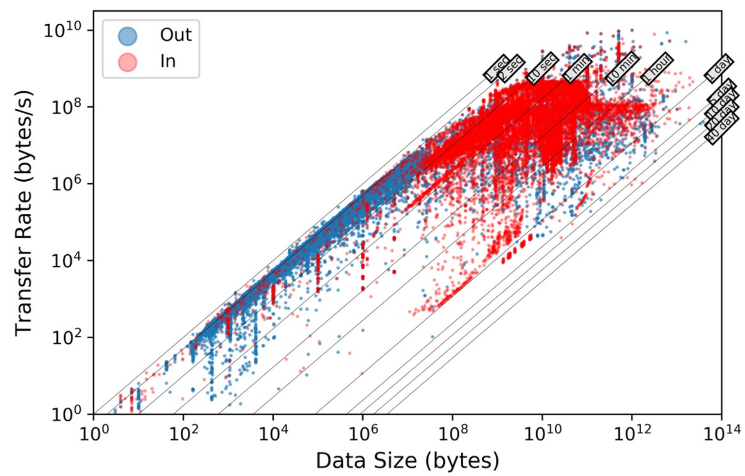




How can we improve data management to/from/between functions?

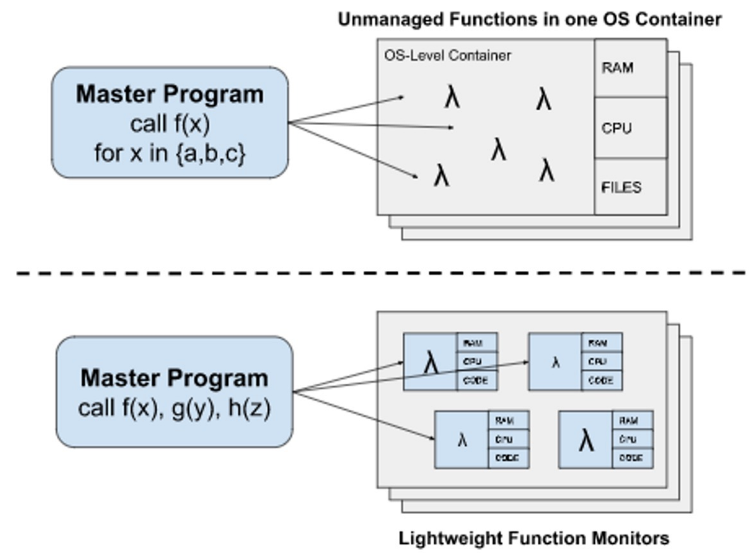
- Research functions are reliant on data
  - Input, output, and between functions
  - Federated environments may have huge latency and bandwidth limitations
- Files, objects, other data?
- Stateless or stateful functions?
  - ML steering and coordination
- Research directions:
  - Low latency communication that supports application patterns
  - Programming models that are data centric
  - Transparent wide-area movement
  - Intuitive and intelligent caching
  - Dataspace-like models





How can we reduce the overheads associated with managing compute environments?

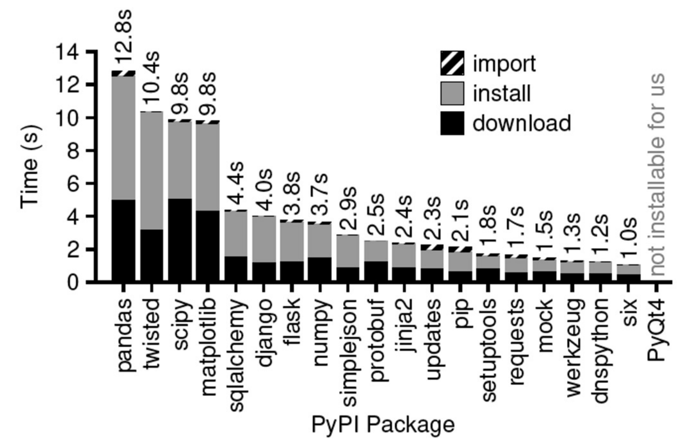
- Container technologies are becoming increasingly diverse (Docker, Singularity, Firecracker, etc.); no one solution works everywhere
- Containers are relatively heavyweight (especially those used in HPC environments)
- Programming virtualization faster, yet insecure
- Research directions:
  - New methods at the function level for
    - Creating execution environment
    - Sandboxing execution
    - Managing resource usage



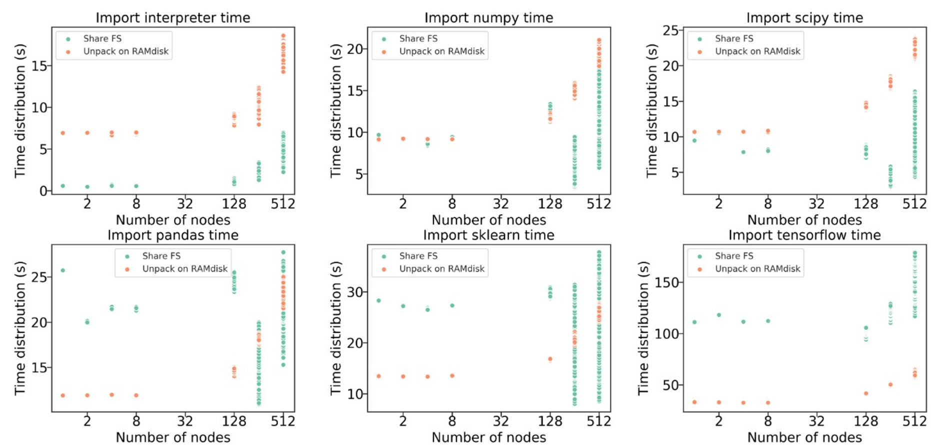
T. Shafter, et al. Lightweight Function Monitors (LFMs) @IPDPS

Can we balance the trade off between start time and resource utilization?

- Cold-starts are challenging in the cloud and more so on research CI
    - ***Node Acquisition:*** For endpoints in HPC clusters, latency of allocating nodes
    - ***Container Instantiation:*** For functions that require containers, starting them
    - ***Package Loading:*** Installing and importing necessary packages

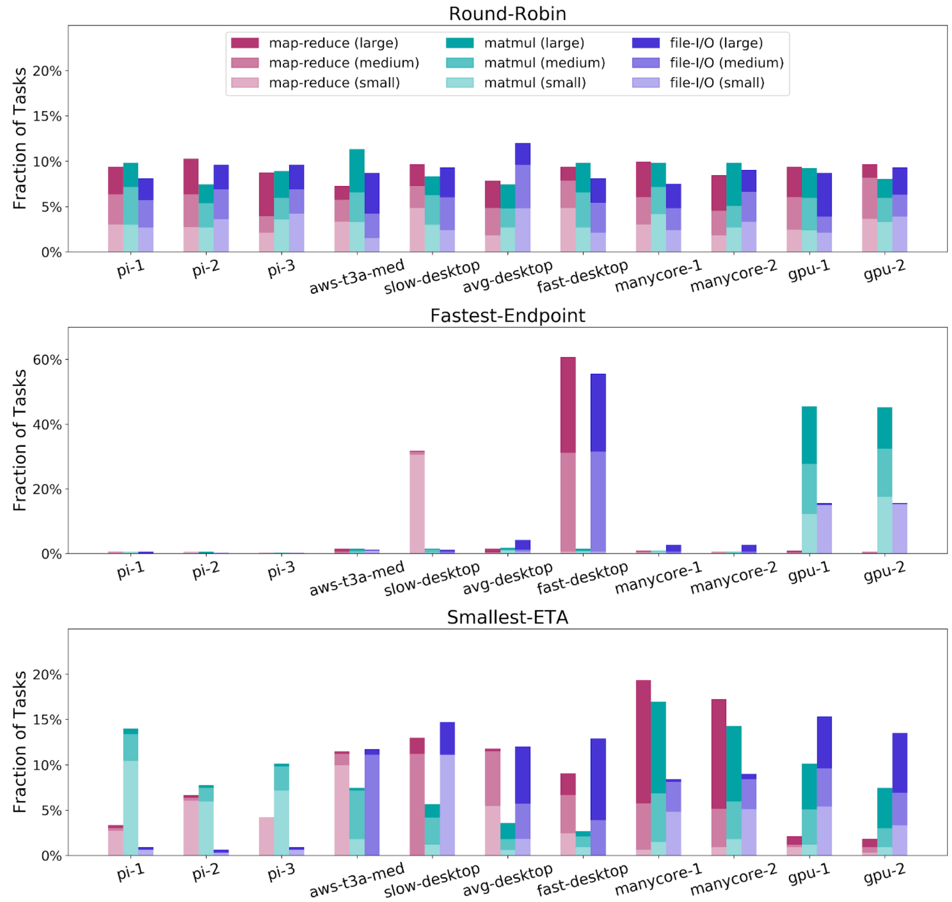


- Research directions:
    - Lightweight virtualization (e.g., Firecracker)
    - Intelligent environment caching, transfer, loading



Can we efficiently schedule function executions in a federated environment?

- We have an environment with varying performance and overheads
  - Execution, transfer, cold start, …
- **Delta**: Experiment with scheduling across heterogenous funcX endpoints
  - Raspberry Pis, Desktops, Cloud instances, GPUs
  - Three scheduling algorithms: Round robin, Fastest endpoint, smallest ETA
  - Smaller tasks distributed across slower endpoints
- Research directions:
  - Modelling various overheads
  - New FaaS scheduling algorithms
  - Workflow scheduling



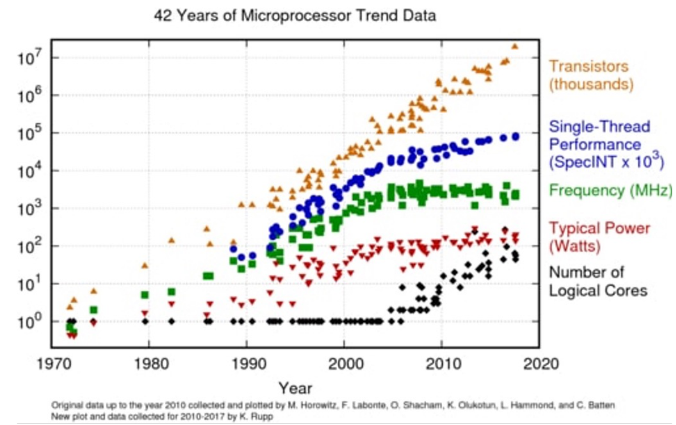
How should we deal with the other hard stuff: security, policies, regulations, …?

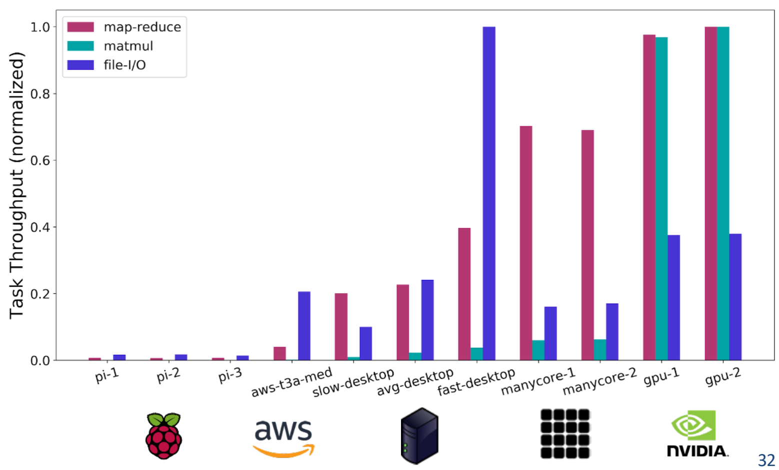
- FaaS creates new security challenges
  - Simple sharing of functions and endpoints, remote access to resources, containerized execution environments, …
  - Likely an attractive target to attackers

- Research directions:
  - Scalable monitoring, logging, auditing
  - Web-based authentication and authorization frameworks
  - Container/function security (e.g., application whitelists)

Can we make it as easy to compute remotely as we compute locally?

- Increasing heterogeneity makes this:
  - necessary (to improve efficiency and gain access to specialized capabilities)
  - challenging (to abstract differences between systems)
- Inflexible authentication and authorization models (e.g., 2FA)
- Widely varying performance makes it hard to optimize performance
- Diverse workloads
  - Event-based and interactive computing impose real-time requirements
  - Machine learning steering requires iterative feedback loops





Lessons learned applying funcX to science use cases

- Abstracts the complexity of using diverse compute resources
- Simplicity: automatic scaling, single interface
- Flexible web-based authentication model
- Enables event-based processing and automated pipelines
- Increases portability between sites, systems, etc.
- Resources can be used efficiently and opportunistically
- Enables secure function/endpoint sharing with collaborators

- FaaS is not suitable for some applications
- Ratio of data size to compute must be reasonable
- Containerization does not always provide entirely portable codes
- Coarse allocation models do not map well to fine grain/short functions
- Decomposing applications is not always easy (or possible)

Parsl & funcX Fest 2022

funcx.org

Parsl & funcX Fest 2022 - The Parsl/funcX Community Meeting (Sep 13-14)

Join us for the second Parsl & funcXFest Community Meeting. The meeting will be held as a hybrid meeting on September 13-14, 2022. The in-person component will be held at the University of Chicago.

The meeting will bring together researchers, developers, and cyberinfrastructure experts from around the world to discuss experiences using and developing funcX and Parsl. Parsl is a parallel programming library and underpins funcX's endpoint software.

Registration (free): https://forms.gle/TEeuGPo4MwHNZML79. We invite lightning talks from the community and would love to hear about your recent work.

We have limited travel support available to attend the workshop. Please contact Kyle Chard (chard@uchicago.edu) for information.

Agenda

Tuesday, September 13, 12 pm - 5 pm CDT (17:00 - 22:00 UTC)

12:00 - 1:00 pm Lunch

1:00 pm - Welcome! - Kyle Chard, University of Chicago/Argonne National Laboratory. slides, video funcX Documentation Support Publications News Blog

funcX

Federated function as a service

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

https://funcx.org

https://funcx.org/binder

foster@uchicago.edu