Improving Kubernetes for HPC/AI/ML Workloads

Kevin Hannon
FOSDEM 2024
Who I am

- Senior Software Engineer at Red Hat
- Red Hat Engineer focused on Kubernetes and Openshift
- Developer on Kubernetes (Job, JobSet and contributions to other CNCF batch projects)
Outline of Talk

- Kubernetes
- Survey of existing tools
- Batch working group
- Job API additions
- Queueing
- JobSet

Not Covered :(
- Maximing performance
- Storage
- Networking
- Cloudfu
What is Kubernetes

- Toolkit for building distributed systems
  - What can fail will fail!
- Declarative API for workloads
- Everything starts with the YAML!
  - API first
Lifecycle of a pod

User submits Pod YAML

User

Persist pod to database

apiserver

Update pod object with node that can run pod

etcd

Find node that satisfies pod resources

scheduler

Given a node name, kubelet communicates with CRI/host to run pod

kubelet
Why you shouldn’t use a direct Pod

- Stateless
- No Self healing
  - Failures happen
- API difficult to change
- No queuing!
Open source tools for Batch workloads

- Volcano
- Armada
- Kubeflow
- MCAD
- YuniKorn
# Existing Jobs

<table>
<thead>
<tr>
<th>Job</th>
<th>Project</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlowJob</td>
<td>Kubeflow</td>
<td>Pod + Service</td>
</tr>
<tr>
<td>PyTorchJob</td>
<td>Kubeflow</td>
<td>Pod + Service</td>
</tr>
<tr>
<td>MPIJobV1</td>
<td>Kubeflow</td>
<td>Pod + Service</td>
</tr>
<tr>
<td>MPIJobV2</td>
<td>Kubeflow</td>
<td>IndexedJob + Headless Service (for launcher) and Pod + Service for workers</td>
</tr>
<tr>
<td>TrainingOperator</td>
<td>Kubeflow</td>
<td>Pod + Service</td>
</tr>
<tr>
<td>PodSpec</td>
<td>Armada</td>
<td>Pod + Service + Ingress</td>
</tr>
<tr>
<td>vcJob</td>
<td>Volcano</td>
<td>Pod</td>
</tr>
</tbody>
</table>
Working Group Batch

Scope

Discuss and enhance the support for Batch (e.g., HPC, AI/ML, data analytics, CI) workloads in core Kubernetes. We want to unify the way users deploy batch workloads to improve portability and to simplify supportability for Kubernetes providers.

In scope

- To reduce fragmentation in the k8s batch ecosystem: congregate leads and users from different external and internal projects and user groups (CNCF TAGs, k8s sub-projects focused on batch-related features such as topology-aware scheduling) in the batch ecosystem to gather requirements, validate designs and encourage reutilization of core kubernetes APIs.
- The following recommendations for enhancements:
  - Additions to the batch API group, currently including Job and CronJob resources that benefit batch use cases such as HPC, AI/ML, data analytics and CI.
  - Primitives for job-level queueing, not limited to the k8s Job resource. Long-term, this could include multi-cluster support.
  - Primitives to control and maximize utilization of resources in fixed-size clusters (on-prem) and elastic clusters (cloud).
  - Runtime and scheduling support for specialized hardware (GPUs, NUMA, RDMA, etc.)
Job API

apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  parallelism: 1
  completions: 1
  activeDeadlineSeconds: 1800
  backoffLimit: 6
  template:
    metadata:
      name: pi
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"
        restartPolicy: OnFailure
IndexedJob + Headless service

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'
Suspend

User submits Pod YAML

Persist pod to database

Update pod object with node that can run pod

Find node that satisfies pod resources

Given a node name, kubelet communicates with CRI/host to run pod
High-level Kueue operation

2. Admit Job (based on order, quota etc.)
3. Inject nodeAffinity based on selected flavor and unsuspend the job
4. Create `v1.Pod(s)`
5. Schedule `v1.Pod(s)`
6. Provision more nodes

Batch Admin

`ResourceFlavor(s)`, `ClusterQueue(s)`, `LocalQueue(s)`

Batch User

apiserver

`scheduler`

cluster-autoscaler

kueue

job-controller
Kueue Supported Jobs

- Design requires suspend field or pod support
- Supported jobs:
  - KubeRay
  - Kubeflow operators
  - JobSet
  - Job
  - FluxMiniCluster
JobSet

- 2023 Kubernetes Project for defining a API for multiple jobs
- Support lifecycle operators around DistributedJobs
- Use IndexedJob + HeadlessService
- Failure + Success Policies for replicated jobs
# Distributed training of a ResNet18 model to do image classification

# using the CIFAR-10 dataset and PyTorch.

apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  name: pytorch
spec:
  replicatedJobs:
    - name: workers
      template:
        spec:
          parallelism: 4
          completions: 4
          backoffLimit: 0
          template:
            spec:
              containers:
                - name: pytorch
                  image: gcr.io/k8s-staging-jobset/pytorch-resnet:latest
                  ports:
                    - containerPort: 3389
                  env:
                    - name: MASTER_ADDR
                      value: "pytorch-workers:0-0.pytorch"
                    - name: MASTER_PORT
                      value: "3389"
                  command:
                    - bash
                    - -xc
                    - torchrun --nproc_per_node=1 --master_addr=$MASTER_ADDR --master_port=$MASTER_PORT resnet.py --backend=gloo --num_epochs=2
Leader + Workers

```
apiVersion: jobset.x-k8s.io/v1alpha1
kind: JobSet
metadata:
  name: success-policy
spec:
  # We want to declare our JobSet successful if workers finish.
  # If workers finish we should clean up the remaining replicatedJobs.
  successPolicy:
    operator: All
    targetReplicatedJobs:
    - workers
  replicatedJobs:
  - name: leader
    replicas: 1
    template:
      spec:
        # Set backoff limit to 0 so job will immediately fail if any pod fails.
        backoffLimit: 0
        completions: 1
        parallelism: 1
        template:
          spec:
            containers:
            - name: leader
              image: bash:latest
              command:
                - bash
                - -xc
                - |
                - sleep 10000
  - name: workers
    replicas: 1
    template:
      spec:
        backoffLimit: 0
        completions: 2
        parallelism: 2
        template:
          spec:
            containers:
            - name: worker
              image: bash:latest
              command:
                - bash
                - -xc
                - |
                - sleep 10
```
Demo!
To install JobSet:

```
VERSION=v0.3.1
kubect1 apply --server-side -f https://github.com/kubernetes-sigs/jobset/releases/download/$VERSION/manifests.yaml
```

To install Kueue:

```
VERSION=v0.5.2
kubect1 apply --server-side -f https://github.com/kubernetes-sigs/kueue/releases/download/$VERSION/manifests.yaml
```
Thank you!