REU BigDataX 2023

**Linux & Tools HomeWork**

Question 3

1. Script “generate-dataset.sh <filename> <num\_records>” was generated with two

command line arguments specifying the file name to output and the number of

records, where each record is separated by new line character, and each has the following format: <integer> <integer> <ASCII\_string>. The integers were stored in binary format (4 bytes) and the ASCII\_string as a string.

Main script run-generate-dataset.sh runs generate-dataset.sh using the “time” command to show how long the benchmark took to complete. The benchmark runs for at least 10 seconds by the parent process sleeping. It uses a nohup command to ignore all hangup (SIGHUP) signals and to keep running program in the background after logging off.

As an example here is the output of running the process for 1000 records:

./run-generate-dataset.sh out1.txt 1000

Number of records generated: 1000

more nohup.out

0.92user 0.14system 0:01.09elapsed 98%CPU (0avgtext+0avgdata 3060maxresident)k

0inputs+216outputs (0major+294710minor)pagefaults 0swaps

1

1. Script “sort-data.sh” was generated that takes input a file from part (a) above and

sorts the file based on solely the first column data and not the entire line of data. The linux sort command is used to make sure the data in column 1 is treated as numbers and not text. It uses the “time” command to show how long the sort script took to complete.

As an example here is the output of running the process for 1000 records:

time ./sort-data.sh out1.txt

Output file is out1.txt.sortedoutput

real 0m0.045s

user 0m0.029s

sys 0m0.003s

1. Script “script run-generate-dataset.sh” was used to generate 3 data files with different number of records: 1000, 100000, 10000000. Using the time command it measures time taken to generate these records.

./run-generate-dataset.sh out1.txt 1000

Number of records generated: 1000

more nohup.out

0.92user 0.14system 0:01.09elapsed 98%CPU (0avgtext+0avgdata 3060maxresident)k

0inputs+216outputs (0major+294710minor)pagefaults 0swaps

1

user+system time = 1.06 secs

./run-generate-dataset.sh out2.txt 100000

Number of records generated: 100000

more nohup.out

91.31user 13.85system 1:47.36elapsed 97%CPU (0avgtext+0avgdata 3068maxresident)k

0inputs+20920outputs (0major+28973695minor)pagefaults 0swaps

user+system time = 105.16 secs

./run-generate-dataset.sh out3.txt 10000000

Number of records generated: 10000000

more nohup.out

22866.49user 4565.22system 8:15:10elapsed 92%CPU (0avgtext+0avgdata 3096maxresident)k

2248inputs+2096976outputs (14major+2927733408minor)pagefaults 0swaps

user+system time = 27431.71 secs

Script “sort-data.sh” was used to sort the data files from part a) and measure the time.

time ./sort-data.sh out1.txt

Output file is out1.txt.sortedoutput

real 0m0.045s

user 0m0.029s

sys 0m0.003s

user+system time = 0.032 secs

time ./sort-data.sh out2.txt

Output file is out2.txt.sortedoutput

real 0m9.400s

user 0m7.166s

sys 0m0.016s

user+system time = 7.182 secs

time ./sort-data.sh out3.txt

Output file is out3.txt.sortedoutput

real 135m9.231s

user 124m58.416s

sys 0m12.636s

user+system time = 7511.052 secs

Python script plot.py was generated that uses matplotlib script to generate a graph for the time taken to generate the data and the time taken to sort the data at the 3 different scales. I used a log plot. The graph automatically adjust to the number of entries, and the scale of the data by using the plt.axis((xmin,xmax,ymin,ymax)) command.

The resulting plot is below:

