Datacenter Simulation Methodologies: Spark

Tamara Silbergleit Lehman, Qiuyun Wang, Seyed Majid Zahedi and Benjamin C. Lee

Tutorial Schedule

Time	Topic
9:00 - 10:00	Setting up MARSSx86 and DRAMSim2
10:00 - 10:30	Web search simulation
10:30 - 11:00	GraphLab simulation
11:00 - 11:30	**Spark simulation**
11:30 - 12:30	Questions and hands on session

Agenda

- Objectives
  - be able to deploy data analytics framework
  - be able to simulate Spark engine, tasks
- Outline
  - Learn Spark with interactive shell
  - Instrument Spark for simulation
  - Create checkpoints
  - Simulate from checkpoints

What is Spark?

Apache Spark is a fast and general engine for large-scale data processing

- Efficiency sources
  - General execution graphs
  - In-memory storage
- Usability sources
  - Rich APIs in Python, Scala, Java
  - Interactive shell

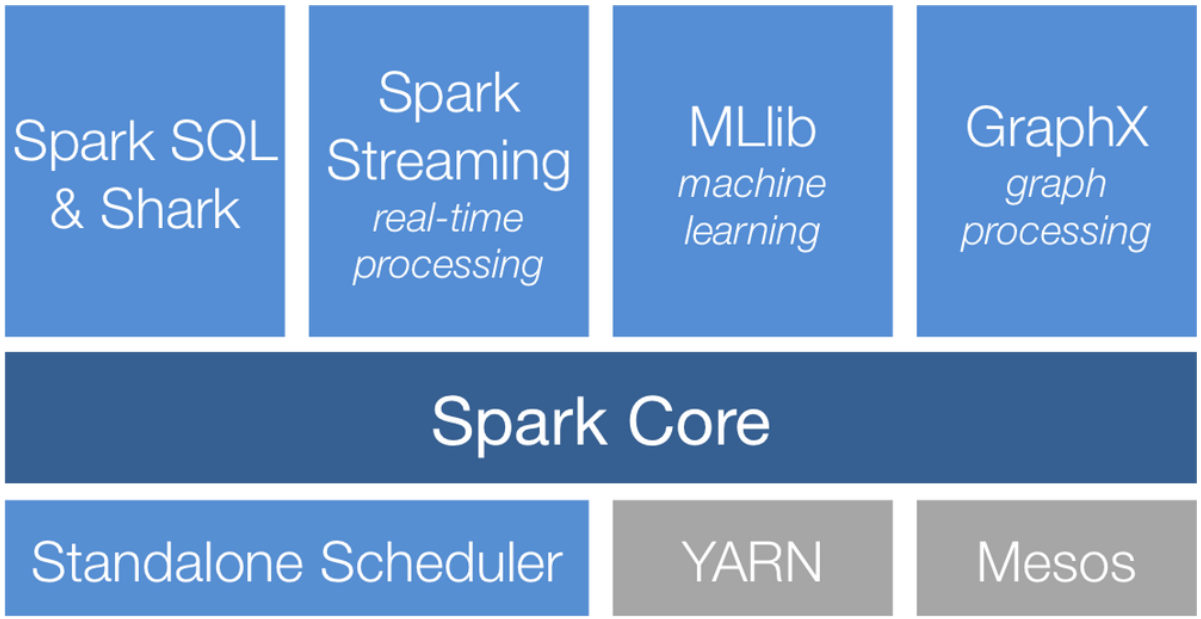
http://spark.apache.org/

Spark History

- Started in 2009
- Open sourced in 2010
- Many companies use Spark
  - Yahoo!, Intel, Adobe, Quantifind, Conviva, Ooyala, Bizo and others
- Many companies are contributing to Spark
  - Over 24 companies
- More information: `http://spark.apache.org/`

Spark Stack

- Spark is a part of the Berkeley Data Analytics Stack
- Spark unifies multiple programming models on same engine
  - SQL, streaming, machine learning, and graphs



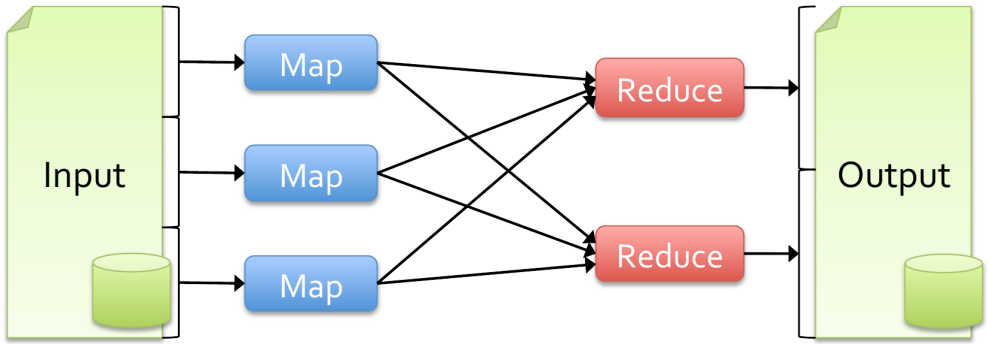
https://www.safaribooksonline.com

Benefits of Unification

- For the engine
  - Reduction in engine code size
  - Improvement in engine performance
- For users
  - Composition of different models
    (e.g. run SQL query then PageRank on results)
  - Fast composition (no writing to disk)
  - Easy status inspection with Spark shell

Duke
UNIVERSITY

Why Spark?

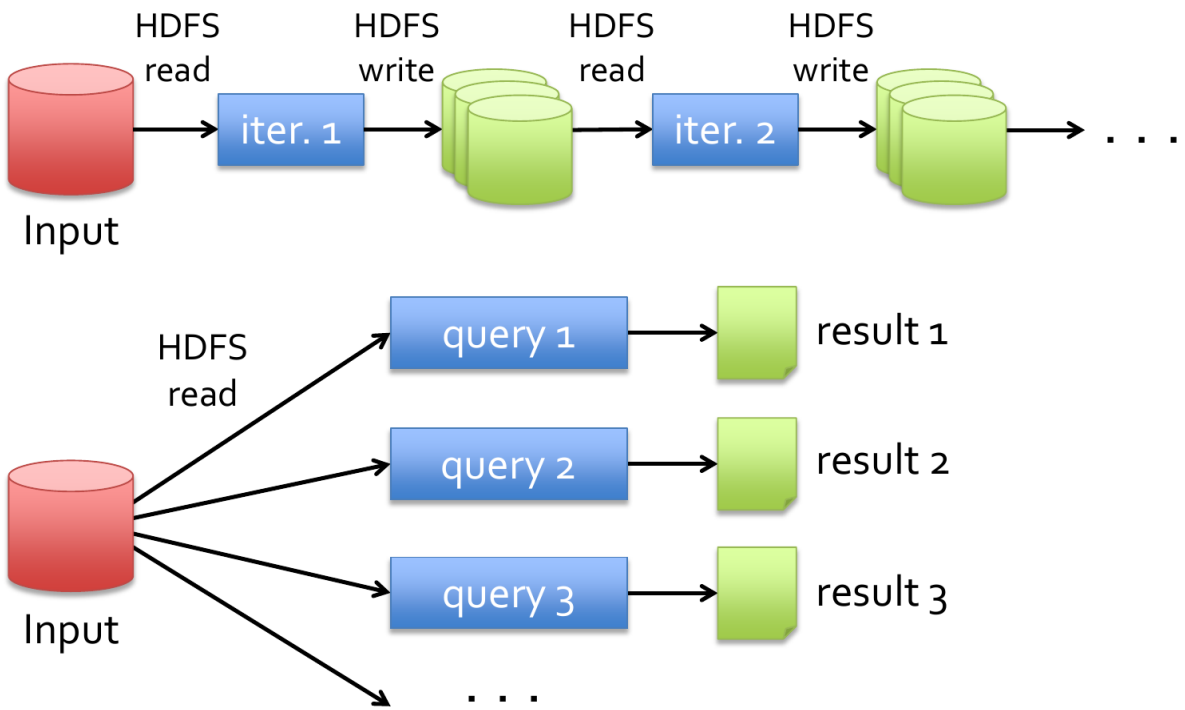


[biglearn_spark_novideo.pdf]

- MapReduce simplifies big data analysis
- However, it performs poorly for:
  - Complex, multi-pass analytics (e.g. ML, graph)
  - Interactive ad-hoc queries
  - Real-time stream processing

Data Sharing in MapReduce

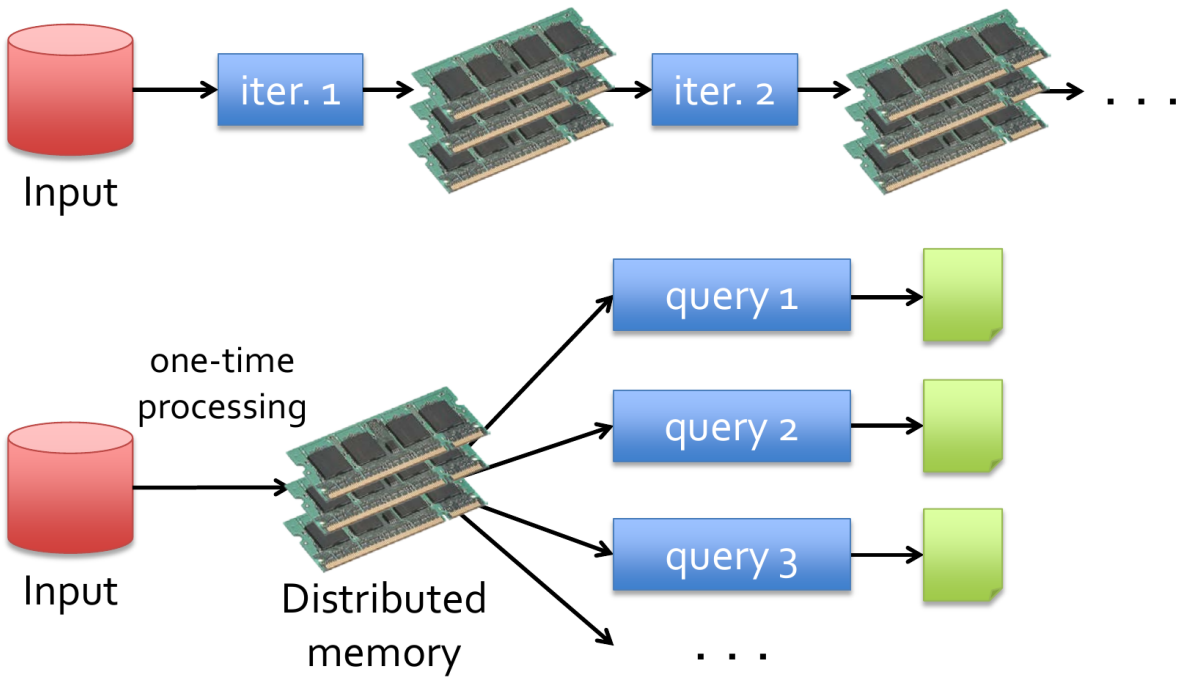
- Mapreduce model is slow
  - replication, serialization, and disk IO



[biglearn_spark_novideo.pdf]

Data Sharing in Spark

- Spark is fast
  - In-memory accesses **10-100x** faster than network and disk



[biglearn_spark_novideo.pdf]

Key Idea: Resilient distributed datasets (RDDs)

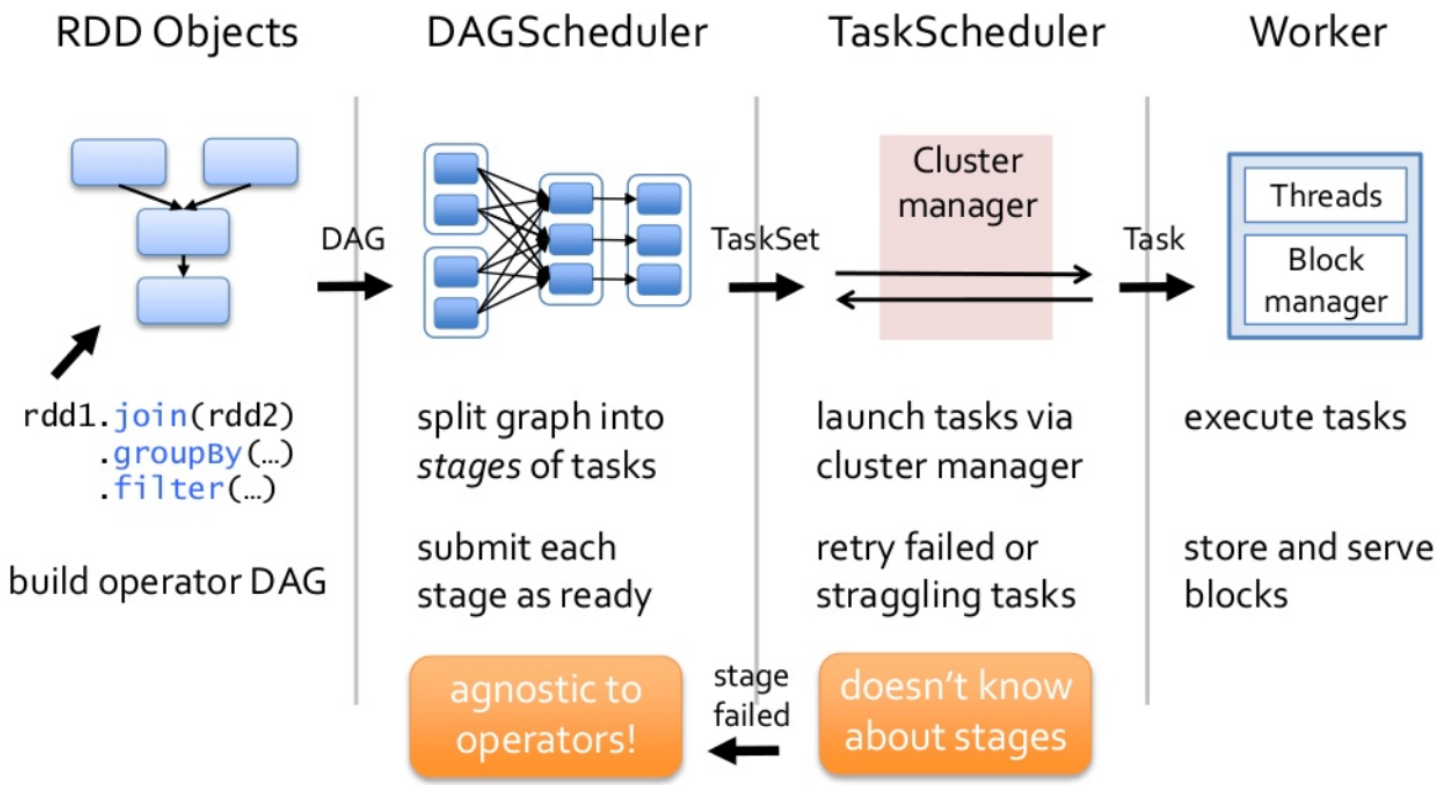
- Fault-tolerant collection of elements
  - Can be cached in memory
  - Can be manipulated through parallel operators
- Two ways to create RDDs
  - Modifying existing RDD
  - Referencing a dataset in external storage system
    - E.g., shared filesystem, HDFS, HBase, ...

Some Important Spark Operations

Transformations (define a new RDD)	map filter sample groupByKey reduceByKey sortByKey	flatMap union join cogroup cross mapValues
Actions (return a result to driver program)	collect reduce count save lookupKey	

[biglearn_spark_novideo.pdf]

Scheduling Process



[Spark Internals (http://www.slideshare.net)

Conclusion

- Spark provides faster framework for big data analytics
  - Complex analytics (e.g. machine learning)
  - Interactive ad-hoc queries
  - Real-time stream processing
- Spark unifies different models and enables sophisticated apps

Datacenter Simulation Methodologies
Getting Started with Spark

Tamara Silbergleit Lehman, Qiuyun Wang, Seyed Majid Zahedi
and Benjamin C. Lee

Agenda

- Objectives
  - be able to deploy data analytics framework
  - be able to simulate Spark engine, tasks
- Outline
  - Experiment with Spark
  - Instrument Spark tasks for simulation
  - Create checkpoints
  - Simulate from checkpoints

Install Spark

- Launch Qemu emulator

```
$ qemu-system-x86_64 -m 4G -nographic -drive
    file=demo.qcow2,cache=unsafe
```

- Install Java (may take ∼15min)

```
# apt-get update
# apt-get install openjdk-7-jdk openjdk-7-jre
```

- Download pre-built Spark

```
# wget http://d3kbcqa49mib13.cloudfront.net/
    spark-1.1.0-bin-hadoop1.tgz
# tar -xvf spark-1.1.0-bin-hadoop1.tgz
```

Duke UNIVERSITY

Interactive Analysis with the Spark Shell

- Launch Spark interactive Python interpreter

```
# cd spark-1.1.0-bin-hadoop1
# ./bin/pyspark
```

- Create RDD from input file

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

- Count number of items in RDD

```
>>> textFile.count()
```

- See first item in RDD

```
>>> textFile.first()
```

Duke UNIVERSITY

More on RDD Operations

- Filter all lines with "Spark"

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

- Count number of lines with "Spark"

```
>>> linesWithSpark.count()
```

- Find maximum number of words in lines:

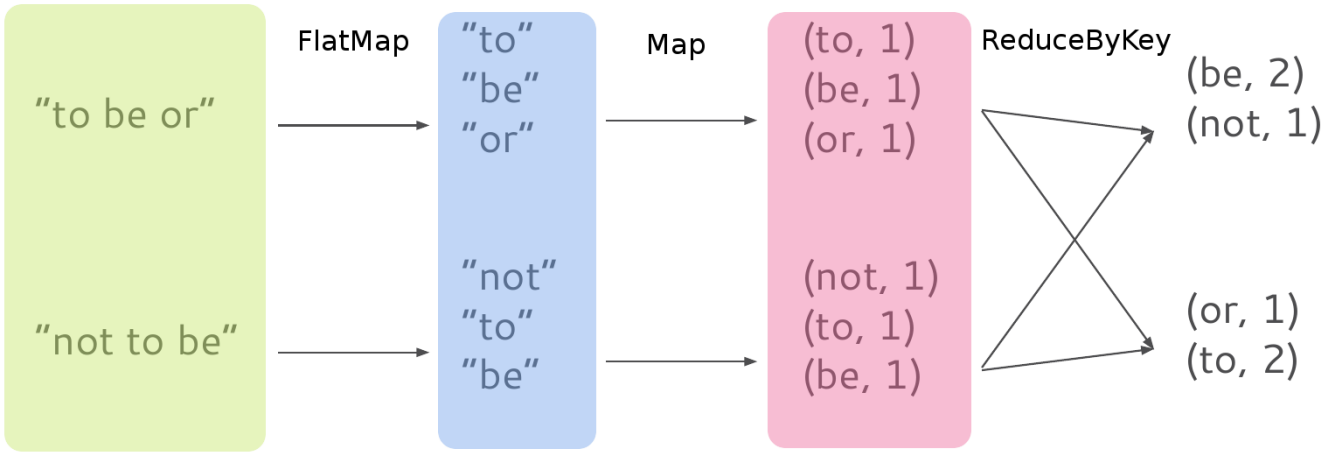
```
>>> textFile.
      map(lambda line: len(line.split())).
      reduce(lambda a, b: a if(a > b) else b)
```

More on RDD Operations (Cont.)

- Do same thing in different way:

```
>>> def max(a, b):
...     if a > b:
...         return a
...     else:
...         return b
>>> textFile.
      map(lambda line: len(line.split())).
      reduce(max)
```

WordCount Example



- Count words with map and reduce functions

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

- Return all the elements of the dataset

```
>>> wordCounts.collect()
```

Caching

- Spark supports cluster-wide in-memory caching

```
>>> linesWithSpark.cache()

>>> linesWithSpark.count()

>>> linesWithSpark.count()
```

- Very useful when data is accessed repeatedly
  - e.g., querying a small hot dataset
  - e.g., running an iterative algorithm like PageRank

Prepare Spark Simulation

- Exit python shell

```
>>> exit()
```

- Copy "ptlcalls.h"

```
# scp user@domain:/path/to/marss/ptlsim/tools
    /ptlcalls.h .
```

- Create ptlcalls.cpp file

```
# vim ptlcalls.cpp
```

ptlcalls.cpp

```
#include <iostream>
#include "ptlcalls.h"
#include <stdlib.h>

extern "C" void create_checkpoint(){
  char *ch_name = getenv("CHECKPOINT_NAME");
  if(ch_name != NULL) {
    printf("creating checkpoint %s\n",ch_name);
    ptlcall_checkpoint_and_shutdown(ch_name);
  }
}

extern "C" void stop_simulation(){
  printf("Stopping simulation\n");
  ptlcall_kill();
}
```

Build lib Library

- Install necessary packages

```
# apt-get install gcc g++ build-essential
```

- Compile C++ code

```
# g++ -c -fPIC ptlcalls.cpp -o ptlcalls.o
```

- Create shared library for Python

```
# g++ -shared -Wl,-soname,libptlcalls.so
    -o libptlcalls.so ptlcalls.o
```

- Copy the library

```
# cp libptlcalls.so ./bin/
```

Create Checkpoint for WordCount

- Include C++ library in WordCount python code (../examples/src/main/python/wordcount.py)

```
from ctypes import cdll
lib = cdll.LoadLibrary('./libptlcalls.so')
```

- Call C++ function to create checkpoint for reduceByKey phase

```
counts = lines.
      flatMap(lambda x: x.split(' ')).
      map(lambda x: (x, 1))
output = counts.collect()
lib.create_checkpoint()
counts = counts.reduceByKey(sample)
output = counts.collect()
lib.stop_simulation()
```

Create WordCount Checkpoint

- Shutdown Qemu emulator and run Marssx86's Qemu

```
# shutdown -h now

$ ./qemu/qemu-system-x86_64 -m 4G -drive
    file=image/spark.qcow2,cache=unsafe
    -nographic
```

- Export CHECKPOINT_NAME

```
# export CHECKPOINT_NAME=wordcount
```

- Run wordcount.py example

```
# cd spark-1.1.1-bin-hadoop1/bin
# ./spark-submit
    ../examples/src/main/python/wordcount.py
    ../README.md
```

Simulate "wordcount" Checkpoint

- Check wordcount checkpoint

```
$ qemu-img info ~/demo.qcow2
```

- Two ways to run from checkpoints
  - Manual run using terminal commands
  - Batch run using run_bench.py code

Manual Run

- Prepare wc.simcfg:

```
-logfile wordcount.log
-run
-machine single_core
-corefreq 3G
-stats wordcount.yml
-startlog 10M
-loglevl 1
-kill-after-run
-quiet
-dramsim-device-ini-file ini/
   DDR3_micron_32M_8B_x4_sg125.ini
-dramsim-results-dir-name wordcount
```

Manual Run (Cont.)

- Run terminal command:

```
$ ./qemu/qemu-system-x86_64 -m 4G -drive
    file=/path/to/image,cache=unsafe
    -nographic -simconfig wc.simcfg
    -loadvm wordcount
```

Batch Run

- Prepare util.cfg:
  (marss.dramsim/util/util.cfg)

```
[DEFAULT]
marss_dir = /path/to/marss/directory
util_dir = %(marss_dir)s/util
img_dir = /path/to/image
qemu_bin = %(marss_dir)s/qemu/qemu-system-x86_64

default_simconfig = -kill-after-run -quiet

[suite spark]
checkpoints = wordcount

[run spark_single]
suite = spark
images = %(img_dir)s/spark.qcow2
memory = 4G
simconfig = -logfile %(out_dir)s/%(bench)s.log
  -machine single_core
  -corefreq 3G
  -run
  -stats %(out_dir)s/%(bench)s.yml
  -dramsim-device-ini-file ini/DDR3_micron_32M_8B_x4_sg125.ini
  -dramsim-results-dir-name %(out_dir)s_%(bench)s
  -startlog 10M
  -loglevel 1
  %(default_simconfig)s
```

Batch Run (Cont.)

- Run run_bench.py

```
$ ./util/run_bench.py -d run/wordcount_test
    spark_single
```

- More information:
  http://marss86.org/~marss86/index.php/Batch_Runs

Other Libraries and Real Data Sets

- Libraries
  - Correlations (Correlation between label and features)
  - Kmeans
  - Decision Tree
  - Logistic Regression
- Data Sets
  - http://www.umass.edu/statdata/statdata/stat-logistic.html
  - http://cs.joensuu.fi/sipu/datasets/
  - http://www.limfinity.com/ir/

Duke UNIVERSITY

Agenda

- Objectives
  - be able to deploy data analytics framework
  - be able to simulate Spark engine, tasks
- Outline
  - Experiment with Spark
  - Instrument Spark tasks for simulation
  - Create checkpoints
  - Simulate from checkpoints

Duke UNIVERSITY

Thank You

- Questions?

Tutorial Schedule

Time	Topic
9:00 - 10:00	Setting up MARSSx86 and DRAMSim2
10:00 - 10:30	Web search simulation
10:30 - 11:00	GraphLab simulation
11:00 - 11:30	Spark simulation
11:30 - 12:30	**Questions and hands on session**