CSL202

Design Document

2016csb1050 - Assignment 4

Model Explanation:

1) **Functioning:** Top command runs every 30 seconds to list out all the processes running at that instant of your machine. The procData hashmap stores these entries with all the necessary details. When the next top command generates its output i.e. after 30 seconds i perform two basic operations. First i check what processes have shut down. For this I have created a temporary hashmap that stores the information of what the current top command has generated. Then I iterate through procData and check which processes are missing in childData, 2the ones that are missing are the processes whose execution has been completed and shut down. Therefore I remove those processes from procData. Now I iterate over child Data to find out which processes have been running since the past i.e. exist in procData. If found I update their cpu usage as per the cpu usage duration limits and their memory usage as per the memory usage duration limits, if not found I create a new entry in the procData and insert it. So my procData efficiently stores correct information of running processes by accumulating their history. Now when i update the details I check for violations and if any process has caused it I add it to another data structure instance violators. After the current top command processing ends I create an object of Email class which does the work of sending emails to the registered users about the violating processes.
2) **Condition on top command running frequency:** I have deduced the result that the running frequency of top command can not be more than 59 secs if the cleaning process has frequency in minutes. Because if a process violates resource usage at lets say t = x and shuts down at t = x+1 (which is highly probable)

then to detect this process top command has to run in less than one minute else it can easily escape. I have set it to 30 seconds.

3) **N2 minutes cleaning and history information:** To store the history of all processes I have create a file history_info.txt which stores the output of all top commands and my code automatically cleans it every quota.window.minutes i.e. n2 . The entries to be cleant are the ones that have been residing there for the past n2 minutes. Thus the user can view the history in human readable format of past n minutes at any time even if his/her machine restarts. The data is stored on disk and automatically cleaned to avoid redundant data and data loss.
4) **processDetail class:** I have created a processDetail class which is basically a data abstraction model for the process data. It has some objects and some set functions. All my process details stored in hashmap are pointers from Process ID to these objects.
5) **Email class:** objects of this class are primarily used to separate the email sending operations from the main code. The class calls appropriate functions to

Design Choices/Observations:

1) I have used process ID to distinguish between the processes, because a process is a running instance of a program. So multiple instances of same program are different processes. The only unique ID is PID, they may run under same user and same command name but are actually different processes.
2) When the computer restarts all processes are re-instantiated and hence are different processes. So all we need to store is past n-minutes data to keep history maintained for the user.
3) Lets say a process P1 executed and successfully shut down, now its process ID is available for any new process coming up. There is a chance that a new process may be assigned the same process Id, so when we are checking

whether it is the same old process I have compared user and command name also. That correctly tells me whether this PID refers to the same process that have been running across 2 runs to top or a different process being assigned the process ID of a process that completed and shut down in between the 2 runs.

4) While parsing the file to delete entries beyond past n minutes I have considered the date also so as to catch the entries that may be some days old.

Important:

1) Do not run the code using IIT Ropar lan ethernet because private ports are blocked and hence code will crib while sending mail. Please use some other source of internet while running.
2) Make sure you have the necessary jar files to run the program