Log Percentiles – Version 2

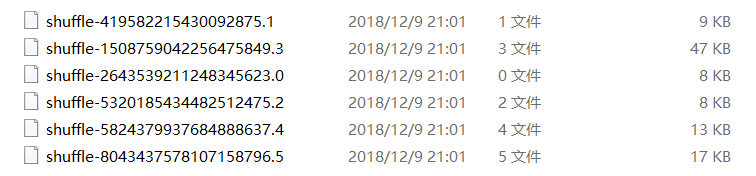
## Prerequisites

* JDK8 or later
* Apache Maven
* Ensure JAVA\_HOME environment variable is set and points to your JDK installation

## Solution

For many real-world, distributed input sources, it’s computationally or operationally impractical to load all access logs into single JVM, but still possible to process log file one by one continuously until we finish all.

Before we dive into spark(Version 3), I will demonstrate how we tackle this problem(sorting) in a general way. First, we need to get a response-time’s range sample by walking through all access logs then distribute each log into the correct shuffle file by comparing its response time with sample value. We call this as “shuffle”. After finishing shuffle stage, we would have lots of shuffle files named by (shuffle-prefix)+(partition-index)



In this case, the minimum in “shuffle-419582215430092875.1” is greater than the maximum in “shuffle-2643539211248345623.0” for we distribute log message by range partitioner. So, the last item in the last partition(shuffle-8043437578107158796.5) receives the largest number.

Then we sort log by response time in asc order within every shuffle file and use a counter to record how many records in every shuffle file

Let’s suppose that N is the length of list, so the target of 90% of requests is (0.9 \* N), Then we need to locate in which shuffle file the target is. The algorithm is very straightforward, we initialize an int variable “current” as the (number of logs in shuffle file 0), then compare with our target. If current < target means that the target is not in this shuffle file, then we set current = current + (number of logs in shuffle file 1) then compare them again. Eventually we could locate the correct shuffle file. And the line number is: {(number of logs in current shuffle file) – (current - target)}

Then we compute 95% and 99% in the same way.

Ordering in spark is quite similar as this except spark does in parallel.

## Configuration

Configurations and defaults are discussed in more detail below.

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | DESCRIPTION | TYPE | DEFAULT |
| log.dir | The access log folder path  You can change it with “—log.dir=/path/to/log” | String | /var/log/httpd |
| shuffle.partitions | Number of partitions | Int | 5 |
| shuffle.dir | Temp shuffle folder | String | /tmp/shuffles |
| shuffle.prefix | Prefix of shuffle files | String | shuffle- |
| shuffle.deleteOnExit | Whether delete shuffle files after finish execution | Boolean | true |

## Execute

Compile source code by maven or use pre-compiled jar file in target folder, then run the jar file in console(change **--shuffle.deleteOnExit=false** if you want to see what had happened during shuffle stage.)

java -jar percentiles-v2-1.jar --log.dir=C:\work\var\log\httpd --shuffle.dir=C:\work\var\log\shuffles --shuffle.deleteOnExit=false

