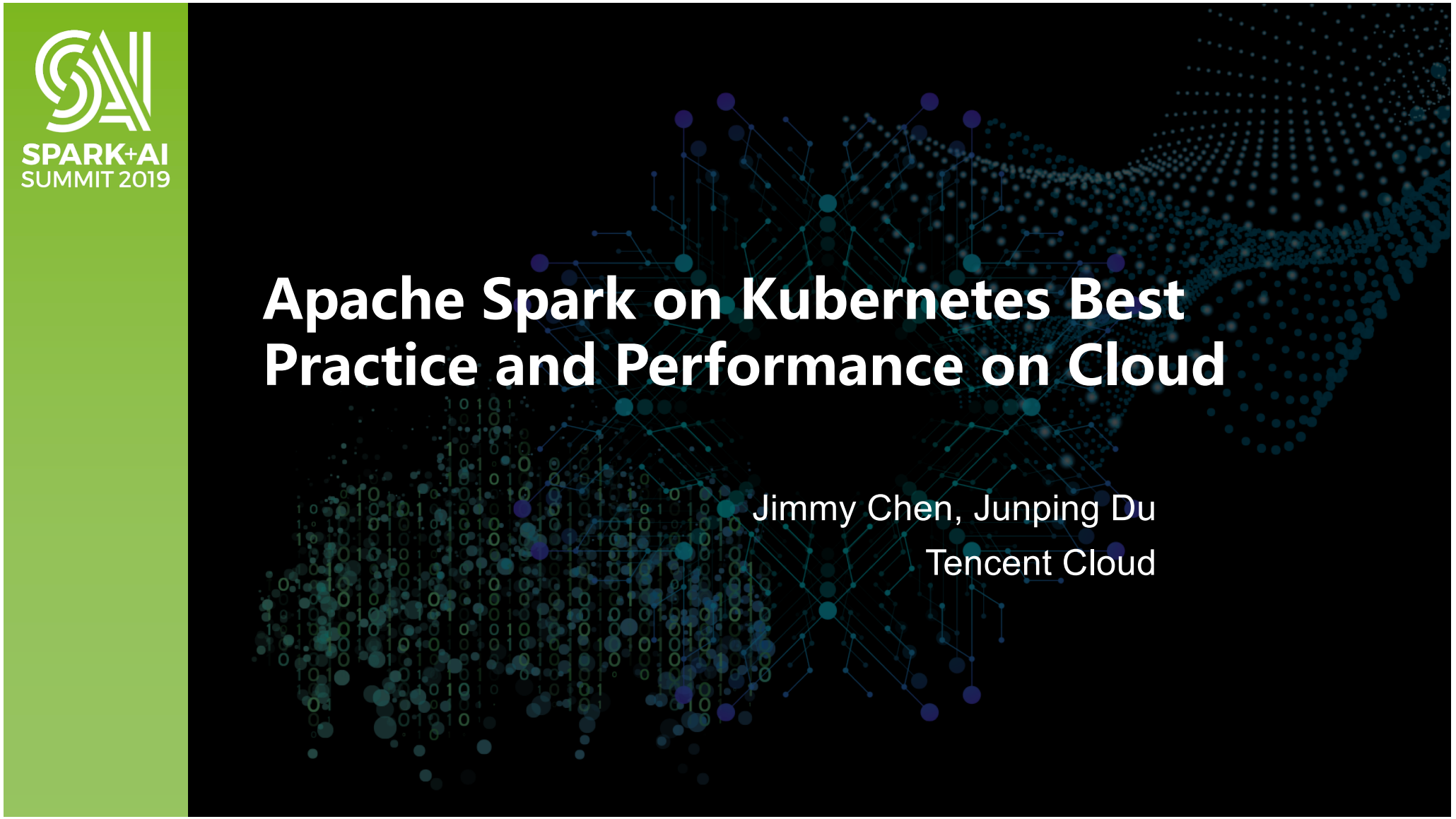


Apache Spark on Kubernetes Best Practice and Performance on Cloud

Jimmy Chen, Junping Du

Tencent Cloud

Agenda

- Spark service in the cloud – our practice
- Spark on Kubernetes – architecture, best practice and performance

About us

Jimmy Chen

Sr. Software Engineer at Tencent Cloud

Apache Hadoop, Spark, Parquet Contributor

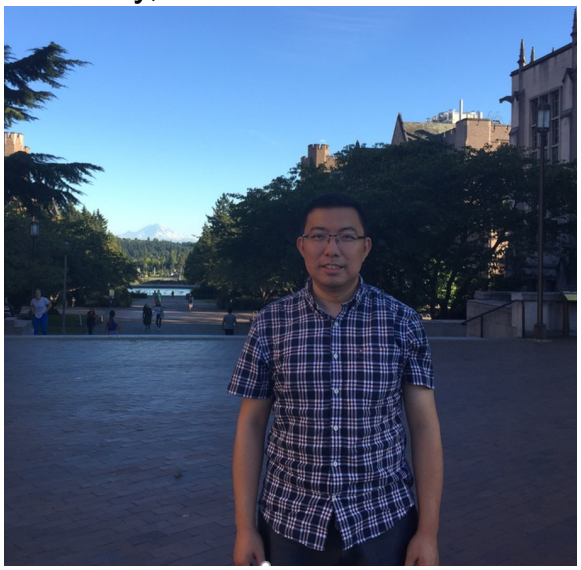
Previously, worked at Intel

Junping Du

Principal Engineer & Director at Tencent Cloud

Apache Hadoop PMC & ASF Member

Previously, worked at Hortonwork and VMware



About Tencent and Tencent Cloud

THE APACHE SOFTWARE FOUNDATION
PLATINUM SPONSORS

500

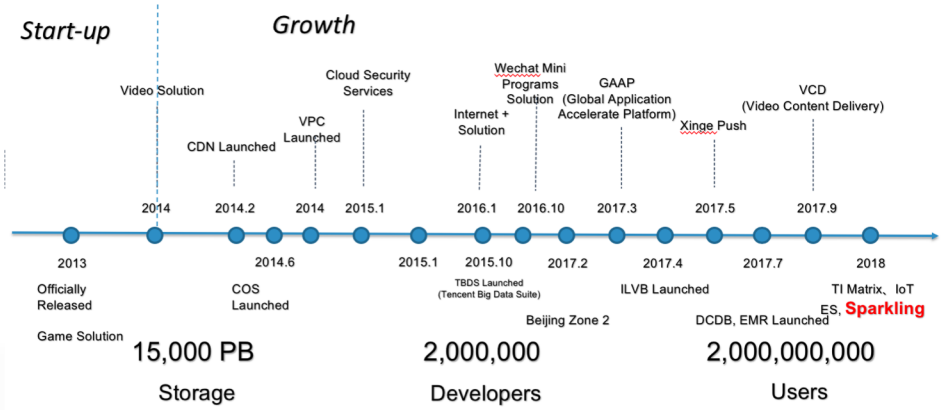
The 331th largest company in the Fortune in 2018

BRANDZ

Top 100 Most Valuable Global Brands 2018 ranked 1st in China, ranked 5th global

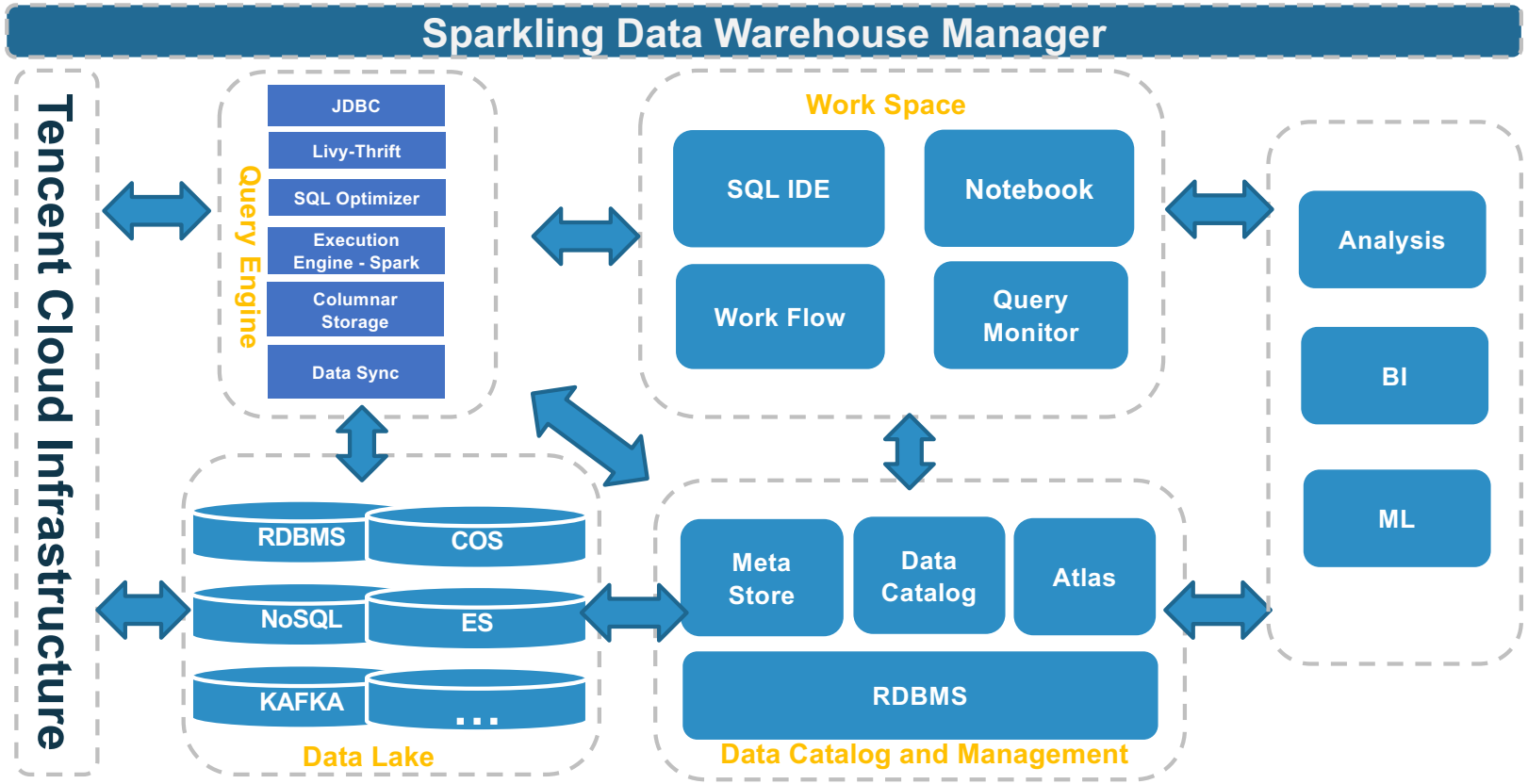
BCG
The Boston Consulting Group

World's 50 Most Innovative Companies 2015 ranked 1st in China



Case study for Spark in the Cloud - Sparkling

Sparkling Architecture



Motivations towards dockerized Spark

- Serverless
  - Better isolation

- On-premises deployment
  - Bare metal infrastructure, control protocol not work
  - DevOps scope
  - Cost

Our Practice

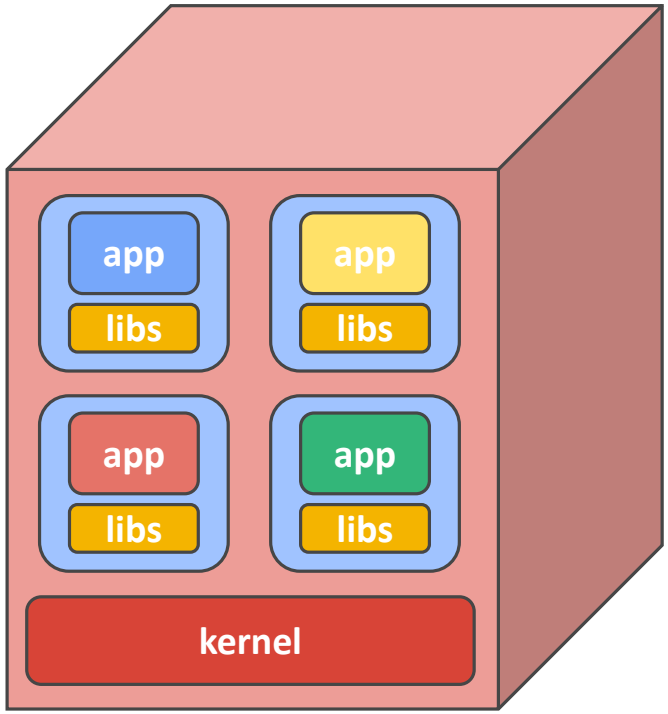
- Sparkling on Kubernetes
  - Cloud API server -> Kubernetes API server
  - CVM resource -> container resource
  - Cloud load balancer -> Kubernetes load balancer
  - Cloud DNS -> Kubernetes ingress/coredns
  - CVM image -> docker image
  - etc.

Kubernetes

New open-source cluster manager.

- github.com/kubernetes/kubernetes

Runs programs in Linux containers.



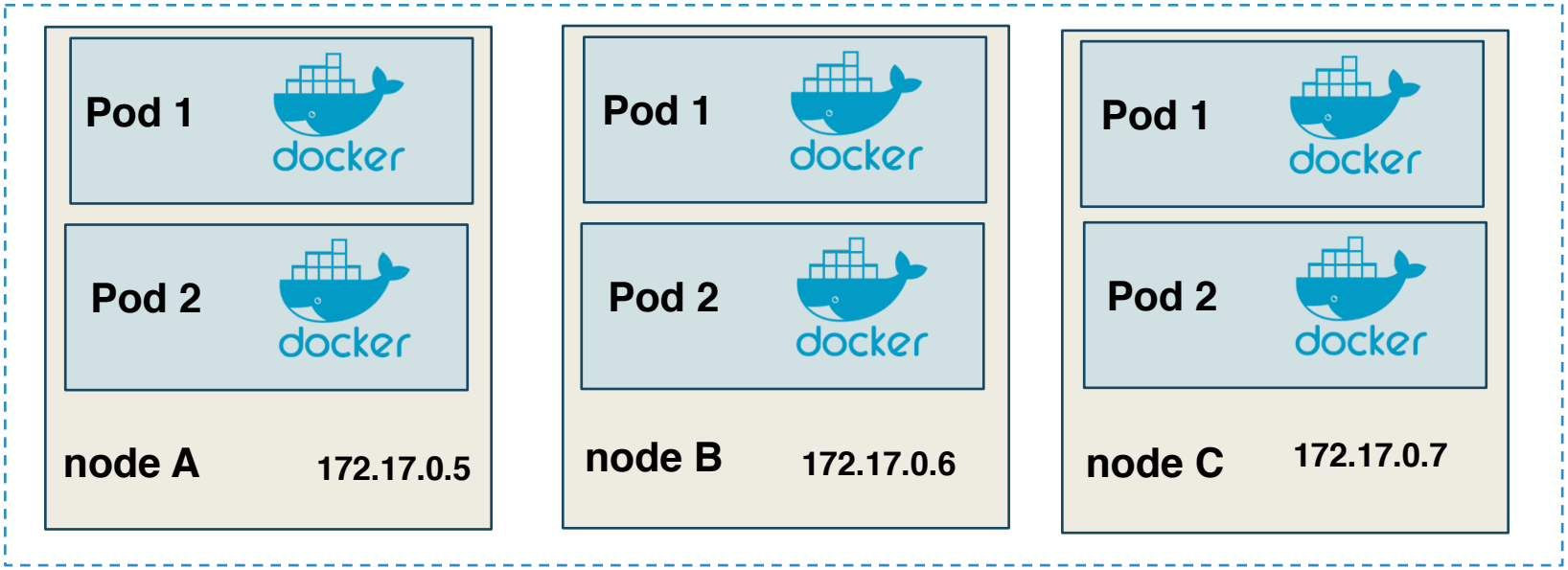
2,000+ contributors and 75,000+ commits.

docker

Kubernetes Architecture

Pod, a unit of scheduling and isolation.

- runs a user program in a primary container
- holds isolation layers like a virtual IP in an infra container

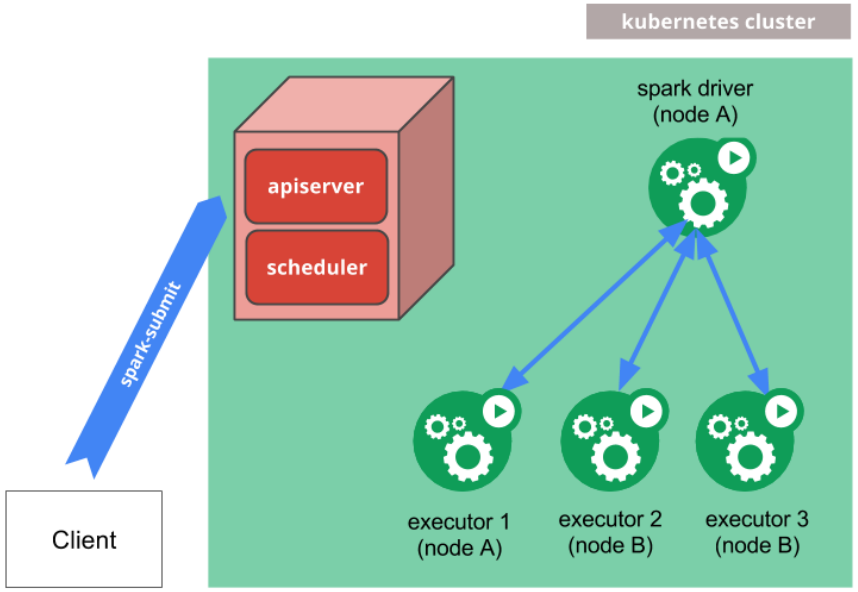


Kubernetes Advantages

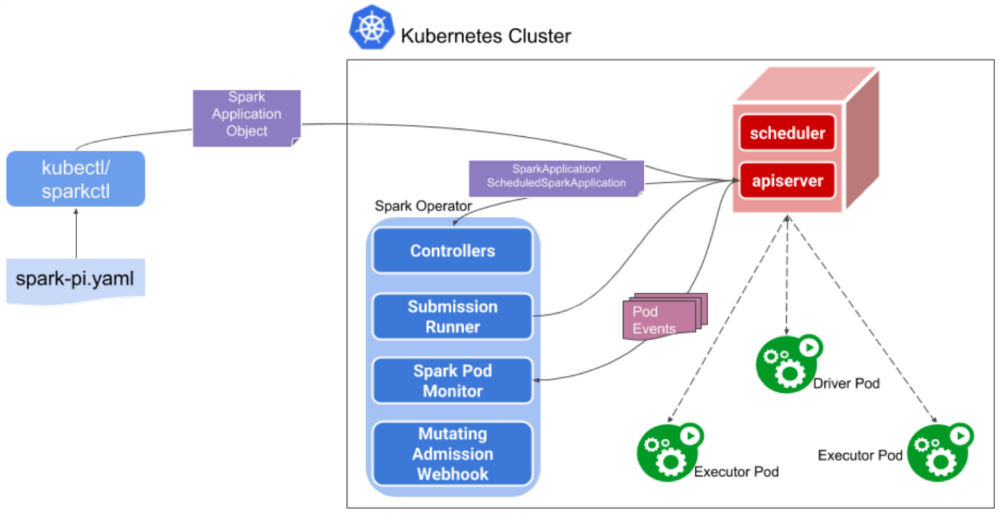
- Portability inherit from Docker, better isolation for performance and security
- Out of box management support, namespace, RBAC, authentication, logging, monitoring, etc..
- Heterogeneous Deployment
- **Ecosystem and large OSS community**

Spark On Kubernetes Workflow

spark-submit



Kubernetes spark operator



Spark on Kubernetes Status

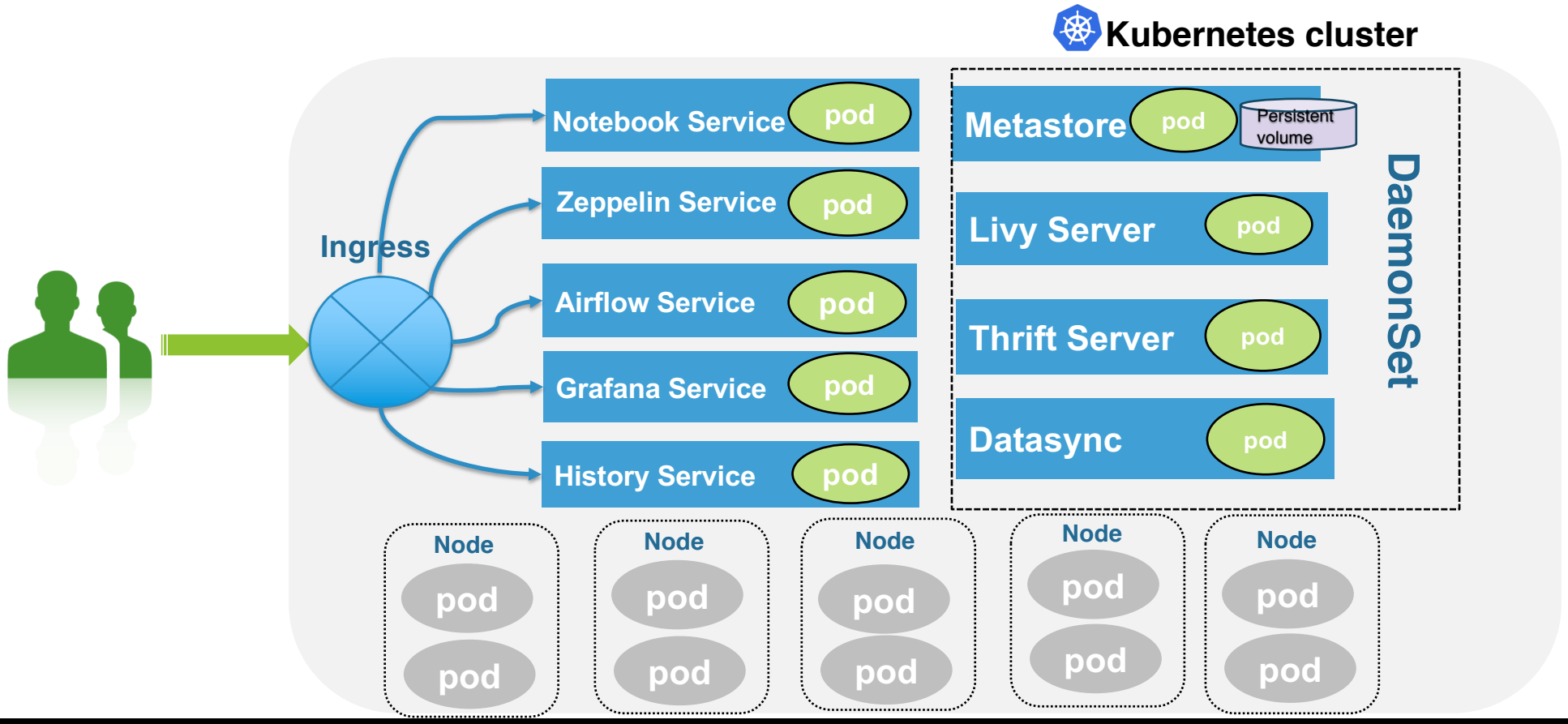
- Spark side

  - Cluster/Client mode
  - Volume mounts
  - Multiple-language support, such as: Python, R
  - Static executor allocation
- On-going Work
  - Dynamic Executor Allocation
  - Job Queues and Resource Management
  - Spark Application Management

- Kubernetes side

  - Support spark-ctl CLI
  - Native cron support for scheduled app
  - Customized Spark pod, e.g. mounting configMap and volume, setting affinity/anti-affinity
  - Monitoring through Prometheus
- On-going Work
  - Support GPU resource scheduler
  - Spark operator HA
  - Multiple spark version support
  - Priority queues

Sparkling on Kubernetes Architecture



Submit via spark-submit

Step 1: check cluster master

```
#kubectl cluster-info
Kubernetes master is running at https://169.254.128.15:60002
```

Step 2: submit application via spark-submit

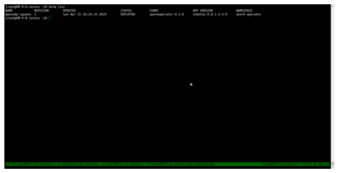
```
#bin/spark-submit \
--master k8s://https://169.254.128.7:60002 \
--deploy-mode cluster \
--name spark-pi \
--class org.apache.spark.examples.SparkPi \
--conf spark.kubernetes.container.image=sherrytima/spark2.4.1:k8s \
local:///opt/spark/examples/jars/spark-examples_2.11-2.4.1.jar
```

Submit via kubectl/sparkctl

- Install spark operator on your kubernetes cluster

```
# helm repo add incubator http://storage.googleapis.com/kubernetes-charts-incubator
# helm install incubator/sparkoperator --namespace spark-operator
```

- Declare application yaml
- Run the application
  - kubectl create -f spark-pi.yaml
  - spark-ctl create spark-pi.yaml



Benchmark Environment

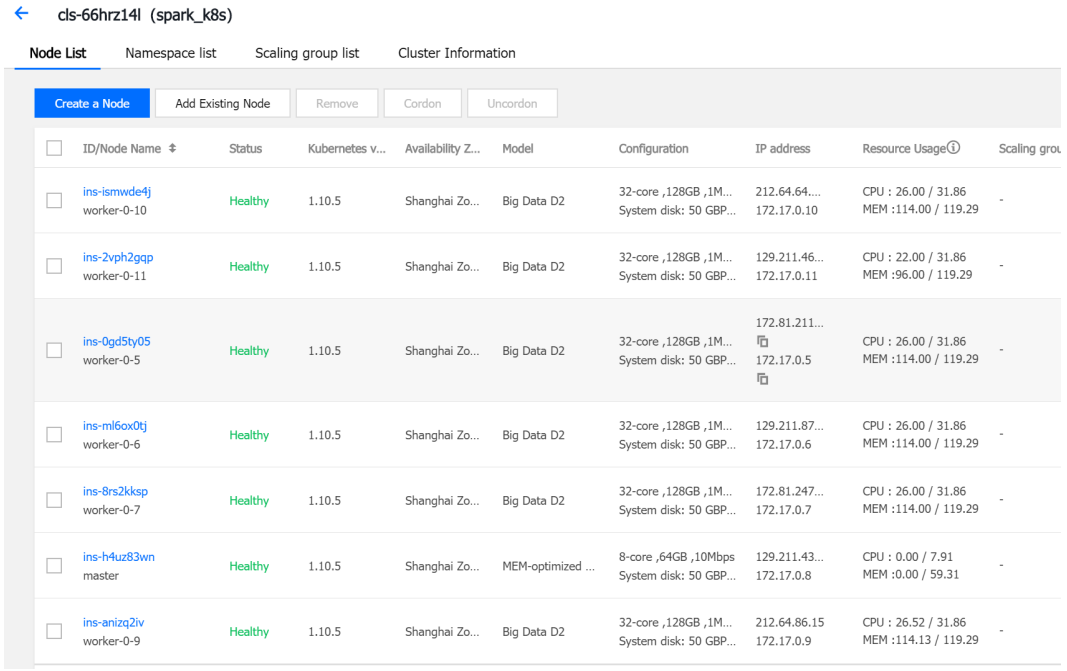
- Software

Kubernetes: 1.10.1
Hadoop/Yarn: 2.7.3
Spark: 2.4.1
OS: Centos-7.2

- Hardware

Master
  - CPU/mem: 8 core + 64G mem
  - Disk: 200G SSD
  - NUM: 1

Slaves
  - CPU/mem:32 core + 128G mem
  - Disk: 4 HDD
  - NUM: 6



Terasort

- Data Size 100G

spark.executor.instance 5

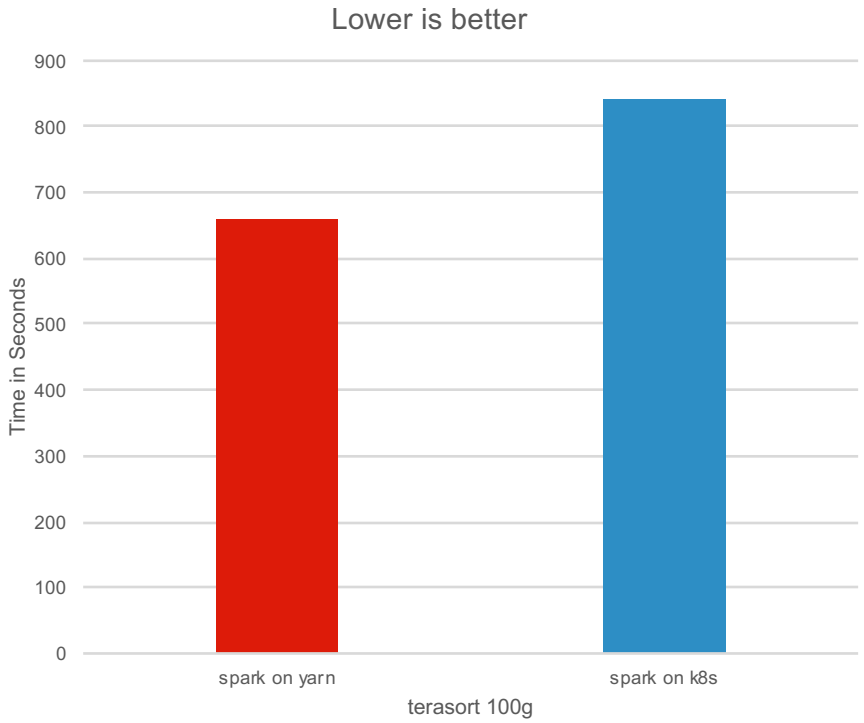
spark.executor.cores 2

spark.executor.memory 10g

spark.executor.memoryoverhead 2g

spark.driver memory 8g

- Data Size 500G
  - Spark on kubernetes **failed**
  - Spark on Yarn (~35 minutes)



Root Cause Analysis

- Pod always be evicted due to disk pressure, the spark.local.dir is mounted as emptyDir.

```
19/04/14 03:01:00 ERROR TaskSchedulerImpl: Lost executor 8 on 172.16.5.186:
The executor with id 8 exited with exit code -1.
The API gave the following brief reason: Evicted
The API gave the following message: The node was low on resource: nodefs.
The API gave the following container statuses:
```

- emptyDir is ephemeral storage on node

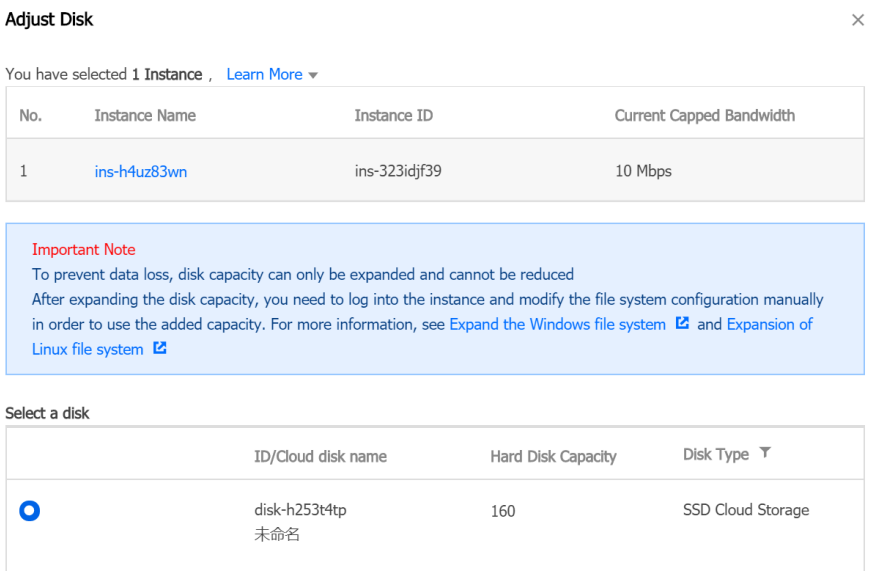
```
{
    "Type": "bind",
    "Source": "/var/lib/kubelet/pods/1efd990a-61a0-11e9-b886-7a113075536f/volumes/kubernetes.io~empty-dir/spark-local-dir-1",
    "Destination": "/tmp",
    "Mode": "",
    "RW": true,
    "Propagation": "rprivate"
}
```

Possible Solutions

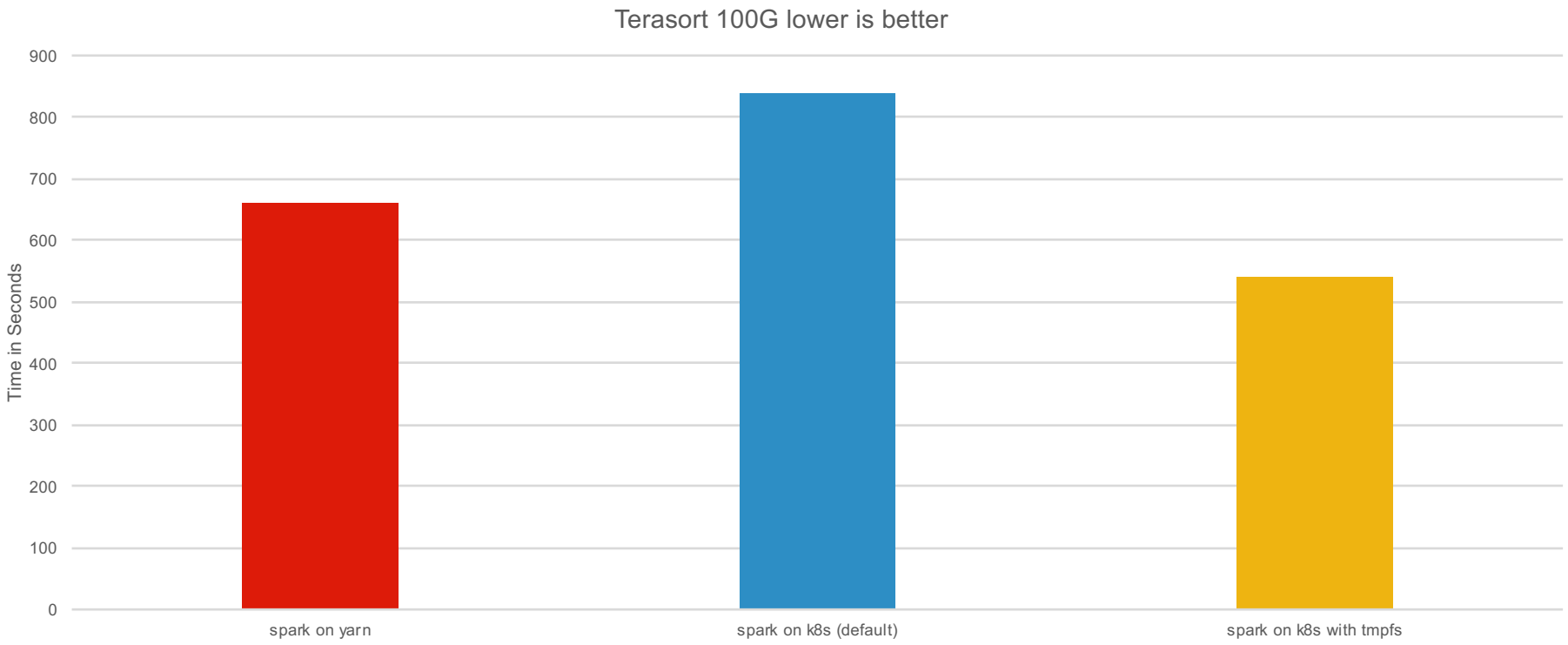
- mount emptyDir with RAM backed volumes.

```
spark.kubernetes.local.dirs.tmpfs=
true
```

- Extend disk capacity of kubelet workspace



Result


Terasort 100G lower is better
Time in Seconds
900
800
700
600
500
400
300
200
100
0
spark on yarn
spark on k8s (default)
spark on k8s with tmpfs

Spark-sql-perf

- Use spark-sql-perf repo from Databricks
  - 10 shuffle bound queries
  - spark application base on spark-sql-perf
- Run in cluster mode for Kubernetes

```
bin/spark-submit  --name tpcds-1000
        --jars hdfs://172.17.0.8:32001/run-lib/*
        --class perf.tpcds.TpcDsPerf
        --conf spark.kubernetes.container.image=sherrytima/spark-2.4.1:k8sv8
        --conf spark.driver.args="hdfs://172.17.0.8:32001/run-tpcds 1000 orc true"
        hdfs://172.17.0.8:32001/tpcds/tpcds_2.11-1.1.jar
```

Benchmark Result

- Configuration

spark.executor.instance 24

spark.executor.cores 6

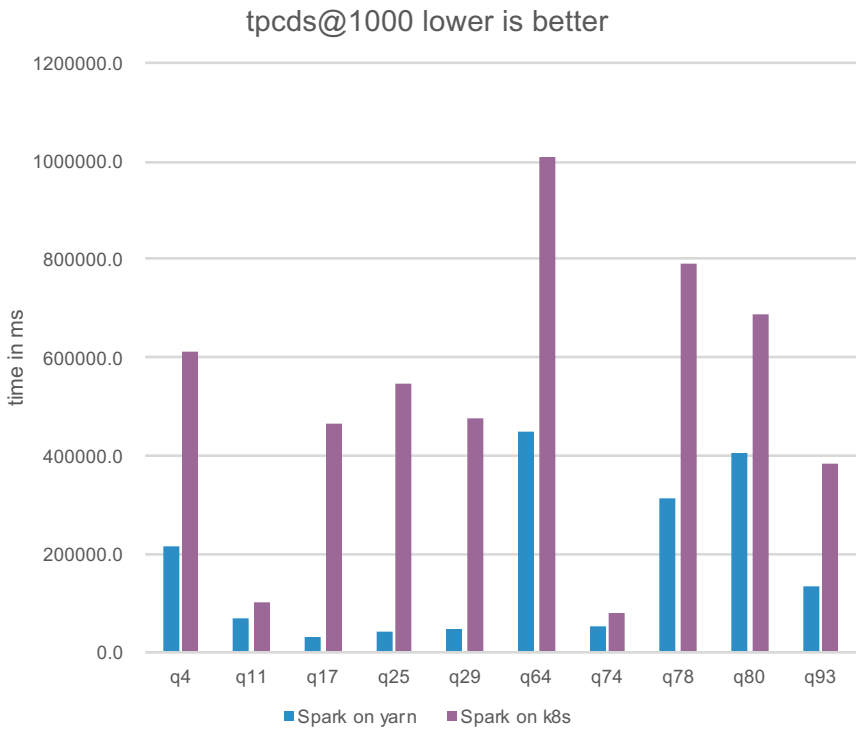
spark.executor.memory 24g

spark.executor.memoryoverhead 4g

spark.driver memory 8g

- Result

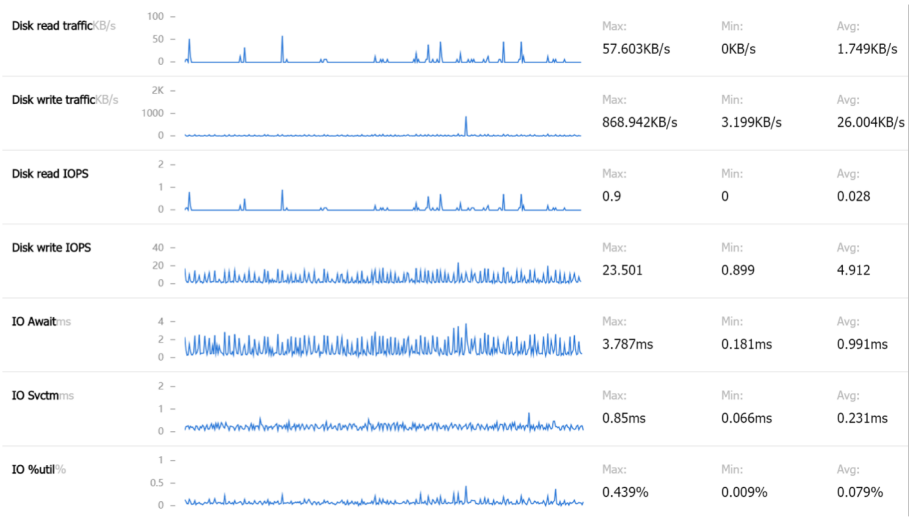
Spark on kubernetes is much slower than spark on yarn!!!



Root Cause

- The kubelet work directory can only be mounted on one disk, so that the spark scratch space only use ONE disk.

- While running in yarn mode, the spark scratch space use as many disk as yarn.local.dir configures.



Possible workaround

- RAM backed media as spark scratch space

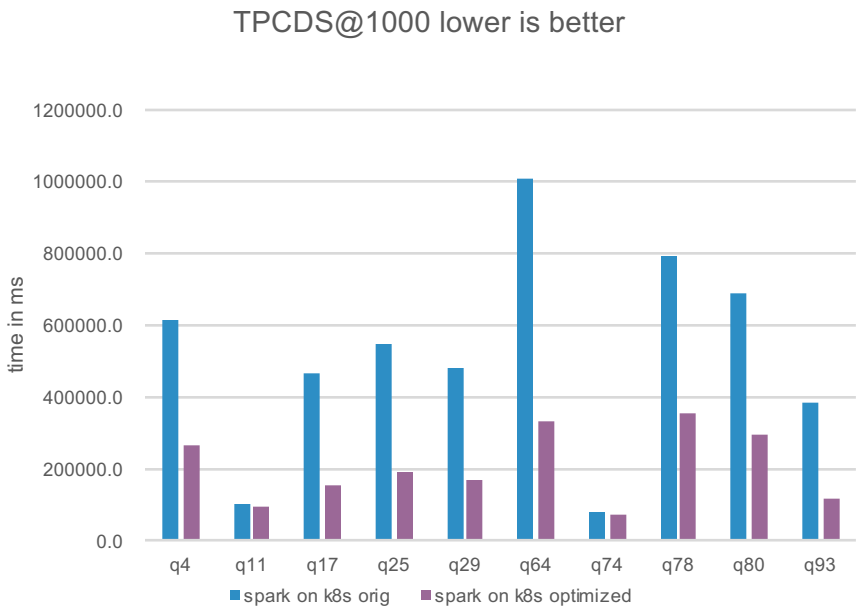
```
19/04/15 09:44:16 WARN TaskSetManager: Lost task 18.0 in stage 0.0 (TID 18, 172.16.10.149, executor 6): org.apache.spark.memory.Spar
kOutOfMemoryError: error while calling spill() on org.apache.spark.shuffle.sort.ShuffleExternalSorter@ff9fe39 : Out of memory
        at org.apache.spark.memory.TaskMemoryManager.acquireExecutionMemory(TaskMemoryManager.java:219)
        at org.apache.spark.memory.TaskMemoryManager.allocatePage(TaskMemoryManager.java:285)
        at org.apache.spark.memory.MemoryConsumer.allocateArray(MemoryConsumer.java:96)
        at org.apache.spark.shuffle.sort.ShuffleExternalSorter.growPointerArrayIfNecessary(ShuffleExternalSorter.java:338)
        at org.apache.spark.shuffle.sort.ShuffleExternalSorter.insertRecord(ShuffleExternalSorter.java:393)
        at org.apache.spark.shuffle.sort.UnsafeShuffleWriter.insertRecordIntoSorter(UnsafeShuffleWriter.java:267)
        at org.apache.spark.shuffle.sort.UnsafeShuffleWriter.write(UnsafeShuffleWriter.java:188)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:55)
        at org.apache.spark.scheduler.Task.run(Task.scala:121)
        at org.apache.spark.executor.Executor$TaskRunner$$anonfun$10.apply(Executor.scala:403)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:409)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
```

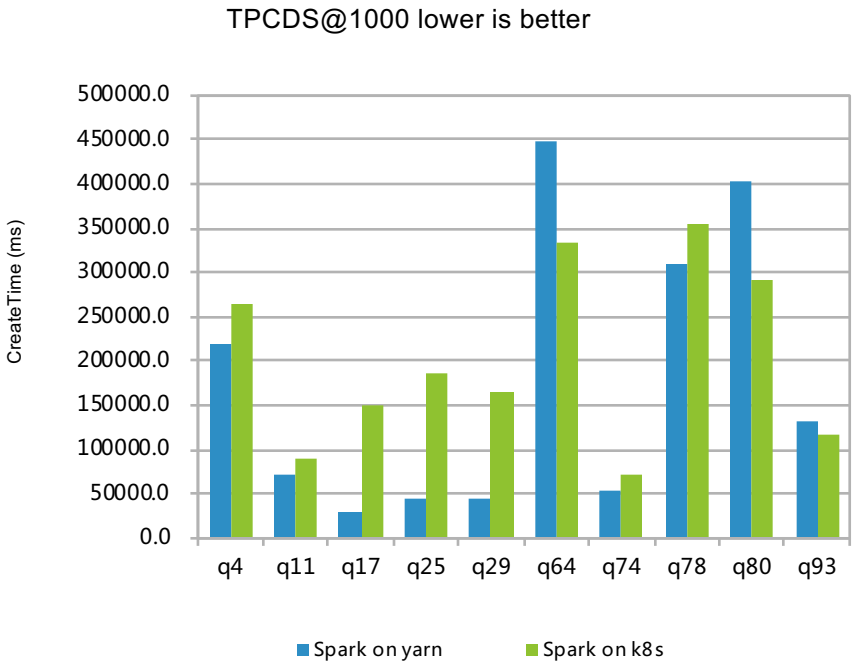
Our Solution

Use specified volumes mount as spark local directory. SPARK-27499

```
spark.kubernetes.executor.volumes.hostPath.spark-local-dir-1.mount.path=/data/mnt-c
spark.kubernetes.executor.volumes.hostPath.spark-local-dir-1.options.path=/data/mnt-c

spark.kubernetes.executor.volumes.hostPath.spark-local-dir-2.mount.path=/data/mnt-d
spark.kubernetes.executor.volumes.hostPath.spark-local-dir-2.options.path=/data/mnt-d

spark.kubernetes.executor.volumes.hostPath.spark-local-dir-3.mount.path=/data/mnt-e
spark.kubernetes.executor.volumes.hostPath.spark-local-dir-3.options.path=/data/mnt-e

spark.kubernetes.executor.volumes.hostPath.spark-local-dir-4.mount.path=/data/mnt-f
spark.kubernetes.executor.volumes.hostPath.spark-local-dir-4.options.path=/data/mnt-f

spark.local.dir=/data/mnt-c,/data/mnt-d,/data/mnt-e,/data/mnt-f
```

Result


TPCDS@1000 lower is better
time in ms
1200000.0
1000000.0
800000.0
600000.0
400000.0
200000.0
0.0
q4
q11
q17
q25
q29
q64
q74
q78
q80
q93
spark on k8s orig
spark on k8s optimized


TPCDS@1000 lower is better
CreateTime (ms)
500000.0
450000.0
400000.0
350000.0
300000.0
250000.0
200000.0
150000.0
100000.0
50000.0
0.0
q4
q11
q17
q25
q29
q64
q74
q78
q80
q93
Spark on yarn
Spark on k8s

Summary

- Spark on Kubernetes practices – scenario in serverless and cloud native
- But we hit problems in production - local storage issues, etc.
- Work with community for better solution and best practices

Future Work

- Close performance gap with yarn mode
- Data locality awareness for Kubernetes scheduler
- Prometheus/Grafana integration

SPARK+AI SUMMIT 2019

Question?

Download on the App Store

GET IT ON Google Play