Lessons Learned From Running Spark On Docker

Thomas Phelan
Chief Architect, BlueData
@tapbluedata

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Outline

- Spark Deployment Models
- Spark on Docker
- Lessons Learned
- Performance
- Demo
- Key Takeaways

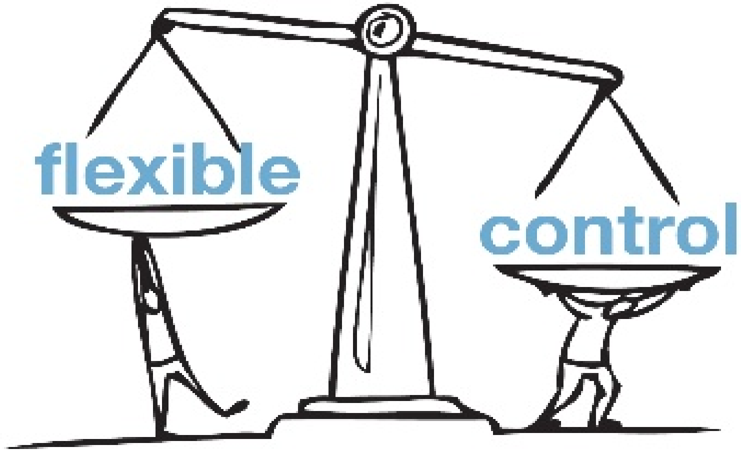
Deploying Multiple Spark Clusters

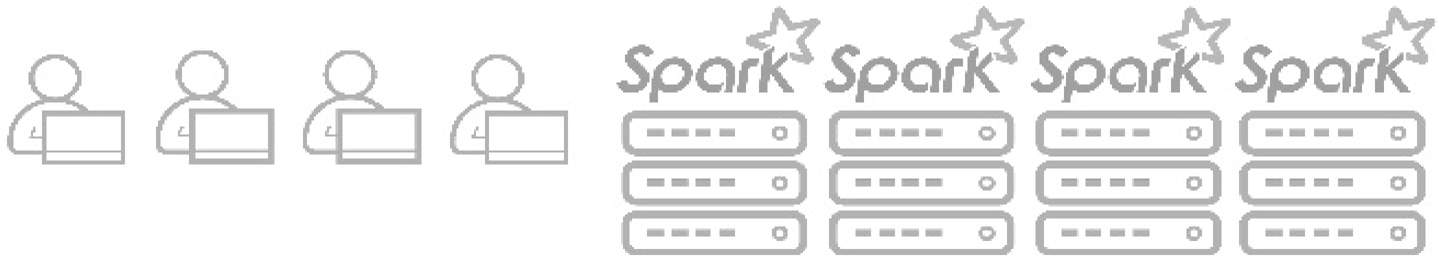
Data scientists want flexibility:

- Different versions of Spark
- Different sets of tools

IT wants control:

- Multi-tenancy
  - Data security
  - Network isolation





Spark Deployment Models

On-premises:

- Hadoop distribution running Spark on YARN
- Spark standalone mode
- Spark using the Mesos container manager/resource scheduler
- **Spark (standalone or on YARN) deployed as a collection of services running within Docker containers**

Spark-as-a-Service in the cloud:

- Databricks
- AWS EMR, Google Dataproc, Microsoft Azure, IBM, others

Advantages of Docker Containers

Property	Description
Hardware-Agnostic	Using operating system primitives (e.g. LXC), containers can run consistently on any server or VM without modification
Content-Agnostic	Can encapsulate any payload and its dependencies
Content Isolation	Resource, network, and content isolation. Avoids dependency hell
Automation	Standard operations to run, start, stop, commit, etc. Perfect for DevOps
Highly Efficient	Lightweight, virtually no performance or start-up penalty

Running Spark on Docker

- Docker containers provide a powerful option for greater agility and flexibility in application deployment on-premises
- Running a complex, multi-service platform such as Spark in containers in a distributed enterprise-grade environment can be daunting
- Here is how we did it ... while maintaining performance comparable to bare-metal

Spark on Docker: Design

- Deploy Spark clusters as Docker containers spanning multiple physical hosts
- Master container runs all Spark services (master, worker, jupyter, zeppelin)
- Worker containers run Spark worker
- Automate Spark service configuration inside containers to facilitate cluster cloning
- Container storage is always ephemeral. Persistent storage is external

Spark Dockerfile

```
# Spark-1.5.2 docker image for RHEL/CentOS 6.x

FROM centos:centos6

# Download and extract spark
RUN mkdir /usr/lib/spark; curl -s http://d3kbcqa49mib13.cloudfront.net/spark-1.5.2-bin-hadoop2.4.tgz | tar -xz -C /usr/lib/spark/

# Download and extract scala
RUN mkdir /usr/lib/scala; curl -s http://www.scala-lang.org/files/archive/scala-2.10.3.tgz | tar xz -C /usr/lib/scala/

# Install zeppelin
RUN mkdir /usr/lib/zeppelin; curl -s http://10.10.10.10:8080/build/thirdparty/zeppelin/zeppelin-0.6.0-incubating-SNAPSHOT-
v2.tar.gz|tar xz -C /usr/lib/zeppelin

RUN yum clean all && rm -rf /tmp/* /var/tmp/* /var/cache/yum/*

ADD configure_spark_services.sh /root/configure_spark_services.sh
RUN chmod -x /root/configure_spark_services.sh && /root/configure_spark_services.sh
```

Spark on Docker: Lessons

Resource Utilization:

- CPU cores vs. CPU shares
- Over-provisioning of CPU recommended
  - noisy-neighbor problem
- No over-provisioning of memory
  - swap

Spark Image Management:

- Utilize Docker's open-source image repository
- Author new Docker images using Dockerfiles
- ***Tip:*** **Docker images can get large. Use "docker squash" to save on size**

Spark on Docker: Lessons

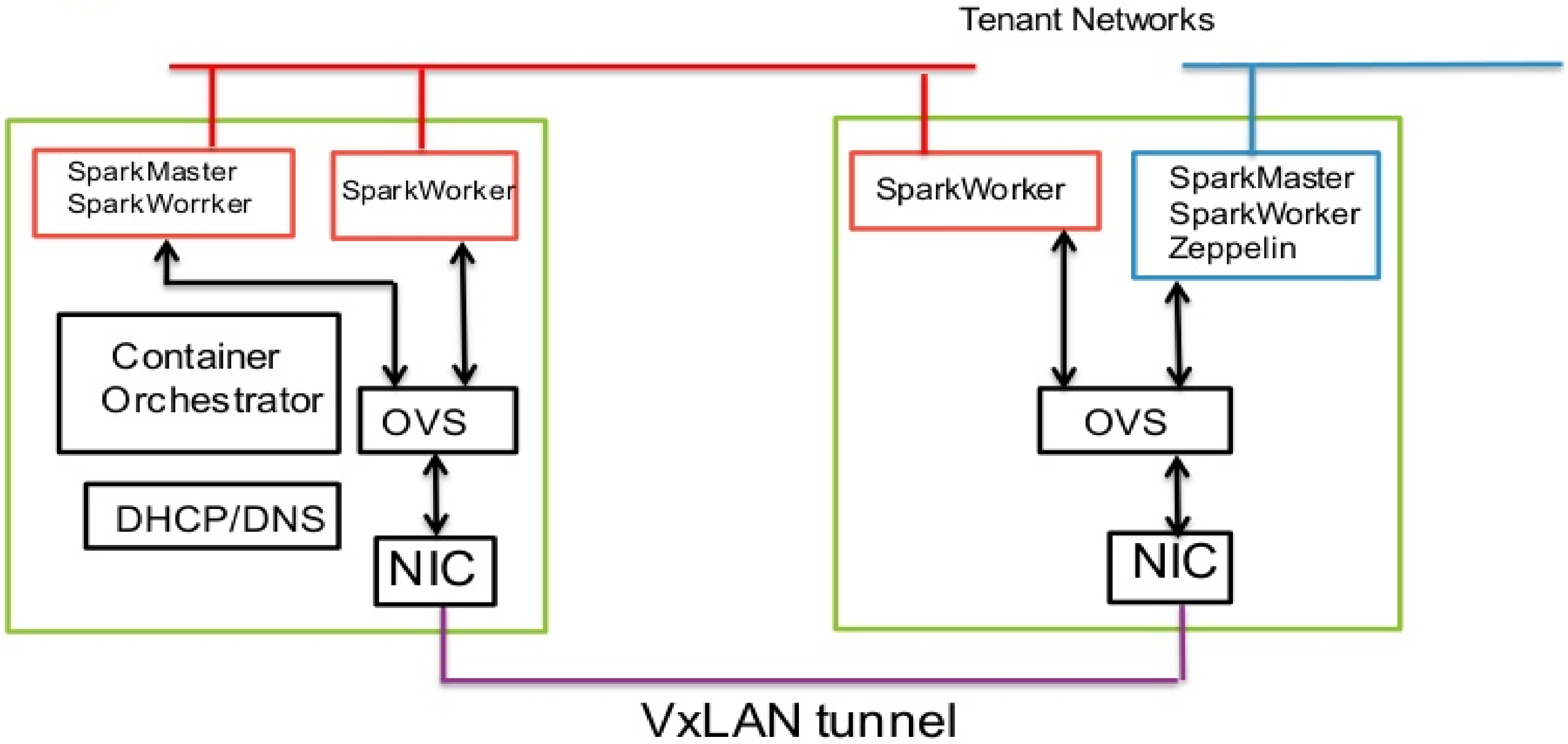
Network:

- Connect containers across hosts
  - – Various network plugins available with Docker v1.10
- Persistence of IP address across container restart
- DHCP/DNS service required for IP allocation and hostname resolution
- Deploy VLANs and VxLAN tunnels for tenant level traffic isolation

Storage:

- Default size of a container's /root needs to be tweaked
  - Resizing of storage inside an existing container is tricky
- Mount /root and /data as block devices
- ***Tip: Mounting block devices into a container does not support symbolic links (IOW: /dev/sdb will not work, /dm/… PCI device can change across host reboot)***

Spark on Docker Architecture



Docker Security Considerations

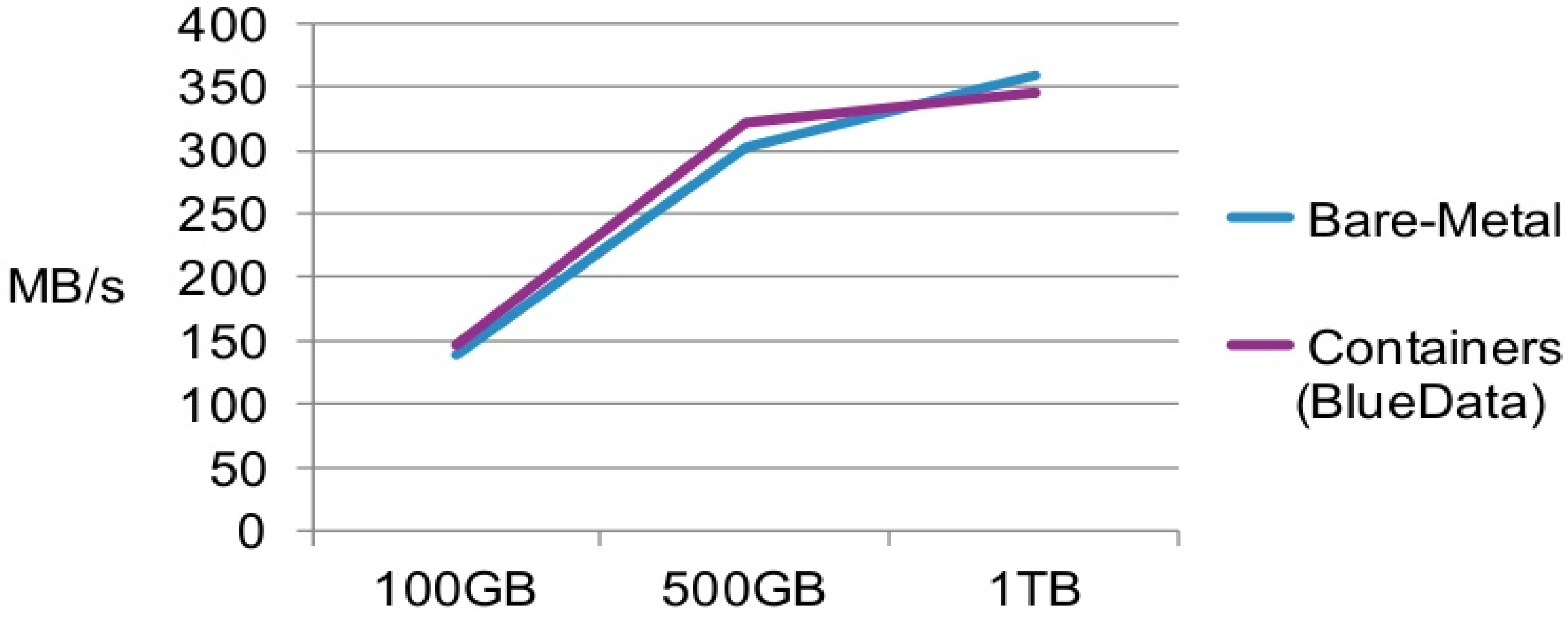
- Security is essential since containers and host share their kernel
  - Non-privileged containers
- Achieved through layered set of capabilities
- Different capabilities provide different levels of isolation and protection
- Add “capabilities” to a container based on what operations are permitted

SETPCAP	Modify process capabilities.
SYS_RESOURCE	Override resource Limits.
AUDIT_WRITE	Write records to kernel auditing log.
CHOWN	Make arbitrary changes to file UIDs and GIDs (see chown(2)).
DAC_OVERRIDE	Bypass file read, write, and execute permission checks.
DAC_READ_SEARCH	Bypass file read permission checks and directory read and execute permission checks.
KILL	Bypass permission checks for sending signals.
SETGID	Make arbitrary manipulations of process GIDs and supplementary GID list.
SETUID	Make arbitrary manipulations of process UIDs.
NET_RAW	Use RAW and PACKET sockets.
NET_BIND_SERVICE	Bind a socket to internet domain privileged ports (port numbers less than 1024).
NET_BROADCAST	Make socket broadcasts, and listen to multicasts.
SYS_CHROOT	Use chroot(2), change root directory.
SYS_PTRACE	Trace arbitrary processes using ptrace(2).
SETFCAP	Set file capabilities.

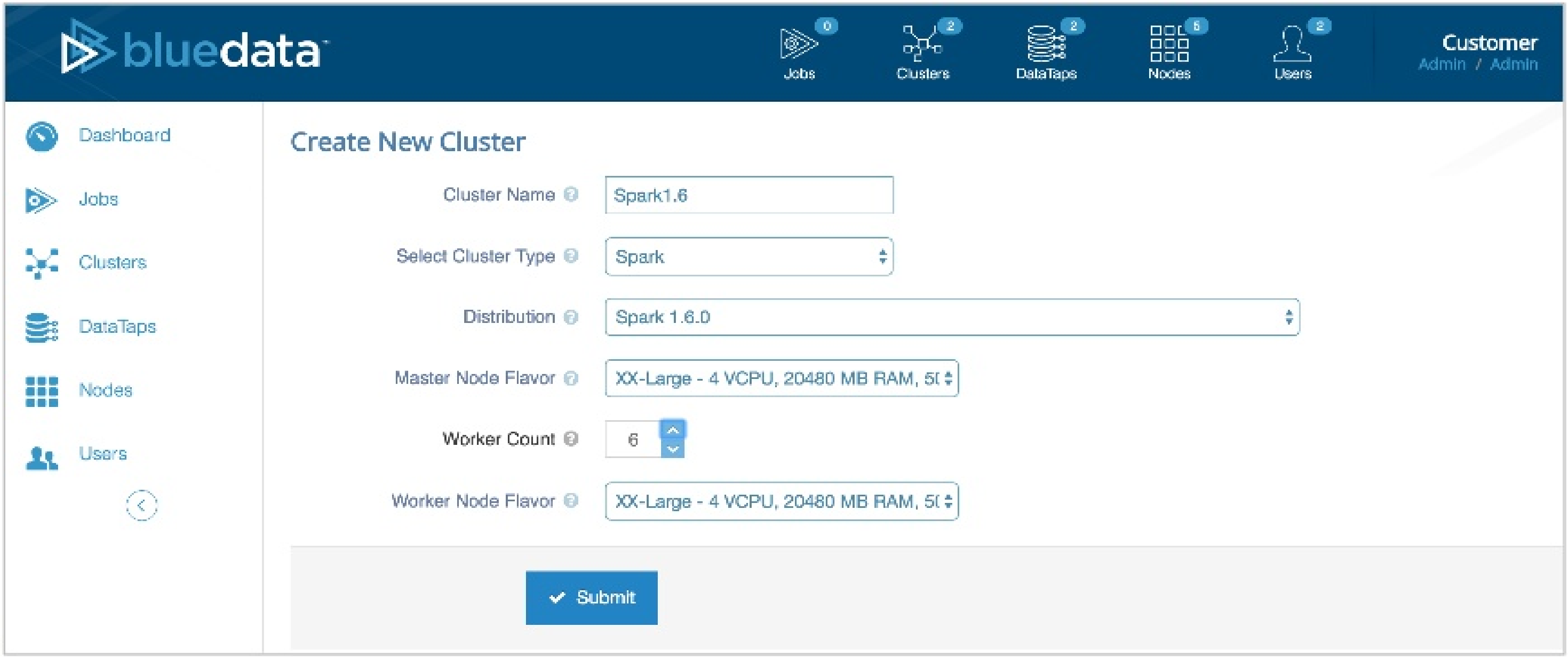
Spark on Docker Performance

- Spark 1.x on YARN
- HiBench - Terasort
  - Data sizes: 100Gb, 500GB, 1TB
- 10 node physical/virtual cluster
- 36 cores and112GB memory per node
- 2TB HDFS storage per node (SSDs)
- 800GB ephemeral storage

Spark Performance Results



Demo


bluedata
Jobs
Clusters
DataTaps
Nodes
Users
Customer
Admin / Admin
Dashboard
Jobs
Clusters
DataTaps
Nodes
Users
Create New Cluster
Cluster Name
Spark1.6
Select Cluster Type
Spark
Distribution
Spark 1.6.0
Master Node Flavor
XX-Large - 4 VCPU, 20480 MB RAM, 5(
Worker Count
6
Worker Node Flavor
XX-Large - 4 VCPU, 20480 MB RAM, 5(
Submit

Key Takeaways of Spark in Docker

- Value for a single cluster deployment
  - Significant benefits and savings for enterprise deployment
- Get best of both worlds:
  - On-premises: security, governance, no data copying/moving
  - Spark-as-a-Service: multi-tenancy, elasticity, self-service
- Performance is good

THANK YOU.

Thomas Phelan
@tapbluedata
tap@bluedata.com
www.bluedata.com

bluedata™

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO