**Syntel Outlook Draft**

**Git setup and pushing code to github**

1. Install git bash

2. Open git bash and go to the directory which need to be a local repository

3. run "git init"

4. copy all files which needs to be pushed into github

5. run "git add ."

6. run 'git commit -m "<message>"'

7. In github page, get the remote repository url by clicking on the "clone or download" button.

8. run "git remote add origin <remote url>"

9. run "ssh-keygen -t rsa" and provide the directory(or go ahead with default directory) and password phrase.

10. Open .ssh/id\_rsa.pub file and copy the public key

11. In github, click add new ssh key button and provide the copied public key

12. run "git push origin master"

Now the files will be moved to universal github repository.

**Spark setup in windows**

a. Set JAVA\_HOME

b. Install Scala and set SCALA\_HOME

c. Add these two env variables in path with their bin directory

d. Download Spark and extract its content in a directory. Create SPARK\_HOME env and set the same in path variable with bin directory.

e. To get Hadoop setup do the following

    1. Run cmd as administrator  
    2. Download winutils.exe binary from [https://github.com/steveloughran/winutils](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fgithub.com%2fsteveloughran%2fwinutils) repository (use hadoop-2.7.1 for Spark 2) - [https://github.com/steveloughran/winutils/blob/master/hadoop-2.7.1/bin/winutils.exe](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fgithub.com%2fsteveloughran%2fwinutils%2fblob%2fmaster%2fhadoop-2.7.1%2fbin%2fwinutils.exe)  
    3. Save winutils.exe binary to a directory of your choice, e.g. c:\hadoop\bin  
    4. Set HADOOP\_HOME to reflect the directory with winutils.exe (without bin), e.g. set HADOOP\_HOME=c:\hadoop  
    5. Set PATH environment variable to include %HADOOP\_HOME%\bin  
    6. Create c:\tmp\hive directory  
    7. Execute winutils.exe chmod -R 777 \tmp\hive  
    8. Open spark-shell and run spark.range(1).show to see a one-row dataset.

For reference -

[http://nishutayaltech.blogspot.in/2015/04/how-to-run-apache-spark-on-windows7-in.html](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fnishutayaltech.blogspot.in%2f2015%2f04%2fhow-to-run-apache-spark-on-windows7-in.html)

[https://stackoverflow.com/questions/43808523/spark-shell-error-on-windows-can-it-be-ignored-if-not-using-hadoop](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fstackoverflow.com%2fquestions%2f43808523%2fspark-shell-error-on-windows-can-it-be-ignored-if-not-using-hadoop)

**Spark – other info**

Parallel programming with spark - [https://www.youtube.com/watch?v=7k4yDKBYOcw](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3d7k4yDKBYOcw)

Analyzing BigData with Twitter: Spark  - [https://www.youtube.com/watch?v=rpXxsp1vSEs](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3drpXxsp1vSEs)

**Spark setup -**

Insall spark locally on windows -

[https://www.youtube.com/watch?v=VwiGHUKAHWM&index=6&list=PL7pSMJuOh5SjFytYLqiRJ9D29CYDXBrCM](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dVwiGHUKAHWM%26index%3d6%26list%3dPL7pSMJuOh5SjFytYLqiRJ9D29CYDXBrCM)

[https://www.youtube.com/watch?v=KvQto\_b3sqw&list=PL7pSMJuOh5SjFytYLqiRJ9D29CYDXBrCM&index=5](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dKvQto_b3sqw%26list%3dPL7pSMJuOh5SjFytYLqiRJ9D29CYDXBrCM%26index%3d5)

How to deploy spark to Mesos -

<https://www.youtube.com/watch?v=iW25_dR_9EE&list=PL7pSMJuOh5SjFytYLqiRJ9D29CYDXBrCM&index=4>

Spark cluster on windows - [https://www.youtube.com/watch?v=bQuoDgz9UMw](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dbQuoDgz9UMw)

**List of installations**

1. Ubuntu -

a. Using Oracle VirtualBox - [http://www.psychocats.net/ubuntu/virtualbox](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fwww.psychocats.net%2fubuntu%2fvirtualbox)

b. Normal Ubuntu installation - [http://www.psychocats.net/ubuntu/installing](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fwww.psychocats.net%2fubuntu%2finstalling)

**Spark – Details**

[https://www.usenix.org/legacy/event/hotcloud10/tech/full\_papers/Zaharia.pdf](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.usenix.org%2flegacy%2fevent%2fhotcloud10%2ftech%2ffull_papers%2fZaharia.pdf)

calling scala class from Java -   
 [http://blog.muhuk.com/2016/05/24/how\_to\_call\_scala\_from\_java\_\_using\_scala\_classes.html#.WYfi0\_l96ig](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fblog.muhuk.com%2f2016%2f05%2f24%2fhow_to_call_scala_from_java__using_scala_classes.html%23.WYfi0_l96ig)  
 [http://ted-gao.blogspot.com/2011/09/mixing-scala-and-java-in-project.html](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fted-gao.blogspot.com%2f2011%2f09%2fmixing-scala-and-java-in-project.html)

Spark -

[https://spark.apache.org/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fspark.apache.org%2f)

[https://spark.apache.org/docs/latest/#launching-on-a-cluster](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fspark.apache.org%2fdocs%2flatest%2f%23launching-on-a-cluster)

[https://spark.apache.org/docs/latest/streaming-programming-guide.html](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fspark.apache.org%2fdocs%2flatest%2fstreaming-programming-guide.html)

Look out for all videos of datamantra - search for "spark datamantra" and view those videos.

Kafka -

[http://kafka.apache.org/documentation.html](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fkafka.apache.org%2fdocumentation.html)

Spark Architecture -

1. RDD (Resilient Distributed DataSet) - Resilient means fault tolerant.  
        RDD is the data set used by Spark with following characteristics

        a. immutable

            - immutable allows to store the data for long time  
            - Lazy transformation allows to recreate data on failure

        b. Lazy evaluation - means do the evaluation only when needed. Until then don't do it

                For eg. if we need to read a file, filter the lines which has "spark" keyword and then get the total count.

                Here reading the file will create an RDD to store the file data and then filter will create another RDD to store the filtered data. These two operation are part of transformations.

                Then getting the count out of filtered data is the action to be performed. This action will be performed on the RDD only when requested. Until then it will never get the count and store in an RDD.

        c. Cacheable

        d. Type inferred

RDD -

1. Spark is built around RDDs. you create, transform, analyze and store RDDs in a Spark program

2. The Dataset contains a collection of elements of any type

    a. Strings, Lines, rows, objects, collections

3. The Dataset can be partitioned and distributed across multiple nodes

4. RDDs are immutable. They cant be changed

5. They can be cached and persisted

6. Transformations act on RDDs to create a new RDD

7. Actions analyze RDDs to provide a result.

8. RDD is resilient, as they rely on lineage and whenever there is a failure in the system they can recompute using the prior info in lineage graph

Difference between RDD and Distributed Shared memory - a comparison

|  |  |  |
| --- | --- | --- |
|  | **RDD** | **DSM** |
| write operation | Coarse grained | Fine grained |
| Recovery | lineage graph | check point - |
| consistency | trivial - bcoz the data is immutable and so wont be up to date | up to the application. |

Apache Spark - Scala - 52 videos on spark and scala in youtube

Best Spark Video - But for 5.5 hrs

Advanced Apache Spark Training - Sameer Farooqui (Databricks) - [https://www.youtube.com/watch?v=7ooZ4S7Ay6Y&spfreload=10](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3d7ooZ4S7Ay6Y%26spfreload%3d10)

Using Apache Spark 2.0 to Analyze the City of San Francisco's Open Data - [https://www.youtube.com/watch?v=K14plpZgy\_c&list=PL-x35fyliRwheNb4GO7je57HTKv1NNCDo&index=1](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dK14plpZgy_c%26list%3dPL-x35fyliRwheNb4GO7je57HTKv1NNCDo%26index%3d1)

**Look out for Sameer Farooqui's video on spark and HDFS**

Longer videos on spark -

1. Understanding Apache Spark in Depth | Spark Explained | Apache Spark Tutorial | Edureka  
2. Spark Streaming | Spark Streaming Tutorial for Beginners | Real Time Processing | Edureka  
3. 5 Things One Must Know About Spark | Spark Tutorial | Spark Features | Edureka

**Learnt from Madhukara Phatak's spark videos**

**From "Evolution of Spark" video**

[http://spark.apache.org/news/spark-1-0-0-released.html](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fspark.apache.org%2fnews%2fspark-1-0-0-released.html)  
     [www.youtube.com/watch?v=ckX6fT3kYG0](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dckX6fT3kYG0)  
     [www.youtube.com/watch?v=iKOGBr-kOks](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fwww.youtube.com%2fwatch%3fv%3diKOGBr-kOks)  
     [www.youtube.com/watch?v=7nIMpD5TyNs](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fwww.youtube.com%2fwatch%3fv%3d7nIMpD5TyNs)  
     [www.youtube.com/watch?v=jErEhxP8LYQ](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fwww.youtube.com%2fwatch%3fv%3djErEhxP8LYQ)

Github link for examples - [https://github.com/phatak-dev/spark2.0-examples](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fgithub.com%2fphatak-dev%2fspark2.0-examples)

**Spark 2.0 -**

Spark 2.0 is going to deal with DataSet abstraction and has come up with libraries for DataSet and so RDD will slowly deprecated.

1. DataSet - strongly typed collection of domain specific objects that can be transformed in parallel using functional or relational operations. DataSet is relational data type representation unlike RDD which has key-value pair form.

2. DataFrame - Untyped view which is a DataSet of Row

3. RDD - represents an immutable, partitioned collection of elements that can be operated on in parallel

4. SparkSession API - Previously SparkContext is been used to create RDD, StreamingContext for DStream, etc. But from 2.0, these context will get deprecated and SparkSession is going to replace them. This SparkSession will help in creating DataSets. DataSet will be more like SQL APIin getting relational form data.

5. SparkSession.read API returns DataFrame object which is unstructured form. To make it domain specific object(nothing but DataSet), we need to specify the type on SparkSession.read API, like appending

".as[String]" to convert the format to String format.

6. DataFrame Vs. DataSet -

        a. DataSet has got  logicsal plan and optimizations of DataFrame

        b. DataFrame is schema less DataSet

        c. DataSet has additional step for serialization and checking for proper schema

        d. DataSet serialization is custom specific serialization. This serialization is different from spark's Java serialization or kryo serialization

        e. In DataFrame the query param passed for action will be validated during run time and so any error with the param will be thrown during runtime. But at the same time with DataSet, the error gets thrown at compile time itself.

7. Catalogue API -

        a. Brings support to manage external metastores from Spark

        b. Integrates well with Hive metastore

        c. Primarily used for DDL operations

        d. API is built on DataSet abstraction

8. Time analysis -

    Analysis on data based on time window. This time abstraction can be used in both batch and streaming operations. Currently this is available only for DataFrames and for DataSet the libraries are not available in 2.0.

9. Plugging Custom Optimizations -

    In Spark 2.0, one can inject custom optimisations without changing majority of the source code. For eg. if the operation is multiply the values by 1 and provide the result then spark will do the multiplication for all the values without knowing that multiplication by 1 is not going to change the value. Hence we can write a custom optimization logic to skip the multiplication here. This will help us in skipping the serialization, action and deserialization logic and reduce the processing time.

**Memory Management in Spark -**

Two usages of memory -

1. Execution - used for shuffles, joins, sorts, etc. Here once the computation is done and arrived at the output then the memory used can be flushed. Memory usage will be temporary.

2. Storage - used to cache the data. This storage will not be temporary and those data can be used for multiple computations.

The Heap memory of JVM is been used by Spark for its processing and storage. Spark 1.0 has the static memory allocation where in some percentage of the available RAM memory was distributed between execution and storage. Here if, either of the allocated memory is full then the disk storage will be used. In case of cache, it tries to clear the cache based on LRU algorithm. But here we could find a problem where in if storage memory is not full but execution memory is full. Even then execution will start using disk memory though there are enough in RAM which are part of storage.

From Spark 1.6, this memory allocation will happen dynamically. The same RAM memory will be used by both execution and storage based on the availability. Here, if the execution needs some more memory where in storage+execution memory is 100% then spark tries to clear the cache to get some memory for execution. But in case of other operations like machine learning, where in the caching is major, we can configure the storage memory size by specifying minimum unevictable amount of cached data(not a reservation but suggestion).

Not only these memory problem will arise between execution and storage. But also the same can happen with tasks running in parallel in execution or storage; and in single tasks handling multiple operations. In all these cases, spark does dynamic allocation and manages to distribute the memory for each task/operations instead of keeping them in wait state.so, the memory will get shared among all these entities.

So, the dealing is when memory contention arises force members to spill so that every member will get memory to do their activity.

Spark never uses the JVM way of allocating memory and uses tungsten format. Because JVM allocates 48 bytes memory to store a string "abcd", though it can be stored in 4 bytes. Tungsten uses different style in storing the data with less memory space which could be within 8 bytes. And in this case, tungsten method will store the data in a serialized manner if the data is an array. Hence we do not need serialization and deserialization of JVM process.

Off heap - Here the memory allocation will happen in the area other than JVM's heap. Here, we use Unsafe API of Java to take care of memory and so the storage also will be done by us and no involvement of JVM.

**Machine Learning in Spark** -

1. You can find the code in github - search for introduction\_to\_ml\_with\_spark

2. Breeze - library in scala for numerical computation. This library internally uses fortran for executing the calculation. Spark MLlib uses Breeze internally for vector manipulation. Also many data structures of Spark MLlib uses Breeze data structures.

3. In machine learning, the data are been converted to RDD[Vectors]. And so any operation which happens on the data will be performed on Vectors which has the same data. But to speed up the process, when the data is huge then the input data will be split into partitions and each partition data will be constructed as matrix of RDD. Hence the mathematical calculation on the data will be performed on matrices. Performing maths calculation on matrix will be much faster as compared to performing on vectors. Because Vectors are single column matrices.

4. MLLib is the library in Spark which is been used for Machine Learning program implementation in Spark. It uses Breeze library of Scala internally.

5. Beyond MLLib -

        a. MLLib Pipelines API

        b. MLLib feature framework - deals with scaling, etc.

        c. Sparkling Water - It is to integrate Spark with another product H2O.

        d. SparkR - To work with R libraries

        e. [http://prediction.io](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fprediction.io) - This exposes RESTful APIs to do machine learning logic implementation.

Spark on YARN: a Deep Dive - Sandy Ryza (Cloudera) - [https://www.youtube.com/watch?v=N6pJhxCPe-Y](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dN6pJhxCPe-Y)

**Videos watched -**

1. Introduction to spark architecture

2. Introduction to spark streaming

3. Introduction to Hadoop (HDFS & Map/Reduce) for Spark Developers  
4. Introduction to Apache Spark  
5. Introduction to spark 2.0

6. Evolution of Apache Spark

7. Deep Dive: Apache Spark Memory Management

**Spark setup and running a sample program -**

Install Spark, Java and Scala in windows - [https://www.youtube.com/watch?v=WlE7RNdtfwE](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dWlE7RNdtfwE)

Setup Spark standalone cluster -  [http://paxcel.net/blog/how-to-setup-apache-spark-standalone-cluster-on-multiple-machine/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fpaxcel.net%2fblog%2fhow-to-setup-apache-spark-standalone-cluster-on-multiple-machine%2f" \t "_blank)

Running a sample program from cmd prompt - [https://www.youtube.com/watch?v=4sO-VgqHLp4&index=7&list=PLIxzgeMkSrQ9VIyTdeqypk8jNJGpg579U](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3d4sO-VgqHLp4%26index%3d7%26list%3dPLIxzgeMkSrQ9VIyTdeqypk8jNJGpg579U)

Calling Java methods from Scala code - [http://stackoverflow.com/questions/15507101/using-java-libraries-in-scala](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fstackoverflow.com%2fquestions%2f15507101%2fusing-java-libraries-in-scala)

[http://spark.training4exam.com/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fspark.training4exam.com%2f) - has training material for spark with HDFS and hands on pdf as well.

**To do for project**

Setup of HDFS

MongoDB setup

How to integrate spark with MongoDB?

How to integrate spark with HDFS?

How to create spark clusters? and their related configurations

How to write a spark program and execute it? - [https://www.youtube.com/watch?v=7k4yDKBYOcw](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3d7k4yDKBYOcw)

How to call the existing Java libraries from scala code that is written as part of spark programming? - [http://stackoverflow.com/questions/15507101/using-java-libraries-in-scala](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fstackoverflow.com%2fquestions%2f15507101%2fusing-java-libraries-in-scala)

**Hadoop installation and setup**

<https://ofirm.wordpress.com/2014/01/05/creating-a-virtualized-fully-distributed-hadoop-cluster-using-linux-containers/>

<http://doctuts.readthedocs.io/en/latest/hadoop.html>

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/>

**MongoDB setup**

Deploy a sharded cluster -

<https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/>

Step by step MongoDB sharded cluster(Ubuntu) -

<http://codingmiles.com/mongodb-sharded-cluster-deployment/>

Setup MongoDB shard(windows) -

<https://coderwall.com/p/bzz1ra/set-up-mongodb-shard-windows-local>

Replica setup(windows) -

<https://blog.basefarm.com/blog/how-to-install-mongodb-on-windows-server-2012-with-a-replication-set/>

**Why MongoDB eventually consistent?**

1. <https://stackoverflow.com/questions/11292215/where-does-mongodb-stand-in-the-cap-theorem>
2. <https://martin.kleppmann.com/2015/05/11/please-stop-calling-databases-cp-or-ap.html>
3. <https://aphyr.com/posts/322-call-me-maybe-mongodb-stale-reads>
4. [https://dzone.com/articles/how-acid-mongodb](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fdzone.com%2farticles%2fhow-acid-mongodb)
5. [http://stackoverflow.com/questions/11424828/why-does-my-mongodb-replica-keep-falling-behind?rq=1](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fstackoverflow.com%2fquestions%2f11424828%2fwhy-does-my-mongodb-replica-keep-falling-behind%3frq%3d1)

**MongoDB and Spark**

1. Spark connector guide - [https://docs.mongodb.com/spark-connector/master/java-api/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fdocs.mongodb.com%2fspark-connector%2fmaster%2fjava-api%2f)

2. With DataSet - [https://docs.mongodb.com/spark-connector/current/java/datasets-and-sql/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fdocs.mongodb.com%2fspark-connector%2fcurrent%2fjava%2fdatasets-and-sql%2f)

**Run mongodb**

1. In a cmd prompt run - mongod.exe --dbpath "C:\Program Files\MongoDB\data"

2. Open another cmd prompt and run – mongo

**Javascript frameworks Architecture**

**AngularJS Architecture -**

[https://stackoverflow.com/questions/14994391/thinking-in-angularjs-if-i-have-a-jquery-background](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fstackoverflow.com%2fquestions%2f14994391%2fthinking-in-angularjs-if-i-have-a-jquery-background)

[https://stackoverflow.com/questions/18414012/what-does-angularjs-do-better-than-jquery](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fstackoverflow.com%2fquestions%2f18414012%2fwhat-does-angularjs-do-better-than-jquery)

**Accessing files in HDFS using Java and accessing mongodb from spark using Java**

[https://stackoverflow.com/questions/17564074/accessing-files-in-hdfs-using-java](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fstackoverflow.com%2fquestions%2f17564074%2faccessing-files-in-hdfs-using-java)

Access Mongo from Spark -

[https://docs.mongodb.com/spark-connector/master/java/datasets-and-sql/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fdocs.mongodb.com%2fspark-connector%2fmaster%2fjava%2fdatasets-and-sql%2f)

Course in mongodb university - [https://university.mongodb.com/courses/M233/about](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2funiversity.mongodb.com%2fcourses%2fM233%2fabout)

**Hadoop – HDFS and Map Reduce**

Components of Distributed System -

1. Distributed Storage - HDFS

2. Distributed processing

    a. Resource management

    b. Distributed computation

3. Network

In Hadoop 1.0, HDFS was the distributed storage and MapReduce was taking care of resource management and distributed computation.

In Hadoop 2.0, HDFS was the distributed storage, YARN(Yet Another Resource Negotiator) takes care of resource management and MapReduce takes care of distributed computation.

**HDFS**

HDFS stores immutable data and also it works based on client server architecture. It is created based on GFS(Google File System). It has software based fault tolerance by replication.

The block size is 64MB by default. The value 64MB is fixed because we need to share this data between nodes and so need to use network for transfer. During when if the size is smaller like 64MB then the possibility of transfer getting failed is less. Also, even if it fails then we need to retry for only 64MB. At the same time, if the block size is 1GB or so then during failure we need to retry the whole 1GB again.

HDFS stores the data in binary format in file system.

HDFS has two type of nodes - Namenode(Master) and Datanode(Slave). The Namenode holds the metadata info and Datanode holds the actual data. The Namenode does the splitting of file as chunks to be stored in Datanode and passes on the info. The Datanode converts the data in chunk into binary format and stores in it. Datanode also takes care of replicating the chunk data in multiple data nodes.

The Namenode holds the following details

1. Name space info (directory structure)

2. File chunk mapping

3. chunk server location

In a node, each chunk will get an unique id for identification. In case of replica in multiple nodes, the same chunk id will be used in multiple nodes. But at any time, within the same node the same chunk id will not be used.

The Namenode stores it data in local master file system and if needed can store in network file system.

In Hadoop 1.0, we had a secondary name node which is not a backup for name node but does the compression of metadata which is been stored by name node.

In Hadoop 2.0, we have primary name node, secondary name node and backup name nodes which will be used when primary name node goes down. Also, zookeeper software is been used to manage the name nodes in switching between the active name nodes. i.e., if primary name node goes down then zookeeper immediately route the incoming request to backup name node which is active. Zookeeper gets the heartbeat of name nodes and keeps track of the active name nodes. This zookeeper software is been used in HBase, Cassandra, etc., for the similar purpose. Data nodes heart beat will be tracked by name nodes.

HDFS Balancer takes care of replica management in the cluster. We do have under-replicated data and more replicated data. Under replicated means due to some issue the data is not been replicated in 3 nodes. More replicated occurs when any one of the actual data node(to hold the replica) is down then the system will bring up another datanode to hold the replica. After some time when the fault data node is up then it will get the replica copy from other data nodes. In this case, we do have 4 copies of the same data instead of 3. This is called More replicated data. HDFS Balancer runs thru the data nodes and corrects these under-replicated and more-replicated data.

**Map/Reduce**

In Map Reduce operation, the file will get split into partitions. Here the splitting will be done by looking at the logical end points.

InputFormat is the API for splitting the file. By default, Hadoop tries to make the number of splits equal to no of chunks. But this split can be configurable as well.

The processing of split/partition can be distributed and the assignment of partition processing can be done like whichever the machine which holds the corresponding chunk, the same machine will be given first preference. There too, the node having chunk copy sitting near to master with good bandwidth will get the highest preference.

Each split processing will be done in a separate JVM. So, before starting the split processing, the JVM will be setup and then the processing will be done in that JVM. So, if we increase the no of split to increase the parallel processing then considerable amount of time will be spent in JVM setup. So, this needs to be considered while configuring the split count.

The processing in Hadoop will be done by Map and Reduce by key. They play will key-value pairs.

Mapper instance - Mapper is the process which holds the logic of processing the data and instance means the JVM instance on which the mapper runs. So, there can be multiple mappers with the same processing logic but works on different partition data. The output of each mapper instance will be persisted in local disk of slave as file and not in HDFS. So, each mapper instance will create its own file and have the processed output in it.

The slave/datanode can be used for both HDFS and MapReduce.

Shuffling - Moving the Mapper output to the place where Reduce can process it.

By default we have one Reducer instance. But if we want to increase it then by max we can have reducers equal to the number of keys.

Before doing Reduce, Hadoop does the following in order

1. Merge - merges the mapper output

2. Group - Group the mapper output based on keys

After doing merge and group, Hadoop kick starts reduce operation. The output of reduce can be stored in HDFS. But again give the directory name to store the reducer output instead of file name. Because if more than one reducer are running then a single file cannot be used by all reducer instances. Hence provide the directory name, so that all reducers can create a separate file to store their output.

**Only HDFS**

Excellent Videos about HDFS - [https://www.youtube.com/watch?v=X\_8LLoiznIA&list=PLjOv0CBS0xcLhG2xvM-Un2Hxji1g0ScrN&index=1](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dX_8LLoiznIA%26list%3dPLjOv0CBS0xcLhG2xvM-Un2Hxji1g0ScrN%26index%3d1)

**HDFS Architecture -**

1. Master Slave architecture. It works with single active namenode and multiple data nodes. Only one name node will be active at any time.

2. Namenode holds metadata, block location and fsimage & edit log.

3. Metadata has info like filename, file size, no. of blocks, block Ids, User, user group, file permission, replication, block size, etc. Block Id is unique across the cluster. Metadata will get stored in main memory. It is also been stored in edit log and later get converted to fsimage. This storage n fsimage & edit log is for recovery purpose.

4. Namenode stores only metadata and not actual data.

5. Data nodes store data as blocks. Default block size in HDFS is 64MB and in Cloudera it is 128MB. Files are broken in to blocks and get stored in HDFS.

6. Datanodes are workhorses of HDFS. The split blocks of the files will get stored in data nodes and 2 of its replica will be other two data nodes.

7. The last block size of a file will always be <= block size.

8. why blocks are replicated? for Reliable storage system.

9. Datanodes send block report to name node. Block report is a report created by scanning all blocks on the disks. The Block report has BlockId, generation stamp and size. Generation stamp holds the block version, which will be used for append operation.

10. Block location has details on block Id to location mapping. The Block location will get updated by block reports and HDFS write operations.

11. In metadata, file name is mapped with block ids. In Block location, block Ids are mapped with Datanodes.

12.Using metadata and block location, name node has complete info about the cluster.

13. Heartbeat will be send by data nodes to name nodes. Using this info, name node will identify the list of data nodes that are active. This heartbeat will be sent on every 3 seconds. If no heartbeat is been sent by the data node then name node will wait for 10 mins and even then no heartbeat received then that data node will be marked as out of service.

14. Heartbeat also carries disk info like total space, used space, free space, data transfer in progress, etc. These info will be used by name node while block allocation and load balancing.

**Note:** Group of data nodes form a rack. One or more racks form a cluster. The HDFS cluster has both racks and name node.

**HDFS write operation -**

To do the write operation in HDFS, we need HDFS client application which will do the write operation.

1. The write operation will be triggered by HDFS client by calling create(<file name>).

2. This create operation will send the RPC call to name node. Here the name node will validate the file by checking if it already exists and also client has write permission or not.

3. If the validation passes then the file info will get stored in main memory and edit log. Note that the data is still not stored in HDFS.

4. Now client starts with the next step in write operation. It starts writing the data in its local disk for the block size or till EOF reaches. Once it creates the local copy for a block size or less than that(if EOF is reached) then client will ask for block allocation to name node.

5. Name node looks out for the replication factor value(consider it as 3), and based on that, it returns three data node info to the client.

6. The client will flush the data to first data node as small packets(generally packet size will be 4kb). The first data node will store the value in its block and acknowledges back to client, so that client will start sending next packet. In the mean while, first data node also sends the data to next data node which will hold replica. Once that node stores data then it sends acknowledgement and also sends to 3rd data node(as replication factor is 3). Once all 3 data nodes stored the packet successfully then all 3 acknowledgements would have reached the client. This way the data will be sent in pipeline and all 3 data nodes will get the data stored in them. Once the data of a block is been stored in data nodes then data nodes will send the info to name node asking to update block location with block details.

7. The client will repeat the same procedure (as in #6) for remaining data and name node & datanode will update them accordingly.

8. After writing the complete file into HDFS, client will call the close() method which informs name node that the file copy is complete.

9. Finally name node's Metadata will hold the file info and Block location will get block info such as block id and data node details.

10. The block and data node allocation will be taken care by name node based on the space availability and data nodes cluster balance. Another point is that data is directly written on data nodes by the client. So, the load is on data nodes and less load on name node. Hence HDFS is scalable for write operation.

**HDFS Read operation -**

1. HDFS Client library will be used for read operation as well.

2. Open(<file name>) method is invoked at client end which will make client to send a RPC call to name node asking for block ids and data node info.

3. The name node will get the details from its metadata and block location and returns the info to the client. The response will have details like #1[DN1, DN3, DN4], where #1 represents block 1 and it is been stored in data node 1, 3 & 4. Note that the data node list sent in the response will be in sorted order based on the distance between the client and the data nodes. This will help to reduce the network latency.

4. The client will now communicate with nearest available data node and get the block data

5. Similar kind of read operation might happen within HDFS where in any of the HDFS data node might need some data for processing and so it might speak to name node and gets the list of data nodes holding the data. Using this, the respective data node will be contacted and the data will be retrieved.

6. Since clients are directly interacting with data nodes for data, at any point of time multiple clients can interact with data node and do the read operation. Hence HDFS is highly scalable.

**Block replica placement policy -**

Generally while writing the data into HDFS, the data nodes will be chosen based on the cluster balance and the space availability. Also, it will be preferred to have the replica in different rack instead of in the same rack. This is to achieve high reliability, availability and network bandwidth utilization. The network bandwidth utilization will be realized when a data node which is executing a task needs to store the data. In this case, the task will persist one copy on the same data node and remaining two copy(with RF = 3) on different data nodes in different racks. Here since the copy is in same data node the network latency will be very less.

Care should be taken while choosing the replication factor value. Because

1. if RF value is high then, highly reliable and available but less performance on network utilization and write performance.

2. If RF value is less then, less reliable and available but good network utilization and write performance.

**Data Node Out of service -**

When data node is out of service then name node will identify it based on the heartbeat it receives. It will wait for 10 mins and after that it will start the process of creating another replica for the blocks which were in failed data node. This is done to maintain the RF=3 value at any point of time.

**Cloud computing**

Planning and Designing cloud infrastructure  - [https://www.youtube.com/watch?v=NqGjklBovaI](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dNqGjklBovaI)

Four pillars for designing cloud based applications

1. Security

2. Reliability

3. Performance efficiency

4. Cost optimization

cloud computing design patterns - [https://msdn.microsoft.com/en-in/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fmsdn.microsoft.com%2fen-in%2f) - inside problem areas in cloud, you can navigate and look out for design patterns

J2EE design patterns - [http://www.j2eebrain.com/java-J2ee-j2ee-design-patterns.html](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fwww.j2eebrain.com%2fjava-J2ee-j2ee-design-patterns.html)

[http://cloudacademy.com/blog/cloud-foundry-benefits/](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=http%3a%2f%2fcloudacademy.com%2fblog%2fcloud-foundry-benefits%2f)

**Pivotal Cloud Foundry**

This helps in deploying and managing the services in cloud platform with less effort. No need to create dockers, configure things in cloud, etc.

1. BOSH - to create Cloud Foundry in IaaS platform

2. cf - command to create services in cloud foundry

a. Spring cloud services is been used for microservices along with spring boot.

b. PCF metrix - to handle logs

Build Your Very Own Private Cloud Foundry - [https://www.youtube.com/watch?v=v85r4Hy3jbs](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dv85r4Hy3jbs)

Inside Cloud Foundry: An Architectural Review - [https://www.youtube.com/watch?v=oXExLtmw0q4](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3doXExLtmw0q4)

Look out for "Getting Started - Cloud Foundry" in youtube

**AWS**

1.       Availability zones – data centers

2.       Edge locations – to take the service closer to the user and this has cache in it.

3.       Virtual Private Cloud(VPC) – AWS resources will be deployed in VPC which has virtual machines, databases, etc.

4.       Networking Direct Connect – this helps in connecting from our private network to VPC. This communication would be private.

5.       Route 53 – supports distributed DNS service. Allows domain name to be hosted on Amazon on port number 53.

**Machine Learning and Neural networks**

Look out for videos of "Luis Serrano". His videos are good about machine learning and deep learning.

**What is Machine Learning?**

**Good video -** [**https://www.youtube.com/watch?v=IpGxLWOIZy4**](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dIpGxLWOIZy4)

1. Linear Regression - Here using the existing data, we will try to get a line that is closer to the points in the graph with x and y axis. This line can be a straight line or circle or even a wave shape line. Using this line, we can predict the point in graph for data which has x axis value. i.e., For eg. Lets take the real estate case where in we have rate values for few of the houses with different sq.ft values. Have the house area in x axis and its rate in y axis. Using this data, draw a line which is closer to the points in the graph for the available x and y axis values. Now, find out the rate of a house whose area is not available in the available data. During this time, using the line drawn with the given data can get us the corresponding rate value by choosing the respective (x,y) point in the graph. This is Linear regression approach. For this, the challenge is we need to come up with the closest line in the graph for the given data, which will help us in getting the accurate value for any new inputs that are coming up.

How will we get this closest line? This can be achieved by summing up the distance between the line and each (x,y) coordinates in the graph. If the sum value is lesser then the line is closer to the points/coordinates, which will be the best. This approach is called Gradient Descent. i.e., trying with multiple values and coming up with the closest one.

2. Naïve Bayes algorithm - Here, we will analyse the data and come up with the list of features/behaviours which help us in finding the output. Using this collected list, we will try to find the output for any input.

For eg. If we want to find out the spam emails from the inbox then we will need to collect the list of cases which says the email is spam. Like, if the email has a specific keyword or spelling mistake or with no subject then the are spam emails.

3. Decision tree - This is a binary tree which we draw based on the data by applying the conditions and segregate them. For eg. if we have the data like people using different mobile apps. Now with this given data we need to group them based on the app which they use and come up with the tree structure. For eg. if the data is like, less than 20 years old are using facebook, and in >20 years, we see males use whatsapp and females use telegram. This way we can create the tree which can be used to predict the output for any input which comes in.

4. Logistic regression - Here too, we will come up with the line, but to segregate two different types of values. For eg. if we have the list of points in graph falling under two categories - Correct and wrong. Then we will come up with a line which can segregate these two values and so for any incoming value, if its point fall under correct area then it will be a correct else wrong value. So, how do we come up with the correct line that segregates correctly? The line should intersect in such a way that the average distance between closest correct values and closest wrong values to the line should be same or closer. This approach is called Linear optimization.

But here too, some times single line will not be sufficient to segregate the two categories. So, we may need to go with more than one lines to arrive at the area segregation. This is called neural networks. Because we will combine the output of multiple lines and arrive at the correct area segregation. In brain, the similar action will be performed like the output of multiple neurons will be collected by another neuron and arrives at a final output.

5. Kernel Trick - this approach will be taken when both the correct and wrong values are in the same straight line. And so only a curve line should be drawn to segregate them. Or even if it is 3-dimensional graph then we need to get an intermediate plate to differentiate both correct and wrong values. This will be done by identifying some mathematical calculation which can segregate the correct and wrong values. Like x+y or x\*y or x^2 or y^2. Out of these calculation any one would get us the values to easily segregate the categories. This is called support vector machine Kernel trick.

6. K-Means Clustering algorithm - This will be applied in cases where we need to find out the right location to place the given input. For eg. in a residential area we need to find out the location where we can place the pizza shop which should be easily accessible by many. We need to try out multiple options by placing the pizza shop location and understand how many houses can easily access. Based on that value we will arrive at the correct location.

7. Hierarchical clustering - Here we will try to find the closest houses and form a cluster. With that info will suggest where to place the pizza shop near by.

**Sharing data between different spark jobs**

To share the data between multiple spark jobs with different functionalities

1. Use Apache Ignite In-memory which shares the RDDs between spark jobs

2. Write the data in HBase and let the other job read from the HBase DB.

**Lenovo A6000 OS Downgrade**

<http://www.droiddosh.com/2015/10/downgrade-lenovo-a6000-from-lollipop-to-kitkat.html>

**Hadoop setup in Ubuntu**

[https://www.dezyre.com/hadoop-tutorial/hadoop-multinode-cluster-setup](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.dezyre.com%2fhadoop-tutorial%2fhadoop-multinode-cluster-setup)

[https://www.tutorialspoint.com/hadoop/hadoop\_multi\_node\_cluster.htm](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.tutorialspoint.com%2fhadoop%2fhadoop_multi_node_cluster.htm)

**Maven setup – To install local jars in maven repo and use them while building**

**To install local jars in maven build**

In eclipse  
1. Goto Run->Run configuration  
2. double click on "Maven Build"  
3. Provide Base directory - directory structure of the project  
4. Goals - install:install-file  
5. In Parameters, add the following  
 file - C:\Users\AF37730\Downloads\db2jcc4-10.1.jar  
 groupId - com.ibm.db2.jcc  
 artifactId - db2jcc4  
 version - 10.1  
 packaging - jar  
 localRepositoryPath - C:\Users\AF37730\.m2\repository  
6. Apply and run it.  
6. After doing the above, add the dependency in the pom.xml file

**Setting in pom.xml to add all the dependent jars in the built jar**

<build>

<plugins>

<!-- Maven Assembly Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-assembly-plugin</artifactId>

<version>2.4.1</version>

<configuration>

<!-- get all project dependencies -->

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<!-- MainClass in mainfest make a executable jar -->

<archive>

<manifest>

<mainClass>com.bala.mongo.MongoJson.controller.MongoDBReader</mainClass>

</manifest>

</archive>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<!-- bind to the packaging phase -->

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

**Microservices**

20-April-2017

=========

1. Microservices can be deployed in multiple machines as a seperate processes. But if, microservice uses its own DB then how do each process of same microservice will sync up their DB? Will all the processes of same microservice type uses the same DB?

2. Microservices example -   
 github.com/redhat-helloworld-msa

3. Spring Boot quick start - videos by Koushik

4. Refactor your Java EE application using Microservices and Containers - by Arun Gupta  
 - blog.arungupta.me

21-April-2017

=========

NoOps in Microservices -   
1. Service replication - Kubernetes  
2. Dependency resolution - Nexus.Kind of service discovery  
3. Failover - Circuit Breaker. A pattern to bring back the service. Netflix has libraries to implement circuit breaker pattern.  
4. Resiliency - Hystrix. This helps in getting the immediate alternate solution for any failover occured.  
5. Service monitoring, alerts and events - Logstash

Containers available in market for microservices - Docker and Rocket. Docker is more matured now.

Docker -   
Used to create containers for software applications  
Package Once Deploy Anywhere(PODA)

Docker Sworm is recently released to overcome the drawbacks in Docker

Kubernetes -   
orchestration system for Docker containers  
schedule across hosts. i.e., it can manage multiple hosts/servers.

The design woule be like  
 User Experience - by Openshift or WildFly  
 Cluster Management - by Kubernetes  
 Container - Docker  
 Container Host - Linux

PDF files created by Arun Gupta reg. microservices -   
[https://github.com/arun-gupta/microservices/tree/master/slides](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fgithub.com%2farun-gupta%2fmicroservices%2ftree%2fmaster%2fslides)

Youtube video -  
Refactor your Java EE application using Microservices and Containers by Arun Gupta - [https://www.youtube.com/watch?v=iJVW7v8O9BU](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3diJVW7v8O9BU)

References -   
github.com/arun-gupta/microservices  
github.com/javaee-samples/docker-java  
dzone.com/refcardz/getting-started-with-microservices

Getting Started with Spring Boot - [https://www.youtube.com/watch?v=sbPSjI4tt10](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dsbPSjI4tt10)

Getting started with Spring Cloud by Josh Long - [https://www.youtube.com/watch?v=SFDYdslOvu8](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dSFDYdslOvu8)

Building Microservices with Spring Cloud - [https://www.youtube.com/watch?v=ZyK5QrKCbwM](https://webapp2.syntelinc.com/OWA/redir.aspx?C=ybXvJyKQL0e83Ddp-IvUZPey9Nki5dQIQTuNTADcminpxFRXVCXUYolLJvz9Qa3RqETOyCDdDqI.&URL=https%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dZyK5QrKCbwM)

**To Learn**

1.       Java multithreading – Executor framework, Fork-Join, etc.

2.       Java memory model

3.       Java – Garbage collection algorithms

4.       Java 8 features – Lambda(closures, etc.), concurrency, etc.

5.       Java thread lifecycle and collections hierarchy

6.       Spring - MVC, Security, Boot, Integration

7.       Hibernate – Details, advantages and disadvantages, n+1 problem

8.       Design principles

9.       Architecture principles

10.   RESTful service design principle

11.   Design patterns – Gang of Four and J2EE design patterns

12.   Cloud Computing – 12 factors, principles to follow while suggesting cloud platform.

13.   Microservices – Docker, Kubernetes, API Management(APIGEE), Circuit Breaker

14.   MongoDB – Eventually consistent, sharding

15.   Cassandra basics

16.   MySQL – horizontal scaling

17.   Tomcat – clustering, configuration

18.   Nginx - Load Balancing, Reverse proxy

19.   Kafka – knowledge

20.   Big data – knowledge

21.  In-memory or cache frameworks

22. Message Queues

23. ZooKeeper – service replication

24. OpenShift and Cloud Foundry