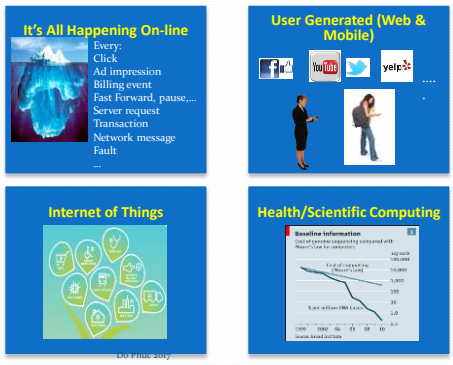
Data Science education and research

A/ Prof. Do Phuc
University of Information Technology, VNU-HCM
Year 2018

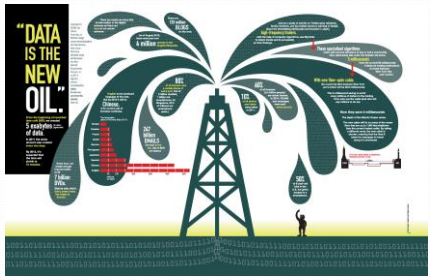
Outline

- Data science
- Big Data
- BSC on Data Science
- MSC on Data Science
- Some research directions
- Parallel Program with Apache Spark
- On Processing project
- References

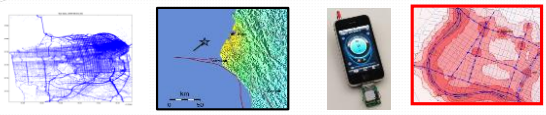
"Big Data" Sources



"Data is the New Oil" – World Economic Forum 2011



What can you do with the data? Traffic Prediction and Earthquake Warning



Crowdsourcing + physical modeling + sensing + data assimilation

to produce:

From Alex Bayen, UCB, Director, Institute for Transportation Studies

"Data Science" an Emerging Field

O'Reilly Radar report, 2011

Data Science – A Definition

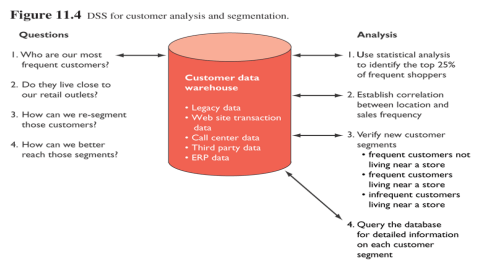
Data Science is the science which uses computer science, statistics and machine learning, visualization and human-computer interactions to collect, clean, integrate, analyze, visualize, interact with data to create data products.

Goal of Data Science

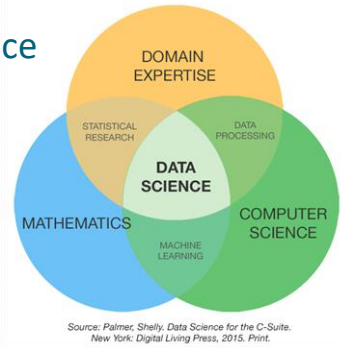
Turn data into data products (value).

Customer Relationship Management

DSS Applications



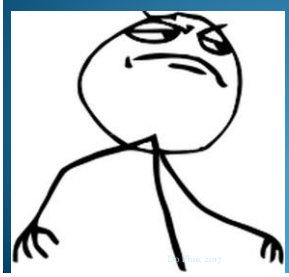
Data Science



Source: Palmer, Shelly. Data Science for the C-Suite. New York: Digital Living Press, 2015. Print.

Data Scientist is the sexiest job of 21st century

(c) Harvard Business Review



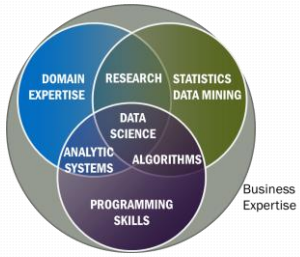
Oh, Really?

Job Trends

Big Data about an order of magnitude larger than data science

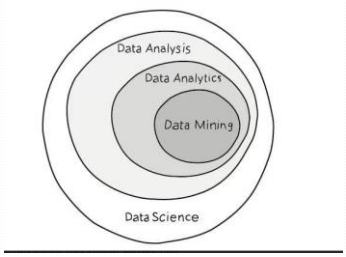
21 September 2014 15,639 jobs have "big data" phrase

- **Data Science** is the extraction of actionable knowledge directly from data through a process of discovery, hypothesis, and analytical hypothesis analysis.
- A **Data Scientist** is a practitioner who has sufficient knowledge of the overlapping regimes of expertise in business needs, domain knowledge, analytical skills and programming expertise to manage the end-to-end scientific method process through each stage in the big data lifecycle.



See Big Data Definitions in http://bigdatawg.nist.gov/V1_output_docs.php

Difference between data mining and data science



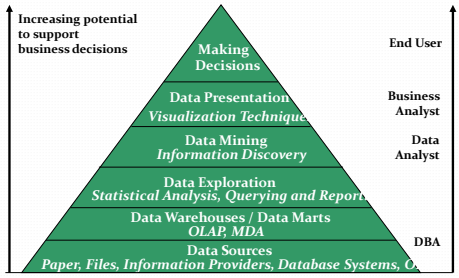
Difference between data mining and data science

- **Data mining :**refers to the science of collecting all the past data and then searching for patterns in this data.
- **Data Science** is an umbrella that contain many other fields like Machine learning, Data Mining, big Data, statistics, Data visualization, data analytics,...
- (source Ricardo Vladimiro, Game Analytics and Data Science Lead @ Miniclip, 2016)

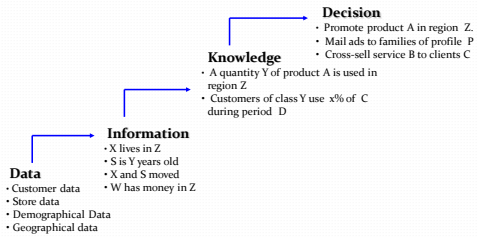
5 Vs of Big Data

- Raw Data: Volume
- Change over time: Velocity
- Data types: Variety (structured, unstructured)
- Data Quality: Veracity
- Information for Decision Making: Value

Utilization



From Data to Knowledge



Statistics

- Descriptive statistics
- Inferential statistics
- Multivariate analysis
  - Confirmation data analysis
  - Exploratory data analysis
- Big data: topic modeling
- Numeric data, machine learning

Machine Learning & Data Mining

- Machine learning
  - Build a system as human learns
  - Symbolic data
  - Statistic: Naïve Bayes
  - Large scale machine learning.
- Data mining
  - Discover implicit and useful knowledge from massive data sets

What is the difference between machine learning and data mining ?

- Machine Learning relates with the study, design and development of the algorithms that give computers the capability to learn without being explicitly programmed (definition of Arthur Samuel).
- Data Mining can be defined as the process that starting from apparently unstructured data tries to extract knowledge and/or unknown interesting patterns. During this process machine Learning algorithms are used.
- (source: Giovanni Di Orio · Institute for the Development of New Technologies)

Data

- Big data (5V)
- Complicated data type,
- Structured, un-structured data
- text, DNA, web,..

Methods extract knowledge from data

- Classification
- Clustering
- Neural networks
- Deep Learning
- Probabilistic Graphical Model: topic model,
- Visual analytics
- Text Mining
- Machine Learning & Data Mining
- Time series forecasting

Distributed Parallel Programming

- Hadoop, Map Reduce
- Spark: Java, Scala, Python

Data base

- SQL
- No SQL
  - MongoDB
  - Graph DB: Neo4j, OrientDB: node, link-> improve searching

Skills Acquired for Data Science Students

Fundamental/Technical :

SQL/ Data Modeling / Cleaning
Data Integration / Warehousing
Statistical Learning / Machine Learning
Distributed Computing
Big Data Management
Classif./Regression/DecisionTrees
Business Intelligence
Distributed Mining Algorithms

Tools :

Oracle /MySQL/DB2/SQLServer
R / SAS / SciKit
Weka /RapidMiner /MatLab
IBM Cognos / SPSS Modeler
Hadoop / Mahout / Cassandra
Python / Java / Cloud Computing
Scala/ Spark / InfoSphere Streams
Spotfire / Tableaux

Professional Skills:

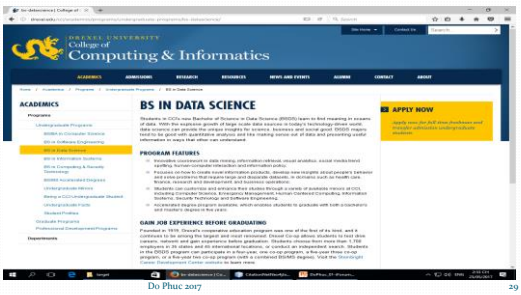
Business Use Cases / Entrepreneurship
Interdisciplinary Teams / Leadership

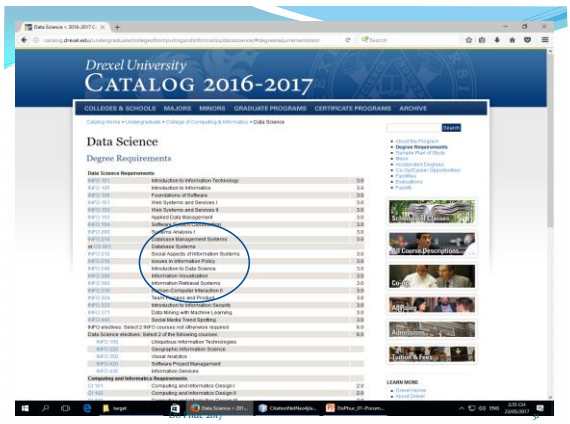
Professional Skills:

Story Telling / Visualization
Presentations / Reports

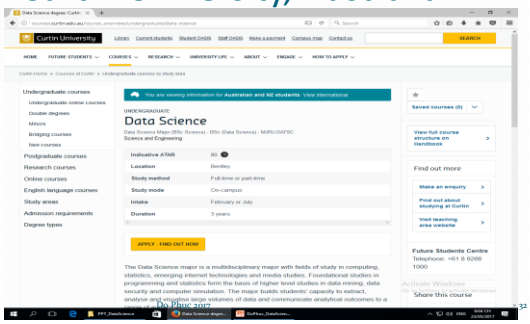
BCS in Data Science

Drexel University, Australia

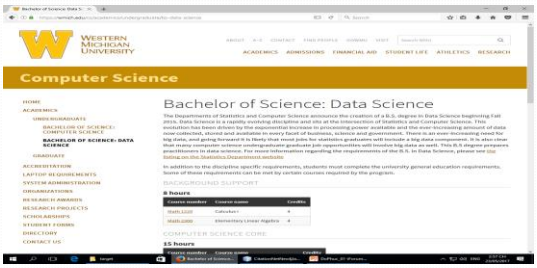


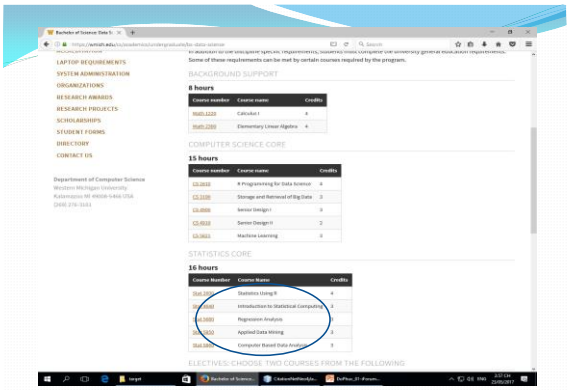


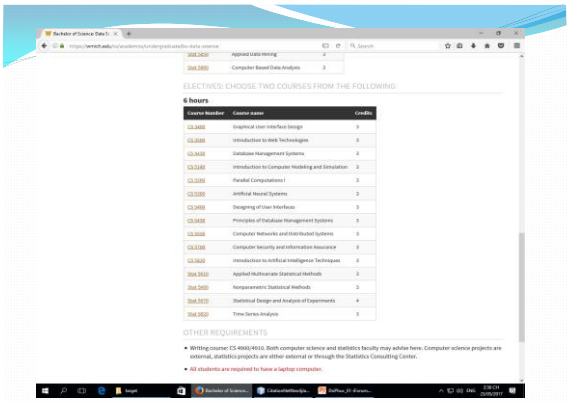
Curtin University, Australia



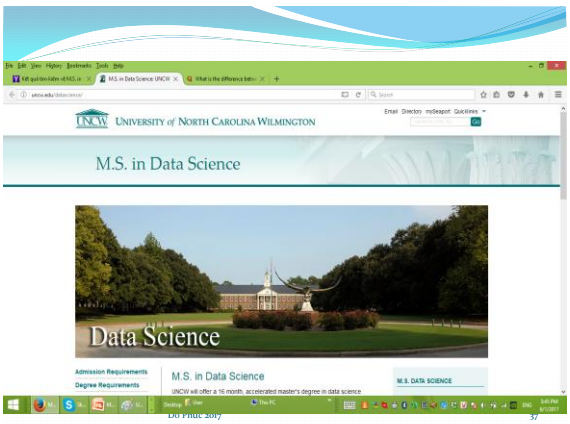
Western Michigan University, USA







MSC in Data Science



What is the Master of Data Science program?

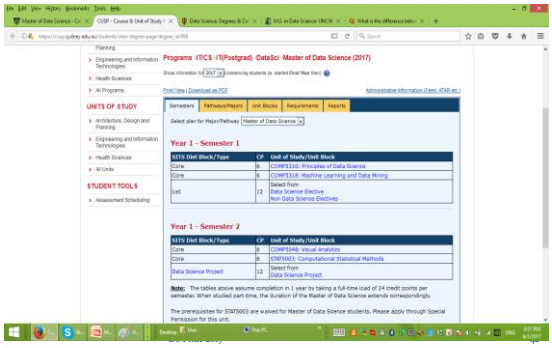
- The Master of Data Science is a professional interdisciplinary program that emphasizes computational-based applications of traditional data analysis methods and current trends in data mining and machine learning to turn data into information, and information into insight.

Students are offered the opportunity to:

- Develop proficiency in both a general-purpose programming language and a statistical programming language
- Advance skills to build appropriate statistical models to solve real-world analytics problems
- Enhance the ability to communicate analytical information effectively that solves data interpretation problems specific to employer needs

Core Coursework

- Computational Data Science
- Fundamentals of Statistics
- Linear Methods
- Machine Learning
- Categorical Data Analysis
- Spatial-Temporal Analysis
- Practicum/Internship



M.S. in Data Science Program

GRADUATE QUALIFYING PROJECT OR MS THESIS (3 TO 9 CREDITS)

CONCENTRATION AND ELECTIVES (9 TO 15 CREDITS)

MATHEMATICAL ANALYTICS (3 CREDITS)	DATA ACCESS & MANAGEMENT (3 CREDITS)	DATA ANALYTICS & MINING (3 CREDITS)	BUSINESS INTELLIGENCE & CASE STUDIES (3 CREDITS)

INTEGRATIVE DATA SCIENCE (3 CREDITS)

Data Science Core

INTEGRATIVE DATA SCIENCE :
DS 501 INTRODUCTION TO DATA SCIENCE (NEW COURSE)

MATHEMATICAL ANALYTICS (SELECT ONE):
MA 543/DS 502 STATISTICAL METHODS FOR DATA SCIENCE (NEW COURSE)
MA 542 REGRESSION ANALYSIS
MA 554 APPLIED MULTIVARIATE ANALYSIS

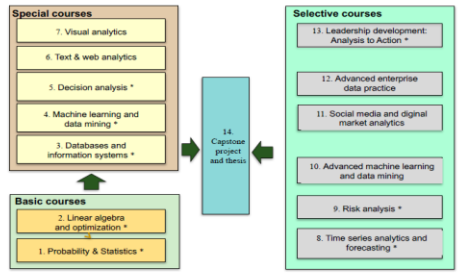
DATA ACCESS AND MANAGEMENT (SELECT ONE):
CS 542 DATABASE MANAGEMENT SYSTEMS
MIS 571 DATABASE APPLICATIONS DEVELOPMENT
CS 561 ADVANCED TOPICS IN DATABASE SYSTEMS
CS 585/DS 503 BIG DATA MANAGEMENT (NEW COURSE)

DATA ANALYTICS AND MINING (SELECT ONE):
CS 548 KNOWLEDGE DISCOVERY AND DATA MINING
CS 539 MACHINE LEARNING
CS 586/DS 504 BIG DATA ANALYTICS (NEW COURSE)

BUSINESS INTELLIGENCE AND CASE STUDIES (SELECT ONE):
MIS 584 BUSINESS INTELLIGENCE
MKT 568 DATA MINING BUSINESS APPLICATIONS

VNU-HCM

Master on data science at JVN



Some research problems

- Big data and high performance analytics
- Business intelligence and predictive analytics
- Visual analytics of large data sets
- Data mining and knowledge discovery for massive data set
- Financial decision-making
- Healthcare data analytics
- Internet big data analytics
- Large-scale data management and infrastructures
- Numerical and statistical data analysis
- Optimization and prescriptive analytics
- Signal processing and information theory
- Social media analytics
- Statistical and machine learning
- Bioinformatics and genomic databases

Programming With Scala and Spark

Source: Matei Zaharia, UC Berkeley

Apache Hadoop

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these modules:

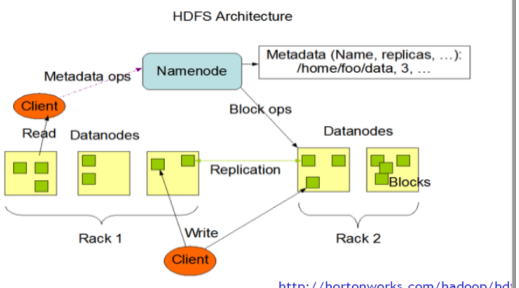
- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

http://hadoop.apache.org

Hadoop-related Apache Projects

- **Ambari™**: A web-based tool for provisioning, managing, and monitoring Hadoop clusters.It also provides a dashboard for viewing cluster health and ability to view MapReduce, Pig and Hive applications visually.
- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- **Mahout™**: A Scalable machine learning and data mining library.
- **Pig™**: A high-level data-flow language and execution framework for parallel computation.
- **Spark™**: A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.
- **Tez™**: A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases.
- **ZooKeeper™**: A high-performance coordination service for distributed applications.

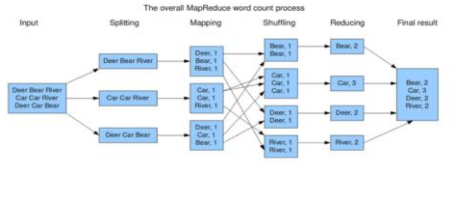
Hadoop Distributed File System (HDFS)



23 E6893 Big Data Analytics – Lecture 1: Overview © 2014 CY Lin, Columbia University

Do Phuc 2017 50

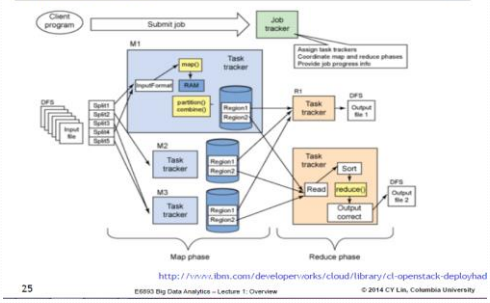
MapReduce example



24 E6893 Big Data Analytics – Lecture 1: Overview © 2014 CY Lin, Columbia University

Do Phuc 2017 51

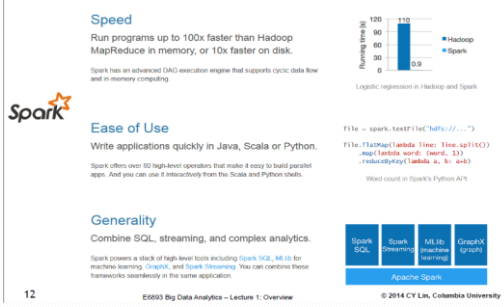
MapReduce Data Flow



25 E6893 Big Data Analytics – Lecture 1: Overview © 2014 CY Lin, Columbia University

Do Phuc 2017 52

In-Memory Example — Apache Spark



12 E6893 Big Data Analytics – Lecture 1: Overview © 2014 CY Lin, Columbia University

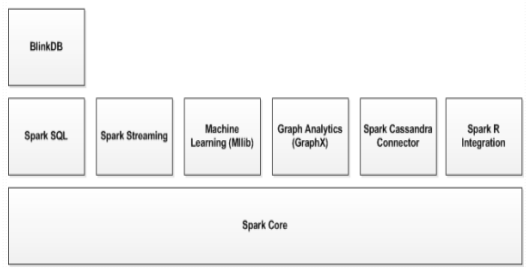
Do Phuc 2017 53

What is Spark?

- Fast, expressive cluster computing system compatible with Apache Hadoop
  - Works with any Hadoop-supported storage system (HDFS, S3, Avro, …) → Up to 100× faster
- Improves **efficiency** through:
  - In-memory computing primitives
  - General computation graphs → Often 2-10× less code
- Improves **usability** through:
  - Rich APIs in Java, Scala, Python
  - Interactive shell

Do Phuc 2017 54

Spark Framework Ecosystem



Do Phuc 2017 55

Spark Ecosystem

- **Spark Streaming:**
  - Spark Streaming can be used for processing the real-time streaming data. This is based on micro batch style of computing and processing. It uses the DStream which is basically a series of RDDs, to process the real-time data.
- **Spark SQL:**
  - Spark SQL provides the capability to expose the Spark datasets over JDBC API and allow running the SQL like queries on Spark data using traditional BI and visualization tools. Spark SQL allows the users to ETL their data from different formats it's currently in (like JSON, Parquet, a Database), transform it, and expose it for ad-hoc querying.
- **Spark MLlib:**
  - MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives.
- **Spark GraphX:**
  - GraphX is the new (alpha) Spark API for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing the Resilient Distributed Property Graph: a directed multi-graph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and aggregateMessages) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.

Languages

- APIs in Java, Scala and Python
- Interactive shells in Scala and Python

Key Idea

- **Work with distributed collections as you would with local ones**
- Concept: resilient distributed datasets (RDDs)
  - Immutable collections of objects spread across a cluster
  - Built through parallel transformations (map, filter, etc)
  - Automatically rebuilt on failure
  - Controllable persistence (e.g. caching in RAM)

Operations

- Transformations (e.g. map, filter, groupBy, join)
  - Lazy operations to build RDDs from other RDDs
- Actions (e.g. count, collect, save)
  - Return a result or write it to storage

Example: Mining Console Logs

- Load error messages from a log into memory, then interactively search for patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split('\t')[2])
messages.cache()

messages.filter(lambda s: "foo" in s).count()
messages.filter(lambda s: "bar" in s).count()
. . .
```

Result: scaled to 1 TB data in 5-7 sec (vs 170 sec for on-disk data)

RDD Fault Tolerance

RDDs track the transformations used to build them (their *lineage*) to re compute lost data

E.g:

```
messages = textFile(...).filter(lambda s: s.contains("ERROR"))
                        .map(lambda s: s.split('\t')[2])
```

Spark in Java and Scala

Java API:

```
JavaRDD<String> lines =
spark.textFile(…);

errors = lines.filter(
  new Function<String, Boolean>() {
    public Boolean call(String s) {
      return s.contains("ERROR");
    }
});

errors.count()
```

Scala API:

```
val lines = spark.textFile(…)

errors = lines.filter(s =>
s.contains("ERROR"))
// can also write
filter(_.contains("ERROR"))

errors.count
```


Scala Cheat Sheet

Variables:

```
var x: Int = 7
var x = 7     // type inferred

val y = "hi"  // read-only
```

Functions:

```
def square(x: Int): Int = x*x

def square(x: Int): Int = {
  x*x   // last line returned
}
```

Collections and closures:

```
val nums = Array(1, 2, 3)

nums.map((x: Int) => x + 2) // => Array(3, 4, 5)

nums.map(x => x + 2)  // => same
nums.map(_ + 2)       // => same

nums.reduce((x, y) => x + y) // => 6
nums.reduce(_ + _)           // => 6
```

More details: scala-lang.org

Creating RDDs

```
# Turn a local collection into an RDD
sc.parallelize([1, 2, 3])

# Load text file from local FS, HDFS, or S3
sc.textFile("file.txt")
sc.textFile("directory/*.txt")
sc.textFile("hdfs://namenode:9000/path/file")

# Use any existing Hadoop InputFormat
sc.hadoopFile(keyClass, valClass, inputFmt, conf)
```


Basic Transformations

```
nums = sc.parallelize([1, 2, 3])

# Pass each element through a function
squares = nums.map(lambda x: x*x)   # => {1, 4, 9}

# Keep elements passing a predicate
even = squares.filter(lambda x: x % 2 == 0) # => {4}

# Map each element to zero or more others
nums.flatMap(lambda x: range(0, x))  # => {0, 0, 1, 0, 1, 2}
```

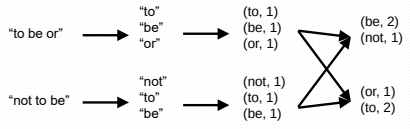
Range object (sequence of numbers 0, 1, …, x-1)

Basic Actions

```
nums = sc.parallelize([1, 2, 3])

# Retrieve RDD contents as a local collection
nums.collect() # => [1, 2, 3]

# Return first K elements
nums.take(2)   # => [1, 2]

# Count number of elements
nums.count()   # => 3

# Merge elements with an associative function
nums.reduce(lambda x, y: x + y)  # => 6

# Write elements to a text file
nums.saveAsTextFile("hdfs://file.txt")
```

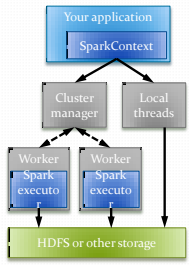

Example: Word Count

```
lines = sc.textFile("hamlet.txt")
counts = lines.flatMap(lambda line: line.split(" ")) \
              .map(lambda word: (word, 1)) \
              .reduceByKey(lambda x, y: x + y)
```



Software Components

- Spark runs as a library in your program
- Runs tasks locally or on a cluster
- Accesses storage via Hadoop InputFormat API (use HBase, HDFS, S3, …)



Create a SparkContext

Scala

```
import spark.SparkContext
import spark.SparkContext._

val sc = new SparkContext("masterUrl", "name", "sparkHome",
Seq("app.jar"))
```

Java

```
import spark.api.java.JavaSparkContext;

JavaSparkContext sc = new JavaSparkContext(
    "masterUrl", "name", "sparkHome", new String[] {"app.jar"}));
```

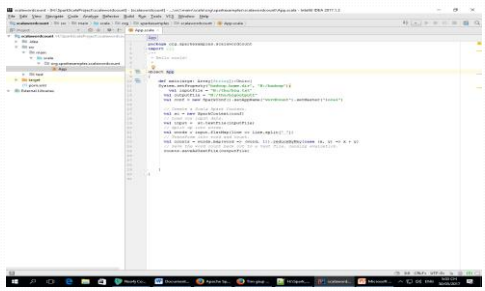
Python

```
from pyspark import SparkContext

sc = SparkContext("masterUrl", "name", "sparkHome", ["library.py"]))
```

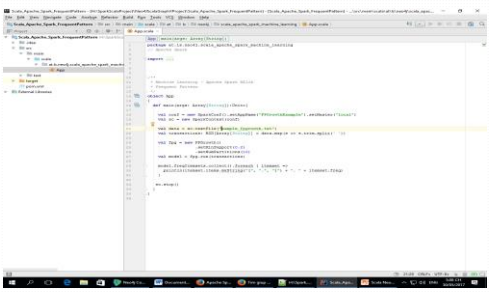
Cluster URL, or local / local[N] | App name | Spark install path on cluster | List of JARs with app code (to ship)

Examples of Scala program with GraphX and MLLib

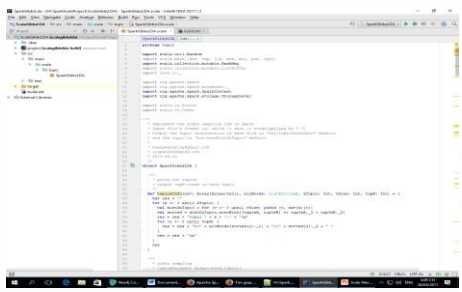
Scala word count



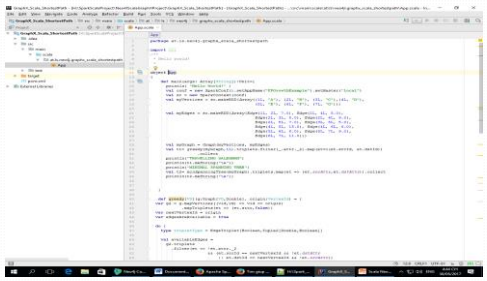
Scala Spark MLLib



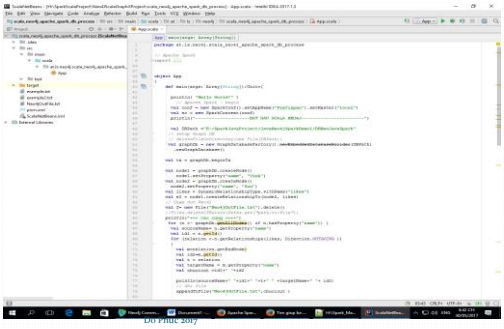
Spark Gibbs LDA, topic modeling



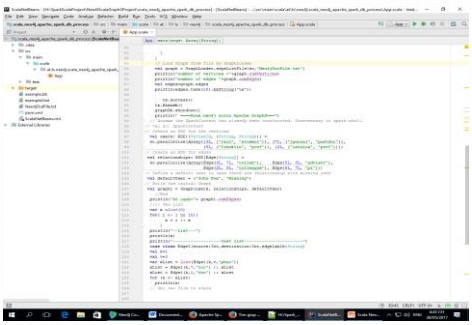
Scala GraphX



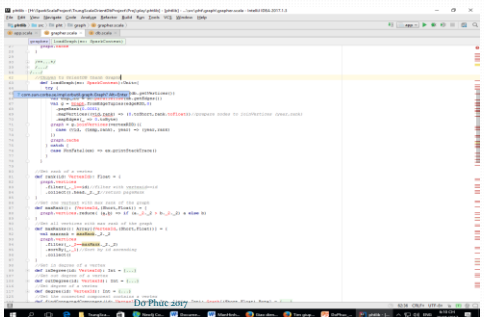
Scala Neo4j GraphX



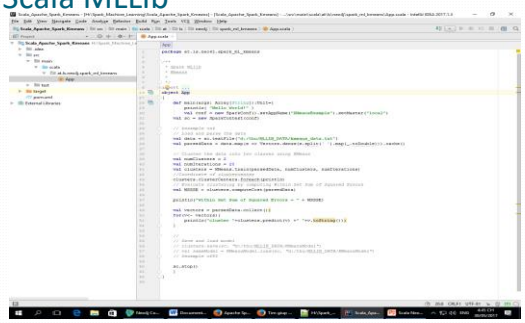
Scala Neo4j GraphX



OrientDB-> graphX

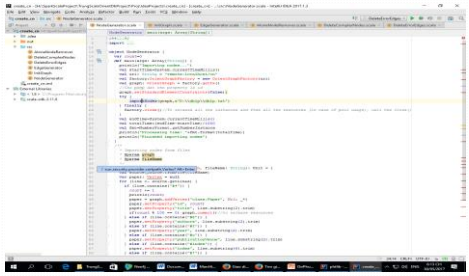


Scala MLLib

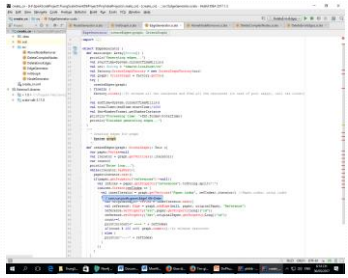


Create graph db

Create vertices from text file



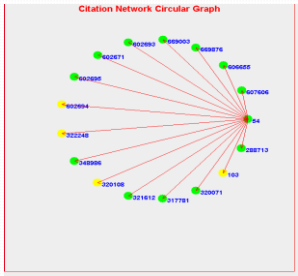
Create edges from text file



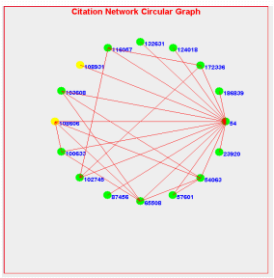
On-processing project

- Find the influence of paper in citation network
- Network: 3 millions vertices, 8 millions edges
- Environment:
- Graph database OrientDB: store Citation network
- Scala, Apache spark, Graph X

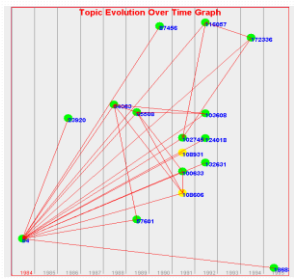
Citation network, Citing papers

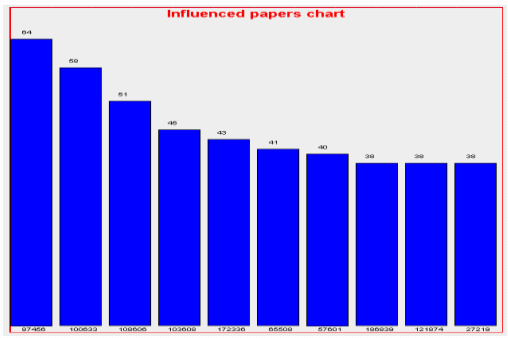


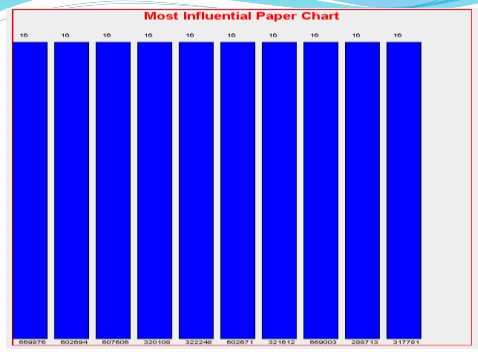
Citation network, cited papers



Topic evolution chart



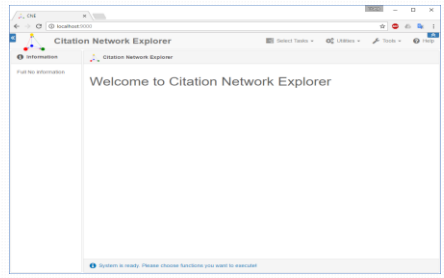




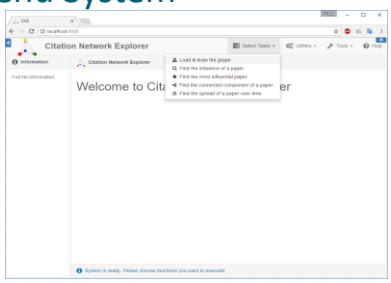
Challenge Problems

- Big data (volume, velocity, variety) for citation network-> Graph database
- Complicated algorithm on graph processing-> parallel graph processing on Apache Spark, GraphX with Scala.

Some results

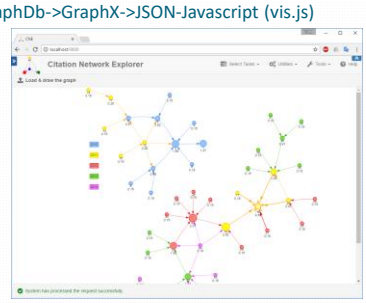


Menu System

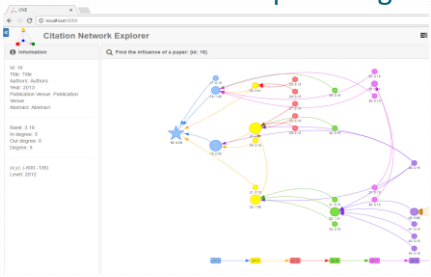


Load and visualize graph

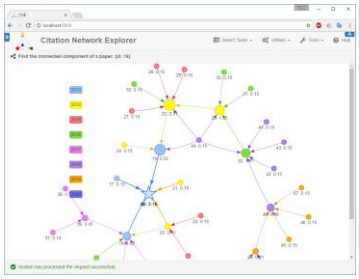
Read graphDb->GraphX->JSON-Javascript (vis.js)



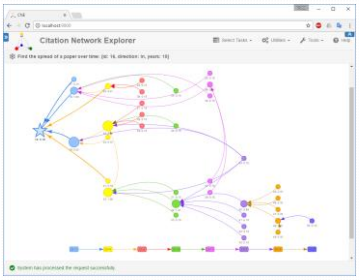
Find the influence of paper.
Scala->OrinentDB->GraphX->Page Rank



Find connected component
Scala -> GraphX



Find the spread of a paper over time



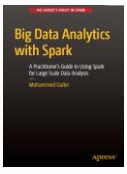
Conclusion

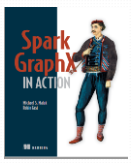
- Data science: turn data into data products
- Data Scientist is the sexiest job of 21st century
- 5 Vs of Big Data
- Skills Acquired for Data Science Students: Programming Skill, Statistics,Machine Learning, Data Mining,Domain Expertise, Analytics System
- BSc and MSc in Data Science
- Parallel/ Distributed Programming: Hadoop, Map Reduce, Apache Spark, Scala, Java, Python

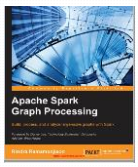
References

1. Ching-Yung Lin, Big Data Analytics Lecture notes, IBM Watson Research Center , 2014
2. Daisy Zhe Wang, Data Science, University of Florida, CISE Department, 2013
3. Matei Zaharia, Parallel Programming With Spark UC Berkeley, 2013
1. Martin Odersky, Scala by Example, EPFL, Switzerland, 2014
2. Michael S. Malak, Robin East, GraphX in Action, Manning 2016
3. Mohammed Guller, Big data Analytics with Spark, Apress, 2015
4. Ho Tu Bao, Data Science.Mini-course on Data Science, HCMC, 2017

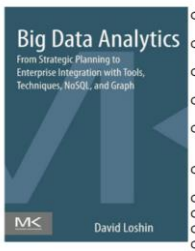
References







Reading Reference for Lecture 1



Chapter 1: Market and Business Drivers for Big Data Analysis
Chapter 2: Business Problems Suited to Big Data Analytics
Chapter 3: Achieving Organizational Alignment for Big Data Analytics
Chapter 4: Developing a Strategy for Integrating Big Data Analytics into the Enterprise
Chapter 5: Data Governance for Big Data Analytics: Considerations for Data Policies and Processes
Chapter 6: Introduction to High-Performance Appliances for Big Data Management
Chapter 7: Big Data Tools and Techniques
Chapter 8: Developing Big Data Applications
Chapter 9: NoSQL Data Management for Big Data
Chapter 10: Using Graph Analytics for Big Data
Chapter 11: Developing the Big Data Roadmap

19 E6893 Big Data Analytics – Lecture 1: Overview

Thank you for your attention!