**Week 4 Assignment**

Guruprasad Velikadu Krishnamoorthy

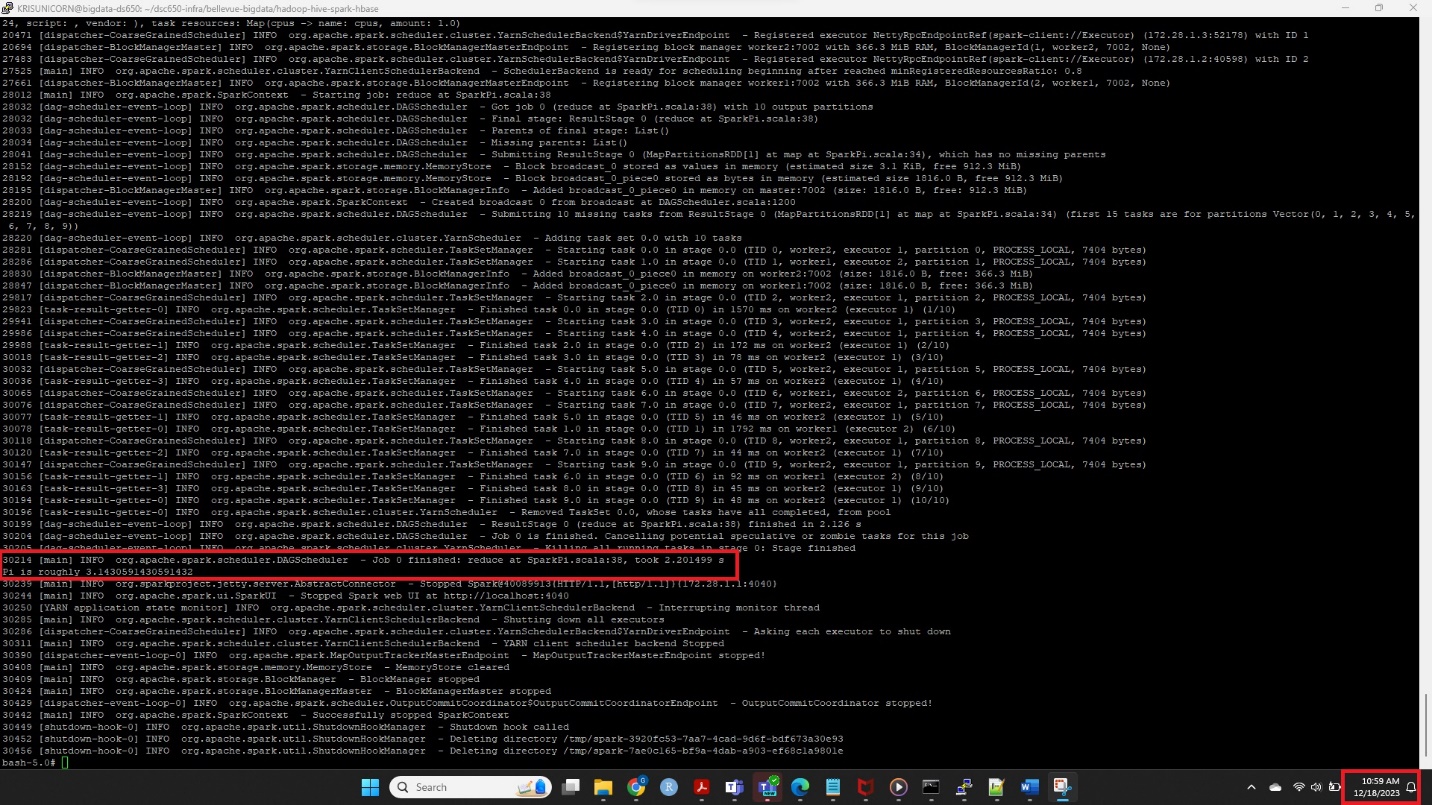
College of Science and Technology, Bellevue University

[**DSC650**-T301: Big](https://cyberactive.bellevue.edu/webapps/blackboard/execute/courseMain?course_id=_512542_1) Data

Professor. Nasheb Ismaily

December 18, 2023

1. **SparkPi Program:**
   1. **Screenshot of SparkPi program:**



* 1. **Output from Spark History Server:**

**A screenshot of a computer

Description automatically generated**

* 1. **Significance of the SparkPi program:**

In the Monte Carlo method for estimating pi, random points are generated within a square, and the number of points falling inside a quarter-circle inside that square is counted. The ratio of the points inside the circle to the total points is then used to estimate the pi, based on the formula of the area of the circle and square.

In SparkPi Program, x and y coordinates of the “n” random points are calculated, and the number of points where x\*2+y\*2 < 1 is calculated. Spark parallelize creates a RDD and distributes the calculation between the executors and returns the result. The Pi is then calculated by using the formula 4 \* count/ n.

Note, in our example the value of n passed is 10 and the program is executed in “client” mode with 2 GB of driver memory, 1 GB of executor memory, and 1 core. Yarn is specified as the Resource manager.

spark-submit --class org.apache.spark.examples.SparkPi \  
 --master yarn \  
 --deploy-mode client \  
 --driver-memory 2g \  
 --executor-memory 1g \  
 --executor-cores 1 \  
 $SPARK\_HOME/examples/jars/spark-examples\*.jar \  
 10

1. **Screenshot of the first 100 generated random numbers.**

**A screenshot of a computer

Description automatically generated**

**A computer screen with a black background

Description automatically generated**

1. **Screenshots of Custom Transformation and their significance:**

Due to space constraints while taking the screenshot of the output, only 10 rows are displayed for each transformation. 7 transformations were used, and their significance is listed below:

* 1. **toUpperCase**:

val transformedSentences = sentencesRDD.map(sentence => sentence.**toUpperCase**)

**Significance**:

This command will convert all the characters to Upper case and return the Uppercase converted values of the RDD. Similarly, **toLowerCase** can also be used.

* 1. **startsWith with Map method:**

val transformedSentences2 = sentencesRDD.**map**(sentence => sentence.startsWith("appl"))

**Significance:**

This command returns a Boolean value if the Array starts with the specified value. In the example, all arrays that start with “appl” returns True and everything else returns False.

* 1. **startsWith used with filter method:**

val transformedSentences2 = sentencesRDD.**filter**(sentence => sentence.**startsWith**("appl"))

**Significance:**

This is similar to the command above; the difference is it applies the filters and returns the array elements that start with the specified value.

A screenshot of a computer

Description automatically generated

* 1. **Contains:**

val transformedSentences3 = sentencesRDD.filter(sentence => sentence.contains("erry"))

**Significance:**

This command returns the list of elements that contains the string **“erry”**

* 1. **substring:**

val transformedSentences4 = sentencesRDD.map(sentence => sentence.substring(0,5))

**Significance:**

This transformation returns the substring of the element from position 0 to 4.

* 1. **Replace:**

val transformedSentences5 = sentencesRDD.map(sentence => sentence.replace(" ","-"))

**Significance:**

This transformation replaces the spaces with “-“ in the Array of elements.

* 1. **indexOf**:

val transformedSentences6 = sentencesRDD.map(sentence => sentence.indexOf("fig"))

**Significance:**

This transformation returns the position of the string “fig” in the array. If the string does not exist, it returns -1.

A screenshot of a computer

Description automatically generated