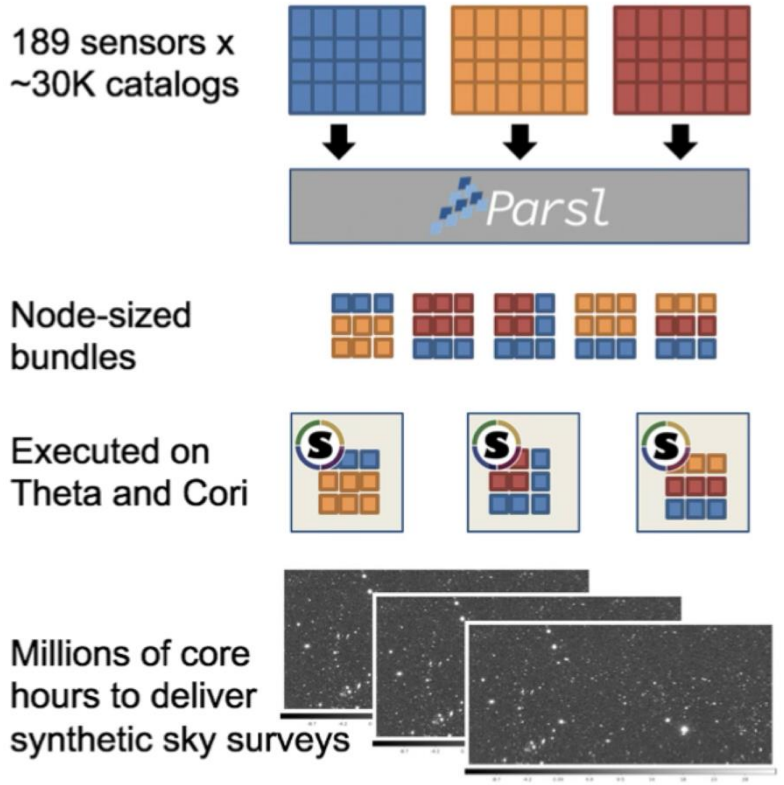
# Workflow Tools for Science

Christine Simpson
Data Services & Workflows Group, ALCF

# What is a Workflow?

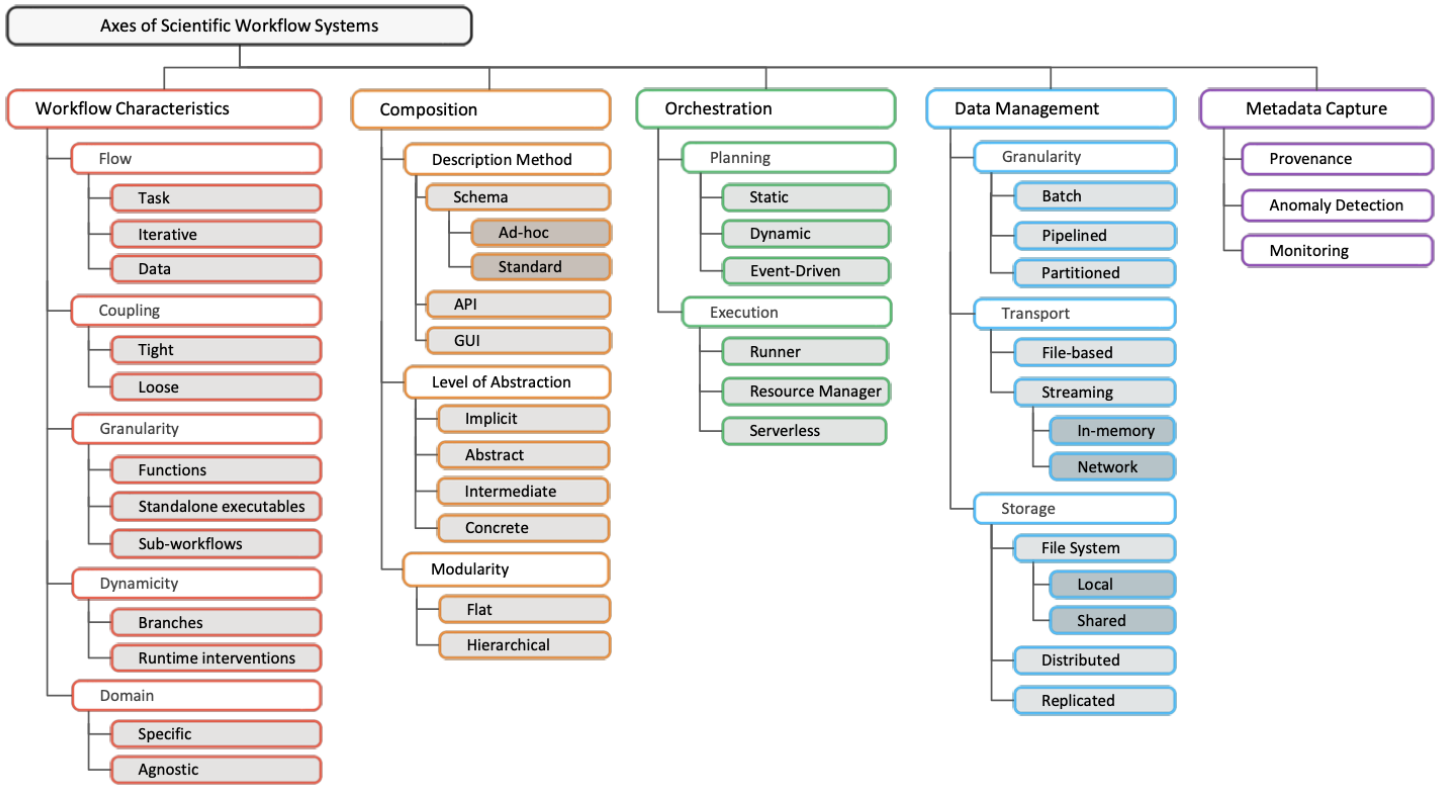
- A workflow is simply any collection of computational tasks run to achieve a result
- Workflows can vary in size, most users have some type of workflow, whether or not they call it that
- Traditionally at ALCF, `workflows` have been large job or task campaigns with some set of programmed dependencies
- Increasingly, AI/ML have become part of workflows, including agents (see later talks in the day!)




Villarreal et al. "Extreme Scale Survey Simulation with Python Workflows."
Proceeding for eScience 2021

# Elements of a Workflow

- Workflows usually include one or more of the following:
  - Task Launching
  - Task Dependencies
  - Data Movement
  - Orchestration
  - Monitoring
- A workflow tool is a software package that can handle one or more of these aspects
- Many tools out there (100s!)




Suter et al. "A terminology for scientific workflow systems". Future Generation Computer Systems, vol 174, Jan. 2026.

# Workflow Tools

- There are many Workflow tools out there!
- We'll cover a few that are used on ALCF systems, i.e. relevant to HPC workloads at scale
- Local Executors:
  - Parsl
  - DragonHPC
- Remote Executors:
  - Globus Compute
  - IRI/Facility API




# Parsl: *A parallel programming library for Python*

- Simple installation with pip
- Apps define how to run tasks
  - Python apps call python functions
  - Bash apps call external applications
- Workflow contained within memory (no database)
- Can elastically scale work on scheduled machine (e.g. submit jobs to PBS/SLURM)
- Configuration (assignment of tasks to hardware) set by user, separate from workflow logic and application definitions
- Community of 70+ developers, several at UChicago & ANL, part of Globus Labs

```python
@python_app
def hello ():
    return 'Hello World!'

print(hello().result())
```

```
Hello World!
```

```python
@bash_app
def echo_hello(stdout='echo-hello.stdout'):
    return 'echo "Hello World!"'

echo_hello().result()

with open('echo-hello.stdout', 'r') as f:
    print(f.read())
```

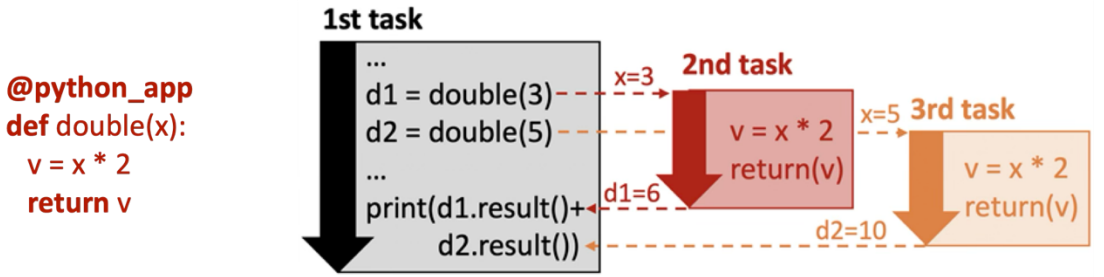
```
Hello World!
```

# Parsl Apps and Futures

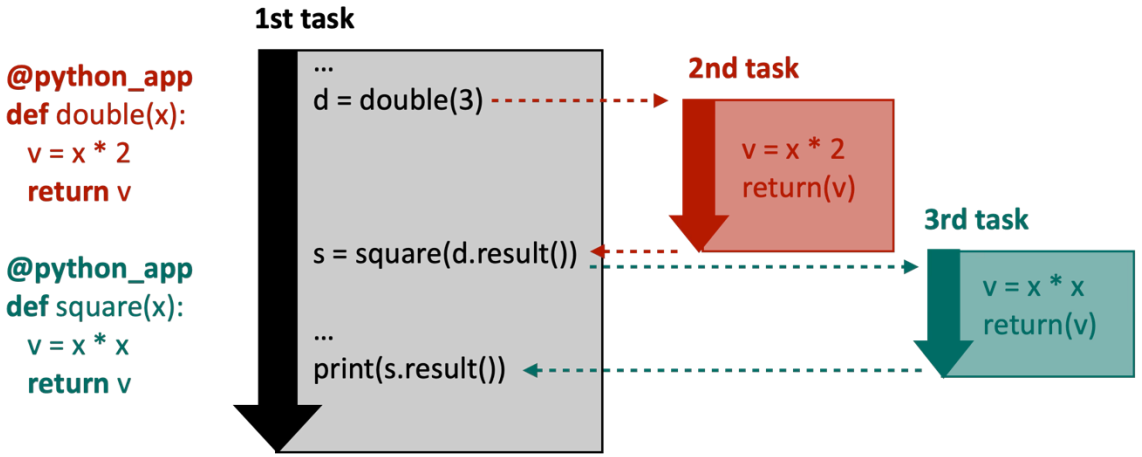
## *How tasks are made and linked*

- Parsl extends the Python concurrent.futures module
- Tasks are created by invoking apps that return an AppFuture
- A future is a proxy for a result that might not yet be available
- Task dependencies can be created by passing the AppFuture from one task to another

### Concurrent Tasks




### Dependent Tasks



# Parsl Demo

# DragonHPC

- Open source project developed by HPE; check out recent PEARC tutorial
- Python API is multi-node extension to Python multiprocessing (e.g. mp.Process, mp.Pool, …)
- Includes a data management layer in the form of a distributed dictionary that spans nodes
- Parallel process launching with fine-grained control of CPU/GPU affinity
- High-speed RDMA transport for off-node communication on Slingshot and Infiniband networks
- C/C++ API included, Fortran API in development; Grafana Dashboard

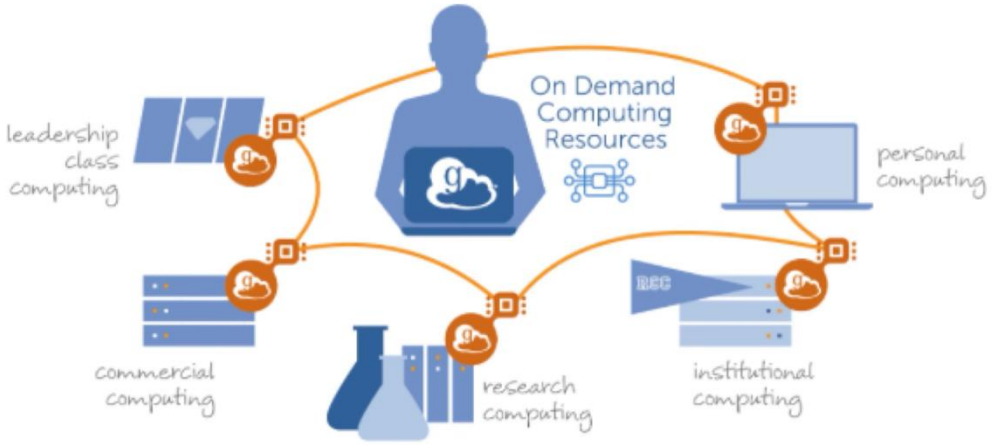
HPE

| Distributed Python (Jupyter Notebooks) and C/C++ support | Resilient training, inference, agents esp. PyTorch | Easy-to-use and HPC-optimized data exchange between applications | Observability for AI/HPC workflows | AI/HPC workflows across systems and sites |
|---|---|---|---|---|
| Multiprocessing/C++ APIs | AI API | Data API | Telemetry API | Workflow API |
| Dragon Core API | | | | |

# Dragon Demo

# Globus Compute

- Launch applications remotely from a laptop, lab workstation, other HPC machine, etc.
- “fire-and-forget” model of function execution
- Built on top of Parsl, similar configuration, also uses python futures
- Globus Compute functions can be integrated with data transfers with Globus Flows
- Requires a Compute Endpoint on the target machine
- ALCF supports 2 facility “multiuser” endpoints on Polaris and Crux
- Users can create their own endpoints on Aurora or other systems from uan/esn/login nodes
- Well suited for use with agent harnesses
- ALCF Docs: https://docs.alcf.anl.gov/services/globus-compute/
- Polaris Endpoint: https://app.globus.org/compute/endpoints/9a947ba5-f537-4681-acf3-cc66485aadec



# Globus Compute Demo

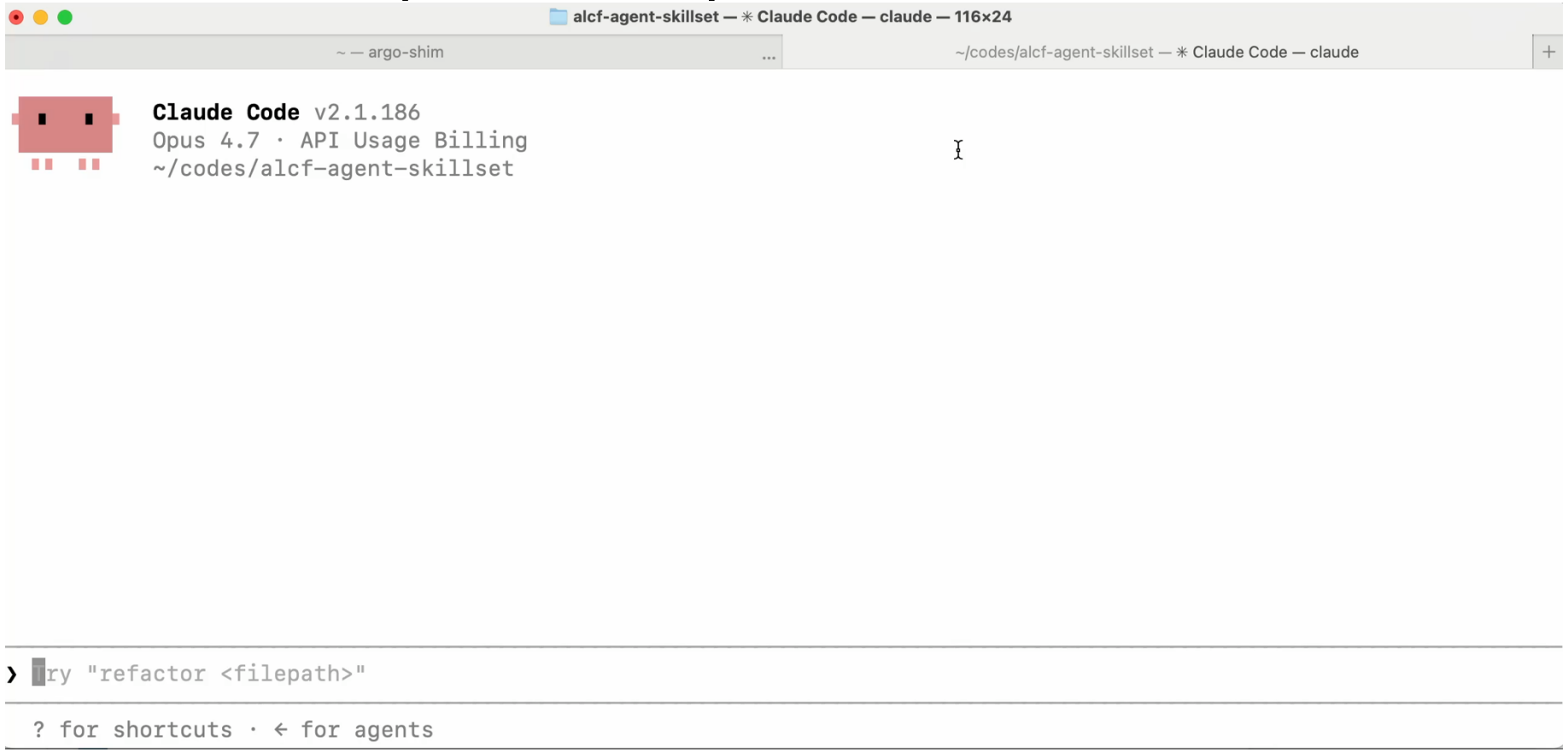
# IRI (Integrated Research Infrastructure) API


- IRI API developed collaboratively between NERSC, OLCF, ALCF & ESNet to create a unified API for remote job submission & management, filesystem queries, and status queries
- IRI API does not constitute a workflow tool by itself, but can be used by workflows to execute work on ASCR HPC systems
- ALCF IRI API open to all users of Polaris and Crux, Aurora coming soon
- Perlmutter at NERSC and Odo at OLCF by request
- ALCF IRI API uses Globus tokens for authentication
- API can be accessed with REST commands, either with cURL or python
- Well suited for use with agent harnesses
- REST API Swagger Page: https://api.alcf.anl.gov
- ALCF Docs: https://docs.alcf.anl.gov/services/iri-api/

# IRI API Demo

# API Skills for Agents

## Globus Compute Example




ALCF Skills: https://github.com/argonne-lcf/alcf-agent-skillset

# Conclusions

- There are many workflow tools out there!
- We covered a few options that are suitable for scaling workloads on HPC systems
- We've seen examples for executing remote work through facility supported APIs (Globus Compute & IRI)
- We will hear more today about the following topics:
  - How workflow tools can couple AI/ML with other applications
  - How workflows can run agents and be run by agents
  - More types of services being offered along side HPC systems that can integrate with workflows (e.g. the ALCF Inference Service)

# ARGONNE TRAINING PROGRAM ON EXTREME-SCALE COMPUTING

Produced by Argonne National Laboratory, a U.S. Department of Energy Laboratory managed by UChicagoArgonne, LLC under contract DE-AC02-06CH11357.

Special thanks to the National Energy Research Scientific Computing Center (NERSC) and Oak Ridge Leadership Computing Facility (OLCF) for the use of their resources during the training event.