Federated HPC for On-demand Data Science

**Ryan Chard**, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Stephen Rosen, Josh Bryan, Ben Galewsky, Dan Katz, Ian Foster, Kyle Chard, and others!
The need for federated, on-demand HPC

1. Support a new class of workloads
   • Real-time, interactive, stream processing, next-gen instruments, ML training, on-demand inference

2. Democratize access to diverse resources
   • Simplify scaling up
   • Facilitate multi-site deployments
   • Standardize access to specialized resources
   • Abstract heterogeneous compute infrastructure

3. Enable fluid function execution across the computing continuum
   • Containers enable portability and sandboxing
   • Networks allow code and data to be sent anywhere
Existing research computing infrastructure has significant barriers for use

- Complex queuing systems with unpredictable delays
- Coarse allocation blocks
  - Not designed for short-duration tasks with variable resource needs
- Steep learning curve and lack of portability
  - Translation to different schedulers (and update when they inevitably break)
  - Heterogeneous architectures
  - Different modules and source code
  - Different container technology

There is an impedance mismatch between many workloads and existing infrastructure available to scientific users
Serverless computing

Provider runs infrastructure and manages allocation of resources

Function as a Service (FaaS)
  • Pick a runtime (python/JS/R etc.)
  • Write function code
  • Run (and scale)

Low latency, on-demand, elastic scaling

Combine functions to solve complex problems
funcX: creating a function serving ecosystem

Functions:
- Register once, run anywhere, any time

Endpoints:
- Dynamically provision resources, deploy containers, and execute functions
- Exploit local architecture/accelerators

funcX Service:
- Register and share endpoints
- Register, share, run functions

Turn *any* machine into a function serving endpoint

Route functions to remote endpoints
- Closest, cheapest, fastest, accelerators
  ...
Deploying a funcX endpoint

- Pip install funcX (e.g., using Conda)
- **Authenticate** and register with the funcX service
- Configure the endpoint for the local resources
- Examples: https://funcx.readthedocs.io/en/latest/endpoints.html#example-configurations
Coding the computing continuum with funcX

1. Define Python functions and register functions with funcX
   - Codes are serialized and stored on the cloud
   - Registration returns a UUID for the function which is used for invocation

2. Run the function on a specified endpoint
   - args* and kwargs* are serialized and sent to funcX
   - Function code and inputs routed to endpoint

3. Retrieve Results
   - Inspect status, wait on results, retrieve outputs

```
[1]: from funcx.sdk.client import FuncXClient
    fxc = FuncXClient()

[2]: def funcx_sum(items):
    return sum(items)

[3]: func_uuid = fxc.register_function(funcx_sum,
                                               description="A sum function")
    print(func_uuid)
    ce23d1c0-91f1-49df-a30b-0672453d8f9b

[4]: payload = [1, 2, 3, 4, 66]
    endpoint_uuid = '4b116d3c-1703-4f8f-9f6f-39921e5864df' # Tutorial endpoint
    res = fxc.run(payload, endpoint_id=endpoint_uuid, function_id=func_uuid)
    print(res)
    7508c2e7-3026-4ee3-95b3-25f7a605d893

[5]: fxc.get_result(res)
[5]: 76

Portable code
Python
Docker, Shifter,
Singularity

Any access
SSH, Globus,
cluster or HPC
scheduler

Any computer
Clusters,
clouds, HPC,
accelerators
Data Science with funcX

- Train ML models
- ML inference on-demand
- Distributed analysis pipelines
- Multi-site HPC campaigns
- Screening 180B molecules
- Solving covid structures
- Fitting as a Service tool for HEP
- Inverse spectroscopy at scale
- On-demand metadata extraction
- RuralAI and robots at the edge
- Delta: smart placement of tasks
- FLoX: federated learning on funcX
Thanks!

https://funcx.org

rchard@anl.gov

Join us on Slack: https://funcx.org/support.html