ECE4721 SU 24 Lab 4 Spark and Drill

> Contributor: ECE4721J 24SU Teaching Group

Target of the lab sessions:

For all groups, you are expected to deploy **a cluster over all computers in your group**, instead of a pseudo-distributed one (multi-nodes on one computer).

Each group should hand in a report showing

- Ex.2: commands and results for the queries
- Ex.3: your code and results

Part I. Spark

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in `Java`, `Scala`, `Python` and `R`, and an optimized engine that supports general execution graphs. 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.

1. Install spark

Spark offical website: http://spark.apache.org/downloads.html

Since we have already installed hadoop, we can install "Spark with user provided Apache Hadoop". For example, `spark-2.4.3-bin-without-hadoop.tgz` (https://spark.apache.org/docs/2.4.3/).

The version we used last year is `spark-3.2.4-bin-without-hadoop`. Make sure you have the same spark version over the whole group. Old archives: https://archive.apache.org/dist/spark/.

After you download and extract the spark, set the PATH:

```
export SPARK_HOME=<extracted_spark_package>
export PATH=$PATH:$SPARK_HOME/bin
```

You can launch spark by the pyspark shell:

```
hadoopuser@hadoop-slave-3:~$ pyspark
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
```

```
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.4
      /_/

Using Python version 3.10.12 (main, Nov 20 2023 15:14:05)
Spark context Web UI available at http://localhost:4040
Spark context available as 'sc' (master = local[*], app id = local-
1718468198259).
SparkSession available as 'spark'.
>>>
```

As you can see, the web ui is available at the port 4040. You may also invoke spark in python interactively:

```
hadoopuser@hadoop-master:~$ python3
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from pyspark import SparkContext
>>> sc = SparkContext()
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
>>>
```

The variable `PYTHONPATH` may need to be configured, see below.

2. RDD

Every Spark application consists of a driver program that runs the user's main function and executes various parallel operations on a cluster. The main abstraction Spark provides is a `resilient distributed dataset (RDD)`, which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel.

RDDs are created by starting with a file in the HDFS (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it. Users may also ask Spark to persist an RDD in memory.

RDD can be manipulated using Spark shell, SparkR, pyspark, scala and java APIs, etc.

RDD programming guide: https://spark.apache.org/docs/2.4.3/rdd-programming-guide.html

[!Important] Two types of operations on an RDD:

- **Transformation**: Create a new dataset from an existing one; Only compute the result when an action is run; Do not return any result to the driver program
- **Action**: Run a computation on a dataset; Return the value to the driver program

Simple examples in pyspark:

```
>>> data = [1, 2, 3, 4, 5]
>>> distData = sc.parallelize(data)
>>> distData = distData.map(lambda x: 2*x)
>>> dataList = distData.collect()
>>> dataList
[2, 4, 6, 8, 10]
>>> listSum = distData.reduce(lambda a, b: a + b)
>>> listSum
30
```

```
## Average number of each word
>>> data = {('apple', 2), ('pine', 1), ('pineapple', 2)}
>>> distData = sc.parallelize(data)
>>> total = distData.map(lambda (x, y): y).reduce(lambda x, y: x + y)  # total
number of words
>>> avg = total / distData.distinct().count()
```

2. Spark on cluster

Spark can run both by itself, or over several existing cluster managers. It currently provides several options for deployment:

- Standalone Deploy Mode: simplest way to deploy Spark - on a private cluster
- Apache Mesos
- Hadoop YARN
- Kubernetes (k8s)

General steps for spark on YARN:

- Find the YARN Master node (i.e. which runs the Resource Manager). The following steps are to be performed on the master node only.
- Download the Spark tgz package and extract it somewhere.
- Configure Hadoop Class Path: https://spark.apache.org/docs/2.4.3/hadoop-provided.html
- Define these environment variables (in `.bashrc`):

  ```
  # Spark variables
  export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
  export LD_LIBRARY_PATH=${HADOOP_HOME}/lib/native
  ```

- Run spark job using `spark-submit`.

  Java Example:

  ```
  spark-submit --class org.apache.spark.examples.SparkPi \
  --master yarn \
  --deploy-mode cluster \
  --driver-memory 1g \
  --executor-memory 1g \
  --executor-cores 1 \
  examples/jars/spark-examples*.jar 5
  ```

  Python Example:

```
spark-submit \
--master yarn \
--deploy-mode cluster \
--conf spark.pyspark.driver.python=python3 \
--conf spark.pyspark.python=python3 \
ex3.py -n <size>
```

If task succeeeded, something like this may appear in your output:

```
client token: N/A
diagnostics: N/A
ApplicationMaster host: hadoop-master
ApplicationMaster RPC port: 34143
queue: default
start time: 1623136363960
final status: SUCCEEDED
tracking URL: http://hadoop-
master:8088/proxy/application_1623135707044_0008/
user: xiejinglei
```

When running on YARN, 2 modes exist :

- `client` : runs the Driver on the client which submits the spark job. The driver runs in the client process, and the application master is only used for requesting resources from YARN.
- `cluster` : runs the Driver on a slave node. The Spark driver runs inside an application master process which is managed by YARN on the cluster, and the client can go away after initiating the application.

Here's what we had last year:

```
# Spark variables
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
export SPARK_HOME=/home/hadoopuser/spark-3.2.4-bin-without-hadoop
export PATH=$PATH:$SPARK_HOME/bin
export LD_LIBRARY_PATH=${HADOOP_HOME}/lib/native
export PYTHONPATH=/home/hadoopuser/spark-3.2.4-bin-without-
hadoop/python:/home/hadoopuser/spark-3.2.4-bin-without-hadoop/python/lib/py4j-
0.10.9.5-src.zip:$PYTHONPATH
```

Part II. Drill

Drill is an Apache open-source SQL query engine for Big Data exploration. Drill is designed from the ground up to support high-performance analysis on the semi-structured and rapidly evolving data coming from modern Big Data applications, while still providing the familiarity and ecosystem of ANSI SQL, the industry-standard query language.

Official tutorial for drill on Hadoop (with zookeeper): https://drill.apache.org/docs/installing-drill-on-the-cluster/. There's also one available drill package manual's server.

Windows Users

You can refer to https://drill.apache.org/docs/installing-drill-on-windows/. You can directly launch drill by

```
F:\apache-drill-1.18.0\bin>drill-embedded.bat

Apache Drill 1.18.0
"Think different, think Drill."
apache drill>
```

Simple examples:

```
apache drill> select columns[0] as name, columns[1] as id, columns[2] as grade
from dfs.`/home/xiejinglei/course/ve472/lab2/l4-names/grades.csv` limit 2;
+----------------+------------+-------+
|      name      |     id     | grade |
+----------------+------------+-------+
| Irvin Kitagawa | 4320355991 | 85    |
| Salina Bryant  | 7615527683 | 96    |
+----------------+------------+-------+
2 rows selected (0.361 seconds)
```

```
apache drill> select * from (select columns[0] as name, columns[1] as id,
columns[2] as grade from dfs.`/home/xiejinglei/course/ve472/lab2/l4-
names/grades.csv`) where name = 'Tad Wyze' order by grade desc;
+----------+------------+-------+
|   name   |     id     | grade |
+----------+------------+-------+
| Tad Wyze | 7345030868 | 93    |
| Tad Wyze | 7345030868 | 64    |
| Tad Wyze | 7345030868 | 60    |
| Tad Wyze | 7345030868 | 55    |
| Tad Wyze | 7345030868 | 52    |
| Tad Wyze | 7345030868 | 51    |
| Tad Wyze | 7345030868 | 38    |
| Tad Wyze | 7345030868 | 37    |
| Tad Wyze | 7345030868 | 37    |
| Tad Wyze | 7345030868 | 32    |
| Tad Wyze | 7345030868 | 29    |
| Tad Wyze | 7345030868 | 28    |
| Tad Wyze | 7345030868 | 22    |
| Tad Wyze | 7345030868 | 13    |
| Tad Wyze | 7345030868 | 13    |
| Tad Wyze | 7345030868 | 13    |
| Tad Wyze | 7345030868 | 0     |
+----------+------------+-------+
```

Drill can also deal with data in avro format, parquet format, etc. with a faster speed.

Lab Submissions

Please finish the ex2 and ex3 as depicted in the manual. Please hand in a pdf report (e.g. exported by markdown), together with your codes onto canvas. **Lab 4 is a group lab**.

Reference

[1] Jinglei Xie. VE472 l4 spark and drill Summer 2022.

[2] ECE4721 Lecture Slides.