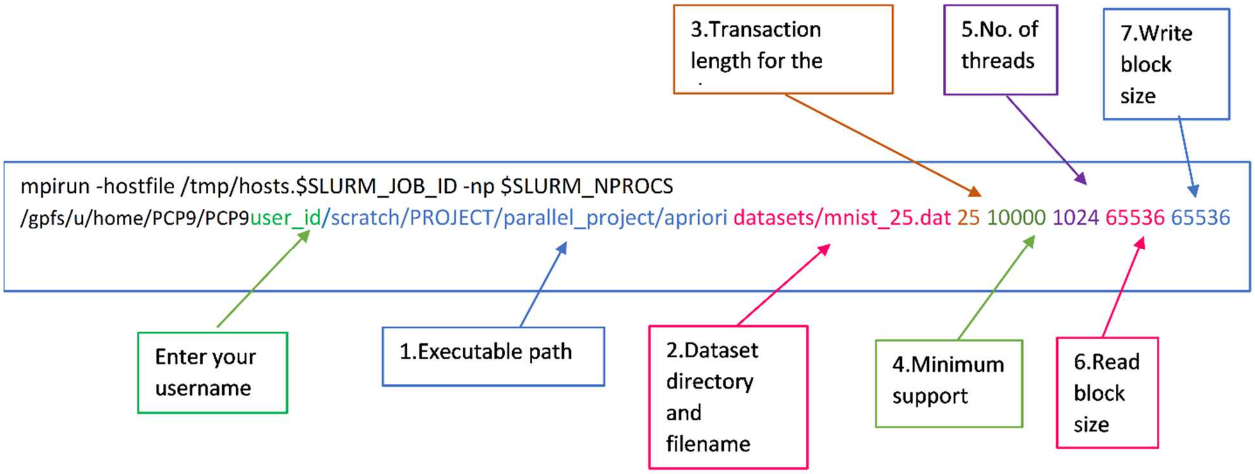
RCS IDs: hussam4@rpi.edu, duttas@rpi.edu, neehan@rpi.edu

To perform an execution on the AiMOS, please follow the following instructions:

1. Enter AiMOS front end and load the required modules:
   a. module load gcc/7.4.0/1
   b. module load spectrum-mpi
   c. module load cuda
2. Go to the directory where you have stored the source files.
3. Run the command "`make all`". This will compile the source files and generate an executable - "apriori".
4. Open the slurmSpectrum.sh file. It contains the mpirun command that has a structure as shown below:



```
mpirun -hostfile /tmp/hosts.$SLURM_JOB_ID -np $SLURM_NPROCS
/gpfs/u/home/PCP9/PCP9user_id/scratch/PROJECT/parallel_project/apriori datasets/mnist_25.dat 25 10000 1024 65536 65536
```

5. Please change the slurmSpectrum.sh file as follows:
   - Modify the home directory with your username (Change the project name as well if you're in a different project than PCP9)
   - **First parameter:** the executable path. DO NOT change the executable name. It should be **apriori.**
   - **Second parameter:** Dataset information. We have stored datasets inside the 'datasets' sub-directory. For this example, we would show how to run "mnist_25.dat" file.
   - **Third parameter:** Transaction byte length for the dataset. This value is 25 for "mnist_25.dat". If you wish to choose any other dataset, the transaction length can be found at the end of the filename just before the ".dat" extension (for example 98 for "mnist_98.dat").
   - **Fourth parameter:** Minimum support. Please provide a moderate minimum support for a reasonable run time. For this dataset of size 60,000 we have chosen minimum support as 10,000 for a quick run.
   - **Fifth (optional) parameter:** Number of threads you want to use.

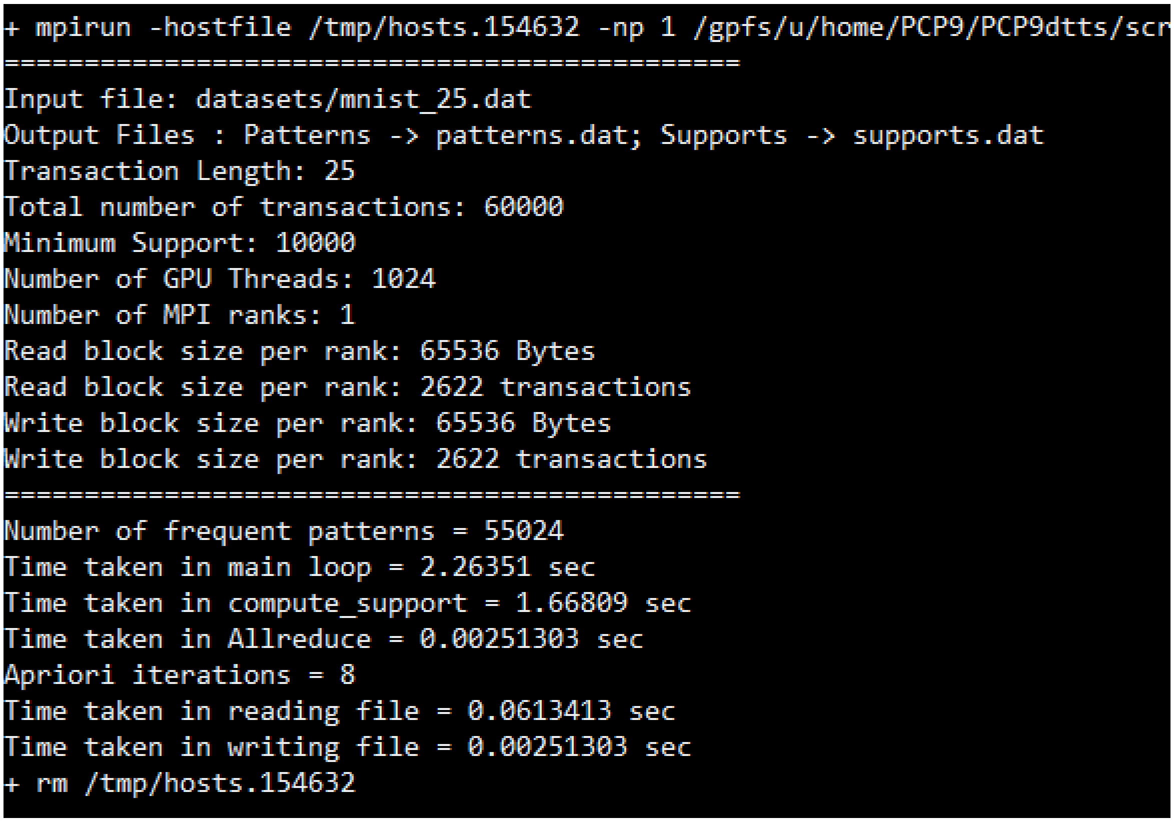
- **Sixth (optional) parameter:** It is an optional parameter that denotes the read block size in bytes. For 64 KB, we have entered 65536 bytes here. The default value is 1024 bytes in case you don't explicitly specify it.
- **Seventh (optional) parameter:** It is an optional parameter that denotes the write block size in bytes. For 64 KB, we have entered 65536 bytes here. The default value is 1024 bytes in case you don't explicitly specify it.

6. After saving the changes to this file, run a command of the following structure to run the executable:

```
make run n=1 t=1 g=1
```

Here, "`n`" denotes number of nodes. "`t`" denotes number of MPI Ranks per node and "`g`" denotes number of GPUs you want to use per node. You can change the parameters accordingly.

7. After the job has been finished you can view the "slurm-*JOB_ID*.out" file and in the end you should see something like this. The section after the first series of "="s describes the configuration used for running the executable. The second section denotes a summary of the results.

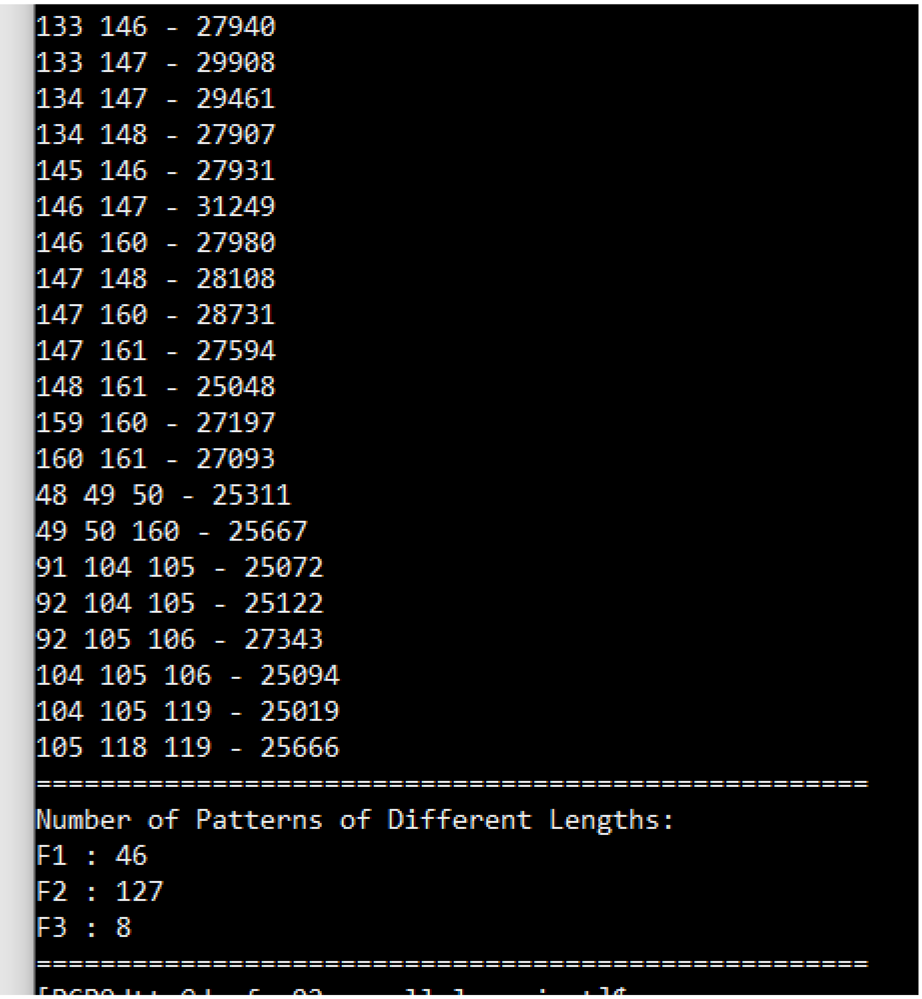


```
+ mpirun -hostfile /tmp/hosts.154632 -np 1 /gpfs/u/home/PCP9/PCP9dtts/scr
=============================================
Input file: datasets/mnist_25.dat
Output Files : Patterns -> patterns.dat; Supports -> supports.dat
Transaction Length: 25
Total number of transactions: 60000
Minimum Support: 10000
Number of GPU Threads: 1024
Number of MPI ranks: 1
Read block size per rank: 65536 Bytes
Read block size per rank: 2622 transactions
Write block size per rank: 65536 Bytes
Write block size per rank: 2622 transactions
=============================================
Number of frequent patterns = 55024
Time taken in main loop = 2.26351 sec
Time taken in compute_support = 1.66809 sec
Time taken in Allreduce = 0.00251303 sec
Apriori iterations = 8
Time taken in reading file = 0.0613413 sec
Time taken in writing file = 0.00251303 sec
+ rm /tmp/hosts.154632
```

8. After the execution the resultant patterns and supports are saved to "patterns.dat" and "supports.dat" files correspondingly. You can view them in a human readable form just by running the following make command:

```
make read tl=25
```

Here “`tl`” denotes transaction length (The third parameter while running the executable). For mnist_25 this value is 25. This compiles and executes the auxiliry program called “reader” (stored in the reader directory along with source code). The output would look something like this:



```
133 146 - 27940
133 147 - 29908
134 147 - 29461
134 148 - 27907
145 146 - 27931
146 147 - 31249
146 160 - 27980
147 148 - 28108
147 160 - 28731
147 161 - 27594
148 161 - 25048
159 160 - 27197
160 161 - 27093
48 49 50 - 25311
49 50 160 - 25667
91 104 105 - 25072
92 104 105 - 25122
92 105 106 - 27343
104 105 106 - 25094
104 105 119 - 25019
105 118 119 - 25666
====================================================
Number of Patterns of Different Lengths:
F1 : 46
F2 : 127
F3 : 8
====================================================
```

In the left of the hyphen we can see the frequent patterns and we can see the support of the pattern in the right side. Number of patterns for different lengths can also be found after the series of “=”s.

9. If you want to compile the program without any CUDA (GPU) activity, please run the following command instead of the usual “`make all`”:

```
make nocuda
```