Log Parser

A Springboot application to parse the logs of web requests stored in file and persist the following metrics in the database:

1. *Api*
2. *Auth_status*
3. *Enterprise_id*
4. *Enterprise_name*
5. *Http_status_code*
6. *Ip_address*
7. *Log_time_stamp*
8. *request_body*
9. *request_from*
10. *Response_time*
11. *user_agent.*

Approach: Since all the logs in the file follow a similar pattern, I've used a regular expression for each line to find out a metric and get the value of that metric. Let's take an example of extracting the Request-Type from the log:
This particular value in the log will present as `!Request-Type=GET#`
The regular expression created for this metric is `!Request-Type=.[^\\#]*`,which means that the value shall end with a `#` and all the characters after `=` will be the value of that particular metric. While processing each line, we search for a particular metric using the regular expression and store them in the database.

Performance Improvement: To improve the performance of the parser, instead of saving one record at a time in the database, I have used batch commits for persistence which will save the list of objects in the database in batches, which has improved the performance by **8 times.** The 250MB file which took almost 6 minutes to process was brought down to 35 seconds using hibernate batch commits.
Another improvement made was regarding sequence generator. Instead of auto generating the IDs for each rows, I have used custom sequence generator which is way more efficient than autoincrementing the id.

Assumptions made:

1. The assumption made is that a new log starts from a new line.
2. For the IDE settings to improve the performance, increase the memory heap to 8gb so that it makes the processing faster. Note that it will be even faster if more heap is allocated.

Database Design: For storing the data, I've used a Relational Database PostgreSQL. There is a single table in the database with the following columns: *api, auth_status,*

enterprise_id, enterprise_name, http_status_code, ip_address, log_time_stamp, request_body, request_from, response_time, user_agent.

The additional parameter that could have been taken as input would be the *response_time* threshold ,so that for a particular log, if the response time is greater than the threshold then only we should process it , otherwise we can skip that log.

Scope of Improvement:

1. A multithreaded approach could have been implemented, so that each thread processes different parts of the file, which could have improved file processing time multiple times for files of very large size.
2. Using a spring batch process to process the file.
3. A rate limiter on the end point for processing the file could have been implemented to refrain users from uploading multiple files in a miniscule time frame, which would have decreased the efficiency.
4. A better regular expression and efficient way to process a particular log.
5. Finding the perfect balance between heap allocation, batch size and number of records stored in the list to save at a particular instance of time to improve the performance.