Demo sprint 1

HPC scheduling with Kubernetes

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

Agenda

- Background
- Problem and end goal
- Environment setup
- Use cases
- Burndown chart
- Plan for next sprint

Background

Kubernetes is a framework that runs distributed systems

- Pods: Containers containing application
- Nodes: Worker Machine that run Pods
- Control Plane: Manages Nodes

Kube-Scheduler selects the nodes that will run the new pods

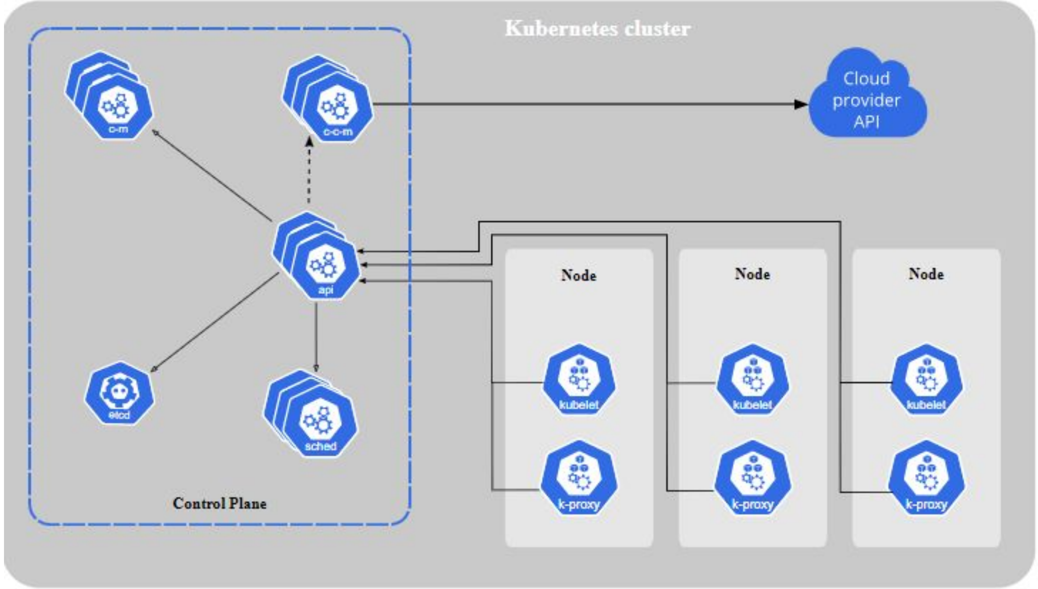
Kubelet starts a pod

APIserver provides communication between components.

Limitations of HPC in Kubernetes:

- No Batch, Co-Scheduling
- No proper queuing system
- Kube-scheduler not adapted to HPC.

Kubernetes allows you to add your own scheduler.



Components of Kubernetes

Kube-flux

Flux is a framework that uses a graph based scheduler called Fluxion.

Kube-Flux is an HPC scheduler that employs the Fluxion library to take scheduling decisions in Kubernetes.

Scheduling Cycle:

- Transform Kubernetes Pod specification to Flux Job Specification
- Submit placement request to Fluxion Library.
- The Fluxion library finds the node on the cluster and returns its name.
- Pass the selected node name to the API to bind with a pod.

Binding Cycle

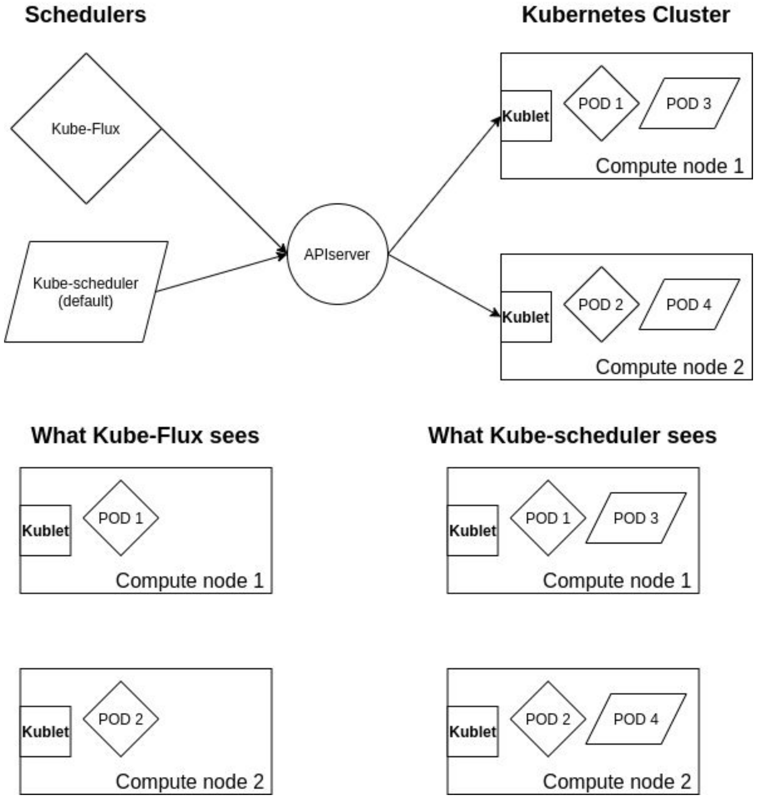
- Pod are bound to a node

Problem

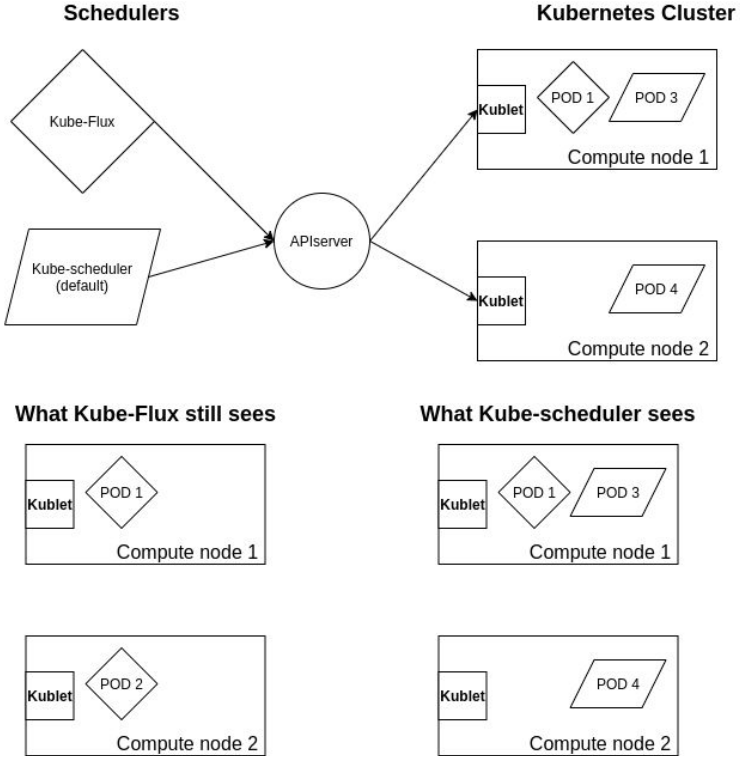
- Flux is not in charge of starting the actual pod, flux simply submits the pod to the Kubernetes APIserver which in turn works with Kublet to start the pod.
- Since Flux is not directly responsible for starting pods, it is not directly aware of their state on the cluster.
- If a pod scheduled by kube-flux finishes or is cancelled, kubernetes will remove said pod from the cluster and free the resources; this information however does not get propagated back to Kube-flux.
- Kube-Flux now sees a resource that has been freed as still occupied, which affects the correctness of future scheduling decisions.
- Consequently, since kubernetes does not receive state updates from the cluster, it is also unaware of the effects of scheduling decisions taken by other schedulers.

Illustration of the problem

State 1



State 2 (POD 2 terminated by kublet)

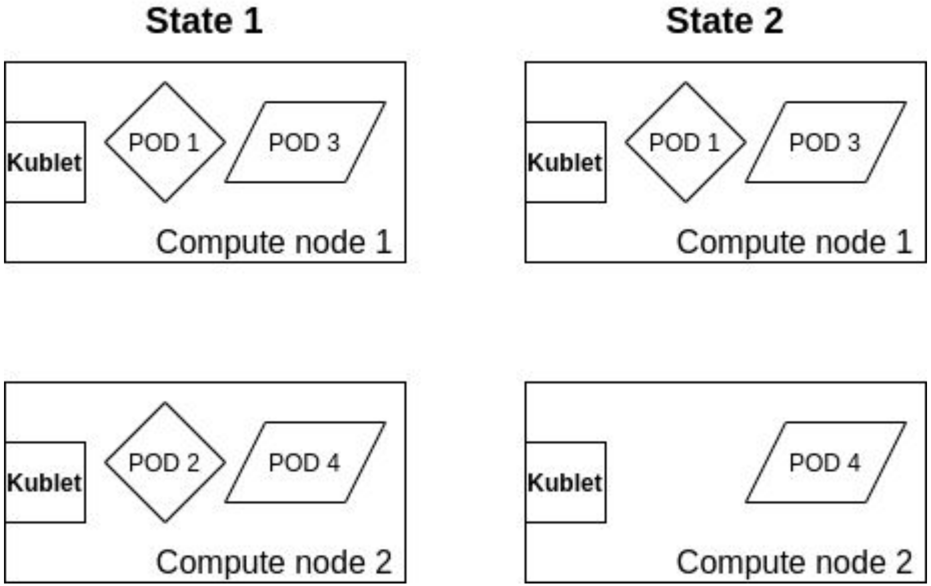


Goal of the project

- Full goal: Continuously synchronize the internal state of Flux with the state of the Kubernetes cluster of compute nodes.
  - Subgoal 1: Synchronize the Kube-flux scheduler to reflect changes in the state of its own scheduled pods.
  - Subgoal 2: Synchronize the Kube-flux scheduler to reflect changes in the state of pods scheduled by other schedulers.

Illustration of our goals (simplified)

What kube-Flux should start seeing by the end of this project



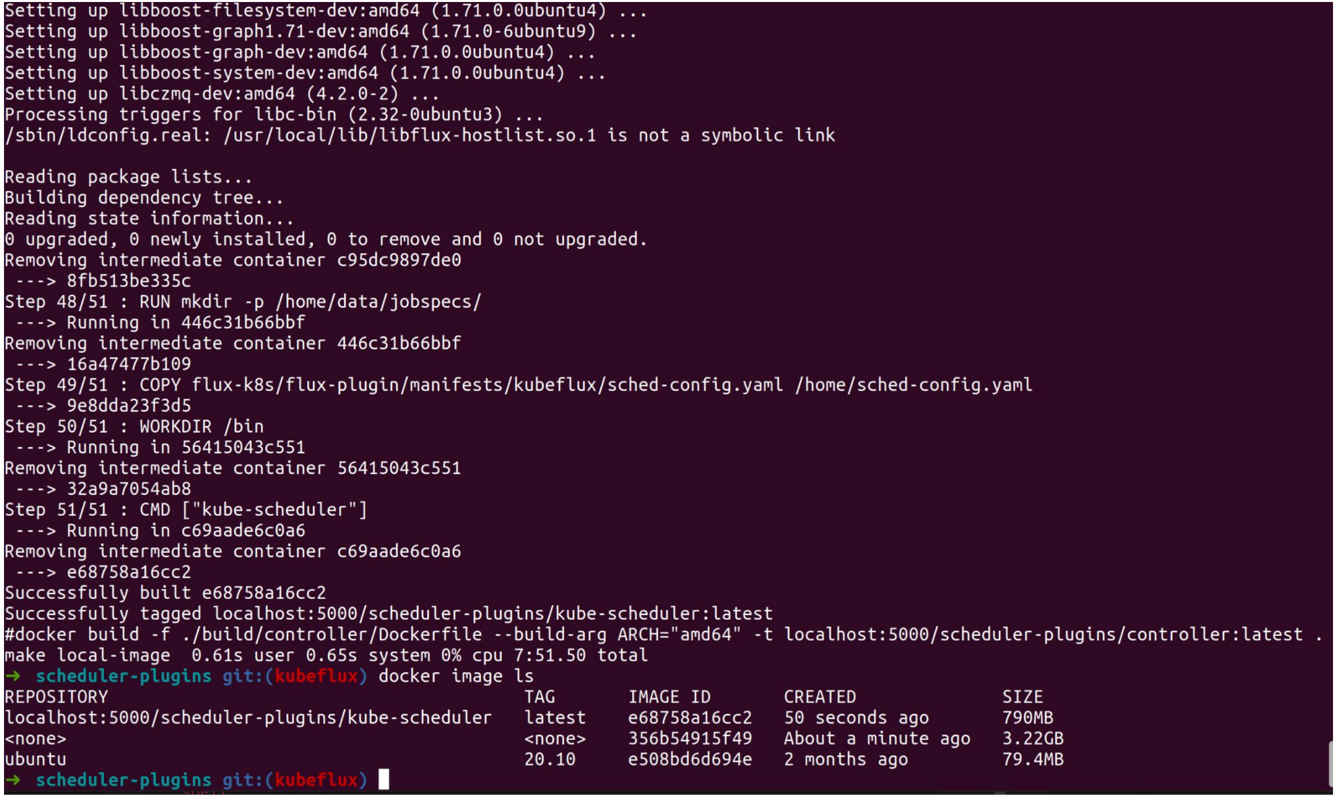
Potential solutions

- Obvious solution
    - Before each kube-flux scheduling round, we re-collect information about the compute nodes current utilization data and rebuild Kube-flux's internal graph using that information.
- Slightly less obvious solution
    - Collect the changes to about the state of the compute nodes and relay this information to Kube-flux which can then use the info to update its internal state without having to rebuild the graph.

Environment

- OS: Linux
- Dependencies:
  - Go: the language used in the development
  - Minikube: used to setup a local Kubernetes cluster
  - Helm: manage Kubernetes application
  - Docker: build images
  - CodeReady Containers (TBD): run local OpenShift clusters on a laptop
- Git Repository
  - Work together with mentors on a GitHub repo
- Local Repository
  - Clone the git repository
  - Test `make local-image`, the process takes about 8 mins

Environment



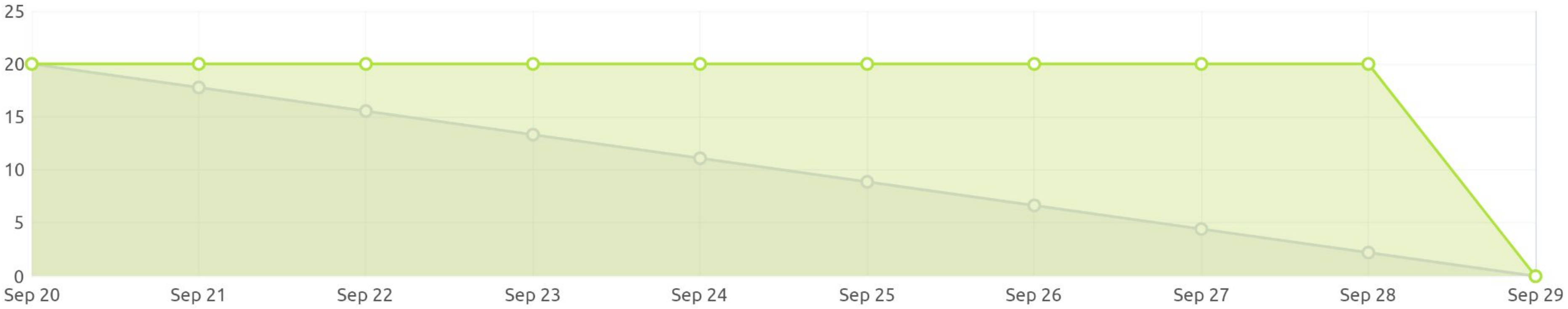
```
Setting up libboost-filesystem-dev:amd64 (1.71.0.0ubuntu4) ...
Setting up libboost-graph1.71-dev:amd64 (1.71.0-6ubuntu9) ...
Setting up libboost-graph-dev:amd64 (1.71.0.0ubuntu4) ...
Setting up libboost-system-dev:amd64 (1.71.0.0ubuntu4) ...
Setting up libczmq-dev:amd64 (4.2.0-2) ...
Processing triggers for libc-bin (2.32-0ubuntu3) ...
/sbin/ldconfig.real: /usr/local/lib/libflux-hostlist.so.1 is not a symbolic link

Reading package lists...
Building dependency tree...
Reading state information...
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Removing intermediate container c95dc9897de0
 ---> 8fb513be335c
Step 48/51 : RUN mkdir -p /home/data/jobspecs/
 ---> Running in 446c31b66bbf
Removing intermediate container 446c31b66bbf
 ---> 16a47477b109
Step 49/51 : COPY flux-k8s/flux-plugin/manifests/kubeflux/sched-config.yaml /home/sched-config.yaml
 ---> 9e8dda23f3d5
Step 50/51 : WORKDIR /bin
 ---> Running in 56415043c551
Removing intermediate container 56415043c551
 ---> 32a9a7054ab8
Step 51/51 : CMD ["kube-scheduler"]
 ---> Running in c69aade6c0a6
Removing intermediate container c69aade6c0a6
 ---> e68758a16cc2
Successfully built e68758a16cc2
Successfully tagged localhost:5000/scheduler-plugins/kube-scheduler:latest
#docker build -f ./build/controller/Dockerfile --build-arg ARCH="amd64" -t localhost:5000/scheduler-plugins/controller:latest .
make local-image  0.61s user 0.65s system 0% cpu 7:51.50 total
➜  scheduler-plugins git:(kubeflux) docker image ls
REPOSITORY                                         TAG       IMAGE ID       CREATED              SIZE
localhost:5000/scheduler-plugins/kube-scheduler   latest    e68758a16cc2   50 seconds ago       790MB
<none>                                             <none>    356b54915f49   About a minute ago   3.22GB
ubuntu                                             20.10     e508bd6d694e   2 months ago         79.4MB
➜  scheduler-plugins git:(kubeflux) █
```

Use cases

- Allow HPC applications that can run on a cloud cluster as well as they can run on an HPC cluster to be scheduled on a cloud cluster.
- Run both HPC and cloud workloads on a unified cluster.

Burndown Chart



20 Points
8 Tasks

Sprint 2

- Setting up OpenShift resources
- Get familiar with Flux’s internal representation of the state of the Kubernetes cluster.
- Devise a testing plan by composing expected correct Flux states to test actual states against.

Thank You