INTRO TO SPARK ASSIGNMENT:

Task 1

Given a list of numbers - List[Int] (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

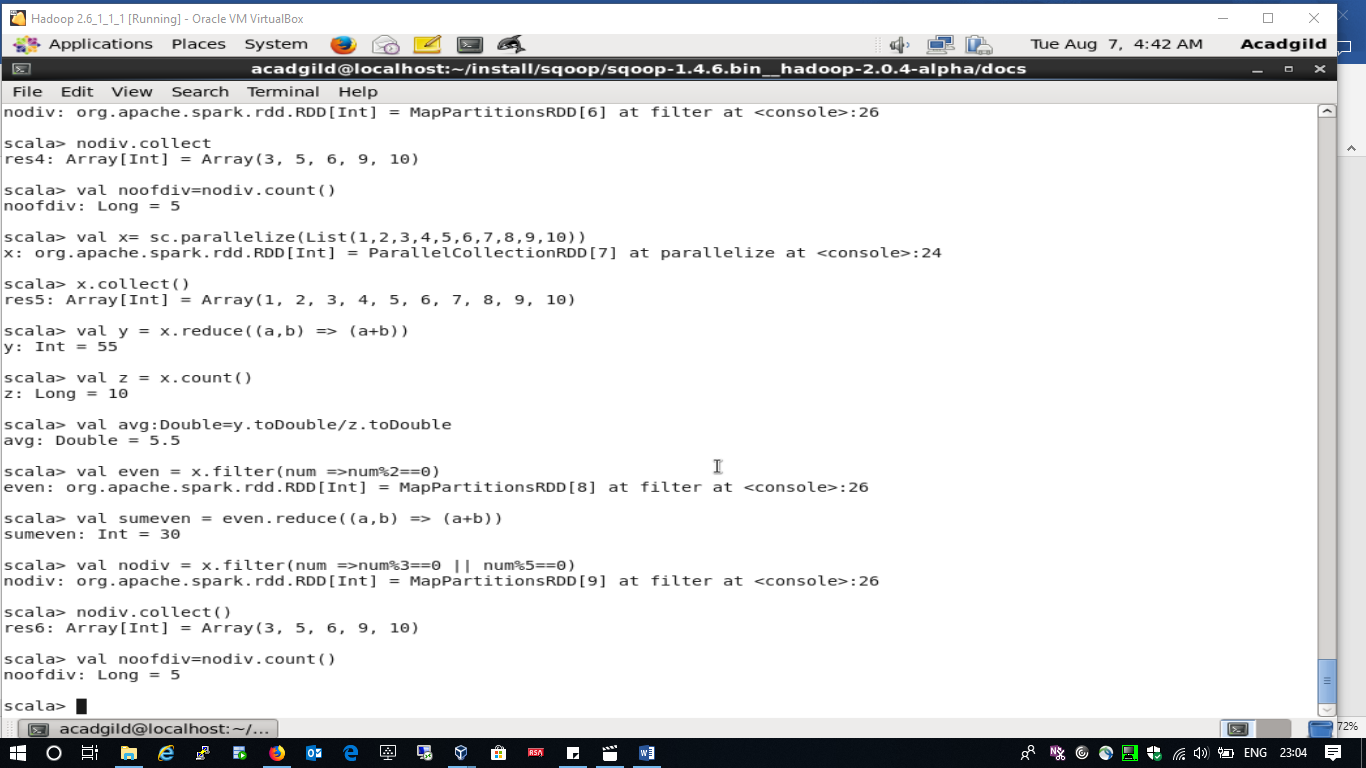
- find the sum of all numbers

- find the total elements in the list

- calculate the average of the numbers in the list

- find the sum of all the even numbers in the list

- find the total number of elements in the list divisible by both 5 and 3



Task 2

1. Pen down the limitations of MapReduce.

Spark is lightning fast **cluster** computing tool. Apache Spark runs applications up to 100x faster in memory and 10x faster on disk than Hadoop. Because of reducing the number of read/write cycle to disk and storing intermediate data in-memory Spark makes it possible.

Spark is easy to program as it has tons of high-level operators with **RDD – Resilient Distributed Dataset where as** In MapReduce, developers need to hand code each and every operation which makes it very difficult to work

Spark is capable of performing batch, interactive and Machine Learning and Streaming all in the same cluster. As a result, makes it a complete **data analytics** engine. Thus, no need to manage different component for each. but, As MapReduce only provides the batch engine. Hence, we are dependent on different engines. For example- Storm, Giraph, Impala, etc. for other requirements. So, it is very difficult to manage many components.

MapReduce fails when it comes to real-time data processing as it was designed to perform batch processing on voluminous amounts of data.

MapReduce is a high latency computing framework

MapReduce doesn’t have an interactive mode.

you can only process data in batch mode not in streaming.

1. What is RDD? Explain few features of RDD?

**RDD** stands for “**Resilient Distributed Dataset”**. It is the fundamental data structure of Apache Spark. RDD in Apache Spark is an immutable collection of objects which computes on the different node of the cluster.

Decomposing the name RDD:

* **Resilient**, i.e. fault-tolerant with the help of RDD lineage graph(DAG) and so able to recompute missing or damaged partitions due to node failures.
* **Distributed**,since Data resides on multiple nodes.
* **Dataset**represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or database via JDBC with no specific data structure.

Hence, each and every dataset in RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster. RDDs are fault tolerant i.e. It posses self-recovery in the case of failure.

1. In-memory computation

### 2. Lazy Evaluation

### 3. Fault Tolerance

### 4. Immutability

### 5. Persistence

### 6. Partitioning

### 7. Parallel

### 8. Location-Stickiness

### 9. Coarse-grained Operation

### 10. Typed

### 11. No limitation

3) List down few Spark RDD operations and explain each of them.

**Map:**  
Map will take each row as input and return an RDD for the row.

**flatMap** will take an iterable data as input and returns the RDD as the contents of the iterator.

**Filter:  
filter** returns an RDD which meets the filter condition. Below is the sample demonstration of the above scenario.

**reduceByKey**takes a pair of key and value pairs and combines all the values for each unique key. Below is the sample demonstration of the above scenario

## **Transformations**:

Any function that returns an RDD is a transformation, elaborating it further we can say that Transformation is functions which create a new data set from an existing one by passing each data set element through a function and returns a new RDD representing the results.

**Collect:**  
collect is used to return all the elements in the RDD.

**Count:  
count**is used to return the number of elements in the RDD.

**countByValue**is used to count the number of occurrences of the elements in the RDD

**Take:  
take**will display the number of records we explicitly specify.

**Reduce:** will gives the reduced set or value by operation given in reduce.