# QUANTIL CODING EXERCISE

Design Document

**By:**

**Anoop S Somashekar**

**Document status overview**

**Document title:** Design Document

**Identification**: Design Document.pdf

**Author**:

Anoop S Somashekar
**Document status**: Complete

| Version | Primary Author(s) | Description of Version | Reason of Change | Date |
|---|---|---|---|---|
| 1.0 | Anoop | Final | Documentation | 08/02/2015 |

# 1. INTRODUCTION

## 1.1 Design Overview

This document describes the various design aspects of the “Quantil Coding Exercise”. The intent is to generate log files for CPU usage of 1000 dual core machines for a single day on every minute and build a query engine so that given a machine ID, CPU ID and time interval retrieve the CPU usage log of that particular machine very efficiently.

In Phase 1, the logs are generated for 1000 dual core machines by randomly generating IP address and CPU usage and stored in a text file.

In Phase 2, the generated log file is read into an in memory data structure and the query results are returned by looking for a specific time interval in the query.

## 1.2Goals and objectives

### 1.2.1 Phase 1

To generate CPU usage log files for 1000 dual core machines for a single day.

### 1.2.2 Phase 2

To build a query engine which returns the CPU usage log data for a given machine and time interval.

### 1.2.3Statement of Scope

The generator software will take path as a command line argument and generates log files in the specified path. The Query software will take path as a command line argument which is the repository of log files generated in Phase I. Output for query software will be the CPU usage log data for a given machine and time interval.

# 2 DETAILED DESCRIPTION OF COMPONENTS

## 2.1 Monitoring System

### 2.1.1 Generator

CPU usage log files are generated for by storing log data in one file per machine. File name is also generated dynamically using machine IP address and date for which the log data is being generated. This is one of the key design decision for faster query result. In this way, the look up for any query is confined to single file and query result will returned in one tenth of a second. The log files are stored in binary format rather than ASCII to compress the log data efficiently. The output directory for log files has to be passed as a command line argument and an optional second parameter for date can be passed. If the optional date parameter is not passed in the command line then the default date used for log generation is “2016-01-01”.

- private ArrayList<String> createIPAddrList(int size) – Returns the list of IP addresses.
- private void generateLog(String date, String ipAddress) – generates log file for a given machine and date.

### 2.1.2 Query

This module will return the result based on the query input. Query will be of the following format:

QUERY IP_ADDR CPU_ID YYYY-MM-DD HH1:MM1 YYYY-MM-DD HH2:MM2

First the query input is validated to check if it is in the above mentioned format, otherwise an error message is thrown to the standard output. Since log files are generated per machine per date and file name for any log file is the combination of IP_ADDR and YYYY-MM-DD, we read the query specific file into an in memory hash map where UNIX time and CPU id is the key and CPU usage is value. Based on the time interval specified in the query, all the CPU usage data are returned for a given CPU ID. Since the log file entry is in UNIX time format and query time interval is in date format, there needs to be conversion from UNIX time to date format and date format to UNIX time which will be provided by Util component below.

- private HashMap<String, Integer> readLog(String date, String ipAddress) – returns log data in hash map for a given IP address and date.
- private boolean validateInput(String inputLine) – validates query input and return true if the query input is valid otherwise return false.
- private void handleQuery(String ip, String core, String d1, String t1, String d2, String t2) – displays query results to the standard output for a given IP address and date.

### 2.1.3 Util

This module will provides utility functions for Query and Generator components. The default value of number of servers and number of cores are 1000 and 2 respectively. If the user intends to change this settings then it has to be updated in configuration property file. The methods getNoOfServers and getNoOfCores first checks the configuration property file, if it is present then it returns as per the property file otherwise it returns the default values.

- public static String generateFName(String ipAddr, String date) – Returns file name as ipAddr_date given ipAddr and date.
- public static String getFormattedDate(long unixTimeMS) – Returns formatted date “YYYY-MM-DD HH:MM
- public static long getUnixTimeMS(String dateString, String time) – Returns Unix time in milli seconds given date in YYYY-MM-DD and time in HH:MM format.
- public static int getNoOfServers() – Returns number of servers for which the log files has to be generated.
- public static int getNoOfCores() – Returns the number of cores in each machine.

### 2.1.4 PropertyFile

This module reads the configuration property file and implements get function for all the parameters specified in the property file.

- public static ReadPropFile getInstance () – Returns single ton object of ReadPropFile class.
- public String getNoOfServers () – Returns number of servers as per configuration property file. If the property file is not present then it returns null.
- public String getNoOfCores() - Returns number of cores as per configuration property file. If the property file is not present then it returns null.