Enabling HPC and ML workloads with the latest Kubernetes Job features

Michał Woźniak - Google
Vanessa Sochat - Lawrence Livermore National Laboratory

Early batch at Kubernetes

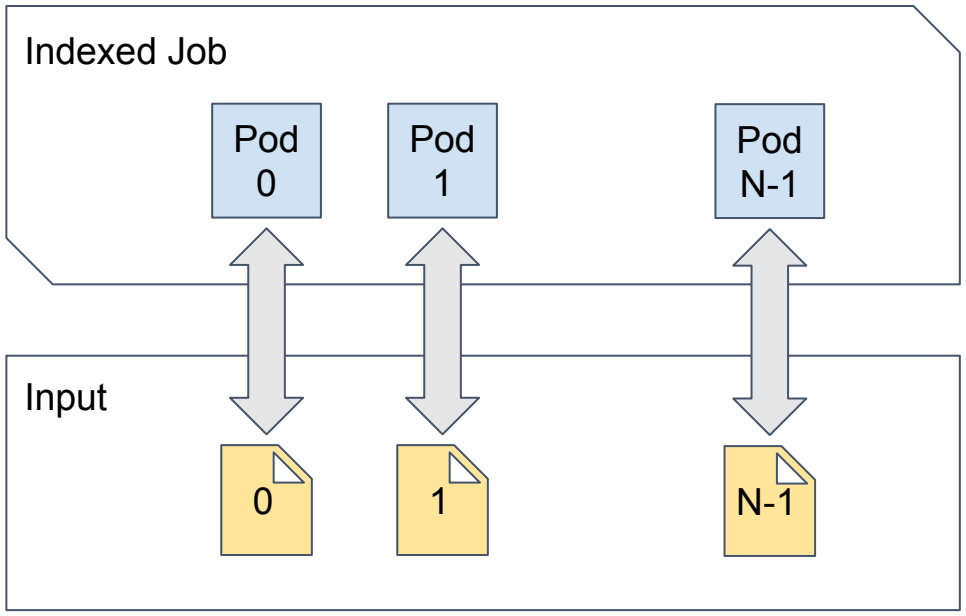
- Kubernetes was built originally for long-running stateless applications
- Feature gaps in the early **batch/v1** Job API
- Re-implementation of the same features by different frameworks
- **Batch WG** initiative to improve batch support in the core Kubernetes

Recent Job enhancements

Incomplete support for periodic jobs	**CronJob** (GA 1.21) **TimeZone support** (GA 1.27)
Job objects accumulating after completion	**TTL After Finished** (GA 1.23)
Parallel jobs require external task queue Difficult pod-to-pod communication	**Indexed Job** (GA 1.24) **Elastic Indexed Jobs** (Beta 1.27)
No direct way to check the number of running pods	**Pods Ready** (Beta 1.24)
No control over job startup and preemption	**Job Suspend** (GA 1.24)
Job controller losing track of completed pods	**Job Tracking with finalizers** (GA 1.26)
Very limited control of control for pod failures	**Pod Failure Policy** (Beta 1.26)

Indexed Jobs

Each worker “knows” its index via the **JOB_COMPLETION_INDEX** environment variable




```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: 'sample-job'
spec:
  completions: 3
  parallelism: 3
  completionMode: Indexed
  template:
    spec:
      restartPolicy: Never
      containers:
      - command:
        - 'bash'
        - '-c'
        - 'echo "My partition: ${JOB_COMPLETION_INDEX}"'
        image: 'docker.io/library/bash'
        name: 'sample-load'
```

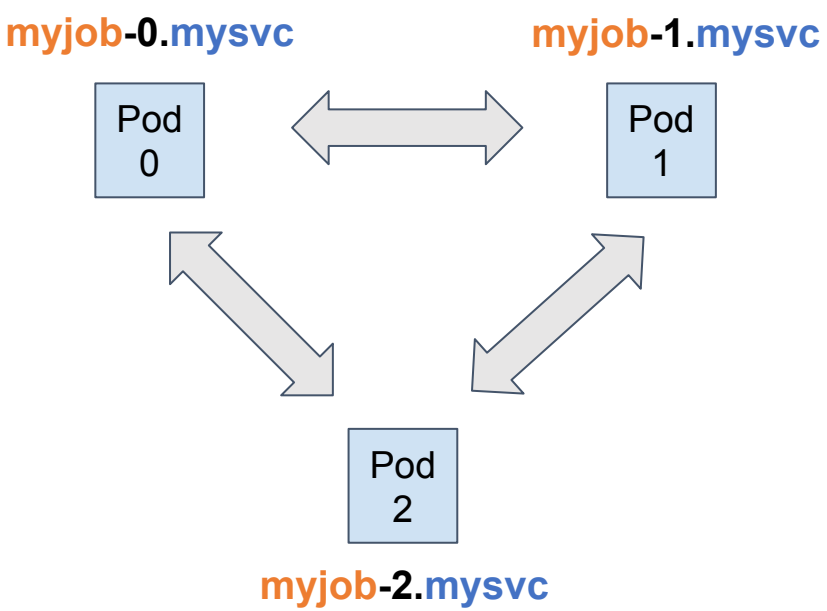
No need for managing a queue of tasks!

Indexed Jobs

When combined with a headless service each worker has a stable DNS name corresponding to its index

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  completions: 3
  parallelism: 3
  completionMode: Indexed
  template:
    spec:
      subdomain: mysvc
      containers:
      - name: main
        ...
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysvc
spec:
  clusterIP: None
  selector:
    job-name: myjob
```

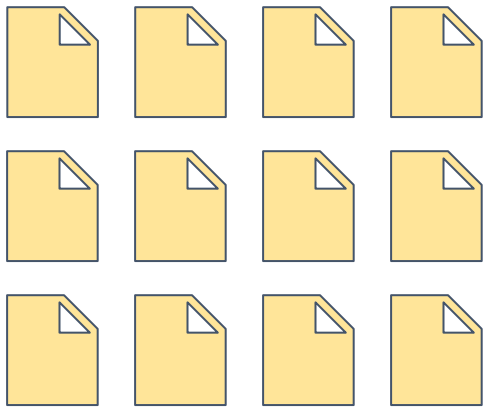


Indexed Jobs at DeepMind

Acknowledgements: George Thomas and Lena Martens (DeepMind)

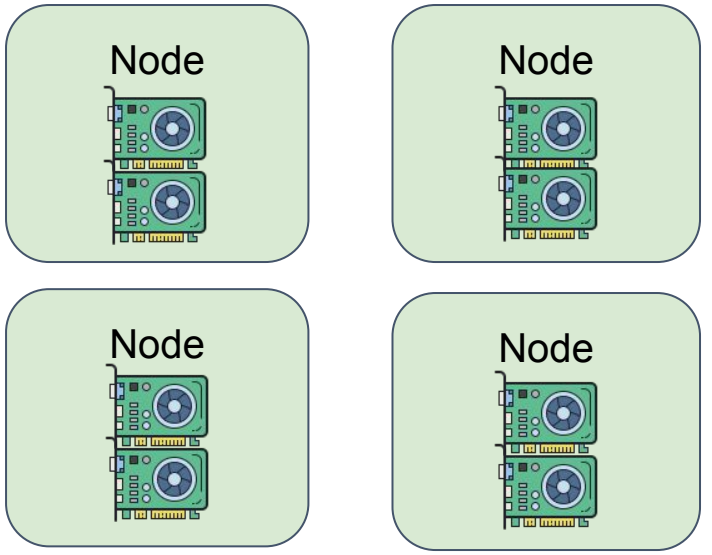
Indexed Job: Distributed ML training

- **Use case**: large input dataset that is sharded across multiple hosts during ML training



Dataset

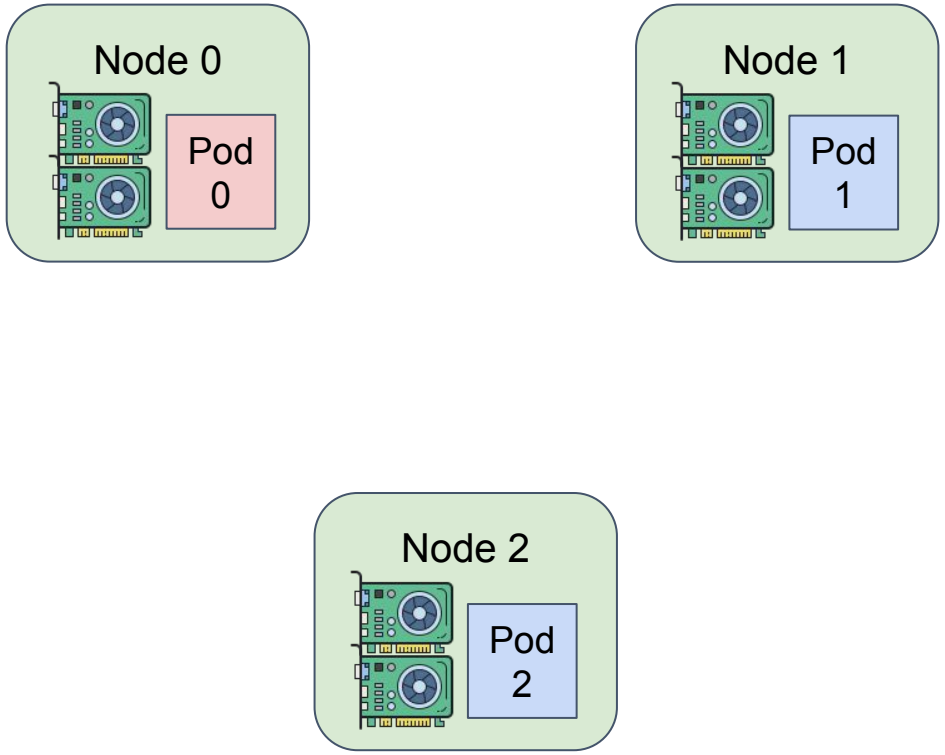
Model



Compute hardware
(typically GPU or TPU)

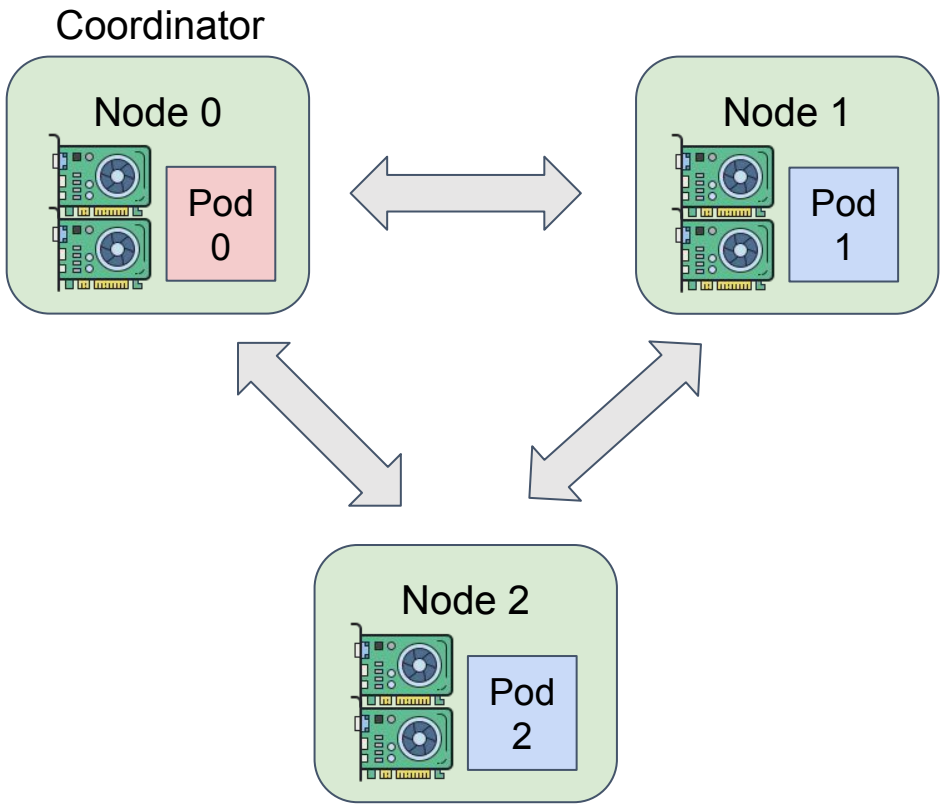
Indexed Job: Distributed ML training

- **Indexed Job creates pods**



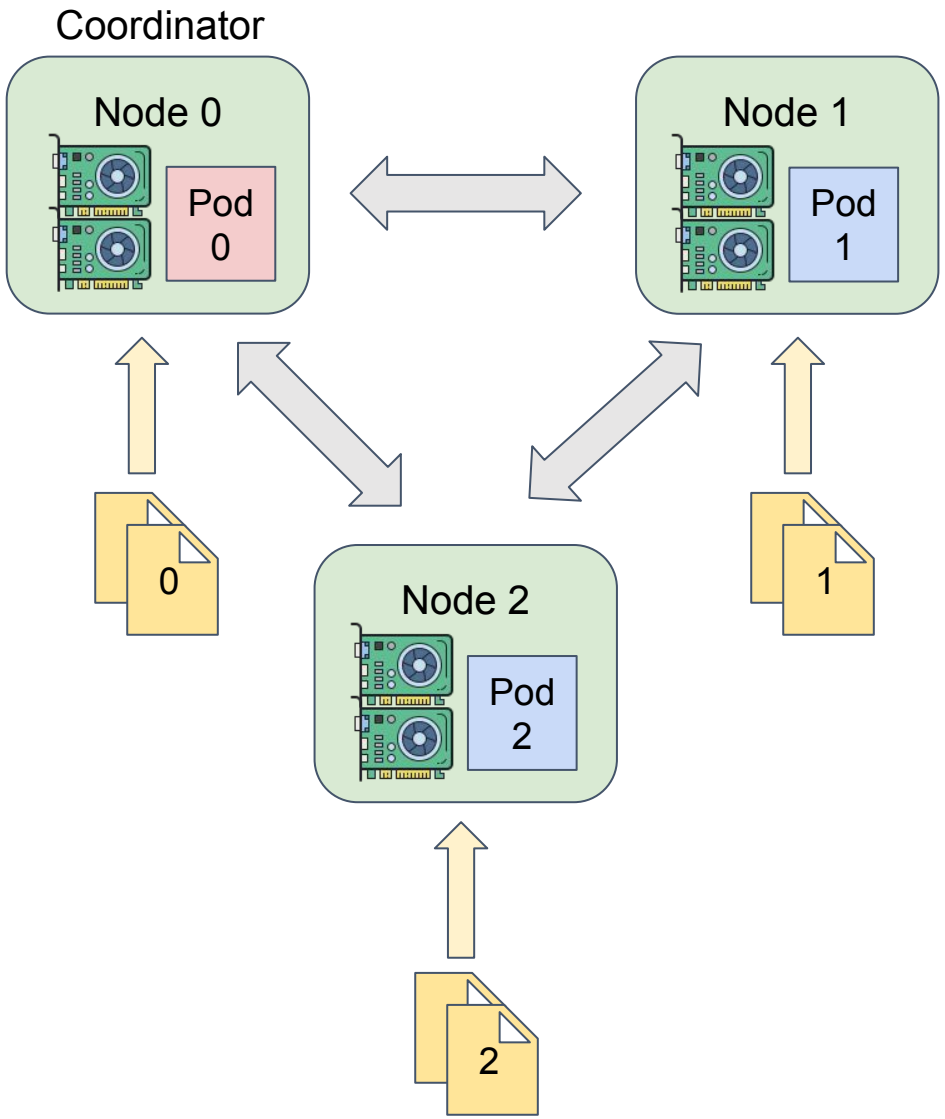
Indexed Job: Distributed ML training

- **Indexed Job creates pods**
- **Distributed environment setup** - coordinator awaits for all pods to be ready



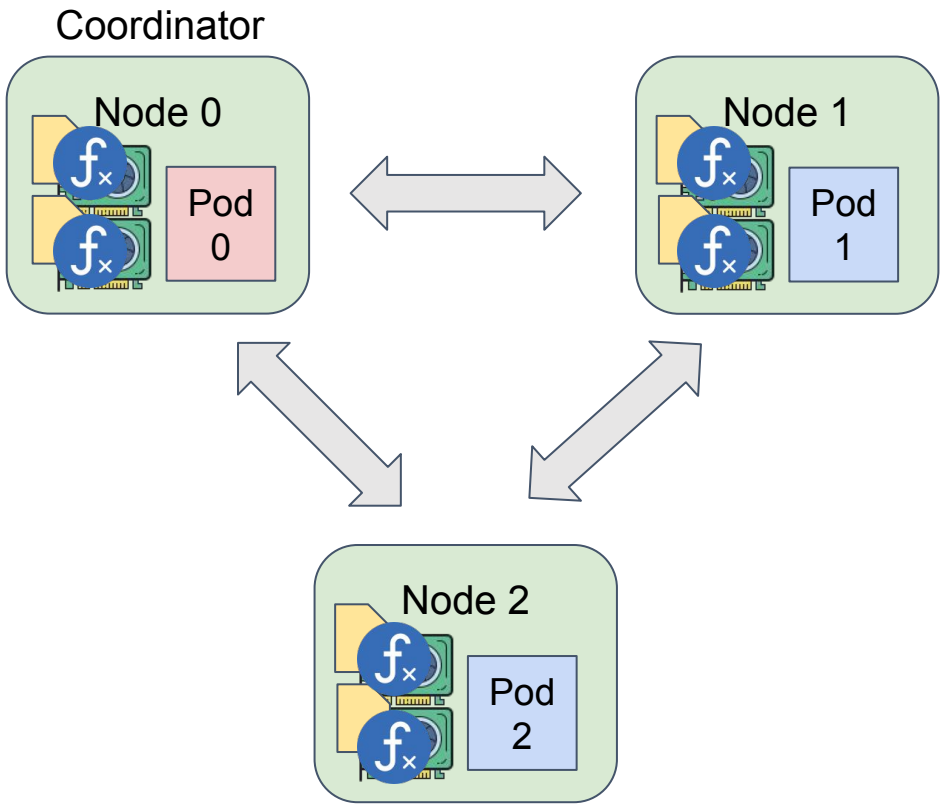
Indexed Job: Distributed ML training

- **Indexed Job creates pods**
- **Distributed environment setup** - coordinator awaits for all pods to be ready
- **Load data batch** and distribute among the local devices



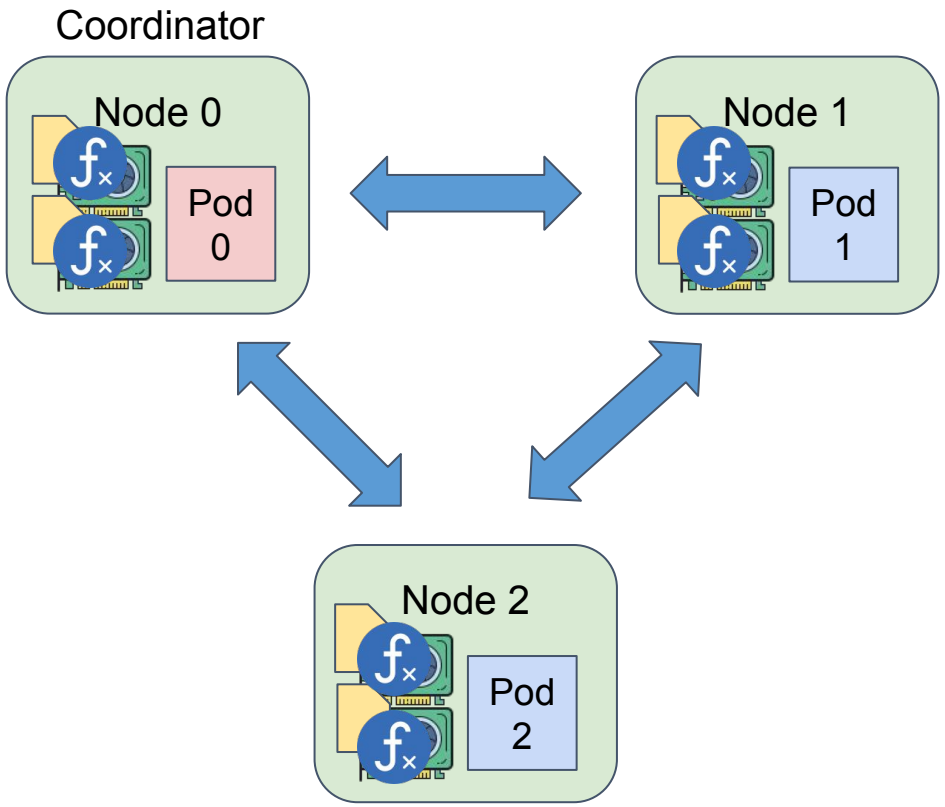
Indexed Job: Distributed ML training

- **Indexed Job creates pods**
- **Distributed environment setup** - coordinator awaits for all pods to be ready
- **Load data batch** and distribute among the local devices
- **Load model** and distribute among local devices. **Simplifying assumption**: model loads on a local device



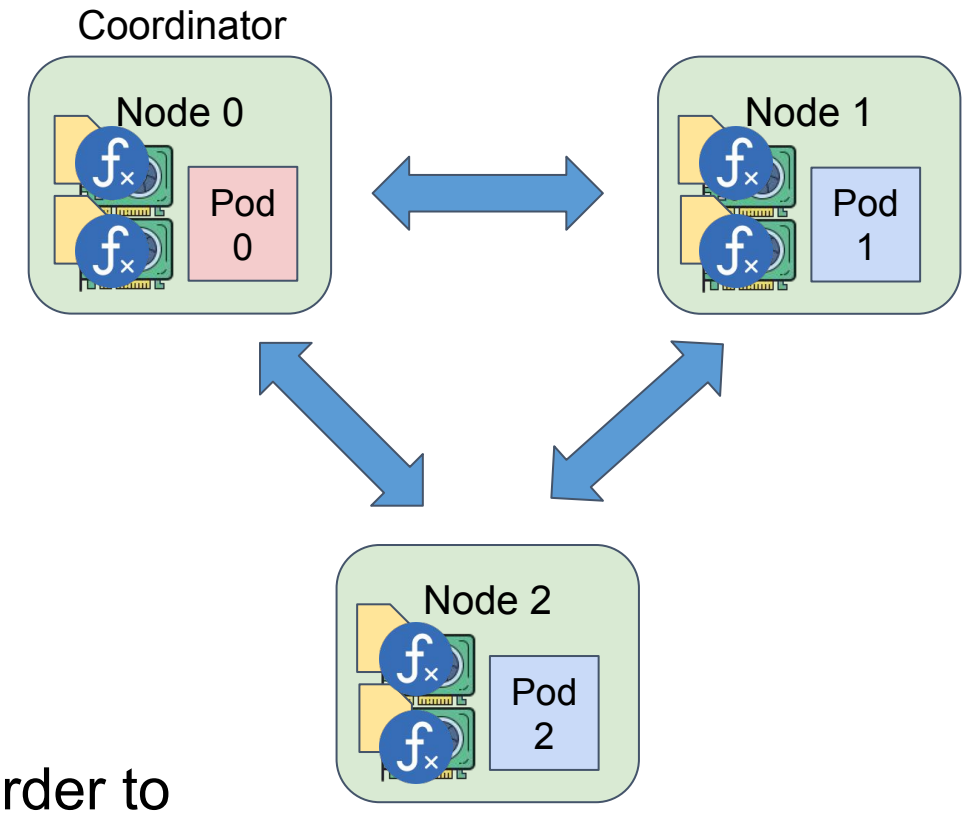
Indexed Job: Distributed ML training

- **Indexed Job creates pods**
- **Distributed environment setup** - coordinator awaits for all pods to be ready
- **Load data batch** and distribute among the local devices
- **Load model** and distribute among local devices. **Simplifying assumption**: model loads on a local device
- **Pod-to-pod communication** during training



Indexed Job: Distributed ML training

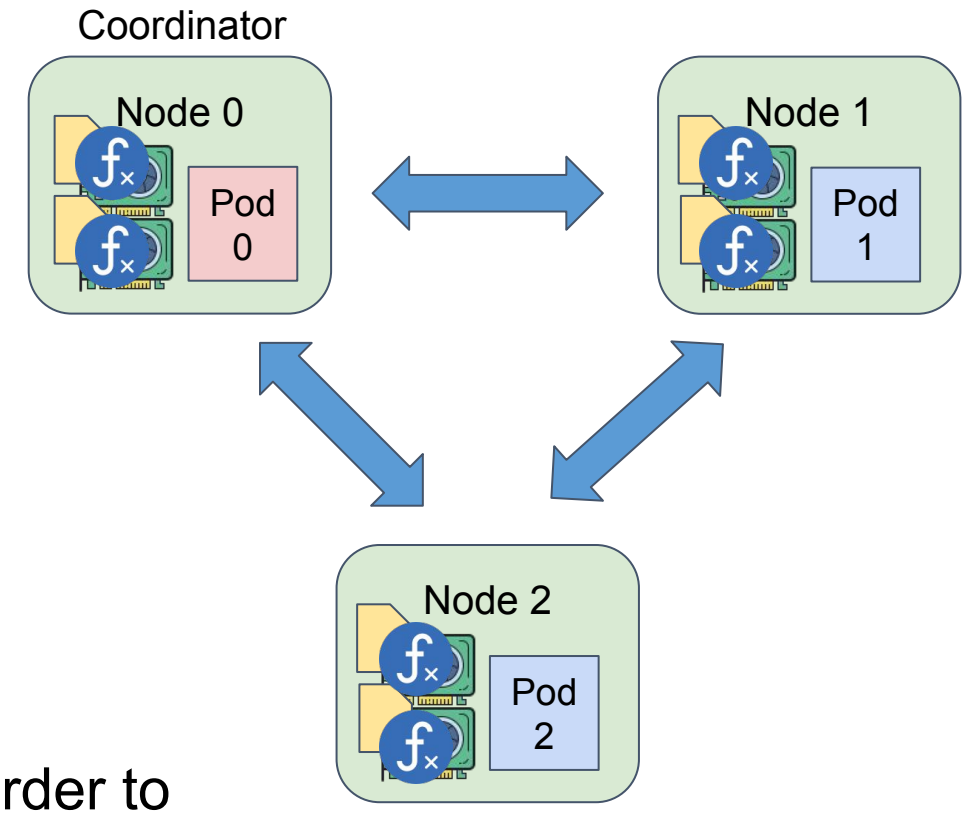
- **Indexed Job creates pods**
- **Distributed environment setup** - coordinator awaits for all pods to be ready
- **Load data batch** and distribute among the local devices
- **Load model** and distribute among local devices. **Simplifying assumption**: model loads on a local device
- **Pod-to-pod communication** during training



Coordinator also performs computations in order to utilize the GPU assigned to it

Indexed Job: Distributed ML training

- **Indexed Job creates pods**
- **Distributed environment setup** - coordinator awaits for all pods to be ready
- **Load data batch** and distribute among the local devices
- **Load model** and distribute among local devices. **Simplifying assumption**: model loads on a local device
- **Pod-to-pod communication** during training

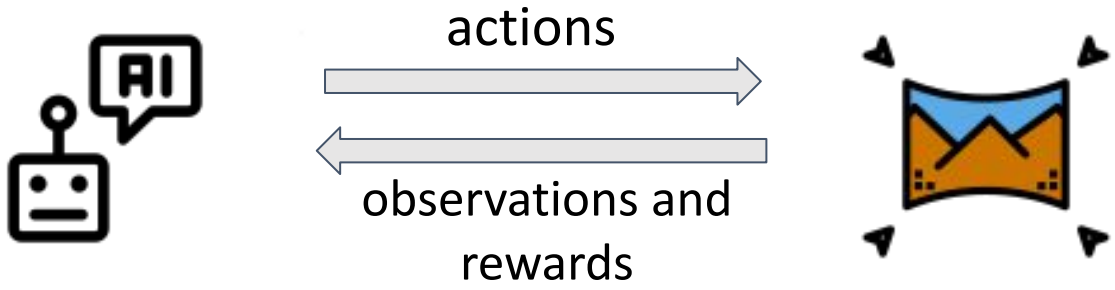


Coordinator also performs computations in order to utilize the GPU assigned to it

Directly usable with frameworks such as **PyTorch** and **JAX**
Inspired code samples: https://github.com/google/learn-oss-with-google

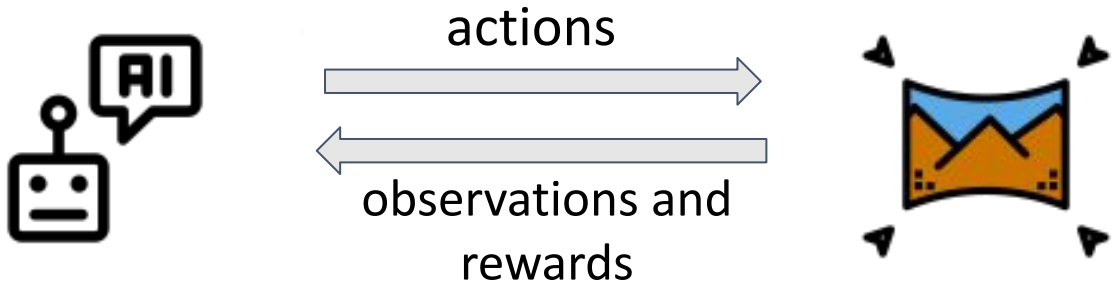
Indexed Job: Agent-environment 1:1

Use case: Agent-environment simulations for Reinforcement Learning



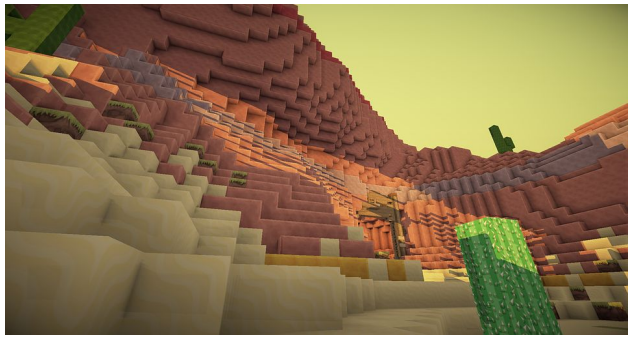
Indexed Job: Agent-environment 1:1

Use case: Agent-environment simulations for Reinforcement Learning

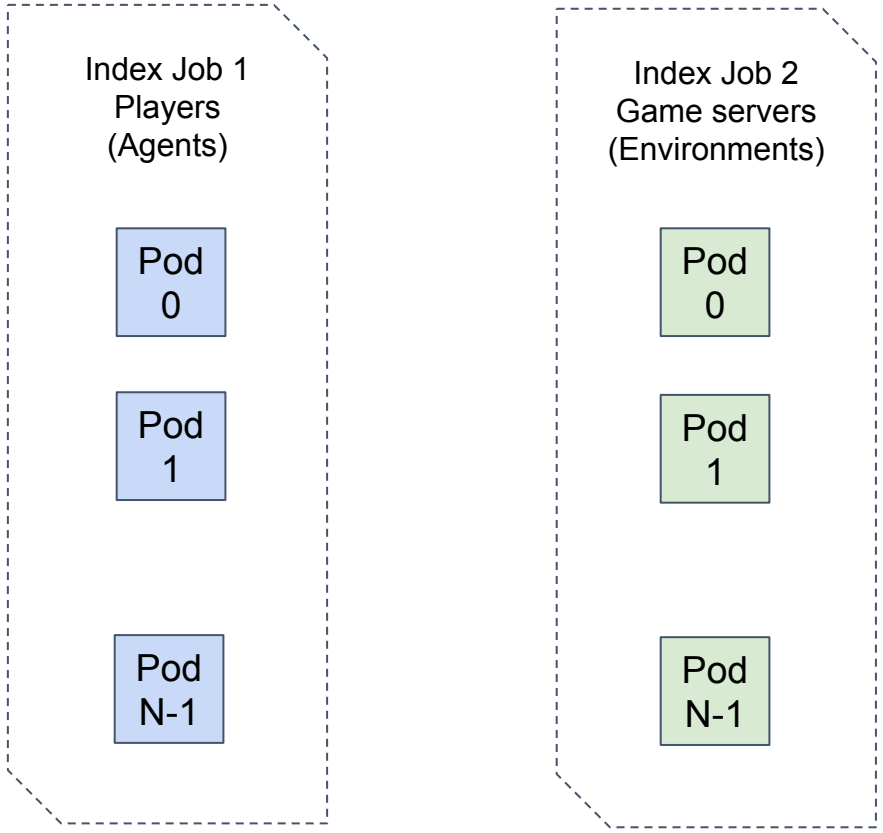


We want to replicate the simulation to:

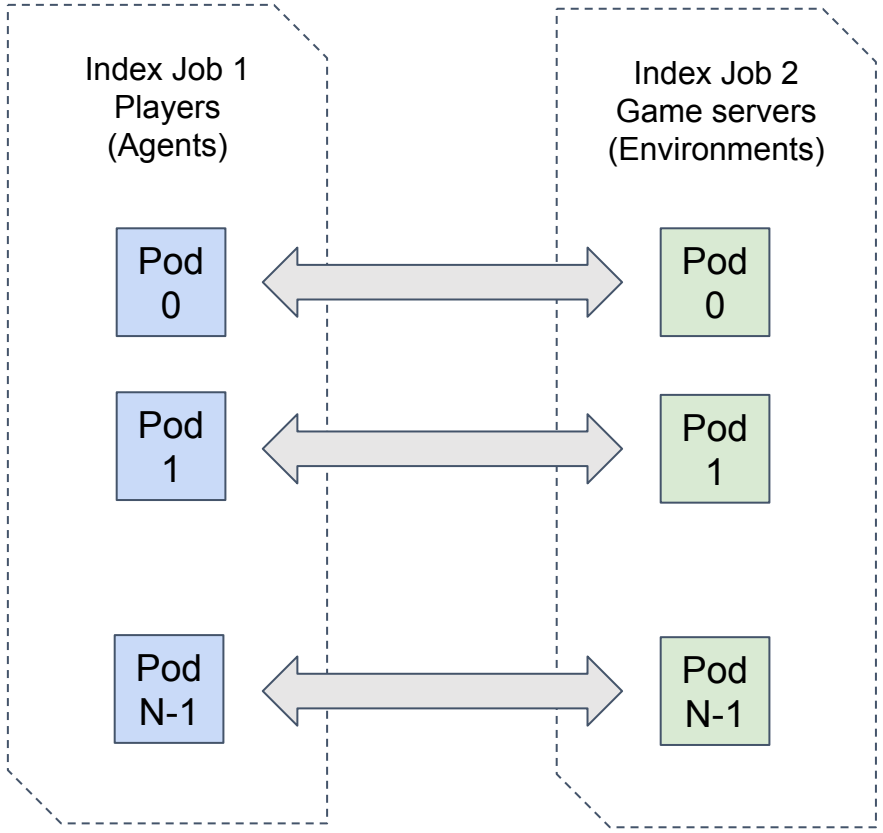
- Collect more data
- Account for variation in initial conditions



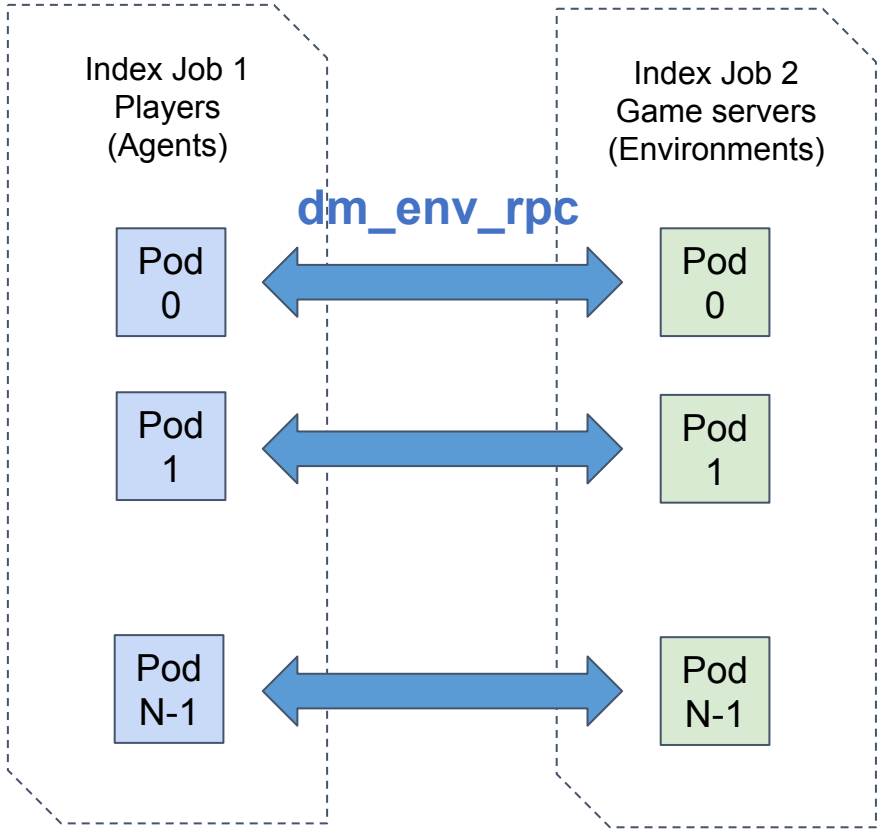
Indexed Job: Agent-environment 1:1



Indexed Job: Agent-environment 1:1

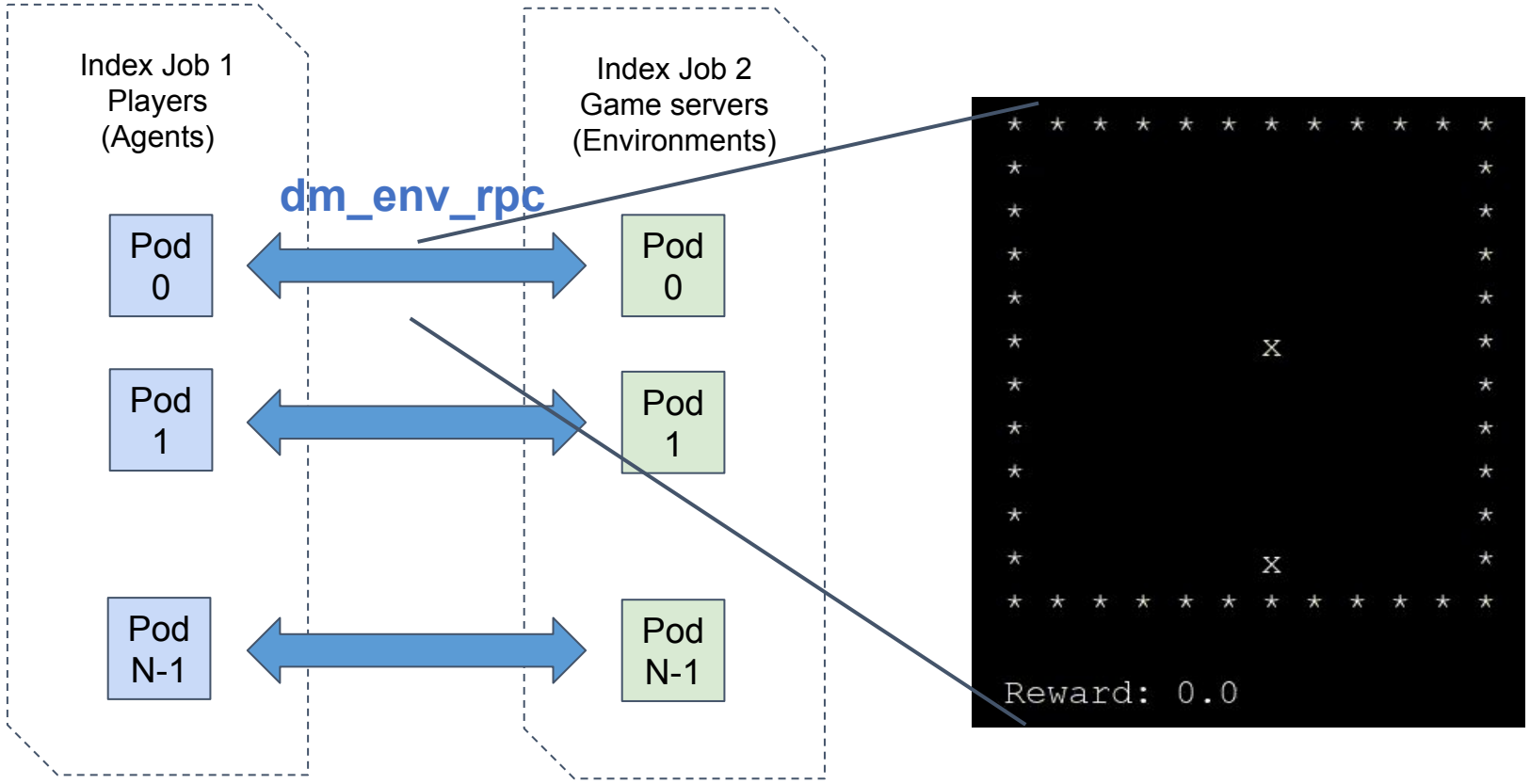


Indexed Job: Agent-environment 1:1



dm_env: The DeepMind RL Environment API
dm_env_rpc: A networking protocol for agent-environment communication

Indexed Job: Agent-environment 1:1



dm_env: The DeepMind RL Environment API
dm_env_rpc: A networking protocol for agent-environment communication

Inspired code sample: https://github.com/google/learn-oss-with-google

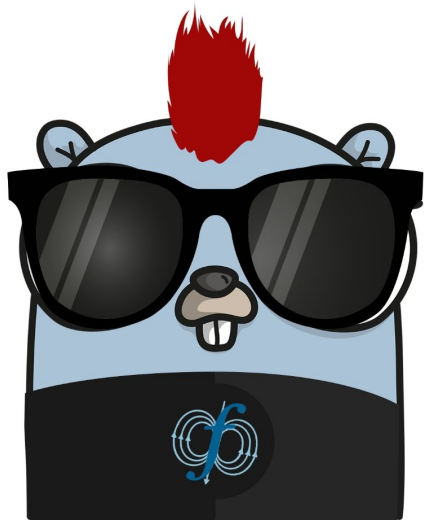
Indexed Jobs at DeepMind

StatefulSets were used before, but:

- no concept of completion
- failed pods would retry continuously

	StatefulSets	Regular Job
Replica knows its index	✅	🧩
Stable DNS names	✅	🧩
Concept of completion	🧩	✅
Limited retries	🧩	✅

Flux Operator



THE OPERATOR

coming to K8s near you...

The Flux Operator

Kubecon 2023, Amsterdam

Vanessa Sochat
Computer Scientist
Livermore Computing

LLNL-PRES-845504

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

Flux Framework

A job scheduler that combines hierarchical management with graph based scheduling.


flux
HPC Land


HPC Land
slurm
workload manager
flux
torque
HTCondor
High Throughput Computing


HPC Land
flux
SHIFTER
Charliecloud


flux
HPC Land

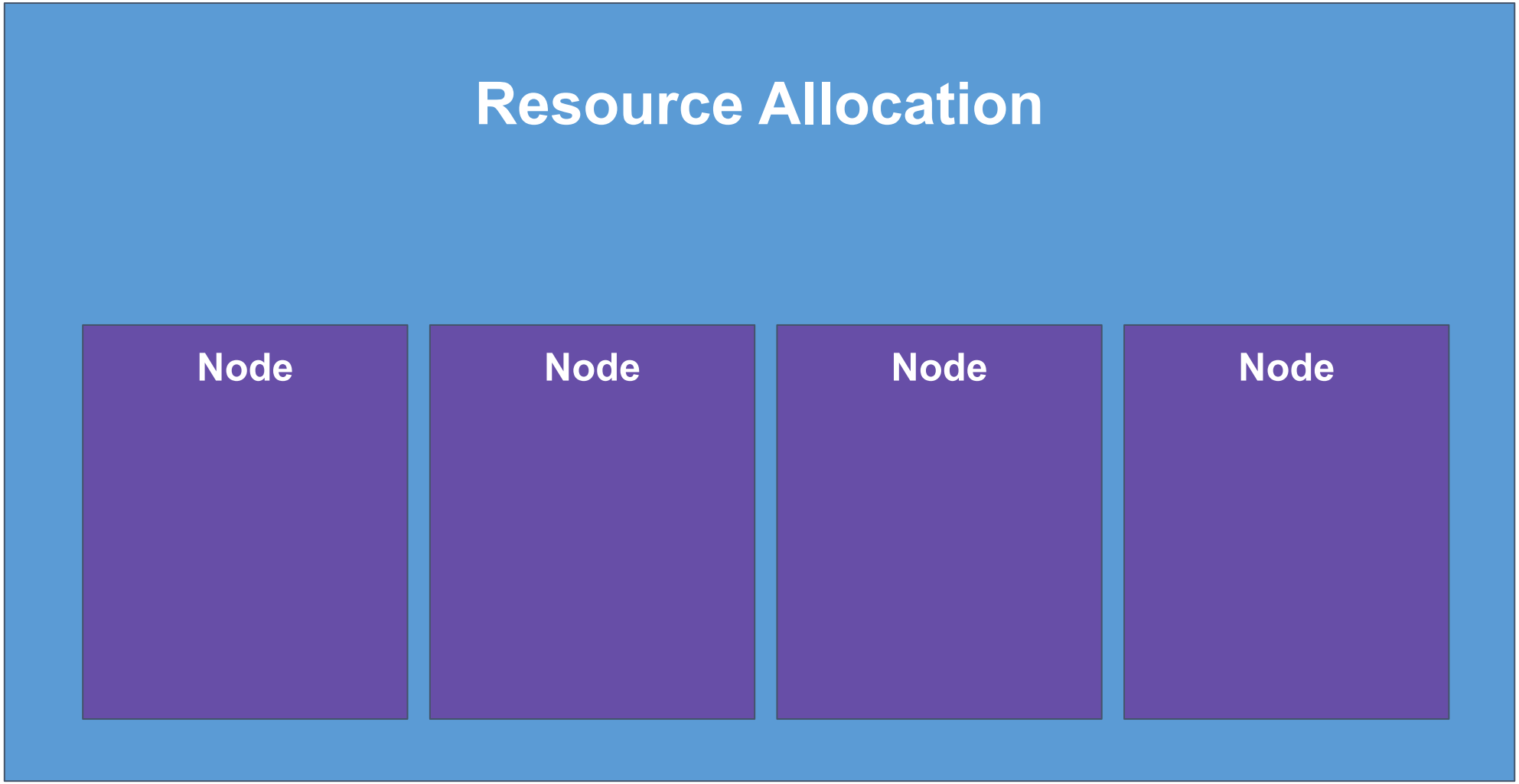
Flux Compared to Other Resource Managers

	Multi-User Mode								
Features	**Flux**	**SLURM**	**PBSPro (OpenPBS)**	**LSF**	**MOAB**	**RadicalPilot**	**Balsam**	**Parsl**	**Nitro**
Multi-user workload management	yes	yes	yes	yes	yes	no	yes	no	no
Full hierarchical resource management	yes	no	no	no	no	n/a	n/a	n/a	n/a
Graph-based advanced resource management	yes	no	no	no	no	n/a	n/a	n/a	n/a
Scheduling specialization	yes	no	no	no	no	n/a	n/a	n/a	n/a
Security: only a small isolated layer running in privileged mode for tighter security	yes	no	no	no	no	n/a	n/a	n/a	n/a
Modern command-line interface (cli) design	yes	outdated	outdated	outdated	outdated	n/a	n/a	n/a	n/a
Application programming interface (APIs) for job management, job monitoring, resource monitoring, low-level messaging	yes (4/4)	some (3/4)	some (2/4)	some (2/4)	some (3/4)	n/a	n/a	n/a	n/a
Language bindings	yes (C, C++, Python, Lua, Rust, Julia, REST)	some (C, REST)	some (C, Python)	some (C, Python)	some (C)	n/a	n/a	n/a	n/a
Bulk job submission	yes	only uniform jobs	only uniform jobs	only uniform jobs	only uniform jobs	n/a	n/a	n/a	n/a
High-speed streaming job submission	yes	no	no	no	no	n/a	n/a	n/a	n/a

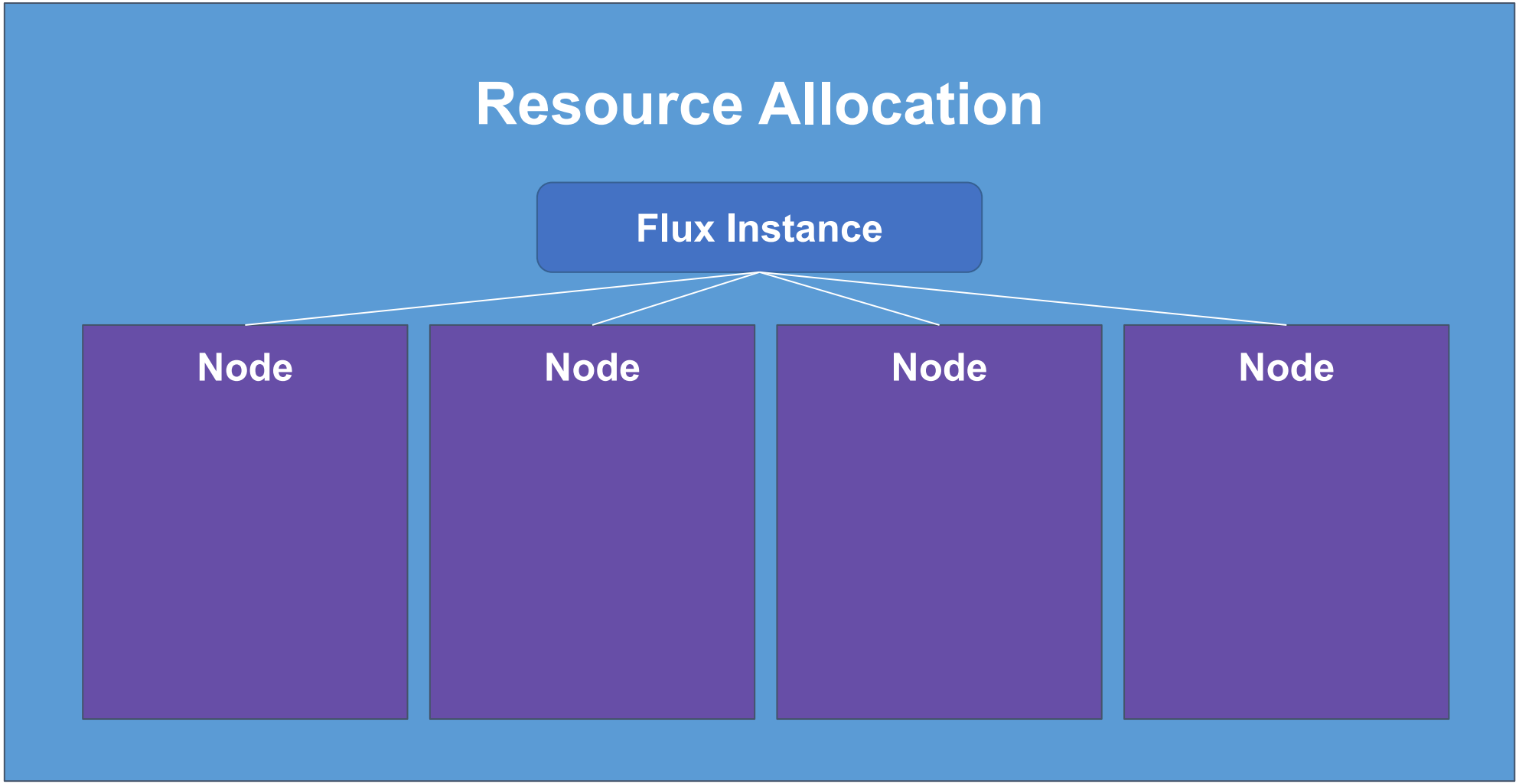
Flux Compared to Other Resource Managers

	Multi-User Mode								
Features	**Flux**	**SLURM**	**PBSPro (OpenPBS)**	**LSF**	**MOAB**	**RadicalPilot**	**Balsam**	**Parsl**	**Nitro**
Multi-user workload management	yes	yes	yes	yes	yes	no	yes	no	no
Full hierarchical resource management	yes	no	no	no	no	n/a	n/a	n/a	n/a
Graph-based advanced resource management	yes	no	no	no	no	n/a	n/a	n/a	n/a
Scheduling specialization	yes	no	no	no	no	n/a	n/a	n/a	n/a
Security: only a small isolated layer running in privileged mode for tighter security	yes	no	no	no	no	n/a	n/a	n/a	n/a
Modern command-line interface (cli) design	yes	outdated	outdated	outdated	outdated	n/a	n/a	n/a	n/a
Application programming interface (APIs) for job management, job monitoring, resource monitoring, low-level messaging	yes (4/4)	some (3/4)	some (2/4)	some (2/4)	some (3/4)	n/a	n/a	n/a	n/a
Language bindings	yes (C, C++, Python, Lua, Rust, Julia, REST)	some (C, REST)	some (C, Python)	some (C, Python)	some (C)	n/a	n/a	n/a	n/a
Bulk job submission	yes	only uniform jobs	only uniform jobs	only uniform jobs	only uniform jobs	n/a	n/a	n/a	n/a
High-speed streaming job submission	yes	no	no	no	no	n/a	n/a	n/a	n/a

What does graph-based resource management mean?

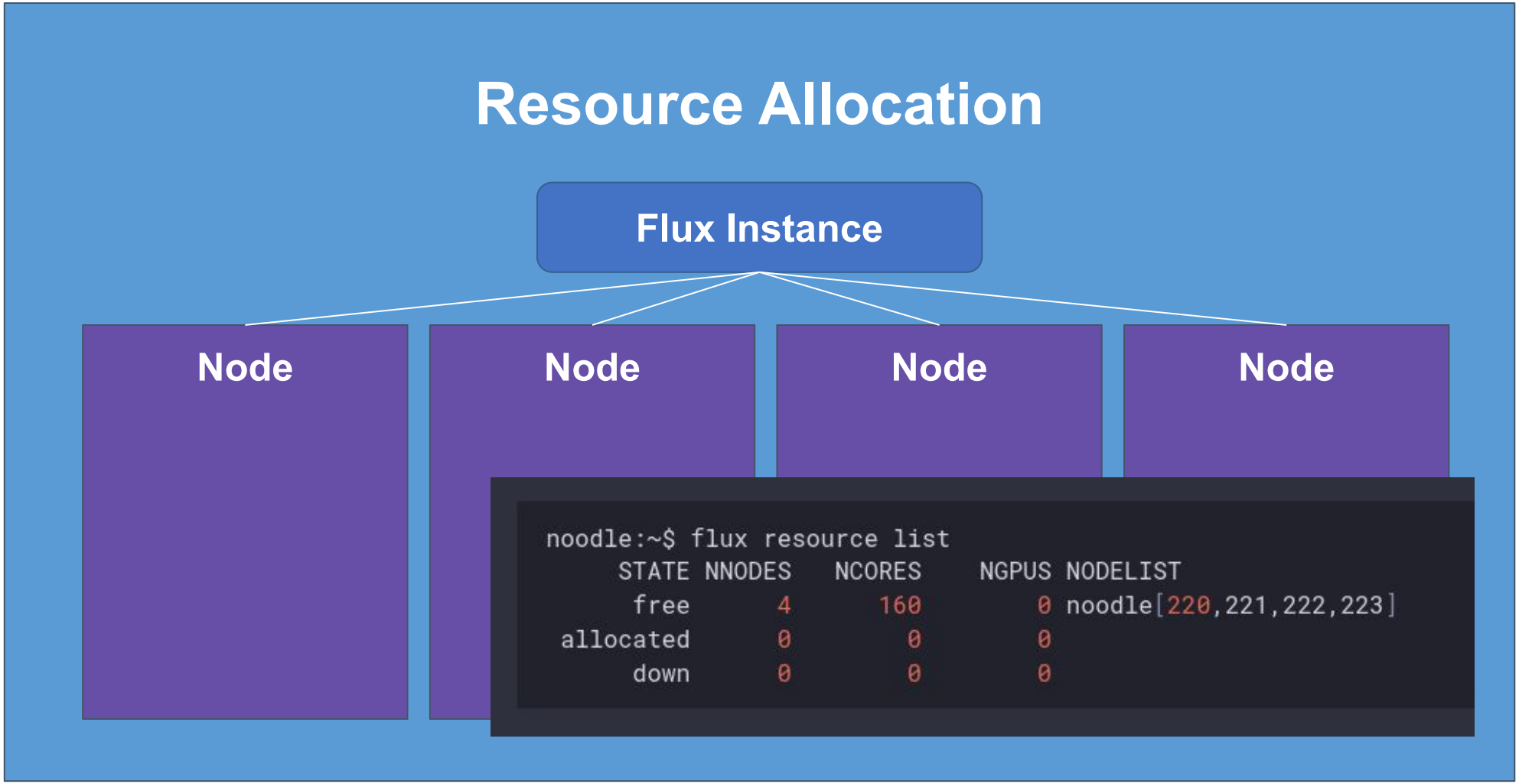


These are my allocated resources on an HPC cluster
(or a set of networked pods)!

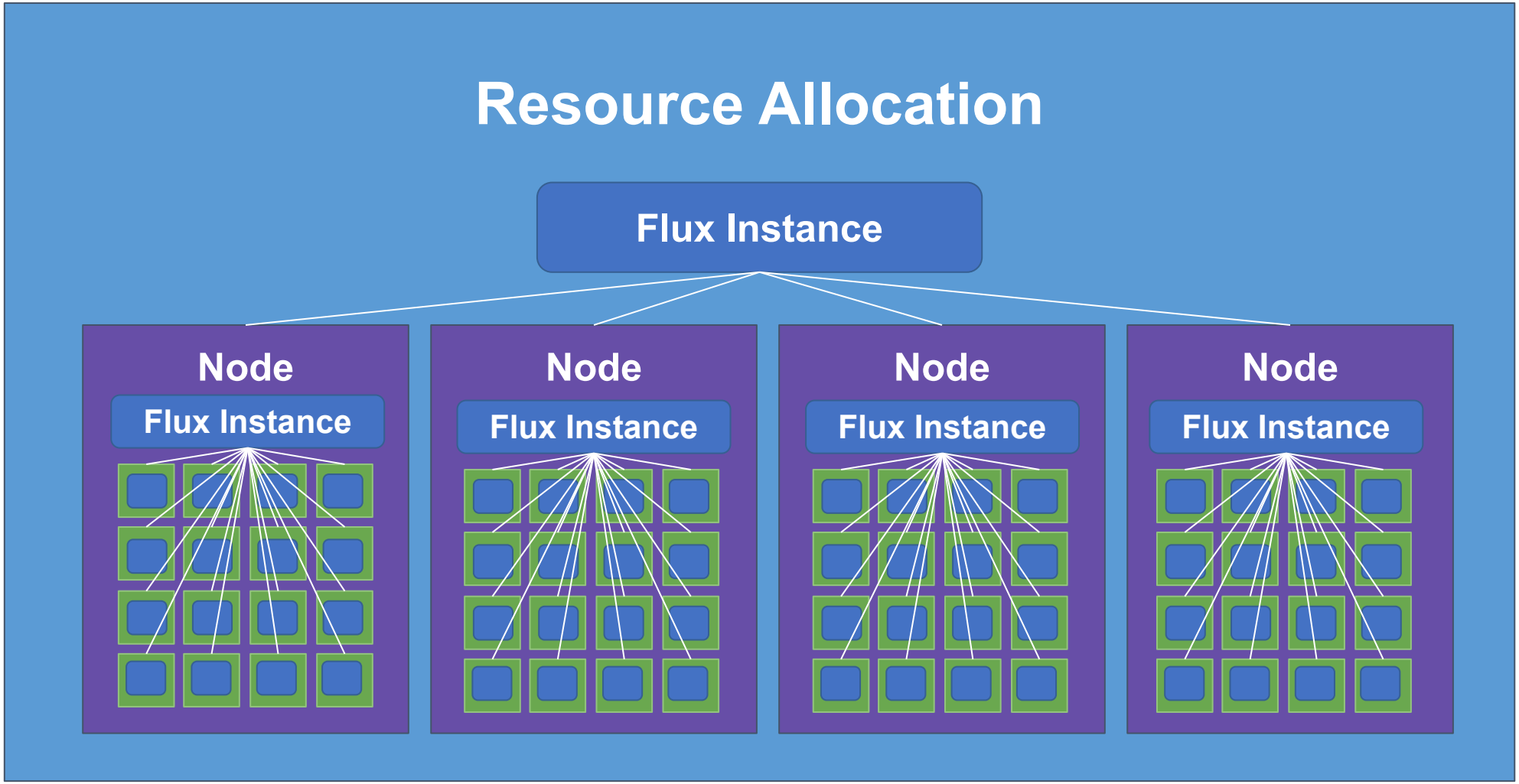


Start a flux instance!

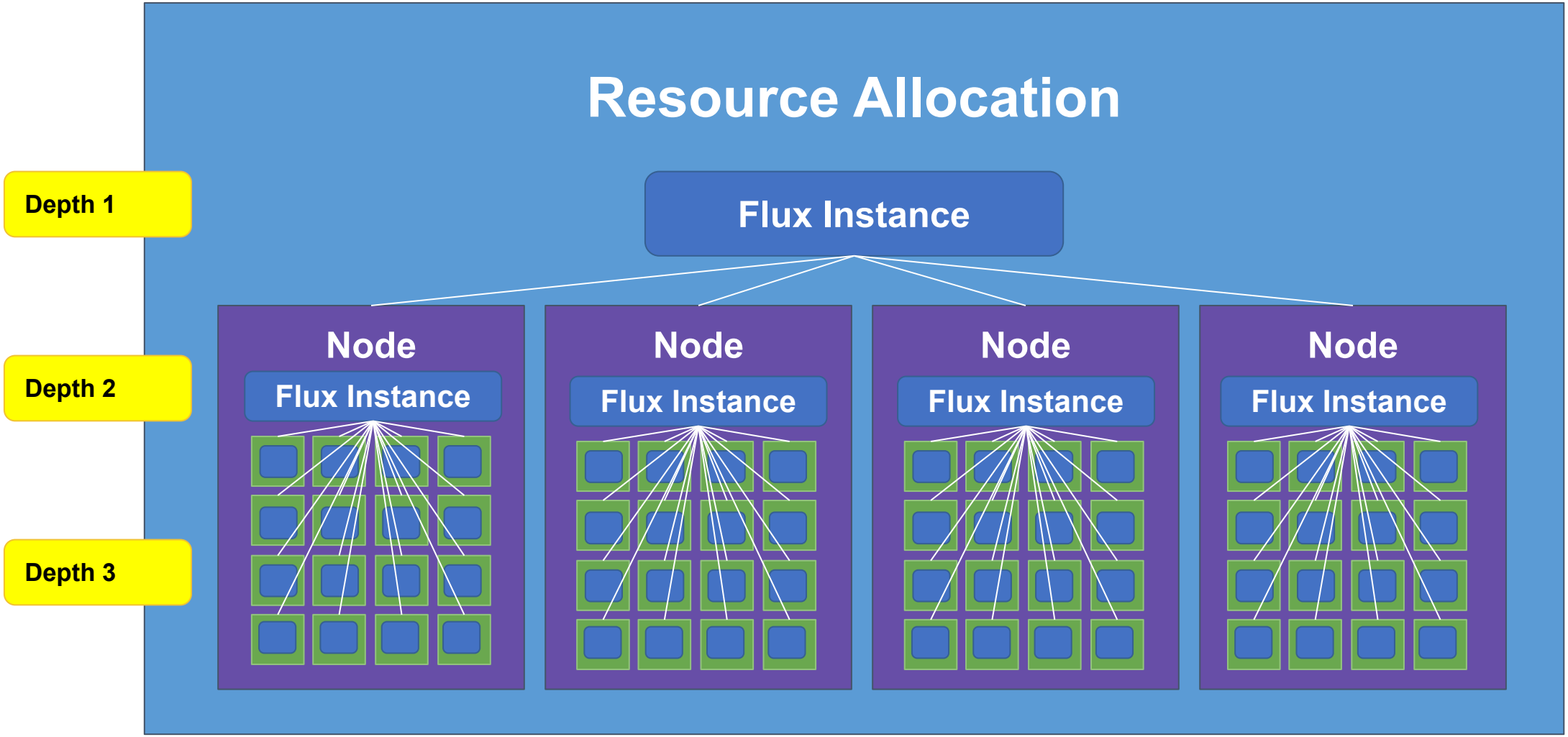
flux start <options> <args> <command>



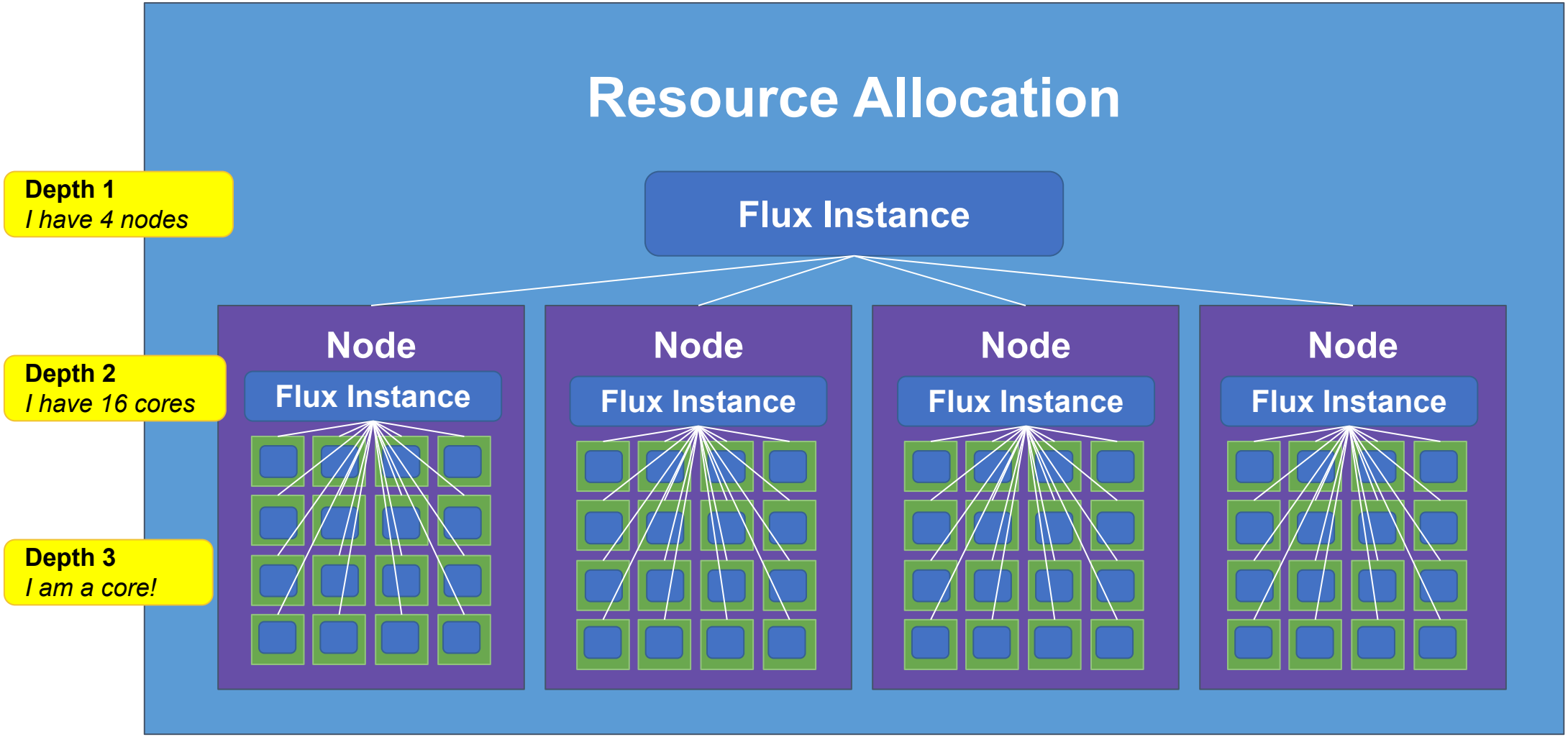
The flux instance sees the resources available!



Flux creates instances of itself to run on sub-resources



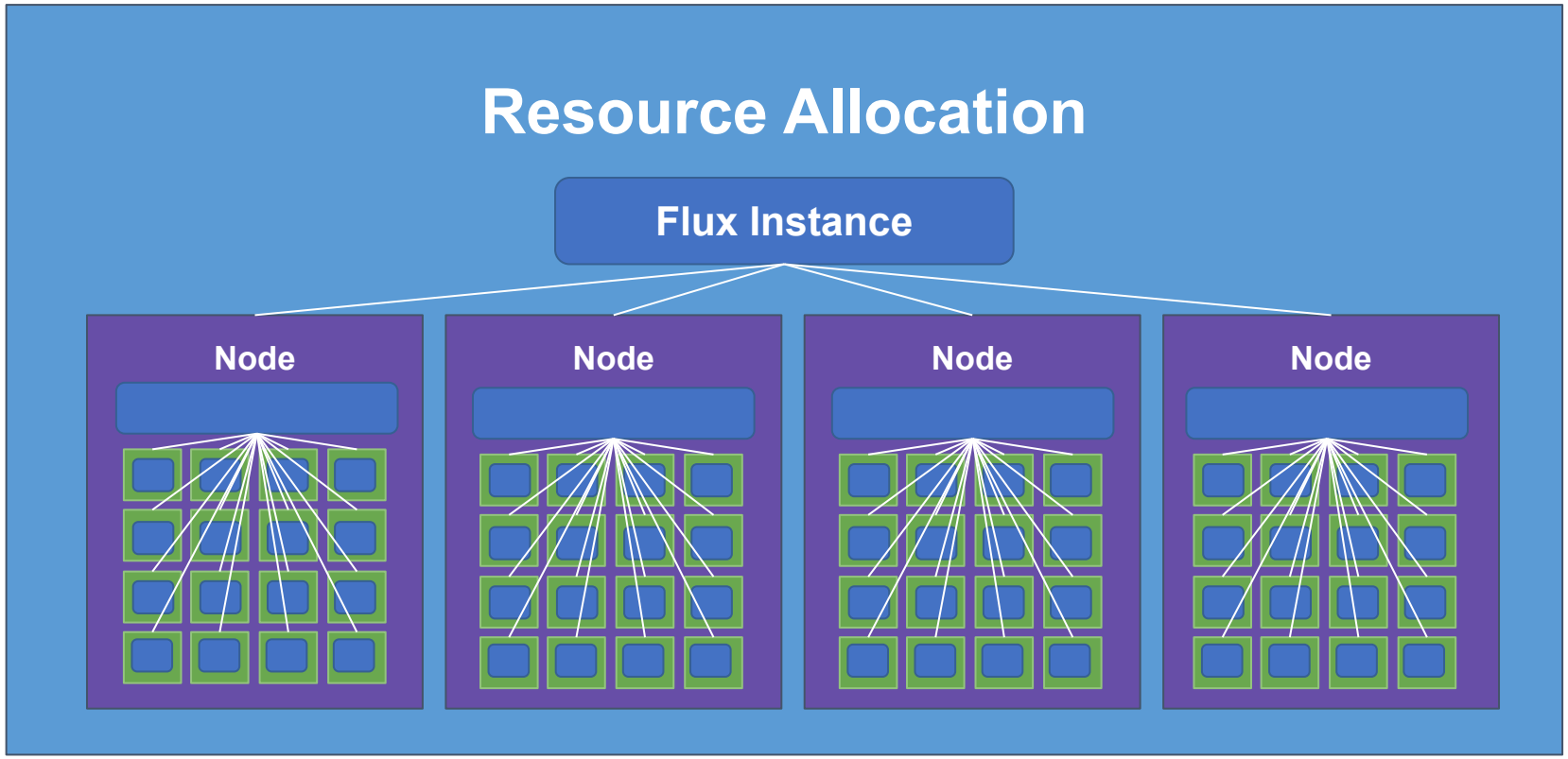
These are different depths in a graph!



Each depth knows about (and validates) its own resources

Flux is really good at:

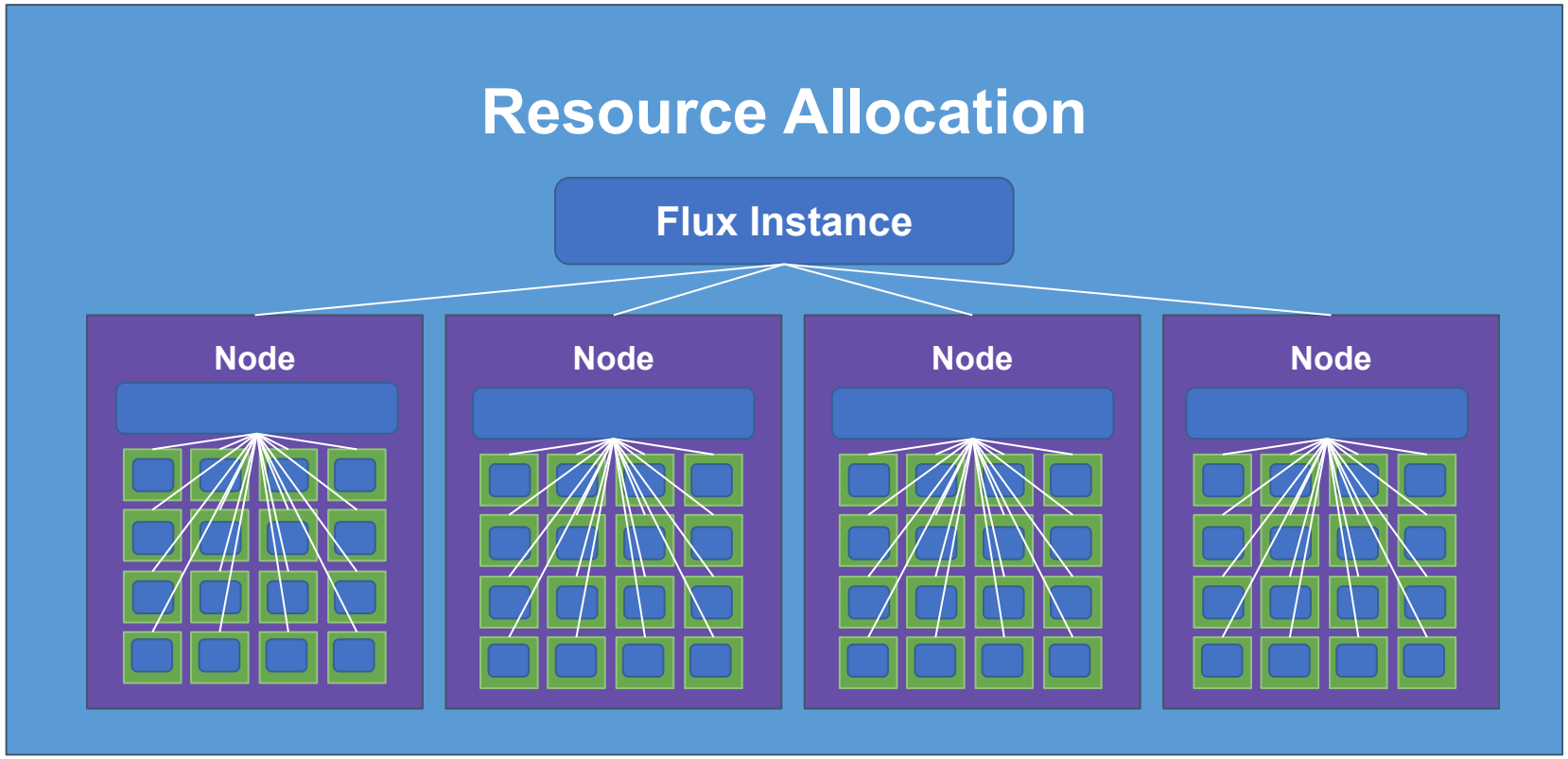
Portability



Flux is really good at:

Portability

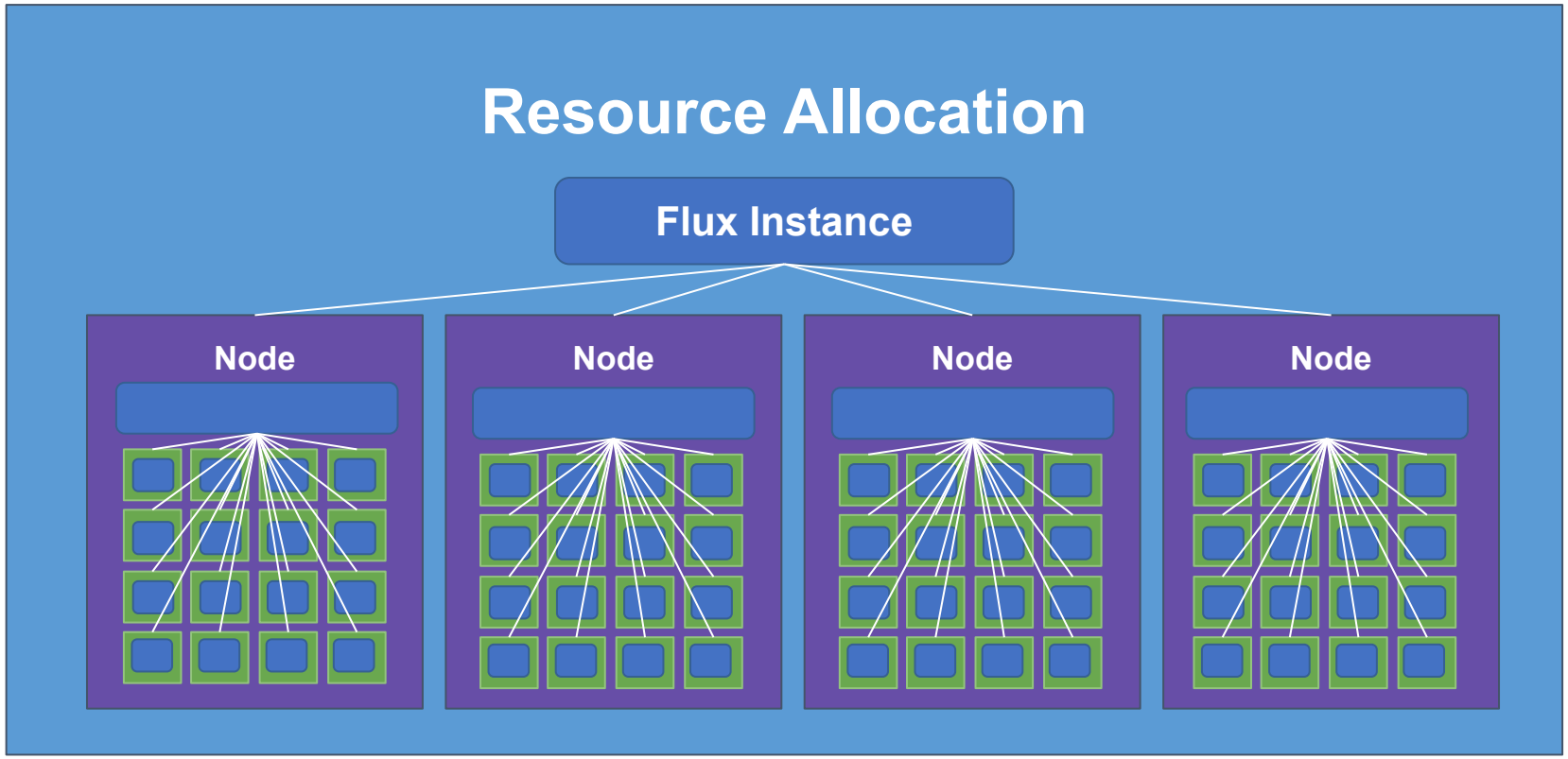
- *Kubernetes*



Flux is really good at:

Portability

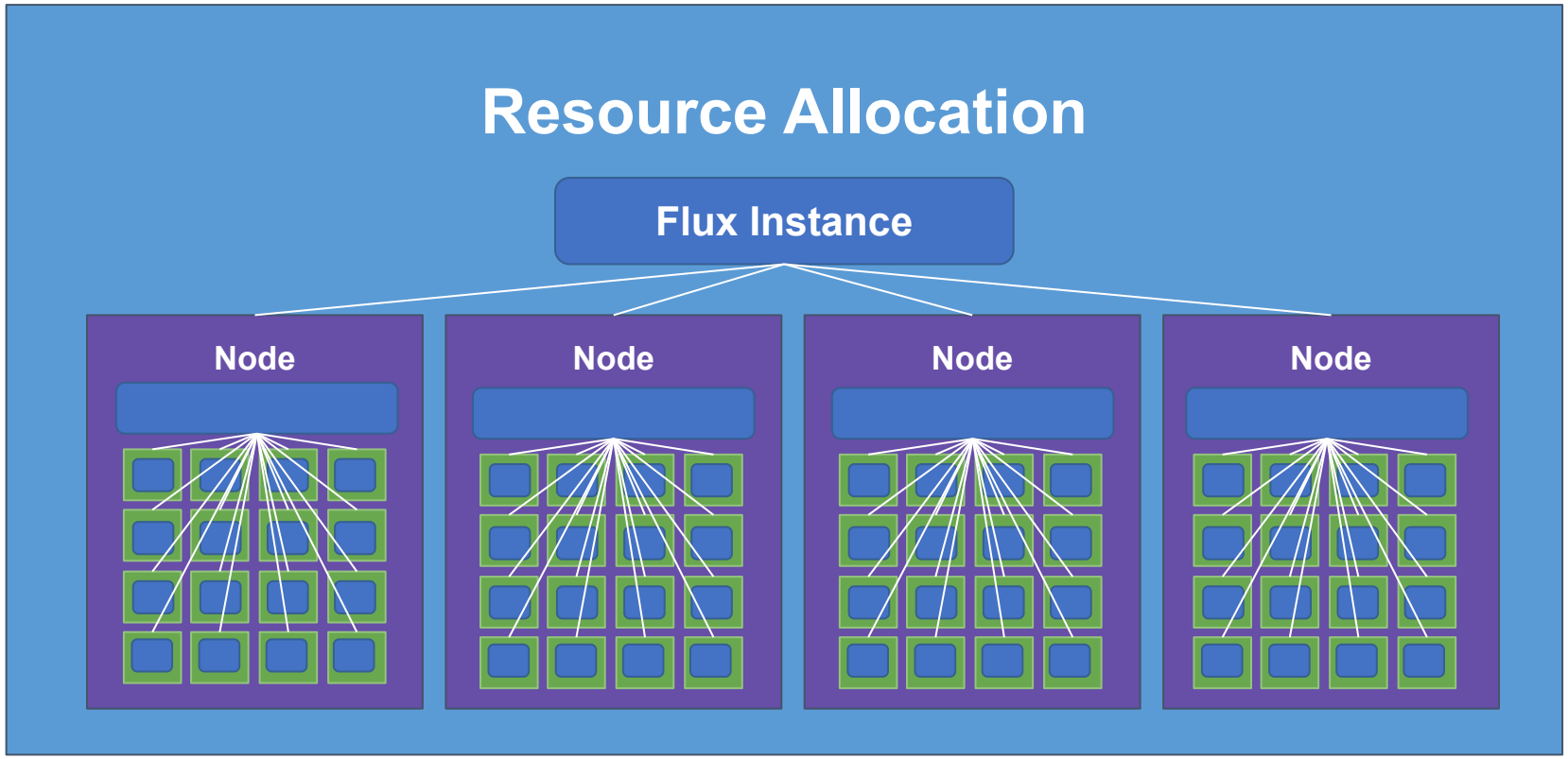
- *Kubernetes*
- *HPC Cluster*



Flux is really good at:

Portability

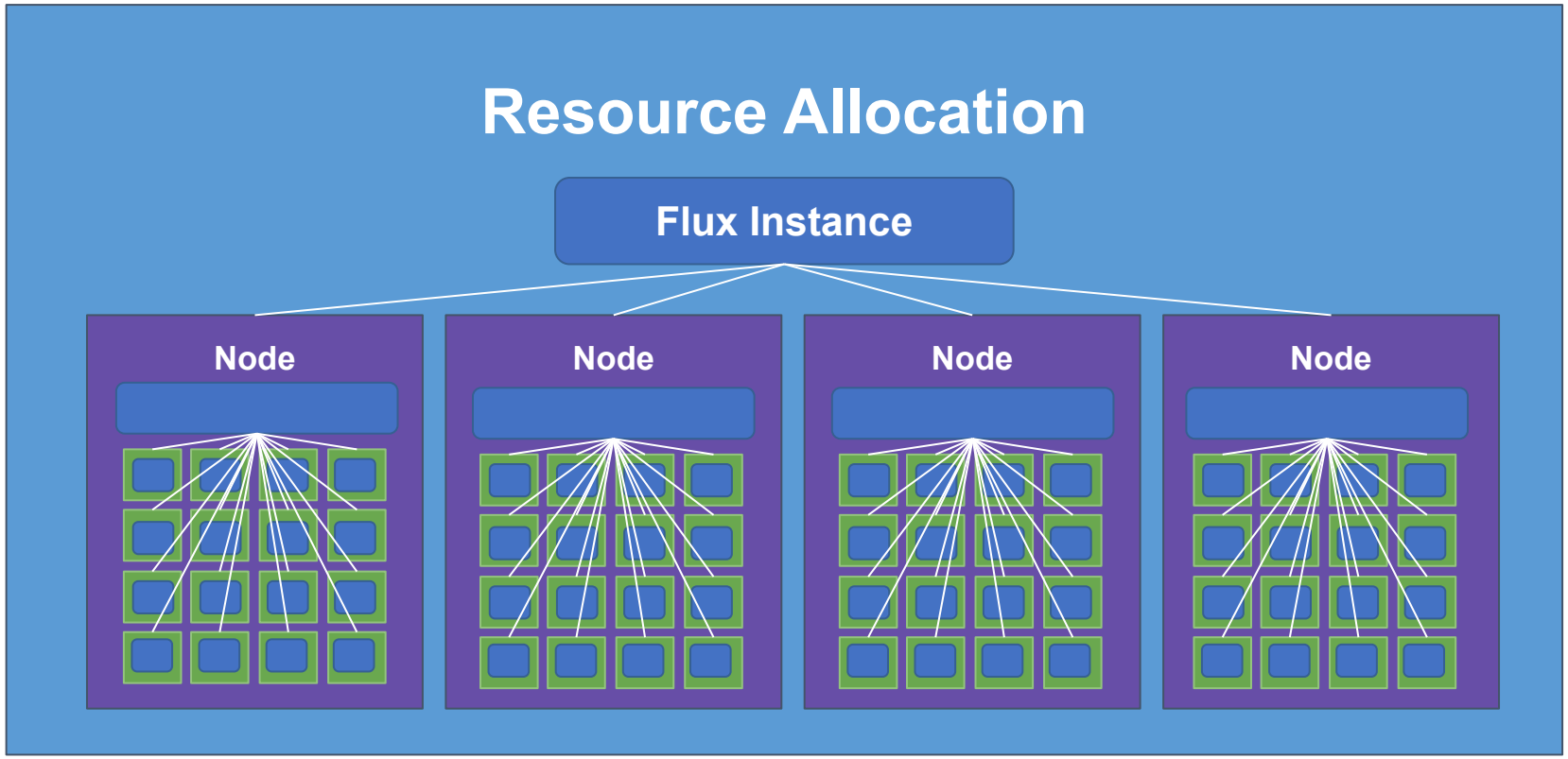
- *Kubernetes*
- *HPC Cluster*
- *SLURM Allocation*



Flux is really good at:

Portability

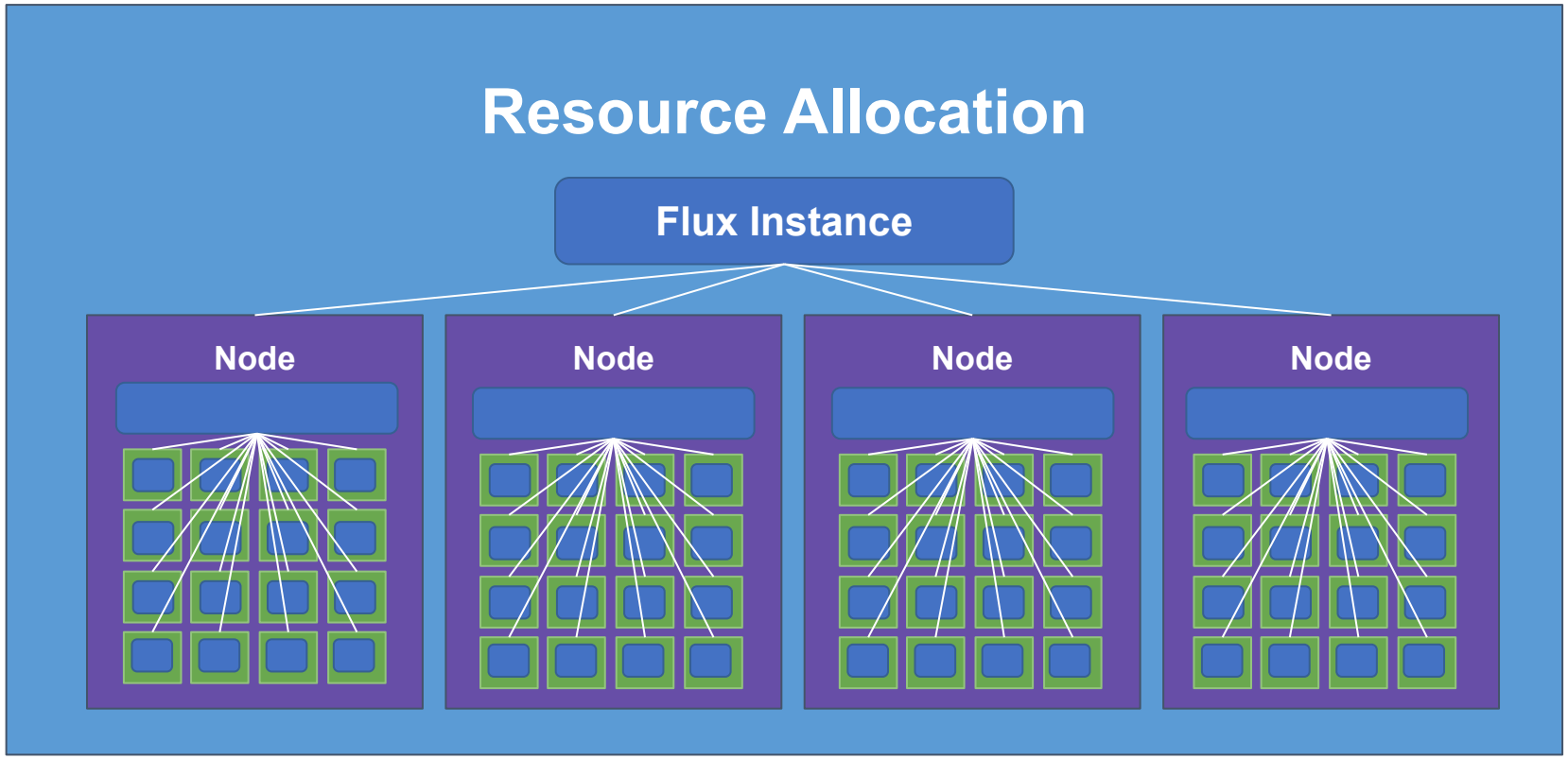
- *Kubernetes*
- *HPC Cluster*
- *SLURM Allocation*
- *Remotely*



Flux is really good at:

Portability

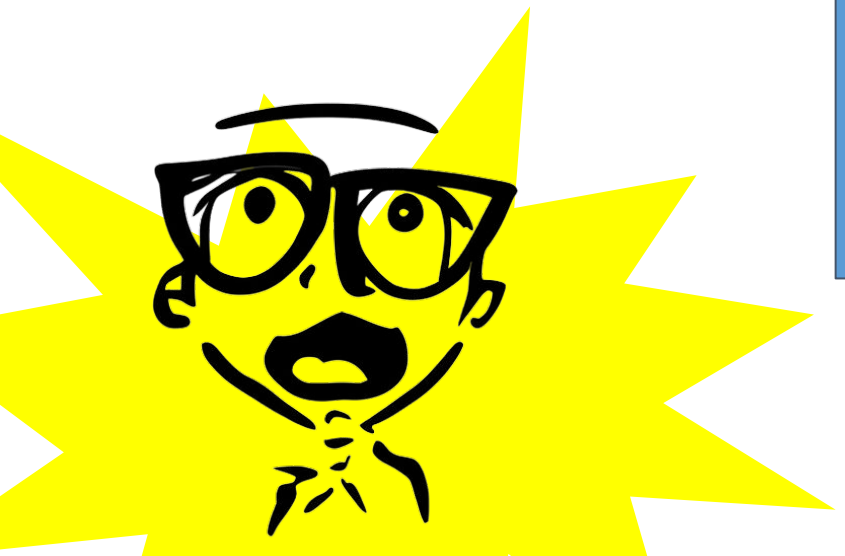
- *Kubernetes*
- *HPC Cluster*
- *SLURM Allocation*
- *Remotely*
- *Container*

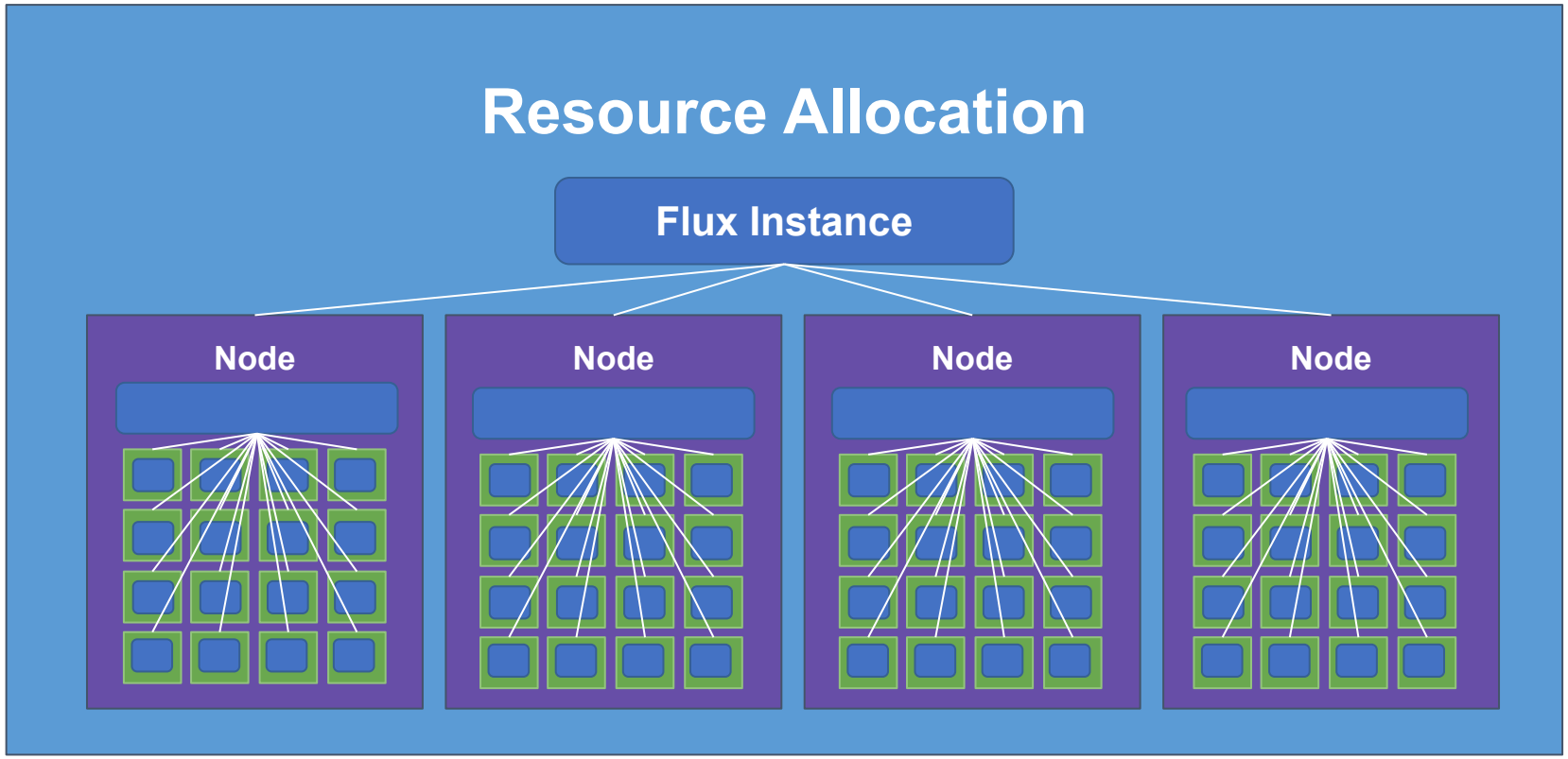


Flux is really good at:

Portability

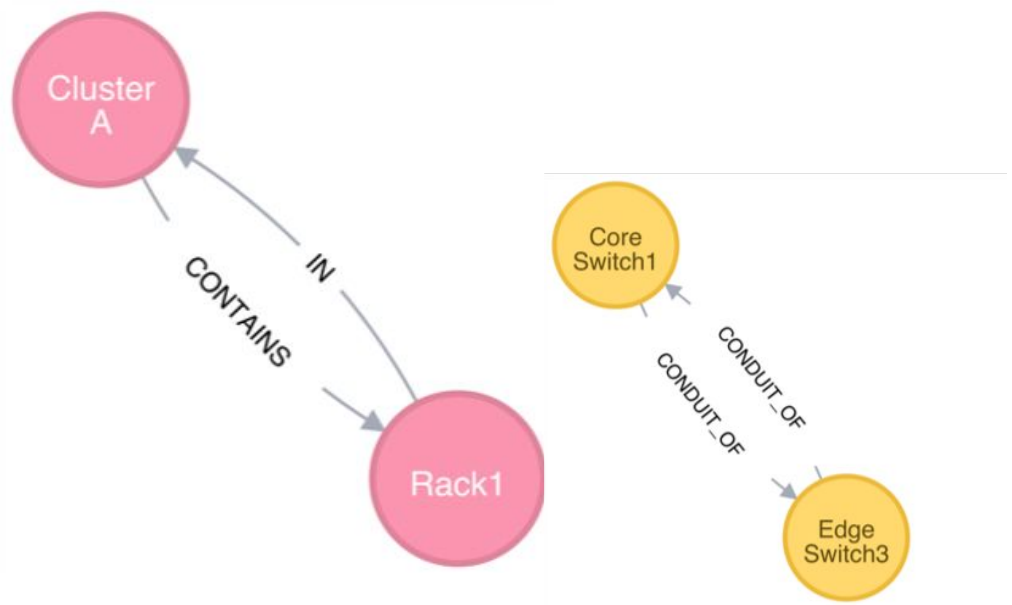
- *HPC Cluster*
- *Kubernetes*
- *SLURM Allocation*
- *Remotely*
- *Container*





Flux is really good at:

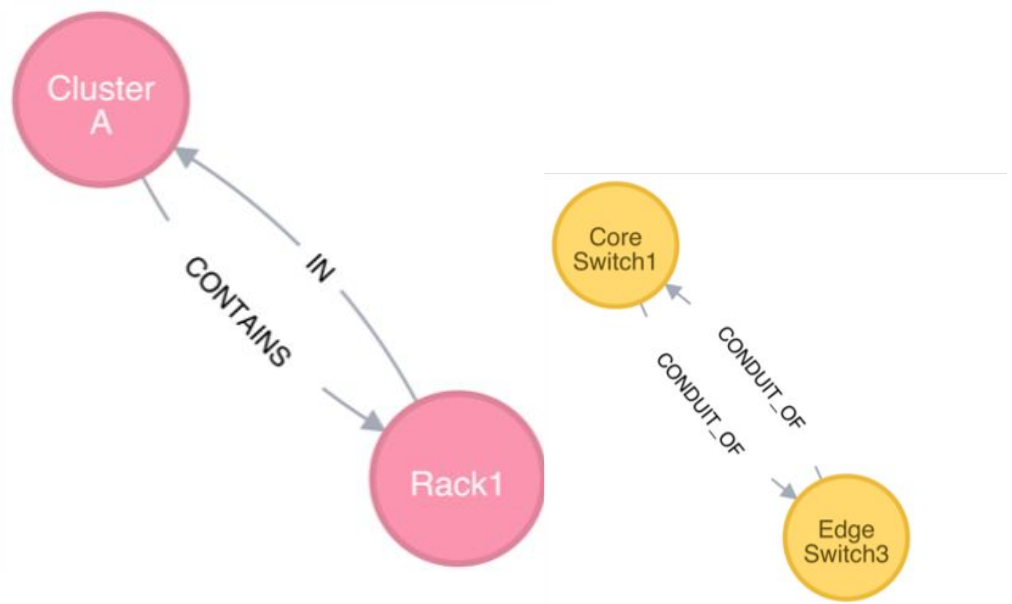
Portability
Co-scheduling



Flux is really good at:

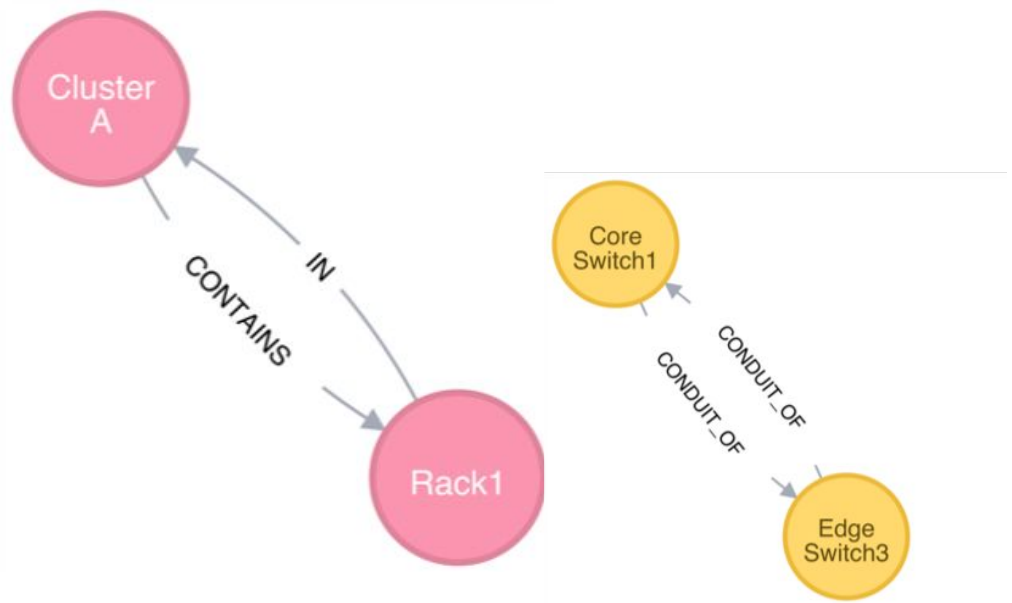
Portability
Co-scheduling

I know the node topology!



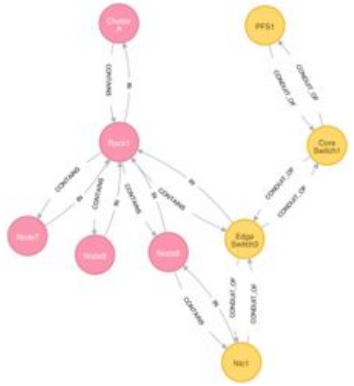
Flux is really good at:

Portability
Co-scheduling



I know the node topology!

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

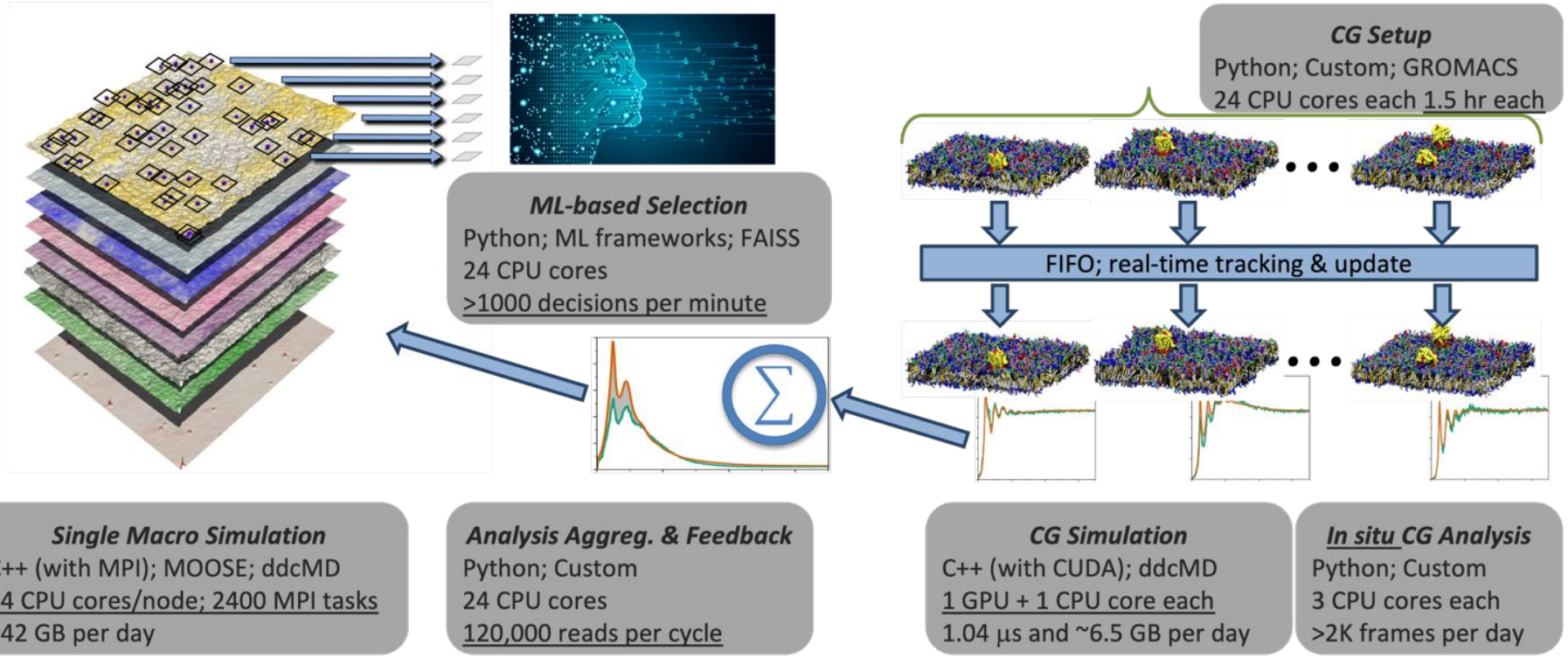


Flux is really good at:

Portability
Co-scheduling
Jobs Coordination

Multiscale Machine-Learned
Modeling Infrastructure

36,000 tasks
176,000 cores
16,000 gpus



*MuMMi SC21 Paper


flux
HPC Land


flux
HPC Land
Cloud Land


flux
HPC Land
Cloud Land


flux
HPC Land
Cloud Land

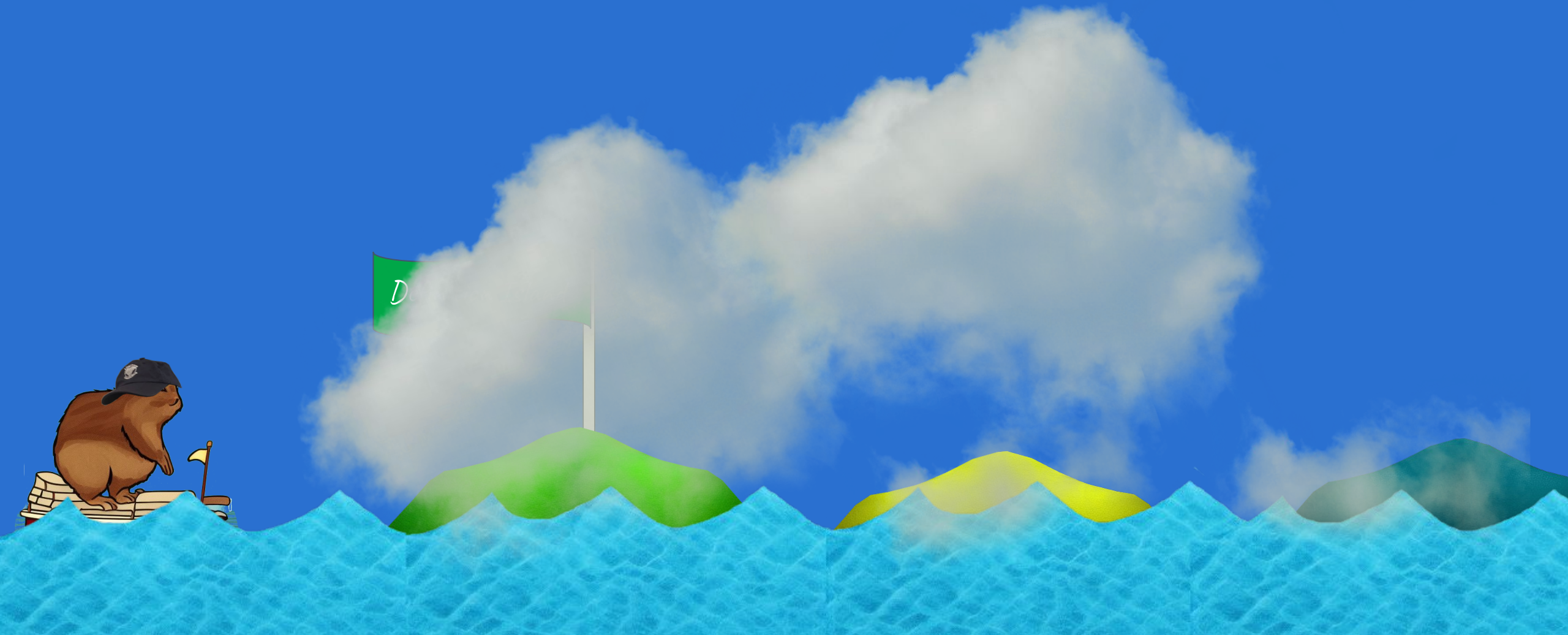
THE OPERATOR

coming to K8s near you...

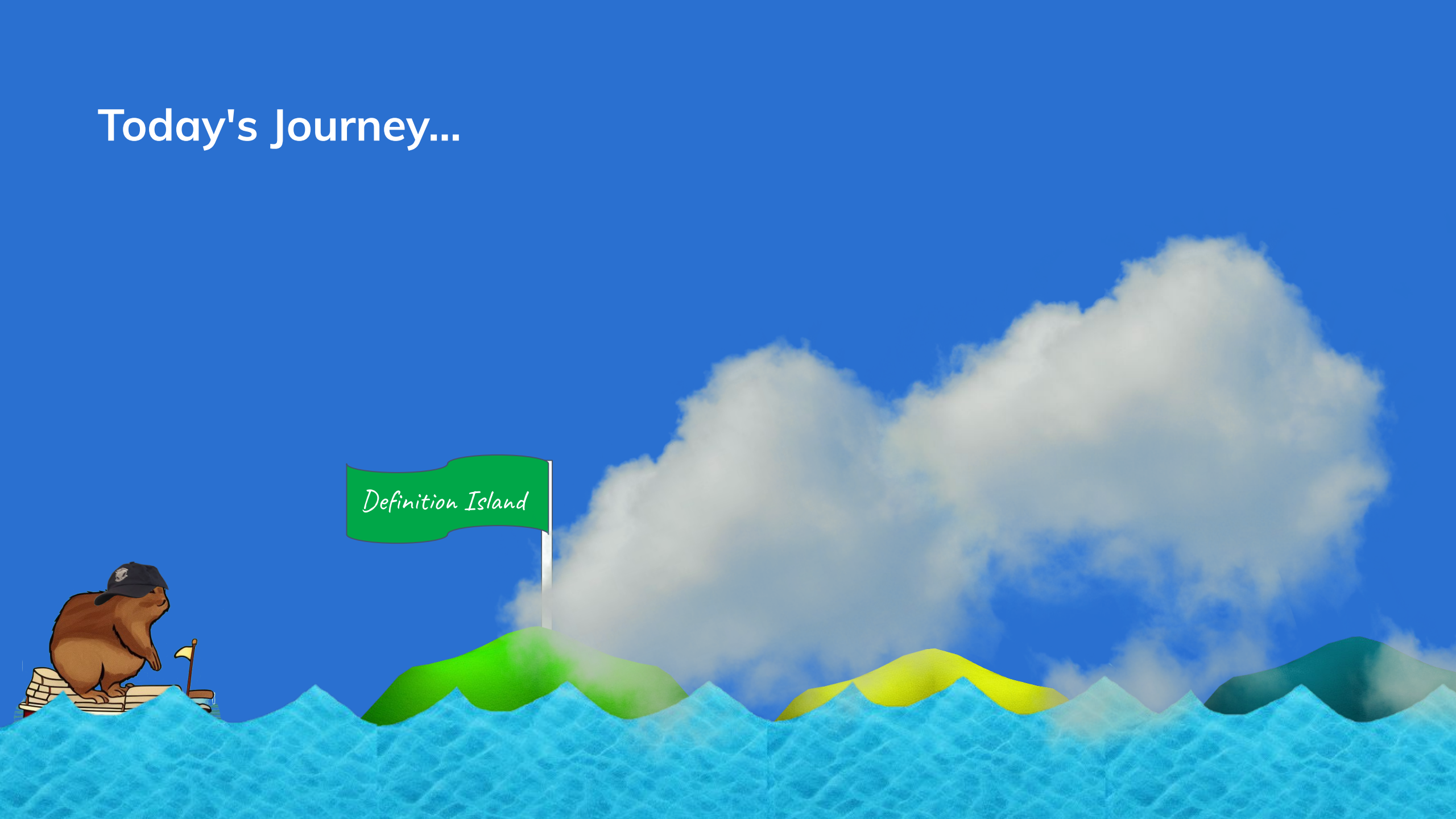
THE OPERATOR

coming to K8s near you...

Today's Journey...



Today's Journey...



Definitions

Operator

A controller for a Kubernetes cluster to manage objects.

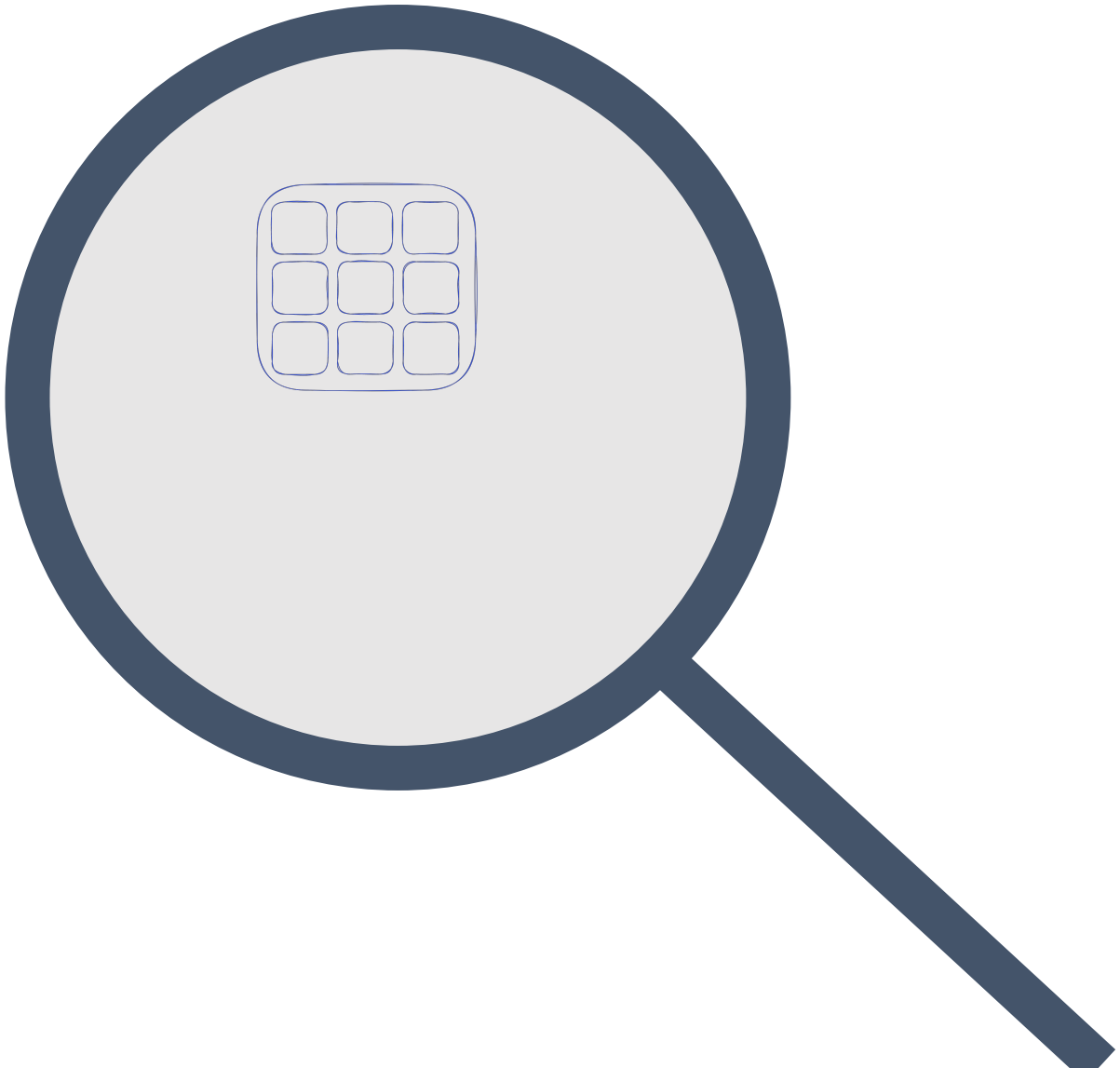
Flux Operator

A controller that allows us to setup a Flux instance to run across pods in a Kubernetes cluster.

Flux Operator

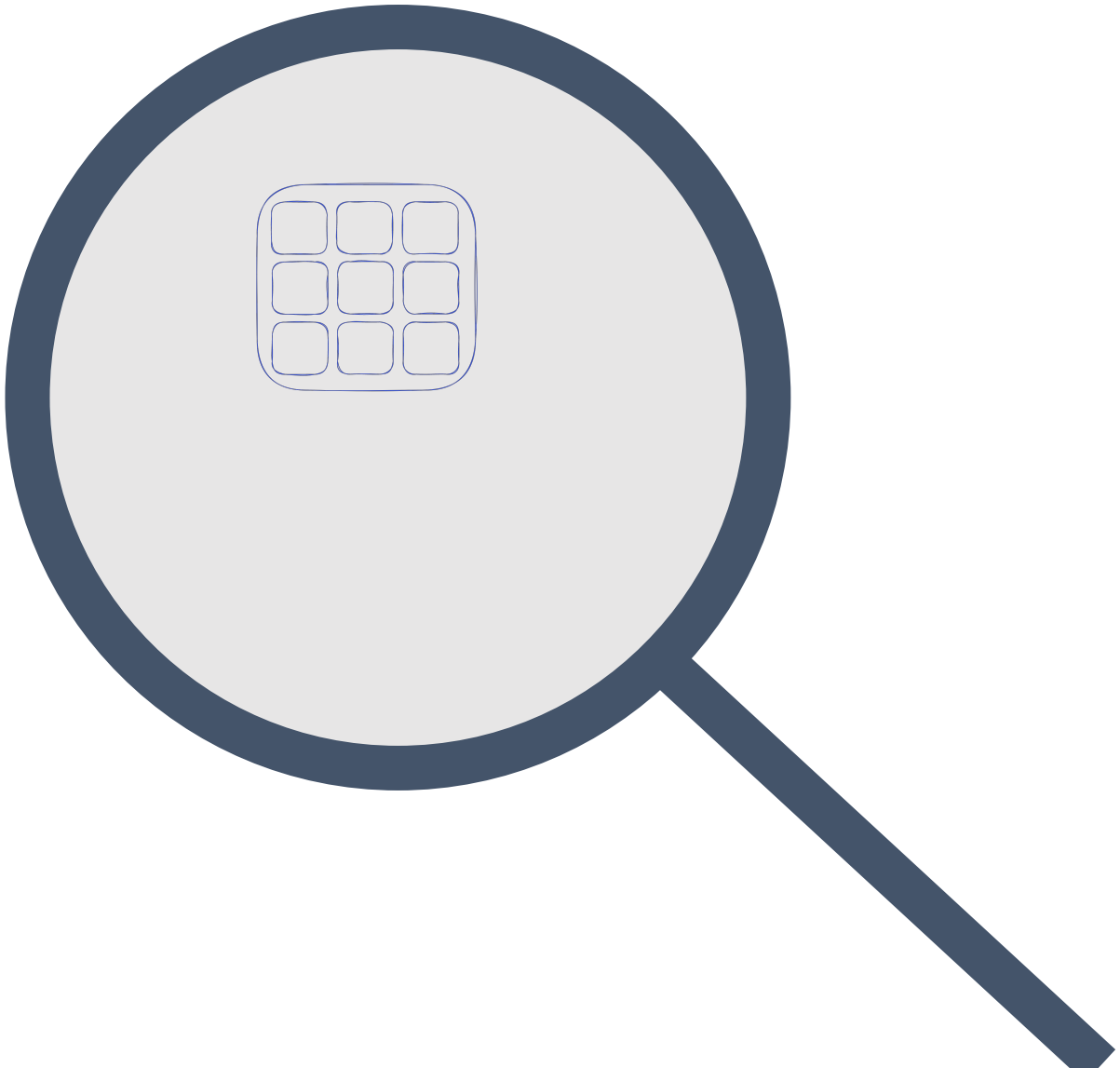
A controller that allows us to setup a Flux instance to run across pods in a Kubernetes cluster.

MiniCluster



MiniCluster

"Is this a cluster for ants?!"



MiniCluster

A set of duplicate pods created by an indexed job in Kubernetes configured to run a Flux instance.

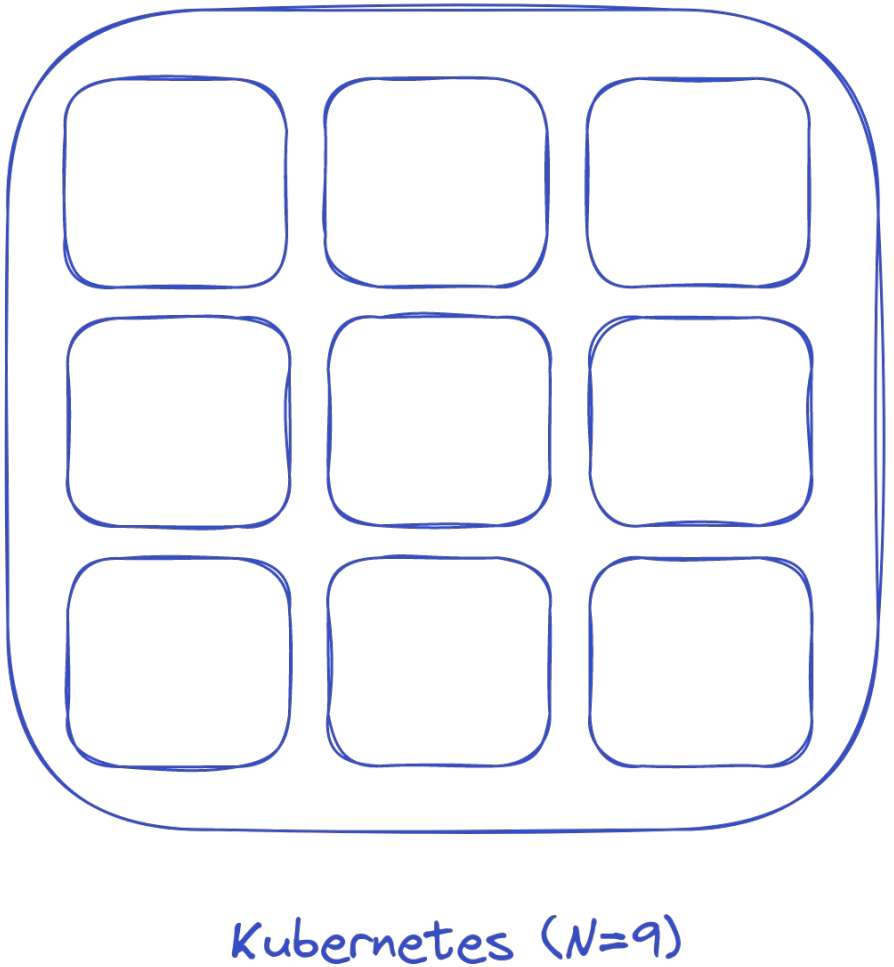
MiniCluster

A set of duplicate pods created by an indexed job in Kubernetes configured to run a Flux instance.

"An entire HPC cluster running in Kubernetes I can control!"

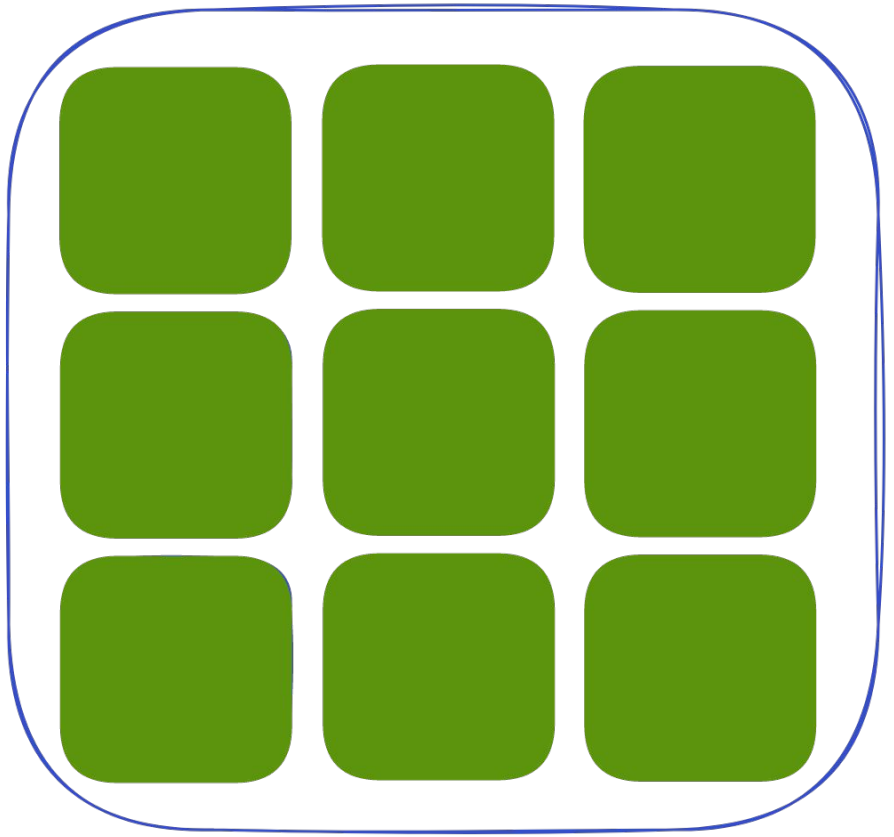
MiniCluster

A set of duplicate pods created by an indexed job in Kubernetes configured to run a Flux instance.



MiniCluster

A set of duplicate pods created by an indexed job in Kubernetes configured to run a Flux instance.

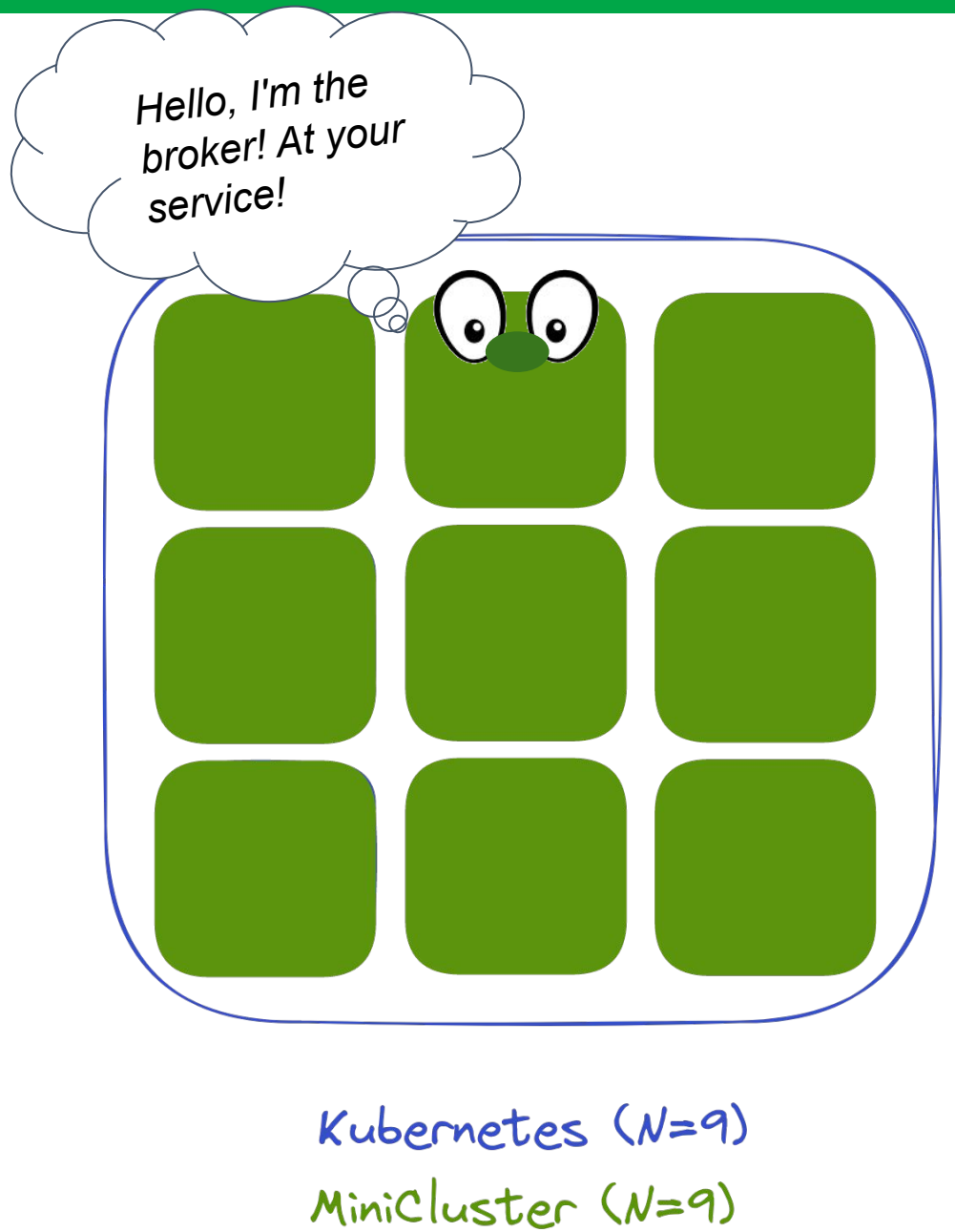


Kubernetes (N=9)

MiniCluster (N=9)

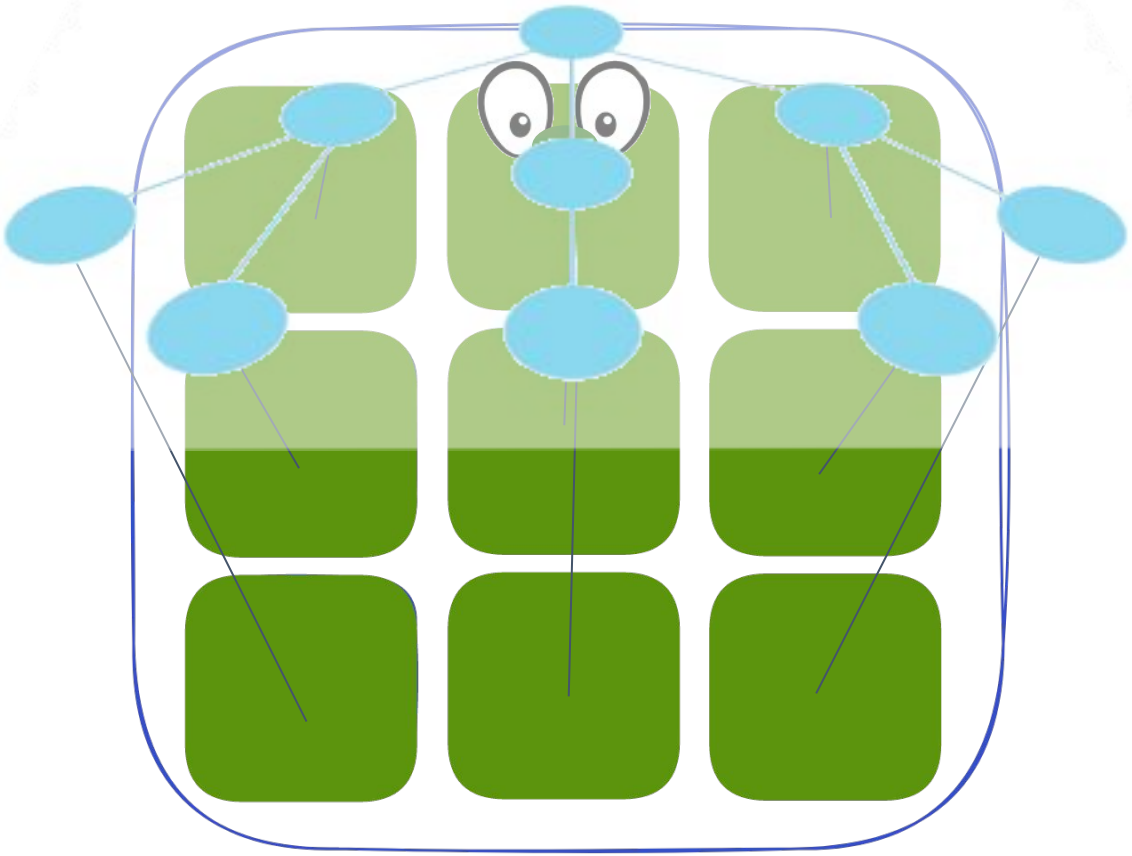
MiniCluster

- Index 0 "broker" orchestrates job



MiniCluster

- Index 0 "broker" orchestrates job
- Communication via tree-based overlay network

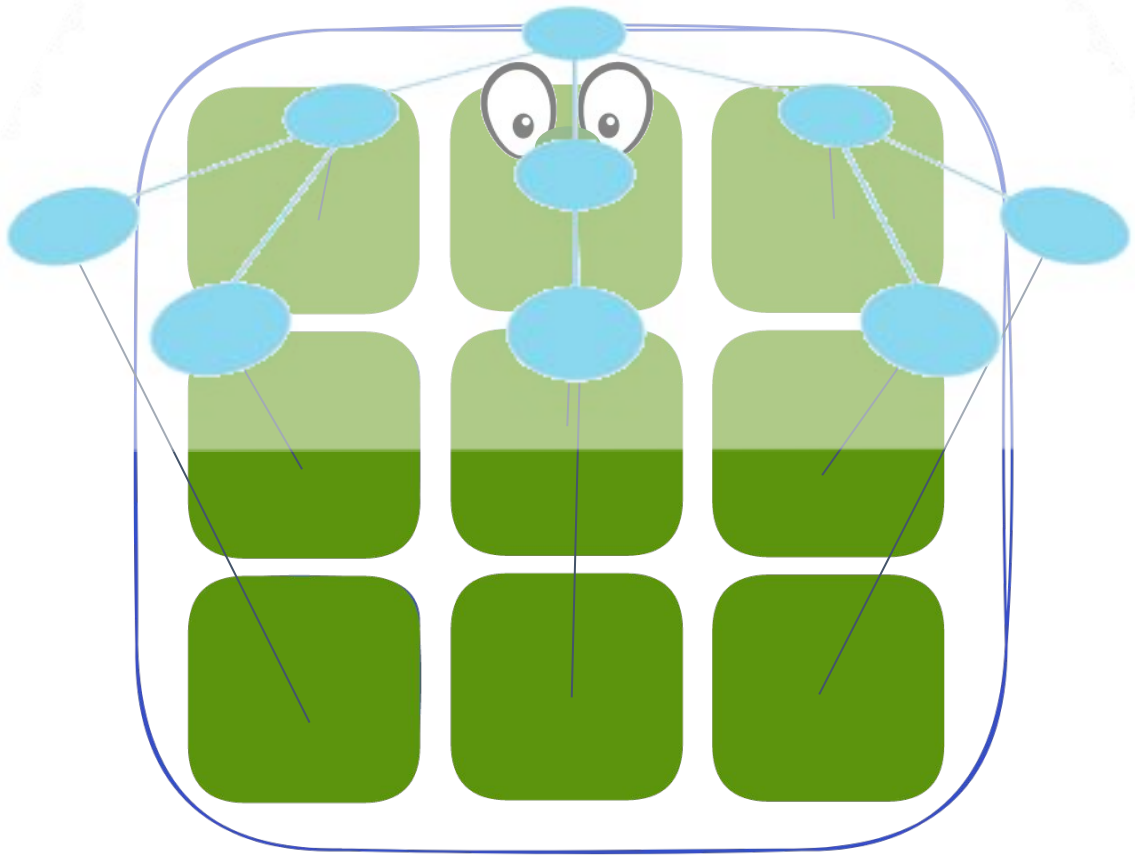


Kubernetes (N=9)

MiniCluster (N=9)

MiniCluster

- Index 0 "broker" orchestrates job
- Communication via tree-based overlay network
- Supports batch jobs, queuing



Kubernetes (N=9)

MiniCluster (N=9)

How do I submit a job?


```
flux submit...
```

HPC Land

```
flux submit...
```

Cloud Land

```
kubectl apply -f service.yaml
```

HPC Land

```
flux submit...
```

Cloud Land

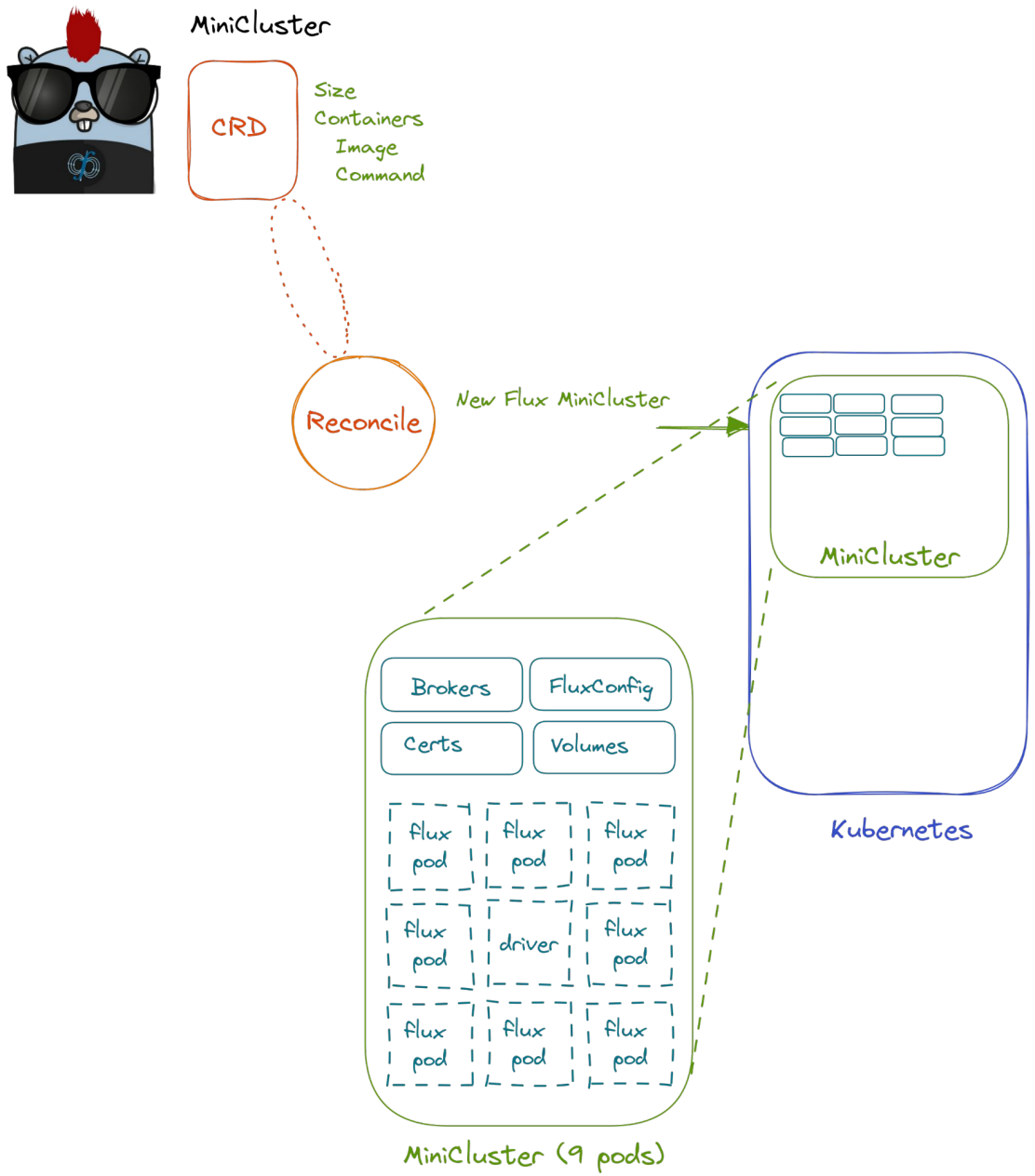
```
kubectl apply -f service.yaml
```

```
kubectl apply -f minicluster.yaml
```

Create a "MiniCluster" with a custom resource definition!

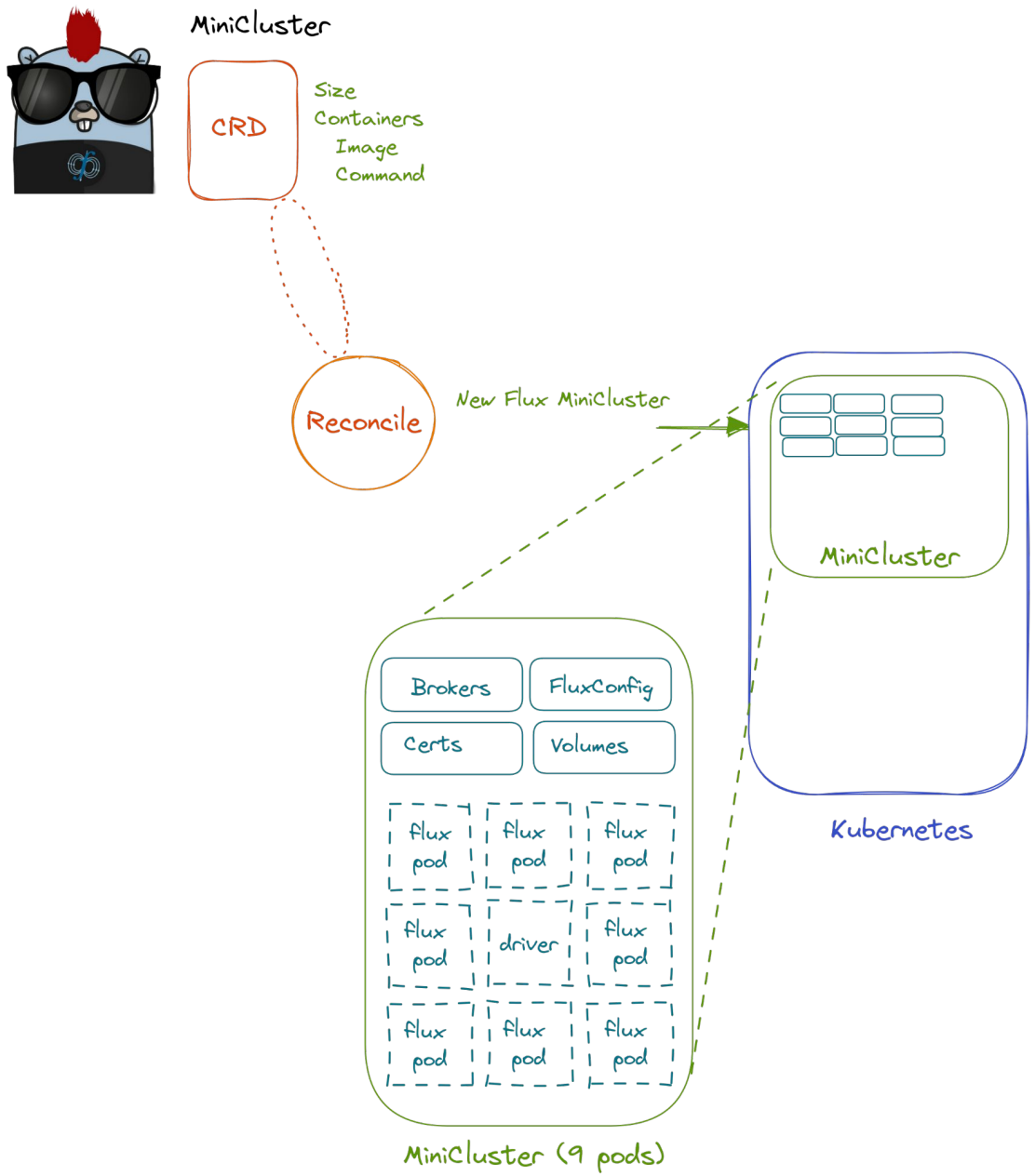
The Flux Operator

- Define your job parameters / containers in the custom resource definition.



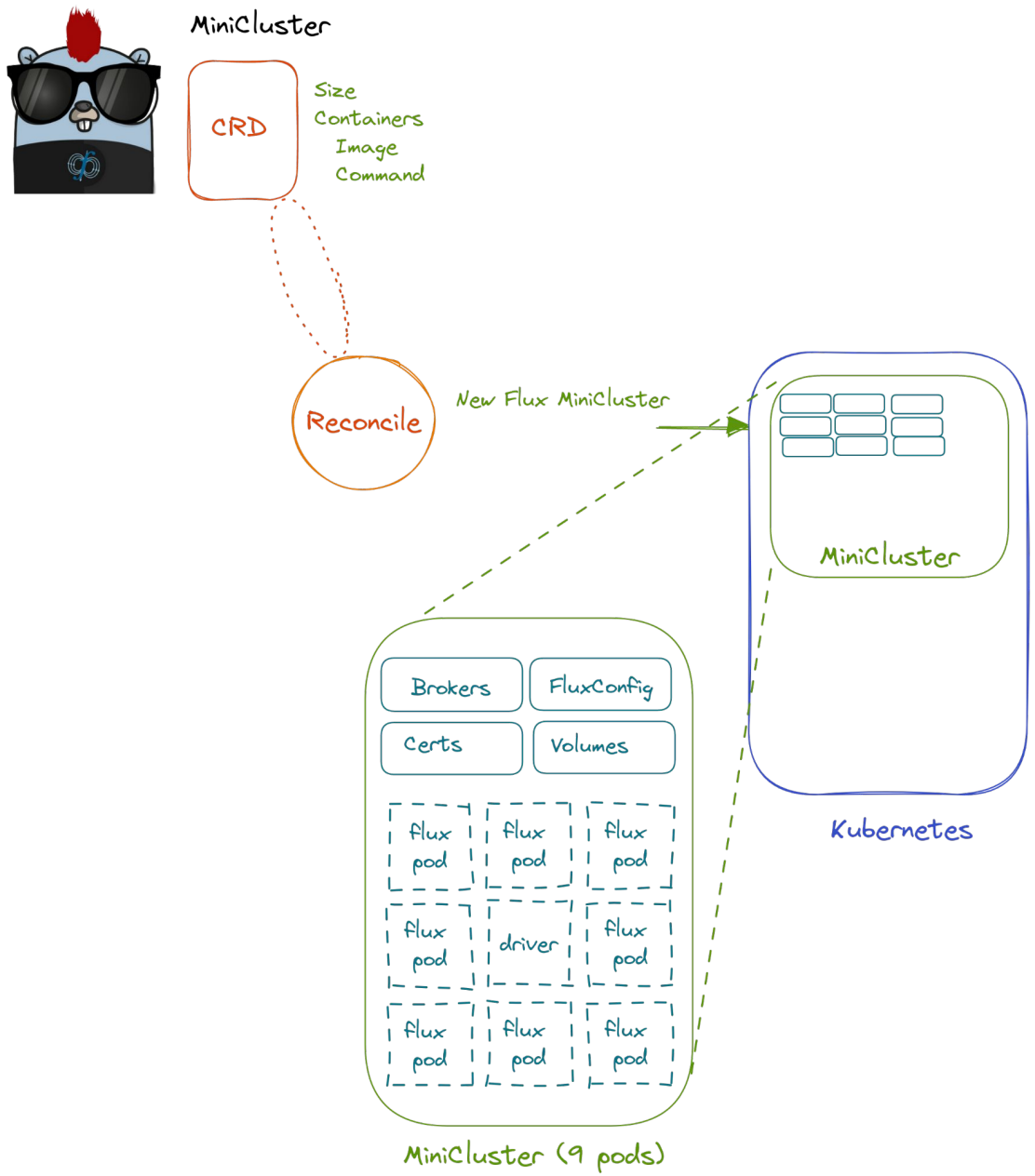
The Flux Operator

- Define your job parameters / containers in the custom resource definition.
- The Flux Operator creates the MiniCluster



The Flux Operator

- Define your job parameters / containers in the custom resource definition.
- The Flux Operator creates the MiniCluster
- The job completes and it cleans up.



I'm ready to make a MiniCluster!


Experiment Empire!

How well does this work?

Compare to the MPI Operator

- Started as part of the KubeFlow project
- Defines an MPIJob custom resource
- A Launcher node coordinates workers via ssh
- Also uses dedicated hostnames and a service for workers
- We use a version modified to scale to > 100 MPI ranks[1]

1. D. J. Milroy *et al.*, "One Step Closer to Converged Computing: Achieving Scalability with Cloud-Native HPC," *2022 IEEE/ACM 4th International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*.

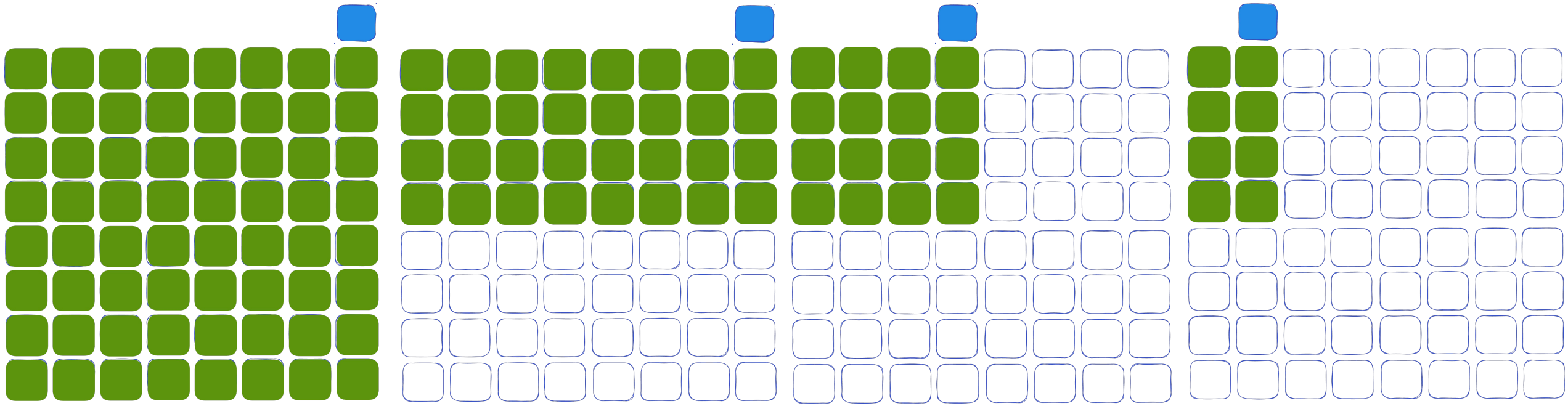
How does the Flux Operator Compare to the MPI Operator?

Experiment Design

- Compare the Flux Operator to the MPI Operator on the same cluster
- **L**arge-scale **A**tomic/**M**olecular **M**assively **P**arallel **S**imulator (LAMMPS)
- Unoptimized container

MPI Operator vs Flux Operator Experiments

Run on the same 65 node cluster (1 node for MPI Operator launcher)



64 pods
6016 ranks

32 pods
3008 ranks

16 pods
1504 ranks

8 pods
752 ranks

Experiment Design

1. Created Kubernetes cluster of size 65
2. For each of the MPI Operator and Flux Operator:
    a. Launch Job/ create MiniCluster for sizes 64, 32, 16, 8
        i. Run LAMMPS x 20
        ii. Record timings
        iii. Save output log

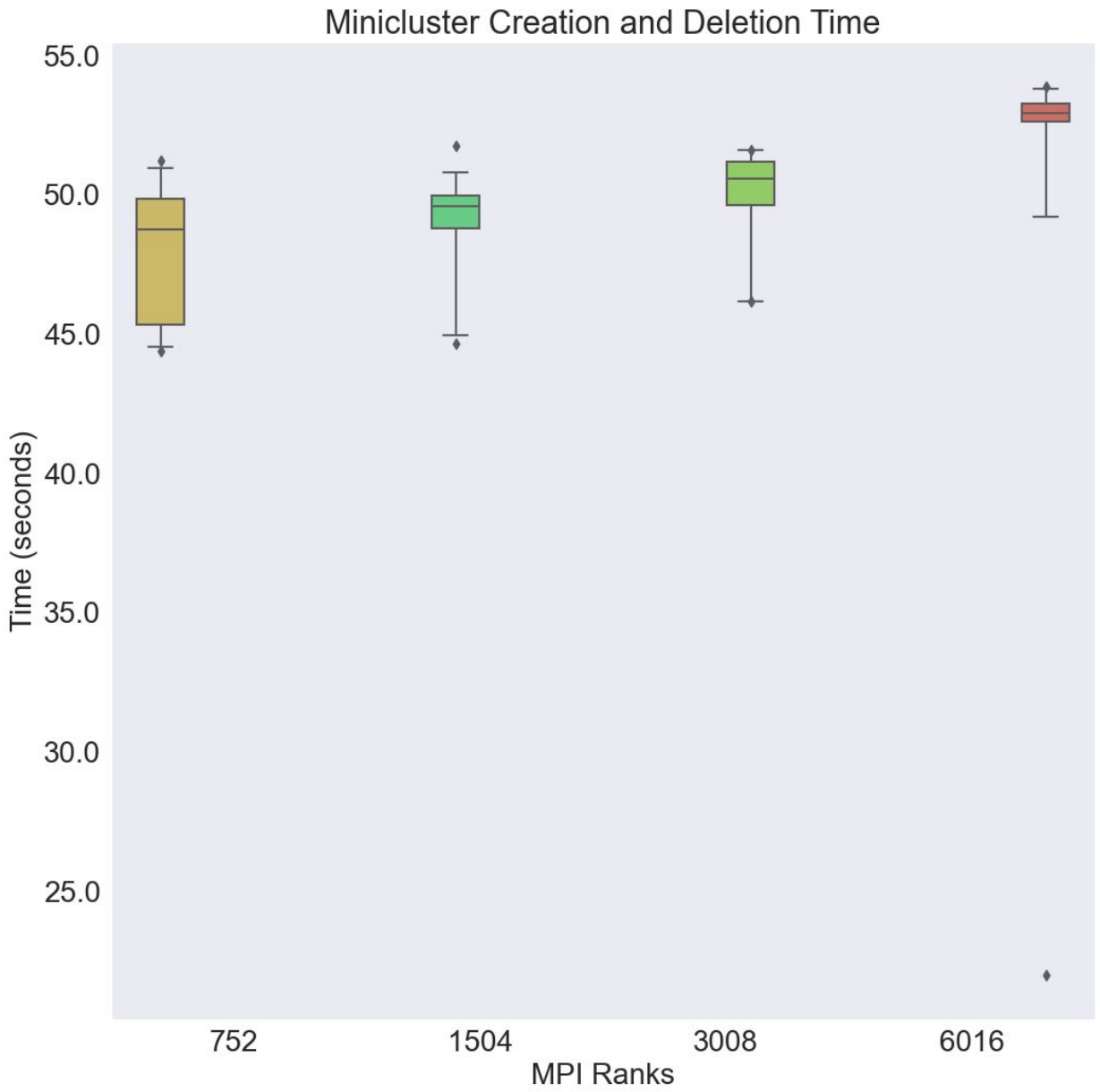
SHOW ME THE RESULTS!

If the Flux Operator MiniCluster is created via an Indexed Job...

If the Flux Operator MiniCluster is created via an Indexed Job...
How well does that scale?

Experiment Results

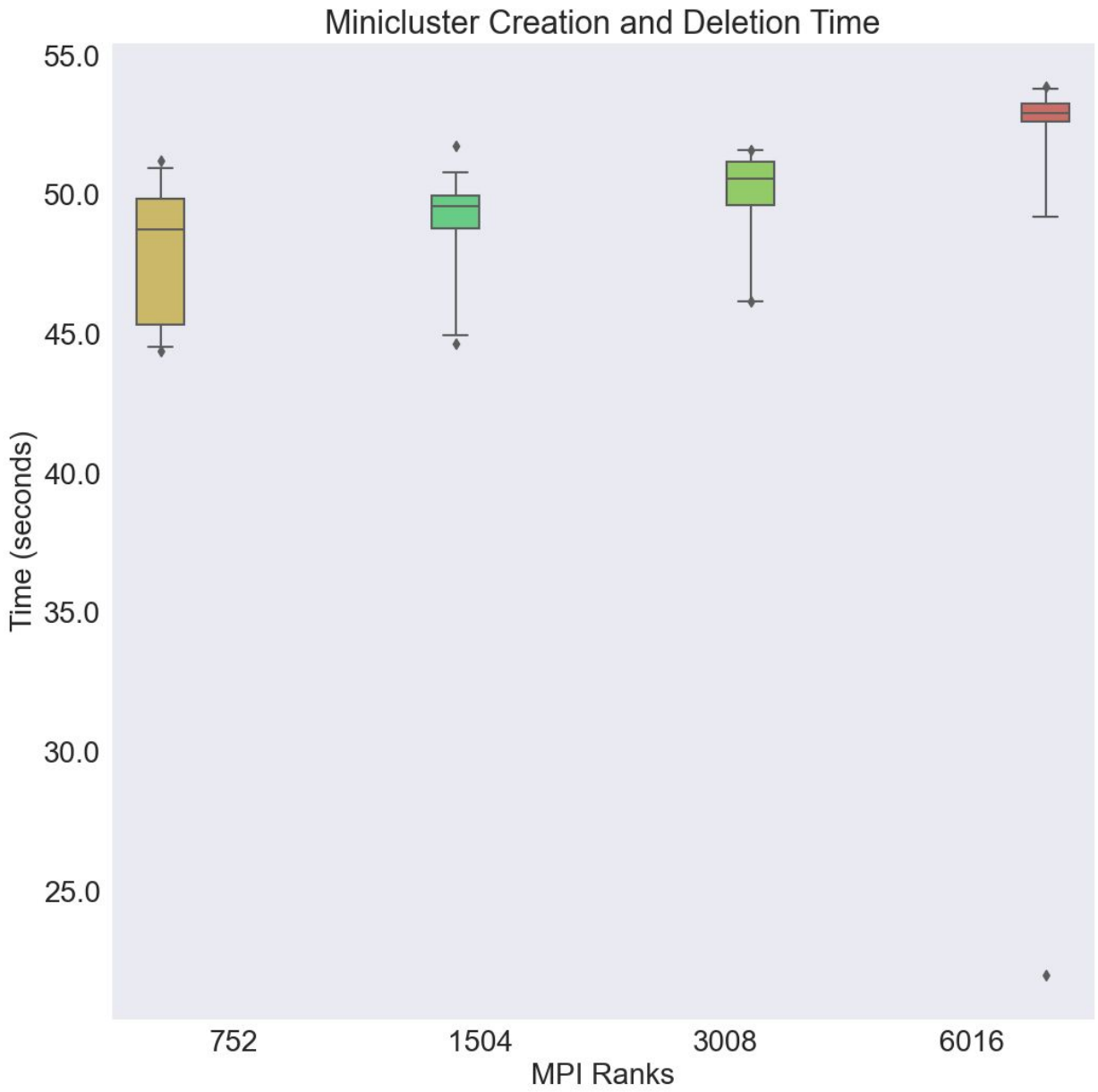
MiniCluster Creation / Deletion Times



Experiment Results

MiniCluster Creation / Deletion Times

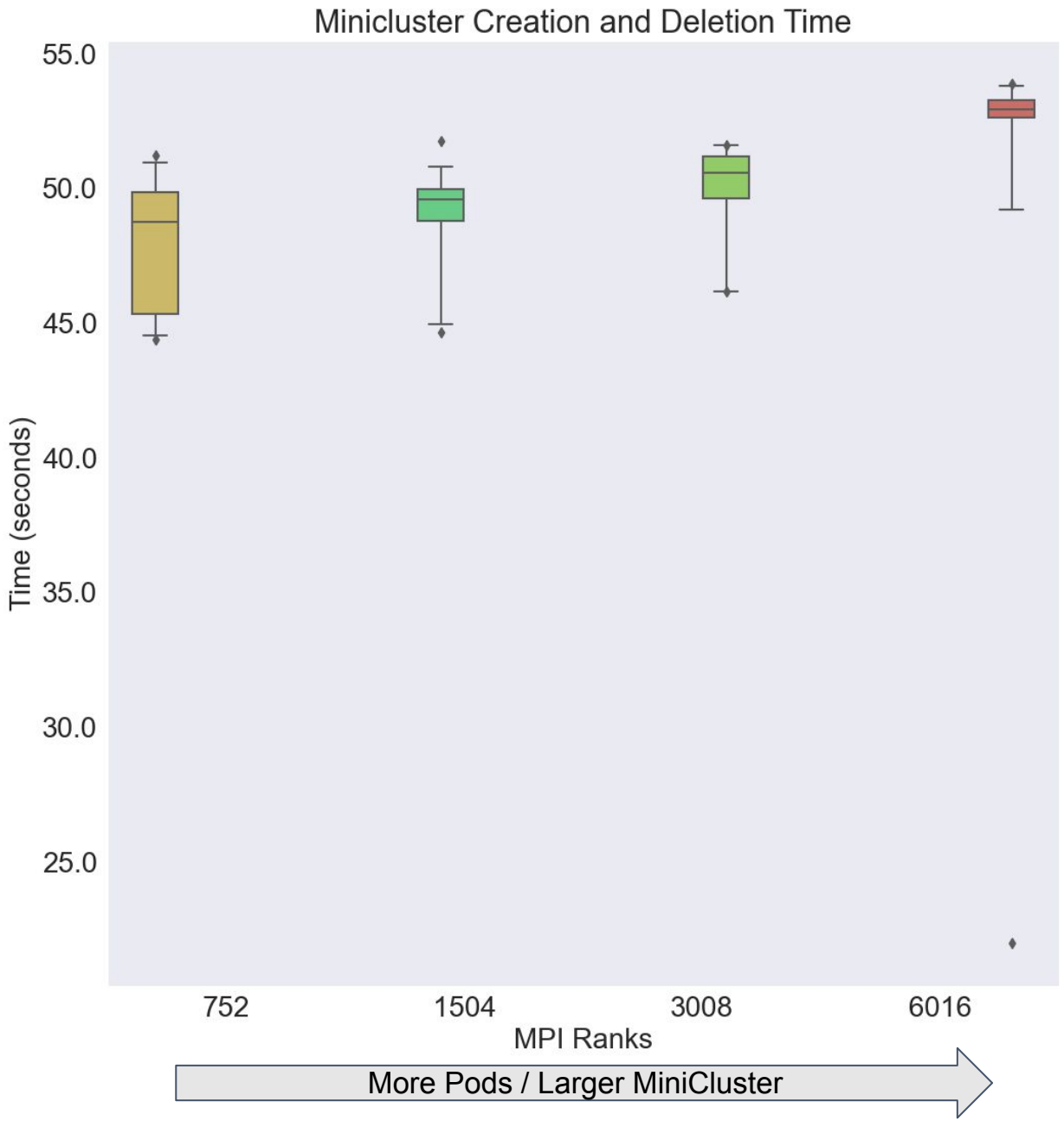
Time is entire up/down minus LAMMPS



Experiment Results

MiniCluster Creation / Deletion Times

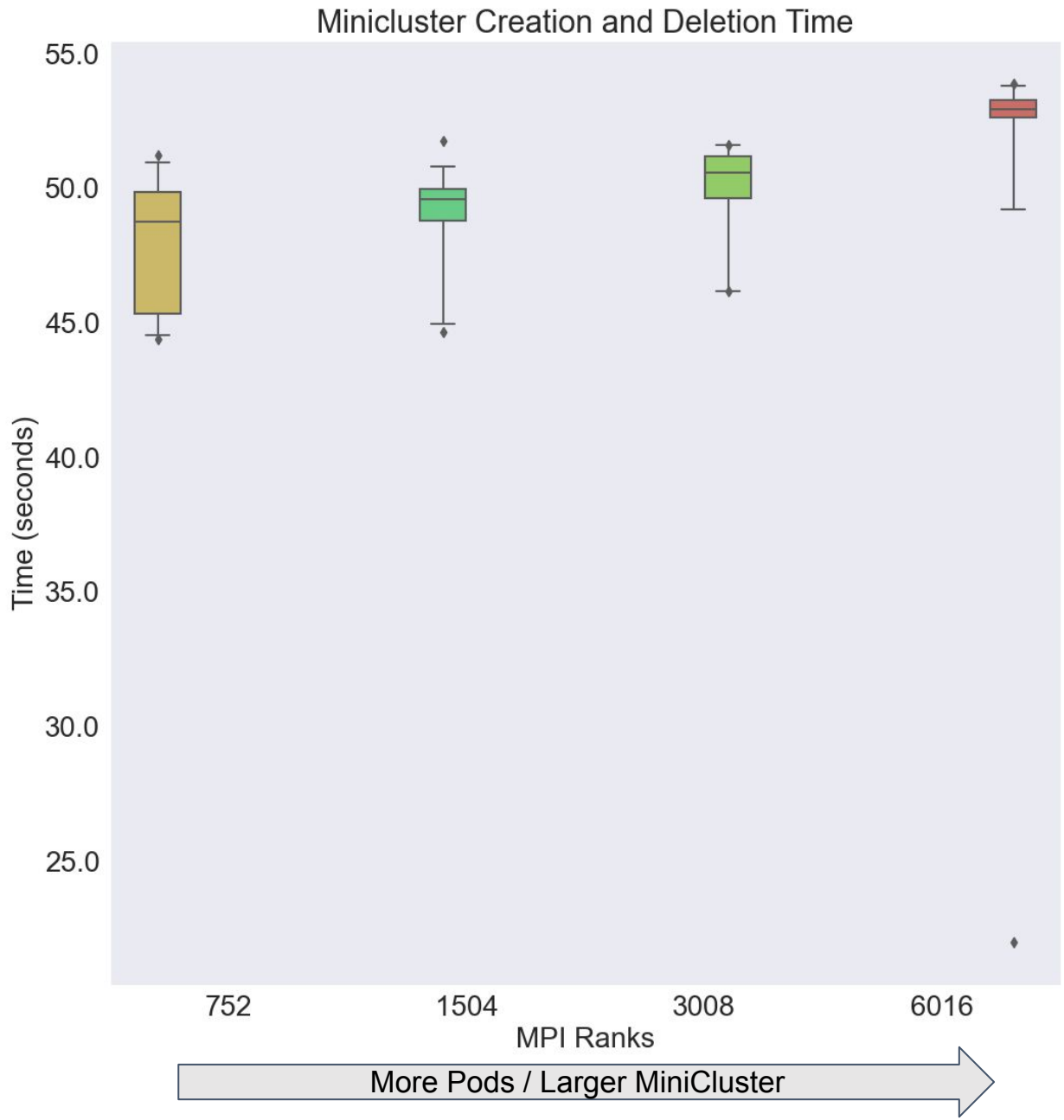
Time is entire up/down minus LAMMPS



Experiment Results

MiniCluster Creation / Deletion Times

Time is entire up/down minus LAMMPS
This scales nicely!



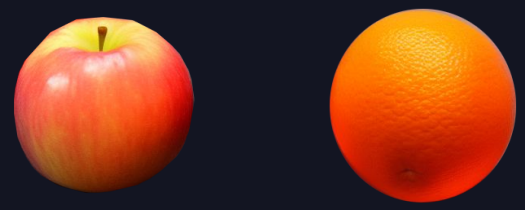
If the Flux and MPI Operator have different designs...

If the Flux and MPI Operator have different designs...

How efficient is each operator's setup?

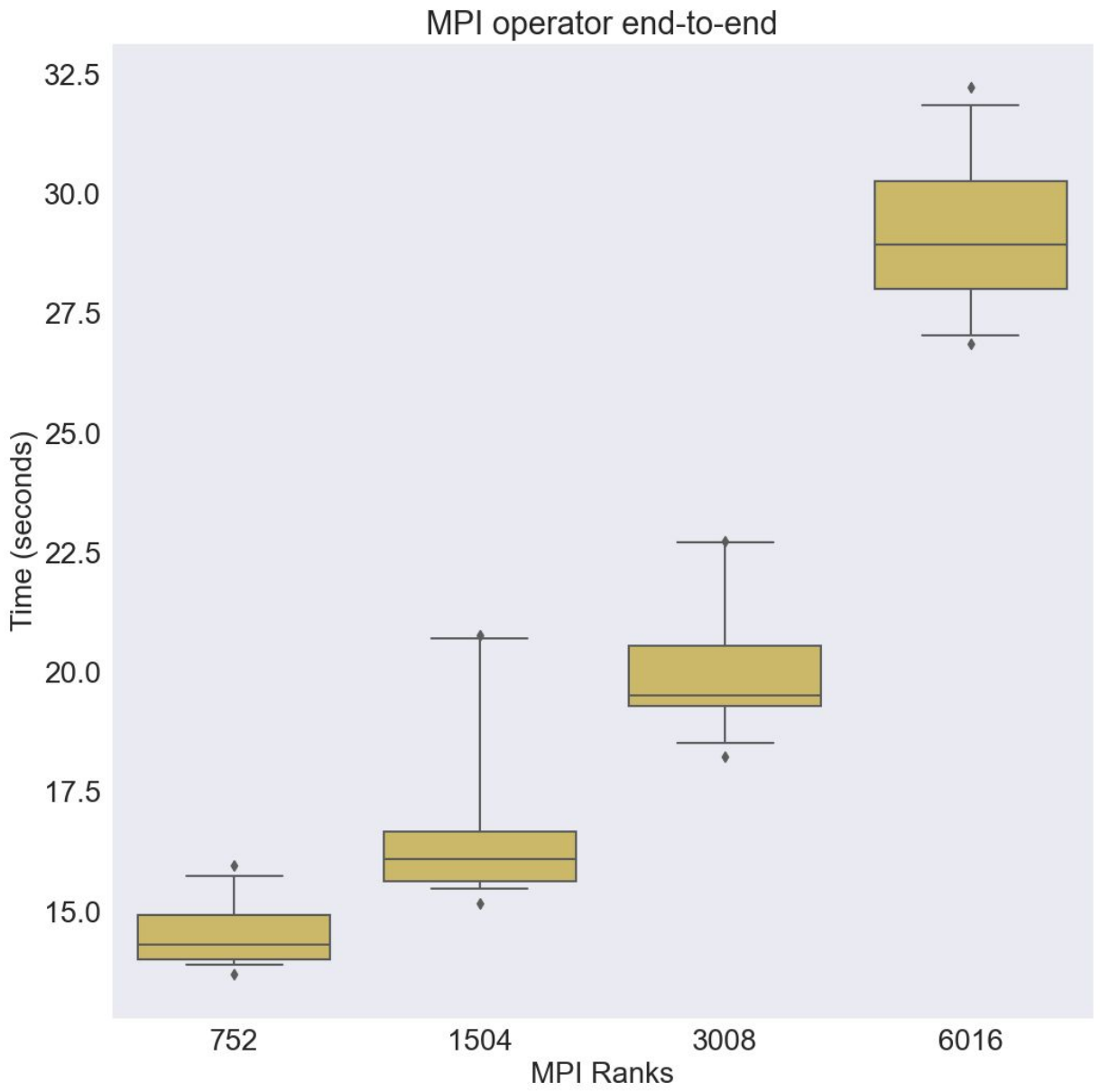
If the Flux and MPI Operator have different designs...

How efficient is each operator's setup?



MPI Operator End to End Time

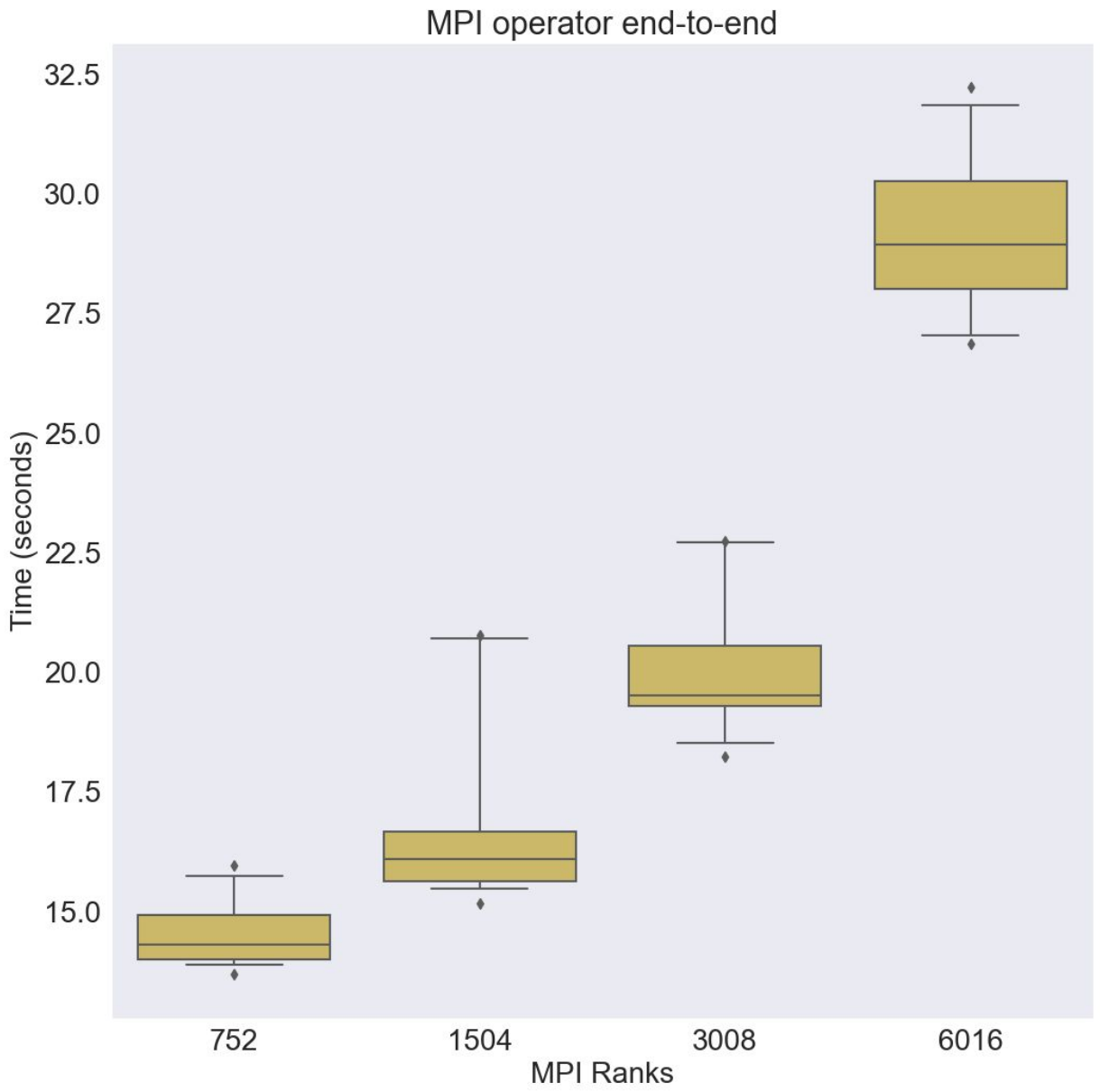
Notification of job through completed



MPI Operator End to End Time

Notification of job through completed

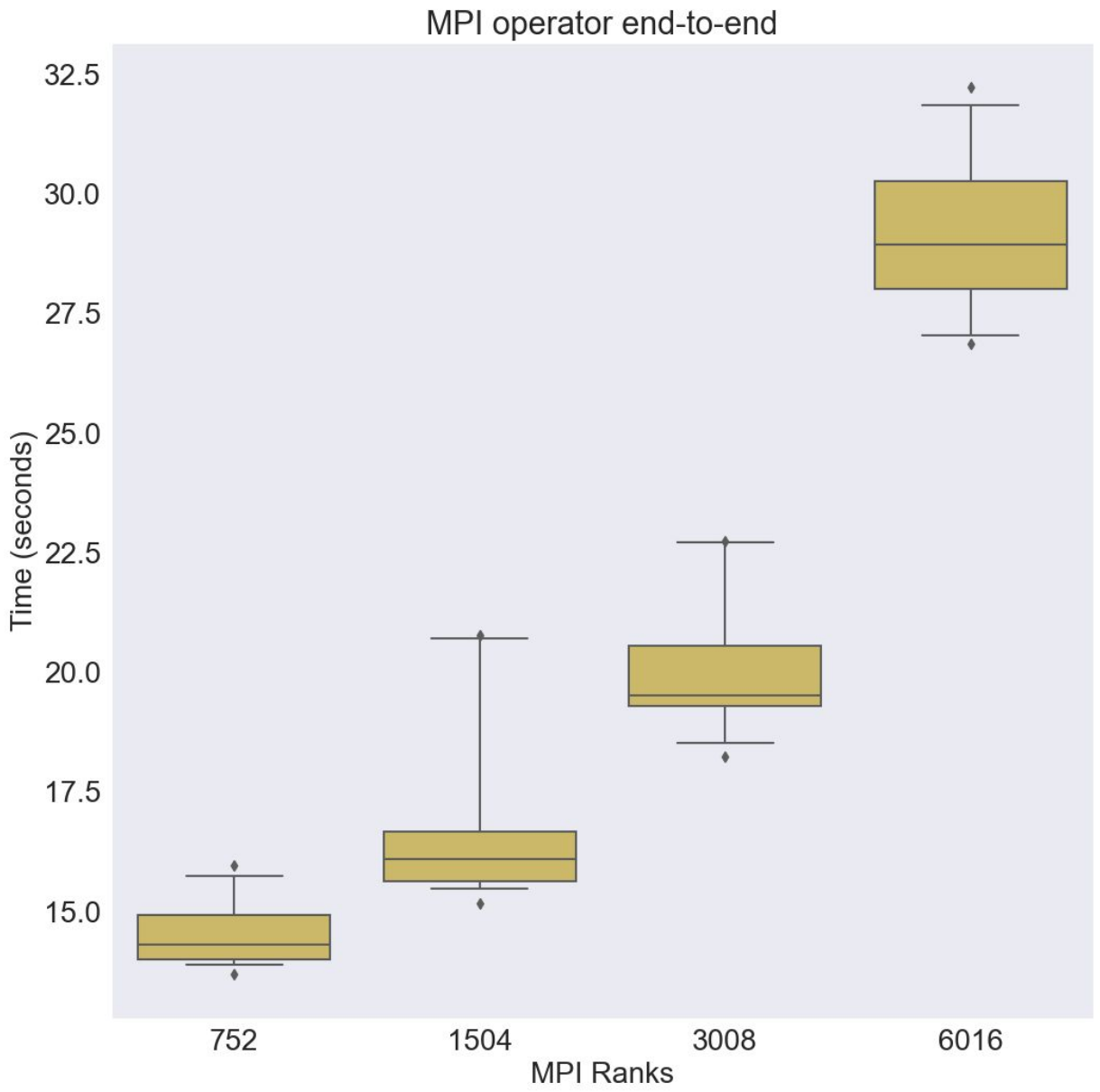
Pods are ready to go!



MPI Operator End to End Time

Notification of job through completed

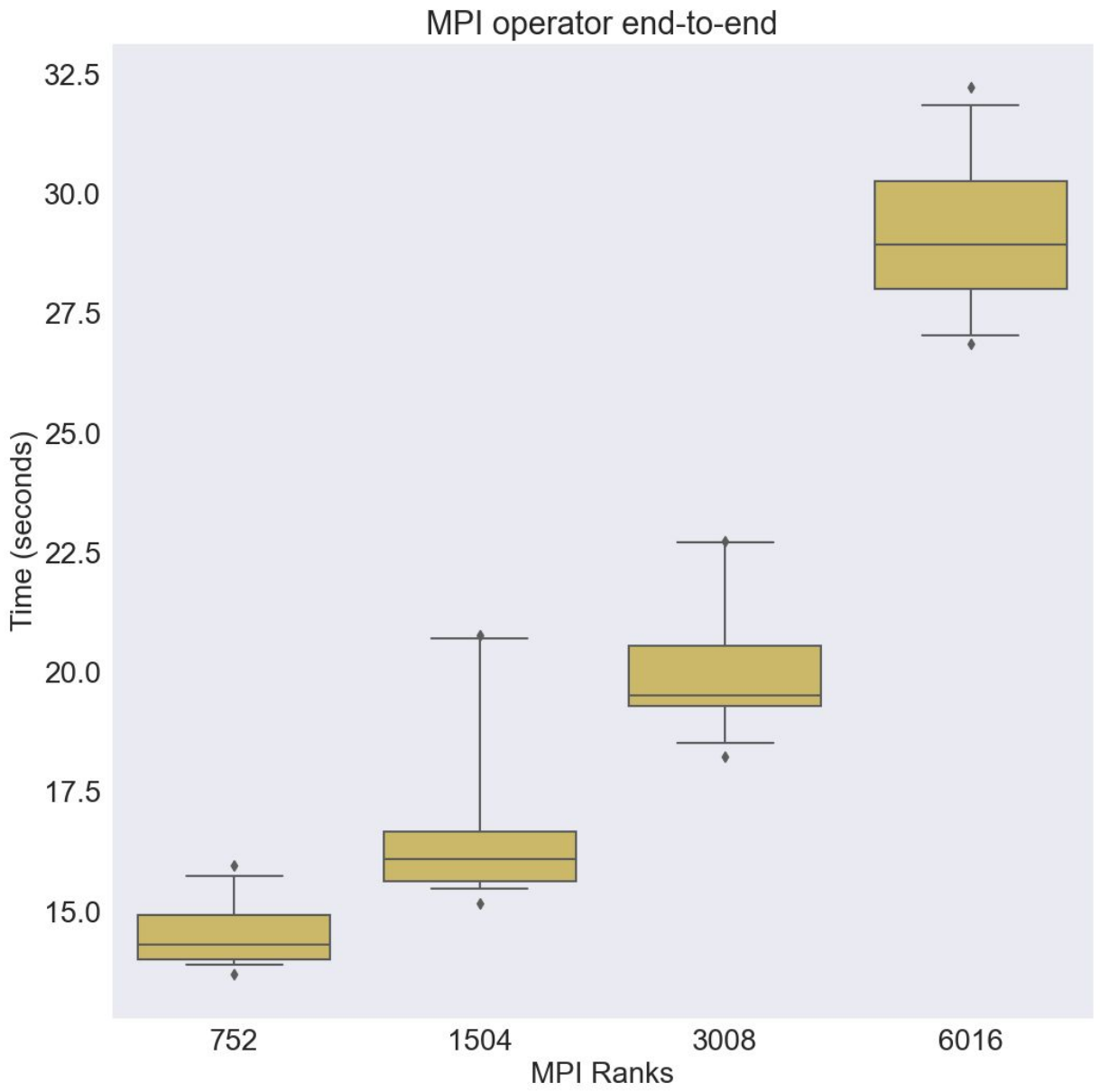
Pods are ready to go!
Not including the LAMMPS run



MPI Operator End to End Time

Notification of job through completed

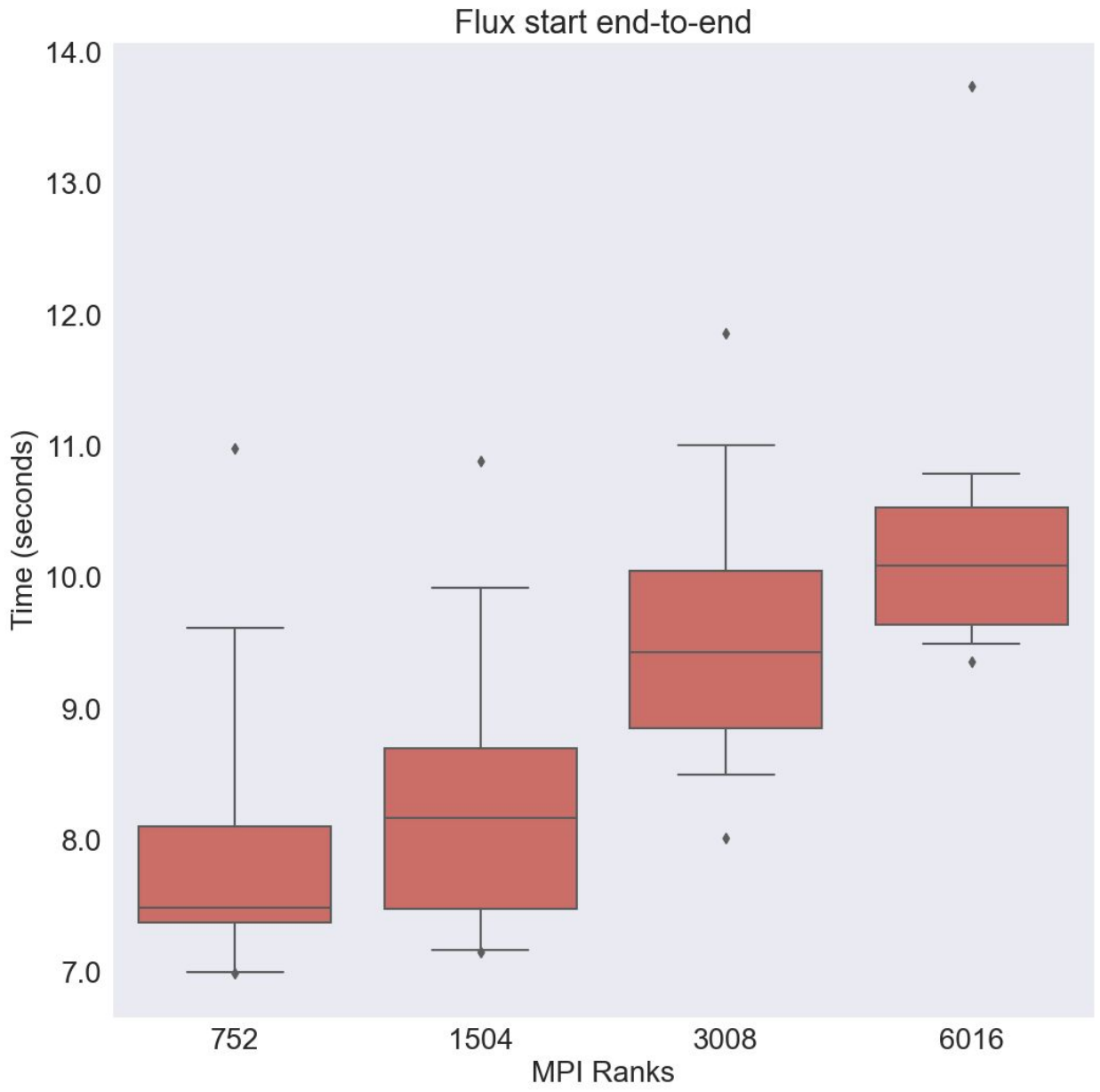
Pods are ready to go!
Not including the LAMMPS run
Time increases 2 fold from size 8 to 64



Flux Start End to End Time

Broker up through broker shutting down

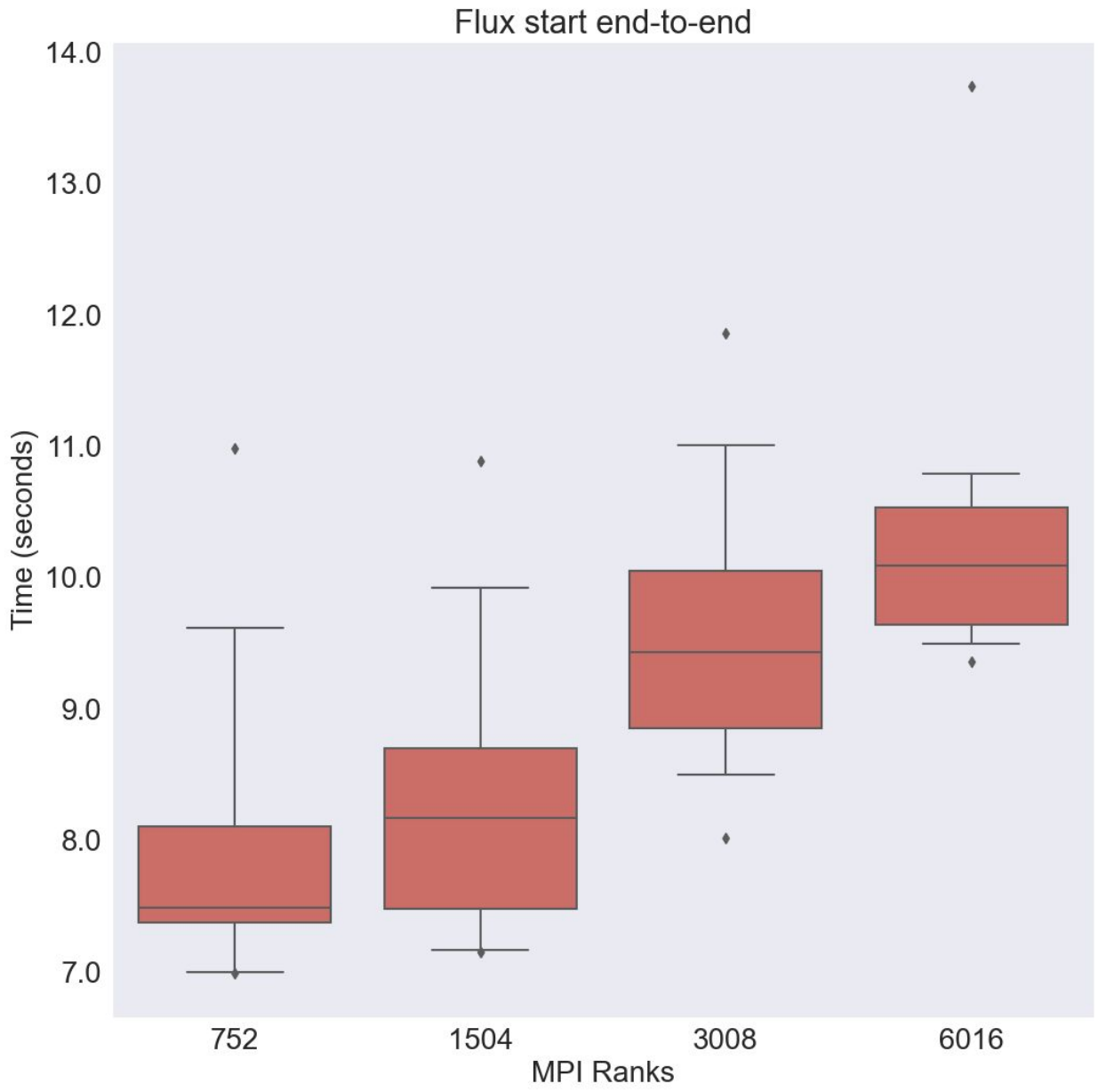
Includes broker waiting for all other pods



Flux Start End to End Time

Broker up through broker shutting down

Includes broker waiting for all other pods
Not including the LAMMPS run

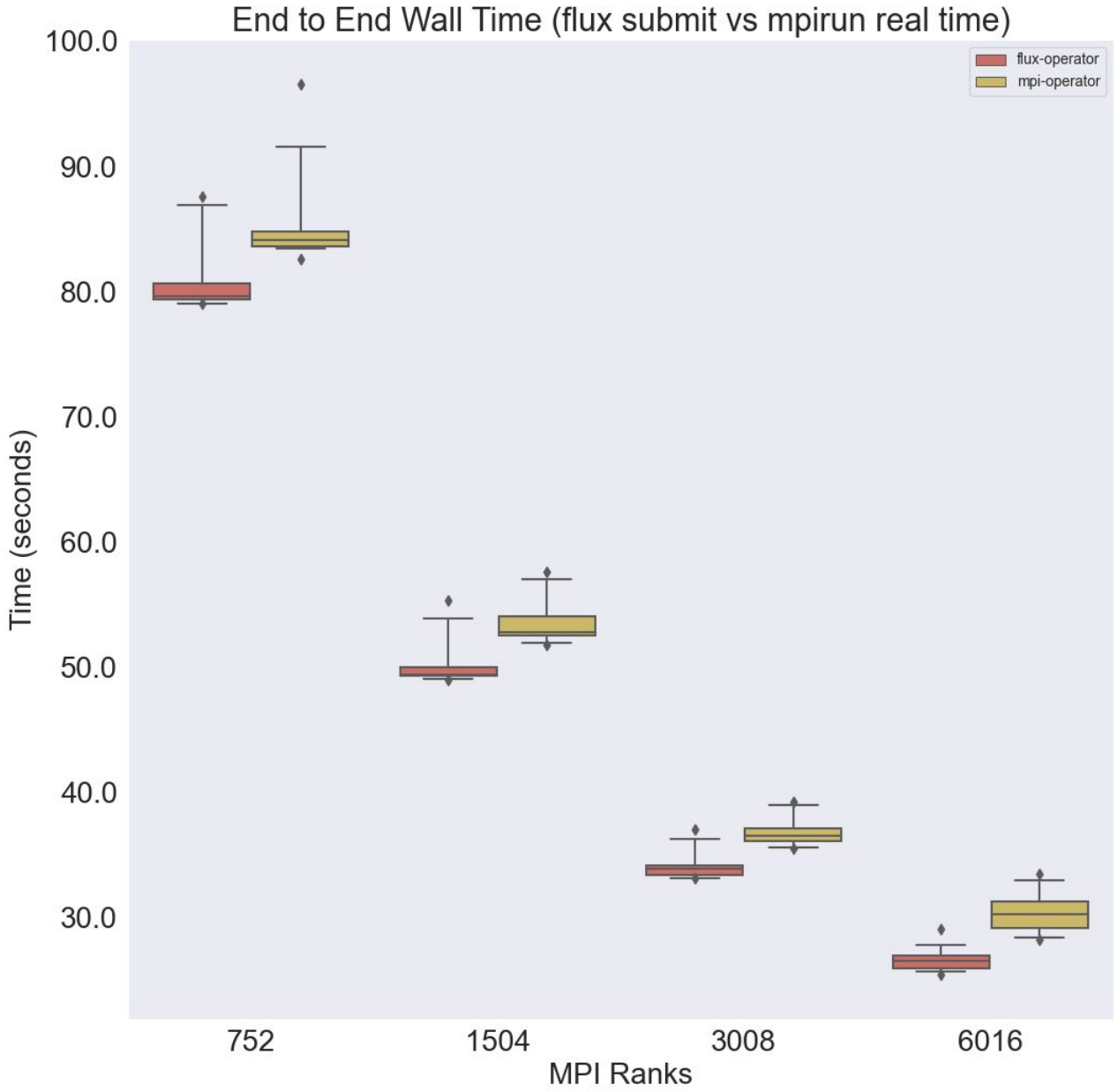


When you remove the setup...

When you remove the setup...
How do the runtimes compare?


Experiment Results

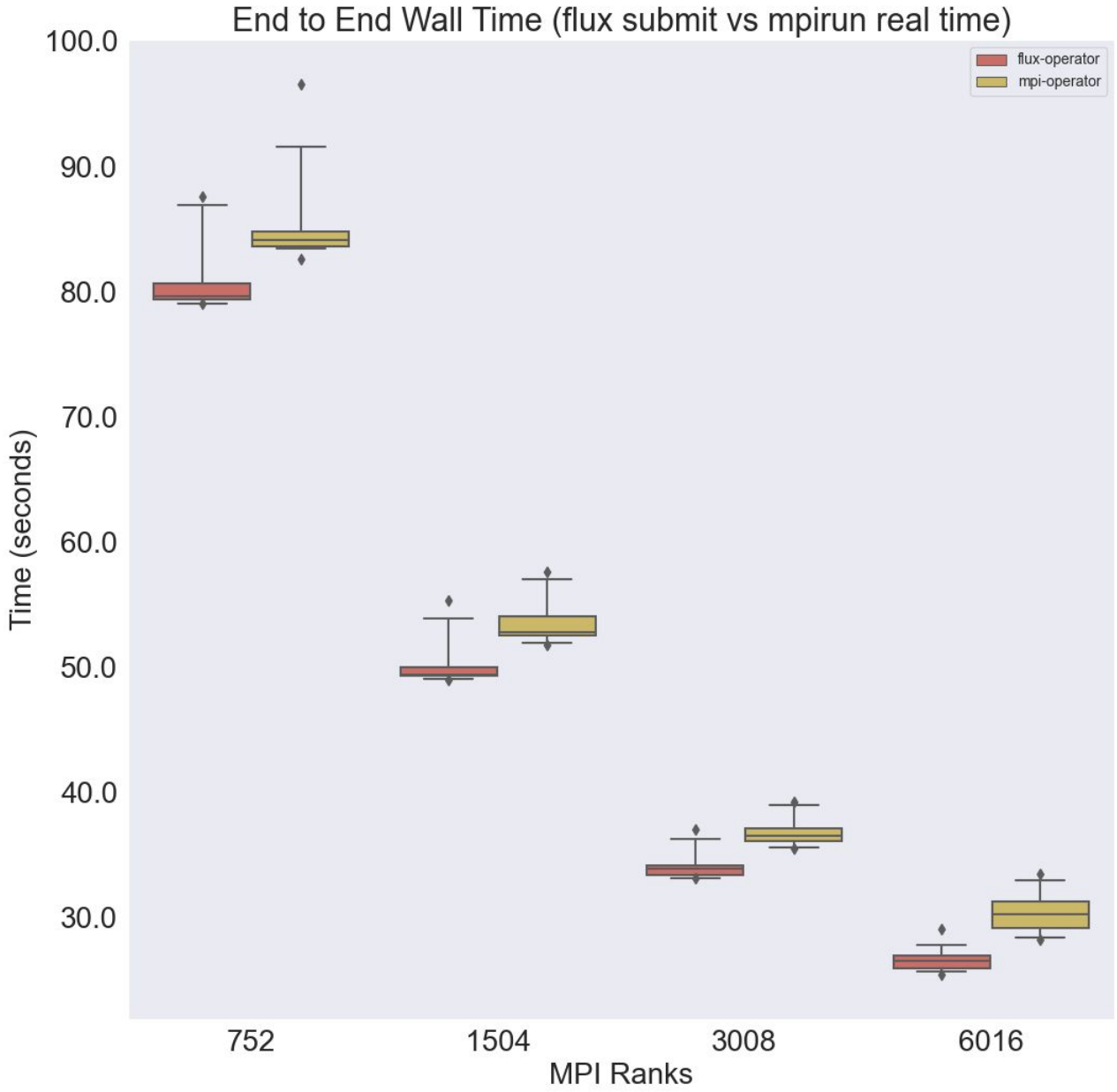
Flux submit vs. mpirun runtimes



Experiment Results

Flux submit vs. mpirun runtimes

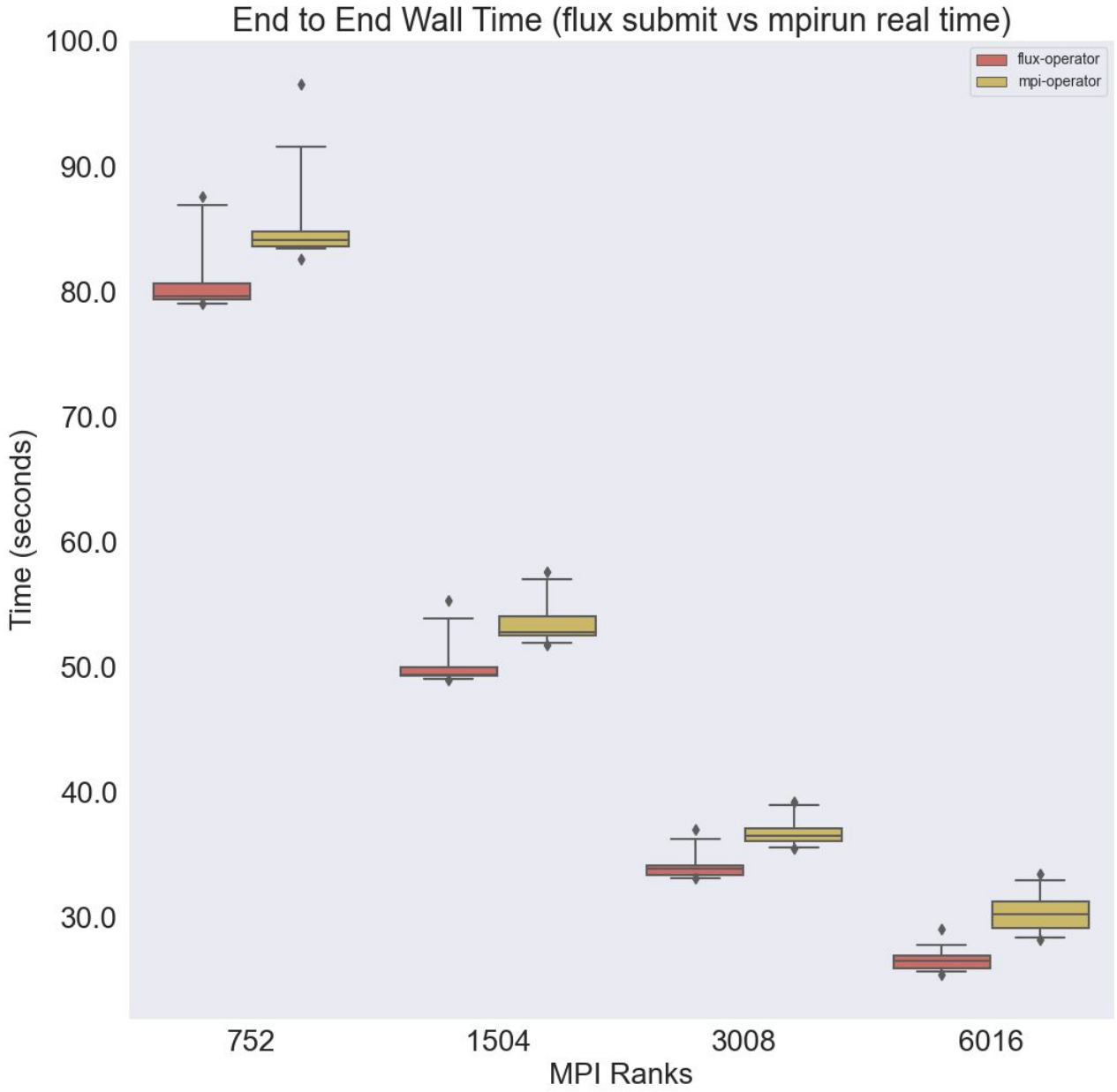
Includes the LAMMPS run



Experiment Results

Flux submit vs. mpirun runtimes

Includes the LAMMPS run
The direct "wrapper" to LAMMPS



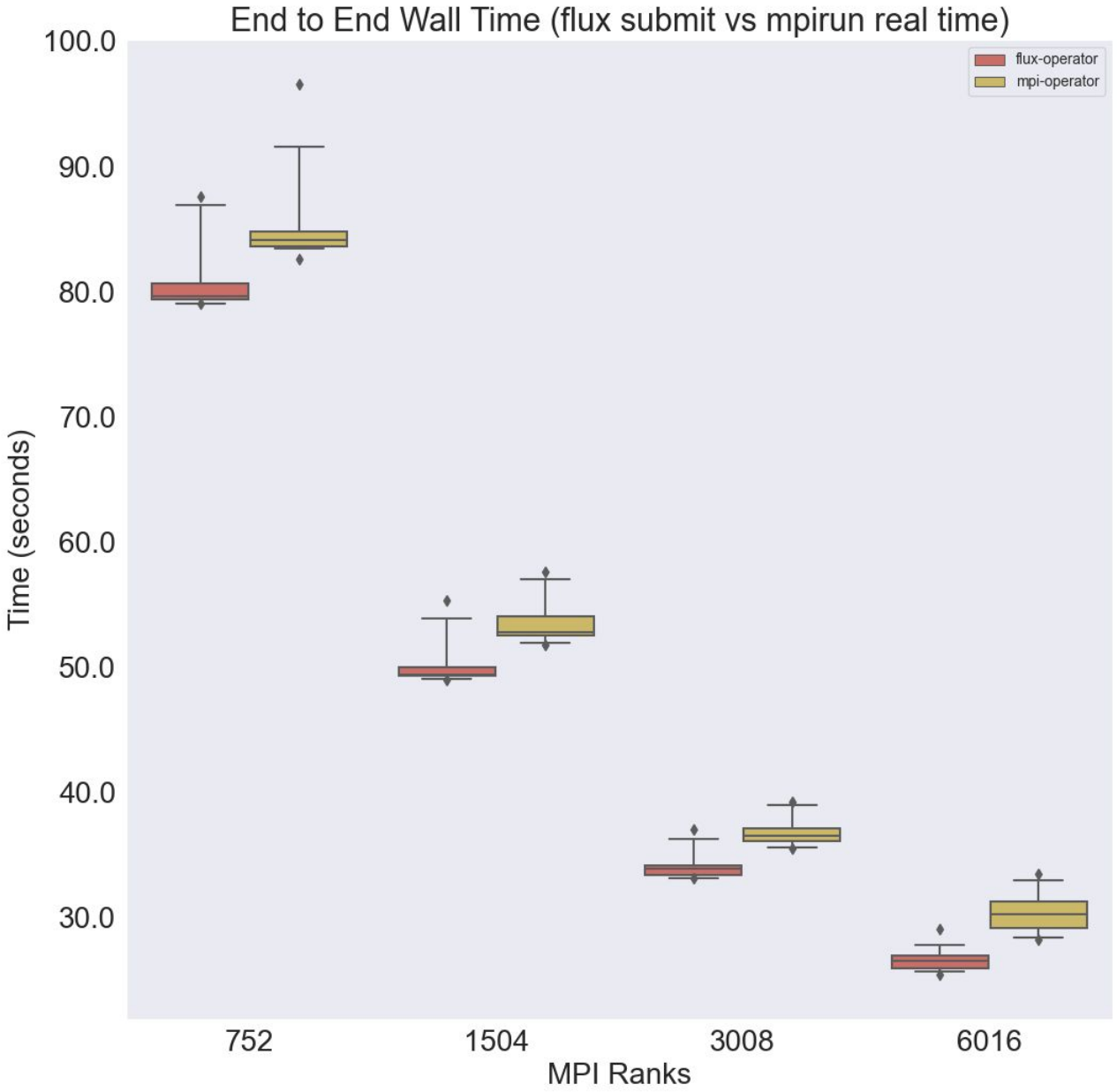
Experiment Results

Flux submit vs. mpirun runtimes

Includes the LAMMPS run
The direct "wrapper" to LAMMPS

For these experiments:

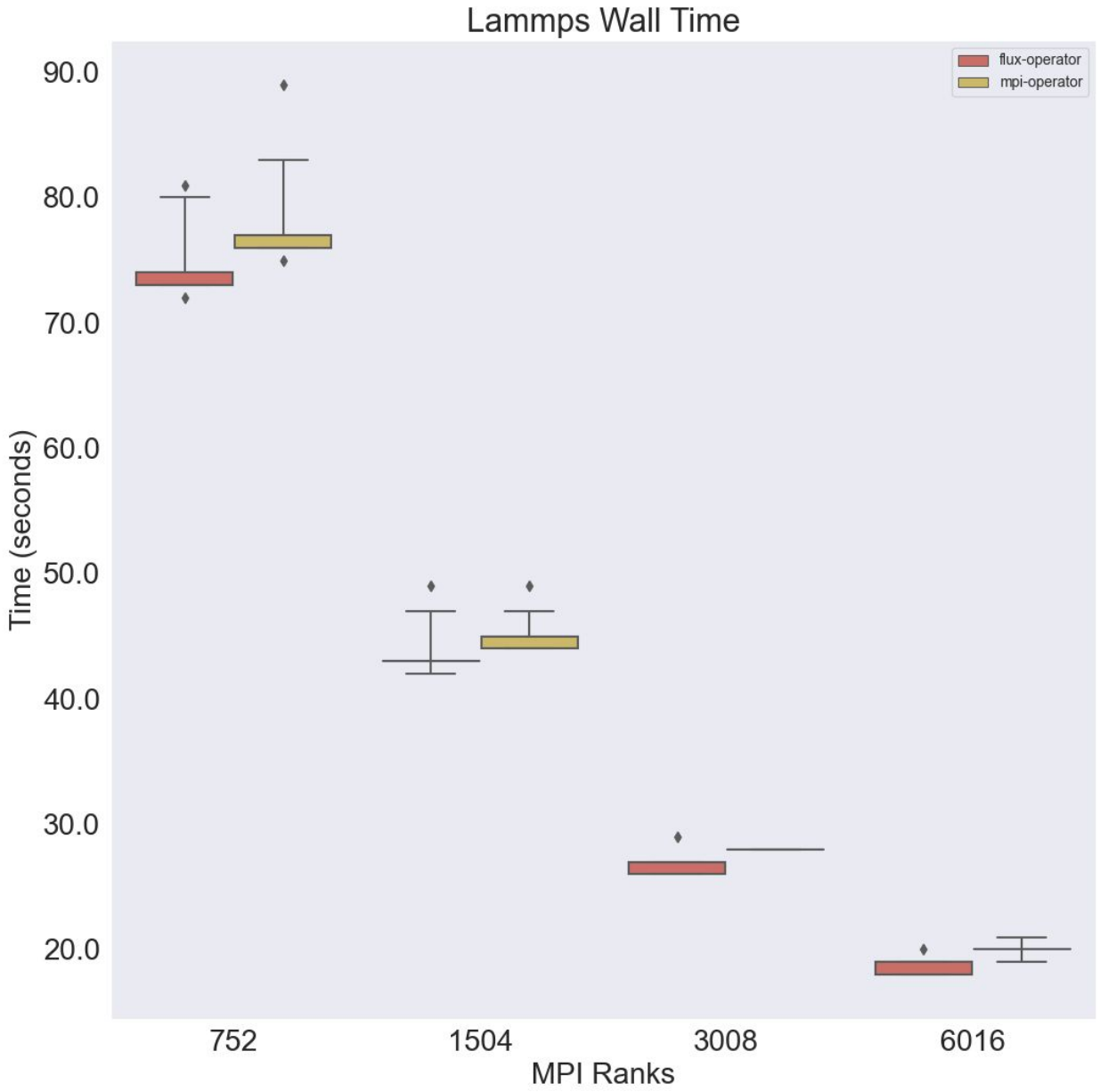
- Flux Operator is consistently faster
- Related to bootstrap?
- The difference is insignificant



Experiment Results

LAMMPS Runtimes

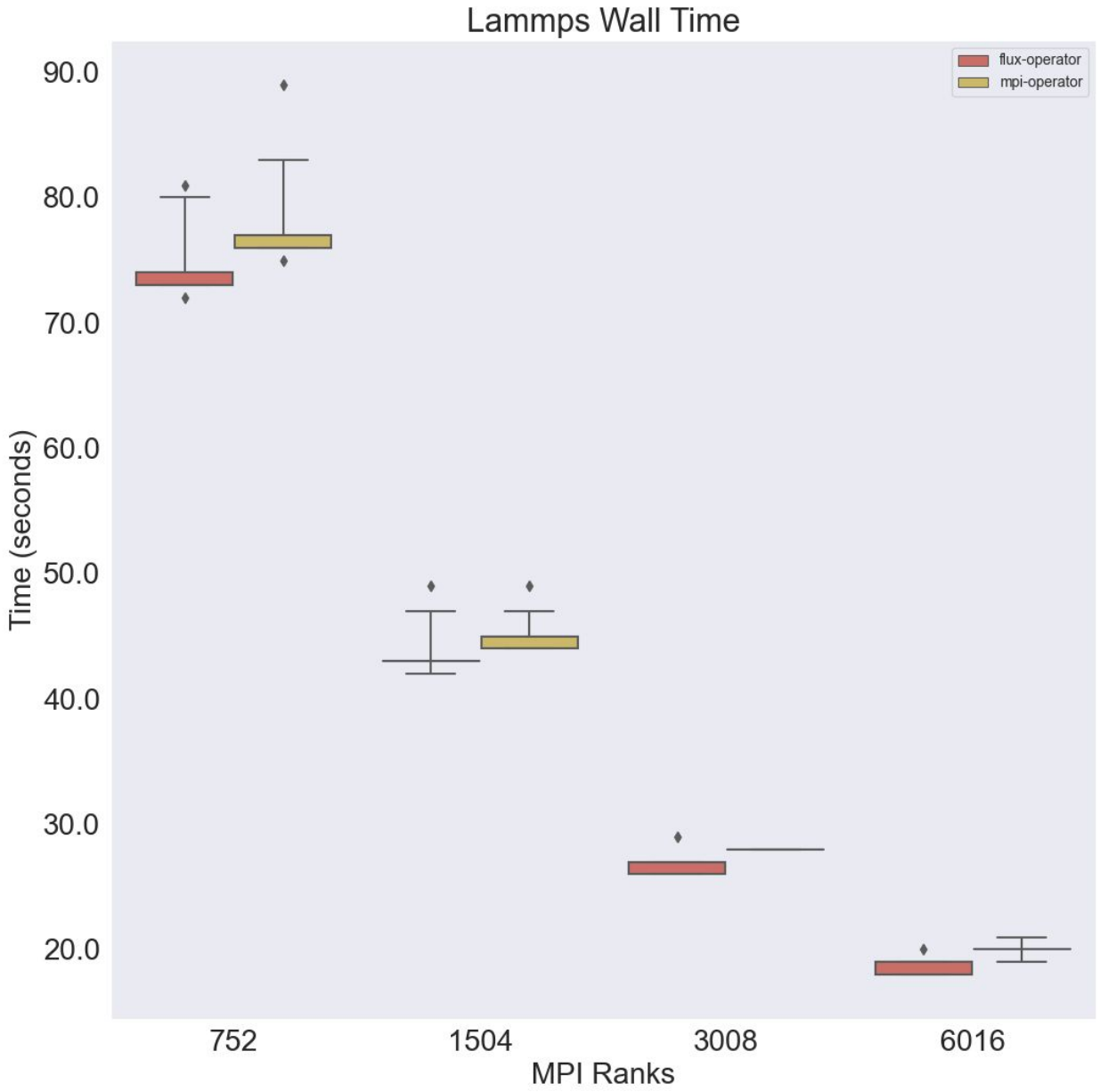
Only LAMMPS, no wrappers



Experiment Results

LAMMPS Runtimes

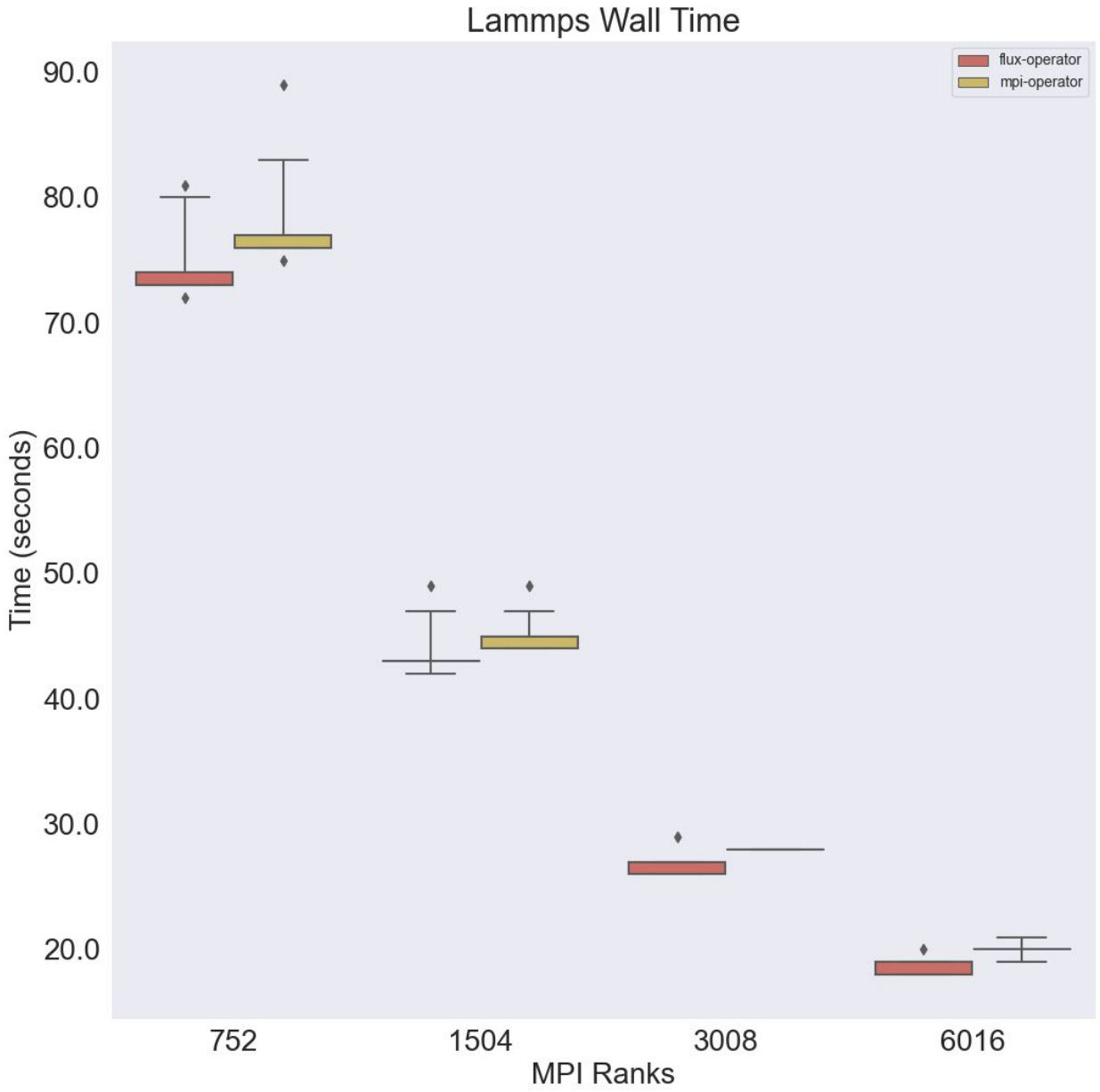
Only LAMMPS, no wrappers
Medians visually ~10% lower for Flux



Experiment Results

LAMMPS Runtimes

Only LAMMPS, no wrappers
Medians visually ~10% lower for Flux
Could translate to cost savings



Conclusions

- *The IndexedJob allows the MiniCluster pods to scale nicely*

Conclusions

- *The IndexedJob allows the MiniCluster pods to scale nicely*
- *The zeromq bootstrap (Flux) seems faster than ssh-bootstrap*

Conclusions

- *The IndexedJob allows the MiniCluster pods to scale nicely*
- *The zeromq bootstrap (Flux) seems faster than ssh-bootstrap*
- *More work is needed to further investigate performance*

Conclusions

- *The IndexedJob allows the MiniCluster pods to scale nicely*
- *The zeromq bootstrap (Flux) seems faster than ssh-bootstrap*
- *More work is needed to further investigate performance*
- *Architecture allows for multiple jobs to be run on the same Minicluster via tasks or Flux sub-instances*
  - *Avoids the infamous etcd/API server bottlenecks,*
  - *Enables high-throughput*

Conclusions

- *The IndexedJob allows the MiniCluster pods to scale nicely*
- *The zeromq bootstrap (Flux) seems faster than ssh-bootstrap*
- *More work is needed to further investigate performance*
- *Architecture allows for multiple jobs to be run on the same Minicluster via tasks or Flux sub-instances*
  - *Avoids the infamous etcd/API server bottlenecks,*
  - *Enables high-throughput*
- *The MPI Operator:*
  - *Requires an extra launcher node*
  - *Could also benefit from using an IndexedJob*

I've learned so much!

The Flux Operator has promise!

But I have questions...

How do I submit a job?

How do I submit a job?

Apply the minicluster.yaml?

How did we actually run these experiments?

Flux Cloud to Automate Flux operator Experiments

Define Your Cluster and Experiments

experiments.yaml

```yaml
# matrix of experiments to run - machine types and sizes are required
matrix:

  # Size of the Kubernetes cluster
  size: [64]

  # See https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/efa.html#efa-instance-types
  # not all support efa
  machine: ["hpc6a.48xlarge"]

kubernetes:
  version: "1.23"

# Flux Mini Cluster experiment attributes
minicluster:
  name: lammps
  namespace: flux-operator

  # These are the sizes of MiniCluster we will iterate over
  size: [64, 32, 16, 8]

variables:
  availability_zones: [us-east-2b, us-east-2c]
  region: us-east-2
  efa_enabled: true
  placement_group: eks-efa-testing

jobs:
  lmp-64:
    command: lmp -v x 64 -v y 16 -v z 16 -in in.reaxc.hns -nocite
    repeats: 21
    memory: 340G
    # 64*94
    tasks: 6016
    size: 64
    cpu: 94
    cores: 96

  lmp-32:
    command: lmp -v x 64 -v y 16 -v z 16 -in in.reaxc.hns -nocite
    repeats: 30
    memory: 340G
    # 32*94
    tasks: 3008
    size: 32
    cpu: 94
    cores: 96
```

Running Your Experiments... Three Commands Away!

flux-cloud up
Brings up the Kubernetes cluster and installs the operator

flux-cloud apply
Runs experiments (jobs) across MiniCluster sizes

flux-cloud-down
Cleans everything up

Keep Updated about What is Going On!

```
lammps-12-xgmg7          0/1     ContainerCreating   0          1s
lammps-13-t9csq          0/1     ContainerCreating   0          1s
lammps-14-rxt5m          0/1     ContainerCreating   0          1s
lammps-15-hx76n          0/1     ContainerCreating   0          1s
lammps-2-l6256           0/1     ContainerCreating   0          1s
lammps-3-hc4dp           0/1     ContainerCreating   0          1s
lammps-4-jcm87           0/1     ContainerCreating   0          1s
lammps-5-tjxfz           0/1     ContainerCreating   0          1s
lammps-6-8gd4n           0/1     ContainerCreating   0          1s
lammps-7-d9jps           0/1     ContainerCreating   0          1s
lammps-8-gs67p           0/1     ContainerCreating   0          1s
lammps-9-wp5qg           0/1     ContainerCreating   0          1s
lammps-cert-generator    0/1     ContainerCreating   0          2s

Waiting for broker pod with prefix lammps-0 to be created...
.
🌀 Broker pod is created.

Waiting for broker pod with prefix lammps-0 to be running...
..
🌀 Broker pod is running.

kubectl -n flux-operator logs lammps-0-hswfv -f > /home/vanessa/Desktop/Code/flux/experiments/aws/lammps/kubecon/run2/d
ata/k8s-size-64-hpc6a.48xlarge/lmp-16-10-minicluster-size-16/log.out
```

```
$ flux-cloud --debug up --cloud aws
$ flux-cloud --debug apply --cloud aws
$ flux-cloud --debug down --cloud aws
```

Flux Cloud Output

Configuration files, data, and scripts saved for reproducibility

```
data/
└── aws
    └── k8s-size-8-hpc6a.48xlarge
        ├── _lmp-8-1-minicluster-size-8
        │   └── log.out
        ├── lmp-8-2-minicluster-size-8
        │   └── log.out
        ├── lmp-8-3-minicluster-size-8
        │   └── log.out
        ├── lmp-8-4-minicluster-size-8
        │   └── log.out
        ├── lmp-8-5-minicluster-size-8
        │   └── log.out
        └── meta.json
```

```
$ tree -a ./data/
./data/
└── minikube
    └── k8s-size-4-local
        ├── lmp-size-2-minicluster-size-2
        │   └── log.out
        ├── lmp-size-4-minicluster-size-4
        │   └── log.out
        ├── meta.json
        └── .scripts
            ├── cluster-create-minikube.sh
            ├── flux-operator.yaml
            ├── kubectl-version.yaml
            ├── minicluster-run-lmp-size-2-minicluster-size-2.sh
            ├── minicluster-run-lmp-size-4-minicluster-size-4.sh
            ├── minicluster-size-2.yaml
            ├── minicluster-size-4.yaml
            ├── minikube-version.json
            ├── nodes-size-4.json
            └── nodes-size-4.txt
```

How do I submit a job?

A comfortable, intuitive user interface!

Reality is informed by...

Vision!

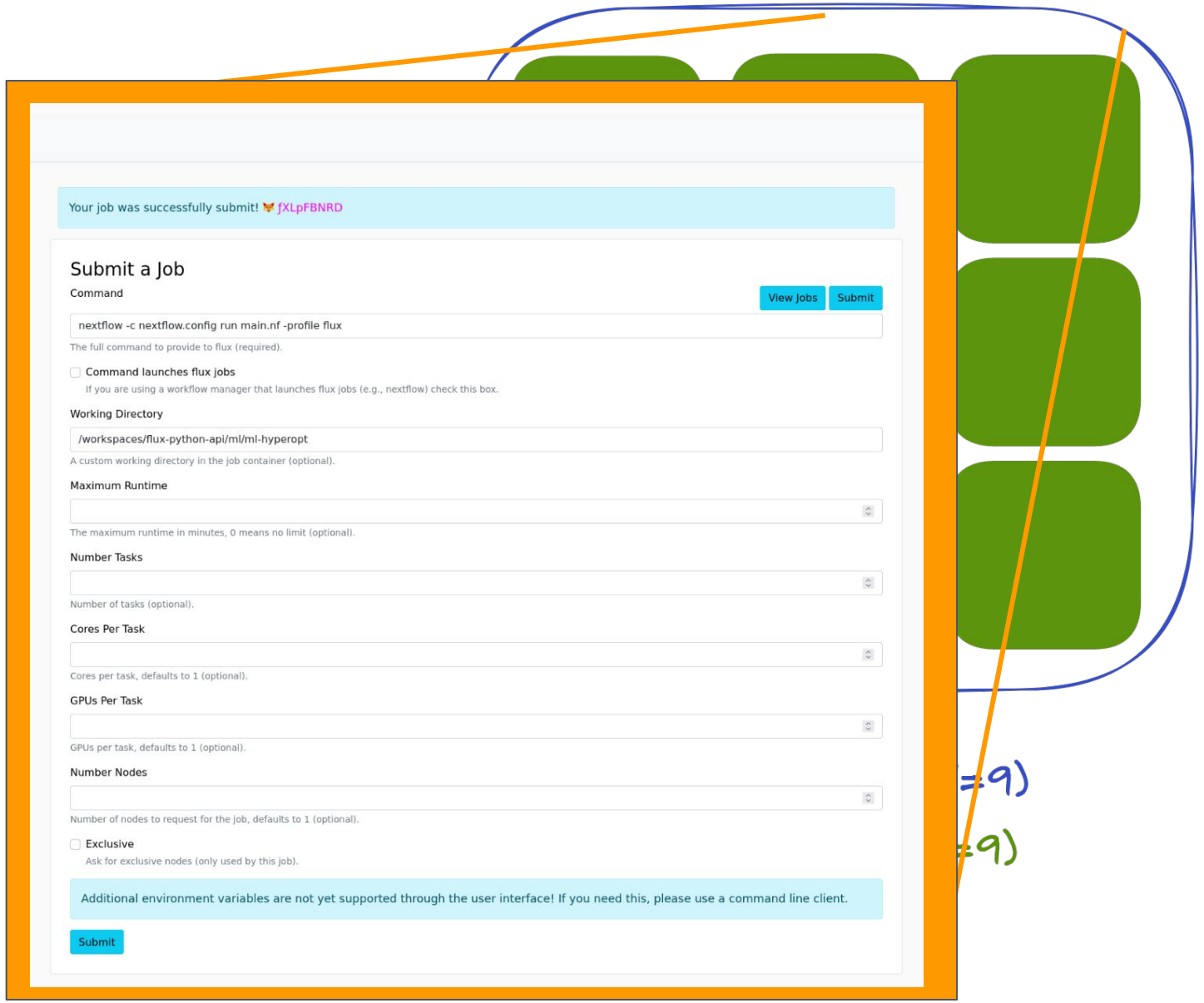
Vision!

Vision!

How *could* we submit jobs?

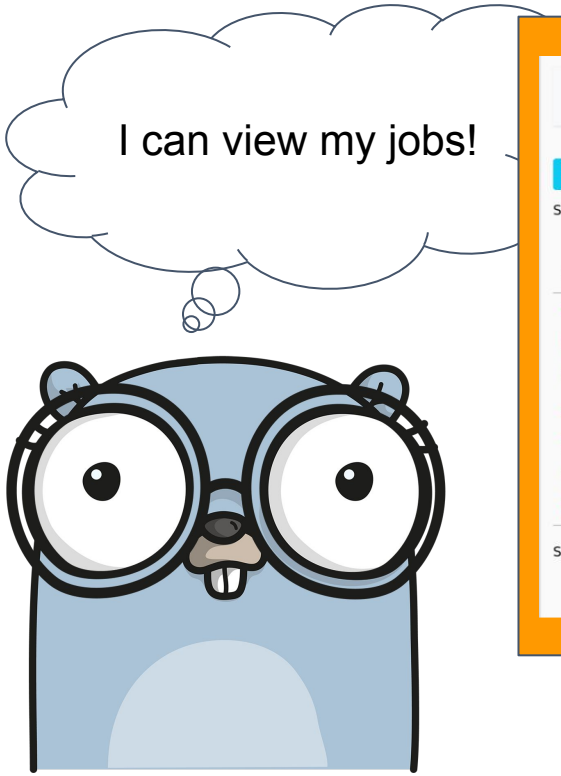
The Flux Operator

Submit via a Web Interface



The Flux Operator

Monitor jobs in an interactive table



Submit Jobs API

Submit Another Job

reset

Show 100 entries

Search:

id	returncode	runtime	result	urgency	priority	state	name	ntasks	ncores	duration	nnodes	ranks	nodelist	expiration	waitstatus	exception
244509770252288	0	0.031345367431640625	COMPLETED	16	16	INACTIVE	echo	1	1	0	1	3	57d0f60fce2e	4821898952	0	no exceptions
116794505297920	0	120.02380442619324	COMPLETED	16	16	INACTIVE	sleep	1	1	0	1	3	57d0f60fce2e	4821891306	0	no exceptions
39064489164800	0	1200.045151233673	COMPLETED	16	16	INACTIVE	sleep	1	1	0	1	3	57d0f60fce2e	4821886673	0	no exceptions
38133571780608	0	60.0217170715332	COMPLETED	16	16	INACTIVE	sleep	1	1	0	1	0	57d0f60fce2e	4821886617	0	no exceptions
36512691388416	0	120.04352164268494	COMPLETED	16	16	INACTIVE	sleep	1	1	0	1	1	57d0f60fce2e	4821886521	0	no exceptions
30820953751552	0	120.0259211063385	COMPLETED	16	16	INACTIVE	sleep	1	1	0	1	2	57d0f60fce2e	4821886182	0	no exceptions
26132493631488	0	60.05410575866699	COMPLETED	16	16	INACTIVE	sleep	1	1	0	1	3	57d0f60fce2e	4821885902	0	no exceptions

Showing 1 to 7 of 7 entries

First Previous 1 Next Last

The Flux Operator

View jobs output and metadata

Job 460215812096

Submit Jobs API

Attribute	Value
ID	460215812096
Name	pokemon
State	INACTIVE 😴
Result	COMPLETED
Urgency	16
Priority	16
Number Tasks	1
Cores	1
Nodes	1 (e6d4b5a7d9bc)
Ranks	0
Runtime	0.08921337127685547
Return Code	0

```
@@@@@@@@@@@@@@@@@@@@@@@@@SS.*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@...SS..S@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@:.S........@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@S,@@@@@@.............@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@S?@@@@,.............%#@@@@@@@@@@@@@@@@@@SS,.@@@@@@@@@@@@
@@@@@@@+.@@@@..............%.%@@@@@@@@@,:::::::S..S@@@@@@@@@@
@@@@@@@#..@@%.................@@@@#:::::+*......S.........@
@@@@@?#@..+@....:**...........S@@:::::.................++.++.*
@@@@?SS@.+SSS.::::*%......@@@@@::*............++++.++...+@@@
@@@@.SS%SSSSS:::::*#...%*..@@+::+...S.?S....+++++....+.@@@@@
@@@@+:+++SSS:*:#***.++...?..+:...%..?S?+++++++++#@@@@@@@@@@@
@@+++SS.S%.+*****?.:**%?....*:...SSSSS.S#+++++...?@@@@@@@@@@
@..S?...+#%.****.+****+%%@@.*++.++.SS+#?+++++++++@@@@@@@@@@@
#...@@@,SSSS#%+++++++++#.S@@*.++SSS+++S,@@+.++.#@@@@@@@@@@@@
@,@@@@@%S++++#@@@@,#.++++SSSS.S.?+.++...@@@@@@@@@@@@@@@@@@@@
@@@@@+++%+*@@@@@@@@@@...+++?SSS*.+++++.*@@@@@@@@@@@@@@@@@@@@
@@@@@@S?S@@@@@@@@@@@@,.+++.:*?SS,@@*S?@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@,?*::*+%.@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@,.........@@@@@@#*%:.?.+#@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@.........@@@@@*:%...++#@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@:::.?@SS..%......@@:::%......+,@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@::S++.SSS..+.....?:.........%@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@,.::#.....S..+.+#:.........*@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@#%..++...S..%+..%......?,@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@..+++++..%+%.%SSSSS@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@,,#+SSS.SSS#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Flapple
```

The Flux Operator

Submit via command line or SDK


```
$ pip install flux-restful-client
```

```
$ flux-restful-cli submit sleep 60
```

```
{
    "Message": "Job submit.",
    "id": 12402053611520
}
```

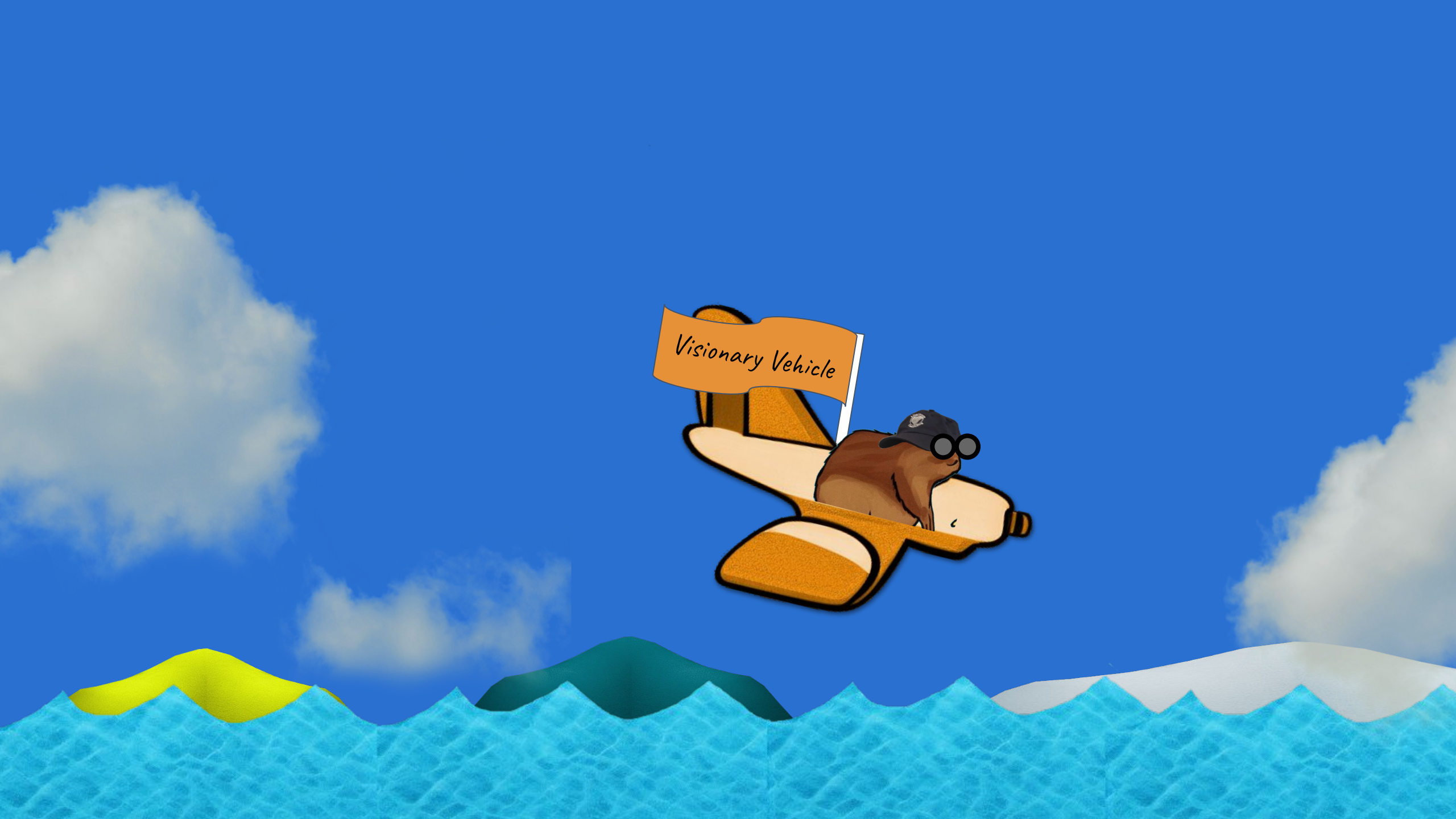
```
# Make sure the directory with the client is in sys.p
from flux_restful_client import get_client

# create with username and token
cli = get_client(user="fluxuser", token="12345")

# Add or update after creation
cli.set_basic_auth(user="fluxuser", token="12345")
```

Coming Soon to a Theater Near You!

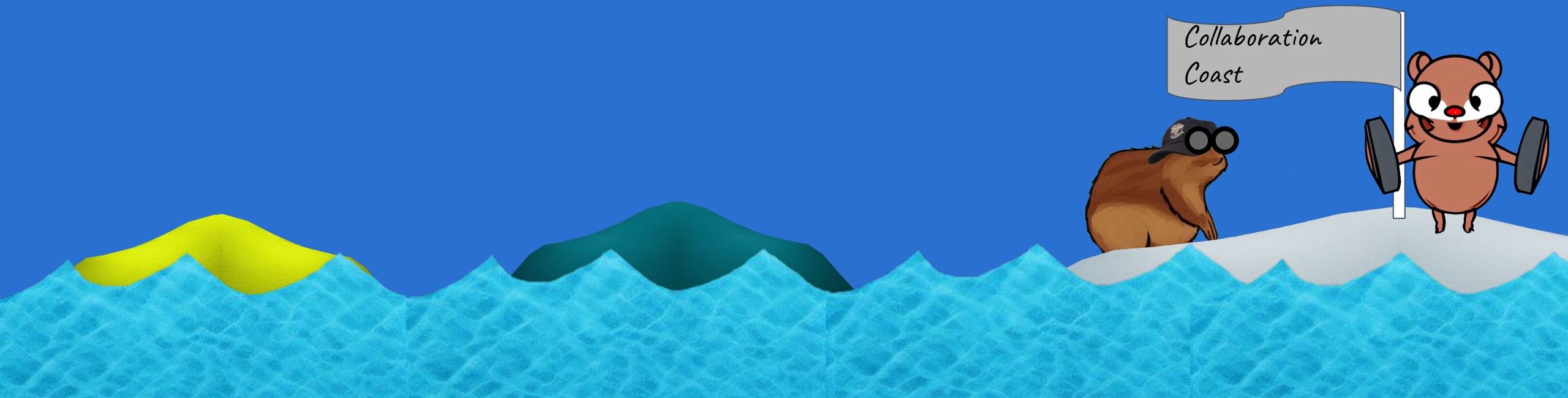
Elasticity
Multi-tenancy
Storage Options
Container Validator


Visionary Vehicle


Visionary Vehicle


Visionary Vehicle


Collaboration
Coast


Collaboration
Coast


We need to work together, and share ideas!
Collaboration Coast

If you were designing a cloud resource manager...

If you were designing a cloud resource manager...

What are some design tips?

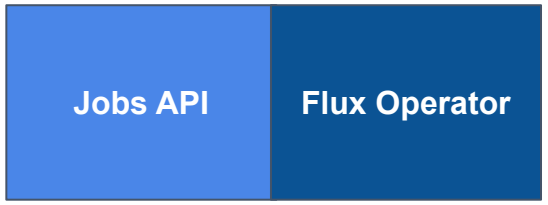
Operator Design Tips / Feedback

Problem: I have a resource manager that communicates via a network.

Operator Design Tips / Feedback

Problem: I have a resource manager that communicates via a network.

Solution: An IndexedJob + headless service is a nice solution (FQDN)



Which component is responsible? 🤔

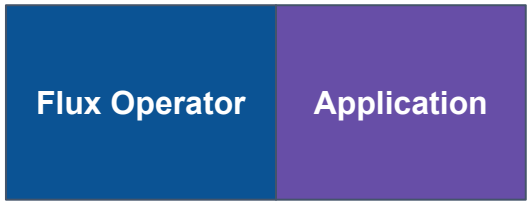
Operator Design Tips / Feedback

Problem: I need specialized logic to generate something!

Operator Design Tips / Feedback

Problem: I need specialized logic to generate something!

Solution: Run via an entrypoint, an isolated pod, or use initContainers



Which component is responsible? 🤔

Operator Design Tips / Feedback

Problem: My workers are specialized!

Operator Design Tips / Feedback

Problem: My workers are specialized!

Solution: Use logic that distinguishes based on the index

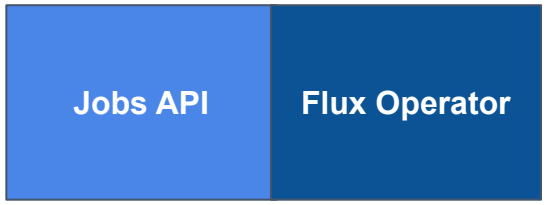
Flux Operator

Which component is responsible? 🤔

Operator Design Tips / Feedback

Problem: My workers are specialized!

Solution: Use logic that distinguishes based on the index



Jobs API Suggestion: Could it be possible to define groups with custom logic within an indexed job?

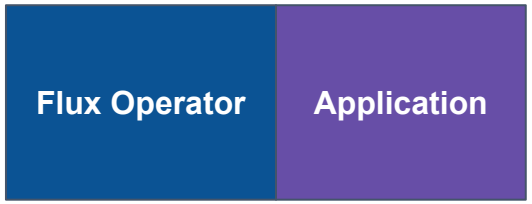
Operator Design Tips / Feedback

Problem: Multi-tenancy or need for root and user permissions/ownership

Operator Design Tips / Feedback

Problem: Multi-tenancy or need for root and user permissions/ownership

Solution: Container is run/started as root, jobs run as user



Which component is responsible? 🤔


HPC Land
flux
flux
Cloud Land


HPC Land
flux
flux
?
Cloud Land

How do I get involved?

Flux Framework
https://github.com/flux-framework

Flux Framework
https://github.com/flux-framework

The Flux Operator
https://github.com/flux-framework/flux-operator

Flux Framework
https://github.com/flux-framework

The Flux Operator
https://github.com/flux-framework/flux-operator

Flux Cloud
https://github.com/converged-computing/flux-cloud

Flux Framework
https://github.com/flux-framework

The Flux Operator
https://github.com/flux-framework/flux-operator

Flux Cloud
https://github.com/converged-computing/flux-cloud

Flux Framework Portal
https://flux-framework.org

📧 sochat1@llnl.gov
👋 @vsoch

📧 sochat1@llnl.gov
👋 @vsoch
🙌 aws

📧 sochat1@llnl.gov
👋 @vsoch
🙌 aws

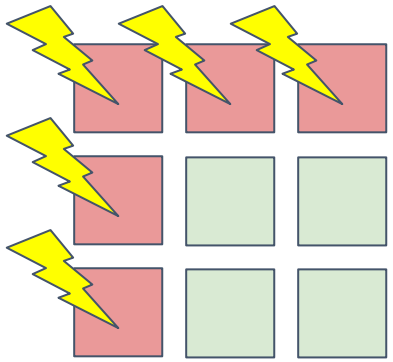
Thank you!

Pod failure policy (use case)

- **Use case:** large parallel workloads, pods fail due to various reasons:
  - Software bugs
  - Disruptions
  - Transient issues
- **backoffLimit** allows to specify the number of retries…

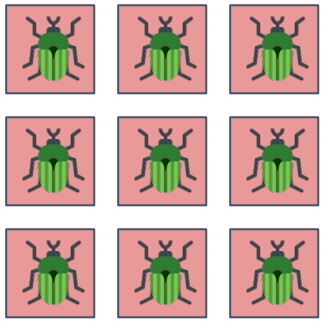
Pod failure policy (use case)

- **Use case:** large parallel workloads, pods fail due to various reasons:
  - Software bugs
  - Disruptions
  - Transient issues
- **backoffLimit** allows to specify the number of retries… but there is a tradeoff:



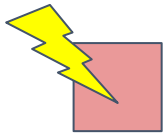
Too low: job failures in case of **disruptions**

vs.



Too high: unnecessary pod restarts in case pod **software bugs**

Pod failure policy (approach)



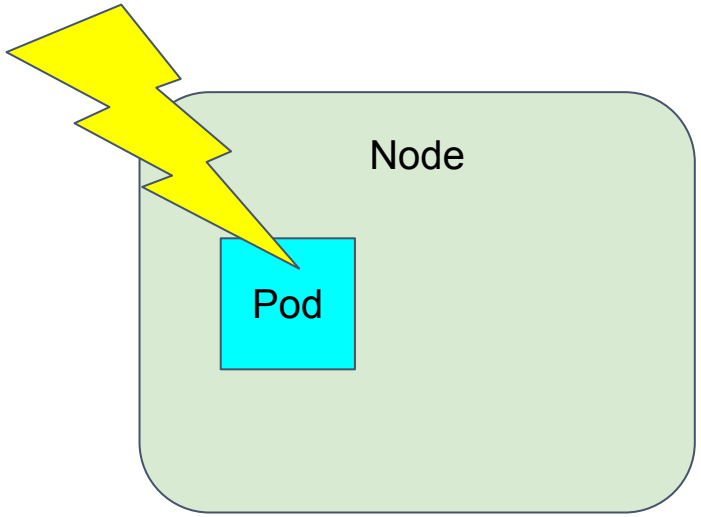
Disruption vs. **Software bug**: how to tell a difference?

- **Container exit codes**
  - Used by Kubeflow by TFJob
  - Some exit code are ambiguous, like 137 (SIGKILL)

- **Pod conditions!**

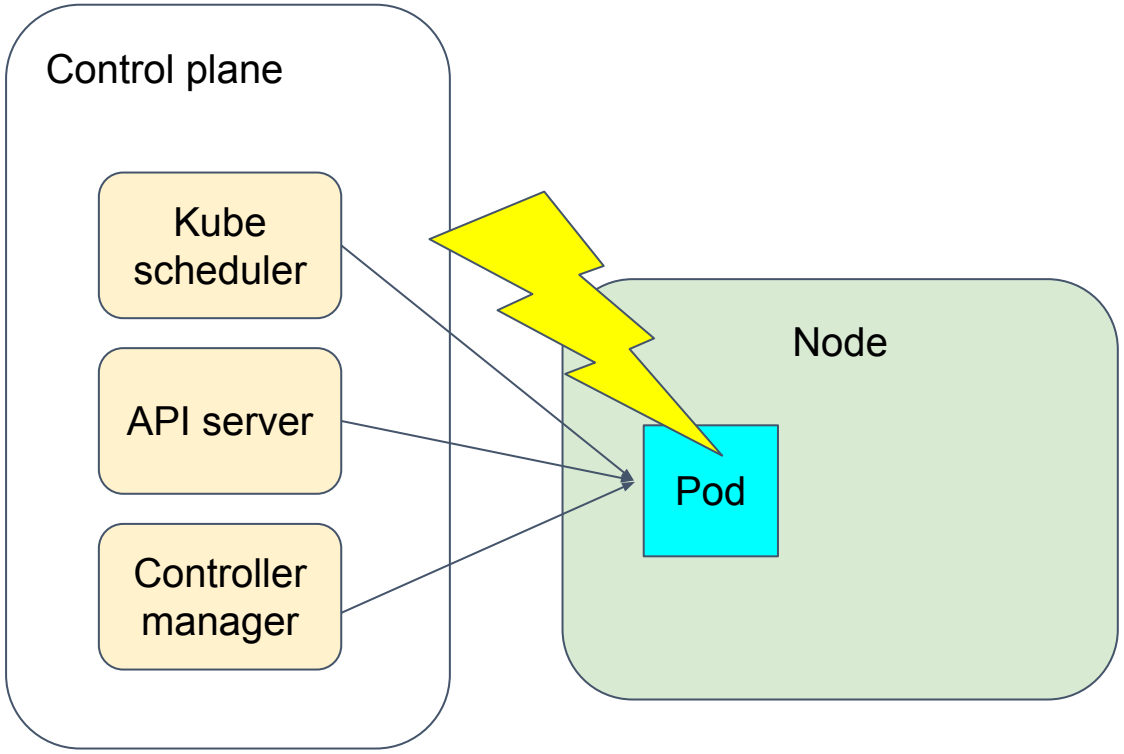
Pod failure policy (pod conditions)

- **DisruptionTarget** is added to indicate the disruption reason



Pod failure policy (pod conditions)

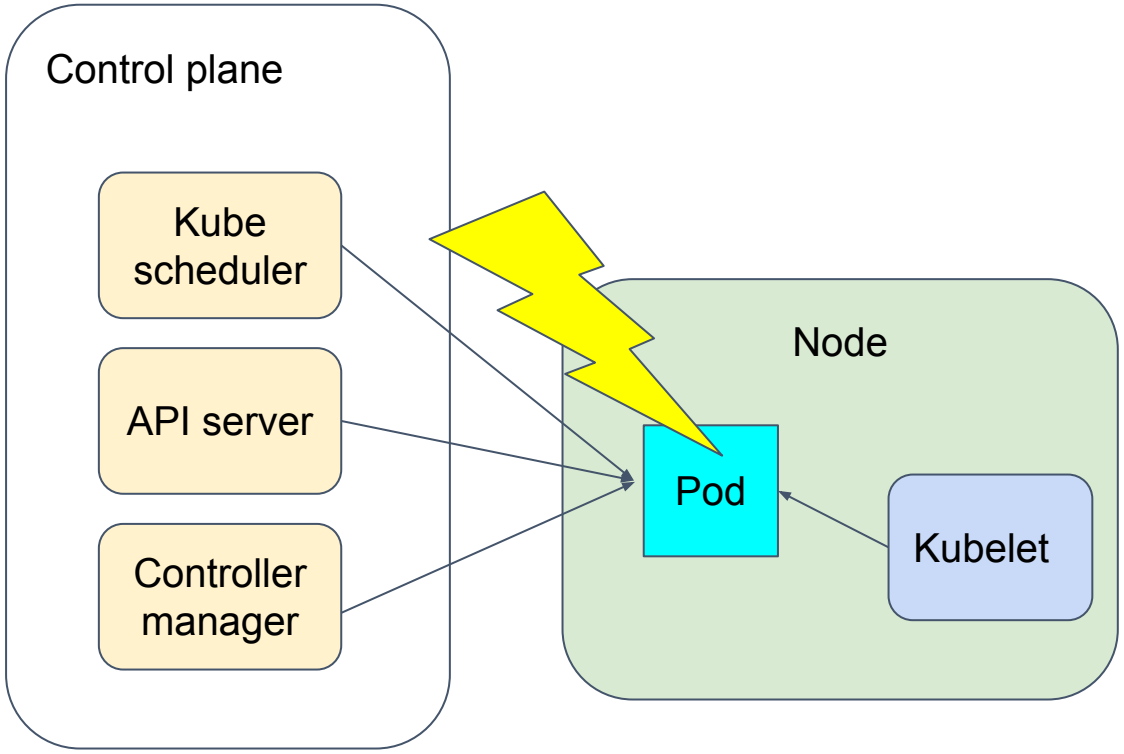
- **DisruptionTarget** is added to indicate the disruption reason
  - Scheduler preemption
  - Eviction API
  - Untolerated NoExecute taint
  - Deletion by PodGC



Pod disruption scenarios

Pod failure policy (pod conditions)

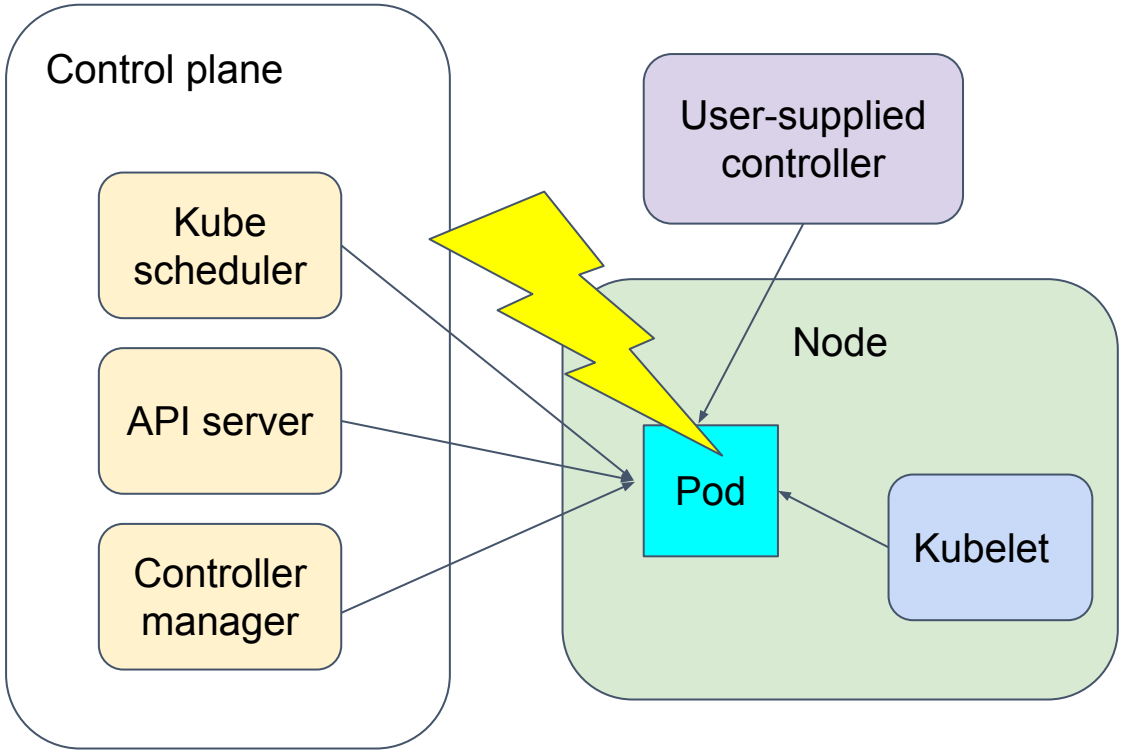
- **DisruptionTarget** is added to indicate the disruption reason
  - Scheduler preemption
  - Eviction API
  - Untolerated NoExecute taint
  - Deletion by PodGC
  - Termination by Kubelet
    - Graceful shutdown
    - Node pressure



Pod disruption scenarios

Pod failure policy (pod conditions)

- **DisruptionTarget** is added to indicate the disruption reason
  - Scheduler preemption
  - Eviction API
  - Untolerated NoExecute taint
  - Deletion by PodGC
  - Termination by Kubelet
    - Graceful shutdown
    - Node pressure
- **Custom conditions** can be added by webhooks or user-supplied controllers
- **Related work**: New Condition for Pending Pods that are stuck (KEP-3815)



Pod disruption scenarios

Pod failure policy (example)

- **ConfigIssue** third-party condition added for:
  - unresolvable container image
  - invalid config map configuration

```
backoffLimit: 6
podFailurePolicy:
  rules:
  - action: FailJob
    onPodConditions:
    - type: ConfigIssue
  - action: Ignore
    onPodConditions:
    - type: DisruptionTarget
  - action: FailJob
    onExitCodes:
      operator: In
      values: [1,2,126,127,128,139]
  - action: Ignore
    onExitCodes:
      operator: In
      values: [130,137,138,147]
```

Pod failure policy (example)

- **ConfigIssue** third-party condition added for:
  - unresolvable container image
  - invalid config map configuration

- Use **DisruptionTarget** to retry any disruptions

```
backoffLimit: 6
podFailurePolicy:
  rules:
  - action: FailJob
    onPodConditions:
    - type: ConfigIssue
  - action: Ignore
    onPodConditions:
    - type: DisruptionTarget
  - action: FailJob
    onExitCodes:
      operator: In
      values: [1,2,126,127,128,139]
  - action: Ignore
    onExitCodes:
      operator: In
      values: [130,137,138,147]
```

Pod failure policy (example)

- **ConfigIssue** third-party condition added for:
  - unresolvable container image
  - invalid config map configuration
- Use **DisruptionTarget** to retry any disruptions
- React to exit codes following the **TFJob** convention

```
backoffLimit: 6
podFailurePolicy:
  rules:
  - action: FailJob
    onPodConditions:
    - type: ConfigIssue
  - action: Ignore
    onPodConditions:
    - type: DisruptionTarget
  - action: FailJob
    onExitCodes:
      operator: In
      values: [1,2,126,127,128,139]
  - action: Ignore
    onExitCodes:
      operator: In
      values: [130,137,138,147]
```

Pod failure policy

- **Our Use Case:** Large embarrassingly-parallel jobs (10k+), where every task needs to complete before the result can be used

- **Before** podFailurePolicy, we tried using Kubernetes but…
  - No backoffLimit = Too many failures
  - backoffLimit = Runs take hours

- **With** podFailurePolicy, we get the best of both worlds. Runs that complete in **half the time** with **no failures**

```
backoffLimit: 5
 podFailurePolicy:
   rules:
     - action: Count
       onPodConditions:
         - type: DisruptionTarget
     - action: FailJob
       onExitCodes:
         operator: NotIn
         values: [0]
```

Acknowledgements: rescale

Summary

- **Indexed Jobs use cases**
  - Embarrassingly parallel jobs
  - Distributed ML training
  - Agent-environment simulations
  - HPC environment (Flux Operator)

Summary

- **Indexed Jobs use cases**
  - Embarrassingly parallel jobs
  - Distributed ML training
  - Agent-environment simulations
  - HPC environment (Flux Operator)

- **Try them out yourself**
  - https://github.com/google/learn-oss-with-google
  - https://github.com/flux-framework/flux-operator

Summary

- **Indexed Jobs use cases**
  - Embarrassingly parallel jobs
  - Distributed ML training
  - Agent-environment simulations
  - HPC environment (Flux Operator)

- **Try them out yourself**
  - https://github.com/google/learn-oss-with-google
  - https://github.com/flux-framework/flux-operator

- **New features of interests**
  - Pod Failure Policy
  - …

How to get involved?

Batch WG:

- Biweekly meetings, Thursdays 4pm CET
- slack.k8s.io #wg-batch
- wg-batch@k8s.io
- git.k8s.io/community/wg-batch

Questions?

Michał Woźniak
- Email: michalwozniak@google.com
- GitHub: @mimowo
- Slack: mimowo

Vanessa Sochat
- Email: sochat1@llnl.gov
- GitHub: @vsoch
- Slack: @v

Please scan the QR Code above to leave feedback on this session

CNCF slack: **#2-kubecon-sessions**