
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

On-Demand Systems and Scaled Training Using the JobSet API

Abdullah Gharaibeh - Google
Vanessa Sochat - Lawrence Livermore National Laboratory

Batch at Kubernetes

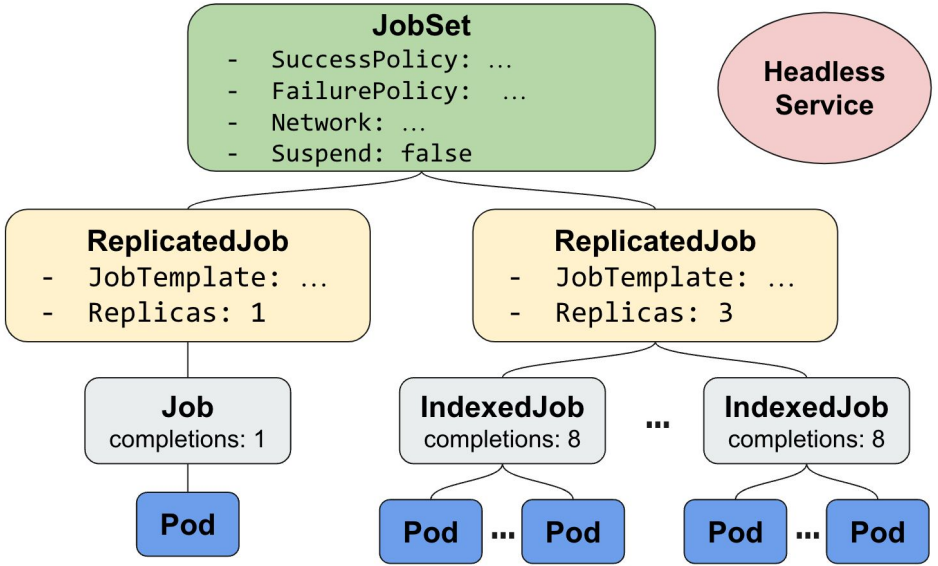
- Kubernetes has become a home for batch workloads
- The **upstream Job API** is now a feature rich and scalable batch API
- Indexed mode enabled new batch use cases in distributed training and HPC

But, **and there is always a “but” …**

we can do even better

JobSet

- A new batch API that uses Job as a building block
- Manages a group of Jobs as a single “workload”
- Automates multiple training and HPC workloads patterns
  - Multi-template pods
  - Pod-to-pod communication
  - Failure and success policy
  - Exclusive placement of child Jobs per domain



JobSet Use Cases

- Scaled distributed training on TPUs
- Building HPC applications with JobSet

Scaled Distributed Training on TPUs

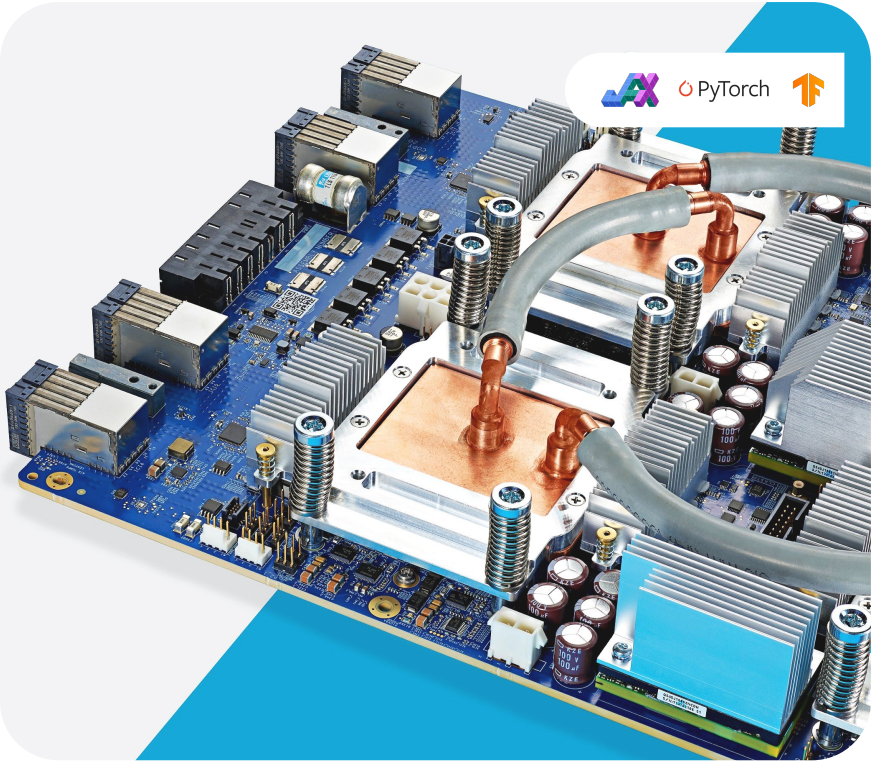
What is TPU?

"Tensor Processing Unit"

AI supercomputing Infrastructure designed by Google **specifically for Machine Learning (ML)**

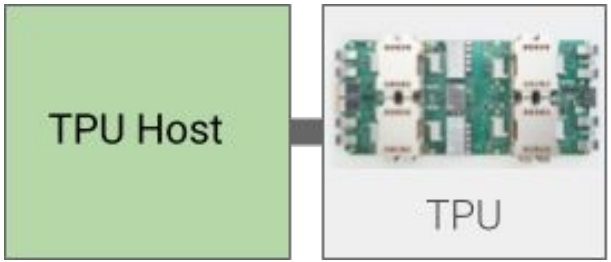
Provides **compelling performance per dollar at scale** for various ML workloads, including **Gen AI, NLP, Computer vision and Recommendations**

Productive development with TPU VMs supporting PyTorch, JAX, TensorFlow and integration with Kubernetes



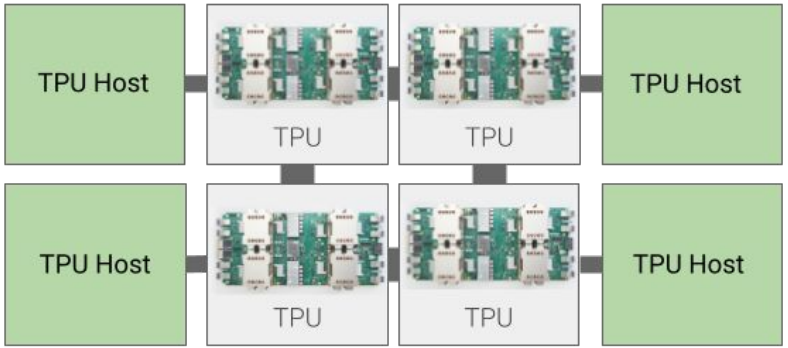
TPUs in the Cloud

TPU device



- No special network connections to other devices in other nodes
- Lifecycle is independent of other nodes

TPU slice

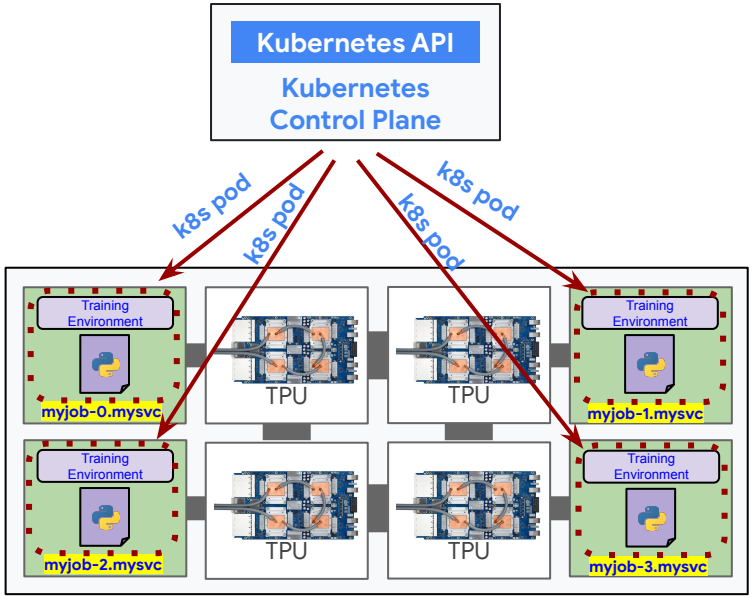


- TPU devices across nodes are connected using a high-speed interconnect (ICI links)
- Lifecycle of all nodes are tied together
- Can be provisioned in different shapes

Distributed Training on TPU Slices

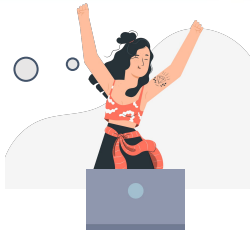
Training setup

- A pod per node representing the workers
- A unique id for each pod
- pod-to-pod communication
- Fail the whole job if any worker fails



Distributed Training on TPU Slices

I know how to do this, use **Indexed Job**!



my-job.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  parallelism: 4
  completions: 4
  completionMode: Indexed
  backoffLimit: 0
  template:
    spec:
      subdomain: svc
      containers:
      - name: job
        env:
        - name: TPU_WORKER_ID
          valueFrom:
            fieldpath:
metadata.annotations['batch.kubernetes.io/job-completion-index']
        - name: TPU_WORKER_HOSTNAMES
          value: myjob-0.svc,myjob-1.svc,myjob-2.svc,myjob-3.svc
```

1 Number of workers

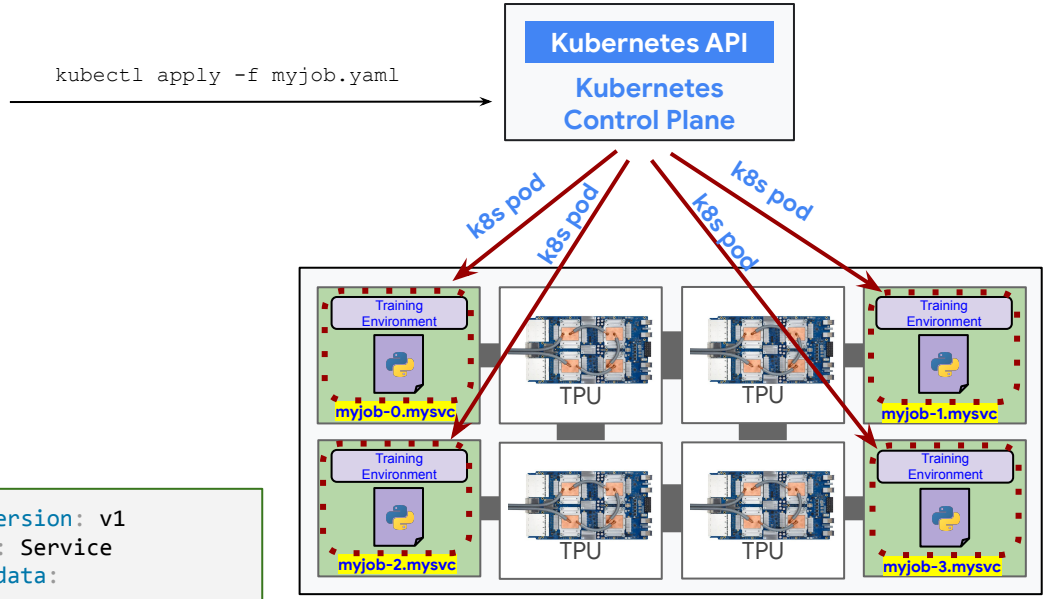
2 Fail the job if any worker fails

3 Enables stable dns hostnames for pods

4 ENV vars for libtpu (distributed comms library) setup

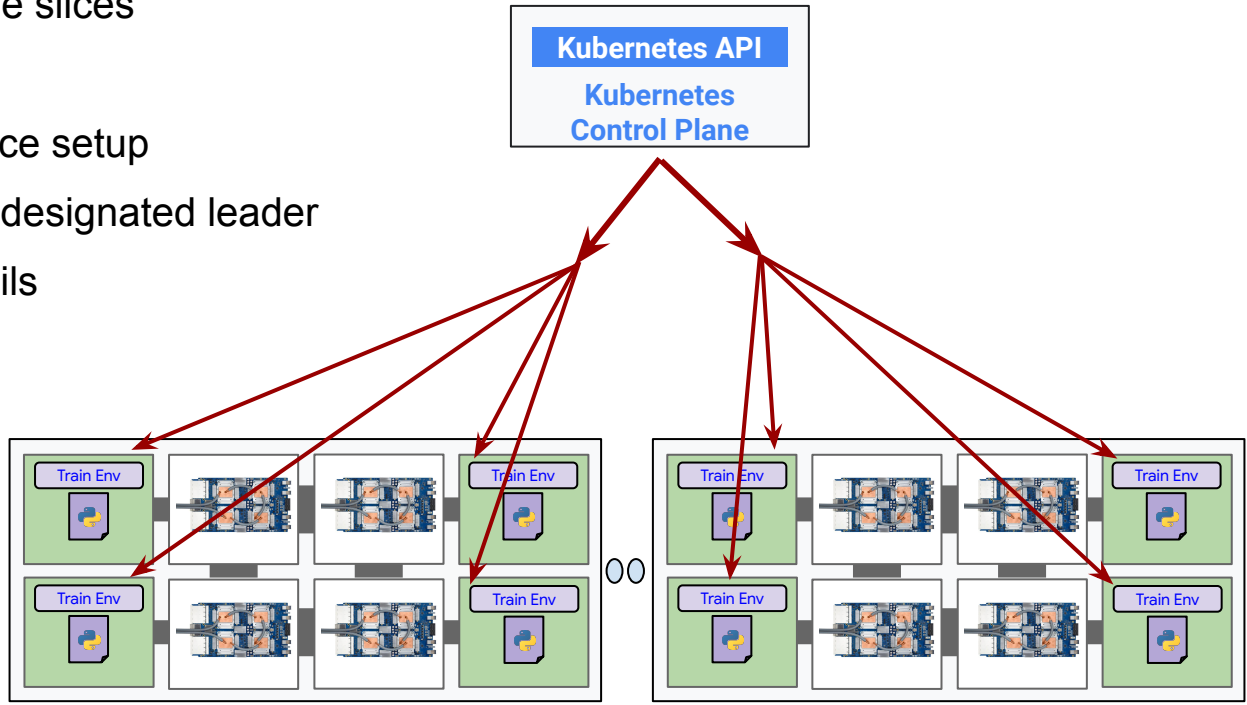
```
apiVersion: v1
kind: Service
metadata:
  name: svc
spec:
  clusterIP: None
  selector:
      job-name: myjob
```

kubectl apply -f myjob.yaml



Scaled Distributed Training

- LLMs are getting even larger
- Training is sharded across multiple slices
- Two levels of orchestration
  - within a slice: same as single slice setup
  - across slices: coordinated via a designated leader
- Fail the whole job if any worker fails



MegaScale Distributed Training

I know how to do this, use **Indexed Job for each slice**!

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: job-0
spec:
  parallelism: 4
  completions: 4
  completionMode: Indexed
  backoffLimit: 0
  template:
    spec:
      subdomain: mysvc
      containers:
      - name: job
        env:
        - name: TPU_WORKER_ID
          valueFrom:
            fieldpath:
metadata.annotations['batch.kubernetes.io/job-completion-index']
        - name: TPU_WORKER_HOSTNAMES
          value:
job-0-0.mysvc,job-0-1.mysvc,job-0-2.mysvc,job-0-3.mysvc
        - name: TPU_MEGASCALE_SLICES
          value: 16
        - name: TPU_MEGASCALE_SLICE_ID
          value: 0
        - name: TPU_MEGASCALE_COORDINATOR
          value: job-0-0.mysvc
```

New ENV vars for multi-slice setup

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: job-1
spec:
  parallelism: 4
  completions: 4
  completionMode: Indexed
  backoffLimit: 0
  template:
    spec:
      subdomain: mysvc
      containers:
      - name: job
        env:
        - name: TPU_WORKER_ID
          valueFrom:
            fieldpath:
metadata.annotations['batch.kubernetes.io/job-completion-index']
        - name: TPU_WORKER_HOSTNAMES
          value:
job-1-0.mysvc,job-1-1.mysvc,job-1-2.mysvc,job-1-3.mysvc
        - name: TPU_MEGASCALE_SLICES
          value: 2
        - name: TPU_MEGASCALE_SLICE_ID
          value: 1
        - name: TPU_MEGASCALE_COORDINATOR
          value: job-0-0.mysvc
```

Scaled Distributed Training

But wait, this is hard to manage!

- What if the training workload spans 100s of slices?
- How do I monitor the status of the whole job?
- How do I fail all the child jobs if anyone fails?
- How do I ensure that each job lands exclusively on a slice?

MegaScale TPU Training on GKE

- Scale: 12736 nodes, 50,944 chips
  - 199 TPU slices → number of IndexedJobs
  - 64 nodes per slice → number of pods per Job
  - 4 chips per node
- Orchestration
  - JobSet: To deploy and manage the training jobs
  - Kueue: Manages queueing and resource sharing (kueue.sigs.k8s.io)

MegaScale Training using JobSet


```yaml
apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  annotations:
    alpha.jobset.sigs.k8s.io/exclusive-topology: tpu-slice-id
  name: megascale
spec:
  failurePolicy:
    maxRestarts: 4
  successPolicy:
    operator: all
  replicatedJobs:
  - name: slice
    replicas: 199
    template:
      spec:
        parallelism: 64
        completions: 64
        template:
          spec:
            containers:
            - name: job
              command: ...
              env:
                - name: TPU_WORKER_ID
                  valueFrom:
                    fieldpath:
metadata.annotations['batch.kubernetes.io/job-completion-index']
                - name: TPU_MEGASCALE_SLICES
                  valueFrom:
                    fieldpath:
metadata.annotations['jobset.sigs.k8s.io/replicatedjob-replicas']
                - name: TPU_MEGASCALE_SLICE_ID
                  valueFrom:
                    fieldpath:
metadata.annotations['jobset.sigs.k8s.io/job-index']
                - name: TPU_MEGASCALE_COORDINATOR
                  value: megascale-slice-0-0.megascale-job
```

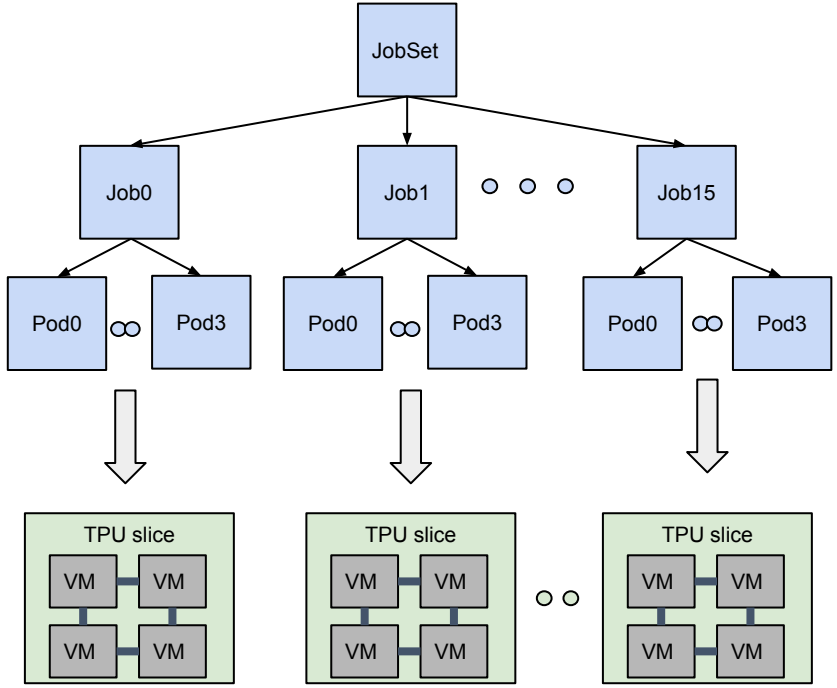
5 1:1 child job to slice assignment

3 restart up to 4 times

2 successful when all child jobs succeed

1 number of slices and nodes per slice

4 GKE injects automatically

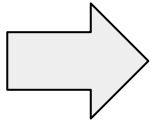


Nodes in the same slice are labelled with a common slice-id

Deep Dive: Placement Policy


```yaml
apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  annotations:
    alpha.jobset.sigs.k8s.io/exclusive-topology:
tpu-slice-id
  name: megascale
spec:

...
...
```



```yaml
kind: Job
  ...
    affinity:
      podAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
            - key: batch.kubernetes.io/job-name
              operator: In
              values:
              - megajob-slice-0  # job name
          topologyKey: tpu-slice-id  # the grouping domain
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions: # exclude the pods of this job
            - key: batch.kubernetes.io/job-name
              operator: NotIn
              values:
              - megaslice-slice-0
          namespaceSelector:
            matchExpressions: # repel only pods from other jobs
            - key: batch.kubernetes.io/job-name
              operator: Exists
          topologyKey: tpu-slice-id
```

Collocate the pods of a job

Repels all other jobs

Building HPC Applications with JobSet

Kubecon 2023, Chicago

Vanessa Sochat
Computer Scientist
Livermore Computing

LLNL-PRES-856096

How do I implement a job manager in Kubernetes?

How do I implement a job manager in Kubernetes?


flux


flux


flux

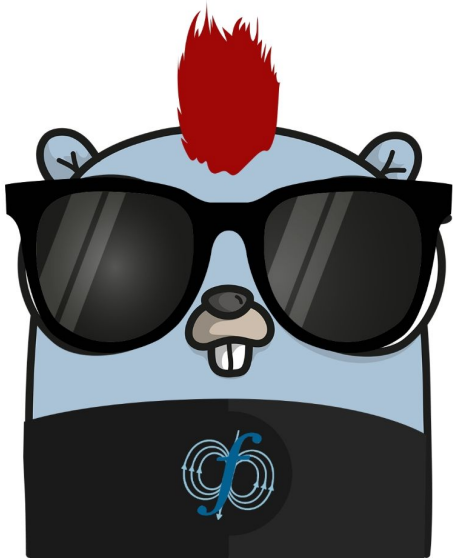

Cloud Land
HPC Land


Cloud Land
Converged Computing
HPC Land


Cloud Land
Converged Computing
flux
HPC Land


Cloud Land
Converged Computing
Yes!
Do you run apps and stuff?
flux
HPC Land


Cloud Land
Converged Computing
Hey I'm modular!
flux
Yarrr me too!
HPC Land



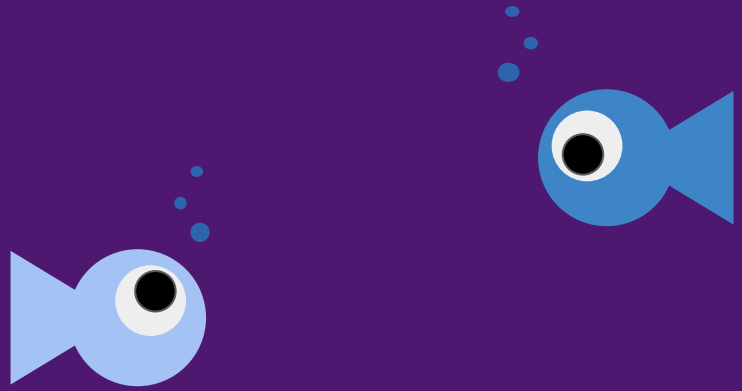
THE OPERATOR

coming to K8s near you...


flux

How do we run applications?

How are cloud and HPC applications different?



Application Coupling

Cloud Land

HPC Land

Application Coupling

Cloud Land

HPC Land

Application Coupling

Cloud Land

HPC Land

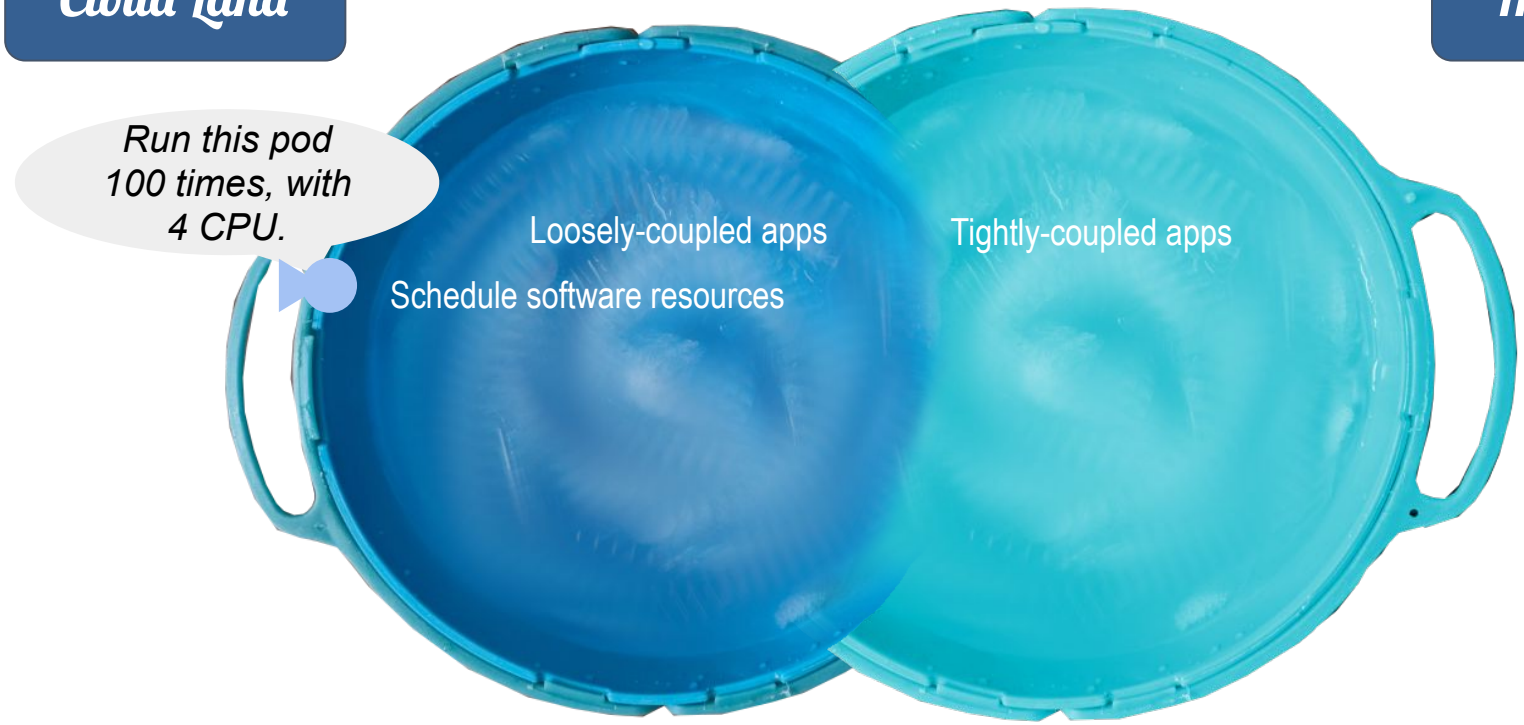
Run this app and database

Message Passing Interface (MPI)!

Resource Scheduling

Cloud Land

HPC Land

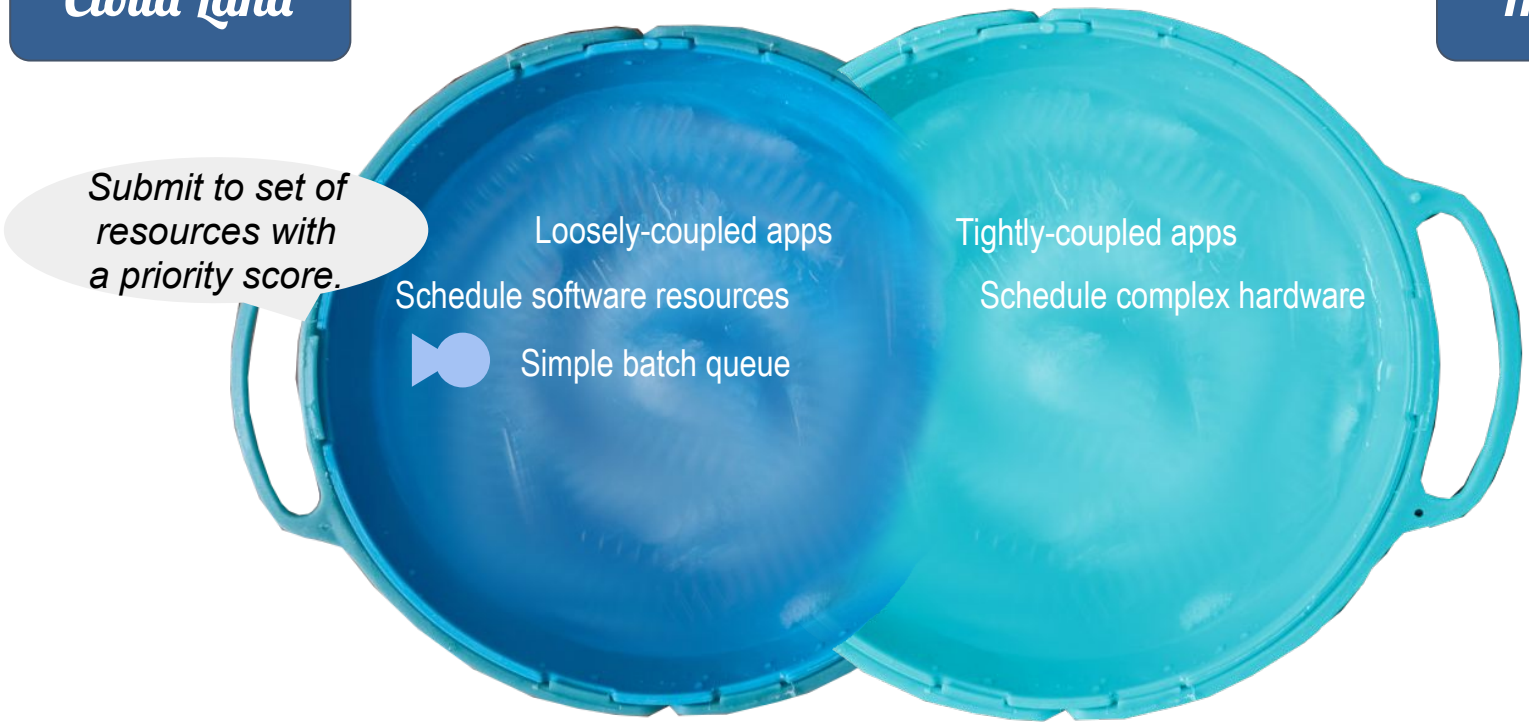


Resource Scheduling

Job Queuing

Cloud Land

HPC Land



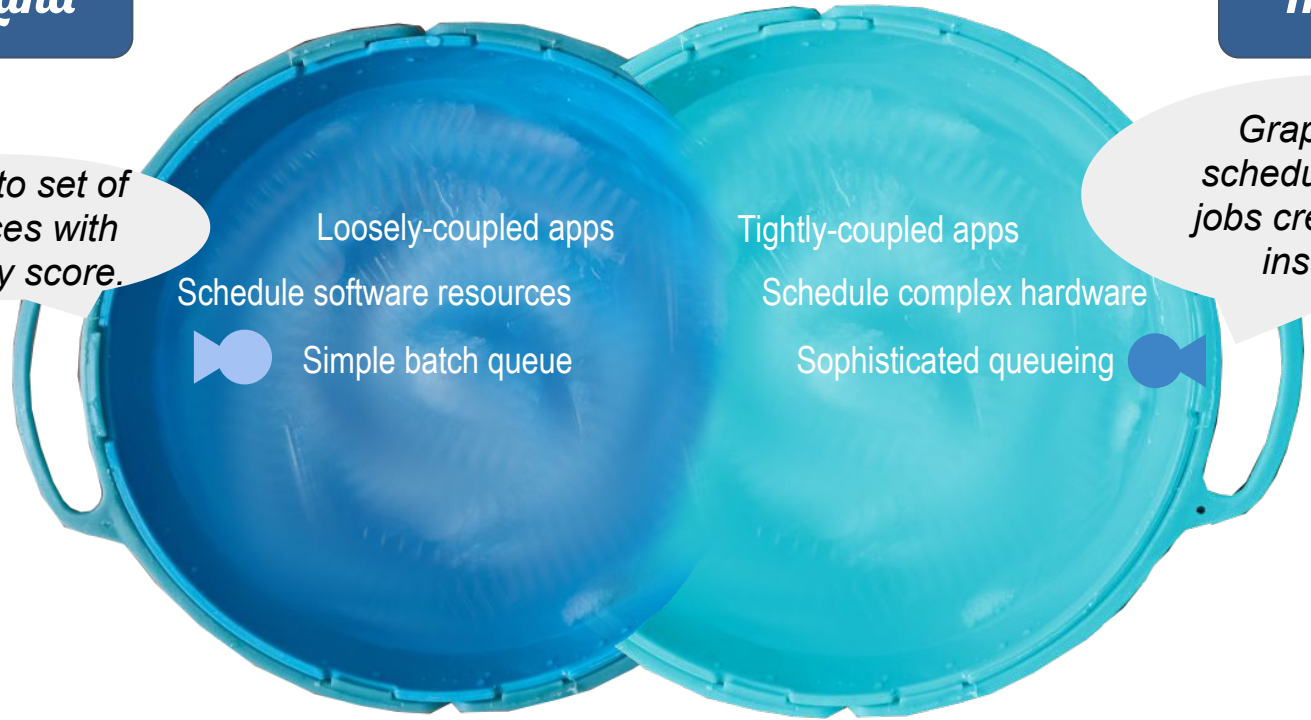
Job Queuing

Cloud Land

HPC Land

Submit to set of resources with a priority score.

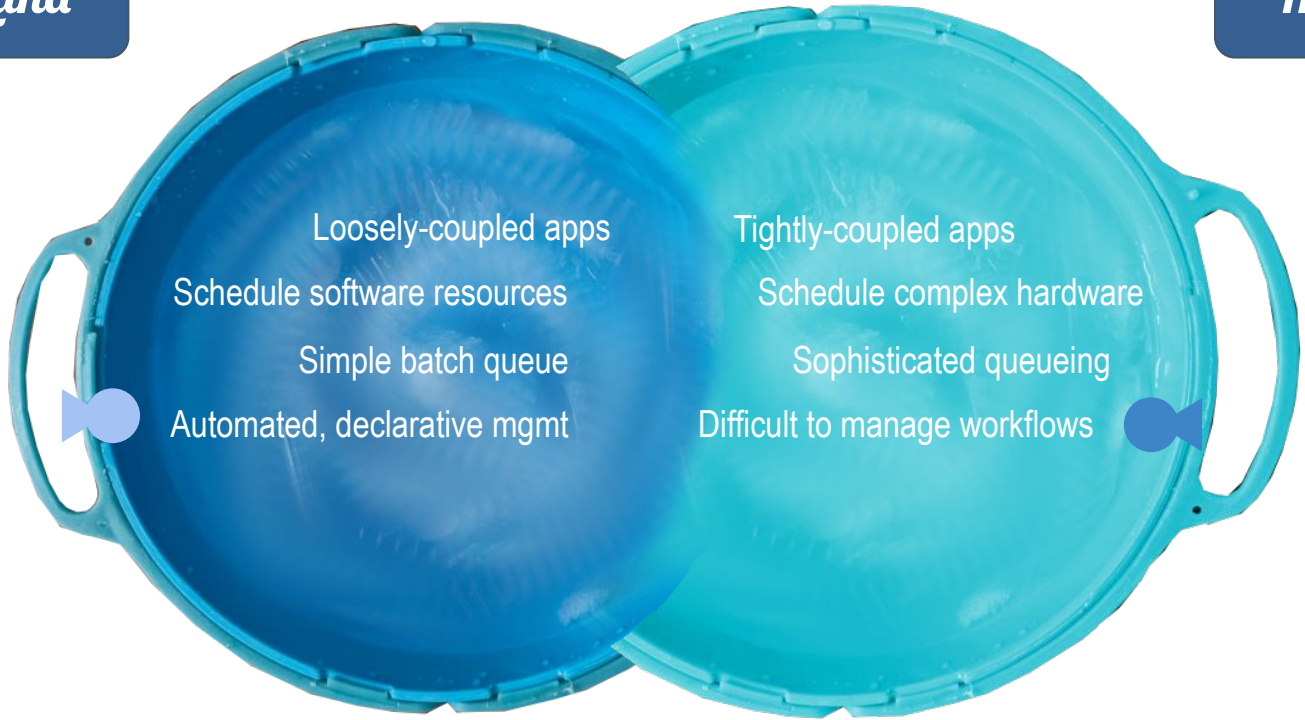
Graph-based scheduling: batch jobs create nested instances.



Workflow Management

Cloud Land

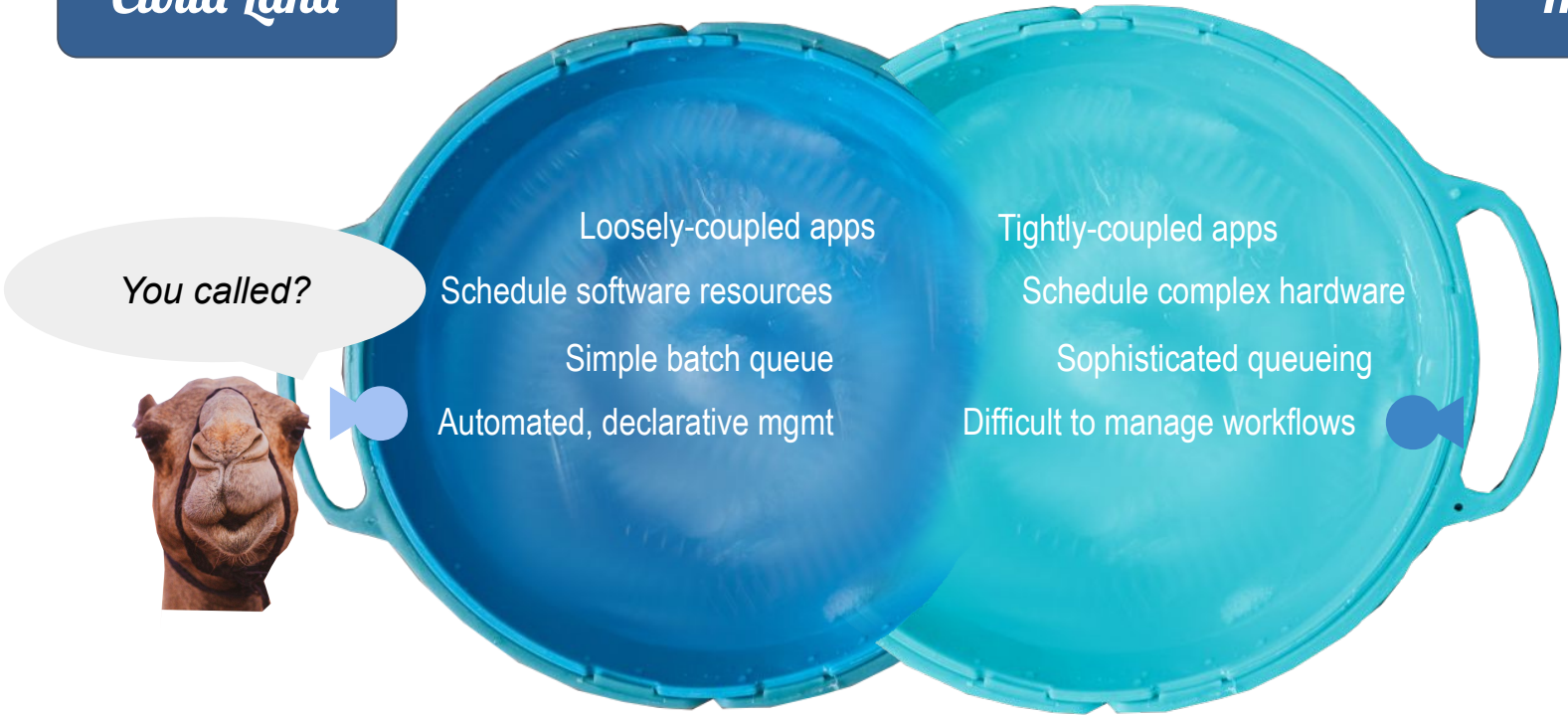
HPC Land



Workflow Management

Cloud Land

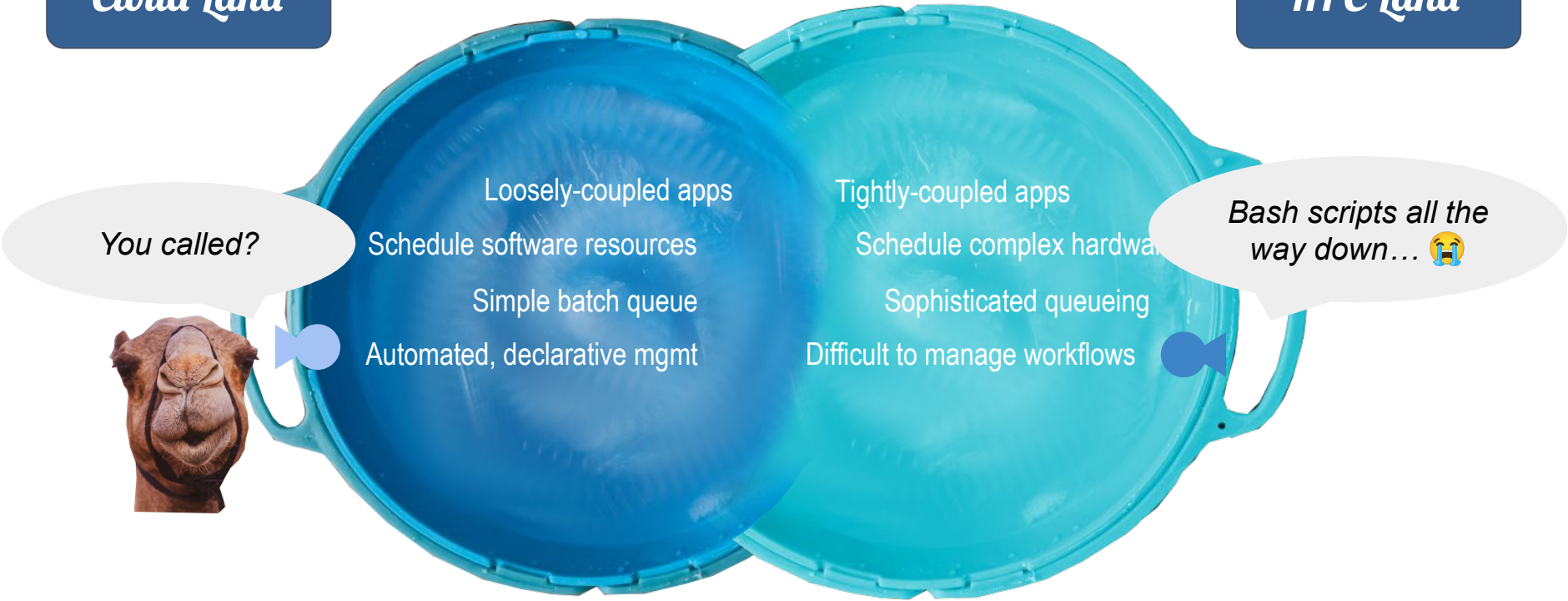
HPC Land



Workflow Management

Cloud Land

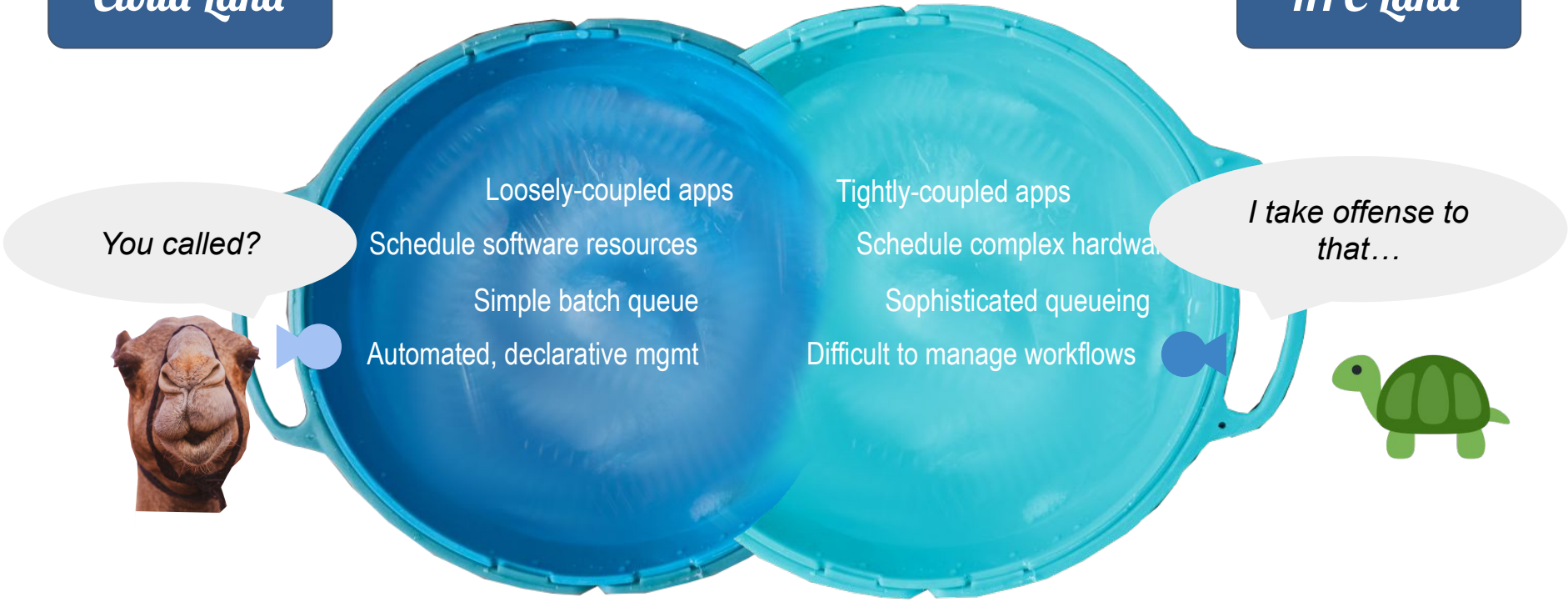
HPC Land



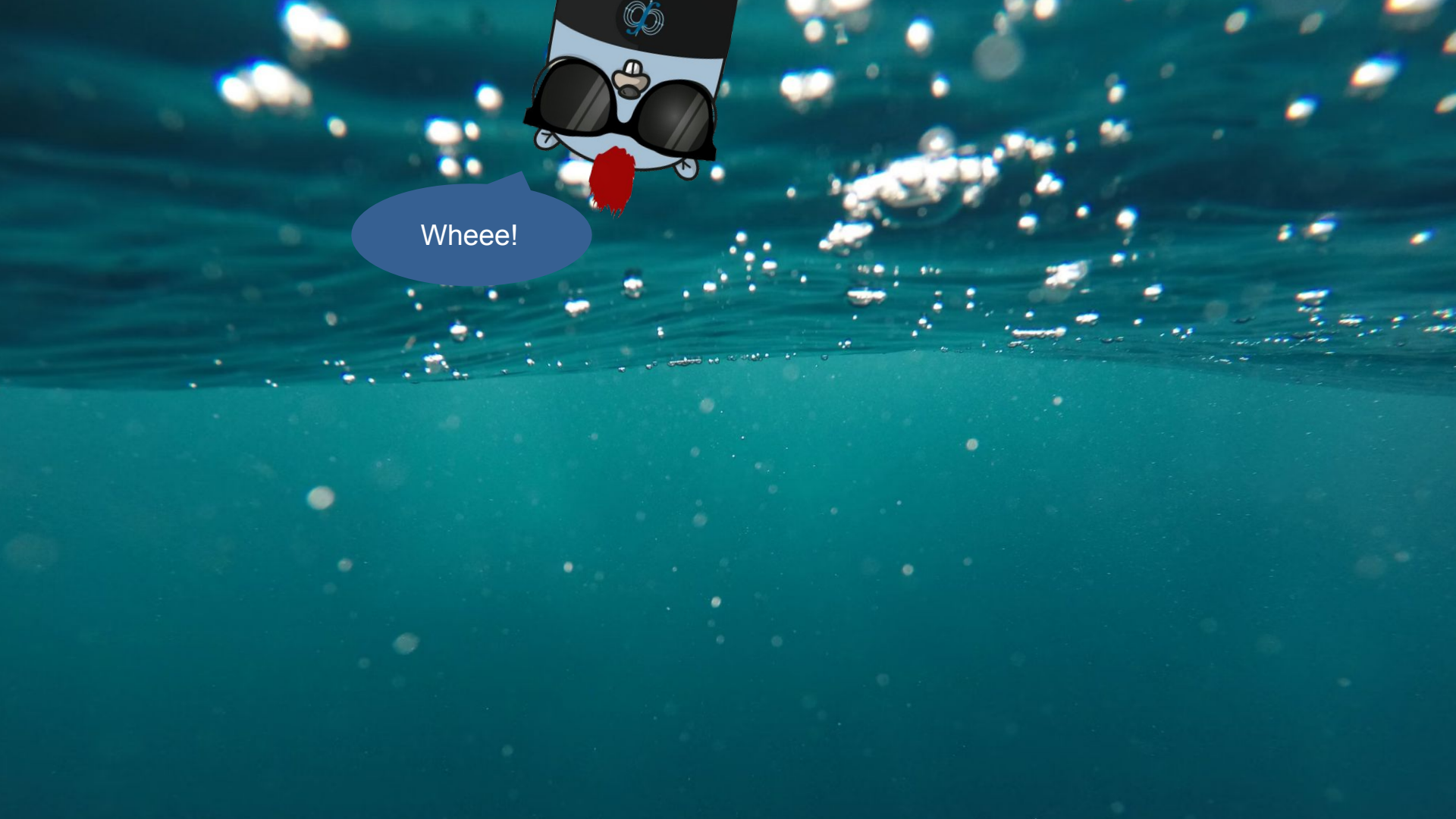
Workflow Management

Cloud Land

HPC Land

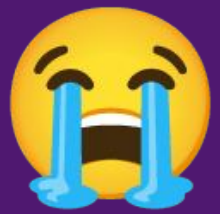



Cloud Land
Converged Computing
flux
HPC Land
Should we run some applications?

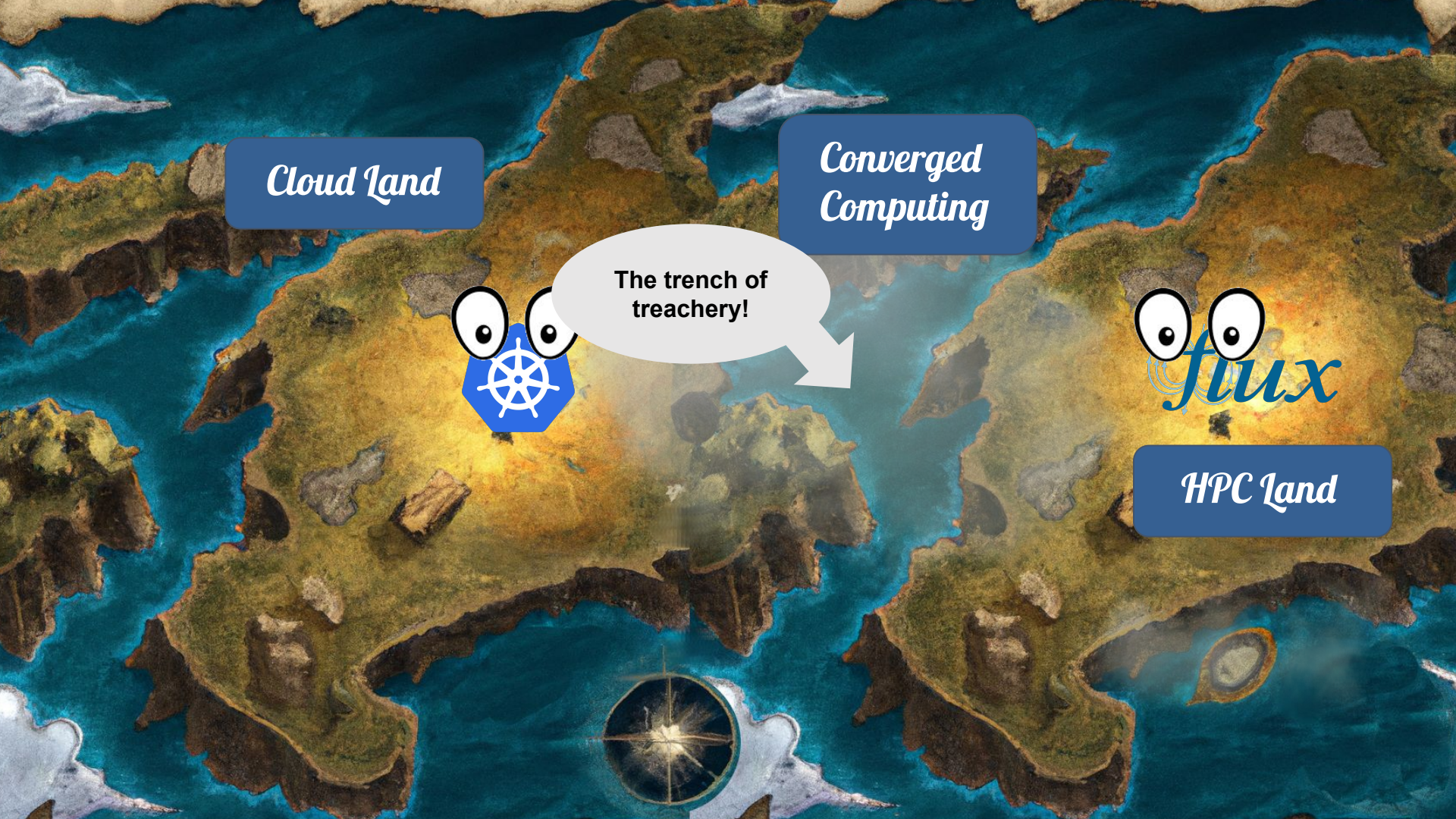

Wheee!

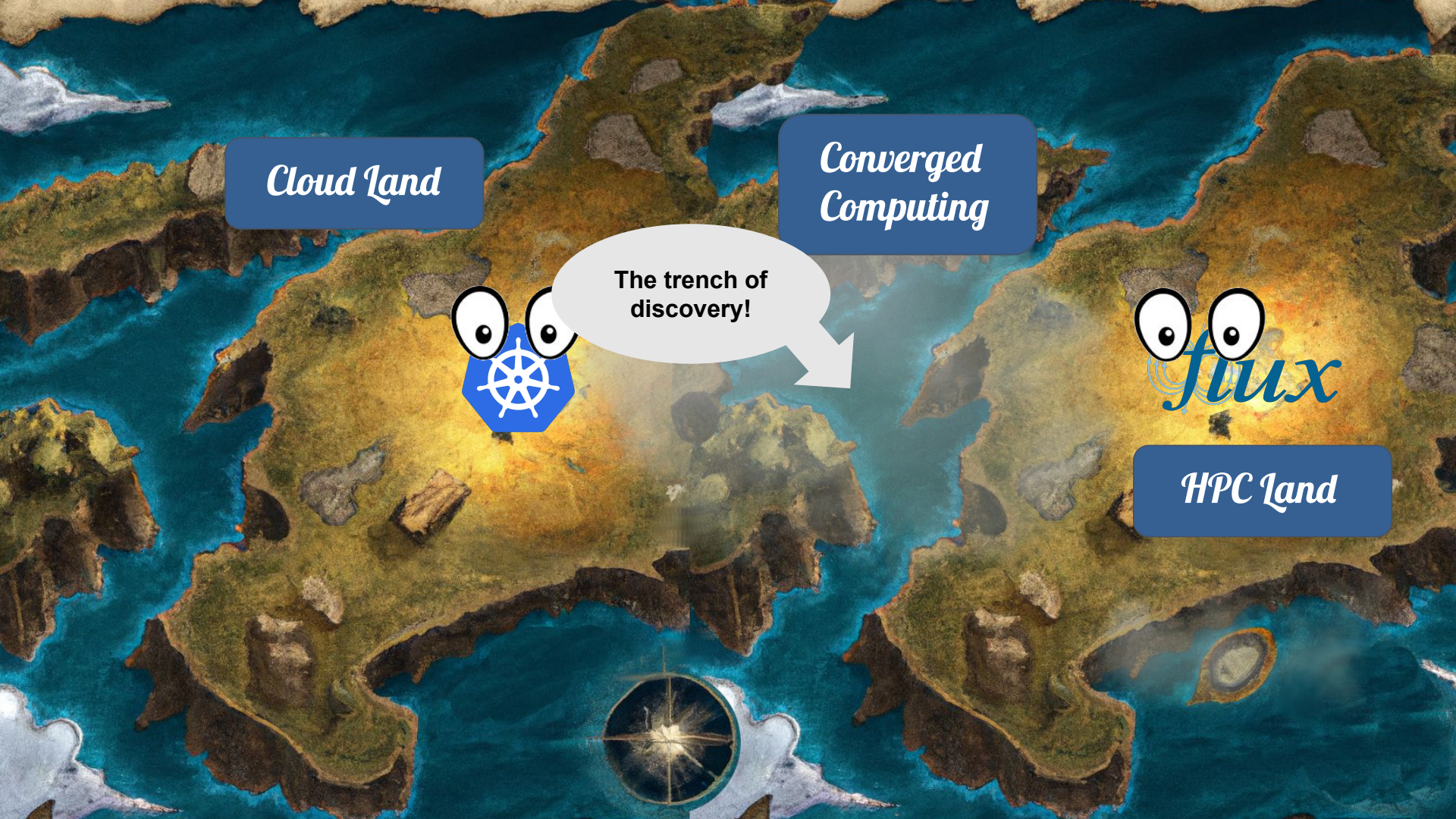
Our applications don't run.

Why doesn't it work?




Cloud Land
Converged Computing
flux
HPC Land


Cloud Land
Converged Computing
The trench of treachery!
flux
HPC Land


Cloud Land
Converged Computing
The trench of discovery!
flux
HPC Land

HPC Applications in Kubernetes Abstractions

An HPC Workload Manager

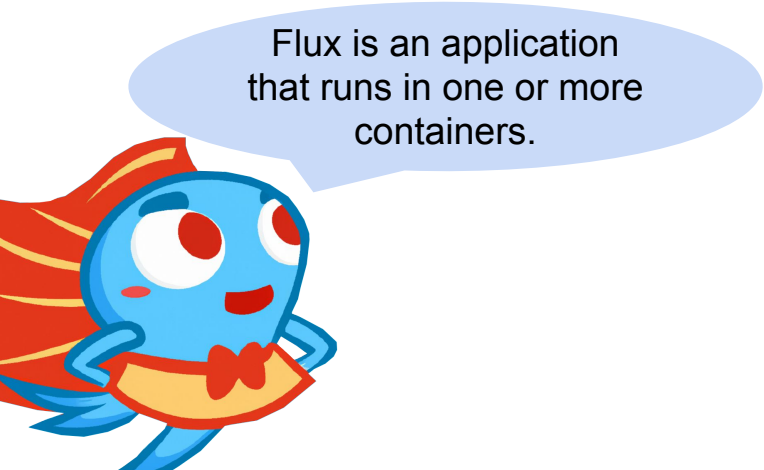
with an Indexed Job

Job

Pod

Container

Volume



An HPC Workload Manager

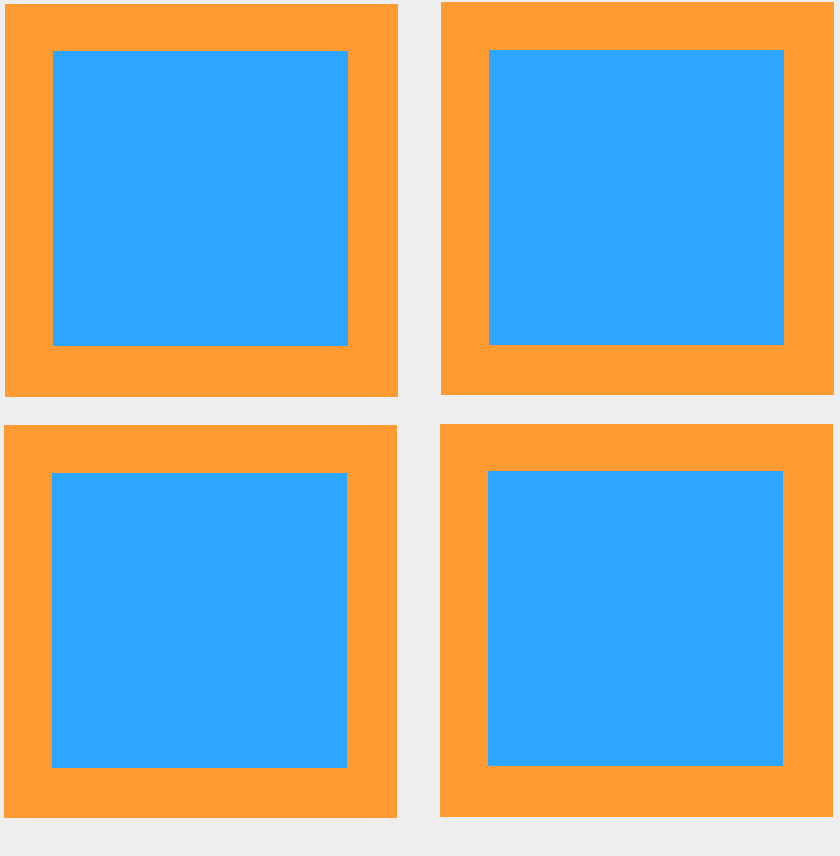
with an Indexed Job

Job

Pod

Container

Volume



Multiple Flux containers in different pods make a Flux cluster!

An HPC Workload Manager

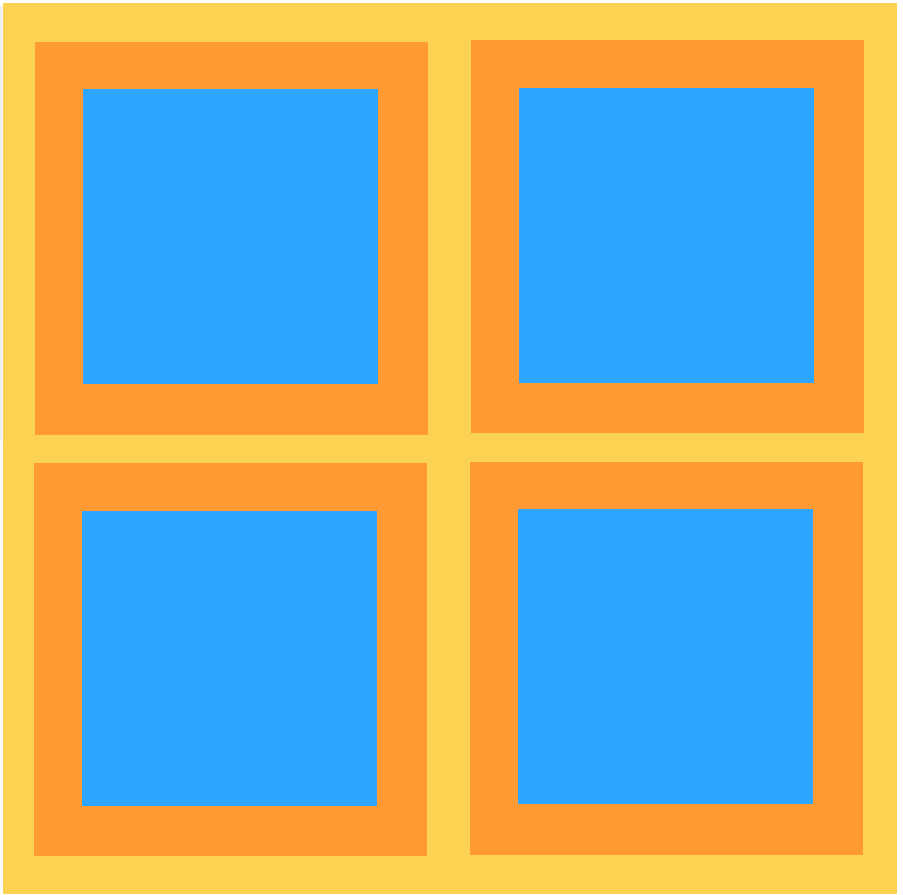
with an Indexed Job

Job

Pod

Container

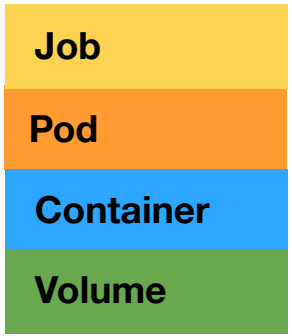
Volume

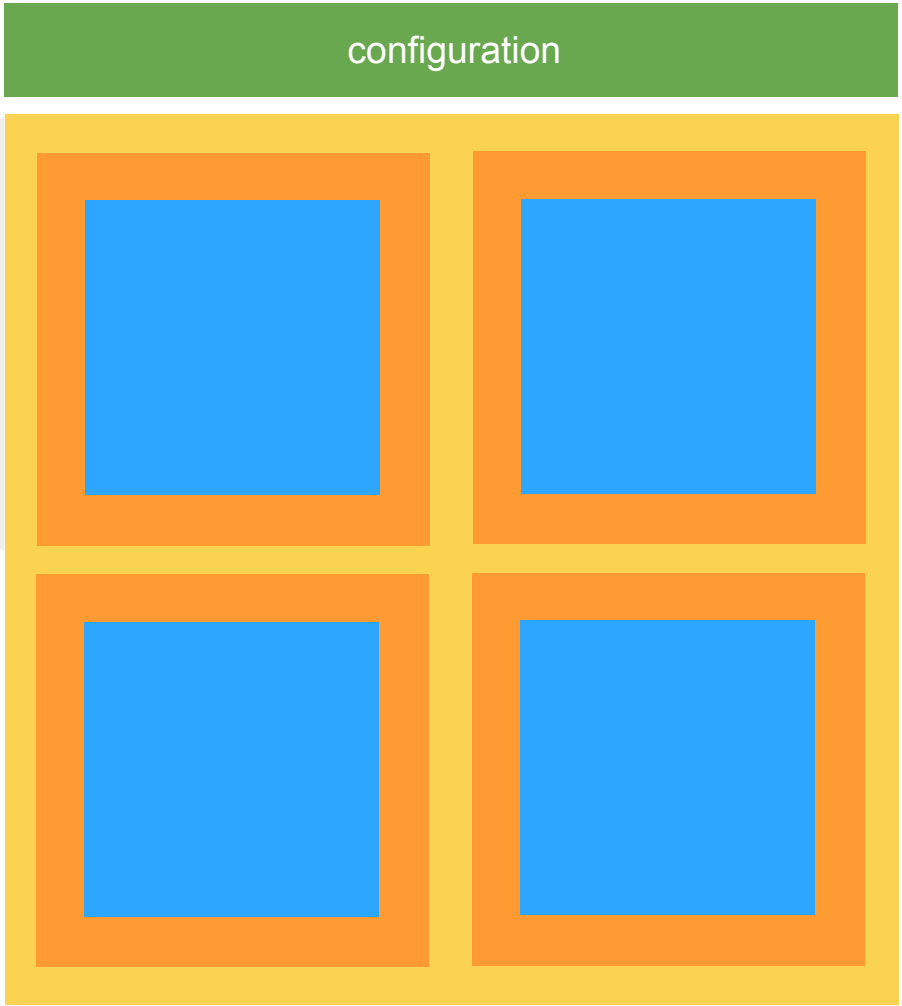


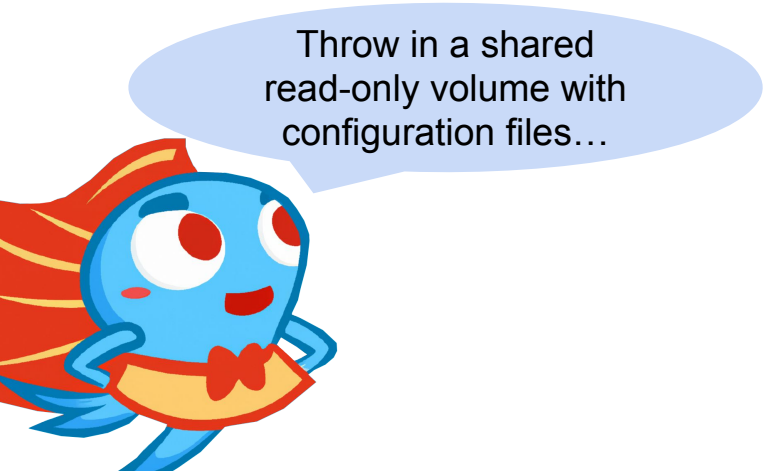
The indexed job creates these duplicated pods, the cluster "nodes"

An HPC Workload Manager

with an Indexed Job

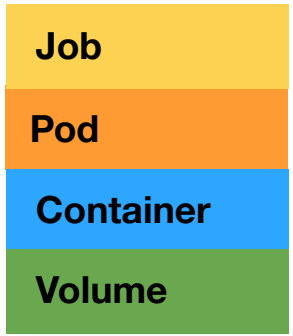


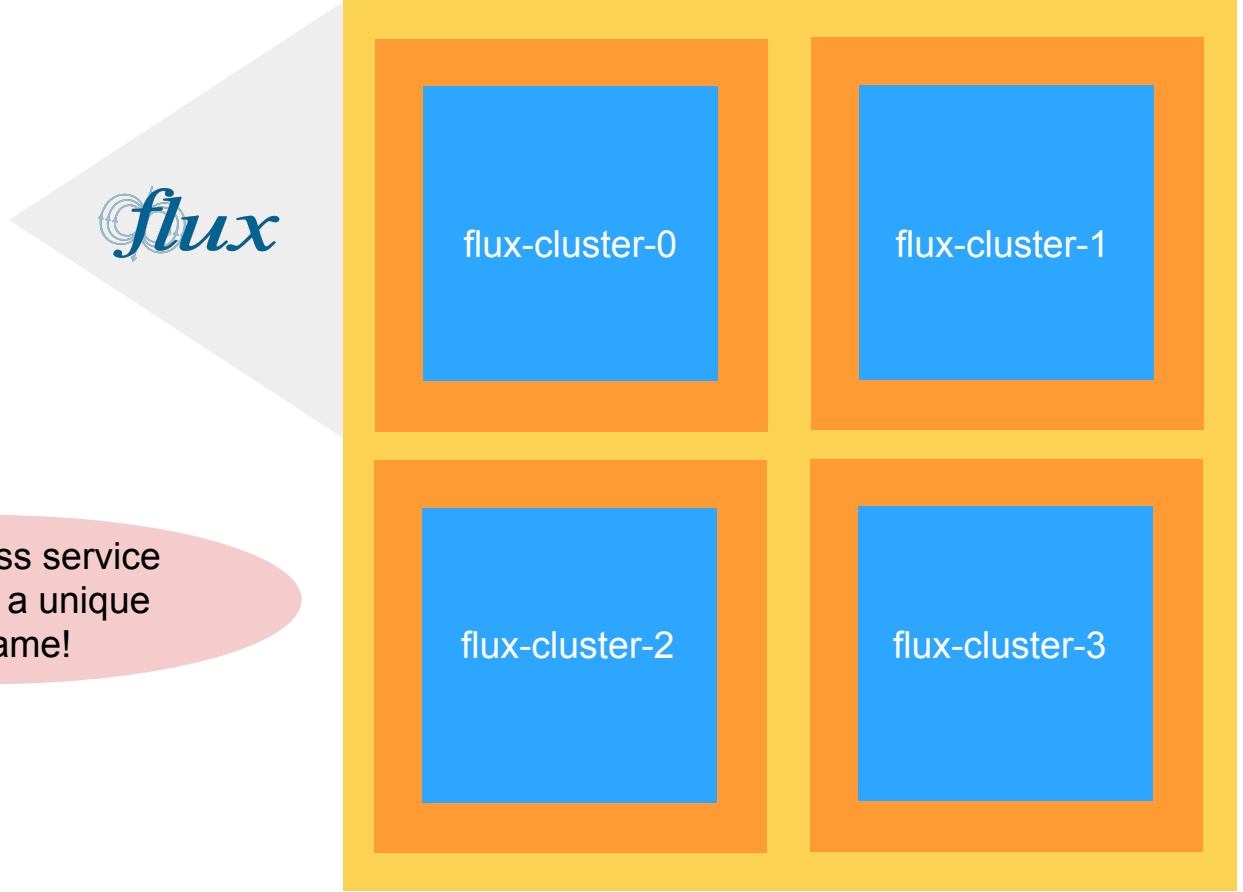




An HPC Workload Manager

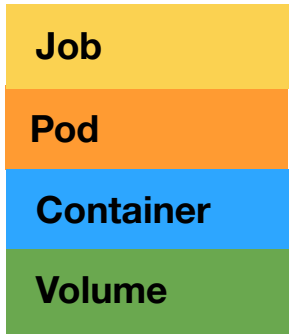
The Flux Operator

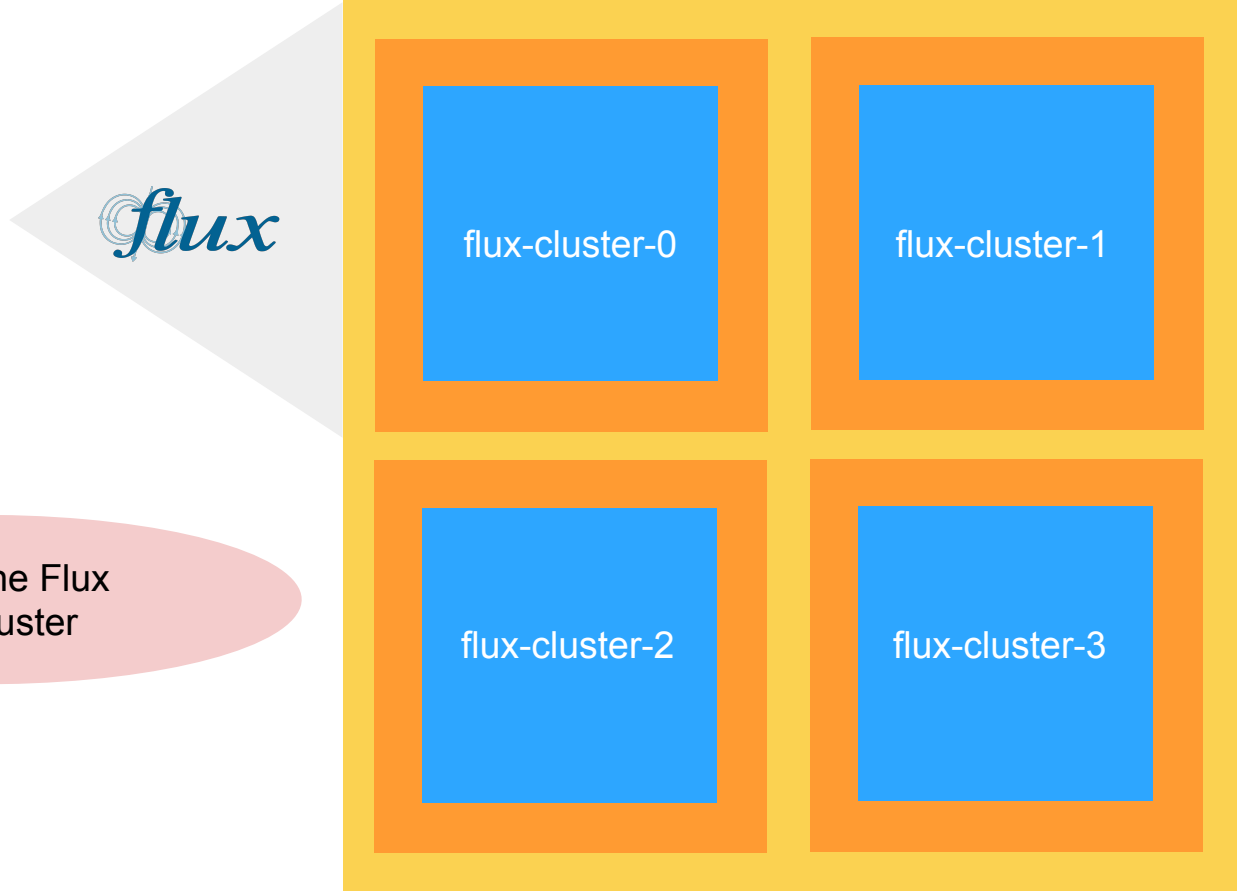


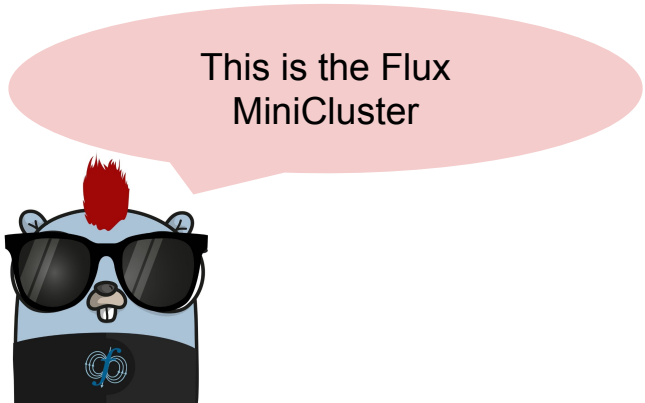


An HPC Workload Manager

The Flux Operator

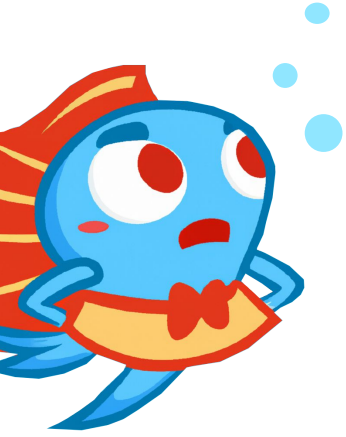


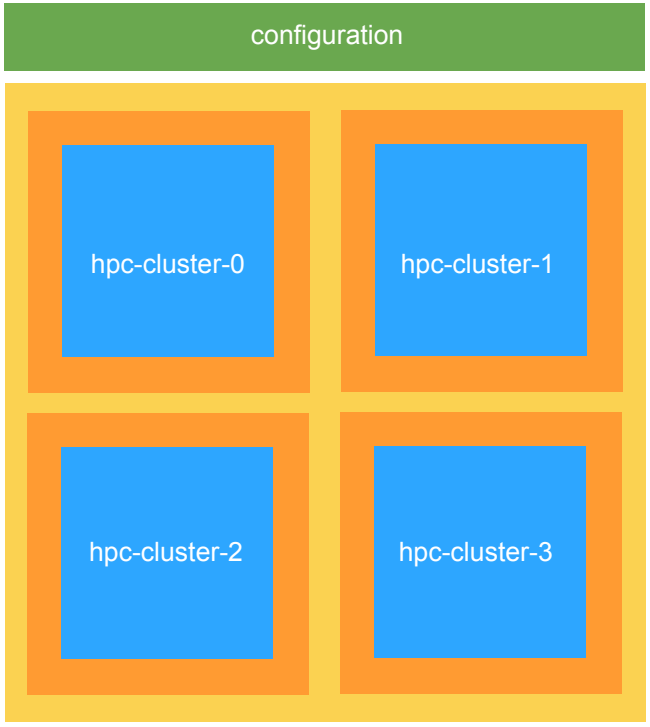




Problems with A Single Job

#developerproblems

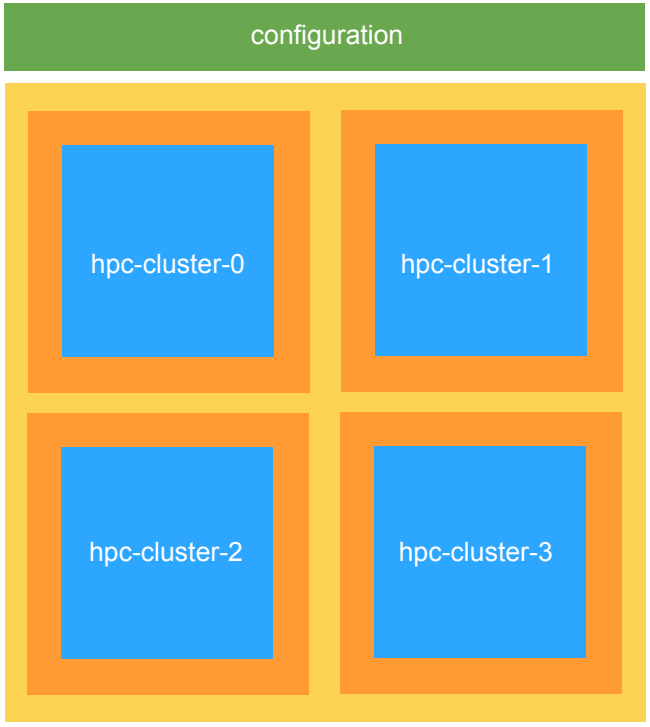




Problems with A Single Job

#developerproblems

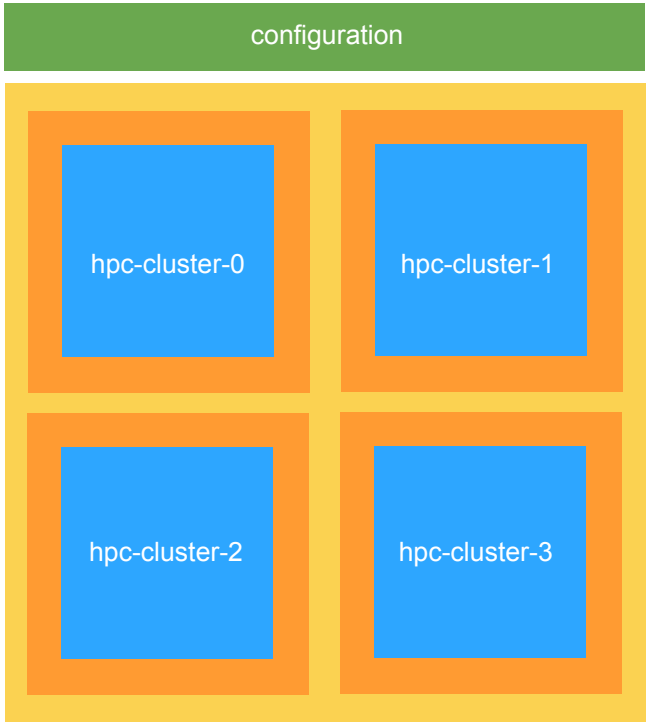
- Different entry point logic based on index



Problems with A Single Job

#developerproblems

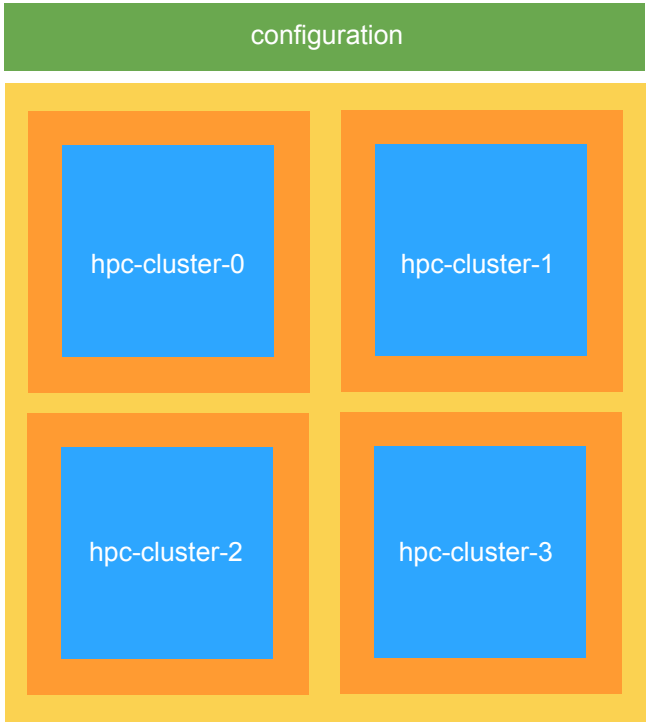
- Different entry point logic based on index
- Manual creation of headless service



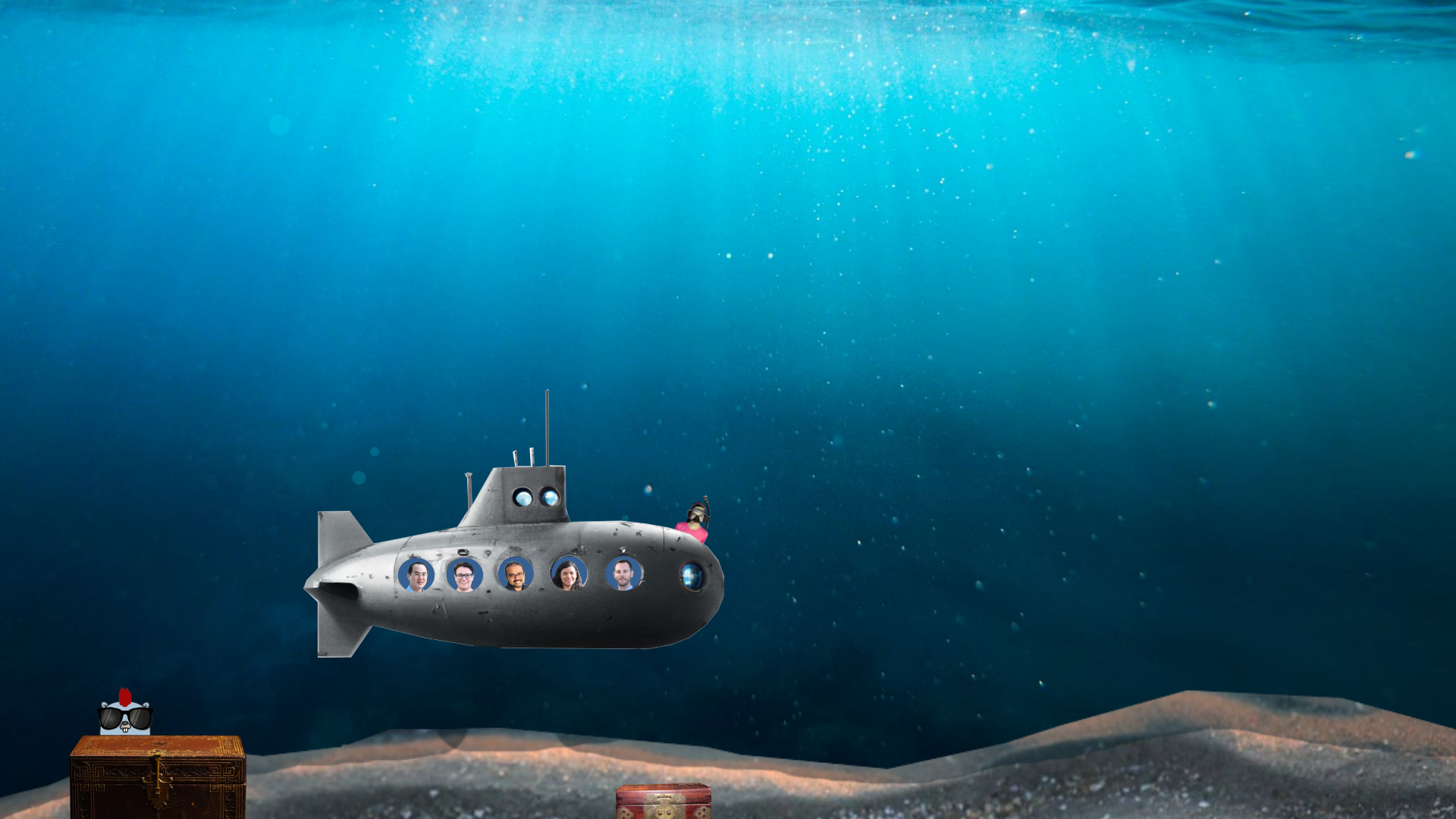
Problems with A Single Job

#developerproblems

- Different entry point logic based on index
- Manual creation of headless service
- We need success / failure policy




OK, we understand the limitations of Job…



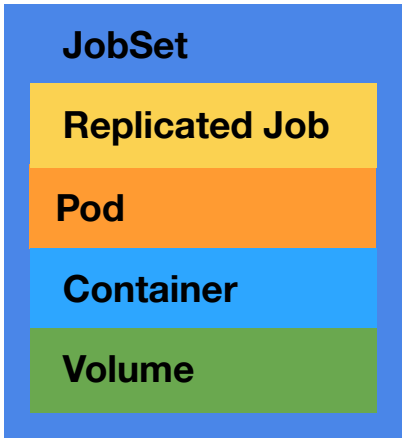

JobSet

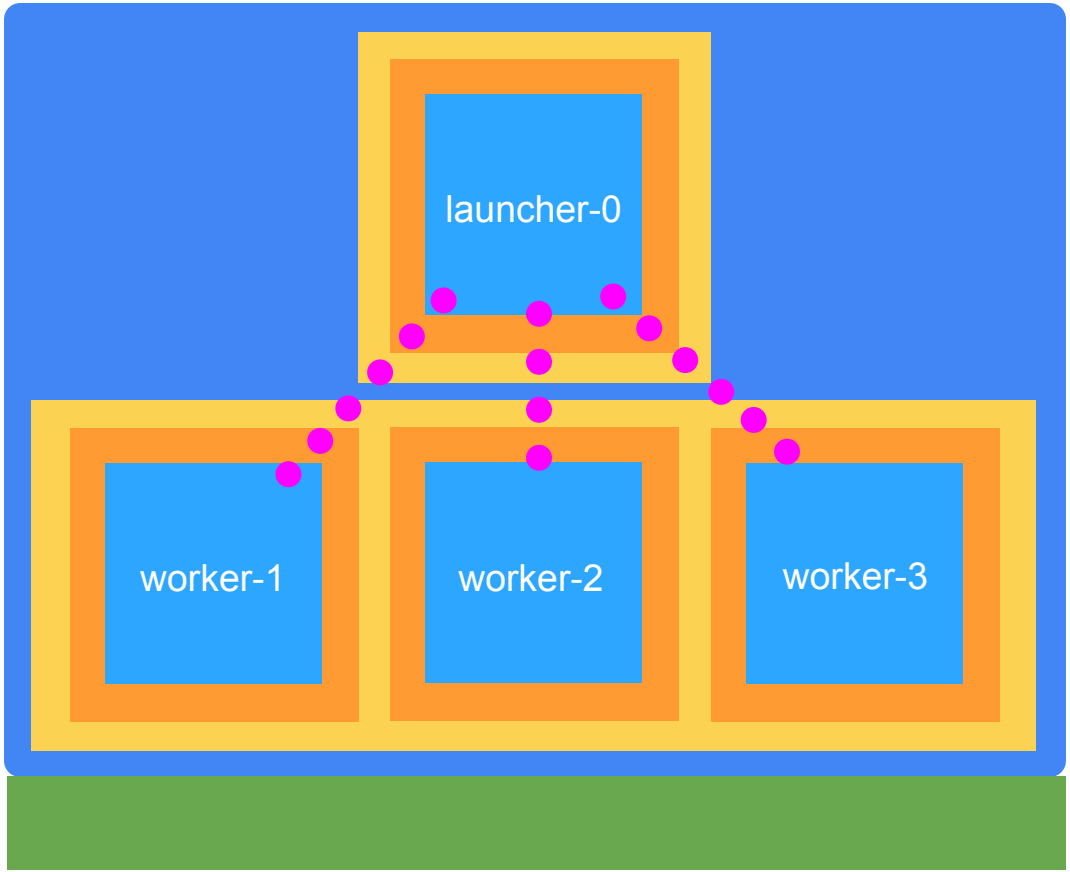

Hey V! 👋


Can I help?

An HPC Workload Manager

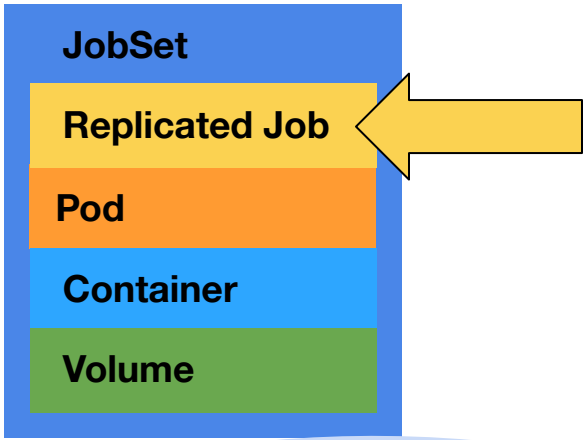
Using JobSet

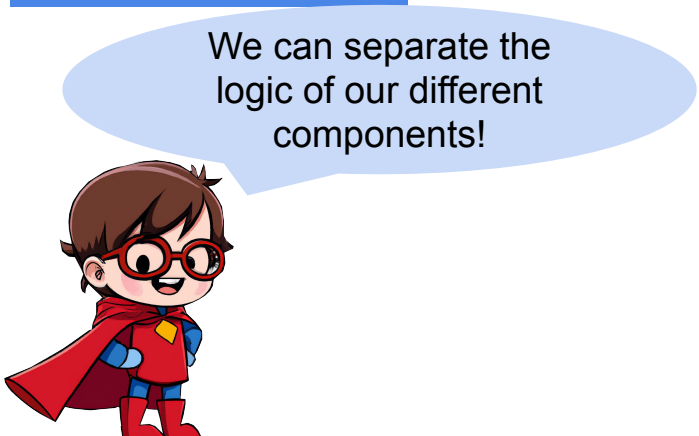


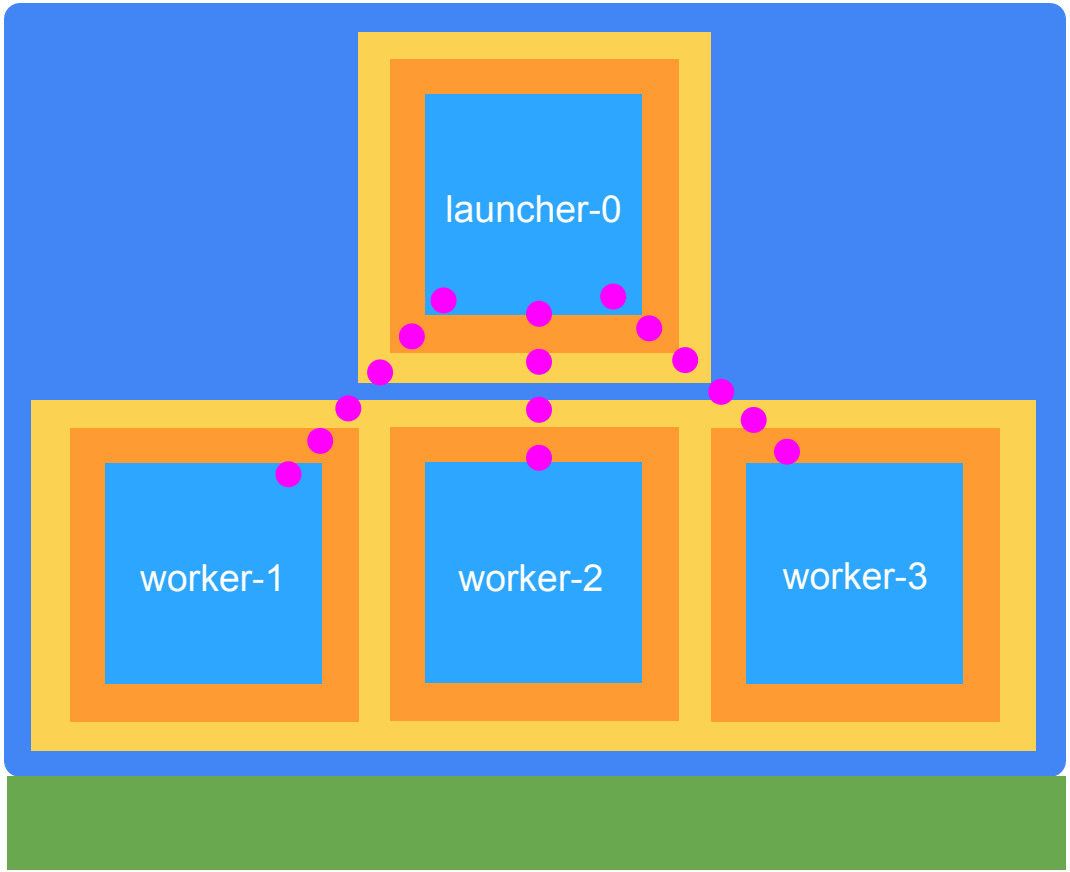


An HPC Workload Manager

Using JobSet

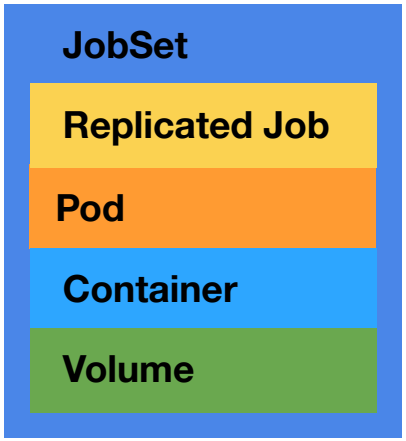


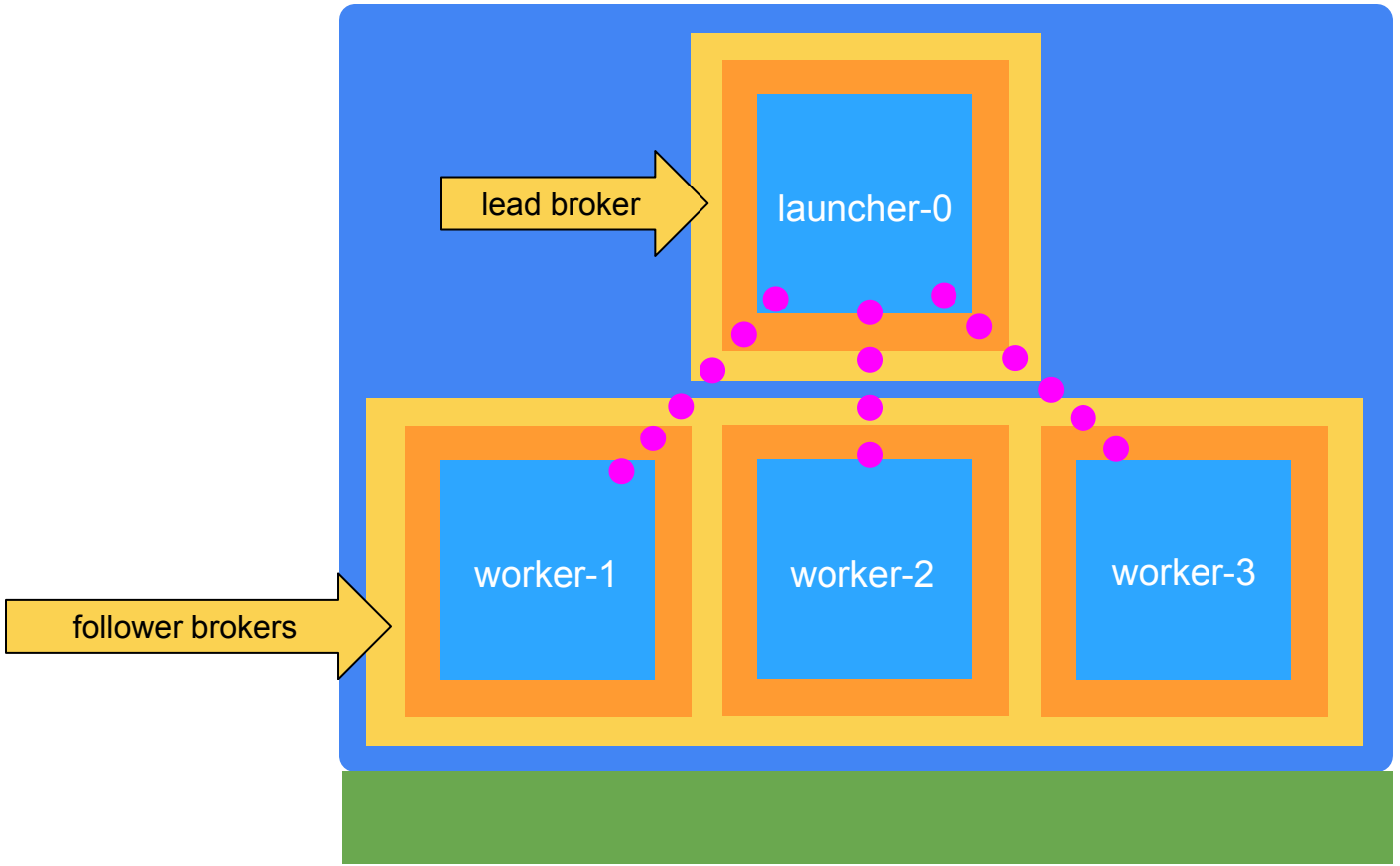




An HPC Workload Manager

Using JobSet





An HPC Workload Manager

Using JobSet

JobSet

Replicated Job

Pod

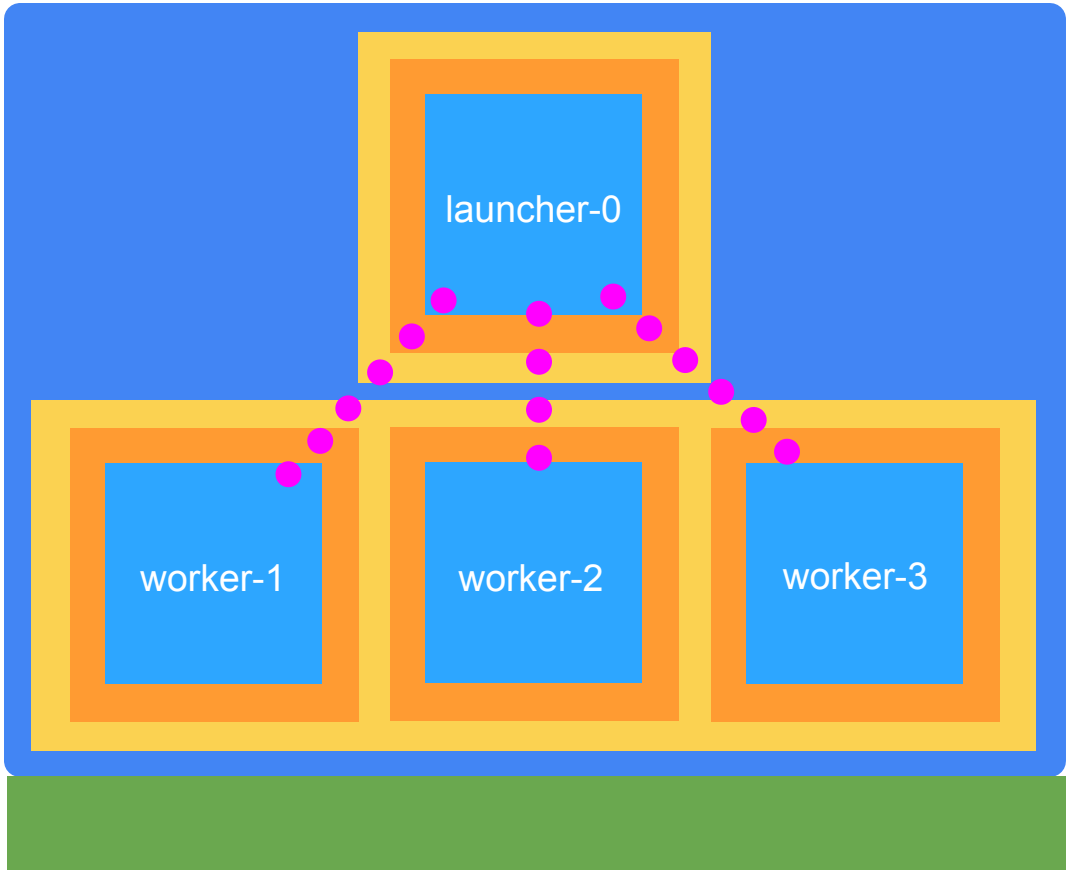
Container

Volume

They can still be on the same network.

hpc-cluster-0-0.hpc-svc-default.svc.cluster.local

<jobname>-<replicated-index>-<index>.<service>-<namespace>.svc.cluster.local



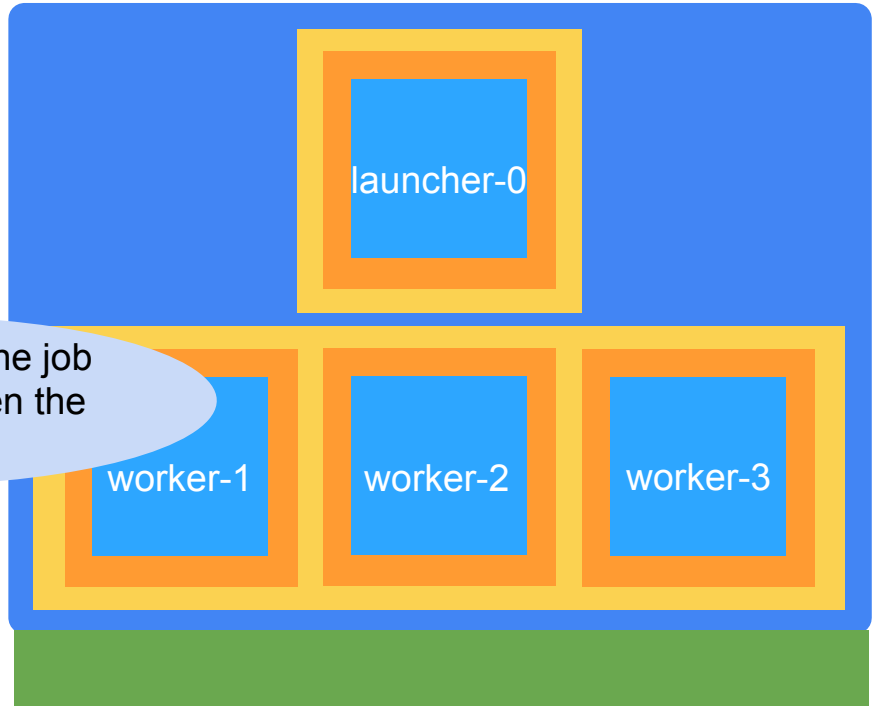
Solutions Offered By JobSet

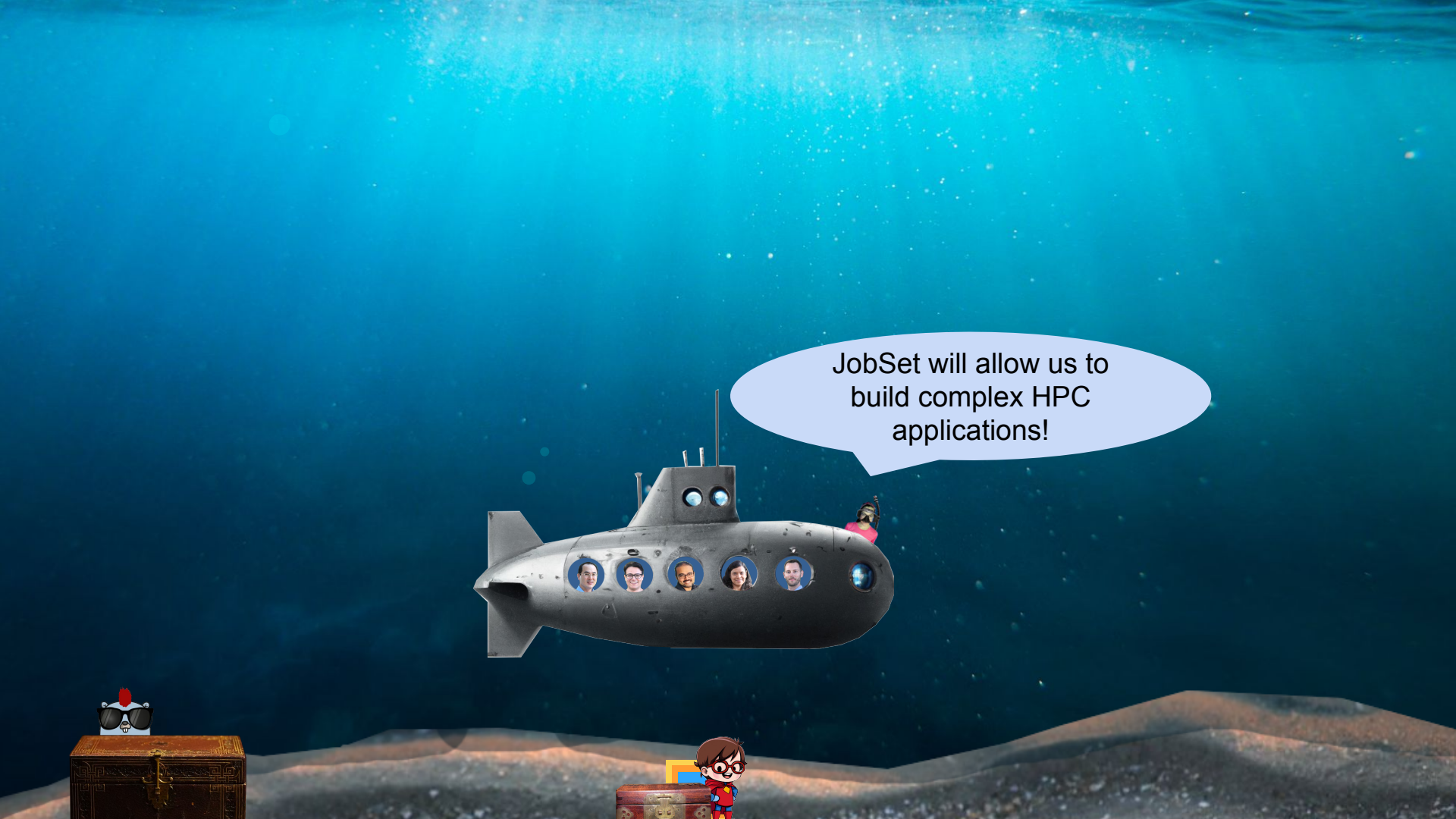
#developeryay!

- ✅ Different entry points logic for Replicated Jobs
- ✅ Automated creation of network
- ✅ Success and failure policies

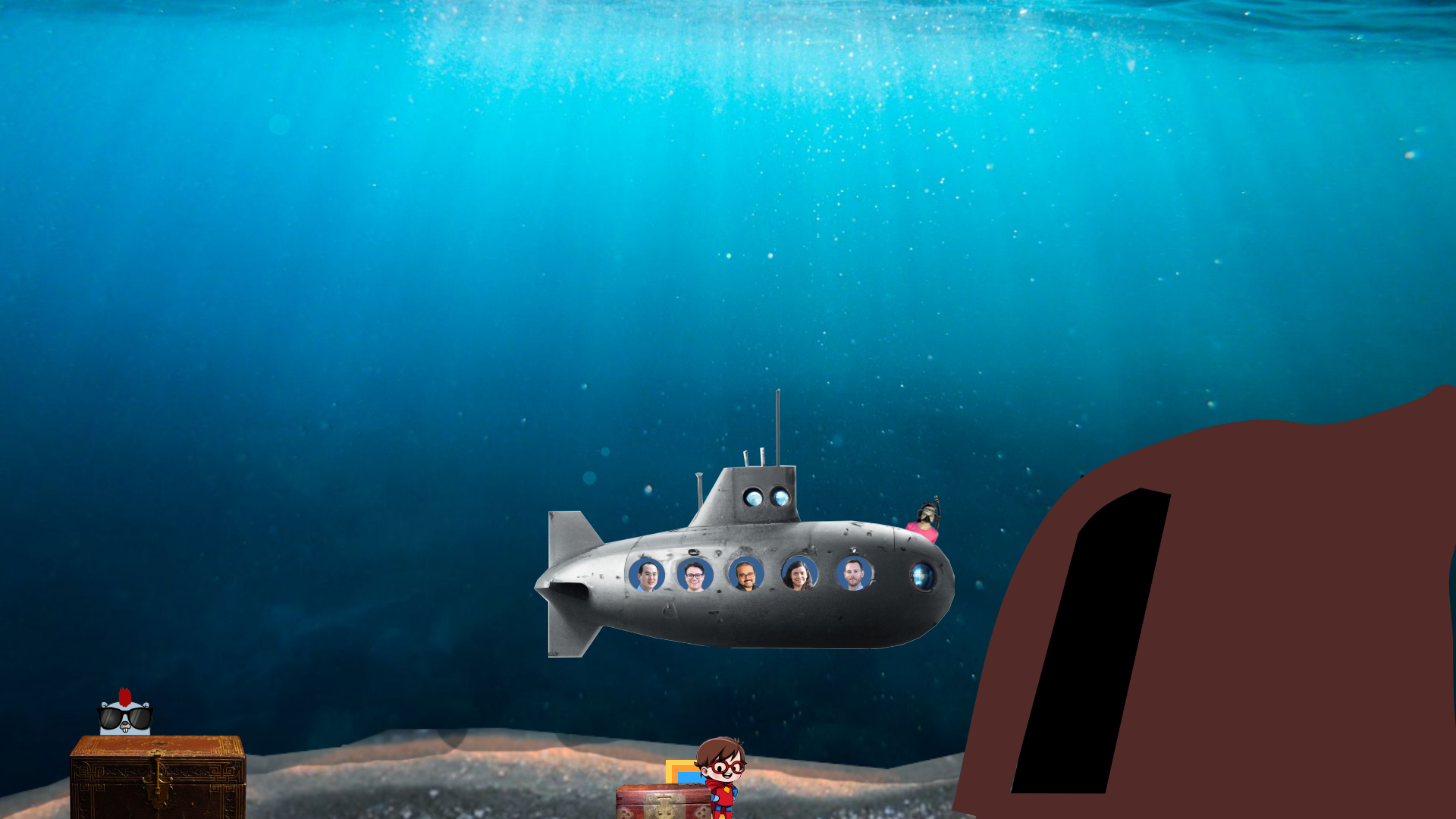
```yaml
apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  name: hpc-cluster
spec:
  successPolicy:
    operator: All
    targetReplicatedJobs:
    - launcher
```

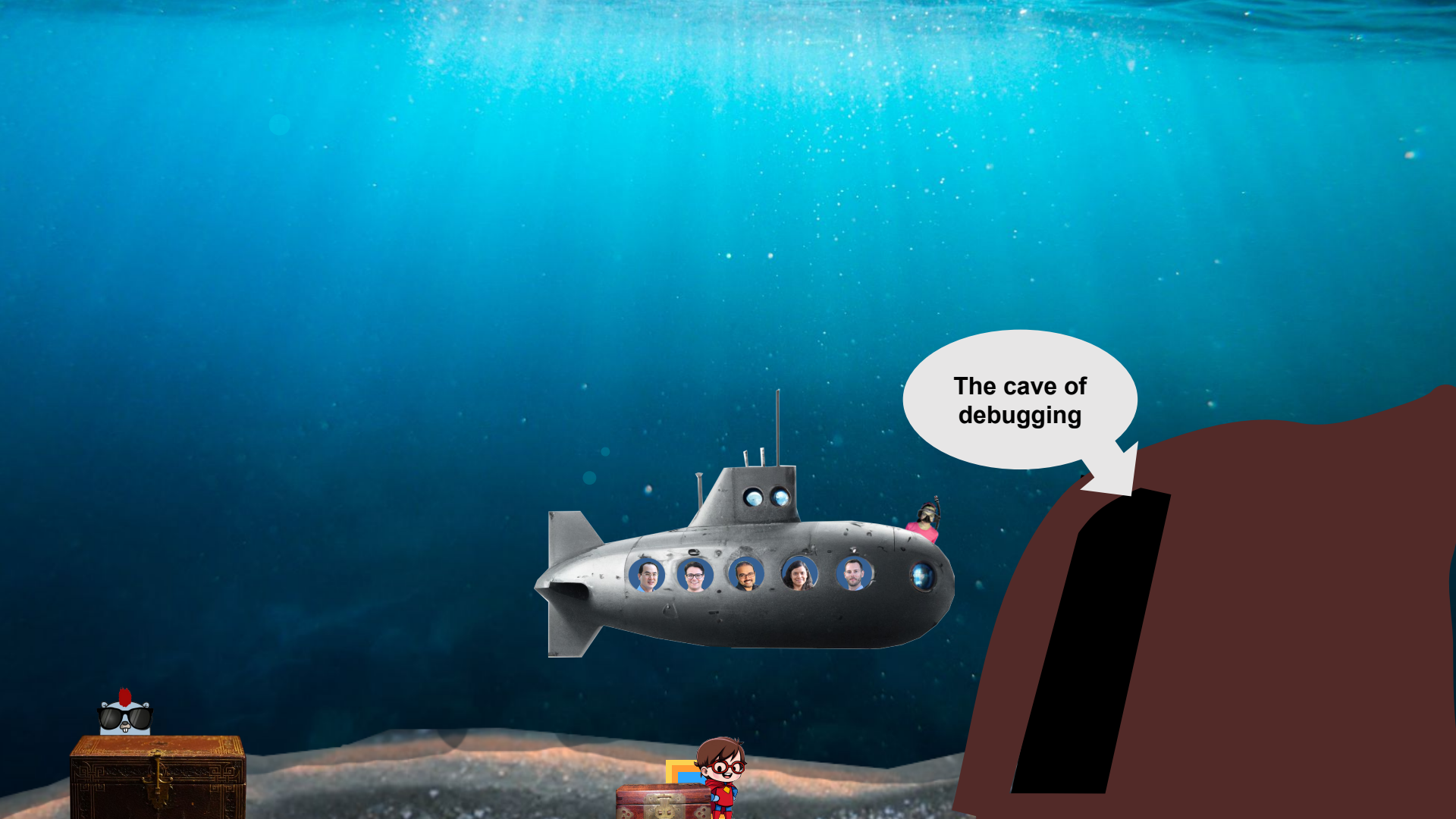
This policy says the job is successful when the launcher is.

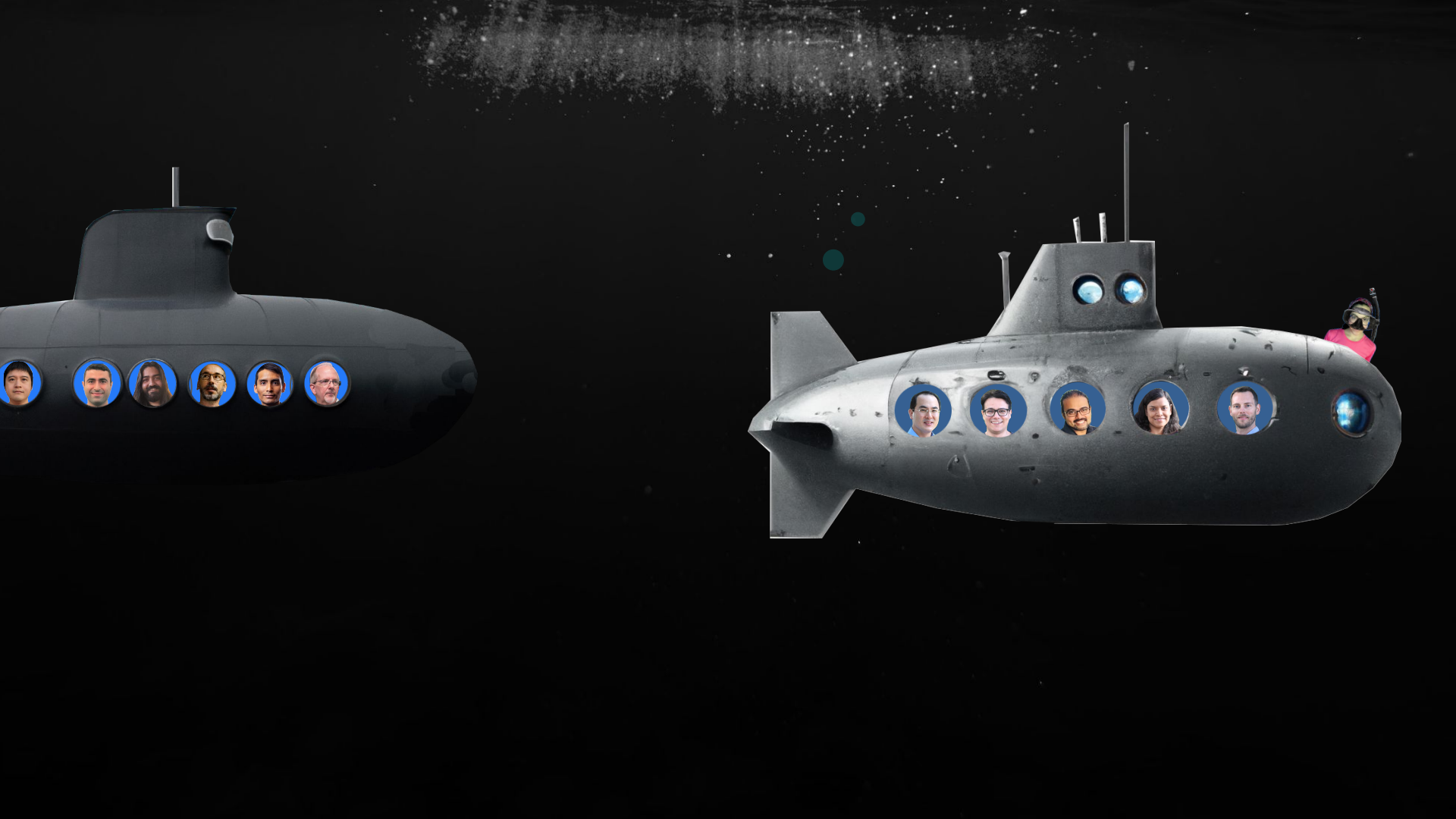


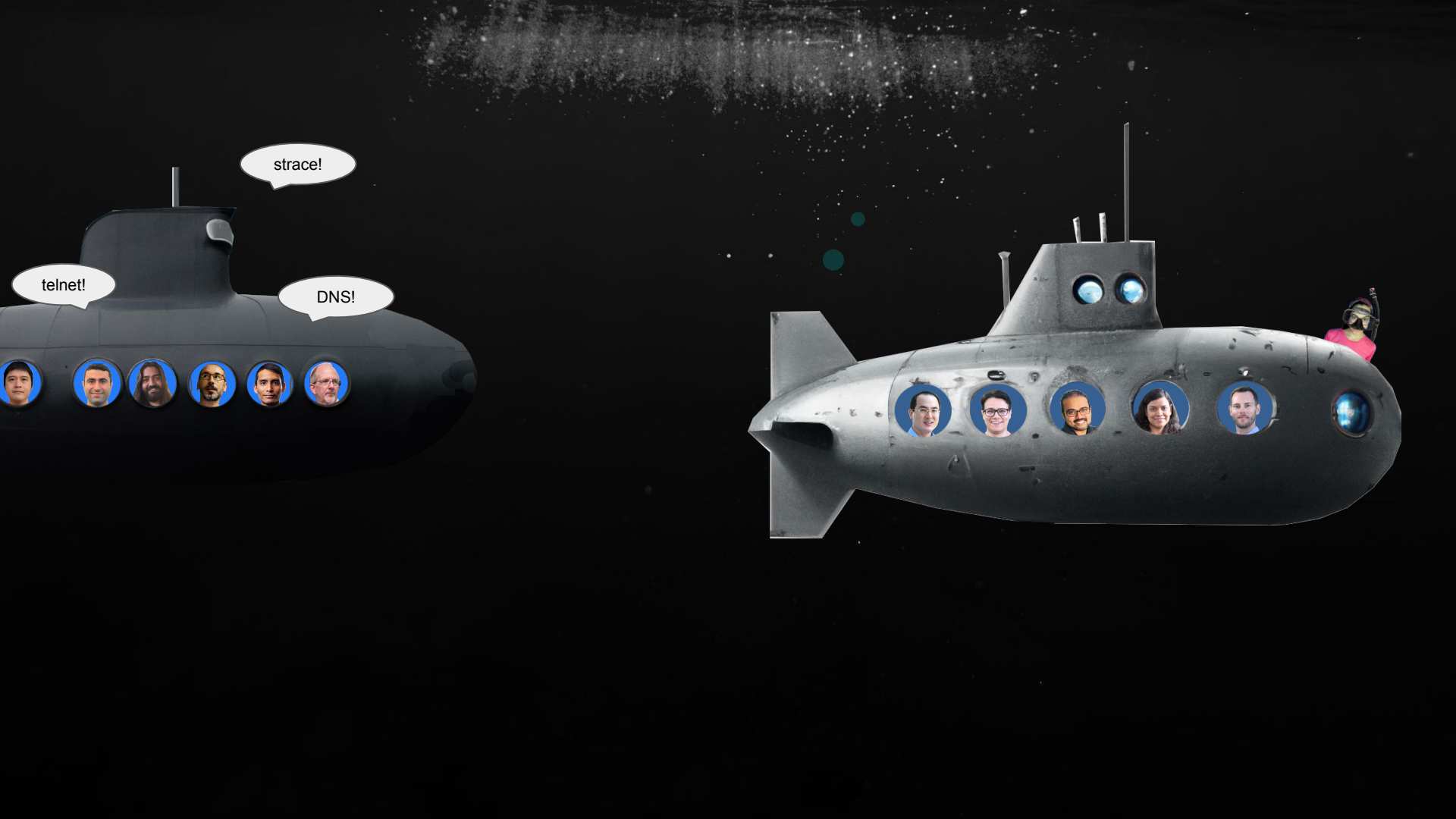

JobSet will allow us to build complex HPC applications!

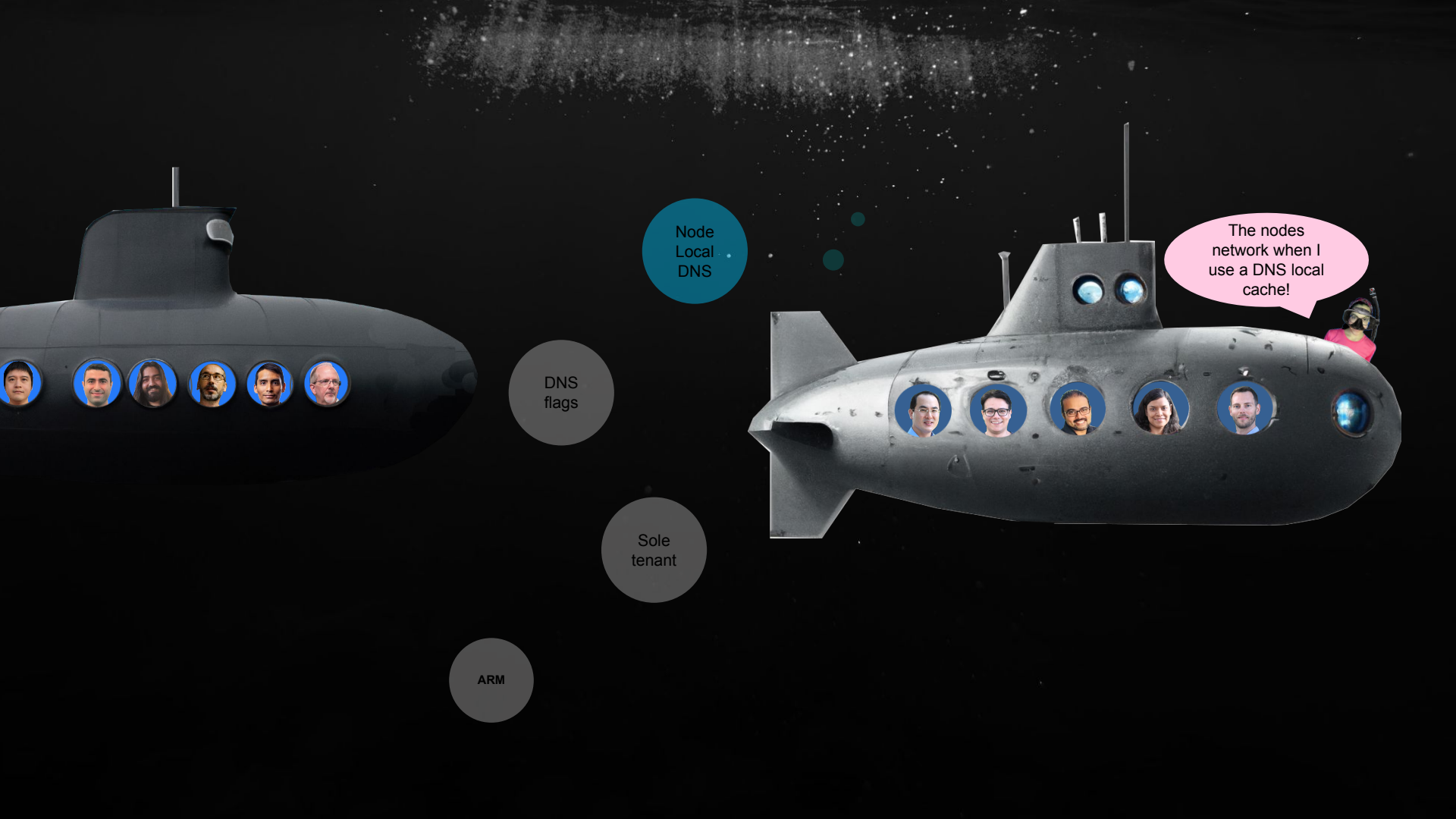
Our applications don't run.

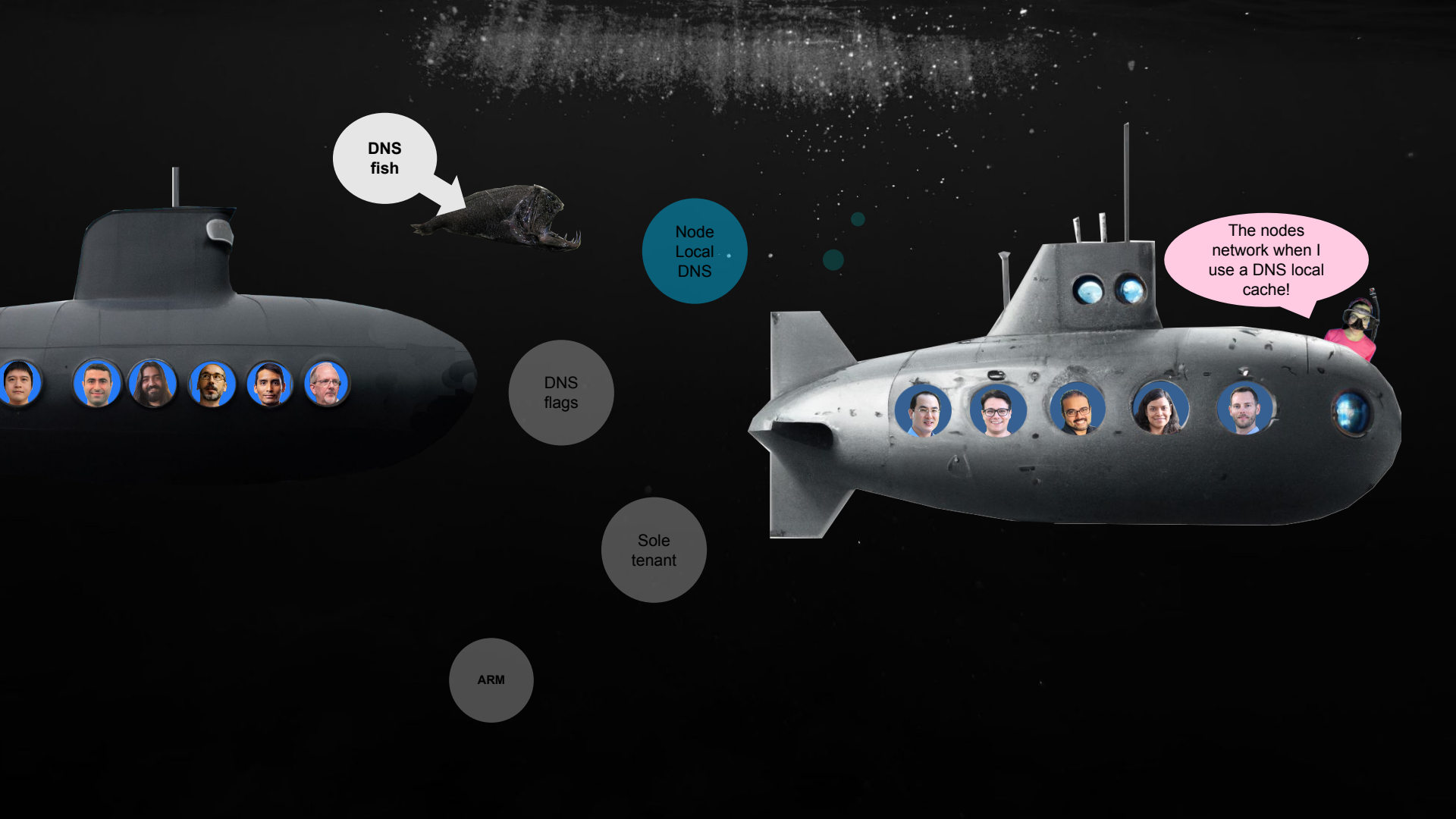


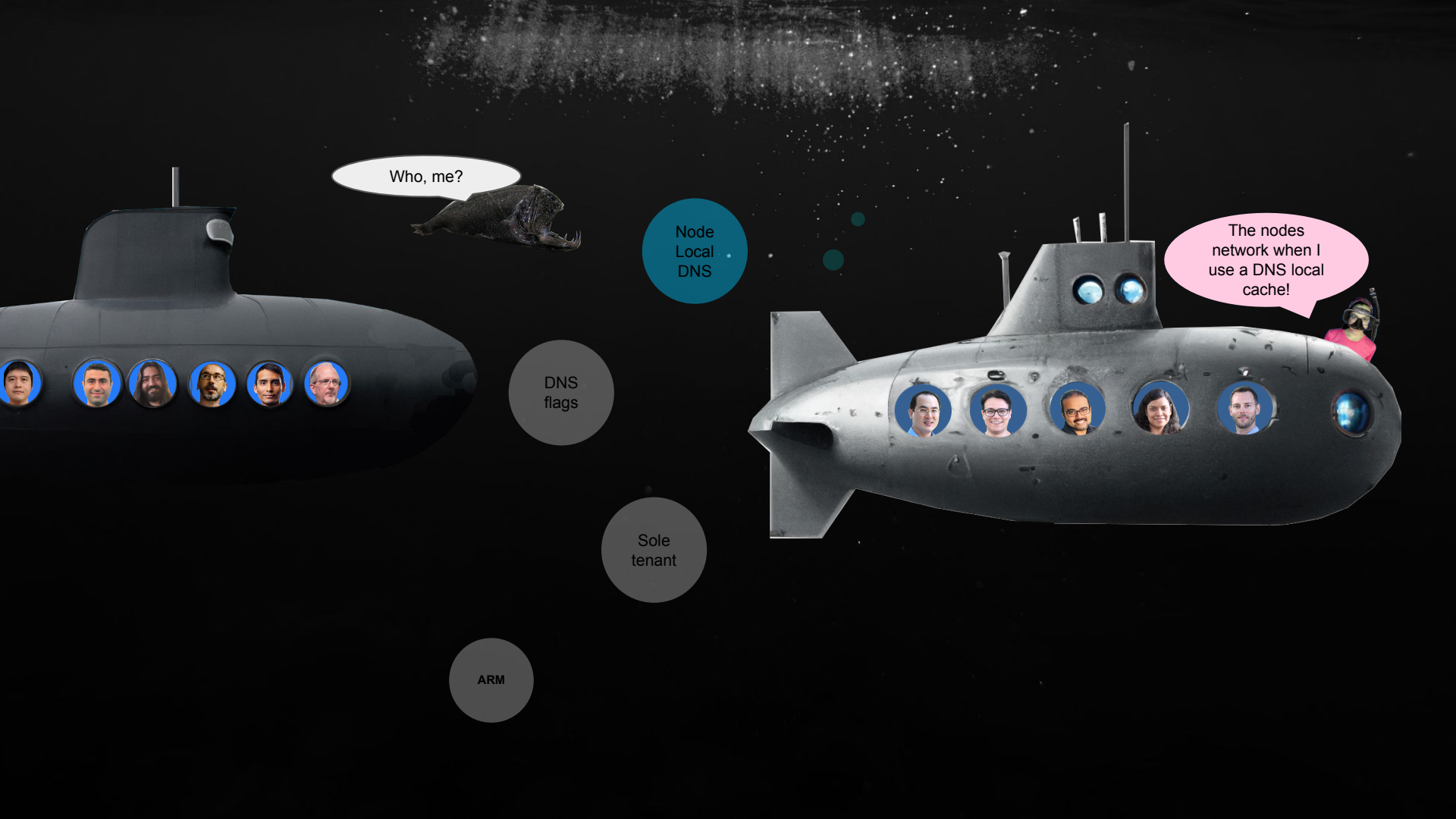

The cave of debugging




strace!
telnet!
DNS!

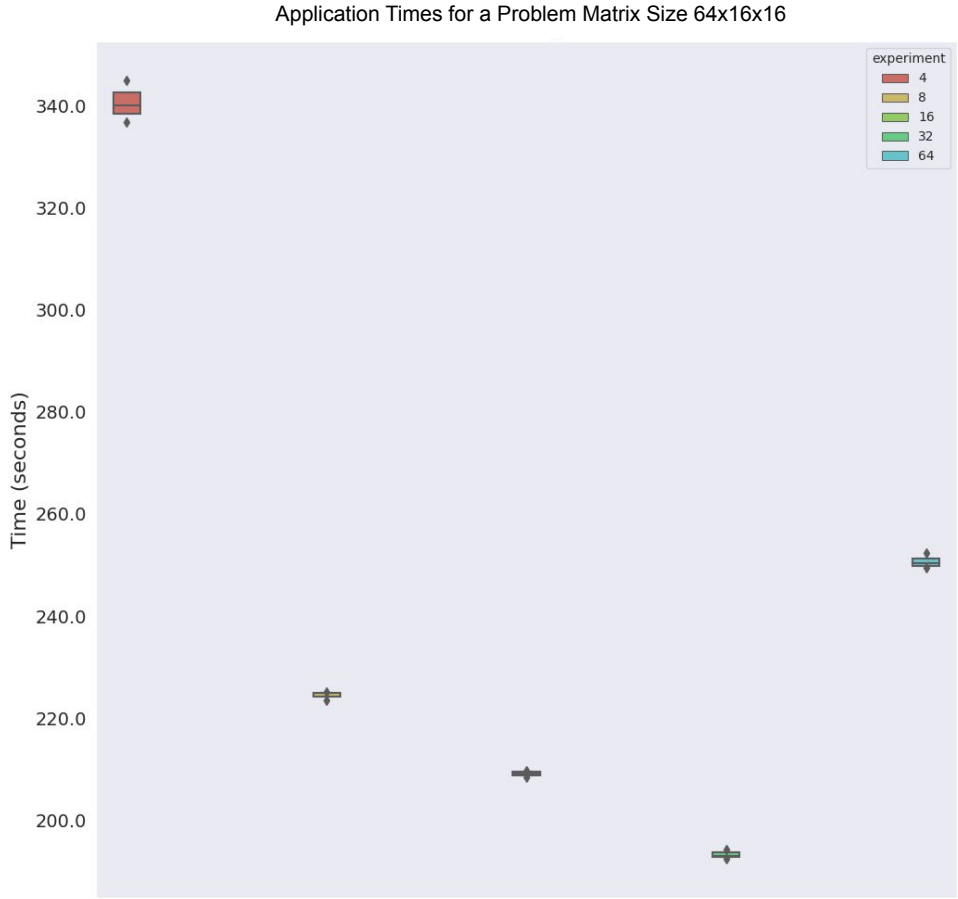

Node Local DNS
DNS flags
Sole tenant
ARM
The nodes network when I use a DNS local cache!


DNS fish
Node Local DNS
The nodes network when I use a DNS local cache!
DNS flags
Sole tenant
ARM

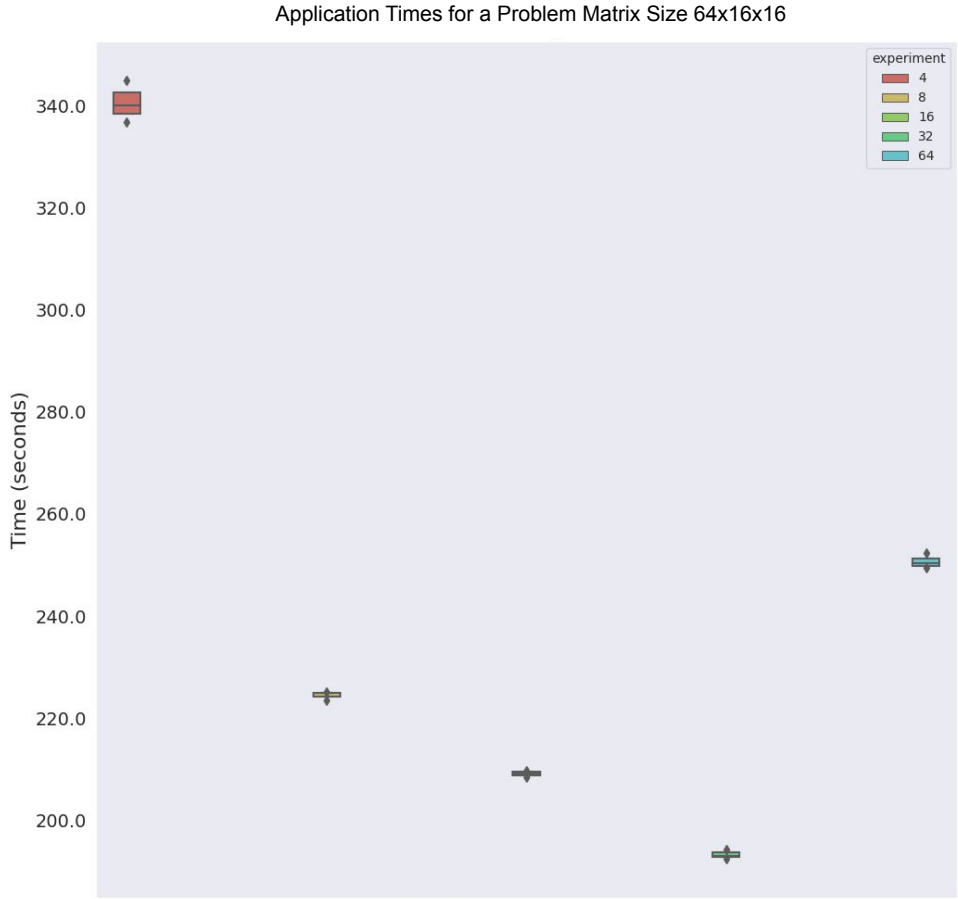

Who, me?
Node Local DNS
DNS flags
Sole tenant
ARM
The nodes network when I use a DNS local cache!


Let's run our application!

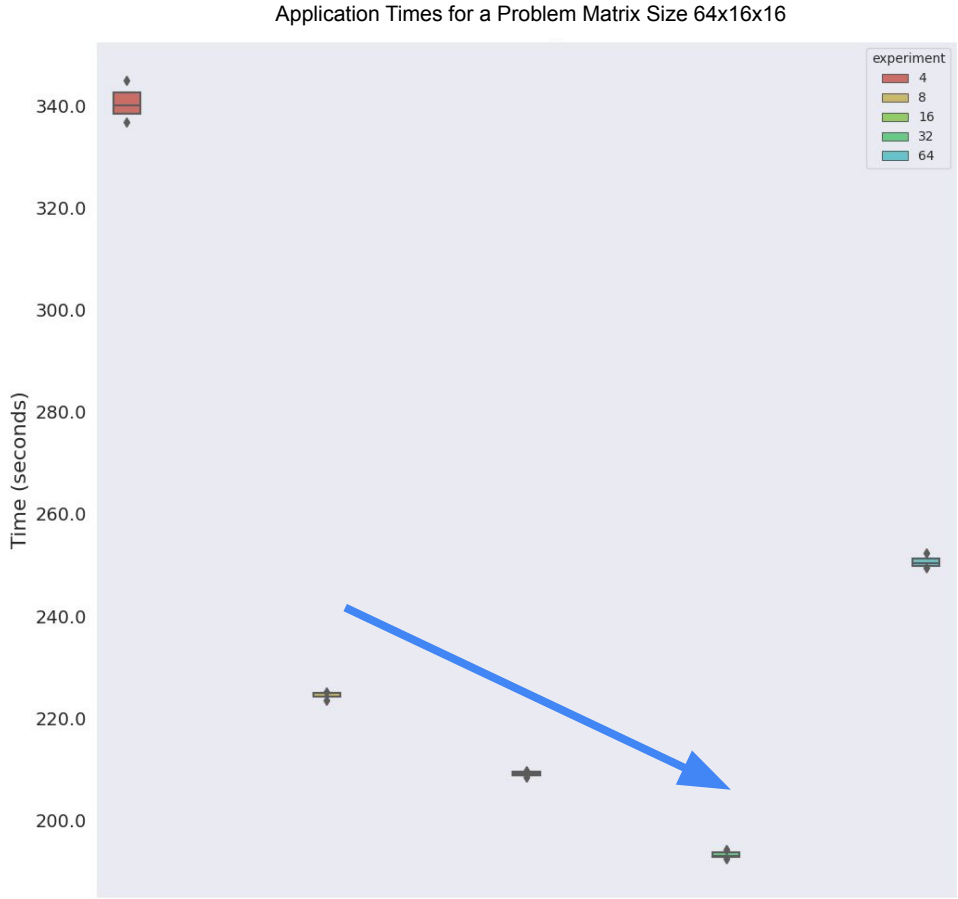
This is application time (Y) as a function of cluster size (nodes) or MPI ranks (X).



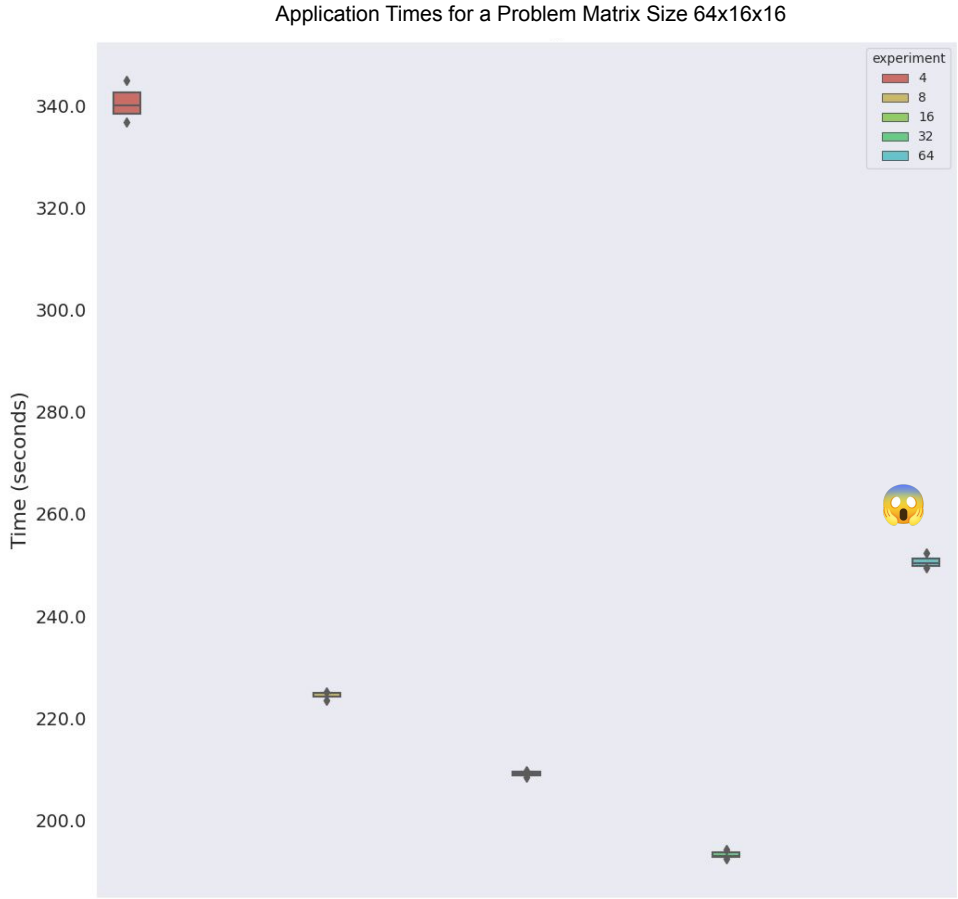
With **strong scaling** the runtime should get faster as we add resources!



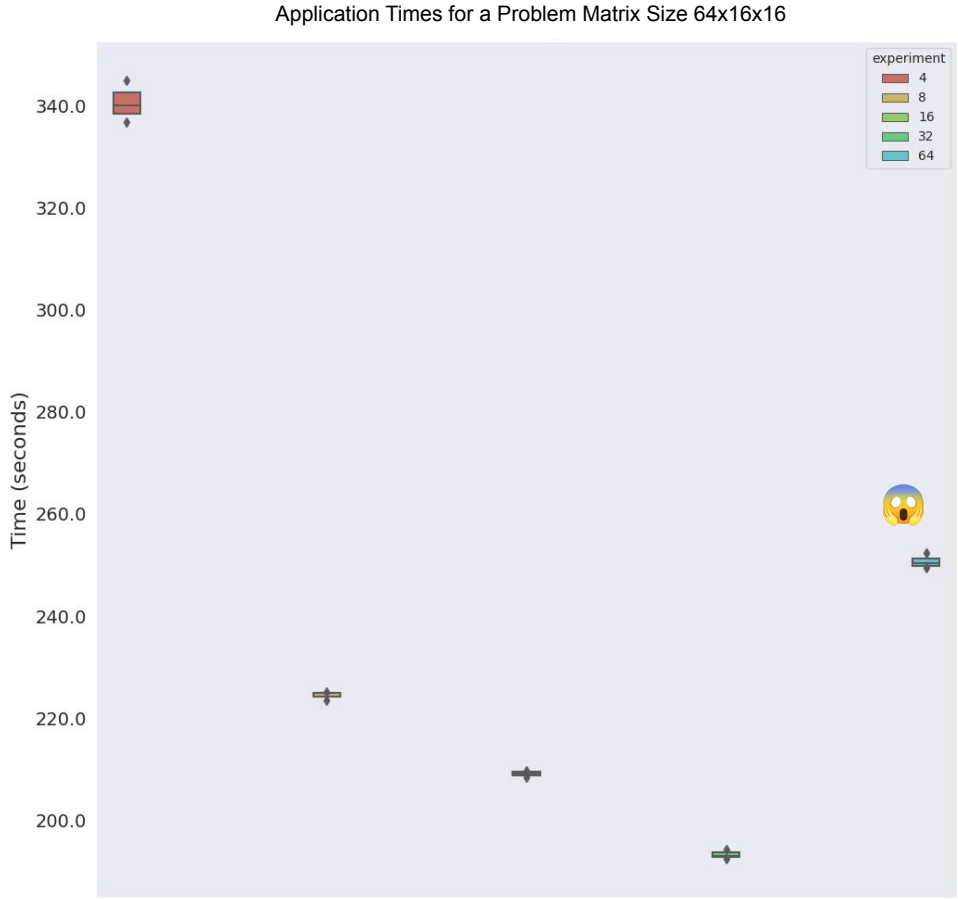
With **strong scaling** the runtime should get faster as we add resources!



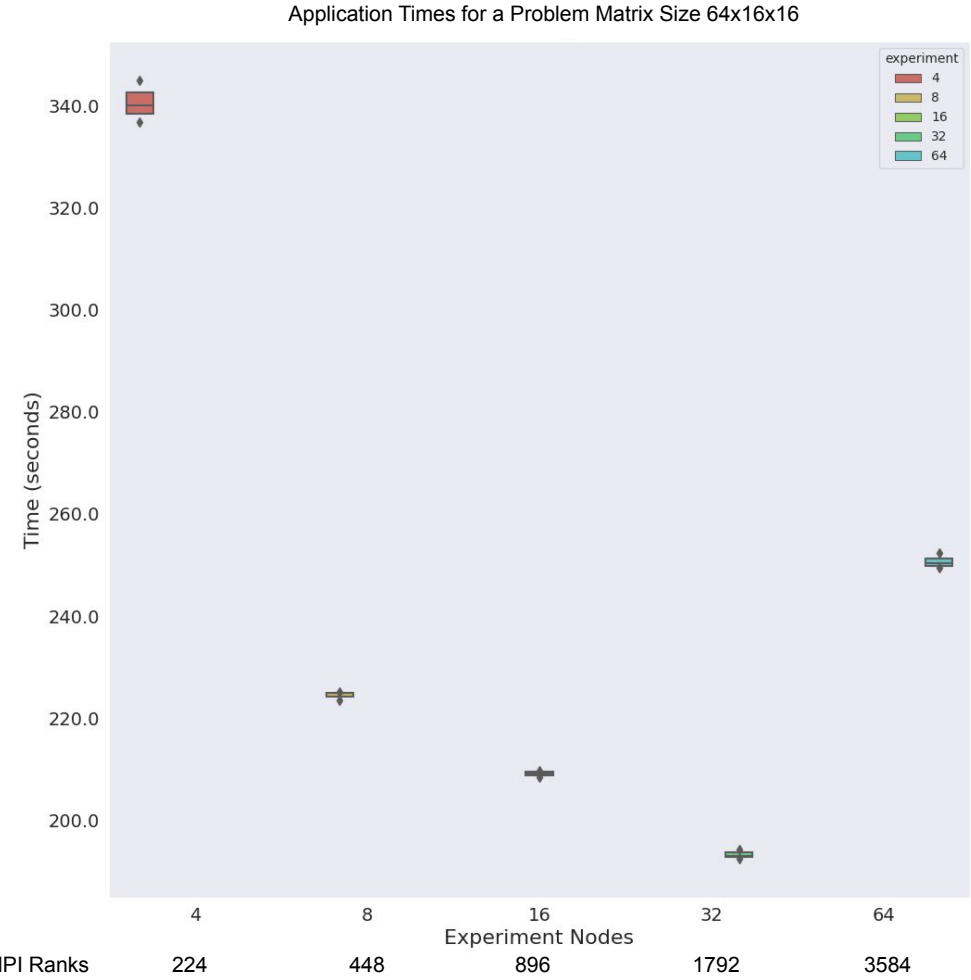
With **strong scaling** the runtime should get faster as we add resources!



Does the **cost of communication** offset the potential benefit to adding more workers?

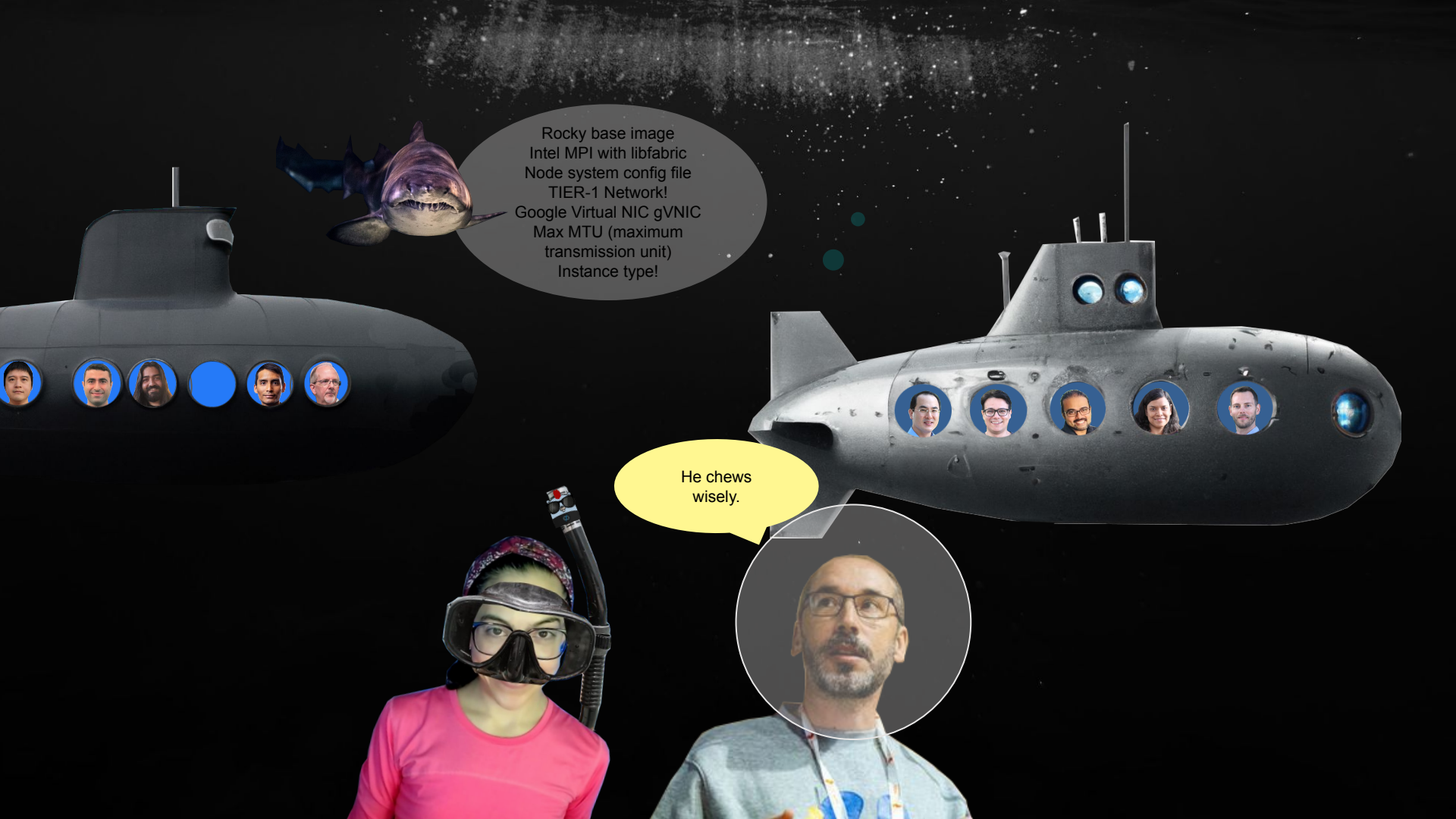


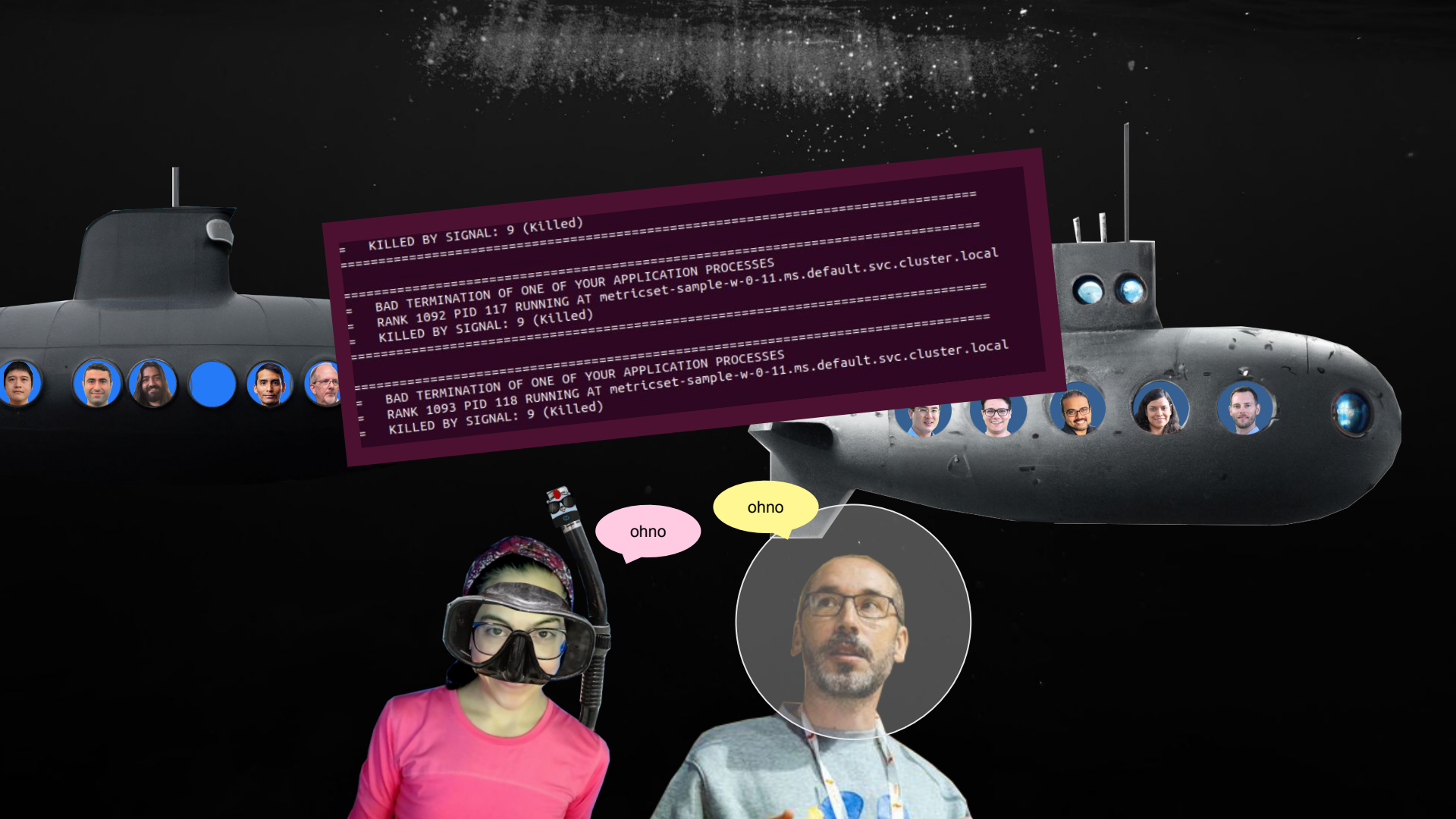
These times aren't very good. 😱



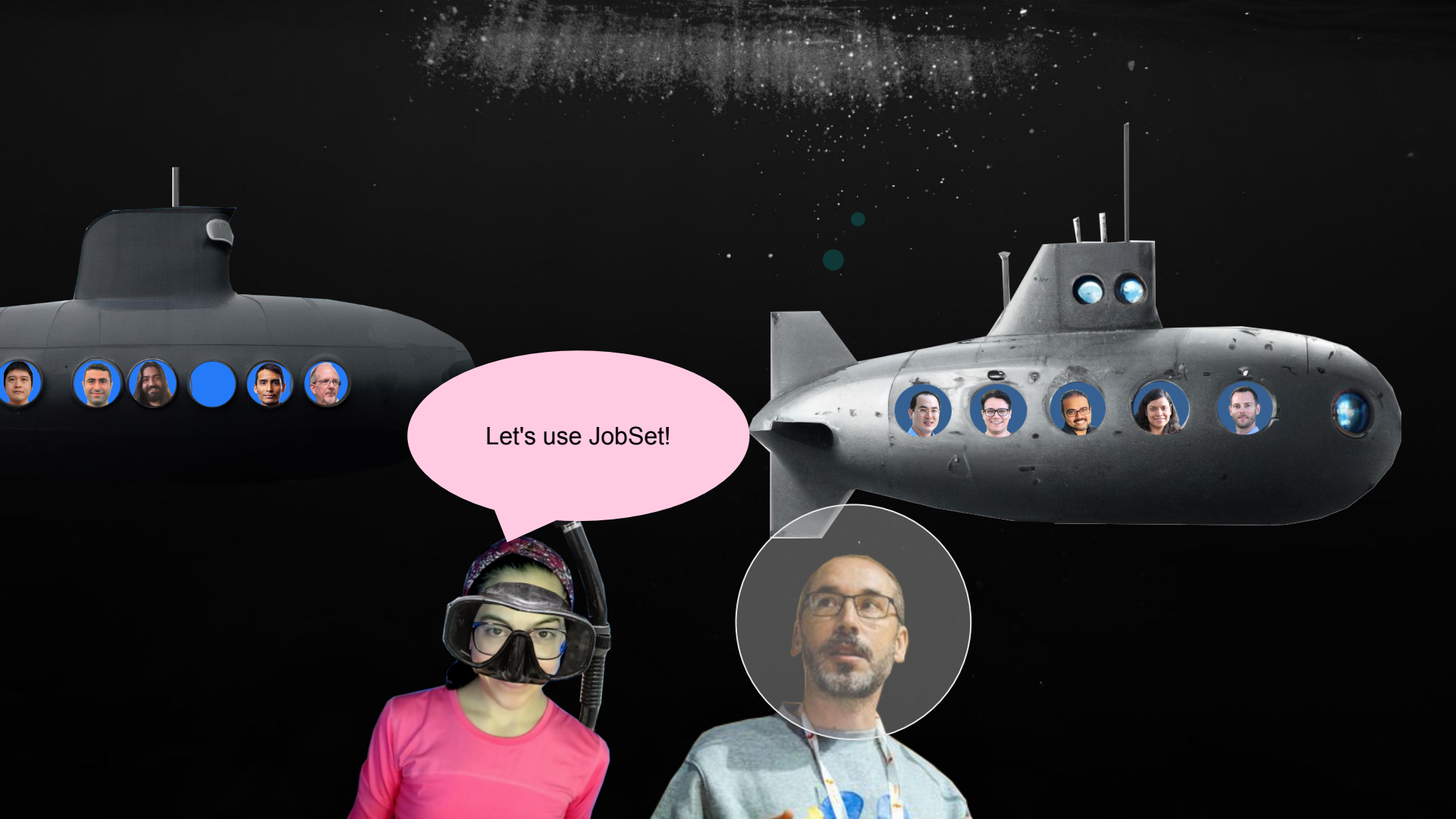

It's still slow!


I can help, V!


Rocky base image
Intel MPI with libfabric
Node system config file
TIER-1 Network!
Google Virtual NIC gVNIC
Max MTU (maximum transmission unit)
Instance type!
He chews wisely.


=   KILLED BY SIGNAL: 9 (Killed)
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   RANK 1092 PID 117 RUNNING AT metricset-sample-w-0-11.ms.default.svc.cluster.local
=   KILLED BY SIGNAL: 9 (Killed)
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   RANK 1093 PID 118 RUNNING AT metricset-sample-w-0-11.ms.default.svc.cluster.local
=   KILLED BY SIGNAL: 9 (Killed)
ohno
ohno

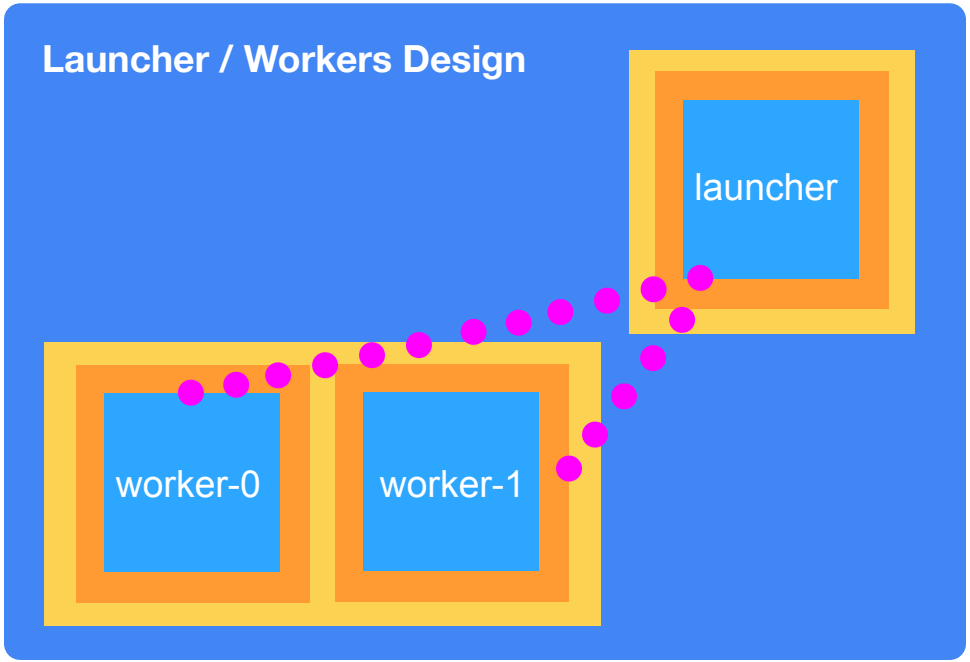

We need to
measure things!


Let's use JobSet!


OK but it's kind of creepy down here…

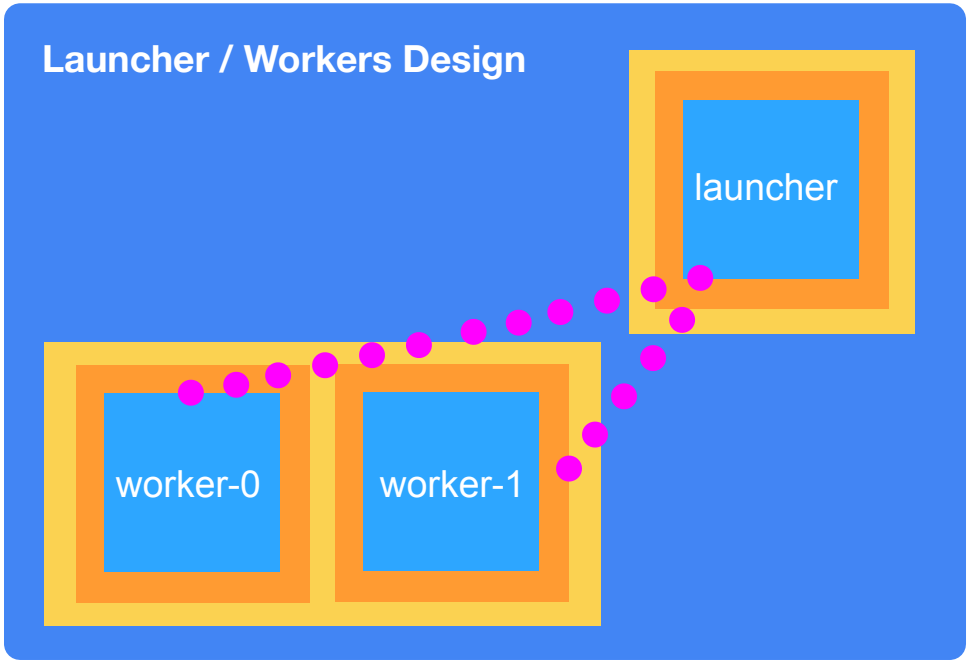
JobSet gives us flexibility to model applications and tools

1. JobSet as a Means to Quickly Measure and Run Applications



Many applications in HPC (and beyond!) have a launcher and worker design!

1. JobSet as a Means to Quickly Measure and Run Applications



HPC Proxy Apps

- LAMMPS
- AMG
- Nekbone
- Kripke
- Pennant
- Quicksilver
- …

Networking

- OSU Benchmarks
- Netmark
- Chatterbug
- …

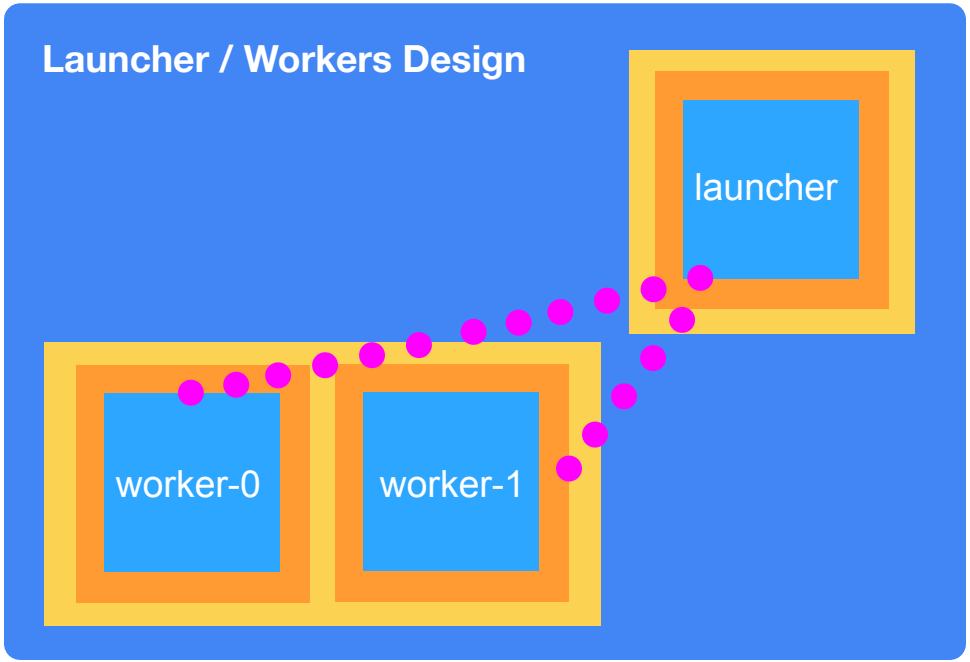
Storage (IO)

- IOR
- …

THE METRICS OPERATOR

You can count on me!

2. The Metrics Operator creates Metric Set



HPC Proxy Apps

- LAMMPS
- AMG
- Nekbone
- Kripke
- Pennant
- Quicksilver
- …

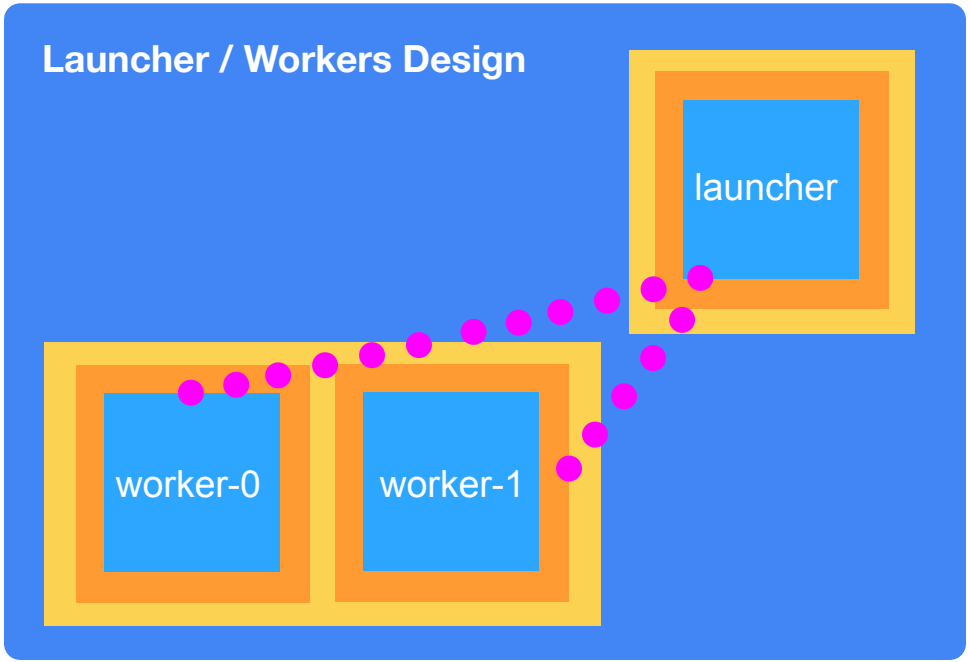
Networking

- OSU Benchmarks
- Netmark
- Chatterbug
- …

Storage (IO)

- IOR
- …

3. The OSU Benchmarks allow us to look at networking



HPC Proxy Apps

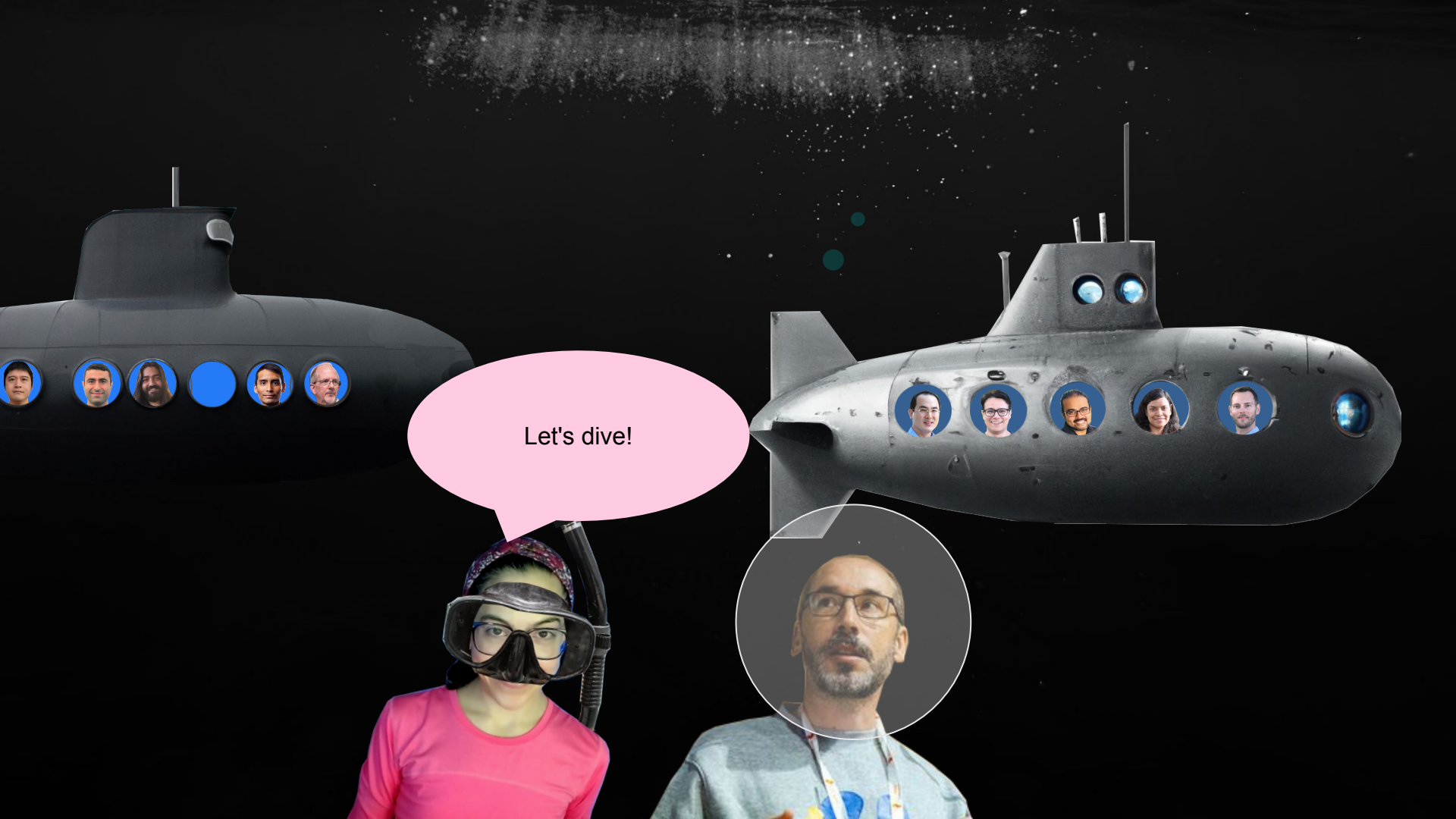
- LAMMPS
- AMG
- Nekbone
- Kripke
- Pennant
- Quicksilver
- …

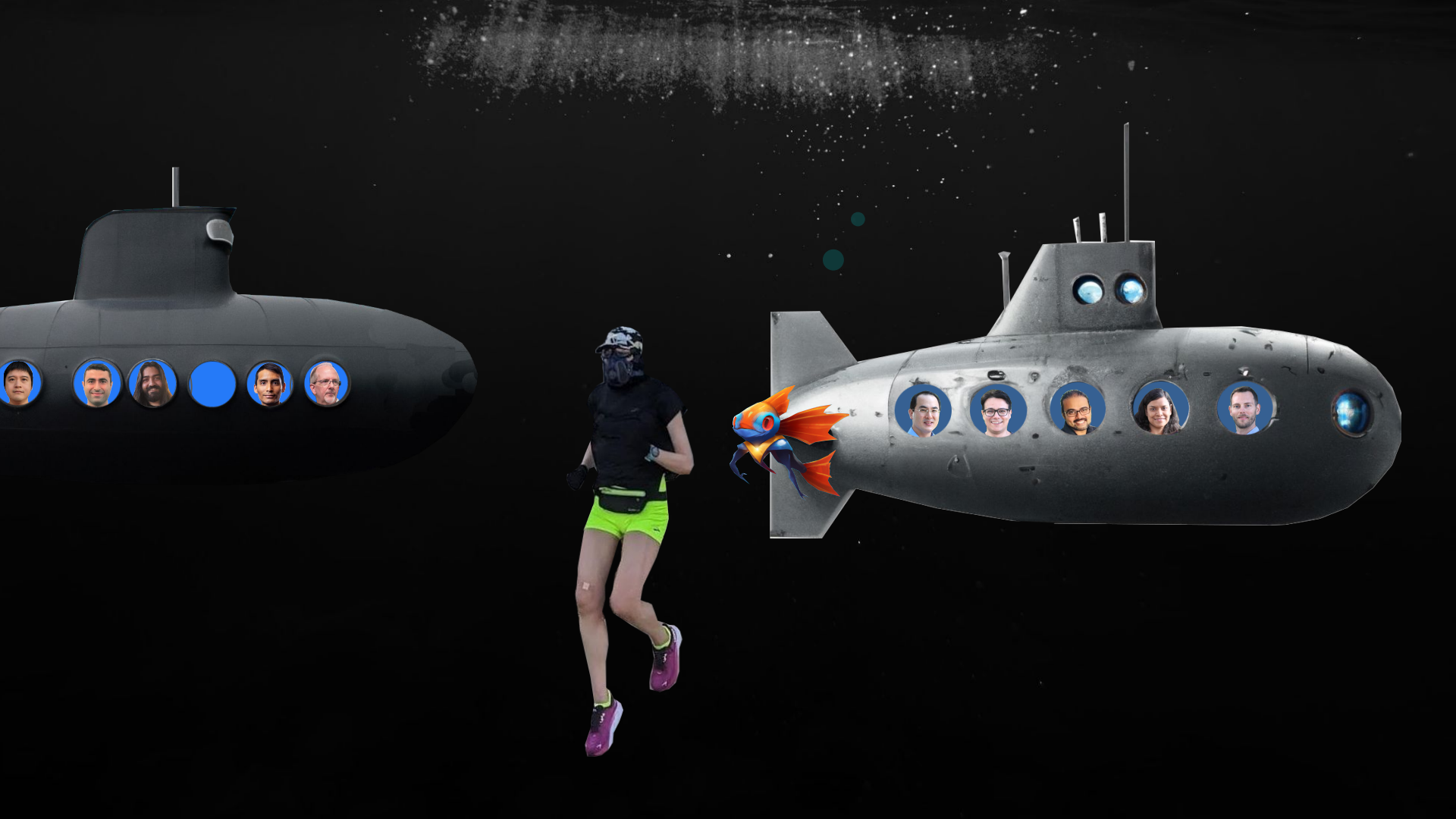
Networking

- OSU Benchmarks
- Netmark
- Chatterbug
- …

Storage (IO)

- IOR
- …


Let's dive!




The abyss of expectations


Be careful…
The abyss of expectations

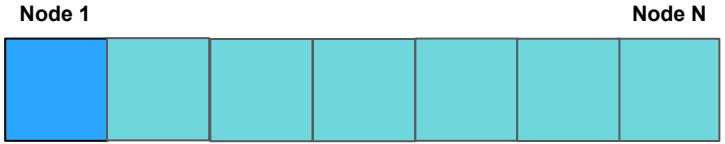
HPC Applications Need Low Latency

What does it mean to be tightly coupled?

What does it mean to be tightly coupled?

Tightly coupled apps use **MPI**, the Message Passing Interface

HPC Uses a **Tightly** Coupled Design

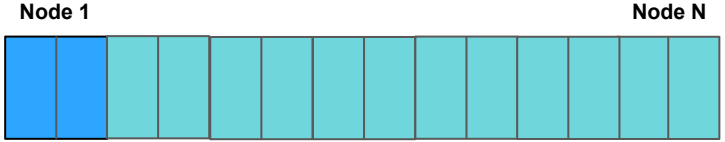


- Allows for complex scientific simulations

launcher

worker

HPC Uses a **Tightly** Coupled Design

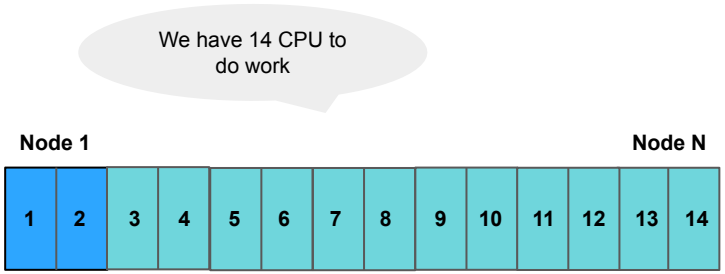


- Launcher assigns work to worker *processes*
- The relevant unit of work isn't a node, but a *process*

launcher

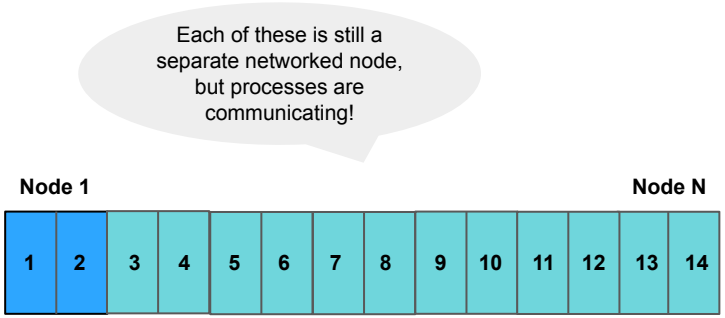
worker

HPC Uses a **Tightly** Coupled Design



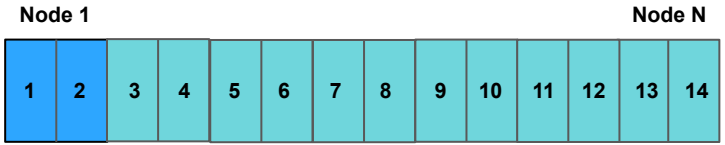
- Launcher assigns work to worker *processes*
- The relevant unit of work isn't a node, but a *process*
- Typically using MPI, the Message Passing Interface

HPC Uses a **Tightly** Coupled Design



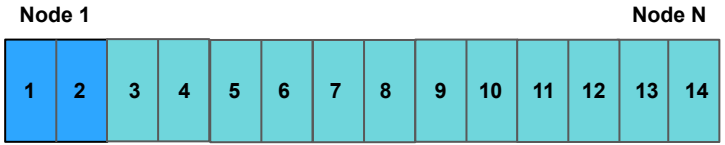
- Launcher assigns work to worker *processes*
- The relevant unit of work isn't a node, but a *process*
- Typically using MPI, the Message Passing Interface

HPC Uses a **Tightly** Coupled Design



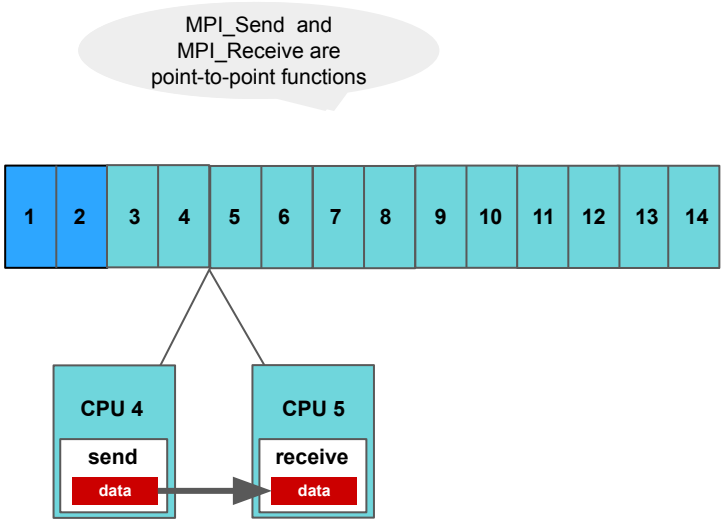
- Different patterns of communication between processes

HPC Uses a **Tightly** Coupled Design



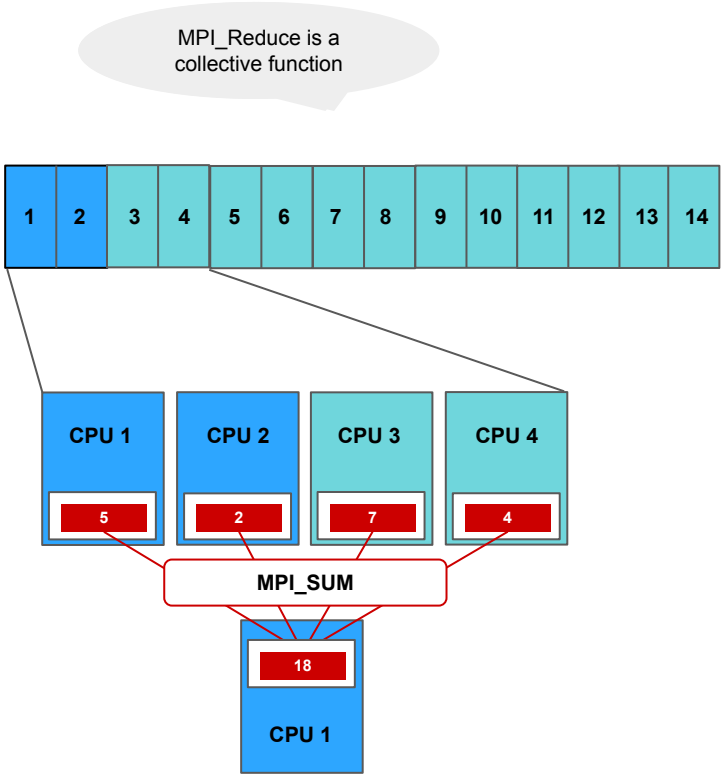
- Different patterns of communication between processes
  - Point-to-point
  - Collective

HPC Uses a **Tightly** Coupled Design



- **Point-to-point** communications are between pairs

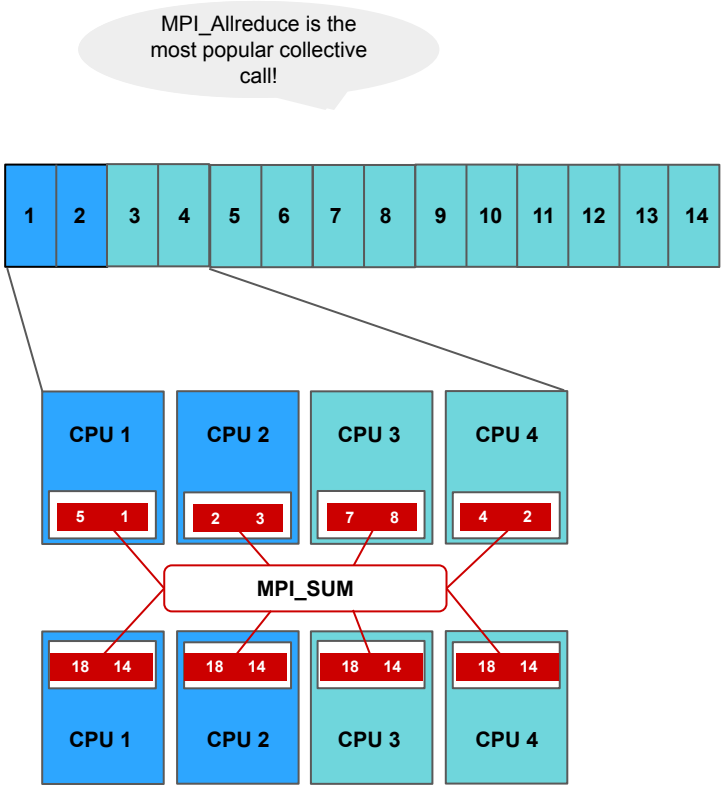
HPC Uses a **Tightly** Coupled Design



- **Point-to-point** communications are between pairs
- **Collective** communications are between collections (groups) of processes

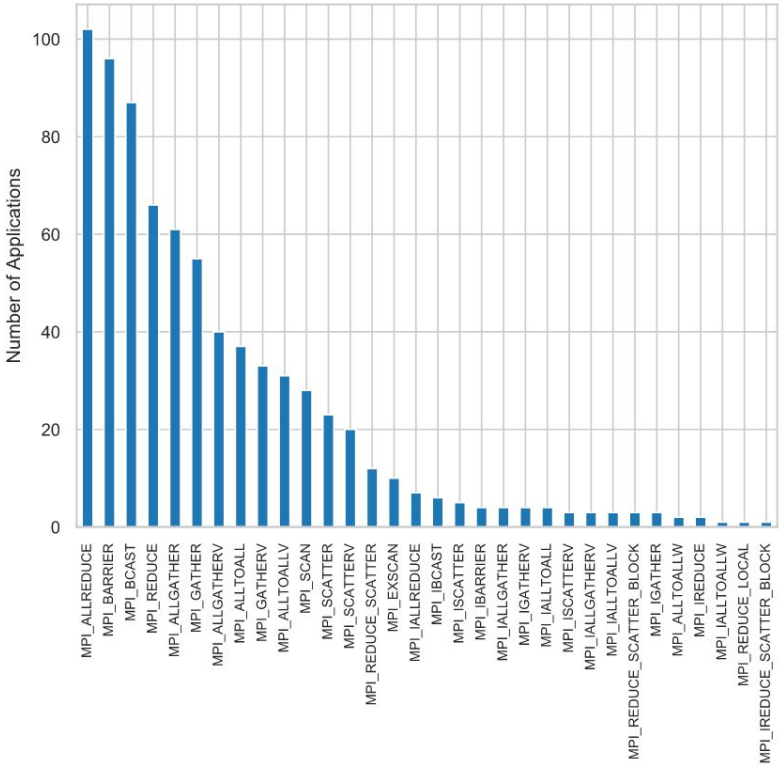
From Wes Kendall's Tutorial

HPC Uses a **Tightly** Coupled Design

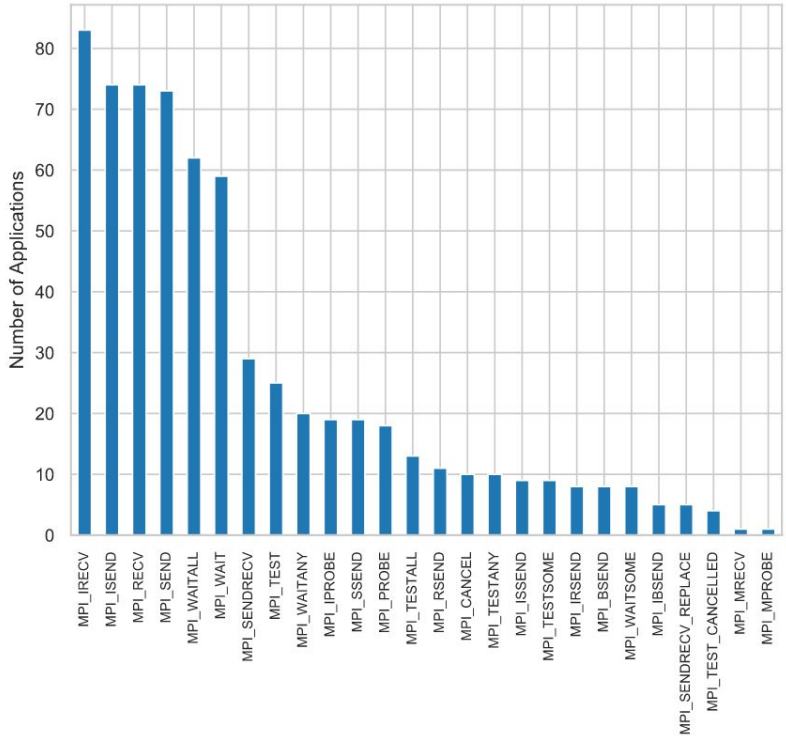


- **Point-to-point** communications are between pairs
- **Collective** communications are between collections (groups) of processes
- **MPI_Allreduce** adds MPI_Bcast to broadcast data

Usage of MPI Features Across HPC Applications



Collective Functions

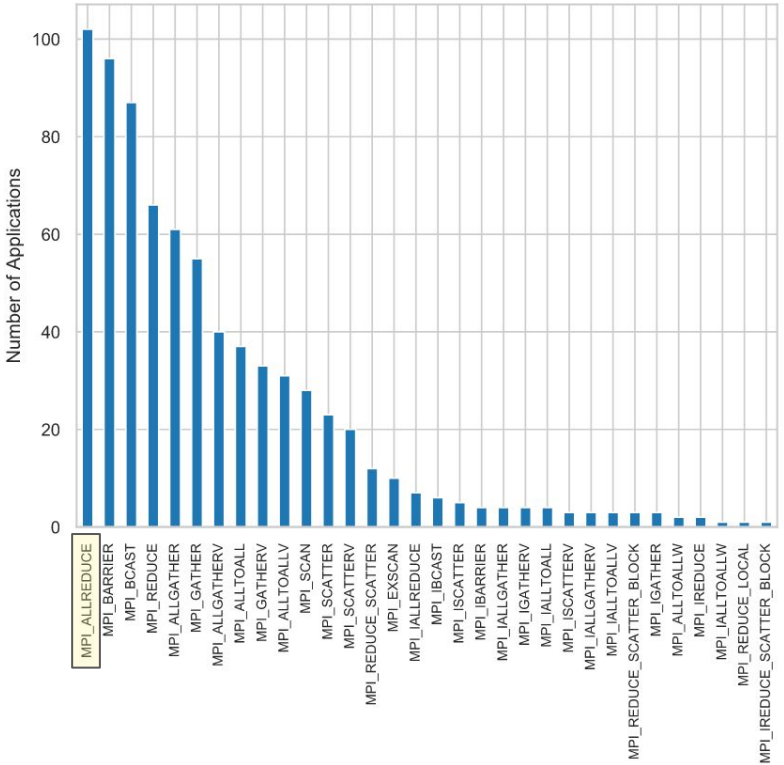


Point-to-point Functions

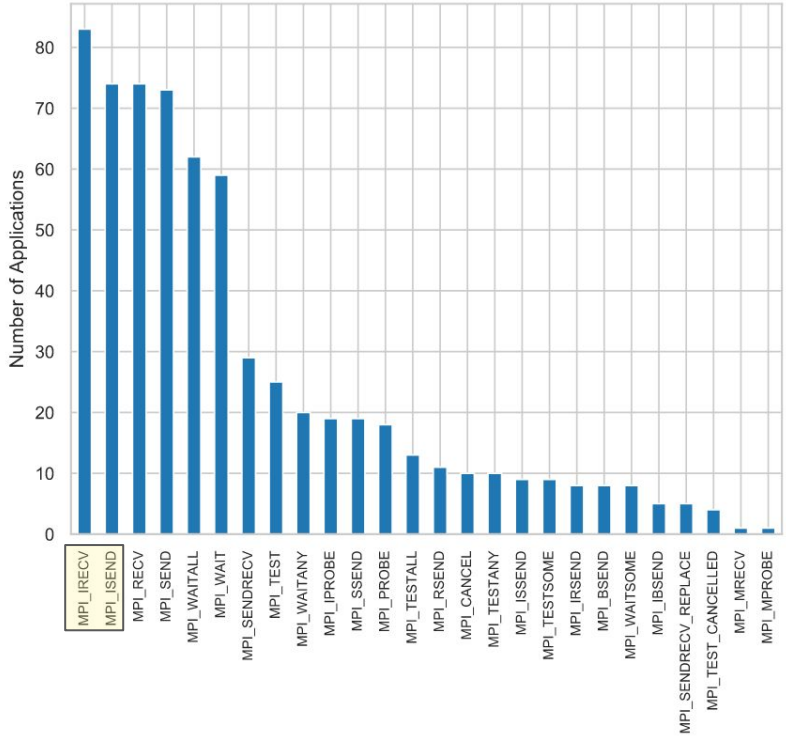
the "i" prefix is the equivalent pattern, but non-blocking

A large-scale study of MPI usage in open-source HPC applications

Usage of MPI Features Across HPC Applications



Collective Functions

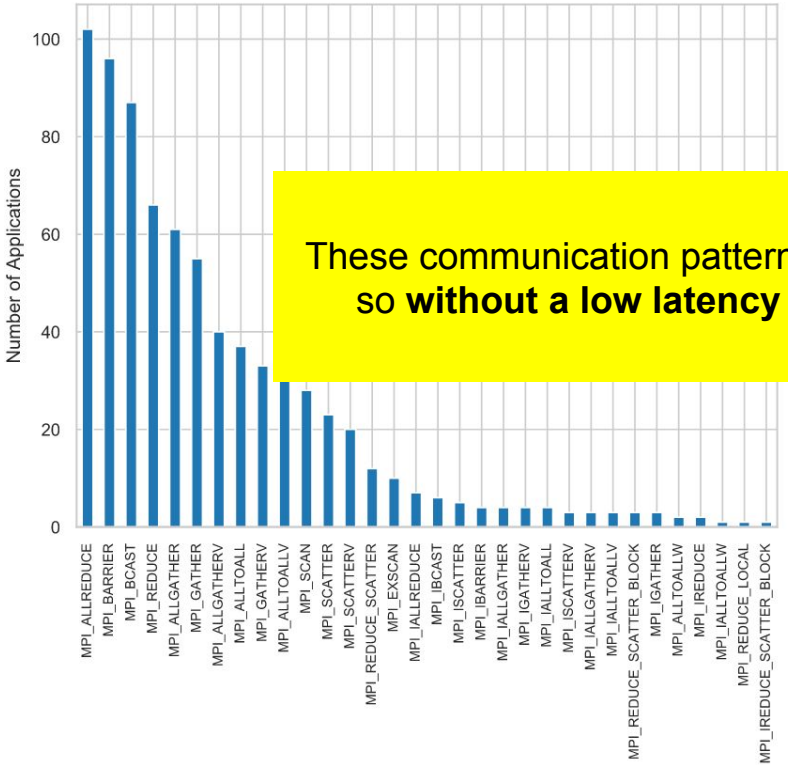


Point-to-point Functions

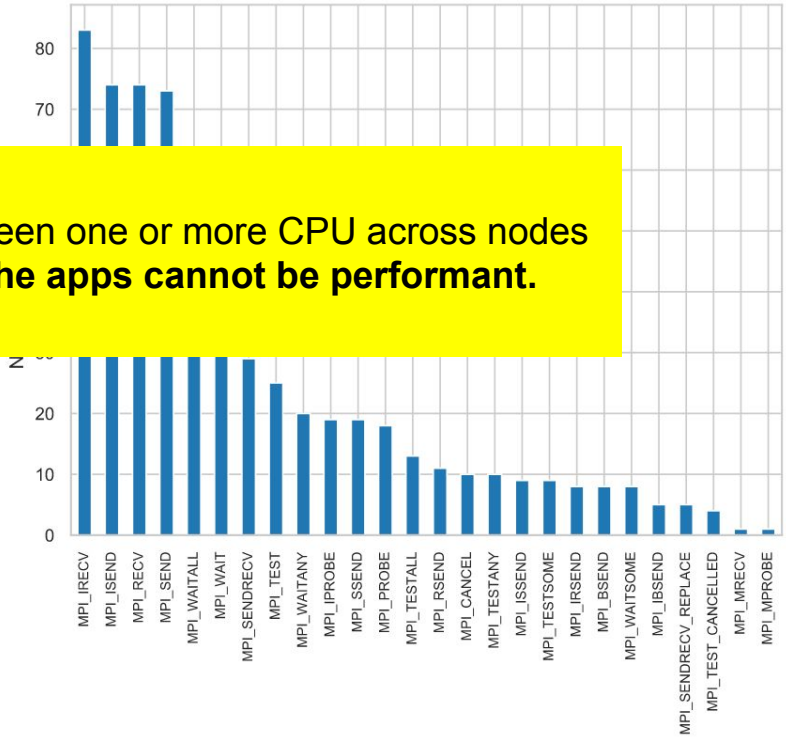
the "i" prefix is the equivalent pattern, but non-blocking

A large-scale study of MPI usage in open-source HPC applications

Usage of MPI Features Across HPC Applications



Collective Functions



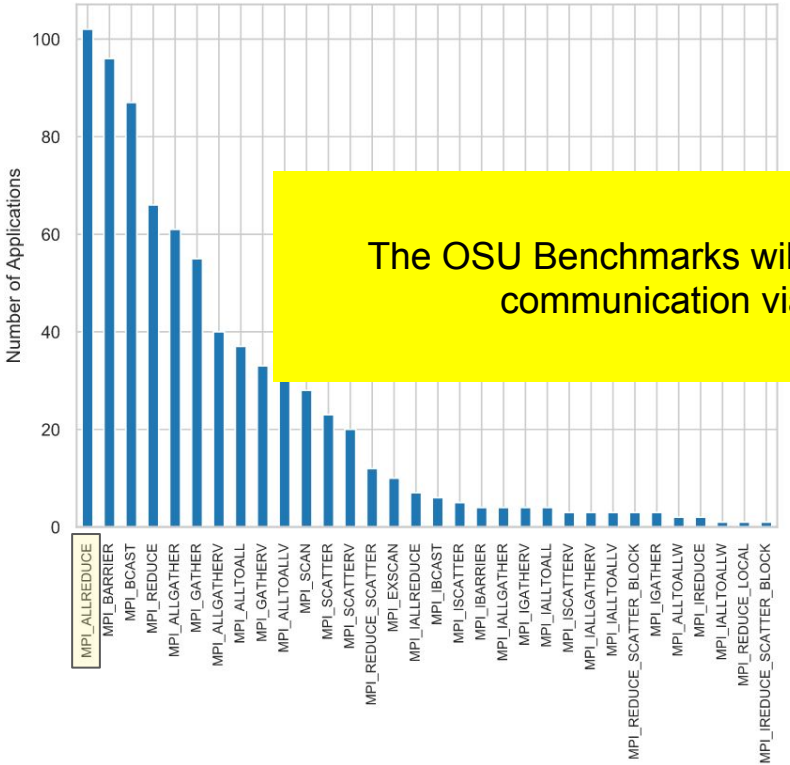
Point-to-point Functions

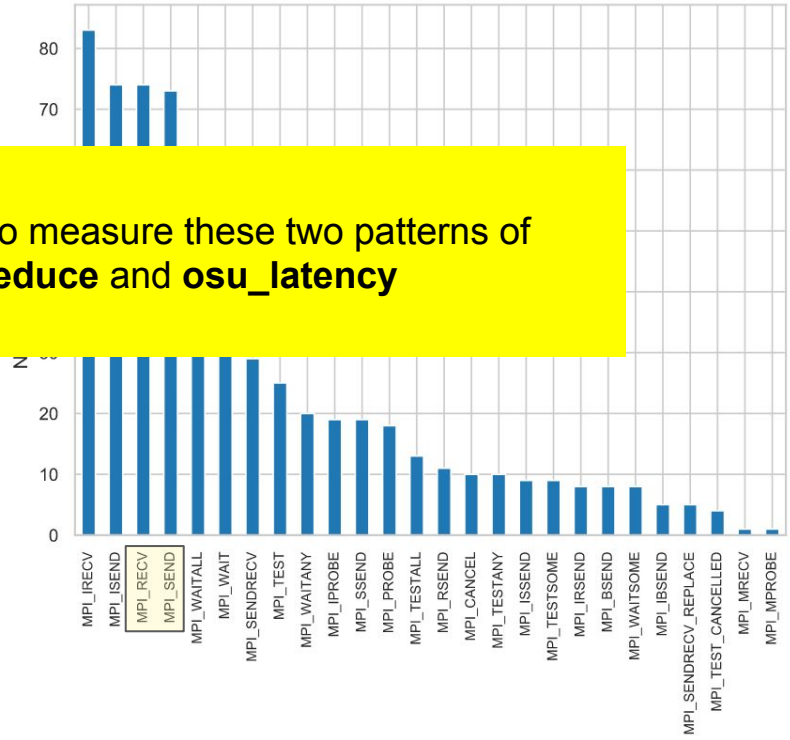
These communication patterns are between one or more CPU across nodes so **without a low latency network, the apps cannot be performant.**

the "i" prefix is the equivalent pattern, but non-blocking

A large-scale study of MPI usage in open-source HPC applications

Usage of MPI Features Across HPC Applications





The OSU Benchmarks will allow us to measure these two patterns of communication via **osu_allreduce** and **osu_latency**

Collective Functions

Point-to-point Functions

the "i" prefix is the equivalent pattern, but non-blocking

A large-scale study of MPI usage in open-source HPC applications

We used the Metrics Operator to assess network latency

```yaml
apiVersion: flux-framework.org/v1alpha1
kind: MetricSet
metadata:
  labels:
    app.kubernetes.io/name: metricset
    app.kubernetes.io/instance: metricset-sample
  name: metricset-sample
spec:
  pods: 64
  metrics:
   - name: network-osu-benchmark
     # Custom list of commands to run
     # See https://converged-computing.github.io/metrics-operator/getting_started/metrics.html#network-osu-benchmark
     listOptions:
       commands:
        - osu_get_acc_latency
        - osu_acc_latency
        - osu_fop_latency
        - osu_get_latency
        - osu_put_latency
        - osu_latency
        - osu_bibw
        - osu_bw
        - osu_put_bw
        - osu_latency_mp
        - osu_put_bibw
        - osu_init
        - osu_get_bw
        - osu_cas_latency
        - osu_latency_mt
        - osu_hello
        - osu_barrier
     options:
       # Wrap each one in time
       timed: "true"
```

Make it stupid-easy to run these HPC metrics and applications!

We used the Metrics Operator to assess network latency

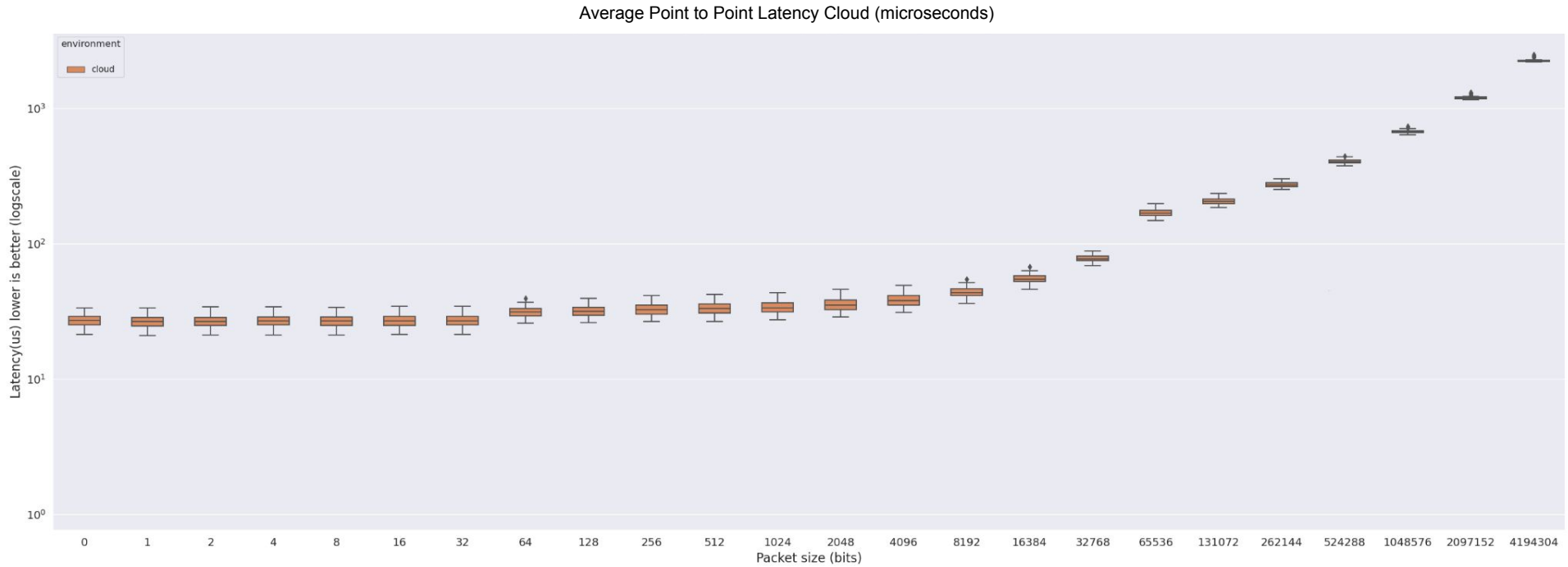
```yaml
apiVersion: flux-framework.org/v1alpha1
kind: MetricSet
metadata:
  labels:
    app.kubernetes.io/name: metricset
    app.kubernetes.io/instance: metricset-sample
  name: metricset-sample
spec:
  pods: 64
  metrics:
   - name: network-osu-benchmark
     # Custom list of commands to run
     # See https://converged-computing.github.io/metrics-operator/getting_started/metrics.html#network-osu-benchmark
     listOptions:
       commands:
        - osu_get_acc_latency
        - osu_acc_latency
        - osu_fop_latency
        - osu_get_latency
        - osu_put_latency
        - osu_latency
        - osu_bibw
        - osu_bw
        - osu_put_bw
        - osu_latency_mp
        - osu_put_bibw
        - osu_init
        - osu_get_bw
        - osu_cas_latency
        - osu_latency_mt
        - osu_hello
        - osu_barrier
     options:
       # Wrap each one in time
       timed: "true"
```

Powered by JobSet!

Make it stupid-easy to run these HPC metrics and applications!

Point to Point Benchmark "osu_latency" using the Metrics Operator

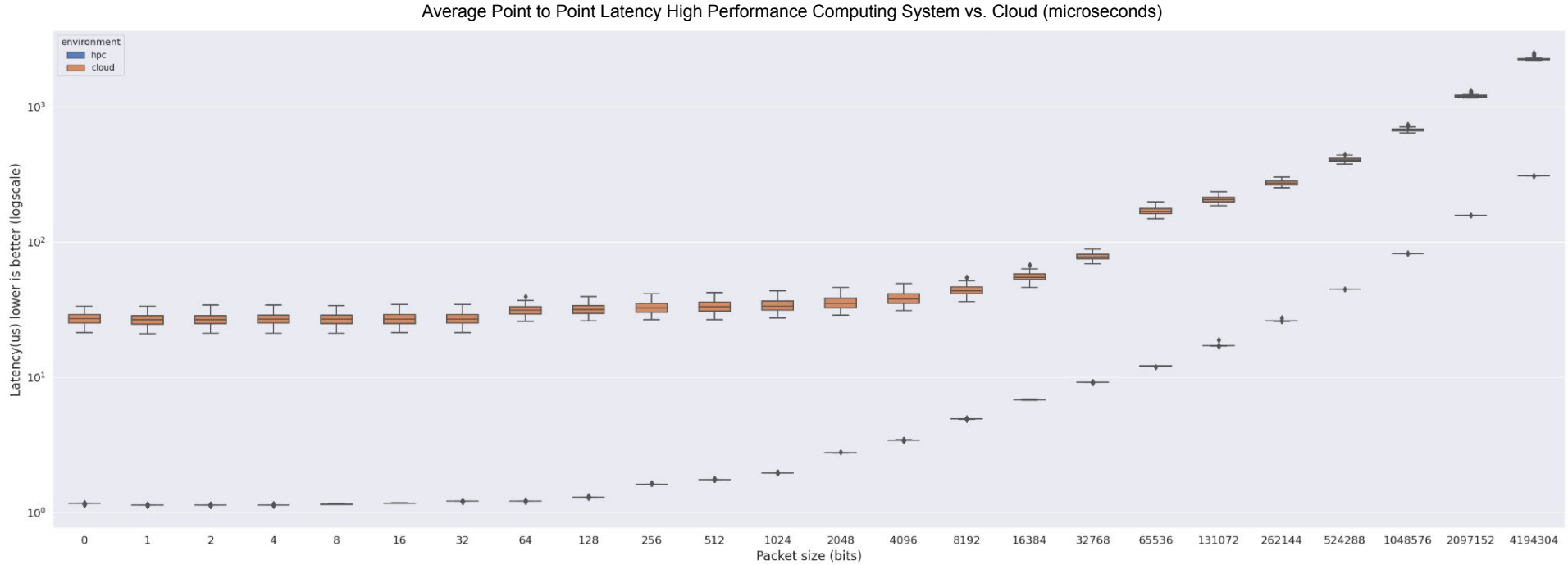
Cloud instance optimized with TIER-1, max MTU optimized libfabric with tcp



Point to Point Benchmark "osu_latency" using the Metrics Operator

Cloud instance optimized with TIER-1, max MTU optimized libfabric with tcp

Heavily used HPC cluster provisioned in 2018 with infiniband



Welcome to the
Abyss of Expectations


Laaaaaatency!
Welcome to the
Abyss of Expectations

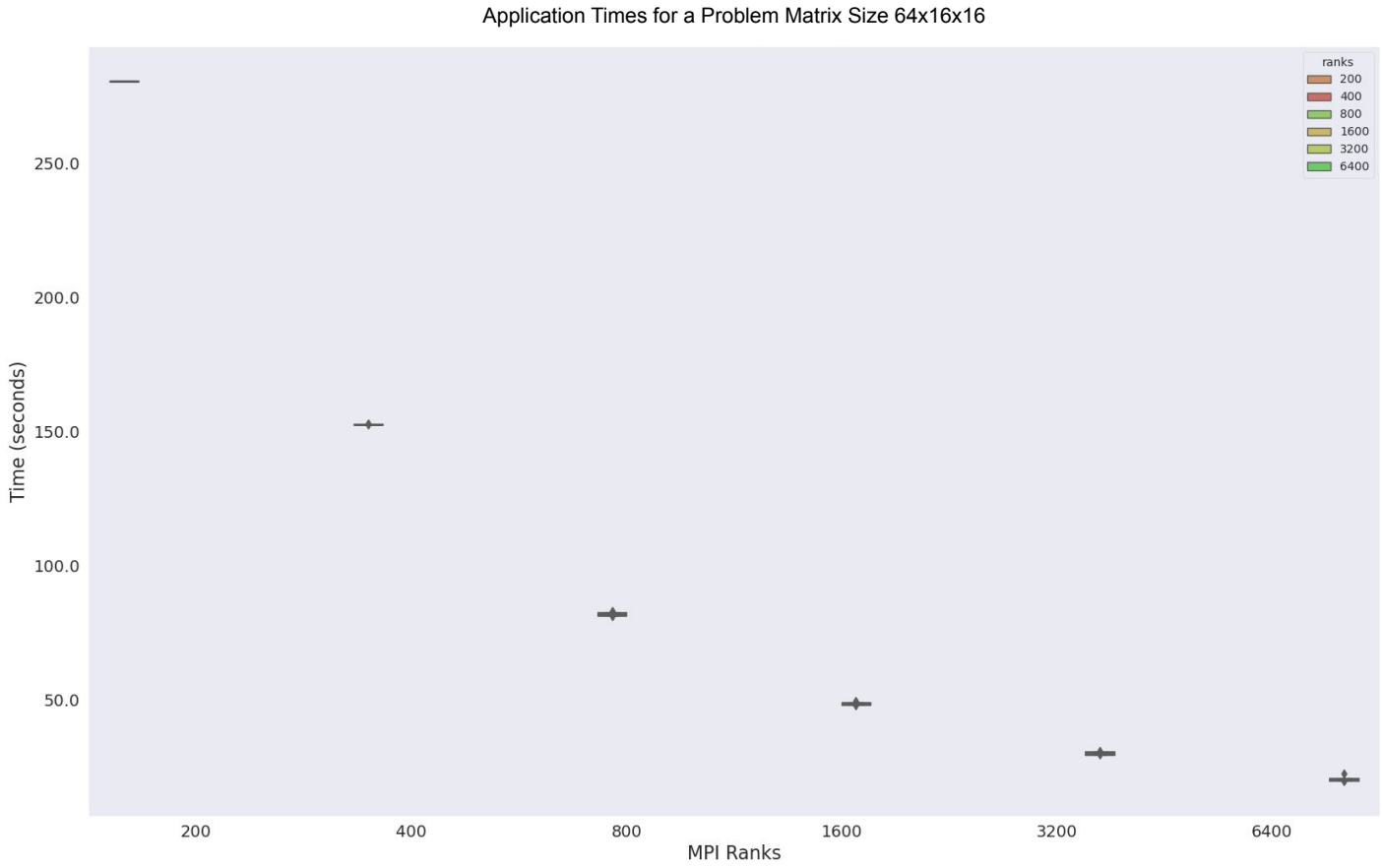
I *know* my application runs better on HPC…

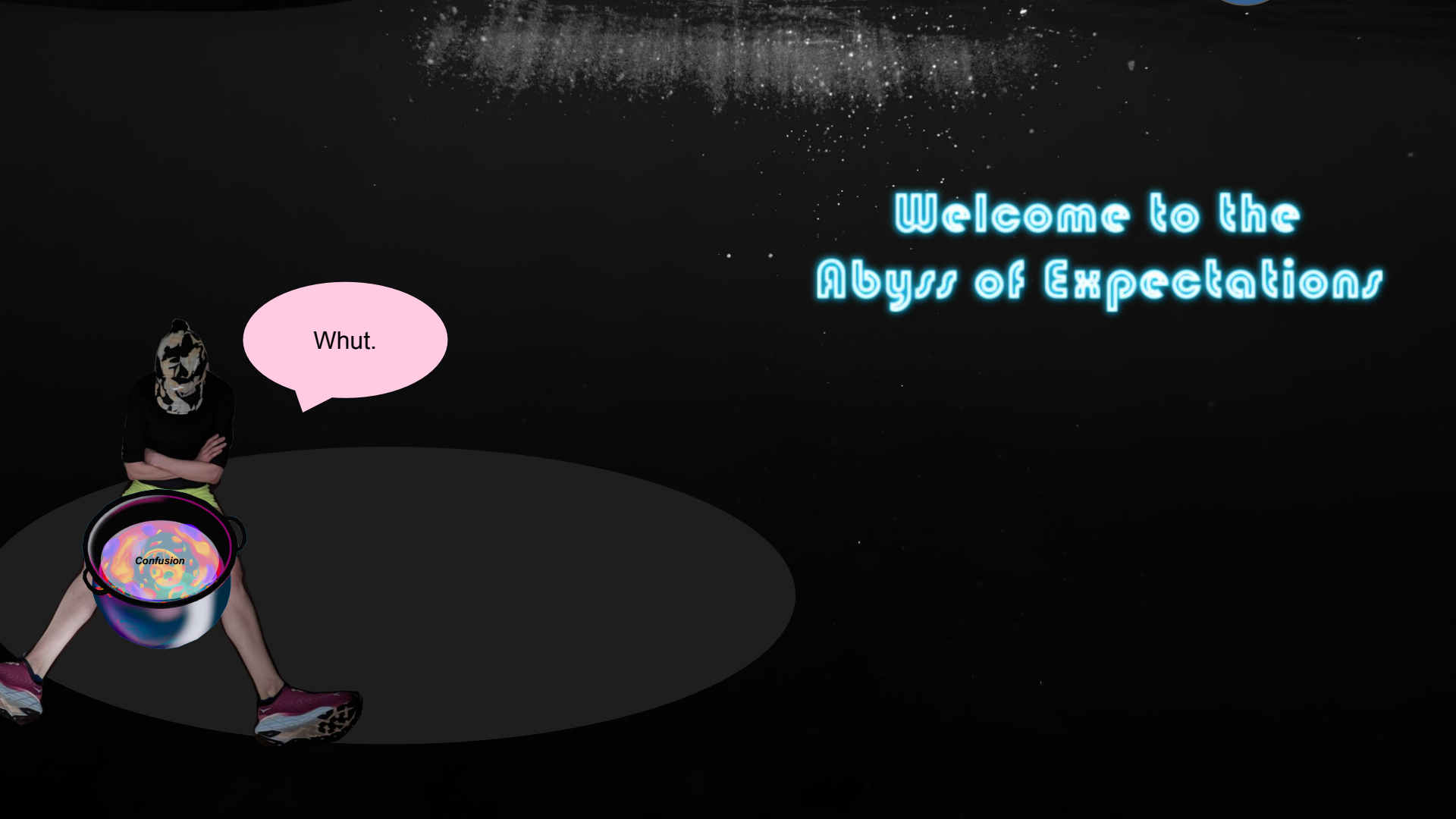
A gift in disguise?

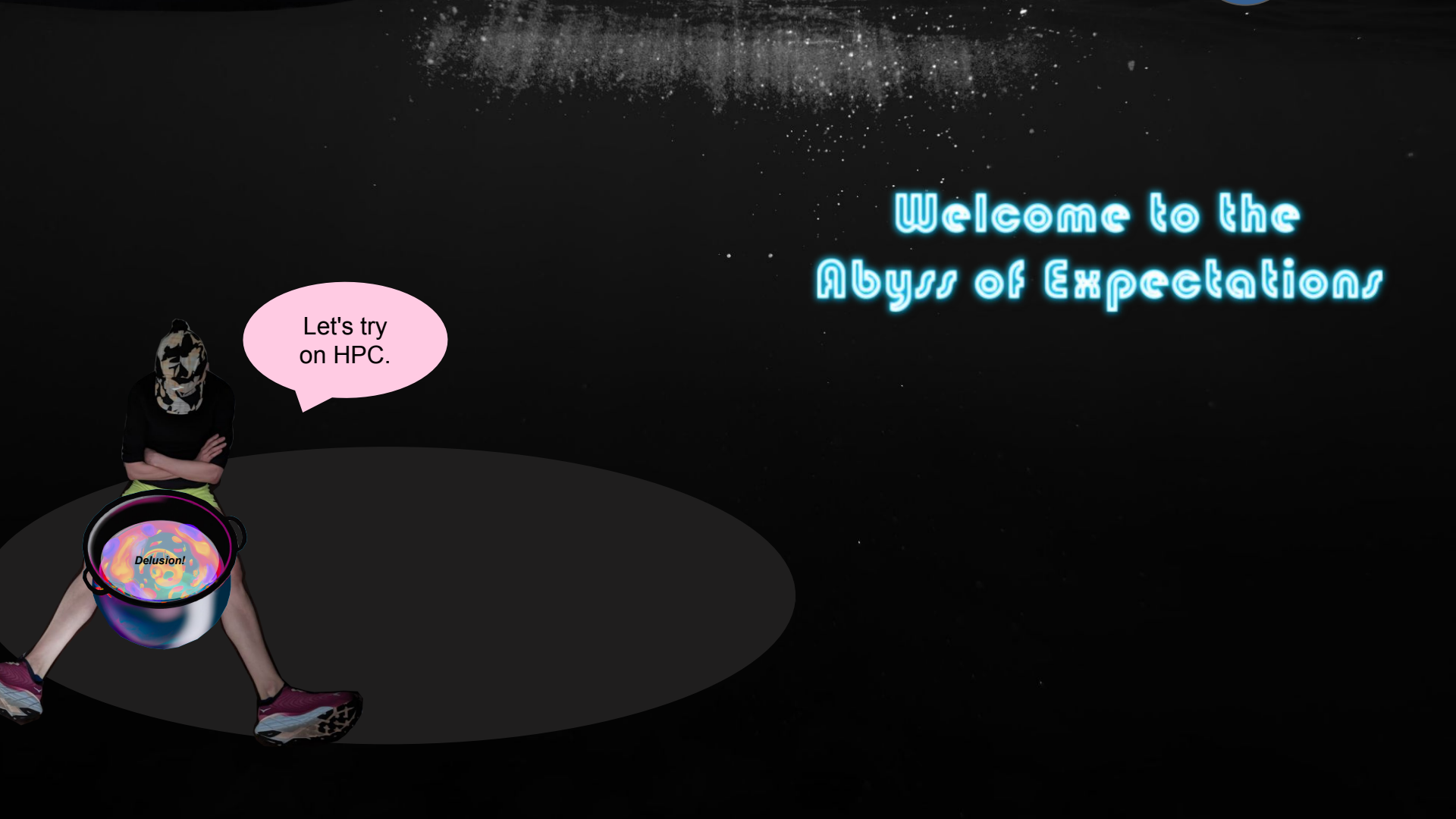

THOU DOST NOT
HAVE QUOTA!

Running our MPI Application with Kubernetes using the Metrics Operator

With **strong scaling**
runtime gets faster
as we add resources!

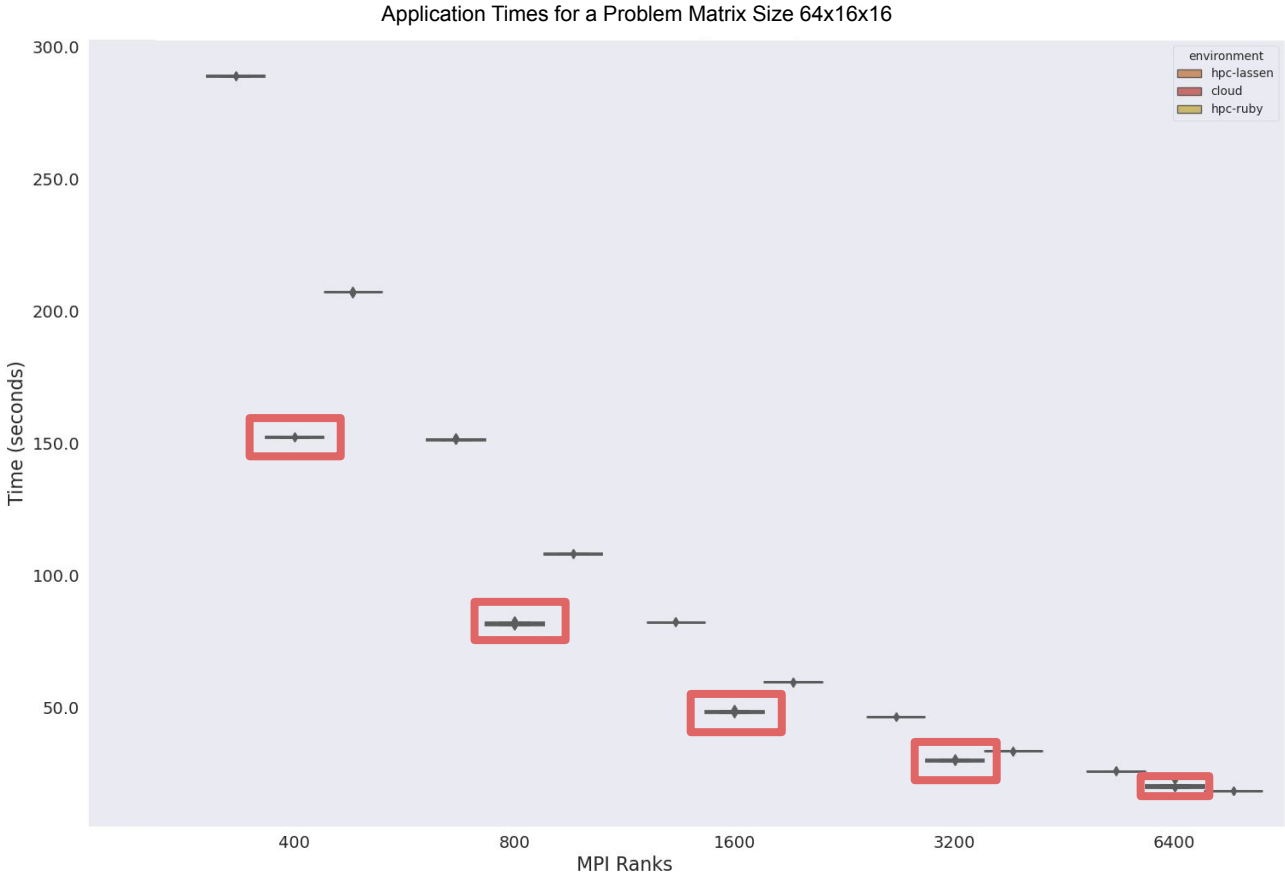



Welcome to the
Abyss of Expectations
Whut.
Confusion


Welcome to the
Abyss of Expectations
Let's try
on HPC.
Delusion!

Cloud Times were *Faster* for our MPI Application

Except for the largest size…



Cloud Times were *Faster* for our MPI Application

Except for the largest size…

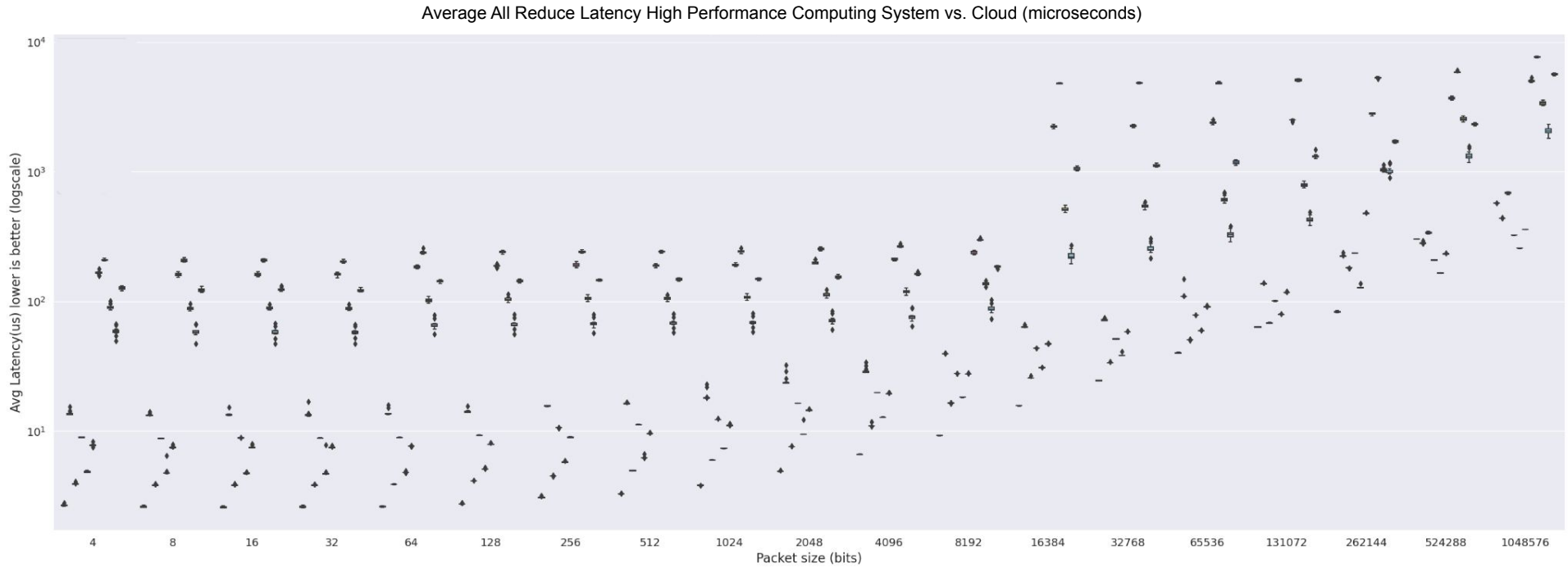
As the cluster gets bigger there is more communication between nodes…

Cloud Times were *Faster* for our MPI Application

Except for the largest size…

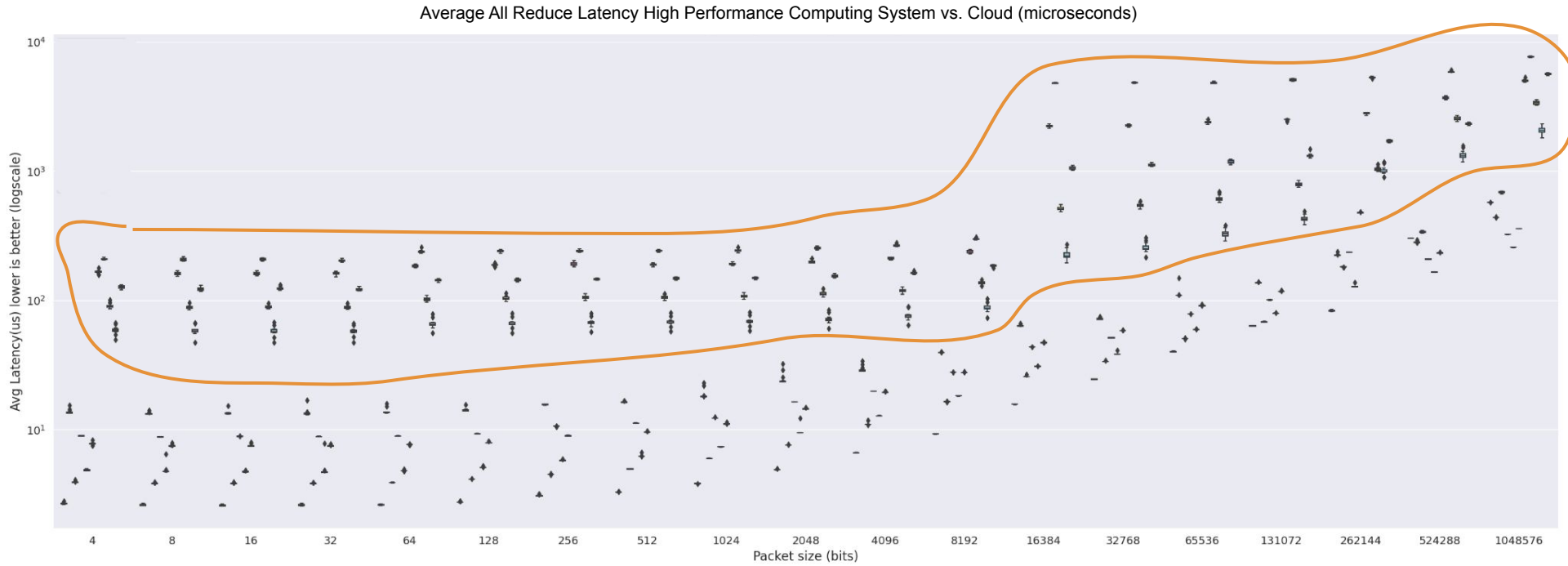
There looks like there is still a cost to that.

Collective Benchmark "osu_allreduce" is a proxy for communication



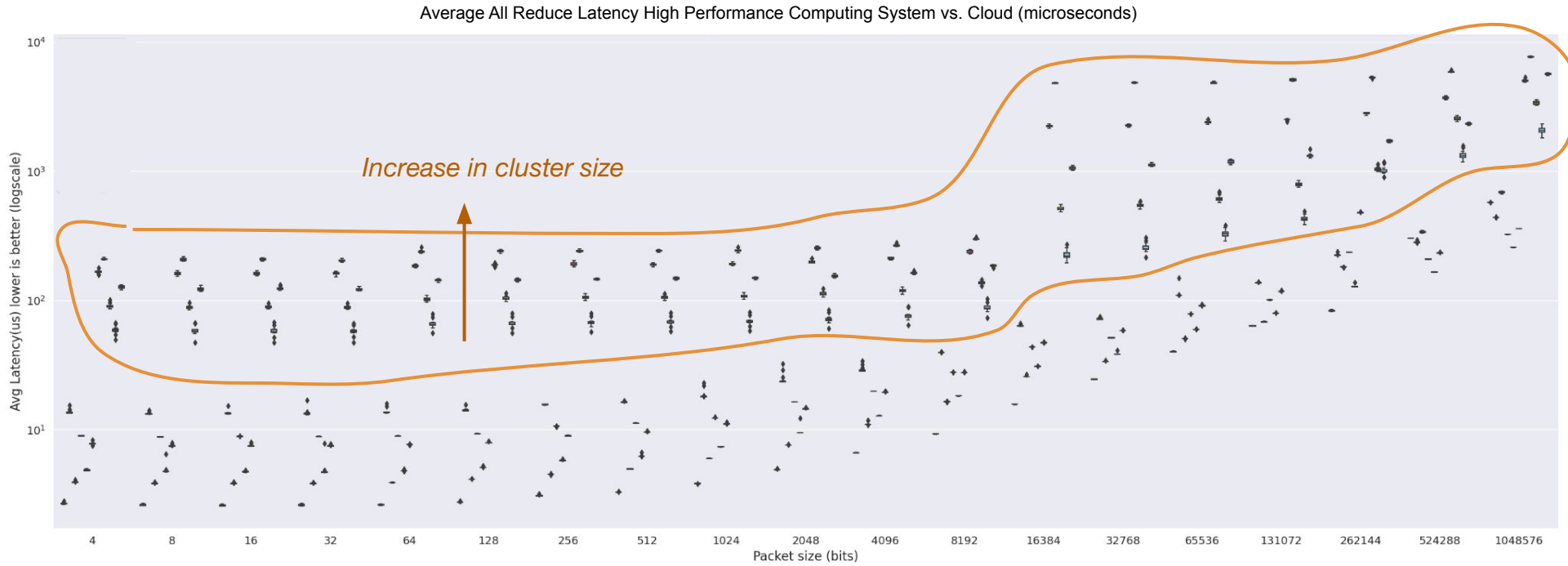
Collective Benchmark "osu_allreduce" is a proxy for communication

Communication between nodes for cloud across sizes is depending on tcp



Collective Benchmark "osu_allreduce" is a proxy for communication

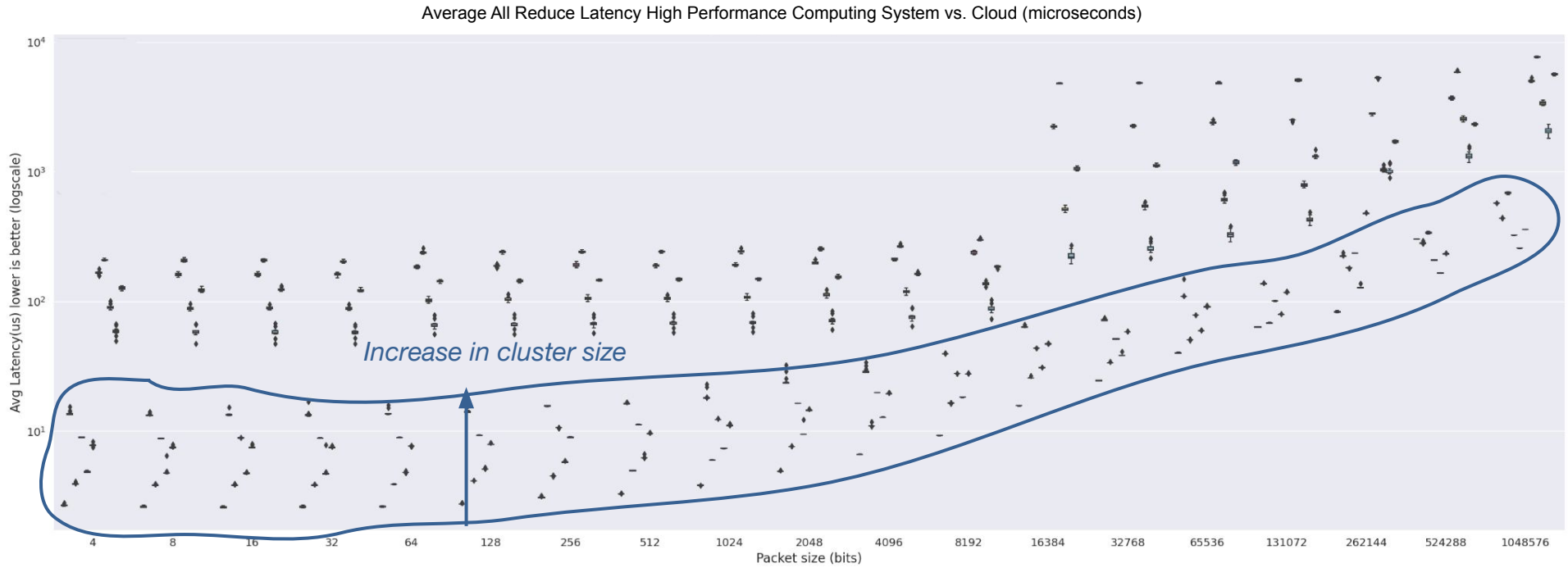
Communication between nodes for cloud across sizes is depending on tcp



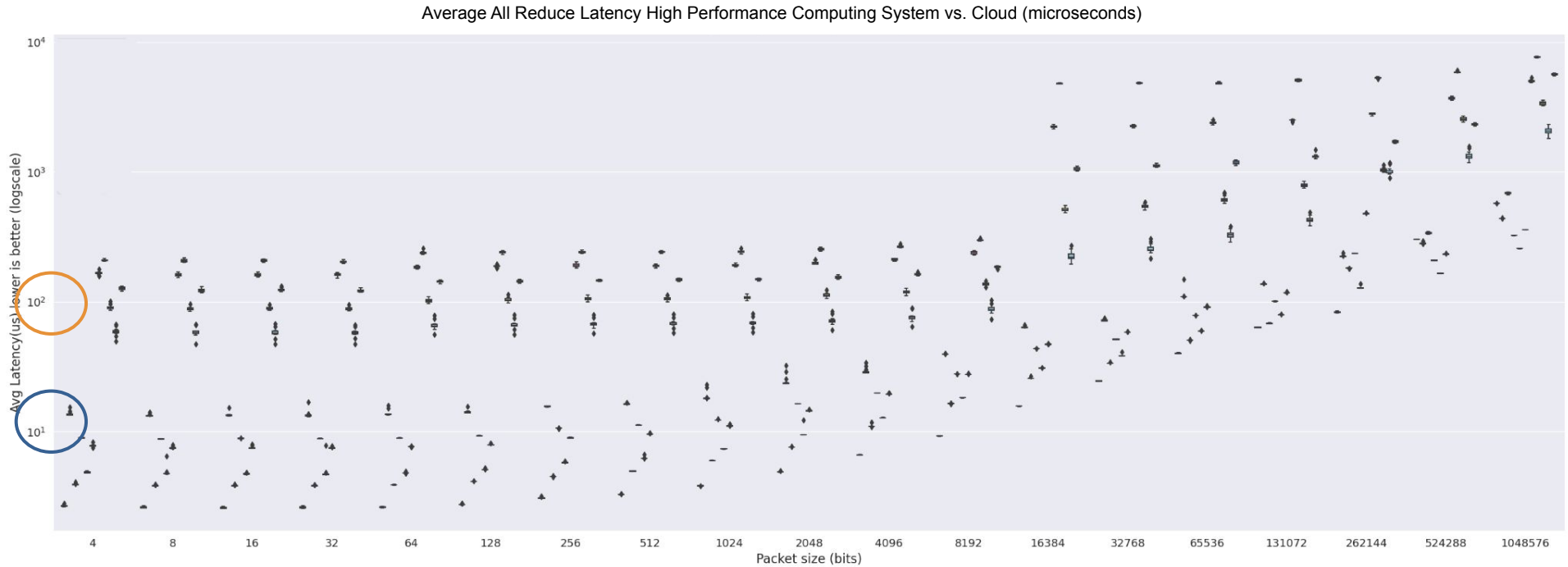
Collective Benchmark "osu_allreduce" is a proxy for communication

Communication between nodes for cloud across sizes is depending on tcp

Communication between nodes for HPC across sizes is depending on infiniband

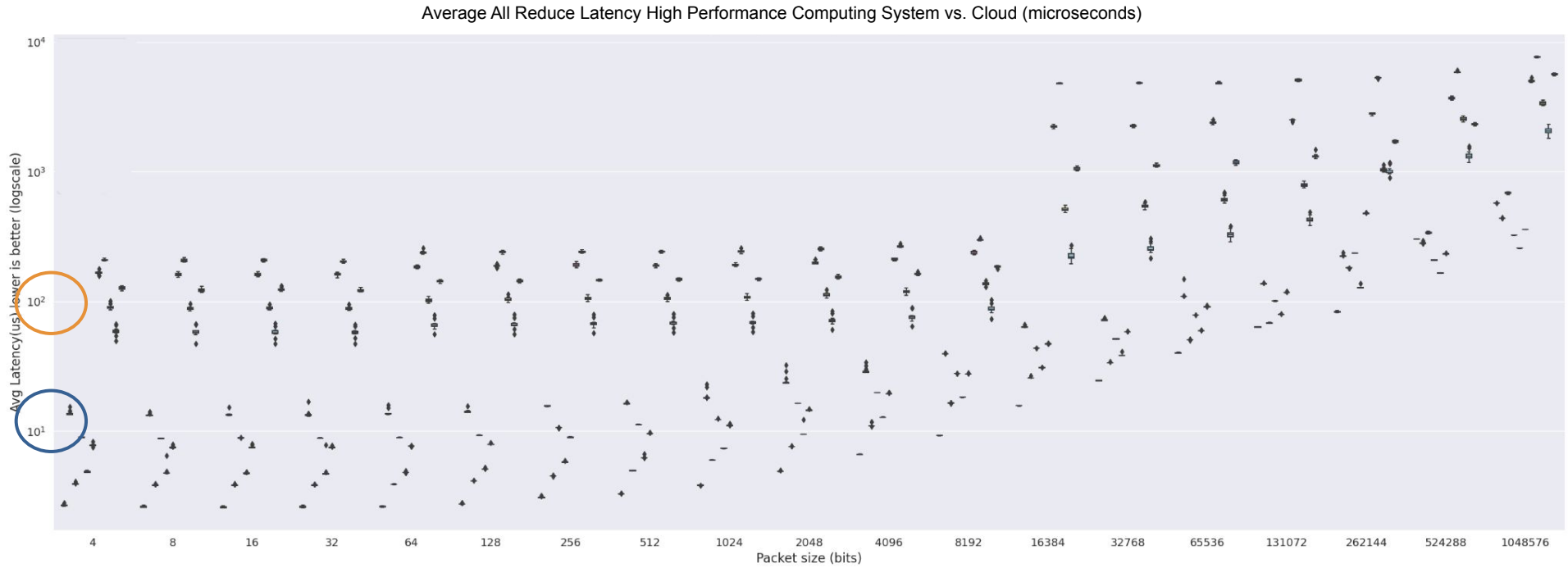


Communication has a much higher cost in cloud due to the network

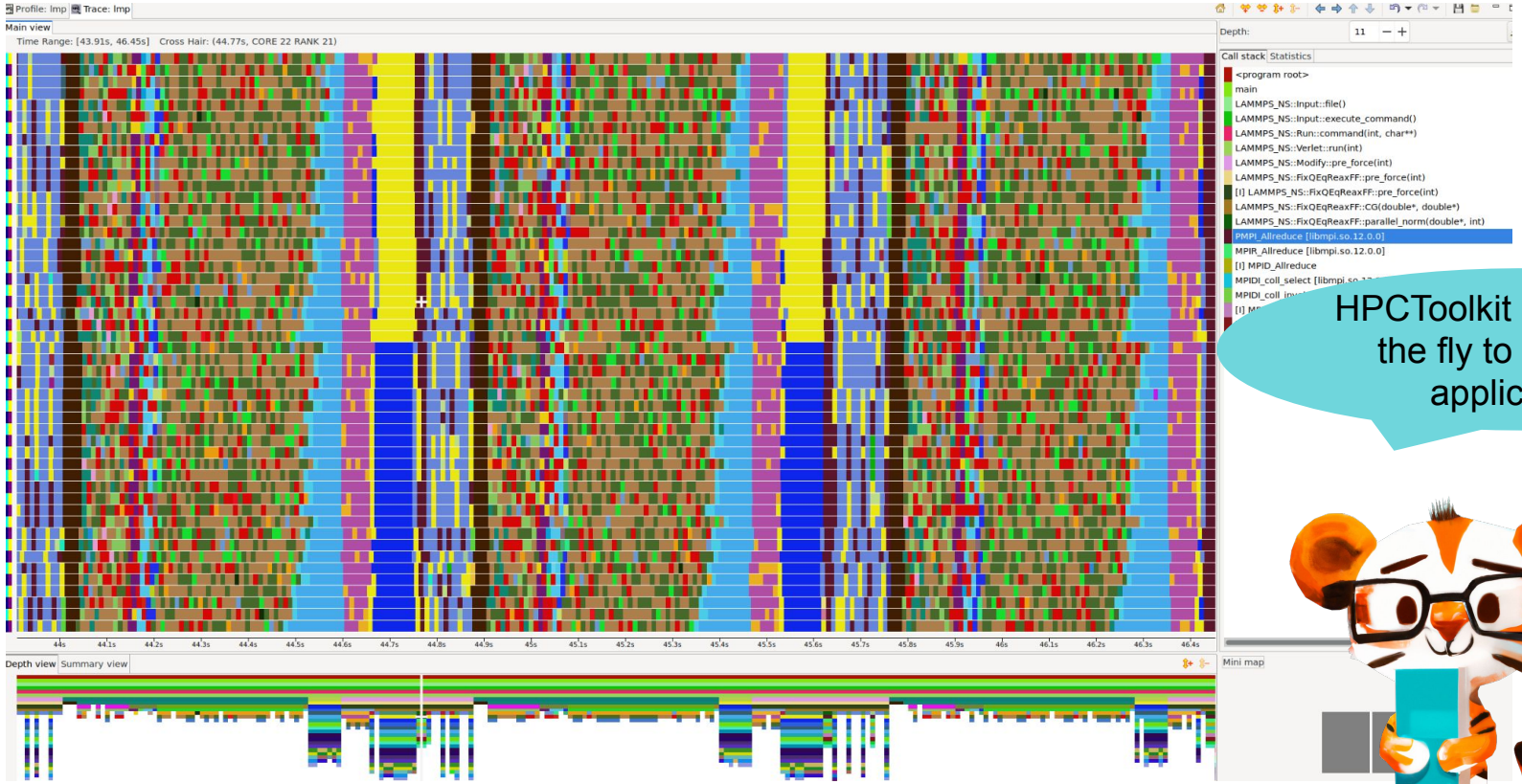


Communication has a much higher cost in cloud due to the network

Does network become an issue at the larger sizes?

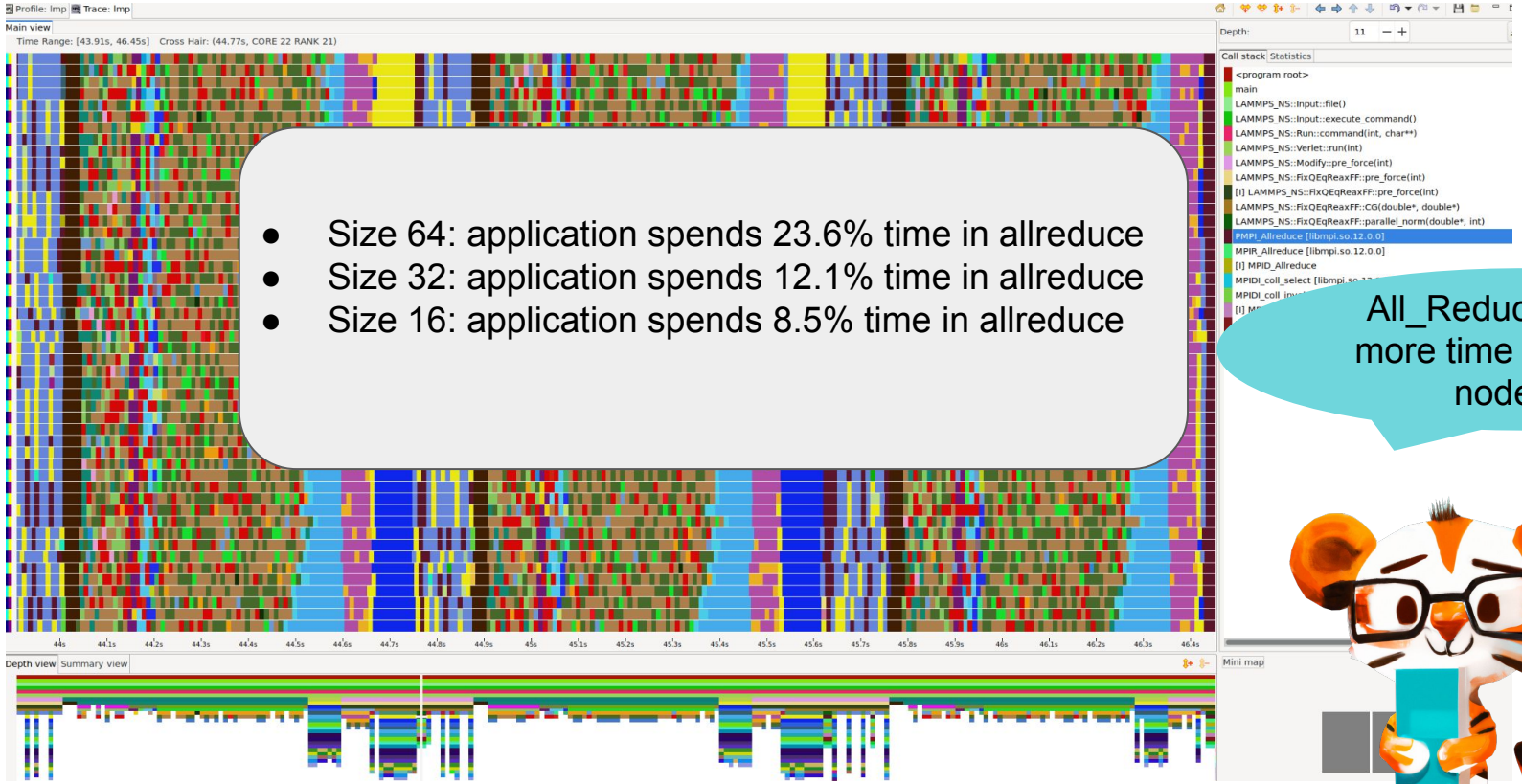


We used the Metrics Operator to do performance analysis



HPCToolkit is added on the fly to trace our application!

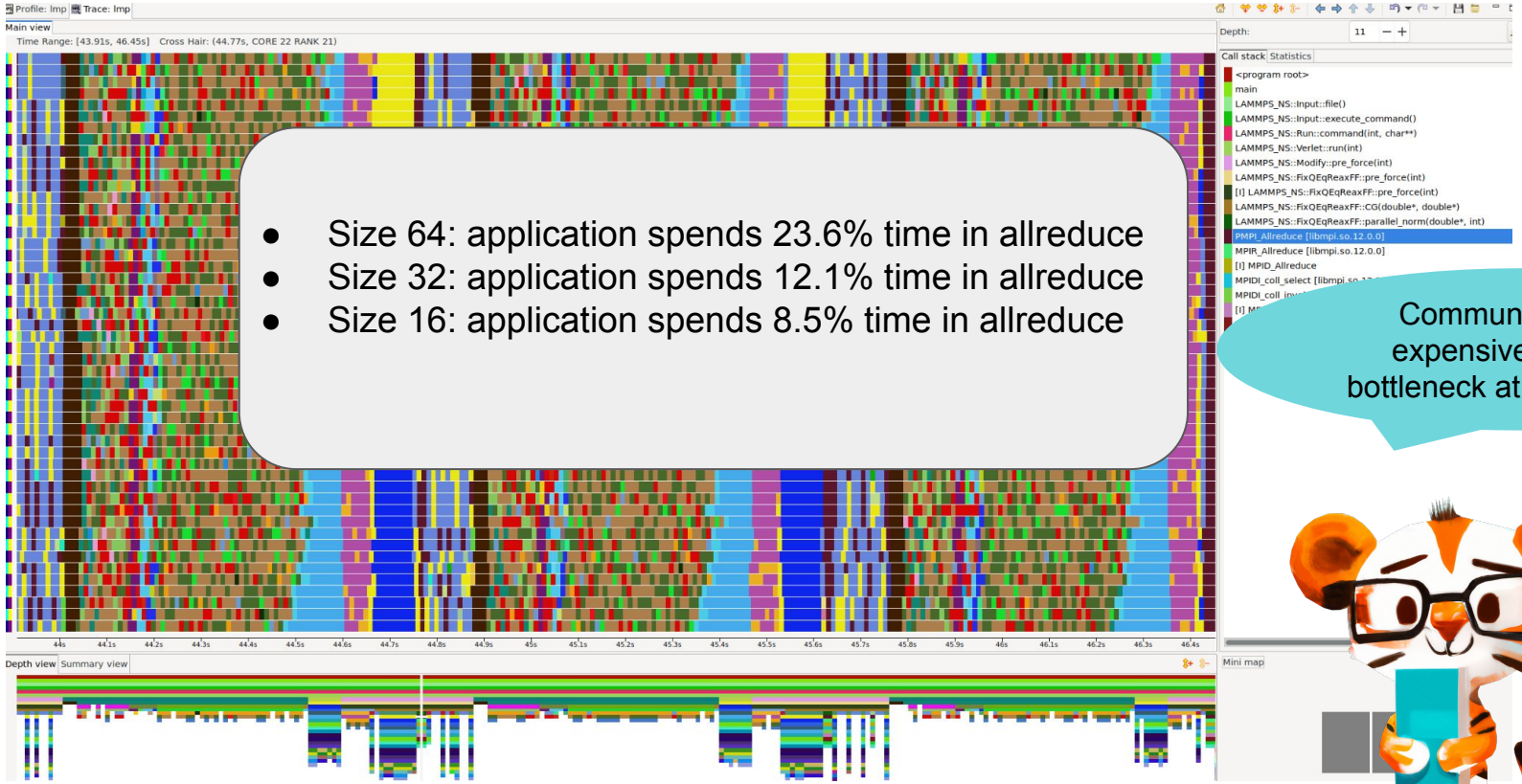
We used the Metrics Operator to do performance analysis



- Size 64: application spends 23.6% time in allreduce
- Size 32: application spends 12.1% time in allreduce
- Size 16: application spends 8.5% time in allreduce

All_Reduce spends more time as we add nodes…

We used the Metrics Operator to do performance analysis



- Size 64: application spends 23.6% time in allreduce
- Size 32: application spends 12.1% time in allreduce
- Size 16: application spends 8.5% time in allreduce

Communication is expensive, and the bottleneck at larger sizes.

Welcome to the Abyss of Expectations

Welcome to the Abyss of Expectations

NETWORK FABRIC

MPI IMPLEMENTATION

TOPOLOGY

MAXIMUM TRANSMISSION UNIT

NETWORK FLAGS

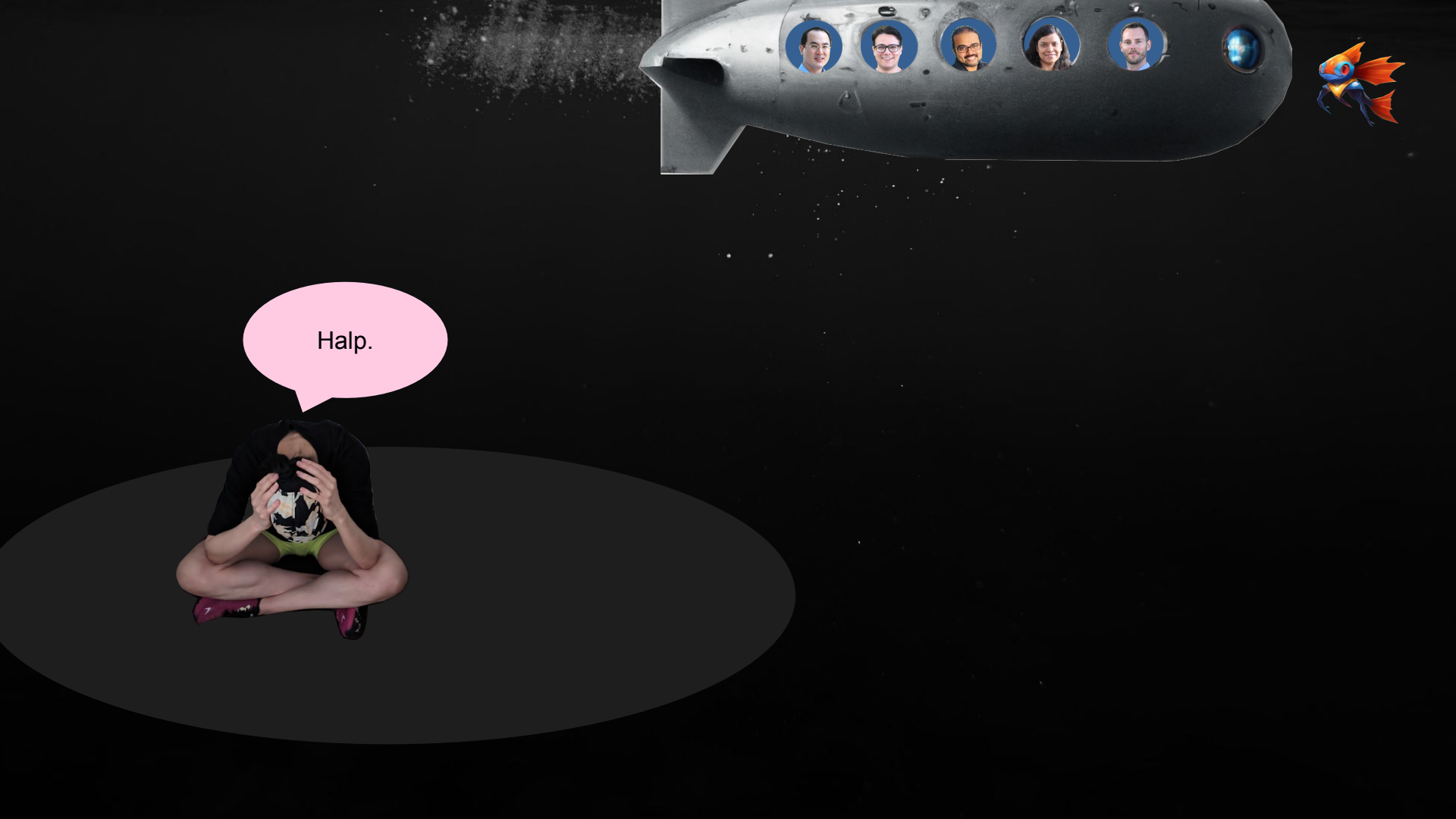
OS

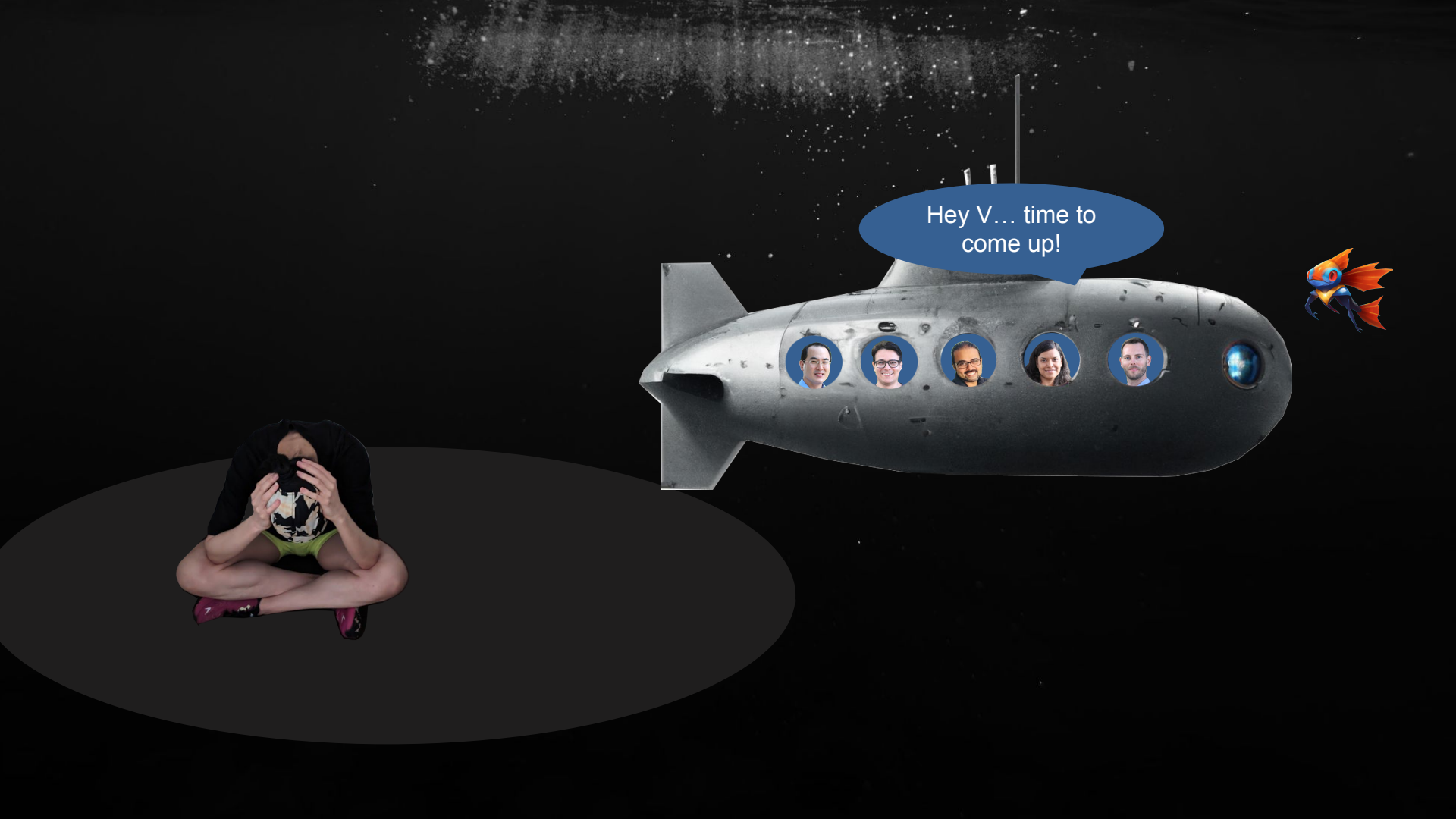
INSTANCE TYPE

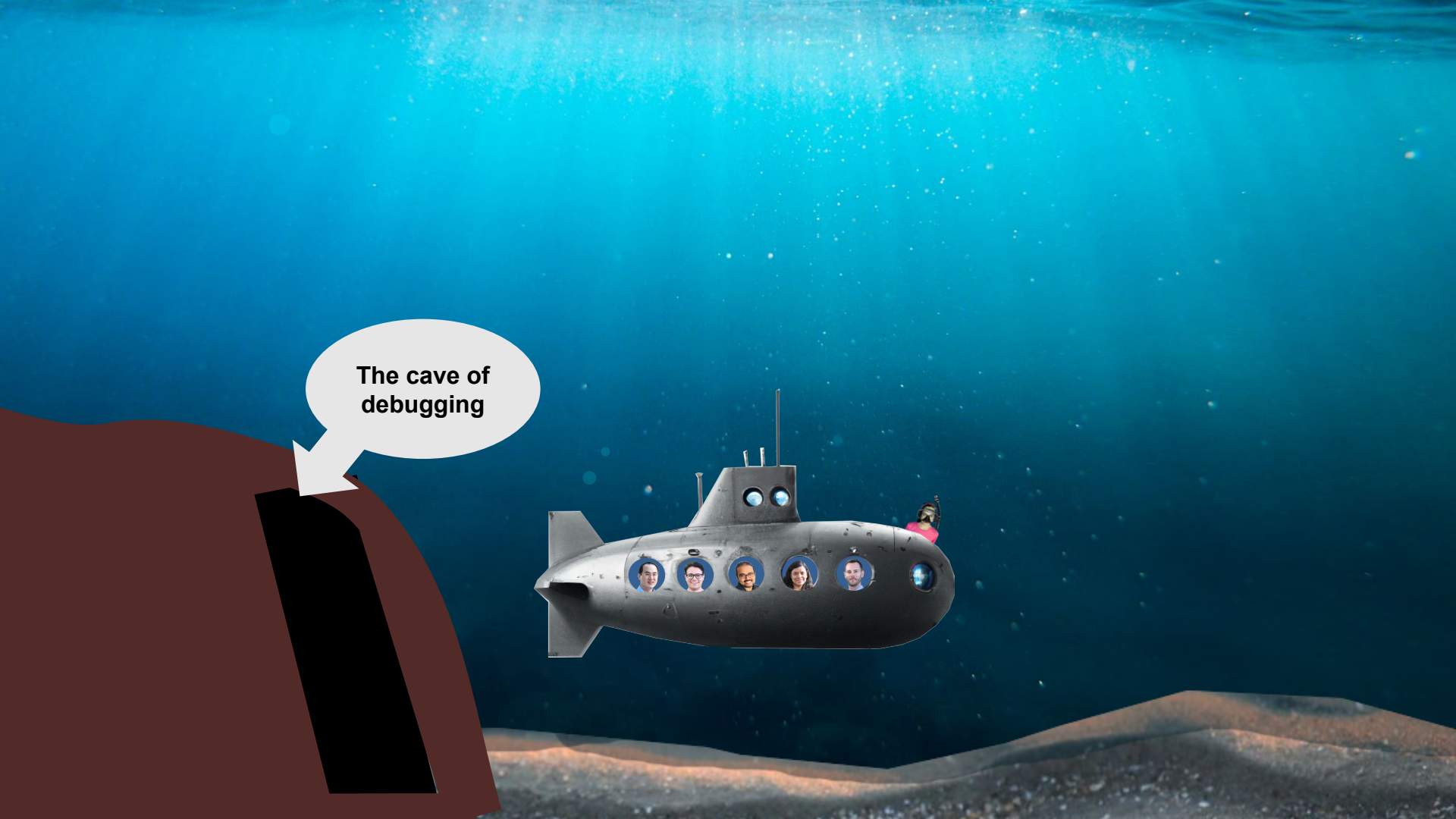
POINT-TO-POINT
COLLECTIVE

AVAILABILITY/
TIME OF DAY?


Halp.

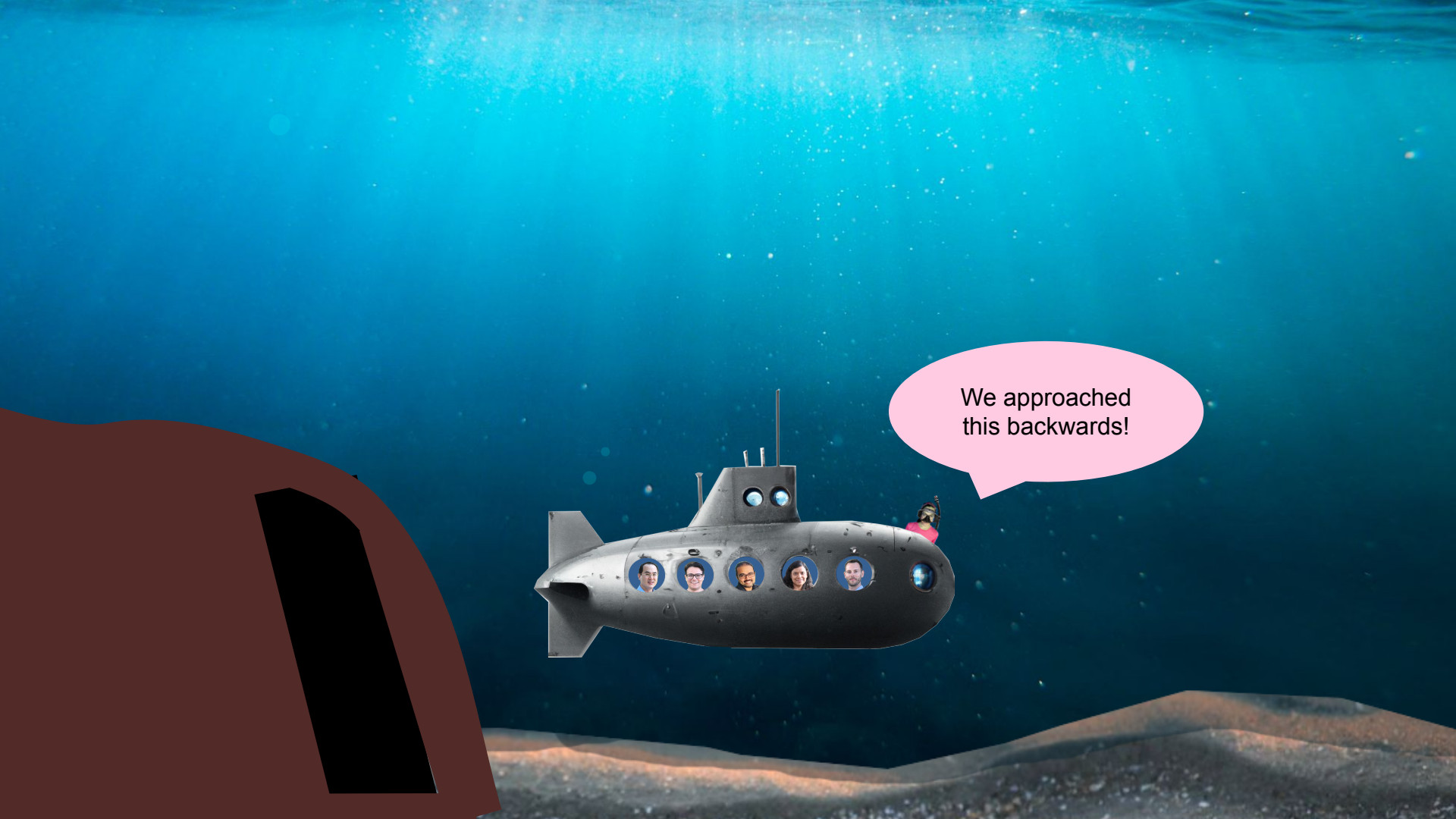

Halp.


Hey V… time to come up!


The cave of debugging

We were wrong about the
need for the lowest of latency!

The application problem is CPU bound at lower ranks, and network latency at higher ranks.


We approached this backwards!

Design Application

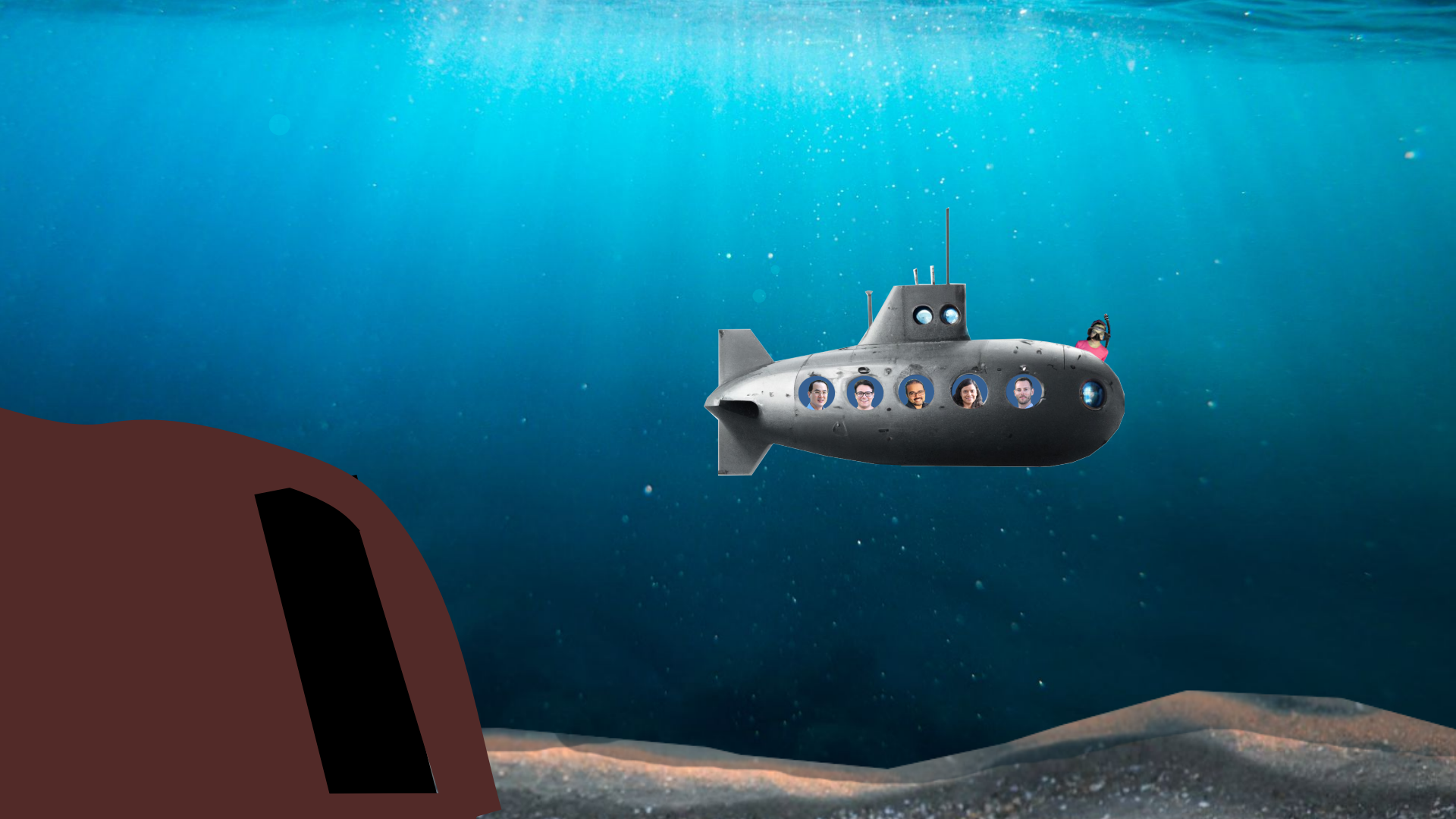
Run It!

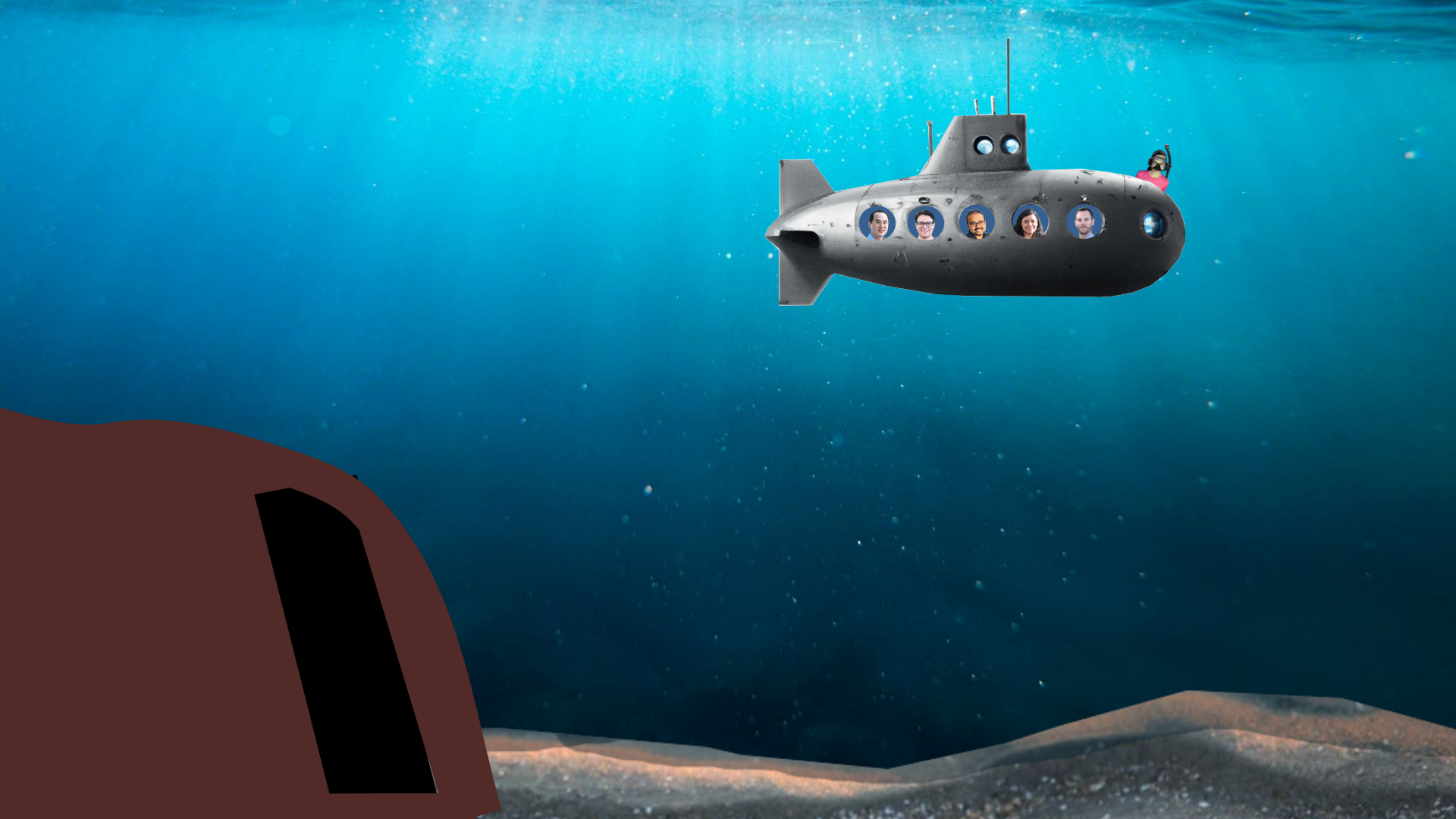
What happened?

Understand Application

Select the environment

Tune to be better





Things We Do Not Understand

- **Instance Types:**
  - Why didn't the c3 instance type work beyond a small problem size or scale?
  - Why when it ran on c3 was it so slow?

Things We Do Not Understand

- **Instance Types:**
  - Why didn't the c3 instance type work beyond a small problem size or scale?
  - Why when it ran on c3 was it so slow?

- **Expectations:**
  - What HPC lore (that drives expectation) is wrong?

Things We Do Not Understand

- **Instance Types:**
  - Why didn't the c3 instance type work beyond a small problem size or scale?
  - Why when it ran on c3 was it so slow?
- **Expectations:**
  - What HPC lore (that drives expectation) is wrong?
- **Application Design**
  - How do we define and assess patterns of applications?


1
"A new space I don't understand."

1

"A new space I don't understand."

2

A design strategy to map between spaces

JobSet provides a generic design to think about HPC apps in the cloud

1. "A new space I don't understand."
2. A design strategy to map between spaces
3. A means to *easily* measure and run things

- OSU benchmarks (network)
- MPI application (proxy app)
- HPCToolkit (performance)

Kubernetes operators allow us to package applications


1
"A new space I don't understand."
2
A design strategy to map between spaces
3
A means to *easily* measure and run things
OSU benchmarks (network)
MPI application (proxy app)
HPCToolkit (performance)
Understanding of application patterns / needs
Network / communication
IO patterns (e.g., read/write)
Hardware


Cloud Land
HPC Land

Cloud Land

Best of Both Worlds

HPC Land

Converged Computing


aws
IBM
So many adventures ahead!


We hope that you join us.
📧 sochat1@llnl.gov
👋 @vsoch

JobSet Future Work

- Placement policy
  - Evolve to a proper API and support additional use cases
    - Colocate, but not exclusive placement
    - Preferential placement
  - Pod Affinity/Anti-Affinity performance limitations
    - Solution under consideration: a leader-follower pattern
    - Schedule a leader pod of each job (e.g., index 0), then inject node affinity to the rest to follow it

- Failure Policy
  - In-place restart of jobs instead of full recreation
  - Configurable restart policy based on child jobs failure reasons

Getting Involved

JobSet

- https://github.com/kubernetes-sigs/jobset
- A Batch WG sponsored project

Batch WG:

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

Acknowledgements

- Daniel Vega-Myhre (github: @danielvegamyhre)

Questions?