Standard Chartered

YARN 3.1 and Beyond !

Naganarasimha

ApacheCon

Here for good

Introduction

Standard Chartered

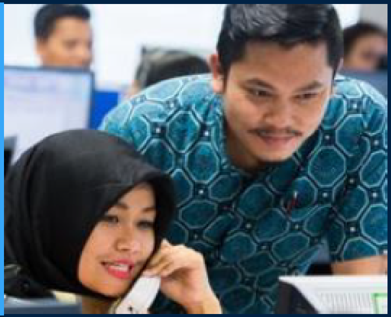
Naganarasimha GR

- Apache Hadoop PMC
- Contributing since 5 years
- Senior BigData Architect @ Standard Chartered Bank SG
- Contributed in key YARN features
  - Node attributes
  - ATS V2

Leading the way in international banking

86,000

We employ more than 86,000 people around the world



125

Our staff come from 125 different countries

150

More than 150 years in business

>1100

Branches worldwide

60

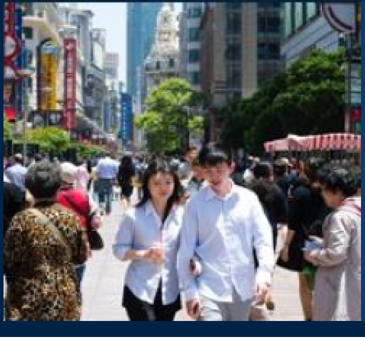
We operate in 60 countries globally

Best Consumer Digital Bank

Global Finance 2017

100

Among the top 100 largest companies listed on the London Stock Exchange



2

We're listed on two of Asia's largest stock exchanges

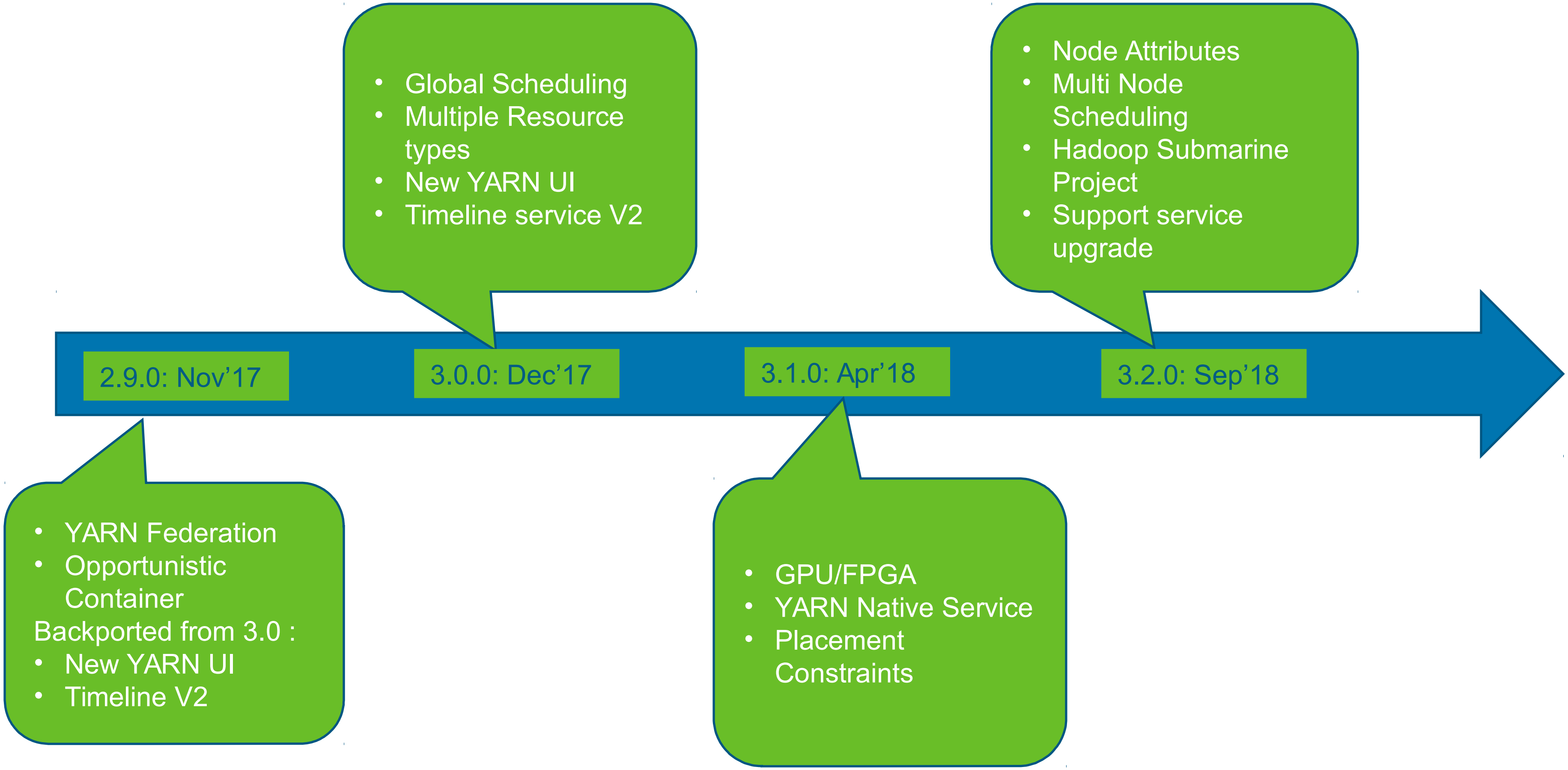
- Introduction
- Present
- Upcoming

Standard Chartered

Introduction

Here for good

A brief timeline of Hadoop Releases:



2.9.0: Nov’17

- YARN Federation
- Opportunistic Container

Backported from 3.0 :

- New YARN UI
- Timeline V2

3.0.0: Dec’17

- Global Scheduling
- Multiple Resource types
- New YARN UI
- Timeline service V2

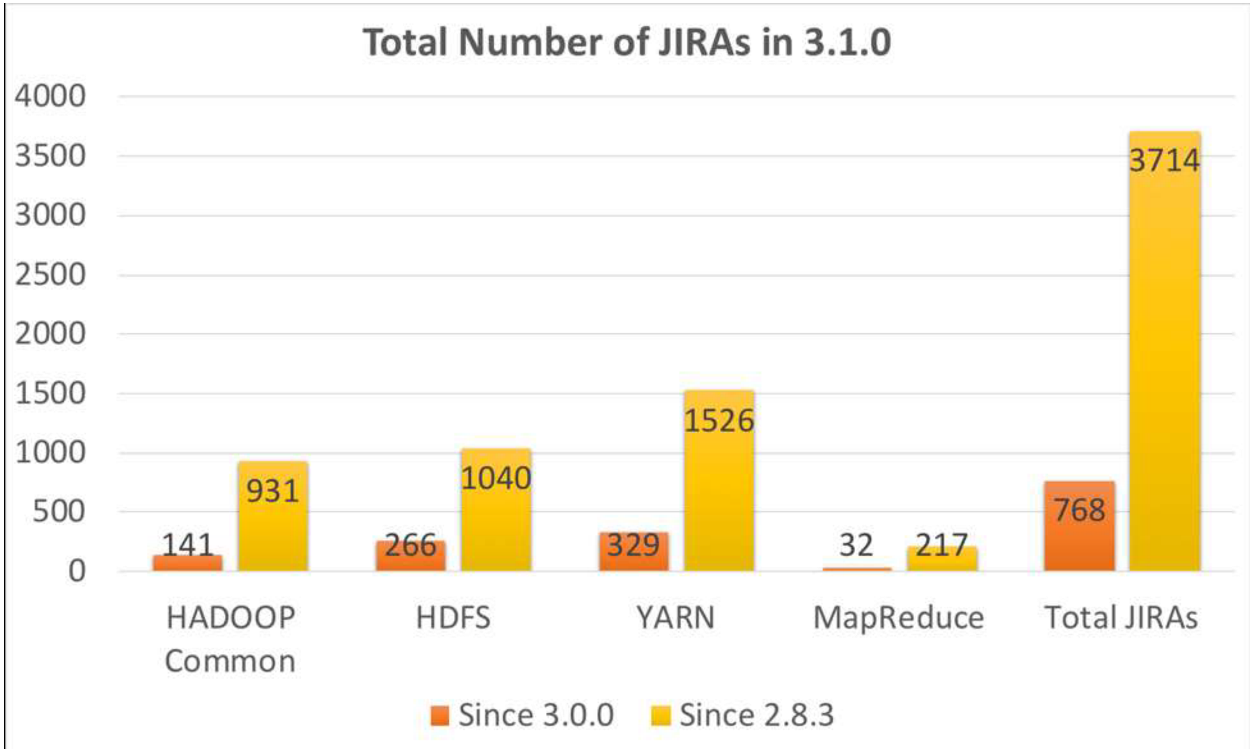
3.1.0: Apr’18

- GPU/FPGA
- YARN Native Service
- Placement Constraints

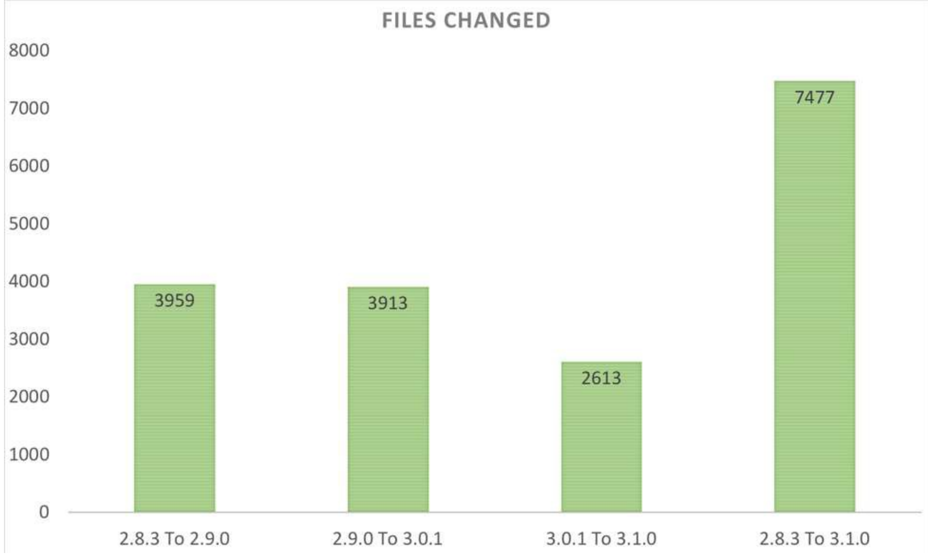
3.2.0: Sep’18

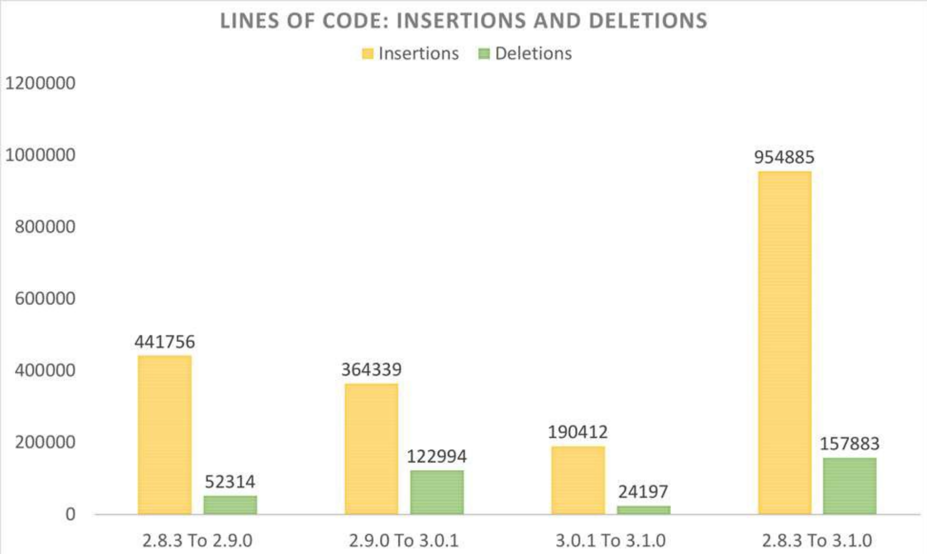
- Node Attributes
- Multi Node Scheduling
- Hadoop Submarine Project
- Support service upgrade

Community Update: JIRAs in 3.1.0

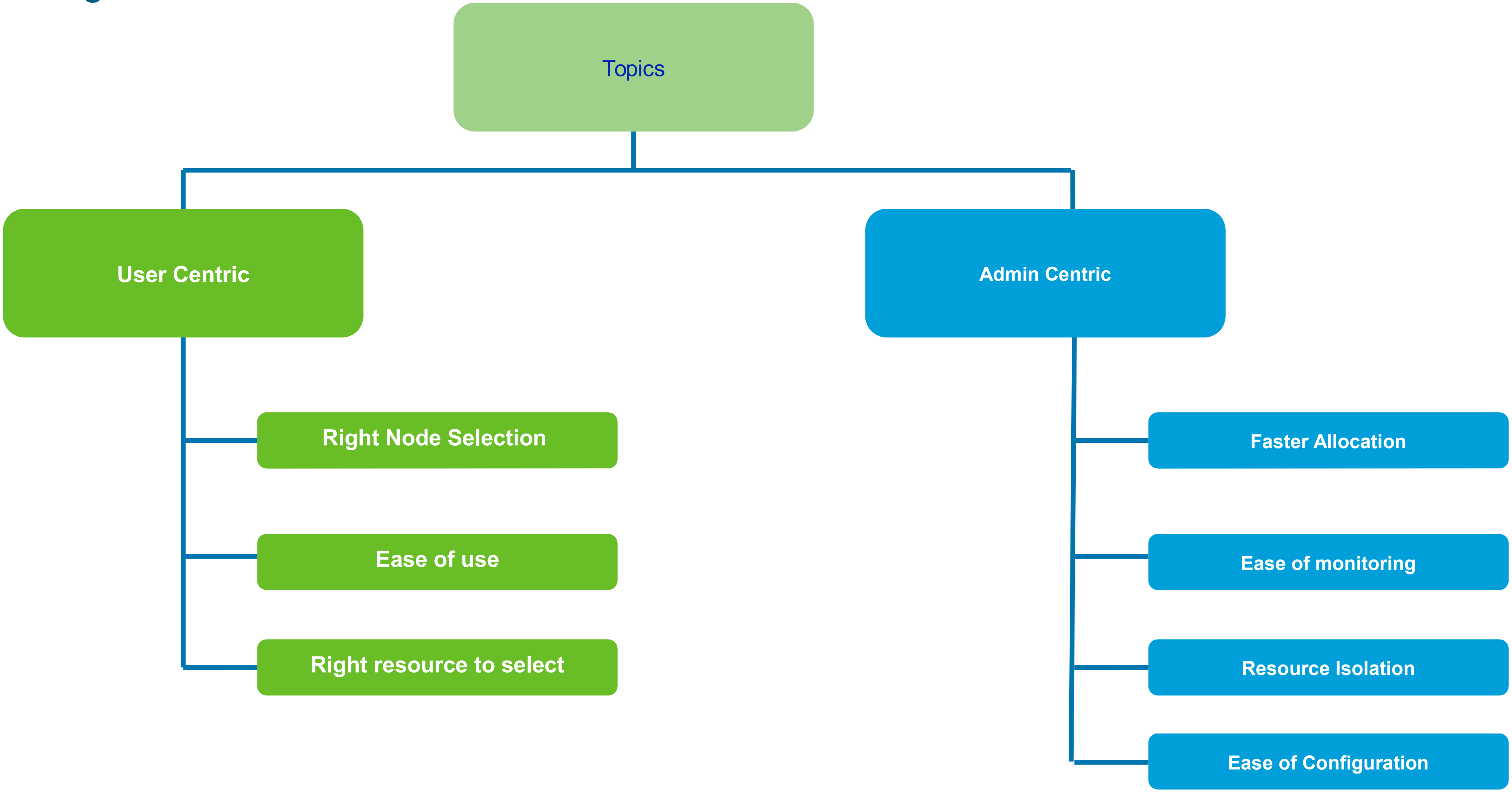


Community Update: Source code changes

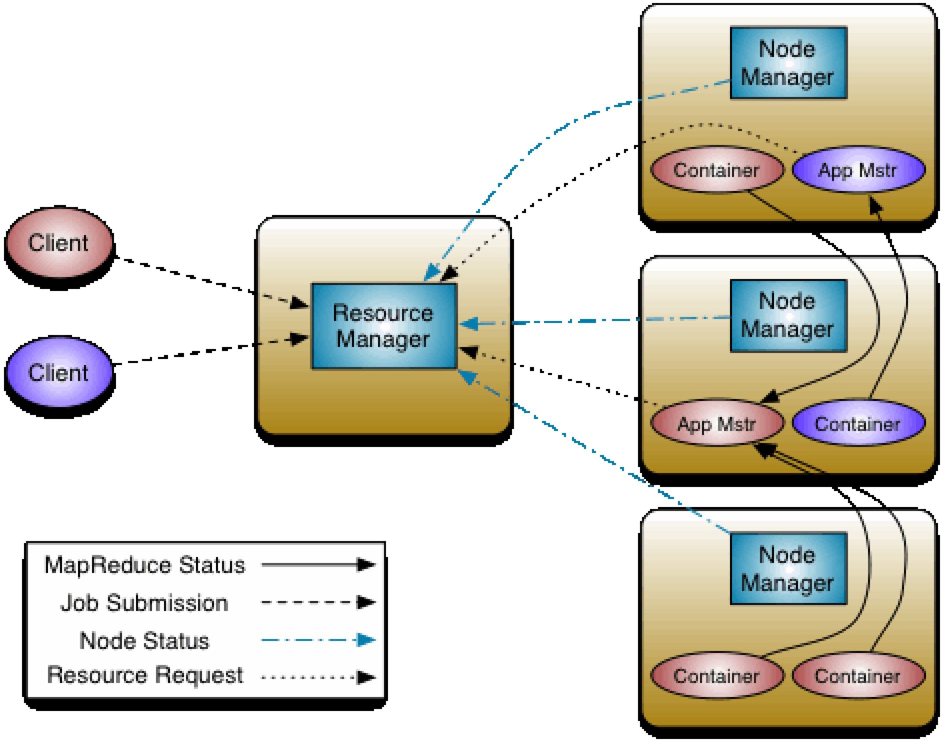




Categorization:



YARN Overview :



Apache Hadoop 3.1

Here for good

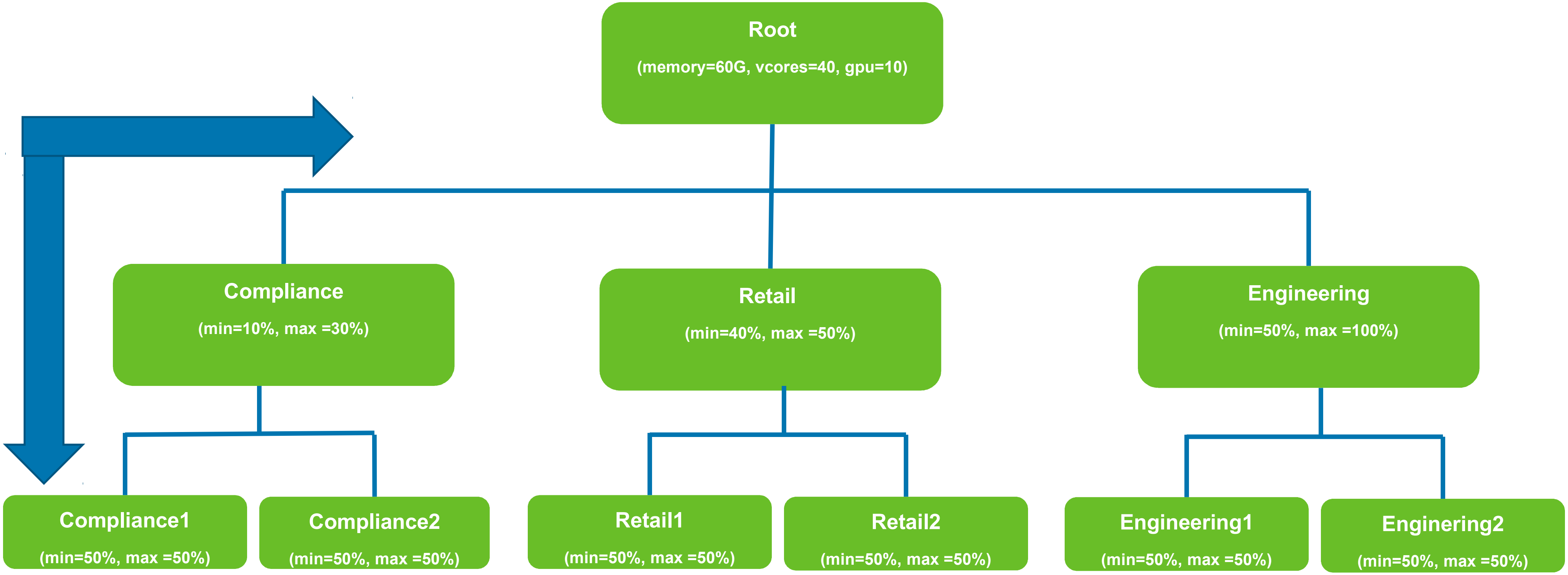
Moving towards Global & Fast Scheduling

Admin Centric

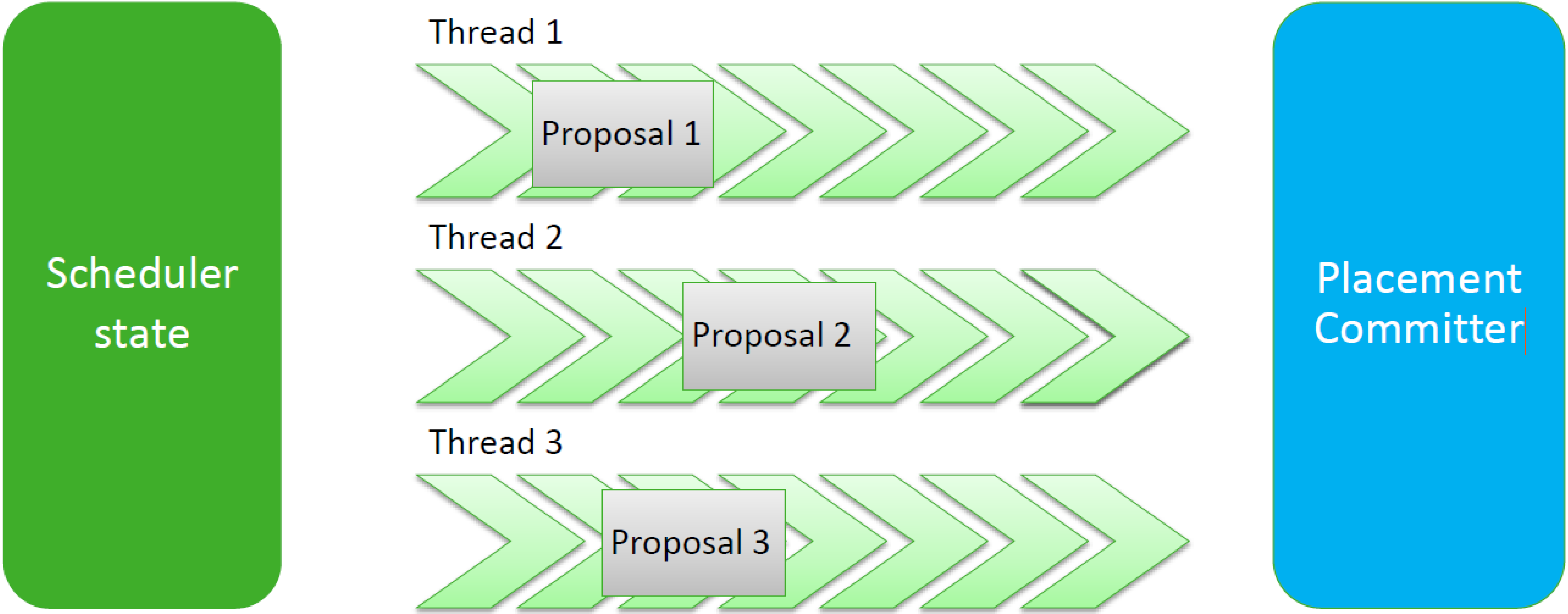
YARN-5139

- **Problems**
  - Current design of one-node-at-a-time allocation cycle can lead to suboptimal decisions.
  - Several coarse grained locks.
- **With this, we improved to**
  - Look at several nodes at a time
  - Fine grained locks
  - Multiple allocator threads
  - YARN scheduler can allocate 3k+ containers per second ≈ 10 mil allocations / hour!
  - **10X throughput gains**
  - Much better placement decisions

Traditional scheduling



Global Scheduling explained



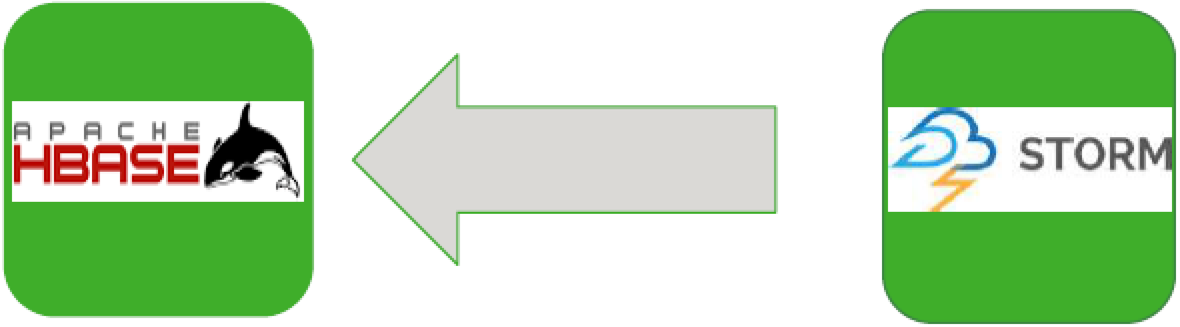
Better placement strategies (YARN-6592)

User Centric

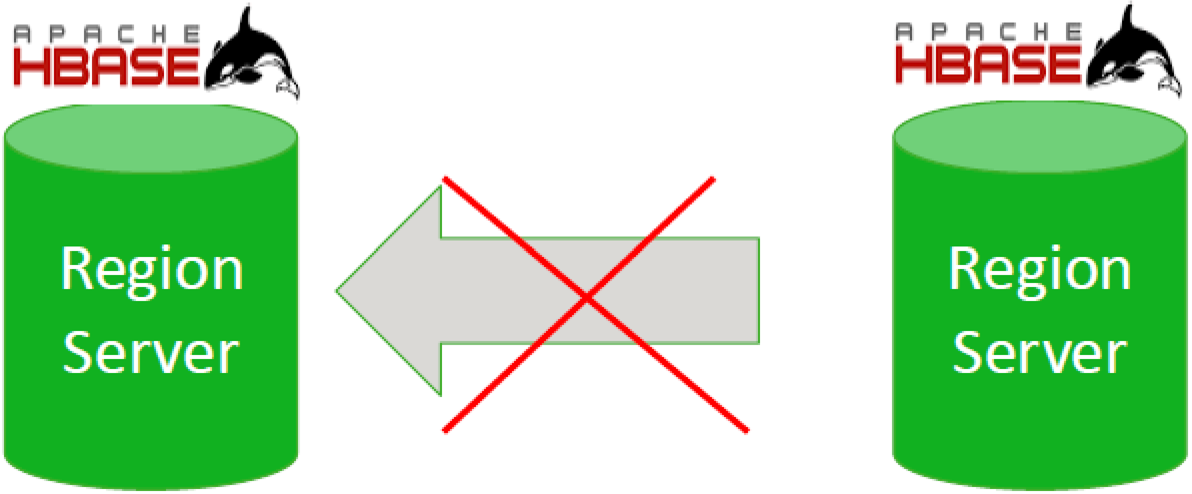
- **Past**
  - Supported constraints in form of Node Locality.
- **Now YARN can support a lot more use cases**
  - Co-locate the allocations of a job on the same rack (**affinity**)
  - Spread allocations across machines (**anti-affinity**) to minimize resource interference
  - Allow up to a specific number of allocations in a node group (**cardinality**)

Better placement strategies (YARN-6592)

- Affinity



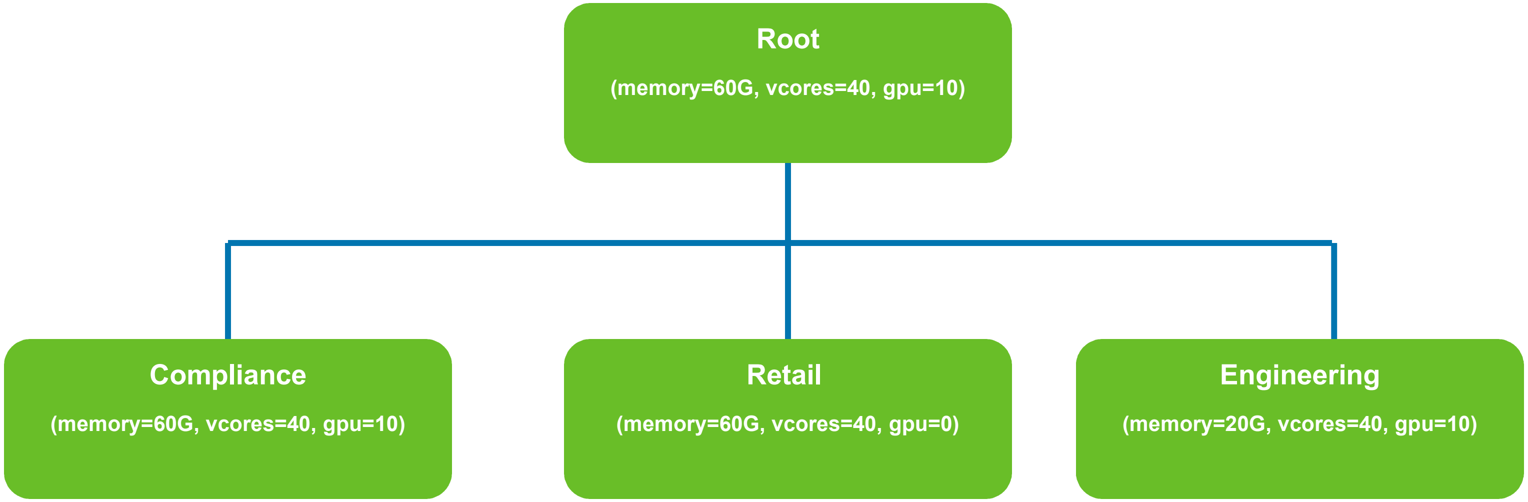
- Anti-affinit



Absolute Resources Configuration in CS – YARN-5881

Admin Centric

- Gives ability to configure Queue resources as below
  <memory=24GB, vcores=20, yarn.io/gpu=2>
- Enables admins to assign different quotas of different resource-types
- No more **“Single percentage value limitation for all resource-types”**


Root
(memory=60G, vcores=40, gpu=10)
Compliance
(memory=60G, vcores=40, gpu=10)
Retail
(memory=60G, vcores=40, gpu=0)
Engineering
(memory=20G, vcores=40, gpu=10)

Auto Creation of Leaf Queues - YARN-7117

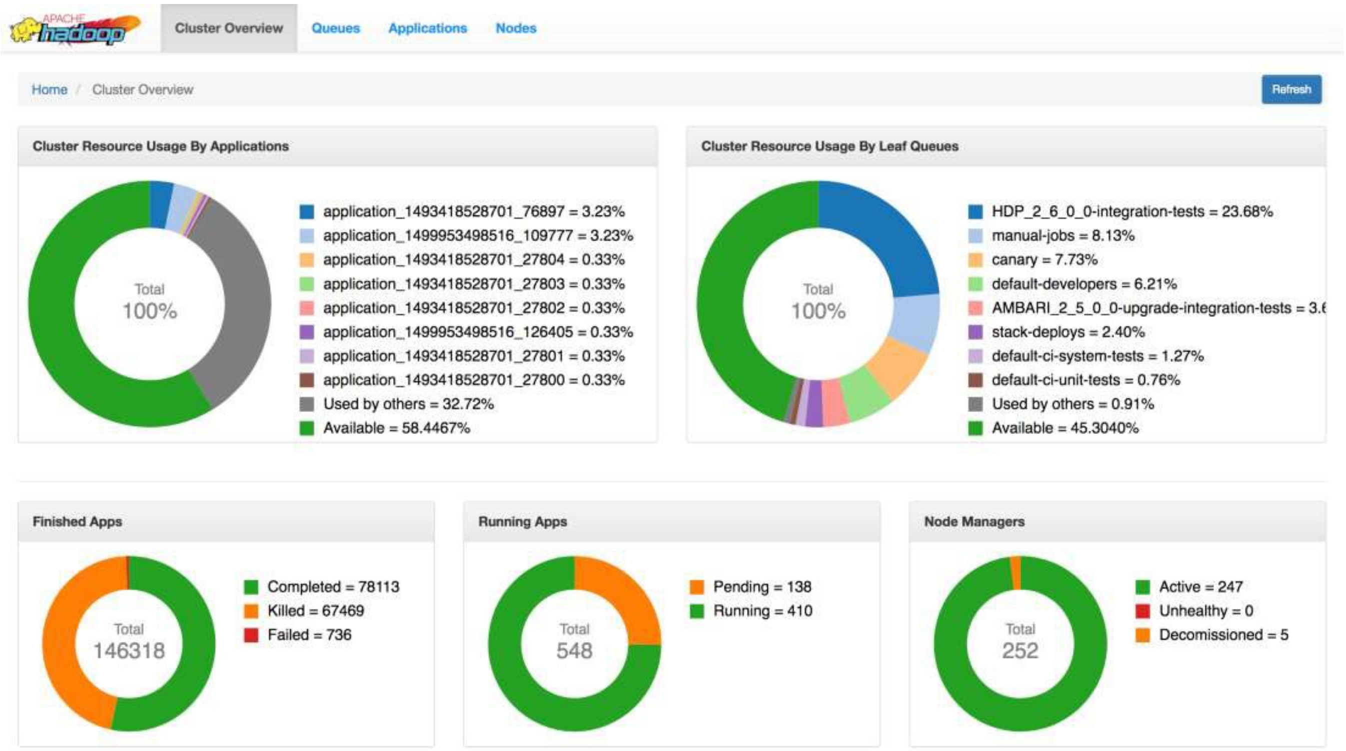
Admin Centric

- **Easily map a queue** explicitly to user or group with out additional configs
  - For e.g, User X comes in, automatically create a queue for user X with a templated capacity requirements

- **Auto created Queues will be**
  - created runtime based on user mapping
  - cleaned up after use
  - adhering to ACLs

Usability : UI 1/2

Admin Centric



Cluster Resource Usage By Applications

- application_1493418528701_76897 = 3.23%
- application_1499953498516_109777 = 3.23%
- application_1493418528701_27804 = 0.33%
- application_1493418528701_27803 = 0.33%
- application_1493418528701_27802 = 0.33%
- application_1499953498516_126405 = 0.33%
- application_1493418528701_27801 = 0.33%
- application_1493418528701_27800 = 0.33%
- Used by others = 32.72%
- Available = 58.4467%

Cluster Resource Usage By Leaf Queues

- HDP_2_6_0_0-integration-tests = 23.68%
- manual-jobs = 8.13%
- canary = 7.73%
- default-developers = 6.21%
- AMBARI_2_5_0_0-upgrade-integration-tests = 3.6
- stack-deploys = 2.40%
- default-ci-system-tests = 1.27%
- default-ci-unit-tests = 0.76%
- Used by others = 0.91%
- Available = 45.3040%

Finished Apps

- 🟩 Completed = 78113
- 🟧 Killed = 67469
- 🟥 Failed = 736

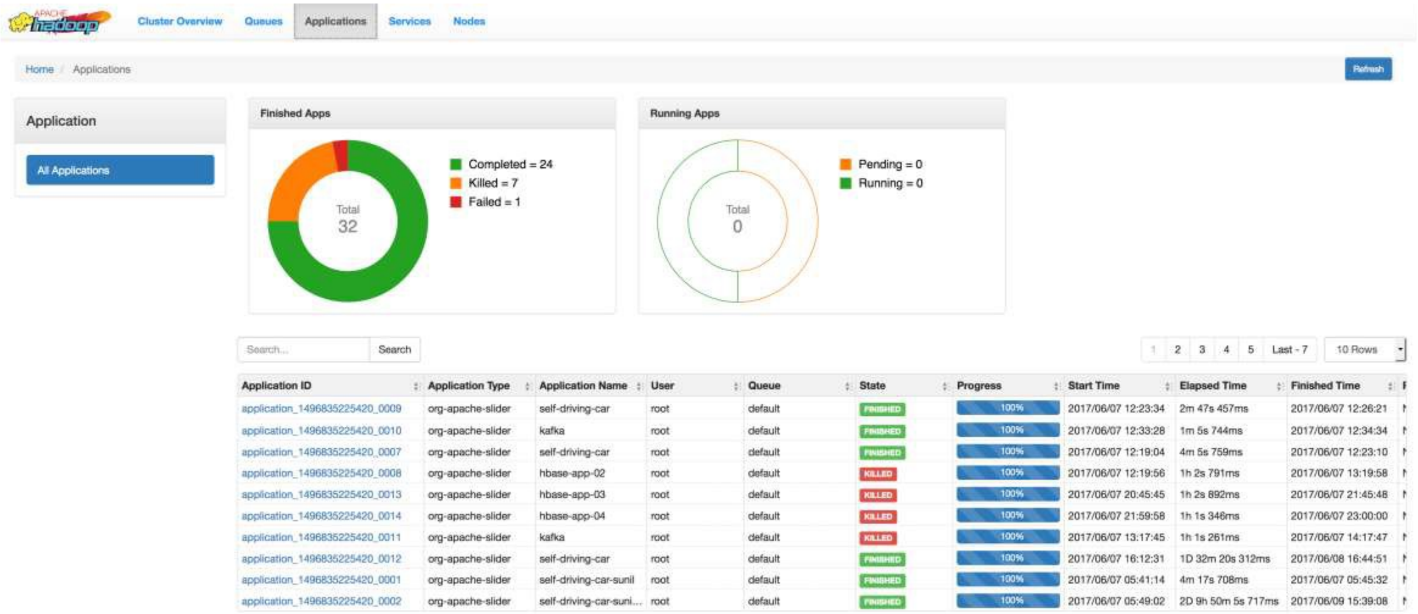
Running Apps

- 🟧 Pending = 138
- 🟩 Running = 410

Node Managers

- 🟩 Active = 247
- 🟥 Unhealthy = 0
- 🟧 Decomissioned = 5

Usability : UI 2/2



Finished Apps

- 🟩 Completed = 24
- 🟧 Killed = 7
- 🟥 Failed = 1

Running Apps

- 🟧 Pending = 0
- 🟩 Running = 0

Application ID	Application Type	Application Name	User	Queue	State	Progress	Start Time	Elapsed Time	Finished Time
application_1496835225420_0009	org-apache-slider	self-driving-car	root	default	FINISHED	100%	2017/06/07 12:23:34	2m 47s 457ms	2017/06/07 12:26:21
application_1496835225420_0010	org-apache-slider	kafka	root	default	FINISHED	100%	2017/06/07 12:33:28	1m 5s 744ms	2017/06/07 12:34:34
application_1496835225420_0007	org-apache-slider	self-driving-car	root	default	FINISHED	100%	2017/06/07 12:19:04	4m 5s 759ms	2017/06/07 12:23:10
application_1496835225420_0008	org-apache-slider	hbase-app-02	root	default	KILLED	100%	2017/06/07 12:19:56	1h 2s 791ms	2017/06/07 13:19:58
application_1496835225420_0013	org-apache-slider	hbase-app-03	root	default	KILLED	100%	2017/06/07 20:45:45	1h 2s 892ms	2017/06/07 21:45:48
application_1496835225420_0014	org-apache-slider	hbase-app-04	root	default	KILLED	100%	2017/06/07 21:59:58	1h 1s 346ms	2017/06/07 23:00:00
application_1496835225420_0011	org-apache-slider	kafka	root	default	KILLED	100%	2017/06/07 13:17:45	1h 1s 261ms	2017/06/07 14:17:47
application_1496835225420_0012	org-apache-slider	self-driving-car	root	default	FINISHED	100%	2017/06/07 16:12:31	1D 32m 20s 312ms	2017/06/08 16:44:51
application_1496835225420_0001	org-apache-slider	self-driving-car-sunil	root	default	FINISHED	100%	2017/06/07 05:41:14	4m 17s 708ms	2017/06/07 05:45:32
application_1496835225420_0002	org-apache-slider	self-driving-car-suni...	root	default	FINISHED	100%	2017/06/07 05:49:02	2D 9h 50m 5s 717ms	2017/06/09 15:39:08

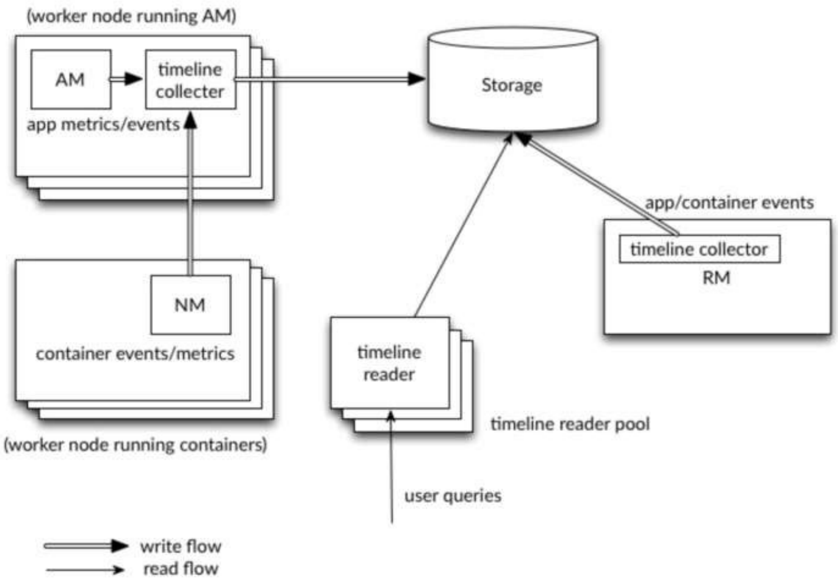
Timeline Service 2.0 Improvements

User Centric

Admin Centric

Understanding and Monitoring a Hadoop cluster itself is a BigData problem

- Using **HBase as backend** for better scalability for read/write
- More robust storage fault tolerance
- Migration and compatibility with v.1.5

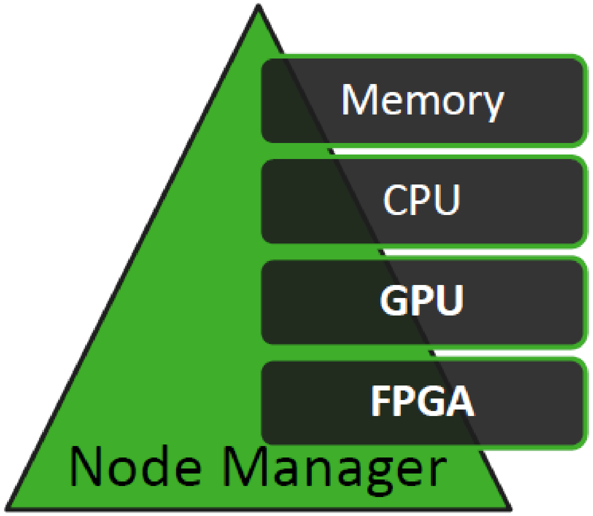


Resource profiles and custom resource types

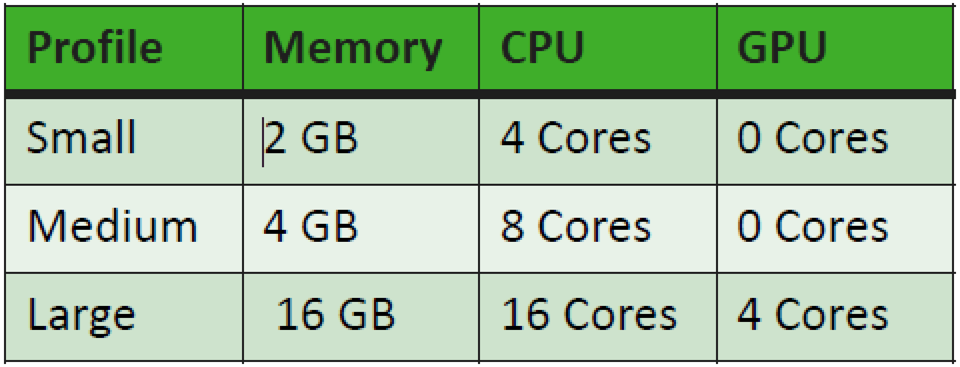
User Centric

Admin Centric

- **YARN** supported only **Memory** and **CPU**
- **Now**
  - A generalized vector for all resources
  - Admin could add arbitrary resource types!



Ease of resource requesting model using profiles for apps

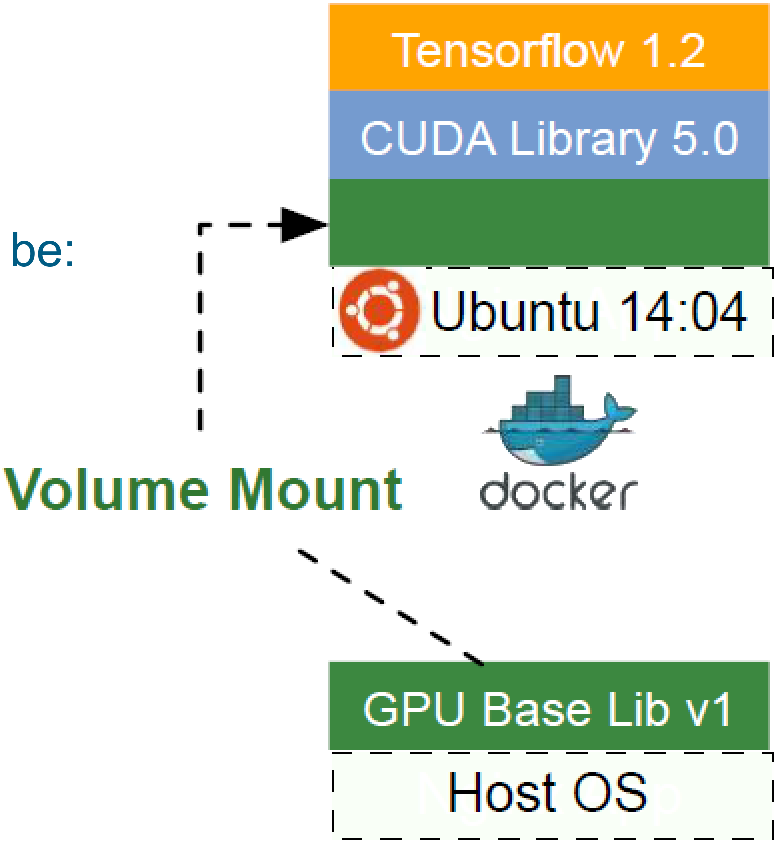


Profile	Memory	CPU	GPU
Small	2 GB	4 Cores	0 Cores
Medium	4 GB	8 Cores	0 Cores
Large	16 GB	16 Cores	4 Cores

GPU support on YARN

User Centric

- **Why?**
  - No need to setup separate clusters
  - Leverage shared compute!
- **Why need isolation?**
  - Multiple processes use the single GPU will be:
    - Serialized.
    - Cause OOM easily.
- **GPU isolation on YARN:**
  - Granularity is for per-GPU device.
  - Use cgroups / docker to enforce isolation.



FPGA on YARN

User Centric

- **FPGA isolation on YARN:**
  - Granularity is for per-FPGA device
  - Use Cgroups to enforce the isolation
- Currently, only Intel OpenCL SDK for FPGA is supported.
- Implementation is extensible to other FPGA SDK.

Services support in YARN

User Centric

- **A native YARN services framework (YARN-5079)**
  - Native Yarn support to Services
  - Apache Slider retired from Incubator – lessons and key code carried over to YARN

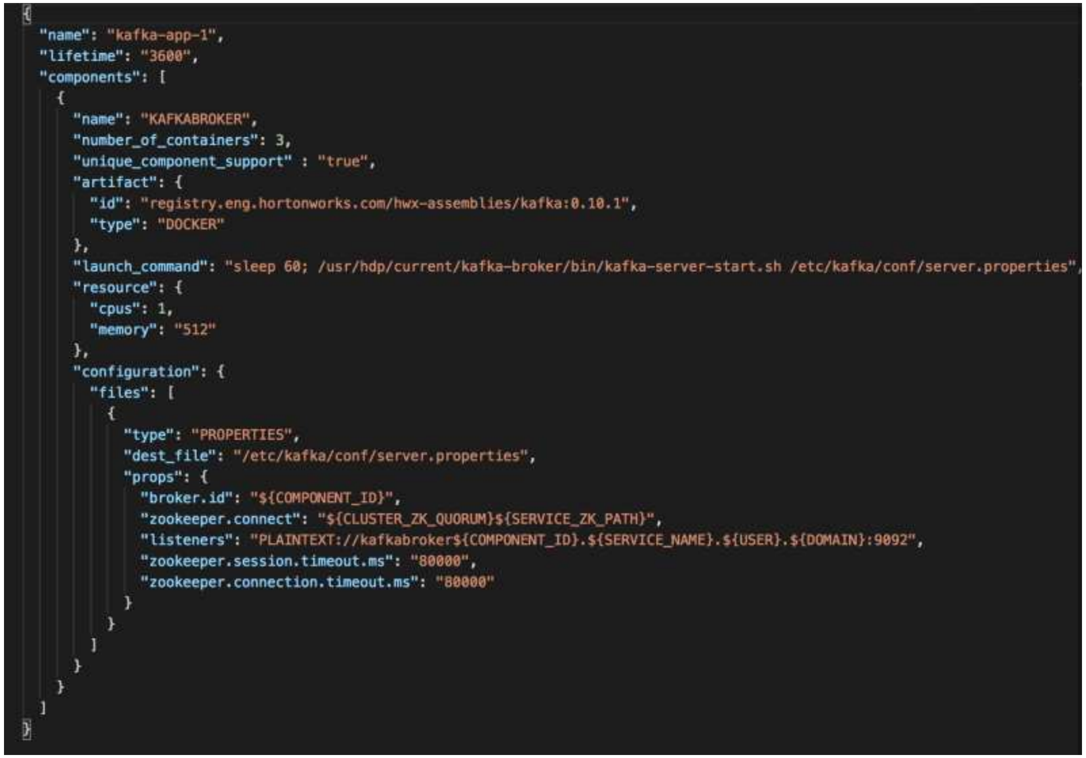
- **Simplified discovery of services via DNS mechanisms: YARN-4757**
  - *regionserver-0.hbase-app-3.hadoop.yarn.site*

- **Application & Services upgrades: YARN-4726**
  - “Do an upgrade of my HBase app with minimal impact to end-users”.

Simplified APIs for service definitions

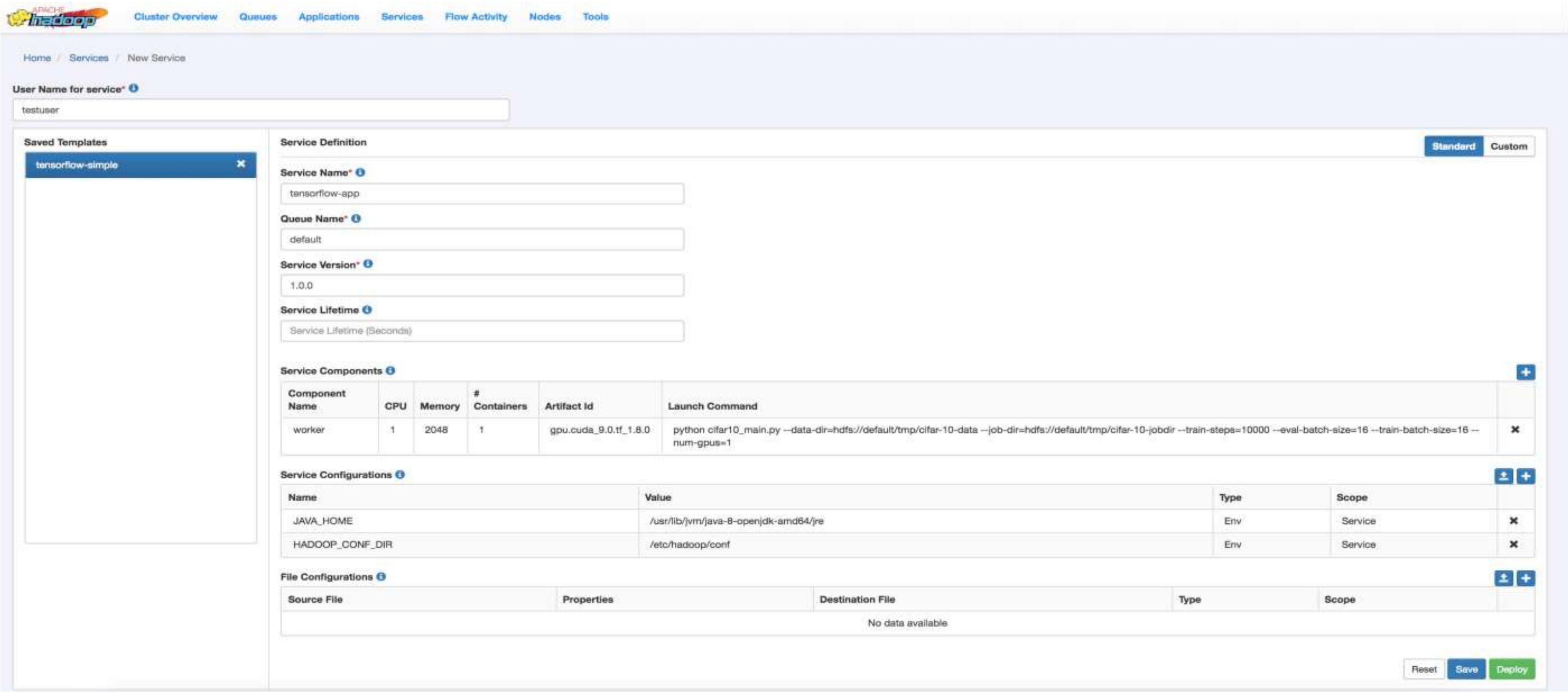
User Centric

- Applications need simple APIs
- Need to be deployable “easily”
- Simple REST API layer (YARN-4793)
- Spawn services & Manage them



```
{
  "name": "kafka-app-1",
  "lifetime": "3600",
  "components": [
    {
      "name": "KAFKABROKER",
      "number_of_containers": 3,
      "unique_component_support" : "true",
      "artifact": {
        "id": "registry.eng.hortonworks.com/hwx-assemblies/kafka:0.10.1",
        "type": "DOCKER"
      },
      "launch_command": "sleep 60; /usr/hdp/current/kafka-broker/bin/kafka-server-start.sh /etc/kafka/conf/server.properties",
      "resource": {
        "cpus": 1,
        "memory": "512"
      },
      "configuration": {
        "files": [
          {
            "type": "PROPERTIES",
            "dest_file": "/etc/kafka/conf/server.properties",
            "props": {
              "broker.id": "${COMPONENT_ID}",
              "zookeeper.connect": "${CLUSTER_ZK_QUORUM}${SERVICE_ZK_PATH}",
              "listeners": "PLAINTEXT://kafkabroker${COMPONENT_ID}.${SERVICE_NAME}.${USER}.${DOMAIN}:9092",
              "zookeeper.session.timeout.ms": "80000",
              "zookeeper.connection.timeout.ms": "80000"
            }
          }
        ]
      }
    }
  ]
}
```

How to run a new service in YARN ?



Home / Services / New Service

User Name for service*

testuser

Saved Templates

tensorflow-simple ✖

Service Definition

Service Name*

tensorflow-app

Queue Name*

default

Service Version*

1.0.0

Service Lifetime

Service Lifetime (Seconds)

Service Components

Component Name	CPU	Memory	# Containers	Artifact Id	Launch Command	
worker	1	2048	1	gpu.cuda_9.0.tf_1.8.0	python cifar10_main.py --data-dir=hdfs://default/tmp/cifar-10-data --job-dir=hdfs://default/tmp/cifar-10-jobdir --train-steps=10000 --eval-batch-size=16 --train-batch-size=16 --num-gpus=1	✖

Service Configurations

Name	Value	Type	Scope	
JAVA_HOME	/usr/lib/jvm/java-8-openjdk-amd64/jre	Env	Service	✖
HADOOP_CONF_DIR	/etc/hadoop/conf	Env	Service	✖

File Configurations

Source File	Properties	Destination File	Type	Scope
No data available				

Apache Hadoop 3.2

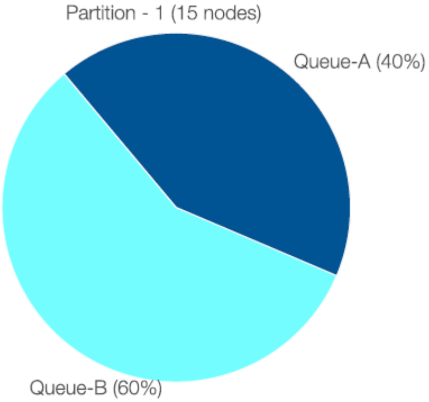
Here for good

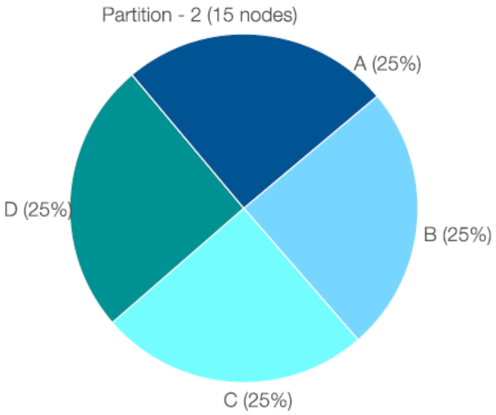
Node Attributes (YARN-3409)

User Centric

- “Take me to a node with JDK 10”

- Node Partition vs. Node Attribute
- **Node Partition**
  - One partition for one node
  - ACL
  - Shares between queues
  - Pre emption enforced
- **Attribute**
  - For Container placement
  - No ACL’s and Shares
  - First come first serve





Node 1
os.type=ubuntu
os.version=14.10
glibc.version=2.20
JDK.version=8u20

Node 2
os.type=RHEL
os.version=5.1
GPU.type=x86_64
JDK.version=7u20

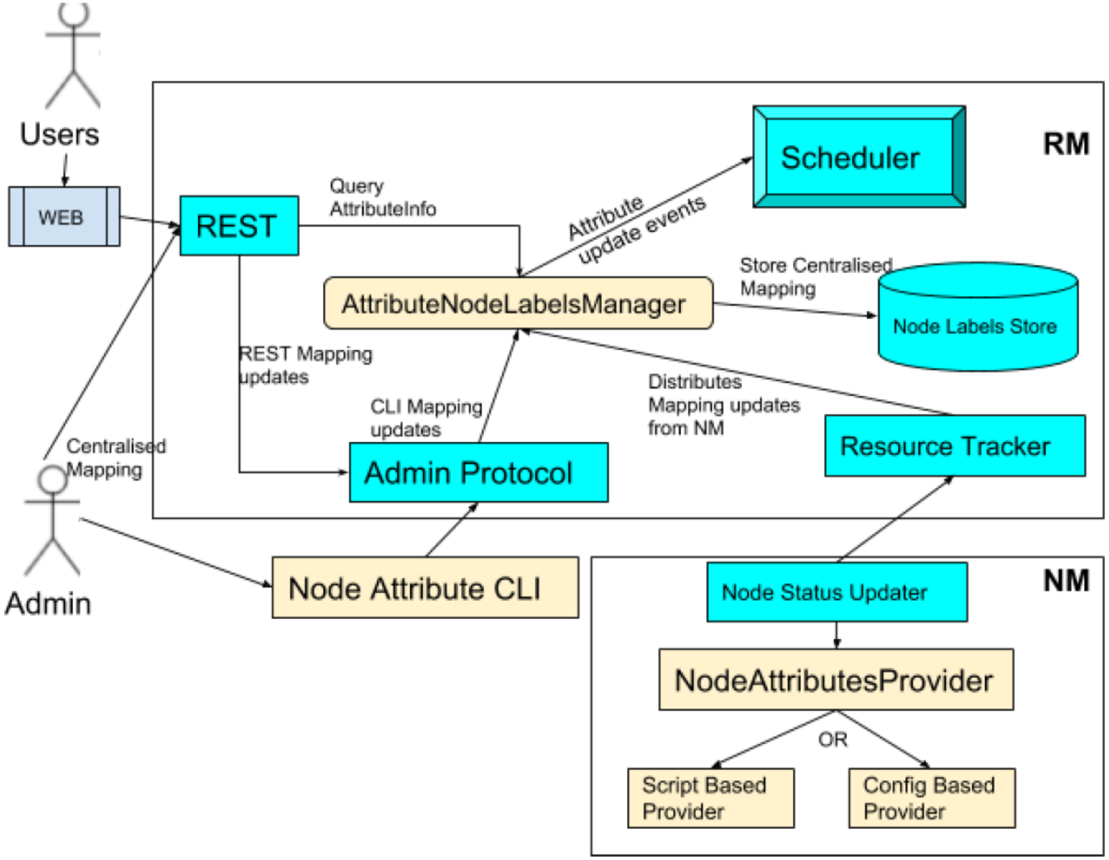
Node 16
os.type=windows
os.version=7
JDK.version=8u20

Node 17
os.type=SUSE
os.version=12
GPU.type=i686
JDK.version=7u20

Node Attributes (YARN-3409)

User Centric

- **Distributed Node Attributes**
  - NM can detect its attributes
  - Script based and Config based detection.
  - Attribute prefix : *yarn.nm.io*
- **Centralised Node Attributes**
  - Configured through CLI and REST
  - Admin ACL's to configure
  - Attribute prefix : *yarn.rm.io*



Node Attributes (YARN-3409)

User Centric

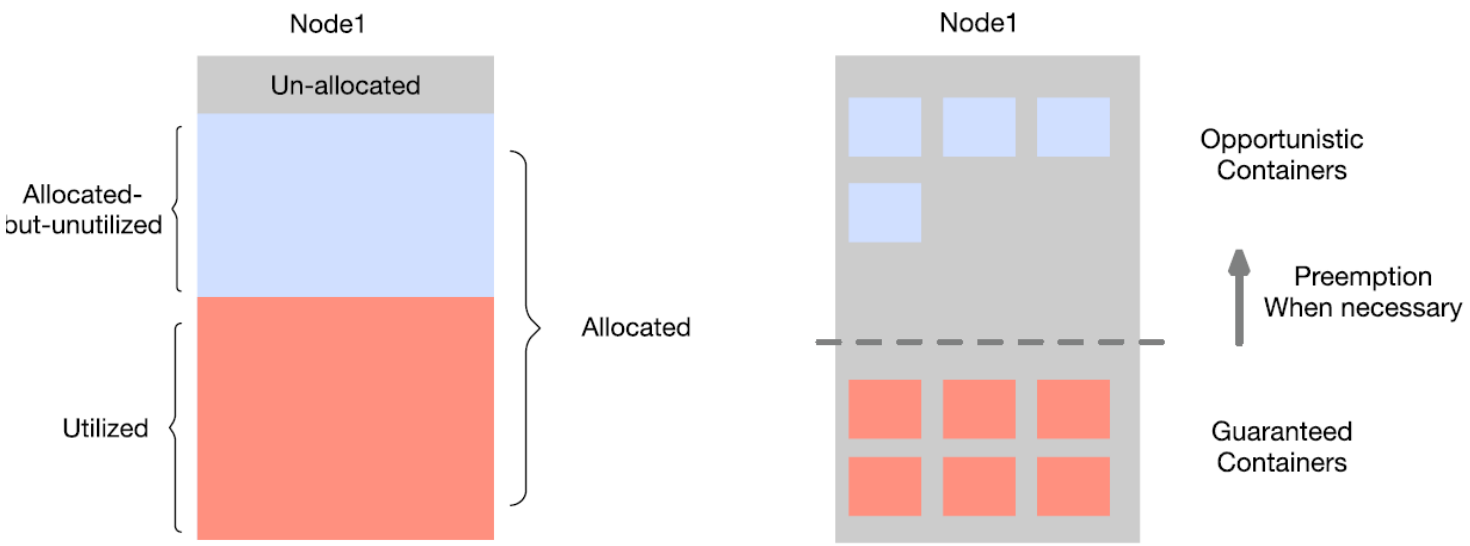
Use cases :

- **Hardware Constraints** : To identify specific kind of resources like
  - GPU, FPGA,
  - SSD, # of disks,
  - InfiniBand,
  - (dual) network cards,
- **Task Constraints:** Task or container specific constraints like
  - To run on specific Operating system versions.
  - Processor architecture
  - software library versions
- **Experimental** : Based on dynamic attributes like
  - Load average,
  - disk usage
  - Network

Container overcommit (YARN-1011)

Admin Centric

- Every user says “Give me 16GB for my task”, even though it’s only needed at peak
- Each node has some allocated but unutilized capacity. Use such capacity to run opportunistic tasks
- Preempt such tasks when needed

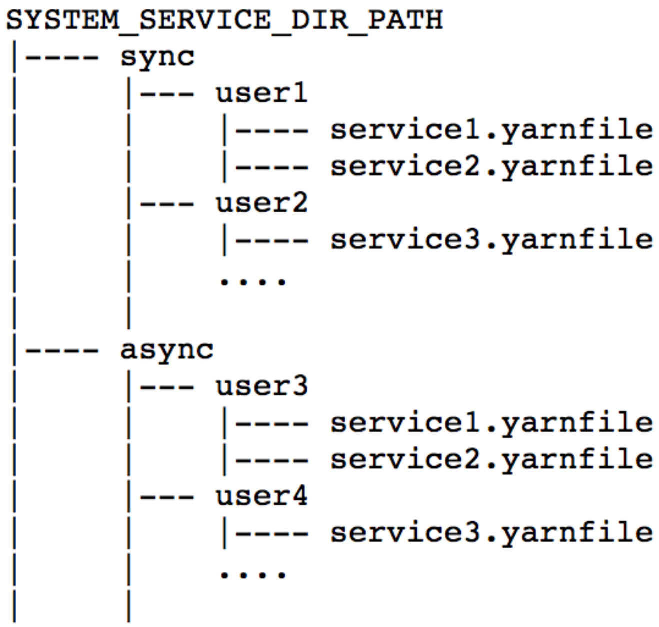


Auto-spawning of system services (YARN-8048)

Admin Centric

- “Start this service when YARN starts”

- “initd for YARN”
- Services are started during the yarn bootstrap
  - For example YARN ATSv2 needs Hbase, so Hbase is system service of YARN.
  - Only Admin can configure
  - Started along with ResourceManager
  - Place spec files under yarn.service.systemservice.dir FS path



```
SYSTEM_SERVICE_DIR_PATH
|---- sync
|          |--- user1
|          |        |---- service1.yarnfile
|          |        |---- service2.yarnfile
|          |--- user2
|          |        |---- service3.yarnfile
|          |        ....
|          |
|---- async
|          |--- user3
|          |        |---- service1.yarnfile
|          |        |---- service2.yarnfile
|          |--- user4
|          |        |---- service3.yarnfile
|          |        ....
|          |
```

TensorFlow on YARN (YARN-8220)

- Run deep learning workloads on the same cluster as analytics, stream processing etc!

- Integrated with latest TensorFlow 1.8 and has **GPU** support
  - Use simple command to run TensorFlow app by using Native Service spec file (Yarnfile)

    yarn app -launch distributed-tf <path-to-saved-yarnfile>
  - A simple python command line utility also could be used to auto-create Yarnfile

```
python submit_tf_job.py
--remote_conf_path hdfs:///tf-job-conf
--input_spec example_tf_job_spec.json
--docker_image gpu.cuda_9.0.tf_1.8.0
--job_name distributed-tf-gpu
--user tf-user
--domain tensorflow.site
--distributed --kerberos
```

TensorFlow on YARN (YARN-8220)

- Run deep learning workloads on the same cluster as analytics, stream processing etc!

- Integrated with latest TensorFlow 1.8 and has **GPU** support
  - Use simple command to run TensorFlow app by using Native Service spec file (Yarnfile)

    ```
    yarn app -launch distributed-tf <path-to-saved-yarnfile>
    ```

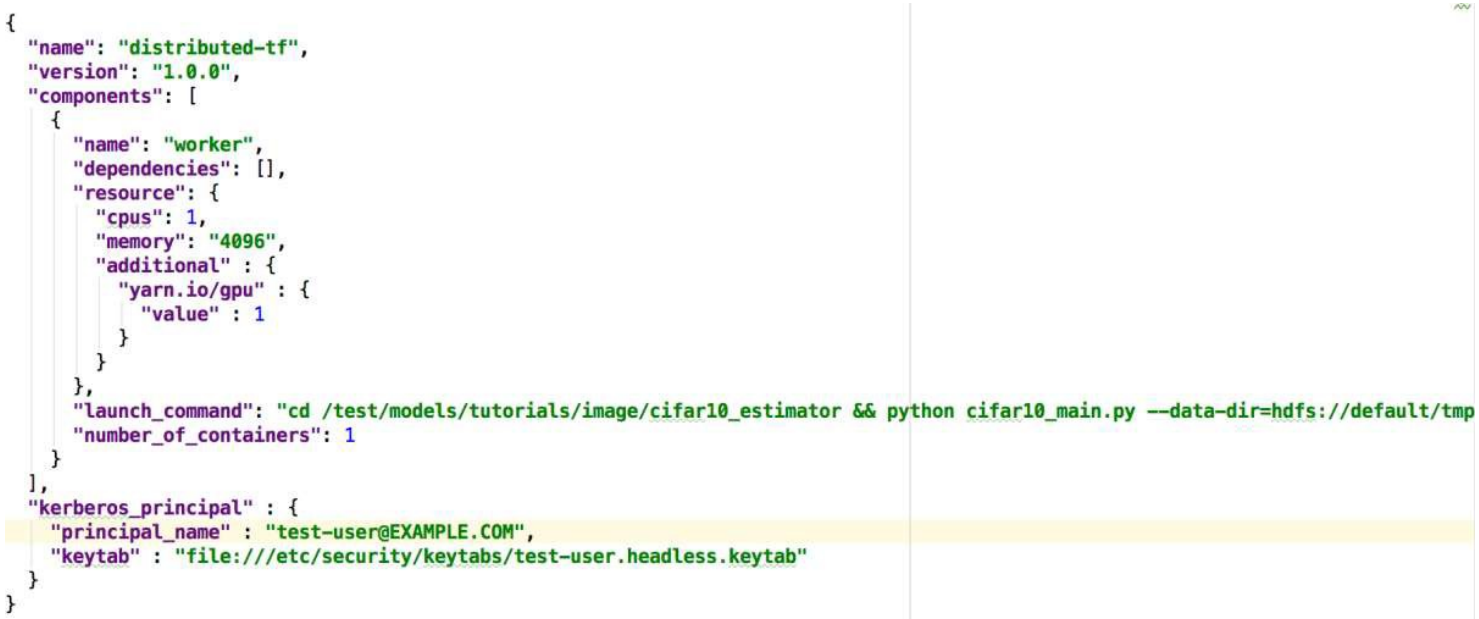
  - A simple python command line utility also could be used to auto-create Yarnfile

    ```
    python submit_tf_job.py
    --remote_conf_path hdfs:///tf-job-conf
    --input_spec example_tf_job_spec.json
    --docker_image gpu.cuda_9.0.tf_1.8.0
    --job_name distributed-tf-gpu
    --user tf-user
    --domain tensorflow.site
    --distributed --kerberos
    ```

TensorFlow on YARN (YARN-8220)

User Centric

- **Sample Yarnfile for TensorFlow job**



```
{
  "name": "distributed-tf",
  "version": "1.0.0",
  "components": [
    {
      "name": "worker",
      "dependencies": [],
      "resource": {
        "cpus": 1,
        "memory": "4096",
        "additional" : {
          "yarn.io/gpu" : {
            "value" : 1
          }
        }
      },
      "launch_command": "cd /test/models/tutorials/image/cifar10_estimator && python cifar10_main.py --data-dir=hdfs://default/tmp
      "number_of_containers": 1
    }
  ],
  "kerberos_principal" : {
    "principal_name" : "test-user@EXAMPLE.COM",
    "keytab" : "file:///etc/security/keytabs/test-user.headless.keytab"
  }
}
```

Other related talks :

- **Deep learning on YARN - Running distributed Tensorflow / MXNet / Caffe / XGBoost on Hadoop clusters**
    - **Speakers** : Wangda Tan
    - Thursday, 27th Sep, 11:20
    - Ballroom

- **Running distributed TensorFlow in production: challenges and solutions on YARN 3.0**
    - **Speakers** : Wangda Tan & Yanbo Liang
    - Thursday, 27th Sep, 15:40
    - Ballroom

Queries ?