Имеется access-лог web-сервера. Файл со следующей структурой.

192.168.32.181 - - [14/06/2017:16:47:02 +1000] "PUT /rest/v1.4/documents?zone=default&\_rid=6076537c HTTP/1.1" 200 2 44.510983 "-" "@list-item-updater" prio:0  
192.168.32.181 - - [14/06/2017:16:47:02 +1000] "PUT /rest/v1.4/documents?zone=default&\_rid=7ae28555 HTTP/1.1" 200 2 23.251219 "-" "@list-item-updater" prio:0  
192.168.32.181 - - [14/06/2017:16:47:02 +1000] "PUT /rest/v1.4/documents?zone=default&\_rid=e356713 HTTP/1.1" 200 2 30.164372 "-" "@list-item-updater" prio:0

У каждой записи есть HTTP-код ответа (9-е поле, в первом примере "200") и время обработки запроса в миллисекундах (11-е поле, в первом примере: "44.510983"). Каждый день оператор выполняет анализ лога локализуя диапазоны времени, когда доля отказов сервиса превышала указанную границу. С этими инцидентами позже разбирается группа разработки. Требуется написать алгоритм читающий access-лог и выполняющий анализ отказов автоматически.

Отказом считается запрос завершившийся с любым 500-м кодом возврата (5xx) или обрабатываемый дольше чем указанный интервал времени.

На входе программе дается:

\* поток данных из access-лог'а;  
\* минимально допустимый уровень доступности (проценты. Например, "99.9");  
\* приемлемое время ответа (миллисекунды. Например, "45").

На выходе программа предоставляет временные интервалы, в которые доля отказов системы превышала указанную границу, а также уровень доступности в этот интервал времени. Интервалы должны быть отсортированы по времени начала.

Пример использования программы:

$ cat access.log | java -jar analyze -u 99.9 -t 45  
13:32:26 13:33:15 94.5  
15:23:02 15:23:08 99.8

Требования и ограничения.

\* максимальный размер access-log'а не позволяет загрузить все записи в оперативную память. Анализ необходимо выполнять потоково. Объем доступной памяти 512 мегабайт (-Xmx512M)

\* разработчику нужно учитывать, что программа должна обрабатывать большие объемы данных и, при выборе решений, ориентироваться на оптимизацию расхода системных ресурсов (процессор, память)  
\* допускается использования версии JDK 8-11 и новее;  
\* допускается использование сторонних библиотек;  
\* сборка проекта должна осуществляться утилитой Apache Maven;  
\* проект должен содержать автоматические тесты фиксирующие поведение системы в объеме по вашему усмотрению;  
\* результатом сборки должен являться self-executable jar-файл, который можно запустить командой "java -jar ..."