Improving Batch Support in Open Source Kubernetes

Abdullah Gharaibeh
Our Team

- Part of the Google Kubernetes Engine (GKE)
- Focused mostly on Open Source (OSS) work in both sig-scheduling* and sig-apps
- Our goal is to improve the batch experience in Kubernetes

* SIG: Special Interest Group that operate under an open governance model
Background: Kubernetes Architecture

- **Master**
  - etcd
  - apiserver
  - controller-manager (Job, Deployment, ..)
  - scheduler

- **Node**
  - kubelet
  - pod

Create Job, Deployment, Service ...
Background: Kubernetes Job vs HTCondor ClassAds

A similar key/value abstraction of resources and requests

```
apiVersion: batch/v1
kind: Job
metadata:
  name: simple
spec:
  template:
    spec:
      containers:
        - name: simple
          image: debian:buster
          command: ["/bin/true"]
          resources:
            requests:
              memory: "64Mi"
              cpu: "250m"
            limits:
              memory: "128Mi"
              cpu: "500m"
          nodeSelector:
            node.kubernetes.io/instance-type: e2-micro
          backoffLimit: 4
          parallelism: 10
          completions: 100
```

- Arguments = ""
- Cmd = "/bin/true"
- RequestCpus = 1
- RequestGPUs = 0
- RequestMemory = 64
- Requirements = (TARGET.InstanceType =?= "e2-micro") && ...

Requests "shares" for ¼ CPU both in cgroups scheduling sense and in the accounting of available resources for matchmaking.

Tightly pack containers for maximum hardware utilization!

Assign cgroups quota for ½ CPU cycles per 100msec period

```
nodeSelector requires nodes with labels;
nodeAffinity expresses more complex selection, including ranking
```
Background: Kubernetes Core OSS Job Model

1. create v1.Job
2. Create v1.Pods
3. Assign pods to nodes
4. Notify of new pods
5. Start pods
6. Update pod status
7. Update job status
The Problem

- Support for batch lagged in core OSS Kubernetes
  - Most core kubernetes components are pod centric, not quite compatible with batch workloads

- A fragmented kubernetes batch ecosystem
  - Volcano, kubeflow, Spark operator each have their own job APIs and semantics
High-level Approach

- Enhance the core OSS Kubernetes Job model, such that it can be used
  - directly for simple workloads
  - as a core resource for more advanced orchestrators or workloads (Queueing, MPI, ML etc.)
Enhancing the Job Model

● Support indexed Jobs semantics with variable parallelism
  ○ **Problem:** Running large scale parallel jobs required setting up a message queue

● Robust tracking of Job completion status
  ○ **Problem:** All pod objects of a job must continue to exist in etcd until the job completes

● Automatic Job objects cleanup
  ○ **Problem:** Completed jobs continue to exist in etcd unless explicitly removed, hence impacting etcd performance
Shallow Dive on parallel execution for Jobs

with a work queue

RabbitMQ

with Indexed Job

Job

apiVersion: batch/v1
kind: Job
spec:
  parallelism: 3
  completions: 6
  completionMode: Indexed

item 0
item 1
item 2
item 3
item 4
item 5

${INDEX} = 0
${INDEX} = 1
${INDEX} = 2
${INDEX} = 3
${INDEX} = 4
${INDEX} = 5
Quickly producing similar tasks in HTCondor

To produce a *cluster* of similar processes we can iterate in `condor_submit`:

```
Universe = Vanilla
Executable = cook
Output = meal$(Process).out
Args = -i $(Item)
Queue Item in (pasta, chicken)
```

Thanks, T.J.!
Unifying the ecosystem around the k8s Job Model

We proposed hooks to allow higher level job schedulers to

- manage a k8s Job as a single entity
  - Added job suspend/resume semantics, job schedulers can use it to control when the job should start and whole job preemption

- define scheduling and provisioning properties
  - Proposed “JobClass” API to uniformly define job attributes

- define job grouping
  - Proposed a “JobGroup” API to uniformly define groups of related jobs (e.g., actors and learners in reinforcement learning)

- control Job scale-down behavior
  - Introduced “pod-deletion-cost” API to inform controllers which pods to delete first on scale down
  - Very similar to HTCondor preemption/defrag ranking. Cost can be proportional to "badput" estimate or literal financial cost of cloud hardware.
Hooking a job-level manager

1. Create v1.Job in suspended state
2. Unssuspend the job
3. Create v1.Pods
4. Assign pods to nodes
5. Notify of new pods
6. Start pods
7. Update pod status
8. Update job status
Summary

- The Kubernetes community is investing more in Batch
- Our goal is to help unify the k8s batch ecosystem around the core API model
  - Makes the user experience more uniform, while allowing vendors to offer differentiation
Questions
HTCondor on Google Cloud

- Provision your own [HTCondor cluster using the Google Cloud Marketplace](https://console.cloud.google.com/marketplace/details/htcondor/htcondor?region=us-central1)!  
  - Google Cloud Storage is S3-interoperable with custom endpoint specification!  
    - Working with ToddM to allow use of Cloud Storage gs:// URIs and knob naming that access keys are for S3/HMAC generally (AWS, Google Cloud, IBM Cloud, Ceph, MinIO, etc.).

```ini
aws_access_key_id_file = key_id_filename
aws_secret_access_key_file = secret_access_key_filename
transfer_input_files = s3://storage.googleapis.com/test-adaf/infile
transfer_output_remaps = "output.dat = s3://storage.googleapis.com/test-adaf/outfile"
```

- Several key customers in research and private sector using HTCondor  
- Working with CHTC team to adopt / improve 9.x security practices in the cloud