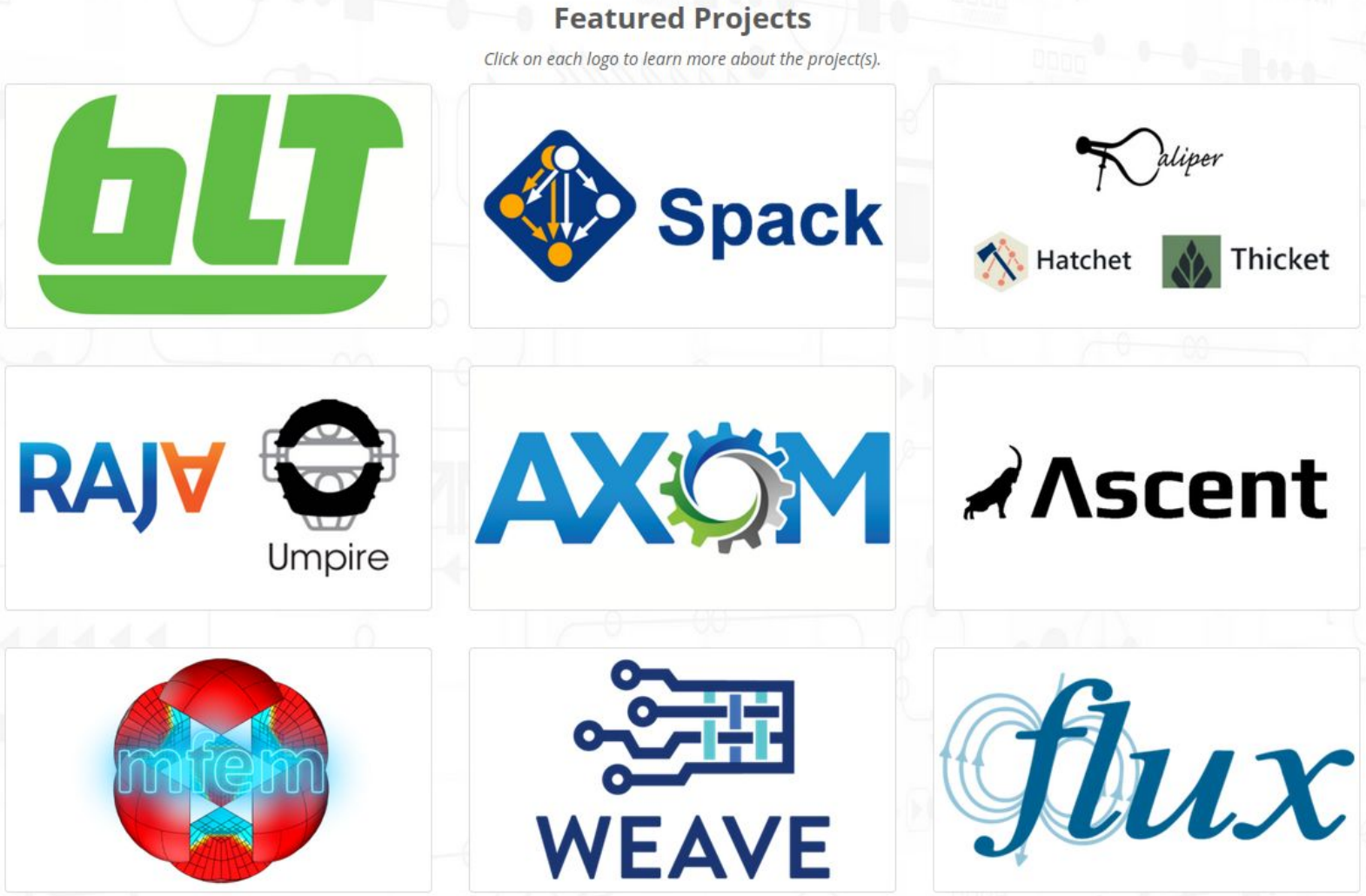
HIGH PERFORMANCE COMPUTING
INNOVATION CENTER

Welcome to the HPCIC AWS Tutorial Series!

Go to:
https://hpcic.llnl.gov/tutorials/2024-hpc-tutorials
to learn more about our other tutorials and documentation!



The Flux Framework Tutorial

HPCIC AWS Tutorial Series

August 29, 2024

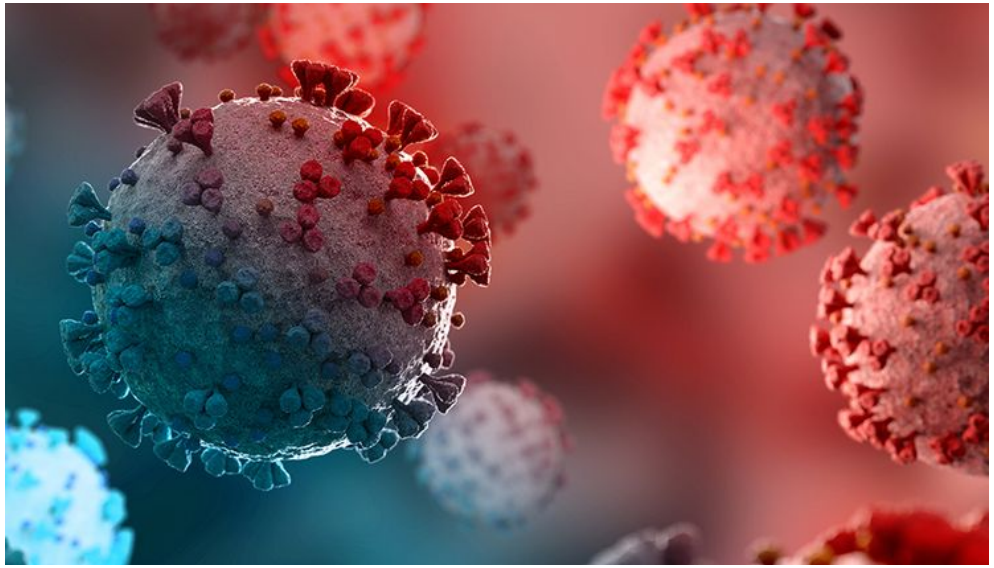
Al Chu, James Corbett, Ryan Day, Jim Garlick, Mark Grondona, William Hobbs, Aniruddha Marathe, **Dan Milroy**, Zeke Morton, Chris Moussa, Tapasya Patki, Barry Rountree, Abhik Sarkar, Tom Scogland, **Vanessa Sochat**, Becky Springmeyer, Jae-Seung Yeom

LLNL-PRES-868703
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Why do we need workload managers?

We need to run complex scientific workflows and simulations that use heterogeneous resources (machine architectures, accelerators / GPUs, and devices) for the advancement of science.



What is a workload manager?

A workload manager receives units of work, for example scientific simulations, and ensures that they are planned, assigned, and executed on (typically) many computers.

What do you mean "many computers" ?

Scientists use *supercomputers* to run these complex workflows - many thousands of machines linked together. This introduces the problem of scheduling and job management.

Scheduling

Which jobs get to run, and where?

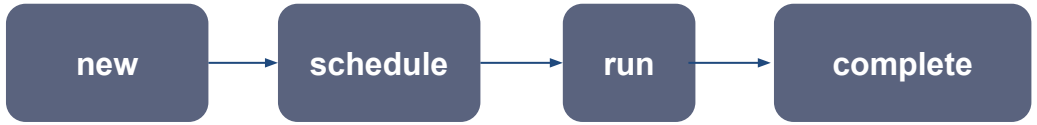
I can pencil you in for nodes **compute[1-10]** with 2 GPU in about an hour.

Scheduling

Which jobs get to run, and where?

Job Management

What is the lifecycle of my job?



What is Flux Framework?

Flux Framework "Flux" is a workload manager developed by Lawrence Livermore National Laboratory that combines hierarchical job management with graph-based scheduling.

Did you know!

Flux has 15+ contributors, including principal engineers behind Slurm, and developers with expertise in container technologies, scheduling, and cloud computing.

How did Flux come to be?

Project Established

Initial project funding
proposal and staffing

How did Flux come to be?

Project Established

Initial project funding proposal and staffing

Design

Preliminary design review

How did Flux come to be?

Project Established
Initial project funding proposal and staffing

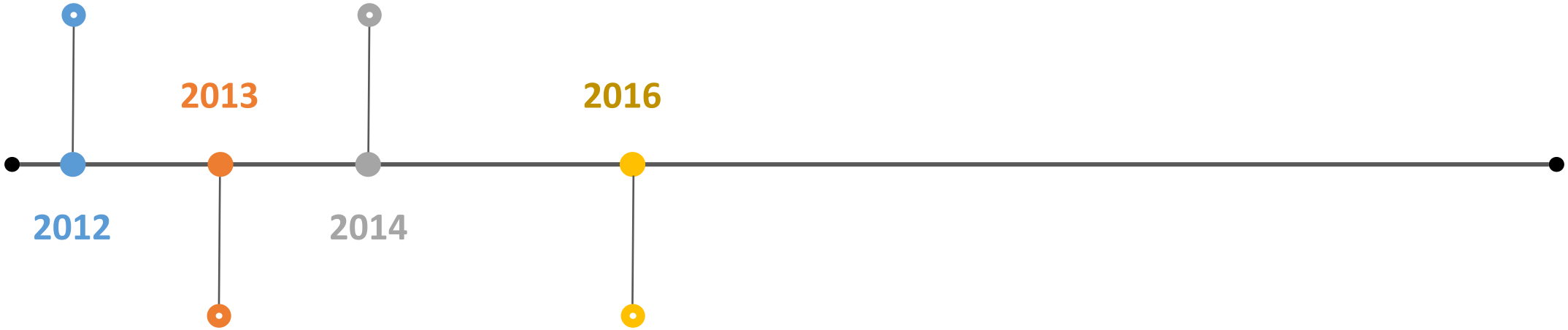
Instance Launch
Prototype Flux instance that could be launched by Slurm

Design
Preliminary design review

How did Flux come to be?

Project Established
Initial project funding proposal and staffing

Instance Launch
Prototype Flux instance that could be launched by Slurm



Design
Preliminary design review

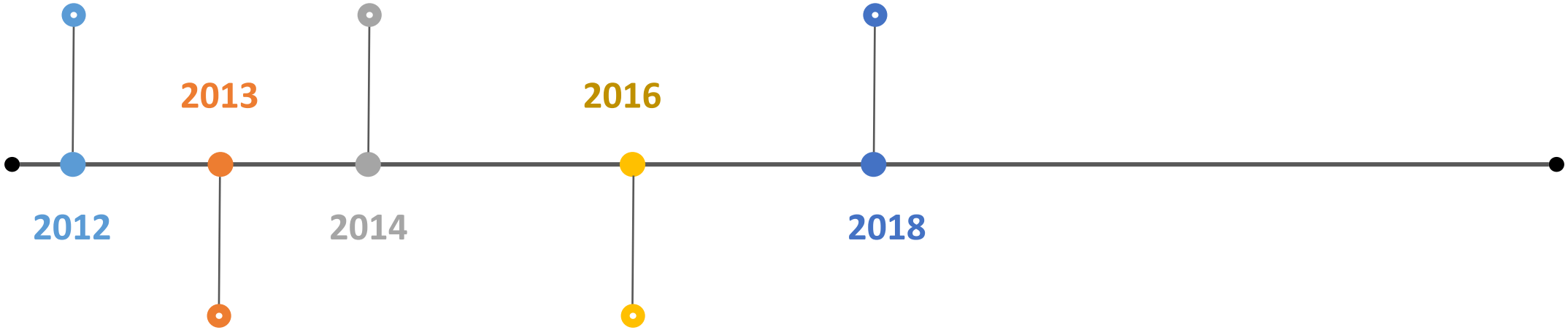
Flux Core 0.1.0
First public release of Flux Core

How did Flux come to be?

Project Established
Initial project funding proposal and staffing

Instance Launch
Prototype Flux instance that could be launched by Slurm

Science
Flux used by Pilot2 cancer moonshot ensemble



Design
Preliminary design review

Flux Core 0.1.0
First public release of Flux Core

How did Flux come to be?

2012 — Project Established
Initial project funding proposal and staffing

2013 — Design
Preliminary design review

2014 — Instance Launch
Prototype Flux instance that could be launched by Slurm

2016 — Flux Core 0.1.0
First public release of Flux Core

2018 — Science
Flux used by Pilot2 cancer moonshot ensemble

2020 — System Instances
Release of Flux Core with system instance support

How did Flux come to be?

2012 — Project Established
Initial project funding proposal and staffing

2013 — Design
Preliminary design review

2014 — Instance Launch
Prototype Flux instance that could be launched by Slurm

2016 — Flux Core 0.1.0
First public release of Flux Core

2018 — Science
Flux used by Pilot2 cancer moonshot ensemble

2020 — System Instances
Release of Flux Core with system instance support

2021 — MuMMI Workflow
Flux+MuMMI Integration used for RAS-RAF membrane dynamics ensemble

How did Flux come to be?

2012 — Project Established
Initial project funding proposal and staffing

2013 — Design
Preliminary design review

2014 — Instance Launch
Prototype Flux instance that could be launched by Slurm

2016 — Flux Core 0.1.0
First public release of Flux Core

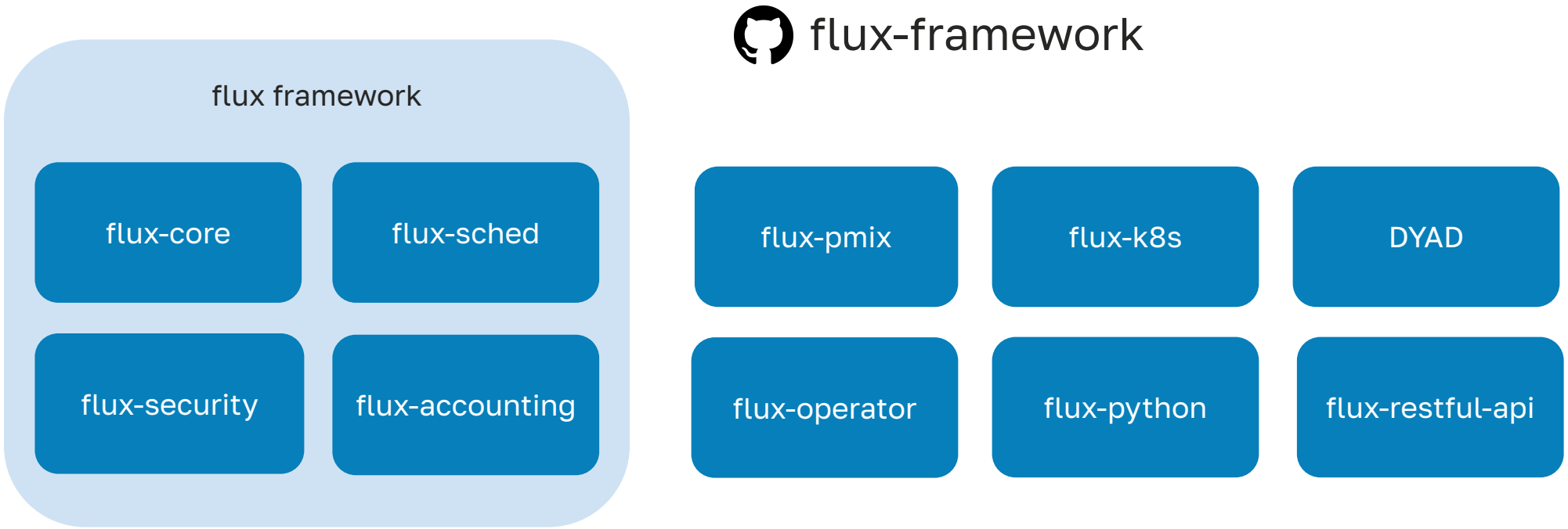
2018 — Science
Flux used by Pilot2 cancer moonshot ensemble

2020 — System Instances
Release of Flux Core with system instance support

2021 — MuMMI Workflow
Flux+MuMMI Integration used for RAS-RAF membrane dynamics ensemble

2023 — El Capitan
Flux system instance deployed on early El Capitan hardware

Flux has evolved into a suite of open source projects

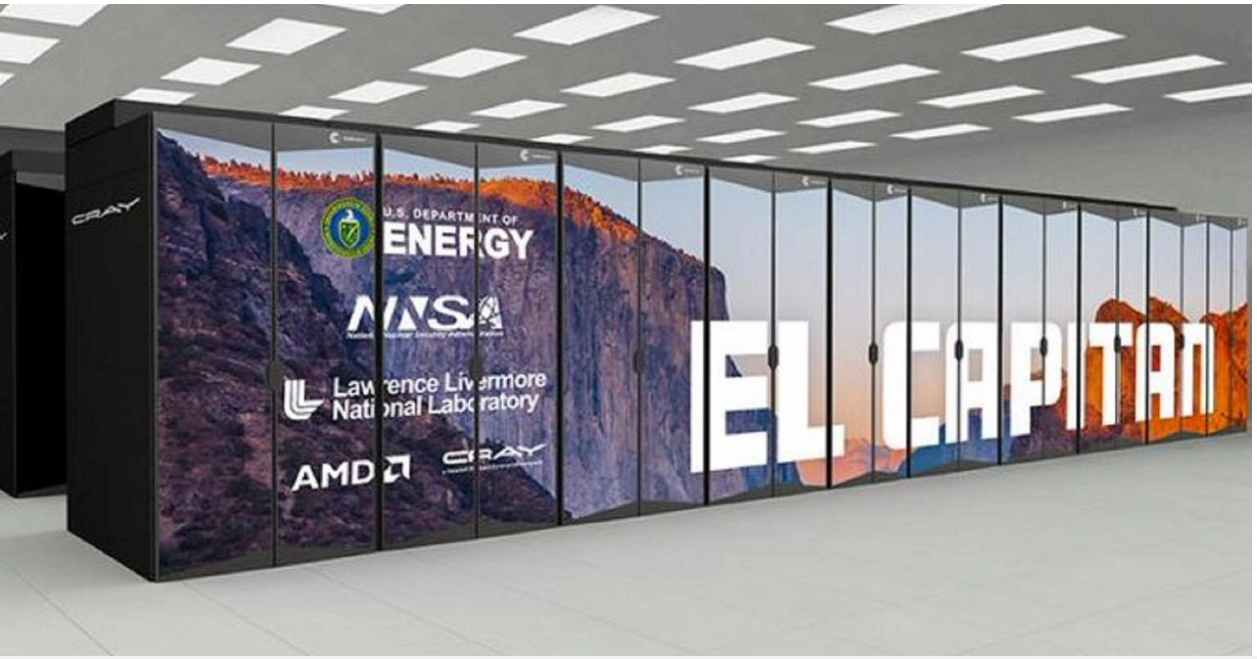


We will show some of these more in detail later in the presentation!

Flux is running as the system scheduler on LLNL production clusters and on El Capitan.

Rolling out on production systems, addressing user groups one at a time.

- El Capitan (pictured here)



Smaller clusters for user feedback (three are in the top 200 of the Top500)

- Tioga, Corona (pictured here), RZVernal, Tenaya
- Hetchy, Fluke, Elmerfudd

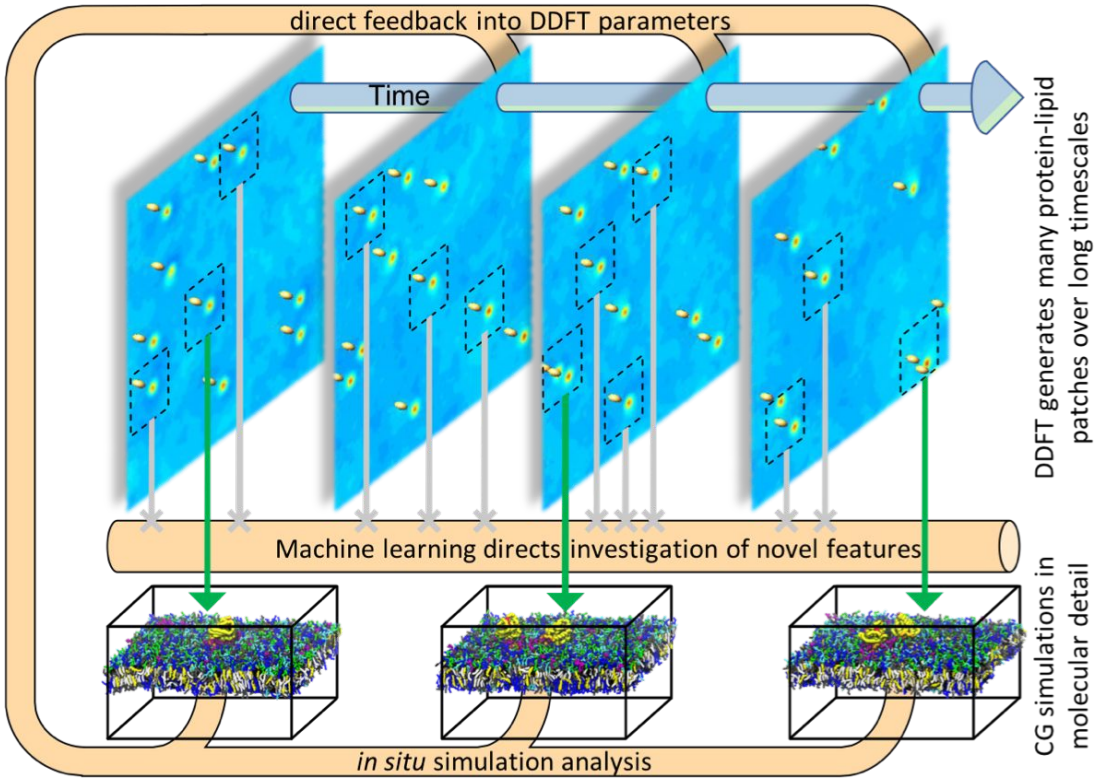
What is Flux Framework?

Flux Framework "Flux" is a workload manager developed by Lawrence Livermore National Laboratory that combines hierarchical job management with graph-based scheduling.

Did you know!

Flux has 15+ contributors, including principal engineers behind Slurm, and developers with expertise in container technologies, scheduling, and cloud computing.

Trends towards complex workflows, extreme resource heterogeneity, and converged computing render traditional workload managers increasingly ineffective.



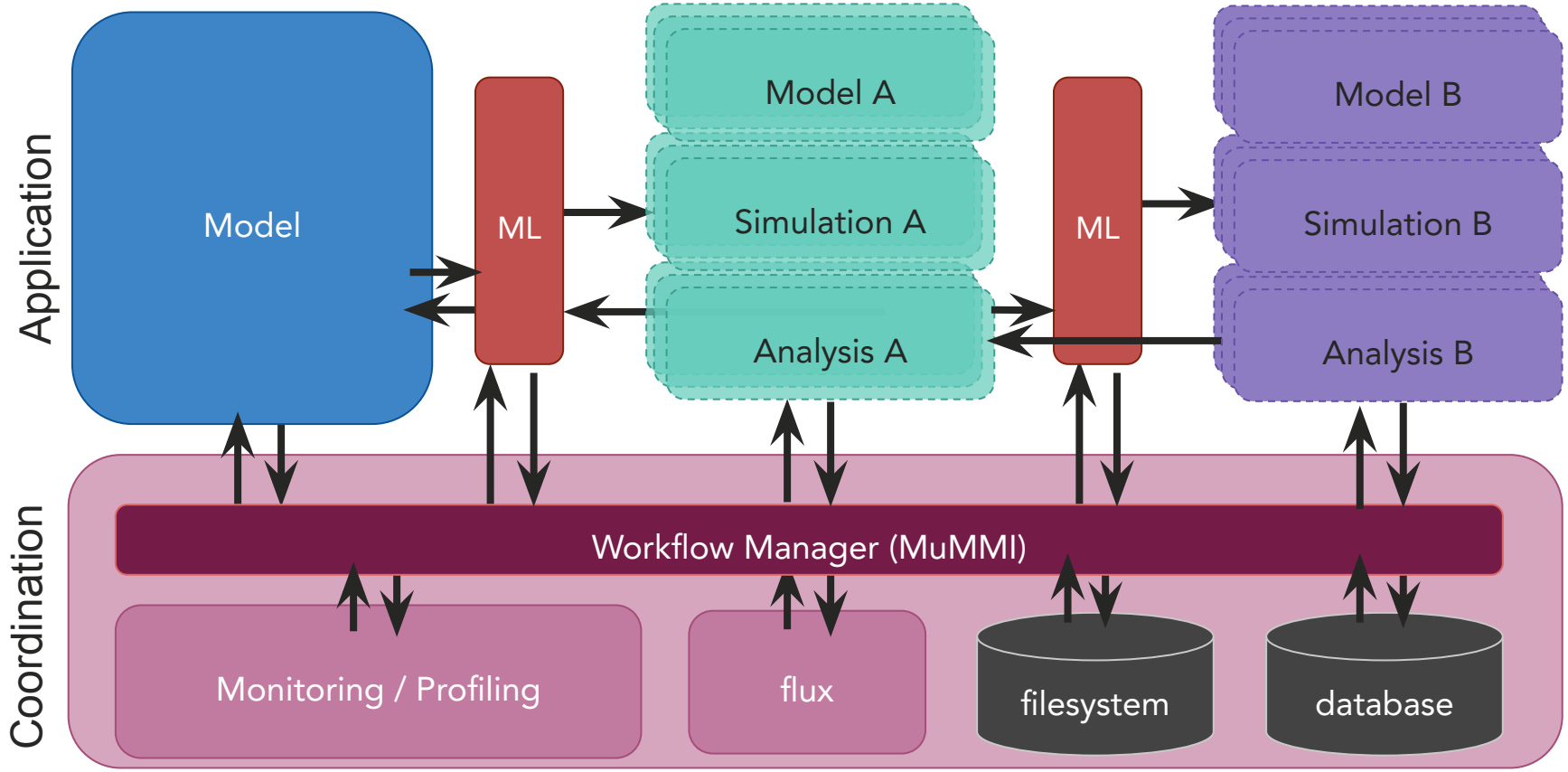
- Co-scheduling
- Job throughput
- Job communication/coordination
- Portability
- Extremely heterogeneous resources

Pre-exascale scientific workflows strain the capabilities of traditional HPC resource managers and schedulers.

Co-scheduling:
CG, analysis bound to cores nearest PCIe buses

Job comms/coordination:
36,000 concurrent tasks;
176,000 cores, 16,000 GPUs

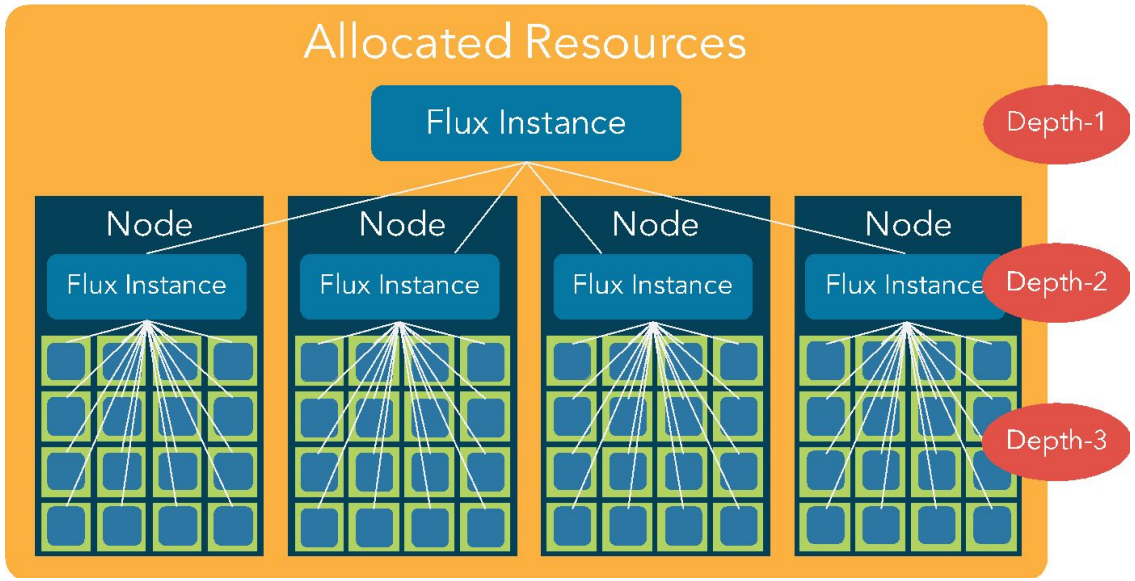
Portability:
adapt tasks to different schedulers/managers



MuMMI: SC'19 best paper, SC'21 paper

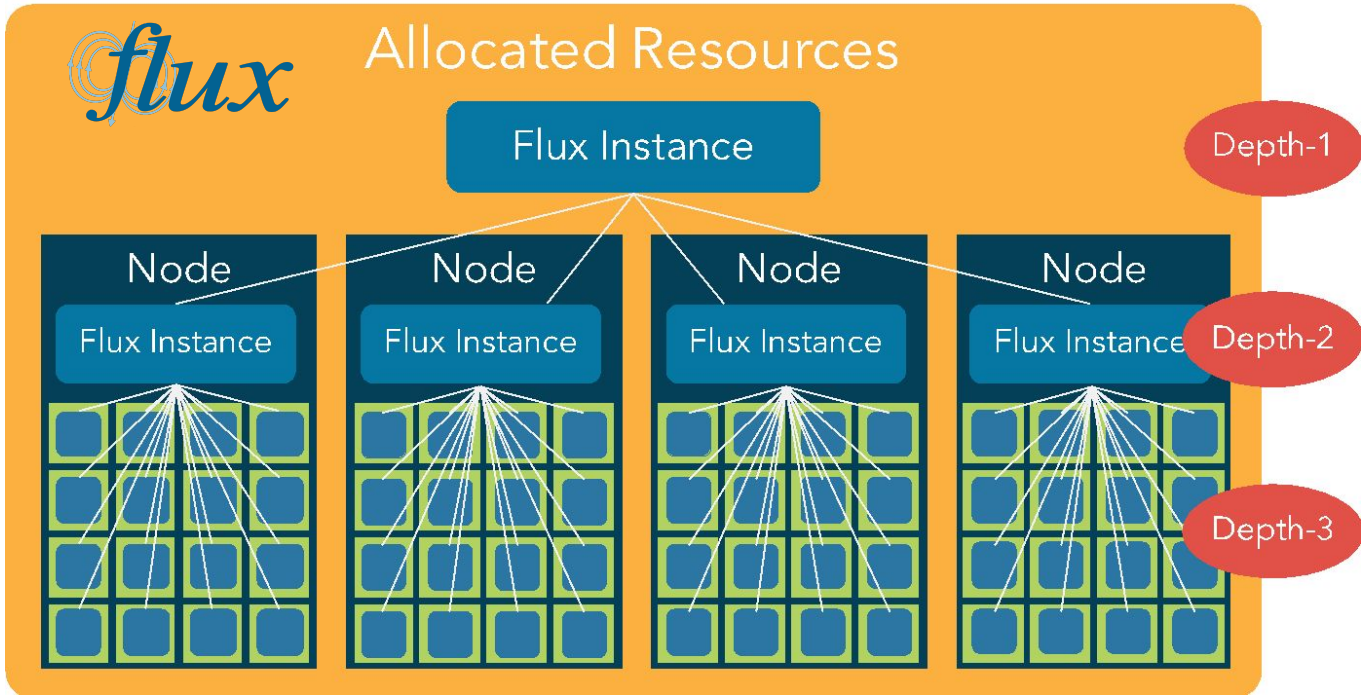
MPI-based simulation with in-situ analysis plus AI/ML

Flux hierarchical management addresses exascale and converged computing challenges.



- Co-scheduling
- Job throughput
- Job communication/coordination
- Portability
- Extremely heterogeneous resources

Hierarchical management addresses exascale and converged computing challenges.



"Fractal scheduling" mitigates centralized scheduler bottleneck

- handles high throughput
- job steps needn't hit central scheduler

Modular, hierarchical design

- Hierarchical resource management and scheduling (separate modules)
- Sub-manager with specialized scheduler
- Schedules cloud resources

Manages resources nearly anywhere

- Bare metal resources, virtual machines in the cloud, HPC resources in another workload manager, pods in Kubernetes
- Workflows only need to program to Flux

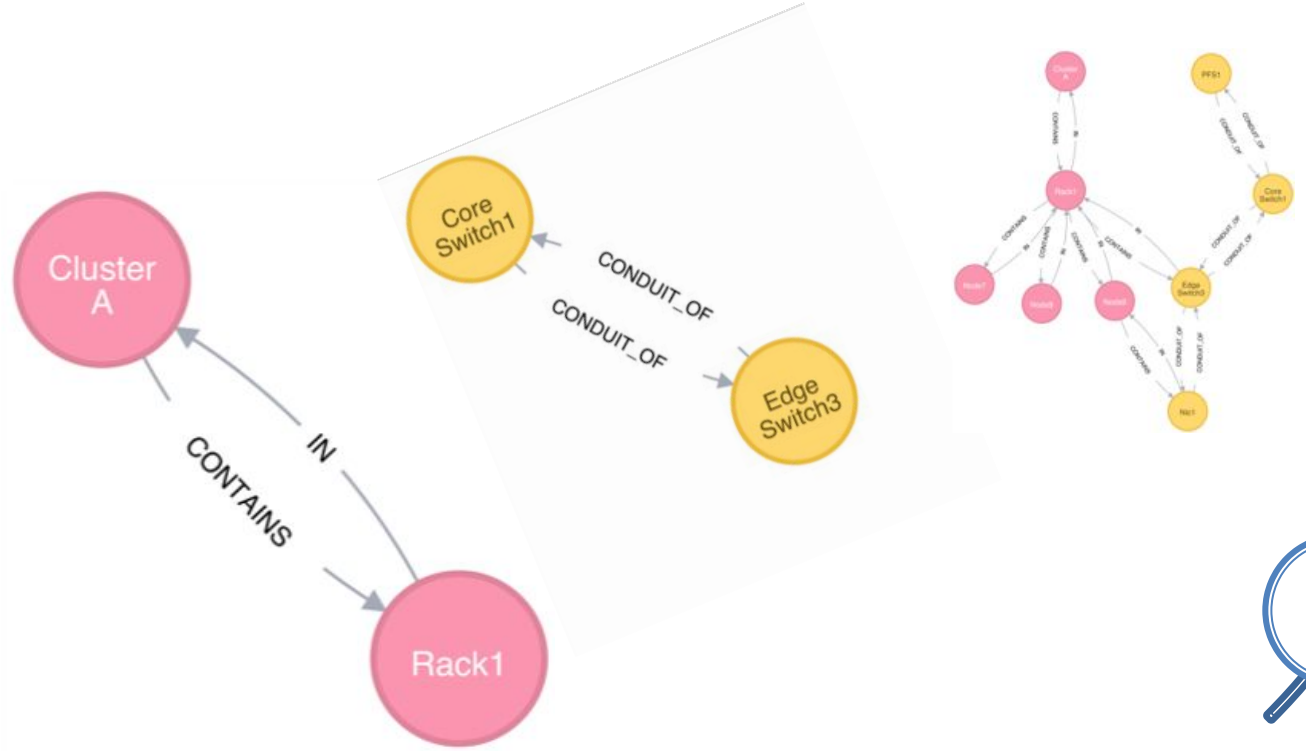
Rich, well-defined interfaces

- Facilitate communications and coordination among tasks within a workflow
- CLI, Python, C, C++, Rust, *Go*, etc.

Graph-based scheduling means modeling cluster resources as a graph, from the nodes down to the cores.

I know the detailed topology.

"These GPUs need to communicate? Let's schedule them to be physically close together!"

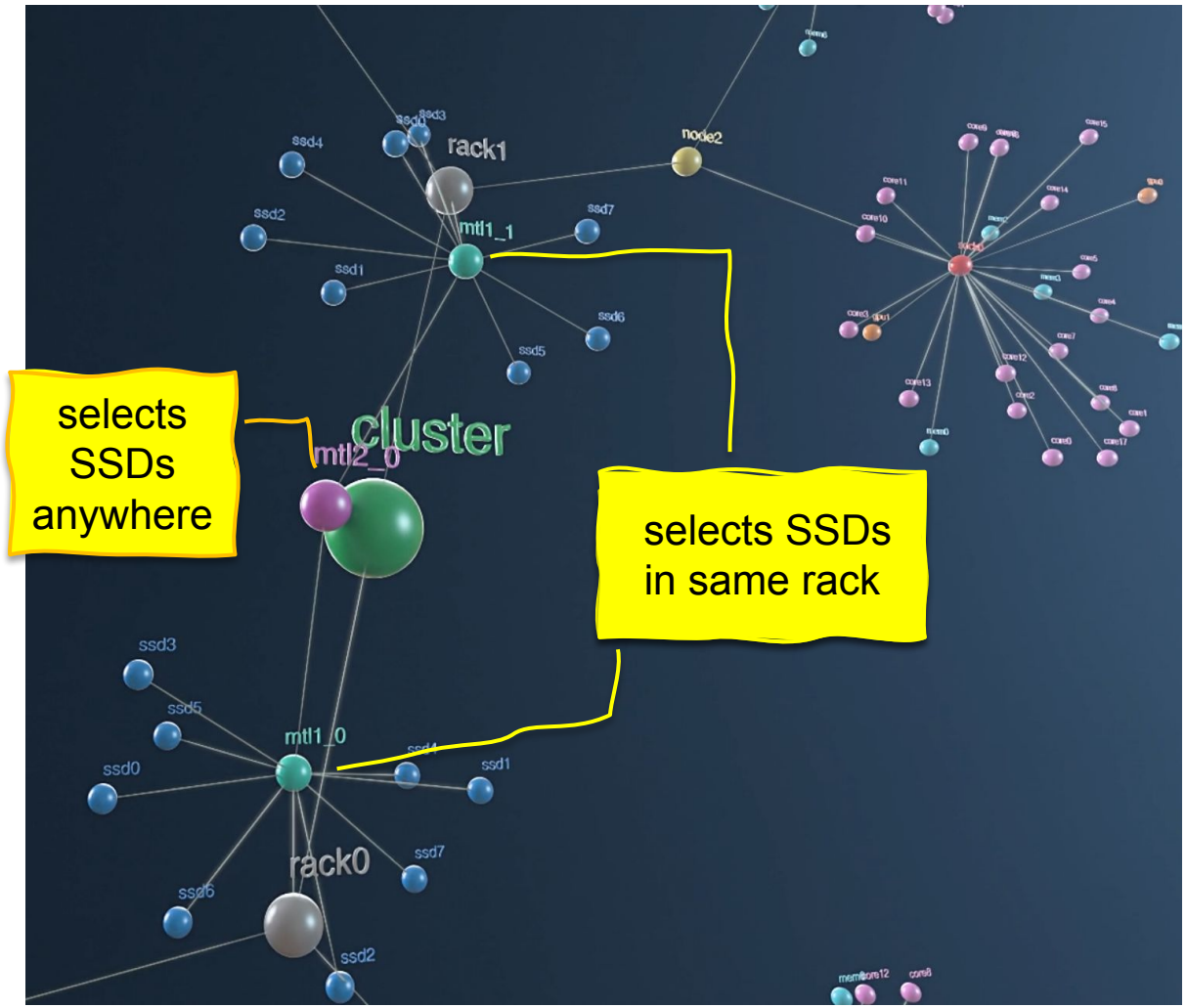


- Co-scheduling
- Job throughput
- Job communication/coordination
- Portability
- Extremely heterogeneous resources

The rabbits of El Cap present a fearsome scheduling problem.

- Multi-tiered storage features “rabbit” nodes
  - 16 NVMEs, direct PCIe to chassis compute nodes
- Dynamically configured as node-local storage or job-global
  - node-local via PCIe, global via network
  - single rabbit can serve both at once
  - scheduler must handle both
- Can be allocated independent of jobs
- Too difficult for traditional schedulers

Fluxion's directed-graph approach addresses the rabbit challenge and facilitates cloud integration.



- Fluxion schedules rack-local and global storage with no code change (caveats)
  - inefficient for scheduling same resource type multiple times
  - affects jobs requesting multiple rabbit allocations
  - known issue to be fixed before El Cap deployment
- Directed-graph representation enables scheduling, managing dynamic, cloud resources

Flux's graph-oriented jobspec allows for highly expressive resource requests.

```
flux run -N 3 -n 18 ./app
```

```
version: 1
resources:
  - type: cluster
    count: 1
    with:
      - type: rack
        count: 2
        with:
          - type: slot
            label: myslot
            count: 3
            with:
              - type: node
                count: 1
                with:
                  - type: socket
                    count: 2
                    with:
                      - type: core
                        count: 18

# a comment
attributes:
  system:
    duration: 3600
tasks:
  - command: app
    slot: myslot
    count:
      per_slot: 1
```

Flux's graph-oriented jobspec allows for highly expressive resource requests.

`flux run -N 3 -n 18 ./app` 

- Graph-oriented resource requirements
  - Express the resource requirements of a program to the scheduler
  - Express program attributes such as arguments, run time, and task layout, to be considered by the execution service
- cluster->racks[2]->slot[3]->node[1]->sockets[2]->core[18]

```
version: 1
resources:
  - type: cluster
    count: 1
    with:
      - type: rack
        count: 2
        with:
          - type: slot
            label: myslot
            count: 3
            with:
              - type: node
                count: 1
                with:
                  - type: socket
                    count: 2
                    with:
                      - type: core
                        count: 18

# a comment
attributes:
  system:
    duration: 3600
tasks:
  - command: app
    slot: myslot
    count:
      per_slot: 1
```

Flux's graph-oriented jobspec allows for highly expressive resource requests.

flux run -N 3 -n 18 ./app

- Graph-oriented resource requirements
  - Express the resource requirements of a program to the scheduler
  - Express program attributes such as arguments, run time, and task layout, to be considered by the execution service
- cluster->racks[2]->slot[3]->node[1]->sockets[2]->core[18]
- slot is the only non-physical resource type
  - Represent a schedulable place where program process or processes will be spawned and contained

```
version: 1
resources:
  - type: cluster
    count: 1
    with:
      - type: rack
        count: 2
        with:
          - type: slot
            label: myslot
            count: 3
            with:
              - type: node
                count: 1
                with:
                  - type: socket
                    count: 2
                    with:
                      - type: core
                        count: 18

# a comment
attributes:
  system:
    duration: 3600
tasks:
  - command: app
    slot: myslot
    count:
      per_slot: 1
```

Flux's graph-oriented jobspec allows for highly expressive resource requests.

flux run -N 3 -n 18 ./app

- Graph-oriented resource requirements
  - Express the resource requirements of a program to the scheduler
  - Express program attributes such as arguments, run time, and task layout, to be considered by the execution service
- cluster->racks[2]->slot[3]->node[1]->sockets[2]->core[18]
- slot is the only non-physical resource type
  - Represent a schedulable place where program process or processes will be spawned and contained
- Referenced from the tasks section

```
1  version: 1
2  resources:
3    - type: cluster
4      count: 1
5      with:
6        - type: rack
7          count: 2
8          with:
9            - type: slot
10             label: myslot
11             count: 3
12             with:
13               - type: node
14                 count: 1
15                 with:
16                   - type: socket
17                     count: 2
18                     with:
19                       - type: core
20                         count: 18
21
22 # a comment
23 attributes:
24   system:
25     duration: 3600
26 tasks:
27   - command: app
28     slot: myslot
29     count:
30       per_slot: 1
```

Next-generation, cross-cluster scientific workflows are demanding portability and cloud integration.

Complex workflows are integrating cloud technologies at LLNL *and beyond!* 🚀

- MPI-based simulation with in-situ analysis plus AI/ML (MuMMI)
- Scalable message broker couples MPI-based tasks (AHA MoleS)
- HPC simulation with AI/ML surrogates, orchestrated data (AMS)
- Orchestration of simulations and services

The what and why of movement to the cloud; HPC doesn't want to be left behind.

The cloud is an environment (public, private) that supports:

- Portability, reproducibility (e.g., containerization)
- Resiliency, efficiency (e.g., resource dynamism, elasticity, declarative management)
- Reduced complexity via automation (autoscaling, elasticity, declarative management)

Companies rent this environment; hugely profitable

- projected to $920B by 2025, 20% CAGR (20-25)[1] vs HPC: $40B by 2025, 20-25 CAGR 8%[2]

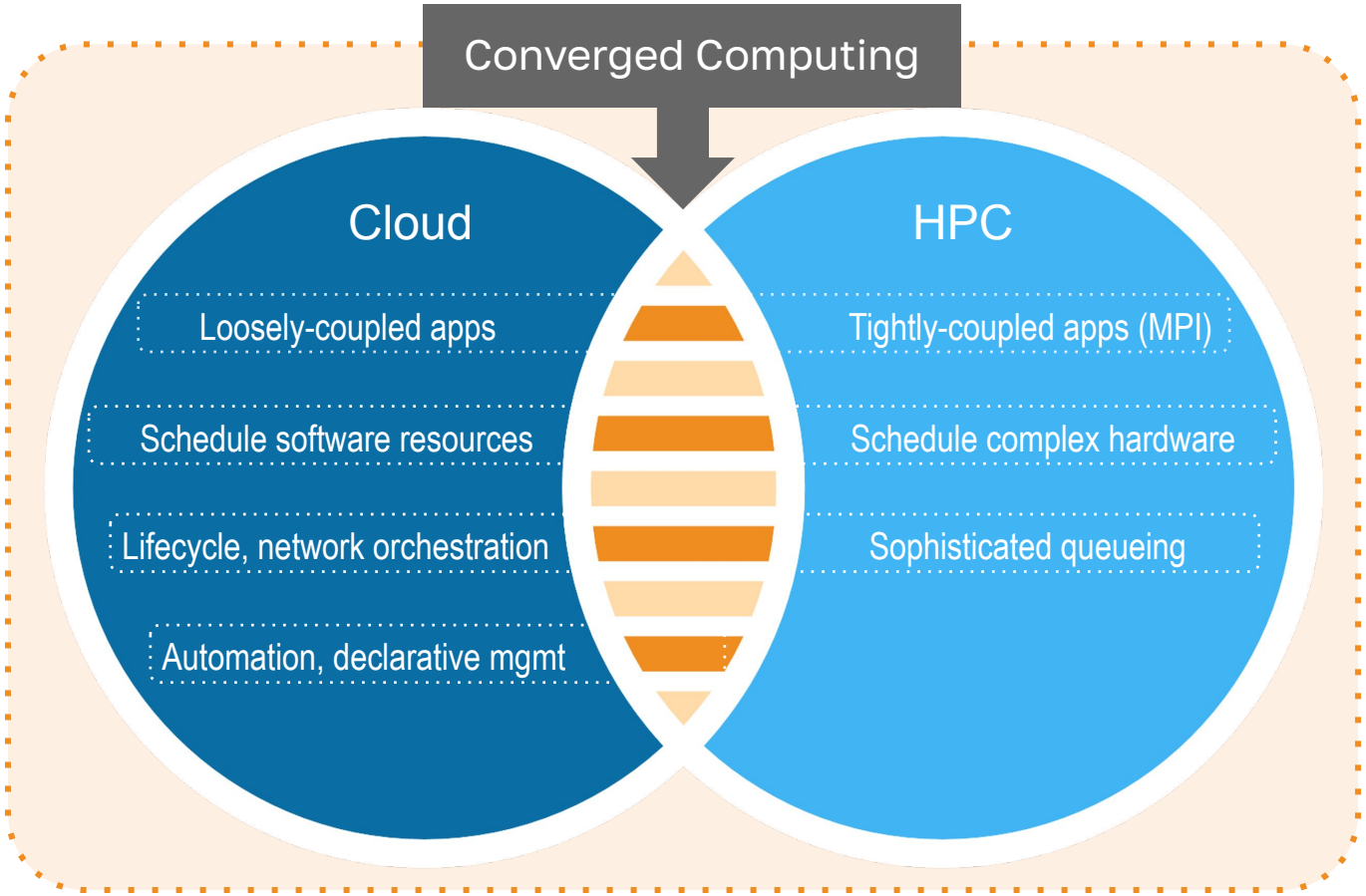
CACM: New economic cycle of computing leads to greater hardware specialization. Areas more distinct and provide fewer benefits to others. **Areas that get left behind[3]:**

- **See little performance benefit**
- **Market too small to justify upfront costs**
- **Cannot coordinate demand (cloud)**

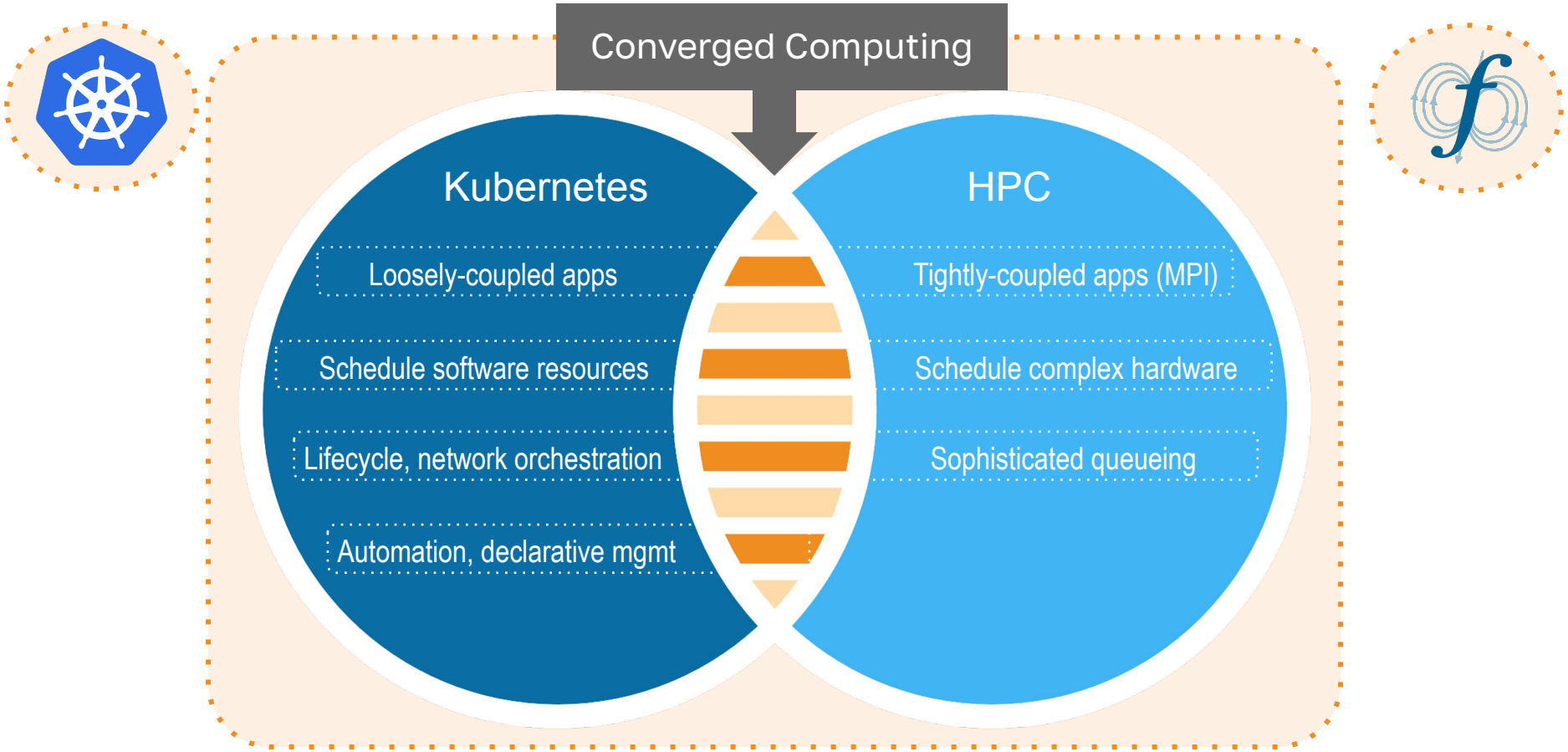
Focuses on hardware, but software development is crucial, closely connected

[1]Gartner 2022, [2]Hyperion 2021, [3]*The Decline of Computers as a General Purpose Technology,* CACM March 2021

Converged computing is combining HPC and cloud, getting the best of both worlds technologically and culturally



A key to converged computing is combining HPC scheduling and resource management with Kubernetes.



Kubernetes and Flux Framework are analogous:

flux-core | flux-sched | flux-security | flux-accounting | flux-pmix

Both are modular – composed of interchangeable components!

kube apiserver | kube scheduler | kube controller manager | kube proxy | kubelet | container runtime

Converged computing strategies enable integration of technologies across the two spaces

(a) Modularity

An application or infrastructure that is modular can have components used interchangeably.

(b) We have shared goals

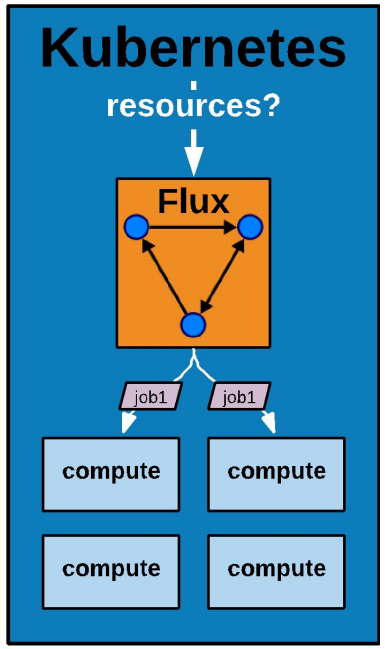
Shared goals align incentives correctly for collaboration toward shared needs, and potentially mutual vision. (we both want to run batch workloads)

(c) Integration

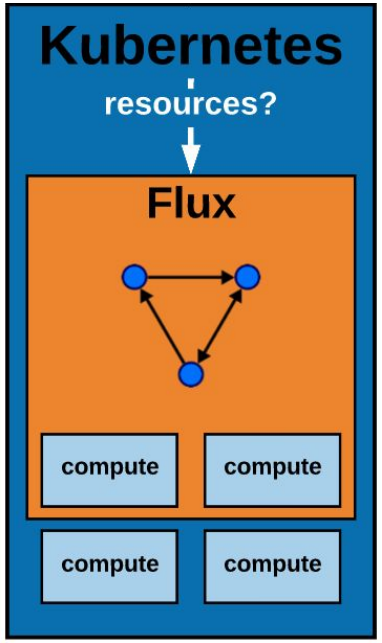
Consumption or deploying components or entirety of one inside of another.

(enabled by containers and language bindings)

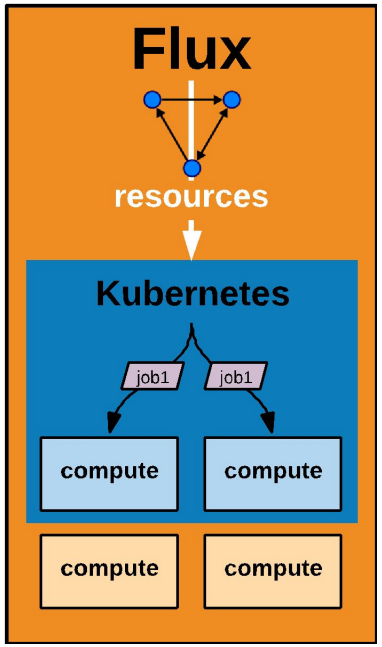
LLNL converged computing project advances convergence with representative, Flux-based models.



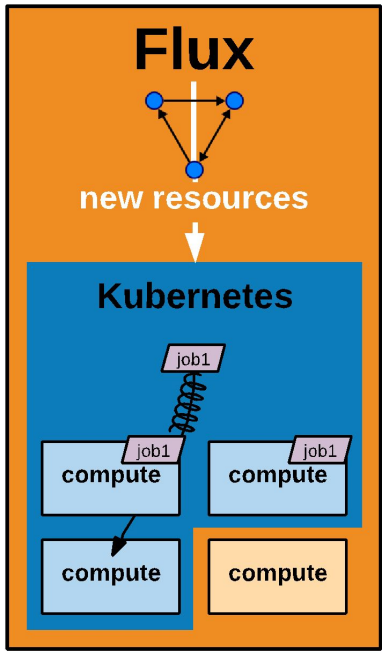
"Fluence" uses the Flux scheduler to schedule pods in Kubernetes.



The "**Flux Operator**" deploys an entire Flux cluster inside of Kubernetes.

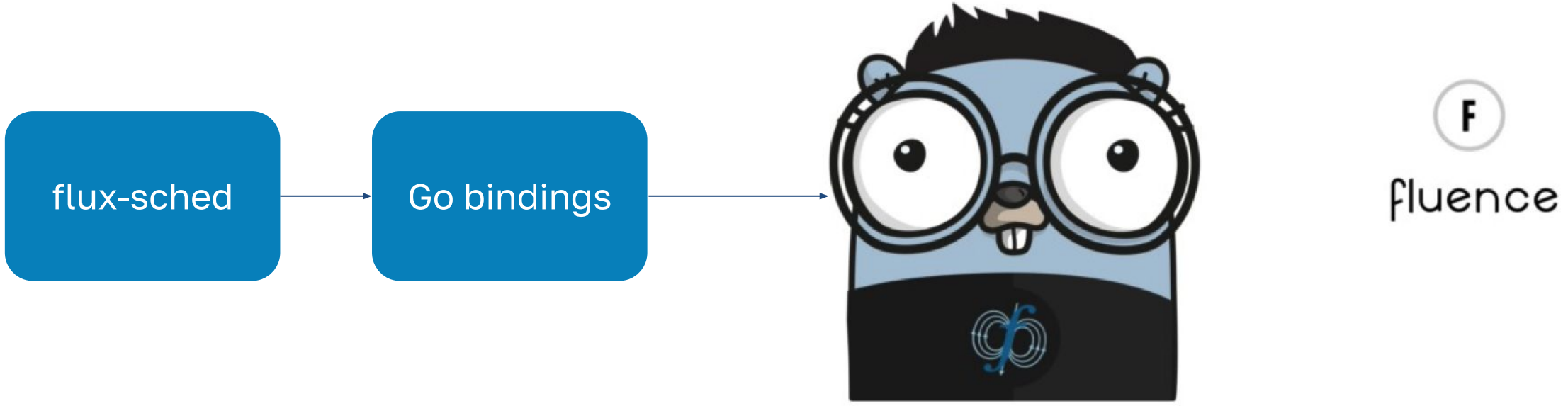


"Usernetes" with HPC (Flux) deploys user-space Kubernetes under Flux



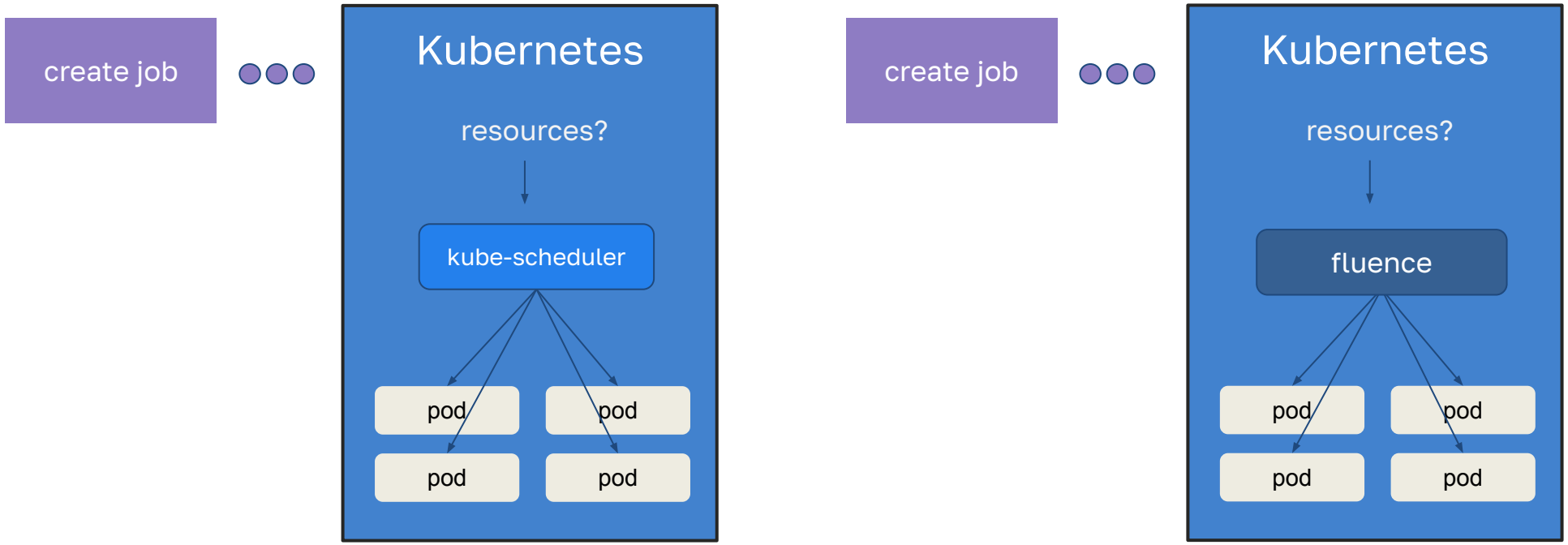
Elasticity of Flux and Kubernetes enables auto-scaling and dynamism

"Fluence" (flux-k8s) schedules work in Kubernetes



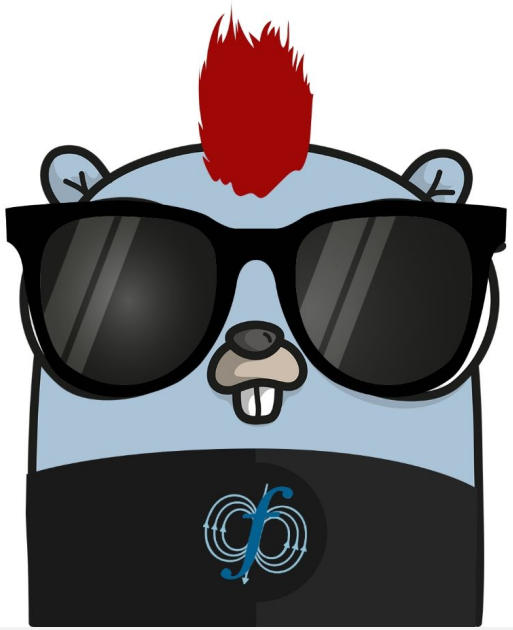
"Fluence" (flux-k8s) schedules work in Kubernetes

The dual *modularity* of **flux-sched** and of **Kubernetes**, and ability to *integrate* the two by way of Go bindings means that we can swap our Fluence scheduler with the Kubernetes default scheduler!



The Flux Operator: Flux implemented inside of Kubernetes

We can map Flux components into containers and Kubernetes abstractions to implement the entirety of Flux inside of Kubernetes via an *operator*, a controller that knows how to manage objects in a Kubernetes cluster.



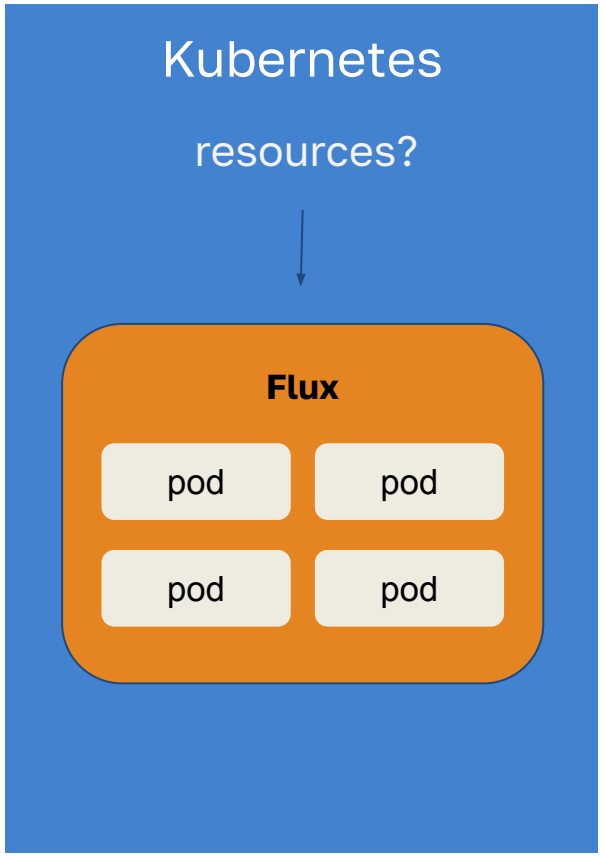
Sochat V, Culquicondor A, Ojea A and Milroy D. The Flux Operator [version 1; peer review: 2 approved]. F1000Research 2024, 13:203

The Flux Operator: Flux implemented inside of Kubernetes

You get an entire Flux cluster, called a "MiniCluster" inside of Kubernetes. Once running, we schedule jobs inside of it using Flux, and avoid stressing the Kubernetes scheduler.

Did you know!

The Flux Operator, at 4 months old, out-performed the MPI Operator, a leader in the space for running MPI workloads in Kubernetes at the time.

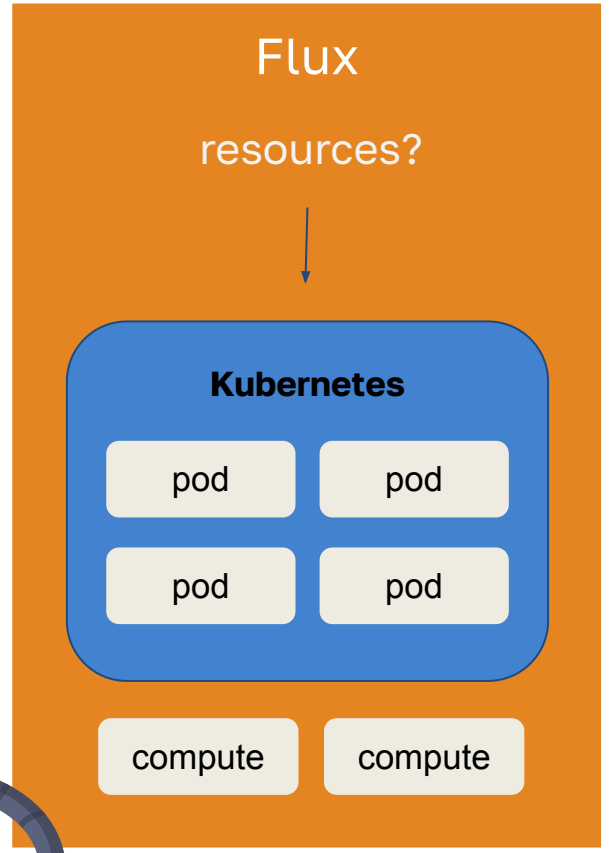


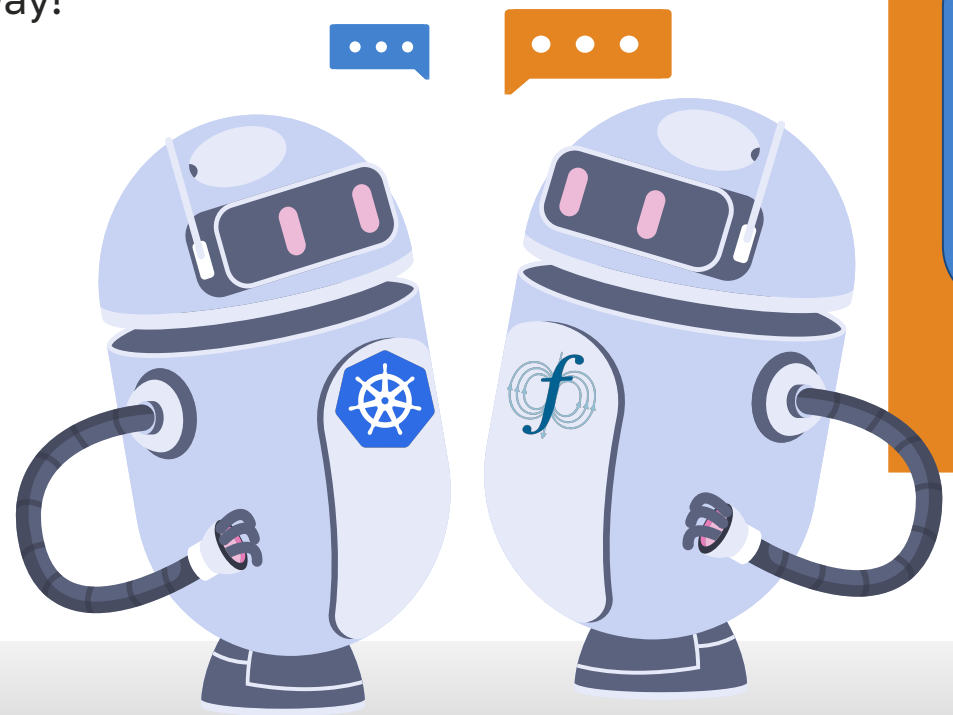
User-Space Kubernetes "Usernetes" Alongside HPC

"The Bare Metal Bros"

Deploy Kubernetes alongside Flux to get the best of both worlds - HPC simulations alongside services. We have achieved equal performance between bare metal and Kubernetes for environments, including:

- AWS using the Elastic Fabric Adapter
- Azure using Infiniband
- Production on-premises underway!

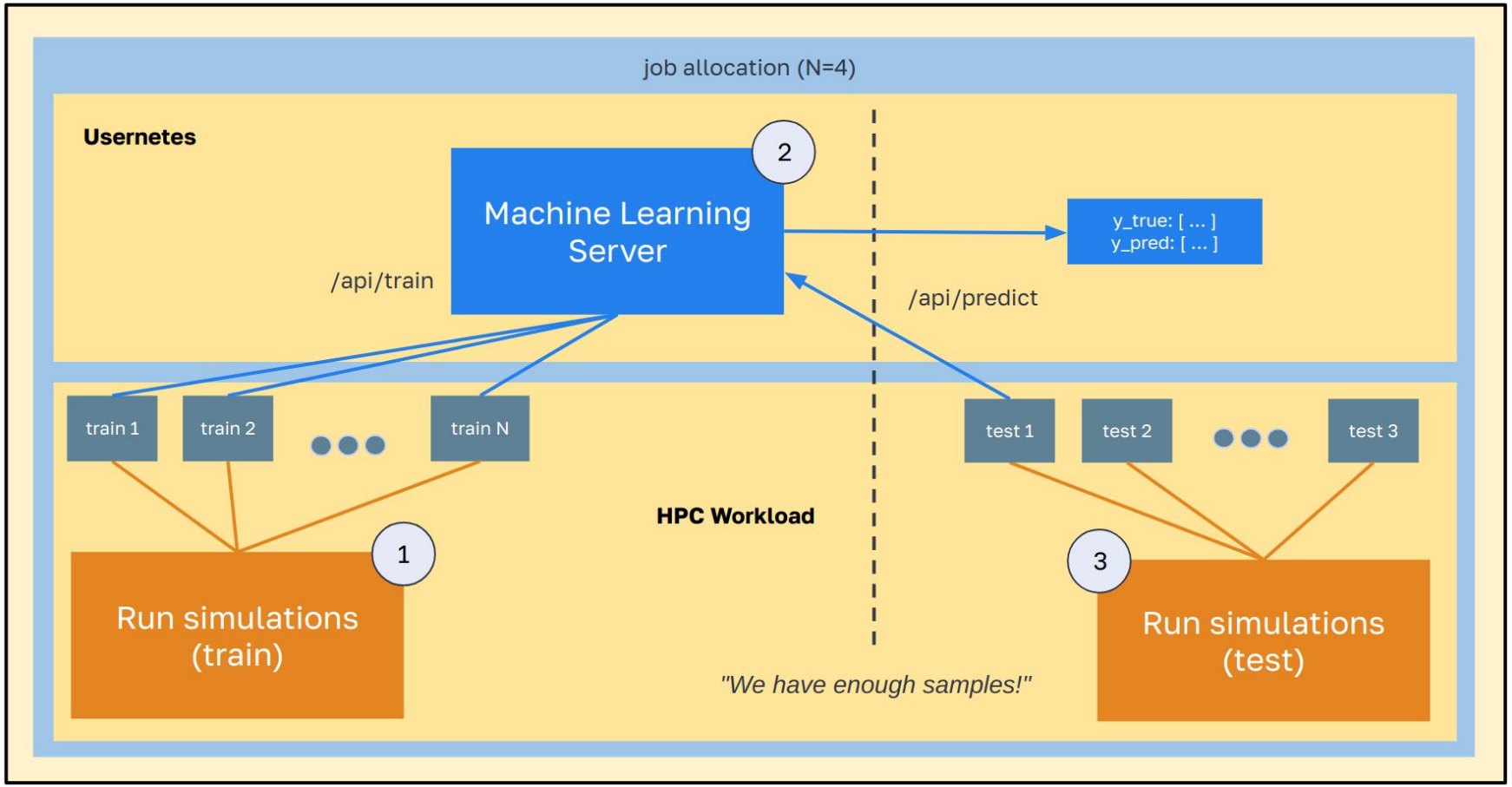




V. Sochat, D. Fox, and D. Milroy, “HPC Alongside User-space Kubernetes,” 2024. https://arxiv.org/abs/2406.06996

User-Space Kubernetes "Usernetes" Alongside HPC

"The Bare Metal Bros"



Sochat, Fox, and Milroy 2024

Flux team has made excellent progress for El Capitan deployment in response to feedback from stakeholders

- ✓ New interfaces for watching and querying jobs and queues
- ✓ Improved tools for file broadcast
- ✓ Utility for posting manual job events
- ✓ Scalability/reliability improvements to running on large systems
- ✓ Improvements to Flux user interfaces and human readability of logs
- ✓ More options for generating taskmaps
- ✓ Command flux update to change job duration after submit
- ✓ Housekeeping: epilog tasks decoupled from jobs

We formed industry and academic collaborations to tackle converged computing, contribute to community, and advocate for HPC.

IBM T.J. Watson Research Center:

Expertise in K8s Scheduling

Red Hat:

Developers of OpenShift K8s platform

AWS:

Largest cloud platform, deep HPC expertise

Google:

Collaborated on Flux Operator

UTK:

Science workflows, converged computing

LLNL:

Dan Milroy
Aniruddha Marathe
Md Rajib Hossen
Zeke Morton
Tapasya Patki
Abhik Sarkar
Vanessa Sochat
Jae-Seung Yeom

Accessing the hands-on tutorial

- Using our EKS cluster at https://tutorial.flux-framework.org
  - Choose a unique username (if not, you'll be sharing a pod)
  - It's a shared instance! Please don't run computationally demanding tasks.
  - The cluster will disappear shortly after the tutorial
- Running locally with Docker:
  - git clone https://github.com/flux-framework/Tutorials.git
    - README in the 2024-HPCIC JupyterNotebook directory
- We'd like your feedback! Please take our tutorial survey:

Thank you!
Questions?

milroy1@llnl.gov | sochat1@llnl.gov

https://github.com/flux-framework/Tutorials.git