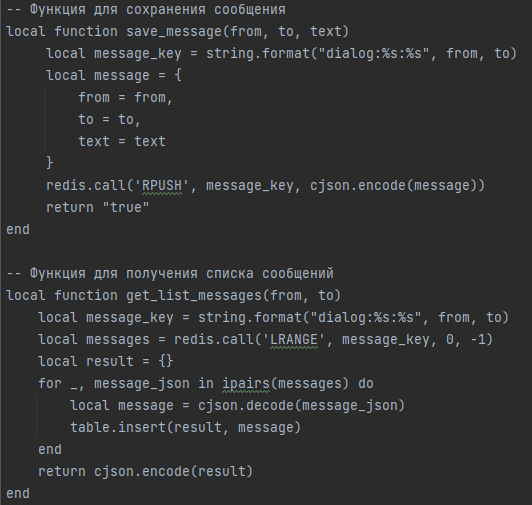
## **Отчёт по теме «In-memory СУБД»**

1. В качестве in-memory СУБД используем Redis. Выделяем систему диалогов в отдельный микросервис. Работу с Redis производим при помощи lua-функций.



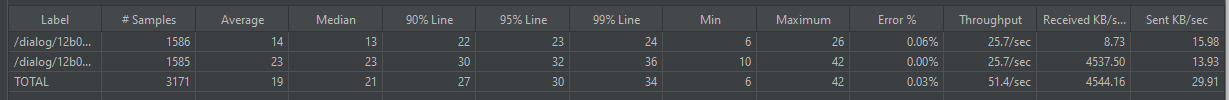
## Проводим нагрузочное тестирование при помощи JMeter.

## **Сценарий без Redis (СУБД Postgres)**

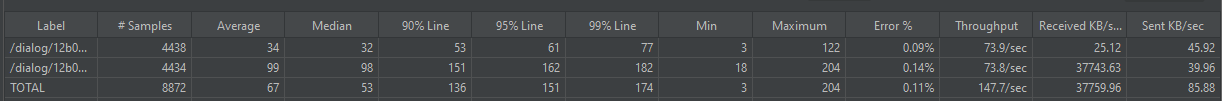
1. Отправка пользователю сообщения (эндпойнт /dialog/user/send) с количеством одновременных запросов 1/10/100/1000 на протяжении 60 секунд.
2. Получение пользователем списка сообщений (эндпойнт /dialog/{userId}/list) с количеством одновременных запросов 1/10/100/1000 на протяжении 60 секунд.

## **Основные выводы**

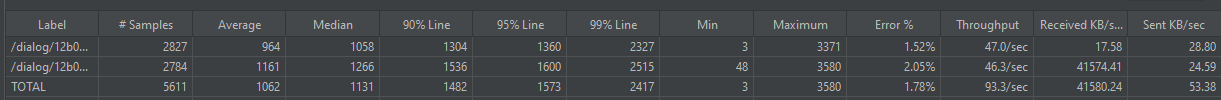
* 1 поток 60 сек



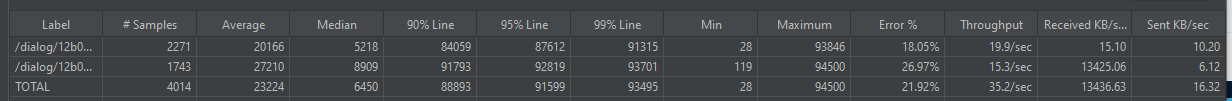
* 10 потоков 60 сек



* 100 потоков 60 сек



* 1000 потоков 60 сек



По первому этапу можно сделать следующие выводы:

1. эндпойнт /dialog/user/send

* По мере увеличения количества потоков 1 -> 10 -> 100 -> 1000 на 90 перцентиле зафиксированы следующие изменения:
* latency: 22 -> 53 -> 1304 -> 84059.
* Изменение throughput: 25,7/sec -> 73,8/sec -> 46,3/sec -> 15.3/sec

С увеличением потоков свыше 100 скорость записи и объем передаваемой информации снижаются.

* При 100 - 1000 потоках появились ошибки в асинхронной обработке запросов

` Resolved [org.springframework.web.context.request.async.AsyncRequestNotUsableException: ServletOutputStream failed to write: java.io.IOException: Connection reset by peer] `.

2. эндпойнт /dialog/{userId}/list

* По мере увеличения количества потоков 1 -> 10 -> 100 -> 1000 на 90 перцентиле зафиксированы следующие изменения:
* latency: 30 -> 151 -> 1536 -> 91793.
* Изменение throughput: 25,7/sec -> 73,9/sec -> 47,0/sec -> 19,9/sec

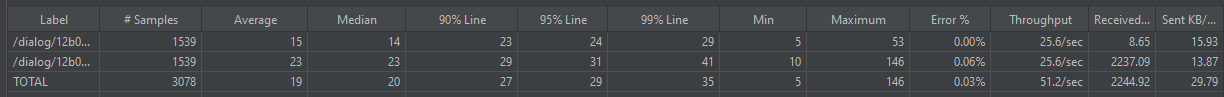
С увеличением потоков свыше 100 скорость чтения и объем передаваемой информации снижаются.

* При 100 - 1000 потоках появились ошибки в получении токена авторизации ` Resolved [org.springframework.web.context.request.async.AsyncRequestNotUsableException: ServletOutputStream failed to write: java.io.IOException: Connection reset by peer] `.

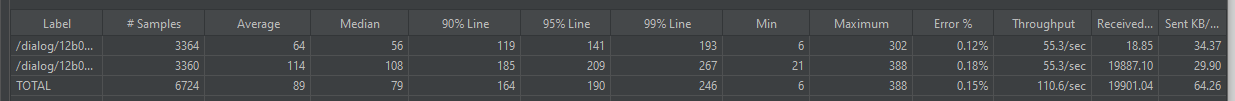
Данные ошибки можно связать с увеличением времени обработки запроса, клиент может закрыть соединение из-за таймаута.

**Сценарий с Redis**

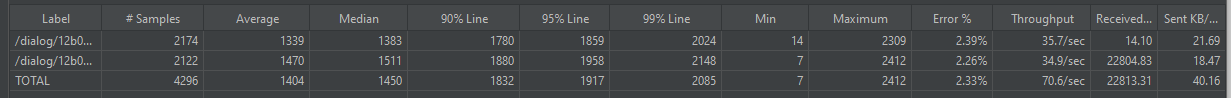
* 1 поток 60 сек



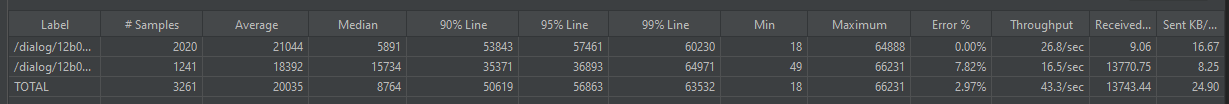
* 10 потоков 60 сек



* 100 потоков 60 сек



* 1000 потоков 60 сек



По второму этапу можно сделать следующие выводы:

1. эндпойнт /dialog/user/send

* По мере увеличения количества потоков 1 -> 10 -> 100 -> 1000 на 90 перцентиле зафиксированы следующие изменения:

- latency: 23 -> 119 -> 1780 -> 53843.

- Изменение throughput: 25,6/sec -> 55,3/sec -> 35,7/sec -> 26,8/sec.

С увеличением потоков свыше 100 скорость записи и объем передаваемой информации снижаются. Скорость записи в in-memory БД незначительно снизилась (какая-то аномалия 😊), объем передаваемой информации снизился.

* При 1000 потоках ошибки в работе сервиса практически отсутствовали.

2. эндпойнт /dialog/{userId}/list

* По мере увеличения количества потоков 1 -> 10 -> 100 -> 1000 на 90 перцентиле зафиксированы следующие изменения:

- latency: 29 -> 185 -> 1880 -> 35371.

- Изменение throughput: 25,6/sec -> 55,3/sec -> 34,9/sec -> 16,5/sec.

С увеличением потоков свыше 100 скорость записи и объем передаваемой информации снижаются. Скорость чтения в in-memory БД изменилась незначительно, объем передаваемой информации при этом снизился.

* При 1000 потоках ошибки в работе сервиса практически отсутствовали.

Итоговые выводы: в текущей реализации системы переключение работы сервиса диалогов на «in-memory СУБД» не привел к значительному увеличению скорости работы, однако значительно увеличил стабильность работы системы за счет отсутствия проблем с асинхронностью обработки запросов.