Towards Diversified Exascale NWP Workflows

Christopher W. Harrop, Stefan F. Gary, Alvaro Vidal Torreira, Christina Holt, Naureen Bharwani, Venita Hagerty, Isidora Jankov, Kyle Chard, Michael Wilde

ECMWF Workshop on HPC in Meteorology - Oct 10, 2023, Bologna, Italy

Acknowledgements

This work is funded by the NOAA Software Engineering for Novel Architectures (SENA) Program

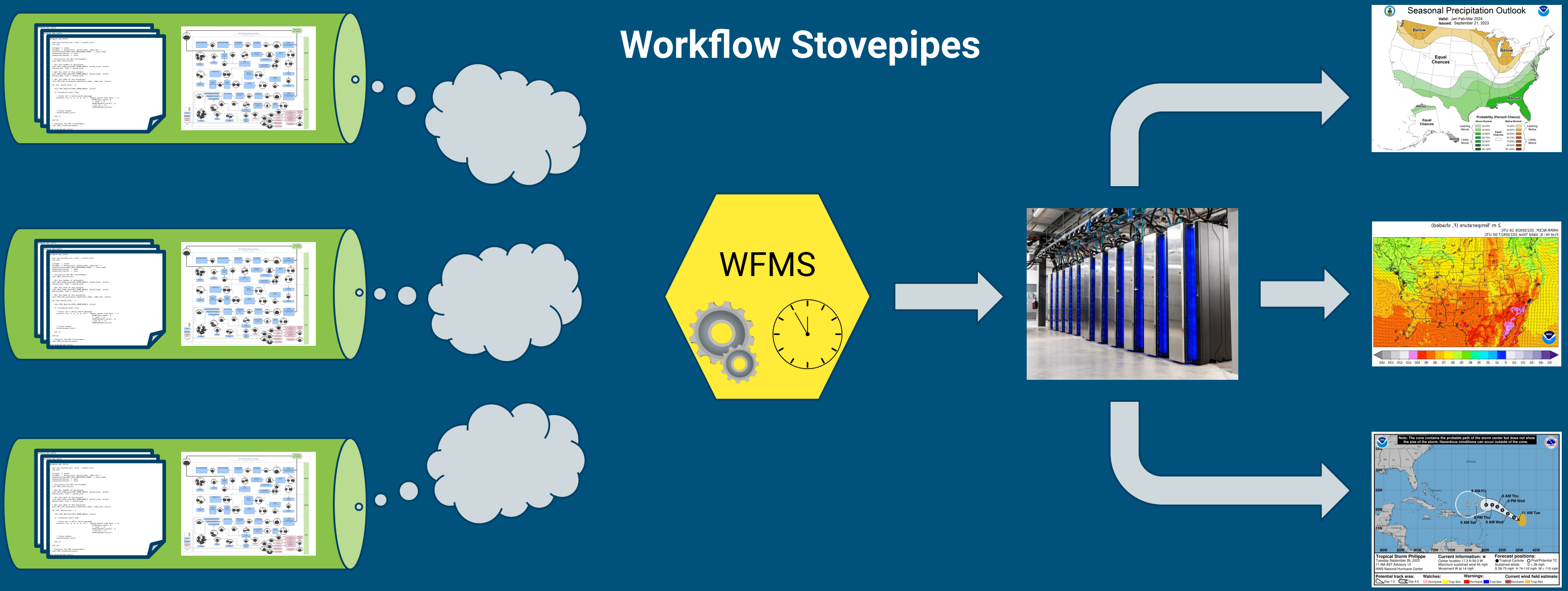
In collaboration with U.S. Department of Energy Exascale Computing Project & Parallel Works (funded by DOE Small Business Innovation Research)

Supported by NOAA Research and Development HPC (RDHPCS) Program

Special thanks to: Todd Munson, Eric Schnepp, James Corbett, Jim Garlick

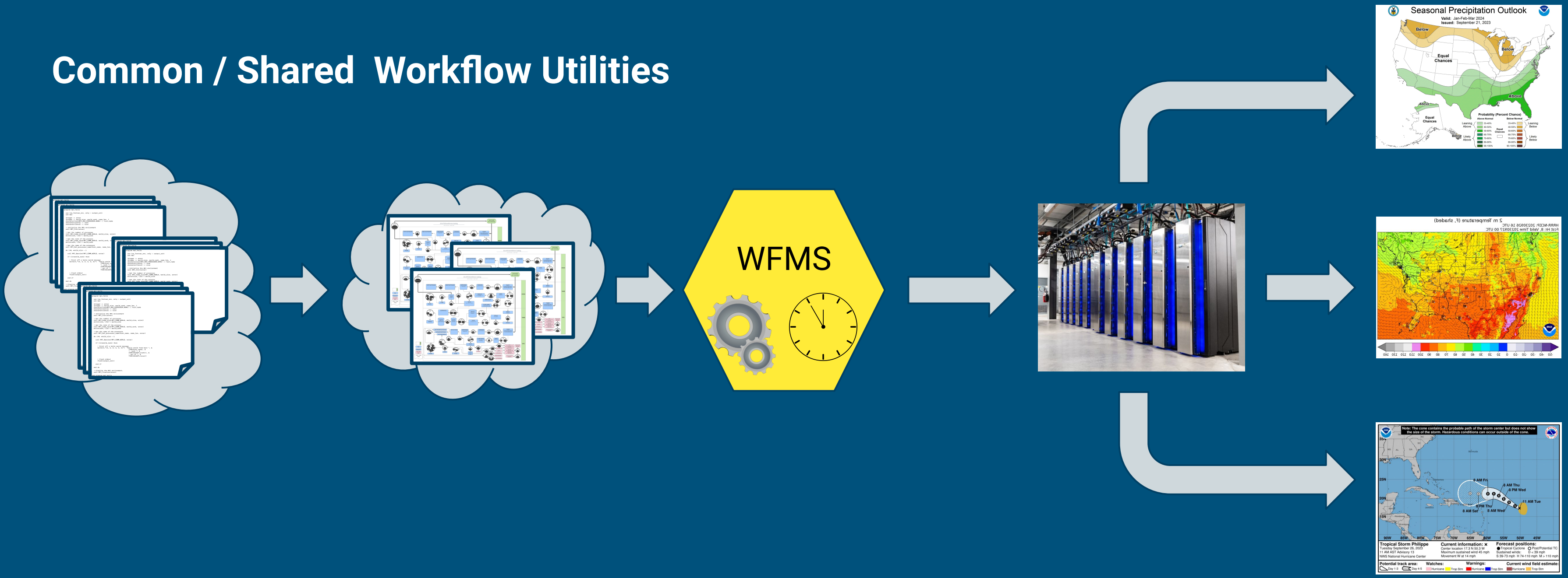
Contents

The Dark Ages

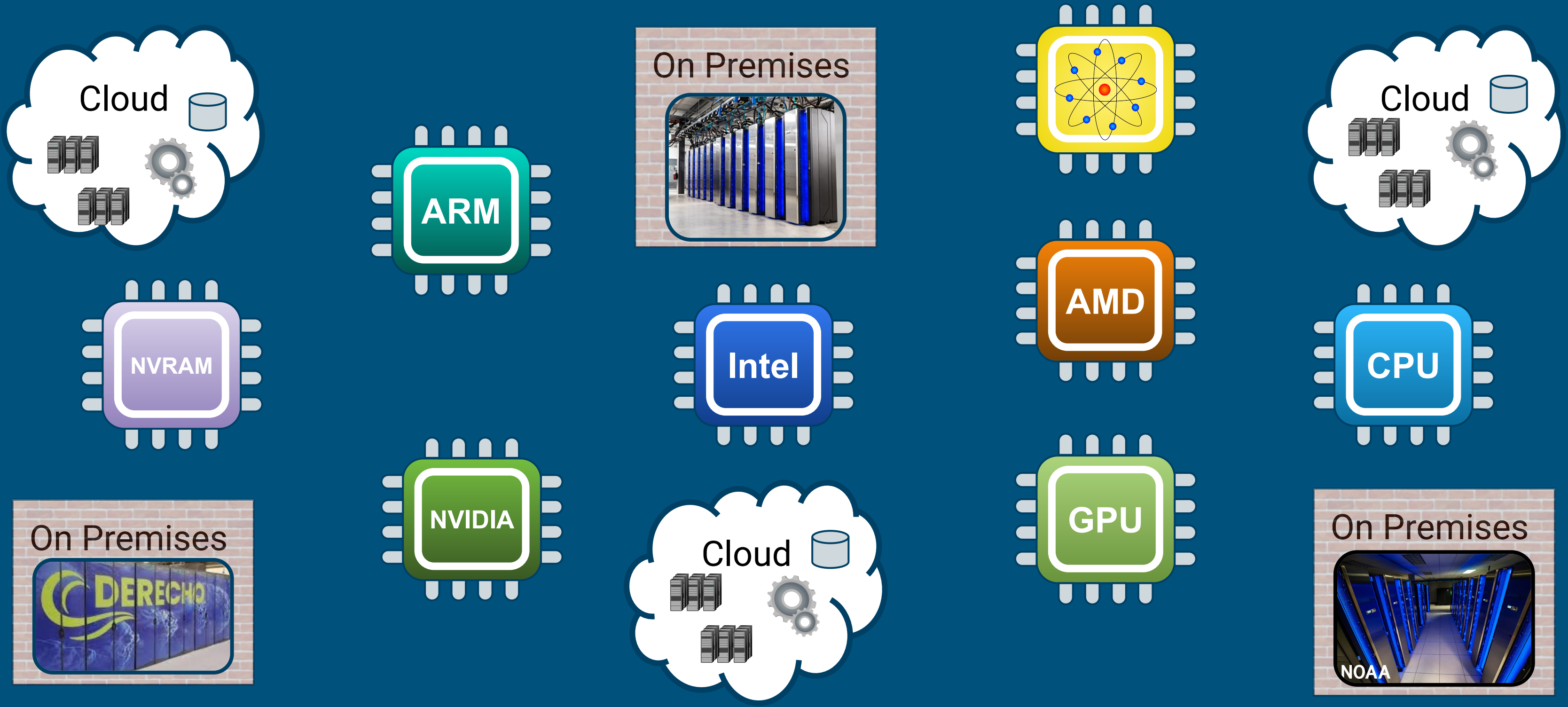


The (Unification) Enlightenment

Common / Shared Workflow Utilities



A Diversification Disruption



A Diversification Disruption

- The HPC landscape is changing
  - Diversification of hardware architectures
  - Diversification of programming models
  - Diversification of computational methods
  - High Throughput Computing (HTC) growing in importance
- Applications are changing
  - Diversification of components and services
  - Scaling up and out for capacity and access to diverse resources
  - Core services may be distributed or only available in certain places
  - Collaboration and sharing of data and products

A Tale of Two Workflow Regimes

R2O Pipeline

- Transition NWP Research to Operations
- Run the Operational system in Research environment
- Conform to requirements at the bottom of the “funnel”
- Use Operational workflow to make incremental forecast quality improvements

Features

- Operational mission is difficult - many modeling systems
- “Funnel” requirements evolve very slowly and meticulously
- Extremely risk averse
- Workflows are a means to an end – A tool in the toolbox

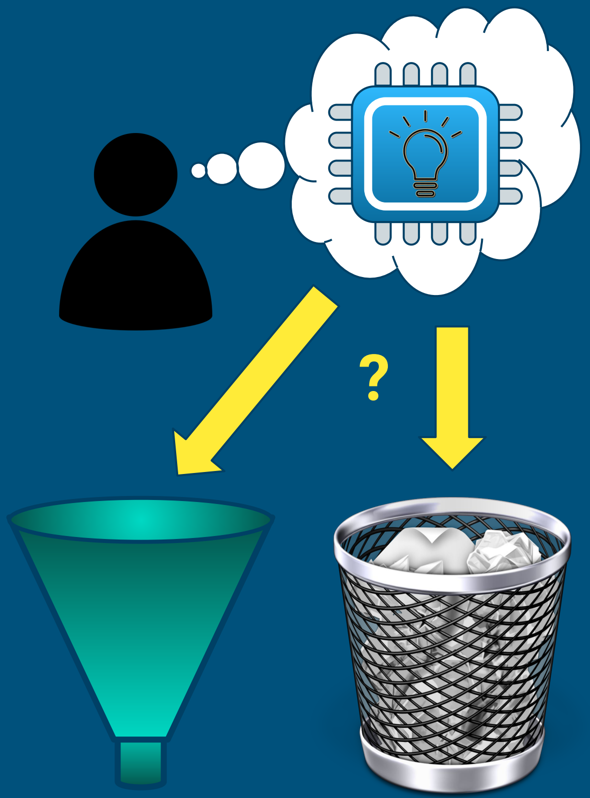
A Tale of Two Workflow Regimes

Long Range HPC (Exascale) Research

- Test and explore viability of emerging technologies
- Adapt the Operational system to use new technologies
- R2O “funnel” requirements lack the necessary agility & flexibility
- Develop workflows that apply HPC advancements to NWP

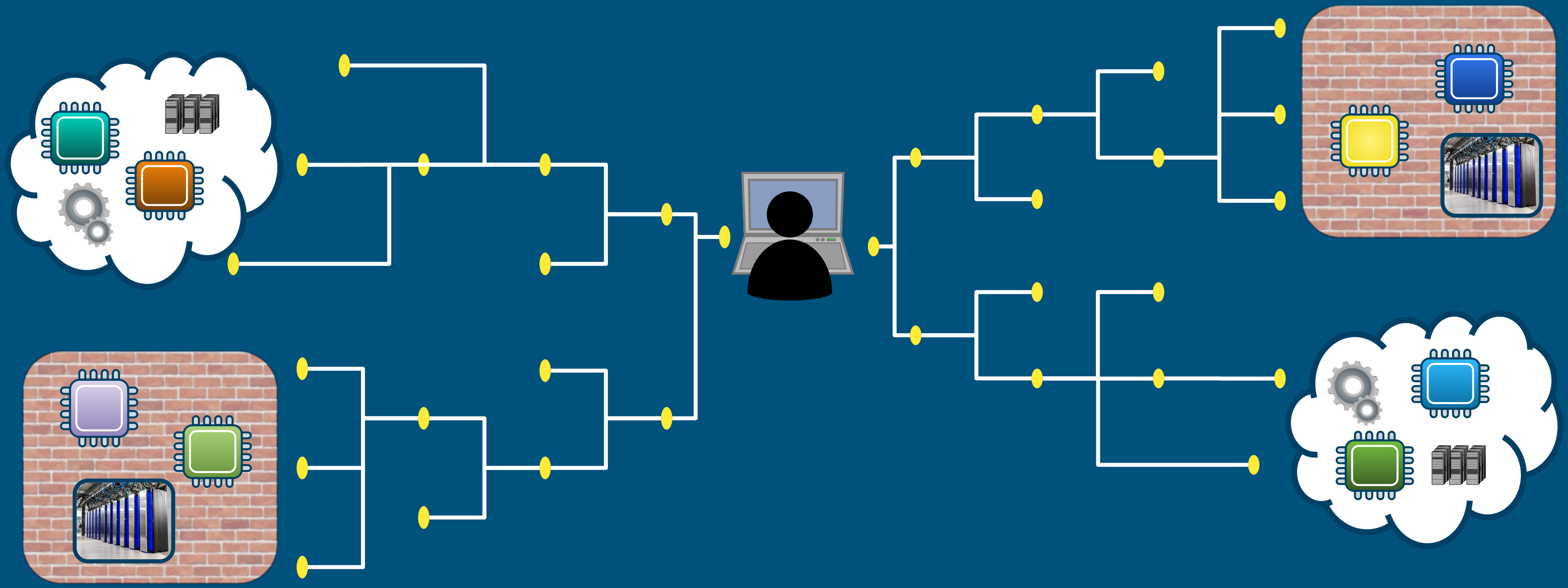
Features

- Diversification of HPC presents many challenges
- “Funnel” requirements inhibit “out-of-the-box” exploration
- Accept risk of using tools not currently available in Operations
- Diversified workflows are the product of the research



Our Vision

Diversified distributed workflows to enable NWP HPC Research



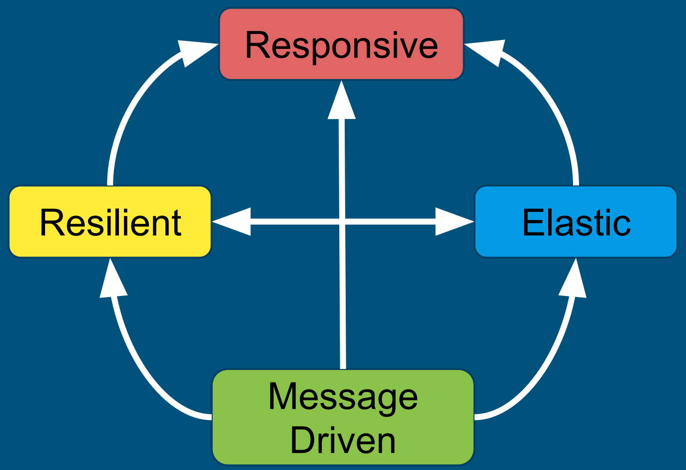
Guiding Principles

- Seamlessly distributed NWP application workflows
- Use existing, hardened, exascale-ready, tools
- Use UFS Unified Workflow building blocks
- No privileged access required for installation or use
- Configuration fully decoupled from execution
- “Let it fail” design for service lifecycle management

Inspiration From The Reactive Manifesto

Properties of reliable distributed applications

- **Responsive** - Consistent, rapid, response times to user requests
- **Resilient** - Maintain responsiveness during failure with isolation, replication, and delegation of failure recovery
- **Elastic** - Stay responsive under varying workloads by scaling resources and designing without points of contention
- **Message Driven** - Asynchronous, location transparent, messaging, between loosely coupled, isolated, components



Which Tools Should We Use?

A Scientific Workflow Management Community Challenge

Massive proliferation of bespoke workflow systems

ExaWorks

Democratize access to hardened, composable HPC workflow tools

- PSI/J
- Flux
- Parsl
- RADICAL
- Swift/T

Support for MPI

Portable Python-based workflow expression with multiple executors for different back-ends

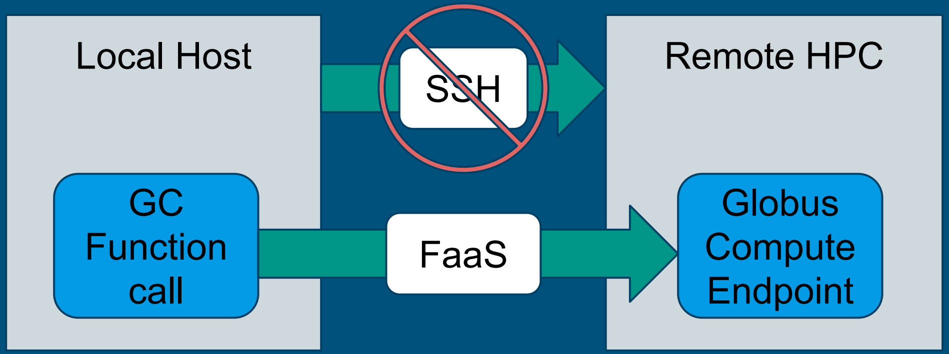
Which Tools Should We Use?

A Distributed Computing Challenge

Reliable management of secure communications between remote systems

Globus Compute

Secure function as a service (FaaS) for remote, high performance, “fire and forget” execution



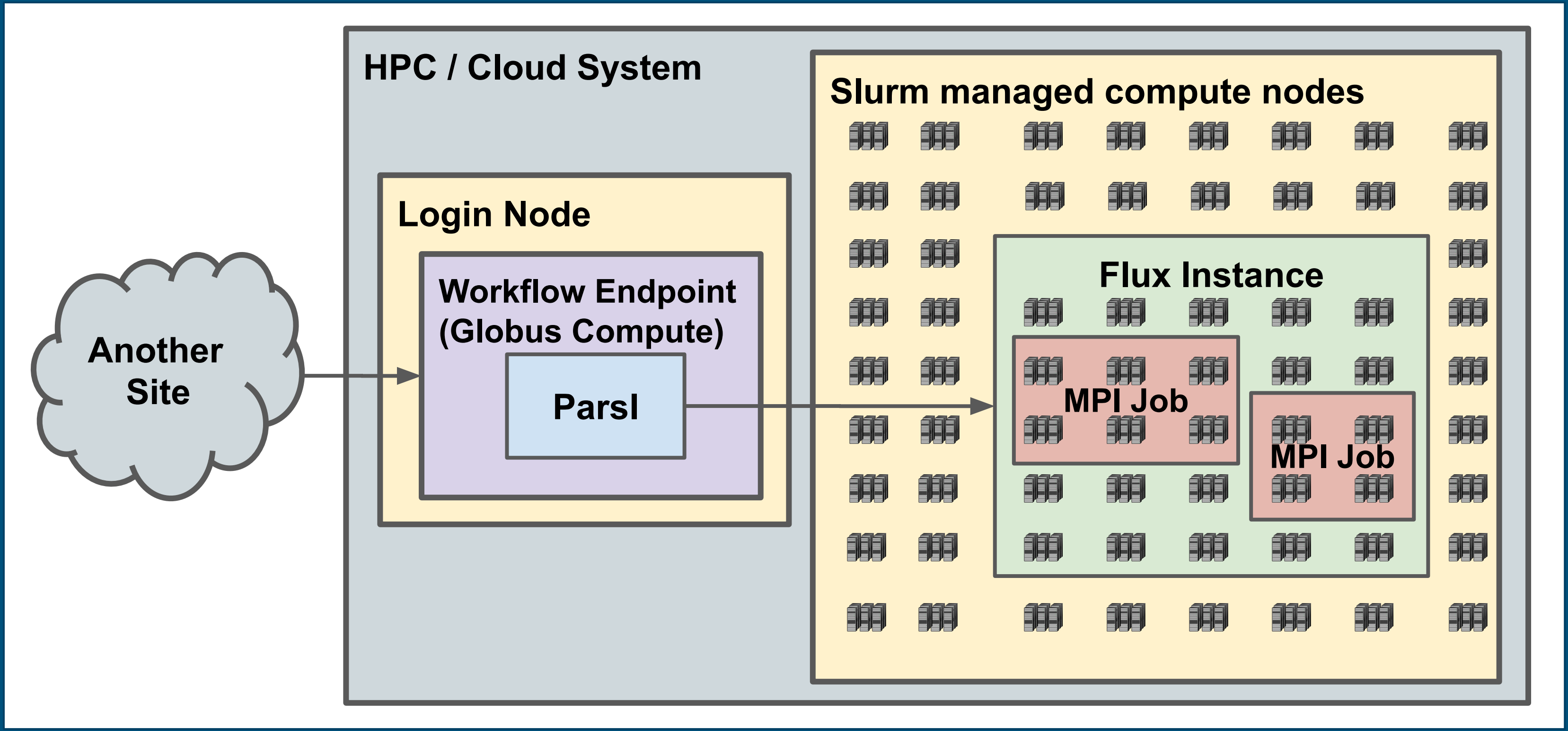
More Challenges

- Should workflows be expressed as programs or as configurations?
  - Python or YAML?
- What about the data?
  - How and when should we move it?
  - Flexible slice and dice - ECMWF Polytope data cube access?
  - Egress and storage costs
  - Globus Flows for multi-hop transfers?

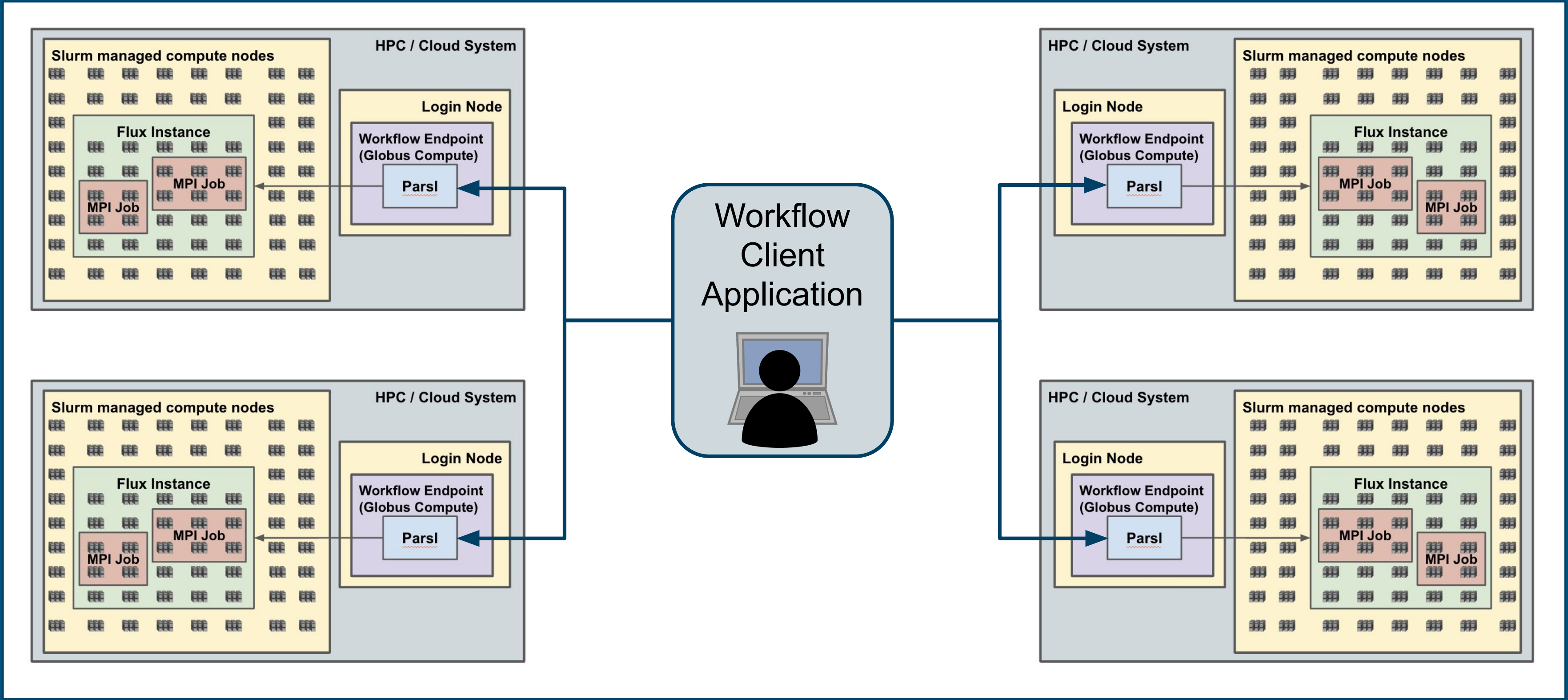
More Challenges

- How do we monitor and steer distributed workflows to diagnose problems?
  - If a workflow is a Python program, how do you interrogate and control it?
  - EPMT: The GFDL Experiment Performance Metrics Tracking Infrastructure
- How do we test while developing distributed workflow capabilities
  - Requires large, complex supporting software stacks
  - CI / CD using containerized Slurm clusters
  - Globus client credentials with GitHub secrets

The Current Architecture



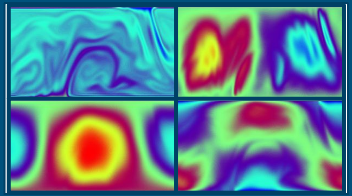
The Current Architecture

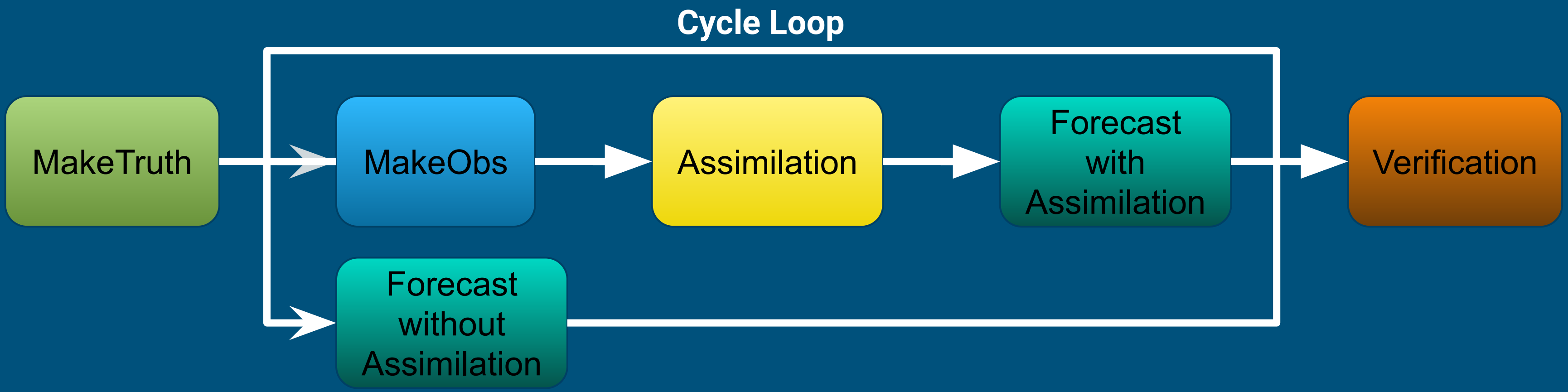


The Test Application

Cycled Data Assimilation with JEDI's Quasi-Geostrophic (QG) Model

- Start with 3D/4D variational methods
- Progress to fully distributed 3D/4D EnsVar ensembles





The Test Application

```
# Define "truth" forecast task
@bash_app
def truth_run_app(jedi_path, stdout=None, stderr=None, parsl_resource_specification={}):
    return '''
    export JEDI_PATH={}
    . $JEDI_PATH/bin/setupenv-hercules.sh
    unset I_MPI_PMI_LIBRARY
    $JEDI_PATH/bin/qg_forecast.x $JEDI_PATH/yaml/truth.yaml
    '''.format(jedi_path)
```

```
# Run the truth forecast
truth = truth_run_app(jedi_path='/path/to/JEDI',
                      stdout=os.path.join(jedi_path, 'truth.out'),
                      stderr=os.path.join(jedi_path, 'truth.err'),
                      parsl_resource_specification={"num_tasks": 1, "num_nodes": 1}
                      ).result()
```

Summary Remarks

- We have a vision, and many questions, but do not yet have all the answers
- We are testing Parsl + Flux + Globus Compute
  - Parsl → High throughput computing and powerful programming interface
  - Flux → MPI-aware scheduling within Parsl workflows
  - Globus Compute → Function as a service (FaaS) for secure distributed execution
- We are starting small for testing and exploration purposes
  - Simple JEDI QG data assimilation workflow
- We are making use of UFS Unified Workflow development where possible
- Demonstration with a “real” model once foundational pieces are settled

Questions / Discussion