Department of Computing

School of Mathematical, Physical and Computational Sciences

Assessed Coursework Set Front Page

Module code: CSMCC16

Lecturer responsible: Atta Badii

Coursework description: Develop a Software Prototype of a MapReduce-like system

Deadlines:
Work to be handed in by 12:00 on: TBA

Work will be marked and returned by: TBA

NOTES:

All coursework code and final report files should be uploaded to Blackboard in a compressed format. No hardcopy submission is required.

If your work is submitted after the deadline, 10% of the maximum possible mark will be deducted for each working day (or part of) it is late. A mark of zero will be awarded if your work is submitted more than 5 working days late. You are strongly recommended to submit work by the deadline as a late submission on one piece of work can impact on other work.

If you believe that you have a valid reason for failing to meet a deadline then you should complete an Extenuating Circumstances form and submit it to the Hub before the deadline.

CSMCC16: Cloud Computing 2018

Coursework: Passenger Airline Flights

Description:

For this coursework there are two files containing lists of data. These are located on the Blackboard system in the Cloud Computing assignments directory – download them from:

Blackboard → Enrollments → CSMCC16

In the Lecture Notes tab, select as follows

Assignments → Coursework Data

The coursework data includes:

Top30_airports_LatLong.csv **AComp_Passenger_data.csv**

The first data file contains details of passengers that have flown between airports over a certain period. The data is in a comma delimited text file, one line per record using this format:

Passenger id:	Format: $XXXnnnnXXn$
Flight id:	Format: $XXXnnnnX$
From airport IATA/FAA code:	Format: XXX
Destination airport IATA/FAA code:	Format: XXX
Departure time (GMT):	Format: $n[10]$ (This is in Unix 'epoch' time)
Total flight time (mins):	Format: $n[1..4]$

The second data file is a list of airport data comprising the name, IATA/FAA code, and location of the airport. The data is in a comma delimited text file, one line per record using this format:

Airport name:	Format: $X[3..20]$
Airport IATA/FAA code:	Format: XXX
Latitude:	Format: $n.n[3..13]$
Longitude:	Format: $n.n[3..13]$

Where X is Uppercase ASCII, n is digit 0..9 and $[n..m]$ is the min/max range of the number of digits/characters in a string.

There are various errors in the **AComp_Passenger_data.csv** input data file, your code should successfully handle these in an **appropriate** manner. The output can be to screen, but must also be written to text files, the format of which is your decision.

There are two additional data input files that can be downloaded from this directory. These two additional input files are as follows:

AComp_Passenger_data_no_error_DateTime.csv **AComp_Passenger_data_no_error.csv**

These can be used during the **initial** development and debugging phases only.

For the final stages of development (i.e. requiring error handling) use the **AComp_Passenger_data.csv** file. Thus the 'no_error' files are not to be used for the software runs that generate the data for the final report, to do so will result in loss of marks.

Objectives

- Determine the number of flights from each airport; include a list of any airports not used.
- Create a list of flights based on the Flight id, this output should include the passenger Id, relevant IATA/FAA codes, the departure time, the arrival time (times to be converted to HH:MM:SS format), and the flight times.
- Calculate the number of passengers on each flight.
- Calculate the line-of-sight (nautical) miles for each flight and the total travelled by each passenger and thus output the passenger having earned the highest air miles.

Tasks

1. For this task in the development process, develop a **non-MapReduce** executable **prototype**, (in Java, C, or C++). The objective is to develop the basic functional 'building-blocks' that will support the development objectives listed above, in a way that mimics something of the operation of the MapReduce/Hadoop framework. This does not mean that you have to implement a client/server infrastructure such as Hadoop! The solution may use multi-threading if this suits your particular design and implementation strategy, the marking strategy will reflect the appropriate use of: coding techniques, succinct standard and appropriate usage of Javadoc comments, data structures and overall program design. The code should be subject to command line version control using a Git repository under your university username under https://csgitlab.reading.ac.uk.

   The final results/output must use the **AComp_Passenger_data.csv** file. Error detection and handling for this task can be quite basic, but it must be robust and follow a logical, well considered strategy – the latter is entirely for you to decide.

2. Write a **brief** report (no more than 7 pages for the actual content, not including title page) explaining:

   a) The high-level description of the development of the prototype software.
   b) A simple description of the Git command line process undertaken.
   c) A detailed description of the MapReduce functions you are replicating.
   d) The output format of any reports that each job produces.
   e) The strategy derived to handle input data error detection/correction and/or run-time recovery.
   f) A self-appraisal of your (equivalent) MapReduce run-time software, with suggestions as to how it may be usefully improved upon. You may comment on any aspect of the development process.

Marking Criteria

The table below shows what is typically expected of the work to obtain a given mark. The assignment carries 30% of the total course marks.

Classification Range	Typically the work should meet these requirements
First Class (>= 70%)	The coursework demonstrates: An exceptional understanding of the principles of MapReduce and software prototyping. Excellent technical skills in designing, implementing and testing the software structure. Excellent understanding and implementation of a suitable error detection/correction strategy Documentation is clear, logical, and well-presented.
Upper Second (60..69)	The coursework demonstrates: Good understanding of the principles of MapReduce and software prototyping. Good technical skills in designing, implementing and testing the software structure. Good understanding and implementation of a suitable error detection/correction strategy. Documentation is clear, logical, and well-presented.
Lower Second (50..59)	The coursework demonstrates: Reasonable understanding of the principles of MapReduce and software prototyping. Reasonable technical skills in designing, implementing and testing the software structure. Reasonable understanding and implementation of a suitable error detection/correction strategy. Documentation is clear, logical, and well-presented.
Third (40..49)	The coursework demonstrates: Basic understanding of the principles of MapReduce and software prototyping. Basic technical skills in designing, implementing and testing the software structure. Basic understanding and implementation of a suitable error detection/correction Strategy. Documentation is clear, logical, and well-presented.
Fail (<40)	The coursework fails to demonstrate any real understanding of the implementation of MapReduce, Hadoop and software prototyping. The documentation is incomplete or contains significant errors.
Breakdown	
Part of Submission	**Coursework Marks (%)**
Task 1: Software Prototype	60
Task 2: Documentation	40