Teaching Apache Spark: Demonstrations on the Databricks Cloud Platform

Yao Yao and Mooyoung Lee

***Abstract*—Apache Spark is a fast and general engine for big data analytics processing with libraries for SQL, streaming, and advanced analytics. In this project, we will present an overview of Spark with the goal of giving students enough guidance to enable them to quickly begin using Spark 2.1 via the online Databricks Community Edition for future projects.**

***Index Terms*—Spark, Databricks, MapReduce, Python, SQL**

I. Introduction

Our goal is to present a concise overview of Apache Spark to our cohorts about its capabilities, uses, and applications via the Databricks cloud platform. The presentation will include original content, but will leverage existing resources where appropriate. This project consists of three deliverables: a 90-minute in-class presentation, recorded videos suitable for use on the 2DS asynchronous platform, and this paper documenting the key information. At the end of the presentation, students should feel confident knowing the basics of Spark to implement for future projects.

II. Problem Statement

There are three main challenges that Spark 2.0 intends to solve: CPU speed, end-to-end applications using one engine, and decision implementation based on reliable real-time data[1]. We plan to not only teach our fellow cohorts about the fundamentals of how to use Spark and the capabilities of the Spark ecosystem, but also provide compelling demonstrations of Spark code that would inspire future data science projects to be implemented on the online Databricks cloud platform.

III. Research Methodoligy

We learned how to use Spark on the online Databricks cloud platform via the Databricks documentation, Lynda.com, IBM's cognitiveclass, and O'Reilly's Definitive Guide. The videos are broken up into four parts to address the basic introduction and three demonstrations that use case studies to highlight the functionality of Spark.

Yao Yao is a graduate student in the Masters of Science in Data Science program at Southern Methodist University (e-mail: yaoyao@smu.edu). He resides in Chicago, Illinois.

Mooyoung Lee is a graduate student in the Masters of Science in Data Science program at Southern Methodist University (e-mail: mooyoungl@smu.edu). He resides in Plano, Texas.

The first video introduces Spark with its development history, ecosystem, and integrations. The first demo covers how to sign up for the community edition of Databricks.com, create a cluster, load in datasets and libraries, find datasets using the shell command, run various code using different programming commands and languages, check revision history, comment code, create a dashboard, and publish a notebook.

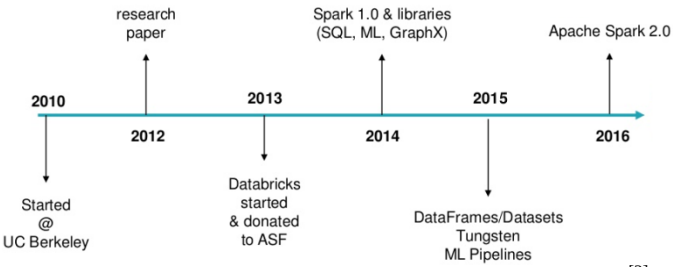
The second video addresses the CPU speed challenges, parallel computing, provides Spark's alternative solution to MapReduce, and Project Tungsten for Spark 2.0[1]. The second demo shows how sc.parallelize is used for parallel computing in a word count example, linear regression speed, and visualizations with GraphX.

The third video addresses the challenge to unify all analytics under one engine and how Spark allows for end-to-end applications to be built by being able to switch between languages, implement specific libraries, call on the same dataset in the same notebook, and call high level APIs to allow for vertical integration[1]. The third demo shows different programming languages calling on the same data set, Spark visualizations, Spark SQL, and the ability to read in and write the schema for JSON data.

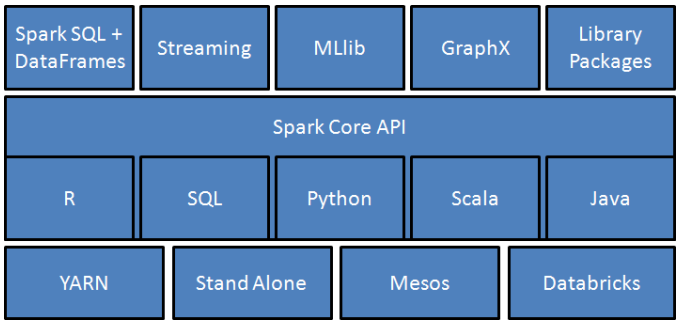
The fourth video addresses the challenges with the reliability of data streaming and the solutions that Spark provides for node crashes, asequential data, and data consistency[2]. The fourth demo shows a streaming example and a custom application with imported library packages.

IV. History / Ecosystem

Apache Spark is an unified analytics platform developed in 2009 at UC Berkley's AMPLab. It became open source in 2010, donated to the Apache Software Foundation in 2013, and is a top-contributed Apache project since 2014. Spark 1.0 was released in 2014, Project Tungsten was released in 2015, and Spark 2.0 was released in 2016 (Display 1).



Display 1. Timeline of Apache Spark and its developments[3].



Display 2: Spark's Ecosystem with Core API, platform, and high level APIs[4].

Spark uses common programming languages such as R, SQL, Python, Scala, and Java in the Core API for the end-user to be more productive towards various tasks (Display 2). Users can build their predictive models based on various library capabilities, while coding effectively in the language of their choice, being able to switch languages, and call on the same dataset in the same notebook[4].

High level APIs such as Spark SQL and DataFrames, data streaming, machine learning library, and GraphX allows users to build end-to-end apps[1]. The Spark open source ecosystem also allows the integration to external environments, data sources, and applications. Databricks cluster manager is used for the online platform. Spark can also run with Hadoop's YARN, Mesos, or as a standalone[5].

V. Spark 2.0's Solutions

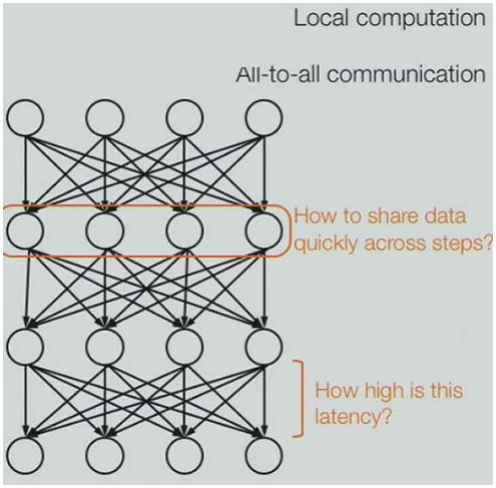
1. CPU Usage

	2010	2017
Storage	100 MB/s (HDD)	1000 MB/s (SSD)
Network	1 GB/s	10 GB/s
CPU	~3 GHz	~3 GHz

Display 3. Hardware trends of current computing systems[1]

As storage capacity and network increased by 10-fold, CPU speed remained relatively the same. Hardware may have more cores, but the speed is not 10 times faster (Display 3)[1]. We are close to the end of frequency scaling for CPU, where the speed cannot run more cycles per second without using more power and generating excessive heat.

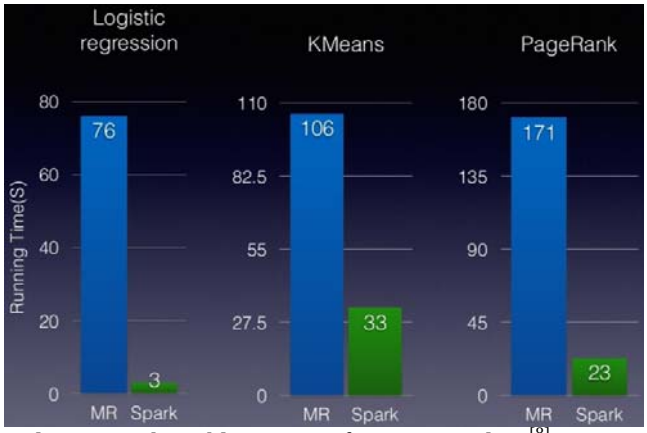
To accommodate for the CPU speed, hardware manufacturers have created multiple core parallel computing processes, which requires a form of MapReduce to compensate for distributed computation[1,6].



Display 4: MapReduce jobs execute in sequence due to the synchronization step[6]

The original MapReduce executed jobs in a simple but rigid structure, where "map" is the transformation step for local computation for each record, "shuffle" is the synchronization step, and "reduce" is the communication step to combine the results from all the nodes in a cluster (Display 4)[7]. MapReduce jobs execute in sequence and each of those jobs are high-latency, where no subsequent job could start until the previous job had finished completely.

Spark instead uses column ranges that keep their schema model and indexing, which could be read inside MapReduce records. Spark also uses an alternative multi-step directed acyclic graphs (DAGs), which mitigates slow nodes by executing nodes all at once and not step by step[7]. This eliminates the synchronization step required by MapReduce and lowers latency (Display 5). In addition, Spark supports in-memory data sharing across DAGs, so that different jobs can work with the same data at very high speeds[7].



Display 5: Spark is able to outperform MapReduce[8].

To further optimize Spark's CPU usage, Project Tungsten for Spark 2.0 mitigates runtime code generation, removes expensive iterator calls, and fuse multiple operators, where less excessive data is created in the memory[1]. User code is converted into binary, which complies to pointer arithmetic for the runtime to be more efficient (Display 6)[1]. This allows the user to code efficiently while performance is increased.

Dataframe code	df.where(df("year") > 2015)
Logical expression	GreaterThan(year#345, literal(2015))
Java Bytecode	bool filter(Object baseObject) { int offset = baseOffset + bitSetWidthInBytes + 3*8L; int value = Platform.getInt(baseObject, offset); return value > 2015;}

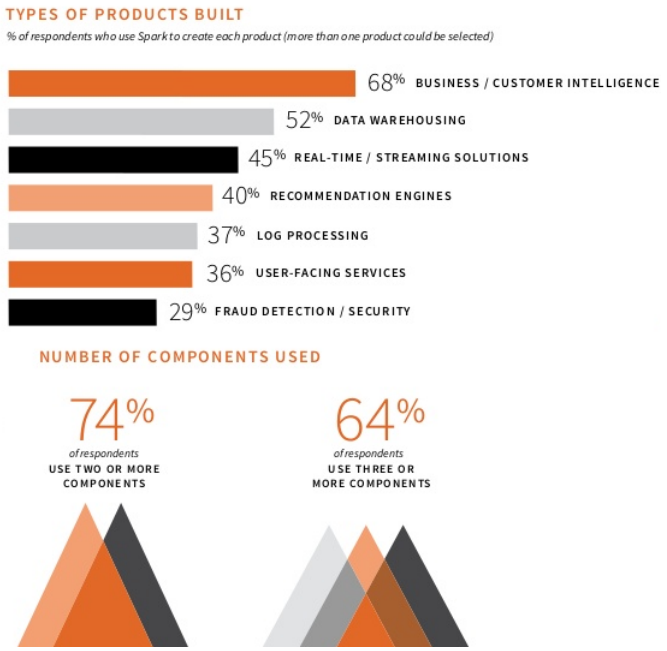
Display 6: The conversion of DataFrame code by the user into Java Bytecode for faster performance[1]

The Databricks online platform also benefits from cloud computing, where one can increase memory by purchasing multiple clusters for running parallel jobs[9]. Cloud computing also mitigates load time and frees up your local machine for other purposes.

2. End-to-End Applications

With multiple specialized engines intended for different applications, there are more systems to install, configure, connect, manage, and debug. Performance dwindles because it is hard to move big data across nodes and dynamically allocate resources for different computations. Writing temp data to file for another engine to run analysis slows down processes between systems[1].

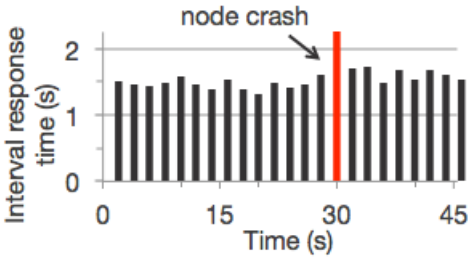
With the same engine, data sets could be cached in RAM while implementing different computations and dynamically allocating resources[10]. Built-in visualizations, high level APIs, and the ability to run multiple languages in the same notebook allow for vertical integration[4]. Performance gains are cumulative due to the aggregation of marginal gains while creating diverse products using multiple components (Display 7).



Display 7: Diversity in application solutions using multiple components in Spark[11]

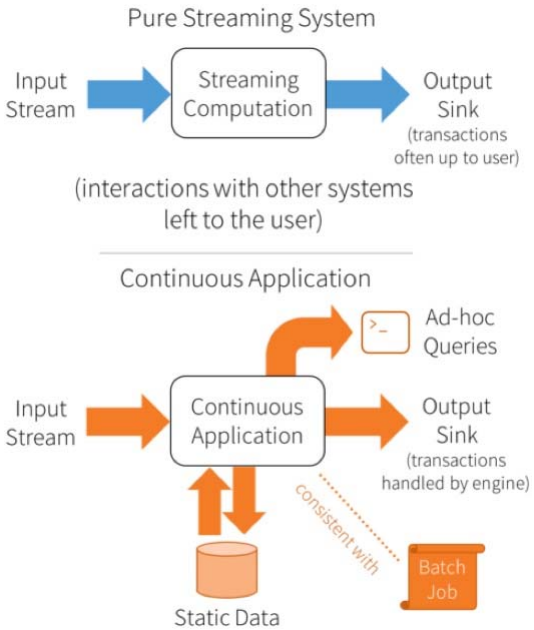
3. Decisions Based on Data Streaming

Streamed data may not be reliable due to node crashes, where artificial fluctuations in the data are caused by latency of the system, asequential data, where data is not in order, and data inconsistency, where multiple data storage locations may have different results (Display 8)[12]. Since Spark nodes have columns indexing, each node is tagged with time stamps and special referencing, where the fault tolerance fixes node crashes, special filtering fixes asequential data, and overall indexing fixes data consistency.



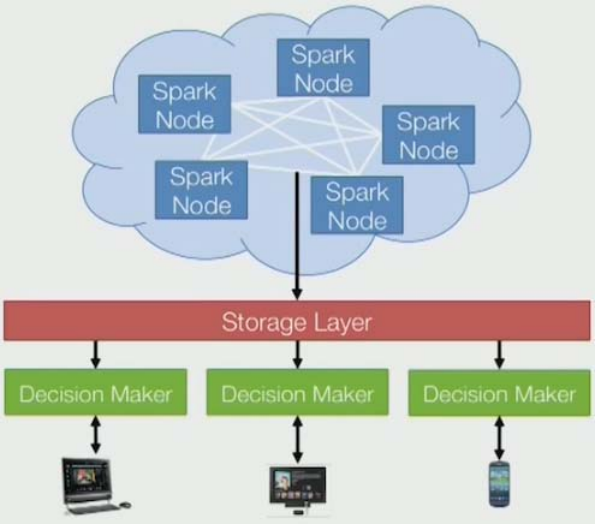
Display 8: Artificial fluctuations in the streaming data are caused by latency of the system and node crashes[12]

The benefits of using Spark is the speed needed to quickly use streaming data to make real time decisions and the sophistication needed to run best algorithms on the data. Being able to implement ad-hoc queries on top of streaming data, static data, and batch processes would allow more flexibility in the real-time decision making process than a pure streaming system (Display 9)[12].



Display 9: Pure streaming system verses continuous application streaming with queries and batch jobs[12]

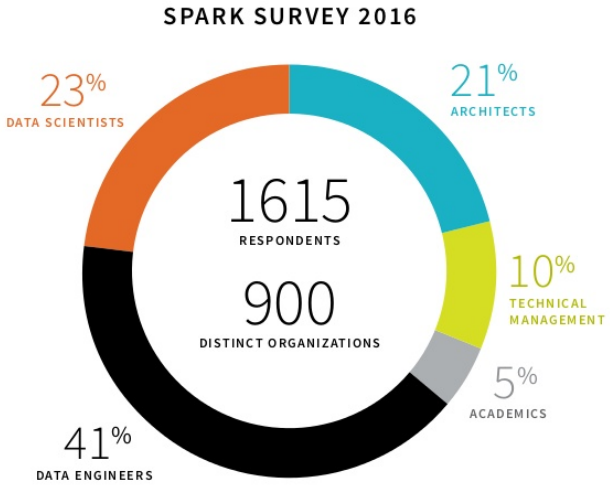
Continuous applications built on structured streaming allow for personalized results from web searches, automated stock trading, trends in news events, credit card fraud prevention, and video streaming, which benefit from implementing quick decisions per device based on various Spark applications from the cloud (Display 10)[13].



Display 10: Spark streaming solutions where each device can make decisions based on real-time data[13]

VI. Conclusion

Spark continues to dominate the analytical landscape with its efficient solutions to CPU usage, end-to-end applications, and data streaming[1]. The diversity of organizations, job fields, and applications that use Spark will continue to grow as more people find use in its various implementations and advanced analytics (Display 11)[11].



Display 11: Diversity in jobs that use Apache Spark[11]

VII. Deliverables

A. Combined Video

Getting Started with Spark, Spark's Approach to CPU Usage, End-to-End Applications, and Data Streaming Applications:

B. In-Class Presentation

Conducted on 8/2/17 at 8:30 PM CST to Dr. Sohail Rafiqi's MSDS7330 section 403 class

C. Other Resources

This paper, demonstration notebooks, and PowerPoint slides are located here: https://github.com/yaowser/learn-spark

VIII. Previous Work

[1] M. Zaharia. What to Expect for Big Data and Apache Spark in 2017. Available: https://youtu.be/vtxwXSGl9V8 and https://www.slideshare.net/databricks/spark-summit-east-2017-matei-zaharia-keynote-trends-for-big-data-and-apache-spark-in-2017

[2] Apache Spark Official Website. Available: http://spark.apache.org/

[3] Jump Start with Apache Spark 2.0 on Databricks. Available: https://www.slideshare.net/databricks/jump-start-with-apache-spark-20-on-databricks

[4] R. Rivera. The future of Apache Spark. Available: https://www.linkedin.com/pulse/future-apache-spark-rodrigo-rivera

[5] D. Lynn. Apache Spark Cluster Managers: YARN, Mesos, or Standalone? Available: http://www.agildata.com/apache-spark-cluster-managers-yarn-mesos-or-standalone/

[6] M. Zaharia. Three Ways Spark Went Against Conventional Wisdom. Available: https://youtu.be/y7KQcwK2w9I and https://www.slideshare.net/databricks/unified-big-data-processing-with-apache-spark-qcon-2014

[7] M. Olsen. MapReduce and Spark. Available: http://vision.cloudera.com/mapreduce-spark/

[8] Performance Optimization Case Study: Shattering Hadoop's Sort Record with Spark and Scala. Available: https://www.slideshare.net/databricks/2015-0317-scala-days

[9] Databricks Cloud Community Edition. Available: http://community.cloud.databricks.com/

[10] Introduction to Apache Spark on Databricks. Available: https://community.cloud.databricks.com/?o=7187633045765022#notebook/418623867444693/command/418623867444694

[11] A. Choudhary. 2016 Spark Survey. Available: https://www.slideshare.net/abhishekcreate/2016-spark-survey

[12] M. Zaharia, et al. Structured Streaming In Apache Spark. Available: https://databricks.com/blog/2016/07/28/structured-streaming-in-apache-spark.html

[13] M. Zaharia. How Companies are Using Spark, and Where the Edge in Big Data Will Be. Available: https://youtu.be/KspReT2JjeE

IX. Resources

Databricks Spark Documentation. Available: https://docs.databricks.com/spark/latest/

M. Mayo. Top Spark Ecosystem Projects. Available: http://www.kdnuggets.com/2016/03/top-spark-ecosystem-projects.html

B. Yavuz. Using 3rd Party Libraries in Databricks: Apache Spark Packages and Maven Libraries. Available: https://databricks.com/blog/2015/07/28/using-3rd-party-libraries-in-databricks-apache-spark-packages-and-maven-libraries.html

Apache Spark YouTube Channel. Available: http://www.youtube.com/channel/UCRzsq7k4-kT-h3TDUBQ82-w

B. Sullins. Apache Spark Essential Training. Available: https://www.lynda.com/Apache-Spark-tutorials/Apache-Spark-Essential-Training/550568-2.html

L. Langit. Extending Hadoop Data Sceince Streaming Spark Storm Kafka. Available: https://www.lynda.com/Hadoop-tutorials/Extending-Hadoop-Data-Science-Streaming-Spark-Storm-Kafka/516574-2.html

Spark Fundamentals I. Available: https://courses.cognitiveclass.ai/courses/course-v1:BigDataUniversity+BD0211EN+2016/info

Spark Fundamentals II. Available: https://courses.cognitiveclass.ai/courses/course-v1:BigDataUniversity+BD0212EN+2016/info

Apache Spark Makers Build. Available: https://courses.cognitiveclass.ai/courses/course-v1:BigDataUniversity+TMP0105EN+2016/info

Exploring Spark's GraphX. Available: https://courses.cognitiveclass.ai/courses/course-v1:BigDataUniversity+BD0223EN+2016/info

Analyzing Big Data in R using Apache Spark. Available: https://courses.cognitiveclass.ai/courses/course-v1:BigDataUniversity+RP0105EN+2016/info

Definitive Guide Apache Spark Excerpts. Available: http://go.databricks.com/definitive-guide-apache-spark

Definitive Guide Apache Spark Raw Chapters. Available: http://shop.oreilly.com/product/0636920034957.do

Spark The Definitive Guide Community Edition. Available: https://github.com/databricks/Spark-The-Definitive-Guide