WELCOME DAY

Школа по подготовке специалистов по нагрузочному тестированию

PerformanceLab
Тестирование и обеспечение качества

План презентации

1 Введение

2 Путь тестировщика

3 Теоретическая часть

4 О технологиях

5 О проектах НТ

6 План школы

7 Заключение

Введение →

О компании

Работа в Перфоманс Лаб – это обеспечение качества и производительности ИТ-систем в промышленных масштабах.

Банкоматы, онлайн банкинг, электронные государственные услуги и прочие полезные технологии в масштабах страны – мы всё это тестируем.

А сейчас и у тебя появилась возможность делать мир лучше вместе с нами!

Миссия компании

Мы улучшаем цифровой мир

В масштабах космоса еще секунду назад мы охотились на мамонтов , а уже сейчас летаем на Марс. Везде есть Интернет, а машины становятся умнее некоторых людей и компьютерные программы есть даже в чайниках.

Ты можешь быть не просто зрителем, а находится в гуще событий – добиваться безупречного качества современных технологий.

Мы, Команда Перфоманс Лаб, объединяем талантливых людей и улучшаем цифровой мир. Каждый день!

Мы не только предоставляем услуги по тестированию, но и разрабатываем свои собственные продукты

BOOMQ.IO

UXCrowd

Ваш путь к нам

1 этап

= 1065 резюме

2 этап

= 213 контактов

3 этап

= 15 человек

Путь тестировщика

+ ИПР*

Ведущий инженер по производительности

≈ 2-3 года

Инженер по производительности

≈ 1,5-2 года

Дизайнер тестов

≈ 1 год

Инженер по тестированию

≈ 9 месяцев

Школьник

≈ 3 недели

*ИПР - Индивидуальный План Развития

Основные направления услуг:

- Функциональное тестирование
- Автоматизированное тестирование
- **Нагрузочное тестирование**
- Юзабитили-тестирование
- Разработка ПО

Что такое тестирование?

Тестирование ПО

- процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом

Функциональное тестирование

- это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Автоматизированное тестирование

- часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

Что такое тестирование производительности?

Нагрузчик – это аналитик, разработчик, тестировщик и архитектор одновременно.

Мы видим систему целиком и это позволяет работать с глобальными проблемы.

Что мы хотим от тестирования производительности?

1. Определение параметров производительности системы

2. Определение максимальной производительности системы

3. Проверка отказоустойчивости системы

4. Проверка надежности системы

5. Подбор оптимальной конфигурации системы

6. Воспроизведение проблем промышленной среды

7. Выявление «узких мест»

Инструментарий

THE APACHE SOFTWARE FOUNDATION

MICRO FOCUS LoadRunner Professional

Gatling

A distributed Load Testing Tool TSUNG

SILK PERFORMER

BOOMQ.IO

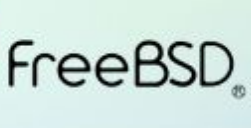
О технологиях →

С чем приходится работать?

Языки

С чем приходится работать?

Операционные системы



С чем приходится работать?

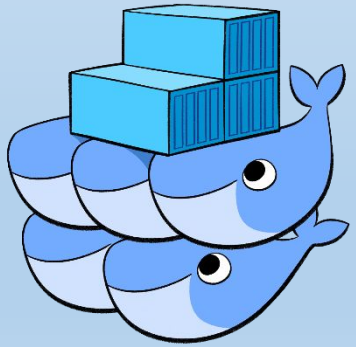
Специальное ПО

С чем приходится работать?

Базы данных

APP сервера

Технологии контейнеризации



Continuous integration

Jenkins

Bamboo

Стандартные этапы

Что надо сделать, чтобы достичь цели?

1. Анализ системы. Разработка и согласование методики нагрузочного тестирования;
2. Установка и настройка инфраструктуры для проведения НТ;
3. Проверка и отладка контура для разработки средств НТ (СНТ);
4. Подготовка СНТ (скрипты, сценарии, эмуляторы\заглушки, тестовые данные);
5. Настройка средств мониторинга на тестовых серверах;
6. Проведение испытаний в тестовой среде;
7. Системный анализ результатов, выявление "узких мест";
8. Проверка рекомендаций по оптимизации системы;
9. Подготовка итогового отчёта.

Что дальше?

Тестирование производительности, как метод прогнозирования утилизации ресурсов для отдельно взятой ИС.

На выходе:

- Определена максимальная производительность системы → ясна «дата смерти» системы
- Понятна картина утилизации оборудования (какой ресурс наиболее востребован)
- Выявлены узкие места в ПО, коде и алгоритмах работы с системой → есть возможность повысить эффективность использования текущих ресурсов

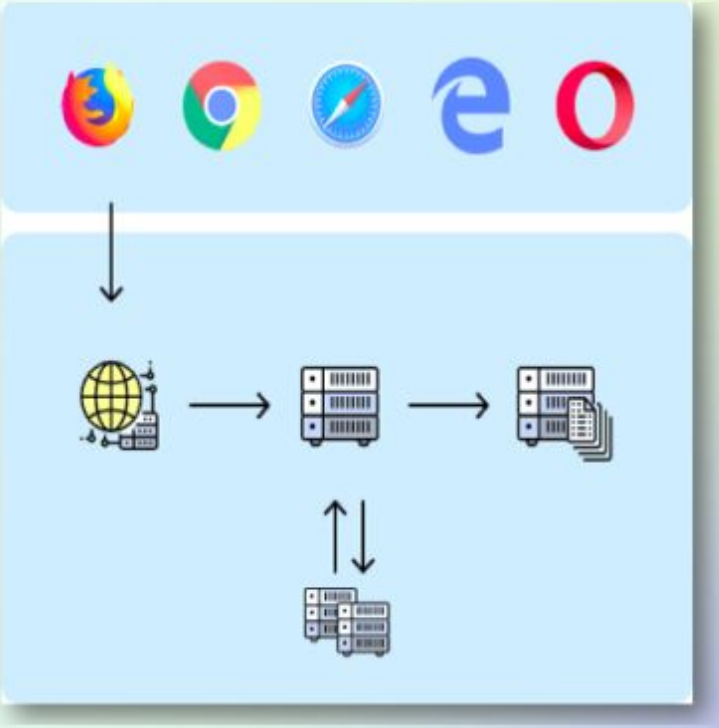
Описание исследуемой системы

Система

Новая высоконагруженная система для приёма и обработки платежей (на микросервисах)

Требуется

- Надёжная защита
- Высокая производительность
- Доступность 24/7



Первые результаты

Оцениваем

В первых тестах производительность системы равнялась:

5% или **5000**

от требуемой транзакций в час

Нужно:

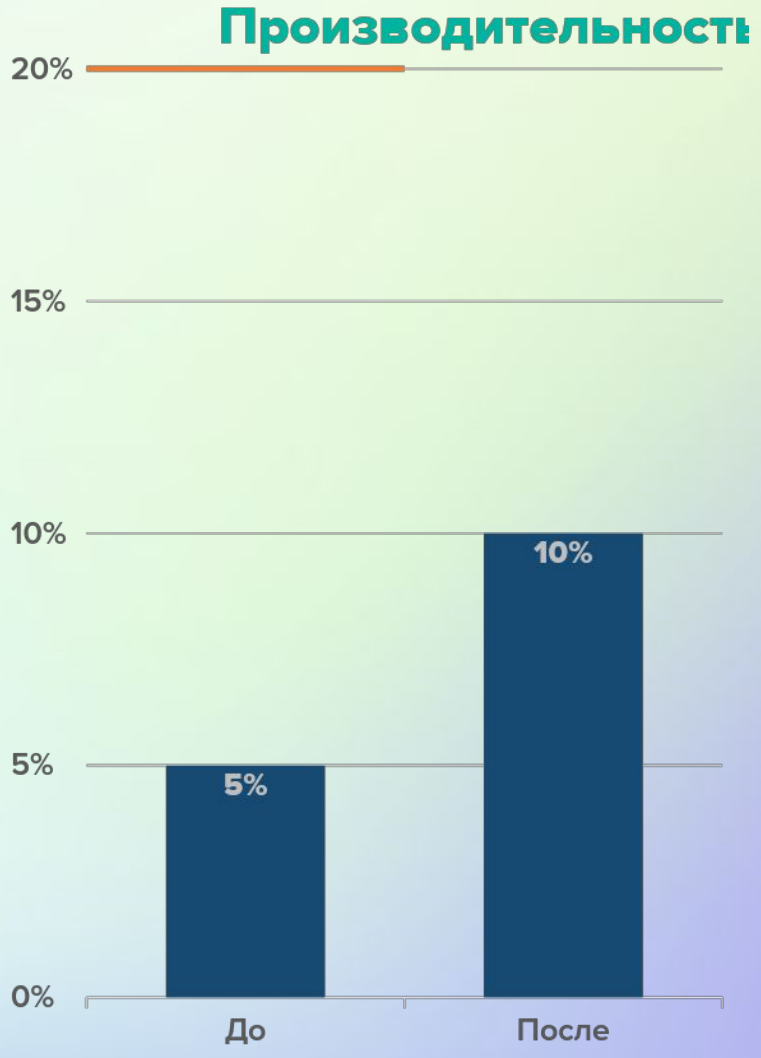
100 000

транзакций в час

Проблема №1 - Сеть

Web-кэширование

При загрузке статические ресурсы занимали всю сеть (в системе не было клиентского кэширования страниц)

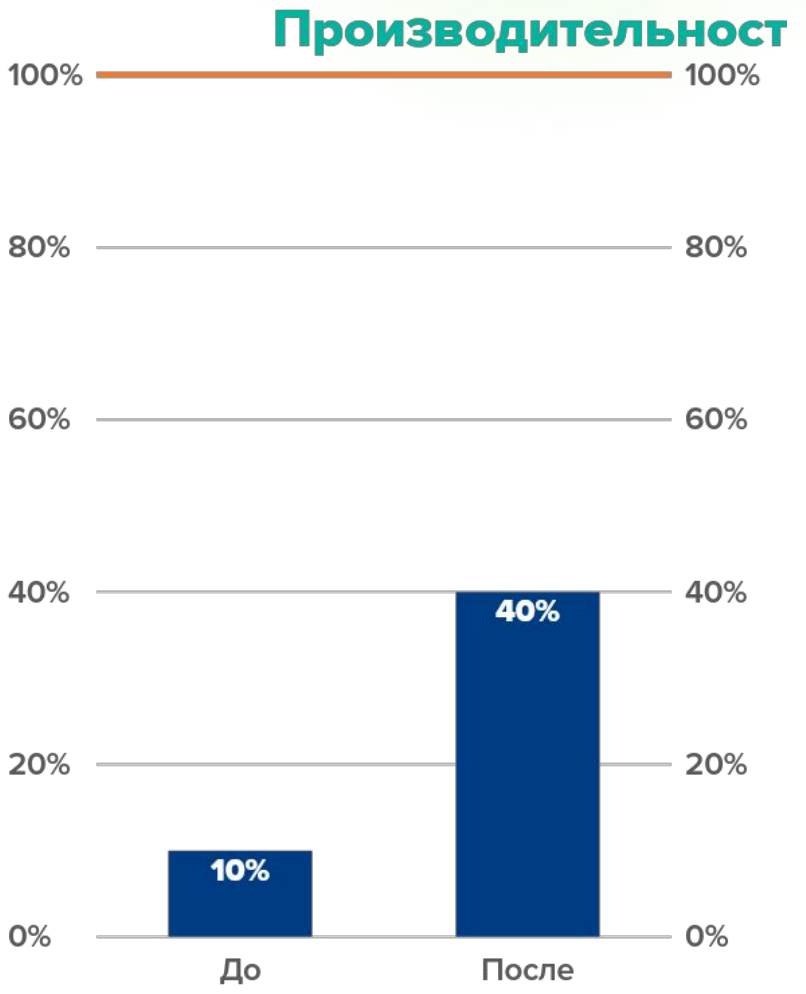


Проблема №2 — Работа с базой данных

Проблемы БД

Выбрать все красные машины
- Выбрать из них все быстрые машины
  - Выбрать из них машины этого года
    - Выбрать из них только BMW

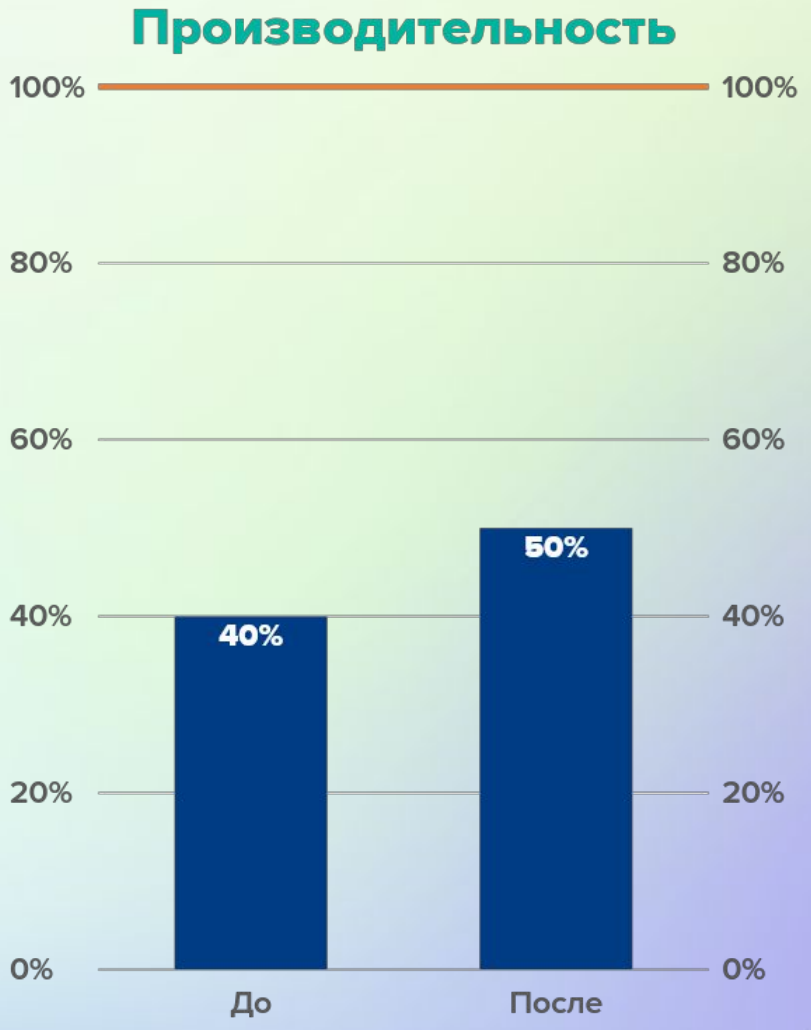
А может лучше наоброт?



Проблема №3 — Виртуализация

Производительность виртуальных машин

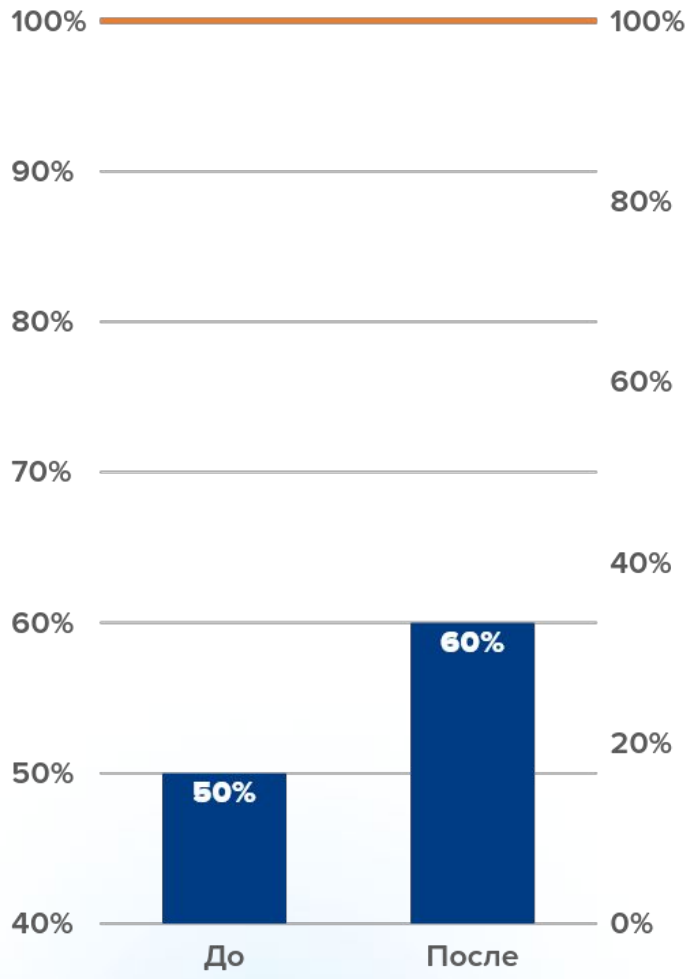
VM одинаковые.
А гипервизоры разные...



Проблема №4 — Коннекты

```
100 %  All Calls • 4.424 ms
20.8 %  MoveNext • 922 ms • LinqToDB.Data.DataConnection+<ExecuteReaderAsync>
20.8 %  SetExistingTaskResult • 922 ms • System.Runtime.CompilerServices.Async
20.8 %  MoveNext • 922 ms • LinqToDB.Data.DataConnection+QueryRunner+<Exec
20.8 %  SetExistingTaskResult • 922 ms • System.Runtime.CompilerServices.As
20.8 %  MoveNext • 922 ms • LinqToDB.Linq.QueryRunner+<ExecuteQueryAsyn
20.8 %  SetExistingTaskResult • 922 ms • System.Runtime.CompilerServices
20.8 %  MoveNext • 921 ms • LinqToDB.Linq.Builder.FirstSingleBuilder+FirstS
20.8 %  SetExistingTaskResult • 921 ms • System.Runtime.CompilerServ
20.8 %  MoveNext • 921 ms • LinqToDB.Linq.ExpressionQuery`1+<LinqTo
20.8 %  SetExistingTaskResult • 921 ms • System.Runtime.Compiler
20.8 %  MoveNext • 921 ms • Rbp.Core.AppServices.Docs.Files.Services.
20.8 %  SetExistingTaskResult • 921 ms • System.Runtime.Compile
20.8 %  MoveNext • 920 ms • Rbp.Core.AppServices.Docs.Files.Serv
20.8 %  async UpdateContentHashes • 920 ms • Rbp.Core.App
20.1 %  ComputeHash • 888 ms • Rbp.Common.Security.Crypto
18.8 %  BlockUpdate • 833 ms • Org.BouncyCastle.Crypto.Dig
```

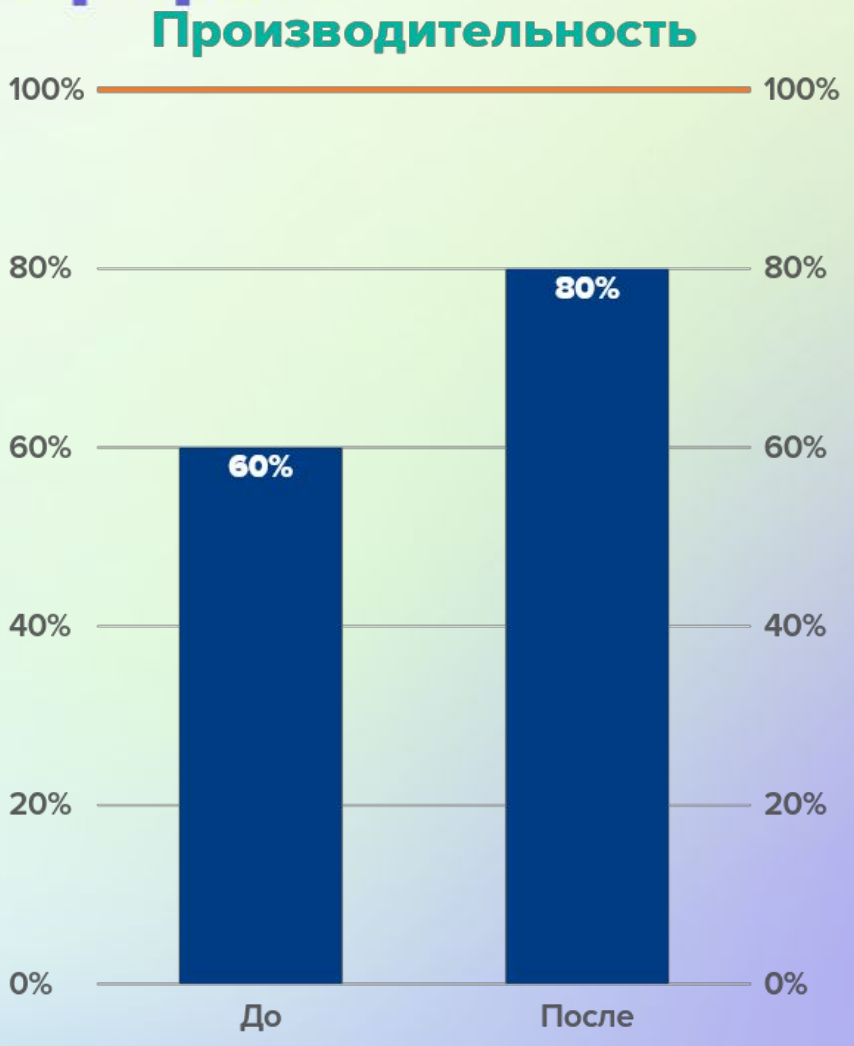
Производительность



Проблема №5 — Платформа

Прибавка к производительности

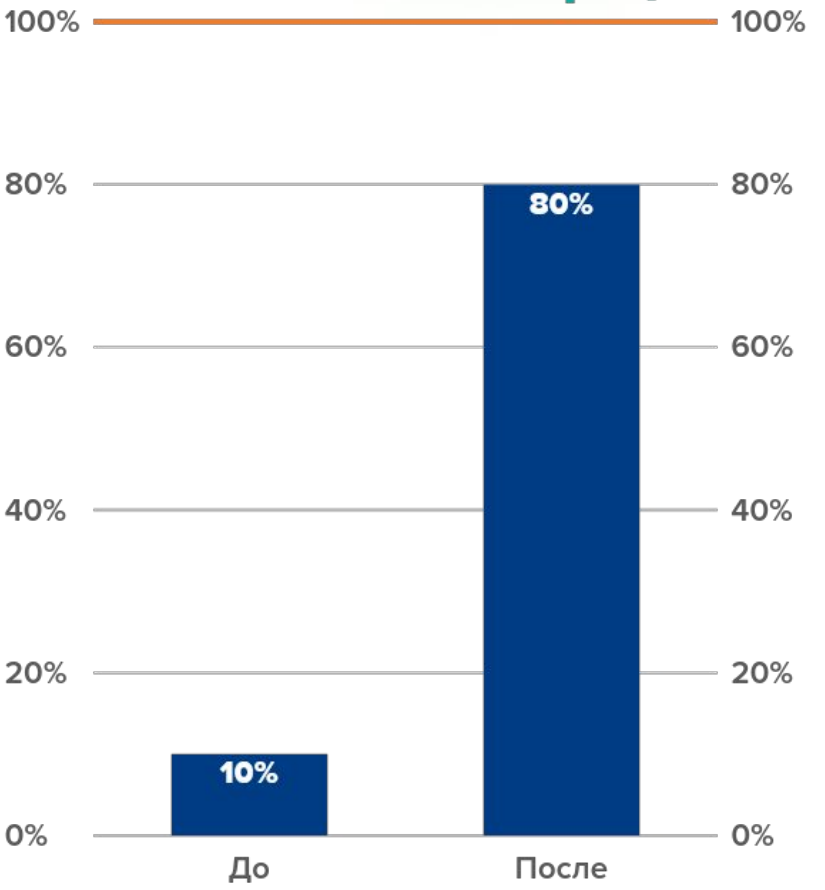
.Net Core 2.2 вместо старого .Net 4



Проблема №6 — Очереди

Длительные операции и система очередей

Производительность «легких" операций



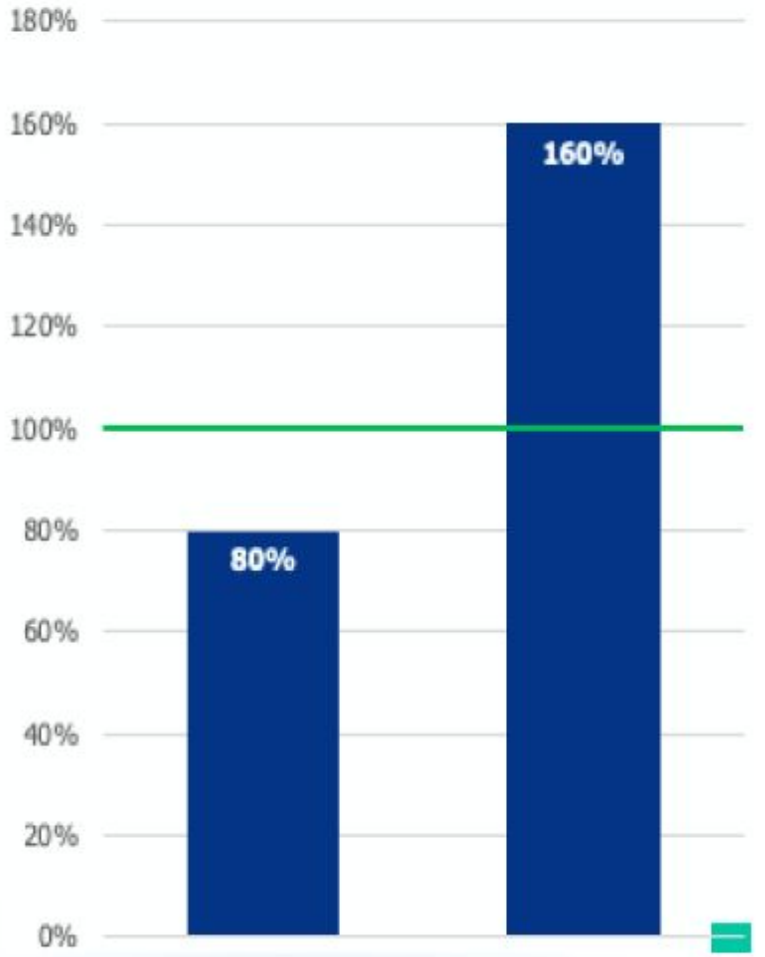
Проблемы. Инфографика

Список проблем, которые мы выявили и устранили

Проблема №7— Производительность

Скорректировать профиль нагрузочного тестирования на основе реальной статистики.

Операция	Было	Стало
Покупка (ключевая операция)	33 000	68 000
Логины (вход в систему)	50 000	20 000
Выписка (генерация отчетов)	17 000	12 000
Итого:	100 000	100 000



Результат

Тысяча платежей в час

До PFLB

После PFLB

цель

План школы

1. Теория НТ;
2. Инструменты НТ (LR, Jmeter, ALM, Boomq);
3. SQL;
4. Языки программирования (Java, Python);
5. Учебный проект;
6. Внутреннее собеседование (экзамен);
7. Собеседование с Заказчиком;
8. Трудоустройство.

P.S. Дедлайн сдачи всех домашек – допуск к внутреннему собесу 70%

Спасибо за внимание

ООО “Перфоманс Лаб”

г.Москва, ул.Барклая, д.6, стр.5, оф.511

+7 495 989 6165

info@pflb.ru

www.performance-lab.ru