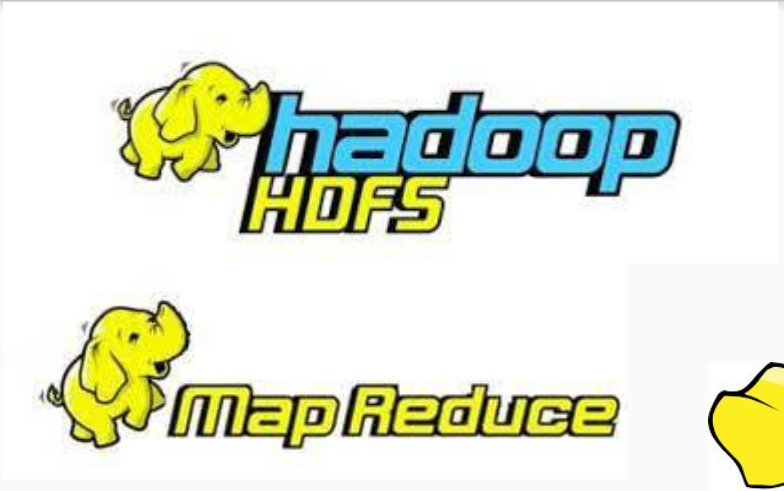
Bank Loan

By: Dawit Abay

INTRO

For my project I combines HDFS data and MapReduce algorithms to evaluate the risk of granting a loan to a person based on their location, loan type, and average risk.



Input Data

CustomerID	Customer Name	Loan Account Number	Sanctioned Loan Amount	Currency	Disbused Loan Amount	Loan Status	Risk in %	City	State	Country	Reason For Taking Loan
1	Michael	HHLNOI235648	3000000	US-Dollar	2500000	Current	20	Atlanta	Georgia	United State	Purchased Home
2	Christopher	HPLNOD24578	1500000	US-Dollar	1300000	Late	10	Augusta-Richmond County	Georgia	United State	Medical Emergency
3	Jessica	HVLOU458792	700000	US-Dollar	500000	Current	7	Columbus	Georgia	United State	Purchased Viechel
4	Matthew	HRLOF854586	200000	US-Dollar	200000	Closed	0	Macon-Bibb County	Georgia	United State	Purchased Furniture
5	Ashley	HHLNOI235649	3000000	US-Dollar	2500000	Current	20	Savannah	Georgia	United State	Purchased Home

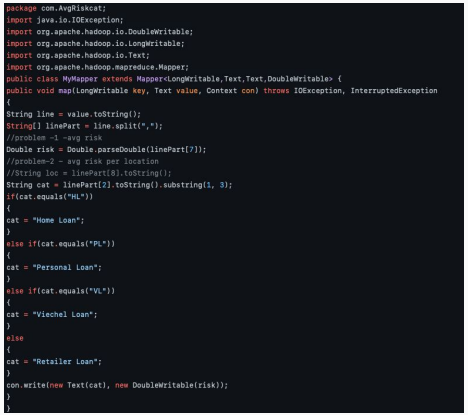
This is CSV data that show collections of CustomerID ,Customer Name ,Loan Account Number, Sanctioned Loan Amount, Currency ,Disbursed Loan Amount ,Loan Status,Risk in % ,City, State, Country, Reason For Taking Loan.

There are over 500 account in the csv data

Hadoop Code Mapper

Mapper maps input key/value pairs to a set of intermediate key/value pairs.

Reducer reduces a set of intermediate values which share a key to a smaller set of values.



```
package com.AvgRiskcat;
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MyMapper extends Mapper<LongWritable,Text,Text,DoubleWritable> {
public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException
{
String line = value.toString();
String[] linePart = line.split(",");
//problem -1 -avg risk
Double risk = Double.parseDouble(linePart[7]);
//problem-2 - avg risk per location
//String loc = linePart[8].toString();
String cat = linePart[2].toString().substring(1, 3);
if(cat.equals("HL"))
{
cat = "Home Loan";
}
else if(cat.equals("PL"))
{
cat = "Personal Loan";
}
else if(cat.equals("VL"))
{
cat = "Viechel Loan";
}
else
{
cat = "Retailer Loan";
}
con.write(new Text(cat), new DoubleWritable(risk));
}
}
```

Hive

Hive is a data warehouse infrastructure that facilitates querying and managing large data sets which resides in distributed storage system

After starting hdfs,yarn,mapreduce put input file in hdfs.

>hadoop fs -put source_path destination_path_with_file_name.

then start hive

>hive

To see database

>show database;

to create database

>create database bank;

To use a database

>use bank;

To create table

>create table bankloan(cos_id,cos_name,loan_no,amount,currency,amount_paid,status,risk,local,reason)

>row format delimited

>fields termineted by ','

>stored as textfile;

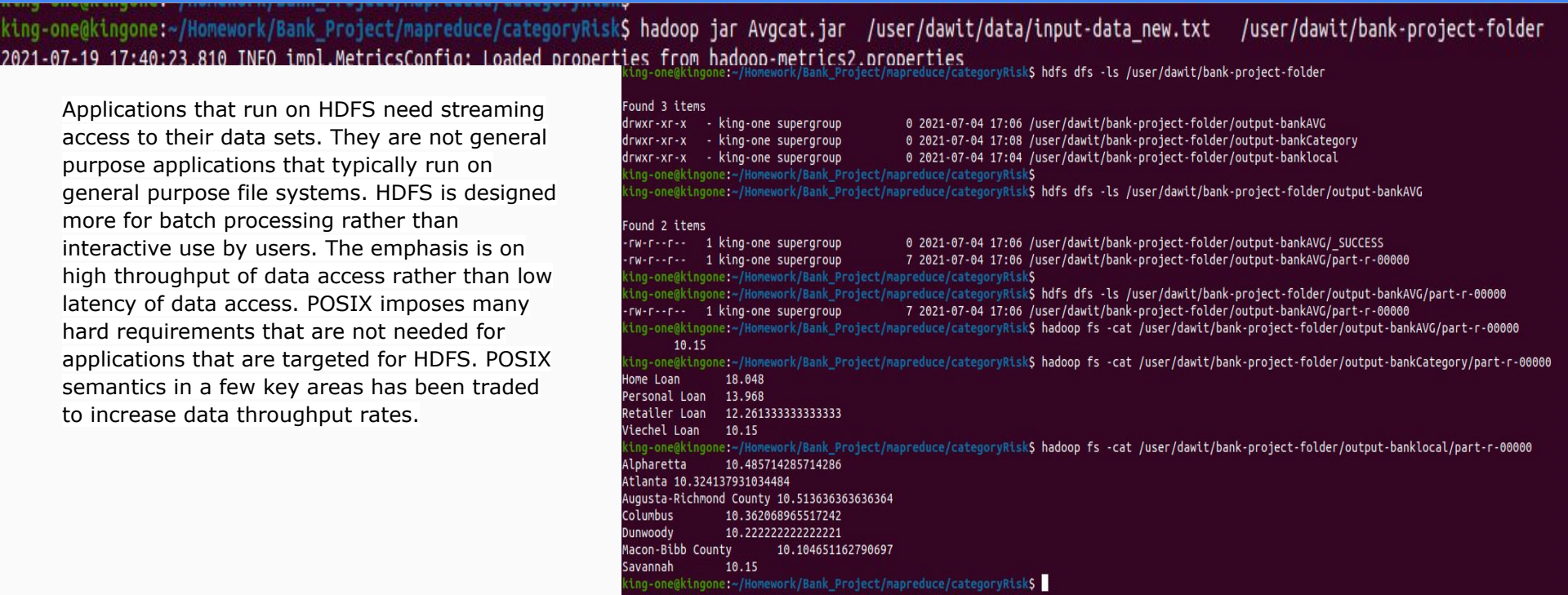
Hadoop Code Reducer

This code shows us how to calculate the risk of each loan

```
double glSum = 0;
int glcount = 0;
//double avg = 0;
public void reduce(Text key,Iterable<DoubleWritable> val,Context con) throws IOException, InterruptedException
{
for (DoubleWritable values : val) {
glSum = glSum + values.get();
glcount = glcount +1;
}
con.write(new Text(key), new DoubleWritable(glSum/glcount));
}

}
```

Run HDFS

Applications that run on HDFS need streaming access to their data sets. They are not general purpose applications that typically run on general purpose file systems. HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access. POSIX imposes many hard requirements that are not needed for applications that are targeted for HDFS. POSIX semantics in a few key areas has been traded to increase data throughput rates.

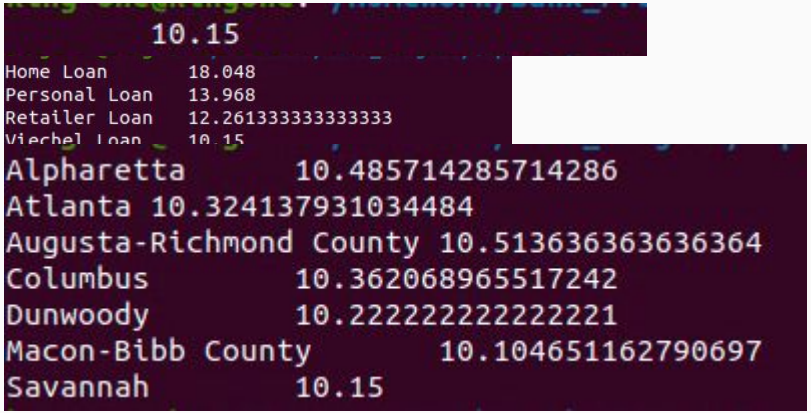


Hadoop OutPut

Avg risk

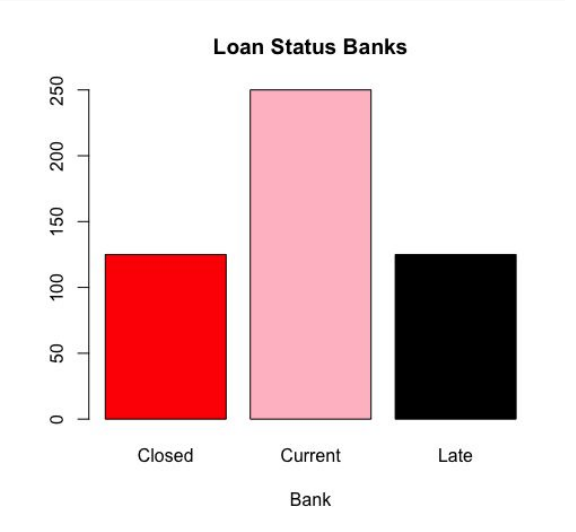
The risk of each loan for home,medical retailer and vehicle

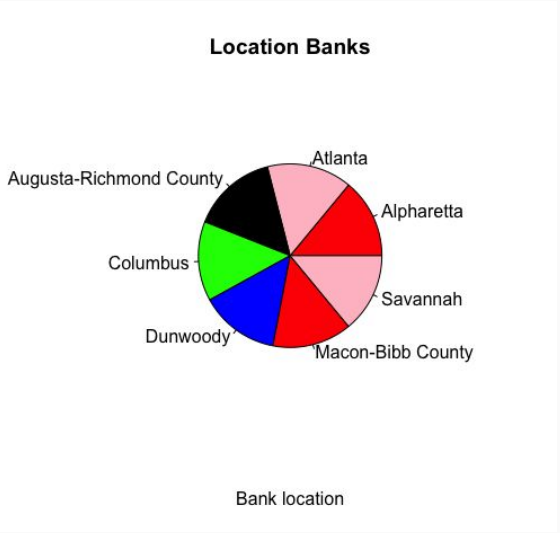
The risk for each location in Georgia



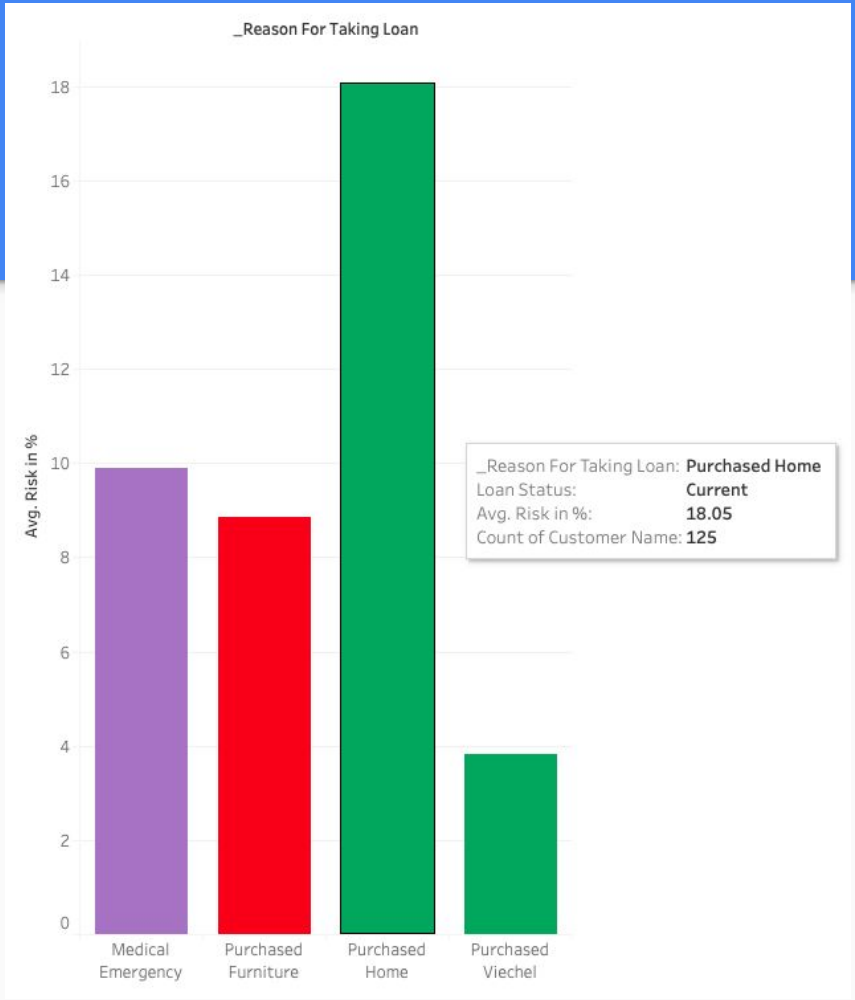
R Studio code

```
library(readr)

library(plotrix)

input_data <-
read.csv("~/Desktop/hadoop_bank_project/input-data.
csv",TRUE, sep = ",")

require("RColorBrewer")

pie(xtabs(~input_data$Location),main="Location
Banks",xlab="Bank location",col =
c("red","pink","black","green","blue"))

#barplot(xtabs(~input_data$`Loan
Status`),main="Loan Status Banks",xlab="Bank",col =
c("red","pink","black","green","blue"))
```

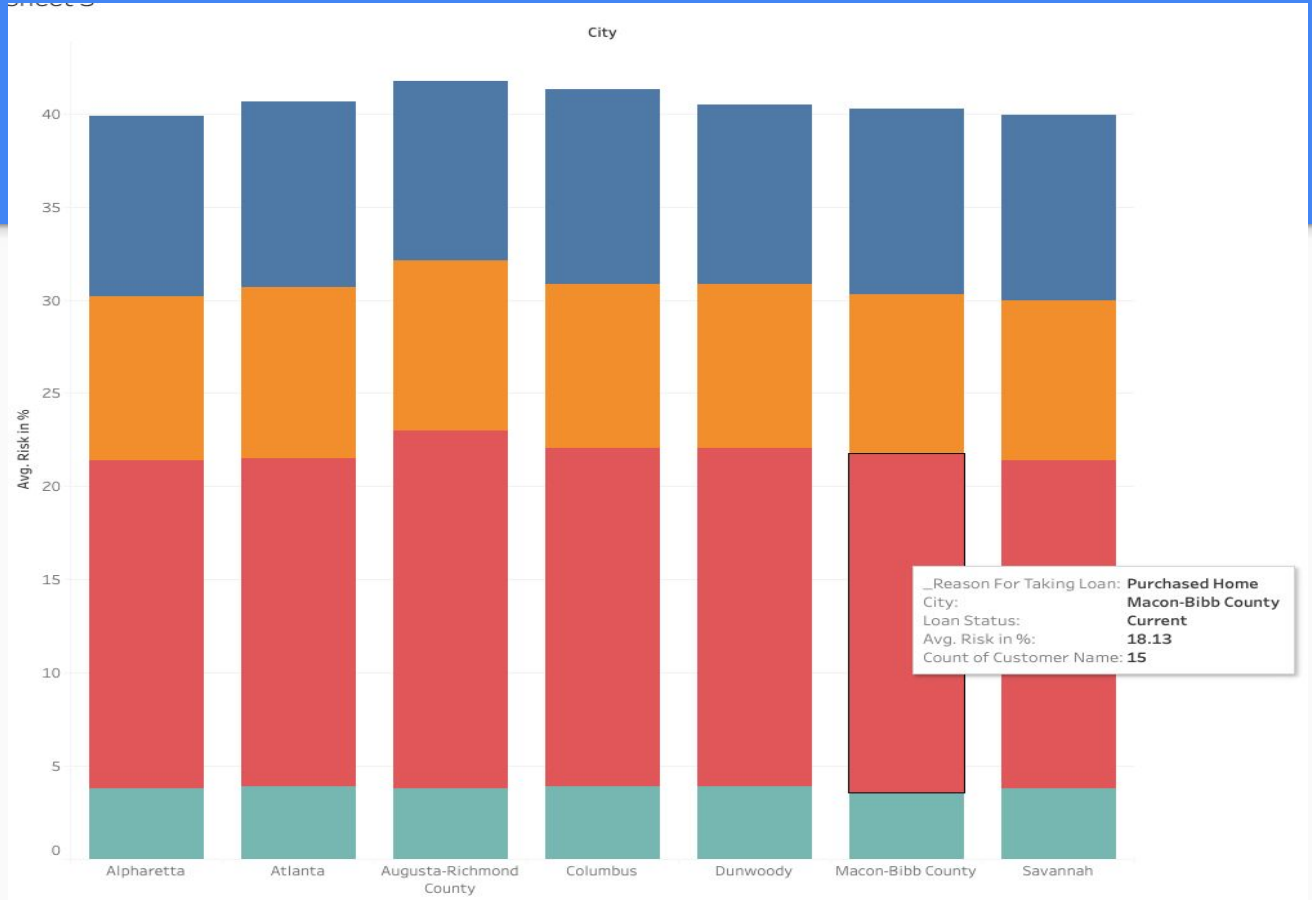




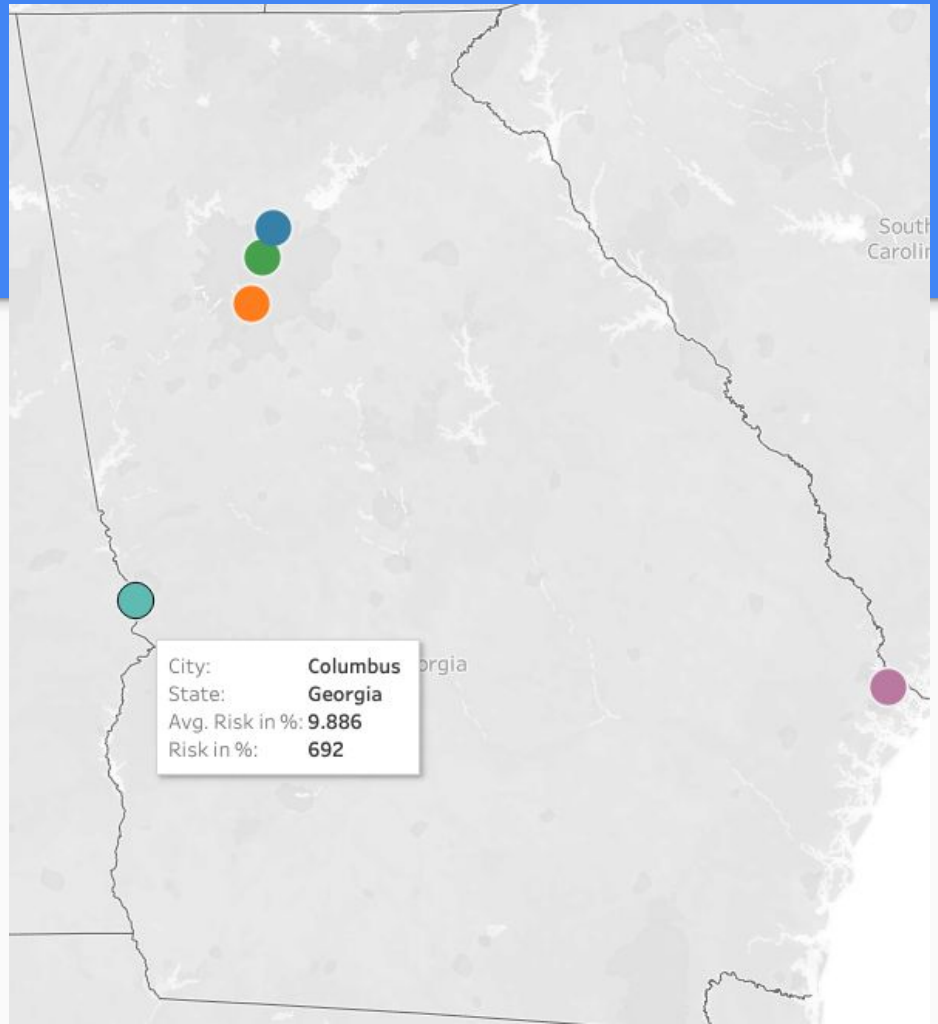
Graph’s


_Reason For Taking Loan
Avg. Risk in %
18
16
14
12
10
8
6
4
2
0
Medical Emergency
Purchased Furniture
Purchased Home
Purchased Viechel
_Reason For Taking Loan: Purchased Home
Loan Status: Current
Avg. Risk in %: 18.05
Count of Customer Name: 125

Graph’s


City
Avg. Risk in %
0
5
10
15
20
25
30
35
40
Alpharetta
Atlanta
Augusta-Richmond County
Columbus
Dunwoody
Macon-Bibb County
Savannah
_Reason For Taking Loan: Purchased Home
City: Macon-Bibb County
Loan Status: Current
Avg. Risk in %: 18.13
Count of Customer Name: 15

Map



Run Time

Do to size of the data the run time was fast

Resource

key algorithms/technology:

- Eclipse
- Hadoop
- Linux OS(Ubuntu 20.04)

Language - Java

Which part of Hadoop used in this project?

- HDFS
- MapReduce
- hadoop-common-3.1.2.jar
- hadoop-mapreduce-client-core-3.1.2.jar

data source: https://data.world/datasets/loan
https://www.kaggle.com/zaurbegiev/my-dataset
https://www.kaggle.com/panamby/bank-loan-status-dataset