twiml

AI Enterprise Workflow Study Group

Course 5, Week 1

5/2/2020

Agenda

- Check in
- Discussion
- Next steps

Course & Study Group Schedule

AI Enterprise Workflow Study Group		
Session	**Topic**	**Date**
Overview Webinar	Webinar with instructor, Ray Lopez	15-Feb
Course 1 Week 1	Course intro	22-Feb
Course 1 Week 2	Data ingestion, cleaning, parsing, assembly	29-Feb
Course 2 Week 1	Exploratory data analysis & visualization	7-Mar
Course 2 Week 2	Estimation and NHT	14-Mar
Course 3 Week 1	Data transformation and feature engineering	21-Mar
Course 3 Week 2	Pattern recognition and data mining best practices	28-Mar
Course 4 Week 1	Model evaluation and performance metrics	18-Apr
Course 4 Week 2	Building machine learning and deep learning models	25-Apr
Course 5 Week 1	Deploying models	2-May
Course 5 Week 2	Deploying models using Spark	9-May
Course 6 Week 1	Feedback loops and monitoring	16-May
Course 6 Week 2	Hands on with OpenScale and Kubernetes	23-May
Course 6 Week 3	Captsone project week 1	30-May
Course 6 Week 4	Captsone project week 2	6-Jun

Course 5 Week 1 learning objectives

1. Employ Apache Spark's RDDs, dataframes, and a pipeline
2. Employ spark-submit scripts to interface with Spark environments
3. Deploy a Flask app with Docker to deploy the ML model
4. Deploy a machine learning model from Watson Studio to Watson Machine Learning

Model Productionalization

- Make it Work
- Make it Better
- Make it Faster

Code Optimization

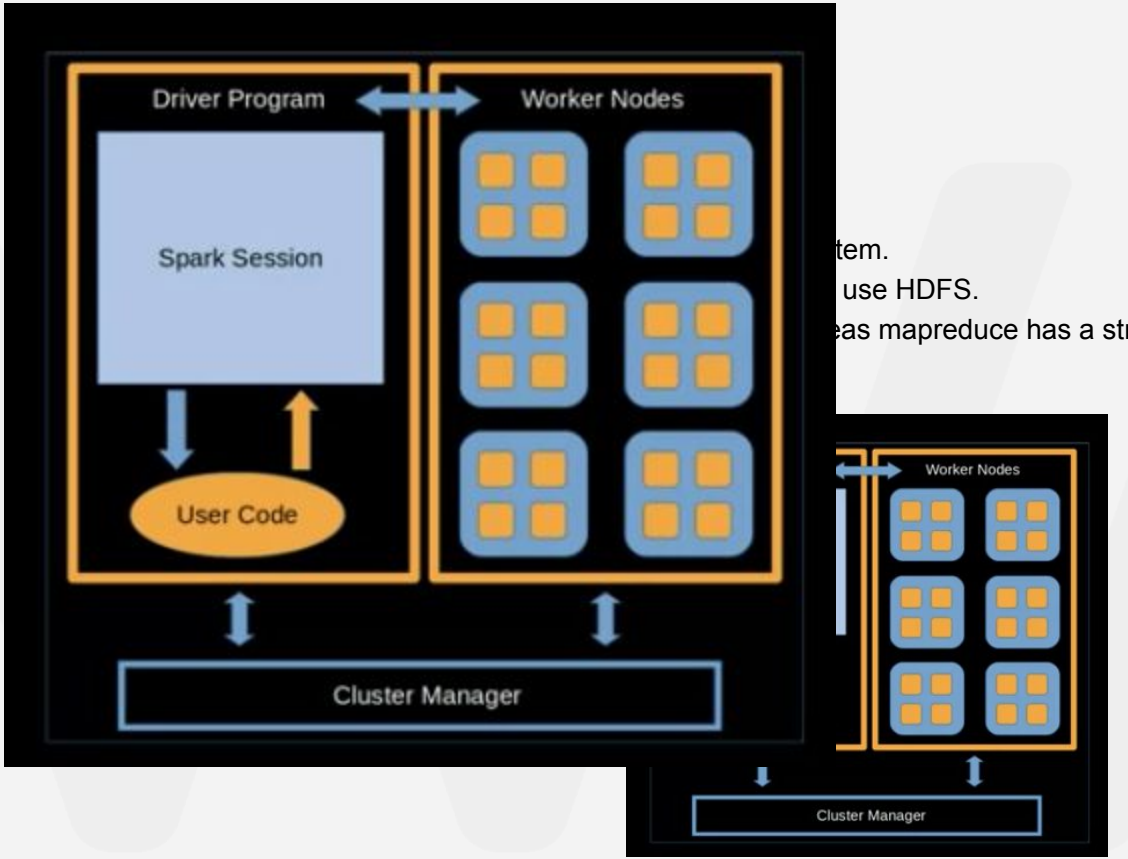
- **Use appropriate data containers** - For example, a Python set has a shorter look-up time than a Python list. Similarly, use dictionaries and NumPy arrays whenever possible.
- **Multiprocessing** - This is a package in the standard Python library and it supports spawning processes (for each core) using an API similar to the threading module. The multiprocessing package offers both local and remote concurrency.
- **Threading** - Another package in the standard library that allows separate flows of execution at a lower level than multiprocessing.
- **Subprocessing** - A module that allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. You may run **and control** non-Python processes like Bash or R with the subprocessing module.
- **mpi4py** - MPI for Python provides bindings of the Message Passing Interface (MPI) standard for the Python programming language, allowing any Python program to exploit multiple processors.
- **ipyparallel** - Parallel computing tools for use with Jupyter notebooks and IPython. Can be used with mpi4py.
- **Cython** - An optimizing static compiler for both the Python programming language and the extended Cython programming language. It is generally used to write C extensions for slow portions of code.
- **CUDA (Compute Unified Device Architecture)** - Parallel computing platform and API created by Nvidia for use with CUDA-enabled GPUs. CUDA in the Python environment is often run using the package PyCUDA.

Amdahl's & Gustafson’s Laws & Scale

The theoretical speedup of a program as a function of the number of processors executing it is limited by the serial part of the program.

Examples of Scale

- Does my service train in a reasonable amount of time given a lot more data?
- Does my service predict in a reasonable amount of time given a lot more data?
- Is my service ready to support additional request load?



tem.

use HDFS.

eas mapreduce has a strictly disk-based processing approach.

Spark RDDs

RDD - Resilient Distributed Dataset

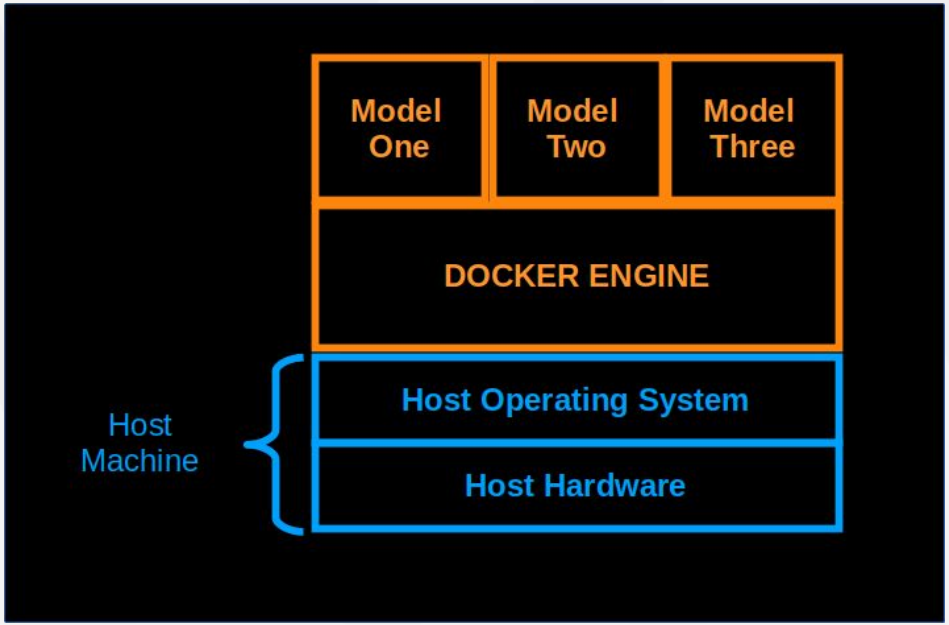
RDD is a full tolerant collection of elements that can be operated on in parallel. The RDD API uses two types of operations, transformations and actions. On top of Spark's RDD API, higher level APIs are provided including the data frame API and the machine learning API. Higher level API’s including data frame and ML APIs are provided.

Spark Workflow

1. Create environment to run spark from python
2. Extract RDDs or DataFrames from files
3. Carry out transformations
4. Execute actions to obtain values (local objects in python)

Note, spark uses lazy evaluation via DAG

Docker


Model One
Model Two
Model Three
DOCKER ENGINE
Host Operating System
Host Hardware
Host Machine

Additional Discussion

What did you learn?

What stumbling blocks did you run into?

How do these lessons relate to your experience?

What did you learn/find interesting in this week's lesson?

What are you doing as homework?

What interesting resources have you found?

Other?