LERTs reproducibility experiment list

Paper: Timely Reporting of Heavy Hitters using External memory
(https://dl.acm.org/doi/pdf/10.1145/3318464.3380598)

There are two folders in the submission:

1. LERT-src: contains source codes for all the data structures and a script containing the commands to run benchmarks. The benchmarks dump the results in text files in dir "sigmod20_raw". This directory is initially empty and gets populated after running benchmarks.
2. SIGMOD20-paper: contains tex files to generate the paper.pdf. The tex file uses the data generated by the benchmarks in the “sigmod20_raw” directory to create the graphs in the paper. There is also a “data” directory that contains all the results from the published version of the paper. In case someone wants to skip running some or all of the experiments, the data files for the experiments can be copied to the “sigmod20_raw” directory to build the paper with original results.

Note: “sigmod20_raw_tested” and “sigmod20_figs_tested” contains the output during script testing runs. These folders can be renamed to “sigmod20_raw” and “sigmod20_figs” to build the paper.

Requirements:

1. OpenSSL version 1.0.2g 1 Mar 2016 or higher.
2. CPU: Intel 4th generation Haswell or higher.
3. Flash drive (SSD) > 64GB
4. RAM > 64GB
5. Number of cores 64
6. cgroups
    a. We have included an installation script (limitMem.sh) to install and setup a cgroup profile. The cgroup profile is used for on-ssd experiments.

Instructions to build the paper:

1. ./limitMem.sh : To setup cgroups profile.
2. cd LERT-src
3. ./run_experiments.sh
4. cd SIGMOD20-paper
5. make : To build the paper using the results generated from benchmarks.

The paper contains graphs for the following benchmarks. The bash script contains functions to perform these experiments and plot the output. These functions can also be executed one-by-one in the order.

1. Generate Data
    a. This function generates input datasets for the experiments in the paper
    b. It uses Firehose code base to generate a simulated cyber stream
    c. The expected files at the successful completion of the function are listed in the bash script
    d. It requires RAM ~64GB and 4+ Hrs
2. Validate count and time Stretch
    a. It corresponds to Figure 1 in the paper
    b. This requires RAM ~ 64GB and 3+ Hrs
3. Validate time stretch for arbitrary datasets
    a. It corresponds to Figure 2a in the paper
    b. This requires RAM ~ 64GB and 2+ Hrs
4. Validate count stretch for buffer strategy
    a. It corresponds to Figure 2b in the paper
    b. This requires RAM ~ 64GB and 2+ Hrs
5. Validate time and count stretch with lifetime
    a. It corresponds to Figure 3 in the paper
    b. This requires RAM ~ 64GB and 2+ Hrs
6. Compute I/O and throughput
    a. It corresponds to Figure 4 and Figure 5a in the paper
    b. This requires RAM ~ 2GB and < 1 Hr
7. Scalability experiment
    a. It corresponds to Figure 5b in the paper
    b. This requires RAM ~ 2GB and more than a day
8. Instantaneous throughput
    a. It corresponds to Figure 5c in the paper
    b. This requires RAM ~ 2GB and < 1 Hr

Note: Figure 4 compares the theoretical and empirical I/O of the different LERTs. The theoretical read/write is computed based on the number of shuffle-merges while inserting observations from the stream. The empirical read/write is measured using iotop. We only plot the empirical I/O in the output plot.

Note: Figure 5c was created using the instantaneous throughput numbers in Gnumeric. The Gnumeric is included in the repo to compare with instantaneous throughput numbers generated during reproducibility. The two output files generated after the experiments are “Instantaneous_throughput_1C_1T.txt” and “Instantaneous_throughput_1024C_4T.txt”. We do not have a way to automate this process.