Cluster and Cloud Computing Assignment 1 – Mastodon Data Analytics

Problem Description

Your task in this programming assignment is to implement a parallelized application leveraging the University of Melbourne HPC facility SPARTAN. Your application will process a large Mastodon dataset. The objective is to identify the:

- The 5 happiest hours in the data,
- The 5 saddest hours in the data,
- The 5 happiest users in the data, i.e. based on posts from a given user account,
- The 5 saddest users in the data, i.e. based on posts from a given user account.

You should be able to log in to SPARTAN through running the following command:

ssh your-unimelb-username@spartan.hpc.unimelb.edu.au

with the password you set for yourself on *karaage* (https://dashboard.hpc.unimelb.edu.au/karaage). Thus, I would log in as:

ssh rsinnott@spartan.hpc.unimelb.edu.au
password = my karaage password (not my UniMelb password)

If you are a Windows user then you may need to install an application like Putty.exe to run *ssh*. (If you are coming from elsewhere with different firewall rules, then you may need to use a VPN).

The files to be used in this assignment are accessible at:

- /data/gpfs/projects/COMP90024/mastodon-144g.ndjson
  - this is the main large file to use for your final analysis and report write up, i.e., **do not use this file for software development and testing**.
- /data/gpfs/projects/COMP90024/mastodon-16m.ndjson
  - is a much smaller JSON file that should be used for testing.
- /data/gpfs/projects/COMP90024/mastodon-106k.ndjson
  - is the smallest JSON file that should be used for initial testing and exploring the data.

You may decide to use the smaller JSON files on your own PC/laptop to start with for development and testing. As noted in the lecture/workshop, we cannot help you with the setup of software on your own local (PC/laptop).

Once on SPARTAN, you should make a symbolic link to these files, i.e., you should run the following commands at the Unix prompt **from your own user directory on SPARTAN**:

```
ln –s /data/gpfs/projects/COMP90024/mastodon-106k.ndjson
ln –s /data/gpfs/projects/COMP90024/mastodon-16m.ndjson
ln –s /data/gpfs/projects/COMP90024/mastodon-144g.ndjson
```

(Do not cut and paste these commands as it will include the carriage return. Type them in manually).

Once done you should see something like the following **in your home directory**:

```
lrwxrwxrwx 1 rsinnott punim1502 50 Mar 7 11:43 mastodon-106k.ndjson -> /data/gpfs/projects/COMP90024/mastodon-106k.ndjson
lrwxrwxrwx 1 rsinnott punim1502 49 Mar 7 11:43 mastodon-16m.ndjson -> /data/gpfs/projects/COMP90024/mastodon-16m.ndjson
lrwxrwxrwx 1 rsinnott punim1502 51 Mar 7 11:43 mastodon-144g.ndjson -> /data/gpfs/projects/COMP90024/mastodon-144g.ndjson
```

Your assignment is to (*eventually*!) search the large Mastodon data file and identify:

- the 5 happiest hours, e.g. (1) 3-4pm on 23rd November with an overall sentiment score of +12, (2) 1-2pm on 1st March with an overall sentiment score of +11, (3)… etc,
- the 5 saddest hours, e.g. (1) 25th February 2-3pm with an overall sentiment score of -12, (2) 11th June with an overall sentiment score of -11, (3)… etc,

- the 5 happiest people, e.g. (1) "*martingelin*, account id *109491602528257114* with a total positive sentiment score of +99", (2) …etc
- the 5 saddest people, e.g. (1) "*InstantArcade*, account id *111825369875335844* with a total negative sentiment score of -66", (2) …etc

Noting that these dates and the sentiment scores are representative! There is no need for deeper statistical analysis of the data. You will note that the vast majority of posts have a sentiment score already, as shown below.

doc.sensitive	doc.createdAt	doc.content	doc.sentiment
false	2025-01-30T11:55:33.000Z	<p>Väldigt bra samtal förra veckan me…	0.02127659574468085
false	2025-02-01T00:40:17.000Z	<p>My Philips Hue hub randomly went o…	0.05
false	2025-01-29T14:03:25.000Z	<p><a href="https://sfba.social/tags/…	0
false	2025-02-01T13:15:02.000Z	<p>Watch: Three more Israeli hostages…	-0.009523809523809525
false	2025-01-29T15:14:43.000Z	<p>Exclusive: Netflix launches new Se…	-0.02040816326530612

Your task is to use information on when the post was made (*doc.created_at*), the pre-calculated sentiment score (*doc.sentiment*) and each unique user account (*doc.account.id* and *doc.account.username*) to answer the bullet points above. Your code should deal with potentially ill-formatted JSON posts and potentially other issues, e.g. no sentiment score.

Your application should allow a given number of nodes and cores to be utilized. Specifically, **your application should be run once** to search the *largest NDJSON* file on each of the following resources:

- 1 node and 1 core;
- 1 node and 8 cores;
- 2 nodes and 8 cores (with 4 cores per node).

The resources should be set when submitting the search application with the appropriate *SLURM* options. Note that you should run a single *SLURM* job three separate times on each of the resources given here, i.e. you should not need to run the same job 3 times on 1 node 1 core for example to benchmark the application. (This is a shared facility and this many COMP90024 students will consume a lot of resources especially with the large file!).

You can implement your solution using any routines and libraries you wish however it is strongly recommended that you follow the guidelines provided on access and use of the SPARTAN cluster. Do not for example think that the job scheduler/SPARTAN automatically parallelizes your code – it doesn't! You may wish to use the pre-existing MPI libraries that have been installed for C, C++ or Python, e.g., mpi4py. You should feel free to make use of the Internet to identify which JSON processing libraries you might use. You may also use any regular expression libraries that you might need for string comparison.

Your application should return the final results and the time to run the job itself, i.e. the time for the first job starting on a given SPARTAN node to the time the last job completes. You may ignore the queuing time. The focus of this assignment is not to optimize the application to run faster, but to learn about HPC and how basic benchmarking of applications on a HPC facility can be achieved and the lessons learned in doing this on a shared resource.

Final packaging and delivery

You should write a brief report on the application – **no more than 4 pages**!, outlining how it can be invoked, i.e. it should include the scripts used for submitting the job to SPARTAN, the approach you took to parallelize your code, and describe variations in its performance on different numbers of nodes and cores. Your report should include the actual results tables as outlined above and a single graph (e.g., a bar chart) showing the time for execution of your solution on 1 node with 1 core, on 1 node with 8 cores and on 2 nodes with 8 cores. You should relate your results back to Amdahl's law to describe the potential performance changes.

Deadline

The assignment should be submitted to Canvas as a zip file. The zip file must be named with the students named in each team and their student Ids. That is, *ForenameSurname-StudentId:ForenameSurname-StudentId* might be *<SteveJobs-12345:BillGates-23456>.zip*. Only one report is required per student pair and only one student needs to upload this report. The deadline for submitting the assignment is: **Friday 11th April (by 12 noon!)**.

It is strongly recommended that you do not do this assignment at the last minute, as it may be the case that the Spartan HPC facility is under heavy load when you need it and hence it may not be available! You have been warned…!!!!

Marking

The marking process will be structured by evaluating whether the assignment (application + report) is compliant with the specification given. This implies the following:

- A working demonstration – **60% marks**
- Report and write up discussion – **40% marks**

Timeliness in submitting the assignment in the proper format is important. **A 10% deduction per day will be made for late submissions.**

You are free to develop your system where you are more comfortable with (at home, on your PC/laptop, in the labs, on SPARTAN itself - but not on the largest file until you are ready!). Your code should of course work on SPARTAN.