INTRODUCTION TO SPARK

Theresa Csar

DBAI Research Seminar, October 12th, 2017

HISTORY

GraphX
APACHE Spark™
hadoop Map Reduce
Apache Hive
HIVE
APACHE HBASE
cassandra
Apache Hadoop Ecosystem
hadoop HDFS

MAPREDUCE - FRUITCOUNT

Input: „Apple, Pear, Kiwi, Pear“

1. Map to key-value pairs: (Apple,1), (Pear, 1), (Kiwi, 1), (Pear, 1)

2. Shuffle: (Apple,1)   (Pear, 1), (Pear, 1)   (Kiwi, 1)

3. Reduce (sum):   (Apple, 1)   (Pear, 2)   (Kiwi,1)

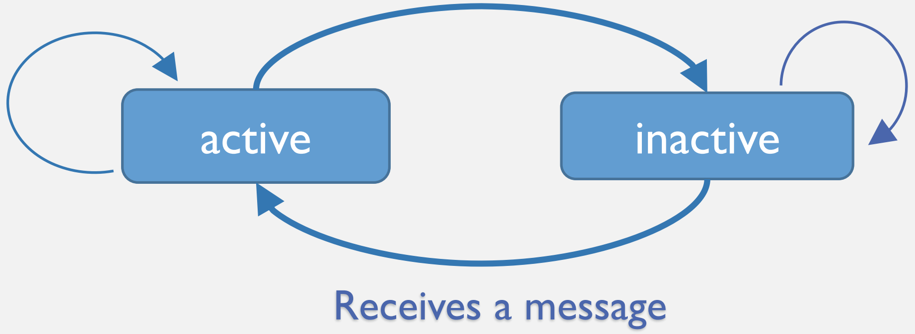
MAPREDUCE -> SPARK

Spark is the answer to Hadoop Mapreduces Disadvantages

- Slow
- Batch-processing
- Lots of reads and writes to the file system

PREGEL COMPUTATION – THINK LIKE A VERTEX

- vertices send messages to each other (along edges)
- In each superstep the vertex executes a vertex program on tthe received messages
- The state of a vertex is set to „inactive“ if it does not receive a message, or if it votes to halt
- The computation stops when all vertices are inactive



APACHE SPARK

- Runs both locally or distributed on a cluster
- Gains a lot of speed in comparison to traditional mapreduce/hadoop by perfoming computations in memory.
  - Key concept: Resilient Distributed Datasets (RDDs) and lazy evaluation

RDDS - RESILIENT DISTRIBUTED DATASETS

- Spark‘s core abstraction for working with data
- Immutable distributed collection of objects
  (split into multiple partitions)

- Three possible operations in Spark (→ lazy evaluation)
  - **Create** a new RDD
  - **Transform** an exisiting RDD
  - **Action**: call an operation on RDDs to compute a result

RDDS – OPERATIONS / LAZY EVALUATION

- Creating: load a dataset, or distribute a collection of objects (parallelize())

- Transformations: for example filtering creates a new RDD
  - are computed only on action

- Actions: calculated right away and return a result or save it to a storage

CREATE RDDS

- Parallelize existing collection of object
  - Usually not practicable since it requries you to have the whole dataset in memory on one machine
- Read from Files in a storage (SparkContext.textFile())

RDDS – PERSIST()

- RDDs are by default recomputed each time you run an action
- If you want to run multiple queries on the same dataset use persist() to keep the RDD in memory or on disk

RDDS - BASIC TRANSFORMATIONS

rdd =	{1,2,2,3}
• rdd.map(x => x*x)	{1,4,4,9}
• rdd.flatMap(x => x.to(3))	{1,2,3,2,3,2,3,3}
• rdd.filter(x => x!=2)	{1,3}
• rdd.distinct()	{1,3}

rdd.groupBy(), rdd.orderBy(), rdd.union(other), rdd.intersection(other), rdd.subtract(other), rdd.cartesian(other)

RDDS - ACTIONS

rdd = {1,2,2,3}

val sum = rdd.reduce((x,y) => x+y)

Similar actions: aggregate(), fold()

RDD – KEY VALUES

- rdd.groupByKey()
- rdd.reduceByKey()

EXAMPLE 1

- www.dbai.tuwien.ac.at/staff/csar/spark
- Create a useraccount at databricks
- Import notebook to workspace

SPARK – OTHER DATATYPES THAN RDDS

- Dataframes (Spark 1.6)
  - Immutable distributed collection of data
  - Organized into named columns
  - Untyped Rows -> Does not support compile time type safety
- Datasets (Spark 1.6)
  - Typed Rows → Supports compile time type safety

→ RDD, Dataframes and Datasets are slowly merging into one datatype: DataSet

https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html

SPARK PACKAGES

- Machine Learning: Mlib
- Analytics: SparkR
- Spark Streaming
- GraphX
- Many more: https://spark.apache.org/third-party-projects.html

SPARK SQL

- Only works on relational data (dataframes or datasets).
- SparkSQL can connect to many different Database systems (Hbase, Hive, Cassandra, …)
- SparkSQL always returns DataFrames

- Spark SQL Language Manual: https://docs.databricks.com/spark/latest/spark-sql/index.html#spark-sql-language-manual

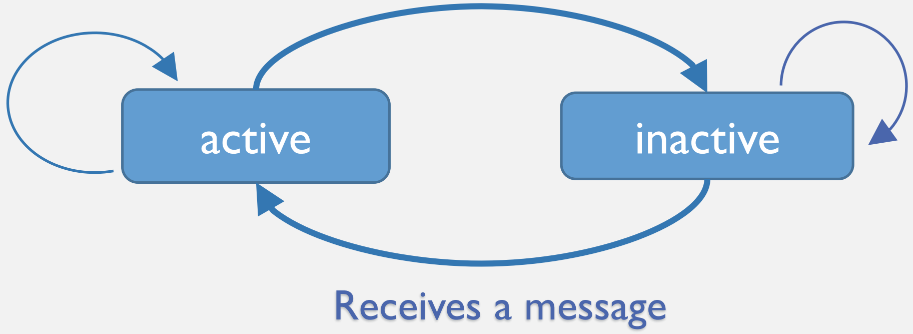
SPARK SQL – CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [db_name.]table_name
    [(col_name1 col_type1. ..)]
    USING datasource
    [OPTIONS (key1=val1, key2=val2, ...)]
    [PARTITIONED BY (col_name1, col_name2, ...)]
    [CLUSTERED BY (col_name3, col_name4, ...) INTO num_buckets BUCKETS]
    [LOCATION path]
    [TBLPROPERTIES (key1=val1, key2=val2, ...)]
    [AS select_statement]
```

EXAMPLE 2

PREGEL COMPUTATION – THINK LIKE A VERTEX

- vertices send messages to each other (along edges)
- In each superstep the vertex executes a vertex program on tthe received messages
- The state of a vertex is set to „inactive“ if it does not receive a message, or if it votes to halt
- The computation stops when all vertices are inactive



GRAPHX

GraphX is built on top of spark

- extends the Resilient Distributed Dataset by the Resilient Distributed Property Graph
- fundamental graph operation
- collection of graph algorithms (page rank, triangle counting, …)
- Pregel API

GRAPHX

- Graph[VD, ED] = Graph(vertices, edges)
- vertices: RDD[(VertexId, VD)]
  Each vertex has a VertexID and a value of type VD
- edges: RDD[Edge[ED]]
  Each edge connects two vertices (src and dst VertexIDs) and has an edge attribute of type ED

EXAMPLE 3

RECENT DEVELOPMENTS

- The concept of DataFrames and are an extension to RDDs
- GraphFrames is a new alternative to GraphX and is based on DataFrames (where GraphX was based on RDDs)

REFERENCES (PAPERS)

- GraphX: Graph Processing in a Distributed Dataflow Framework, Gonzalez et al, OSDI '14
- Pregel: A System for Large-Scale Graph Processing, Malewicz et al., SIGMOD'10
- MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean and Sanjay Ghemawat, in Proc. 6th USENIX Symp. on Operating Syst. Design and Impl., 2004

REFERENCES (BOOKS)

- Hadoop: The Definitive Guide 4th Edition, Tom White, O’Reilly Media, April 2015
- Learning Spark, Lightning-Fast Big Data Analysis, Matei Zaharia et al., O’Reilly Media, Mai 2015
- High Performance Spark, Best Practices for Scaling and Optimizing Apache Spark, Holden Karau and Rachel Warren, O’Reilly Media, June 2017

REFERENCES (LINKS)

- https://hadoop.apache.org/
- https://spark.apache.org/
- https://spark.apache.org/graphx/
- https://community.cloud.databricks.com/
- https://docs.databricks.com/spark/latest/spark-sql/index.html#spark-sql-language-manual

WHERE TO GO FROM HERE?

- Get your own local installation of spark
- Use a virtual machine:
  - https://de.hortonworks.com/products/sandbox/
  - https://www.cloudera.com/downloads/quickstart_vms/5-12.html
- Rent a cluster:
  - https://aws.amazon.com/de/ec2/?nc2=h_m1
  - https://cloud.google.com/compute/
  - https://azure.microsoft.com/de-de/
- (in my opinion) best point to start programming your own scala code on a spark cluster: https://de.hortonworks.com/tutorial/setting-up-a-spark-development-environment-with-scala/
- Tutorials by Databricks, Cloudera, Hortonworks

THANK YOU FOR YOUR ATTENTION!