

**Vishwakarma Government Engineering College, Chandkheda****Computer Engineering Department****Big Data Analytics (3170722)****Practical list – winter 2021**

#	Aim of Practical	CO	Page
1	To Demonstrate Installation and Configuration of MongoDB client Server.	CO1	1
2	Write the Mongoddb queries for creating database, collection, inserting documents, updating document, deleting documents.	CO1	3
3	Write the MongoDB queries for the given collection.	CO1	6
4	Write MongoDB queries for aggregate methods such as Count, Limit, Sort and similar to LIKE predicate in SQL.	CO1	10
5	To Demonstrate the Installation and Configuration of Single node and multimode Hadoop clusters.	CO4	17
6	To Develop a Map Reduce program for Word count for Hadoop cluster.	CO2	27
7	To study stream mining using case study approach.	CO3	32
8	To demonstrate Pig and Hive SQL queries using various operators.	CO3	40
9	To show the Installation steps for the SPARK on single node system.	CO3	44
10	Develop a word count program using SPARK.	CO3	46

## Practical – 1

**Aim:** To Demonstrate Installation and Configuration of MongoDB client Server.

STEP 1: Download the installer

- In the Version dropdown, select the version of MongoDB to download.
- In the Platform dropdown, select Windows.
- In the Package dropdown, select msi.
- Click Download.

STEP 2: Run the MongoDB installer

- Choose Setup Type

You can choose either the Complete (recommended for most users) or Custom setup type. The Complete setup option installs MongoDB and the MongoDB tools to the default location. The Custom setup option allows you to specify which executables are installed and where.

- Service Configuration

Starting in MongoDB 4.0, you can set up MongoDB as a Windows service during the install or just install the binaries.

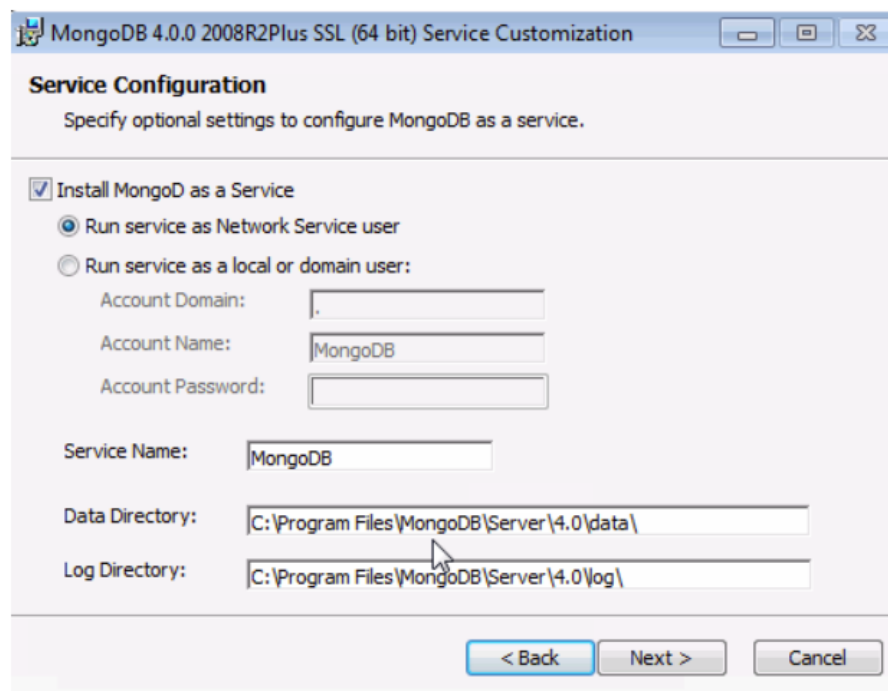


Fig 1.1 installation

STEP 3: Create database directory.

- Create the data directory where MongoDB stores data. MongoDB's default data directory path is the absolute path \data\db on the drive from which you start MongoDB.

STEP 4: Start your MongoDB database

To start MongoDB, run exe.

```
"C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe" --dbpath="c:\data\db"
```

The --dbpath option points to your database directory.

## Practical – 2

**Aim:** Write the Mongodb queries for creating database, collection, inserting documents, updating document, deleting documents.

```
> show dbs
admin  0.000GB
config 0.000GB
local  0.000GB
mydb   0.000GB

> use test
switched to db test

> show dbs
admin  0.000GB
config 0.000GB
local  0.000GB
mydb   0.000GB

> db.createCollection('students')
{ "ok" : 1 }

> show dbs
admin  0.000GB
config 0.000GB
local  0.000GB
mydb   0.000GB
test   0.000GB

> db.students.insertMany([ { "_id": NumberInt(0), "name": "aimee Zank", "scores": [ {
"score": 1.463179736705023, "type": "exam" }, { "score": 11.78273309957772, "type": "quiz" },
{ "score": 35.8740349954354, "type": "homework" } ] },
{ "_id": NumberInt(4), "name": "Zachary Langlais", "scores": [ { "score": 78.68385091304332,
"type": "exam" }, { "score": 90.2963101368042, "type": "quiz" }, {
"score": 34.41620148042529, "type": "homework" } ] },
{ "_id": NumberInt(5), "name": "Wilburn Spiess", "scores": [ { "score": 44.87186330181261,
"type": "exam" }, { "score": 25.72395114668016, "type": "quiz" }, {
"score": 63.42288310628662, "type": "homework" } ] },
{ "_id": NumberInt(7), "name": "Salena Olmos", "scores": [ { "score": 90.37826509157176,
"type": "exam" }, { "score": 42.48780666956811, "type": "quiz" }, {
"score": 96.52986171633331, "type": "homework" } ] },
{ "_id": NumberInt(12), "name": "Quincy Danaheer", "scores": [ { "score": 54.29841278520669,
"type": "exam" }, { "score": 85.61270164694737, "type": "quiz" }, {
"score": 80.40732356118075, "type": "homework" } ] },
```

```

{ "_id": NumberInt(13), "name": "Jessika Dagenais", "scores": [ { "score": 90.47179954427436,
"type": "exam" }, { "score": 90.3001402468489, "type": "quiz" }, {
"score": 95.17753772405909, "type": "homework" } ] },
{ "_id": NumberInt(14), "name": "Alix Sherrill", "scores": [ { "score": 25.15924151998215,
"type": "exam" }, { "score": 68.64484047692098, "type": "quiz" }, {
"score": 24.68462152686763, "type": "homework" } ] },
{ "_id": NumberInt(15), "name": "Tambra Mercure", "scores": [ { "score": 69.1565022533158,
"type": "exam" }, { "score": 3.311794422000724, "type": "quiz" }, {
"score": 45.03178973642521, "type": "homework" } ] },
{ "_id": NumberInt(9), "name": "Sanda Ryba", "scores": [ { "score": 97.00509953654694,
"type": "exam" }, { "score": 97.80449632538915, "type": "quiz" }, {
"score": 25.27368532432955, "type": "homework" } ] },
{ "_id": NumberInt(21), "name": "Rosana Vales", "scores": [ { "score": 46.2289476258328,
"type": "exam" }, { "score": 98.34164225207036, "type": "quiz" }, {
"score": 36.18769746805938, "type": "homework" } ] },
{ "_id": NumberInt(22), "name": "Margart Vitello", "scores": [ { "score": 75.04996547553947,
"type": "exam" }, { "score": 10.23046475899236, "type": "quiz" }, {
"score": 96.72520512117761, "type": "homework" } ] },
{ "_id": NumberInt(24), "name": "Jesusa Rickenbacker", "scores": [ {
"score": 86.0319702155683, "type": "exam" }, { "score": 1.967495200433389, "type": "quiz" },
{ "score": 61.10861071547914, "type": "homework" } ] },
{ "_id": NumberInt(2), "name": "Corliss Zuk", "scores": [ { "score": 67.03077096065002,
"type": "exam" }, { "score": 6.301851677835235, "type": "quiz" }, {
"score": 66.28344683278382, "type": "homework" } ] },
{ "_id": NumberInt(1), "name": "Aurelia Menendez", "scores": [ { "score": 60.06045071030959,
"type": "exam" }, { "score": 52.79790691903873, "type": "quiz" }, {
"score": 71.76133439165544, "type": "homework" } ] },
{ "_id": NumberInt(6), "name": "Jenette Flanders", "scores": [ { "score": 37.32285459166097,
"type": "exam" }, { "score": 28.32634976913737, "type": "quiz" }, {
"score": 81.57115318686338, "type": "homework" } ] },
{ "_id": NumberInt(8), "name": "Daphne Zheng", "scores": [ { "score": 22.13583712862635,
"type": "exam" }, { "score": 14.63969941335069, "type": "quiz" }, {
"score": 75.94123677556644, "type": "homework" } ] },
{ "_id": NumberInt(10), "name": "Denisha Cast", "scores": [ { "score": 45.61876862259409,
"type": "exam" }, { "score": 98.35723209418343, "type": "quiz" }, {
"score": 55.90835657173456, "type": "homework" } ] },
{ "_id": NumberInt(11), "name": "Marcus Blohm", "scores": [ { "score": 78.42617835651868,
"type": "exam" }, { "score": 82.58372817930675, "type": "quiz" }, {
"score": 87.49924733328717, "type": "homework" } ] },
{ "_id": NumberInt(16), "name": "Dodie Staller", "scores": [ { "score": 7.772386442858281,
"type": "exam" }, { "score": 31.84300235104542, "type": "quiz" }, {
"score": 80.52136407989194, "type": "homework" } ] },
{ "_id": NumberInt(17), "name": "Fletcher McConnell", "scores": [ {
"score": 39.41011069729274, "type": "exam" }, { "score": 81.13270307809924, "type": "quiz" },
{ "score": 97.70116640402922, "type": "homework" } ] },

```

```
{ "_id": NumberInt(20), "name": "Tressa Schwing", "scores": [ { "score": 42.17439799514388,
"score": 71.99314840599558, "type": "quiz" }, {
"score": 81.23972632069464, "type": "homework" } ] },
{ "_id": NumberInt(19), "name": "Gisela Levin", "scores": [ { "score": 44.51211101958831,
"score": 0.6578497966368002, "type": "quiz" }, {
"score": 93.36341655949683, "type": "homework" } ] },
{ "_id": NumberInt(23), "name": "Tamika Schildgen", "scores": [ {
"score": 45.65432764125526, "type": "exam" }, { "score": 64.32927049658846, "type": "quiz" },
{ "score": 83.53933351660562, "type": "homework" } ] },
{ "_id": NumberInt(18), "name": "Verdell Sowinski", "scores": [ {
"score": 62.12870233109035, "type": "exam" }, { "score": 84.74586220889356, "type": "quiz" },
{ "score": 81.58947824932574, "type": "homework" } ] ] }
```

```
db.friends.update(search term,new data)
```

```
db.collection.updateOne(<filter>, <update>, <options>)
```

```
db.collection.updateMany(<filter>, <update>, <options>)
```

```
db.collection.replaceOne(<filter>, <update>, <options>)
```

```
> db.friends.update({},{ $set: { gender: "Male" } })
```

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.friends.update({},{ $set: { gender: "Male" } }, { multi: true })
```

```
WriteResult({ "nMatched" : 3, "nUpserted" : 0, "nModified" : 2 })
```

```
> db.friends.update({},{ $unset: { gender: "Male" } }, { multi: true })
```

```
WriteResult({ "nMatched" : 3, "nUpserted" : 0, "nModified" : 3 })
```

```
> show collections
```

```
students
```

```
> db.students.drop()
```

```
True
```

### Practical – 3

**Aim:** Solve following queries using various mongodb search criteria.

(a) Find the document wherein the name of student has value 'Fletcher McConnell'.

```
> db.students.find({ name:"Fletcher McConnell"})
{"_id": 17, "name": "Fletcher McConnell", "scores": [{"score": 39.41011069729274, "type": "exam"}, {"score": 81.13270307809924, "type": "quiz"}, {"score": 97.70116640402922, "type": "homework"}]}
```

(b) Display name of students from student collection.

```
> db.students.find({}, {name:true, _id:false})
{"name": "aimee Zank"}
{"name": "Zachary Langlais"}
{"name": "Wilburn Spiess"}
{"name": "Salena Olmos"}
{"name": "Quincy Danaher"}
{"name": "Jessika Dagenais"}
{"name": "Alix Sherrill"}
{"name": "Tambra Mercure"}
{"name": "Sanda Ryba"}
{"name": "Rosana Vales"}
{"name": "Margart Vitello"}
{"name": "Jesusa Rickenbacker"}
{"name": "Corliss Zuk"}
{"name": "Aurelia Menendez"}
{"name": "Jenette Flanders"}
{"name": "Daphne Zheng"}
{"name": "Denisha Cast"}
{"name": "Marcus Blohm"}
{"name": "Dodie Staller"}
{"name": "Fletcher McConnell"}
```

(c) Display name of student with id of the student having id value 22.

```
> db.students.find({_id:22})
{"_id": 22, "name": "Margart Vitello", "scores": [{"score": 75.04996547553947, "type": "exam"}, {"score": 10.23046475899236, "type": "quiz"}, {"score": 96.72520512117761, "type": "homework"}]}
```

(d) display documents with students id with 1 to 3.

```
> db.students.find({_id:{$gt:0,$lt:4}})
{"_id": 1, "name": "Aurelia Menendez", "scores": [{"score": 60.06045071030959, "type": "exam"}, {"score": 52.79790691903873, "type": "quiz"}, {"score": 71.76133439165544, "type": "homework"}]}
{"_id": 2, "name": "Corliss Zuk", "scores": [{"score": 67.03077096065002, "type": "exam"}, {"score": 6.301851677835235, "type": "quiz"}, {"score": 66.28344683278382, "type": "homework"}]}
```

(e) display documents with students name is 'Tressa Schwing' and 'exam' score greater than 85.35.

```
> db.students.find({name:"Tressa Schwing","scores.type":"exam","scores.score":{$gt:85.35}})
{"_id": 20, "name": "Tressa Schwing", "scores": [{"score": 42.17439799514388, "type": "exam"}, {"score": 71.99314840599558, "type": "quiz"}, {"score": 81.23972632069464, "type": "homework"}]}
```

(f) display all documents with 'homework' type score is less than 95.

```
> db.students.find({ scores: { $elemMatch: { score: {$lt:95.0}, type: "homework" } } })
{"_id": 0, "name": "aimee Zank", "scores": [{"score": 1.463179736705023, "type": "exam"}, {"score": 11.78273309957772, "type": "quiz"}, {"score": 35.8740349954354, "type": "homework"}]}
{"_id": 4, "name": "Zachary Langlais", "scores": [{"score": 78.68385091304332, "type": "exam"}, {"score": 90.2963101368042, "type": "quiz"}, {"score": 34.41620148042529, "type": "homework"}]}
{"_id": 5, "name": "Wilburn Spiess", "scores": [{"score": 44.87186330181261, "type": "exam"}, {"score": 25.72395114668016, "type": "quiz"}, {"score": 63.42288310628662, "type": "homework"}]}
{"_id": 12, "name": "Quincy Danaher", "scores": [{"score": 54.29841278520669, "type": "exam"}, {"score": 85.61270164694737, "type": "quiz"}, {"score": 80.40732356118075, "type": "homework"}]}
{"_id": 14, "name": "Alix Sherrill", "scores": [{"score": 25.15924151998215, "type": "exam"}, {"score": 68.64484047692098, "type": "quiz"}, {"score": 24.68462152686763, "type": "homework"}]}
{"_id": 15, "name": "Tambra Mercure", "scores": [{"score": 69.1565022533158, "type": "exam"}, {"score": 3.311794422000724, "type": "quiz"}, {"score": 45.03178973642521, "type": "homework"}]}
{"_id": 9, "name": "Sanda Ryba", "scores": [{"score": 97.00509953654694, "type": "exam"}, {"score": 97.80449632538915, "type": "quiz"}, {"score": 25.27368532432955, "type": "homework"}]}
```



```
{ "_id" : 21, "name" : "Rosana Vales", "scores" : [ { "score" : 46.2289476258328, "type" :
"exam" }, { "score" : 98.34164225207036, "type" : "quiz" }, { "score" : 36.18769746805938,
"type" : "homework" } ] }
{ "_id" : 24, "name" : "Jesusa Rickenbacker", "scores" : [ { "score" : 86.0319702155683,
"type" : "exam" }, { "score" : 1.967495200433389, "type" : "quiz" }, { "score" :
61.10861071547914, "type" : "homework" } ] }
{ "_id" : 2, "name" : "Corliss Zuk", "scores" : [ { "score" : 67.03077096065002, "type" :
"exam" }, { "score" : 6.301851677835235, "type" : "quiz" }, { "score" : 66.28344683278382,
"type" : "homework" } ] }
{ "_id" : 1, "name" : "Aurelia Menendez", "scores" : [ { "score" : 60.06045071030959, "type" :
"exam" }, { "score" : 52.79790691903873, "type" : "quiz" }, { "score" : 71.76133439165544,
"type" : "homework" } ] }
{ "_id" : 6, "name" : "Jenette Flanders", "scores" : [ { "score" : 37.32285459166097, "type" :
"exam" }, { "score" : 28.32634976913737, "type" : "quiz" }, { "score" : 81.57115318686338,
"type" : "homework" } ] }
{ "_id" : 8, "name" : "Daphne Zheng", "scores" : [ { "score" : 22.13583712862635, "type" :
"exam" }, { "score" : 14.63969941335069, "type" : "quiz" }, { "score" : 75.94123677556644,
"type" : "homework" } ] }
{ "_id" : 10, "name" : "Denisha Cast", "scores" : [ { "score" : 45.61876862259409, "type" :
"exam" }, { "score" : 98.35723209418343, "type" : "quiz" }, { "score" : 55.90835657173456,
"type" : "homework" } ] }
{ "_id" : 11, "name" : "Marcus Blohm", "scores" : [ { "score" : 78.42617835651868, "type" :
"exam" }, { "score" : 82.58372817930675, "type" : "quiz" }, { "score" : 87.49924733328717,
"type" : "homework" } ] }
{ "_id" : 16, "name" : "Dodie Staller", "scores" : [ { "score" : 7.772386442858281, "type" :
"exam" }, { "score" : 31.84300235104542, "type" : "quiz" }, { "score" : 80.52136407989194,
"type" : "homework" } ] }
{ "_id" : 20, "name" : "Tressa Schwing", "scores" : [ { "score" : 42.17439799514388, "type" :
"exam" }, { "score" : 71.99314840599558, "type" : "quiz" }, { "score" : 81.23972632069464,
"type" : "homework" } ] }
{ "_id" : 19, "name" : "Gisela Levin", "scores" : [ { "score" : 44.51211101958831, "type" :
"exam" }, { "score" : 0.6578497966368002, "type" : "quiz" }, { "score" : 93.36341655949683,
"type" : "homework" } ] }
{ "_id" : 23, "name" : "Tamika Schildgen", "scores" : [ { "score" : 45.65432764125526, "type" :
"exam" }, { "score" : 64.32927049658846, "type" : "quiz" }, { "score" : 83.53933351660562,
"type" : "homework" } ] }
{ "_id" : 18, "name" : "Verdell Sowinski", "scores" : [ { "score" : 62.12870233109035, "type" :
"exam" }, { "score" : 84.74586220889356, "type" : "quiz" }, { "score" : 81.58947824932574,
"type" : "homework" } ] }
```

(g) retrieve all documents with 'quiz' score between 80 and 90 inclusively.

```
> db.students.find({ scores: { $elemMatch: { score: { $gte: 80, $lte: 90 }, type: "homework" } } })
```

```
{ "_id" : 12, "name" : "Quincy Danaher", "scores" : [ { "score" : 54.29841278520669, "type" :  
"exam" }, { "score" : 85.61270164694737, "type" : "quiz" }, { "score" : 80.40732356118075,  
"type" : "homework" } ] }  
{ "_id" : 6, "name" : "Jenette Flanders", "scores" : [ { "score" : 37.32285459166097, "type" :  
"exam" }, { "score" : 28.32634976913737, "type" : "quiz" }, { "score" : 81.57115318686338,  
"type" : "homework" } ] }  
{ "_id" : 11, "name" : "Marcus Blohm", "scores" : [ { "score" : 78.42617835651868, "type" :  
"exam" }, { "score" : 82.58372817930675, "type" : "quiz" }, { "score" : 87.49924733328717,  
"type" : "homework" } ] }  
{ "_id" : 16, "name" : "Dodie Staller", "scores" : [ { "score" : 7.772386442858281, "type" :  
"exam" }, { "score" : 31.84300235104542, "type" : "quiz" }, { "score" : 80.52136407989194,  
"type" : "homework" } ] }  
{ "_id" : 20, "name" : "Tressa Schwing", "scores" : [ { "score" : 42.17439799514388, "type" :  
"exam" }, { "score" : 71.99314840599558, "type" : "quiz" }, { "score" : 81.23972632069464,  
"type" : "homework" } ] }  
{ "_id" : 23, "name" : "Tamika Schildgen", "scores" : [ { "score" : 45.65432764125526, "type"  
: "exam" }, { "score" : 64.32927049658846, "type" : "quiz" }, { "score" : 83.53933351660562,  
"type" : "homework" } ] }  
{ "_id" : 18, "name" : "Verdell Sowinski", "scores" : [ { "score" : 62.12870233109035, "type"  
: "exam" }, { "score" : 84.74586220889356, "type" : "quiz" }, { "score" : 81.58947824932574,  
"type" : "homework" } ] }
```

## Practical – 4

**Aim:** Write MongoDB queries for aggregate methods such as Count, Limit, Sort and similar to LIKE predicate in SQL.

(a) Display documents in the ascending order of `_id`.

```
> db.students.find({}).sort({_id:1})
{"_id": 0, "name": "aimee Zank", "scores": [{"score": 1.463179736705023, "type": "exam"}, {"score": 11.78273309957772, "type": "quiz"}, {"score": 35.8740349954354, "type": "homework"}] }
{"_id": 1, "name": "Aurelia Menendez", "scores": [{"score": 60.06045071030959, "type": "exam"}, {"score": 52.79790691903873, "type": "quiz"}, {"score": 71.76133439165544, "type": "homework"}] }
{"_id": 2, "name": "Corliss Zuk", "scores": [{"score": 67.03077096065002, "type": "exam"}, {"score": 6.301851677835235, "type": "quiz"}, {"score": 66.28344683278382, "type": "homework"}] }
{"_id": 4, "name": "Zachary Langlais", "scores": [{"score": 78.68385091304332, "type": "exam"}, {"score": 90.2963101368042, "type": "quiz"}, {"score": 34.41620148042529, "type": "homework"}] }
{"_id": 5, "name": "Wilburn Spiess", "scores": [{"score": 44.87186330181261, "type": "exam"}, {"score": 25.72395114668016, "type": "quiz"}, {"score": 63.42288310628662, "type": "homework"}] }
{"_id": 6, "name": "Jenette Flanders", "scores": [{"score": 37.32285459166097, "type": "exam"}, {"score": 28.32634976913737, "type": "quiz"}, {"score": 81.57115318686338, "type": "homework"}] }
{"_id": 7, "name": "Salena Olmos", "scores": [{"score": 90.37826509157176, "type": "exam"}, {"score": 42.48780666956811, "type": "quiz"}, {"score": 96.52986171633331, "type": "homework"}] }
{"_id": 8, "name": "Daphne Zheng", "scores": [{"score": 22.13583712862635, "type": "exam"}, {"score": 14.63969941335069, "type": "quiz"}, {"score": 75.94123677556644, "type": "homework"}] }
{"_id": 9, "name": "Sanda Ryba", "scores": [{"score": 97.00509953654694, "type": "exam"}, {"score": 97.80449632538915, "type": "quiz"}, {"score": 25.27368532432955, "type": "homework"}] }
{"_id": 10, "name": "Denisha Cast", "scores": [{"score": 45.61876862259409, "type": "exam"}, {"score": 98.35723209418343, "type": "quiz"}, {"score": 55.90835657173456, "type": "homework"}] }
{"_id": 11, "name": "Marcus Blohm", "scores": [{"score": 78.42617835651868, "type": "exam"}, {"score": 82.58372817930675, "type": "quiz"}, {"score": 87.49924733328717, "type": "homework"}] }
{"_id": 12, "name": "Quincy Danaher", "scores": [{"score": 54.29841278520669, "type": "exam"}, {"score": 85.61270164694737, "type": "quiz"}, {"score": 80.40732356118075, "type": "homework"}] }
```

```

{"_id": 13, "name": "Jessika Dagenais", "scores": [{"score": 90.47179954427436, "type": "exam"}, {"score": 90.3001402468489, "type": "quiz"}, {"score": 95.17753772405909, "type": "homework"}] }
{"_id": 14, "name": "Alix Sherrill", "scores": [{"score": 25.15924151998215, "type": "exam"}, {"score": 68.64484047692098, "type": "quiz"}, {"score": 24.68462152686763, "type": "homework"}] }
{"_id": 15, "name": "Tambra Mercure", "scores": [{"score": 69.1565022533158, "type": "exam"}, {"score": 3.311794422000724, "type": "quiz"}, {"score": 45.03178973642521, "type": "homework"}] }
{"_id": 16, "name": "Dodie Staller", "scores": [{"score": 7.772386442858281, "type": "exam"}, {"score": 31.84300235104542, "type": "quiz"}, {"score": 80.52136407989194, "type": "homework"}] }
{"_id": 17, "name": "Fletcher McConnell", "scores": [{"score": 39.41011069729274, "type": "exam"}, {"score": 81.13270307809924, "type": "quiz"}, {"score": 97.70116640402922, "type": "homework"}] }
{"_id": 18, "name": "Verdell Sowinski", "scores": [{"score": 62.12870233109035, "type": "exam"}, {"score": 84.74586220889356, "type": "quiz"}, {"score": 81.58947824932574, "type": "homework"}] }
{"_id": 19, "name": "Gisela Levin", "scores": [{"score": 44.51211101958831, "type": "exam"}, {"score": 0.6578497966368002, "type": "quiz"}, {"score": 93.36341655949683, "type": "homework"}] }
{"_id": 20, "name": "Tressa Schwing", "scores": [{"score": 42.17439799514388, "type": "exam"}, {"score": 71.99314840599558, "type": "quiz"}, {"score": 81.23972632069464, "type": "homework"}] }
Type "it" for more
> it
{"_id": 21, "name": "Rosana Vales", "scores": [{"score": 46.2289476258328, "type": "exam"}, {"score": 98.34164225207036, "type": "quiz"}, {"score": 36.18769746805938, "type": "homework"}] }
{"_id": 22, "name": "Margart Vitello", "scores": [{"score": 75.04996547553947, "type": "exam"}, {"score": 10.23046475899236, "type": "quiz"}, {"score": 96.72520512117761, "type": "homework"}] }
{"_id": 23, "name": "Tamika Schildgen", "scores": [{"score": 45.65432764125526, "type": "exam"}, {"score": 64.32927049658846, "type": "quiz"}, {"score": 83.53933351660562, "type": "homework"}] }
{"_id": 24, "name": "Jesusa Rickenbacker", "scores": [{"score": 86.0319702155683, "type": "exam"}, {"score": 1.967495200433389, "type": "quiz"}, {"score": 61.10861071547914, "type": "homework"}] }

```

(b) Display documents in the descending order of name.

```

> db.students.find({}, {name:true}).sort({"name":-1})
{"_id": 0, "name": "aimee Zank"}
{"_id": 4, "name": "Zachary Langlais"}
{"_id": 5, "name": "Wilburn Spiess"}
{"_id": 18, "name": "Verdell Sowinski"}

```

```

{"_id": 20, "name": "Tressa Schwing"}
{"_id": 23, "name": "Tamika Schildgen"}
{"_id": 15, "name": "Tambra Mercure"}
{"_id": 9, "name": "Sanda Ryba"}
{"_id": 7, "name": "Salena Olmos"}
{"_id": 21, "name": "Rosana Vales"}
{"_id": 12, "name": "Quincy Danaher"}
{"_id": 22, "name": "Margart Vitello"}
{"_id": 11, "name": "Marcus Blohm"}
{"_id": 24, "name": "Jesusa Rickenbacker"}
{"_id": 13, "name": "Jessika Dagenais"}
{"_id": 6, "name": "Jenette Flanders"}
{"_id": 19, "name": "Gisela Levin"}
{"_id": 17, "name": "Fletcher McConnell"}
{"_id": 16, "name": "Dodie Staller"}
{"_id": 10, "name": "Denisha Cast"}
Type "it" for more
> it
{"_id": 8, "name": "Daphne Zheng"}
{"_id": 2, "name": "Corliss Zuk"}
{"_id": 1, "name": "Aurelia Menendez"}
{"_id": 14, "name": "Alix Sherrill"}

```

(c) Display documents first in the ascending order of `_id` and then descending order of name.

```

> db.students.find({}, {name:true}).sort({_id:1,"name":-1})
{"_id": 0, "name": "aimee Zank"}
{"_id": 1, "name": "Aurelia Menendez"}
{"_id": 2, "name": "Corliss Zuk"}
{"_id": 4, "name": "Zachary Langlais"}
{"_id": 5, "name": "Wilburn Spiess"}
{"_id": 6, "name": "Jenette Flanders"}
{"_id": 7, "name": "Salena Olmos"}
{"_id": 8, "name": "Daphne Zheng"}
{"_id": 9, "name": "Sanda Ryba"}
{"_id": 10, "name": "Denisha Cast"}
{"_id": 11, "name": "Marcus Blohm"}
{"_id": 12, "name": "Quincy Danaher"}
{"_id": 13, "name": "Jessika Dagenais"}
{"_id": 14, "name": "Alix Sherrill"}
{"_id": 15, "name": "Tambra Mercure"}
{"_id": 16, "name": "Dodie Staller"}
{"_id": 17, "name": "Fletcher McConnell"}
{"_id": 18, "name": "Verdell Sowinski"}

```

```

{"_id": 19, "name": "Gisela Levin"}
{"_id": 20, "name": "Tressa Schwing"}
Type "it" for more
> it
{"_id": 21, "name": "Rosana Vales"}
{"_id": 22, "name": "Margart Vitello"}
{"_id": 23, "name": "Tamika Schildgen"}
{"_id": 24, "name": "Jesusa Rickenbacker"}

```

(d) Display all documents except first two from students collection.

```

> db.students.find().sort({_id:1}).skip(2)
{ "_id" : 2, "name" : "Corliss Zuk", "scores" : [ { "score" : 67.03077096065002, "type" :
"exam" }, { "score" : 6.301851677835235, "type" : "quiz" }, { "score" : 66.28344683278382,
"type" : "homework" } ] }
{ "_id" : 4, "name" : "Zachary Langlais", "scores" : [ { "score" : 78.68385091304332, "type" :
"exam" }, { "score" : 90.2963101368042, "type" : "quiz" }, { "score" : 34.41620148042529,
"type" : "homework" } ] }
{ "_id" : 5, "name" : "Wilburn Spiess", "scores" : [ { "score" : 44.87186330181261, "type" :
"exam" }, { "score" : 25.72395114668016, "type" : "quiz" }, { "score" : 63.42288310628662,
"type" : "homework" } ] }
{ "_id" : 6, "name" : "Jenette Flanders", "scores" : [ { "score" : 37.32285459166097, "type" :
"exam" }, { "score" : 28.32634976913737, "type" : "quiz" }, { "score" : 81.57115318686338,
"type" : "homework" } ] }
{ "_id" : 7, "name" : "Salena Olmos", "scores" : [ { "score" : 90.37826509157176, "type" :
"exam" }, { "score" : 42.48780666956811, "type" : "quiz" }, { "score" : 96.52986171633331,
"type" : "homework" } ] }
{ "_id" : 8, "name" : "Daphne Zheng", "scores" : [ { "score" : 22.13583712862635, "type" :
"exam" }, { "score" : 14.63969941335069, "type" : "quiz" }, { "score" : 75.94123677556644,
"type" : "homework" } ] }
{ "_id" : 9, "name" : "Sanda Ryba", "scores" : [ { "score" : 97.00509953654694, "type" :
"exam" }, { "score" : 97.80449632538915, "type" : "quiz" }, { "score" : 25.27368532432955,
"type" : "homework" } ] }
{ "_id" : 10, "name" : "Denisha Cast", "scores" : [ { "score" : 45.61876862259409, "type" :
"exam" }, { "score" : 98.35723209418343, "type" : "quiz" }, { "score" : 55.90835657173456,
"type" : "homework" } ] }
{ "_id" : 11, "name" : "Marcus Blohm", "scores" : [ { "score" : 78.42617835651868, "type" :
"exam" }, { "score" : 82.58372817930675, "type" : "quiz" }, { "score" : 87.49924733328717,
"type" : "homework" } ] }
{ "_id" : 12, "name" : "Quincy Danaher", "scores" : [ { "score" : 54.29841278520669, "type" :
"exam" }, { "score" : 85.61270164694737, "type" : "quiz" }, { "score" : 80.40732356118075,
"type" : "homework" } ] }
{ "_id" : 13, "name" : "Jessika Dagenais", "scores" : [ { "score" : 90.47179954427436, "type" :
"exam" }, { "score" : 90.3001402468489, "type" : "quiz" }, { "score" : 95.17753772405909,
"type" : "homework" } ] }

```

```
{ "_id" : 14, "name" : "Alix Sherrill", "scores" : [ { "score" : 25.15924151998215, "type" :
"exam" }, { "score" : 68.64484047692098, "type" : "quiz" }, { "score" : 24.68462152686763,
"type" : "homework" } ] }
{ "_id" : 15, "name" : "Tambra Mercure", "scores" : [ { "score" : 69.1565022533158, "type" :
"exam" }, { "score" : 3.311794422000724, "type" : "quiz" }, { "score" : 45.03178973642521,
"type" : "homework" } ] }
{ "_id" : 16, "name" : "Dodie Staller", "scores" : [ { "score" : 7.772386442858281, "type" :
"exam" }, { "score" : 31.84300235104542, "type" : "quiz" }, { "score" : 80.52136407989194,
"type" : "homework" } ] }
{ "_id" : 17, "name" : "Fletcher Mcconnell", "scores" : [ { "score" : 39.41011069729274,
"type" : "exam" }, { "score" : 81.13270307809924, "type" : "quiz" }, { "score" :
97.70116640402922, "type" : "homework" } ] }
{ "_id" : 18, "name" : "Verdell Sowinski", "scores" : [ { "score" : 62.12870233109035, "type"
: "exam" }, { "score" : 84.74586220889356, "type" : "quiz" }, { "score" : 81.58947824932574,
"type" : "homework" } ] }
{ "_id" : 19, "name" : "Gisela Levin", "scores" : [ { "score" : 44.51211101958831, "type" :
"exam" }, { "score" : 0.6578497966368002, "type" : "quiz" }, { "score" : 93.36341655949683,
"type" : "homework" } ] }
{ "_id" : 20, "name" : "Tressa Schwing", "scores" : [ { "score" : 42.17439799514388, "type" :
"exam" }, { "score" : 71.99314840599558, "type" : "quiz" }, { "score" : 81.23972632069464,
"type" : "homework" } ] }
{ "_id" : 21, "name" : "Rosana Vales", "scores" : [ { "score" : 46.2289476258328, "type" :
"exam" }, { "score" : 98.34164225207036, "type" : "quiz" }, { "score" : 36.18769746805938,
"type" : "homework" } ] }
{ "_id" : 22, "name" : "Margart Vitello", "scores" : [ { "score" : 75.04996547553947, "type" :
"exam" }, { "score" : 10.23046475899236, "type" : "quiz" }, { "score" : 96.72520512117761,
"type" : "homework" } ] }
Type "it" for more
> it
{ "_id" : 23, "name" : "Tamika Schildgen", "scores" : [ { "score" : 45.65432764125526, "type"
: "exam" }, { "score" : 64.32927049658846, "type" : "quiz" }, { "score" : 83.53933351660562,
"type" : "homework" } ] }
{ "_id" : 24, "name" : "Jesusa Rickenbacker", "scores" : [ { "score" : 86.0319702155683,
"type" : "exam" }, { "score" : 1.967495200433389, "type" : "quiz" }, { "score" :
61.10861071547914, "type" : "homework" } ] }
```

(e) Display 5<sup>th</sup> and 6<sup>th</sup> documents from the students collection.

```
> db.students.find().sort({_id:1}).skip(4).limit(2)
{ "_id" : 5, "name" : "Wilburn Spiess", "scores" : [ { "score" : 44.87186330181261, "type" :
"exam" }, { "score" : 25.72395114668016, "type" : "quiz" }, { "score" : 63.42288310628662,
"type" : "homework" } ] }
{ "_id" : 6, "name" : "Jenette Flanders", "scores" : [ { "score" : 37.32285459166097, "type" :
"exam" }, { "score" : 28.32634976913737, "type" : "quiz" }, { "score" : 81.57115318686338,
"type" : "homework" } ] }
```

(f) Display total number of documents in the students collection.

```
> db.students.find().length()
24
```

(g) Display last three documents from the students collection.

```
> db.students.find().sort({_id:-1}).limit(3)
{ "_id" : 24, "name" : "Jesusa Rickenbacker", "scores" : [ { "score" : 86.0319702155683,
"type" : "exam" }, { "score" : 1.967495200433389, "type" : "quiz" }, { "score" :
61.10861071547914, "type" : "homework" } ] }
{ "_id" : 23, "name" : "Tamika Schildgen", "scores" : [ { "score" : 45.65432764125526, "type" :
"exam" }, { "score" : 64.32927049658846, "type" : "quiz" }, { "score" : 83.53933351660562,
"type" : "homework" } ] }
{ "_id" : 22, "name" : "Margart Vitello", "scores" : [ { "score" : 75.04996547553947, "type" :
"exam" }, { "score" : 10.23046475899236, "type" : "quiz" }, { "score" : 96.72520512117761,
"type" : "homework" } ] }
```

(h) Find the ids of students whose name begins with the letter “A”.

```
> db.students.find({name:{$regex:/^A/}},{name:1})
{ "_id" : 14, "name" : "Alix Sherrill" }
{ "_id" : 1, "name" : "Aurelia Menendez" }
```

(i) Display all documents in which student name ends with the letter 'r'.

```
> db.students.find({name:{$regex:/r$/}})
{ "_id" : 12, "name" : "Quincy Danaher", "scores" : [ { "score" : 54.29841278520669, "type" :
"exam" }, { "score" : 85.61270164694737, "type" : "quiz" }, { "score" : 80.40732356118075,
"type" : "homework" } ] }
{ "_id" : 24, "name" : "Jesusa Rickenbacker", "scores" : [ { "score" : 86.0319702155683,
"type" : "exam" }, { "score" : 1.967495200433389, "type" : "quiz" }, { "score" :
61.10861071547914, "type" : "homework" } ] }
{ "_id" : 16, "name" : "Dodie Staller", "scores" : [ { "score" : 7.772386442858281, "type" :
"exam" }, { "score" : 31.84300235104542, "type" : "quiz" }, { "score" : 80.52136407989194,
"type" : "homework" } ] }
```

(j) Find all documents in student name contains ‘t’ in any position.

```
> db.students.find({name:{$regex:/t/}})
{ "_id" : 22, "name" : "Margart Vitello", "scores" : [ { "score" : 75.04996547553947, "type" :
"exam" }, { "score" : 10.23046475899236, "type" : "quiz" }, { "score" : 96.72520512117761,
"type" : "homework" } ] }
```



```
{ "_id" : 6, "name" : "Jenette Flanders", "scores" : [ { "score" : 37.32285459166097, "type" :  
"exam" }, { "score" : 28.32634976913737, "type" : "quiz" }, { "score" : 81.57115318686338,  
"type" : "homework" } ] }  
{ "_id" : 10, "name" : "Denisha Cast", "scores" : [ { "score" : 45.61876862259409, "type" :  
"exam" }, { "score" : 98.35723209418343, "type" : "quiz" }, { "score" : 55.90835657173456,  
"type" : "homework" } ] }  
{ "_id" : 16, "name" : "Dodie Staller", "scores" : [ { "score" : 7.772386442858281, "type" :  
"exam" }, { "score" : 31.84300235104542, "type" : "quiz" }, { "score" : 80.52136407989194,  
"type" : "homework" } ] }  
{ "_id" : 17, "name" : "Fletcher McConnell", "scores" : [ { "score" : 39.41011069729274,  
"type" : "exam" }, { "score" : 81.13270307809924, "type" : "quiz" }, { "score" :  
97.70116640402922, "type" : "homework" } ] }
```

## Practical – 5

**Aim:** To Demonstrate the Installation and Configuration of Single node and multimode Hadoop clusters.

1. Download JDK 8 and set it up

```
C:\Windows\system32>SET PATH=F:/SOFT/JDK_8/bin

C:\Windows\system32>java -version
java version "1.8.0_291"
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)
```

Fig 6.1 – JAVA 8

2. Download Hadoop and extract the files

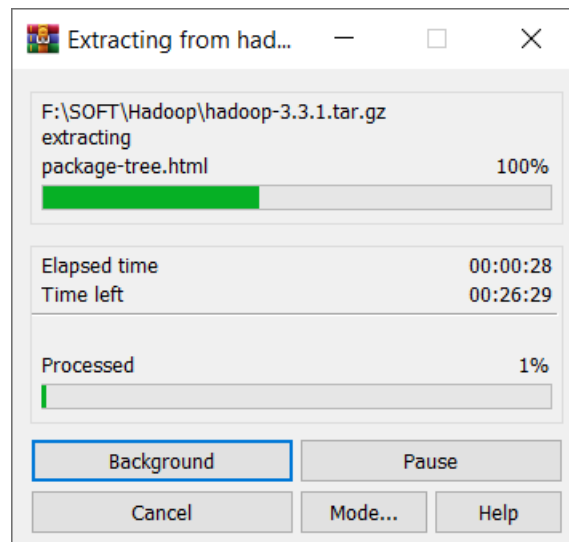


Fig 6.2 – Hadoop installation

3. Go To -> F:\SOFT\Hadoop\hadoop-3.3.1

create folder name "data"

4. Go To -> F:\SOFT\Hadoop\hadoop-3.3.1\data

create two folder name "datanode" and "namenode"

5. Go To -> F:\SOFT\Hadoop\hadoop-3.3.1\etc\hadoop

Edit files:

### **core-site.xml**

Repalce <configuration> with below

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
</property>
</configuration>
```

### **hadoop-env.cmd**

Set JAVA\_HOME

```
@rem The java implementation to use.  Required.
set JAVA_HOME=F:\SOFT\JDK_8
```

### **hdfs-site.xml**

Repalce <configuration> with below

```
<configuration>
<property>
  <name>dfs.replication</name>
  <!-- we are working on local computer so we set it to 1
       by default it is 3-->
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:///F:/SOFT/Hadoop/hadoop-3.3.1/data/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>/F:/SOFT/Hadoop/hadoop-3.3.1/data/datanode</value>
</property>
</configuration>
```

### **mapred-site.xml**

Repalce <configuration> with below

```
<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
</configuration>
```

### yarn-site.xml

Replace <configuration> with below

```
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

#### 6. Add to Path variable

F:\SOFT\Hadoop\hadoop-3.3.1\bin  
 F:\SOFT\Hadoop\hadoop-3.3.1\sbin  
 Create new Path  
 HADOOP\_HOME  
 F:\SOFT\Hadoop\hadoop-3.3.1\bin

#### 7. check installation

>>**hdfs namenode -format**

above command will give you warning that winutils.exe is missing.

so, download it and past it in the path that warning is showing

```
C:\Windows\system32>hdfs namenode -format
2021-09-05 12:37:51,177 WARN util.Shell: Did not find winutils.exe: {}
java.io.FileNotFoundException: Could not locate Hadoop executable: F:\SOFT\Hadoop\hadoop-3.3.1\bin\winutils.exe -see http
ps://wiki.apache.org/hadoop/WindowsProblems
    at org.apache.hadoop.util.Shell.getQualifiedBinInner(Shell.java:619)
    at org.apache.hadoop.util.Shell.getQualifiedBin(Shell.java:592)
    at org.apache.hadoop.util.Shell.<clinit>(Shell.java:689)
    at org.apache.hadoop.util.StringUtils.<clinit>(StringUtils.java:79)
    at org.apache.hadoop.hdfs.server.common.HdfsServerConstants$RollingUpgradeStartupOption.getAllOptionString(HdfsS
erverConstants.java:129)
    at org.apache.hadoop.hdfs.server.namenode.NameNode.<clinit>(NameNode.java:349)
2021-09-05 12:37:51,255 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = LAPTOP-B1C68170/192.168.43.217
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 3.3.1
STARTUP_MSG: classpath = F:\SOFT\Hadoop\hadoop-3.3.1\etc\hadoop;F:\SOFT\Hadoop\hadoop-3.3.1\share\hadoop\common;F:\SOF
```

Fig 6.3 – winutil.exe

>>start-dfs

after above command if your datanode get shutdown than inspect error it will cause by clusterID that is different from namenode's clusterID.

so, copy datanode's clusterID and run below command

>>hdfs namenode -format -clusterID CID-4ee3ccb3-8bdb-42e9-9edb-a7f297540a77

>>start-dfs

>>start-yarn

After above commands you have **new 4 cmd window** if any of them don't get shutdown than installation is successful

```

Administrator: Command Prompt
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>start-dfs
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>start-yarn
starting yarn daemons
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>jps
13392 Jps
15908 DataNode
12360 ResourceManager
6232 NodeManager
15292 NameNode
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>

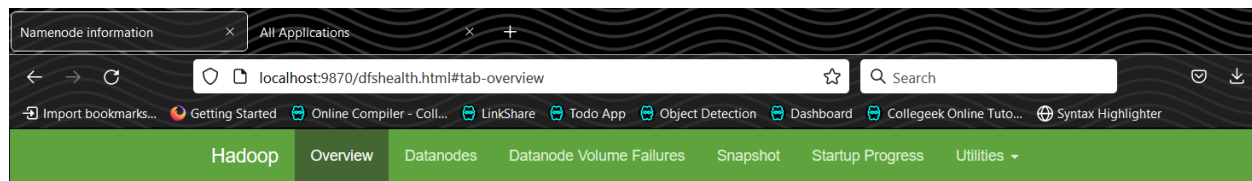
Apache Hadoop Distribution - hadoop namenode
2021-09-05 13:12:52,259 INFO blockmanagement.BlockManager: Number of over-replicated blocks = 0
2021-09-05 13:12:52,259 INFO blockmanagement.BlockManager: Number of blocks being written = 0
2021-09-05 13:12:52,260 INFO hdfs.StateChange: STATE* Replication Queue initialization scan for invalid, over- and under-replicated blocks completed in 18 msec

Apache Hadoop Distribution - hadoop datanode
2021-09-05 13:12:52,822 INFO impl.FsDatasetImpl: Time to add replicas to map for block pool BP-1186939487-192.168.43.217-1630827735994 on volume F:\SOFT\Hadoop\hadoop-3.3.1\data\datanode: 1ms
2021-09-05 13:12:52,822 INFO impl.FsDatasetImpl: Total time to add all replicas to map for block pool BP-1186939487-192.168.43.217-1630827735994: 4ms
2021-09-05 13:12:52,823 INFO checker.ThrottledAsyncChecker: Scheduling a check for F:\SOFT\Hadoop\hadoop-3.3.1\data\data

Apache Hadoop Distribution - yarn nodemanager
Sep 05, 2021 1:17:11 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory register
INFO: Registering org.apache.hadoop.yarn.server.nodemanager.webapp.JAXBContextResolver as a provider class
Sep 05, 2021 1:17:11 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFO: Initiating Jersey application, version 'Jersey: 1.19 02/11/2015 03:25 AM'

Apache Hadoop Distribution - yarn resourcemanager
eCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2021-09-05 13:17:11,648 INFO ipc.Server: Starting Socket Reader #1 for port 8031
2021-09-05 13:17:11,651 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.server.api.ResourceTrackerPB to the server
2021-09-05 13:17:11,652 INFO ipc.Server: IPC Server Responder: starting
2021-09-05 13:17:11,653 INFO ipc.Server: IPC Server listener on 8031: starting
2021-09-05 13:17:11,663 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2021-09-05 13:17:11,677 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queueCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2021-09-05 13:17:11,686 INFO ipc.Server: Starting Socket Reader #1 for port 8030
2021-09-05 13:17:11,696 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.api.ApplicationMasterProtocolPB to the server
2021-09-05 13:17:11,697 INFO ipc.Server: IPC Server Responder: starting
  
```

Fig 6.4 – Hadoop single node



## Overview 'localhost:9000' (✓active)

Started:	Sun Sep 05 13:12:50 +0530 2021
Version:	3.3.1, ra3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled:	Tue Jun 15 10:43:00 +0530 2021 by ubuntu from (HEAD detached at release-3.3.1-RC3)
Cluster ID:	CID-4ee3ccb3-8bdb-42e9-9edb-a7f297540a77
Block Pool ID:	BP-1186939487-192.168.43.217-1630827735994

## Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 78.88 MB of 186.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 49.27 MB of 51.4 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	200 GB
Configured Remote Capacity:	0 B

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
0	0	0	0	0

**Cluster Nodes Metrics**

Active Nodes	Decommissioning Nodes	Decom
1	0	0

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Mir
Capacity Scheduler	[memory-mb (unit=M), vcores]	<memory:1024, vCores

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	Finis
Showing 0 to 0 of 0 entries									

Fig 6.5 & Fig 6.6 – Hadoop web interface

8. close the hadoop

```
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>start-dfs

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>start-yarn
starting yarn daemons

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>jps
13392 Jps
15908 DataNode
12360 ResourceManager
6232 NodeManager
15292 NameNode

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>stop-dfs
SUCCESS: Sent termination signal to the process with PID 2544.
SUCCESS: Sent termination signal to the process with PID 2056.

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>stop-yarn
stopping yarn daemons
SUCCESS: Sent termination signal to the process with PID 14988.
SUCCESS: Sent termination signal to the process with PID 13560.

INFO: No tasks running with the specified criteria.

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>
```

Fig 6.9 – closing of hadoop

## Multinode Hadoop Cluster

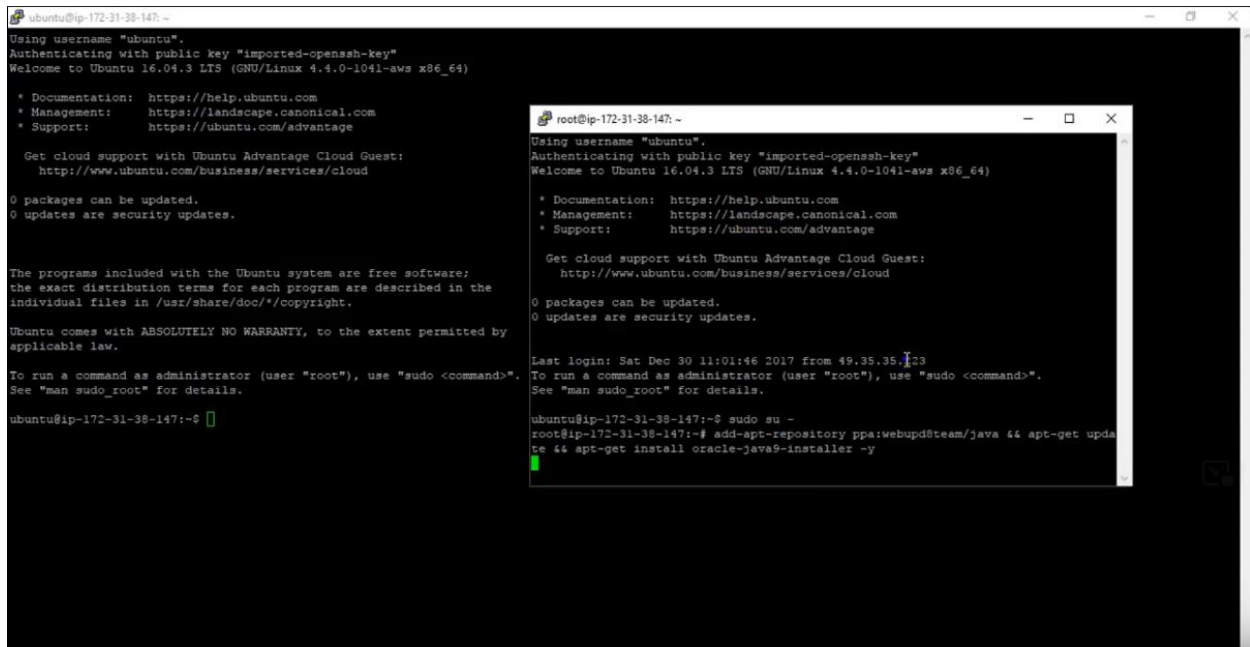
Strat ec2 instance and install java and hadoop

Download the latest version of JAVA

add-apt-repository ppa:webupd8team/java && apt-get install oracle-java9-installer -y

Download the latest version of Hadoop Software.

wget <http://apache.mirror.gtcomm.net/hadoop/common/hadoop-2.9.0/hadoop-2.9.0.tar.gz>



```

ubuntu@ip-172-31-38-147: ~
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1041-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-38-147:~$

root@ip-172-31-38-147: ~
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1041-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

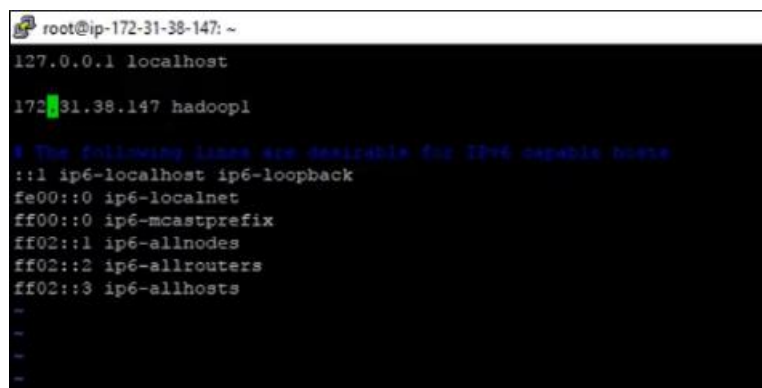
0 packages can be updated.
0 updates are security updates.

Last login: Sat Dec 30 11:01:46 2017 from 49.35.35.123
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-38-147:~$ sudo su -
root@ip-172-31-38-147:~$ add-apt-repository ppa:webupd8team/java && apt-get update && apt-get install oracle-java8-installer -y

```

Fig 6.10 install java and hadoop



```

root@ip-172-31-38-147: ~
127.0.0.1 localhost

172.31.38.147 hadoop1

# The following lines are available for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

```

Fig 6.11 setup ip of instance

Create the required Hadoop. Users.

```

addgroup hadoop
adduser --ingroup hadoop yarn
adduser --ingroup hadoop hdfs
adduser --ingroup hadoop mapred

```

Create the required directories.

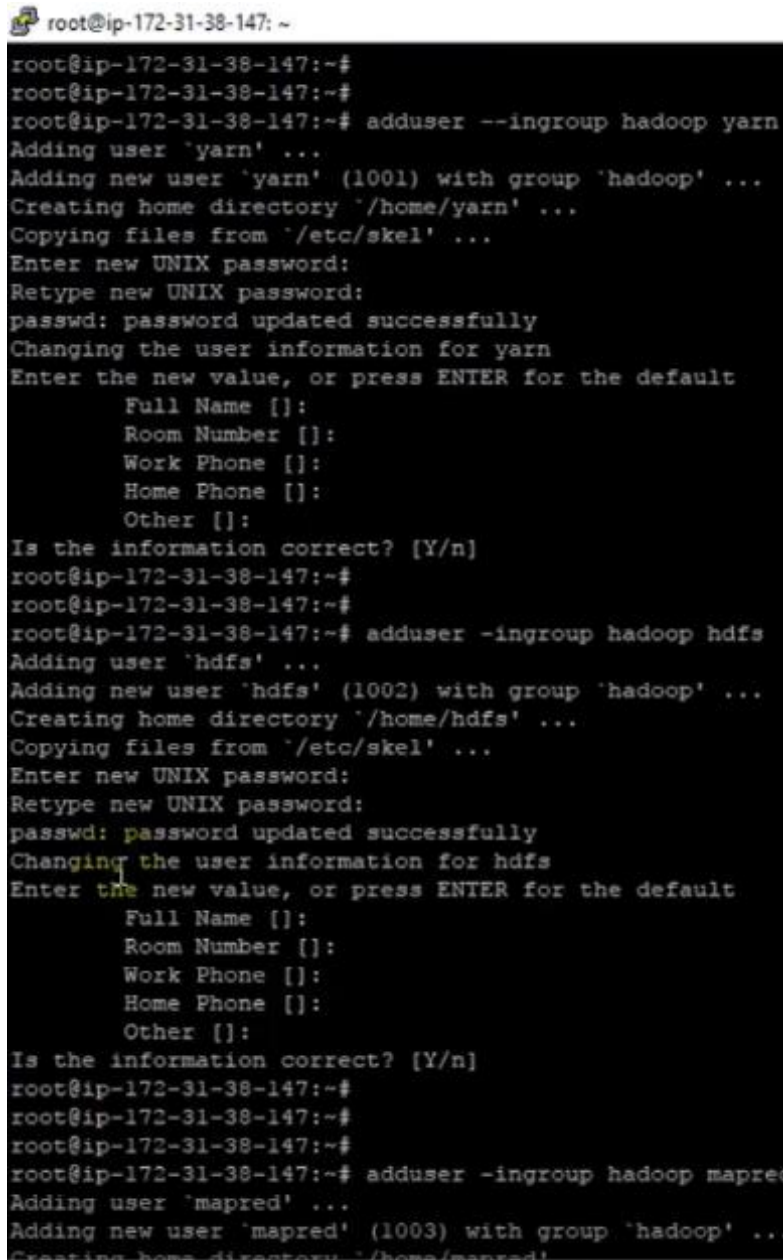
```

mkdir -p /opt/yarn/hadoop/
mkdir -p /var/data/hadoop/hdfs/nn
mkdir -p /var/data/hadoop/hdfs/snn

```



```
mkdir -p /var/data/hadoop/hdfs/dn
mkdir -p /var/data/hadoop/hdfs/tmp
chown -R hdfs:hadoop /var/data/
mkdir -p /var/log/hadoop/logs
chmod -R 777 /var/log/hadoop
chown -R yarn:hadoop /var/log/hadoop
chown -R hdfs:hadoop /opt/yarn/hadoop/
chmod -R 777 /opt/yarn/hadoop/
```



```
root@ip-172-31-38-147: ~
root@ip-172-31-38-147:~#
root@ip-172-31-38-147:~# adduser --ingroup hadoop yarn
Adding user 'yarn' ...
Adding new user 'yarn' (1001) with group 'hadoop' ...
Creating home directory '/home/yarn' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for yarn
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
root@ip-172-31-38-147:~#
root@ip-172-31-38-147:~#
root@ip-172-31-38-147:~# adduser -ingroup hadoop hdfs
Adding user 'hdfs' ...
Adding new user 'hdfs' (1002) with group 'hadoop' ...
Creating home directory '/home/hdfs' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hdfs
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
root@ip-172-31-38-147:~#
root@ip-172-31-38-147:~#
root@ip-172-31-38-147:~#
root@ip-172-31-38-147:~# adduser -ingroup hadoop mapred
Adding user 'mapred' ...
Adding new user 'mapred' (1003) with group 'hadoop' ...
Creating home directory '/home/mapred'
```

Fig 6.12 setup hadoop users

Edit all files that we have edited during single node hadoop cluster setup

```
hdfs@hadoop2: /opt/yarn/hadoop/etc/hadoop
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

  <property>
    <name>fs.default.name</name>
    <value>hdfs://hadoop1:9000</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/var/data/hadoop/hdfs/nn</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/var/data/hadoop/hdfs/dn</value>
  </property>
  <property>
    <name>fs.checkpoint.dir</name>
    <value>file:/var/data/hadoop/hdfs/snn</value>
  </property>
  <property>
    <name>fs.checkpoint.edits.dir</name>
```

Fig 6.13 edit all xml files

After start datanode and namenode

Then yarn service by

Start-dfs

Start-yarn

And Hadoop multimode setup is done.

```

hdfs@hadoop1: /opt/yarn/hadoop/sbin
17/12/30 12:10:59 INFO namenode.FSImageFormatProtobuf: Saving image file /var/data/hadoop/hdfs/nn/current/fsimage.ckpt_00000000000000000000 using no compression
17/12/30 12:10:59 INFO namenode.FSImageFormatProtobuf: Image file /var/data/hadoop/hdfs/nn/current/fsimage.ckpt_00000000000000000000 of size 321 bytes saved in 0 seconds
17/12/30 12:10:59 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
17/12/30 12:10:59 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at hadoop1/172.31.38.147
*****/
hdfs@hadoop1: /opt/yarn/hadoop/sbin$ hsbins
hdfs@hadoop1: /opt/yarn/hadoop/sbin$ ./st
start-all.cmd      start-dfs.cmd      start-yarn.cmd      stop-all.sh        stop-dfs.sh        stop-yarn.sh
start-all.sh       start-dfs.sh       start-yarn.sh       stop-balancer.sh    stop-secure-dns.sh
start-balancer.sh   start-secure-dns.sh stop-all.cmd        stop-dfs.cmd        stop-yarn.cmd
hdfs@hadoop1: /opt/yarn/hadoop/sbin$ ./st
start-all.cmd      start-dfs.cmd      start-yarn.cmd      stop-all.sh        stop-dfs.sh        stop-yarn.sh
start-all.sh       start-dfs.sh       start-yarn.sh       stop-balancer.sh    stop-secure-dns.sh
start-balancer.sh   start-secure-dns.sh stop-all.cmd        stop-dfs.cmd        stop-yarn.cmd
hdfs@hadoop1: /opt/yarn/hadoop/sbin$ ./st
start-all.cmd      start-dfs.cmd      start-yarn.cmd      stop-all.sh        stop-dfs.sh        stop-yarn.sh
start-all.sh       start-dfs.sh       start-yarn.sh       stop-balancer.sh    stop-secure-dns.sh
start-balancer.sh   start-secure-dns.sh stop-all.cmd        stop-dfs.cmd        stop-yarn.cmd
hdfs@hadoop1: /opt/yarn/hadoop/sbin$ ./start-dfs.sh
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/opt/yarn/hadoop/share/hadoop/common/lib/hadoop-auth-2.9.0.jar)
to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Starting namenodes on [hadoop1]
hadoop1: starting namenode, logging to /var/log/hadoop/logs/hdfs/hadoop-hdfs-namenode-hadoop1.out
hadoop3: starting datanode, logging to /var/log/hadoop/logs/hdfs/hadoop-hdfs-datanode-hadoop3.out
hadoop2: starting datanode, logging to /var/log/hadoop/logs/hdfs/hadoop-hdfs-datanode-hadoop2.out
hadoop1: starting datanode, logging to /var/log/hadoop/logs/hdfs/hadoop-hdfs-datanode-hadoop1.out
Starting secondary namenodes [hadoop2]
hadoop2: starting secondarynamenode, logging to /var/log/hadoop/logs/hdfs/hadoop-hdfs-secondarynamenode-hadoop2.out
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/opt/yarn/hadoop/share/hadoop/common/lib/hadoop-auth-2.9.0.jar)
to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
hdfs@hadoop1: /opt/yarn/hadoop/sbin$ jps

```

Fig 6.14 Hadoop multinode cluster setup

## Practical – 6

**Aim:** To Develop a Map Reduce program for Word count for Hadoop cluster.

First we will create **input** directory to store input file **data.txt** that contains **some words** and we will use example program of **mapreduce.jar** to count word in data.txt file. Output will be storing in **out** directory.

HDFS: Hadoop Distributed File System

YARN: Yet Another Resource Negotiator

```
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>start-dfs

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>start-yarn
starting yarn daemons

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>jps
10960 ResourceManager
7904 Jps
10916 NameNode
13336 NodeManager
10940 DataNode

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop fs -mkdir /input

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop fs -put "F:\work of pro\SEM7\3170722-Big
Data\data.txt" /input

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop fs -ls /input
Found 1 items
-rw-r--r--  1 gabur supergroup    110 2021-09-05 15:04 /input/data.txt

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop fs -cat /input/data.txt
big data
hadoop
hi
hello
good
big data
big data
hadoop
hello
good
bad
```

```

morning
hadoop
big data
hi
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop jar F:\SOFT\Hadoop\hadoop-
3.3.1\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.3.1.jar
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the words in the
input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of
the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets
  multifielwc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in the input
files.
  wordmedian: A map/reduce program that counts the median length of the words in the input
files.
  wordstandarddeviation: A map/reduce program that counts the standard deviation of the
length of the words in the input files.

F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop jar F:\SOFT\Hadoop\hadoop-
3.3.1\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.3.1.jar wordcount /input /out
2021-09-05 15:15:28,223 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting
to ResourceManager at /0.0.0.0:8032
2021-09-05 15:15:29,775 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding
for path: /tmp/hadoop-yarn/staging/gabur/.staging/job_1630834150687_0001
2021-09-05 15:15:30,184 INFO input.FileInputFormat: Total input files to process : 1
2021-09-05 15:15:30,569 INFO mapreduce.JobSubmitter: number of splits:1
2021-09-05 15:15:30,827 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1630834150687_0001

```

```
2021-09-05 15:15:30,828 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-09-05 15:15:31,148 INFO conf.Configuration: resource-types.xml not found
2021-09-05 15:15:31,149 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-09-05 15:15:31,719 INFO impl.YarnClientImpl: Submitted application
application_1630834150687_0001
2021-09-05 15:15:31,787 INFO mapreduce.Job: The url to track the job: http://LAPTOP-
B1C68170:8088/proxy/application_1630834150687_0001/
2021-09-05 15:15:31,789 INFO mapreduce.Job: Running job: job_1630834150687_0001
2021-09-05 15:15:44,011 INFO mapreduce.Job: Job job_1630834150687_0001 running in
uber mode : false
2021-09-05 15:15:44,013 INFO mapreduce.Job: map 0% reduce 0%
2021-09-05 15:15:50,102 INFO mapreduce.Job: map 100% reduce 0%
2021-09-05 15:15:57,181 INFO mapreduce.Job: map 100% reduce 100%
2021-09-05 15:15:58,204 INFO mapreduce.Job: Job job_1630834150687_0001 completed
successfully
2021-09-05 15:15:58,363 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=96
    FILE: Number of bytes written=548319
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=211
    HDFS: Number of bytes written=58
    HDFS: Number of read operations=8
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0
  Job Counters
    Launched map tasks=1
    Launched reduce tasks=1
    Data-local map tasks=1
    Total time spent by all maps in occupied slots (ms)=4236
    Total time spent by all reduces in occupied slots (ms)=4756
    Total time spent by all map tasks (ms)=4236
    Total time spent by all reduce tasks (ms)=4756
    Total vcore-milliseconds taken by all map tasks=4236
    Total vcore-milliseconds taken by all reduce tasks=4756
    Total megabyte-milliseconds taken by all map tasks=4337664
    Total megabyte-milliseconds taken by all reduce tasks=4870144
  Map-Reduce Framework
    Map input records=15
    Map output records=19
    Map output bytes=173
    Map output materialized bytes=96
    Input split bytes=101
```

```
Combine input records=19
Combine output records=8
Reduce input groups=8
Reduce shuffle bytes=96
Reduce input records=8
Reduce output records=8
Spilled Records=16
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=109
CPU time spent (ms)=1855
Physical memory (bytes) snapshot=496582656
Virtual memory (bytes) snapshot=743444480
Total committed heap usage (bytes)=347602944
Peak Map Physical memory (bytes)=295948288
Peak Map Virtual memory (bytes)=414810112
Peak Reduce Physical memory (bytes)=200634368
Peak Reduce Virtual memory (bytes)=328728576
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=110
File Output Format Counters
  Bytes Written=58
```

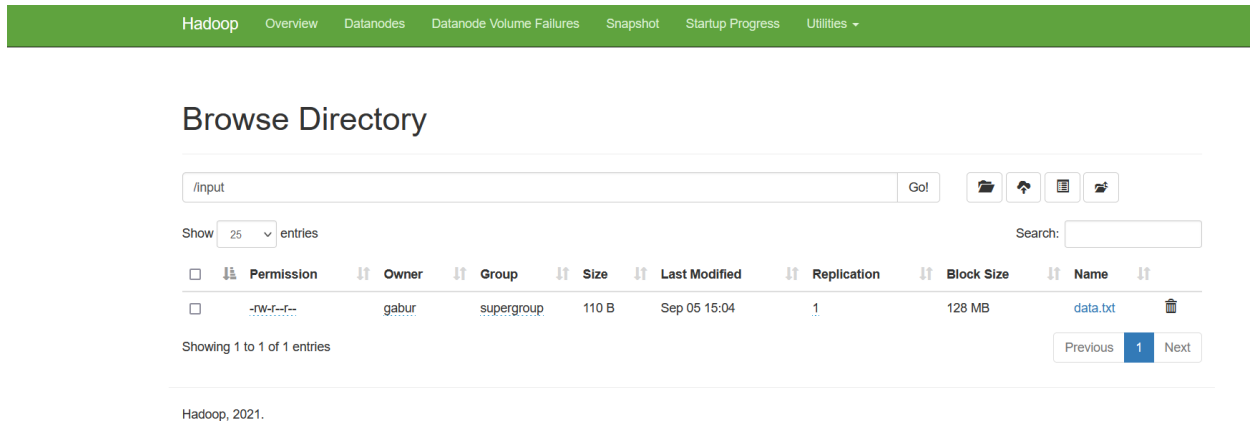
```
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop fs -ls /out
```

```
Found 2 items
```

```
-rw-r--r--  1 gabur supergroup      0 2021-09-05 15:15 /out/_SUCCESS
-rw-r--r--  1 gabur supergroup    58 2021-09-05 15:15 /out/part-r-00000
```

```
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>hadoop fs -cat /out/*
```

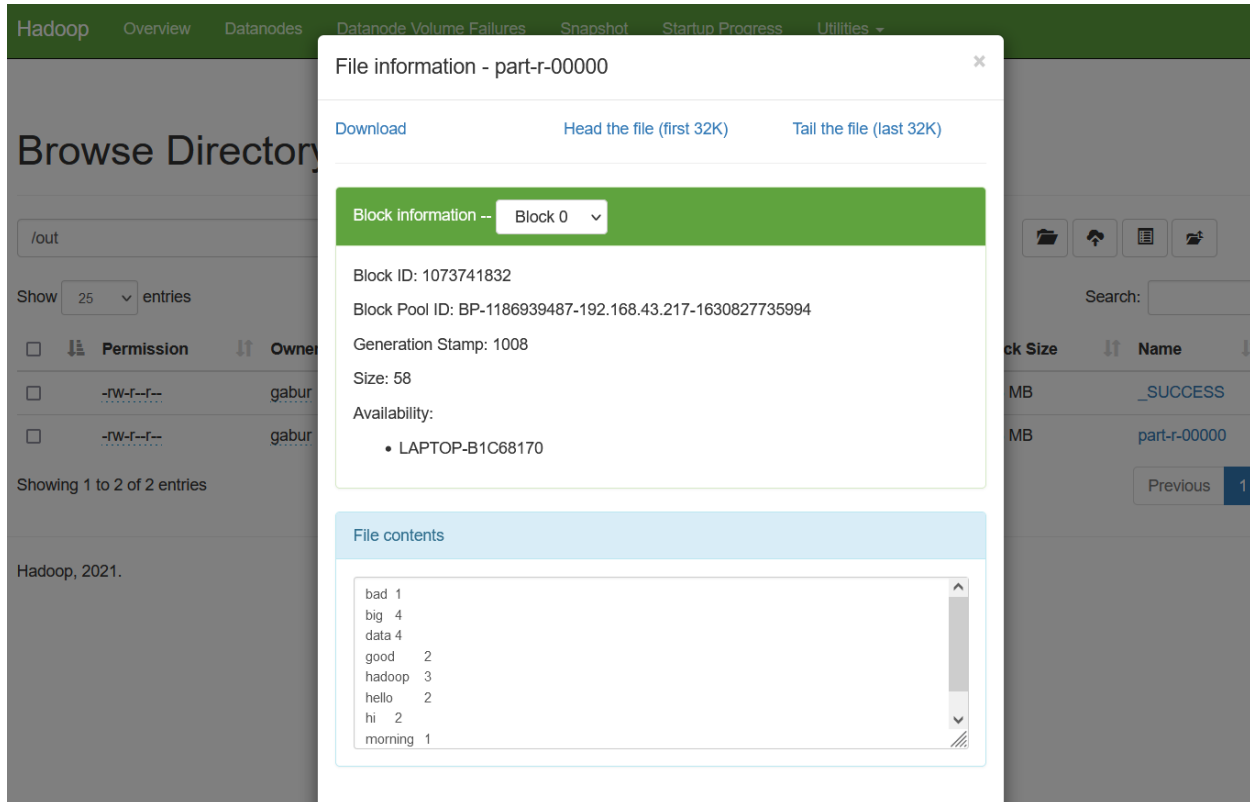
```
bad  1
big  4
data 4
good 2
hadoop 3
hello 2
hi  2
morning 1
```



The screenshot shows the Hadoop web interface's 'Browse Directory' page. The path '/input' is entered in the search bar. A table lists the contents of the directory, showing a single file named 'data.txt' with a size of 110 B, last modified on Sep 05 15:04, and a replication factor of 1. The file is owned by 'gabur' and belongs to the 'supergroup'. The interface includes navigation links like 'Previous', '1', and 'Next'.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	gabur	supergroup	110 B	Sep 05 15:04	1	128 MB	<a href="#">data.txt</a>

Fig 8.1 data.txt



The screenshot shows the Hadoop web interface with a 'File information - part-r-00000' overlay. The overlay displays details for a specific block (Block 0) of the file 'part-r-00000'. The file information includes the block ID, block pool ID, generation stamp, size, and availability. The file contents are also displayed in a scrollable text area.

File information - part-r-00000

Download Head the file (first 32K) Tail the file (last 32K)

Block information - Block 0

Block ID: 1073741832  
Block Pool ID: BP-1186939487-192.168.43.217-1630827735994  
Generation Stamp: 1008  
Size: 58  
Availability:  
• LAPTOP-B1C68170

File contents

```
bad 1
big 4
data 4
good 2
hadoop 3
hello 2
hi 2
morning 1
```

Fig 8.2 output of word count



## Practical – 7

**Aim:** To study stream mining using case study approach.

### Introduction:

**Data Stream Mining** (also known as stream learning) is the process of extracting knowledge structures from continuous, rapid data records. A data stream is an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities.

In many data stream mining applications, **the goal** is to predict the class or value of new instances in the data stream given some knowledge about the class membership or values of previous instances in the data stream.

Machine learning techniques can be used to learn this prediction task from labeled examples in an automated fashion. Often, concepts from the field of incremental learning are applied to cope with structural changes, on-line learning and real-time demands. In many applications, especially operating within non-stationary environments, the distribution underlying the instances or the rules underlying their labeling may change over time, i.e. the goal of the prediction, **the class to be predicted or the target value to be predicted, may change over time**. This problem is referred to as **concept drift**. Detecting concept drift is a **central issue** to data stream mining. Other challenges that arise when applying machine learning to streaming data include: **partially and delayed labeled data, recovery from concept drifts, and temporal dependencies**.

Examples of data streams include computer network traffic, phone conversations, ATM transactions, web searches, and sensor data. **Data stream mining** can be considered a subfield of data mining, machine learning, and knowledge discovery.

### Data-Stream-Management System:

We can view a stream processor as a kind of data-management system, the high-level organization of which is suggested in Fig. Any number of streams can enter the system.

Each stream can provide elements at its own schedule; they need not have the same data rates or data types, and the time between elements of one stream need not be uniform. The fact that the rate of arrival of stream elements is not under the control of the system distinguishes stream processing from the processing of data that goes on within a database-management system.

The latter system controls the rate at which data is read from the disk, and therefore never has to worry about data getting lost as it attempts to execute queries. Streams may be archived in a large archival store, but we assume it is not possible to answer queries from the archival store. It could be examined only under special circumstances using time-consuming retrieval processes.

There is also a working store, into which summaries or parts of streams may be placed, and which can be used for answering queries. The working store might be disk, or it might be main memory, depending on how fast we need to process queries. But either way, it is of sufficiently limited capacity that it cannot store all the data from all the streams.

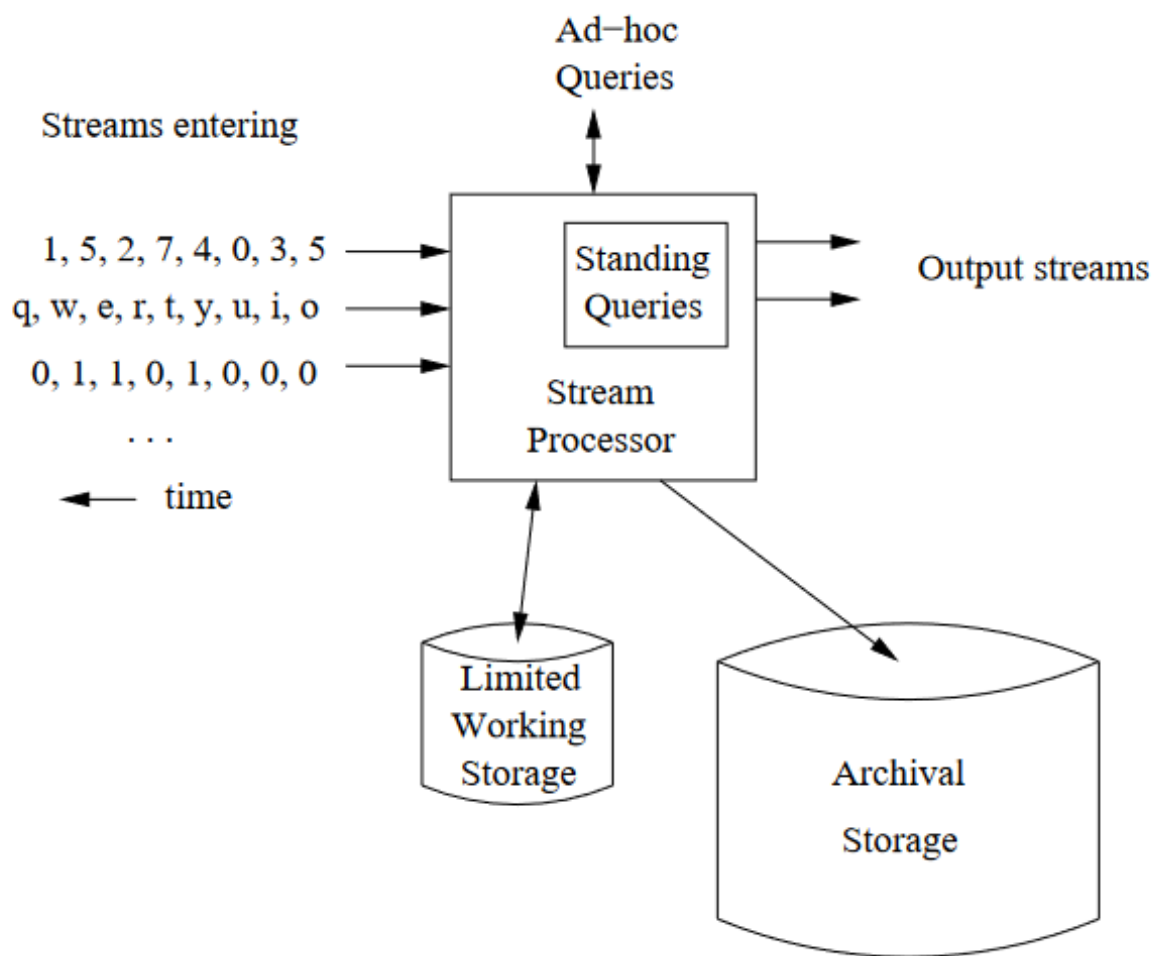


Fig 7.1 data-stream-management system

## Examples of Stream Sources:

### Sensor Data

Imagine a temperature sensor bobbing about in the ocean, sending back to a base station a reading of the surface temperature each hour. The data produced by this sensor is a stream of real numbers. It is not a very interesting stream, since the data rate is so low. It would not stress modern technology, and the entire stream could be kept in main memory, essentially forever. Now, give the sensor a GPS unit, and let it report surface height instead of temperature.

The surface height varies quite rapidly compared with temperature, **so we might have the sensor send back a reading every tenth of a second. If it sends a 4-byte real number each time, then it produces 3.5 megabytes per day.** It will still take some time to fill up main memory, let alone a single disk. But one sensor might not be that interesting.

To learn something about ocean behavior, we might want to deploy a million sensors, each sending back a stream, at the rate of ten per second. A million sensors isn't very many; there would be one for every 150 square miles of ocean. Now we have 3.5 terabytes arriving every day, and we definitely need to think about what can be kept in working storage and what can only be archived.

### Image Data

Satellites often send down to earth streams consisting of many terabytes of images per day. Surveillance cameras produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at intervals like one second. London is said to have six million such cameras, each producing a stream.

### Internet and Web Traffic

A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its outputs. Normally, the job of the switch is to transmit data and not to retain it or query it. But there is a tendency to put more capability into the switch, e.g., the ability to detect denial-of-service attacks or the ability to reroute packets based on information about congestion in the network. Web sites receive streams of various types. For example, Google receives several hundred million search queries per day. Yahoo! accepts billions of "clicks" per day on its various sites. Many interesting things can be learned from these streams.

## Data stream mining methods:

For extracting knowledge or patterns from data streams, it is crucial to develop methods that analyze and process streams of data in multidimensional, multi-level, single pass and online manner. These methods should not be limited to data streams only, because they are also needed when we have large volume of data. Moreover, because of the limitation of data streams, the proposed methods are based on statistic, calculation and complexity theories. For example, by using summarization techniques that are derived from statistic science, we can confront with memory limitation. In addition, some of the techniques in computation theory can be used for implementing time and space efficient algorithms. By using these techniques we can also use common data mining approaches by enforcing some changes in data streams. Some solutions have been proposed based on data stream mining problems and challenges.

These solutions can be categorized to data-based and task-based solutions. This classification is depicted in Fig 7.2. Data-based techniques refer to summarizing the whole dataset or choosing a subset of the incoming stream to be analyzed. Sampling, load and sketching techniques represent the former one. Synopsis data structures and aggregation represent the later one. Task-based techniques are those methods that modify existing techniques or invent new ones in order to address the computational challenges of data stream processing. Approximation algorithms, sliding window and algorithm output granularity represent this category.

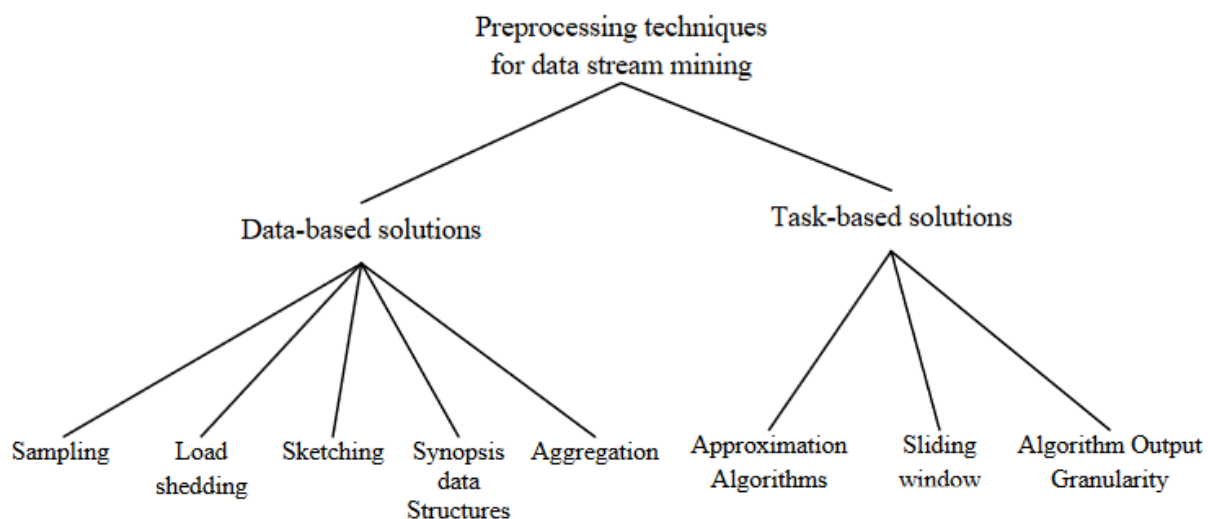


Fig 7.2 classification of data stream methods

**Sampling** refers to the process of probabilistic choice of a data item to be processed or not. The problem with using sampling in the context of data stream analysis is the unknown dataset size. Thus the treatment of data stream should follow a special analysis to find the error bounds. Another problem with sampling is that it would be important to check for anomalies for surveillance analysis as an application in mining data streams. Sampling may not be the right choice for such an application. Sampling also does not address the problem of fluctuating data rates. It would be worth investigating the relationship among the three parameters: data rate, sampling rate and error bounds.

**Load shedding** refers to the process of dropping a sequence of data streams. Load shedding has been used successfully in querying data streams. It has the same problems of sampling. Load shedding is difficult to be used with mining algorithms because it drops chunks of data streams that could be used in the structuring of the generated models or it might represent a pattern of interest in time series analysis.

**Sketching** is the process of randomly project a subset of the features. It is the process of vertically sample the incoming stream. Sketching has been applied in comparing different data streams and in aggregate queries. The major drawback of sketching is that of accuracy. It is hard to use it in the context of data stream mining.

**Creating synopsis** of data refers to the process of applying summarization techniques that are capable of summarizing the incoming stream for further analysis. Wavelet analysis, histograms, quantiles and frequency moments have been proposed as synopsis data structures. Since synopsis of data does not represent all the characteristics of the dataset, approximate answers are produced when using such data structures.

The process in which the input stream is represented in a summarized form is called **Aggregation**. This aggregate data can be used in data mining algorithms. The main problem of this method is that highly fluctuating data distributions reduce the method's efficiency.

**Approximation algorithms** have their roots in algorithm design. It is concerned with design algorithms for computationally hard problems. These algorithms can result in an approximate solution with error bounds. The idea is that mining algorithms are considered hard computational problems given its features of continuity and speed and the generating environment that is featured by being resource constrained. Approximation algorithms have attracted researchers as a direct solution to data stream mining problems. However, the problem of data rates with regard to the available resources could not be solved using approximation algorithms. Other tools should be used along with these algorithms in order to adapt to the available resources. Approximation algorithms have been used in.

The inspiration behind **sliding window** is that the user is more concerned with the analysis of most recent data streams. Thus the detailed analysis is done over the most recent data items and summarized versions of the old ones.

The **algorithm output granularity (AOG)** introduces the first resource-aware data analysis approach that can cope with fluctuating very high data rates according to the available memory and the processing speed represented in time constraints. The AOG performs the local data analysis on a resource constrained device that generates or receive streams of information. AOG has three main stages. Mining followed by adaptation to resources and data stream rates represent the first two stages. Merging the generated knowledge structures when running out of memory represents the last stage. AOG has been used in clustering, classification and frequency counting. The function of the AOG algorithm is depicted in

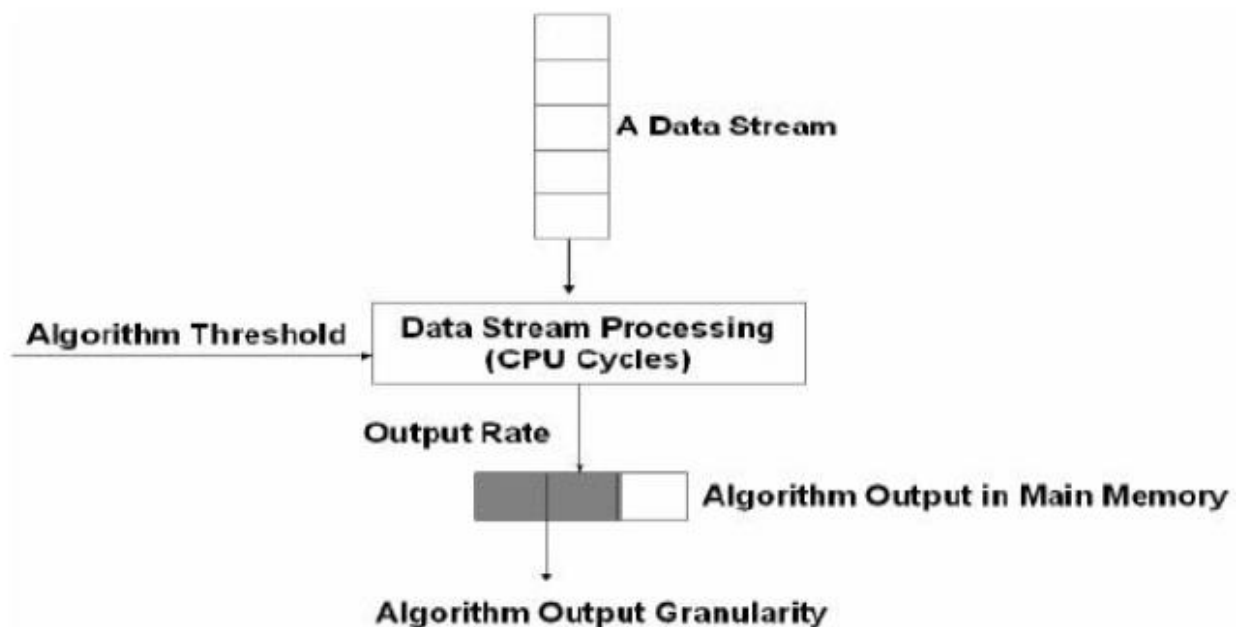


Fig 7.3 The AOG algorithm

## **Classification of data stream challenges:**

There are different challenges in data stream mining that cause many research issues in this field. Regarding to data stream requirements, developing stream mining algorithms is needed more studying than traditional mining methods. We can classify stream mining challenges in 5 categories;

Irregular rate of arrival and variant data

Arrival rate over time,

Quality of mining results,

Bounded memory size and huge amount of data streams,

Limited resources, e.g., memory space and computation power and to facilitate data analysis and take a quick decision for users. In the following each of them will be described.

One of the most important issues in data stream mining is optimization of memory space consumed by the mining algorithm. Memory management is a main challenge in stream processing because many real data streams have irregular arrival rate and variation of data arrival rate over time. In many applications like sensor networks, stream mining algorithms with high memory cost is not applicable. Therefore, it is necessary to develop summarizing techniques for collecting valuable information from data streams. Data pre-processing is an important and time consuming phase in the knowledge discovery process and must be taken into consideration when mining data streams. Designing a light-weight preprocessing techniques that can guarantee quality of the mining results is crucial. The challenge here is to automate such a process and integrate it with the mining techniques.

By considering the size of memory and the huge amount of data stream that continuously arrive to the system, it is needed to have a compact data structure to store, update and retrieve the collected information. Without such a data structure, the efficiency of mining algorithm will largely decrease. Even if we store the information in disks, the additional I/O operations will increase the processing time.

While it is impossible to rescan the entire input data, incremental maintaining of data structure is indispensable. Furthermore, novel indexing, storage and querying techniques are required to manage continuous and changing flow of data streams. It is crucial to consider the limited resources such as memory space and computation power for reaching accurate estimates in data streams mining. If stream data mining algorithms consume the available resources without any consideration, the accuracy of their results would decrease dramatically. In several papers this issue is discussed and their solutions for resource-aware mining are proposed. One of the proposed solutions is AOG which use a control parameter to control its output rate according to memory, time constraints and data stream rate. Also in another algorithm is proposed that not

only reduces the memory required for data storage but also retains good approximation given limited resources like memory space and computation power.

Visualization is a powerful way to facilitate data analysis. Absence of suitable tools for visualization of mining result makes many problems in data analysis and quick decision making by user. This challenge still is a research issue that one of the proposed approaches is intelligent monitoring.

Research Issues	Challenges	Approaches
Memory management	Fluctuated and irregular data arrival rate and variant data arrival rate over time	Summarizing techniques
Data preprocessing	Quality of mining results and automation of preprocessing techniques	Light-weight preprocessing techniques
Compact data structure	Limited memory size and large volume of data streams	Incremental maintaining of data structure, novel indexing, storage and querying techniques
Resource aware	Limited resources like storage and computation capabilities	AOG and [31]
Visualization of results	Problems in data analysis and quick decision making by user	Still is a research issue (one of the proposed approaches is: intelligent monitoring)

Fig 7.3 Classification of data stream mining challenges



## Practical – 8

**Aim:** To demonstrate Pig and Hive SQL queries using various operators.

```
F:\SOFT\pig\pig-0.17.0\bin>pig -x local
2021-09-27 19:02:27,629 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2021-09-27 19:02:27,630 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2021-09-27 19:02:27,958 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0
(r1797386) compiled Jun 02 2017, 15:41:58
2021-09-27 19:02:27,959 [main] INFO org.apache.pig.Main - Logging error messages to:
F:\SOFT\Hadoop\hadoop-3.3.1\logs\pig_1632749547955.log
2021-09-27 19:02:27,980 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file
C:\Users\gabur/.pigbootup not found
2021-09-27 19:02:28,130 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2021-09-27 19:02:28,132 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop
file system at: file:///
2021-09-27 19:02:28,428 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2021-09-27 19:02:28,456 [main] INFO org.apache.pig.PigServer - Pig Script ID for the
session: PIG-default-ba48df0c-6cea-44d2-9782-69f8b73cce92
2021-09-27 19:02:28,456 [main] WARN org.apache.pig.PigServer - ATS is disabled since
yarn.timeline-service.enabled set to false

grunt> agriculture= LOAD 'F:/work of pro/SEM7/3170722-Big
Data/archive/crop_production.csv' using PigStorage(',') as ( State_Name:chararray ,
District_Name:chararray , Crop_Year:int , Season:chararray , Crop:chararray , Area:int ,
Production:int );
2021-09-27 19:03:34,719 [main] INFO org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum

grunt> describe agriculture;
agriculture: {State_Name: chararray,District_Name: chararray,Crop_Year: int,Season:
chararray,Crop: chararray,Area: int,Production: int}

grunt> statewisecrop = GROUP agriculture BY State_Name;

grunt> desc
desc      describe

grunt> describe statewisecrop;

statewisecrop: {group: chararray,agriculture: {(State_Name: chararray,District_Name:
chararray,Crop_Year: int,Season: chararray,Crop: chararray,Area: int,Production: int)}}
```

```

grunt> STORE statewisecrop INTO 'F:/work of pro/SEM7/3170722-Big
Data/archive/statewiseinfo';
HadoopVersion  PigVersion  UserId StartedAt   FinishedAt   Features
3.3.1  0.17.0  gabur  2021-09-27 19:06:08   2021-09-27 19:06:13   GROUP_BY

Success!

Job Stats (time in seconds):
JobId  Maps  Reduces MaxMapTime  MinMapTime  AvgMapTime
MedianMapTime  MaxReduceTime  MinReduceTime
      AvgReduceTime  MedianReducetime  Alias  Feature Outputs
job_local2107346029_0001    1    1    n/a    n/a    n/a    n/a    n/a    n/a    n/a    n/a
agriculture,statewisecrop  GROUP_BY    F:/work of pro/SEM7/3170722-Big
Data/archive/statewiseinfo,

Input(s):
Successfully read 246092 records from: "F:/work of pro/SEM7/3170722-Big
Data/archive/crop_production.csv"

Output(s):
Successfully stored 34 records in: "F:/work of pro/SEM7/3170722-Big
Data/archive/statewiseinfo"

Counters:
Total records written : 34
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local2107346029_0001

grunt> cropinfo = FOREACH( GROUP agriculture BY Crop )
>> GENERATE group AS Crop, SUM(agriculture.Area) as AreaPerCrop ,
>> SUM(agriculture.Production) as ProductionPerCrop;
2021-09-27 19:10:10,159 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum

grunt> describe cropinfo;
cropinfo: {Crop: chararray,AreaPerCrop: long,ProductionPerCrop: long}

```

```
F:\SOFT\hive\apache-hive-2.3.9-bin\bin>hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/F:/SOFT/hive/apache-hive-2.3.9-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/F:/SOFT/Hadoop/hadoop-3.3.1/share/hadoop/common/lib/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2021-09-27T16:49:32,538 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Found configuration file file:/F:/SOFT/hive/apache-hive-2.3.9-bin/conf/hive-site.xml

hive> create database demo;
OK
Time taken: 6.606 seconds

hive> show databases;
OK
default
demo
Time taken: 0.179 seconds, Fetched: 2 row(s)

hive> create table demo.employee (Id int, Name string , Salary float);
OK
Time taken: 2.333 seconds

hive> describe demo.employee;
OK
id                int
name              string
salary           float
Time taken: 0.403 seconds, Fetched: 3 row(s)

hive> INSERT INTO TABLE employee VALUES (1, 'Gaurav' ,30000),(2, 'Aryan' ,20000),(3, 'Vishal' ,40000),(4, 'Siddharth' ,60000),(5, 'Henry' ,25000),(6, 'Chirag' ,60000),(7, 'Lisa' ,25000),(8, 'Ronit' ,20000);
Loading data to table demo.employee
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Cumulative CPU: 3.061 sec  HDFS Read: 4645 HDFS Write: 204
SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 61 msec
OK
Time taken: 28.138 seconds

hive> select * from employee;
```

OK

```
1  Gaurav 30000.0
2  Aryan  20000.0
3  Vishal 40000.0
4  Siddharth 60000.0
5  Henry  25000.0
6  Chirag 60000.0
7  Lisa   25000.0
8  Ronit  20000.0
```

Time taken: 0.839 seconds, Fetched: 8 row(s)

hive> select id, name, (salary \* 10) /100 from employee;

OK

```
1  Gaurav 3000.0
2  Aryan  2000.0
3  Vishal 4000.0
4  Siddharth 6000.0
5  Henry  2500.0
6  Chirag 6000.0
7  Lisa   2500.0
8  Ronit  2000.0
```

Time taken: 0.938 seconds, Fetched: 8 row(s)

hive> select \* from employee where salary >= 25000;

OK

```
1  Gaurav 30000.0
3  Vishal 40000.0
4  Siddharth 60000.0
5  Henry  25000.0
6  Chirag 60000.0
7  Lisa   25000.0
```

Time taken: 0.943 seconds, Fetched: 6 row(s)

hive> select Id, Name, sqrt(Salary) from employee;

OK

```
1  Gaurav 173.20508075688772
2  Aryan  141.4213562373095
3  Vishal 200.0
4  Siddharth 244.94897427831782
5  Henry  158.11388300841898
6  Chirag 244.94897427831782
7  Lisa   158.11388300841898
8  Ronit  141.4213562373095
```

Time taken: 0.654 seconds, Fetched: 8 row(s)

## Practical – 9

**Aim:** To show the Installation steps for the SPARK on single node system.

Step1: Download SPARK and extract the files

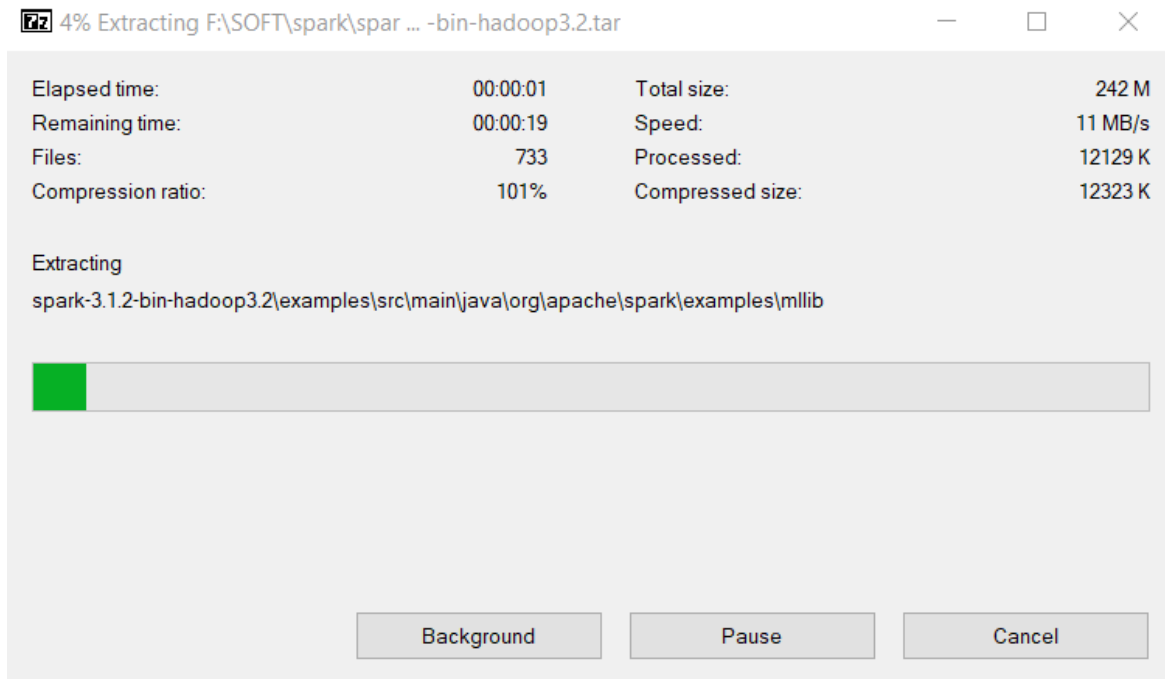


Fig 9.1 spark installation

Step 2: download winutils.exe and set up path variable

SPARK\_HOME

F:\SOFT\spark\spark-3.1.2-bin-hadoop3.2

In Path

%SPARK\_HOME%\bin

Already setup winutils.exe in Hadoop installation

Step 3: check installation

Fig 9.2 – Spark shell

Fig 9.3 – Spark UI

## Practical – 10

**Aim:** Develop a word count program using SPARK.

First start hadoop *start-dfs* for input file that contains list of words.

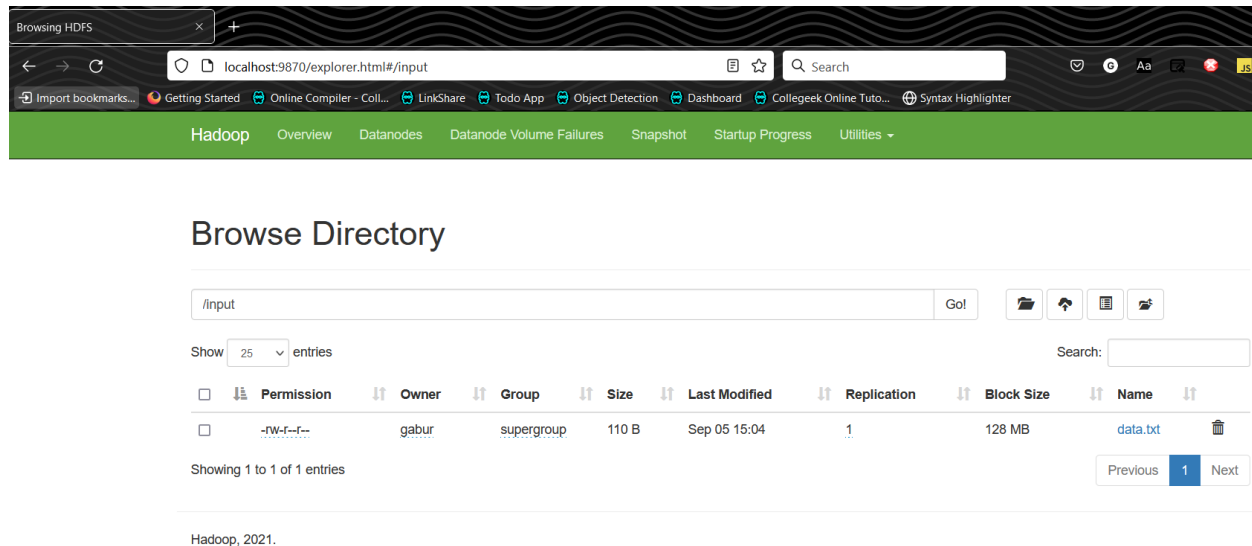


Fig 10.1 – Hadoop files

```
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>spark-shell
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/F:/SOFT/spark/spark-3.1.2-bin-hadoop3.2/jars/spark-unsafe_2.12-3.1.2.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://LAPTOP-B1C68170:4040
Spark context available as 'sc' (master = local[*], app id = local-1631936933222).
Spark session available as 'spark'.
Welcome to

  ____
 /_  __ \
/_/_/  \_/_/
version 3.1.2

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 13.0.1)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val text = sc.textFile("hdfs://localhost:9000/input")
21/09/18 09:19:06 WARN ProcsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of ProcessTree metrics is stopped
text: org.apache.spark.rdd.RDD[String] = hdfs://localhost:9000/input MapPartitionsRDD[1] at textFile at <console>:24
scala> val text = sc.textFile("hdfs://localhost:9000/input")
```

Fig 10.2 – Spark input file

Using spark action we can see file content.

```
scala> text.collect;  
res0: Array[String] = Array(big data, hadoop, hi, hello, good, big data, big data, hadoop, hello,  
good, bad, morning, hadoop, big data, hi)
```

Create Map of string => int for every word in file.

```
scala> val mapf = text.map(word => (word,1))  
mapf: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[4] at map at  
<console>:25  
  
scala> mapf.collect;  
res1: Array[(String, Int)] = Array((big data,1), (hadoop,1), (hi,1), (hello,1), (good,1), (big  
data,1), (big data,1), (hadoop,1), (hello,1), (good,1), (bad,1), (morning,1), (hadoop,1), (big  
data,1), (hi,1))
```

Reduce map to get count of words

```
scala> val reducef = mapf.reduceByKey(_+_);  
reducef: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[5] at reduceByKey at  
<console>:25  
  
scala> reducef.collect;  
res2: Array[(String, Int)] = Array((big data,4), (hello,2), (morning,1), (hadoop,3), (hi,2),  
(bad,1), (good,2))  
  
scala> reducef.collect.foreach(println)  
(big data,4)  
(hello,2)  
(morning,1)  
(hadoop,3)  
(hi,2)  
(bad,1)  
(good,2)  
  
scala> val tablef = reducef.toDF("word","count")  
tablef: org.apache.spark.sql.DataFrame = [word: string, count: int]  
  
scala> tablef.collect;  
res4: Array[org.apache.spark.sql.Row] = Array([big data,4], [hello,2], [morning,1],  
[hadoop,3], [hi,2], [bad,1], [good,2])
```



```
scala> tablef.show;
```

```
+-----+-----+
|  word|count|
+-----+-----+
|big data|  4|
|  hello|  2|
|morning|  1|
|  hadoop|  3|
|     hi|  2|
|    bad|  1|
|   good|  2|
+-----+-----+
```

```
scala> :quit
```

```
F:\SOFT\Hadoop\hadoop-3.3.1\sbin>stop-dfs
```

```
SUCCESS: Sent termination signal to the process with PID 17560.
```

```
SUCCESS: Sent termination signal to the process with PID 4964.
```

```
scala> reducef.collect.foreach(println)
```

```
(big data,4)
(hello,2)
(morning,1)
(hadoop,3)
(hi,2)
(bad,1)
(good,2)
```

```
scala> val tablef = reducef.toDF("word","count")
```

```
tablef: org.apache.spark.sql.DataFrame = [word: string, count: int]
```

```
scala> tablef.collect;
```

```
res4: Array[org.apache.spark.sql.Row] = Array([big data,4], [hello,2], [morning,1], [hadoop,3], [hi,2], [bad,1], [good,2])
```

```
scala> tablef.show;
```

```
+-----+-----+
|  word|count|
+-----+-----+
|big data|  4|
|  hello|  2|
|morning|  1|
|  hadoop|  3|
|     hi|  2|
|    bad|  1|
|   good|  2|
+-----+-----+
```

Fig 10.3 word count