OS 1 Course Project

Time period: FOUR weeks(June 26 - July 23)

Goal: Build a further understanding of Operating Systems

Source data: Systor'17 trace (systor17-01.tar) from (iotta.snia.org/tracetypes/3)

Where to get the trace:

- Go to: iotta.snia.org/tracetypes/3
- You'll find the *Systor'17 Traces*, use the drop-down menu under *Actions* choose the following options:
  - "*Download README*", to get the README file
  - "*Download Sample*", to get a sample trace
  - "*view subtraces*", to go to the next level page:
    - choose "*view subtraces*" under "Systor'17 Paper Traces", to go to the subtraces download page
    - choose "*Download via HTTP*" under "Systor'17 Part 01 "
    - before a download starts, you are supposed to fill a Download License Acceptance form online by providing the required information
    - Once you click the "*Yes, I Accept*", the download will start automatically
    - The filename "*systor17-01.tar*", size is 1007MB

Analysis: Once the trace is downloaded correctlly, it's your turn to analysis and manipulate the trace data.

- Task 0: You might need to de-compress the .tar file twice to get the .csv files (e.g. 2016022207-LUN0.csv). According to the trace sample file, there will be SIX types of entries in each .csv file: *Timestamp, Response, IOType, LUN, Offset,* and *Size*
- Task 1: Separate .csv files into two independent files according to "*IOType*", there should be two files after the separation (R.csv and W.csv)
- Task 2: Sort the entries of those two files by "*Size*", if the same "*Size*", sort by "*Timestamp*", the sorted data should be written back in separate to the R.csv and W.csv accordingly
- Task 3: Collect the data access pattern based on the sorted data: collect the number of entries of each "*Size*"

Present Results: You are supposed to present the following topics:

- Two independent files in total(R.csv & W.csv), one of which holds the analized data with the "*IOType*" "R" while another holds that of "W" (Task 1). Note that the summary of the entries of R/W files should equal to the total number of entries in the orignial *systor17-01* after the decompresion.
- In each of the file, there should have two segments with a blank line between them: the first segment is the sorted data by "*Size*" (Task 2), and the second segment is the statistic analysis result (Taks 3).

- A report describes **Design** of the project, followed by the **Performance Results** that includes the **total time span** (calcuate by *T_analysis_completes - T_analysis_starts*), the **average CPU utilization**, and the **average I/O bandwidth**. Note that we assume the analysis starts at Task 0（ecompression of *systor17-01.tar*）, and ends after Task 3
- You may need to improve the system efficiency by reducing the total time span.

Assignment Requests: You must pack the following document tree as a "*[your name].zip*" file and upload it to the Bloackboard.

- [student id]
    - code
        - src_before // containing your source codes before optimizing
        - src_after // containing your source codes after optimizing
        - *README // a file showing how to run your codes
    - results // containing one R.csv and one W.csv, each of which holds sorted entries and analized results.
    - *report // your report file

Grading: You should run your code and collect results for twice: the first time when your code runs correctly , collect the time span as the base line; then start to optimize your code to make sure that the time span should be at least 10% shorter. Otherwise you fail the project.

Hints: Any kinds of optimizations based on what you learned from the OS are encouraged such as multiply threads, parallel IO, interleaved IO. You may consider the following issues during your design and testing:

- You may need to use multiple processes/threads, that some of them are handling decompression, some are handling read data, and some are analyzing.
- You may also need to think about the situation that if the main memory is not large enough to place the entire decompressed *systor17-01*, some techniques and eviction policies may be involved
- There will be a decent numbers of I/Os, you may need to consider the memory management to reudes small writes.
- In order to reduce the total time span, you may need to use different processes/threads to parallelization (e.g. one thread for decompress, one thread for read, and other thread(s) for analysis).
- Since the I/O operations during the analysis are mixed with decompress, read, and write, you need to be careful of synchronlization issue. Especially, try to make sure that two concurrent I/O, two writes don't write to the same location, always read the updated files after writes.
- You may need to write a "*shell*" to run the task 0-3 while collecting the timestamp information as part of performance results.