Big Data Analysis
- Part I : Introduction to Hadoop and Spark at TACC

April. 20, 2017

Dr. Weijia Xu(Manager), Dr. Ruizhu Huang
Data Mining & Statistics Group
Texas Advanced Computing Center (TACC)
The University of Texas at Austin

Big Data Training Series at TACC

- Introduction to Hadoop and Spark On Wrangler – Today
- Introduction to Scala/Spark - April 27, 2017 1:00- 4:30
  - 1:00-2:00 Introduce Scala programing
  - 2:00-3:00 Programming with spark using Scala
  - 3:00-4:00 Build spark application, unit test, submit, and running on Wrangler
  - 4:00-4:30 Hands on
- Data Analysis Using Hadoop/Spark - May 4, 2017
  - 1:00-2:00 Dataframe, SparkSQL
  - 2:00-3:00 Data analysis with MLlib and Graphx
  - 3:00-4:00 Spark streaming, advanced topic configuration/ optimization
  - 4:00-4:30 Hands on

Today's Agenda

1:00 – 1:30 Overview of Big Data Processing

- Software in Big Data/TACC
- Big Data and HPC
- Intro. to MapReduce Model

1:30 – 2:30 What's Hadoop

- What are HDFS and YARN
- Hadoop on Wrangler

2:30 – 3:00 Breaks / Hands-on exercise

3:00 – 4:00 Programming with Hadoop and Spark

- Using Hadoop with other languages
- Hadoop vs. Spark
- Intro. of Zeppelin
- Using Spark with Scala, Python and R

4:00 – 4:30 Hands-on exercise

What's the big challenge with the big data analysis?

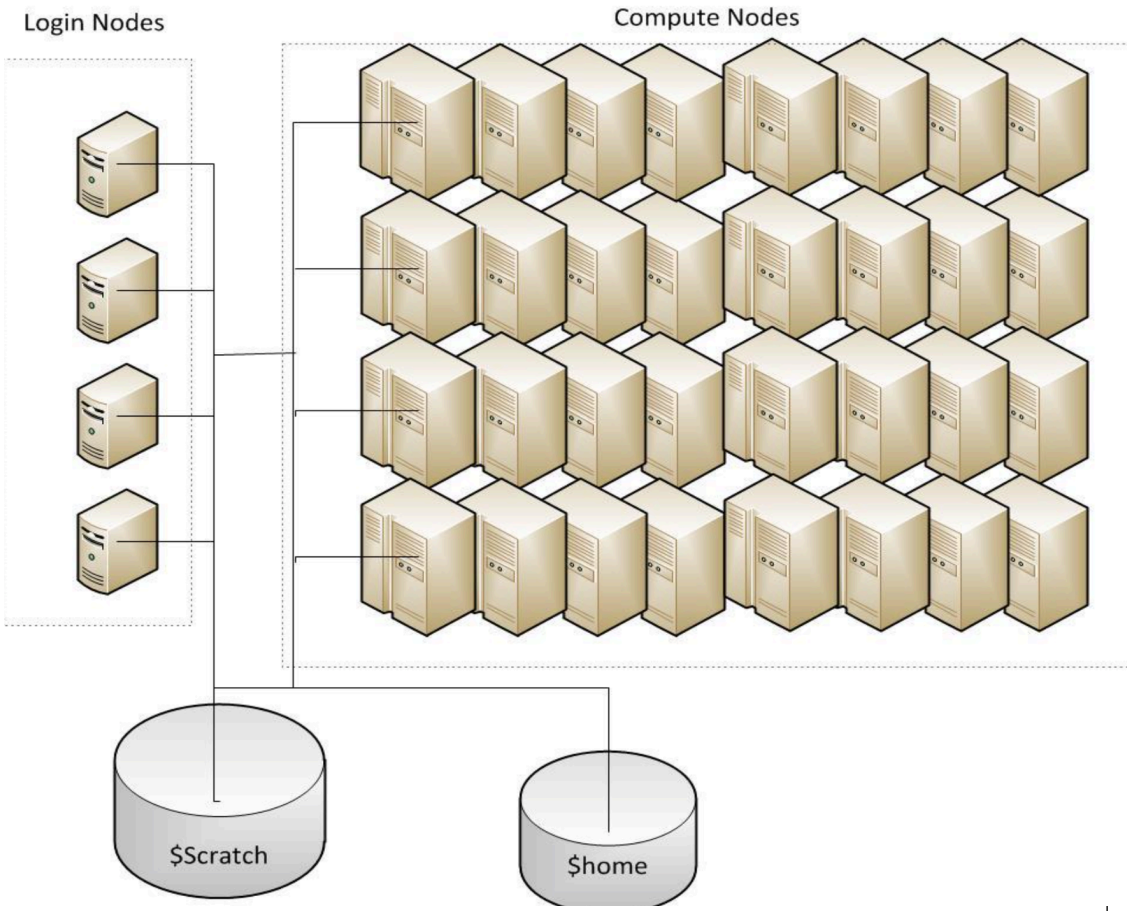
Data analysis process requires even more computational resources

- Storage, triple the size of the raw data to store the intermediate files, output etc.
- Memory, e.g. algorithm want to store the pair-wise distance matrix among data points

The analysis process would take much longer

- Typical hard drive read speed is about 150MB/sec,
  - Reading 1TB ~ 2 hours
- Analysis could require processing time quadratic to the size of the data
  - Analysis took 1 second for 1GB data, would require 11 days to finish for 1TB data

Big Data v.s. High Performance Computing



Traditional Computation at Computing Cluster

Data are stored separately and accessed by compute nodes during the access,

- Typically a high speed connection between the storage and computing resources, e.g. 40Gbps

User need develop code that can run on multiple nodes concurrently

- Typically, master-slaves mode, where a master thread to control the process of multiple workers.

Do not really solve the Big Data problem.

MapReduce Model for Big Data

A programing model proposed and used by Google.

Serving as a platform for customized computation over large scale of data. :

- At the core, users implement interface of two main functions:

```
map   (in_key, in_value) ->
          (out_key, intermediate_value) list
reduce (out_key, intermediate_value list) ->
          out_value list
```

It's not a brand new idea, but several old ideas put together to handle data at scale

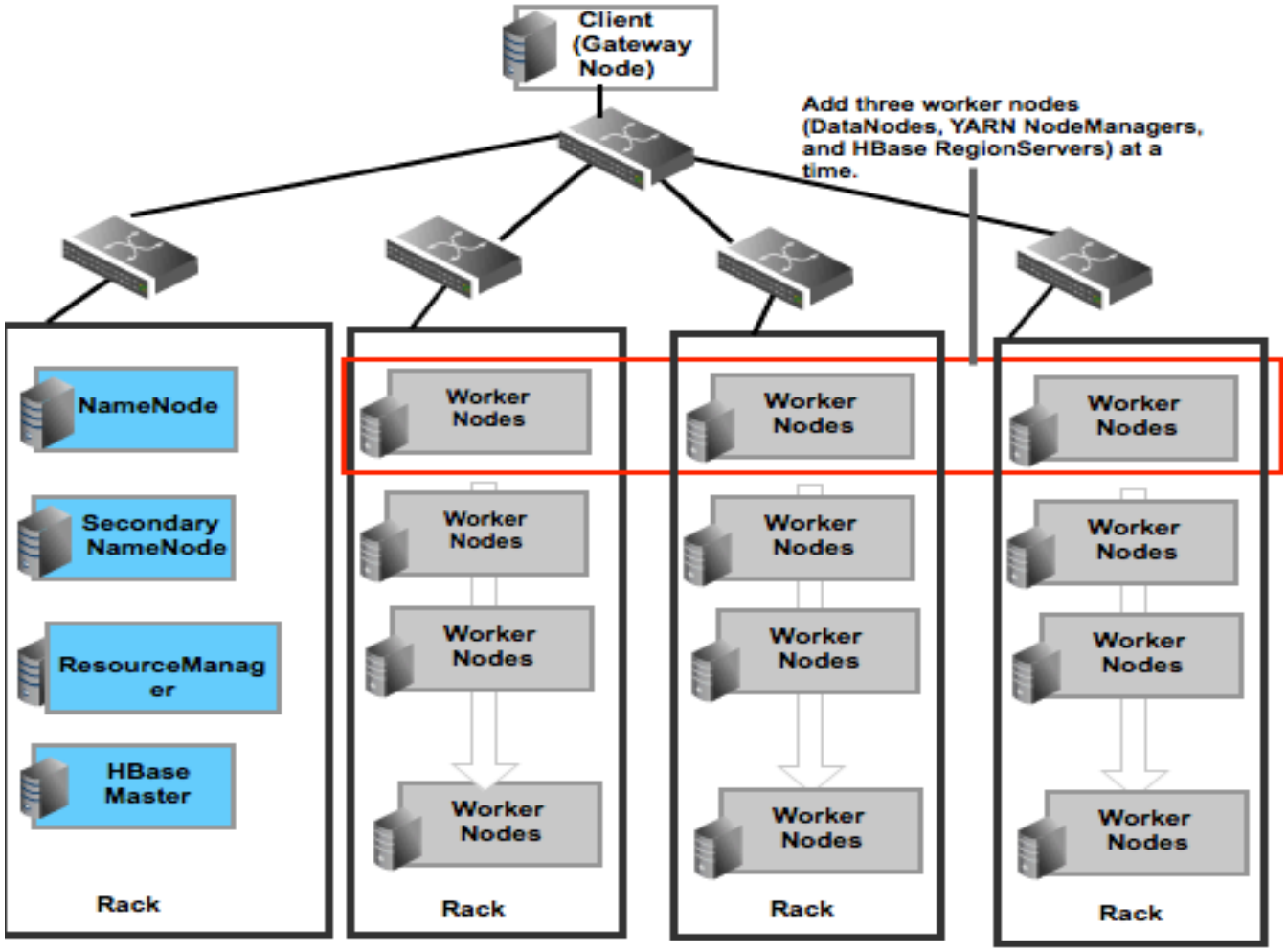
“Big” Ideas in MapReduce

- Move computations to data.
  - Do not use/assume high bandwidth inter-connection between nodes.
  - Avoid and reduce the need of data transfer over the network is often the bottleneck.
- Processing data in sequential order, avoid random access
  - Same idea as applied in the RDBMS
  - However, assume data can be processed any order,
  - All data are packed in to blocks (default at 64MB).

“Big” Ideas in MapReduce

- Scale out instead of scale up:
  - The same code can run with 1 node or 100 nodes.
- Hide system details from user
  - Providing abstraction in writing parallel code.
    - Mapper and reducer
    - Partitioner and combiner
  - Isolates developer from (and independent from) system hardware details
  - Once all required components are specified, data is automatically “sliced” and processed in parallel.

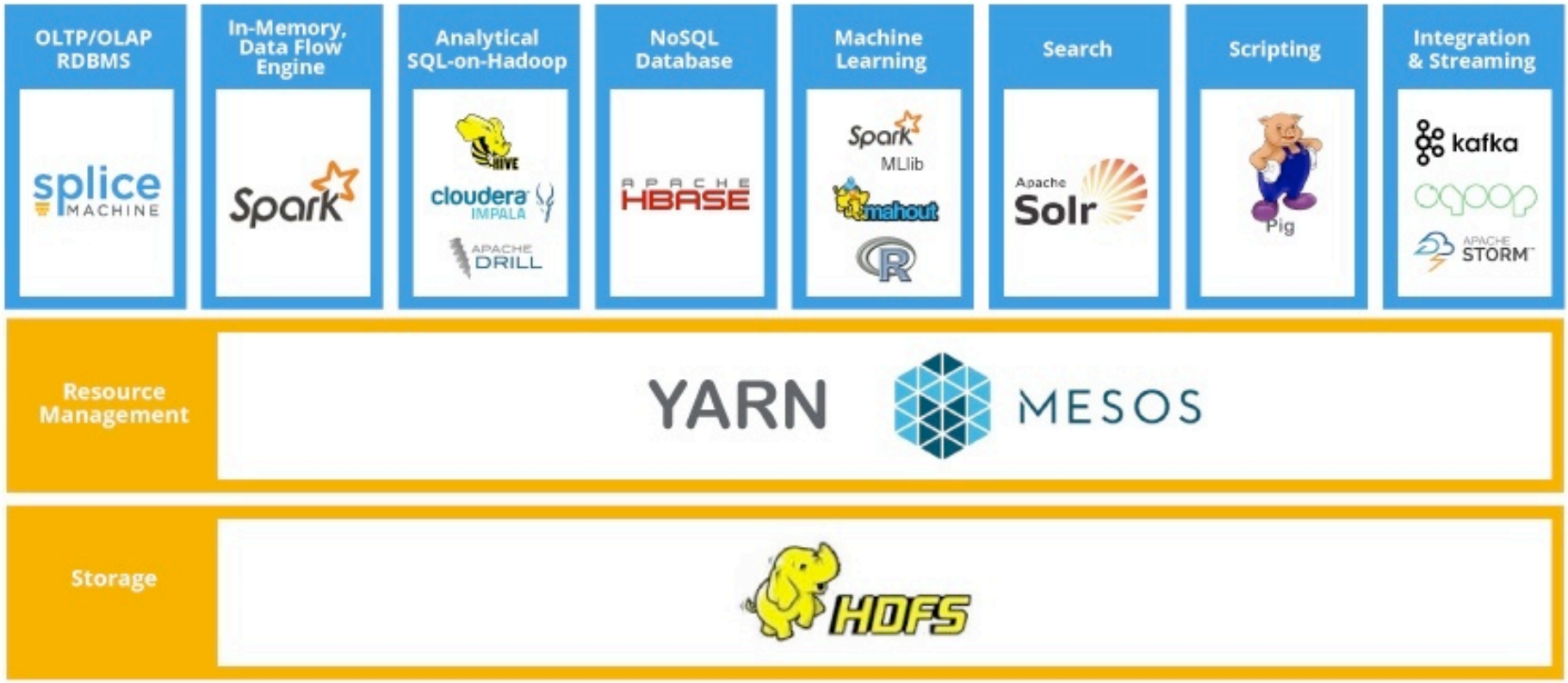
Support MapReduce with Hadoop Cluster



Support MapReduce with Hadoop Cluster

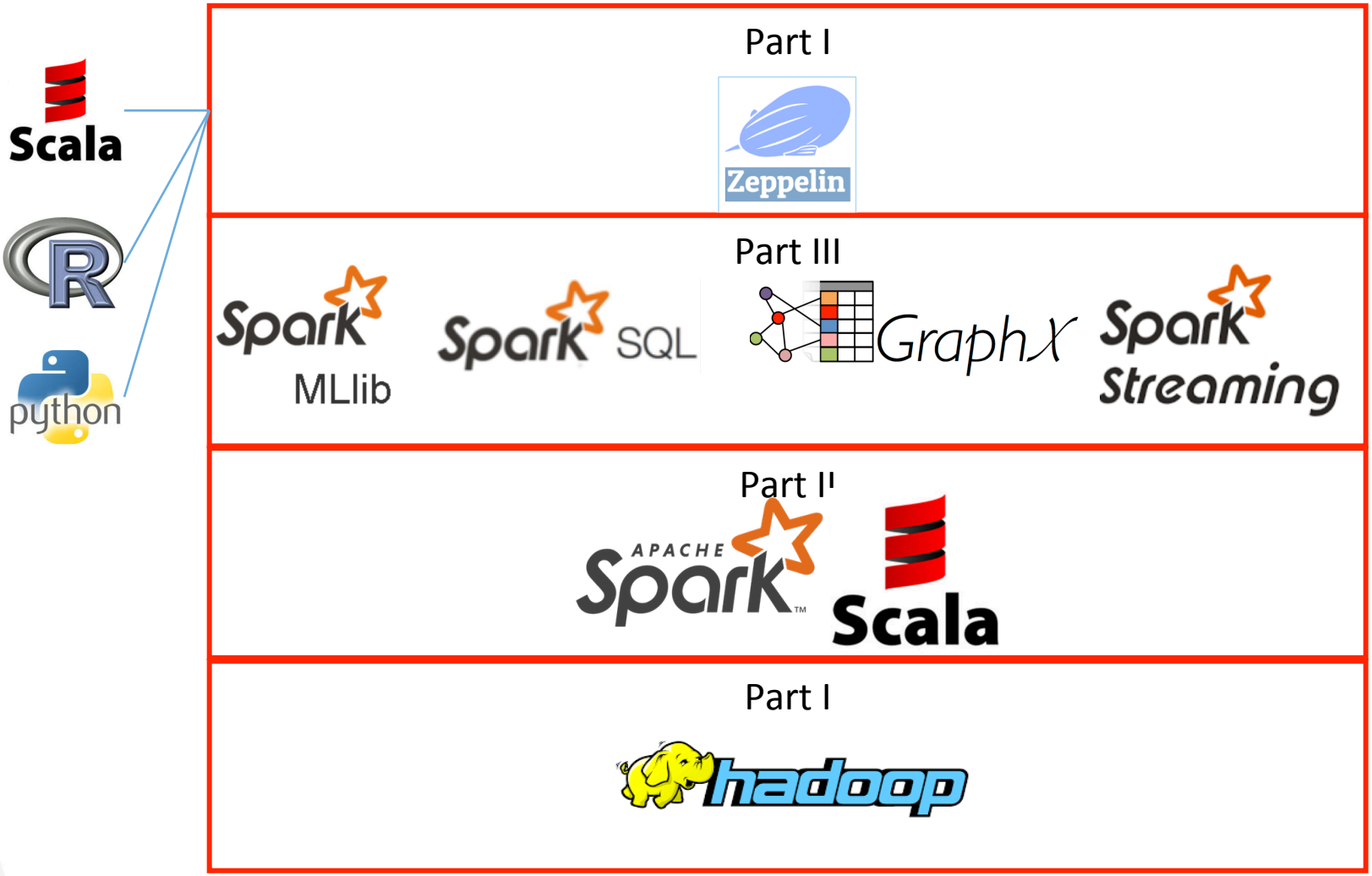
- Data are stored distributed across nodes within the hadoop cluster
  - The workload distribution are automatically handled by the Hadoop cluster
- Data can be treated as key-value pair
  - Either the “Key” or “Value” in a key-value pair could be any type of data.
- The computation is broke down in to two major steps:
  - Map instances:
    - process the data stored within one data block sequentially,
    - The result is in the format of list of Key-value pairs
  - Reduce instances:
    - Collect (key, value) pairs emitted by Map instances
    - Pairs with the same key are sent to the same reducer,
    - Each Reducer process the key-value pairs received and write output to a file.
  - User only required to develop functions for Map and Reduce

Hadoop Ecosystem



Big Data Analysis Support at TACC

About this series



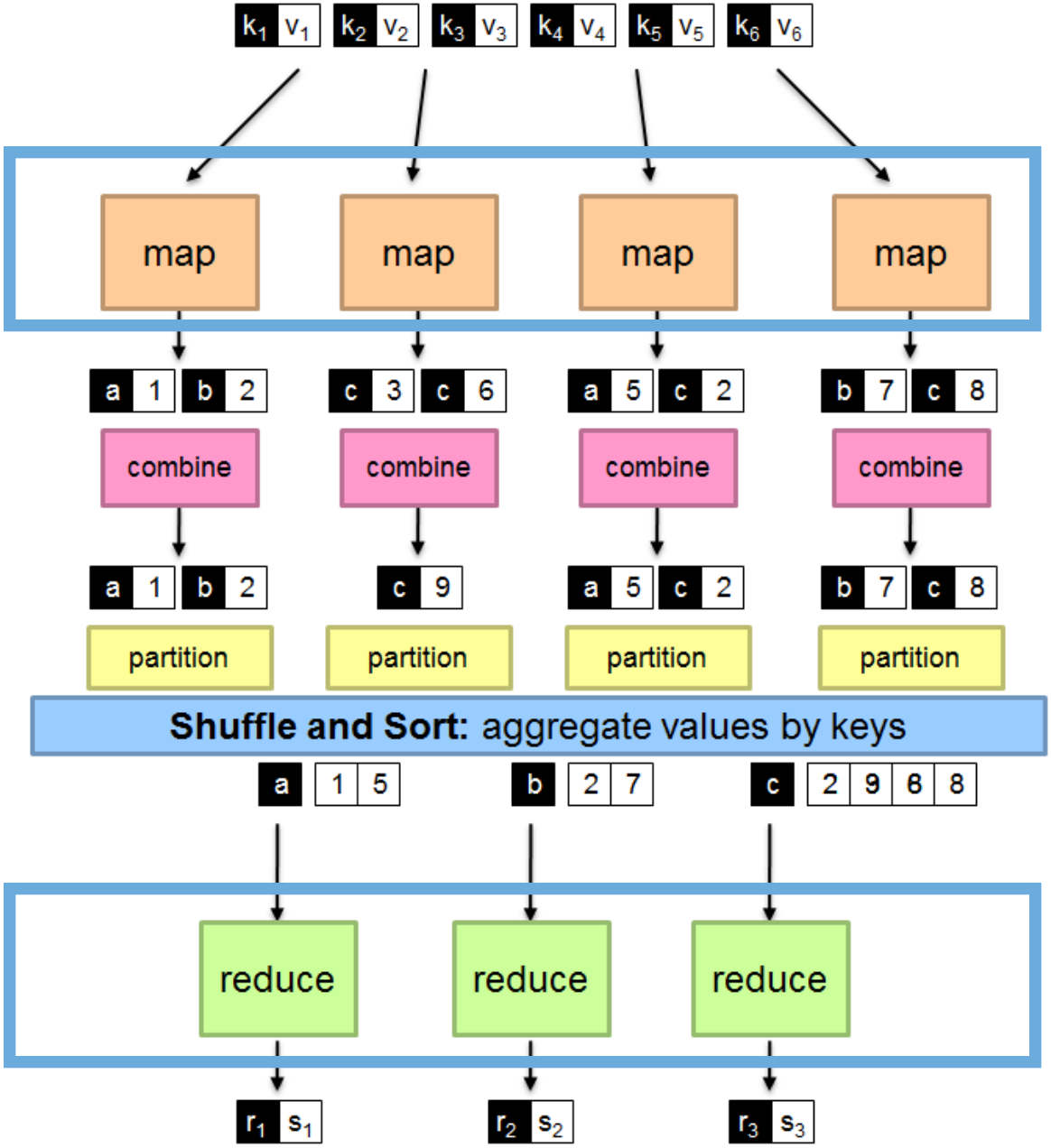
Recap

- Big Data Challenges
  - storing capacity
  - processing power
- MapReduce Model
  - Move computations to data
  - Processing data in sequential order, avoid random access
- Hadoop Infrastructure
  - NameNode
  - Secondary NameNode
  - Resource Manager
  - Worker/Compute/Data Nodes
- Software in Big Data/TACC

Hadoop and Yarn

- Hadoop is an open source implementation of MapReduce programming model in JAVA with interface to other programming language such as C/C++, python.
- Hadoop includes HDFS, YARN, MapReduce

Map Reduce Model



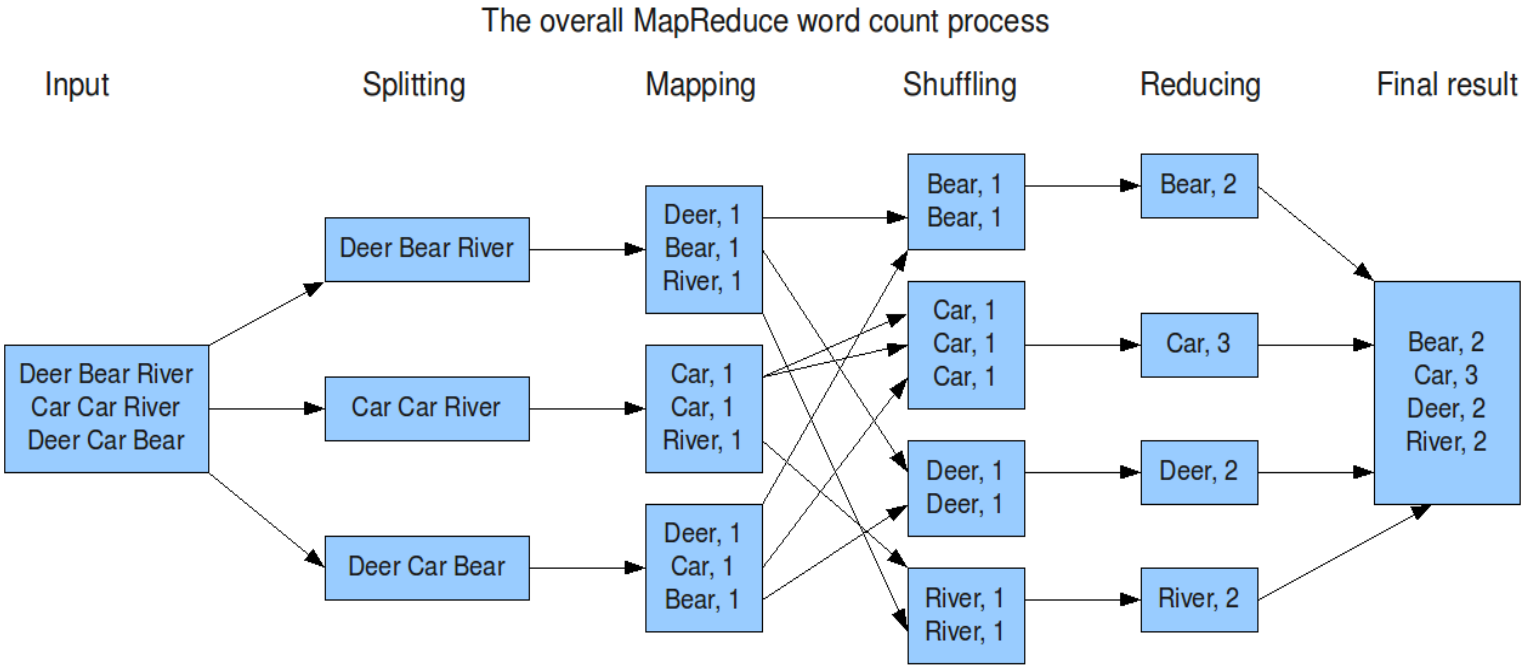
Word Count Example

Read text files and count how often words occur.

- The input is text files
- The output is a text file
  - each line: word, tab, count

Map: Produce pairs of (word, count)

Reduce: For each word, sum up the counts.


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

WordCount-Java Overview

```
 3 import ...
12 public class WordCount {
14                      public static class Map extends Mapper<Object, Text, Text, IntWritable> {
18                       public void map ...
26                      }
27
28                      public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable> {
30                      public void reduce ...
37                      }
38
39                      public static void main(String[] args) throws Exception {
40                      Job job = Job.getInstance(new Configuration(), "word count");;
41                      ...
53                      FileInputFormat.setInputPaths(conf, new Path(args[0]));
54                      FileOutputFormat.setOutputPath(conf, new Path(args[1]));
55
56                      System.exit(job.waitForCompletion(true) ? 0 : 1);;
57                      }
59 }
```

wordCount Mapper - Java

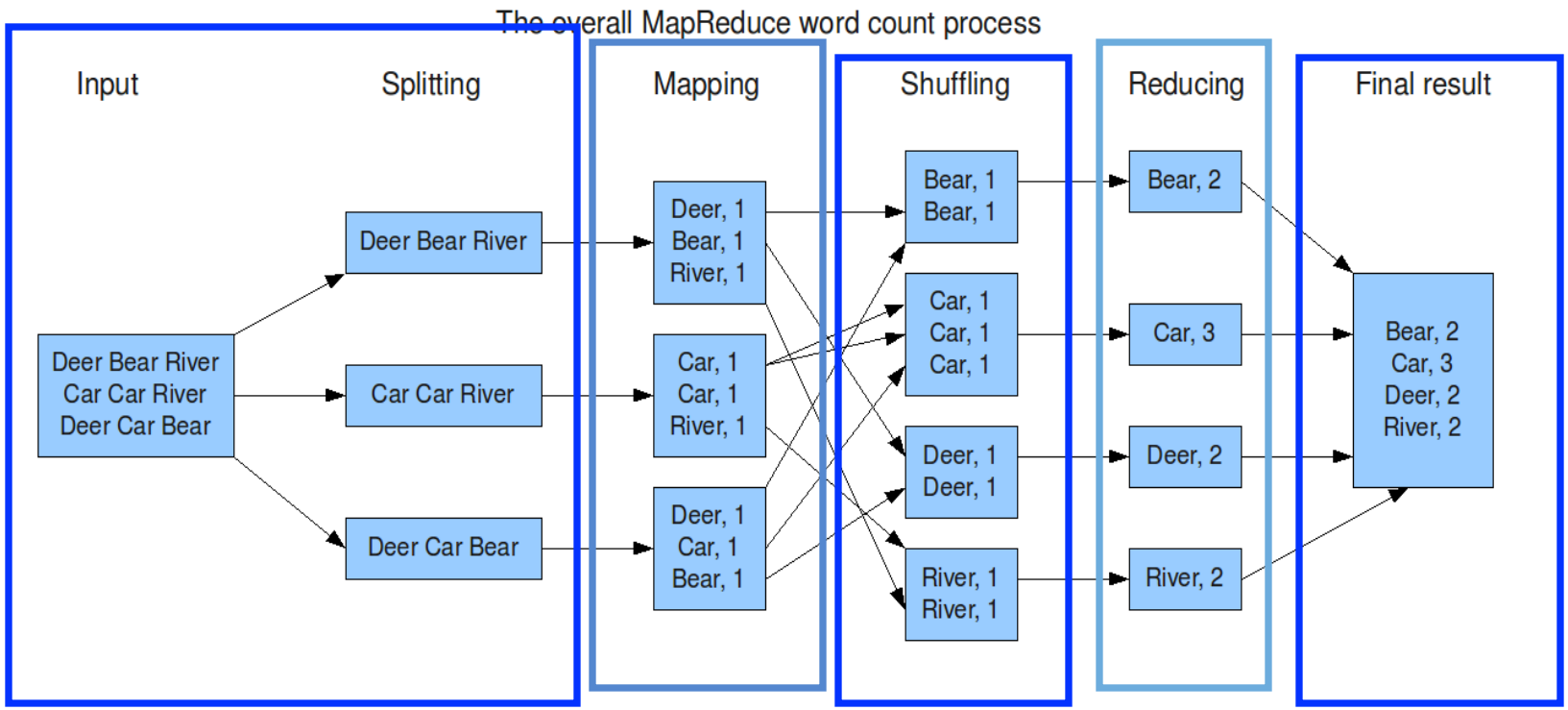
```
14  public static class Map extends Mapper<Object, Text, Text, IntWritable> {
15        private final static IntWritable one = new IntWritable(1);
16        private Text word = new Text();
17
18        public void map( Object key, Text value, Context context)
                    throws IOException {
19                        String line = value.toString();
20                        StringTokenizer tokenizer = new
StringTokenizer(line);
21                        while (tokenizer.hasMoreTokens()) {
22                              word.set(tokenizer.nextToken());
23                              context.write(word, one);
24                        }
25        }
26  }
```

wordCount Reducer - Jave

```
28  public static class Reduce extends extends Reducer<Text, IntWritable,
Text, IntWritable> {
29
30      public void reduce(Text key, Iterable<IntWritable> values,
                    Context context)
            throws IOException {
31                      int sum = 0;
32                      while (values.hasNext()) {
33                              sum += values.next().get();
34          }
35           result.set(sum);
36                      context.write(key, result);;
37      }
38  }
```

WordCount main - Java

```
public static void main(String[] args) throws Exception {
        Job job = Job.getInstance(new Configuration(), "word count");

        job.setJarByClass(WordCount.class);
        job.setMapperClass(Map.class);
        job.setCombinerClass(Reduce.class);
        job.setReducerClass(Reduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Mapper - Java

Maps input key-value pairs to a set of intermediate key-value pair

Class for Individual tasks to run.

One mapper task per *InputSplit.*

Map function are automatically called per key Value pair.

```
public static class Map extends Mapper<Object, Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map( Object key, Text value, Context context)
        throws IOException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
              word.set(tokenizer.nextToken());
              context.write(word, one);
        }
      }
}
```

Reducer - Java

Reduces the set of values of the same key to a smaller set.

- Each reducer will process subset generated by partitioner
- Each reducer will generate an output file

```
public static class Reduce extends extends Reducer<Text, IntWritable,  Text,
    IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values,
                Context context)  throws IOException {
      int sum = 0;
      while (values.hasNext()) {
            sum += values.next().get();
      }
      result.set(sum);
      context.write(key, result);
    }
}
```

Mapper - Python

```
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

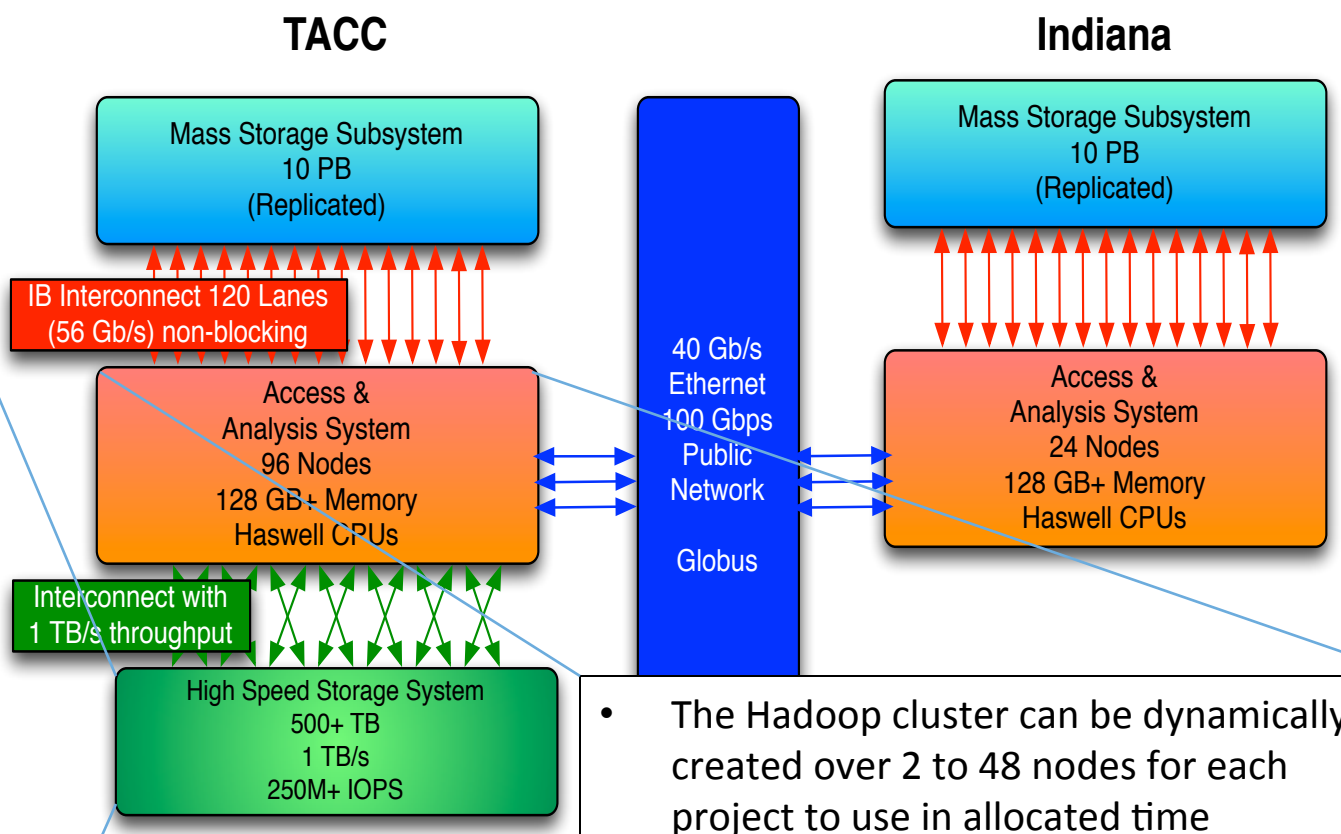
Reducer - Python

```
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
```

```
 # this IF-switch only works because Hadoop
sorts map output
    # by key (here: word) before it is passed to
the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word,
current_count)
        current_count = count
        current_word = word

# do not forget to output the last word if
needed!
if current_word == word:
    print '%s\t%s' % (current_word,
current_count)
```

Accessing Hadoop Cluster on Wrangler

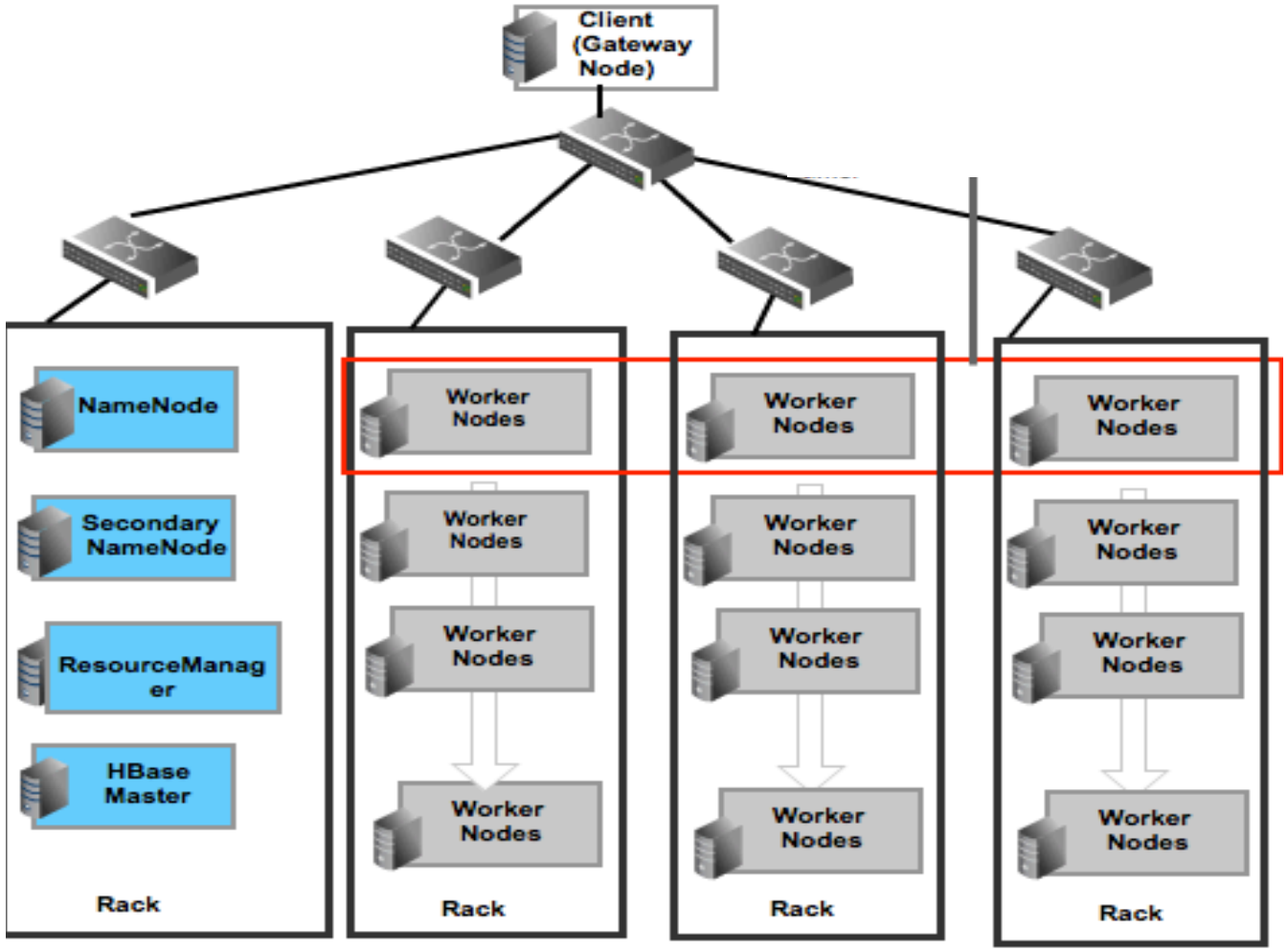
Wrangler



- A direct attached PCI interface allows access to the NAND flash.
- Not limited by networking connection
- Flash storage not tied to individual nodes

- The Hadoop cluster can be dynamically created over 2 to 48 nodes for each project to use in allocated time
- Each node has access to 4 TB flash storage across four channels
- Accessible via the Hadoop cluster via idev, batch job submission and VNC sessions.

A Typical Hadoop Infrastructure



Get Started with Hadoop on Wrangler

Step 1: cre… reservation through Wrangler data portal

What do…

Any…

Step 2: Access your Ha… jobs

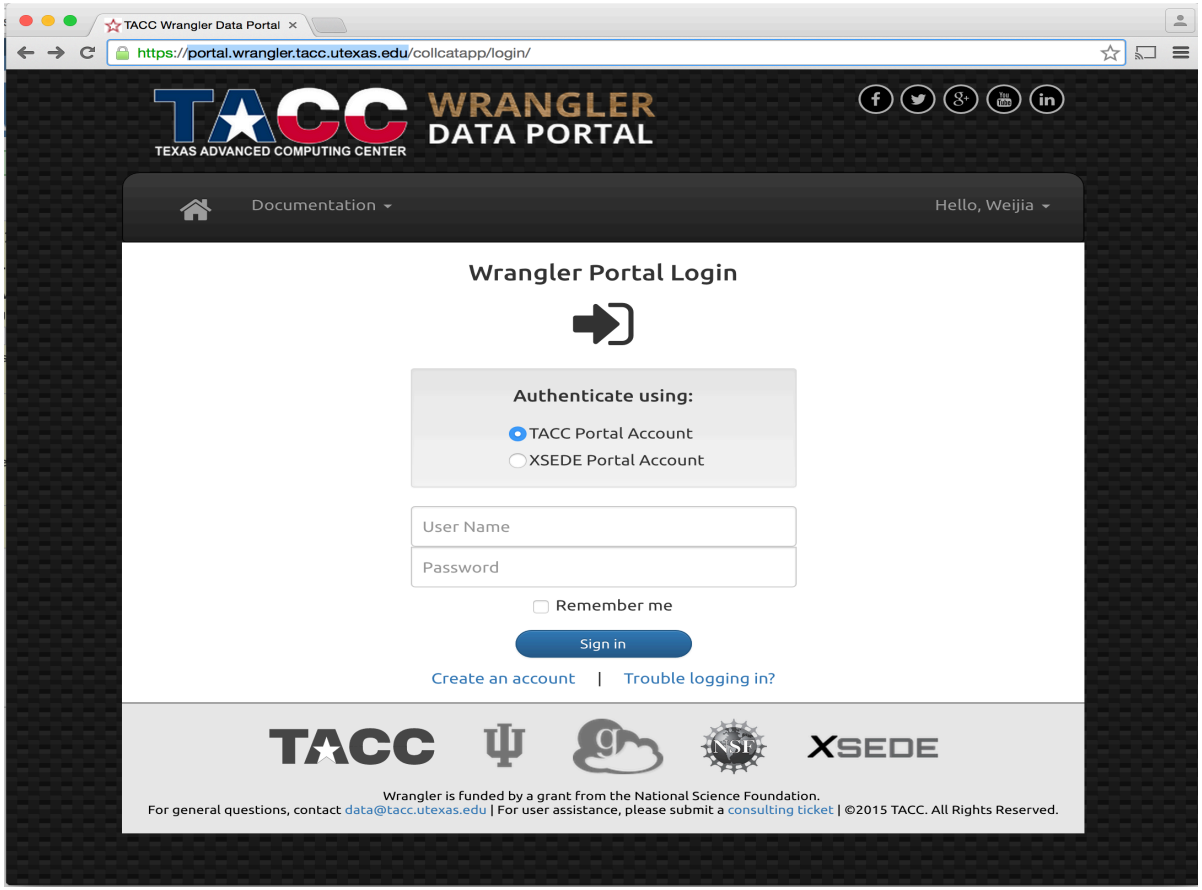
What do you need?

Secure Shell Client

Any VNC client

Create Hadoop Reservation

Wrangler data portal: portal.wrangler.tacc.utexas.edu

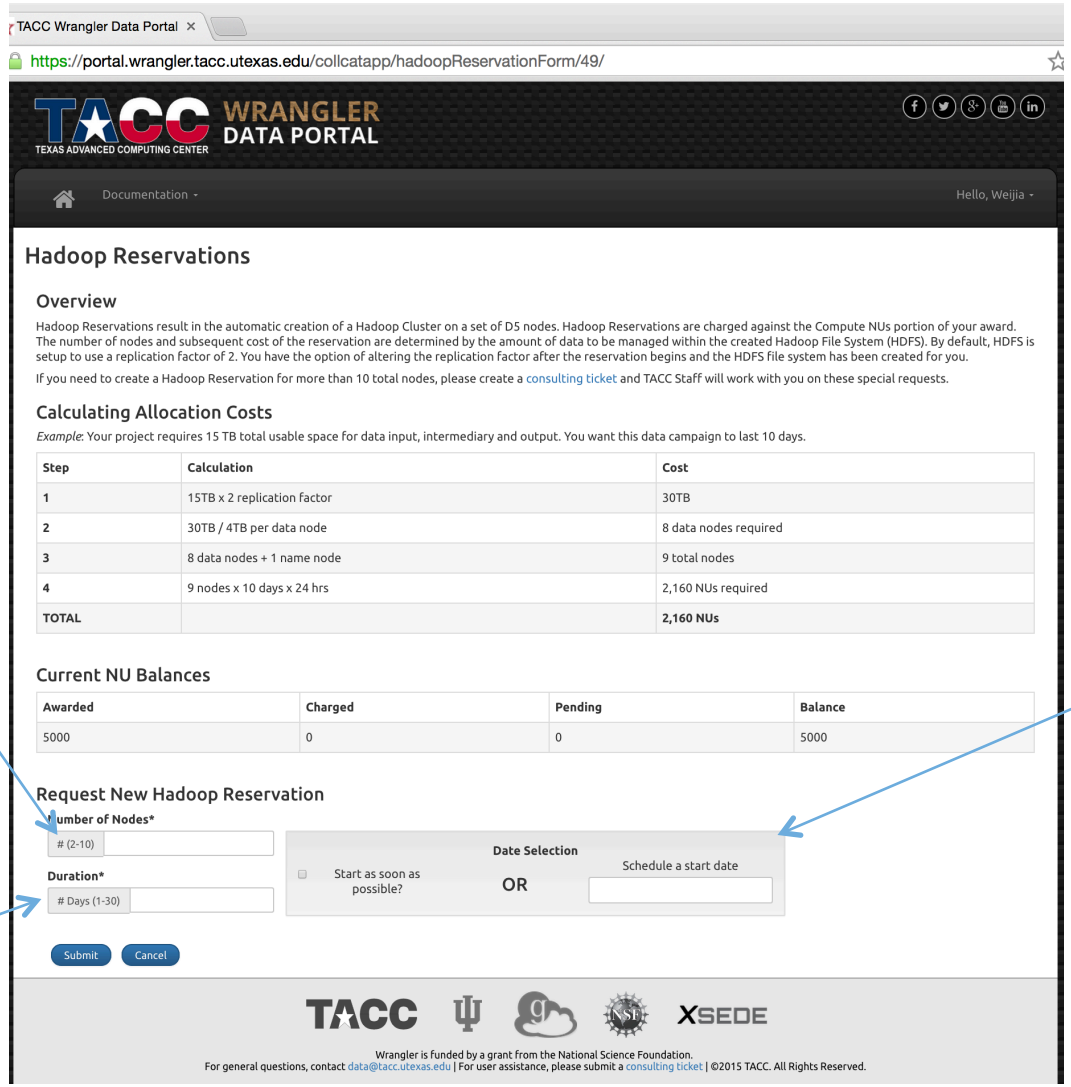


On project page choose: Manage -> Create Hadoop Reservation

the number of nodes (1 ~10) to be used for the Hadoop cluster.

Duration (1-30 Days)

Schedule Start time



Check Hadoop Reservation

- Once log on to Wrangler login node, user can check the reservation status with `scontrol` command:

>scontrol show reservation

```
login1.wrangler(9)$ scontrol show reservation
ReservationName=Service_nodes_equivalent StartTime=2017-03-28T12:59:00 EndTime=2017-04-25T13:00:00 Duration=28-00:01:00
   Nodes=c251-[140-143] NodeCnt=4 CoreCnt=192 Features=(null) PartitionName=(null) Flags=MAINT,OVERLAP,IGNORE_JOBS,SPEC_NODES
   Users=root,nthorne,plubbs,ctjordan,ngaffney,xwj Accounts=(null) Licenses=(null) State=ACTIVE

ReservationName=hadoop+PEMFS+2178 StartTime=2017-04-15T10:20:02 EndTime=2017-04-20T10:20:02 Duration=5-00:00:00
   Nodes=c252-[109-112] NodeCnt=4 CoreCnt=192 Features=(null) PartitionName=hadoop Flags=STATIC
   Users=gruan,xwj,agupta,rhuang,avliu,ychu,zzhang,parnell Accounts=(null) Licenses=(null) State=ACTIVE
```

- The reservation will include all users from the projects
- The first node in the reservation will be used as ***namenode***
- The hadoop cluster will start with a set of default settings
  - User may override most settings such as duplication factor, block size at run time per application.
  - Hadoop cluster with specific settings upon request

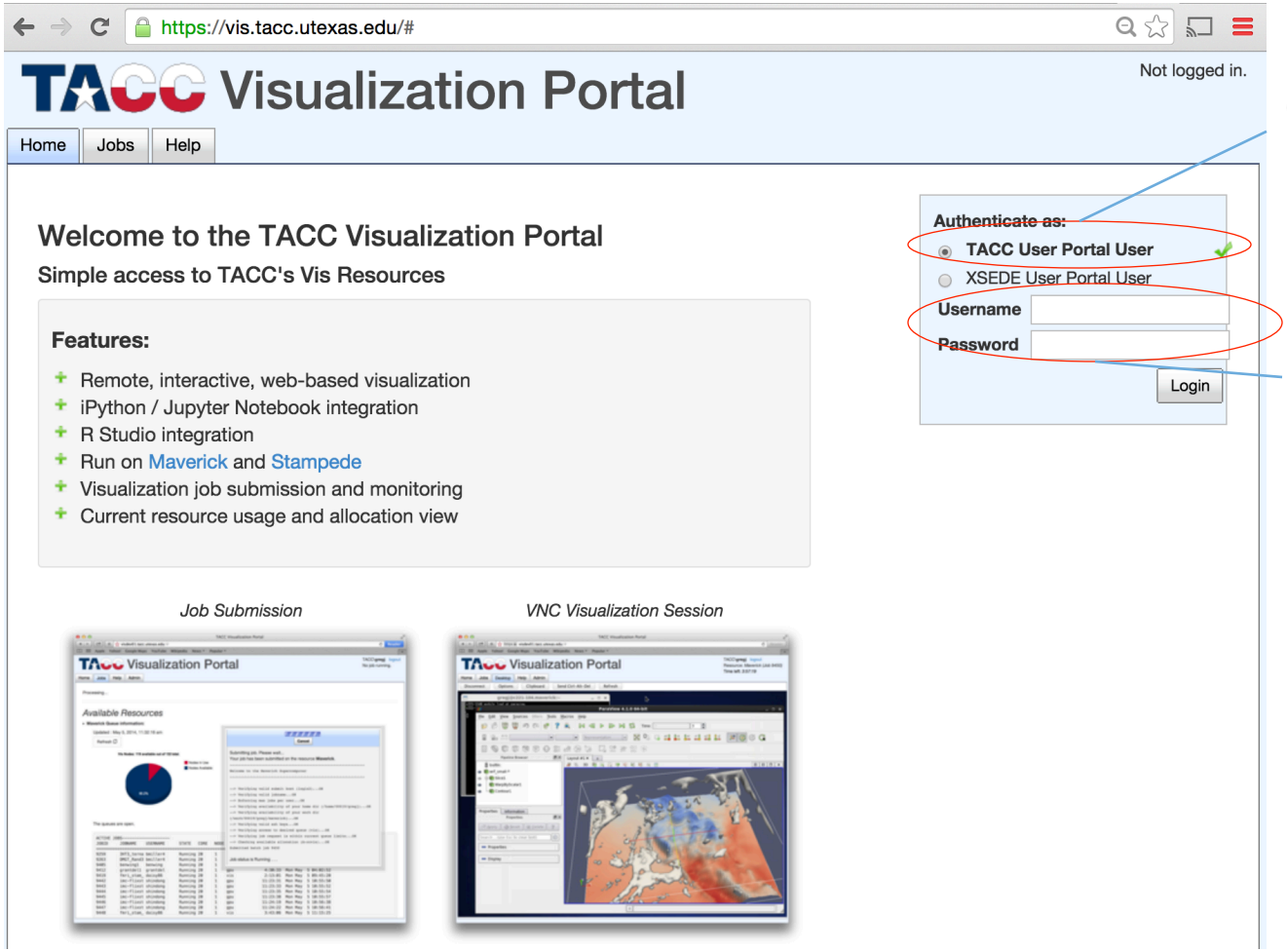
Access Hadoop Reservation

Once the reservation status is *“active”,* a user can access through slurm job:

- VNC job: starts a vnc server session on one of the node in Hadoop cluster,
  - Check cluster information and hadoop job status
  - Application with Graphical/Web user interface
- idev job: Assign one node in Hadoop cluster to user
  - Manage data in and out hadoop cluster,
  - Submit Hadoop jobs via command line
  - Code testing
- Batch job: submitting jobs to YARN resource manager in Hadoop cluster.
  - Submit large analysis job
  - Submit batch of processing jobs to run sequentially

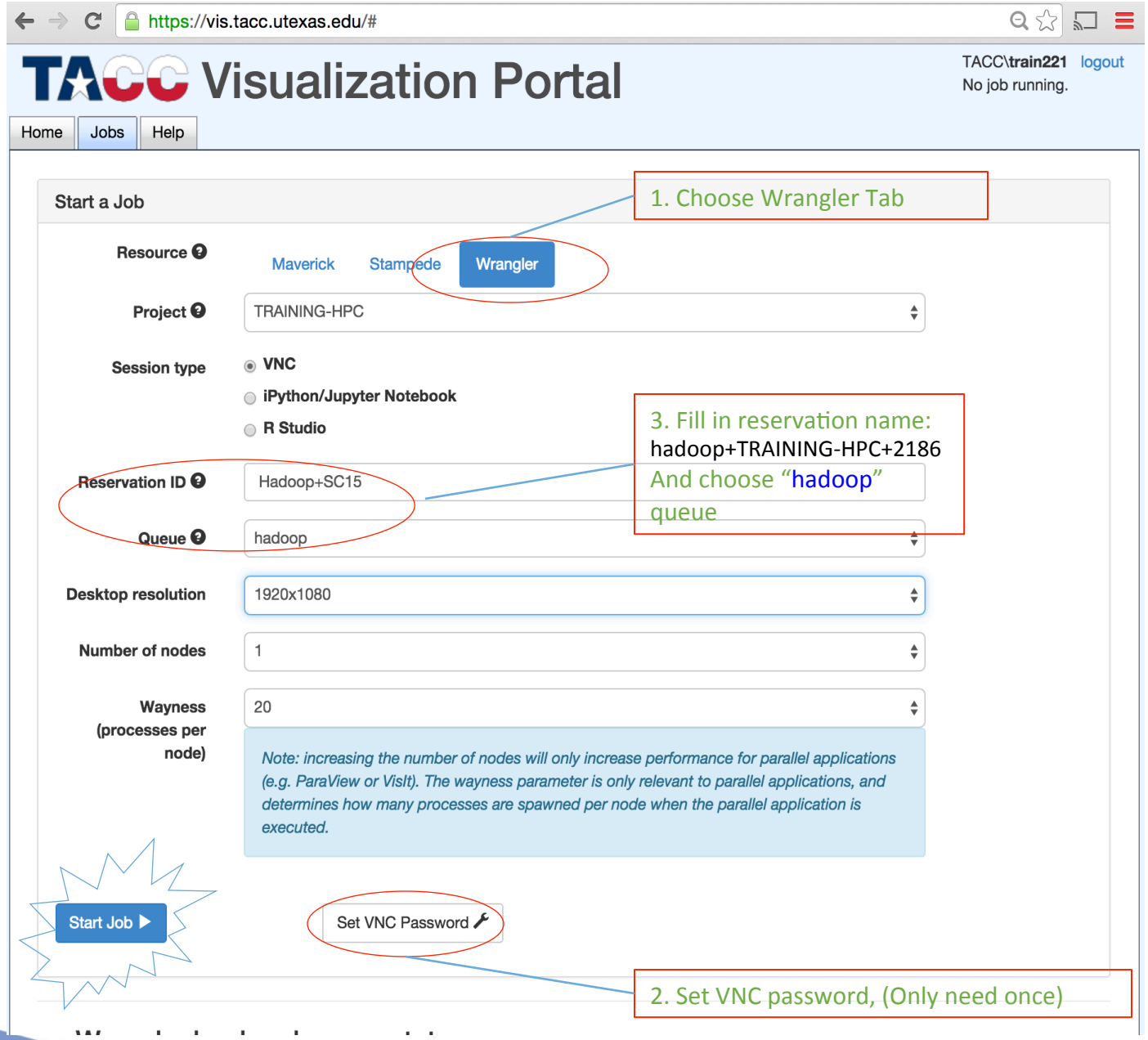
Access Hadoop Cluster with VNC

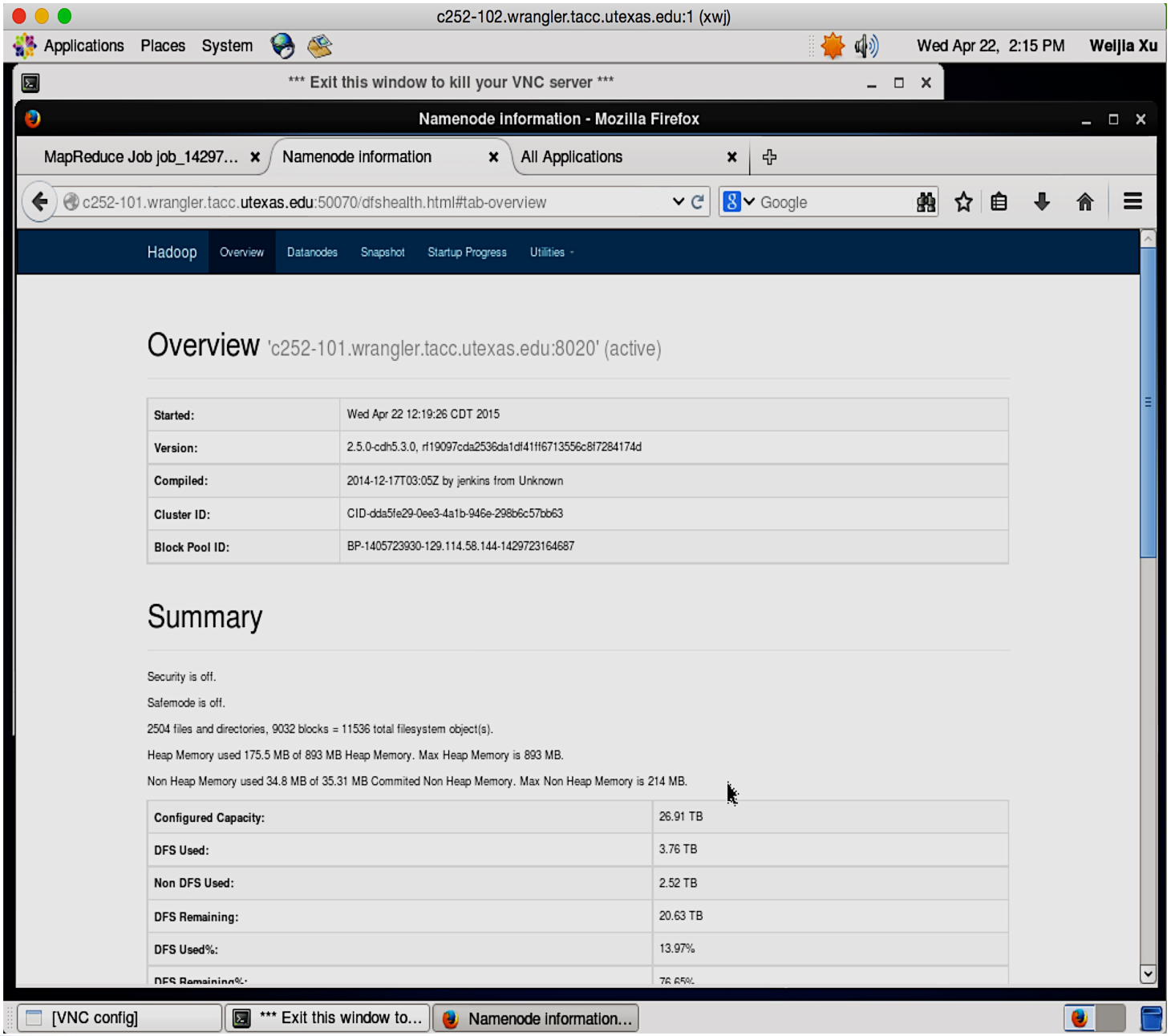
Please visit: vis.tacc.utexas.edu

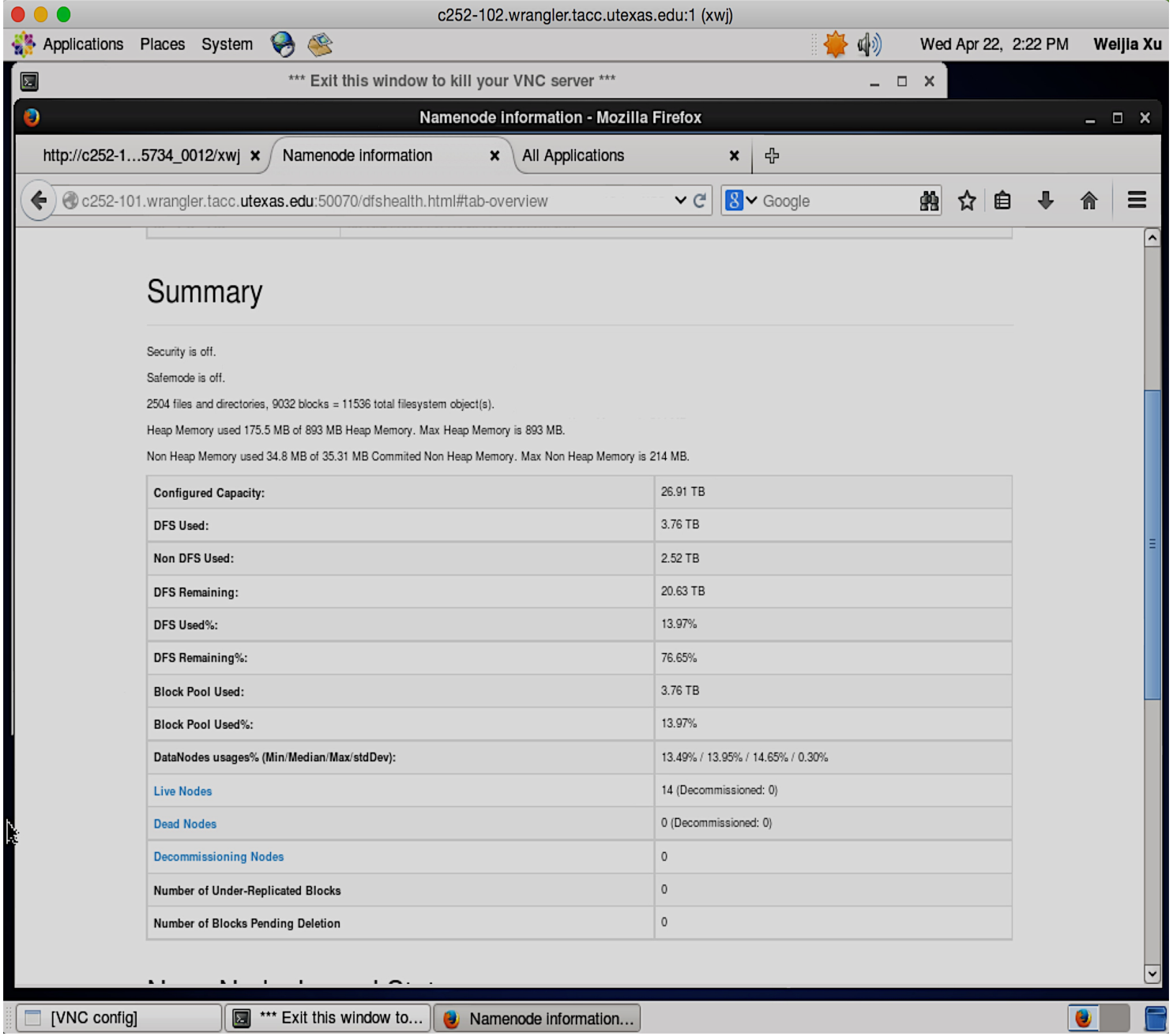


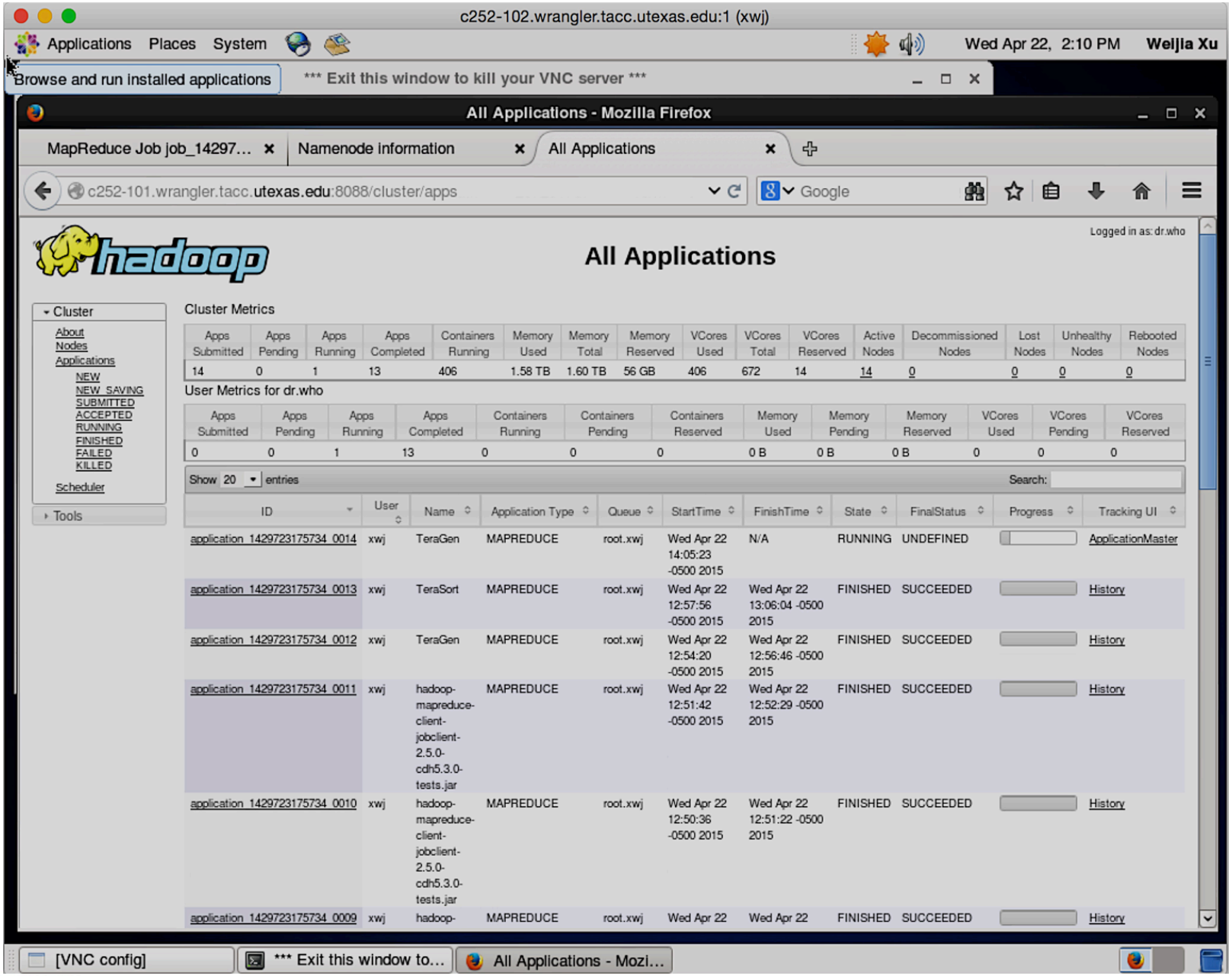
Choose “TACC User Portal User”

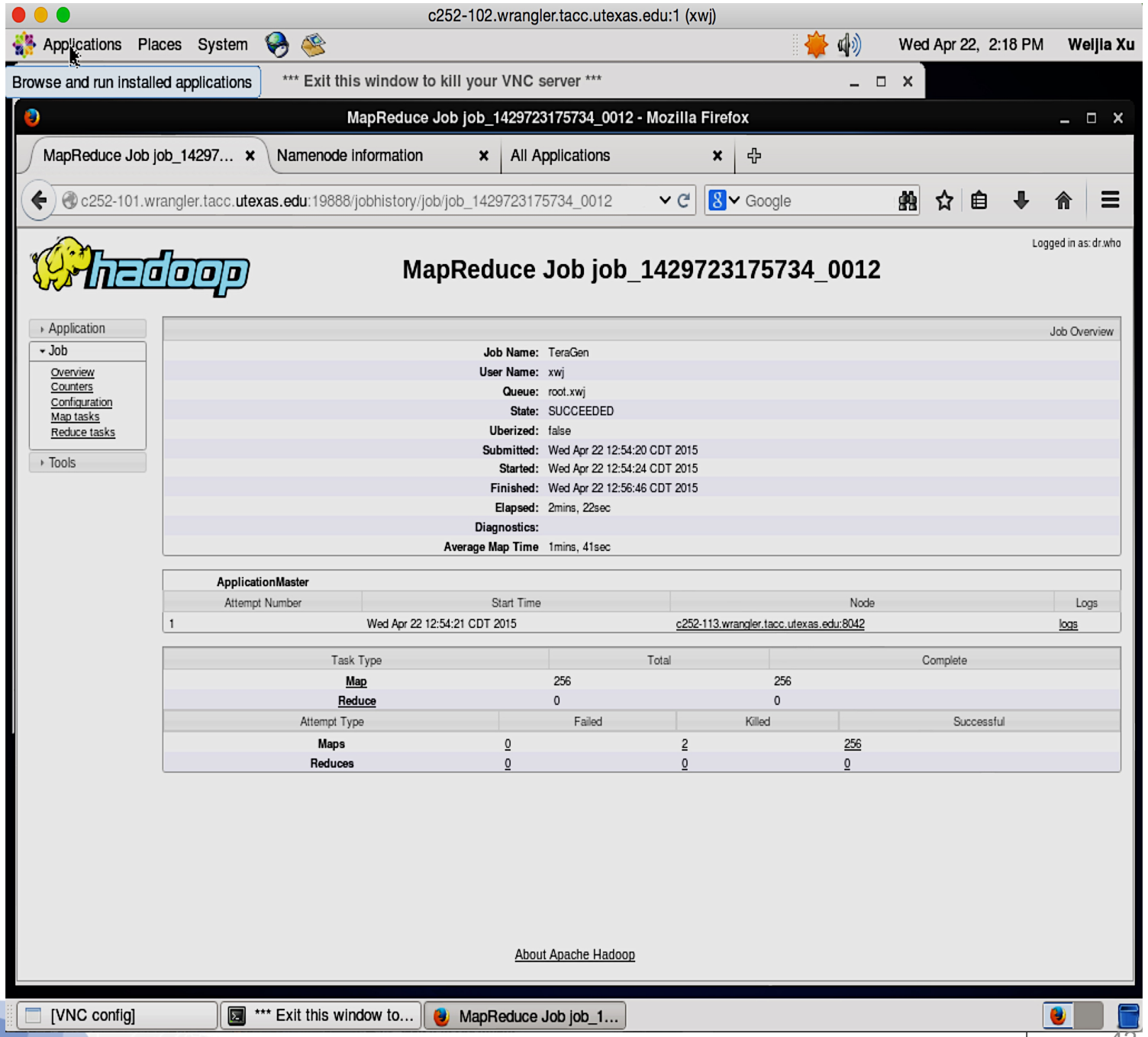
Enter credential


https://vis.tacc.utexas.edu/#
TACC Visualization Portal
TACC\train221 logout
No job running.
Home
Jobs
Help
Start a Job
1. Choose Wrangler Tab
Resource
Maverick
Stampede
Wrangler
Project
TRAINING-HPC
Session type
VNC
iPython/Jupyter Notebook
R Studio
3. Fill in reservation name:
hadoop+TRAINING-HPC+2186
And choose "hadoop" queue
Reservation ID
Hadoop+SC15
Queue
hadoop
Desktop resolution
1920x1080
Number of nodes
1
Wayness (processes per node)
20
Note: increasing the number of nodes will only increase performance for parallel applications (e.g. ParaView or VisIt). The wayness parameter is only relevant to parallel applications, and determines how many processes are spawned per node when the parallel application is executed.
Start Job
Set VNC Password
2. Set VNC password, (Only need once)


c252-102.wrangler.tacc.utexas.edu:1 (xwj)
Applications Places System
Wed Apr 22, 2:15 PM Weijia Xu
*** Exit this window to kill your VNC server ***
Namenode information - Mozilla Firefox
MapReduce Job job_14297...
Namenode information
All Applications
c252-101.wrangler.tacc.utexas.edu:50070/dfshealth.html#tab-overview
Google
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Overview 'c252-101.wrangler.tacc.utexas.edu:8020' (active)
Started: Wed Apr 22 12:19:26 CDT 2015
Version: 2.5.0-cdh5.3.0, rl19097cda2536da1df41ff6713556c8f7284174d
Compiled: 2014-12-17T03:05Z by jenkins from Unknown
Cluster ID: CID-dda5fe29-0ee3-4a1b-946e-298b6c57bb63
Block Pool ID: BP-1405723930-129.114.58.144-1429723164687
Summary
Security is off.
Safemode is off.
2504 files and directories, 9032 blocks = 11536 total filesystem object(s).
Heap Memory used 175.5 MB of 893 MB Heap Memory. Max Heap Memory is 893 MB.
Non Heap Memory used 34.8 MB of 35.31 MB Commited Non Heap Memory. Max Non Heap Memory is 214 MB.
Configured Capacity: 26.91 TB
DFS Used: 3.76 TB
Non DFS Used: 2.52 TB
DFS Remaining: 20.63 TB
DFS Used%: 13.97%
[VNC config]
*** Exit this window to...
Namenode information...


c252-102.wrangler.tacc.utexas.edu:1 (xwj)
Applications Places System
Wed Apr 22, 2:22 PM Weijia Xu
*** Exit this window to kill your VNC server ***
Namenode information - Mozilla Firefox
http://c252-1...5734_0012/xwj
Namenode information
All Applications
c252-101.wrangler.tacc.utexas.edu:50070/dfshealth.html#tab-overview
Google
Summary
Security is off.
Safemode is off.
2504 files and directories, 9032 blocks = 11536 total filesystem object(s).
Heap Memory used 175.5 MB of 893 MB Heap Memory. Max Heap Memory is 893 MB.
Non Heap Memory used 34.8 MB of 35.31 MB Commited Non Heap Memory. Max Non Heap Memory is 214 MB.
Configured Capacity: 26.91 TB
DFS Used: 3.76 TB
Non DFS Used: 2.52 TB
DFS Remaining: 20.63 TB
DFS Used%: 13.97%
DFS Remaining%: 76.65%
Block Pool Used: 3.76 TB
Block Pool Used%: 13.97%
DataNodes usages% (Min/Median/Max/stdDev): 13.49% / 13.95% / 14.65% / 0.30%
Live Nodes 14 (Decommissioned: 0)
Dead Nodes 0 (Decommissioned: 0)
Decommissioning Nodes 0
Number of Under-Replicated Blocks 0
Number of Blocks Pending Deletion 0
[VNC config]
*** Exit this window to...
Namenode information...


c252-102.wrangler.tacc.utexas.edu:1 (xwj)
Applications Places System
Wed Apr 22, 2:10 PM Weijia Xu
Browse and run installed applications
*** Exit this window to kill your VNC server ***
All Applications - Mozilla Firefox
MapReduce Job job_14297...
Namenode information
All Applications
c252-101.wrangler.tacc.utexas.edu:8088/cluster/apps
Google
hadoop
Logged in as: dr.who
All Applications
Cluster
About
Nodes
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools
Cluster Metrics
Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | VCores Used | VCores Total | VCores Reserved | Active Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes | Rebooted Nodes
14 | 0 | 1 | 13 | 406 | 1.58 TB | 1.60 TB | 56 GB | 406 | 672 | 14 | 14 | 0 | 0 | 0 | 0
User Metrics for dr.who
Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Containers Pending | Containers Reserved | Memory Used | Memory Pending | Memory Reserved | VCores Used | VCores Pending | VCores Reserved
0 | 0 | 1 | 13 | 0 | 0 | 0 | 0 B | 0 B | 0 B | 0 | 0 | 0
Show 20 entries
Search:
ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress | Tracking UI
application_1429723175734_0014 | xwj | TeraGen | MAPREDUCE | root.xwj | Wed Apr 22 14:05:23 -0500 2015 | N/A | RUNNING | UNDEFINED | ApplicationMaster
application_1429723175734_0013 | xwj | TeraSort | MAPREDUCE | root.xwj | Wed Apr 22 12:57:56 -0500 2015 | Wed Apr 22 13:06:04 -0500 2015 | FINISHED | SUCCEEDED | History
application_1429723175734_0012 | xwj | TeraGen | MAPREDUCE | root.xwj | Wed Apr 22 12:54:20 -0500 2015 | Wed Apr 22 12:56:46 -0500 2015 | FINISHED | SUCCEEDED | History
application_1429723175734_0011 | xwj | hadoop-mapreduce-client-jobclient-2.5.0-cdh5.3.0-tests.jar | MAPREDUCE | root.xwj | Wed Apr 22 12:51:42 -0500 2015 | Wed Apr 22 12:52:29 -0500 2015 | FINISHED | SUCCEEDED | History
application_1429723175734_0010 | xwj | hadoop-mapreduce-client-jobclient-2.5.0-cdh5.3.0-tests.jar | MAPREDUCE | root.xwj | Wed Apr 22 12:50:36 -0500 2015 | Wed Apr 22 12:51:22 -0500 2015 | FINISHED | SUCCEEDED | History
application_1429723175734_0009 | xwj | hadoop- | MAPREDUCE | root.xwj | Wed Apr 22 | Wed Apr 22 | FINISHED | SUCCEEDED | History
[VNC config]
*** Exit this window to...
All Applications - Mozi...


c252-102.wrangler.tacc.utexas.edu:1 (xwj)
Applications Places System
Wed Apr 22, 2:18 PM Weijia Xu
Browse and run installed applications
*** Exit this window to kill your VNC server ***
MapReduce Job job_1429723175734_0012 - Mozilla Firefox
MapReduce Job job_14297...
Namenode information
All Applications
c252-101.wrangler.tacc.utexas.edu:19888/jobhistory/job/job_1429723175734_0012
Google
Logged in as: dr.who
hadoop
MapReduce Job job_1429723175734_0012
Application
Job
Overview
Counters
Configuration
Map tasks
Reduce tasks
Tools
Job Overview
Job Name: TeraGen
User Name: xwj
Queue: root.xwj
State: SUCCEEDED
Uberized: false
Submitted: Wed Apr 22 12:54:20 CDT 2015
Started: Wed Apr 22 12:54:24 CDT 2015
Finished: Wed Apr 22 12:56:46 CDT 2015
Elapsed: 2mins, 22sec
Diagnostics:
Average Map Time 1mins, 41sec
ApplicationMaster
Attempt Number
Start Time
Node
Logs
1
Wed Apr 22 12:54:21 CDT 2015
c252-113.wrangler.tacc.utexas.edu:8042
logs
Task Type
Total
Complete
Map
256
256
Reduce
0
0
Attempt Type
Failed
Killed
Successful
Maps
0
2
256
Reduces
0
0
0
About Apache Hadoop
[VNC config]
*** Exit this window to...
MapReduce Job job_1...

Access Hadoop Reservation via idev Session

User can submit idev session to hadoop cluster reservation

- ***idev –r*** hadoop+TRAINING-HPC+2186

It defaults to use your default project,

The ***–A allocaiton_name*** option to specify allocation to use

The default duration for idev is 30 minutes

The ***-m minutes*** option can specify the time of the idev session

Please limit your usage to Hadoop related tasks, you can also submit idev without using reservation for non-hadoop tasks.

Access by secure shell client

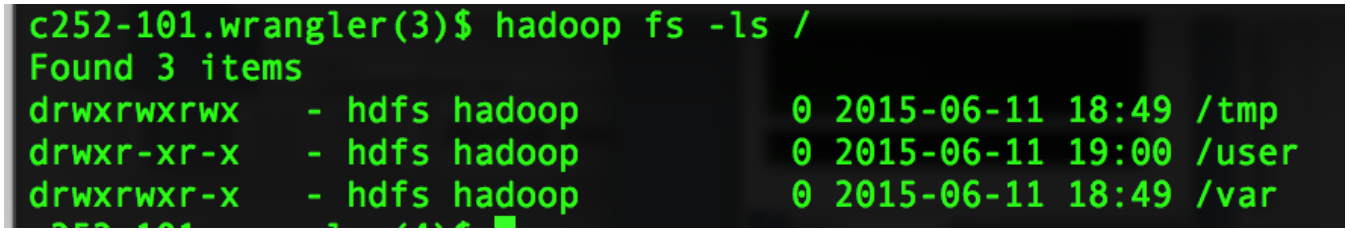
ssh username@wrangler.tacc.utexas.edu

idev –r hadoop+TRAINING-HPC+2186 –m 240 –p hadoop

- Access by vis portal
  - Go to vis.tacc.utexas.edu using web browser
  - Login with your credential
  - Goto Wrangler tab to start VNC sessions using reservation hadoop+TRAINING-HPC+2186 **and using Hadoop queue**

Hadoop Distributed File System (HDFS)

The hdfs will be set up with three top level directories



/tmp public writeable, used by many hadoop based application as temporary space.

/user all users home directory /user/$USERNAME

/var public readable, used by many hadoop based application to store log files etc.

Working with HDFS

HDFS has file system shell

hadoop fs [commands]

The file system shell includes a set of command to work with hdfs.

Command are similar to common linux commands e.g.

```
>hadoop fs -ls   #to list content of the default user directory.
>hadoop fs -mkdir abc #to make a directory in hdfs.
```

Getting Data in and out HDFS

hadoop fs -put local_file [path_in_HDFS]

- Put a file in your local system into the HDFS
- Each file would be stored in one or more “blocks”
  - The default block size is 128MB.
  - The block size used can be override by users

Hadoop fs -get path_in_hdfs [path_in_local]

- Get a file from the hadoop cluster to the local file system.

Other File Shell commands

-stat returns stat information of a path

-cat/tail output to stdout

-setrep set replication factor

For a complete lists just do

hadoop fs

Or

visit http://hadoop.apache.org/docs/r2.5.2/hadoop-project-dist/hadoop-common/FileSystemShell.html

YARN

YARN: Yet Another Resource Manager

- Managing computing resources within Hadoop cluster
- All jobs should be submitted to yarn to run.
  - E.g. using either ***yarn Jar*** or ***hadoop Jar***
- When use other hadoop-supported application, please also specify YARN as resource manager. Such as SPARK.

YARN commands

- Show cluster status
- Help managing jobs running inside the Hadoop cluster.

YARN Commands

yarn application

-list to list applications submitted to YARN

default will show active/queued application

```
Total number of applications (application-types: [] and st
                Application-Id      Application-Name
application_1431812610805_4832                GSO108
```

-kill to kill application specified by application-ID.

-appStates/appTypes filter options

YARN Commands

yarn node

-list list of status of data nodes

Let us know if there is less than expected live data nodes.

yarn logs

dump logs of a finished application

-applicationID specific log from which application

-containerID specify log from which container

Running Hadoop Application

All Hadoop application can be run as a console command.

The basic format is like following:

hadoop jar java_jar_name java_class_name [parameters]

The user can use –D to specify more Hadoop options. e.g.

-D mapred.map.tasks #number of map instances to be generated.

-D mapred.reduce.tasks # number of reduce instances to be used.

Running Wordcount example code

The Hadoop distribution comes with an example jar which includes a set of exemplar map reduce code:

To run wordcount example from that jar file, you can run command like following:

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar \
  wordcount   \  #java class name to run
  -D mapred.map.tasks=500     \  #number of mapper instance
  -D mapred.reduce.tasks=256 \ # number of reducer instance
  /tmp/data/enwiki-20120104-pages-articles.xml  \    #input file on hdfs
  wiki_wc     #folder to store the output
```

Running Wordcount Example code

The on-screen output will show job running status

```
15/05/29 02:50:59 INFO client.RMProxy: Connecting to ResourceManager at name.rustler.tacc.utexas.edu/129.114.57.132:8032
15/05/29 02:51:01 INFO input.FileInputFormat: Total input paths to process : 1
15/05/29 02:51:02 INFO mapreduce.JobSubmitter: number of splits:266
15/05/29 02:51:02 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
15/05/29 02:51:02 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
15/05/29 02:51:02 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1431812610805_3390
15/05/29 02:51:03 INFO impl.YarnClientImpl: Submitted application application_1431812610805_3390
15/05/29 02:51:03 INFO mapreduce.Job: The url to track the job: http://name.rustler.tacc.utexas.edu:8088/proxy/application_1431812610805_3390/
15/05/29 02:51:03 INFO mapreduce.Job: Running job: job_1431812610805_3390
15/05/29 02:51:09 INFO mapreduce.Job: Job job_1431812610805_3390 running in uber mode : false
15/05/29 02:51:09 INFO mapreduce.Job:  map 0% reduce 0%
15/05/29 02:51:20 INFO mapreduce.Job:  map 14% reduce 0%
15/05/29 02:51:21 INFO mapreduce.Job:  map 24% reduce 0%
15/05/29 02:51:22 INFO mapreduce.Job:  map 32% reduce 0%
15/05/29 02:51:23 INFO mapreduce.Job:  map 45% reduce 0%
15/05/29 02:51:24 INFO mapreduce.Job:  map 52% reduce 0%
15/05/29 02:51:25 INFO mapreduce.Job:  map 57% reduce 0%
15/05/29 02:51:26 INFO mapreduce.Job:  map 62% reduce 0%
15/05/29 02:51:27 INFO mapreduce.Job:  map 65% reduce 0%
15/05/29 02:51:28 INFO mapreduce.Job:  map 66% reduce 0%
15/05/29 02:51:30 INFO mapreduce.Job:  map 67% reduce 0%
15/05/29 02:51:38 INFO mapreduce.Job:  map 68% reduce 0%
15/05/29 02:51:39 INFO mapreduce.Job:  map 72% reduce 0%
15/05/29 02:51:40 INFO mapreduce.Job:  map 76% reduce 0%
15/05/29 02:51:41 INFO mapreduce.Job:  map 81% reduce 0%
15/05/29 02:51:42 INFO mapreduce.Job:  map 87% reduce 0%
15/05/29 02:51:43 INFO mapreduce.Job:  map 93% reduce 0%
15/05/29 02:51:44 INFO mapreduce.Job:  map 97% reduce 0%
15/05/29 02:51:45 INFO mapreduce.Job:  map 98% reduce 0%
15/05/29 02:51:46 INFO mapreduce.Job:  map 99% reduce 0%
15/05/29 02:51:47 INFO mapreduce.Job:  map 100% reduce 0%
15/05/29 02:51:49 INFO mapreduce.Job:  map 100% reduce 10%
15/05/29 02:51:50 INFO mapreduce.Job:  map 100% reduce 24%
15/05/29 02:51:51 INFO mapreduce.Job:  map 100% reduce 42%
15/05/29 02:51:52 INFO mapreduce.Job:  map 100% reduce 76%
15/05/29 02:51:53 INFO mapreduce.Job:  map 100% reduce 98%
15/05/29 02:51:54 INFO mapreduce.Job:  map 100% reduce 100%
15/05/29 02:51:57 INFO mapreduce.Job: Job job_1431812610805_3390 completed successfully
15/05/29 02:51:58 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=12501618427
                FILE: Number of bytes written=25053078456
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=35618393055
                HDFS: Number of bytes written=6489193532
                HDFS: Number of read operations=1566
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=512
        Job Counters
                Launched map tasks=266
                Launched reduce tasks=256
                Data-local map tasks=213
                Rack-local map tasks=53
                Total time spent by all maps in occupied slots (ms)=15587438
                Total time spent by all reduces in occupied slots (ms)=5572760
                Total time spent by all map tasks (ms)=7793719
                Total time spent by all reduce tasks (ms)=2786380
```

Parallelism with HDFS

The number of actual mapper created may be limited by the number of actual data blocks in the hdfs.

- In the example, the input file is about ~35GB, with default block size as 128MB, the file is stored in 266 blocks in hdfs.
- All mappers may not be run at the same time, if there is not enough resources.

Each reducer will generate an output file independent from each other.

- So 256 reducer would result 256 files in the output folder.

Running WordCount Example

Each output file is a text file as well,

- Each line contains a word and its count,
- You will notice for each output file, the word is sorted in alphabetic order
- You can copy the file out of hdfs using command like
  - Hadoop fs –get /tmp/wiki_wc wiki_wc
- Or you can view the content of each output using command like
  - Hadoop fs –cat /tmp/wiki_wc/part-r-00238 | more

Other Examples

There are more examples in the example.jar. You can use following command to get the list

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar
```

Here are a few examples:

- grep: A map/reduce program that counts the matches of a regex in the input.
- pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
- terasort: sorting large set of random generated 100bytes data
- sudoku: A sudoku solver.

Recap

- Hadoop and YARN
- Word Count Example – key value pair
  - Mapper (word, 1)
  - Reducer (word,count)
- Create a Hadoop Reservation
  - https://portal.wrangler.tacc.utexas.edu/
- Access Hadoop Reservation
  - VNC job – access Hadoop web UI etc.(TACC vis portal)
  - Idev job – interactive development/debugging
  - Batch job – submit bug-free code to Hadoop cluster

Try It Out

- Example code/data location

  /work/00791/xwj/DMS/hadoop-training/

  Copy to your home directory e.g.

  > cp –r /work/00791/xwj/DMS/hadoop-training ~/hadoop

  ➢ cd ~/hadoop

Take a look of exercise.txt (https://goo.gl/pUXO7g)

Try exercise 1 and exercise 2 first

hadoop fs command syntax

hadoop fs –{ls | rm | cat| mkdir | put | get}

Using Hadoop with other programming language

Enabling MR jobs in other scripting language: Python, Perl, R, C, etc...

User need to provide scripts/programs for Map and Reduce processing

- The input/output format need to be compatible with key-value pair

Intermediate data are passed through stdin, stdout

- A trade-off between convenience and performance

Hadoop Streaming API

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar
    -input /path/to/input/in/hdfs                    #input file location
    -output /path/to/output/in/hdfs                  #output file location
    -mapper map                                      # mapper
implementation
     -reducer reduce                                 #reduce
implementation
     -file map                  #location of the map code on local file
system
     -file reduce               # location of the reduce code on local file
system.
```

The map and reduce could be implemented in any programming language, even with bash script.

WordCount using Hadoop with bash

Putting together:

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar  \
    -D mapred.map.tasks=512  \
    -D mapred.reduce.tasks=256  \
    -D stream.num.map.output.key.fields=1 \  # specify the position of the key
     -input /tmp/data/20news-all/alt.atheism \
     -output wiki_wc_bash \
     -mapper ./mapwc.sh -reducer ./reducewc.sh \
     -file ./mapwc.sh -file ./reducewc.sh
```

WordCount using using Hadoop with Python

```
hadoop jar $HADOOP_STREAMING \
-D mapred.map.tasks=4 -D mapred.reduce.tasks=2 \
-file mapper.py -mapper mapper.py \
-file reducer.py -reducer reducer.py \
-input /user/$USER/data/book.txt  -output /user/$USER/output-streaming-py
```

Hadoop vs Spark

Parameters	Spark	Hadoop
Data Storage	Spark stores data in-memory.	Hadoop stores data on disk.
Fault tolerance	Spark's data storage model, resilient distributed datasets (RDD) guarantees fault tolerance.	It uses replication to achieve fault tolerance.
Line of code	Apache Spark is project of 20,000 Line of code.	Hadoop 2.0 has 1,20,000 Line of code
Speed	It is Faster due to In-memory computation.	It is relatively slower than Spark.
OS Support	• Linux • Windows • Mac OS	• Linux
High level language	• Scala • Python • Java • R	• Java
Streaming data	Spark can be used to process as well as modify real-time data with Spark streaming.	With Hadoop Map-Reduce one can process batch of stored data.
Machine Learning	Spark has its own set of Machine learning libraries (MLib).	Hadoop requires interface with other Machine learning library. Eg: Apache Mahout.

Introduction of Zeppelin

- A web-based notebook that enables interactive data analytics and visualization.
- Multiple Language Backend

Start Zeppelin on Wrangler

login1.wrangler(5)$cd $DATA

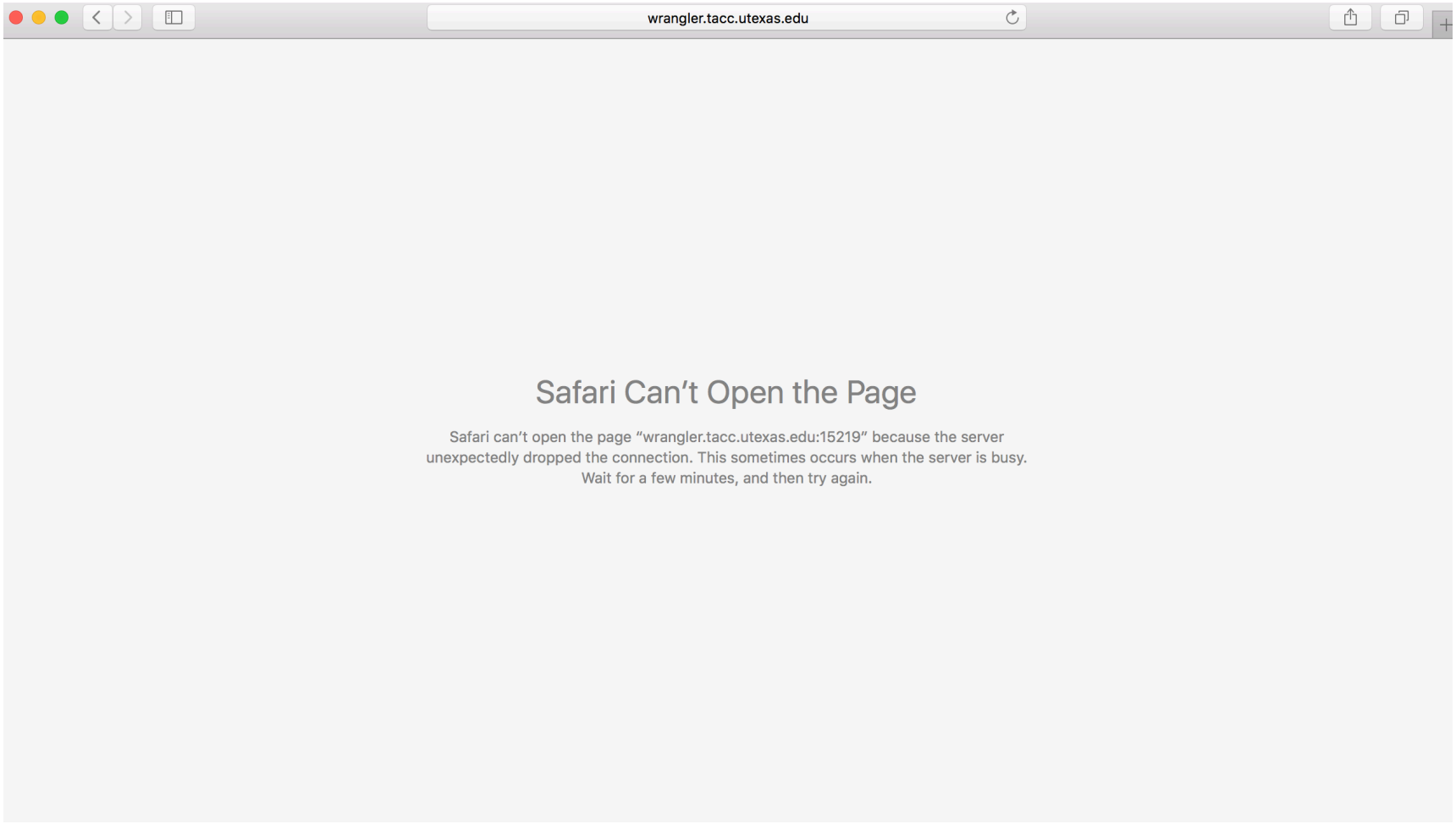
login1.wrangler(6)$cp /data/apps/.zeppelin/job.zeppelin .

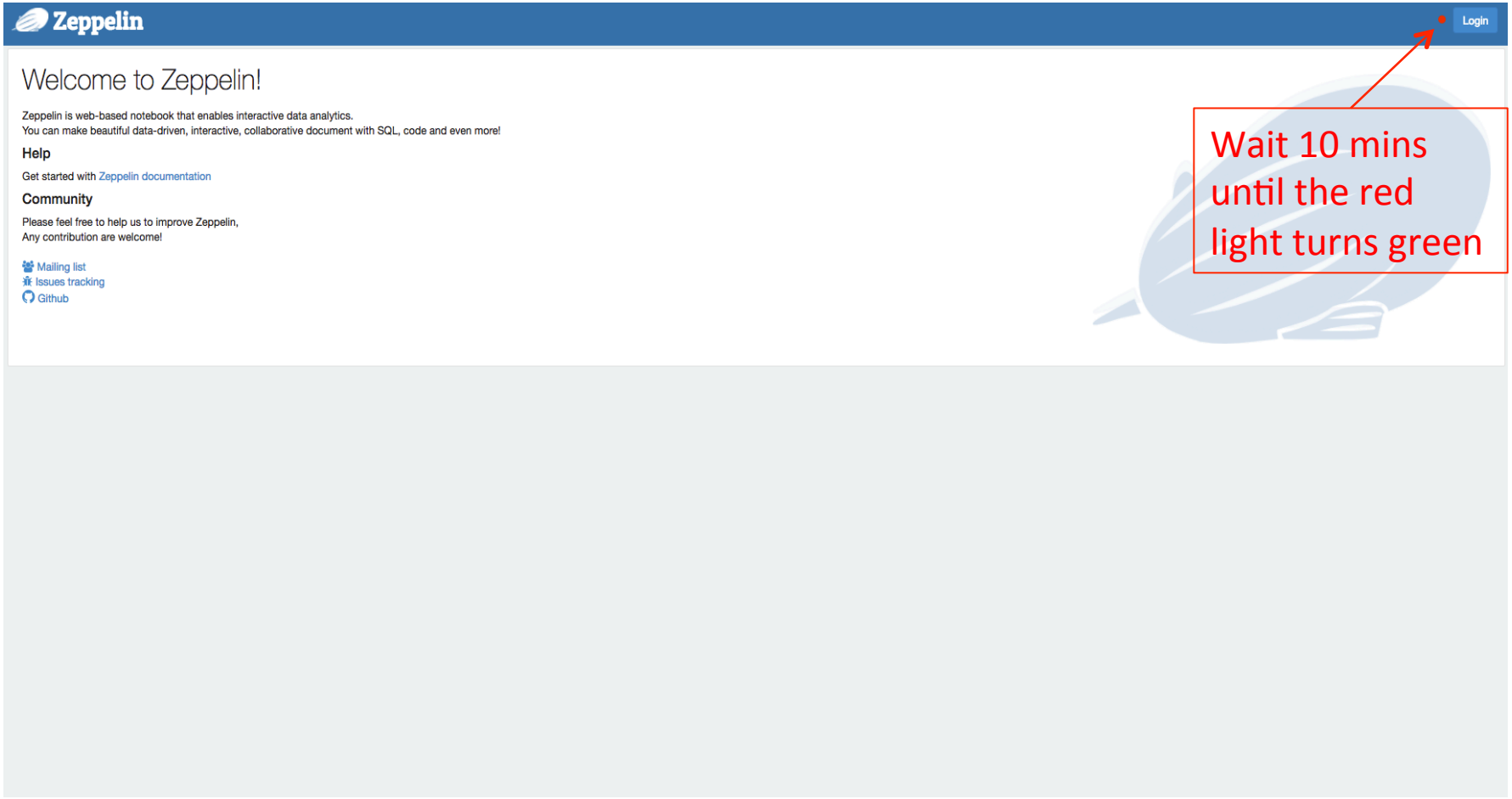
login1.wrangler(7)$sbatch –reservation=hadoop+TRAINING-HPC+2186 job.zeppelin

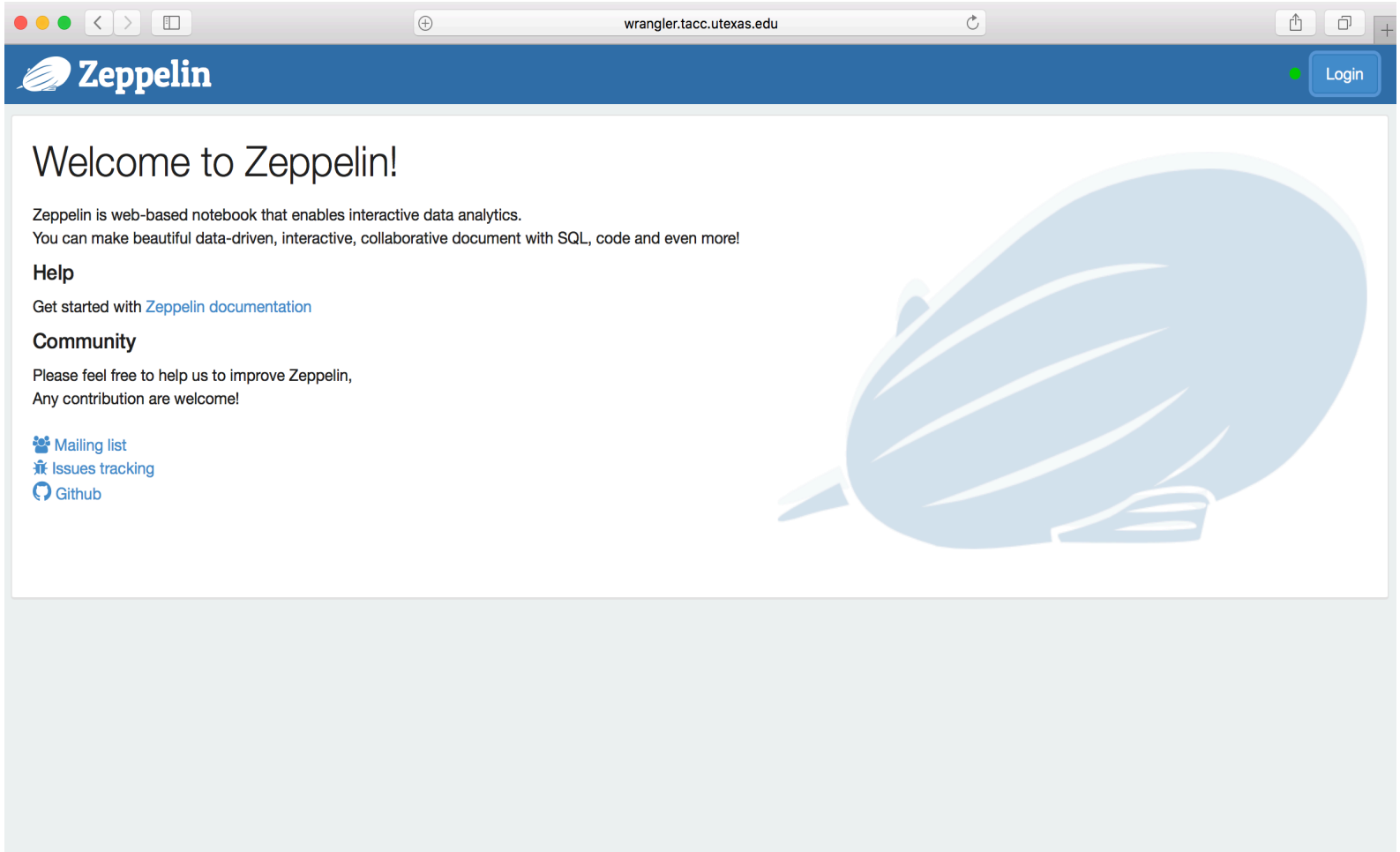
```
login1.wrangler(15)$ tail zeppelin.out |tail -n 3
Your applicatin  is now running!
Application UI is at http://wrangler.tacc.utexas.edu:15211
Zeppelin username and password: user9062
```

Wait 10 minutes for Zeppelin UI to start, copy and paste http://wrangler.tacc.utexas.edu:XXXXX to your web browser.

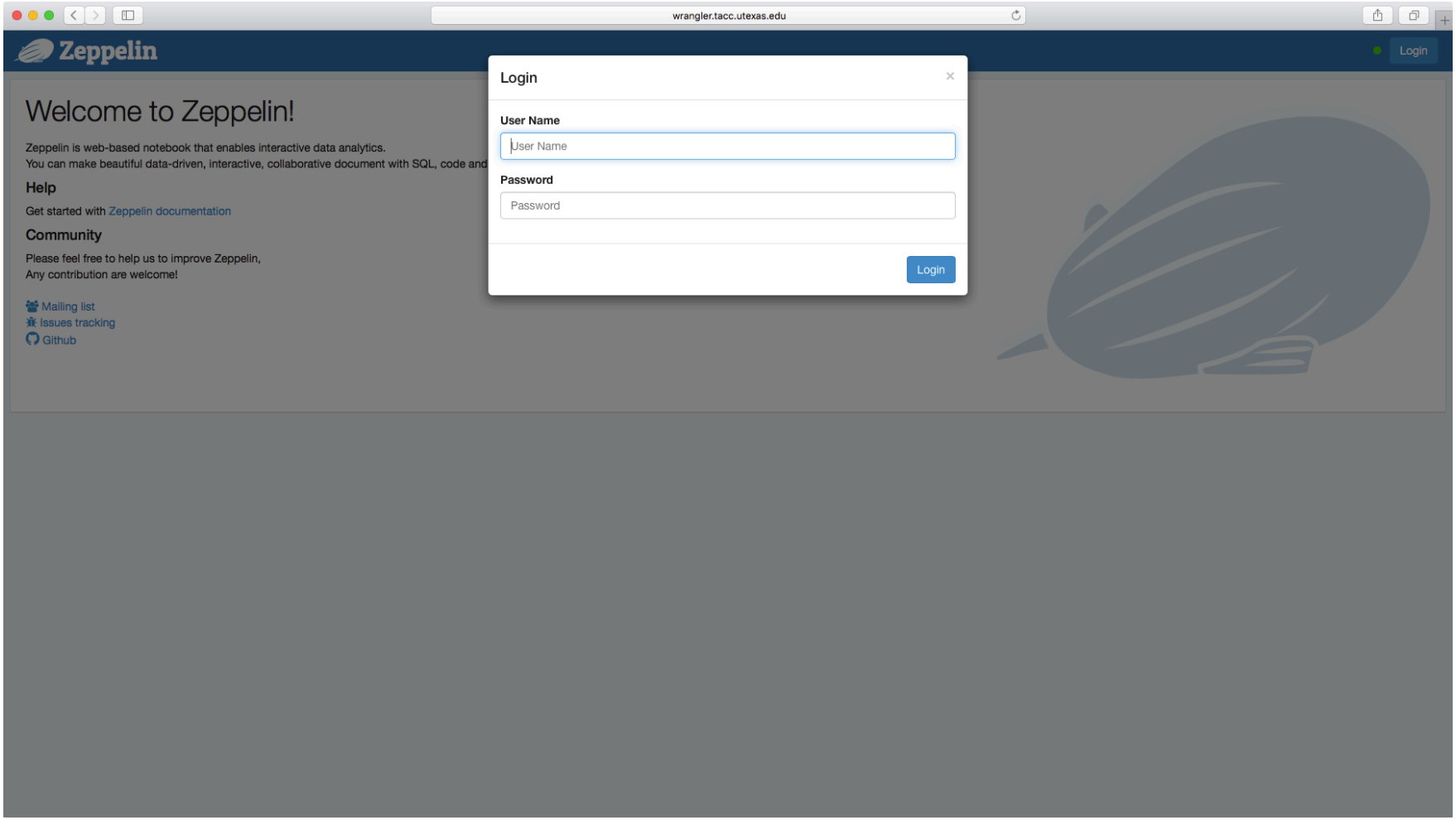
Use the username and password to login: userXXXX

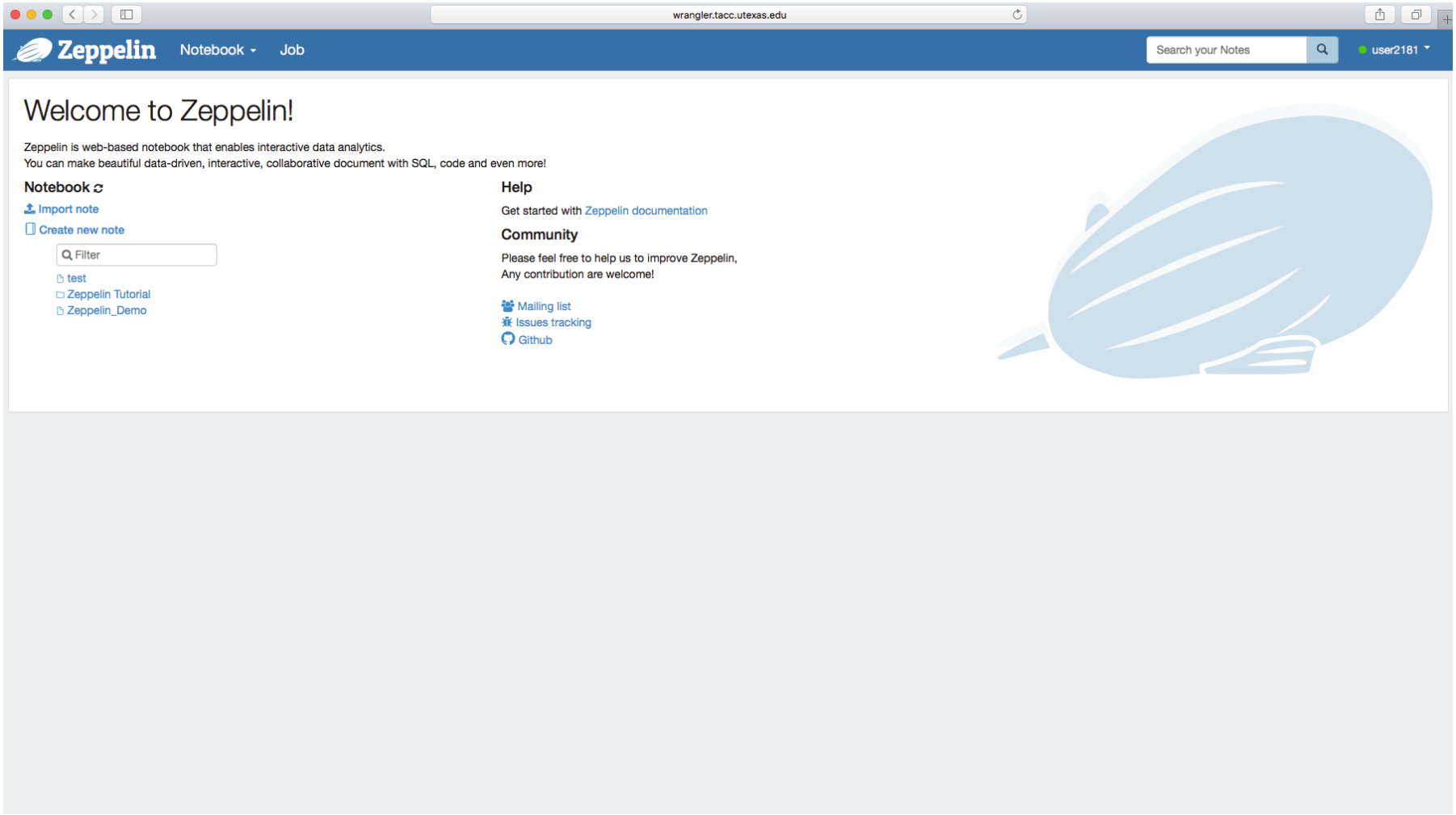

wrangler.tacc.utexas.edu
Safari Can’t Open the Page
Safari can’t open the page “wrangler.tacc.utexas.edu:15219” because the server unexpectedly dropped the connection. This sometimes occurs when the server is busy. Wait for a few minutes, and then try again.

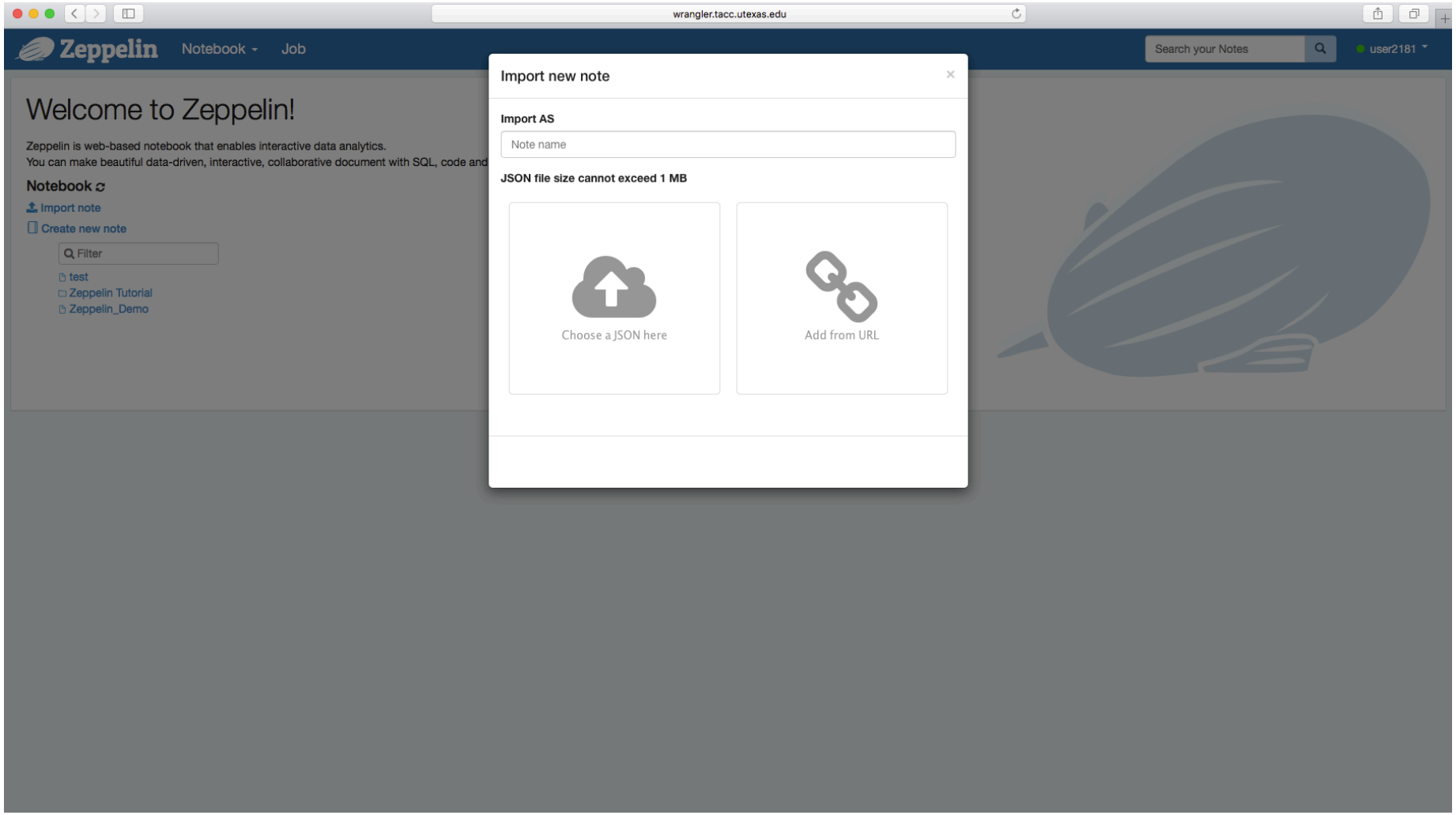

Zeppelin
Login
Welcome to Zeppelin!
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!
Help
Get started with Zeppelin documentation
Community
Please feel free to help us to improve Zeppelin,
Any contribution are welcome!
Mailing list
Issues tracking
Github
Wait 10 mins until the red light turns green

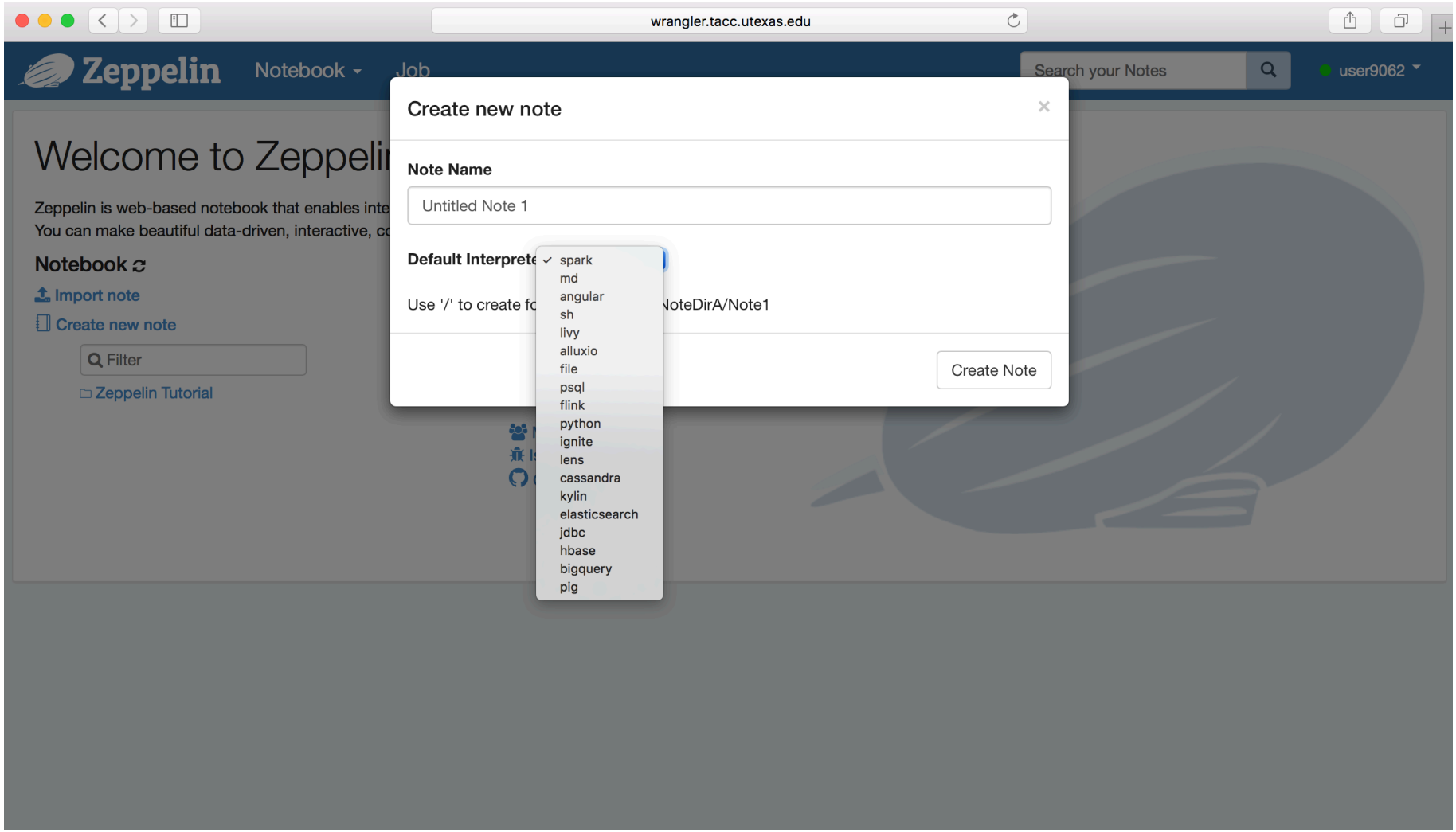

wrangler.tacc.utexas.edu
Zeppelin
Login
Welcome to Zeppelin!
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!
Help
Get started with Zeppelin documentation
Community
Please feel free to help us to improve Zeppelin,
Any contribution are welcome!
Mailing list
Issues tracking
Github

Type in your username and password: userxxxx




wrangler.tacc.utexas.edu
Zeppelin
Notebook
Job
Search your Notes
user2181
Welcome to Zeppelin!
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!
Notebook
Import note
Create new note
Filter
test
Zeppelin Tutorial
Zeppelin_Demo
Help
Get started with Zeppelin documentation
Community
Please feel free to help us to improve Zeppelin,
Any contribution are welcome!
Mailing list
Issues tracking
Github


wrangler.tacc.utexas.edu
Zeppelin
Notebook
Job
Search your Notes
user2181
Welcome to Zeppelin!
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and
Notebook
Import note
Create new note
Filter
test
Zeppelin Tutorial
Zeppelin_Demo
Import new note
Import AS
Note name
JSON file size cannot exceed 1 MB
Choose a JSON here
Add from URL

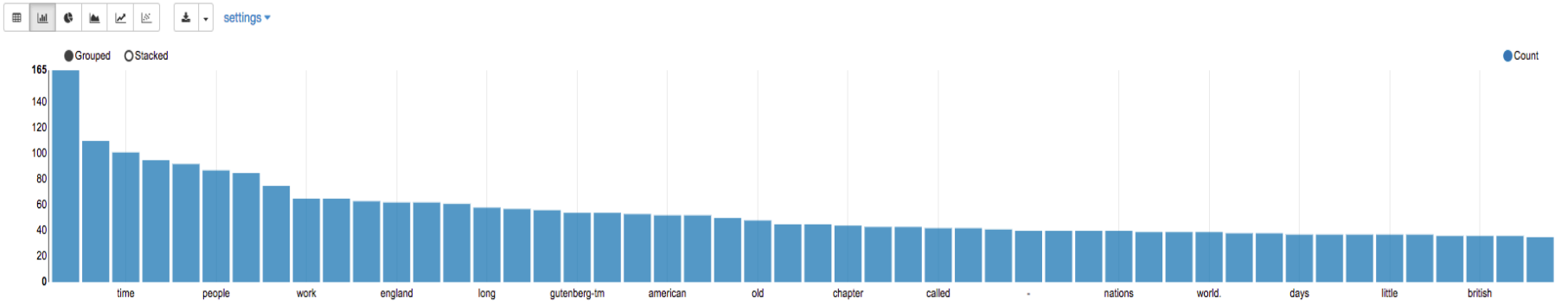

wrangler.tacc.utexas.edu
Zeppelin
Notebook
Job
Search your Notes
user9062
Welcome to Zeppelin
Zeppelin is web-based notebook that enables inte
You can make beautiful data-driven, interactive, co
Notebook
Import note
Create new note
Filter
Zeppelin Tutorial
Create new note
Note Name
Untitled Note 1
Default Interprete
spark
md
angular
sh
livy
alluxio
file
psql
flink
python
ignite
lens
cassandra
kylin
elasticsearch
jdbc
hbase
bigquery
pig
Use '/' to create fo
NoteDirA/Note1
Create Note

Using Spark with Scala

```
%spark
val stopWords = sc.textFile("stopwords.txt")
val stopWordSet = stopWords.collect.toSet
val stopWordSetBC = sc.broadcast(stopWordSet)

import org.apache.spark.sql.Row
//textFile.flatMap(_.toLowerCase.split(" ")).subtract(stopWords).take(100)
val wordCounts =  textFile.flatMap(_.toLowerCase.split(" ")).filter( w => !stopWordSetBC.value.contains(w)&&w!="").map(word => (word, 1)).reduceByKey((a, b) => a + b)

val top50 = wordCounts.sortBy(_._2,ascending=false).map{case (word:String,count:Int) => {word + "\t" + count} }.take(50)
//top50.mkString("\n")
print("%table Word\t Count\n" + top50.mkString("\n"))
```

FINISHED

```
stopWords: org.apache.spark.rdd.RDD[String] = stopwords.txt MapPartitionsRDD[3] at textFile at <console>:28
stopWordSet: scala.collection.immutable.Set[String] = Set(serious, latterly, down, side, moreover, please, ourselves, behind, for, find, further, mill, due, any, wherein, across, twenty, name, this, in, move, itse", have, your, off, once, are, is, his, why, too, among, everyone, show,
 empty, already, nobody, less, am, hence, system, than, four, fire, anyhow, three, whereby, con, twelve, throughout, but, whether, below, co, mine, becomes, eleven, what, would, although, elsewhere, another, front, if, hereby, own, neither, bottom, up, etc, so, our, per, therein, must,
 beforehand, keep, do, all, him, had, somehow, re, onto, nor, every, herein, full, before, afterwards, somewhere, whither, else, namely, us, it, whereupon, two, thence, a, herse", sometimes, became, though, within, as, because...
stopWordSetBC: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Set[String]] = Broadcast(4)
import org.apache.spark.sql.Row
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[7] at reduceByKey at <console>:37
top50: Array[String] = Array(great	165, men	110, time	101, years	95, new 92, people	87, project	85, thousand	75, work	65, god 65, history	63, england	62, ancient	62, religious	61, long	58, city	57, human	56, gu
tenberg-tm	54, man 54, english	53, american	52, world	52, king	50, old 48, came	45, europe	45, chapter	44, took	43, did 43, called	42, roman	42, length	41, -	40, day 40, life	40, nations	40, gave	39, ne
arly	39, world.	39, a.	38, d.	38, days	37, received	37, hand	37, little	37, vast	37, place	36, british	36, [image:	36, modern	35)
```



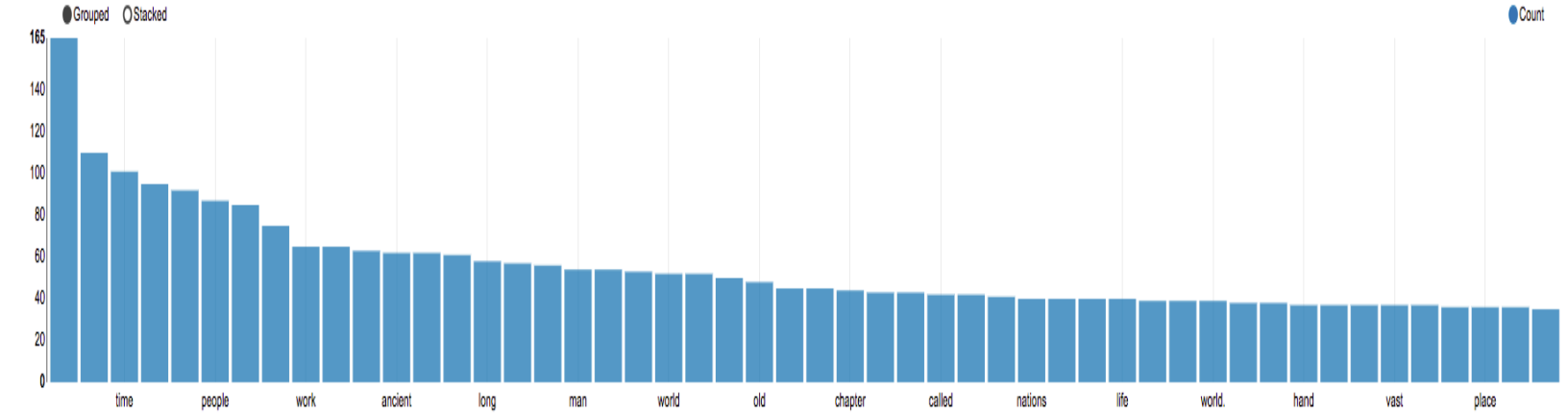
Took 1 min 57 sec. Last updated by anonymous at April 17 2017, 1:18:18 PM. (outdated)

Using Spark with Python

```
%pyspark
textFile = sc.textFile("book.txt")

stopWords = sc.textFile("stopwords.txt")


wordCounts = textFile.flatMap(lambda line: line.lower().split()).subtract(stopWords).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b)
top50 = wordCounts.sortBy(lambda a: a[1],ascending=False).map(lambda x: x[0]+ "\t" + str(x[1])).take(50)

print("%table Word\t Count\n" + "\n".join(top50))
```

READY

settings



Using Spark with R

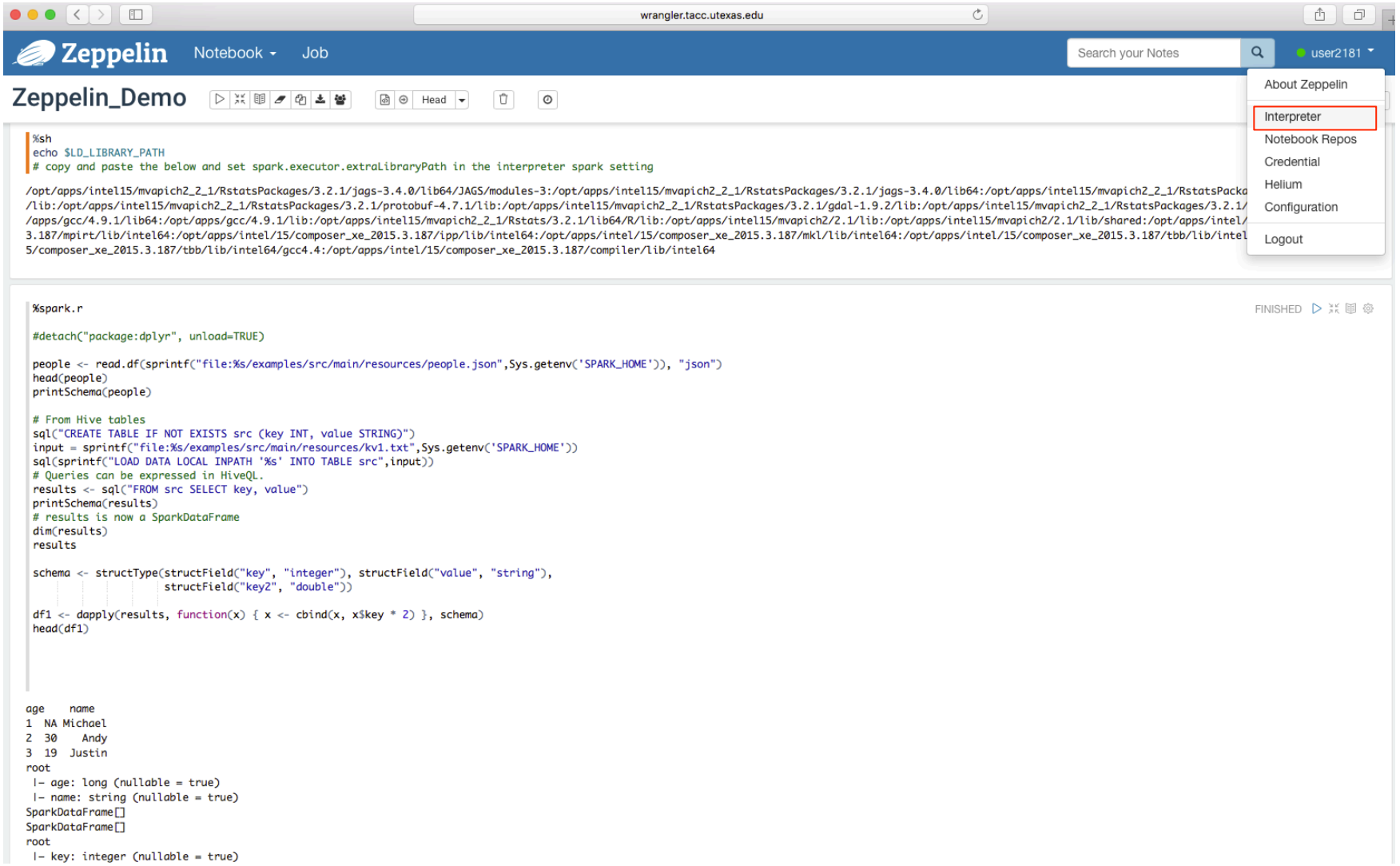
SparkR interpreter setting(spark.executor.extraLibraryPath):

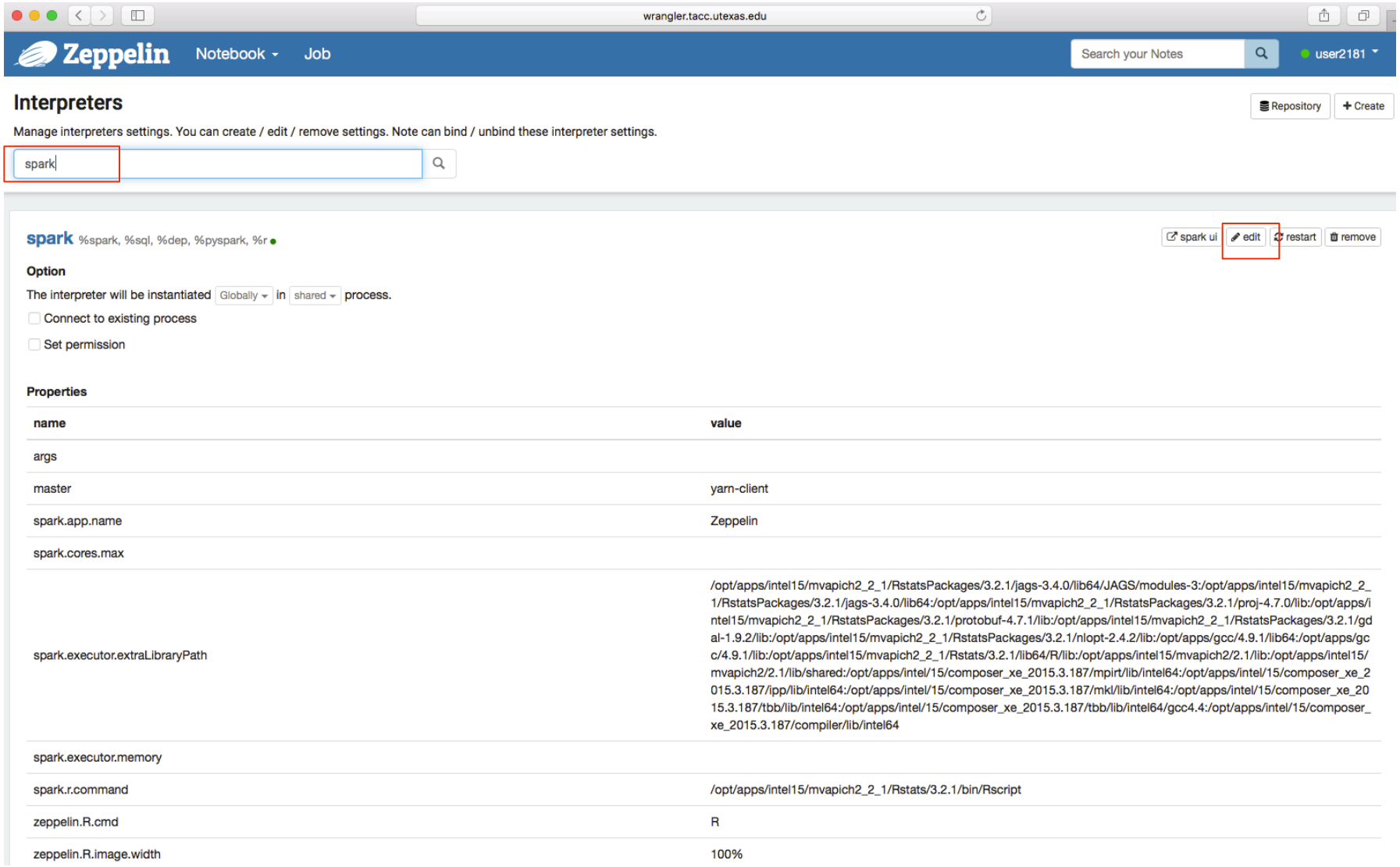
spark.executor.extraLibraryPath	/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/jags-3.4.0/lib64/JAGS/modules-3:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/jags-3.4.0/lib64:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/proj-4.7.0/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/protobuf-4.7.1/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/gdal-1.9.2/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/nlopt-2.4.2/lib:/opt/apps/gcc/4.9.1/lib64:/opt/apps/gcc/4.9.1/lib:/opt/apps/intel15/mvapich2_2_1/Rstats/3.2.1/lib64/R/lib:/opt/apps/intel15/mvapich2/2.1/lib:/opt/apps/intel15/mvapich2/2.1/lib/shared:/opt/apps/intel/15/composer_xe_2015.3.187/mpirt/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/ipp/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/mkl/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/tbb/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/tbb/lib/intel64/gcc4.4:/opt/apps/intel/15/composer_xe_2015.3.187/compiler/lib/intel64
spark.executor.memory	
spark.r.command	/opt/apps/intel15/mvapich2_2_1/Rstats/3.2.1/bin/Rscript

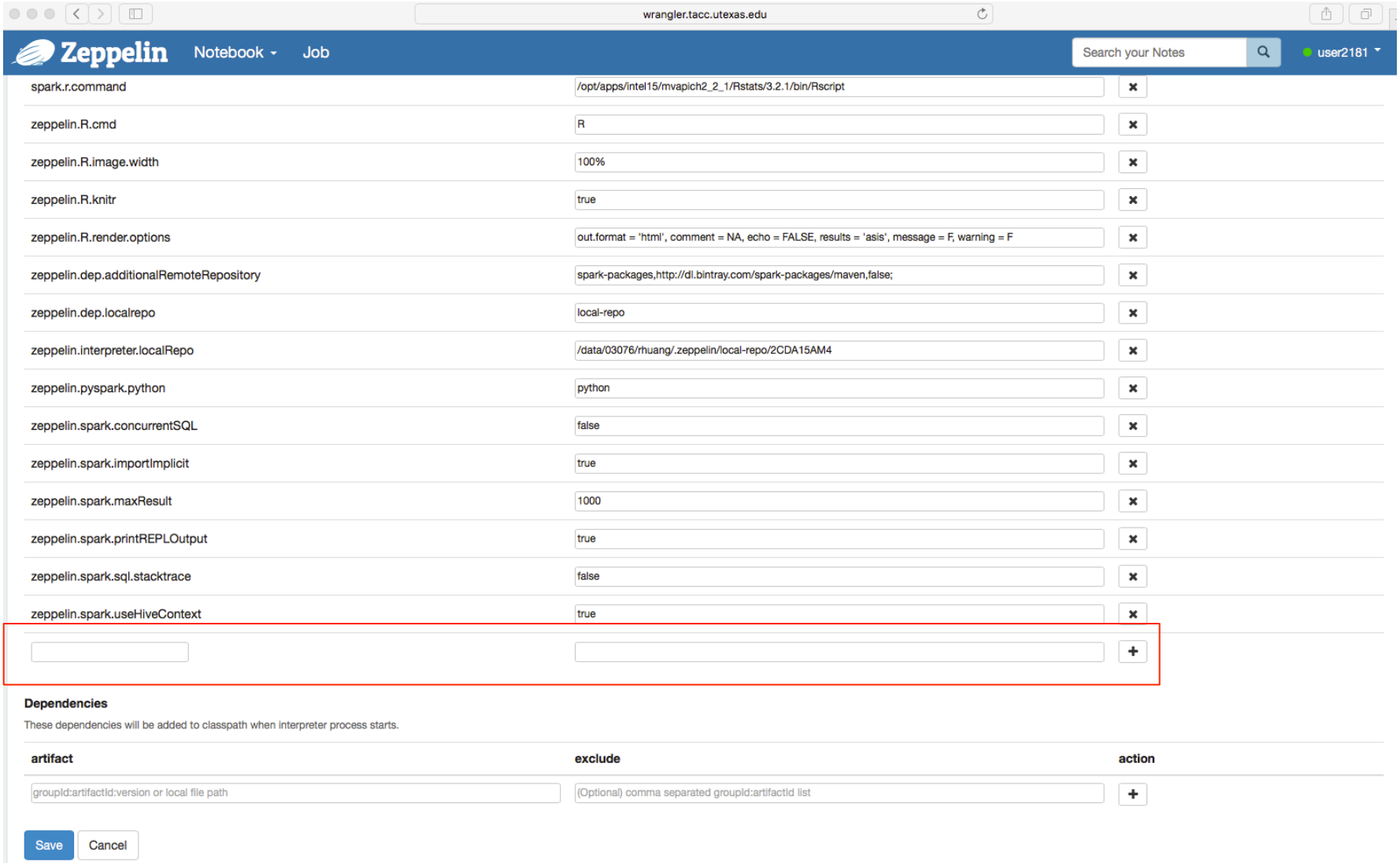
FINISHED

```
%spark.r
##### Run a given function on a large dataset using dapply or dapplyCollect
# Convert waiting time from hours to seconds.
# Note that we can apply UDF to DataFrame.
schema <- structType(structField("eruptions", "double"), structField("waiting", "double"),
                     structField("waiting_secs", "double"))
df1 <- dapply(df, function(x) { x <- cbind(x, x$waiting * 60) }, schema)
head(collect(df1))
```

```
eruptions waiting waiting_secs
1     3.600      79         4740
2     1.800      54         3240
3     3.333      74         4440
4     2.283      62         3720
5     4.533      85         5100
6     2.883      55         3300
```


wrangler.tacc.utexas.edu
Zeppelin
Notebook
Job
Search your Notes
user2181
About Zeppelin
Interpreter
Notebook Repos
Credential
Helium
Configuration
Logout
Zeppelin_Demo
Head
%sh
echo $LD_LIBRARY_PATH
copy and paste the below and set spark.executor.extraLibraryPath in the interpreter spark setting
/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/jags-3.4.0/lib64/JAGS/modules-3:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/jags-3.4.0/lib64:/opt/apps/intel15/mvapich2_2_1/RstatsPacka
/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/protobuf-4.7.1/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/gdal-1.9.2/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/
/apps/gcc/4.9.1/lib64:/opt/apps/gcc/4.9.1/lib:/opt/apps/intel15/mvapich2_2_1/Rstats/3.2.1/lib64/R/lib:/opt/apps/intel15/mvapich2/2.1/lib:/opt/apps/intel15/mvapich2/2.1/lib/shared:/opt/apps/intel/
3.187/mpirt/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/ipp/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/mkl/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/tbb/lib/intel
5/composer_xe_2015.3.187/tbb/lib/intel64/gcc4.4:/opt/apps/intel/15/composer_xe_2015.3.187/compiler/lib/intel64
%spark.r
FINISHED
#detach("package:dplyr", unload=TRUE)
people <- read.df(sprintf("file:%s/examples/src/main/resources/people.json",Sys.getenv('SPARK_HOME')), "json")
head(people)
printSchema(people)
# From Hive tables
sql("CREATE TABLE IF NOT EXISTS src (key INT, value STRING)")
input = sprintf("file:%s/examples/src/main/resources/kv1.txt",Sys.getenv('SPARK_HOME'))
sql(sprintf("LOAD DATA LOCAL INPATH '%s' INTO TABLE src",input))
# Queries can be expressed in HiveQL.
results <- sql("FROM src SELECT key, value")
printSchema(results)
# results is now a SparkDataFrame
dim(results)
results
schema <- structType(structField("key", "integer"), structField("value", "string"),
structField("key2", "double"))
df1 <- dapply(results, function(x) { x <- cbind(x, x$key * 2) }, schema)
head(df1)
age name
1 NA Michael
2 30 Andy
3 19 Justin
root
|- age: long (nullable = true)
|- name: string (nullable = true)
SparkDataFrame[]
SparkDataFrame[]
root
|- key: integer (nullable = true)


wrangler.tacc.utexas.edu
Zeppelin
Notebook
Job
Search your Notes
user2181
Interpreters
Repository
+ Create
Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.
spark
spark %spark, %sql, %dep, %pyspark, %r
spark ui
edit
restart
remove
Option
The interpreter will be instantiated Globally in shared process.
Connect to existing process
Set permission
Properties
name
value
args
master
yarn-client
spark.app.name
Zeppelin
spark.cores.max
spark.executor.extraLibraryPath
/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/jags-3.4.0/lib64/JAGS/modules-3:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/jags-3.4.0/lib64:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/proj-4.7.0/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/protobuf-4.7.1/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/gdal-1.9.2/lib:/opt/apps/intel15/mvapich2_2_1/RstatsPackages/3.2.1/nlopt-2.4.2/lib:/opt/apps/gcc/4.9.1/lib64:/opt/apps/gcc/4.9.1/lib:/opt/apps/intel15/mvapich2_2_1/Rstats/3.2.1/lib64/R/lib:/opt/apps/intel15/mvapich2/2.1/lib:/opt/apps/intel15/mvapich2/2.1/lib/shared:/opt/apps/intel/15/composer_xe_2015.3.187/mpirt/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/ipp/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/mkl/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/tbb/lib/intel64:/opt/apps/intel/15/composer_xe_2015.3.187/tbb/lib/intel64/gcc4.4:/opt/apps/intel/15/composer_xe_2015.3.187/compiler/lib/intel64
spark.executor.memory
spark.r.command
/opt/apps/intel15/mvapich2_2_1/Rstats/3.2.1/bin/Rscript
zeppelin.R.cmd
R
zeppelin.R.image.width
100%


wrangler.tacc.utexas.edu
Zeppelin
Notebook
Job
Search your Notes
user2181
spark.r.command
/opt/apps/intel15/mvapich2_2_1/Rstats/3.2.1/bin/Rscript
zeppelin.R.cmd
R
zeppelin.R.image.width
100%
zeppelin.R.knitr
true
zeppelin.R.render.options
out.format = 'html', comment = NA, echo = FALSE, results = 'asis', message = F, warning = F
zeppelin.dep.additionalRemoteRepository
spark-packages,http://dl.bintray.com/spark-packages/maven,false;
zeppelin.dep.localrepo
local-repo
zeppelin.interpreter.localRepo
/data/03076/rhuang/.zeppelin/local-repo/2CDA15AM4
zeppelin.pyspark.python
python
zeppelin.spark.concurrentSQL
false
zeppelin.spark.importImplicit
true
zeppelin.spark.maxResult
1000
zeppelin.spark.printREPLOutput
true
zeppelin.spark.sql.stacktrace
false
zeppelin.spark.useHiveContext
true
Dependencies
These dependencies will be added to classpath when interpreter process starts.
artifact
exclude
action
groupId:artifactId:version or local file path
(Optional) comma separated groupId:artifactId list
Save
Cancel

Demo - Zeppelin

Note:

- Clear browser data (cached images and files) to avoid loading the cached image of Zeppelin UI.

- To change default setting in the Zeppelin's interpreter:
  - spark.executor.instances: 12
  - spark.executor.memory: 2g
  - spark.driver.memory: 4g

Recap

- Hadoop vs. Spark
  - Spark: in-memory, faster, interactive development, multi-languages (Scala, Java, Python, R)
- Using Hadoop with other languages
  - Also called Hadoop streaming – confusing naming
  - Hard to debug
- Zeppelin
  - Interactive multi-interpreters notebook with great visualization support
- Using Spark with Scala, Python and R
  - SparkR api quite different from Scala and Python

Try It Out

Exercise 3 shows an example of compile from user's java code to Hadoop application and run it.

Exercise 4 is about hadoop streaming

- Can you write a word count example with your favorite programming language?
- Exemplar code for Python, R and bash are available at the corresponding directory.

Questions to think?

- Where/how did we specify parallelisms?

Exercise 5 is an example of a completed workflow for using Mahout for document clustering.

More Questions

Data Group @ TACC
data@tacc.utexas.edu

Weijia Xu
xwj@tacc.utexas.edu
Ruizhu Huang
rhuang@tacc.utexas.edu

Thank You!

Misc.

The reservation name is hadoop+TRAINING-HPC+2186

Exemplar data, code, and practice command suggestions are also in ***/work/00791/xwj/*DMS/hadoop-training**

idev session command

idev –r hadoop+TRAINING-HPC+2186 –m 240 –p hadoop

Vis portal vis.tacc.utexas.edu

Refer to exercise text or https://goo.gl/pUXO7g