Big Data Analytics on Docker in Practice

Shi, Dongjie Wang, Jun
A “simple” Big Data Analytics Vertical Application

*Other names and brands may be claimed as the property of others
Simple?

- Any Big Data analytics application is a very **complex** system with many components (micro-services)
  - HDFS*/Ceph*/GlusterFS*
  - YARN*/Mesos*/Kubernetes*
  - Hbase*/MangoDB*/Cassandra*/Redis*/Neo4j*
  - Kafka*/Flume*
  - MapReduce*/Spark*/Flink*/Storm*

- Big Data Components need to be managed
  - Dependency Management, Provisioning, Deployment, Monitoring, Resilience, Resource Management

- Infrastructure need to be managed
  - Provisioning, Monitoring, Repair, Resource Management

*Other names and brands may be claimed as the property of others*
Orchestration Tool, is an underlying “plumbing” layer, or a Lego player.

- Big Data System Oriented
- Container (Docker) Based
- (Micro-)Service Orchestration
- Infrastructure (Cluster) Management
- DevOps Automation
Big Data System Oriented & Container(Docker) Based
A Real-World Example

*Other names and brands may be claimed as the property of others
Big Data System Oriented & Container (Docker) Based
A Real-World Example

*Other names and brands may be claimed as the property of others
(Micro-) Service Orchestration
Arda Micro-Service Architecture

Multiple **verticals** runs on a cluster of **nodes**
- A **vertical** can be an E2E vertical solution (e.g., video analysis) or vertical platform.
- A **node** can be a physical server or VM

Each vertical consists of multiple **services**
- A **service** can be a big data service. (e.g., Kafka, HDFS, HBase, YARN, Spark, ZooKeeper, etc.)

Each service in turn consists of multiple **micro-services** (or roles)
- A **micro-service** can be a role in the big data service. (e.g., master and worker for Spark, NameNode and DataNode for HDFS, etc.)

A micro-service has multiple replicated and/or partitioned instances, each running on a different node.

*Other names and brands may be claimed as the property of others*
The Definition of a vertical DSL

```scala
val nodes = defineNodes("ValInor-Node-", 1, 2, 3)

val dockerHub = "10.239.45.219:15000"
val hdfsNameNodeImage = "$dockerHub/arda/hdfs-namenode:2.7.1-a3.2-0.6-SNAPSHOT"
val hdfsDataNodeImage = "$dockerHub/arda/hdfs-datanode:2.7.1-a3.2-0.6-SNAPSHOT"
val yarnResourceManagerImage = "$dockerHub/arda/yarn-resource-manager:2.7.1-a3.2-0.6-SNAPSHOT"
val yarnNodeManagerImage = "$dockerHub/arda/yarn-nodemanager:2.7.1-a3.2-0.6-SNAPSHOT"

val hdfsParameters = s"-c fs.defaultFS=hdfs://$nodes(0):9000 -c dfs.namenode.datanode.registration.ip-hostname-check=false -c dfs.namenode.name.dir=/dir1/dfs/name -c dfs.datanode.data.dir=/dir1/dfs/data"
val yarnParameters = s"-s fs.defaultFS=hdfs://$nodes(0):9000 -c yarn.resourcemanager.hostname=$nodes(0) -c yarn.nodemanager.resource.memory-mb=12800 -c yarn.scheduling-policy=capacity"

val hdfsNameNode = defineMicroServiceWithName("hdfsNameNode") withId("1") useImage(hdfsNameNodeImage) atHost(nodes(0)) withPorts("9000 50070") withParameters(hdfsParameters)
val hdfsDataNodes = range(1, 2) map { i ->
    defineMicroServiceWithName("hdfsDataNode") + i useImage(hdfsDataNodeImage) atHost(nodes(1 + i)) withPorts("50075") withParameters(hdfsParameters) dependsOn(hdfsNameNode)
}
val hdfsService = defineServiceWithName("HDFS Service") addUris("http://$nodes(0):50070") addMicroService(hdfsNameNode) addMicroServices(hdfsDataNodes)

val yarnResourceManagerNode = defineMicroServiceWithName(yarnResourceManagerImage) atHost(nodes(1)) withPorts("8088") withParameters(yarnParameters)
val yarnNodeManagerNodes = range(2, 3) map { i ->
    defineMicroServiceWithName("yarnNodeManager") + i useImage(yarnNodeManagerImage) atHost(nodes(2 + i)) withParameters(yarnParameters) dependsOn(yarnResourceManagerNode)
}
val yarnService = defineServiceWithName("Yarn Service") addUris("http://$nodes(2):8088") addMicroService(yarnResourceManagerNode) addMicroServices(yarnNodeManagerNodes) dependsOn(hdfsService)

val commonParameters = hdfsParameters.zip(yarnParameters).takeWhile(Function.tupled(_._1, _._2, _._3, _._4, _._5), unzip_1, mkString)
val wordcountParameter = s"$commonParameters.split(commonParameters.size()).sort().mkString(\"","\") -- $nodes(0):9000 4 2 4 1"
val wordcount = defineMicroServiceWithName("spark-wordcount") useImage("$dockerHub/arda/spark-wordcount:1.0-a3.2-0.6-SNAPSHOT") atHost(nodes(0)) withParameters(wordcountParameter)

val vertical = defineVertical withName("spark-vertical") addService(hdfsService) addService(yarnService) addDriver(wordcount)
```

*Other names and brands may be claimed as the property of others*
Infrastructure (Cluster) Management Architecture

Orchestration Services

*Other names and brands may be claimed as the property of others*
DevOps Automation
Lifecycle of a Vertical

NEW/STOPPED
(no containers and resources)

RUNNING
(containers up and watchdog up)

SUSPENDED
(all containers paused)

ERROR

None

Snap shot

CREATE
DELETE
UPDATE*
deploy
stop
suspend
resume
clear

error

Vertical Resource
Vertical Operation

Snaphot

READ

*Other names and brands may be claimed as the property of others
Some Concerns

- Performance Overhead
- Big Data Service Specific Configurations
- Container Watchdog
- Lifecycle Interceptors
- Log Management
- Performance Monitoring
An unified boot pattern for Docker image development

- **load_parameters**
  - `-c key1=value1` `-c key2=value2` `-r para1, para2, para3` `-w para4`

- **config**
  - `${config.key=config.default-value}` e.g. core-site.xml, hdfs-site.xml

- **Watchdog**
  - watch the processes/ports
  - customized watchdog

- **logs collection**

- **startup**

- **lifecycle interceptors**
  - `pre_start`
  - `post_start`
  - `pre_shutdown`

---

```bash
#!/bin/bash

CONFIG_FILES=()
LOG_FILES=()
WATCHED_PROCESSES=()
WATCHED_PORTS=()

function do_run(){
    echo "do_run() at boot.conf has not been implemented"
}

function do_pre_start(){
}
function do_post_start(){
}
function do_pre_shutdown(){
}
```

*Other names and brands may be claimed as the property of others*
Log Management

• Docker logging drivers
  • none/json/syslog/journal/gelf/fluentd/...
  • multiple logging drivers

• How about In-Container Log Files?
  • Unified boot pattern

• Docker*  →  fluentd*  →  Elasticsearch* [EFK]
Performance Monitoring

- Monitoring each node/container
  - CPU, Memory, DISK, NetWork
  - via cAdvisor*

- Monitoring the cluster
  - CPU, Memory, DISK, NetWork
  - per Node
  - per Container
  - via Heapster*
  - with History

*Other names and brands may be claimed as the property of others
Legal Disclaimer

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase.

For more information go to http://www.intel.com/performance.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of other.

Copyright ©2016 Intel Corporation.