Kubernetes Service Discovery

Service Discovery

- for large scale deep learning we need multiple processes that talk to each other
- this requires
  - service discovery
  - networking
  - name resolution

K8s Service Discovery

Simple:

- every pod gets assigned a hostname and domain
- you can simply connect directly to these well-known names

Requirements:

- create a "headless service" to start the name resolver
- add ports, host name, and subdomain to your pods

Headless Service

The `clusterIP: None` makes it headless. (Other services are load balancing, which we don't want.)

In [28]:

```
kubectl apply -f - <<'EOF'
apiVersion: v1
kind: Service
metadata:
  name: bigdata19
spec:
  clusterIP: None
  ports:
    - port: 7880
      targetPort: 7880
  selector:
    app: bigdata19
EOF
```

```
service/bigdata19 unchanged
```

A Visible Pod

This pod will be assigned the DNS name `shards.bigdata19` .

In [30]:

```
# nodes get assigned DNS names if they have a port and the app label matches the headless service

kubectl delete pod/shards || true
kubectl apply -f - <<'EOF'
apiVersion: v1
kind: Pod
metadata:
  name: shards
  labels:
    app: bigdata19
spec:
  containers:
  - name: shards
    image: gcr.io/research-191823/bigdata19
    command: ["serve-imagenet-shards", "-b", "96", "zpub://0.0.0.0:7880"]
    ports:
      - containerPort: 7880
  restartPolicy: Never
  hostname: shards
  subdomain: bigdata19
EOF
```

```
Error from server (NotFound): pods "shards" not found
pod/shards created
```

In [ ]:

```
sleep 15
```

DNS Debugging

In [32]:

```
# make sure resolution is working
kubectl exec -ti shards -- nslookup shards.bigdata19

# check resolv.conf file
kubectl exec -ti shards -- cat /etc/resolv.conf
```

```
Server:         10.64.0.10
Address:        10.64.0.10#53

Name:   shards.bigdata19.default.svc.cluster.local
Address: 10.0.1.149

nameserver 10.64.0.10
search default.svc.cluster.local svc.cluster.local cluster.local c.research-191823.internal google.i
nternal
options ndots:5
```

In [33]:

```
# check service running
kubectl get svc --namespace=kube-system

# check endpoints
kubectl get ep kube-dns --namespace=kube-system
```

```
NAME                   TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)         AGE
default-http-backend   NodePort    10.64.0.106    <none>        80:31530/TCP    11h
heapster               ClusterIP   10.64.5.246    <none>        80/TCP          11h
kube-dns               ClusterIP   10.64.0.10     <none>        53/UDP,53/TCP   11h
metrics-server         ClusterIP   10.64.15.252   <none>        443/TCP         11h
NAME       ENDPOINTS                                               AGE
kube-dns   10.0.0.130:53,10.0.2.130:53,10.0.0.130:53 + 1 more...   11h
```

In [34]:

```
# when desperate, you can look through the kube-dns logs
kubectl get pods --namespace=kube-system -l k8s-app=kube-dns
kubectl get pods --namespace=kube-system -l k8s-app=kube-dns -o name | sed 1q |
while read pod; do
kubectl logs --tail=3 --namespace=kube-system $pod -c kubedns
kubectl logs --tail=3 --namespace=kube-system $pod -c dnsmasq
kubectl logs --tail=3 --namespace=kube-system $pod -c sidecar
kubectl logs --tail=3 --namespace=kube-system $pod -c prometheus-to-sd
done
```

```
NAME                        READY   STATUS    RESTARTS   AGE
kube-dns-79868f54c5-h9m5c   4/4     Running   0          11h
kube-dns-79868f54c5-kh556   4/4     Running   0          11h
E1209 05:11:04.345848       1 reflector.go:201] k8s.io/dns/pkg/dns/dns.go:192: Failed to list *v1.Se
rvice: Get https://10.64.0.1:443/api/v1/services?resourceVersion=0: net/http: TLS handshake timeout
E1209 05:11:04.346614       1 reflector.go:201] k8s.io/dns/pkg/dns/dns.go:189: Failed to list *v1.En
dpoints: Get https://10.64.0.1:443/api/v1/endpoints?resourceVersion=0: net/http: TLS handshake timeo
ut
I1209 07:01:00.234994       1 dns.go:601] Could not find endpoints for service "bigdata19" in namesp
ace "default". DNS records will be created once endpoints show up.
I1209 05:04:01.718338       1 nanny.go:146] dnsmasq[23]: using nameserver 127.0.0.1#10053 for domain
cluster.local
I1209 05:04:01.718345       1 nanny.go:146] dnsmasq[23]: using nameserver 169.254.169.254#53
I1209 05:04:01.718354       1 nanny.go:146] dnsmasq[23]: read /etc/hosts - 7 addresses
I1209 05:04:02.521706       1 server.go:45] Starting server (options {DnsMasqPort:53 DnsMasqAddr:127
.0.0.1 DnsMasqPollIntervalMs:5000 Probes:[{Label:kubedns Server:127.0.0.1:10053 Name:kubernetes.defa
ult.svc.cluster.local. Interval:5s Type:33} {Label:dnsmasq Server:127.0.0.1:53 Name:kubernetes.defau
lt.svc.cluster.local. Interval:5s Type:33}] PrometheusAddr:0.0.0.0 PrometheusPort:10054 PrometheusPa
th:/metrics PrometheusNamespace:kubedns})
I1209 05:04:02.521760       1 dnsprobe.go:75] Starting dnsProbe {Label:kubedns Server:127.0.0.1:1005
3 Name:kubernetes.default.svc.cluster.local. Interval:5s Type:33}
I1209 05:04:02.521898       1 dnsprobe.go:75] Starting dnsProbe {Label:dnsmasq Server:127.0.0.1:53 N
ame:kubernetes.default.svc.cluster.local. Interval:5s Type:33}
I1209 05:04:03.749028       1 main.go:124] Taking source configs from kubernetes api server
I1209 05:04:03.749033       1 main.go:85] Built the following source configs: [{kubedns localhost 10
054 /metrics [probe_kubedns_latency_ms probe_kubedns_errors dnsmasq_misses dnsmasq_hits] 0xc4203e40a
0}]
I1209 05:04:03.749061       1 main.go:133] Running prometheus-to-sd, monitored target is kubedns loc
alhost:10054
```

In [35]:

```
sleep 15
```

Logs of the Running Server

The server is chugging along nicely, sending out training batches to anybody who will listen.

In [36]:

```
kubectl logs shards | sed 10q
```

```
serving zpub://0.0.0.0:7880
0 rate 0.000000 msg/s throughput 0.00e+00 bytes/s
10 rate 5.538139 msg/s throughput 8.00e+07 bytes/s
20 rate 5.260502 msg/s throughput 7.60e+07 bytes/s
30 rate 5.167468 msg/s throughput 7.47e+07 bytes/s
40 rate 5.068958 msg/s throughput 7.33e+07 bytes/s
50 rate 5.030015 msg/s throughput 7.27e+07 bytes/s
60 rate 4.998792 msg/s throughput 7.22e+07 bytes/s
70 rate 5.002644 msg/s throughput 7.23e+07 bytes/s
80 rate 4.998627 msg/s throughput 7.22e+07 bytes/s
```

In [37]:

```
kubectl get pods
```

```
NAME     READY   STATUS    RESTARTS   AGE
shards   1/1     Running   0          4m32s
```

Starting a Client

Here is a small network client that listens to training data and outputs statistics.

In [38]:

```
kubectl delete pod/client || true
kubectl apply -f - <<'EOF'
apiVersion: v1
kind: Pod
metadata:
  name: client
  labels:
    app: bigdata19
spec:
  containers:
  - name: client
    image: gcr.io/research-191823/bigdata19
    command: ["tensormon", "zsub://shards.bigdata19:7880"]
    stdin: true
    tty: true
  restartPolicy: Never
  hostname: client
  subdomain: bigdata19
EOF
```

```
Error from server (NotFound): pods "client" not found
pod/client created
```

In [39]:

```
sleep 15
```

Client Output

In [40]:

```
kubectl logs client | sed 10q
```

```
input: ['zsub://shards.bigdata19:7880']
zsub://shards.bigdata19:7880
connected
                  10    1.818 batches/s  174.525 samples/s (batchsize: 96)
                  20    5.192 batches/s  498.429 samples/s (batchsize: 96)
                  30    5.199 batches/s  499.126 samples/s (batchsize: 96)
                  40    4.700 batches/s  451.197 samples/s (batchsize: 96)
                  50    5.196 batches/s  498.807 samples/s (batchsize: 96)
```

Starting a DL Client on a GPU Node

In [41]:

```
kubectl delete job/myjob || true
kubectl apply -f - <<'EOF'
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
  labels:
    app: bigdata19
spec:
  backoffLimit: 0
  template:
    spec:
      containers:
        - name: myjob
          image: gcr.io/research-191823/bigdata19
          command:
            - "/bin/bash"
            - "-c"
            - |
              cp /files/*.py .
              python3 training.py --tensorcom zsub://shards.bigdata19:7880
          stdin: true
          tty: true
          resources:
            limits:
              nvidia.com/gpu: "1"
          volumeMounts:
            - mountPath: /files
              name: files
      nodeSelector:
        cloud.google.com/gke-accelerator: nvidia-tesla-t4
      restartPolicy: Never
      volumes:
        - configMap:
            name: files
          name: files
EOF
```

```
Error from server (NotFound): jobs.batch "myjob" not found
job.batch/myjob created
```

In [42]:

```
sleep 10
```

In [43]:

```
kubectl logs job/myjob
```

```
/opt/conda/lib/python3.6/site-packages/torchvision/io/_video_opt.py:17: UserWarning: video reader ba
sed on ffmpeg c++ ops not available
  warnings.warn("video reader based on ffmpeg c++ ops not available")
Mon Dec  9 16:40:23 UTC 2019; myjob-q2d62; root; /workspace; GPU 0: Tesla T4 (UUID: GPU-fd29201b-d66
3-6697-b413-a761dceb23c8);
creating resnet50
        0 bs    96 per sample loss 7.45e-02 loading 6.03e-04 training 1.85e-02
```

In [44]:

```
sleep 60
```

Training

- Note that with distributed preprocessing, loading is very fast.
- We will talk about the Tensorcom package late.

In [45]:

```
kubectl logs job/myjob
```

```
/opt/conda/lib/python3.6/site-packages/torchvision/io/_video_opt.py:17: UserWarning: video reader ba
sed on ffmpeg c++ ops not available
  warnings.warn("video reader based on ffmpeg c++ ops not available")
Mon Dec  9 16:40:23 UTC 2019; myjob-q2d62; root; /workspace; GPU 0: Tesla T4 (UUID: GPU-fd29201b-d66
3-6697-b413-a761dceb23c8);
creating resnet50
        0 bs    96 per sample loss 7.45e-02 loading 6.03e-04 training 1.85e-02
     1152 bs    96 per sample loss 7.38e-02 loading 5.74e-04 training 7.39e-03
     2304 bs    96 per sample loss 7.34e-02 loading 5.62e-04 training 4.28e-03
     3456 bs    96 per sample loss 7.31e-02 loading 5.57e-04 training 3.42e-03
     4608 bs    96 per sample loss 7.29e-02 loading 5.56e-04 training 3.18e-03
     5760 bs    96 per sample loss 7.27e-02 loading 5.58e-04 training 3.13e-03
     6816 bs    96 per sample loss 7.26e-02 loading 5.56e-04 training 3.12e-03
```

In [46]:

```
kubectl get jobs
```

```
NAME    COMPLETIONS   DURATION   AGE
myjob   0/1           72s        72s
```

In [47]:

```
kubectl delete jobs --all
kubectl delete pods --all
```

```
job.batch "myjob" deleted
pod "client" deleted
pod "myjob-q2d62" deleted
pod "shards" deleted
```

Kubernetes Service Discovery

- it's like creating new server out of thin air
- you can define your distributed application as a collection of pods
- K8s also provides load balancing and more complex name spaces