HPC Scheduling with Kubernetes

Sprint 2

Juhi Paliwal, Siyuan Chen, Yilin Xu, Nidhi Shah, Soufiane Jounaid

Claudia Misale(IBM), Carlos Eduardo Arango Gutierrez (RedHat),
Daniel Milroy (Lawrence Livermore National Laboratory)

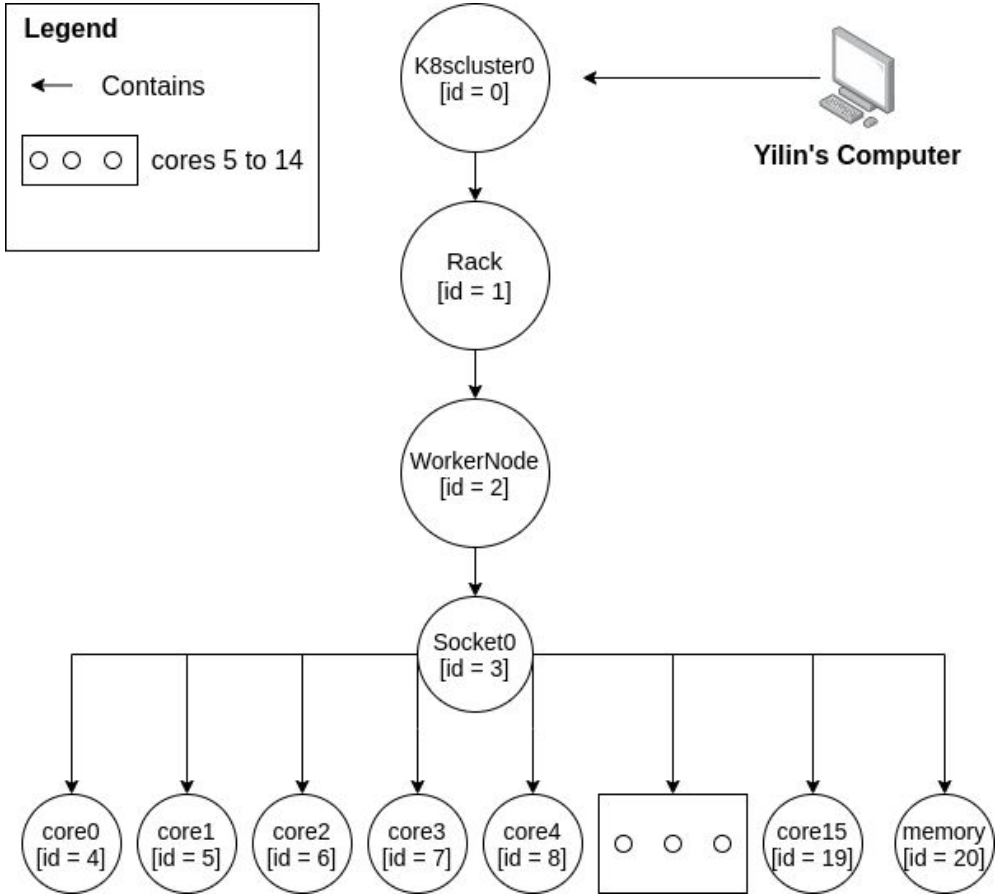
Agenda

1. Quick recap of last sprint.
2. Flux’s internal cluster state representation: The resource graph.
3. Demo outline.
4. Pi application test.
5. Reproduction of the state inconsistency issue in practice.
6. Tasks and Burndown chart.
7. Plans for next sprint.

About the Project so far

- State inconsistencies between the Kubernetes cluster and the Kube-Flux scheduler.
- Hinders utilization of the cluster, queued jobs wait forever.
- We want to design and implement an informer that carries updated cluster state information to the Kube-Flux resource graph.

Heart of Kube-Flux: The resource graph



Demo outline

1. Pi application test description
2. Expected results of the Pi experiment
3. Create local cluster
4. Deploy containerized Kube-flux scheduler
5. Run Pi test with default scheduler
    a. Display list of completed pods
6. Run Pi test with kubeflux scheduler
    a. Explore the resource graph after every pod placement.

Pi Application test

- Environment: Cluster running on local machine
  - 2 compute nodes
    - 1 Master node: Kubernetes control plane
    - 1 worker node
      - 16 virtual cores in 1 socket and 1 memory node
- Tools:
  - Kind
  - Docker
  - Match allocate cmd
- Task: (pod spec on next slide)
  - Deploy 4 pods each on 8 compute cores.
- Goal
  - Demonstrate the limitation of Flux

```
→  pi git:(yilin/test_setup) ✗ kubectl get nodes
NAME                     STATUS   ROLES                  AGE   VERSION
xyl-kind-control-plane   Ready    control-plane,master   52m   v1.21.1
xyl-kind-worker          Ready    <none>                 52m   v1.21.1
```

Pi test desired cluster state

1 * Kubernetes jobspec

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: pi-job-sched
spec:
  completions: 4
  template:
    metadata:
      labels:
        app: pi-test-default
    spec:
      containers:
      # comment below out to use default scheduler
      schedulerName: scheduling-plugin
      - name: pi-test
        image: localhost:5000/pi:latest
        resources:
          limits:
            cpu: "8"
          requests:
            cpu: "8"
      restartPolicy: Never
```

Kube-Flux translates

4 * Flux jobspec

```
[JobSpec] JobSpec in YAML:
version: 1
resources:
- type: node
  count: 1
  with:
  - type: socket
    count: 1
    with:
    - type: slot
      count: 1
      label: default
      with:
      - type: core
        count: 8
attributes:
  system:
    duration: 3600
tasks:
- command: []
  slot: default
  count:
    per_slot: 1
```

Expected experiment result

- The default kube-scheduler will successfully complete the job.
- Flux will successfully schedule 2 pods of 8 cores each.
- Flux will fail to schedule the third pod.

Runtime DEMO by Yillin

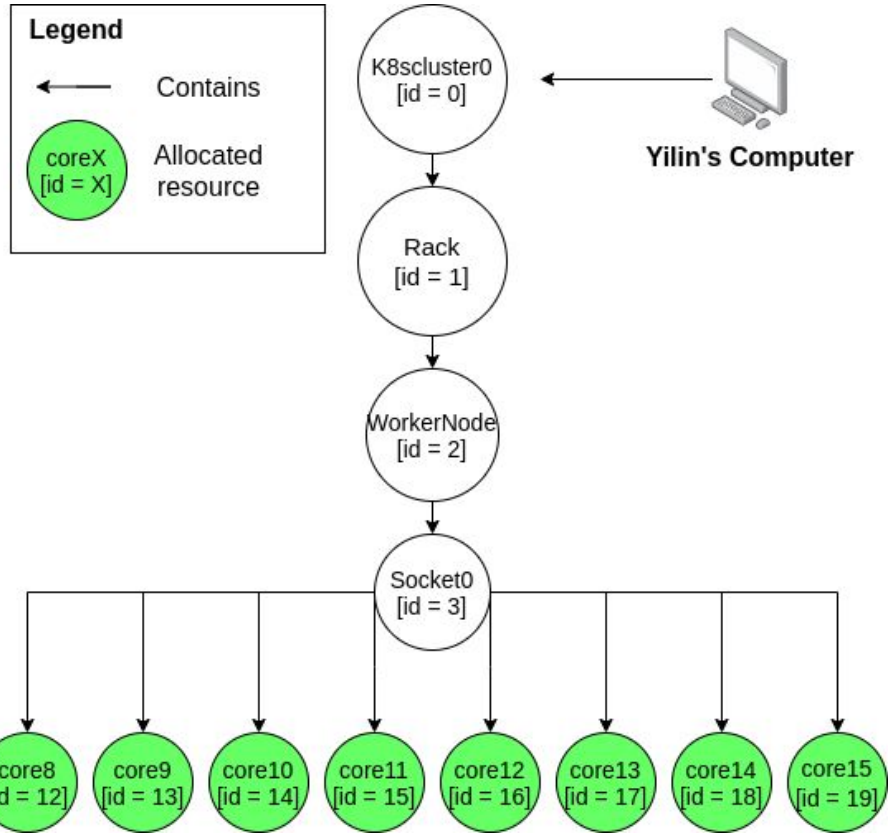
Flux resource graph after first pod placement

Pod placement output

```
 ----Match Allocate output---
jobid: 1
reserved: false
allocated: Resource Graph starting at
allocated node

at: 0
preorder visit count: 13
postorder visit count: 12
overhead: 0.000306
error: 0
nodename  xyl-kind-worker
Node Selected:  xyl-kind-worker
Filtering input node xyl-kind-worker
Filter: node selected by Flux
xyl-kind-worker
```

Resource Graph



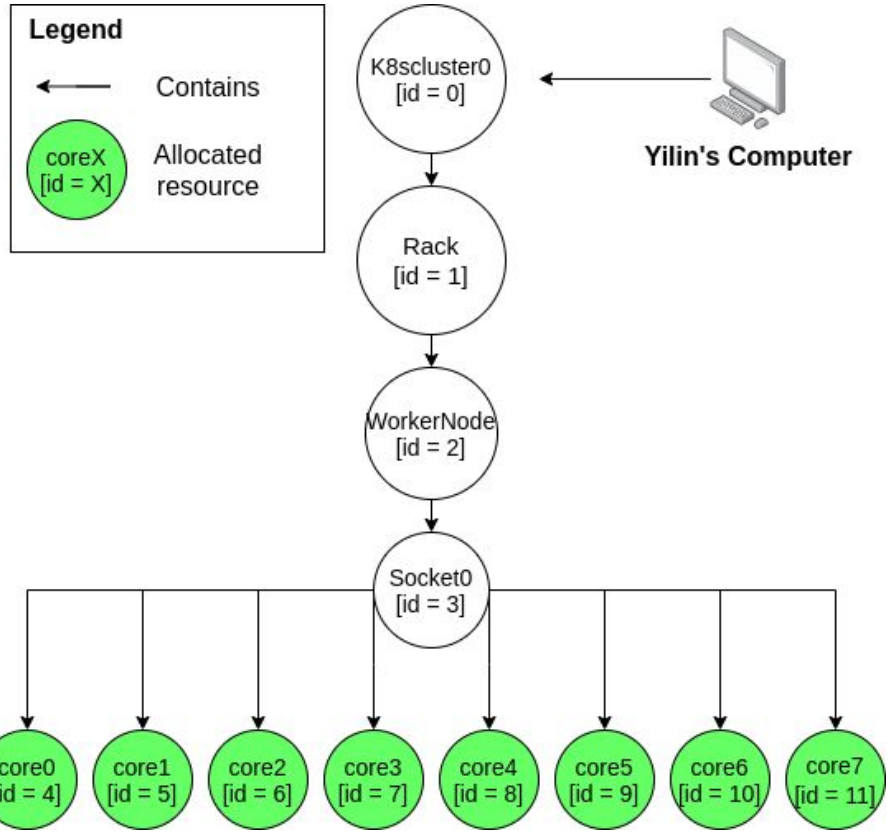
Flux resource graph after second pod placement

Pod placement output

```
 ----Match Allocate output---
jobid: 2
reserved: false
allocated: Resource Graph starting at
allocated node

at: 0
preorder visit count: 21
postorder visit count: 12
overhead: 0.000376
error: 0
nodename  xyl-kind-worker
Node Selected:  xyl-kind-worker
Filtering input node xyl-kind-worker
Filter: node selected by Flux
xyl-kind-worker
```

Resource Graph



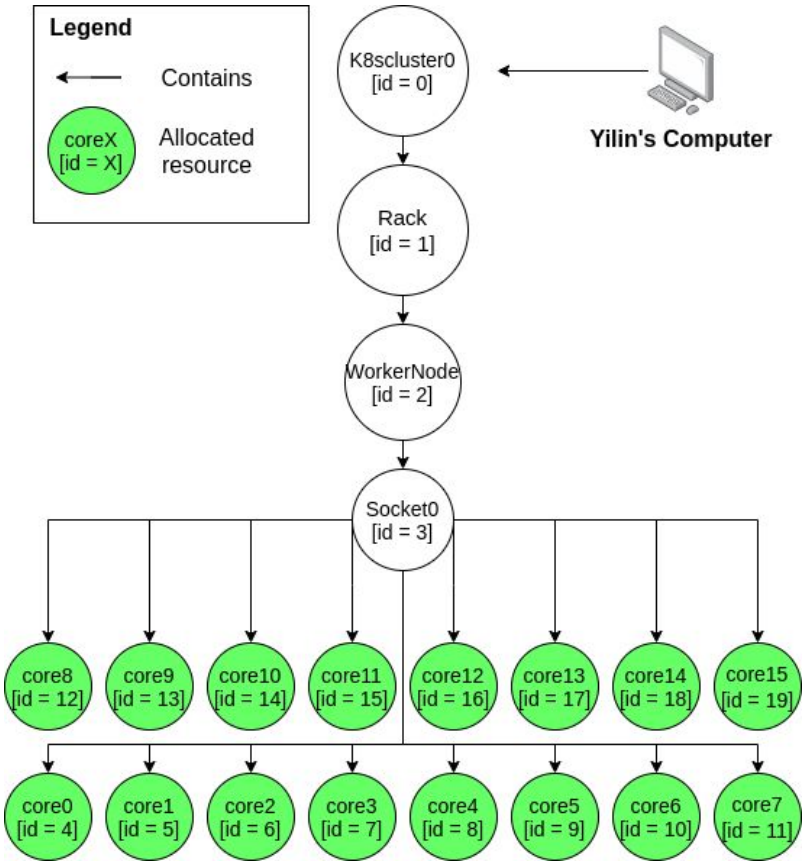
Flux resource graph after failed pod 3 placement attempt

Pod placement output

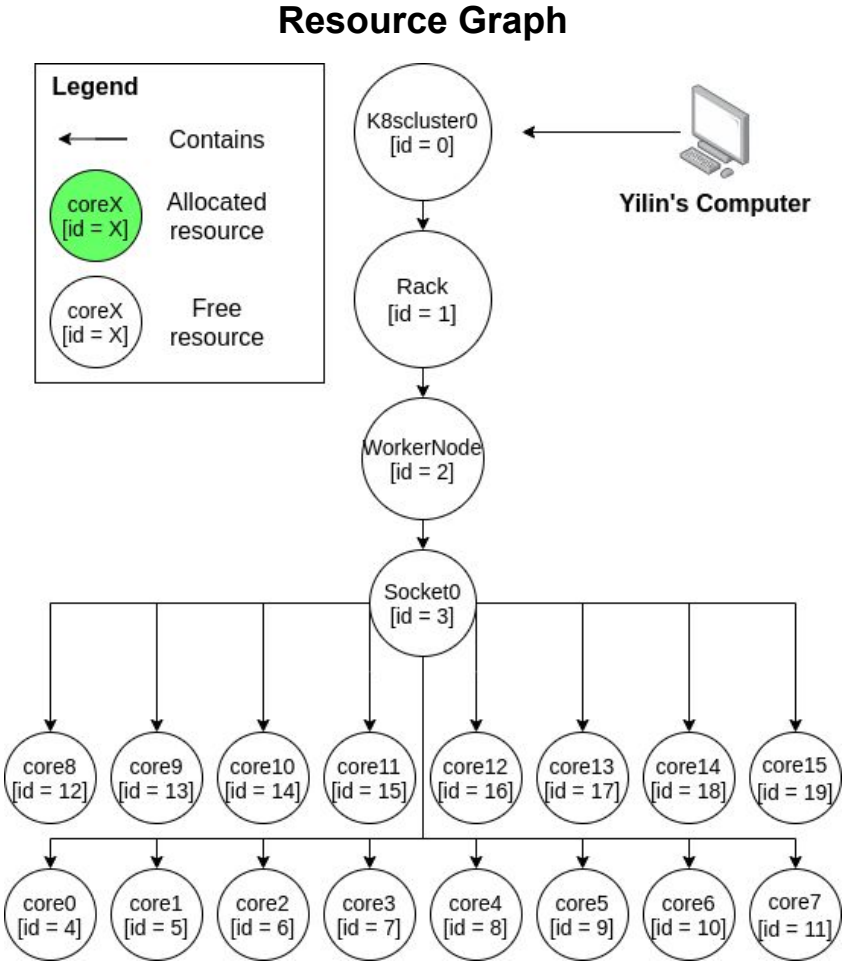
```
Pod cannot be scheduled by
KubeFlux, nodename  NONE
I1013 00:01:33.511431       1
default_preemption.go:219] from
a pool of 2 nodes (offset: 0,
sample 2 nodes:
[xyl-kind-control-plane
xyl-kind-worker]), ~2 candidates
will be chosen
"Unable to schedule pod; no fit;
waiting"
pod="default/pi-job-kubeflux-sch
ed-c959p" err="0/2 nodes are
available: 2 Pod cannot be
scheduled by KubeFlux, nodename
NONE."
```

Here's why

Resource Graph



The Reality


Resource Graph
Legend
Contains
coreX [id = X]
Allocated resource
coreX [id = X]
Free resource
K8scluster0 [id = 0]
Yilin's Computer
Rack [id = 1]
WorkerNode [id = 2]
Socket0 [id = 3]
core8 [id = 12]
core9 [id = 13]
core10 [id = 14]
core11 [id = 15]
core12 [id = 16]
core13 [id = 17]
core14 [id = 18]
core15 [id = 19]
core0 [id = 4]
core1 [id = 5]
core2 [id = 6]
core3 [id = 7]
core4 [id = 8]
core5 [id = 9]
core6 [id = 10]
core7 [id = 11]

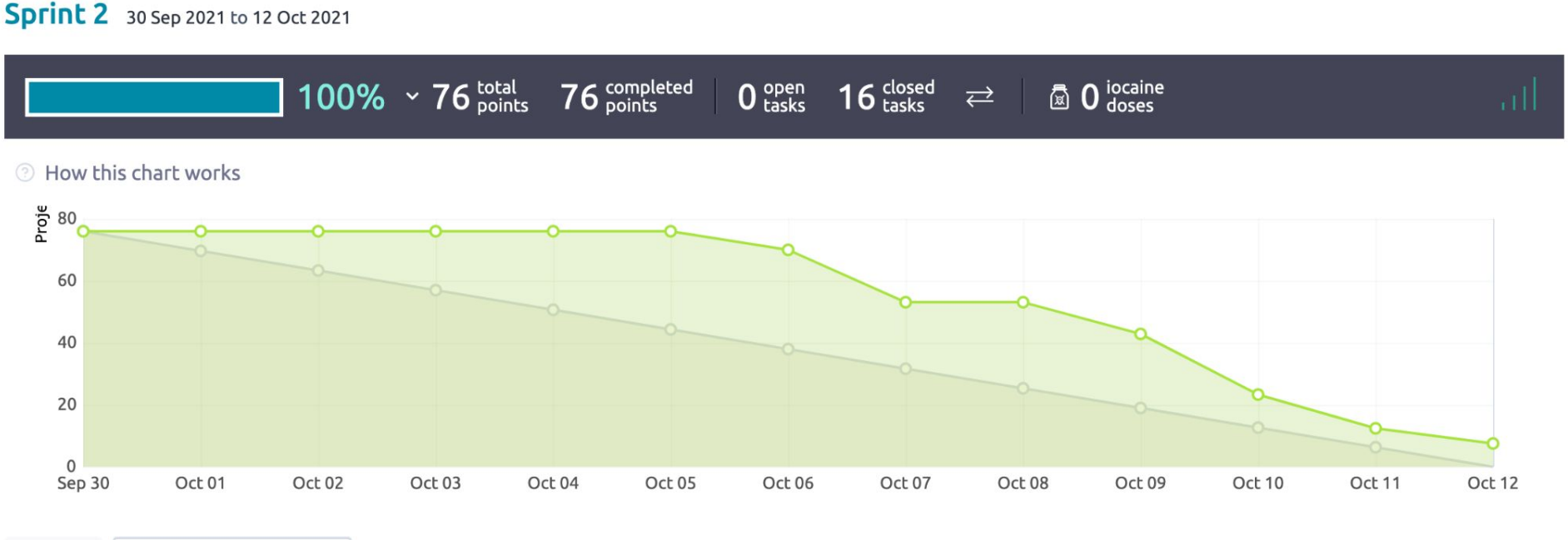
Sprint 2 Tasks

- Analysis of the underlying resource graph used by Flux to represent the Kubernetes cluster
- Replicate the problem statement successfully
- Designed Pi program test and reproduced the state inconsistency issue in practice

  Challenges

  - Spent time working on an application called Gromacs before moving to the Pi application
  - Gromacs needs MPI operator setup which gave us an installation issue

Burn-down Chart



Plans For The Next Sprint

1. Understand the existing code structure from the Product Owners and brainstorm on the solutions.
2. Work on the development of the informer.
3. Complete the pending kubernetes setup for all members.

Thank You!