Notes

Kerberos authentication

<https://kb.iu.edu/d/aumh>

<http://web.mit.edu/kerberos/krb5-1.12/doc/admin/admin_commands/kadmin_local.html>

Common queries - use the below link – code available

<http://tiku.io/total/4/>

**Pig integration with Hbase** - via default approach and Hcatalog approach – provides ACID Property

http://chimera.labs.oreilly.com/books/1234000001811/ch12.html#hbase

// Schema of user\_links is (id, name, email, links).

// Notice how the id (key) field is omitted in the argument.

store user\_links into 'hbase://users'

using org.apache.pig.backend.hadoop.hbase.HBaseStorage(

'user\_info:name, user\_info:email, links:\*');Accessing Cassandra via pig

<http://chimera.labs.oreilly.com/books/1234000001811/ch12.html#hbase>

HIVE

Lateral View –

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+LateralView>

[How to feed XML to your Pig](http://blog.mortardata.com/post/61678005593/xml-pig-loader)

<http://blog.mortardata.com/post/61678005593/xml-pig-loader>

data = LOAD '/path/to/speaking\_events.xml'  
 USING org.apache.pig.piggybank.storage.StreamingXMLLoader(  
 'Document',  
 'event, gathering'  
 ) AS (  
 event: {(attr:map[], content:chararray)}  
 gathering {(attr:map[], content:chararray)}  
 );

|  |  |
| --- | --- |
|  | The below regex works for me.  b= foreach a generate REGEX\_EXTRACT(x,'Id="(?[^"]+)"',1),REGEX\_EXTRACT(x,'UserId="(?[^"]+)"',1),REGEX\_EXTRACT(x,'Name="(?[^"]+)"',1),REGEX\_EXTRACT(x,'Date="(?[^"]+)"',1);  dump b;  Here is the output.  (1815,829,Student,2014-01-18T16:02:02.837)  (1816,401,Commentator,2014-01-18T17:07:19.713) |

Using XPath

XPath is a function that allows text extraction from xml. Starting PIG 0.13 , Piggy bank UDF comes with XPath support. It eases the XML parsing in PIG scripts. A sample script using XPath is as shown below.

Using XPath

REGISTER piggybank.jar

DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();

A = LOAD 'xmls/hadoop\_books.xml' using org.apache.pig.piggybank.storage.XMLLoader('BOOK') as (x:chararray);

B = FOREACH A GENERATE XPath(x, 'BOOK/AUTHOR'), XPath(x, 'BOOK/PRICE');

dump B;

**pig Join**

A = LOAD 'a.txt' AS (n:chararray, a:int);

B = LOAD 'b.txt' AS (n:chararray, m:chararray);

C = JOIN A by $0 LEFT OUTER, B BY $0;

|  |  |
| --- | --- |
| **BY** | **Keyword** |
| **LEFT** | **Left outer join.** |
| **RIGHT** | **Right outer join.** |
| **FULL** | **Full outer join.** |
| **OUTER** | **(Optional) Keyword** |

**Oozie**

Dryrun – tests the workflow application with ok or failed status without creating job

Submitting pig jobs through HTTP

Submitting hive jobs through HTTP

Web HDFS

<http://hortonworks.com/community/forums/topic/web-hdfs-in-java/>

Service running status

service --status-all

**Authentication SharePoint link**

<https://share.ams.bnymellon.net/sites/GlobalCoE/CoE_BigData/hadoop/SitePages/Authentication%20through%20client%20host.aspx>

TEZ Session Timeout Error

When a hive session running with TEZ is not used(Inactive) for more than 5 mins default. Then Session got aborted with below error

hive> select A.emp\_id,B.commit\_id

> FROM (select \* from temp\_tbl1) A

> LEFT JOIN

> (select \* from temp\_tbl2) B;

No encryption was performed by peer.

Warning: Map Join MAPJOIN[9][bigTable=?] in task 'Map 2' is a cross product

Query ID = root\_20150311081717\_430ea42a-e755-444e-9057-15c6419a2a9a

Total jobs = 1

Launching Job 1 out of 1

No encryption was performed by peer.

Tez session was closed. Reopening...

Session re-established.

Status: Running (application id: application\_1425643808115\_0469)

Map 1: -/- Map 2: -/-

Status: Failed

Vertex failed, vertexName=Map 1, vertexId=vertex\_1425643808115\_0469\_1\_01, diagnostics=[Vertex Input: temp\_tbl2 initializer failed., token (HDFS\_DELEGATION\_TOKEN token 26113 for dshmbtm) can't be found in cache]

Vertex failed, vertexName=Map 2, vertexId=vertex\_1425643808115\_0469\_1\_00, diagnostics=[Vertex Input: temp\_tbl1 initializer failed., token (HDFS\_DELEGATION\_TOKEN token 26113 for dshmbtm) can't be found in cache]

DAG failed due to vertex failure. failedVertices:2 killedVertices:0

FAILED: Execution Error, return code 2 from org.apache.hadoop.hive.ql.exec.tez.TezTask

**R**

**R to HDFS via rhdfs**

Download package from github

>rhdfs\_1.0.8.tar.gz

Requred packages

>rJava

Install step

>export HADOOP\_CMD=/usr/lib/hadoop/bin/hadoop

> R CMD INSTALL ~/rhdfs\_1.0.8.tar.gz

Config step

>library(rhdfs)

>hdfs.init()

>hdfs.ls('/')

conn <- hdfs.file("/tmp/testr.txt","r",buffersize=104857600);

fread <- hdfs.read(conn);

fil <- unserialize(fread);

f <- rawToChar(fread)

data = read.table(textConnection(f), sep = "\t");

----------------------------------------------

####Reading a text file

con <- hdfs.file("/tmp/rest.txt","r",buffersize=104857600);

fread1 <- hdfs.read.text.file("/tmp/rest.txt")

print(fread1)

> print(fread1)

[1] "hadoop" "rest" "api" ""

-------------------------------------------------------

####Reading a numeric file

conn <- hdfs.file("/tmp/testr.txt","r",buffersize=104857600);

fread <- hdfs.read(conn);

f <- rawToChar(fread)

> print(f)

[1] "0\t21.5\t43\t0\n0.5\t28.2\t56.4\t14.1\n1\t32.5\t65\t32.5\n1.5\t35.3\t70.6\t52.95\n2\t37.7\t75.4\t75.4\n2.5\t39.2\t78.4\t98\n3\t40.1\t80.2\t120.3\n4\t41.2\t82.4\t164.8\n5\t42.2\t84.4\t211\n7\t43.6\t87.2\t305.2\n10\t45.6\t91.2\t456\n"

-----------------------------------------------------------

Same data with read.table

data = read.table(textConnection(f), sep = "\t");

> print(data)

V1 V2 V3 V4

1 0.0 21.5 43.0 0.00

2 0.5 28.2 56.4 14.10

3 1.0 32.5 65.0 32.50

4 1.5 35.3 70.6 52.95

5 2.0 37.7 75.4 75.40

6 2.5 39.2 78.4 98.00

7 3.0 40.1 80.2 120.30

8 4.0 41.2 82.4 164.80

9 5.0 42.2 84.4 211.00

10 7.0 43.6 87.2 305.20

11 10.0 45.6 91.2 456.00

d1 <-data.frame(data)

f1<-d1$V1

print(f1)

[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 4.0 5.0 7.0 10.0

logic <- f1>4

> print(logic)

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE

f2<-d1$V2

f3<-d1$V3

f4<-d1$V4

mat <- cbind(f1,f2) ---------Column binding

mat <- rbind(f1,f2,f3,f4) ----------row binding

out<-outer(f1,f2) --------outer product.

matr<-f1%\*%f2;

curve(f1, from = 0, to = 10, n = 101,

type = "l", xname = "x", xlab = xname, ylab = NULL,

log = NULL, xlim = NULL)

-----------------------

summary(data)

V1 V2 V3 V4

Min. : 0.000 Min. :21.50 Min. :43.00 Min. : 0.00

1st Qu.: 1.250 1st Qu.:33.90 1st Qu.:67.80 1st Qu.: 42.73

Median : 2.500 Median :39.20 Median :78.40 Median : 98.00

Mean : 3.318 Mean :37.01 Mean :74.02 Mean :139.11

3rd Qu.: 4.500 3rd Qu.:41.70 3rd Qu.:83.40 3rd Qu.:187.90

Max. :10.000 Max. :45.60 Max. :91.20 Max. :456.00

-----------------------

> t.test(f1,f2)

Welch Two Sample t-test

data: f1 and f2

t = -14.3115, df = 13.431, p-value = 1.623e-09

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-38.76013 -28.62169

sample estimates:

mean of x mean of y

3.318182 37.009091

-----------------------

search()

[1] ".GlobalEnv" "package:rhdfs" "package:rJava"

[4] "package:stats" "package:graphics" "package:grDevices"

[7] "package:utils" "package:datasets" "package:methods"

[10] "Autoloads" "package:base"

------------------------

points(f1, col = 1:4, pch = 8, cex = 2)

------------------------------------------------------------

1. The entities that R creates and manipulates are known as objects.

>objects() or ls()

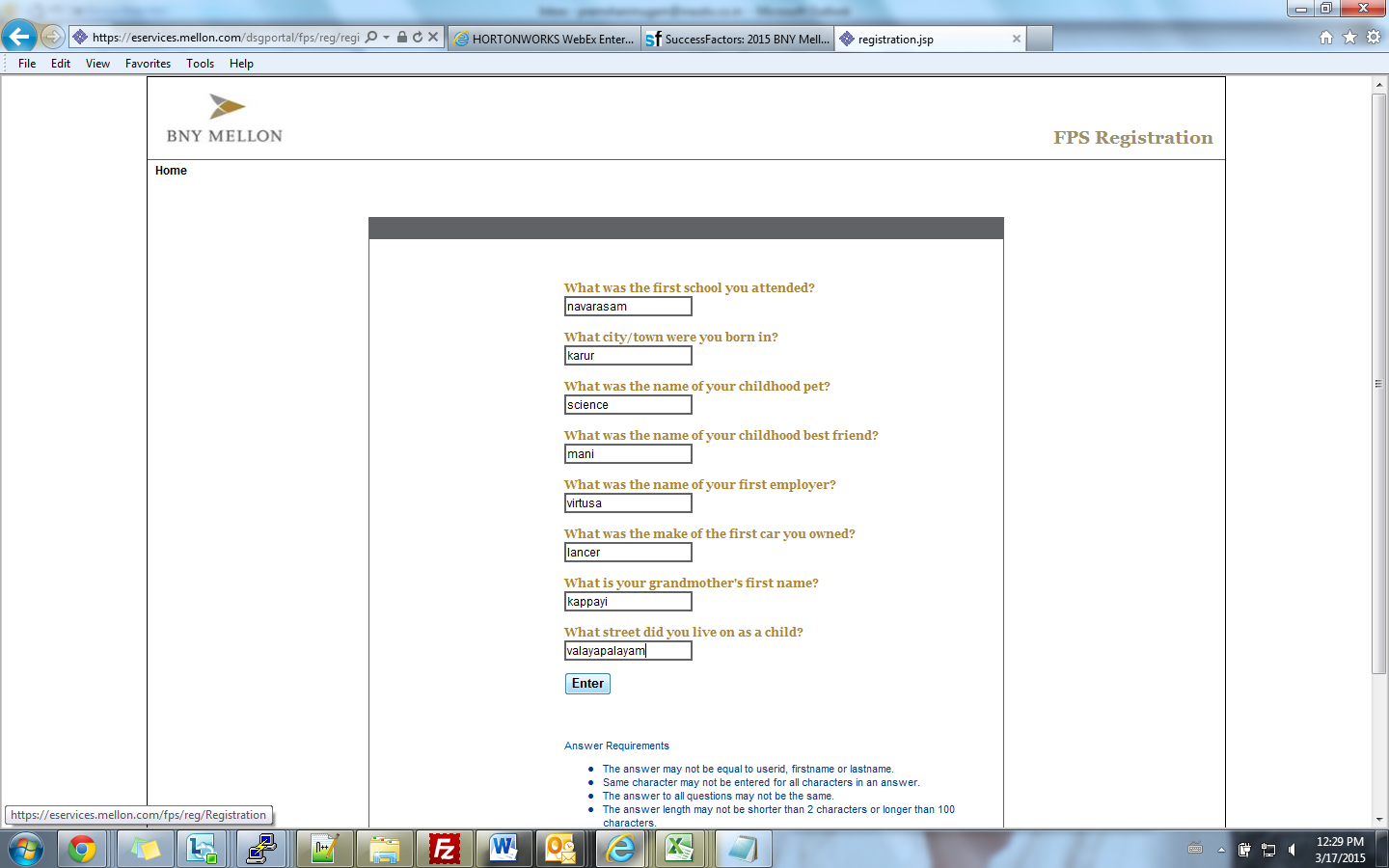
2. Can remove those objects by

>rm(obj\_name)

3. these obj ll be stored in .RData

setwd(

out<-read.table(pipe("hadoop dfs -cat '/tmp/testdata/test.csv'"), sep=",", header=TRUE)



**Hive debug mode**

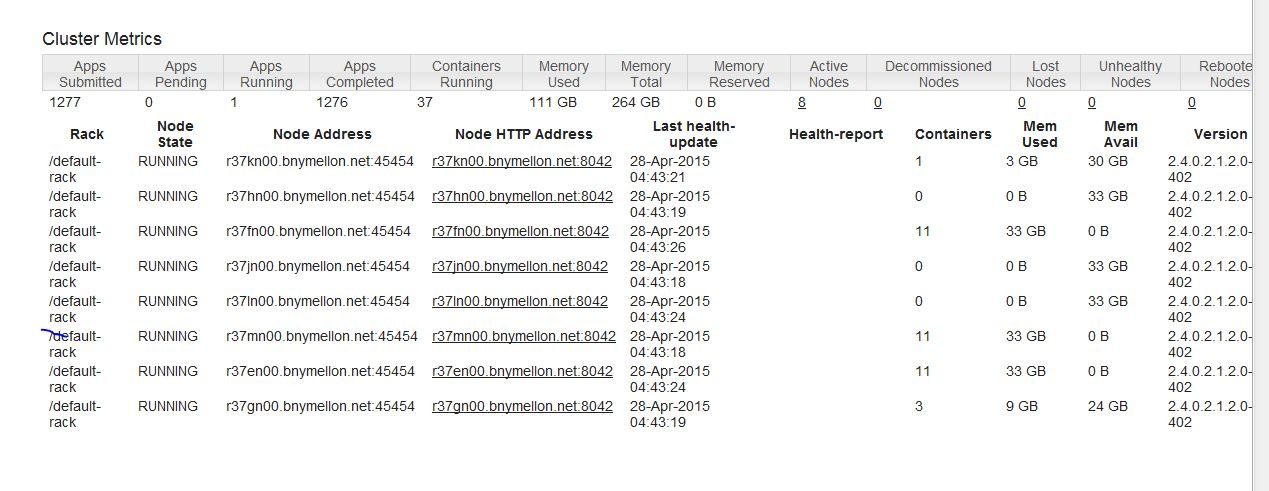
hive --hiveconf hive.root.logger=DEBUG,console

**Capacity-scheduler.xml – r37bn00 (dev)**

**Case 00045911 Raised for Container allocation in cluster is not even**

**Case details**

**** 



HWX Suggested us to change few parameters and explained about Capacity-scheduler.

**yarn.scheduler.maximum-allocation-mb**

**SET mapred.output.compression.codec org.apache.hadoop.io.compress.SnappyCodec;**

**SET mapred.output.compression.type BLOCK;**

V changed it in Pig with above parameters. And also they suggested us to check after changing the “yarn.scheduler.capacity.root.default.capacity“ from 40 to 80.

-------

We change the “yarn.scheduler.capacity.root.default.capacity” from 40 to 80 and now have increased it to 100%

**So we are able to achive double usage of containers.**

After changing

**yarn rmadmin -refreshQueues**

Lists all the

**hadoop queue -list**

**hadoop queue -showacls**

Reference

1. <https://support.pivotal.io/hc/en-us/articles/201623853-How-to-configure-queues-using-YARN-capacity-scheduler-xml->
2. <http://pivotalhd.docs.pivotal.io/docs/yarn-resource-management.html>

Process

Ps – ef

Ps –xw – for background running processes

**Balancer**

The balancer, in general, should be run when you have a large number of datanodes that are significantly disproportionate in how much data they are handling. For large sites with a significant amount of node failures, they often run balancer 'continuously'.

That said:

Run it whenever you add a rack.

Run it whenever you are in a 'near full' situation. Do not run it when you are full (HDFS at 80% used). Clear space first!

Be mindful of the -threshold parameter. Setting it \*too low\* means balancer will never finish. Setting it \*too high\* makes it useless. The default (10%) is generally too high. I usually go for 5% myself.

Don't run the balancer on the name node or anywhere else a hadoop daemon is running.

The namenode has protections to prevent mass block movement so that services keep running as expected. In other words, there are limits on how many blocks can be moved at one time. This scales based on the size of the grid.

Don't forget to tune dfs.balance.bandwidthPerSec to something realistic for your network. In most cases, the default is too low.

Syntax:

hdfs balancer

[-threshold <threshold>]

[-policy <policy>]

[-exclude [-f <hosts-file> | <comma-separated list of hosts>]]

[-include [-f <hosts-file> | <comma-separated list of hosts>]]

[-idleiterations <idleiterations>]

| **COMMAND\_OPTION** | **Description** |
| --- | --- |
| -policy <policy> | datanode (default): Cluster is balanced if each datanode is balanced. blockpool: Cluster is balanced if each block pool in each datanode is balanced. |
| -threshold <threshold> | Percentage of disk capacity. This overwrites the default threshold. |
| -exclude -f <hosts-file> | <comma-separated list of hosts> | Excludes the specified datanodes from being balanced by the balancer. |
| -include -f <hosts-file> | <comma-separated list of hosts> | Includes only the specified datanodes to be balanced by the balancer. |
| -idleiterations <iterations> | Maximum number of idle iterations before exit. This overwrites the default idleiterations(5). |

Suggested by Ambari

**Have modified the below properties as suggested by ambari on 30-july-2015**

Some service configurations are not configured properly. We recommend you review and change the highlighted configuration values. Are you sure you want to proceed without correcting configurations?

| **Service** | **Property** | **Value** | **Description** |
| --- | --- | --- | --- |
| YARN | yarn.scheduler.maximum-allocation-mb | 33792 | The maximum allocation for every container request at the RM, in MBs. Memory requests higher than this won't take effect, and will get capped to this value.  **Value is less than the recommended default of 67584** |
| YARN | yarn.nodemanager.resource.memory-mb | 33792 | Amount of physical memory, in MB, that can be allocated for containers. **Value is less than the recommended default of 67584** |

**Result : After changing this we could see the application running time got reduced to 1/3.**



**BXP to Hadoop Connection steps**

Kindly refer the below link to connect hadoop cluster from BXP servers.

<https://mysourcesocial.bnymellon.net/docs/DOC-20702>

**To find the empty line in a file**

awk '!NF {s+=1} END {print s}'

**Ontology**

References :

<http://www.w3.org/standards/semanticweb/ontology#examples>

<https://en.wikipedia.org/wiki/Turtle_(syntax)>

<http://www.cambridgesemantics.com/semantic-university/rdf-101>

<http://www.linkeddatatools.com/introducing-rdfs-owl>

<http://www.w3.org/TR/owl-features/>

<https://github.com/d2rq/d2rq>

<http://d2rq.org/d2rq-language#examples>

<http://www.w3schools.com/xml/xml_rdf.asp>

**AWS**

In Namenode

> jps

39971 NameNode

64452 Jps

43250 DFSZKFailoverController

37136 JournalNode

/usr/jdk64/jdk1.7.0\_45/bin/jmap -heap 39971

=================================================

JVM - Memory Model

------------------

New Gen

-Eden

-Survivor

Old Gen

Permanent Gen

Ref :: <http://www.journaldev.com/2856/java-jvm-memory-model-and-garbage-collection-monitoring-tuning>

JVM

Each Java application runs inside a runtime instance of some concrete implementation of the abstract specification of the Java virtual machine.

* to run one Java application. When a Java application starts, a runtime instance is born. When the application completes, the instance dies. If you start three Java applications at the same time, on the same computer, using the same concrete implementation, you'll get three Java virtual machine instances. Each Java application runs inside its own Java virtual machine

https://www.artima.com/insidejvm/ed2/jvm2.html

http://www.cubrid.org/blog/dev-platform/understanding-jvm-internals/

=================================================

Yarn Arch-“ Yet-Another-Resource-Negotiator”

The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker, in other words resource management and job scheduling/monitoring, into separate daemons: a global ResourceManager and per-application ApplicationMaster (AM).

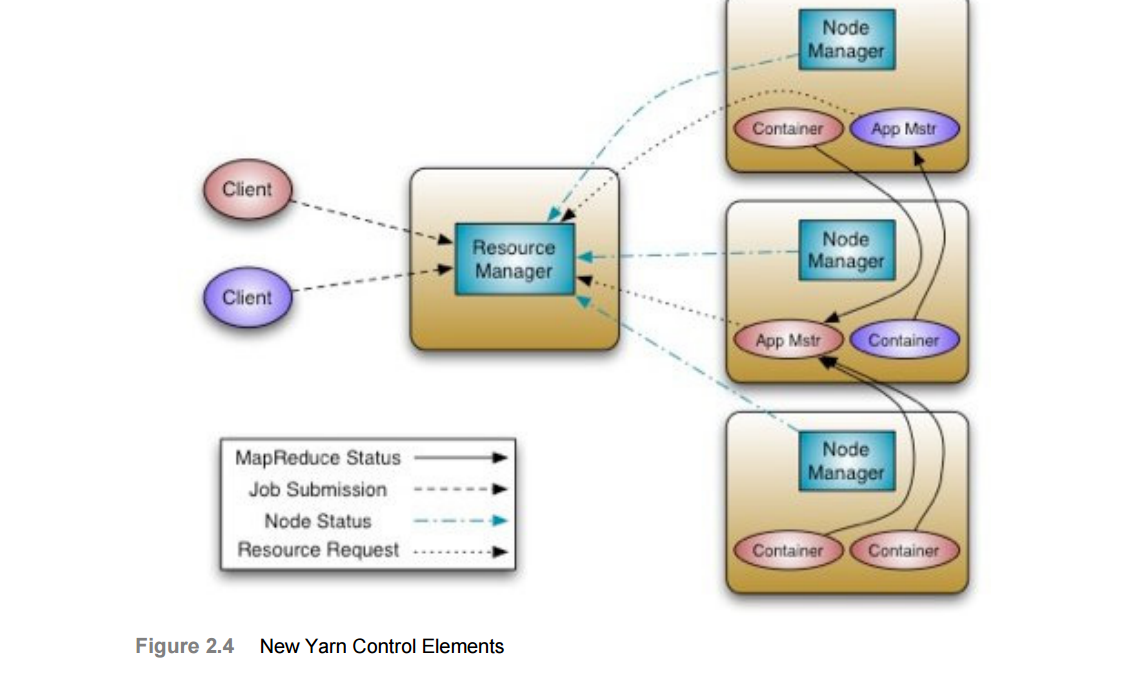
YARN provides the daemons and APIs necessary to develop generic distributed applications of any kind, handles and schedules resource requests (such as memory and CPU) from such applications, and supervises their execution.

Under YARN, there is no distinction between resources available for maps and resources available for reduces – all resources are available for both

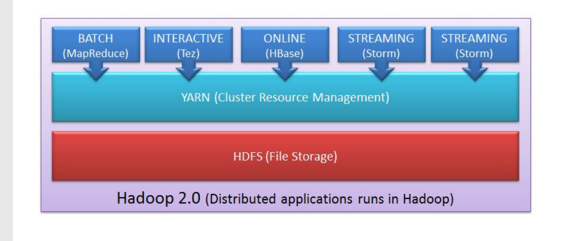
MR2

The old MR1 framework was rewritten to run within a submitted application on top of YARN.

In MR2, Every application controls its own density via application master.



YARN’s execution model is more generic than the earlier MapReduce implementation. YARN can run applications that do not follow the MapReduce model, unlike the original Apache Hadoop MapReduce (also called MR1).



**Resource manager allocation.**

For each application running on cluster will have its own Application Master(1-to-1).

Client will request for new application request(1). The RM respose, marked (2), will typically contain a newly generated unique application ID, in addition to information about cluster resource capabilities that the client will need in requesting resources for running the application’s AM.

Using the information received from the RM, the client can construct and submit an “Application Submission Context”, marked (3), which typically contains information like scheduler queue, priority and user information, in addition to information needed by the RM to be able to launch the AM. This information is contained in a “Container Launch Context”, which contains the application’s jar, job files, security tokens and any resource requirements.

**Application Master**

The first step, marked (1) in the diagram, is for the AM to register itself with the RM. This step consists of a handshaking procedure and also conveys information like the RPC port that the AM will be listening on, the tracking URL for monitoring the application’s status and progress, etc.

The RM registration response, marked (2), will convey essential information for the AM master like minimum and maximum resource capabilities for this cluster. The AM will use such information in calculating and requesting any resource requests for the application’s individual tasks. The resource allocation request from the AM to the RM, marked (3), mainly contains a list of requested containers, and may also contain a list of released containers by this AM. Heartbeat and progress information are also relayed through resource allocation requests as shown by arrow (4).

When the Scheduler component of the RM receives a resource allocation request, it computes, based on the scheduling policy, a list of containers that satisfy the request and sends back an allocation response, marked (5), which contains a list of allocated resources. Using the resource list, the AM starts contacting the associated node managers (as will be soon seen), and finally, as depicted by arrow (6), when the job finishes, the AM sends a Finish Application message to the Resource Manager and exits.

 Resources are requested in the form of containers, where each container has a number of non-static attributes?????

Reference

http://blog.cloudera.com/blog/2012/02/mapreduce-2-0-in-hadoop-0-23/

JobHistoryServer- information about completed jobs.

NodeManager is responsible for launching containers, each of which can house a map or reduce task

MR2 supports both the old (“mapred”) and new (“mapreduce”) MapReduce APIs