ons/logo$_r cc.png$ icons/spark.png

Introduction to Spark

Igor Yakushin
ivy2@uchicago.edu

December 27, 2019

Introduction

- Apache Spark is a fast and general-purpose cluster computing system.
- It provides high-level APIs in Java, Scala, Python and R
- Typically all the functionality is available in Scala and Java while APIs in other languages might be behind. For example, Spark's GraphX library is only available in Scala.
- It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.
- Spark can run within Hadoop cluster and knows how to deal with various Hadoop components (like HDFS, Hive, HBase, etc) and data formats.
- Spark can also run on a standalone computer or a regular HPC cluster like midway.
- Spark can utilize multiple nodes and multiple CPU cores in a node.

Introduction

- Python API, pyspark, can be used
  - in batch mode
  - interactively
    - in a python command prompt
    - in Jupyter notebook
- In this tutorial we shall use only Python API to Spark mostly on midway cluster without Hadoop.
- The simplest way to install pyspark is to use pip install pyspark.
- While MLlib - Spark's machine learning library - has some Neural Network routines, for more advanced Deep Learning framework consider using Spark in combination with BigDL library from Intel that knows how to work with Spark's RDDs and can also be installed with pip install bigdl.

Introduction

- Spark was initially started by Matei Zaharia at UC Berkeley's AMPLab in 2009, and open sourced in 2010 under a BSD license.
- In 2013, the project was donated to the Apache Software Foundation and switched its license to Apache 2.0.
- Michael Franklin, who co-founded and directed AMPLab when Spark was created, is now professor and chair at the Computer Science Department of the University of Chicago.
- The current version of Spark is 2.3.0.
- We are using version 2.3.0 on midway and 2.2.0 on Hadoop cluster.

Introduction

- Before Spark 2.0, the main abstraction of Spark was the Resilient Distributed Dataset - RDD
- After Spark 2.0, RDD is replaced by DataFrame
- RDDs are still supported
- It is recommended now to use DataFrames instead of RDDs:
    - provides SQL interface to data - more convenient to program
    - much faster since a query optimization, similar the one used for SQL in RDBM, is applied to queries on DataFrames but not on RDDs
- We shall cover both since Spark is still in the state of transition. For example:
    - There are two streaming libraries:
        - Structured Streaming - based on DataFrames,
        - DStreams - based on RDDs
    - There are two APIs to machine learning library:
        - the old one based on RDD is in a maintenance state - only bug fixes are applied to it but no new features are introduced;
        - the new machine learning library based on DataFrames is still catching up with the functionality of the old library