# Отчет по тесту "Надежность" (Leak)

Данные испытания проводились в рамках задачи - [XSETA-2](http://jira.xset.ru:8080/browse/XSETA-2) (Проведение тестов на поиск максимума).

* [Конфигурация среды для нагрузочного тестирования](#scroll-bookmark-2)
* [Цели тестирования](#scroll-bookmark-3)
* [Выводы](#scroll-bookmark-4)
* [Рекомендации](#scroll-bookmark-5)
* [Описание тестов](#scroll-bookmark-6)
  + [№](#scroll-bookmark-7)
  + Надежность
* [Результаты тестирования](#scroll-bookmark-12)
  + Надежность

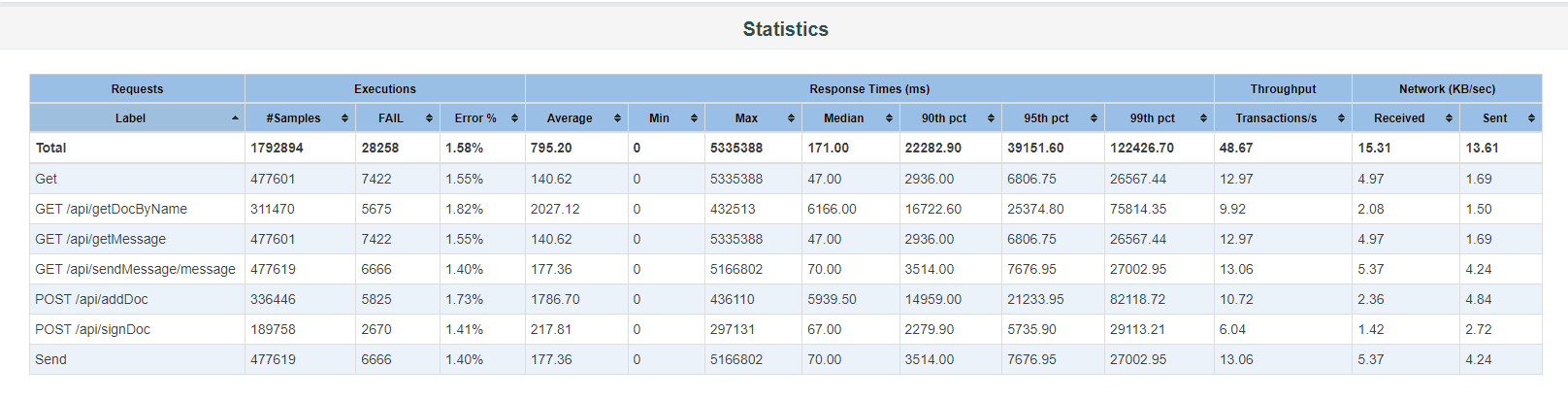
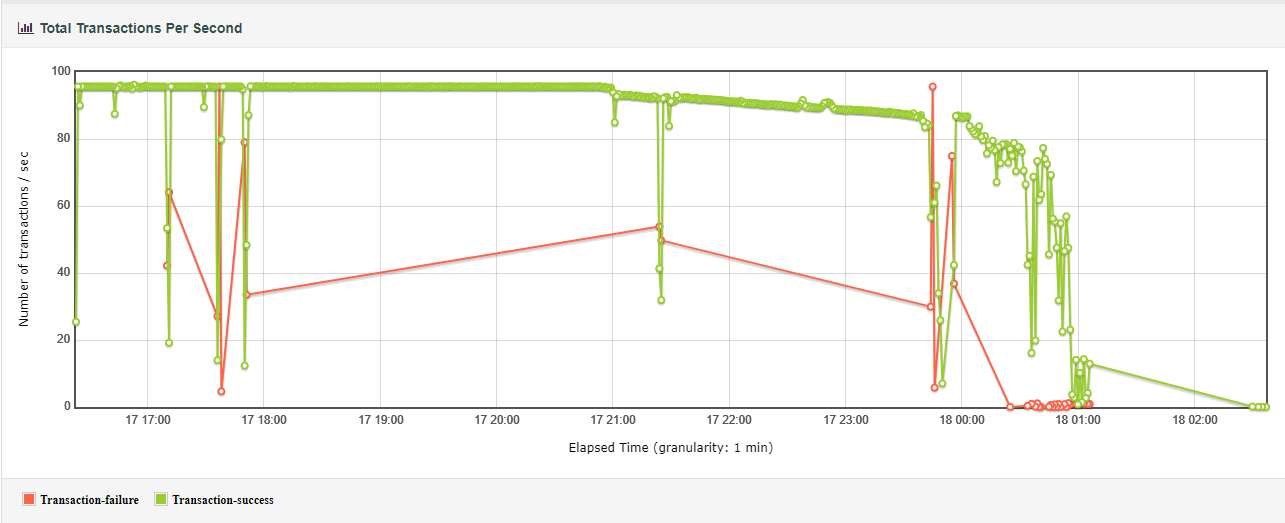
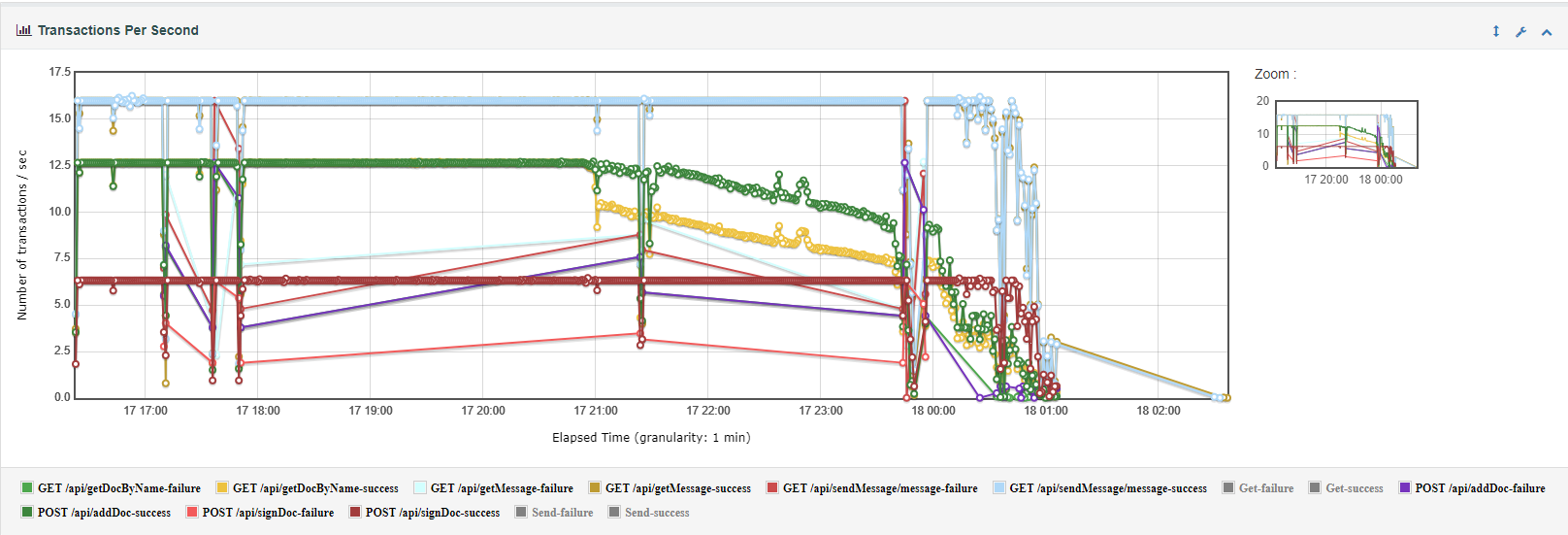
# ****1 Конфигурация среды для нагрузочного тестирования****

|  |  |  |
| --- | --- | --- |
| № | Наименования ресурса | Значение |
| 1 | ОС | Выпуск: Windows 11 Домашняя для одного языка  Версия: 22H2  Сборка ОС: 22621.2428  Взаимодействие: Windows Feature Experience Pack 1000.22674.1000.0  разрядность: 64 bit. |
| 2 | CPU | 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz   2.70 GHz  4 ядра |
| 3 | RAM | 16,0 ГБ |
| 4 | Жесткий диск | 512 Гб,  тип SSD. |

# ****2 Цели тестирования****

* Определение максимальной и пиковой производительности системы
* Выявление «узких мест»

# ****Выводы****



На графике "Total Transactions Per Second" видно, что максимальная производительность была достигнута на уровне около 100 запросов в секунду в период с 17:00 до 21:00 для успешных транзакций (зеленая линия). В этот период система работала стабильно.В таблице "Statistics" видно, что для периода максимальной нагрузки с 20:41 до 21:01:

* **Общее количество операций**: 1792894
* **Количество ошибок**: 28258(Error %: 1.58%)

В период максимальной нагрузки с 17:00 до 21:00:

* **GET /api/getDocByName-success:** Количество успешных транзакций стабильно держалось на уровне 15 транзакций в секунду.
* **GET /api/getMessage-success:** Количество успешных транзакций стабильно держалось на уровне 12 транзакций в секунду.
* **POST /api/addDoc-success:** Количество успешных транзакций стабильно держалось на уровне 7 транзакций в секунду.
* **POST /api/signDoc-success:** Количество успешных транзакций стабильно держалось на уровне 5 транзакций в секунду.

График демонстрирует значительное увеличение количества заблокированных таблиц после 21:00, что указывает на проблемы с транзакциями и конкуренцию за ресурсы.

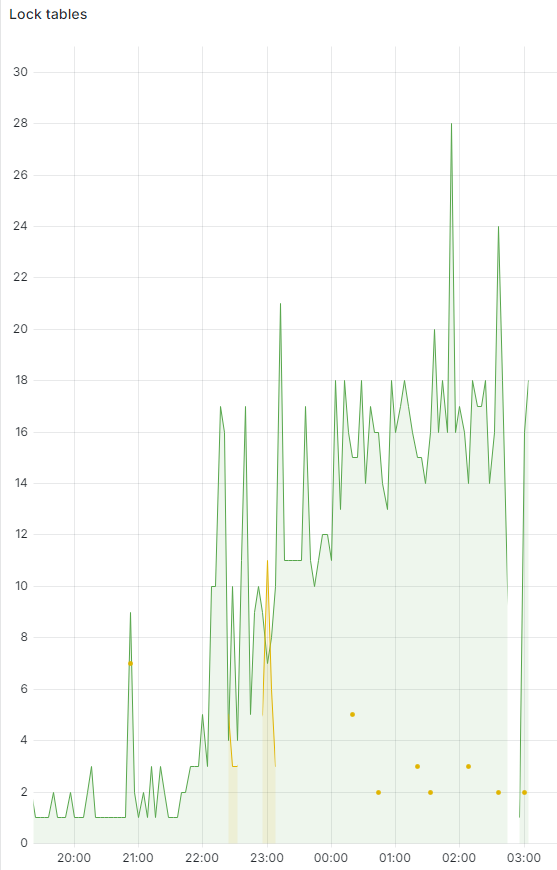
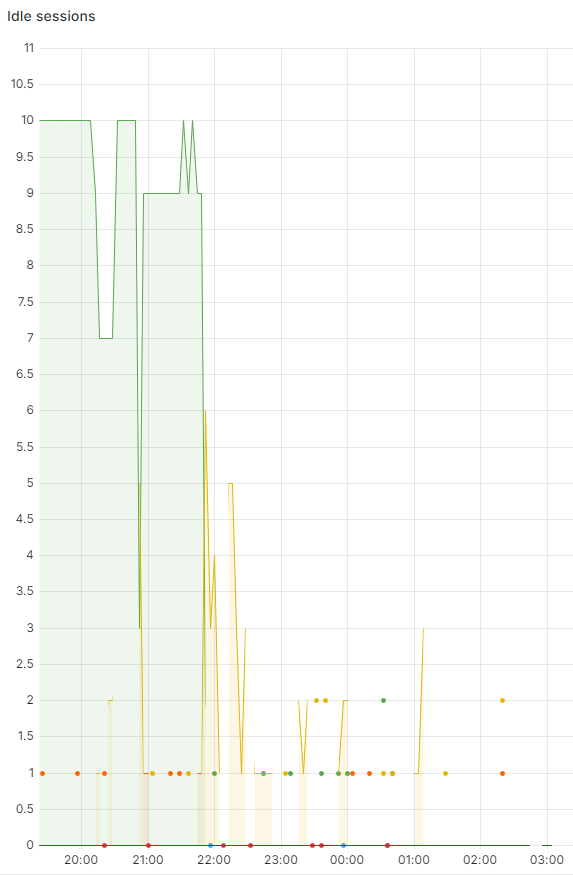


График показывает резкое увеличение числа неактивных сессий после 21:00, что может указывать на долгосрочные транзакции и проблемы с управлением сессиями.



# ****4 Рекомендации****

1. Анализ и оптимизация запросов к базе данных для уменьшения заблокированных таблиц и повышения эффективности транзакций.

* 1. Улучшение механизмов управления сессиями для минимизации числа неактивных сессий и повышения производительности.
  2. Рассмотрение балансировки нагрузки для равномерного распределения запросов и уменьшения конкуренции за ресурсы

# ****5 Описание тестов****

**Профиль НТ**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя операции** | **Интенсивность ОП/ч** | **Интенсивность в %** |
| |  | | --- | | 1 | | |  |  | | --- | --- | | 1 | POST /api/signDoc | | |  | | --- | | 12000 | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  | 10% | |
| |  | | --- | | 2 | | |  |  | | --- | --- | | 2 | GET /api/sendMessage | | |  | | --- | | 23684 | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  | 25% | |
| |  | | --- | | 3 | | |  |  | | --- | --- | | 3 | GET /api/getMessage | | |  |  | | --- | --- | | 29990 |  | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  | 25% | |
| |  | | --- | | 4 | | |  |  | | --- | --- | | 4 | POST /api/addDoc | | |  |  | | --- | --- | |  | 29990 | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  | 20% | |
| |  | | --- | | 5 | | |  |  | | --- | --- | | 5 | GET /api/getDocByName | | |  |  | | --- | --- | | 23684 |  | | |  |  |  |  | | --- | --- | --- | --- | |  |  |  | 20% | |

## ****5.1 Поиск максимальной нагрузки****

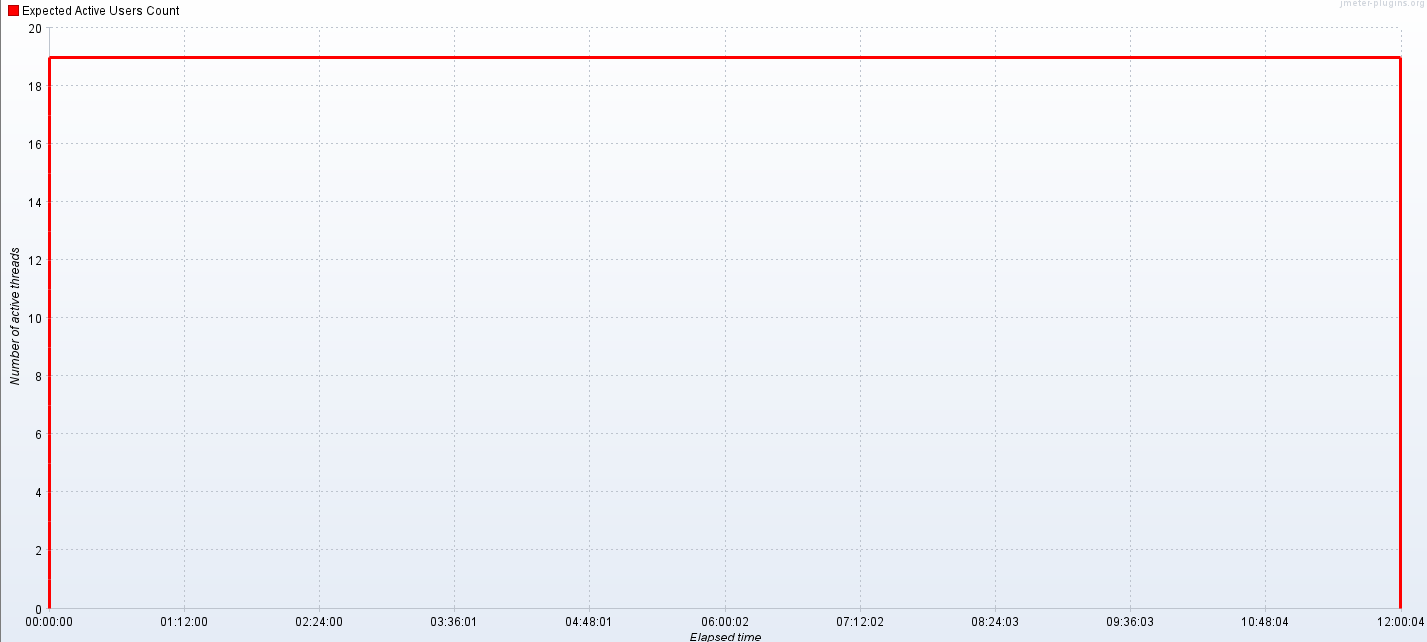
SLA:

1. **Допустимый процент ошибок:** не более 5%
2. **Допустимая загрузка CPU:** не более 80%

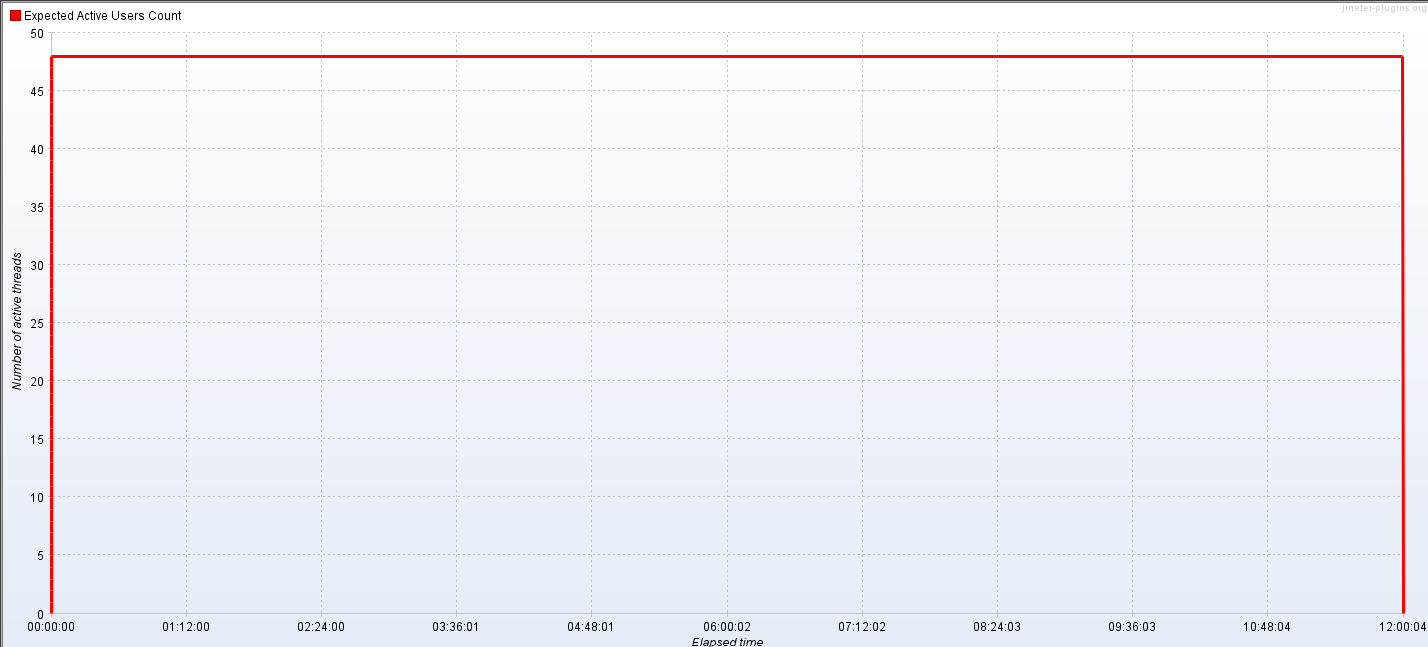
* 80% от максимальной ступени
* Распределение нагрузки по сценариям:

Характер нагрузки

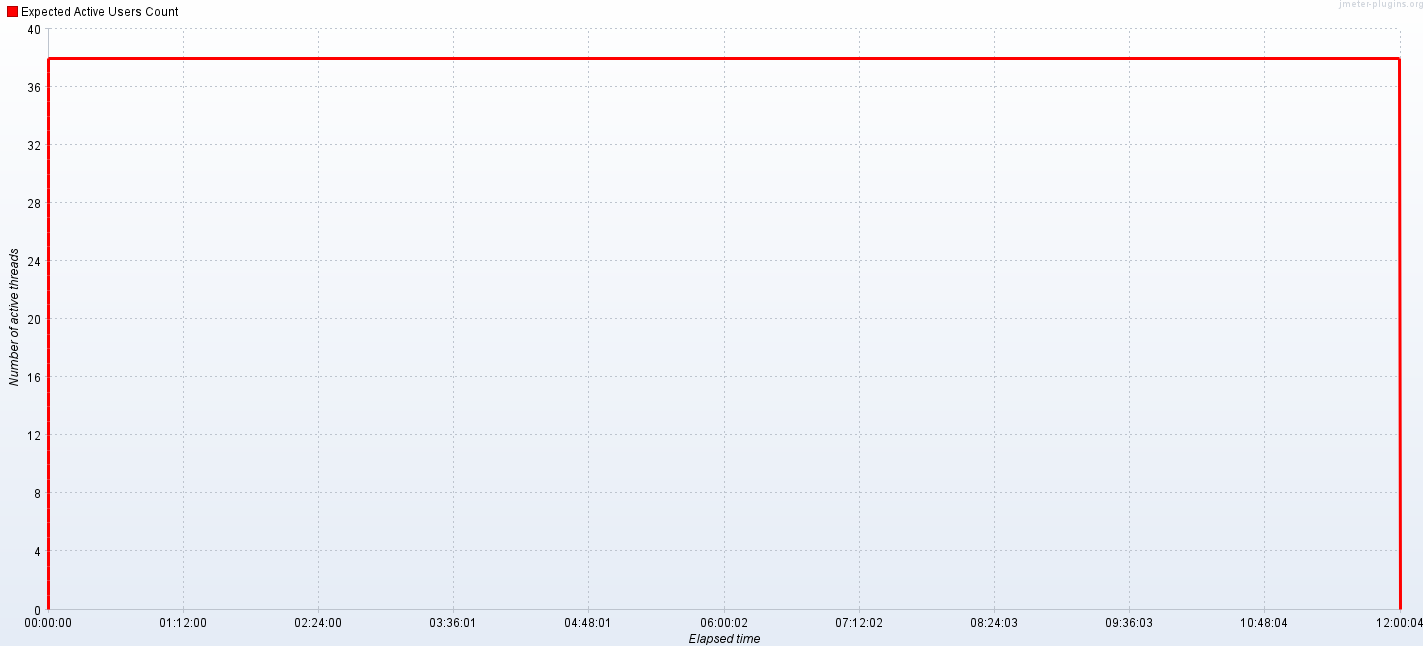
1. /api/signDoc



1. /api/sendMessage
2. /api/getMessage



1. /api/addDoc
2. /api/getDocByName



# ****6 Результаты тестирования****

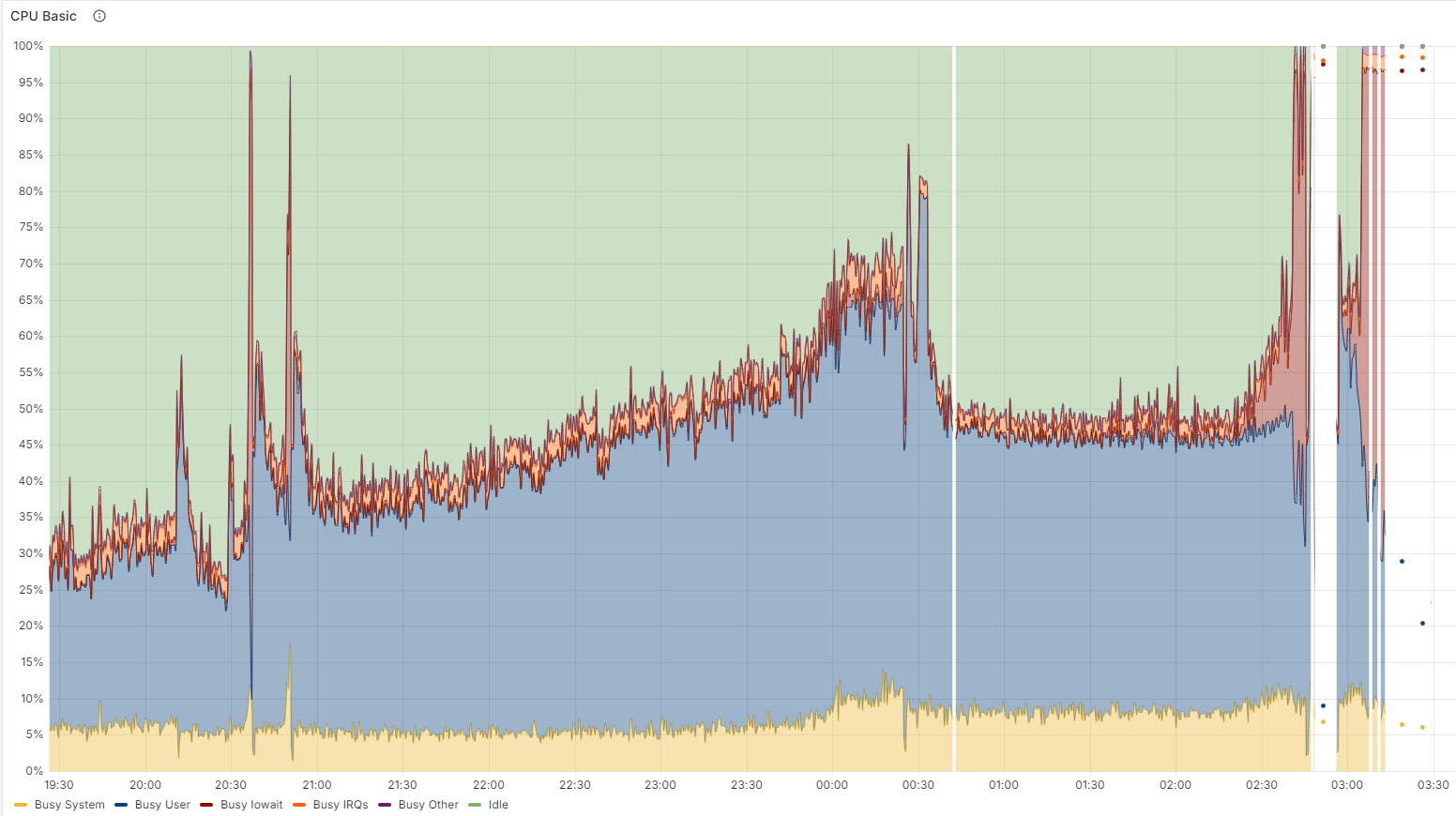
## ****6.1 Тесть на надежность****

Данное тестирование проведено в соответствии с МНТ - [ссылка МНТ](http://wiki.xset.ru:8090/pages/viewpage.action?pageId=3440676)

Начало теста 17/07/24, 19:23

Окончание теста 18/07/24, 07:23

**График - Утилизация CPU(Сервер)**

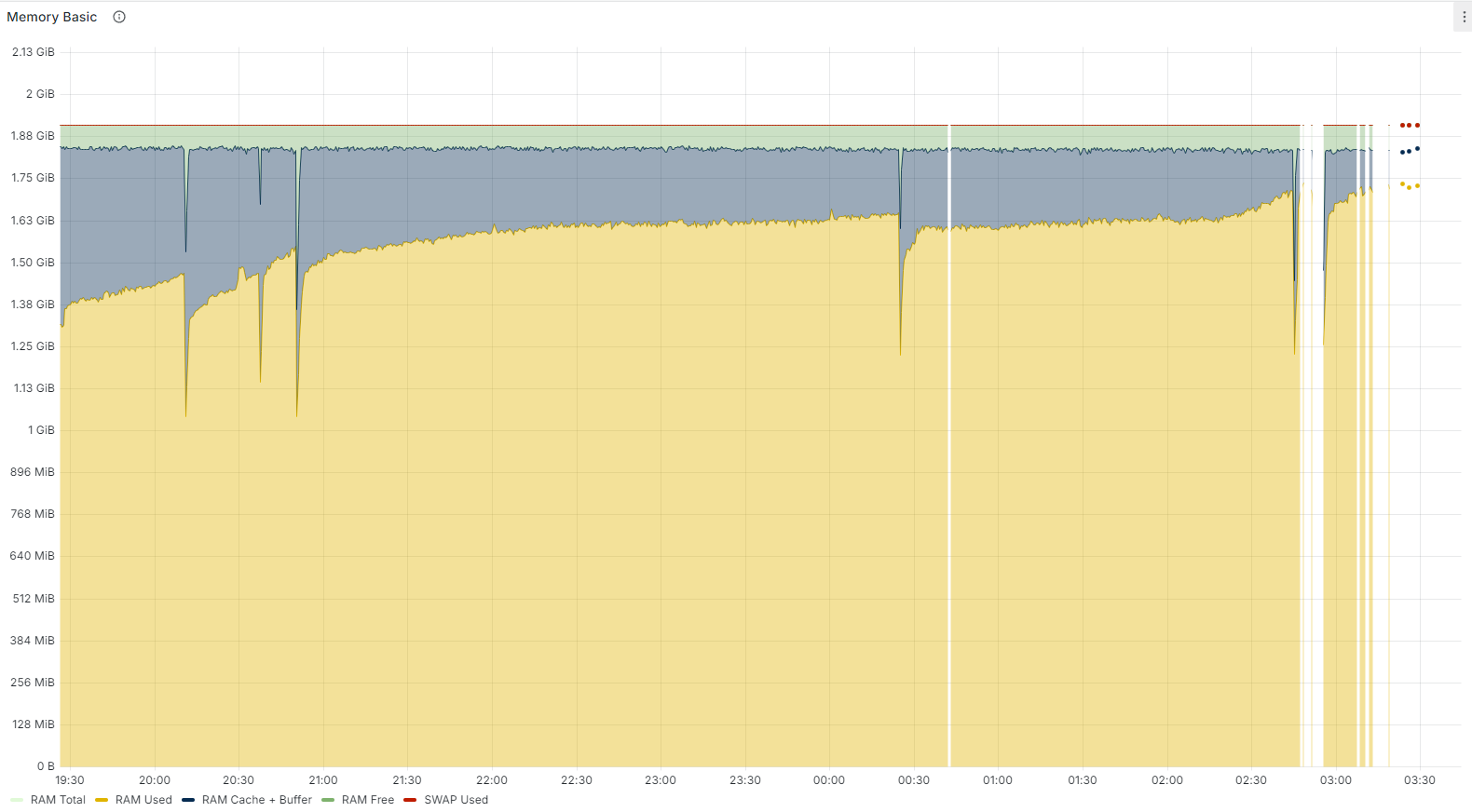


На графике выше видно, что:

* В начале теста (19:23) использование CPU составляет около 35%.
* В период с 19:23 до 20:30 нагрузка на CPU колеблется, оставаясь в пределах 30-40%.
* С 20:30 до 22:45 наблюдаются значительные пики нагрузки, достигающие до 100%, с последующим возвратом к уровню около 40%.
* Начиная с 22:45 нагрузка на CPU постепенно увеличивается с 50%.
* С 22:45 до 03:00 нагрузка на CPU постепенно увеличивается, начиная с 50% и достигая 85% к концу периода.
* После 03:00 наблюдаются значительные пики нагрузки, достигающие 90-100%, что приводит к нестабильности системы.

Максимальная производительность системы достигается в период с 22:45 до 03:00, так как после этого начинается значительное увеличение нагрузки до 90-100% на CPU, что приводит к нестабильности системы. Исходя из этих данных, можно сделать вывод, что система стабильно работала при нагрузке до 70-80%, но дальнейшее увеличение нагрузки приводит к снижению стабильности работы системы.

**График - Утилизация RAM(Сервер)**



На графике выше видно, что:

На графике выше видно, что:

* В начале теста (19:23) утилизация RAM составляет примерно 1.38 GiB.
* В период с 19:23 до 21:00 наблюдается стабильное увеличение использования RAM, доходя до 1.75 GiB.
* С 21:00 до 21:30 использование RAM продолжает медленно расти, стабилизируясь на уровне около 1.75 GiB.
* С 21:30 до 02:00 утилизация RAM стабилизируется и остается на уровне около 1.88 GiB, с несколькими незначительными спадами.
* С 02:00 до 03:30 утилизация RAM остается на уровне около 1.88 GiB, с редкими резкими спадами до 1.50 GiB.

На протяжении всего теста использование RAM остается стабильным, с плавным ростом в начале и дальнейшей стабилизацией. Максимальное значение использования RAM достигает 1.88 GiB, что является устойчивым показателем для текущей конфигурации системы.

**График - Скорость чтения дисковой подсистемы(Сервер)**

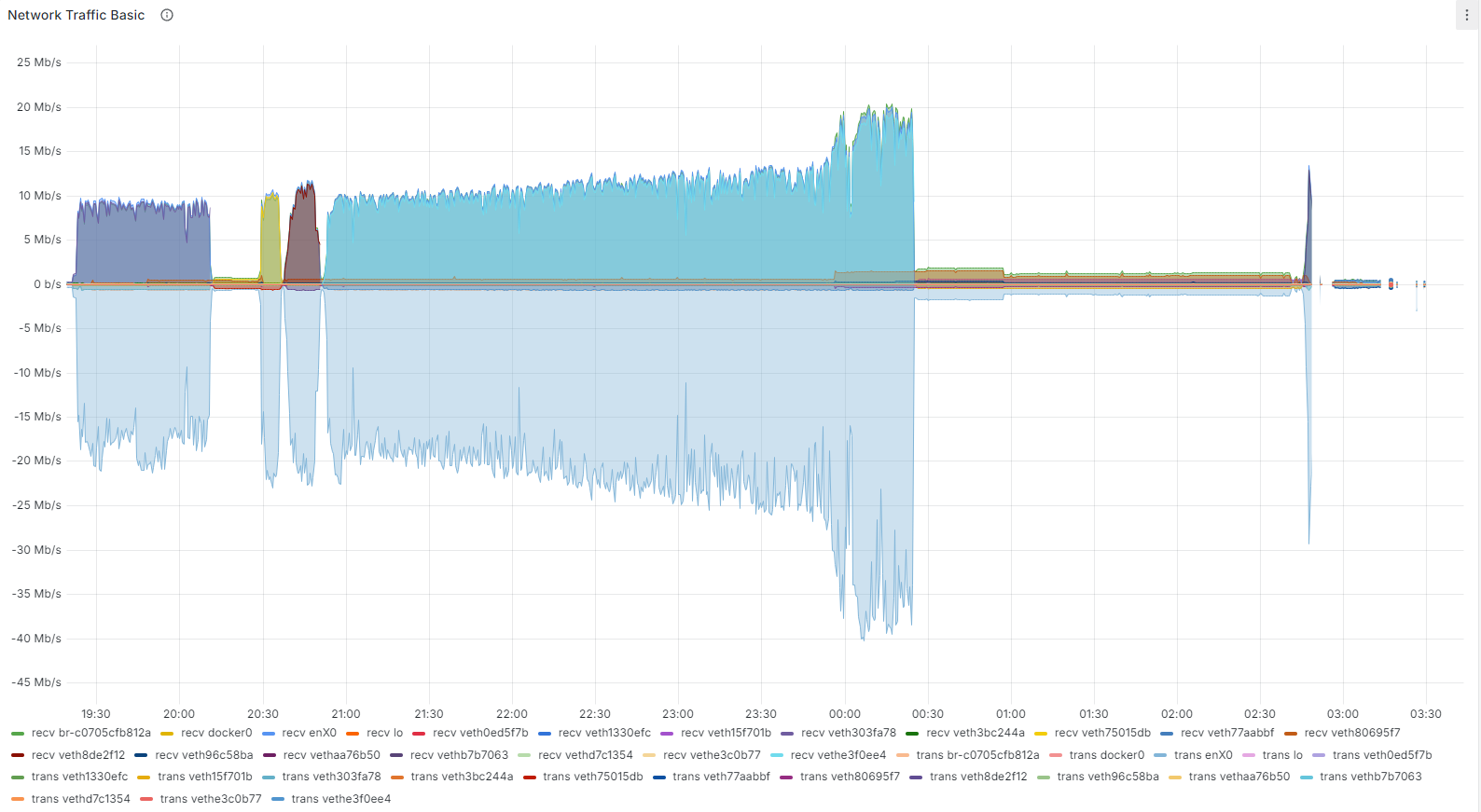


На графике выше видно, что:

* В начале теста скорость чтения дисковой подсистемы составляет около 0 MB/s.
* В течение теста наблюдаются значительные колебания скорости чтения, включая несколько глубоких спадов, доходящих до -65 MB/s.
* Средняя скорость чтения составляет примерно 5.76 MB/s.
* Максимальная скорость чтения достигает 52.7 MB/s.
* Скорость записи также демонстрирует значительные колебания, с несколькими пиками, достигающими до 8.59 MB/s.
* Средняя скорость записи составляет примерно 438 kB/s.
* Минимальная зафиксированная скорость записи – 52.7 kB/s, максимальная – 8.59 MB/s.

В течение теста скорость чтения дисковой подсистемы варьируется, с несколькими значительными пиками и глубокими спадениями. Максимальная скорость чтения достигает 52.7 MB/s, что является высоким показателем для текущей конфигурации системы.

**График - Потребление сетевых ресурсов(Сервер)**

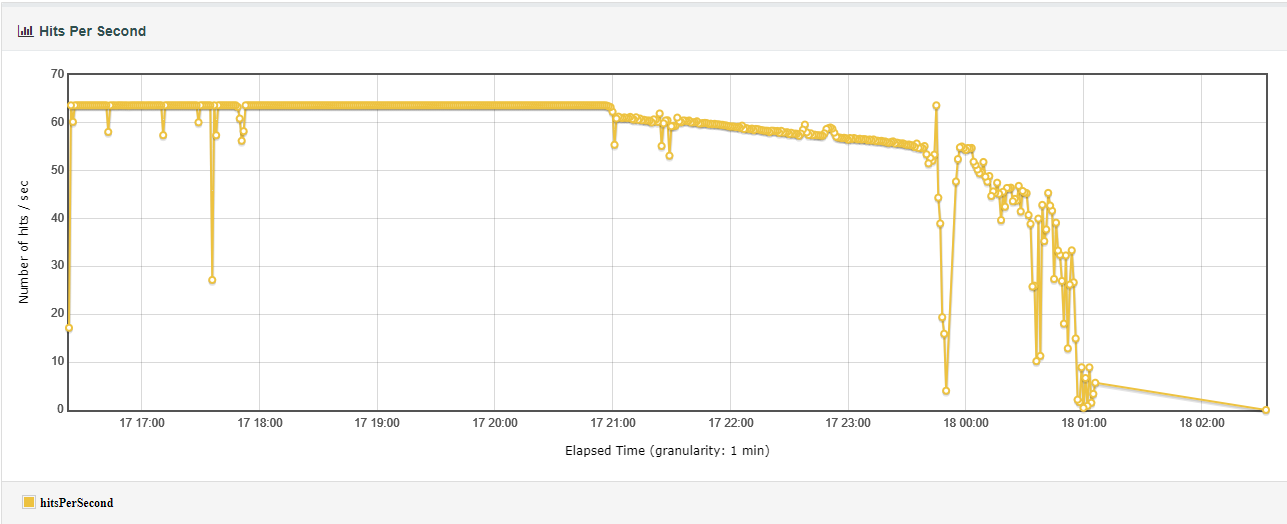


На графике выше видно, что:

* Средняя скорость передачи данных составляет около 241 KB/s, с пиковыми значениями до 1.13 MB/s.
* Средняя скорость получения данных составляет около 1.07 MB/s, с пиковыми значениями до 10.3 MB/s.
* Начиная с 22:30, наблюдается явная деградация производительности сетевой подсистемы.
* Наблюдаются кратковременные пики передачи и получения данных, что может указывать на интенсивные сетевые операции в эти моменты.
* В конце теста, после 02:30, пиковые значения скорости передачи и получения данных снижаются, что указывает на ухудшение сетевой производительности.

На основании данных графика, сетевая подсистема стабильно справляется с нагрузкой на протяжении большей части теста, однако начиная с 22:30 наблюдается явная деградация производительности.

**График - Hits per Second**



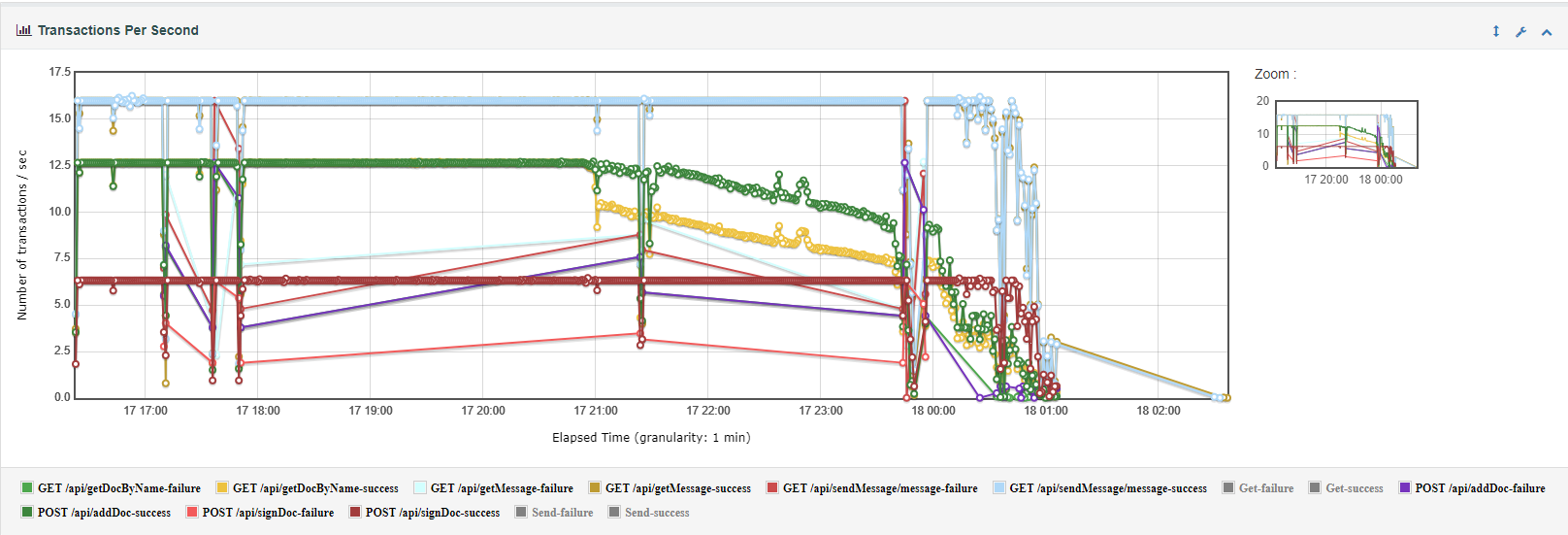
На графике выше видно, что:

* После начального роста количество запросов в секунду стабилизируется около 60 запросов в секунду.
* Начиная с 22:00, наблюдаются значительные колебания в количестве запросов, указывая на нестабильную работу системы.
* После 00:00 количество запросов начинает резко снижаться, продолжая колебаться вплоть до окончания теста.

На основании данных графика можно сделать вывод, что система способна стабильно обрабатывать нагрузку, поддерживая необходимое количество запросов в секунду на протяжении большей части теста. Однако начиная с 22:00 наблюдается нестабильная работа, которая усиливается после 00:00, что приводит к значительному снижению количества запросов в секунду.

Конец формы

**График - Transaction per second**

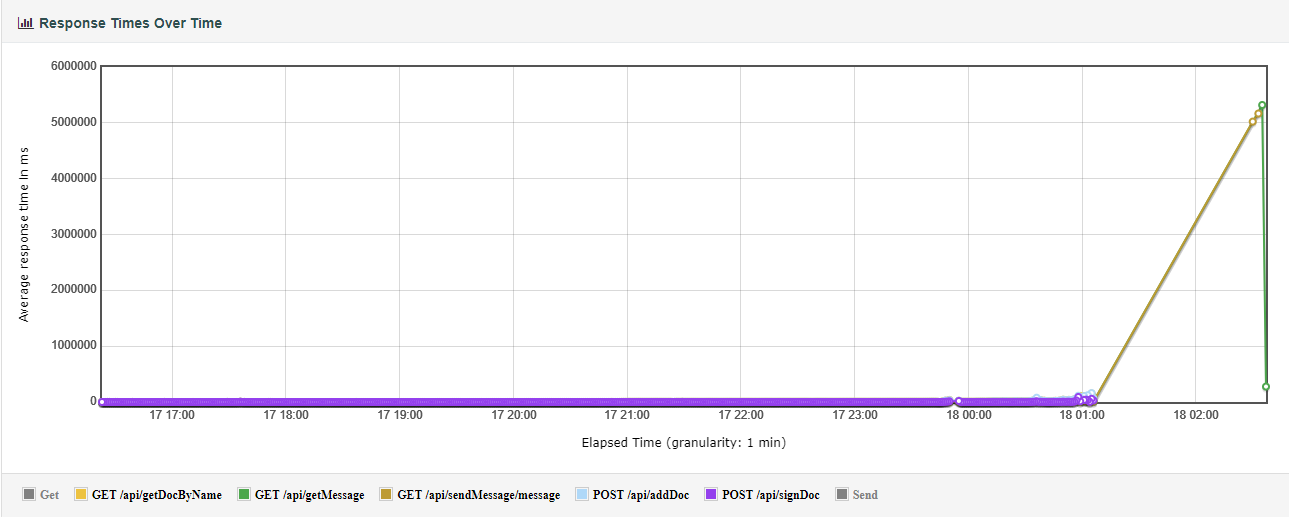


На графике выше видно, что:

* **GET /api/sendMessage-success (голубая линия):**
  + Начало теста: плавный рост до 5 транзакций в секунду.
  + Далее: стабилизация на уровне 15-17 транзакций в секунду.
* **GET /api/getDocByName-success (желтая линия):**
  + Начало теста: плавный рост до 7 транзакций в секунду.
  + Далее: стабилизация на уровне около 8 транзакций в секунду.
* **GET /api/getMessage-success (зеленая линия):**
  + Начало теста: плавный рост до 5 транзакций в секунду.
  + Далее: стабилизация на уровне около 10 транзакций в секунду.
* **POST /api/addDoc-success (светло-голубая линия):**
  + Начало теста: плавный рост до 10 транзакций в секунду.
  + Далее: стабилизация на уровне 15-17 транзакций в секунду.
* **POST /api/signDoc-success (фиолетовая линия):**
  + Начало теста: плавный рост до 5 транзакций в секунду.
  + Далее: стабилизация на уровне 5-8 транзакций в секунду.
* До 17:00 система показывает стабильную производительность по всем основным API-методам.
* После 17:00 наблюдаются значительные колебания и снижение количества транзакций по всем API-методам.

На основании данных графика можно сделать вывод, что система способна стабильно обрабатывать высокий уровень транзакций в секунду на протяжении большей части теста. Однако начиная с 17:00 наблюдается нестабильная работа, что приводит к значительному снижению количества транзакций в секунду.

**График - Времена отклика**



На графике выше видно, что:

* **GET /api/getDocByName (желтая линия):**
  + Начало теста: время отклика около 0 мс.
  + Далее: плавное увеличение до примерно 4000 мс.
  + В течение теста: постепенное увеличение времени отклика до 8000 мс.
  + В конце теста: значительное увеличение и колебания времени отклика, достигая почти 10000 мс.
* **GET /api/getMessage (зеленая линия):**
  + Начало теста: время отклика около 0 мс.
  + Далее: постепенное увеличение до примерно 6000 мс.
  + В конце теста: значительное увеличение и колебания времени отклика, достигая почти 6000000 мс.
* **GET /api/sendMessage/message (фиолетовая линия):**
  + Начало теста: время отклика около 0 мс.
  + Далее: постепенное увеличение до примерно 10000 мс.
  + В конце теста: значительное увеличение и колебания времени отклика, достигая почти 5000000 мс.
* **POST /api/addDoc (голубая линия):**
  + Начало теста: время отклика около 0 мс.
  + Далее: стабилизация до 4000 мс.
  + В течение теста: значительное увеличение времени отклика до 6000 мс.
  + В конце теста: значительное увеличение и колебания времени отклика.
* **POST /api/signDoc (зеленая линия):**
  + Время отклика остается стабильно низким на протяжении всего теста, около 0 мс.

Наблюдается значительное увеличение времени отклика для операций GET /api/getDocByName и POST /api/addDoc, что может указывать на увеличение нагрузки на систему или узкие места в производительности. Время отклика для GET /api/getMessage и POST /api/signDoc остаётся стабильным и низким на протяжении всего теста, что указывает на эффективную обработку этих запросов. График демонстрирует стабильное и предсказуемое время отклика для большинства операций, с явной деградацией производительности в конце теста.

**График – Kafka: Message in per second**



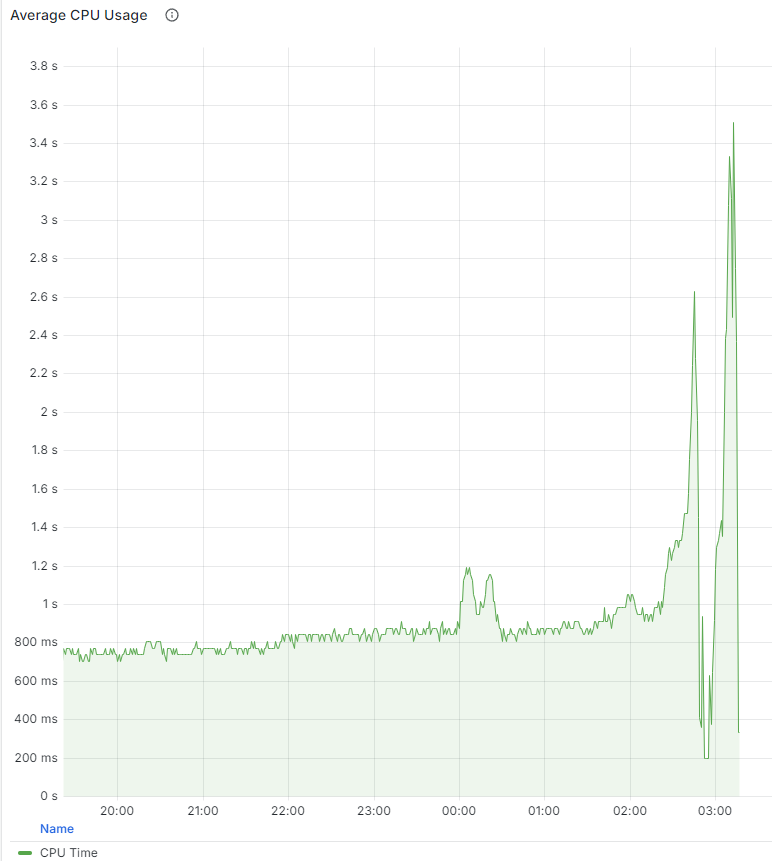
На графике выше видно, что:

* В начале теста количество сообщений в секунду для Kafka постепенно увеличивалось до уровня около 16 сообщений/секунду.
* После достижения уровня 16 сообщений/секунду, скорость поступления сообщений оставалась относительно стабильной.
* В течение теста средняя скорость поступления сообщений составляла около 15 сообщений/секунду.
* Начиная с 03:00, наблюдается значительное снижение количества сообщений в секунду, что может указывать на деградацию производительности системы.
* В конце теста количество сообщений в секунду снижается до 0, что указывает на остановку процесса.

Стабильный уровень поступления сообщений в течение большей части теста свидетельствует о том, что система справлялась с нагрузкой до определенного момента. Однако, начиная с 03:00, система начала испытывать трудности с обработкой сообщений, что привело к значительному снижению производительности.

**График –PostgresSQL**

* 1. **Average CPU Usage**

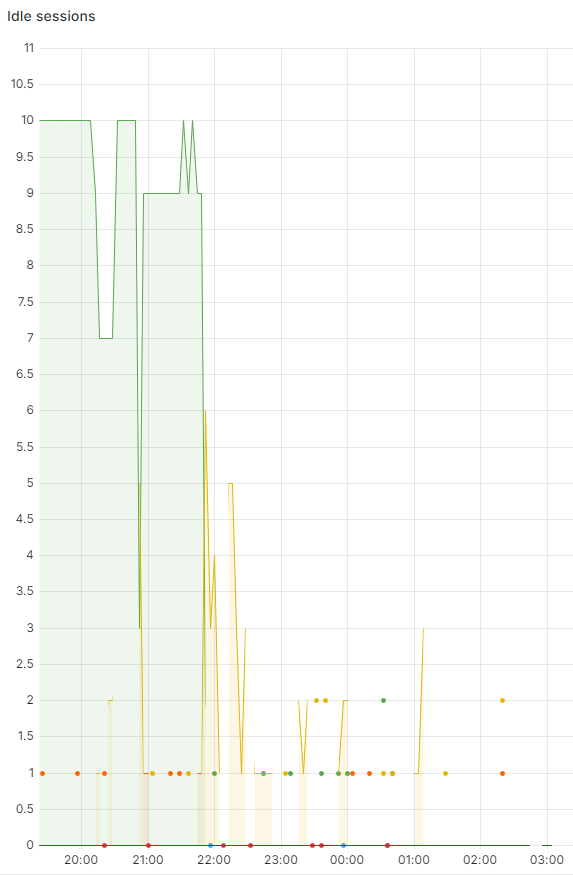


На графике выше видно, что:

* В начале теста среднее использование CPU для PostgreSQL колеблется в пределах 680-760 мс.
* В течение теста наблюдаются колебания использования CPU, но с общей тенденцией к увеличению.
* Около 23:00 начинается более резкий рост использования CPU, достигая пиковых значений до 1.2 секунд.
* После 01:00 использование CPU продолжает увеличиваться, достигая пиковых значений до 3.6 секунд.
* В конце теста, около 03:00, использование CPU значительно возрастает и наблюдаются значительные колебания.

На основании данных графика, можно сделать вывод, что основная проблема заключается в значительном росте использования CPU после 23:00 и особенно после 01:00. Эти колебания и увеличение использования CPU могут указывать на проблемы с нагрузкой на систему, неэффективные запросы или блокировки, которые приводят к деградации производительности системы.

* 1. **Idle session**

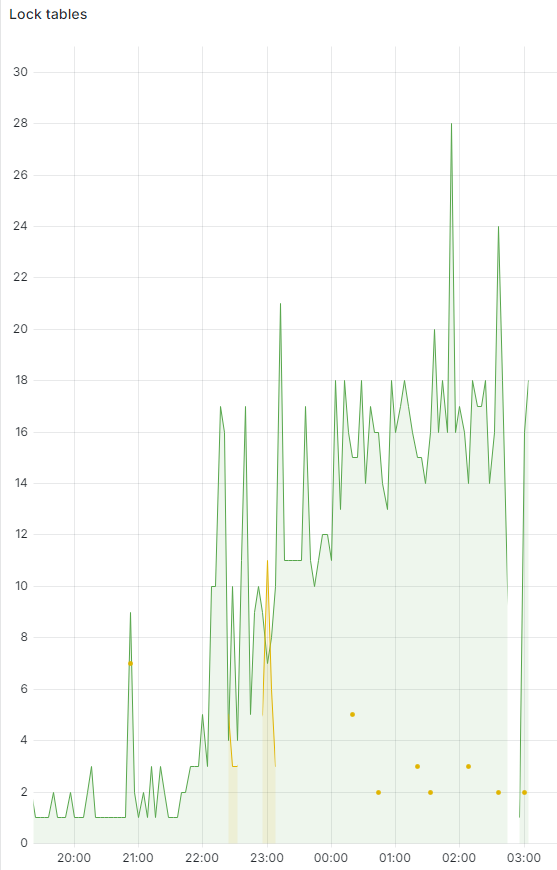


На графике выше видно, что:

* В начале теста количество неактивных сессий для PostgreSQL остается стабильным на уровне около 10 сессий.
* Начиная примерно с 20:45, количество неактивных сессий начинает колебаться и значительно уменьшается, достигая минимальных значений около 0 сессий.
* В то же время, с 20:45 наблюдается некоторое количество сессий, находящихся в состоянии "idle in transaction", что указывает на наличие долгосрочных транзакций, которые не завершаются вовремя.
* Периодические пики в неактивных сессиях и сессиях в состоянии "idle in transaction" совпадают с резким снижением количества транзакций в секунду, как видно на предыдущем графике.

На основании данных графика, можно сделать вывод, что основная проблема заключается в значительном уменьшении количества неактивных сессий после 20:45 и наличии сессий в состоянии "idle in transaction". Это указывает на возможные проблемы с производительностью, такие как блокировки, долгосрочные транзакции или проблемы с соединением, которые приводят к деградации производительности системы.

1. **Lock tables**



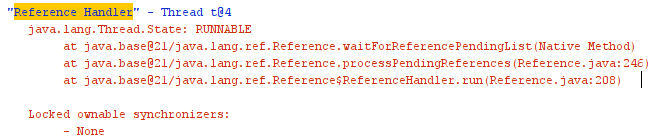
На графике выше видно, что:

* В начале теста количество заблокированных таблиц остается относительно стабильным на уровне 1-2 блокировок.
* До 23:20 наблюдаются минимальные колебания, и количество блокировок не превышает 2.
* Начиная с 23:20, количество блокировок начинает резко увеличиваться.
* К 23:30 количество заблокированных таблиц достигает пиковых значений до 23 блокировок.
* После 23:30 количество блокировок остается высоким и колеблется в пределах 15-23 блокировок, что указывает на значительные проблемы с блокировками.

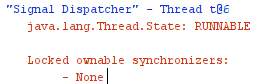
На основании данных графика, можно сделать вывод, что основная проблема заключается в резком увеличении количества заблокированных таблиц после 23:20. Это указывает на возможные проблемы с транзакциями, конкуренцию за ресурсы или неэффективное управление блокировками, что приводит к деградации производительности системы.

**Thread dump**

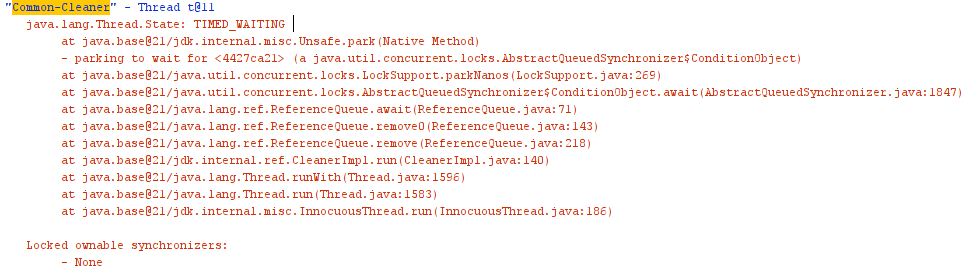
* 1. **RUNNABLE потоки**:
  + Поток "Reference Handler": Выполняет обработку ссылок.



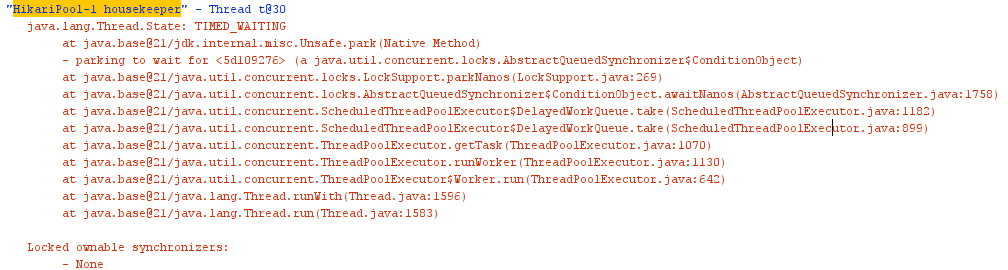
* + Поток "Signal Dispatcher": Обрабатывает системные сигналы.



* **Возможные проблемы**:
  + Потоки, которые постоянно находятся в состоянии RUNNABLE, указывают на высокую загрузку CPU.
  1. **TIMED\_WAITING потоки**:
  + Поток "Common-Cleaner": Ожидает задачи очистки.

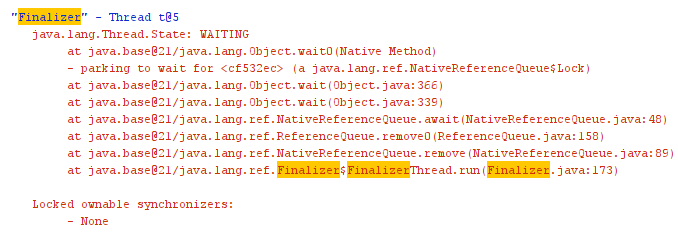


* + Поток "HikariPool-1 housekeeper": Ожидает следующего задания от планировщика задач.

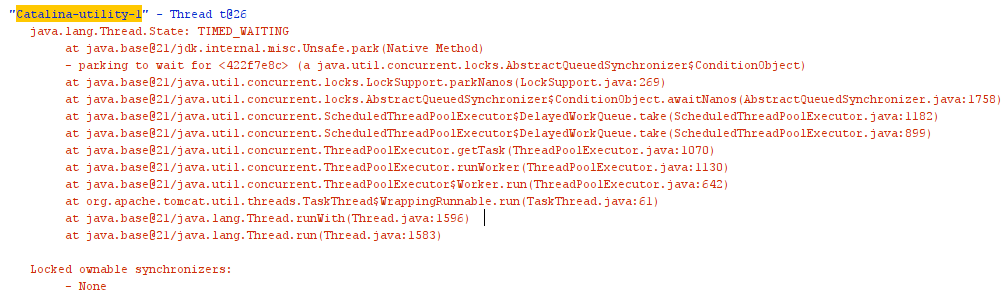


* **Возможные проблемы**:
  + Потоки в состоянии TIMED\_WAITING ожидают завершения временного периода, это может указывать на задержки в обработке задач.

1. **WAITING потоки**:
   * Поток "Finalizer": Ожидает завершения финализации объектов.



* + Поток "Catalina-utility-1": Ожидает новой задачи от планировщика.



* **Возможные проблемы**:
  + Потоки в состоянии WAITING могут ожидать завершения определенных операций, что может указывать на возможные узкие места в производительности.

1. **BLOCKED потоки**:
   * Потоки "http-nio-8080-exec-1" - "http-nio-8080-exec-9": Ожидают освобождения мьютекса для обработки HTTP-запросов.



* **Возможные проблемы**:
  + Высокая конкуренция за монитор, связанный с обработкой HTTP-запросов, что приводит к значительным задержкам.
  + Продолжительные блокировки могут замедлять выполнение программы.

**Heap dump**



1. **java.lang.Object**:
   * **Count**: +99,122
   * **Size**: +1,585,952 B

Это может указывать на создание большого количества объектов в приложении, что может быть связано с интенсивной деятельностью по созданию и использованию объектов.

1. **java.util.concurrent.ConcurrentHashMap$Node**:
   * **Count**: +26,782
   * **Size**: +857,024 B

Это может указывать на использование ConcurrentHashMap для управления коллекциями данных, что может быть необходимо для обеспечения конкурентного доступа к данным.

1. **byte[]**:
   * **Count**: +22,716
   * **Size**: +9,227,584 B

Наибольшее увеличение по объему памяти. Это указывает на интенсивное использование байтовых массивов, что связано с обработкой больших объемов данных или буферизацией ввода/вывода.

1. **java.lang.String**:
   * **Count**: +19,840
   * **Size**: +476,160 B

Активное создание строк в приложении, связанное с обработкой текстовых данных.

1. **org.aspectj.weaver.patterns.ExposedState**:
   * **Count**: +11,597
   * **Size**: +371,104 B

Использование библиотеки AspectJ, вероятно, связано с аспектно-ориентированным программированием (AOP).

1. **org.aspectj.weaver.reflect.ShadowMatchImpl**:
   * **Count**: +11,597
   * **Size**: +556,656 B

Также связано с AspectJ, что может указывать на интенсивное использование аспектов для обработки.

1. **org.apache.tomcat.util.buf.ByteChunk**:
   * **Count**: +5,936
   * **Size**: +284,928 B

Связано с обработкой данных в Apache Tomcat, что может быть частью веб-сервера.

1. **org.apache.tomcat.util.buf.CharChunk**:
   * **Count**: +5,407
   * **Size**: +259,536 B

Обработка символов в Tomcat, вероятно, связана с парсингом и обработкой HTTP-запросов и ответов.

1. **org.apache.tomcat.util.buf.MessageBytes**:
   * **Count**: +5,277
   * **Size**: +253,296 B

Используется для обработки сообщений в Tomcat, указывает на активную работу веб-сервера.

1. **java.lang.Object[]**:
   * **Count**: +4,647
   * **Size**: +842,592 B

Массивы объектов, используемые для хранения и управления коллекциями данных.

### Возможные проблемы:

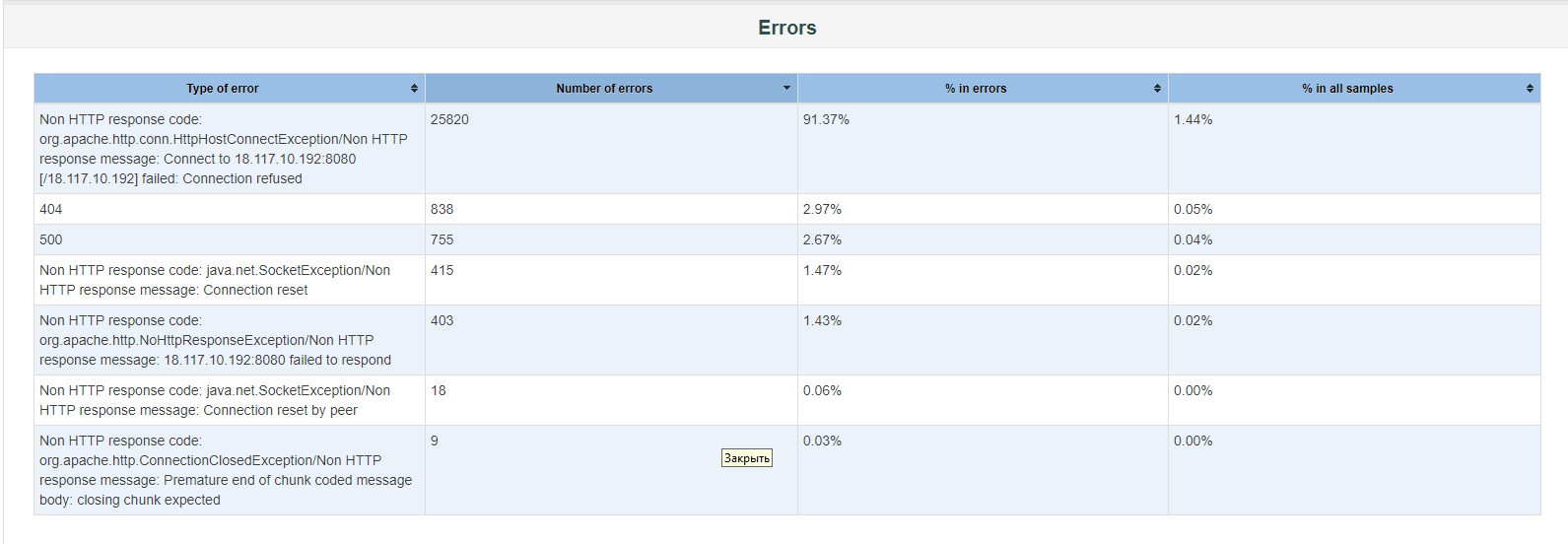
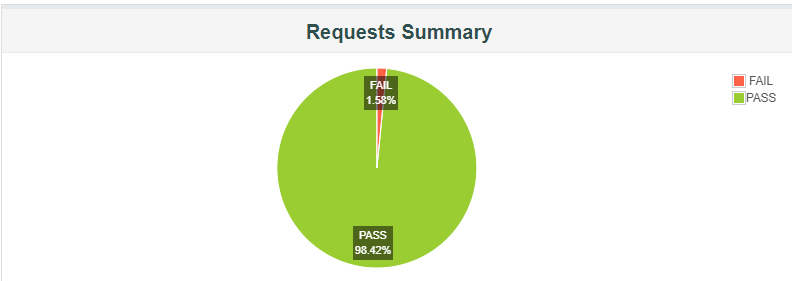
* **Интенсивное использование байтовых массивов byte[] и строк java.lang.String**:
  + Может привести к высокому потреблению памяти и увеличению сборок мусора.
* **Увеличение использования ConcurrentHashMap$Node**:
  + Указывает на рост сложности управления данными и потенциальные проблемы с производительностью при высоком уровне конкурентного доступа.

### Возможные решения:

* **Оптимизация использования байтовых массивов и строк**:
  + Минимизировать создание временных объектов там, где это возможно.
* **Оптимизация аспектно-ориентированного программирования**:
  + Нужен анализ использование библиотек AspectJ и рассмотреть возможность оптимизации аспектов для уменьшения накладных расходов.

### Заключение: основные источники увеличения потребления памяти связаны с интенсивным использованием байтовых массивов, строк и структур данных для управления коллекциями. Рекомендуется рассмотреть возможность оптимизации этих областей для улучшения производительности и снижения потребления памяти.

**График - Errors**



На графике и таблицах выше видно, что: • Общий процент успешных запросов составляет 98.42%, а процент неуспешных запросов составляет 1.58%. • Из 179,289,4 запросов, 28,528 запросов завершились с ошибкой. • Основные типы ошибок включают:

* Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to 18.117.10.192:8080 [/18.117.10.192] failed: Connection refused (91.37% ошибок)
* 404 (2.97% ошибок)
* 500 (2.67% ошибок)
* Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset (1.47% ошибок)
* Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 18.117.10.192:8080 failed to respond (1.43% ошибок)
* Другие незначительные ошибки включают: Connection reset by peer и Premature end of chunk coded message body: closing chunk expected. • Наиболее частыми ошибками являются ошибки подключения к серверу, такие как Connection refused и Connection reset, что указывает на проблемы с доступностью сервера. • Запрос GET /api/getDocByName имеет высокий процент ошибок, в основном связанные с ошибками подключения (5160 из 5675 ошибок). • Запрос GET /api/getMessage также имеет значительное количество ошибок подключения (6452 из 7422 ошибок). • Запрос GET /api/sendMessage/message и POST /api/addDoc также показывают высокий уровень ошибок подключения. • Ошибки 500 указывают на внутренние серверные ошибки, связанные с запросами GET /api/getDocByName и POST /api/addDoc, что требует дополнительного анализа серверного кода и баз данных для выявления причин этих ошибок.

На основании данных графика и таблиц можно сделать вывод, что основная проблема связана с доступностью и стабильностью сервера, а также с внутренними ошибками сервера, которые необходимо решить для улучшения производительности и стабильности системы.