Supercomputing for Big Data ET4310
Assignment 1

Using Spark for In-Memory Computation

Spark is a fast and powerful big data engine for processing Hadoop data. It runs in Hadoop clusters through Hadoop YARN or in Spark's standalone mode, and it can process data in HDFS, HBase, Hive, and any Hadoop input format. It is designed to perform both general data processing (similar to MapReduce) and new workloads like streaming, interactive queries, and machine learning.

Introduction
In this assignment you'll learn about Spark, how to use it and the tools and facilities it provides for processing your big data analytics. You'll be introduced to a set of Spark commands and you'll experiment with the processing they are capable of. Finally you'll write and execute a twitter based stream processing application with Spark.

Spark installation
Spark is open source and publicly available for download from Spark Apache project site at spark.apache.org. It is possible to install Spark on a Win, Linux or Mac system, since it is written in Java and runs on a Java VM. We will use Spark version 1.5.0 for the exercises in this assignment. Installation of the pre-built packages on Linux, Mac and Windows is straightforward. However, in the lab we will assume access to a Linux environment. Please log into the Linux partition during lab hours.
In order to compile stand-alone Spark programs, you'll need to install the Scala Simple Build Tool or sbt from www.scala-sbt.org as well.
If you need help installing these programs, let us know.

Spark basics
The first exercise of the assignment is to get you acquainted with Spark and run some simple tasks. Visit the page (spark.apache.org/documentation.html) and watch the Screencast 3 and 4 of the "Screencast Tutorial Videos" under the Videos section.
Run the assignments in the videos as they are described in the Quick Start page of the Spark website (http://spark.apache.org/docs/latest/quick-start.html).

Exercise 1. Data exploration using spark (30% of the grade)

This exercise is to perform some data exploration of page view statistics for Wikimedia projects. The questions here are based on the Berkley Spark tutorial to be found at the following link. There, you can get good hints on how to solve these questions.
http://ampcamp.berkeley.edu/big-data-mini-course/data-exploration-using-spark.html

First, you'll need to download the page view statistics for 6pm on Dec 9, 2007 from
http://dumps.wikimedia.org/other/pagecounts-raw/2007/2007-12/pagecounts-20071209-180000.gz
each line, delimited by a space, contains stats for one Wikimedia page. The schema is:
<project_code> <page_title> <num_hits> <page_size>

Use this statistics file to find answers to the following questions.
1. First, launch the Spark shell and then create an RDD (Resilient Distributed Dataset) named pagecounts from the input files.

2. Let's take a peek at the data. Use the take() operation of an RDD to get the first K records, with K = 10. The take() operation returns an array and Scala simply prints the array with each element separated by a comma. This is not easy to read. Make the output prettier by traversing the array to print each record on its own line.

3. Check how many records in total are in this dataset.

4. Derive an RDD containing only English pages from pagecounts. This can be done by applying a filter function to pagecounts. For each record, we can split it by the field delimiter (i.e. a space) and get the first field and then compare it with the string "en". (experiment in this and following questions with caching the RDDs using the ".cache" action).

5. How many records are there for English pages?

6. Generate a histogram of total English page views on Wikimedia pages for the different titles included in the dataset. The high level idea of what you'll be doing is as follows. First, generate a key value pair for each line; the key is the page title (the second field), and the value is the number of pageviews for that page (the third field).

7. Find English pages that were viewed more than 1000 times in the dataset.

8. Write a short report describing your results.

Exercise 2. Spark Streaming (70% of the grade)

The task in this exercise is to write a standalone Spark streaming application with the following characteristics:

1. Continuously reads Twitter feed every second
2. Collects the feed in a sliding window of 60 seconds
3. Calculates the most 5 retweeted tweets within the 60 second window (see Tip c below)
4. Prints the top 5 tweets calculated every second for the sliding window
5. Write a short report describing your work

Tips and notes:

a. Accustom yourself first with Spark streaming using the example on https://databricks-training.s3.amazonaws.com/realtime-processing-with-spark-streaming.html

b. This exercise requires you to setup Twitter credentials. You can learn how to do that by following information on the link given in the tip example.

c. Note that only retweets that appear within the last minute have to be considered. So, for example, if a specific tweet has 1000 retweets overall, but only 75 retweets appear within the last minute, the number of retweets to be considered is 75.

d. Note that since we are using public Twitter stream, we are limited to 1% for the global traffic (this limits us to around 60 tweets/seconds). So do not be surprised in case you were expecting more tweets.

e. Twitter sets a tag that indicates if a tweet represents a retweet of an original tweet, and has a link to the original tweet in the retweet. Twitter also keeps the retweet count in the status of the original tweet (tracking of the number of retweets).