REPORT

The monitoring system creates the directory has 3 java files: 1. ServerDetails.java, 2. LogGenerator.java and 3. QueryCommand.java.

Check readMe file for brief description.

ServerDetails Class:

The ServerDetails class contains the properties of an individual server, like the IP Address, the CPU IDs of the 2 CPUs of the server, CPU usage of the 2 CPUs and the timestamp at which a usage was recorded. All these details are used by the LogGenerator class to create the Log file.

LogGenerator Class:

The LogGenerator class as previously mentioned creates the Log file. It logs each CPU usage for 1000 servers every minute. Therefore, writing 2000 logs every minute for 24 hrs i.e. 1440 minutes. All the logs are written to a single text file.

Each record in the log file consists the Unix time at which the log was recorded, the IP Address of the server, CPU's ID and its respective usage for that particular time.

The LogGenerator.java code accepts command line arguments if the user wants to create the log file in a particular directory. If no arguments are given, the output log file will be created in the root directory by default as log.txt

The logs are recorded by default from 2014-10-31 00:00 to 2014-10-31 23:59. These are values are fixed in the program i.e. hardcoded as constants.

Input:

Command Line Arguments (both optional) :

Log_File_Name Path_of_the_directory_at_which_the_file_must_be_created

Output:

Log file created with around 2,880,000 logs (1440 * 1000 * 2) at the desired directory with the desired Log file name.

QueryCommand:

The QueryCommand Class parses the the input query, validates the correctness of the query using regular expressions and executes the query.

Before asking for the Query, the program splits the existing log file with 24 hrs worth of log into 3 separate log files with each file consisting of 8hrs worth of log for each cpu across 1000 servers. That is each files consists of 960,000 worth of logs (8 hrs * 60 mins/hr * 1000 servers * 2 cpu/server).

We then keep track of the starting times in each of these individual log files. By doing this, we can compare the start time given in the command and the starting times on the individual files and focus only on the log files where

the given start time >= the start times of the files. This effectively allows to use to only search a subset of the logs, thus reducing run-time.

Also, the all the logs where the given start time >= the start times of the files in the current file will be cached using a hash map which gives us constant access time to determine the CPU usage at a given time range, IP Address and CPU. The cache will be cleared each time we move from one log file to another. The Key for the hash map is a combination of the IP Address and the CPU ID while the value for each key is a list of CPU usages.

There are two inputs, one is the optional command line argument to specify the path to the log file.

Input:

Command Line Arguments (optional; ignore if the log file is in the root):

Log_file_name

Stdin (case sensitive):

QUERY IP_Address CPU_ID time_start time_end

Example: QUERY 192.168.1.12 0 2014-10-31 00:00 2014-10-31 00:05

Note:

time_start > time_end

Either time_start or time_end should be between the range 2014-10-31 00:00 and 2014-10-31 23:59.

EXIT

Exits out of the system.

Output (Stdout):

CPU usage for the given CPU Id and IP across the given time range.

Example:

CPU0 usage on 192.168.1.12:

(2014-10-31 00:00, 90%), (2014-10-31 00:01, 89%), (2014-10-31 00:02, 87%), (2014-10-31

00:03, 94%), (2014-10-31 00:04, 88%)

Run time:

Worst case: For the maximum time range, 1440 minutes, the program should parse through the all 2,880,000 records. For this the run time is 1 sec.

Total time spent on the Exercise, documentation and the report: 6 to 7 hrs approx