Intro to Apache Spark

Installation and setup

- Download latest version from http://spark.apache.org/downloads.html by selecting the package type of "Pre-built for Hadoop 2.7 and later." The download is a compressed tarball, called spark-2.4.4-bin-hadoop2.7.tgz
- Create a top-level folder and unpack inside it as follows:

```
cd ~
mkdir spark-install
cd spark-install
tar xzvf ../Downloads/spark-2.4.4-bin-hadoop2.7.tgz
ln -s  spark-2.4.4-bin-hadoop2.7 spark
```

- Then add Spark programs to your path as follows:

```
echo "export PATH=~/spark-install/spark/bin:~/spark-
    install/spark/sbin/:$PATH" >> ~/.bashrc
source ~/.bashrc
```

- Test it by starting one of the shells. For example, try spark-shell (for Scala-based shell) or try pyspark for Python-based shell. Use Ctrl-d to terminate the shell.

First Example

- Examine the https://github.com/BoiseState/CS535-resources/tree/master/examples/Spark/simpleapp. To build the example, use the following commands (install maven if it is missing on your system):

```
mvn clean; mvn package
```

- Now, we submit the jar file to run on Spark as follows:

```
spark-submit --class "SimpleApp" --master local[4]
    target/simple-project-1.0.jar
```

- Note that we are running Spark locally on our system. That is denoted by the `local[4]` option. The number in square brackets it is the number of threads to use.

Operation of a Spark Application

- A Spark application consists of a driver program that launches various parallel operations on the cluster. It creates distributed datasets on the cluster, and then applies operations to them.
- Driver programs access Spark via a `SparkContext` object, which represents a connection to a computing cluster. We use this to create Resilient Distributed Datasets (RDDs) on which we can then apply various parallel operations.
- The driver program manages a number of nodes called executors. These are used to parallelize the operations.
- Spark programs can be written via interactive shells (for Scala and Python, for example) or they can be standalone (for Scala, Python, Java etc). For production environments, standalone would be the norm.