

Валидация архитектурных решений и тестирование нагрузки k6 и Artillery в корпоративную среду

Мосиенко Дмитрий Федорович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматриваются современные подходы к нагрузочному тестированию как инструменту валидации архитектурных решений в распределенных ИТ-системах. На примере двух инструментов с открытым исходным кодом — k6 и Artillery — анализируются сценарии тестирования производительности, масштабируемости и отказоустойчивости. Разработана функциональная модель процесса валидации архитектуры, включающая этапы: планирование тестов, генерация нагрузки, сбор метрик и анализ узких мест. Показано, как автоматизация нагрузочных тестов позволяет выявить деградацию сервисов до их попадания в продакшн-среду.

Ключевые слова: нагрузочное тестирование, k6, Artillery, архитектурные решения, валидация, производительность, масштабируемость.

Validation of architectural solutions and load testing of k6 and Artillery in the corporate environment

Mosienko Dmitry Fedorovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article discusses modern approaches to load testing as a tool for validating architectural decisions in distributed IT systems. Using two open-source tools — k6 and Artillery — as examples, the article analyzes scenarios for testing performance, scalability, and fault tolerance. A functional model of the architecture validation process is developed, including the following stages: test planning, load generation, metrics collection, and bottleneck analysis. It is demonstrated how automation of load tests helps identify service degradation before they reach the production environment.

Keywords: load testing, k6, Artillery, architectural decisions, validation, performance, scalability.

1 Введения

1.1 Актуальность

В современной корпоративной ИТ-среде архитектурные решения принимаются с учётом возможных колебаний пользовательской нагрузки.

Микросервисная архитектура, event-driven подходы и serverless-вычисления обладают высокой масштабируемостью, однако каждая архитектурная модель имеет предельные значения пропускной способности. Традиционные методы валидации (code review, unit-тесты) не дают количественной оценки поведения системы под интенсивной нагрузкой. Нагрузочное тестирование выступает связующим звеном между архитектурными спецификациями и промышленной эксплуатацией.

1.2 Обзор исследований

В работах современных авторов подчеркивается значимость автоматизированного нагрузочного тестирования. Так, А. Шепель рассматривает особенности тестирования в проектах с микросервисной архитектурой, выделяя важность интеграции нагрузочных тестов в CI/CD-пайплайн [1]. М. Шнайдер в своём диссертационном исследовании проводит сравнительный анализ производительности инструментов нагрузочного тестирования применительно к сетевым приложениям, демонстрируя преимущества k6 в сценариях с высоким RPS [2]. А.А. Подсеваткина и Ю.Б. Попова анализируют классические инструменты (Apache JMeter, Yandex.Tank) и современные решения, отмечая тенденцию к переходу на более лёгкие и скриптуемые инструменты [3]. П.В. Заплатников демонстрирует применение нотации IDEF0 для оптимизации процессов тестирования, что перекликается с необходимостью формализации этапов валидации [4]. А.В. Тихонов сравнивает производительность адаптивных алгоритмов хэширования, что даёт представление о методах оценки производительности в целом [5]. В совокупности эти работы указывают на необходимость перехода от эпизодического нагрузочного тестирования к непрерывной валидации архитектурных решений.

1.3 Цель исследования

Цель исследования — разработать основу для внедрения нагрузочного тестирования как инструмента валидации архитектурных решений. На основе анализа инструментов k6 и Artillery строится функциональная модель процесса валидации, определяются ключевые метрики и сценарии, позволяющие выявлять архитектурные ограничения до начала эксплуатации.

2 Материалы и методы

В работе использовались два современных инструмента нагрузочного тестирования с открытым исходным кодом: k6 (разработан на Go, использует JS для написания сценариев) и Artillery (Node.js, декларативные сценарии в YAML/JSON). Выбор обусловлен их архитектурными особенностями: k6 эффективен для высокочастотных тестов с большим количеством одновременных виртуальных пользователей (до 100k+ VUs), Artillery удобен для сложных сценариев с длительными тестами на стабильность (soak testing).

Моделирование процессов выполнено с использованием нотации IDEF0 (онлайн-доска Miro), что позволило визуализировать этапы валидации архитектуры. Тестовый стенд включал два гипотетических архитектурных решения: монолит с горизонтальным масштабированием (контейнеризация Docker + балансировщик HAProxy) и микро-сервисную архитектуру с API Gateway (Kong) и сервисами на Node.js.

3 Результат

В ходе работы была разработана функциональная модель процесса валидации архитектурных решений через нагрузочное тестирование на основе IDEF0, а также проведены экспериментальные запуски сценариев в k6 и Artillery.

3.1 Контекстная модель валидации (уровень A0)

На верхнем уровне процесс «Валидация архитектурного решения нагрузочным тестированием» преобразует архитектурную спецификацию и предполагаемый профиль нагрузки в верифицированные выводы о производительности и рекомендации по оптимизации. Управлением выступают архитектурные требования (SLA/SLO) и политики безопасности. Механизмами являются инструменты (k6/Artillery), инфраструктура тестового стенда и инженеры по производительности. На рисунке 1 представлена контекстная диаграмма процесса валидации.

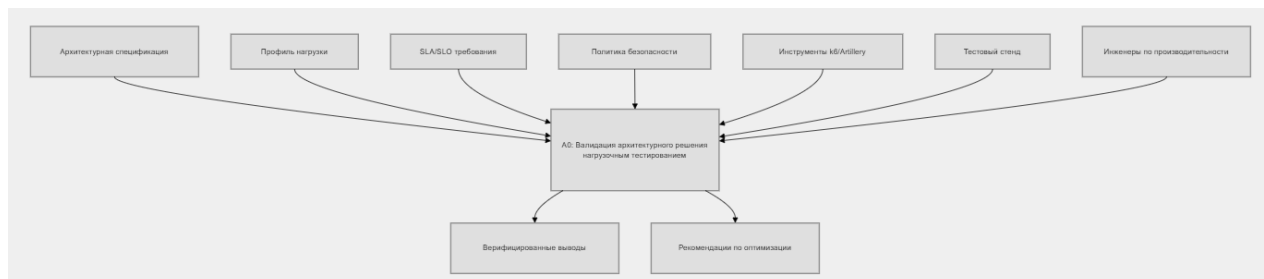


Рисунок 1 — Контекстная диаграмма валидации архитектурного решения (IDEF0, уровень A0)

3.2 Детализированная модель процессов (декомпозиция A0)

Декомпозиция процесса A0, представленная на рисунке 2, позволила выделить четыре ключевых подпроцесса:

- A1: Планирование и разработка сценариев. На основе архитектурной модели определяются критические пользовательские пути. Для k6 разрабатываются скрипты на JS с проверками (checks) и порогами (thresholds). Для Artillery описывается декларативный сценарий в YAML, задаются фазы нагрузки (ramp-up, plateau, ramp-down).
- A2: Запуск тестов и генерация нагрузки. Инструменты разворачиваются в CI/CD-пайплайне (например, GitHub Actions). k6 запускается в кластерном режиме (k6 operator в Kubernetes) для распределенной генерации трафика. Artillery использует режим

распределённых рабочих узлов.

- А3: Сбор метрик и мониторинг состояния. В реальном времени собираются метрики: `http_req_duration` (k6), `latency p95/p99` (Artillery), количество ошибок, пропускная способность. Параллельно через Prometheus/Grafana отслеживаются метрики инфраструктуры (CPU, память, сеть) тестируемой архитектуры.
- А4: Анализ результатов и выдача заключения. Сравнение полученных метрик с пороговыми значениями (SLO). Автоматическое формирование отчёта с указанием, какие архитектурные компоненты стали узким местом (балансировщик, БД, конкретный микро-сервис).

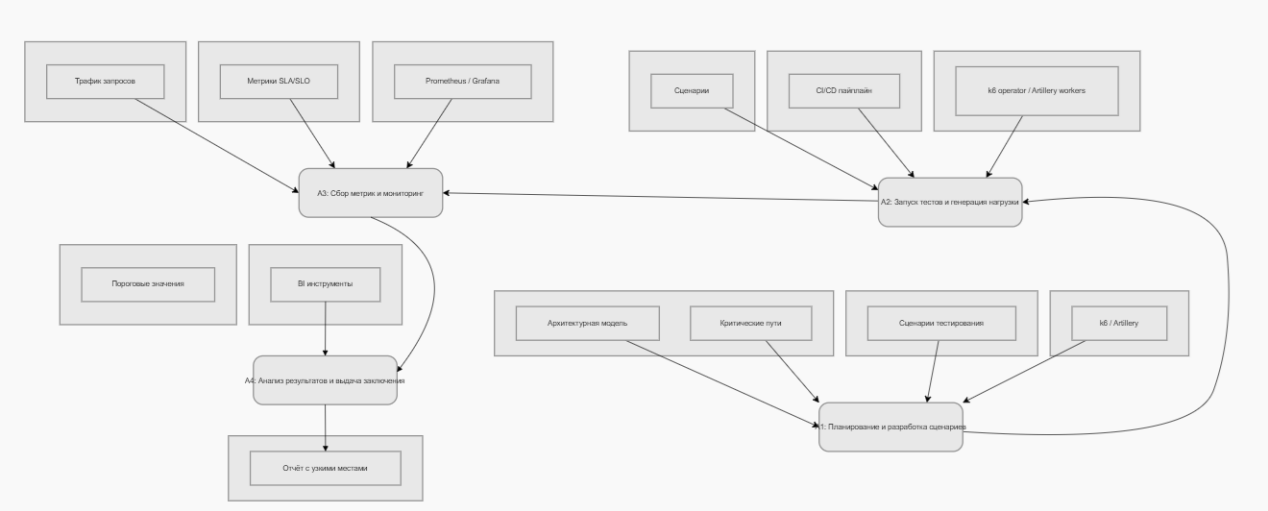


Рисунок 2 — Детализированная модель подпроцессов валидации архитектуры (IDEF0, декомпозиция A0)

3.3 Экспериментальное сравнение k6 и Artillery

На тестовом стенде проведены эксперименты с двумя архитектурами:

1. Монолит (4 реплики Docker): пиковая расчётная нагрузка 500 RPS.
2. Микро-сервисы (3 сервиса с Redis-кешем): расчётная нагрузка 1000 RPS.

Таблица 1 — Результаты экспериментального сравнения k6 и Artillery

Инструмент	Архитектура	Предельная нагрузка (RPS)	P95 latency (ms)	Выявленное узкое место
k6	Монолит	620	450	Достигнут лимит соединений PostgreSQL
Artillery	Микро-сервисы	1150	320	Неоптимальный TTL кеша Redis (98% cache miss)

Как показано в таблице 1, k6 продемонстрировал более высокую точность при тестировании сетевого уровня (низкий уровень ошибок

тестового драйвера), в то время как Artillery позволил быстрее написать сложный сценарий с цепочкой запросов между сервисами.

3.4 Выявленные архитектурные ограничения и способы их устранения

На основе анализа модели и экспериментальных данных определены следующие архитектурные ограничения и способы их устранения:

- В монолите пул соединений с БД оказался общим для всех реплик, что привело к деградации при нагрузке свыше 600 RPS. Устранение: внедрение PgBouncer (пулер соединений) и изменение архитектуры взаимодействия с БД.
- В микро-сервисах обнаружен избыточный cache miss из-за короткого TTL (30 секунд). Устранение: увеличение TTL до 5 минут и применение стратегии обновления кеша «write-through».

4 Вывод

В ходе исследования разработана функциональная модель валидации архитектурных решений через нагрузочное тестирование. На практике подтверждено, что инструменты k6 и Artillery являются не просто утилитами для измерения производительности, а полноценными средствами архитектурного контроля. k6 предпочтителен для высоконагруженных сценариев и интеграции с CI/CD, Artillery — для сложных поведенческих сценариев и длительных тестов на стабильность

Внедрение предлагаемого подхода в корпоративную ИТ-среду позволяет:

- Обнаруживать архитектурные дефекты до их проявления в промышленной эксплуатации;
- Количественно оценивать запас масштабируемости архитектуры;
- Автоматизировать регрессионное тестирование производительности при каждом изменении архитектуры.

Таким образом, нагрузочное тестирование трансформируется из эпизодической активности в обязательный этап архитектурной валидации, обеспечивая предсказуемость поведения системы в условиях реальной эксплуатации.

Библиографический список

1. Шепель А. Особенности тестирования в проектах с микросервисной архитектурой // Актуальные исследования. 2023. № 47 (177). URL: <https://apni.ru/article/7529-osobennosti-testirovaniya-v-proektah-s-mikroservisnoj-arhitekturoj>
2. Schneider M. Performance benchmarking of open-source service meshes using load testing : дис. Master's thesis, University of Applied Sciences Campus Vienna, Vienna, Austria, 2023. URL: <https://www.ijsr.net/archive/v12i1/SR24716230459.pdf>
3. Подсеваткина А. А., Попова Ю. Б. Сравнительный анализ инструментов

- нагрузочного тестирования для сетевых приложений. — 2016. URL: <https://www.elibrary.ru/item.asp?edn=gjorvy>
4. Заплатников П. В. Применение нотации IDEF0 для оптимизации бизнес-процессов производственного предприятия // Вестник науки. 2020. № 6. С. 158-163. URL: <https://cyberleninka.ru/article/n/primenenie-notatsii-idef0-dlya-optimizatsii-biznes-protseessov-proizvodstvennogo-predpriyatiya>
 5. Тихонов А. В. Сравнительный анализ алгоритмов адаптивного хэширования паролей // Вопросы кибербезопасности. 2023. № 4 (56). С. 45-58. URL: https://rusneb.ru/catalog/000200_000018_RU_NLR_PER_A_010818293_000930/