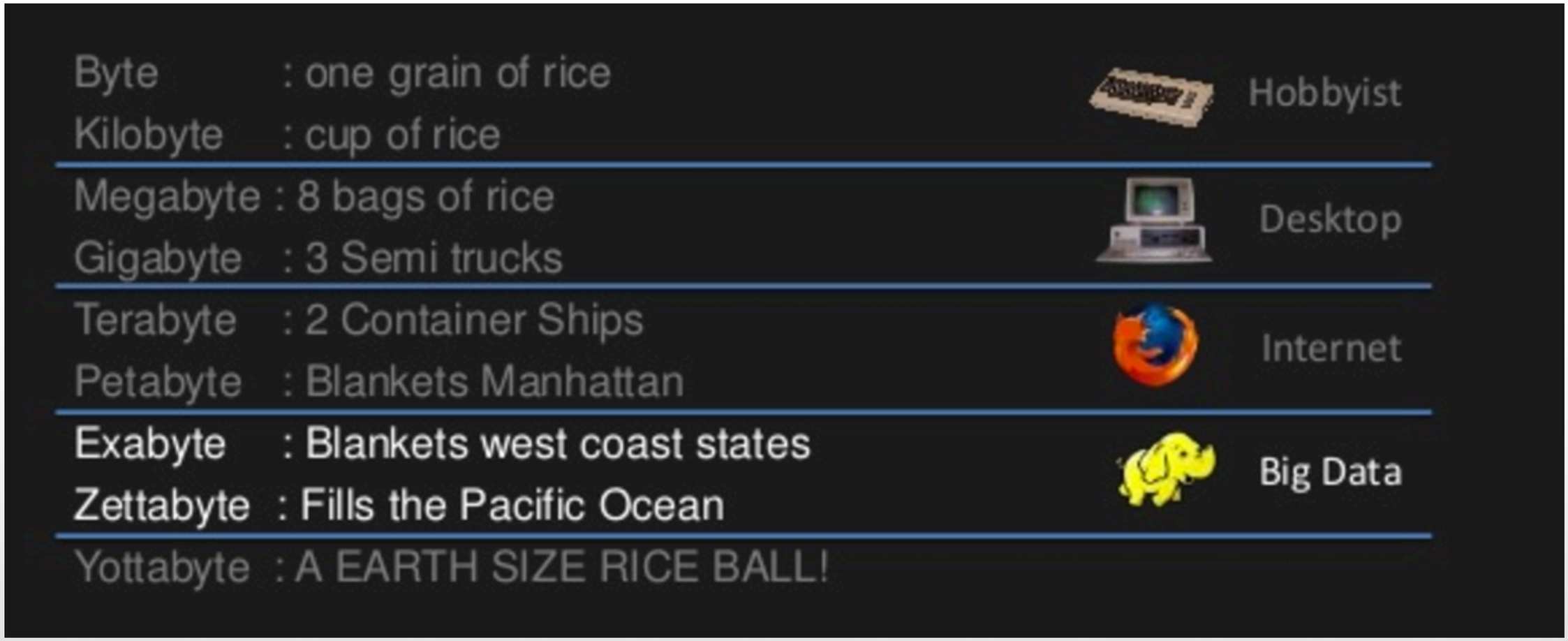
Big Data Analytics with Spark and Hadoop at OSC

10/26/2017
OSC workshop

Shameema Oottikkal
Data Application Engineer
Ohio SuperComputer Center
email:soottikkal@osc.edu

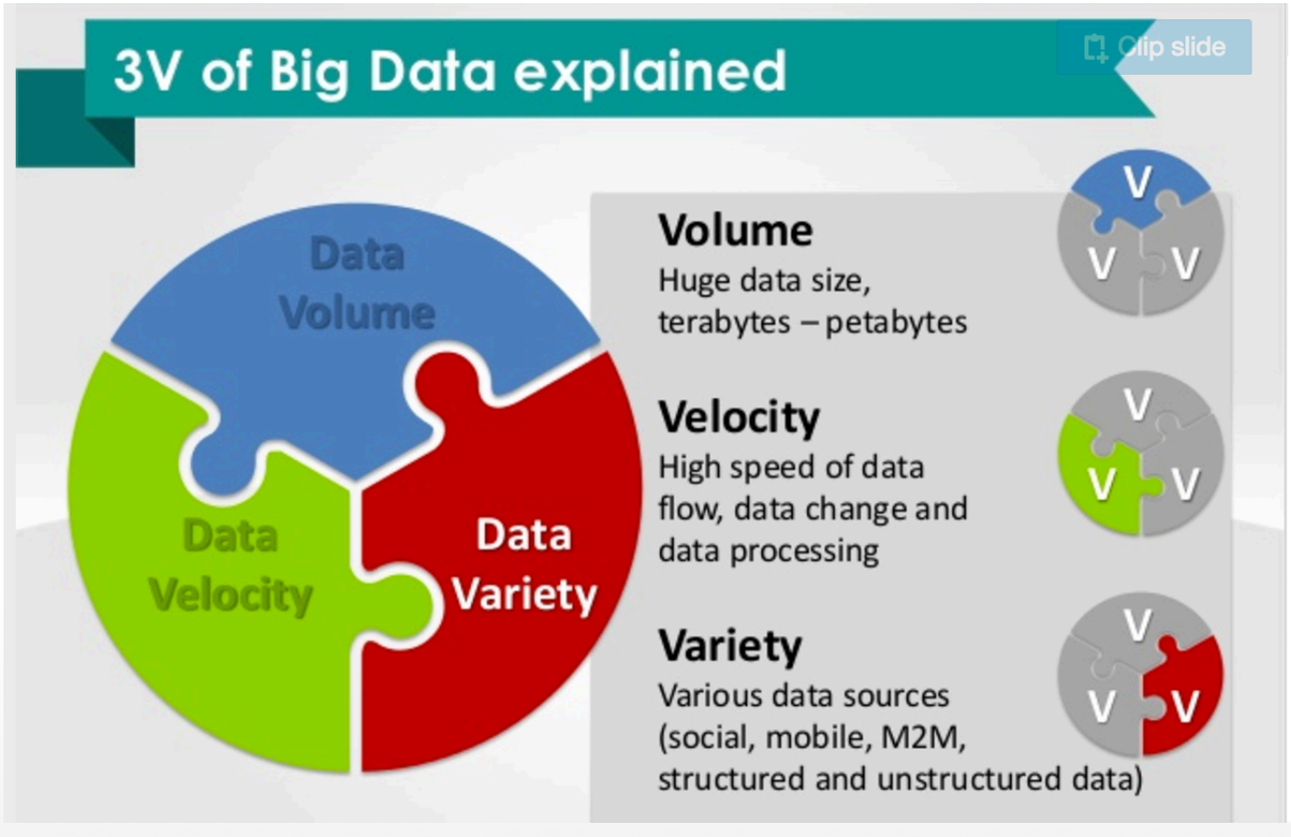
What is Big Data

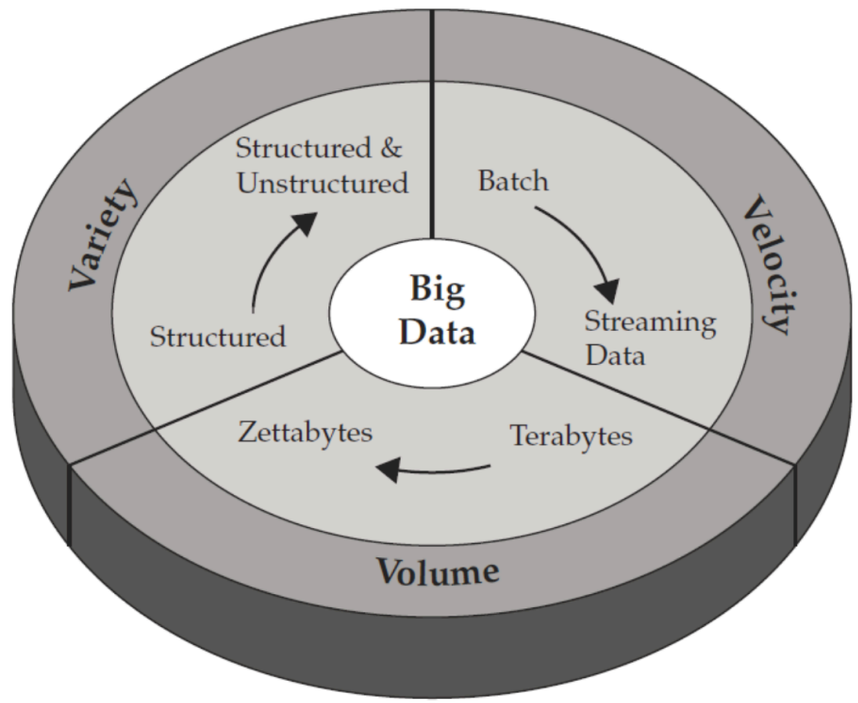
Big data is an evolving term that describes any voluminous amount of structured and unstructured data that has the potential to be mined for information.



Ref: http://www.slideshare.net/dwellman/what-is-big-data-24401517/3

The 3V of Big Data





- Key enablers for the growth of “Big Data” are:
  - Increase of storage capacities
  - Increase of processing power
  - Availability of data

Big Data Applications

AUTOMOTIVE
Auto sensors reporting location, problems

COMMUNICATIONS
Location-based advertising

CONSUMER PACKAGED GOODS
Sentiment analysis of what's hot, problems

FINANCIAL SERVICES
Risk & portfolio analysis
New products

EDUCATION & RESEARCH
Experiment sensor analysis

HIGH TECHNOLOGY / INDUSTRIAL MFG.
Mfg quality
Warranty analysis

LIFE SCIENCES
Clinical trials
Genomics

MEDIA/ ENTERTAINMENT
Viewers / advertising effectiveness

ON-LINE SERVICES / SOCIAL MEDIA
People & career matching
Web-site optimization

HEALTH CARE
Patient sensors, monitoring, EHRs
Quality of care

OIL & GAS
Drilling exploration sensor analysis

RETAIL
Consumer sentime
Optimized market

TRAVEL & TRANSPORTATION
Sensor analysis for optimal traffic flows
Customer sentiment

UTILITIES
Smart Meter analysis for network capacity,

LAW ENFORCEMENT & DEFENSE
Threat analysis - social media monitoring, photo analysis

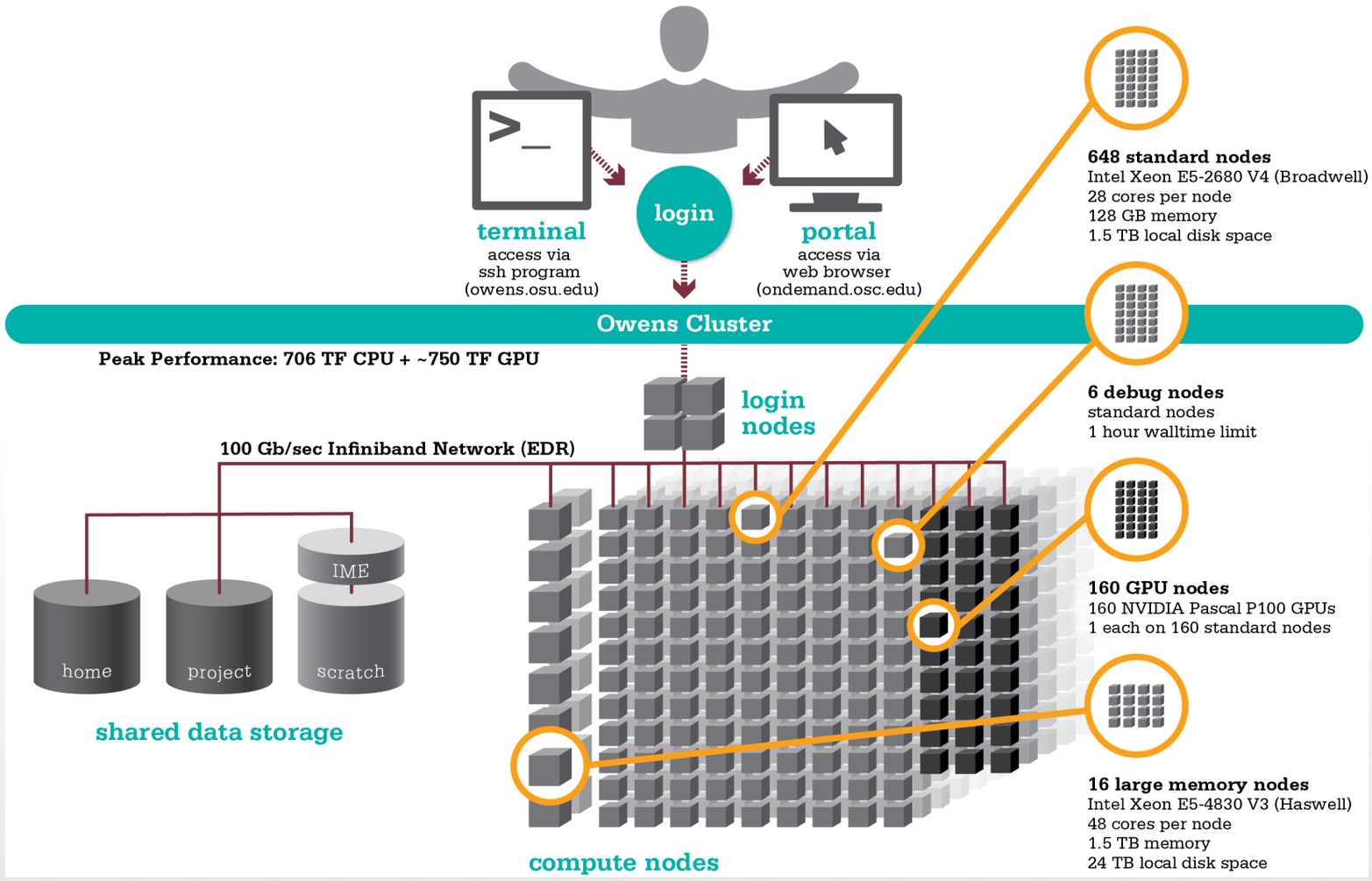
BRILLIX
delivering brilliant DBAs

Data Analytical Tools

	Examples	Characteristics	Typical tools	Analytical methods
Small Data (megabytes)	Sales records, Customers database (small and medium companies)	Hundreds – thousands of records	Personal computer, Excel, R, other basic statistics software	Simple statistics
Large Data (gigabytes-terabytes)	Customer databases (big companies)	Millions of records, mostly structured data	Server workstation computer, Relational database systems, data warehouses	Advanced statistics, business intelligence, data mining,
Big Data (terabytes – petabytes)	Customer interactions (social media, mobile), multimedia (video, images, free text), location-based data, RFIM	Over millions of records, distributed, unstructured	Cloud, data centers, Distributed databases, NoSQL, Hadoop	MapReduce, Distributed File Systems

Data Analytical nodes@OSC

Owens' data analytics environment is comprised of 16 nodes, each with 48 CPU cores, 1.5TB of RAM and 24TB of local disk.



$HOME:
500GB/per user
Backed up daily
Permanent storage

Local disk:$TMPDIR
1.5TB or 24TB
Not backed up
Temporary storage

/fs/scratch:
1200TB
Not backed up
Temporary storage

/fs/project:
Upon request
1-5TB
Backed up daily
1-3 years

OSC OnDemand ondemand.osc.edu

- 1: User Interface
  - Web based
    - Usable from computers, tablets, smartphones
    - Zero installation
  - Single point of entry
    - User needs three things
      - ondemand.osc.edu
      - OSC Username
      - OSC Password
    - Connected to all resources at OSC

- 2: Interactive Services
  - File Access
  - Job Management
  - Visualization Apps
    - Desktop access
    - Single-click apps (Abaqus, Ansys, Comsol, Paraview)
  - Terminal Access

Tutorial available at

osc.edu/ondemand

OSC OnDemand

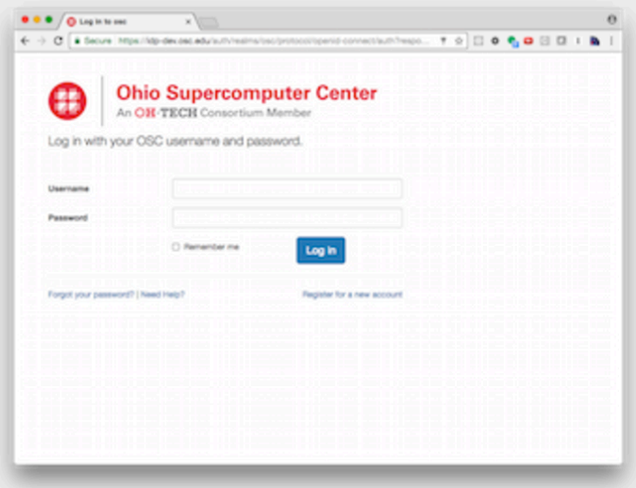
Login to OSC OnDemand

Log in with either your OSC Account or a third party account via CILogon.

Log in with your OSC account

Step 1. Login with your OSC account

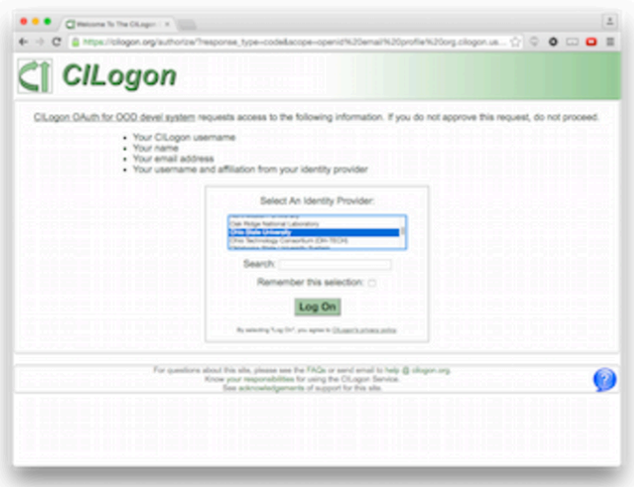
Authenticate with OSC's Open ID Connect server.

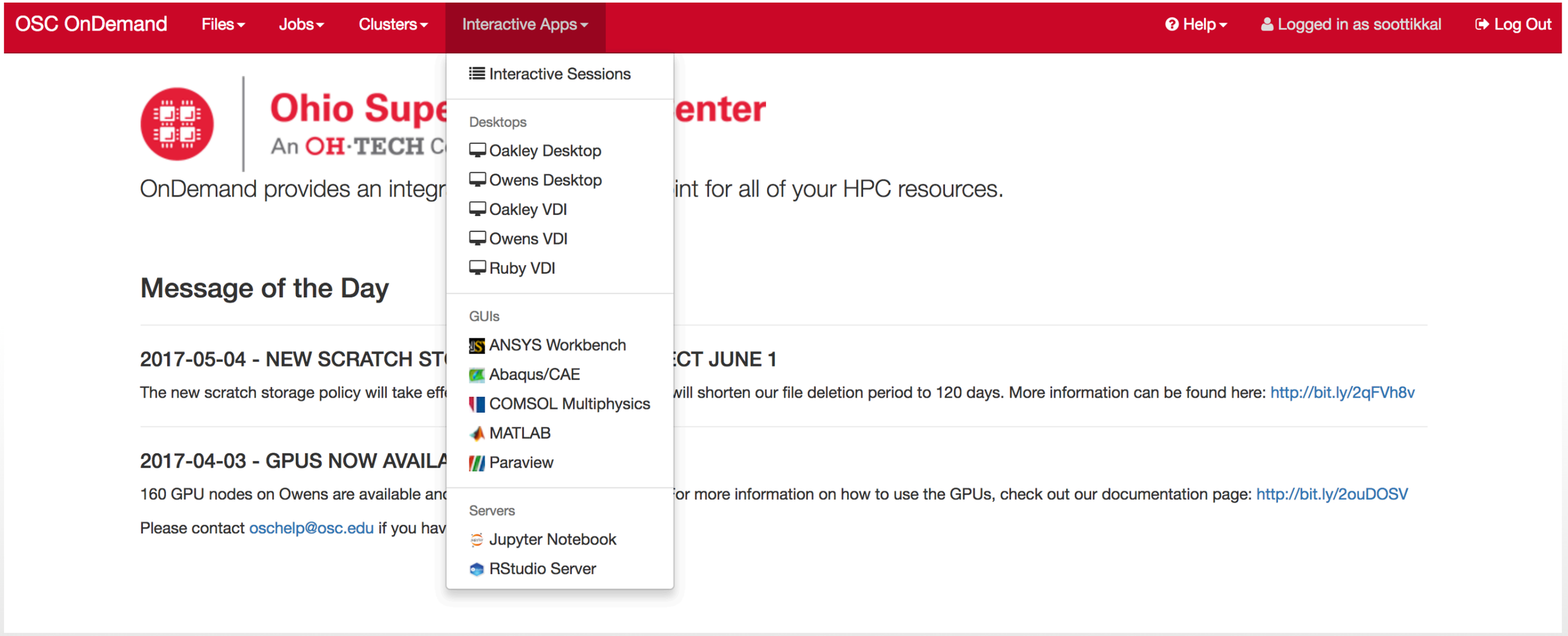


Log in with third party through CILogon

Step 1. Choose your identity provider

CILogon provides access to identity providers from many academic institutions across the state.




OSC OnDemand
Files
Jobs
Clusters
Interactive Apps
Help
Logged in as soottikkal
Log Out
Interactive Sessions
Desktops
Oakley Desktop
Owens Desktop
Oakley VDI
Owens VDI
Ruby VDI
GUIs
ANSYS Workbench
Abaqus/CAE
COMSOL Multiphysics
MATLAB
Paraview
Servers
Jupyter Notebook
RStudio Server
Ohio Supe
enter
An OH·TECH C
OnDemand provides an integr
int for all of your HPC resources.
Message of the Day
2017-05-04 - NEW SCRATCH ST
CT JUNE 1
The new scratch storage policy will take eff
will shorten our file deletion period to 120 days. More information can be found here: http://bit.ly/2qFVh8v
2017-04-03 - GPUS NOW AVAILA
160 GPU nodes on Owens are available and
or more information on how to use the GPUs, check out our documentation page: http://bit.ly/2ouDOSV
Please contact oschelp@osc.edu if you hav

Data Analytics@OSC

Python: A popular general-purpose, high-level programming language with numerous mathematical and scientific packages available for data analytics.

R: A programming language for statistical and machine learning applications with very strong graphical capabilities.

MATLAB:A full featured data analysis toolkit with many advanced algorithms readily available.

Spark and Hadoop: Frameworks for running map reduce algorithms

Intel Compilers: Compilers for generating optimized code for Intel CPUs.

Intel MKL: The Math Kernel Library provides optimized subroutines for common computation tasks such as matrix-matrix calculations.

Statistical software: Octave, Stata, FFTW, ScaLAPACK, MINPACK, sprng2

R and Rstudio

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical and graphical techniques and is highly extensible.

Availability:

The following versions of R are available on OSC systems:

VERSION	OAKLEY	OWENS
2.14.1	X	
2.15.0	X	
2.15.2	X	
3.0.1	X	
3.1.3	X	
3.2.0	X	
3.3.1	X*	X
3.3.2		X*

Running R interactively

Set-up

In order to configure your environment for the usage of R, run the following command:

```
module load R
```

Using R

Once your environment is configured, R can be started simply by entering the following command:

```
R
```

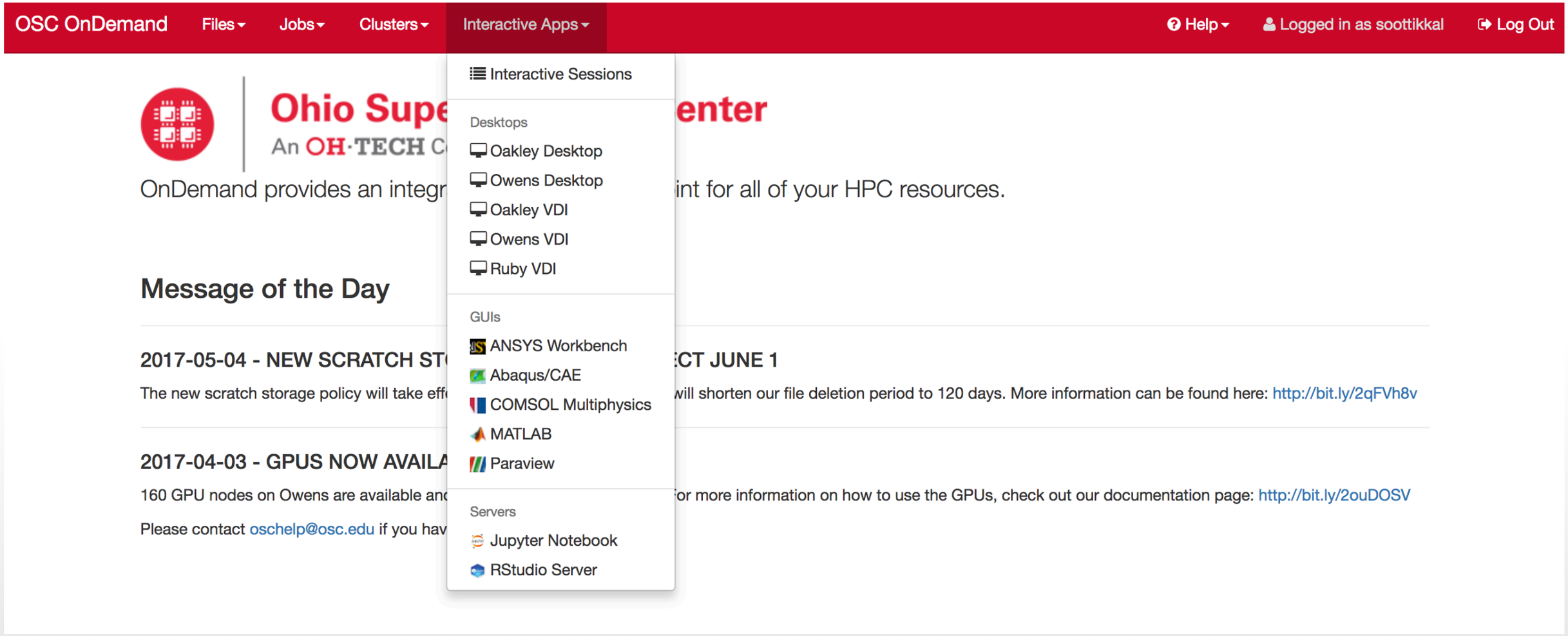
For a listing of command line options, run:

```
R --help
```

Batch Usage

```
#PBS -N R_ExampleJob
#PBS -l nodes=1:ppn=12

module load R
cd $PBS_O_WORKDIR
cp in.dat $TMPDIR
cd $TMPDIR

R CMD BATCH test.R test.Rout

cp test.Rout $PBS_O_WORKDIR
```

Rstudio on Ondemand



Home / Interactive Sessions / RStudio Server

Interactive Apps

Desktops

- Oakley Desktop
- Owens Desktop
- Oakley VDI
- Owens VDI
- Ruby VDI

GUIs

- ANSYS Workbench
- Abaqus/CAE
- COMSOL Multiphysics
- MATLAB
- Paraview

Servers

- Jupyter Notebook

This app will launch an RStudio Server on one or more Owens nodes.

Number of hours

1

Number of nodes

1

Node type

any

- **any** - (*28 cores*) Chooses anyone of the available Owens nodes. This reduces the wait time as you have no requirements.
- **hugemem** - (*48 cores*) This Owens node has 1.5TB of available RAM as well as 48 cores. There are 16 of these nodes on Owens.

Account

PZS0680

You can leave this blank if **not** in multiple projects.

I would like to receive an email when the session starts

Launch

Session was successfully created. ×

Home / Interactive Sessions

Interactive Apps

Desktops

- Oakley Desktop
- Owens Desktop
- Oakley VDI
- Owens VDI
- Ruby VDI

GUIs

RStudio Server (1891978.owens-batch.ten.osc.edu) Queued

Created at: 2017-09-26 11:36:18 EDT

Time Requested: 1 hour

Session ID: 8622e17d-1728-4aeb-b929-48a0012b16c6

Delete

Please be patient as your job currently sits in queue. The wait time depends on the number of cores as well as time requested.

Interactive Apps

Desktops

- Oakley Desktop
- Owens Desktop
- Oakley VDI
- Owens VDI
- Ruby VDI

GUIs

- ANSYS Workbench
- Abaqus/CAE

RStudio Server (1891978.owens-batch.ten.osc.edu) 1 node | 28 cores | Running

Host: o0143.ten.osc.edu

Created at: 2017-09-26 11:36:18 EDT

Time Remaining: about 1 hour

Session ID: 8622e17d-1728-4aeb-b929-48a0012b16c6

Delete

If you see `Failed to connect to ...`, then wait a few seconds before trying the **Connect to Jupyter** button again. This warning appeared because the Jupyter Notebook is still starting up.

Connect to RStudio Server

File Edit Code View Plots Session Build Debug Profile Tools Help

soottikkal

Go to file/function | Addins | Project: (None)

Console ~/

```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> 
```

Environment | **History**

Import Dataset | List

Global Environment

Environment is empty

Files | **Plots** | **Packages** | **Help** | **Viewer**

New Folder | Upload | Delete | Rename | More

Home

Name	Size	Modified
.RData	2.5 KB	Sep 5, 2017, 12:28 PM
.Renviron	90 B	Jul 6, 2017, 11:42 AM
.Rhistory	4.6 KB	Sep 26, 2017, 11:38 AM
1	1.3 KB	Jul 13, 2016, 12:28 PM
4EY4-NCH_amd_ime.out	3.2 KB	Aug 29, 2017, 12:16 PM
4EY4-NCH_md01.rst	3.6 MB	Aug 29, 2017, 12:16 PM
4EY4-NCH_solvated.prmtop	9.7 MB	Aug 29, 2017, 12:16 PM
@	769 B	Mar 15, 2017, 2:35 PM
a	0 B	Jul 25, 2017, 3:16 PM
a.csv	103 B	Jun 8, 2017, 1:40 PM
a.log	250 B	Jul 11, 2017, 3:34 PM
a.parquet		
accouting		

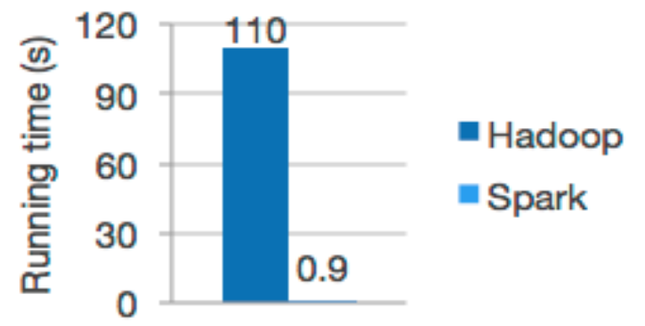
Apache Spark

Apache Spark is an open source cluster computing framework originally developed in the AMPLab at University of California, Berkeley but was later donated to the Apache Software Foundation where it remains today. In contrast to Hadoop's disk-based analytics paradigm, Spark has multi-stage in-memory analytics.

Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.



Logistic regression in Hadoop and Spark

Ease of Use

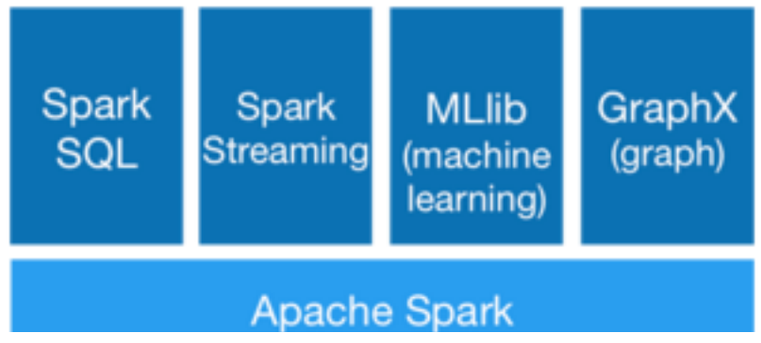
Write applications quickly in Java, Scala, Python, R.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python and R shells.

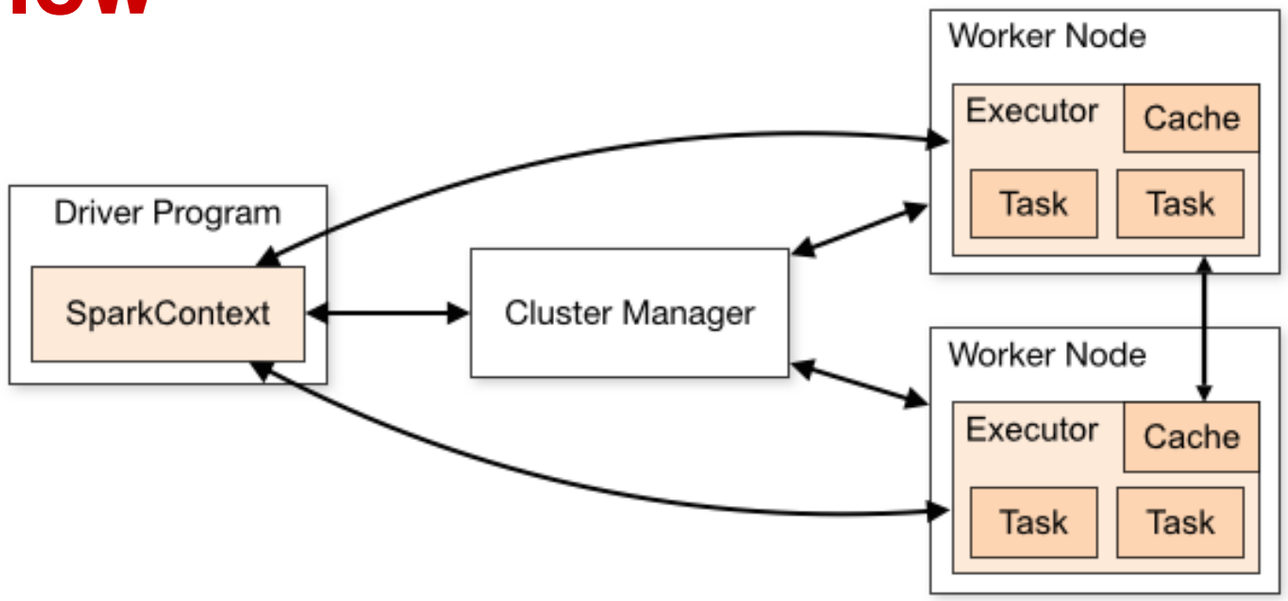
Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.



Spark workflow



Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program).

Requires cluster managers which allocate resources across applications.

Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application.

Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends tasks to the executors to run.

RDD- Resilient Distributed Datasets

RDD (Resilient Distributed Dataset) is the main logical data unit in Spark. They are

- Distributed and partitioned
- Stored in memory
- Immutable
- Partitions recomputed on failure

RDD- Transformations and Actions

Transformations are executed on demand. That means they are computed lazily. Eg: filter, join, sort

Actions return final results of RDD computations. Actions triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system. Eg: collect(), count(), take()

RDD Operations

Transformations	Actions
map(func)	take(N)
flatMap(func)	count()
filter(func)	collect()
groupByKey()	reduce(func)
reduceByKey(func)	takeOrdered(N)
mapValues(func)	top(N)
...	...

Interactive Analysis with the Spark Shell

$SPARK_HOME/bin/pyspark # Opens SparkContext

```
Python 2.7.5 (default, Oct 11 2015, 17:47:16)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/02/23 10:16:30 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Python version 2.7.5 (default, Oct 11 2015 17:47:16)
SparkSession available as 'spark'.
>>>
```

1. Create a RDD

```
>>> data = sc.textFile("README.md")
```

2. Transformation of RDD

```
>>>linesWithSpark = data.filter(lambda line: "Spark" in line)
```

3. Action on RDD

```
>>> linesWithSpark.count()   # Number of items in this RDD
12
```

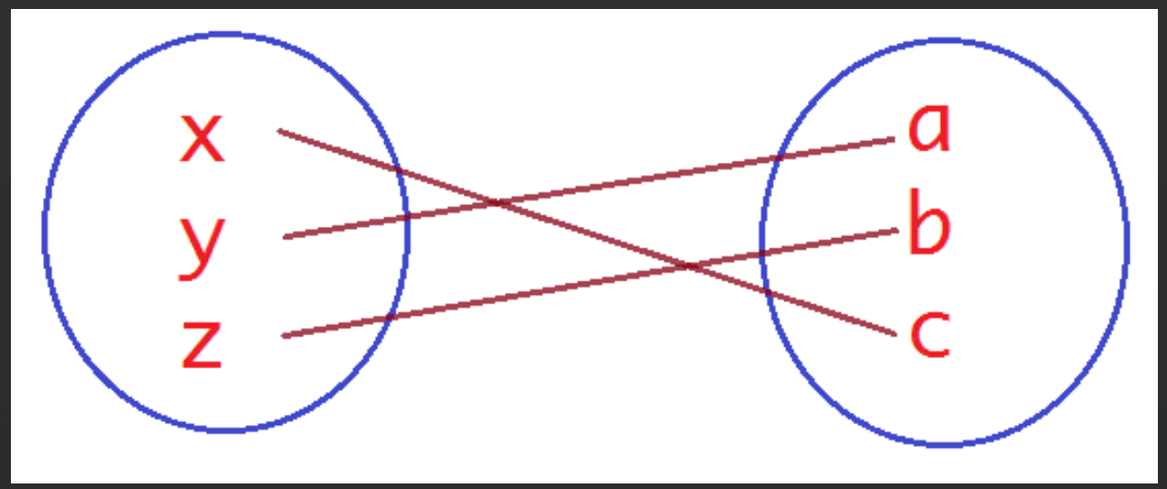
4. Combining Transformation and Actions

```
>>> data.filter(lambda line: "Spark" in line).count() # How many lines contain "Spark"?
12
```

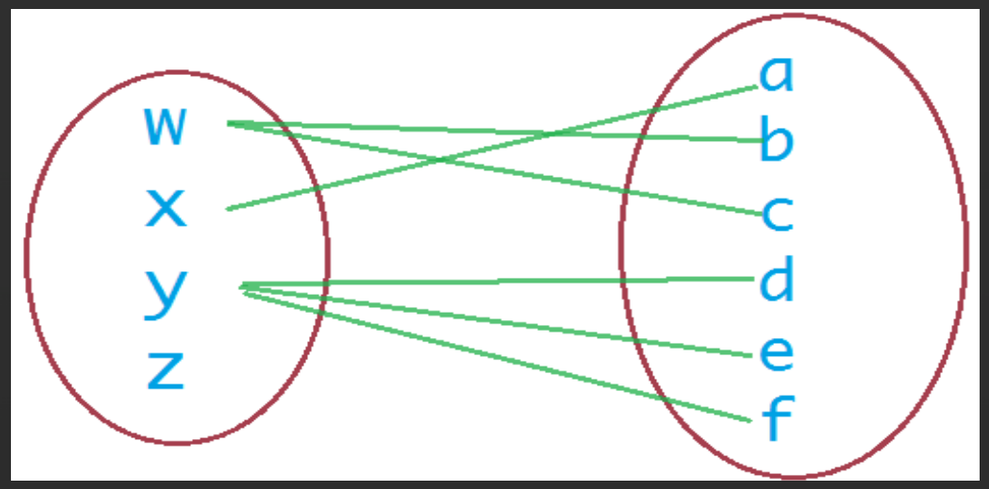
Word count Example

Map: One element in input gets mapped to only one element in output.
Flatmap: One element in input maps to zero or more elements in the output.

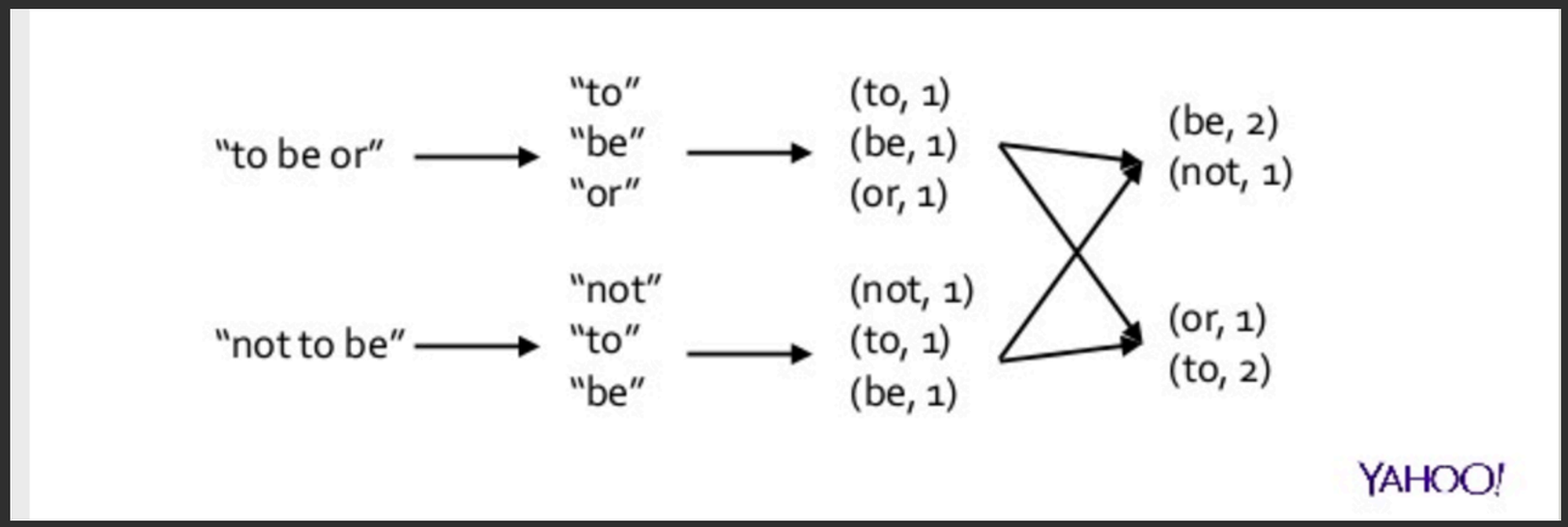
Map



Flatmap



Word count Example




```
>>>wordCounts = data.flatMap(lambda line: line.split()).map(lambda word:
(word,1)).reduceByKey(lambda a, b: a+b)

>>> wordCounts.collect()
```

Spark documentation at OSC

https://www.osc.edu/resources/available_software/software_list/spark

Availability & Restrictions

Spark is available to all OSC users without restriction.

The following versions of Spark are available on OSC systems:

VERSION	OAKLEY	OWENS
1.5.2	X	
1.6.1	X	
2.0.0*	X	X

NOTE: * means it is the default version.

Set-up

In order to configure your environment for the usage of Spark, run the following command:

```
module load spark
```

In order to access a particular version of Spark, run the following command

```
module load spark/2.0.0
```

Running Spark interactively in batch

To run Spark interactively, but in batch on Owens please run the following command,

```
qsub -I -l nodes=4:ppn=28 -l walltime=01:00:00
```

When your interactive shell is ready, please launch spark cluster using the pbs-spark-submit script

```
pbs-spark-submit
```

You can then launch the interface for pyspark as follows,

```
pyspark --master spark://nodename.ten.osc.edu:7070
```

```
Python 2.7.5 (default, Oct 11 2015, 17:47:16)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/02/23 10:16:30 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Python version 2.7.5 (default, Oct 11 2015 17:47:16)
SparkSession available as 'spark'.
>>>
```

Running Spark non-interactively

Using Spark

In order to run Spark in batch, reference the example batch script below. This script requests 6 node on the Oakley cluster for 1 hour of walltime. The script will submit the pyspark script called test.py using pbs-spark-submit command into the PBS queue.

```
#PBS -N Spark-example

#PBS -l nodes=6:ppn=12

#PBS -l walltime=01:00:00

module load spark

cd $PBS_O_WORKDIR

cp test.py $TMPDIR

cd $TMPDIR

pbs-spark-submit test.py  > test.log

cp * $PBS_O_WORKDIR
```

Running Spark using PBS script

1. Create an App in python: stati.py

```
from pyspark import SparkContext
import urllib
f = urllib.urlretrieve ("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data.gz","kddcup.data.gz")

data_file = "./kddcup.data.gz"
sc = SparkContext(appName="Stati")
raw_data = sc.textFile(data_file)

import numpy as np

def parse_interaction(line):
    line_split = line.split(",")
    symbolic_indexes = [1,2,3,41]
    clean_line_split=[item for i, item in enumerate(line_split) if i not in symbolic_indexes]
    return np.array([float(x) for x in clean_line_split])

vector_data=raw_data.map(parse_interaction)


from pyspark.mllib.stat import Statistics
from math import sqrt

summary = Statistics.colStats(vector_data)


print ("Duration Statistics:")
print (" Mean %f" % (round(summary.mean()[0],3)))
print ("St. deviation : %f"%(round(sqrt(summary.variance()[0]),3)))
print (" Max value: %f"%(round(summary.max()[0],3)))
print (" Min value: %f"%(round(summary.min()[0],3)))
```

2. Create a PBS script: stati.pbs

```
#PBS -N spark-statistics
#PBS -l nodes=18:ppn=28
#PBS -l walltime=00:10:00
module load spark/2.0.0
cp stati.py $TMPDIR
cd $TMPDIR
pbs-spark-submit stati.py > stati.log
cp *  $PBS_O_WORKDIR
```

3. Run Spark job

```
qsub stati.pbs
```

4. Output: stati.log

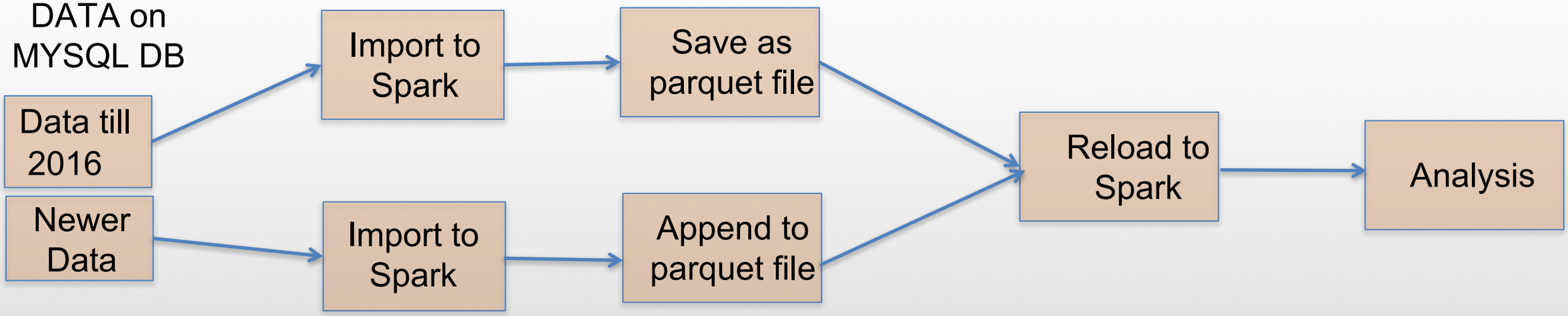
```
sync from spark://n0381.ten.osc.edu:7077
starting org.apache.spark.deploy.master.Master, logging to
/nfs/15/soottikkal/spark/kdd/spark-soottikkal-org.apache.spark.deploy.master.Master-1-
n0381.ten.osc.edu.out
failed to launch org.apache.spark.deploy.master.Master:
full log in /nfs/15/soottikkal/spark/kdd/spark-soottikkal-
org.apache.spark.deploy.master.Master-1-n0381.ten.osc.edu.out

Duration Statistics:
Mean 48.342000
St. deviation : 723.330000
Max value: 58329.000000
Min value: 0.000000
Total value count: 4898431.000000
Number of non-zero values: 118939.000000


SPARK_MASTER=spark://n0381.ten.osc.edu:7077
```

CASE STUDY

Data mining of historical jobs records of OSC's clusters

Aim: To understand client utilizations of OSC recourses.

Data: Historical records of every Job that ran on any OSC clusters that includes information's such as number of nodes, software, CPU time and timestamp.



Pyspark code for data analysis

#importing data

```
df=sqlContext.read.parquet("/fs/scratch/pbsacct/Jobs.parquet")
df.show(5)
```

jobid	username	system	nproc	submit_date	end_date	jobname	sw_app	queue
13780.owens-batch...	[redacted]3	owens	280	2016-09-28	2016-10-08	MMPCDH24EC1-3-2eq...	namd	parallel
13786.owens-batch...	[redacted]4	owens	96	2016-09-28	2016-10-05	FR181-011DS	foam	parallel
13798.owens-batch...	[redacted]0	owens	252	2016-09-28	2016-10-03	TSRD-5-3-012DS	foam	parallel
13800.owens-batch...	[redacted]0	owens	252	2016-09-28	2016-10-02	TSRD-5-3-013MSE	foam	parallel
13804.owens-batch...	[redacted]0	owens	252	2016-09-28	2016-10-02	TSRD-5-3-014MSE	foam	parallel

#Which types of queue is mostly used

```
df.select("jobid","queue").groupBy("queue").count().show()
```

queue	count
debug	157
serial	288174
montecarlo	12
parallel	41214
hugemem	102
largeparallel	60
longserial	66
dedicated	8

#Which software is used most?

```
df.select("jobid","sw_app").groupBy
("sw_app").count().sort(col("count").desc()) .show()
```

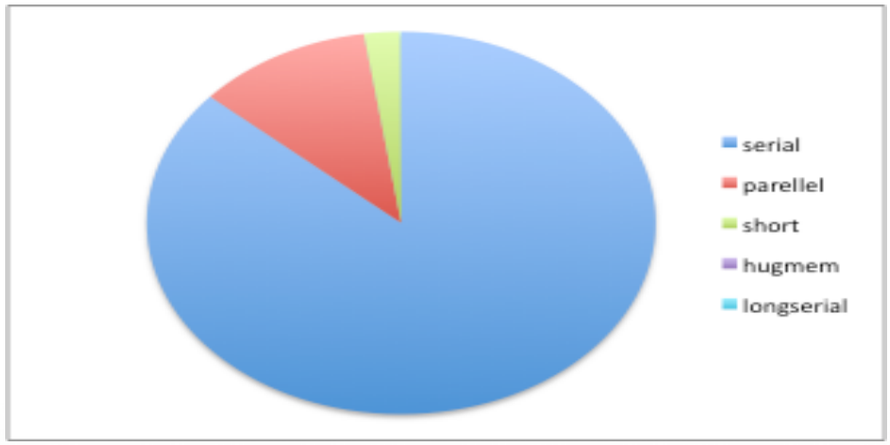
sw_app	count
condor	40199
fastsimcoal	39535
null	36914
amber	35304
real_exe	31076
molcas	23695
vasp	18164
gadget	13880
bam	13189
hpl	9820

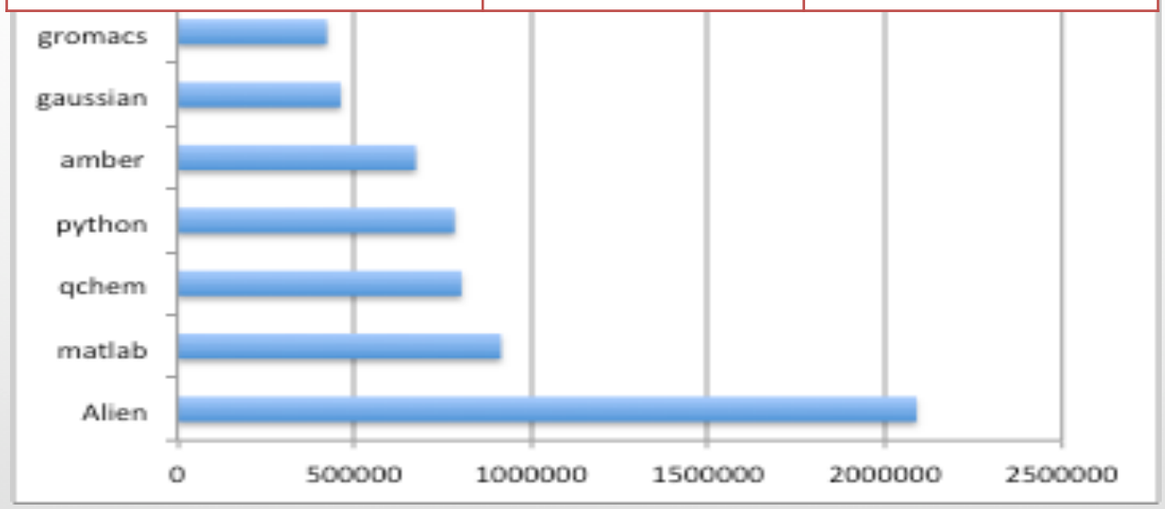
#who uses gaussian software most?

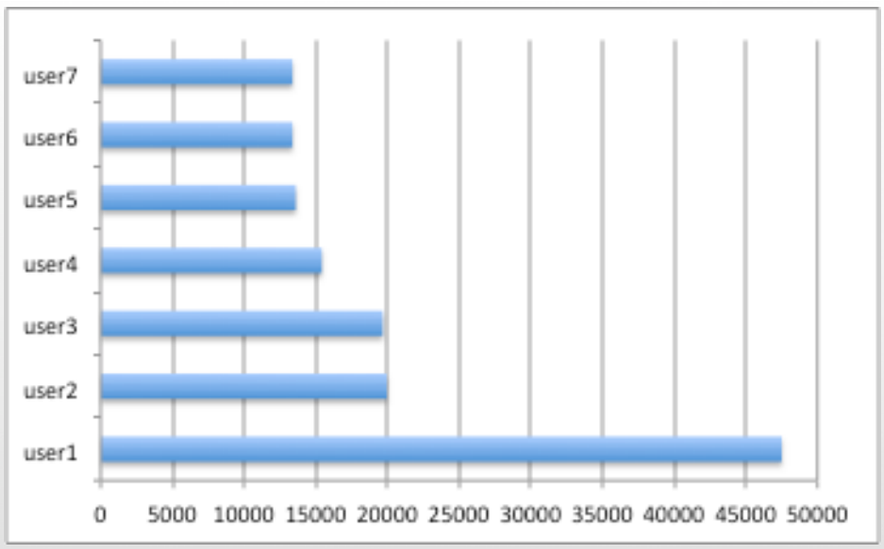
```
df.registerTempTable("Jobs")
sqlContext.sql(" SELECT username FROM
Jobs WHERE sw_app='gaussian' " ).show()
```

Results

Statistics	MYSQL	SPARK
Job vs CPU	1 hour	5 sec
CPU vs Account	1.25 hour	5 sec
Walltime vs user	1.40 hour	5 sec







Running Hadoop at OSC

A Hadoop cluster can be launched within the HPC environment, but managed by the PBS job scheduler using Myhadoop framework developed by San Diego Supercomputer Center. (Please see http://www.sdsc.edu/~allans/MyHadoop.pdf)

Availability & Restrictions

Hadoop is available to all OSC users without restriction.

The following versions of Hadoop are available on OSC systems:

VERSION	OAKLEY	OWENS
3.0.0*		X

NOTE: * means it is the default version.

Set-up

In order to configure your environment for the usage of Hadoop, run the following command:

```
module load hadoop
```

In order to access a particular version of Hadoop, run the following command

```
module load hadoop/3.0.0-alpha1
```

Using Hadoop: Sample PBS Script

```
#PBS -N hadoop-example

#PBS -l nodes=6:ppn=12

#PBS -l walltime=01:00:00

setenv WORK $PBS_O_WORKDIR

module load hadoop/3.0.0-alpha1

module load myhadoop/v0.40

setenv HADOOP_CONF_DIR $TMPDIR/mycluster-conf-$PBS_JOBID

cd $TMPDIR

myhadoop-configure.sh -c $HADOOP_CONF_DIR -s $TMPDIR

$HADOOP_HOME/sbin/start-dfs.sh

hadoop dfsadmin -report

hadoop  dfs -mkdir data

hadoop  dfs -put $HADOOP_HOME/README.txt  data/

hadoop  dfs -ls data

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0-alpha1.jar
wordcount data/README.txt wordcount-out

hadoop  dfs -ls wordcount-out

hadoop  dfs  -copyToLocal -f  wordcount-out  $WORK

$HADOOP_HOME/sbin/stop-dfs.sh

myhadoop-cleanup.sh
```

Using Hadoop: Sample PBS Script

```
#PBS -N hadoop-example

#PBS -l nodes=6:ppn=12

#PBS -l walltime=01:00:00

setenv WORK $PBS_O_WORKDIR

module load hadoop/3.0.0-alpha1

module load myhadoop/v0.40

setenv HADOOP_CONF_DIR $TMPDIR/mycluster-conf-$PBS_JOBID

cd $TMPDIR

myhadoop-configure.sh -c $HADOOP_CONF_DIR -s $TMPDIR

$HADOOP_HOME/sbin/start-dfs.sh

hadoop dfsadmin -report

hadoop  dfs -mkdir data

hadoop  dfs -put $HADOOP_HOME/README.txt  data/

hadoop  dfs -ls data

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0-alpha1.jar
wordcount data/README.txt wordcount-out

hadoop  dfs -ls wordcount-out

hadoop  dfs  -copyToLocal -f  wordcount-out  $WORK

$HADOOP_HOME/sbin/stop-dfs.sh

myhadoop-cleanup.sh
```

Spark Exercise

Connect to Owens cluster through putty terminal:
ssh username@owens.osc.edu
Enter password

```
#Copy necessary files
cp -r ~soottikkal/workshop/Oct17-Bigdata ./

#check files
cd Oct17-Bigdata
ls
cat instructions

# request 1 interactive node

qsub -I -l nodes=1:ppn=28 -l walltime=04:00:00 -A PZS0687

#check files
cd Oct17-Bigdata
ls
cd spark

#launch spark
module load spark/2.0.0
pyspark --executor-memory 10G --driver-memory 10G
```

```
#Example 1: Unstructured Data

#create a RDD
>>> data = sc.textFile("Oct17-Bigdata/spark/README.md")

#count number of lines
>>> data.count()
99

#see the content of the RDD
>>> data.take(3)
[u'# Apache Spark', u'', u'Spark is a fast and general cluster computing system for Big Data. It provides']
>>> data.collect()

#check data type
>>> type(data)
<class 'pyspark.rdd.RDD'>

#transformation of RDD
>>> linesWithSpark = data.filter(lambda line: "Spark" in line)
#action on RDD
>>> linesWithSpark.count()
19

##combining transformation and actions
>>> data.filter(lambda line: "Spark" in line).count()
19
```

```
#Example 2: Structured Data

#About the data: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99

#load data and run basic operations
>>> data=spark.read.csv(”data.csv", header='TRUE’)
>>> data.count()
494021

>>> data.take(1)
[Row(dst_bytes=u'5450', duration=u'0', flag=u'SF', protocal_type=u'tcp', service=u'http',
src_bytes=u'181')]

>>> data.take(3)
 Row(dst_bytes=u'5450', duration=u'0', flag=u'SF', protocal_type=u'tcp', service=u'http',
src_bytes=u'181'), Row(dst_bytes=u'486', duration=u'0', flag=u'SF', protocal_type=u'tcp', service=u'http',
src_bytes=u'239'), Row(dst_bytes=u'1337', duration=u'0', flag=u'SF', protocal_type=u'tcp',
service=u'http', src_bytes=u'235')]

>>> data.printSchema()
root
 |-- dst_bytes: long (nullable = true)
 |-- duration: long (nullable = true)
 |-- flag: string (nullable = true)
 |-- protocal_type: string (nullable = true)
 |-- service: string (nullable = true)
 |-- src_bytes: long (nullable = true)
```

```
>>>data.show(5)
```

```
+---------+--------+----+-------------+-------+---------+
|dst_bytes|duration|flag|protocal_type|service|src_bytes|
+---------+--------+----+-------------+-------+---------+
|    29200|       0|  S1|          tcp|   http|      228|
|     9156|       0|  S1|          tcp|   http|      212|
|        0|       0| REJ|          tcp|  other|        0|
|        0|       0| REJ|          tcp|  other|        0|
|        0|       0| REJ|          tcp|  other|        0|
+---------+--------+----+-------------+-------+---------+
only showing top 5 rows
```

```
>>> data.select("dst_bytes","flag").show(5)
```

```
+---------+----+
|dst_bytes|flag|
+---------+----+
|     5450|  SF|
|      486|  SF|
|     1337|  SF|
|     1337|  SF|
|     2032|  SF|
+---------+----+
```

```
>>> data.filter(interactions_df.flag!="SF").show(5)
```

```
+---------+--------+----+-------------+-------+---------+
|dst_bytes|duration|flag|protocal_type|service|src_bytes|
+---------+--------+----+-------------+-------+---------+
|    29200|       0|  S1|          tcp|   http|      228|
|     9156|       0|  S1|          tcp|   http|      212|
|        0|       0| REJ|          tcp|  other|        0|
|        0|       0| REJ|          tcp|  other|        0|
|        0|       0| REJ|          tcp|  other|        0|
+---------+--------+----+-------------+-------+---------+
only showing top 5 rows
```

Submitting Spark and Hadoop job non-interactively

```
cd spark
ls
qsub stati.pbs
qstat
qstat  | grep `whoami`
ls
qsub sql.pbs


cd hadoop
qsub sub-wordcount.pbs
qsub sub-grep.pbs
```

References

1. Spark Programming Guide

https://spark.apache.org/docs/2.0.0/programming-guide.html

-Programming with Scala, Java and Python

2. Data Exploration with Spark

http://www.cs.berkeley.edu/~rxin/ampcamp-ecnu/data-exploration-using-spark.html

3. Hadoop

http://hadoop.apache.org/

4. OSC Documentation

https://www.osc.edu/documentation/software_list/spark_documentation
https://www.osc.edu/resources/available_software/software_list/hadoop

Thank you!

- Questions or comments: soottikkal@osc.edu
- General questions about OSC service: oschelp@osc.edu