ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.415

ВЕБ-ОРИЕНТИРОВАННАЯ СИСТЕМА МОДЕЛИРОВАНИЯ НАГРУЗКИ ДЛЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ[1]

Шумилов Дмитрий Сергеевич,
студент 4 курса направления подготовки 09.03.03 «Прикладная информатика» ГАОУ ВО ЛО «ЛГУ им. А.С. Пушкина», г. Санкт-Петербург, Россия
E-mail: dim.shumiloff@gmail.com

Черников Сергей Юрьевич,
студент 4 курса направления подготовки 09.03.03 «Прикладная информатика» ГАОУ ВО ЛО «ЛГУ им. А.С. Пушкина», г. Санкт-Петербург, Россия
E-mail: 5wh19s13m8h@gmail.com

Аннотация

Статья посвящена проектированию и реализации веб-ориентированной системы моделирования нагрузки для тестирования программного обеспечения. Актуальность работы обусловлена необходимостью сократить трудоёмкость подготовки нагрузочных экспериментов и повысить воспроизводимость оценки поведения программной системы при изменяющихся профилях пользовательской активности. Рассматриваемая система объединяет дискретно-событийное ядро симуляции, веб-интерфейс настройки, модуль многоформатного логирования, средства статистического сравнения прогонов и оптимизатор конфигураций. Показано, что интеграция этих компонентов позволяет выполнять полный цикл работы с нагрузочным экспериментом: от задания архитектуры стенда и профиля нагрузки до получения графиков, метрик и сравнительных выводов по результатам нескольких запусков.

Ключевые слова: нагрузочное тестирование, имитационное моделирование, дискретно-событийная симуляция, SimPy, Flask, Chart.js, Optuna, логирование, анализ прогонов, веб-приложение.

WEB-ORIENTED LOAD SIMULATION SYSTEM FOR SOFTWARE TESTING

Shumilov Dmitry Sergeevich,
4th year student of the 09.03.03 "Applied Informatics" degree program, Pushkin Leningrad State University, Saint-Petersburg, Russia
E-mail: dim.shumiloff@gmail.com

[1] научный руководитель: **Королёв Владимир Владимирович**, Старший преподаватель, ГАОУ ВО ЛО «ЛГУ им. А.С. Пушкина», г. Санкт-Петербург, Россия
E-mail: vvkorolyov@yandex.ru
academic supervisor: **Korolev Vladimir Vladimirovich**, Senior lecturer, Pushkin Leningrad State University, Saint-Petersburg, Russia
E-mail: vvkorolyov@yandex.ru

Chernikov Sergey Yurievich,
4th year student of the 09.03.03 "Applied Informatics" degree program, Pushkin Leningrad State University, Saint-Petersburg, Russia
E-mail: 5wh19s13m8h@gmail.com

ABSTRACT

The article presents the design and implementation of a web-oriented load simulation system for software testing. The relevance of the work is determined by the need to reduce the effort required to prepare load experiments and to improve the reproducibility of performance assessment under changing user activity profiles. The system combines a discrete-event simulation core, a web configuration interface, multi-format logging, statistical comparison of simulation runs and configuration optimization. The paper shows that the integration of these components supports the complete workflow of a load experiment: from configuring the simulated architecture and workload profile to obtaining charts, metrics and comparative conclusions based on several runs.

Keywords: load testing, simulation modeling, discrete-event simulation, SimPy, Flask, Chart.js, Optuna, logging, run analysis, web application.

Введение

Современные программные системы обслуживают большое число пользователей, взаимодействуют с внешними сервисами и должны сохранять приемлемое время отклика при неравномерной нагрузке. Поэтому нагрузочное тестирование становится не только технической процедурой перед выпуском продукта, но и инструментом управления качеством разработки [1]. В классическом подходе специалист формирует сценарии, поднимает тестовый стенд, запускает генератор нагрузки и анализирует полученные отчёты. Такой процесс даёт точные сведения о поведении уже развёрнутой системы, но требует подготовленной инфраструктуры и заметных трудозатрат.

Для ранней оценки архитектурных решений полезно применять имитационное моделирование. Дискретно-событийная модель описывает систему как совокупность компонентов, очередей, ресурсов и событий, а затем исследует прохождение запросов в виртуальном времени. Этот подход не заменяет реальное нагрузочное тестирование, но позволяет заранее сравнивать варианты архитектуры, выявлять потенциальные узкие места и уточнять параметры будущего тестового стенда [2, 3].

На практике инструменты нагрузочного тестирования, такие как Apache JMeter и Gatling, ориентированы прежде всего на генерацию реальной нагрузки и получение отчётов по фактически работающей системе [4, 5]. Библиотеки дискретно-событийного моделирования, напротив, дают гибкость программного описания модели, но обычно требуют от пользователя навыков разработки. Возникает промежуточная задача: предоставить специалисту контроля качества удобный веб-интерфейс, который скрывает ручное редактирование кода, но сохраняет возможности симуляционного эксперимента.

В настоящей статье рассматривается не отдельный модуль, а общий результат проекта: веб-ориентированная система моделирования нагрузки для тестирования программного обеспечения. Она объединяет настройку архитектуры стенда, запуск симуляции, запись событий в нескольких форматах, визуализацию результатов, сравнение прогонов и автоматический поиск более подходящих конфигураций. Такой взгляд

позволяет оценить проект как целостный программный комплекс, а не как набор изолированных функций.

Цель исследования

Целью исследования является проектирование и реализация системы, обеспечивающей полный цикл подготовки и анализа симуляционного нагрузочного эксперимента через веб-интерфейс: задание параметров нагрузки и архитектуры стенда, запуск дискретно-событийной модели, сбор логов, расчёт метрик, визуализацию результатов и поддержку сравнения нескольких прогонов.

Материалы и методы исследования

Базой для постановки задачи послужил процесс подготовки нагрузочного тестирования в подразделении контроля качества программных решений. В исходном процессе специалисту требовалось вручную изменять параметры конфигурации, запускать симуляцию из командной строки и самостоятельно интерпретировать результаты. Такой порядок увеличивал время итерации, ограничивал воспроизводимость экспериментов и повышал вероятность ошибок при изменении параметров стенда.

Методологическую основу работы составили системный анализ предметной области, функциональное моделирование бизнес-процессов, объектно-ориентированное проектирование и экспериментальная проверка разработанного прототипа. При проектировании были выделены основные сущности симуляции: настройки стенда, параметры запроса, профиль нагрузки, компоненты инфраструктуры, события логирования и агрегированные показатели результата.

Технологическая база проекта включает Python как общий язык серверной части и симуляционного ядра, SimPy как библиотеку дискретно-событийного моделирования [6], Optuna для автоматического перебора и оптимизации конфигурационных параметров [7], Flask как фреймворк веб-приложения [8] и Chart.js для построения интерактивных графиков в браузере [9]. Для статистического сравнения прогонов используются критерий Колмогорова-Смирнова и расстояние Вассерштейна, доступные в библиотеке SciPy [10].

Результаты и их обсуждение

Разработанная система построена как связка веб-интерфейса, серверного API и расширенного симуляционного фреймворка. Пользователь задаёт параметры нагрузки, архитектуру стенда и дополнительные компоненты через форму в браузере. Flask-приложение валидирует входные данные, преобразует их в структурированные объекты StandSettings и RequestSettings, создаёт каталог прогона и запускает симуляционное ядро как отдельный процесс. После завершения прогона результаты доступны через страницу визуализации и API агрегации.

Таблица 1. Функциональные компоненты разработанной системы [составлено авторами]

Компонент	Назначение	Технологии	Практический результат
Веб-интерфейс	Ввод параметров стенда, профиля нагрузки и формата логирования	Flask, HTML, CSS, JavaScript	Снижение необходимости ручного изменения Python-кода
Симуляционное ядро	Моделирование прохождения запросов через балансировщики, серверы и опциональные компоненты	Python, SimPy	Воспроизводимая проверка архитектурных вариантов

Логирование	Запись событий пользователей и запросов в нескольких форматах	Gatling-like log, JSON Lines, CSV	Единая база для визуализации, экспорта и анализа
Аналитика прогонов	Расчёт RPS, среднего времени ответа, p95/p99 и доли отказов	Python, SciPy	Формализованное сравнение конфигураций
Оптимизация	Автоматический перебор параметров при заданных ограничениях	Optuna	Переход от ручного перебора к серии воспроизводимых экспериментов

Ядро симуляции реализует прохождение запросов через цепочку инфраструктурных компонентов. В базовую модель входят балансировщик, веб-серверы и прикладные серверы, а расширение проекта добавляет Cache, Database, MessageQueue и CDN. Такая структура позволяет исследовать влияние кеширования, очередей, ограниченных пулов соединений и внешних задержек на время ответа и долю отказов. Для балансировки поддержаны стратегии Round-Robin, Least-Connections, Weighted-Round-Robin и Consistent-Hashing.

Отдельная часть системы отвечает за генерацию профиля нагрузки. Помимо равномерного линейного роста пользователей реализованы пиковый, ступенчатый, синусоидальный и burst-профили. За счёт этого можно моделировать не только учебный сценарий плавного нарастания нагрузки, но и более близкие к реальной эксплуатации ситуации: кратковременные всплески, удержание нагрузки на плато и периодические колебания активности.

Модуль логирования записывает события симуляции в Gatling-подобный формат, JSON Lines и CSV. Gatling-совместимая запись сохраняет привычный для специалистов вид журнала нагрузочного прогона, JSON Lines удобен для машинной обработки и расширения набора атрибутов, а CSV подходит для быстрой выгрузки в табличные редакторы и отчёты. Унификация событий LogRequest и LogUser позволяет добавлять новые форматы без переписывания логики симуляции.

После отправки формы система сохраняет параметры прогона, запускает симуляционное ядро и отображает состояние выполнения в браузере. На странице результата пользователь видит идентификатор запуска, журнал выполнения, агрегированную сводку и графики, построенные по событиям симуляции. Это позволяет проследить полный путь от настройки эксперимента до интерпретации метрик без обращения к промежуточным файлам.

Общий поток работы системы моделирования нагрузки представлен на рисунке 1.

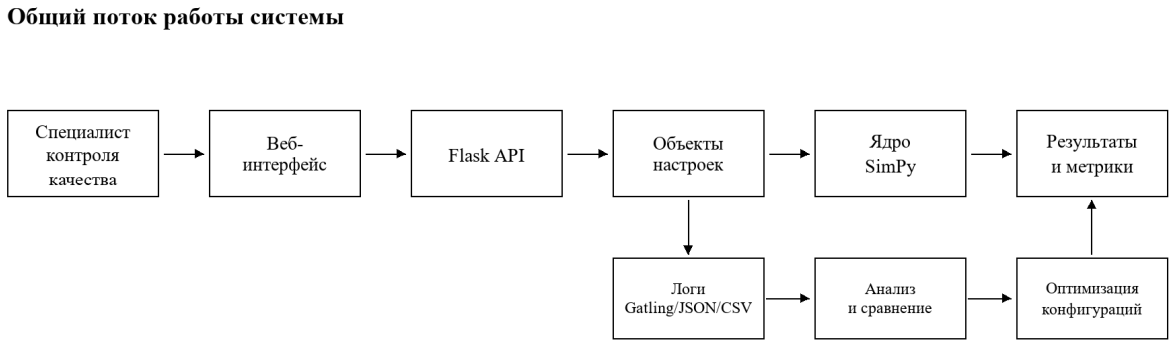

Рисунок 1. Общий поток работы системы моделирования нагрузки. [разработано авторами]

На рисунке 2 представлена модульная структура программного комплекса.

Рисунок 2. Модульная структура программного комплекса. [разработано авторами]

Веб-интерфейс выполняет роль рабочей оболочки для специалиста контроля качества. На главной странице доступны переходы к запуску новой симуляции, истории прогонов и сравнению результатов. Страница настройки объединяет параметры нагрузки, архитектуру стенда, выбор стратегии балансировки, подключение Cache и CDN, настройки серверов и формат логирования. Результаты отображаются в виде графиков времени отклика, RPS и процента отказов, что позволяет оценивать поведение модели без внешнего генератора HTML-отчётов.

Модуль анализа прогонов агрегирует события по секундам, рассчитывает среднее время ответа, перцентили p50, p95 и p99, число успешных и неуспешных запросов, а также процент отказов. Для сравнения двух запусков применяется статистическое сопоставление распределений времени ответа. В отчёте дополнительно фиксируются точки деградации и отказа, что помогает не только визуально сравнить графики, но и получить формализованный вывод о различии конфигураций.

Оптимизационный модуль использует Optuna для подбора параметров и сравнения стратегий поиска. В системе предусмотрены сценарии оптимизации количества серверов и других изменяемых параметров при заданных ограничениях по деградации и отказам. Практическая ценность такого подхода заключается в том, что специалист может перейти от ручного перебора вариантов к серии воспроизводимых экспериментов, где каждая попытка сохраняется как отдельный прогон.

Опытная эксплуатация проводилась на контрольных сценариях, в которых изменялись количество серверов, стратегия балансировки, форма профиля нагрузки и наличие кеширующих компонентов. Результаты подтвердили ожидаемые закономерности: увеличение числа серверов снижает нагрузку на отдельный узел до появления другого узкого места; кеширование и CDN уменьшают среднее и p95 время ответа при высокой вероятности попадания в кеш; разные стратегии балансировки дают различимый эффект при неоднородных параметрах серверов.

Таблица 2. Результаты контрольных симуляционных прогонов [составлено авторами]

Сценарий	Конфигурация	Запросов	avg, мс	p95, мс	max RPS	Отказы, %
Базовый стенд	1 сервер, Round-Robin, без Cache/CDN	6618	339.5	751.0	97	1.15
Масштабирование	2 сервера, Round-Robin, без Cache/CDN	7311	45.7	119.0	119	0.0

Кеширующий слой	2 сервера, Round-Robin, Cache+CDN	7321	7.8	20.0	123	0.0
Least-Connections	2 сервера, Least-Connections, без Cache/CDN	7300	59.5	178.0	123	0.0
Пиковая нагрузка	2 сервера, Round-Robin, peak-профиль	12956	31.0	65.0	98	0.0

Численные результаты показывают, что переход от одного сервера к двум снижает среднее время ответа с 339,5 до 45,7 мс и устраняет отказы в выбранном профиле нагрузки. При добавлении Cache и CDN с высокой вероятностью попадания в кеш среднее время ответа снижается до 7,8 мс, а p95 — до 20,0 мс. Сценарий Least-Connections в данной конфигурации не улучшил результат относительно Round-Robin, что показывает важность экспериментальной проверки стратегий балансировки, а не только их формального наличия.

Главный результат проекта состоит в объединении ранее разрозненных действий в единый цикл. Пользователь больше не ограничен ручным редактированием Python-кода и просмотром отдельных файлов логов: он формирует конфигурацию, запускает симуляцию, получает графики, возвращается к истории прогонов и сравнивает альтернативы в одном веб-приложении. При этом техническая реализация остаётся расширяемой: новые компоненты стенда, профили нагрузки и форматы отчётности могут добавляться в существующую структуру без изменения пользовательского сценария.

К ограничениям текущей версии относится абстрактность модели: симуляция описывает основные закономерности прохождения запроса, но не воспроизводит все детали сетевых протоколов, аппаратных ресурсов, файловых операций и внутренних механизмов конкретных СУБД. Кроме того, поддерживаемая топология ориентирована на типовой стек с балансировщиком, веб-серверами и прикладными узлами. Эти ограничения не отменяют исследовательской ценности системы, но задают направление дальнейшего развития.

Выводы

Проведённая работа показала возможность реализации веб-ориентированной системы, объединяющей дискретно-событийное моделирование нагрузки, настройку архитектуры стенда, многоформатное логирование, визуализацию, статистическое сравнение прогонов и автоматическую оптимизацию конфигураций. В отличие от классических инструментов реального нагрузочного тестирования, система позволяет проводить предварительную оценку архитектурных решений без развёртывания полноценного стенда.

Интеграция SimPy, Flask, Chart.js и Optuna позволила создать единый программный контур, в котором одни и те же структурированные настройки используются веб-интерфейсом, симуляционным ядром, модулем анализа и оптимизатором. Это повышает воспроизводимость экспериментов и снижает трудоёмкость подготовки прогонов для специалистов, не работающих напрямую с программным кодом.

Дальнейшее развитие системы целесообразно связать с визуальным редактором топологии, расширением набора моделируемых компонентов, более строгой серверной валидацией параметров, экспортом отчёта по серии прогонов и переносом фонового выполнения на специализированный механизм очередей. Эти изменения позволят приблизить прототип к промышленному инструменту поддержки нагрузочного тестирования и архитектурного анализа.

Список литературы:

1. Орлов С. А., Цилькер Б. Я. Технологии разработки программного обеспечения: учебник для вузов. - 4-е изд. - СПб.: Питер, 2012. - 608 с.
2. Гнеденко Б. В., Коваленко И. Н. Введение в теорию массового обслуживания. - 4-е изд. - М.: ЛКИ, 2012. - 400 с.
3. Советов Б. Я., Яковлев С. А. Моделирование систем: учебник для вузов. - 4-е изд. - М.: Высшая школа, 2005. - 343 с.
4. Apache Software Foundation. Apache JMeter User's Manual: [сайт]. - URL: https://jmeter.apache.org/usermanual/ (дата обращения: 31.05.2026). - Текст: электронный.
5. Gatling. Gatling Documentation: [сайт]. - URL: https://docs.gatling.io/ (дата обращения: 31.05.2026). - Текст : электронный.
6. SimPy Developers. SimPy Documentation: [сайт]. - URL: https://simpy.readthedocs.io/ (дата обращения: 31.05.2026). - Текст : электронный.
7. Akiba T. Optuna: A Next-generation Hyperparameter Optimization Framework / T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. - 2019. - DOI: 10.1145/3292500.3330701.
8. Pallets Projects. Flask Documentation \: [сайт]. - URL: https://flask.palletsprojects.com/ (дата обращения: 31.05.2026). - Текст: электронный.
9. Chart.js Contributors. Chart.js Documentation: [сайт]. - URL: https://www.chartjs.org/docs/ (дата обращения: 31.05.2026). - Текст : электронный.
10. SciPy Developers. scipy.stats statistical functions : [сайт]. - URL: https://docs.scipy.org/doc/scipy/reference/stats.html (дата обращения: 31.05.2026). - Текст: электронный.

References:

1. Orlov, S. A., & Tsilker, B. Ya. (2012). Software Development Technologies (4th ed.). Saint Petersburg: Piter. (In Russian).
2. Gnedenko, B. V., & Kovalenko, I. N. (2012). Introduction to Queueing Theory (4th ed.). Moscow: LKI. (In Russian).
3. Sovetov, B. Ya., & Yakovlev, S. A. (2005). System Modeling (4th ed.). Moscow: Vysshaya shkola. (In Russian).
4. Apache Software Foundation. Apache JMeter User's Manual. Available at: https://jmeter.apache.org/usermanual/ (accessed: 31.05.2026).
5. Gatling. Gatling Documentation. Available at: https://docs.gatling.io/ (accessed: 31.05.2026).
6. SimPy Developers. SimPy Documentation. Available at: https://simpy.readthedocs.io/ (accessed: 31.05.2026).
7. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. DOI: 10.1145/3292500.3330701.

8. Pallets Projects. Flask Documentation. Available at: https://flask.palletsprojects.com/ (accessed: 31.05.2026).
9. Chart.js Contributors. Chart.js Documentation. Available at: https://www.chartjs.org/docs/ (accessed: 31.05.2026).
10. SciPy Developers. scipy.stats statistical functions. Available at: https://docs.scipy.org/doc/scipy/reference/stats.html (accessed: 31.05.2026).