МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ПРИЛОЖЕНИЯ НЕФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ ВЫСОКОНАГРУЖЕННЫХ РАСПРЕДЕЛЕННЫХ СИСТЕМ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Спасибо Ольги Олеговны

Научный руководитель
ст. преподаватель, _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Подавляющее большинство широко распространенных и часто используемых компьютерных решений и приложений представляет собой высоконагруженные распределенные системы, асинхронно работающие с огромными объемами информации. В связи с этим существует необходимость пристального контроля за бесперебойной работой систем, который достигается, помимо построения соответствующей инфраструктуры, с помощью эффективно организованного процесса тестирования. На сегодняшний день существует множество сложных компьютеризированных систем, цена ошибки которых очень высока: системы мониторинга частоты сердечного ритма; системы автоматического управления полётами и поездками; системы, оказывающие влияние на экономику и производство, и другие.

Актуальность проблемы поиска уязвимостей и анализа надежности и защищенности систем очень высока. Для достижения высокого уровня тестирования необходимо использовать не менее сложные инструменты тестирования.

Целью данной работы является создание приложения для тестирования систем мониторинга биржевых площадок. В ходе работы необходимо проанализировать и сравнить некоторые решения, предназначенные для создания сценариев нефункционального тестирования для оценки отказоустойчивости и стрессоустойчивости системы, скорости ее восстановления после сбоев, анализ процессов репликации и резервного копирования, предельного числа активных пользователей и так далее.

Полученные результаты анализа используются для:

- разработки приложения для тестирования систем мониторинга биржевых площадок;
- анализа полученных в ходе исследования метрик и статистик;
- автоматизированного представления результатов тестирования в доступной визуальной форме в виде графиков и диаграмм.

Объектом исследования является биржевая система, работающая с протоколами обмена финансовой (биржевой) информацией, такими как международный стандарт FIX и проприетарный бинарный протокол NATIVE.

Предметом исследования является возможность создания приложения, оценивающего отказоустойчивость и надежность системы мониторинга биржевой платформы.

Программные средства, используемые в ходе работы:IntelliJ IDEA 2019.1.3 (Community Edition, Build #IC-191.7479.19, built on May 28, 2019) JRE: 1.8.0_202-release-1483-b58 x86_64, JVM: OpenJDK 64-Bit Server VM, Charles Web Debugging Proxy (version 4.5.6), Gatling Tool (version 3.3.1 (November 7th, 2019) License: Apache License 2.0), Apache Maven 3.6.1

1 Теоретические аспекты

1.1 Тестирование ПО

Существует множество общепринятых определений тестирования программного обеспечения.

В наиболее широком смысле, тестирование программного обеспечения — это одна из техник контроля качества, включающая в себя следующие мероприятия:

- планирование работ;
- проектирование тестов;
- выполнение тестирования;
- анализ полученных результатов.

Очень важными понятиями в тестировании программного обеспечения являются верификация и валидация [1].

Верификация – это процесс оценки системы или её компонентов с целью определения того, удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа, то есть выполняются ли цели, сроки, задачи по разработке проекта, определенные в начале текущей фазы.

Валидация – это определение соответствия разрабатываемого программного обеспечения ожиданиям и потребностям пользователя, требованиям к системе.

Множество определений международных стандартов ISO сводится к следующему: тестирование программного обеспечения – это процесс исследования (статическое тестирование) и/или испытания (динамическое тестирование) программного обеспечения, целью которых является проверка соответствия между фактическим поведением программного средства и его ожидаемым поведением после выполнения конечного набора тестов, выбранного определенным (в зависимости от методологии) образом [2].

Можно выделить следующие цели тестирования программного обеспечения [3]:

- повышение вероятности того, что приложение, находящееся под тестированием, будет работать правильно при любых обстоятельствах;
- повышение вероятности того, что приложение, находящееся под тестированием, будет соответствовать всем описанным требованиям;

— предоставление актуальной информации о состоянии продукта в определенный момент времени.

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей и объекта/объектов тестирования, можно разделить на следующие три группы:

- функциональные;
- нефункциональные;
- связанные с изменениями.

Функциональное тестирование – это процесс проверки того, в какой степени реальные функции продукта соответствуют функциональными требованиями, описанным в спецификации (документации) этого продукта. Виды функционального тестирования основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (компонентном, интеграционном, системном, приемочном). Как правило, помимо функциональных спецификаций, эти функции описываются в требованиях или в виде “случаев использования системы”м [4].

Тестирование функциональности может проводиться в двух областях:

- требования;
- бизнес-процессы.

Тестирование в области требований использует спецификацию функциональных требований к системе как основу для дизайна тест-кейсов. Обычно составляются списки того, что должно тестироваться, а что нет, приоритезируются требования на основе рисков (если это не сделано в документе с требованиями), а на основе этого и тестовые сценарии. Это позволяет фокусироваться и не упустить при тестировании наиболее важный функционал.

Тестирование в области бизнес-процессов базируется на знаниях этих бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этом случае тестовые сценарии, как правило, основываются на случаях использования системы. Зачастую тестирование на основе бизнес-процессов находит те баги, что обнаружились бы конечными пользователями в первую очередь, если бы качественное тестирование не было проведено своевременно.

Нефункциональное тестирование – это процесс, относящийся к проверке свойств, которые не относятся к функциям разработанного продукта.

Данные свойства определяются нефункциональными требованиями, которые характеризуют разработанный продукт с таких сторон, как [1]:

1. надежность (способность продукта адекватно реагировать на непредвиденное использование);
2. производительность (способность продукта сохранять работоспособность под разными уровнями нагрузки);
3. удобство (анализ удобства использования приложения с позиции конечного пользователя);
4. масштабируемость (способность продукта функционировать при разных масштабах использования; например, при частном использовании и при использовании для нужд корпорации);
5. безопасность (степень защиты данных пользователей); портируемость (способность продукта функционировать с использованием различных платформ) и много других качеств.

Подытожив описание двух видов тестирования, можно заключить, что функциональное тестирование отвечает на вопрос “как работает система?”, а нефункциональное тестирование отвечает на вопрос “как хорошо система выполняет свои функции?” [5]

1.2 Фреймворк нагрузочного тестирования Gatling

С развитием отрасли нагрузочного тестирования и ростом необходимости его применения разрабатывались различные новые тестовые инструменты - фреймворки. Фреймворки нагрузочного тестирования различаются в архитектурном плане, качеством эмулируемых сетевых соединений, пороговым числом эмулируемых активных пользователей сети, количеством поддерживаемых протоколов, интерфейсом и так далее. В зависимости от особенностей предметной области и специфики решаемой проблемы стоит выбирать соответствующий инструмент нагрузочного тестирования.

Фреймворк Gatling — мощный инструмент нагрузочного тестирования, имеющий высокую производительность и широту поддержки сетевых протоколов “из коробки”.

Gatling отличается отличной поддержкой HTTP-протокола, поэтому данный фреймворк — оптимальное решение для тестирования HTTP-сервера. Так как ядро Gatling не имеет привязки к определенному протоколу, в будущем фреймворк Gatling возможно будет иметь такую же надежную поддержку и

других сетевых протоколов. В настоящее время, фреймворк Gatling имеет поддержку JMS (Java Message Service — стандарт промежуточного ПО для рассылки сообщений, позволяющий приложениям, выполненным на платформе Java EE, создавать, посылать, получать и читать сообщения).

Разработчики фреймворка Gatling стремились создать ресурсоемкий эффективный инструмент нагрузочного тестирования, сценарии для которого легко читались другими разработчиками и отражали предметную область и специфику проекта. Gatling имеет поддержку DSL(domain-specific language) для разработки тестовых сценариев, расширение над языком Scala [6].

Gatling - это фреймворк с асинхронной архитектурой. Так как Gatling работает с протоколами (например, HTTP) Gatling поддерживает асинхронную работу до возникновения блокирующих друг друга сообщений. Поэтому действия пользователей системы реализованы не через разделение потоков на рабочей машине, а через отправку асинхронных сообщений, что позволяет сэкономить затраты на поддержку аппаратных средств. Таким образом, с использованием фреймворка Gatling не является проблемой симуляция одновременной работы тысяч пользователей.

К недостаткам данного фреймворка можно отнести отсутствие подробной документации и отсутствие поддержки функциональностей фреймворка предыдущих версий. Большинство пользователей отмечают, что новые версии Gatling зачастую несовместимы с проектом, написанным с использованием более поздних версий. Gatling ориентирован на достаточно опытных разработчиков ПО для нагрузочного тестирования, так как не содержит сам по себе графического интерфейса для запуска и отладки приложений. Сценарии с использованием фреймворка Gatling могут быть написаны только на языке программирования Scala, который не является одним из самых распространенных, что также увеличивает входные требования для разработчиков [7].

Существует 8 способов выпонения сценария:

- nothingFor (duration): пауза при выполнении сценариев.
- atOnceUsers (nbUsers): внедряет заданное количество пользователей одновременно.
- rampUsers (nbUsers) in (duration): вводит заданное количество пользователей с линейным изменением в течение заданной продолжительности.
- constantUsersPerSec (speed) in (duration): вводит пользователей с посто-

янной скоростью, определенной в “пользователях в секунду”, в течение заданной продолжительности. Пользователи будут вводиться через равные промежутки времени.

— constantUsersPerSec (speed) in (duration) randomized: вводит пользователей с постоянной скоростью, определенной в пользователях в секунду, в течение заданной продолжительности. Пользователи будут вводиться через случайные интервалы времени.
— rampUsersPerSec (rate1) - (rate2) in (duration): вводит пользователей от начальной скорости до целевой скорости, определенной в “пользователях в секунду”, в течение заданной продолжительности. Пользователи будут вводиться через равные промежутки времени.
— rampUsersPerSec (rate1) - (rate2) in (duration) randomized: вводит пользователей от начальной скорости до целевой скорости, определенной пользователями в секунду, в течение заданной продолжительности. Пользователи будут вводиться через случайные интервалы времени.
— heavyisideUsers (nbUsers) in (duration) : вводит заданное число пользователей после плавного приближения пошаговой функции критических значений, определенной на заданной продолжительности.

2 Практика

2.1 Описание тестируемой системы мониторинга

В ходе работы требуется создать симуляцию поведения несколько тысяч пользователей для системы мониторинга трейдинговых операций в биржевой системе. Приложение развернуто на удаленной машине, доступ на которую отсутствует в связи с политикой безопасности. Приложение доступно только для устройств, подключенных к локальной защищенной LAN-сети. На рисунке 1 приведен снимок экрана окна рассматриваемой системы мониторинга.

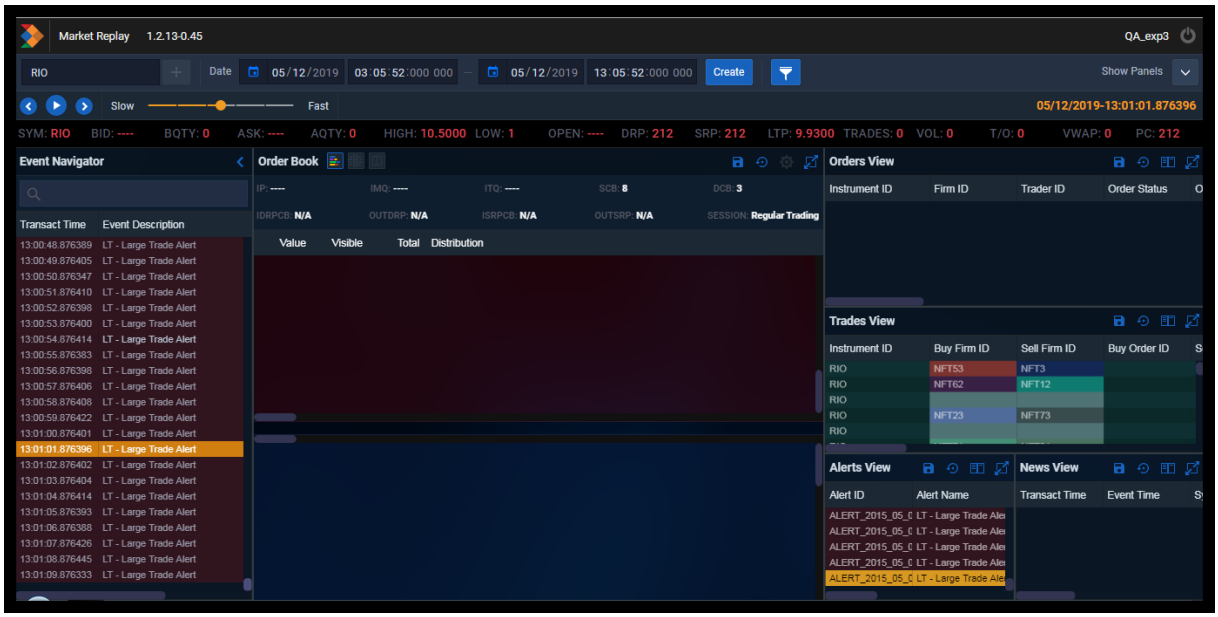

Рисунок 1 – окно системы мониторинга Market Replay после авторизации

2.2 Разработка программного обеспечения

В ходе данной работы были разработаны нагрузочные тесты на языке Scala с применением фреймворка Gatling для высоконагруженного распределенного приложения.

Для реализации поставленных в работе целей будут решены следующие задачи:

- создание приложения с использованием фреймворка Gatling
- реализация нагрузки свыше 1000 пользователей
- реализация механизма аутентификации посредством передачи токена
- исследование различных методов роста числа активных пользователей — линейный, квадратичный рост, резкий скачок (увеличение в десятки и сотни раз) подключений к системе, иными словами - rampUp и так далее

● построение наглядных отчетов о проделанной работе

● определение критических значений, значительно влияющих на работу системы мониторига

В ходе данной работы был создан Maven-проект в интегрированной среде разработки IntelliJ IDEA, разработанной компанией JetBrains. Maven — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющимся подмножеством XML. В файле pom.xml задаются необходимые зависимости, подключаются библиотеки и репозитории, необходимые для успешной сборки проекта. Maven предоставляет возможность автоматически импортировать необходимые библиотеки на локальную машину в директорию .m2, к которой фреймворк непосредственно обращается при компиляции и сборки проекта. Так как локальная машина, с которой выполянется разработка проекта находится в закрытой защещенной сети, доступ в ней на большинство веб-сайтов закрыт, также закрыт доступ к удаленному репозиторию https://mvnrepository.com/artifact/org.apache. Все пакеты и библиотеки, используемые в проекте, были установлены вручную.

На рисунке 2 изображено стартовое окно Gatling Recoder.

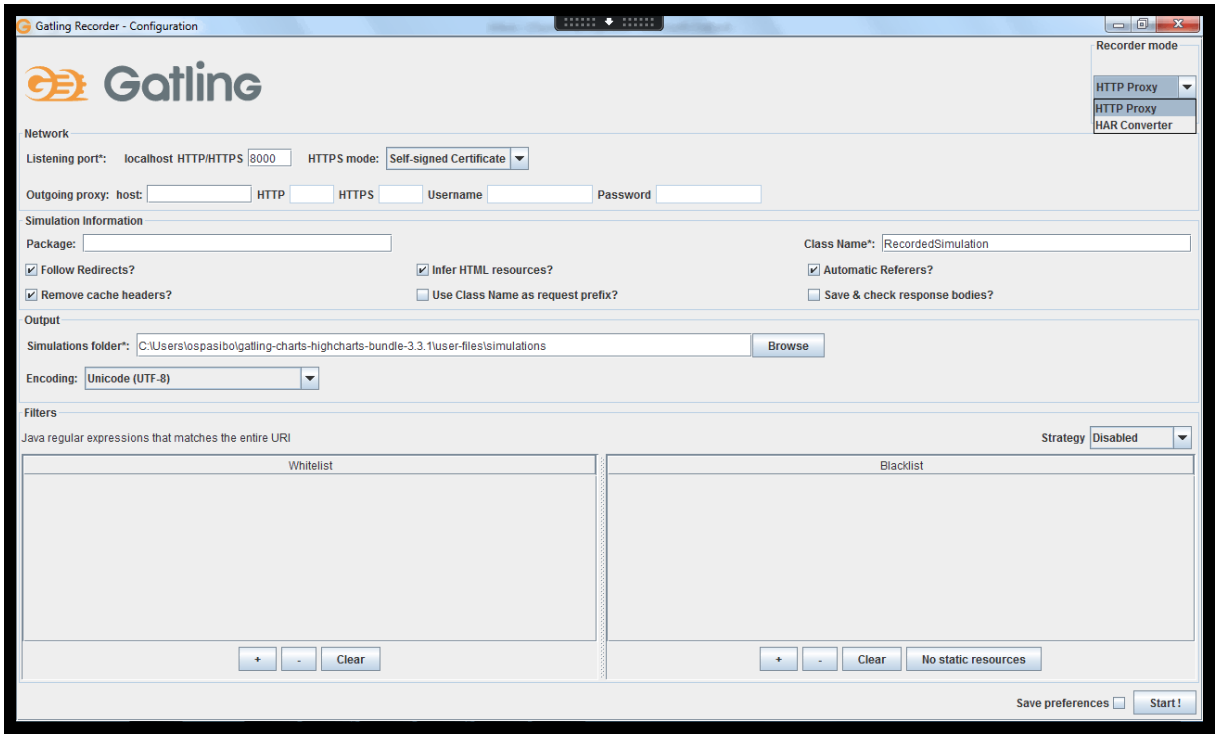

Рисунок 2 – окно запуска фреймворка Gatling

Существуют два вида автоматизированной записи тестовых сценариев — в режиме прокси-сервера [8] и при передаче HAR- файла. Для записи в режиме

прокси-сервера необходимо, чтобы приложение было запущено локально на той же машине на которой запущен Gatling Recoder. Действия пользователя автоматически отражаются в виде посылаемых серверу HTTP-запросов в окне Gatling Recoder. Записать HAR-файл для передачи его Gatling Recoder можно непосредственно в браузере (Например, в Google Chrome это можно сделать в разделе Developer Tools -> Network).

Для решения поставленной задачи невозможно использовать автоматизированную запись действий пользователя системы, так как разрабатываемый проект и приложение по мониторингу биржевой системы расположены на разных машинах, а сеть через которую есть доступ к приложению является защищенной и закрытой, что не позволяет отразить в полной мере запросы клиентов и ответы сервера.

На рисунке 3 отражена структура разработанного проекта.

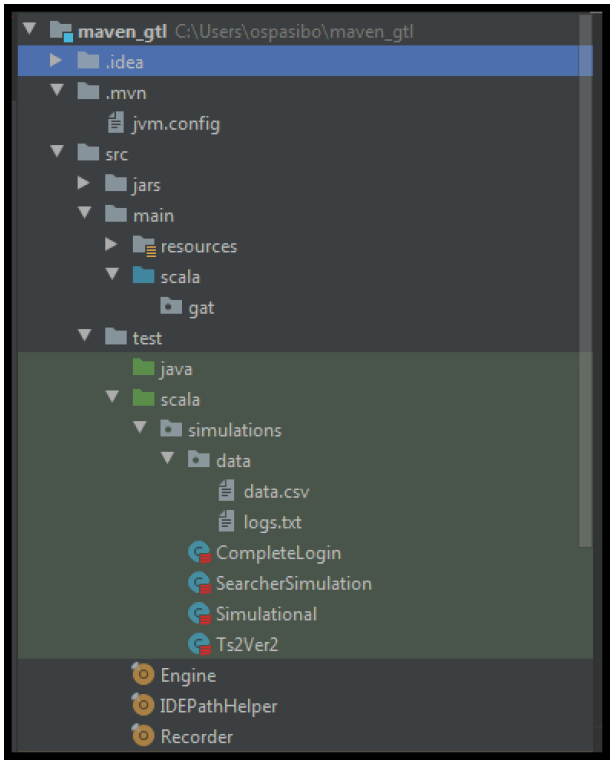

Рисунок 3 – получение и отправка HTTP-запросов от хоста https://10.61.9.104:22650

В директории $project/test/scala/simulations находятся основные тестовые сценарии.

В основном классе классе $project/test/scala/simulations/Ts2Ver2.scala инициализированы :

● переменная url — переменная строкового типа, которая содержит адрес хоста;

● переменные c, hr24, min, sec были созданы для записи в лог-файл данных каждого пользователя и время его подключения к хосту;

● переменная httpProtocol задает непосредственно параметры подключения к хосту https://10.61.9.104:22650 и защищенному веб-сокету;

wss://10.61.9.104:22650/websocket

● переменные headers содержат заголовки к HTTP-запросам, которые меняются в зависимости от характера запроса. В headers перечислены только поля заголовков, которые следует изменить. Неперечисленные поля сохраняют значения присвоенные в предыдущем заголовке;

● переменная csvFeeder содержит путь до csv-файла, из которого приложение собирает данные для парсинга и параллельной отправки данных на хост;

● переменная scn содержит непосредственно вызов HTTP-запросов, изменения заголовков, подключения к веб-сокету и хосту, вызов feeder-a для параллельного одновременного подключения пользователей с разными временными токенами;

● метод setUP запускает сценарий на выполнение, задавая количество пользователей и характер роста нагрузки на приложение.

2.3 Результаты работы

В результате работы были построены графики и отчеты автоматизированными средствами фреймворка нагрузочного тестирования Gatling.

В ходе работы для разных режимов генерации нагрузки были собраны следующие статистики:

● успешность/неуспешность выполнения запроса

● текст сообщений об ошибках

● количество успешных/неуспешных запросов

● количество запросов, выполняющихся за минимальное время обработки запросов

● количество запросов, выполняющихся за наибольшее время обработки запросов

● количество запросов, на выполнение которых требуется среднее количество времени и другие;

Собранные статистики позволили оценить время обработки запросов системой мониторинга биржевой платформы, оценить производительность си-

стемы при разных типах нагрузки, а так же определить среднее число запросов в секунду.

При выполнении разработанного приложения встречается ошибка сервера 401 - Unauthorized, так как логины и пароли, используемые приложением, также используется тестировщиками системы, разработчиками и менеджерами проекта. Если в момент запуска приложения для нагрузочного тестирования, пользователь уже авторизован в системе, то отправляемые через приложение данные учетной записи такого пользователя не смогут быть корректно обработаны сервером. На рисунке **??** видно, что на момент запуска приложения 4 пользователя системы уже были авторизованы.

В ходе работы были определены параметры, оказавшимися критическими для работы системы — 3000 пользователей в режиме rampUsers (3000) in (1200).

На рисунках 4, 5, 6 отражены графики и метрики для вызова сценария в режиме rampUp.

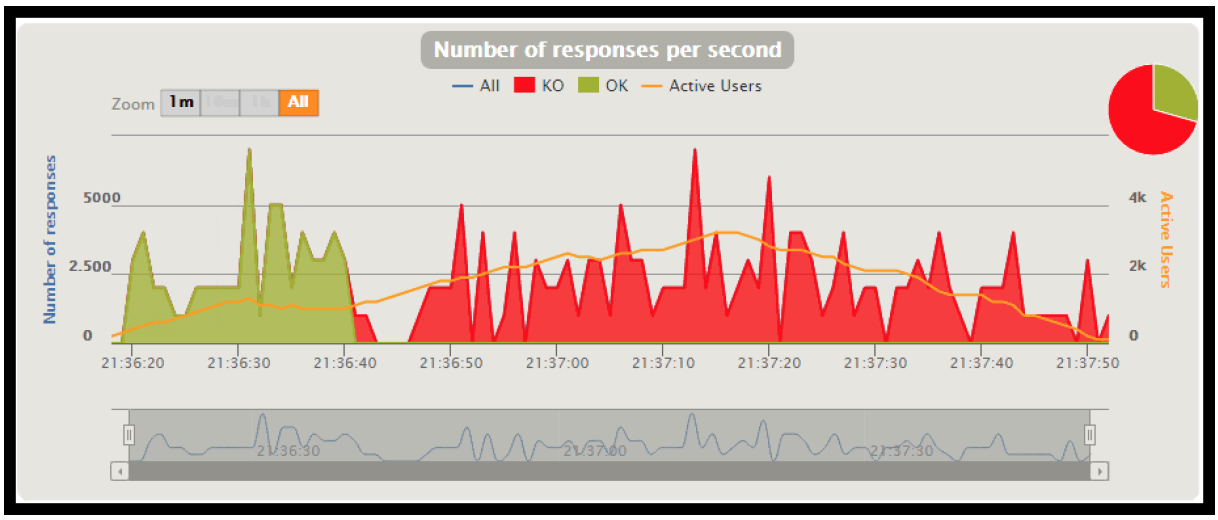

Рисунок 4 – отчет, демонстрирующий число получаемых от сервера ответов в секунду

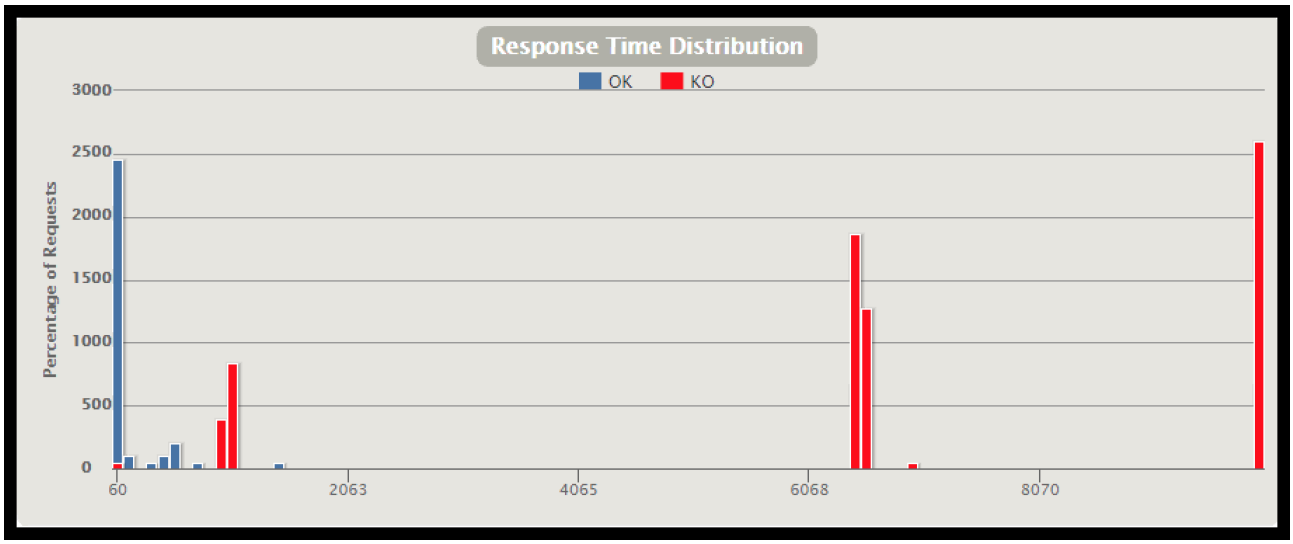

Рисунок 5 – отчет, демонстрирующий распределение времени ответа

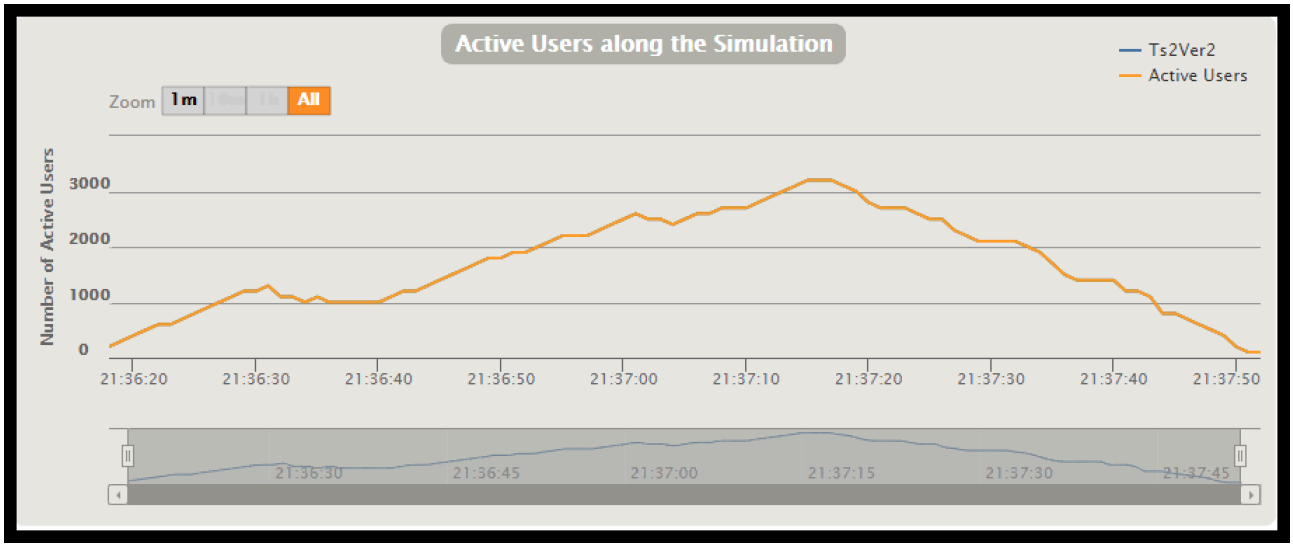

Рисунок 6 – отчет, демонстрирующий число активных пользователей в системе

ЗАКЛЮЧЕНИЕ

В ходе дипломной работы были решены следующие задачи:

- рассмотрение различных фреймворков нагрузочного тестирования;
- разработка приложения нагрузочного тестирования для высоконагруженной распределенной системы мониторинга;
- анализ критических значений параметров работы сиситемы;
- построение наглядных графиков и отчетов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Криспин, Л.* Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд / Л. Криспин. — Вильямс, 2016. — С. 174– 176.

2 *Куликов, С. С.* Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — 2017. — С. 312.

3 *Матвеевский, В. Р.* Надежность технических систем / В. Р. Матвеевский. — Московский государственный институт электроники и математики, 2002. — С. 113.

4 *Савин, Р.* Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах / Р. Савин. — Дело, 2007. — С. 124 – 136.

5 Официальный сайт ISTQB - International Software Testing Qualifications Board [Электронный ресурс]. — URL: https://www.istqb.org (Дата обращения 22.04.2020). Загл. с экр. Яз. англ.

6 *Tolledo, R.* Gatling: Take your performance tests to the next level [Электронный ресурс] / R. Tolledo. — Pp. 12–14. — URL: https://www.thoughtworks.com/gatling-take-your-performance-tests-next-level (Дата обращения 20.04.2020). Загл. с экр. Яз. англ.

7 Официальный сайт фреймворка Gatling [Электронный ресурс]. — URL: https://gatling.io (Дата обращения 22.04.2020). Загл. с экр. Яз. англ.

8 Официальный сайт Charles Web Debugging Proxy [Электронный ресурс]. — URL: https://www.charlesproxy.com (Дата обращения 19.04.2020). Загл. с экр. Яз. англ.