



HADOOP YARN在异构环境下应用与实践

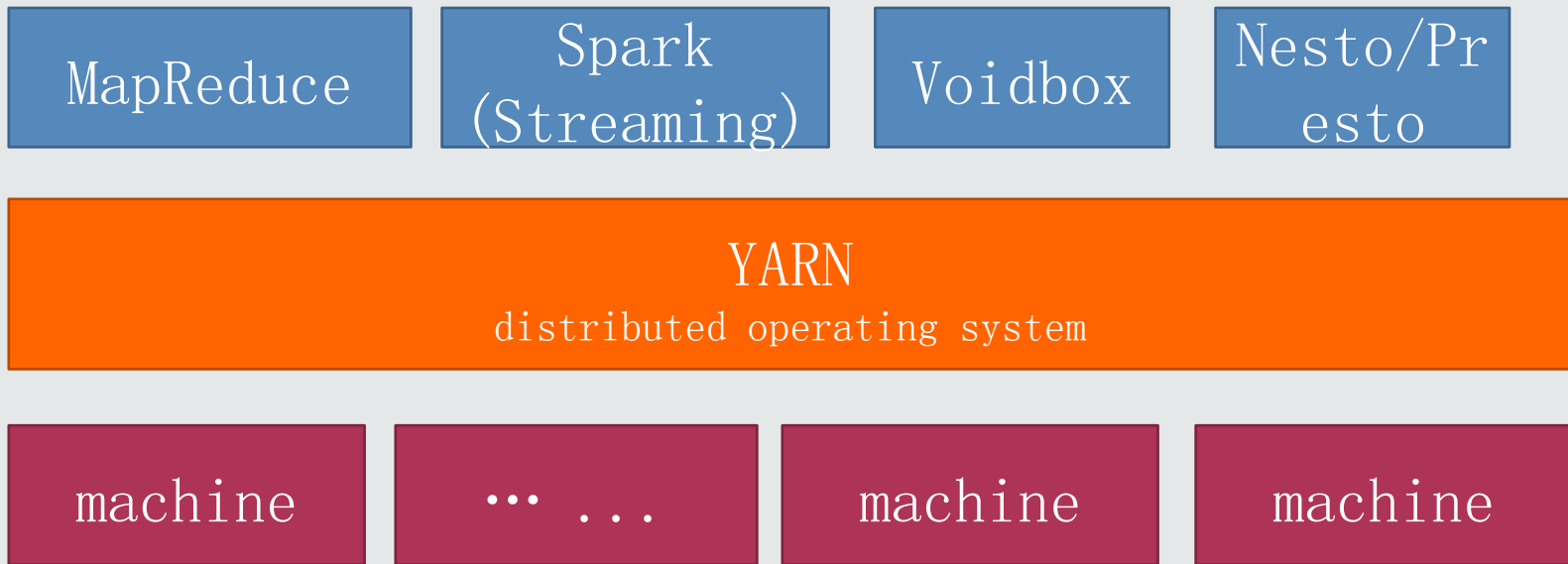
董西成

AGENDA

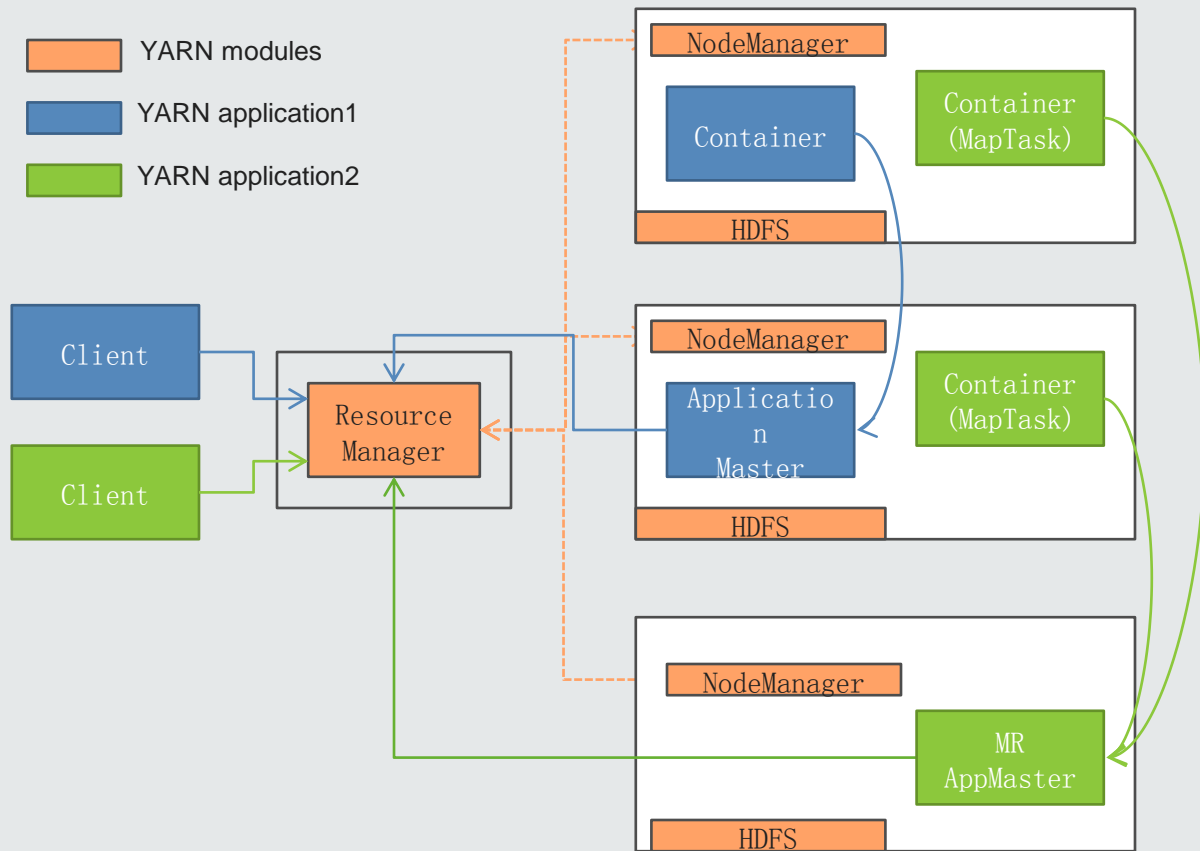
- Background
- Label-based scheduling
- Framework On YARN
- Experience
- Q & A



ARCHITECTURE - YARN



ARCHITECTURE - YARN



DISTRIBUTED APPLICATION TYPES

- Batch Jobs

- Take from a few seconds to a few days to complete
- E. g. MapReduce, Spark

- Long-running Services

- Should “never” go down, handle short-lived latency-sensitive requests
- E. g. Presto/Nesto, Spark Streaming, Web Services

- YARN: Data Operating System

- ResourceManager HA & Recovery
- NodeManager Recovery
- Resource Scheduling(e. g. label-based scheduling)
- [Apache Slider](#) & [Apache Twill](#)

HETEROGENEOUS ISSUES

- Heterogeneous Causes
 - Static factors: Machines(CPU, memory, network, OS, libraries)
 - Dynamic factors: Load, Network, Slow Disk ...
- Solutions for static factors
 - Distinguish machines, and assign to different frameworks
 - E.g. 128GB-memory machines for spark, 10G-network machines for spark streaming/nesto/presto
- Solutions for dynamical factors
 - Solve in framework level
 - For MapReduce/Spark: speculative execution
 - For others: ?

AGENDA

- Background
- Label-based scheduling
- Framework On YARN
- Experience
- Q & A



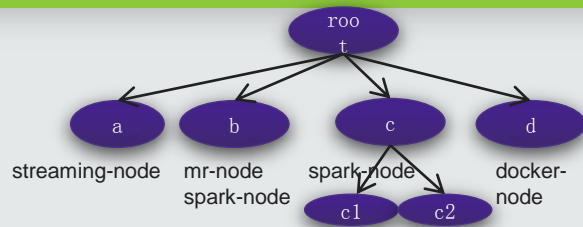
WHY LABEL-BASED SCHEDULING

- Distinguish good/bad machines for different Apps
 - E. g. Large memory machine for spark
- Apps require special libraries/services installed
 - Spark MLlib: native libraries
 - Voidbox: docker engines
- Machine-level isolation for key Apps
 - Low-latency Apps: spark streaming/presto/nesto

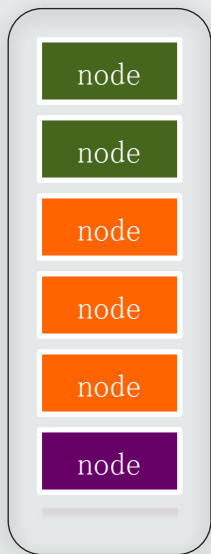
LABEL-BASED SCHEDULING

node	streaming-node
node	mr-node
node	spark-node, mr-node
node	docker-node

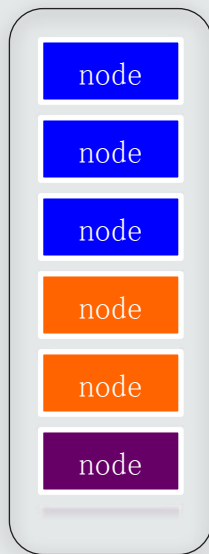
Resource Manager
(Capacity Scheduler)



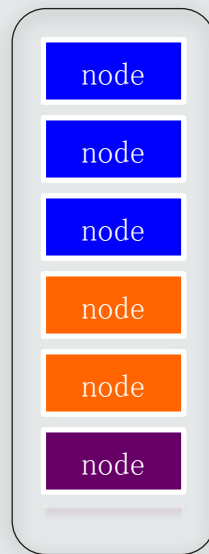
rack1



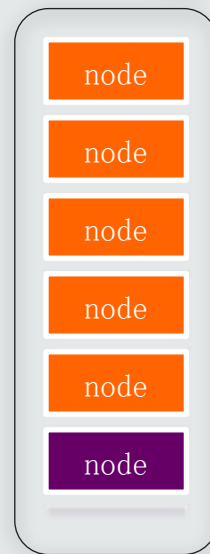
rack2



rack3



rack4



rack5

LEARN FROM LABEL-BASED SCHEDULING

- Benefit
 - Static/dynamic machine-level partition
 - Elastic scalability for Apps
 - Latch to enter maintenance mode
- Issue
 - Only Capacity Scheduler is supported
 - YARN Web Portal does not support well
 - Hurt data locality

AGENDA

- Background
- Label-based scheduling
- Frameworks On YARN
- Experience
- Q & A



FRAMEWORKS ON YARN

- MapReduce
- Spark
- Spark Streaming
 - Real-time processing
- Nesto/Presto
 - MPP engines
- Voidbox
 - Docker on yarn

WHAT IS NESTO

- What is Nesto

- A distributed query engine for parquet-like nested data
- MPP + Parquet (On HDFS)

- Nesto features

- support 1

- support f

- Support js

- use cases i

- Segment ar

- Cohort ana

- Funnel ana

```
behavior
|- watch
| - {"cid":60311148,"duc":"Computer","dlc":"HULU", "s":1}
| - {"cid":60311148,"duc":"Computer","dlc":"HULU", "f":1}
| - {"cid":60311149,"duc":"Computer","dlc":"HULU", "s":1}
| - {"cid":60311150,"duc":"Computer","dlc":"HULU", "s":1}
```

```
{"expr":["greater_than",["sum_list",["time_range_filter","last",1,"months",["get_va  
r","watch"]], "s"],5]}
```

```
public interface Action<DATA, ROW extends Serializable, KEY, VALUE extends Serializable,  
    RESULT extends Serializable> extends Serializable {  
    boolean filter(DATA data);  
    List<Pair<KEY, VALUE>> projection(DATA data);  
    List<ROW> select(DATA data);  
    VALUE combine(VALUE entry, VALUE other);  
    RESULT reduce(VALUE entry);  
}
```

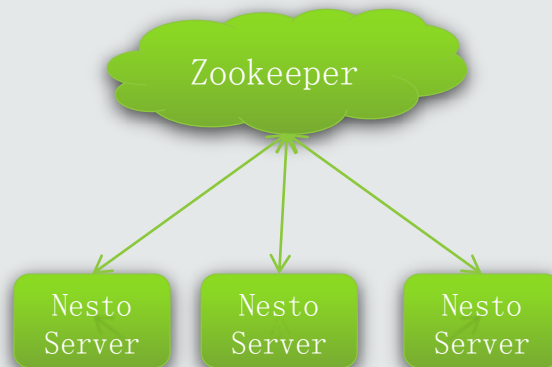
NESTO ON YARN (1)

- Requirements

- Run Specified N containers(servers), One container on node
- Fault tolerance
- Tracking web UI
- Log rotation
- Services (Never die)

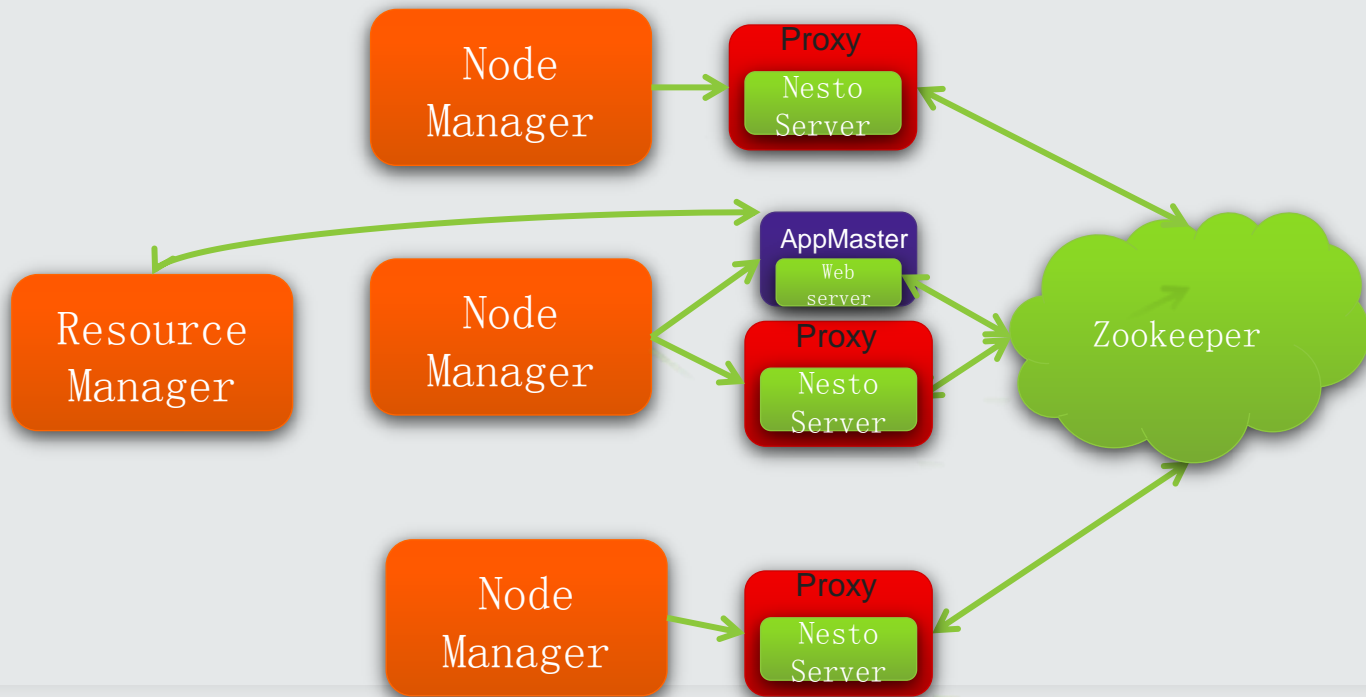
- Implementation

- ApplicationMaster: scheduling & fault tolerance
- Container: NestoServer (Worker)



NESTO ON YARN (2)

```
yarn jar nesto-install/nesto-0.1.0-SNAPSHOT-jar-with-dependencies.jar \  
com.hulu.nesto.yarn.NestoYarnClient \  
-conf_path /mnt/data1/home/xicheng.dong/nesto-src/yarn/nesto-install/conf/ \  
-framework_path hdfs:///tmp/xicheng/nesto/nesto-yarn.tar.gz \  
-master_memory 2048 -container_memory 2048 -container_vcores 1 -num_containers 3
```

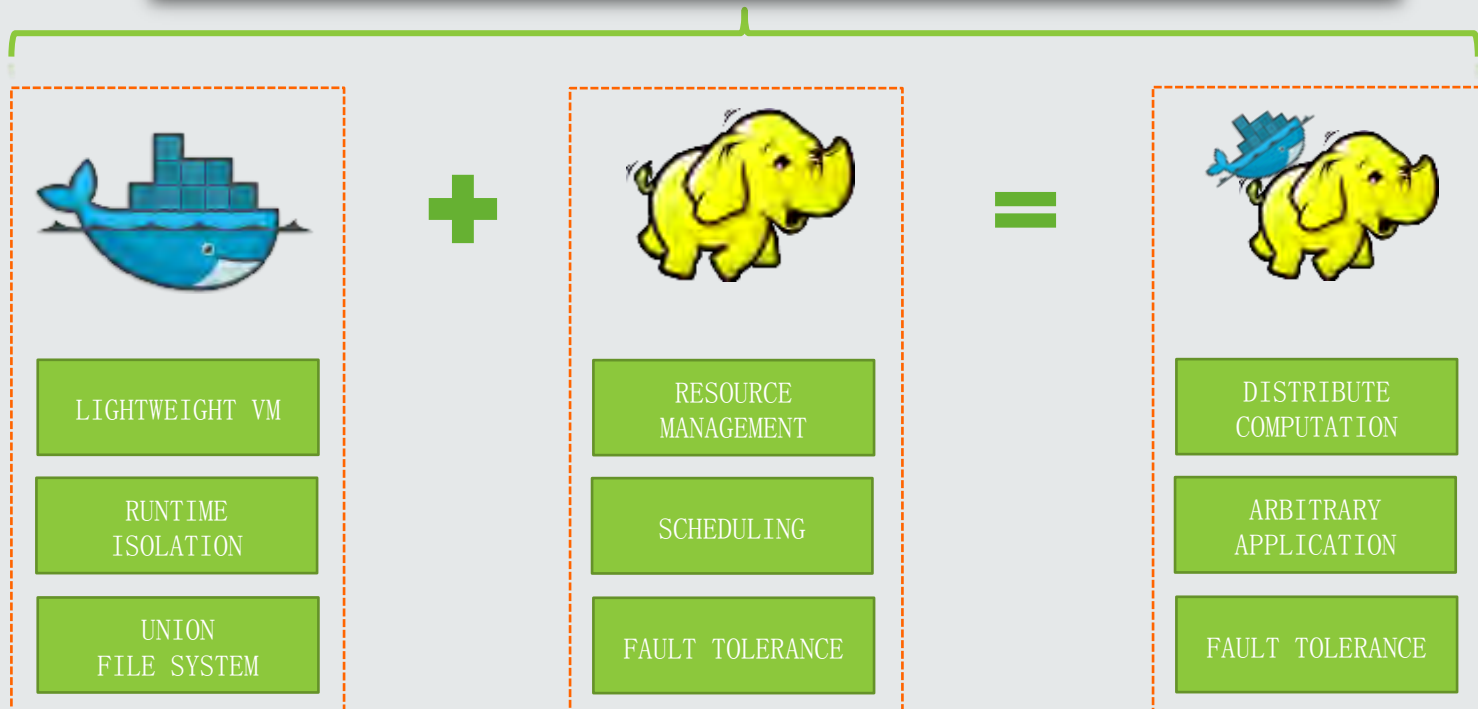


NESTO IN HETEROGENEOUS CLUSTER

- Nesto deployment
 - Nesto is not a high-concurrency system
 - Mixed deployment with other batch/OLAP systems on YARN
- Slow task/worker
 - Slow task detection and re-dispatch
 - Worker priority scheduling based on weight

WHAT IS VOIDBOX

VOIDBOX IS A PROGRAMMING FRAMEWORK



VOIDBOX SERVICE EXAMPLE

```
ServiceWarmUp serviceWarmUp = ServiceWarmUp.newBuilder()  
    .containerPort(5067).protocol(ServiceConstants.Protocol.HTTP).timeOutSeconds(600).path("").build();
```

```
ServiceHealthCheck healthCheck = ServiceHealthCheck.newBuilder()  
    .containerPort(5067)  
    .protocol(ServiceConstants.Protocol.HTTP)  
    .successCode(200)  
    .intervalSeconds(2)  
    .timeoutSeconds(10)  
    .maxConsecutiveFailures(3)  
    .path("")  
    .build();
```

```
IInstancePolicy iInstancePolicy = new ConstantsInstancesPolicy(2);
```

```
sc.serviceBuilder()  
    .setGroup(serviceGroup)  
    .setWarmUp(serviceWarmUp)  
    .setHealthCheck(healthCheck)  
    .setInstance(iInstancePolicy)  
    .build();
```

VOIDBOX COMPONENTS

Framework

DAG

SERVICE

TASK

Core

API

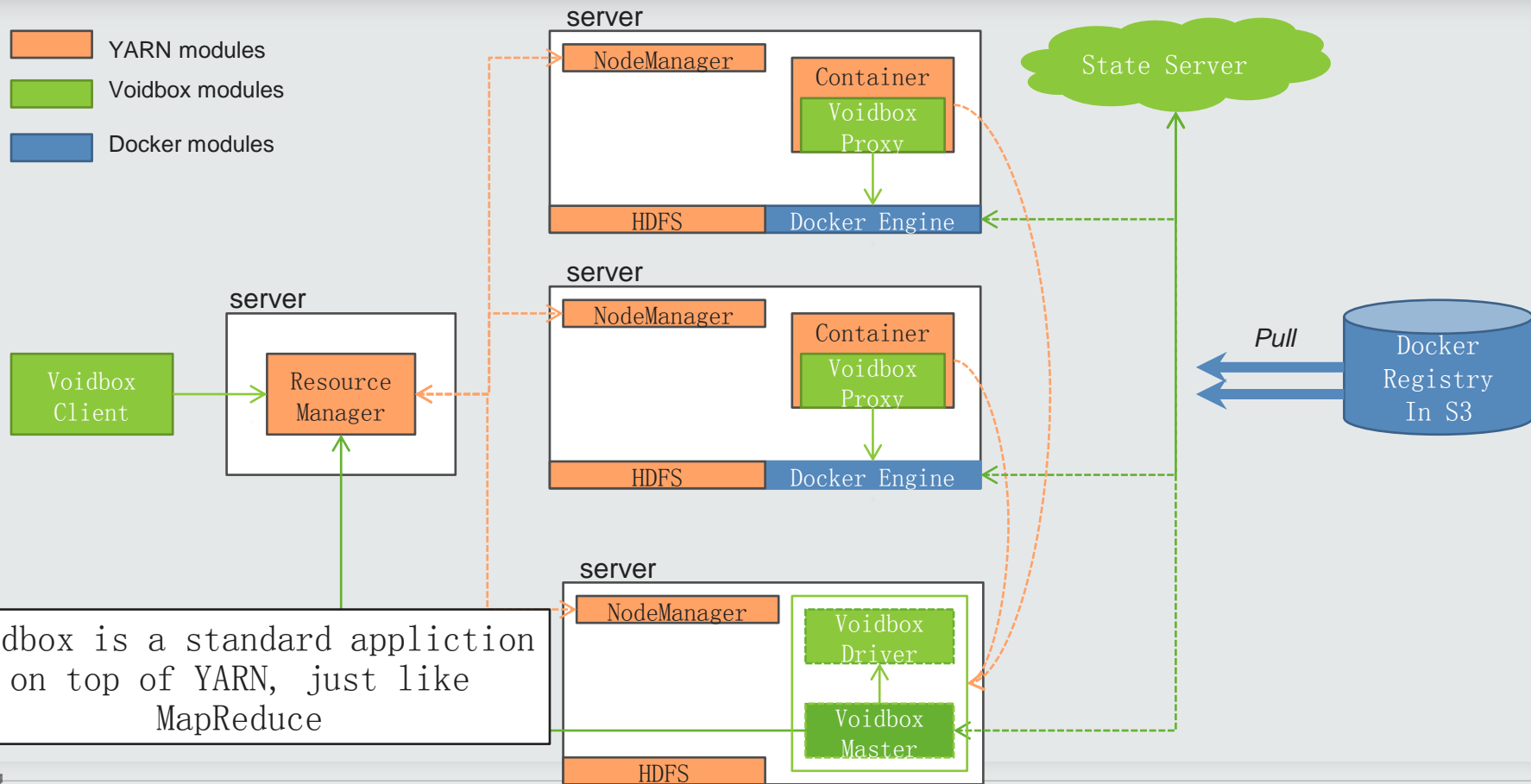
RUNTIME MODE

REGISTRY/MGMT

VOIDBOX MASTER

VOIDBOX PROXY

VOIDBOX ARCHITECTURE



VOIDBOX IN HULU

- Batch Jobs
 - Data Dump
 - Facematch: distributed application, face recognition in video
- Long-running Services
 - Nesto WebServer
 - Spark HistoryServer/JobServer
 - Kinko: thumbnail Generator Service

AGENDA

- Background
- Label-based scheduling
- Frameworks On YARN
- Experience
- Q & A



EXPERIENCE 1: CONTAINER API

- Container Description
 - Host, rack
 - Relax locality
- Blacklisted Container
 - ApplicationMaster could put certain nodes into blacklist, then no more containers on those nodes will be received.
- Release unused containers
 - If AM allocates K containers, more containers may be received.

EXPERIENCE 2: TRACKING URL

- What's tracking url
 - URL of web server inside/outside AM
 - Jetty or netty
- Two tracking URL
 - Register tracking url (running)
 - Unregister tracking url (completed)
- URL Format
 - <http://host:port>
 - Direct URL to a.jsp, use “a.jsp”, not “/a.jsp” ;
“a/b.html” not “/a/b.html”

EXPERIENCE 3: LOG ROTATION

- AM or Container run command
 - \$JAVA_HOME/bin/java -Dnesto.server.container.log=<LOG_DIR> -Dlog4j.configuration=log4j.properties ... com.hulu.NestoServer 1>><LOG_DIR>/server.log 2>><LOG_DIR>/server.log
 - <LOG_DIR> will be replaced by YARN framework
- log4j.properties

```
log4j.rootLogger=INFO,console

log4j.appender.console=org.apache.log4j.RollingFileAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.target=System.err
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %5p %c:%L - %m%n
log4j.appender.console.File=${nesto.server.container.log}/server.log
log4j.appender.console.MaxFileSize=200MB
log4j.appender.console.MaxBackupIndex=10
```

EXPERIENCE 4: MEMORY OVERHEAD

- Allocate more memory than JVM heap size

```
Vargs.add(ApplicationConstants.Environment.JAVA_HOME.ss() + "/bin/java");  
// Set an memory size  
vargs.add("-Xms" + containerMemory + "m");  
vargs.add("-Xmx" + containerMemory + "m");  
Vargs.add("-Djava.io.tmpdir=$PWD/tmp");  
vargs.add("-Dlog4j.configuration=" + NestoYarnConstants.NESTO_YARN_APPCONTAINER_LOG4J);
```

```
Resource capability = Resource.newInstance(containerMemory + memoryOverhead, ← 300MB  
    containerVirtualCores);  
  
AMRMClient.ContainerRequest request = new AMRMClient.ContainerRequest(capability, new String[] {hostname},  
    new String[] {rackname}, pri, false);
```

Java
Server

Java
JVM

Non-Java
memory

Container Memory

Hadoop Streaming is a good example!!!

EXPERIENCE 5: DEBUG

- Understand container environment
 - AM/container command is written to a shell script
- How to check the shell script content
 - Register tracking url (running)

```
vars.add("cat $PWD/launch_container.sh > /tmp/launch_container.sh && ");
vars.add("ls -la $PWD/ > /tmp/container_pwd.sh && ");
vars.add(ApplicationConstants.Environment.JAVA_HOME.$S() + "/bin/java");
// Set an memory size
vars.add("-Xms" + containerMemory + "m");
vars.add("-Xmx" + containerMemory + "m");
vars.add("-Djava.io.tmpdir=$PWD/tmp");
vars.add("-Dlog4j.configuration=" + NestoYarnConstants.NESTO_YARN_APPCONTAINER_LOG4J);
```

```
xicheng.dong@elsauidn027:~$ sudo ls -la /mnt/volume5/yarn/nm/usercache/ap.deploy/appcache/application_1446430358907_1210/container_1446430358907_1210_01_000029/
total 32
drwxr-xr-x 3 yarn yarn 4096 Nov 26 08:24 .
drwxr-xr-x 4 yarn yarn 4096 Nov 26 08:24 ..
-rw-r--r-- 1 yarn yarn 68 Nov 26 08:24 container_tokens
-rwxr-xr-x 1 yarn yarn 4137 Nov 26 08:24 launch_container.sh
lrwxrwxrwx 1 yarn yarn 79 Nov 26 08:24 nesto -> /mnt/volume5/yarn/nm/usercache/ap.deploy/filecache/253/nesto-yarn-mirror.tar.gz
drwxr-xr-x 3 yarn yarn 4096 Nov 26 08:24 tmp
-rw-r--r-- 1 yarn yarn 514 Nov 26 08:24 YarnAppContainerLog4j.properties
```

Q & A

