Apache Hadoop YARN: The Next-generation Distributed Operating System

Zhijie Shen & Jian He @ Hortonworks

About Us

- Software Engineer @ Hortonworks, Inc.
- Hadoop Committer @ The Apache Foundation

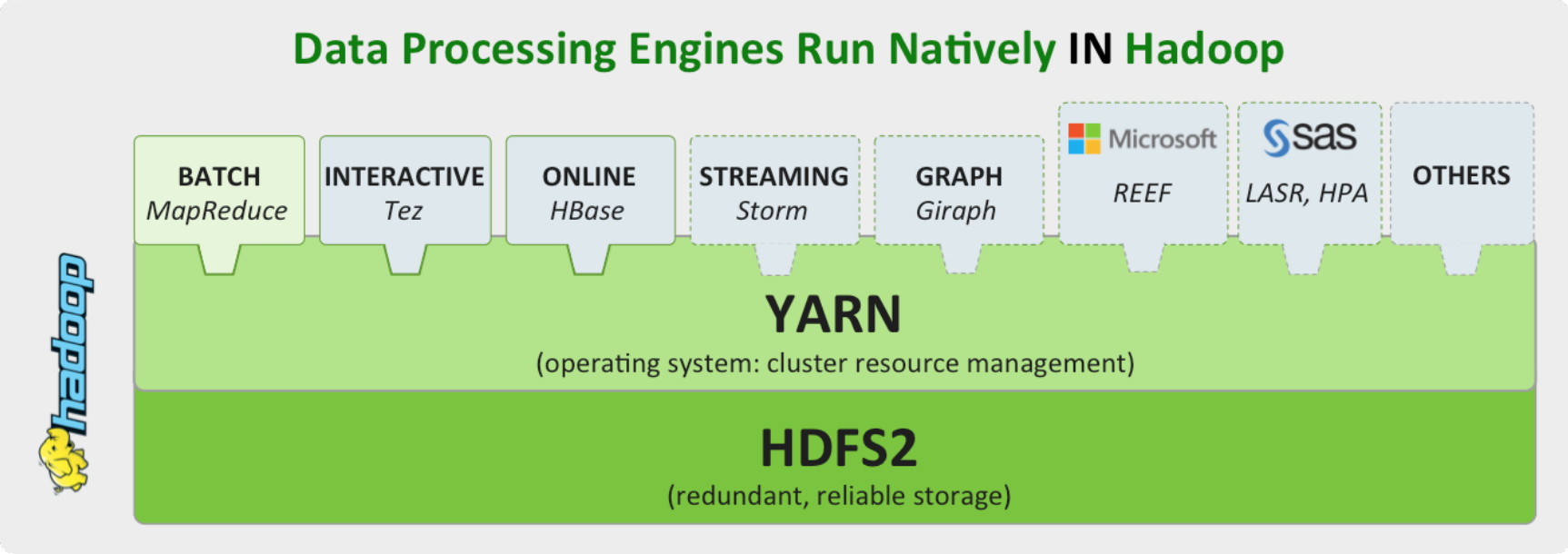
- We’re doing YARN!

Hortonworks

Agenda

- What Is YARN
- YARN Framework
- Recent Development
- Writing Your YARN Applications

What Is YARN (1)



Hortonworks

What Is YARN (2)

Motivation:

- Flexibility - Enabling data processing model more than MapReduce
- Efficiency - Improving performance and QoS
- Resource Sharing - Multiple workloads in cluster

Agenda

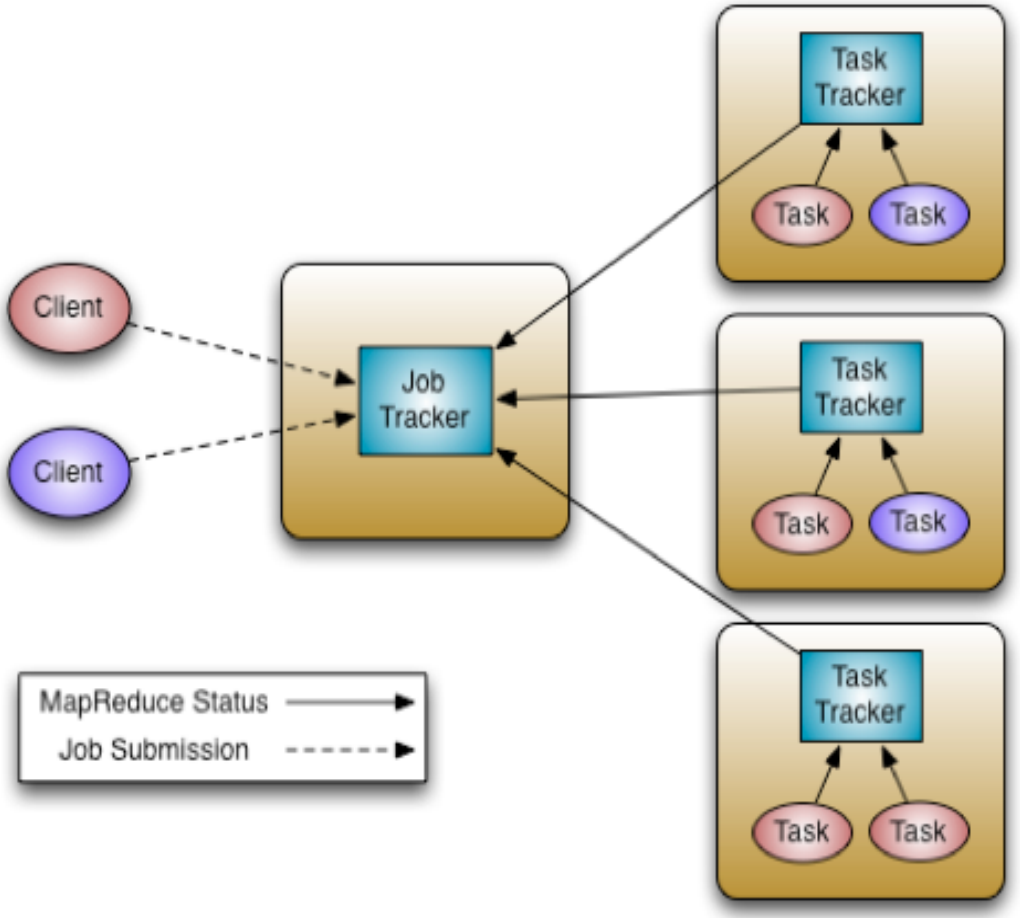
- What Is YARN
- YARN Framework
- Recent Development
- Writing Your YARN Applications

Hortonworks

YARN Framework (1)

JobTracker-TaskTracker

- MapReduce Only
- Scalability
  - **2009 – 8 cores, 16GB of RAM, 4x1TB disk**
  - **2012 – 16+ cores, 48-96GB of RAM, 12x2TB or 12x3TB of disk**
- Poor Cluster Utilization
  - **distinct map slots and reduce slots**



Hortonworks

YARN Framework (2)

RescourceManager:

Arbitrates resources among all the applications in the system

NodeManager:

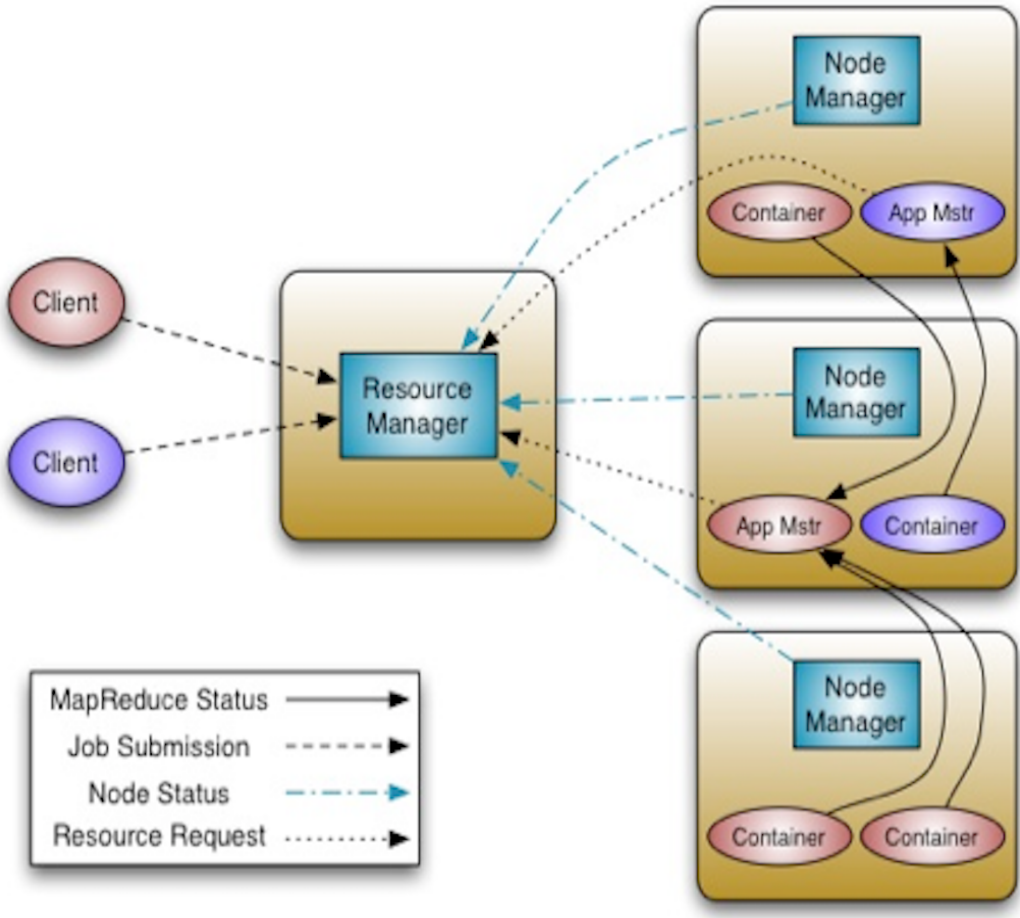
the per-machine slave, which is responsible for launching the applications' containers, monitoring their resource usage

ApplicationMaster:

Negatiate appropriate resource containers from the Scheduler, tracking their status and monitoring for progress

Container:

Unit of allocation incorporating resource elements such as memory, cpu, disk, network etc, to execute a specific task of the application (similar to map/reduce slots in MRv1)

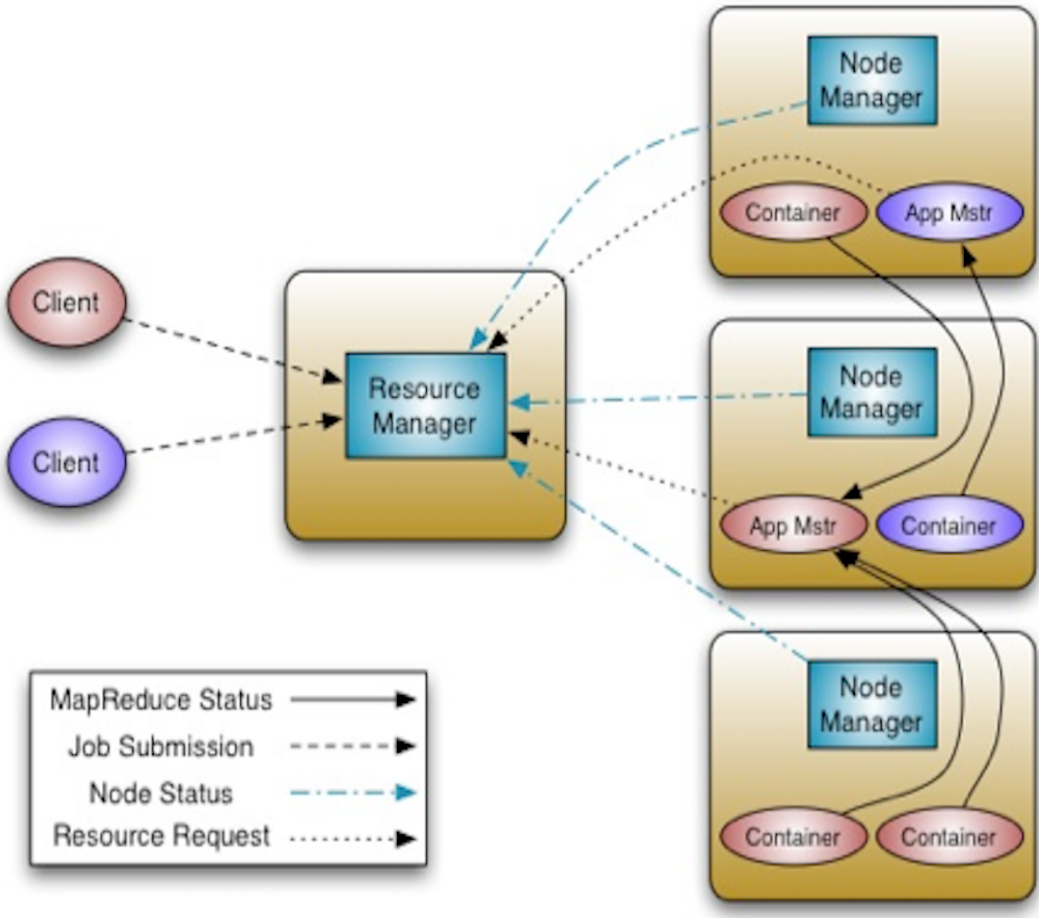


Hortonworks

YARN Framework (3)

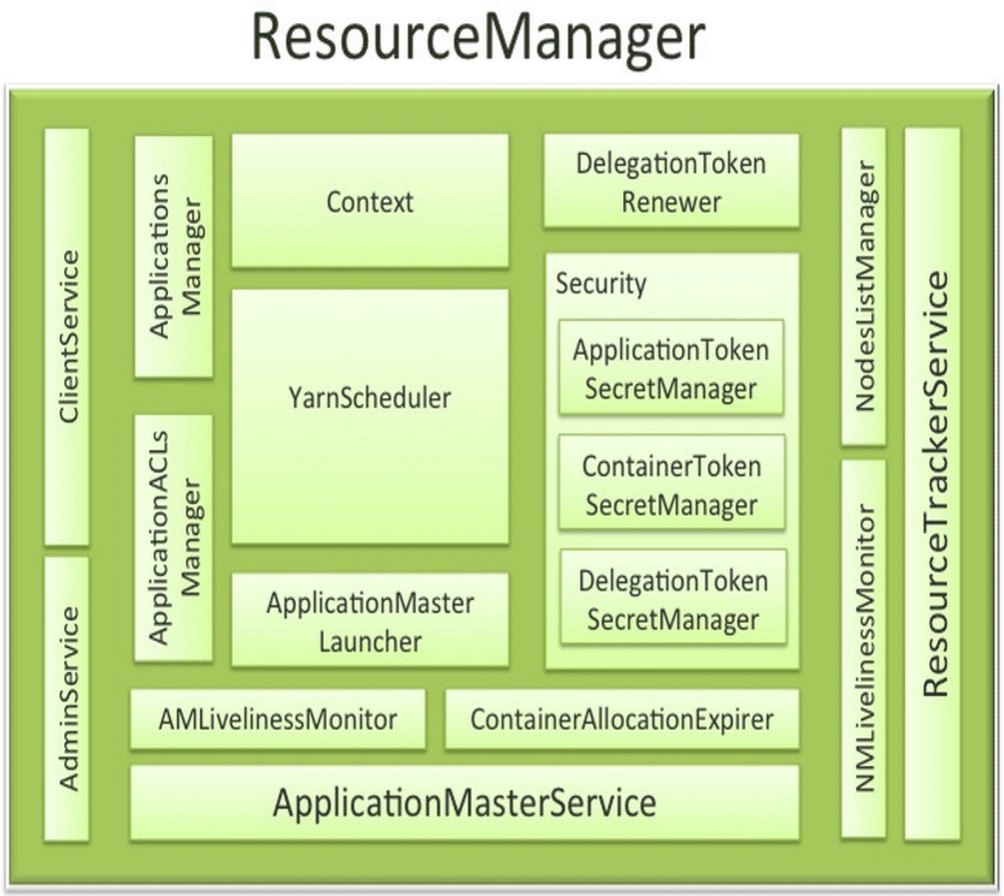
Execution Sequence:

1. A client program submits the application
2. ResourceManager allocates a specified container to start the ApplicationMaster
3. ApplicationMaster, on boot-up, registers with ResourceManager
4. ApplicationMaster negotiates with ResourceManager for appropriate resource containers
5. On successful container allocations, ApplicationMaster contacts NodeManager to launch the container
6. Application code is executed within the container, and then ApplicationMaster is responded with the execution status
7. During execution, the client communicates directly with ApplicationMaster or ResourceManager to get status, progress updates etc.
8. Once the application is complete, ApplicationMaster unregisters with ResourceManager and shuts down, allowing its own container process



Hortonworks

YARN Framework (4)



Components interfacing RM to the clients:

- ClientRMService
- AdminService

Components interacting with the per-application AMs:

- ApplicationMasterService

Components connecting RM to the nodes:

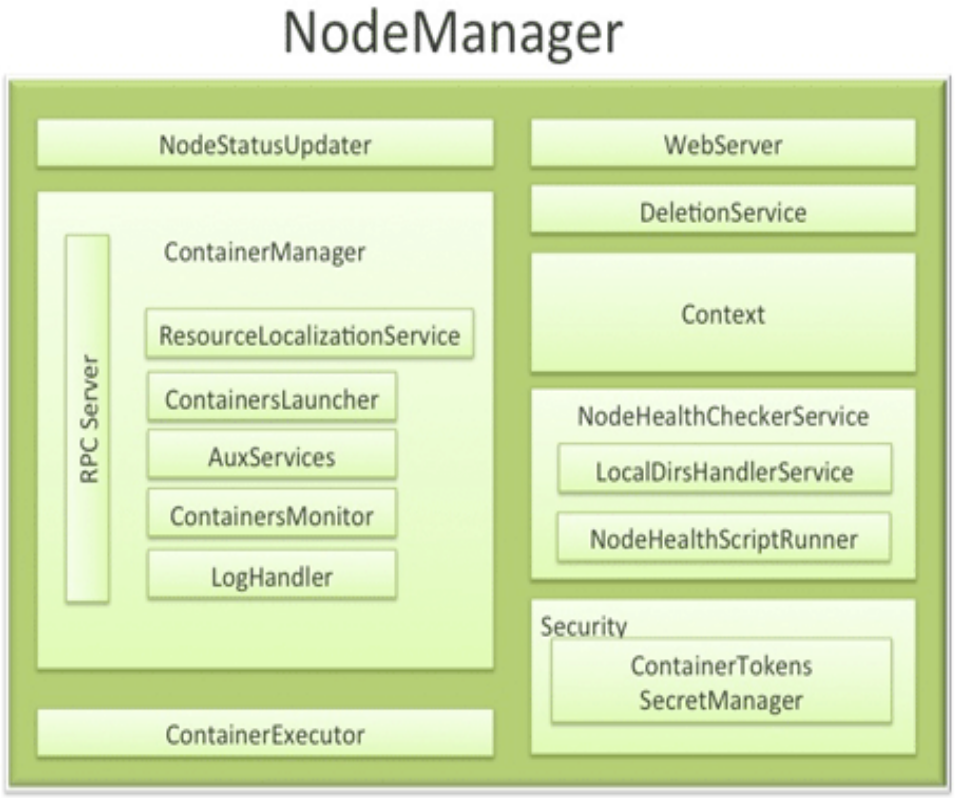
- ResourceTrackerService

Core of the ResourceManager

- ApplicationsManager
- Scheduler
- Security

Hortonworks

YARN Framework (5)



Component for NM-RM communication:

- NodeStatusUpdater

Core component managing containers on the node:

- ContainerManager

Component monitoring node health:

- NodeHealthCheckService

Component interacting with OS to start/stop the container process:

- ContainerExecutor

ACL and Token verification:

- Security

Hortonworks

YARN Framework (7)

Scheduler

- FIFO
- Fair
- Capcity

Agenda

- What Is YARN
- YARN Framework
- Recent Development
- Writing Your YARN Applications

Hortonworks

Recent Development (1)

ResourceManager High Availability

- RM is a single point of failure.
  - Restart for various reasons: Bugs, hardware failures, deliberate down-time for upgrades
- Goal: transparent to users and no need to explicitly monitor such events and re-submit jobs.
- Overly complex in MRv1 for the fact that JobTracker has to save too much information: both cluster state and per-application running state.
  - limitation: JobTracker dies meaning all the applications' running-state are lost

Recent Development (2)

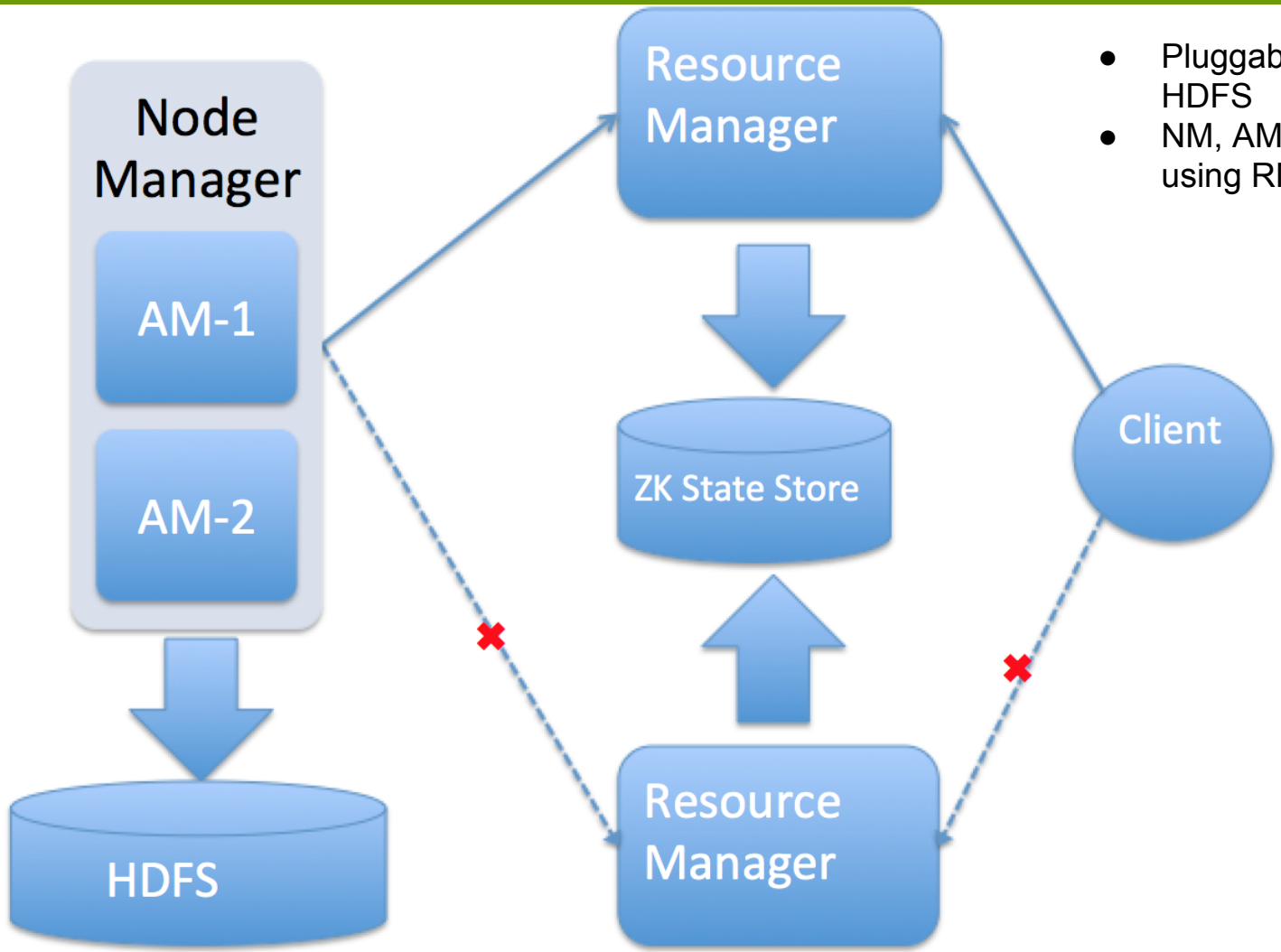
ResourceManager Recovery

- RM Restart Phase 1 (Done): All running Applications are killed after RM restarts.
- RM Restart Phase 2: Applications are not killed and report running state back to RM after RM comes up.

- RM only saves application submission metadata and cluster-level status (eg: Secret keys, tokens)

- Each application is responsible for persisting and recovering its application-specific running state.
  - MR job implements its own recovery mechanism by writing job-specific history data into a separate history file on HDFS

Recent Development (3)



- Pluggable state store: ZooKeeper, HDFS
- NM, AM, Clients retry and redirect using RM proxy

Hortonworks

Recent Development (4)

ResourceManager Failover

- Leader election (ZooKeeper)
- Transfer of resource-management authority to a newly elected leader.
- Clients (NM, AM, clients) redirect to the new RM
  - abstracted by RMProxy.

Hortonworks

Recent Development (5)

Long Running Service

- Work-preserving AM restart.
- Work-preserving RM restart.
- Work-preserving NM restart.

Hortonworks

Recent Development (6)

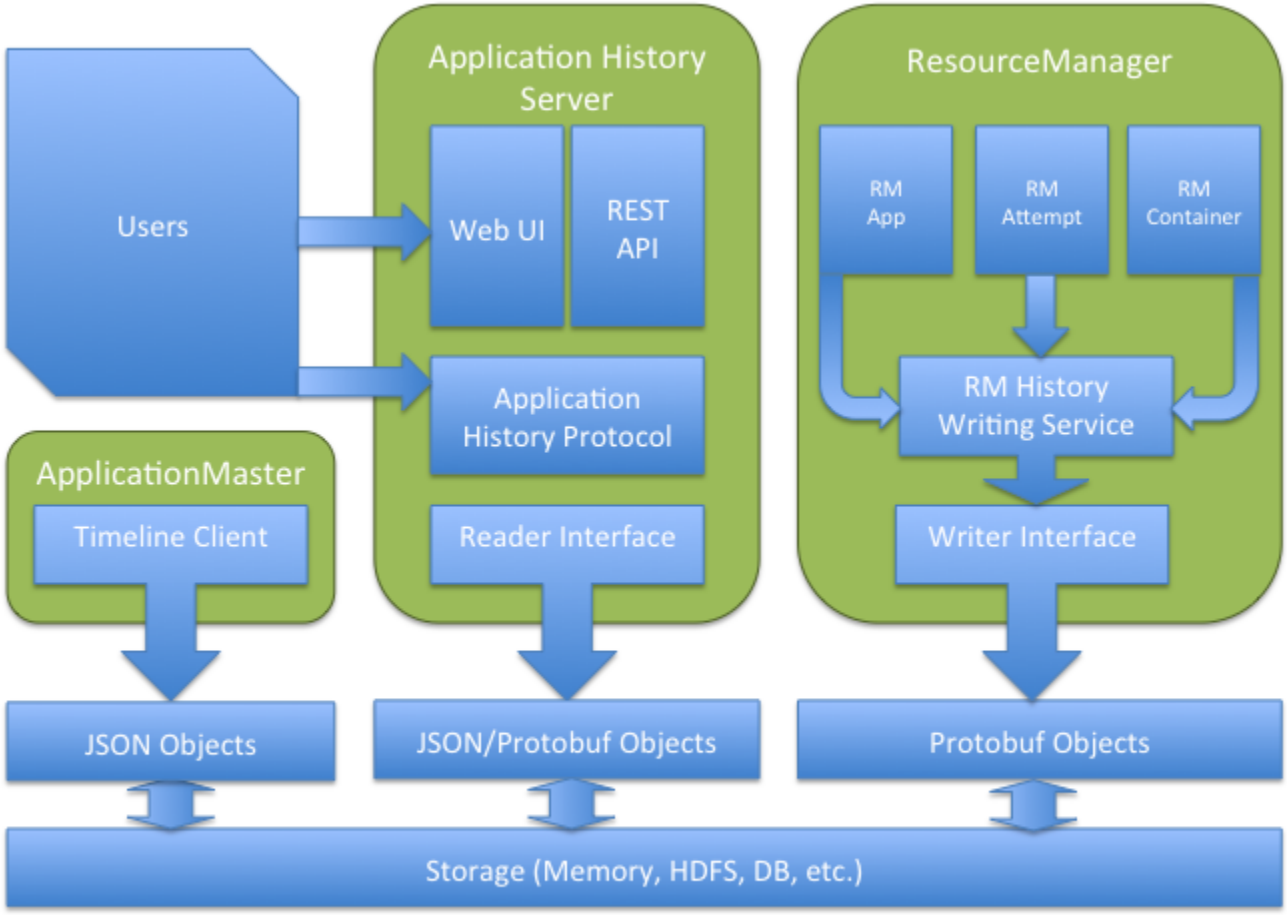
Application Historic Data Service

- ResourceManager records generic application information
  - Application
  - ApplicationAttempt
  - Container
- ApplicationMaster writes framework specific information
  - Free for users to define
- Multiple interfaces to inquiry the historic information
  - RPC
  - Web UI
  - RESTful Services

Hortonworks

Recent Development (7)

Application History Data Service



Hortonworks

Agenda

- What Is YARN
- YARN Framework
- Recent Development
- Writing Your YARN Applications

Writing Your YARN Applications (1)

Client API

- ApplicationClientProtocol
  - The protocol for a client that communicates wi
    ResourceManager
  - submitApplication, getApplicationReport,
    killApplication, etc.
  - YarnClient Library
    - Wrapper over ApplicationClientProtocol to simplify usage

Hortonworks

Writing Your YARN Applications (2)

ApplicationMaster API

- ApplicationMasterProtocol
  - The protocol used by ApplicationMaster to talk to ResourceManager
  - registerApplicationMaster, finisApplicationMaster, allocate
  - AMRMClient, AMRMClientAsync
- ContainerManagementProtocol
  - The protocol used by ApplicationMaster to talk to NodeManager to
  - startContainers, stopContainers, etc.
  - NMClient, NMClientAsync

Hortonworks

Writing Your YARN Applications (3)

Example - Client: submitting an application

```
...
// Get the RPC stub
ApplicationClientProtocol applicationsManager =
    ((ApplicationClientProtocol) rpc.getProxy(
        ApplicationClientProtocol.class, rmAddress, appsManagerServerConf));
// Assign an application ID
GetNewApplicationRequest request =
    Records.newRecord(GetNewApplicationRequest.class);
GetNewApplicationResponse response =
     applicationsManager.getNewApplication(request);
...
// Create the request to send to the ApplicationsManager
SubmitApplicationRequest appRequest =
    Records.newRecord(SubmitApplicationRequest.class);
appRequest.setApplicationSubmissionContext(appContext);
// Submit the application to ResourceManager
applicationsManager.submitApplication(appRequest);
```

Hortonworks

Writing Your YARN Applications (4)

Example - Client: getting an application report

```
…
// Get an application report
GetApplicationReportRequest reportRequest =
    Records.newRecord(GetApplicationReportRequest.class);
reportRequest.setApplicationId(appId);
GetApplicationReportResponse reportResponse =
    applicationsManager.getApplicationReport(reportRequest);
ApplicationReport report = reportResponse.getApplicationReport();
```

Example - Client: killing an application

```
…
// Kill an application
KillApplicationRequest killRequest =
    Records.newRecord(KillApplicationRequest.class);
killRequest.setApplicationId(appId);
applicationsManager.forceKillApplication(killRequest);
```

Hortonworks

Writing Your YARN Applications (5)

Example - AM: registration

```
// Get the RPC stub
ApplicationMasterProtocol resourceManager =
    (ApplicationMasterProtocol) rpc.getProxy(ApplicationMasterProtocol.class, rmAddress, conf);
RegisterApplicationMasterRequest appMasterRequest =
    Records.newRecord(RegisterApplicationMasterRequest.class);
// Set registration details
...
RegisterApplicationMasterResponse response =
    resourceManager.registerApplicationMaster(appMasterRequest);
```

Hortonworks

Writing Your YARN Applications (6)

Example - AM: requesting containers

```
List<ResourceRequest> requestedContainers;
List<ContainerId> releasedContainers
AllocateRequest req = Records.newRecord(AllocateRequest.class);
// The response id set in the request will be sent back in
// the response so that the ApplicationMaster can
// match it to its original ask and act appropriately.
req.setResponseId(rmRequestID);
// Set ApplicationAttemptId
req.setApplicationAttemptId(appAttemptID);
// Add the list of containers being asked for
req.addAllAsks(requestedContainers);
// Add the list of containers which the application don't need any more
req.addAllReleases(releasedContainers);
// Assuming the ApplicationMaster can track its progress
req.setProgress(currentProgress);
AllocateResponse allocateResponse = resourceManager.allocate(req);
```

Hortonworks

Writing Your YARN Applications (7)

Examples - AM: Starting containers

```
// Get the RPC stub
ContainerManagementProtocol cm =
    (ContainerManager)rpc.getProxy(ContainerManagementProtocol.class, cmAddress, conf);
// Now we setup a ContainerLaunchContext
ContainerLaunchContext ctx =
    Records.newRecord(ContainerLaunchContext.class);
…
// Send the start request to the ContainerManager
StartContainerRequest startReq = Records.newRecord(StartContainerRequest.class);
startReq.setContainerLaunchContext(ctx);
cm.startContainer(startReq);
```

Hortonworks

Q&A

http://hortonworks.com/labs/yarn/

Hortonworks