CSE441: DATABASE SYSTEMS

ASSIGNMENT 2 (Two-phase merge sort)

DEADLINE: 9:00 pm, 6th February 2020

In this assignment, you have to implement the two-phase merge sort algorithm to sort a large number of records.

Programming Languages Allowed: Python and C++

Part-1 (70% of marks)

Implementation Specs:

1. The metadata file will contain information about the size of the different columns (in bytes).
2. The data type for all columns will be a string.
3. The number of columns can range from 1 to 20.
4. Your program should be capable of sorting in both ascending and descending order.
5. Your program should run for different values of main memory usage allowed and the different size of files (MBs-GBs).

If we found your program is using main memory more than specified, **you will lose marks**.

Note: Print simple logs while executing on the console.

Example:

```
$> ./sort input.txt output.txt 50 asc C1 C2
```

Output:

```
###start execution
##running Phase-1
Number of sub-files (splits): 10
sorting #1 sublist
Writing to disk #1
…..
Sorting #10 sublist
Writing to disk #10
##running phase-2
Sorting...
Writing to disk
###completed execution
```

Part-2 (30% of marks)

Reimplement two-phase merge sort by the **parallelising first phase of two-phase merge sort** using threads.

Input format:

Metadata.txt

<ColumnName 1>,<Size of the column>
<ColumnName 2>,<Size of the column>
……..
<ColumnName n>,<Size of the column>

Input.txt

Containing the records with the column values. All the values will be string only and might contain space or “,”.

Data

To generate an input file, visit http://www.ordinal.com/gensort.html and download **gensort** code to generate data.
The following command generates the first 100 tuples of three columns.

```
$> ./gensort a 100 input.txt
```

For generating input files with different sizes, you will have to compute the size of each tuple and calculate for what number of tuples a certain file size can be achieved. For instance,

If **Tuple size** = 10 bytes (sum of sizes of all columns)
File size = (10 * total_tuples) bytes.

Command-line inputs:

1. Input file name (containing the raw records)
2. Output filename (containing sorted records)
3. Main memory size (in MB)
4. Number of threads (for part-2)
5. Order code (asc / desc) asc : ascending, desc : descending.

6. ColumnName K_1
7. ColumnName K_2
8. …..

Example

- ./sort input.txt output.txt 50 asc C1 C2 (for part1)
- ./sort input.txt output.txt 100 5 desc C3 C1 (for part2)

In the first one, the records in input.txt to be sorted in ascending order with 50MB space based in C1 and if any row has same C1, sort based on C2.

Observations

1. **Varying FileSize with constant memory**: Take the main memory as 100MB and run your two algorithms for file size 5MB, 50MB, 500MB, 1GB, 2GB, 3GB and note the time taken for each run.
2. **Varying memory with constant FileSize:** Take a file of size 500MB and run your two algorithms with memory as 25MB, 100MB, 250MB, 500MB and note the time taken at each run.

Record your inferences and explain your views on the **execution times** (y-axis: execution time).

Error handling

1. Check feasibility condition for two-way merge sort for provided memory constraints. If not satisfied, provide an error message and halt.
2. The number of command-line arguments to the executable.

Deliverables

Create a folder with RollNuber_Assignment2 and put the following into it
1. Python/C++ source codes
2. A pdf with the name **Analysis.pdf** containing the following information
    a. Configuration of the system
    b. Observations (in tabular format and corresponding graphs).
    c. Explanation

Zip the folder and upload.

Strict action will be taken for copying in the Assignments.