RATIS
RAFT library for Java
RAFT library for Java
RATIS

RAFT library for Java

https://flokkr.github.io
Energy
Manufacturer
Model

More efficient
A
B
C
D
E
F
G

Less efficient
Energy consumption kWh/year
(Based on standard test results for 24h)
325

Actual consumption will depend on how the appliance is used and where it is located

Fresh food volume L
Frozen food volume L
190
126

Noise
(dB(A) re 1 pW)

Further information is contained in product brochures

Norm EN 153 May 1990
Refrigerator Label Directive 94/2/EC
### Docker

Local compose file

**More efficient**

- A
- B
- C
- D
- E
- F
- G

**Less efficient**

<table>
<thead>
<tr>
<th>Configuration management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Preprocessing</td>
</tr>
<tr>
<td>On change</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Provisioning, Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multihost support</td>
</tr>
<tr>
<td>Scheduling</td>
</tr>
<tr>
<td>Cluster definition</td>
</tr>
<tr>
<td>Scaling</td>
</tr>
<tr>
<td>Multi-tenancy</td>
</tr>
<tr>
<td>Fallback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrasevice network</td>
</tr>
<tr>
<td>DNS</td>
</tr>
<tr>
<td>Service discovery</td>
</tr>
<tr>
<td>Data locality</td>
</tr>
<tr>
<td>Availability of the ports</td>
</tr>
<tr>
<td>Less efficient</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td><strong>Configuration management</strong></td>
</tr>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Preprocessing</td>
</tr>
<tr>
<td>On change</td>
</tr>
<tr>
<td>** Provisioning, Scheduling**</td>
</tr>
<tr>
<td>Multihost support</td>
</tr>
<tr>
<td>Scheduling</td>
</tr>
<tr>
<td>Cluster definition</td>
</tr>
<tr>
<td>Scaling</td>
</tr>
<tr>
<td>Multi tenancy</td>
</tr>
<tr>
<td>Failover</td>
</tr>
<tr>
<td><strong>Network</strong></td>
</tr>
<tr>
<td>Intraservice network</td>
</tr>
<tr>
<td>DNS</td>
</tr>
<tr>
<td>Service discovery</td>
</tr>
<tr>
<td>Data locality</td>
</tr>
<tr>
<td>Availability of the ports</td>
</tr>
</tbody>
</table>
What is Apache Hadoop
What is Apache Hadoop in 60 seconds
HDFS
HDFS

YARN

Mapreduce
FROM frolvlad/alpine-oraclejdk8
ADD hadoop-3.2.0.tar.gz /opt
WORKDIR /opt/hadoop
<table>
<thead>
<tr>
<th>Less efficient</th>
<th>docker</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration management</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>Preprocessing</td>
<td></td>
</tr>
<tr>
<td>On change</td>
<td></td>
</tr>
<tr>
<td><strong>Provisioning, Scheduling</strong></td>
<td></td>
</tr>
<tr>
<td>Multihost support</td>
<td></td>
</tr>
<tr>
<td>Scheduling</td>
<td></td>
</tr>
<tr>
<td>Cluster definition</td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td></td>
</tr>
<tr>
<td>Multi tenancy</td>
<td></td>
</tr>
<tr>
<td>Failover</td>
<td></td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td></td>
</tr>
</tbody>
</table>
<configuration>
  <property>
    <name>dfs.namenode.rpc-address</name>
    <value>namenode:9000</value>
  </property>
  <property>
    <name>dfs.datanode.plugins</name>
    <value>org.apache.hadoop.ozone.HddsDatanodeService</value>
  </property>
  <property>
    <name>rpc.metrics.percentiles.intervals</name>
    <value>60,300</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/data/namenode</value>
  </property>
  <property>
    <name>rpc.metrics.quantile.enable</name>
    <value>true</value>
  </property>
</configuration>
version: "3"
services:
  servicel:
    image: apache/imagename
    hostname: namenode
    ports:
      - 9870:9870
    environment:
      CONFIGURATION1: value
      DFS_DIR: /dfs
      THREAD_NUMBER: 1
How to handle configuration?

Create a simple **launcher** script to

- Create config file from environment variables
- Start the application
version: "3"
services:
  namenode:
    image: flokkr/hadoop
    hostname: namenode
    command: ["hdfs","namenode"]
    ports:
      - 9870:9870
    environment:
      ENSURE_NAMENODE_DIR: "/<tmp/hadoop-root/dfs/name"
      CORE-SITE.XML_fs.defaultFS: "hdfs://namenode:9000"
      HDFS-SITE.XML_dfs.namenode.rpc-address: "namenode:9000"
      HDFS-SITE.XML_dfs.replication: "1"
  datanode:
    image: flokkr/hadoop
    command: ["hdfs", "datanode"]
    environment:
      CORE-SITE.XML_fs.defaultFS: "hdfs://namenode:9000"
      HDFS-SITE.XML_dfs.namenode.rpc-address: "namenode:9000"
      HDFS-SITE.XML_dfs.replication: "1"
      LOG4J.PROPERTIES_log4j.rootLogger: "INFO, stdout"
      LOG4J.PROPERTIES_log4j.appender.stdout: "org.apache.log4j.ConsoleAppender"
      LOG4J.PROPERTIES_log4j.appender.stdout.layout: "org.apache.log4j.PatternLayout"
      LOG4J.PROPERTIES_log4j.appender.stdout.layout.ConversionPattern: "%d{yyyy-MM-dd HH:mm:ss} %-5p \
\n"
Configuration management is managing the configuration values
<table>
<thead>
<tr>
<th>Configuration management</th>
<th>ENV (script)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>n/a</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>n/a</td>
</tr>
<tr>
<td>On change</td>
<td></td>
</tr>
<tr>
<td>Provisioning, Scheduling</td>
<td>n/a</td>
</tr>
<tr>
<td>Multihost support</td>
<td></td>
</tr>
<tr>
<td>Scheduling</td>
<td></td>
</tr>
<tr>
<td>Cluster definition</td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td></td>
</tr>
<tr>
<td>Multi tenancy</td>
<td></td>
</tr>
<tr>
<td>Failover</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
</tr>
<tr>
<td>Intraservice network</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td></td>
</tr>
<tr>
<td>Service discovery</td>
<td></td>
</tr>
<tr>
<td>Data locality</td>
<td></td>
</tr>
</tbody>
</table>
Container is the **unit** of packaging.
Launcher script has the power
Launcher script
Launcher script

- Create config files from ENV
Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)

Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)
- Retrieve kerberos/SSL secrets
Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)
- Retrieve kerberos/SSL secrets
- Enable prometheus monitoring (Java agent)
Launcher script

- Create config files from ENV
- Wait for the dependency (TCP check)
- Download additional optional component
- Prepare HDFS (format namenode, ...)
- Retrieve kerberos/SSL secrets
- Enable prometheus monitoring (Java agent)
- Show network traffic (Instrumentation with Java agent)
--- Launching "hdfs datanode"

Formatting using clusterId: CID-51b1d36b-356a-4c2a-9570-57b6300ccd5d

--- Plugin is activated BYTEMAN -----

Connecting to kv.anzix.net (176.9.127.13:443)

- byteeman.jar 14% |-------------------------| 112k 0:00:06 ETA
- byteeman.jar 100% |------------------------------------| 791k 0:00:08 ETA

Connecting to gist.githubusercontent.com (151.101.36.133:443)

- byteeman.btm 100% |------------------------------------| 710 0:00:08 ETA

Process is instrumented with setting JAVA_OPTS to -javaagent:/opt/byteeman/byteeman.jar=script:/tmp/byteeman.btm

Standard output is replaced with btrace output

---

--- Launching "hdfs namenode"

--> RPC message request: VersionRequestProto from 172.23.0.3:52480

--> RPC message response: VersionRequestProto to 172.23.0.3:52480

info {
  buildVersion: "16b70619a24cddf5d3b0fcf4b58ca77238ccbe6d"
  unused: 0
  blockPoolID: "BP-1294221783-172.23.0.2-1528791611084"
  storageInfo {  
    layoutVersion: 4294967232
    namespaceID: 2129242657
    clusterID: "CID-51b1d36b-356a-4c2a-9570-57b6300ccd5d"
    ctime: 1528791611084
  }
  softwareVersion: "3.1.0"
  capabilities: 1
  state: ACTIVE
}

--> RPC message request: RegisterDatanodeRequestProto from 172.23.0.3:52480

registration {
  datanodeID {
    ipAddr: "0.0.0.0"
    hostName: "8d6811399538"
    datanodeUuid: "92926bb4-04b5-4e9c-8f85-694a2d7c61ec"
    xferPort: 9866
    infoPort: 9864
    ipcPort: 9867
    infoSecurePort: 0
  }
}

---
Hashicorp stack
"do it yourself"
Consul
Service Discovery and Configuration Made Easy
Consul
Service Discovery and Configuration Made Easy

Vault
A Tool for Managing Secrets

Nomad
Easily Deploy Applications at Any Scale
node1
node2
node3
node4
node5
Host network
<table>
<thead>
<tr>
<th>Provisioning, Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multihost support</td>
</tr>
<tr>
<td>Scheduling</td>
</tr>
<tr>
<td>Cluster definition</td>
</tr>
<tr>
<td>Scaling</td>
</tr>
<tr>
<td>Multi tenancy</td>
</tr>
<tr>
<td>Failover</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intraservice network</td>
</tr>
<tr>
<td>DNS</td>
</tr>
<tr>
<td>Service discovery</td>
</tr>
<tr>
<td>Data locality</td>
</tr>
<tr>
<td>Availability of the ports</td>
</tr>
</tbody>
</table>
<configuration>
  <property>
    <name>dfs.namenode.rpc-address</name>
    <value>namenode:9000</value>
  </property>
  <property>
    <name>dfs.datanode.plugins</name>
    <value>org.apache.hadoop.ozone.HddsDatanodeService</value>
  </property>
  <property>
    <name>rpc.metrics.percentiles.intervals</name>
    <value>60,300</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/data/namenode</value>
  </property>
  <property>
    <name>rpc.metrics.quantile.enable</name>
    <value>true</value>
  </property>
</configuration>
<table>
<thead>
<tr>
<th>Less efficient</th>
<th>Consul</th>
<th>Nomad</th>
<th>Consul</th>
<th>Nomad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration management</td>
<td>Source</td>
<td>Preprocessing</td>
<td>On change</td>
<td>Restart</td>
</tr>
<tr>
<td>Provisioning, Scheduling</td>
<td>Multihost support</td>
<td>Scheduling</td>
<td>Cluster definition</td>
<td>Nomad .nomad redeploy</td>
</tr>
<tr>
<td></td>
<td>Scaling</td>
<td>Multi tenancy</td>
<td>Failover</td>
<td>no yes</td>
</tr>
<tr>
<td>Network</td>
<td>Intraservice network</td>
<td>DNS</td>
<td>Service discovery</td>
<td>Host netw</td>
</tr>
</tbody>
</table>
|                      | Data locality | Availability of the ports | yes | consul yes | host
Kubernetes
"out of the box"
Network!
Storage!!
(volume, secrets, configs)
DaemonSet

node1  node2  node3  node4  node5
ReplicaSet

node1
node2
node3
node4
node5
DaemonSet

node1
node2
node3
node4
node5
Benefits of Hadoop + k8s?
Benefits of Hadoop + k8s?

Ecosystem

Flexibility
Example:
Monitor Hadoop with Prometheus
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
    spec:
      containers:
        - name: hdfs-namenode
          image: flokkr/ozone:2.1.0
          args: ["hdfs","namenode"]
apiVersion: apps/v1beta1
class kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
template:
  metadata:
    labels:
      app: ozone
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/port: "28942"
spec:
  containers:
    - name: hdfs-namenode
      image: flokkr/ozone:2.1.0
      args: ["hdfs","namenode"]
Sidecar pattern

- container1
- container2
- network
- volume
- ps xa (alpha)

Pod
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: ozone-hdfs-namenode
spec:
  serviceName: ozone2-hdfs-namenode
  replicas: 1
  template:
    metadata:
      labels:
        app: ozone
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "28942"
    spec:
      shareProcessNamespace: true
      containers:
      - name: hdfs-namenode
        image: flokkr/ozone:2.1.0
        args: ["hdfs","namenode"]
      - name: jmxpromo
        image: flokkr/jmxpromo-sidecar
<table>
<thead>
<tr>
<th>Less efficient</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration management</strong></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>configmap</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>helm</td>
</tr>
<tr>
<td>On change</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>Provisioning, Scheduling</strong></td>
<td></td>
</tr>
<tr>
<td>Multihost support</td>
<td>CNI</td>
</tr>
<tr>
<td>Scheduling</td>
<td>kubectl</td>
</tr>
<tr>
<td>Cluster definition</td>
<td>helm, yaml</td>
</tr>
<tr>
<td>Scaling</td>
<td>yes</td>
</tr>
<tr>
<td>Multi tenancy</td>
<td>namespaces</td>
</tr>
<tr>
<td>Failover</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td></td>
</tr>
<tr>
<td>Intraservice network</td>
<td>CNI</td>
</tr>
<tr>
<td>DNS</td>
<td>statefuset</td>
</tr>
<tr>
<td>Service discovery</td>
<td>DNS</td>
</tr>
<tr>
<td>Data locality</td>
<td>no</td>
</tr>
<tr>
<td>Availability of the ports</td>
<td>service/ingress</td>
</tr>
</tbody>
</table>
### Summary

<table>
<thead>
<tr>
<th>More efficient</th>
<th>Less efficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>G</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>Configuration management</td>
<td>Provisioning, Scheduling</td>
</tr>
<tr>
<td>Source</td>
<td>Multihost support</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>Scheduling</td>
</tr>
<tr>
<td>On change</td>
<td>Cluster definition</td>
</tr>
<tr>
<td></td>
<td>Scaling</td>
</tr>
<tr>
<td></td>
<td>Multi-tenancy</td>
</tr>
<tr>
<td></td>
<td>Failover</td>
</tr>
<tr>
<td>Network</td>
<td></td>
</tr>
<tr>
<td>Intraservice network</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td></td>
</tr>
<tr>
<td>Service discovery</td>
<td></td>
</tr>
<tr>
<td>Data locality</td>
<td></td>
</tr>
<tr>
<td>Availability of the ports</td>
<td></td>
</tr>
</tbody>
</table>
Summary

Don't buy without checking the label
Summary

Don't buy without checking the label

Hadoop is first class citizen of cloud-native/containerized environments*
Summary

Don't buy without checking the label

Containerization can help a lot to manage our Bigdata clusters

Hadoop is first class citizen of cloud-native/containerized environments*
Márton Elek @anzix
https://flokkr.github.io (bigdata + containers project)
https://github.com/flokkr (source)
elek@apache.org
Image credits

Yan Pritzker (CC)

Carrie Cizauskas (CC)