Apache Spark Interview Questions

mindmajix.com/apache-spark-interview-questions

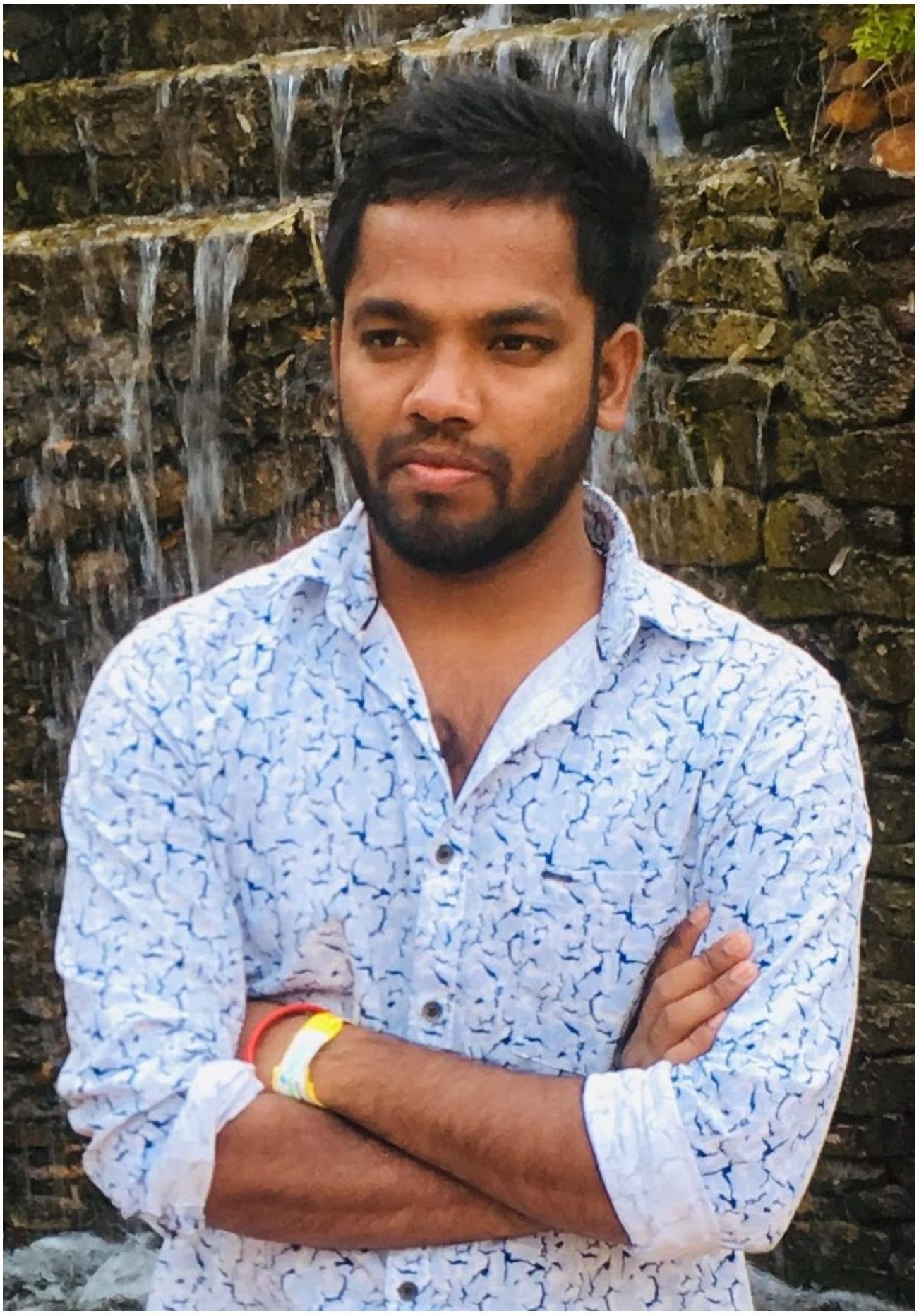
May 5, 2017

(5.0) | 8068 Ratings

About Author

Vinod M CONTENT LEAD AT MINDMAJIX

Vinod M is a Big data expert writer at Mindmajix and contributes in-depth articles on various Big Data Technologies. He also has experience in writing for Docker, Hadoop, Microservices, Commvault, and few BI tools. You can be in touch with him via LinkedIn and Twitter.



If you're looking for Apache Spark Interview Questions for Experienced or Freshers, you are at right place. There are a lot of opportunities from many reputed companies in the world. According to research Apache Spark has a market share of about 4.9%. So, You still have an opportunity to move ahead in your career in Apache Spark Development. Mindmajix offers Advanced Apache Spark Interview Questions 2018 that helps you in cracking your interview & acquire dream career as Apache Spark Developer.

Top 40 Apache Spark Interview Questions

Q1) Apache Spark Vs Hadoop

Spark Vs Hadoop		
Features	**Spark**	**Hadoop**
Data processing	Part of hadoop, hence batch processing	Batch Processing even for high volumes
Streaming Engine	Apache spark straming - micro batches	Map-Reduce
Data Flow	Direct Acyclic Graph-DAG	Map-Reduce
Computation Model	Collect and process	Map-Reduce batch oriented model
Performance	Slow due to batch processing	Slow due to batch processing
Memory Management	Automatic memory management in latest release	Dynamic and static - Configurable
Fault Tolerance	Recovery available without extra code	Highly fault tolerant due to Map-Reduce
Scalability	Highly scalable - sSpark Cluster(8000 Nodes)	Highly scalable - Produces large number of nodes

Q2) What is Spark?

Spark is a parallel data processing framework. It allows to develop fast, unified big data application combine batch, streaming and interactive analytics.

Q3) Why Spark?

Spark is the third generation distributed data processing platform. It’s unified bigdata solution for all bigdata processing problems such as batch , interacting, streaming processing.So it can ease many bigdata problems.

Q4) What is RDD?

Spark’s primary core abstraction is called Resilient Distributed Datasets. RDD is a collection of partitioned data that satisfies these properties. Immutable, distributed, lazily evaluated, catchable are common RDD properties.

Q5) What is Immutable?

Once created and assign a value, it’s not possible to change, this property is called Immutability. Spark is by default immutable, it does not allow updates and modifications. Please note data collection is not immutable, but data value is immutable.

Q6) What is Distributed?

RDD can automatically the data is distributed across different parallel computing nodes.

Q7) What is Lazy evaluated?

If you execute a bunch of programs, it’s not mandatory to evaluate immediately. Especially in Transformations, this Laziness is a trigger.

Q8) What is Catchable?

Keep all the data in-memory for computation, rather than going to the disk. So Spark can catch the data 100 times faster than Hadoop.

Q9) What is Spark engine responsibility?

Spark responsible for scheduling, distributing, and monitoring the application across the cluster.

Q10) What are common Spark Ecosystems?

- Spark SQL(Shark) for SQL developers,
- Spark Streaming for streaming data,
- MLLib for machine learning algorithms,
- GraphX for Graph computation,
- SparkR to run R on Spark engine,
- BlinkDB enabling interactive queries over massive data are common Spark ecosystems. GraphX, SparkR, and BlinkDB are in the incubation stage.

Q11) What is Partitions?

Partition is a logical division of the data, this idea derived from Map-reduce (split). Logical data specifically derived to process the data. Small chunks of data also it can support scalability and speed up the process. Input data, intermediate data, and output data everything is Partitioned RDD.

Learn Spark vs Hadoop What's Better to Learn First.

Q12) How spark partition the data?

Spark use map-reduce API to do the partition the data. In Input format we can create number of partitions. By default HDFS block size is partition size (for best performance), but its’ possible to change partition size like Split.

Q13) How Spark store the data?

Spark is a processing engine, there is no storage engine. It can retrieve data from any storage engine like HDFS, S3 and other data resources.

Q14) Is it mandatory to start Hadoop to run spark application?

No not mandatory, but there is no separate storage in Spark, so it use local file system to store the data. You can load data from local system and process it, Hadoop or HDFS is not mandatory to run spark application.

Q15) What is SparkContext?

When a programmer creates a RDDs, SparkContext connect to the Spark cluster to create a new SparkContext object. SparkContext tell spark how to access the cluster. SparkConf is key factor to create programmer application.

Q16) What is SparkCore functionalities?

SparkCore is a base engine of apache spark framework. Memory management, fault tolarance, scheduling and monitoring jobs, interacting with store systems are primary functionalities of Spark.

Q17) How SparkSQL is different from HQL and SQL?

SparkSQL is a special component on the sparkCore engine that support SQL and HiveQueryLanguage without changing any syntax. It’s possible to join SQL table and HQL table.

Q18) When did we use Spark Streaming?

Spark Streaming is a real time processing of streaming data API. Spark streaming gather streaming data from different resources like web server log files, social media data, stock market data or Hadoop ecosystems like Flume, and Kafka.

Q19) How Spark Streaming API works?

Programmer set a specific time in the configuration, within this time how much data gets into the Spark, that data separates as a batch. The input stream (DStream) goes into spark streaming. Framework breaks up into small chunks called batches, then feeds into the spark engine for processing. Spark Streaming API passes that batches to the core engine. Core engine can generate the final results in the form of streaming batches. The output also in the form of batches. It can allows streaming data and batch data for processing.

Q20) What is Spark MLlib?

Mahout is a machine learning library for Hadoop, similarly MLlib is a Spark library. MetLib provides different algorithms, that algorithms scale out on the cluster for data processing. Most of the data scientists use this MLlib library.

Q21) What is GraphX?

GraphX is a Spark API for manipulating Graphs and collections. It unifies ETL, other analysis, and iterative graph computation. It’s fastest graph system, provides fault tolerance and ease of use without special skills.

Q22) What is File System API?

FS API can read data from different storage devices like HDFS, S3 or local FileSystem. Spark uses FS API to read data from different storage engines.

Q23) Why Partitions are immutable?

Every transformation generates new partition. Partitions use HDFS API so that partition is immutable, distributed and fault tolerance. Partition also aware of data locality.

Q24) What is Transformation in spark?

Spark provides two special operations on RDDs called transformations and Actions. Transformation follows lazy operation and temporary hold the data until unless called the Action. Each transformation generates/return new RDD. Example of transformations: Map, flatMap, groupByKey, reduceByKey, filter, co-group, join, sortByKey, Union, distinct, sample are common spark transformations.

Q25) What is Action in Spark?

Actions are RDD’s operation, that value returns back to the spar driver programs, which kick off a job to execute on a cluster. Transformation’s output is an input of Actions. reduce, collect, takeSample, take, first, saveAsTextfile, saveAsSequenceFile, countByKey, foreach are common actions in Apache spark.

Q26) What is RDD Lineage?

Lineage is an RDD process to reconstruct lost partitions. Spark not replicate the data in memory, if data lost, Rdd use linege to rebuild lost data.Each RDD remembers how the RDD build from other datasets.

Q27) What is Map and flatMap in Spark?

The map is a specific line or row to process that data. In FlatMap each input item can be mapped to multiple output items (so the function should return a Seq rather than a single item). So most frequently used to return Array elements.

Q28) What are broadcast variables?

Broadcast variables let programmer keep a read-only variable cached on each machine, rather than shipping a copy of it with tasks. Spark supports 2 types of shared variables called broadcast variables (like Hadoop distributed cache) and accumulators (like Hadoop counters). Broadcast variables stored as Array Buffers, which sends read-only values to work nodes.

Q29) What are Accumulators in Spark?

Spark of-line debuggers called accumulators. Spark accumulators are similar to Hadoop counters, to count the number of events and what’s happening during job you can use accumulators. Only the driver program can read an accumulator value, not the tasks.

Q30) How RDD persist the data?

There are two methods to persist the data, such as persist() to persist permanently and cache() to persist temporarily in the memory. Different storage level options there such as MEMORY_ONLY, MEMORY_AND_DISK, DISK_ONLY and many more. Both persist() and cache() uses different options depends on the task.

Q31) When do you use apache spark? OR What are the benefits of Spark over Mapreduce?

- Spark is really fast. As per their claims, it runs programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. It aptly utilizes RAM to produce the faster results.
- In map reduce paradigm, you write many Map-reduce tasks and then tie these tasks together using Oozie/shell script. This mechanism is very time consuming and the map-reduce task has heavy latency.
- And quite often, translating the output out of one MR job into the input of another MR job might require writing another code because Oozie may not suffice.
- In Spark, you can basically do everything using single application/console (pyspark or scala console) and get the results immediately. Switching between 'Running something on cluster' and 'doing something locally' is fairly easy and straightforward. This also leads to less context switch of the developer and more productivity.
- Spark kind of equals to MapReduce and Oozie put together.

Q32) Is there is a point of learning MapReduce, then?

Yes. For the following reason:

- MapReduce is a paradigm used by many big data tools including Spark. So, understanding the MapReduce paradigm and how to convert a problem into series of MR tasks is very important.
- When the data grows beyond what can fit into the memory on your cluster, the Hadoop Map-Reduce paradigm is still very relevant.
- Almost, every other tool such as Hive or Pig converts its query into MapReduce phases. If you understand the Mapreduce then you will be able to optimize your queries better.

Q33) When running Spark on Yarn, do I need to install Spark on all nodes of Yarn Cluster?

Apache Spark Certification Training!

Explore Curriculum
Since spark runs on top of Yarn, it utilizes yarn for the execution of its commands over the cluster's nodes.
So, you just have to install Spark on one node.

Check Out Apache Spark Tutorials

Q34) What are the downsides of Spark?

Spark utilizes the memory. The developer has to be careful. A casual developer might make following mistakes:

- She may end up running everything on the local node instead of distributing work over to the cluster.
- She might hit some webservice too many times by the way of using multiple clusters.

The first problem is well tackled by Hadoop Map reduce paradigm as it ensures that the data your code is churning is fairly small a point of time thus you can make a mistake of trying to handle whole data on a single node.
The second mistake is possible in Map-Reduce too. While writing Map-Reduce, user may hit a service from inside of map() or reduce() too many times. This overloading of service is also possible while using Spark.

Q35) What is an RDD?

The full form of RDD is resilience distributed dataset. It is a representation of data located on a network which is

- Immutable – You can operate on the rdd to produce another rdd but you can't alter it.
- Partitioned / Parallel – The data located on RDD is operated in parallel. Any operation on RDD is done using multiple nodes.
- Resilience – If one of the node hosting the partition fails, another nodes takes its data.

RDD provides two kinds of operations: Transformations and Actions.

Q36) What is Transformations?

The transformations are the functions that are applied on an RDD (resilient distributed data set). The transformation results in another RDD. A transformation is not executed until an action follows.
The example of transformations are:

1. map() – applies the function passed to it on each element of RDD resulting in a new RDD.
2. filter() – creates a new RDD by picking the elements from the current RDD which pass the function argument.

Q37) What are Actions?

An action brings back the data from the RDD to the local machine. Execution of an action results in all the previously created transformation. The example of actions are:

- reduce() – executes the function passed again and again until only one value is left. The function should take two argument and return one value.
- take() – take all the values back to the local node form RDD.

Q38) Say I have a huge list of numbers in RDD(say myrdd). And I wrote the following code to compute average:

def myAvg(x, y):
return (x+y)/2.0;
avg = myrdd.reduce(myAvg);

Q39) What is wrong with it? And How would you correct it?

The average function is not commutative and associative;
I would simply sum it and then divide by count.

```
1  def sum(x, y):
2  return  x+y;
3  total = myrdd.reduce(sum);
4  avg = total / myrdd.count();
```

The only problem with the above code is that the total might become very big thus over flow. So, I would rather divide each number by count and then sum in the following way.

```
1  cnt = myrdd.count();
2  def devideByCnd(x):
3  return  x/cnt;
4  myrdd1 = myrdd.map(devideByCnd);
5  avg = myrdd.reduce(sum);
```

Q40) Say I have a huge list of numbers in a file in HDFS. Each line has one number.And I want to compute the square root of sum of squares of these numbers. How would you do it?

We would first load the file as RDD from HDFS on spark

```
numsAsText = sc.textFile("hdfs://hadoop1.knowbigdata.com/user/student/sgiri/mynumbersfile.txt");
```

Define the function to compute the squaresdef toSqInt(str):

```
1  v = int(str);
2  return  v*v;
```

#Run the function on spark rdd as transformation

```
nums = numsAsText.map(toSqInt);
```

#Run the summation as reduce action

```
total = nums.reduce(sum)
```

#finally compute the square root. For which we need to import math.

```
import math;
print math.sqrt(total);
```

Q41) Is the following approach correct? Is the sqrtOfSumOfSq a valid reducer?

```
numsAsText
=sc.textFile(“hdfs:  //hadoop1.knowbigdata.com/user/student/sgiri/mynumbersfile.txt”);
def toInt(str):
return  int(str);
nums = numsAsText.map(toInt);
def sqrtOfSumOfSq(x, y):
return  math.sqrt(x*x+y*y);
total = nums.reduce(sum)
import math;
print math.sqrt(total);
```

A: Yes. The approach is correct and **sqrtOfSumOfSq** is a valid reducer.

Q42) Could you compare the pros and cons of the your approach (in Question 2 above) and my approach (in Question 3 above)?

You are doing the square and square root as part of reduce action while I am squaring in map() and summing in reduce in my approach.
My approach will be faster because in your case the reducer code is heavy as it is calling math.sqrt() and reducer code is generally executed approximately n-1 times the spark RDD.
The only downside of my approach is that there is a huge chance of integer overflow because I am computing the sum of squares as part of map.

Q43) If you have to compute the total counts of each of the unique words on spark, how would you go about it?

#This will load the bigtextfile.txt as RDD in the spark lines =

```
sc.textFile(“hdfs://hadoop1.knowbigdata.com/user/student/sgiri/bigtextfile.txt”);
```

#define a function that can break each line into words

```
def toWords(line):
return  line.split();
```

Run the toWords function on each element of RDD on spark as flatMap transformation.
We are going to flatMap instead of map because our function is returning multiple values.

```
words = lines.flatMap(toWords);
```

Convert each word into (key, value) pair. Her key will be the word itself and value will be 1.

```
def toTuple(word):
return  (word, 1);
wordsTuple = words.map(toTuple);
```

Now we can easily do the reduceByKey() action.

```
def sum(x, y):
return  x+y;
counts = wordsTuple.reduceByKey(sum)
```

Now, print

```
counts.collect()
```

Q44) In a very huge text file, you want to just check if a particular keyword exists. How would you do this using Spark?

```
lines = sc.textFile(“hdfs:  //hadoop1.knowbigdata.com/user/student/sgiri/bigtextfile.txt”);
def isFound(line):
if  line.find(“mykeyword”) > -1:
return  1;
return  0;
foundBits = lines.map(isFound);
sum = foundBits.reduce(sum);
if  sum > 0:
print “FOUND”;
else  :
print “NOT FOUND”;
```

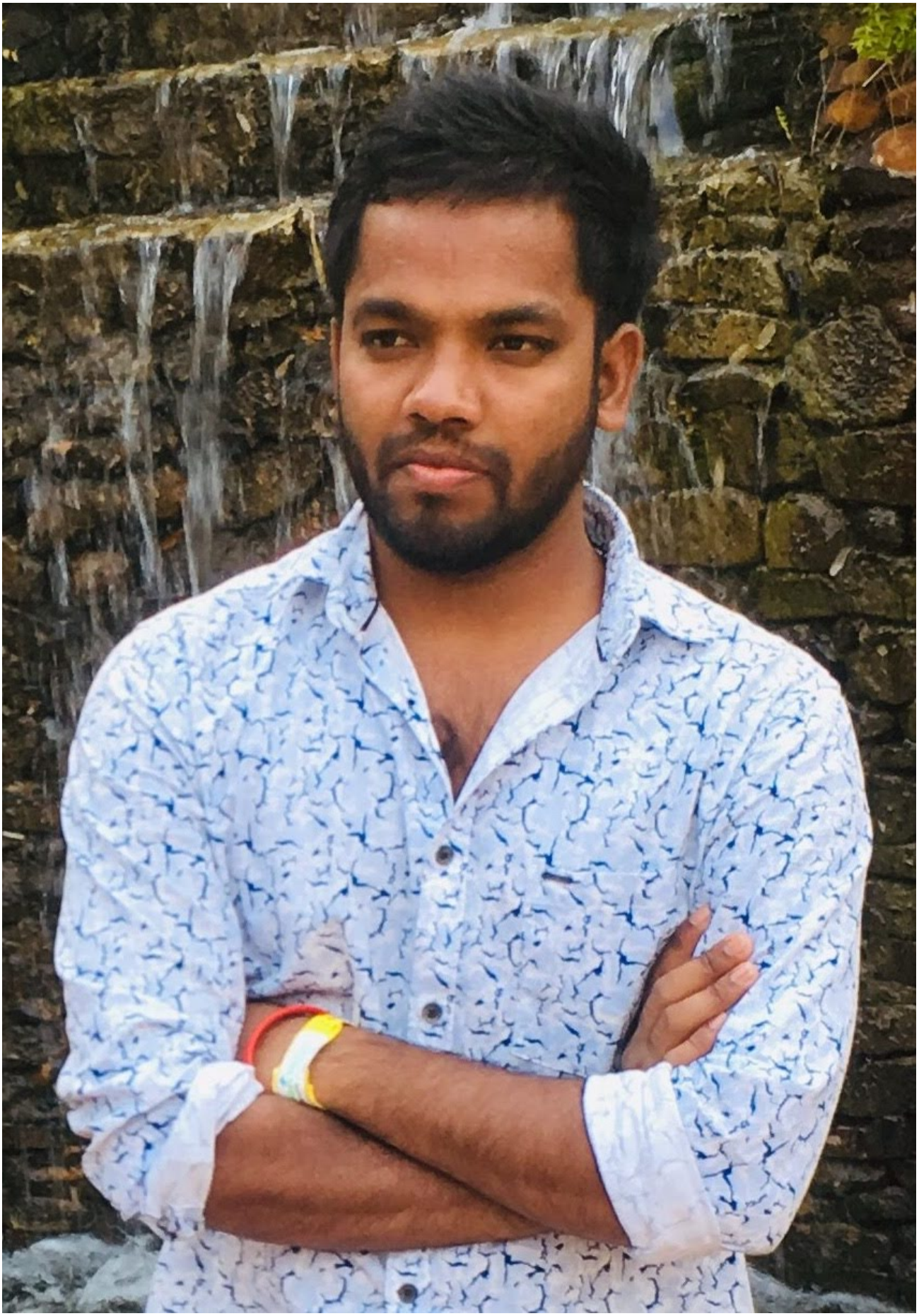
Q45) Can you improve the performance of this code in previous answer?

Yes. The search is not stopping even after the word we are looking for has been found. Our map code would keep executing on all the nodes which is very inefficient.
We could utilize accumulators to report whether the word has been found or not and then stop the job. Something on these line:
import thread, threading
from time import sleep

```
result = “Not Set”
lock = threading.Lock()
accum = sc.accumulator(0)
def map_func(line):
#introduce delay to emulate the slowness
sleep(1);
if  line.find(“Adventures”) > -1:
accum.add(1);
return  1;
return  0;
def start_job():
global result
try  :
sc.setJobGroup(“job_to_cancel”, “some description”)
lines =
sc.textFile(“hdfs:  //hadoop1.knowbigdata.com/user/student/sgiri/wordcount/input/big.txt”);
result = lines.map(map_func);
result.take(1);
except Exception as e:
result = “Cancelled”
lock.release()
def stop_job():
while  accum.value < 3 :
sleep(1);
sc.cancelJobGroup(“job_to_cancel”)
supress = lock.acquire()
supress = thread.start_new_thread(start_job, tuple())
supress = thread.start_new_thread(stop_job, tuple())
supress = lock.acquire()
[/tab]
```

How does the job support work?

* We see your project and technologies used, if we are 100% confident then we agree to support you.
* We work on the monthly basis
* No of hours of Support: Based on customer need and the pricing also varies
* We support you to solve your technical problem and guide you in the right direction.



By Vinod M

- 2019-05-31
- 80677

Call Our Advisor