Course Assignment

Master of

Applied Computer Science

Department of Technology

Assignment title	Exploring Bitcoin
Course code	MA120
Course name	Big Data
Due date	18 Oktober

Declaration:

Through the submission of this assignment, we hereby declare that this report is the result of our own work, and that all sources have been properly cited to throughout the text.

Names of students	Theodore Midtbø Alstad ; Howie Chen
Student ID numbers	865317 ; 866354

Exploring Bitcoin

ABSTRACT

In this assignment we parsed a dataset, in the form of XMLs, through a Hadoop- and Pig apache. We learned how to write MapReduce jobs and Pig scripts, and how effective it is on our dataset based on XML files from Bitcoin stack exchange.

Introduction

We choose to work together because both of us have python background, therefor we choose python as main programming language. We explored Bitcoin as given dataset from archive.org/download/stackexchange/bitcoin.stackexchange.com.7z through Apaches: Hadoop and Pig.

Main functions

Task 1 Warmup

1a) WordCount
Assumption: Count the words in body of questions PostTypeID="1" in the *Posts.xml* file. The result should be how many times a word occur in the body of questions.

ImplementationHello hello

Notes/Reflection bye bye

1b) Unique words
Assumption: Write a MapReduce job where the result should be unique words in the titles PostTypeID="1" in the *Posts.xml* File.

Implementation

Notes/Reflection

1c) MoreThan10
Assumption: Write a simple python code to check how many words there is in the title in the *Posts.xml*. The result should output how many titles are longer than 10 words.

Implementation

Notes/Reflection

1d) Stopwords
Assumption: Write a simple python code based on task 1a to exclude stopwords from body of questions PostTypeID="1" in the *Posts.xml*. The output should be text file without any stopwords.

Implementation

Notes/Reflection

1e) Pig top 10
Assumption: Write a pig script to select top 10 listed words after removing the stopwords from *Posts.xml*. The output should print out top 10 listed words and the corresponding occurrence rate.

Implementation

Notes/Reflection

1f) Tags
Assumption: Write a MapReduce job to create a dictionary over unique tags in *Posts.xml*.

Implementation

Notes/Reflection

Task 2 Discover

2a) Counting
Assumption: We chose to write a MapReduce job to count the total unique users there are in *Users.xml*. The result will print out how many unique users there are.

Implementation

Notes/Reflection

2b) Unique users
Assumption: Write a MapReduce job based on task 2a) to create a mapper and a reducer functions in *Users.xml*. The result should contain unique users in the dataset.

Implementation

Notes/Reflection

2c) Top miners
Assumption:Write a MapReduce job to find top 10 users based on attribute Reputation="x" in *Users.xml*. The result will print out top 10 users based on their reputation.

Implementation

Notes/Reflection

2d Top) questions
Assumption: Write a MapReduce job to find top 10 title questions PostTypeID="1" based on attribute Score="x" in *Posts.xml*.The result lists top 10 questions using id, question and the score.

Implementation

Notes/Reflection

2e) Favorite questions
Assumption: Write a MapReduce job to find top 10 title questions PostTypeID="1" based on attribute FavoriteCount="x" in *Posts.xml*.The result lists top 10 questions like id, question and the score.

Implementation

Notes/Reflection

2f) Average answers
Assumption:

Implementation

Notes/Reflection

2g) Countries
Assumption: We chose to write a MapReduce job to discover users by countries in *Users.xml*. The result lists different countries and corresponding users.

Implementation

Notes/Reflection

2h) Names
AssumptionWe chose to write a MapReduce job to find the most popular names in *Users.xml*. The result lists top 10 common names and how many.

Implementation

Notes/Reflection

2i) Answers
Assumption: Write a simple python code to find how many titles of questions PostTypeId="x" have at least one answers based on attribute AnswerCount in *Users.xml*. The result prints out how many questions have been answered.

Implementation

Notes/Reflection

Task 3 Numbers

3a) Bigram
Assumption: We chose to write a MapReduce job to find the most common pair of adjacent words in *Posts.xml*.For instance, "big data" or "Fast car" are examples of bigram. The result print the most common bigram

Implementation

Notes/Reflection

3b) Trigram
Assumption: This task is based on 3a, to find three words that appear consecutively in *Posts.xml*. The result print the most common trigram.

Implementation

Notes/Reflection

3c) Combiner
Assumption: Write a MapReduce job and add a combiner before the data sent to the reducer. The result should be the same as a reducer, but the reducer recive a smaller volume of data.

Implementation

Notes/Reflection Having a combiner in A MapReduce jobs saves us bandwidth and computational strain by decreasing

the volume of data sent from the mapper. What combiner means is to have a semi-reducer after the mapper before sending it to the reducer.

3d) Useless
Assumption: We chose write a MapReduce job to find how many times the word "useless" in *Posts.xml* occurs in the body of questions PostTypeId="1". The result print how many times the word useless occurs.

Implementation

Notes/Reflection

Task 4 Search engine

4a) Title index
Assumption: We chose to write MapReduce job to create index over titles, bodies and answers of questions in *Posts.xml*. We are after having a simple index that lists publications in which a search term/s and occur/s. The result is a list of words and their index appearance in posts

Implementation

Notes/Reflection

Conclusion