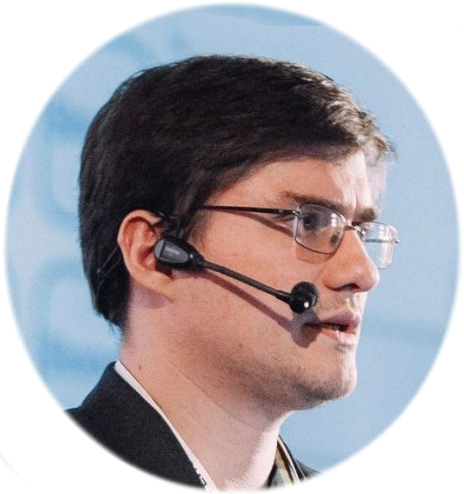
Теория и практика нагрузочного тестирования

Алексей Рагозин
alexey.ragozin@gmail.com

О докладчике



Занимаюсь высоконагруженными системами на Java с 2006.
Разрабатывал софт для торговли на фондовых рынках, телекоме, e-commerce, RTB, здравоохранения.
Выступаю на конференциях, организую митапы, провожу тренинги.

Email: alexey.ragozin@gmail.com
Blog: blog.ragozin.info
Github: https://github.com/aragozin
Митапы и вебинары: https://aragozin.timepad.ru

О чём этом доклад?

Многие IT системы относятся к классу
систем массового обслуживания (СМО)

Микросервисы, веб приложения и пр.

СМО довольно давно изучаются математикой

“Queueing theory”, A. Erlang – 1909, “Исследование операций” – начало XX века

И в теории, теория не должна отличаться от практики,
но на практике это не всегда так.

Виды тестирования

Тестирование производительности

- Load testing – *нагрузочное тестирование*
- Volume testing – *объёмное тестирование*
- Stress testing – *стрессовое тестирование*
- Endurance testing – *тестирование стабильности*
- Scalability testing – *тестирование масштабируемости*

Виды тестирования

Тестирование производительности

- Load testing – *нагрузочное тестирование*
- Volume testing – *объёмное тестирование*
- Stress testing – *стрессовое тестирование*
- Endurance testing – *тестирование стабильности*
- Scalability testing – *тестирование масштабируемости*

Цель нагрузочного тестирования

Нефункциональные требования

- Нагрузка 100 RPS

Цель нагрузочного тестирования

Нефункциональные требования

- ~~Нагрузка 100 RPS~~ – не корректно!

Цель нагрузочного тестирования

Нефункциональные требования

- ~~Нагрузка 100 RPS~~ – не корректно!
- P99 < 100мс при нагрузке 100 RPS

SLA

Проектная нагрузка

Цель нагрузочного тестирования

Нефункциональные требования

- ~~Нагрузка 100 RPS~~ – не корректно!
- P99 < 100мс при нагрузке 1000 пользователей*

SLA

Проектная нагрузка

*) Каждый пользователь создаёт нагрузку с определённым профилем

Цель нагрузочного тестирования

Нефункциональные требования

- ~~Нагрузка 100 RPS~~ – не корректно!
- P99 < 100мс при нагрузке 100 RPS

SLA

Проектная нагрузка

Цель: проверка SLA на проектном уровне нагрузки

Цель нагрузочного тестирования

Нефункциональные требования

- ~~Нагрузка 100 RPS~~ – не корректно!
- P99 < 100мс при нагрузке 100 RPS

SLA

Проектная нагрузка

Цель: проверка SLA на проектном уровне нагрузки

Цель*: поиск точки деградации системы

Системы массового обслуживания

(определение из учебника http://mathhelpplanet.com/static.php?p=sistema-massovogo-obsluzhivaniya)

Математическая модель системы массового обслуживания (СМО) включает три основных элемента:

- ***поток поступающих сообщений,***
- ***систему обслуживания,***
- ***характеристики качества и дисциплину обслуживания.***

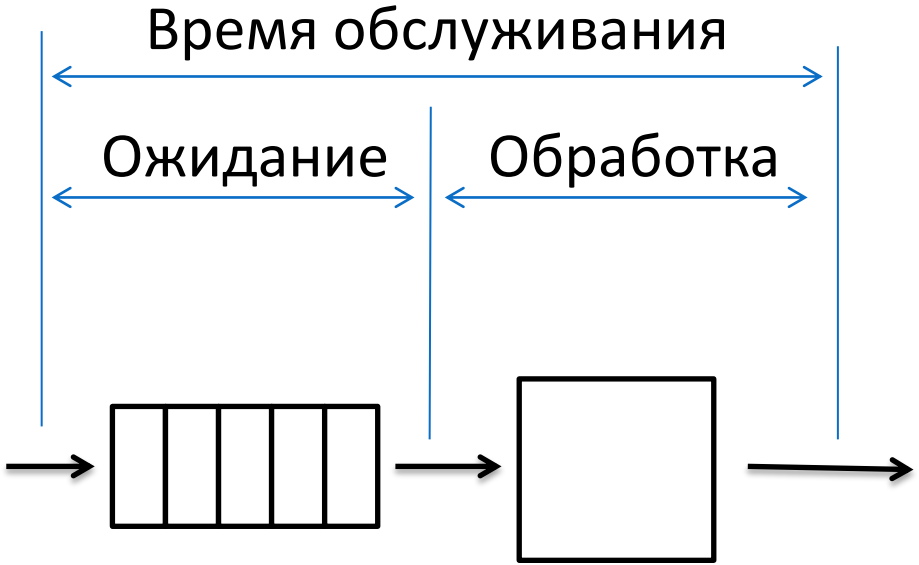
Системы массового обслуживания

(определение из учебника http://mathhelpplanet.com/static.php?p=sistema-massovogo-obsluzhivaniya)

Математическая модель системы массового обслуживания (СМО) включает три основных элемента:

- ***поток поступающих сообщений,***
  интенсивность, статистическое распределение
- ***систему обслуживания,***
  время обработки запроса
- ***характеристики качества и дисциплину обслуживания.***
  FIFO/LIFO/…, таймауты

СМО с очередью



Поток случайный событий

Нагрузка: **100 RPS**

Поток случайный событий

Нагрузка: **100 RPS**

```
long delayNS = TimeUnit.SECONDS.toNanos(1) / rate;
while(true) {
    callService();
    LockSupport.parkNanos(delayNS);
}
```

Поток случайный событий

Нагрузка: **100 RPS**

```
long delayNS = TimeUnit.SECONDS.toNanos(1) / rate;
while(true) {
    threadPool.execute(() -> callService());
    LockSupport.parkNanos(delayNS);
}
```

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

```
long delayNS = TimeUnit.SECONDS.toNanos(1) / rate;
while(true) {
    threadPool.execute(() -> callService());
    LockSupport.parkNanos(delayNS);
}
```

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

Поток случайный событий

Равномерный поток событий

Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

Поток случайный событий

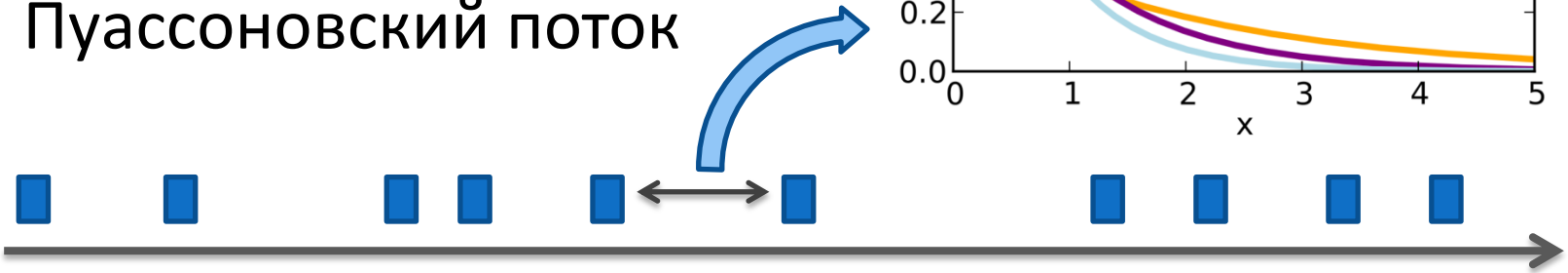
Равномерный поток событий

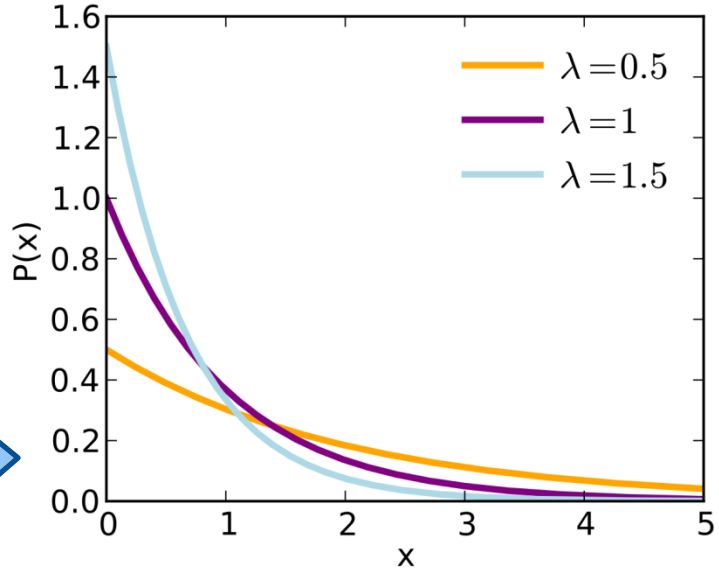
Нагрузка: **100 RPS**

Пуассоновский поток - *поток независимых событий*

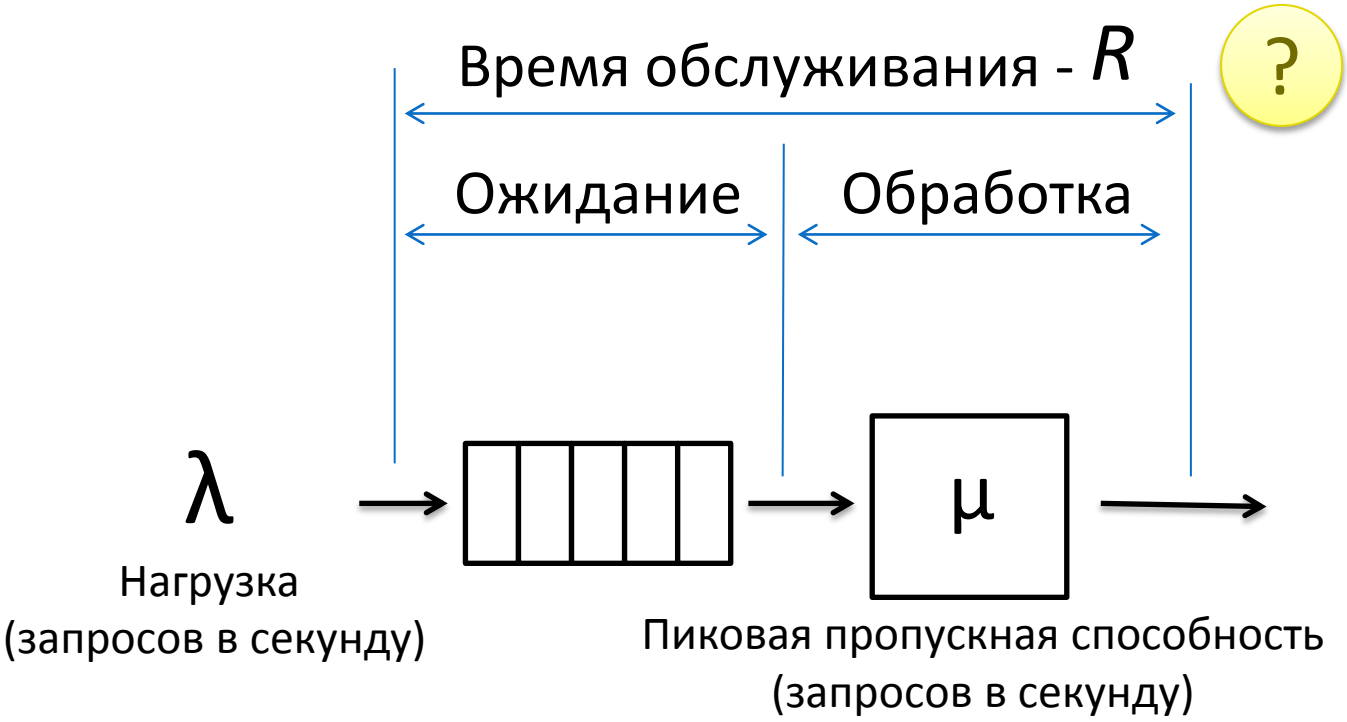
Поток случайный событий

Пуассоновский поток



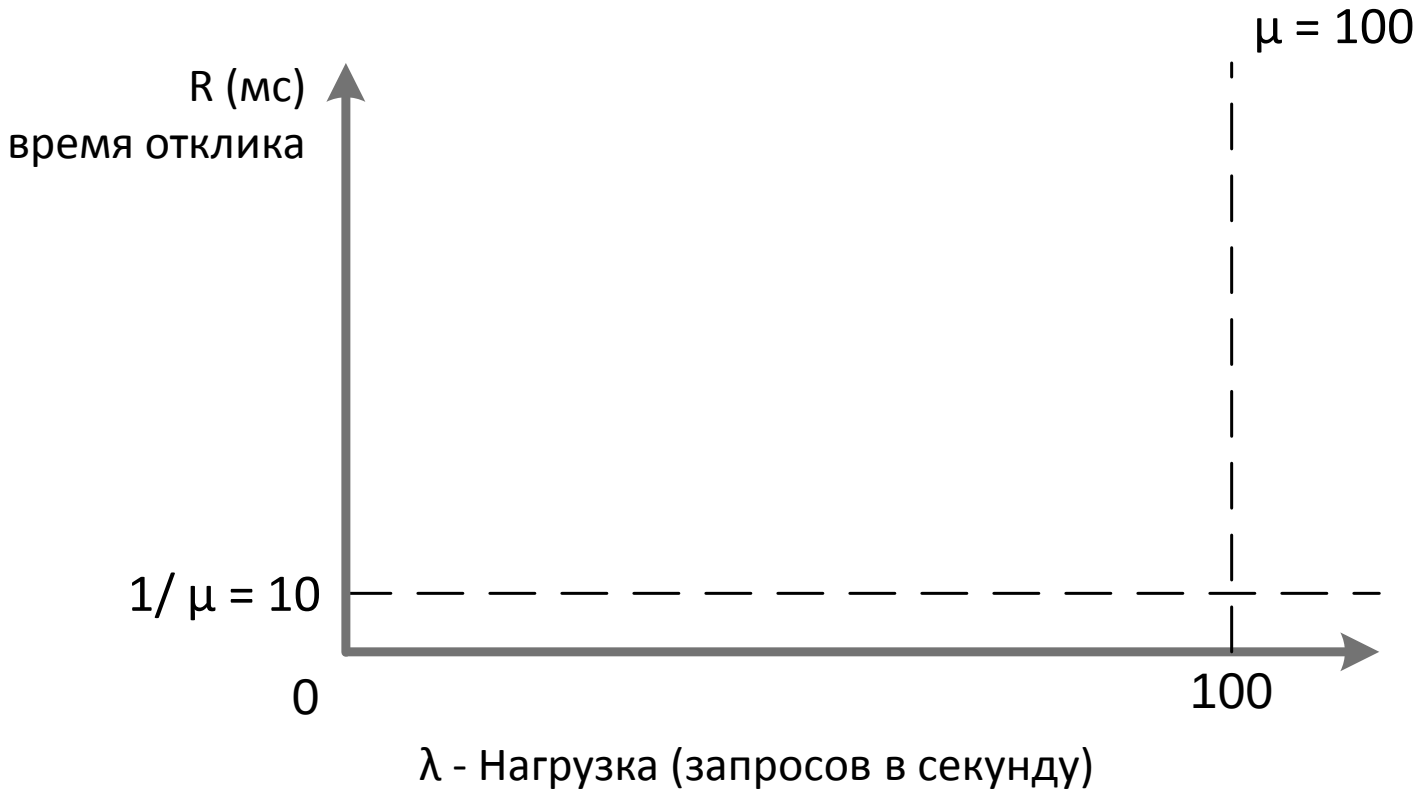


СМО с очередью

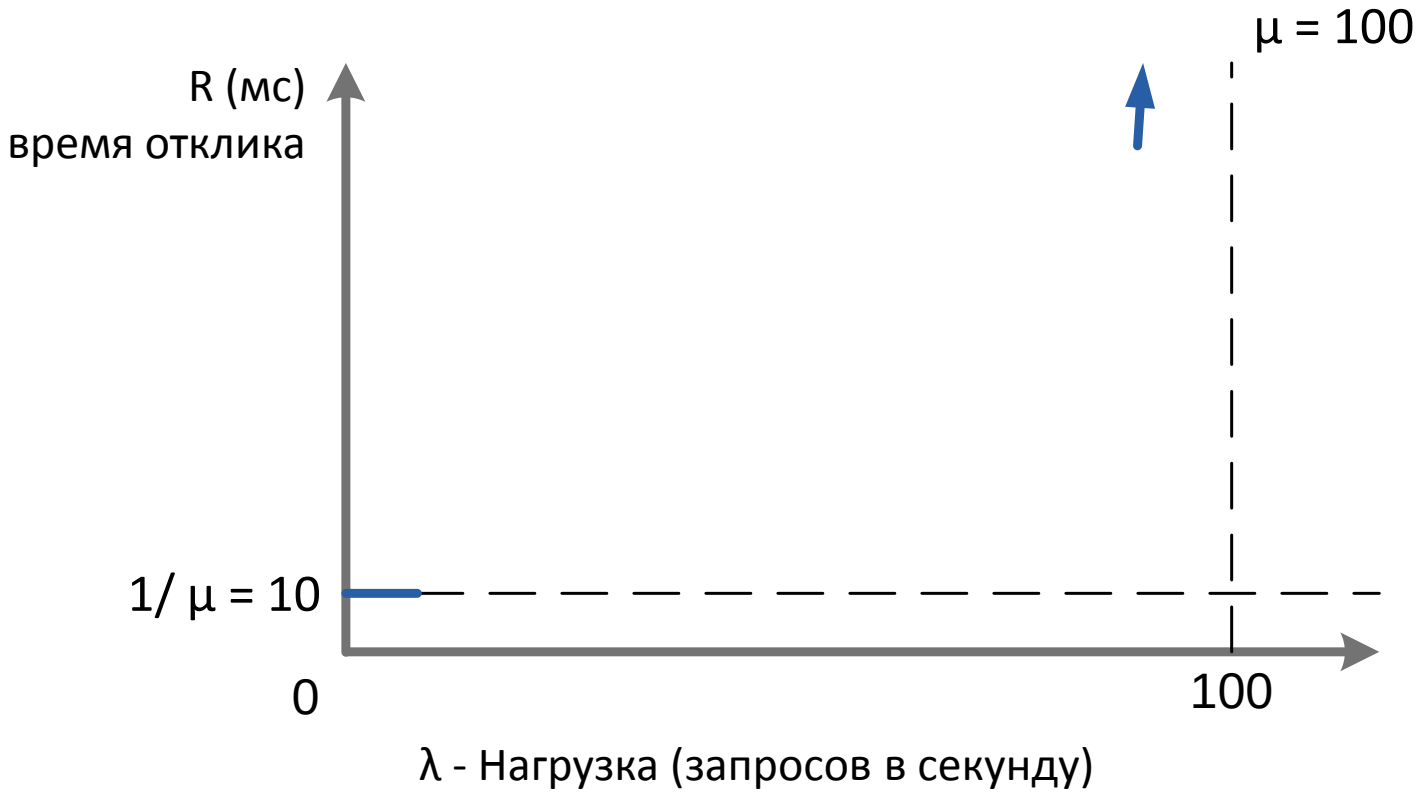


СМО с очередью

СМО с очередью

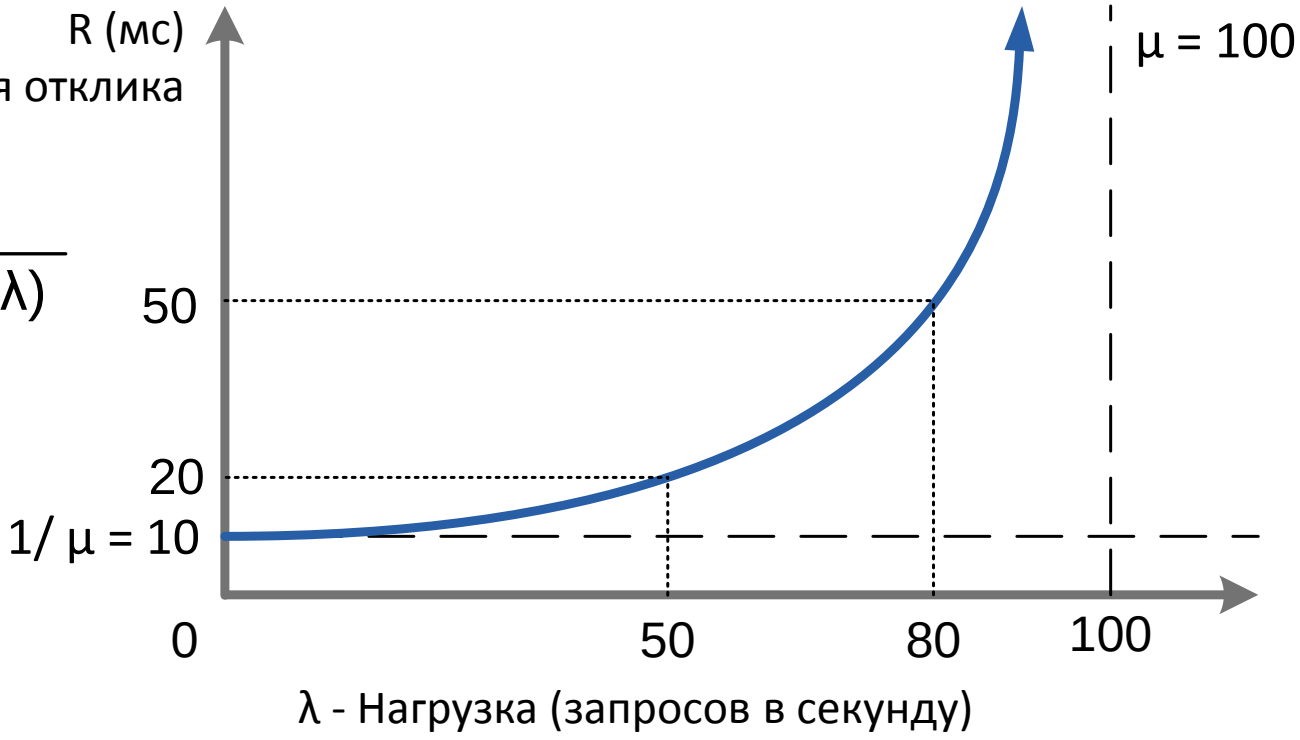


СМО с очередью

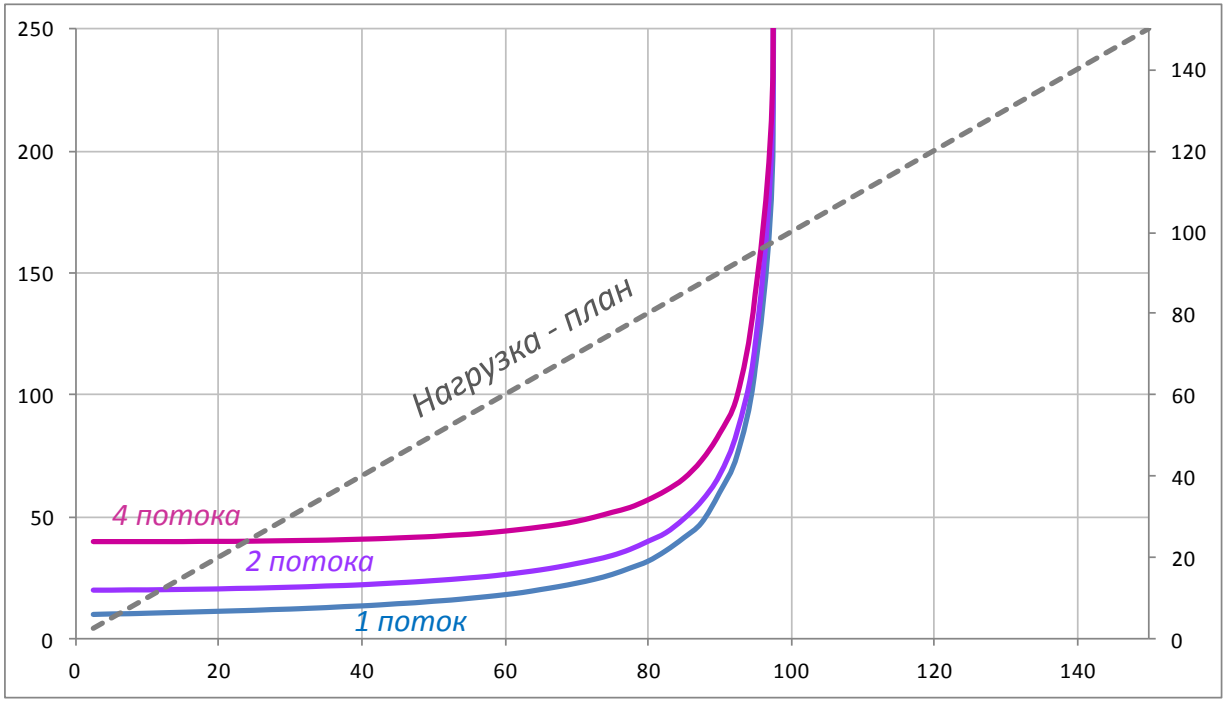


СМО с очередью

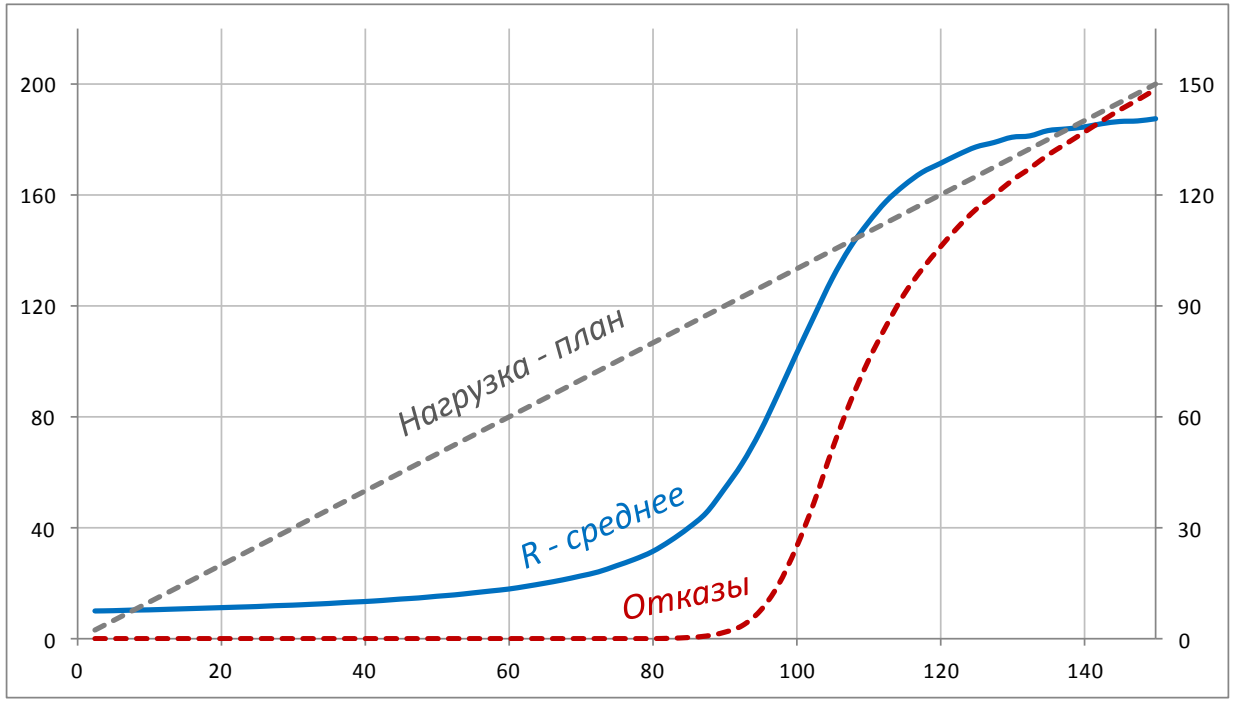
$$R = \frac{1}{(\mu - \lambda)}$$



СМО с очередью



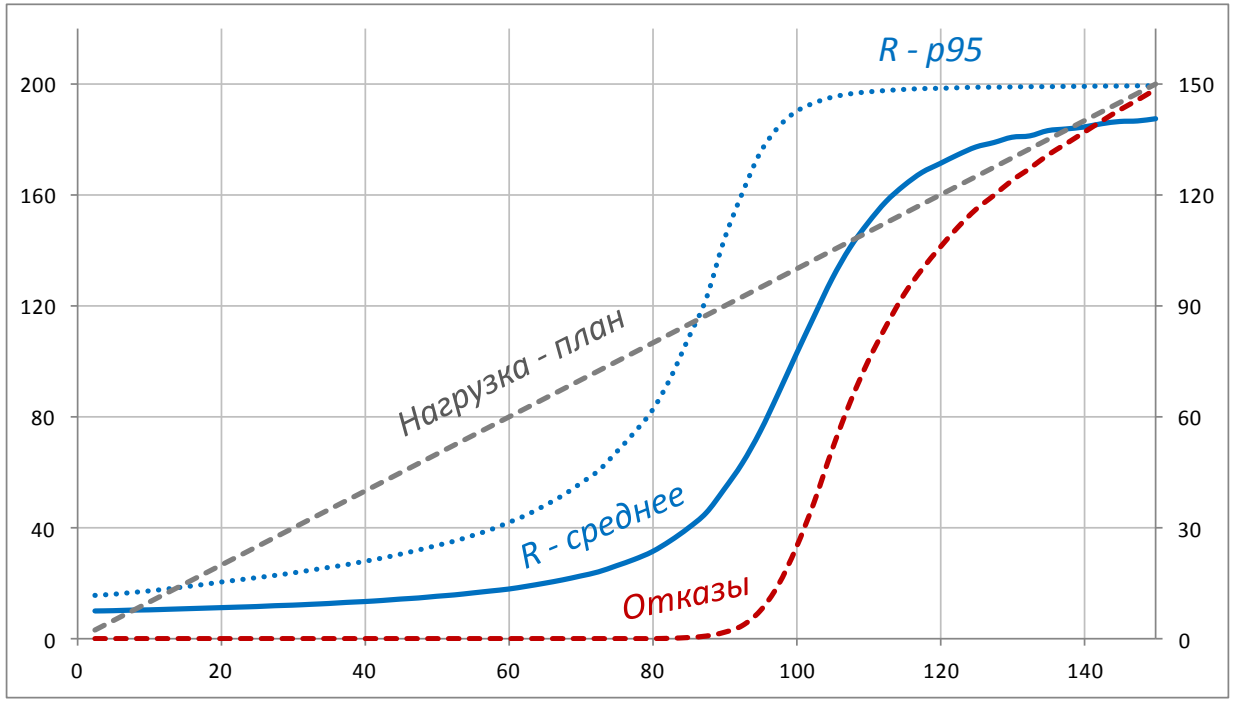
СМО с очередью



μ = 100

ТО = 200мс

СМО с очередью



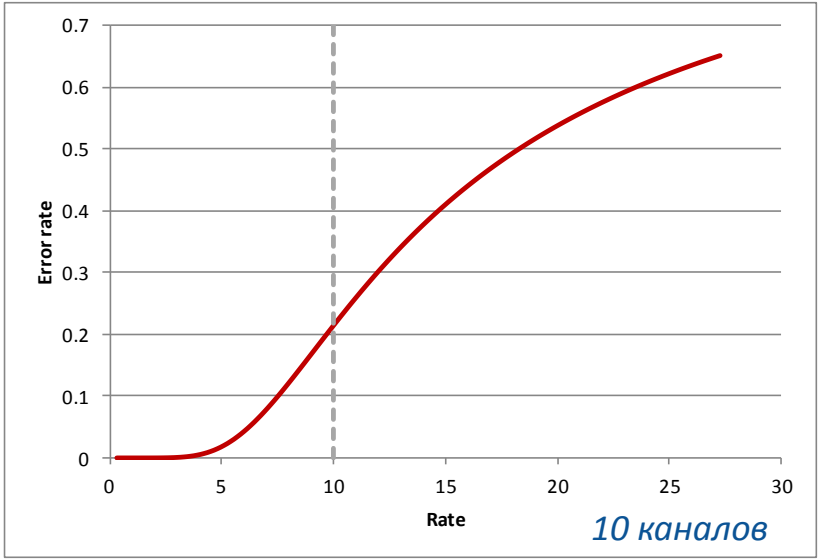
μ = 100

ТО = 200мс

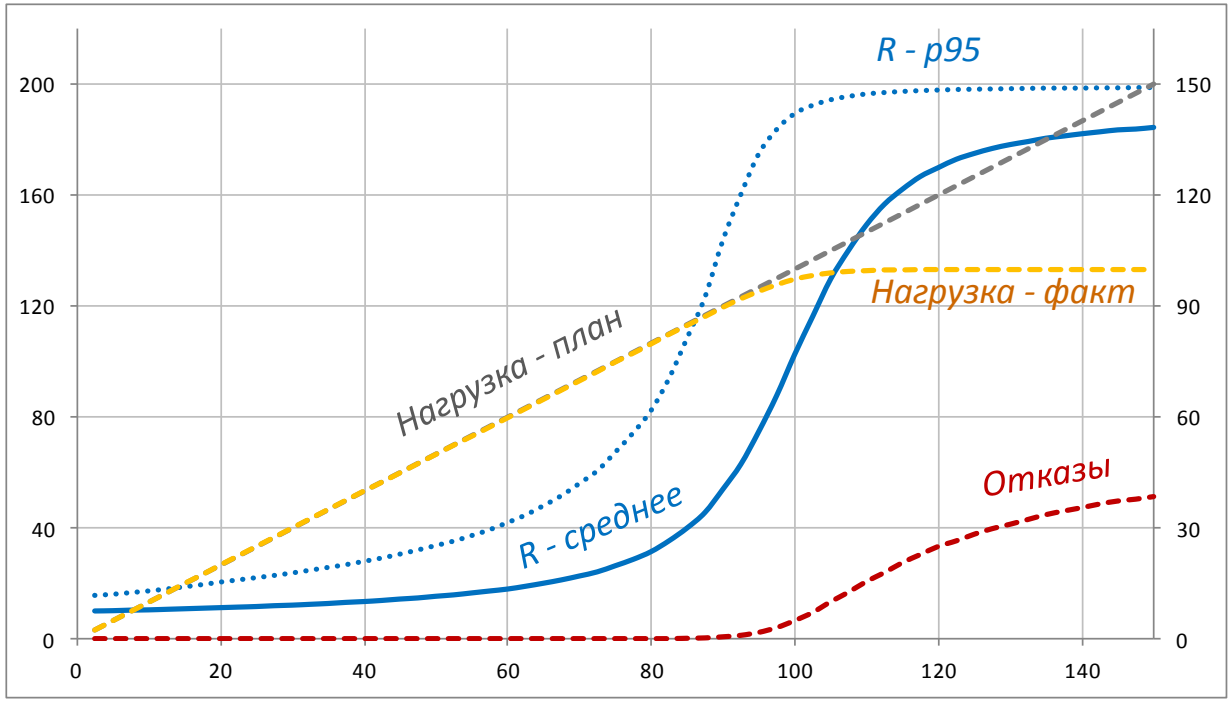
Формула Эрланга

Вероятность отказа (формула Эрланга),
для системы с *n* каналами обслуживания, без очереди.

$$\frac{\lambda}{\mu} = a \qquad P_{отк} = p_n = \frac{\frac{a^n}{n!}}{\sum_{k=0}^{n} \frac{a^k}{k!}}$$



СМО с очередью

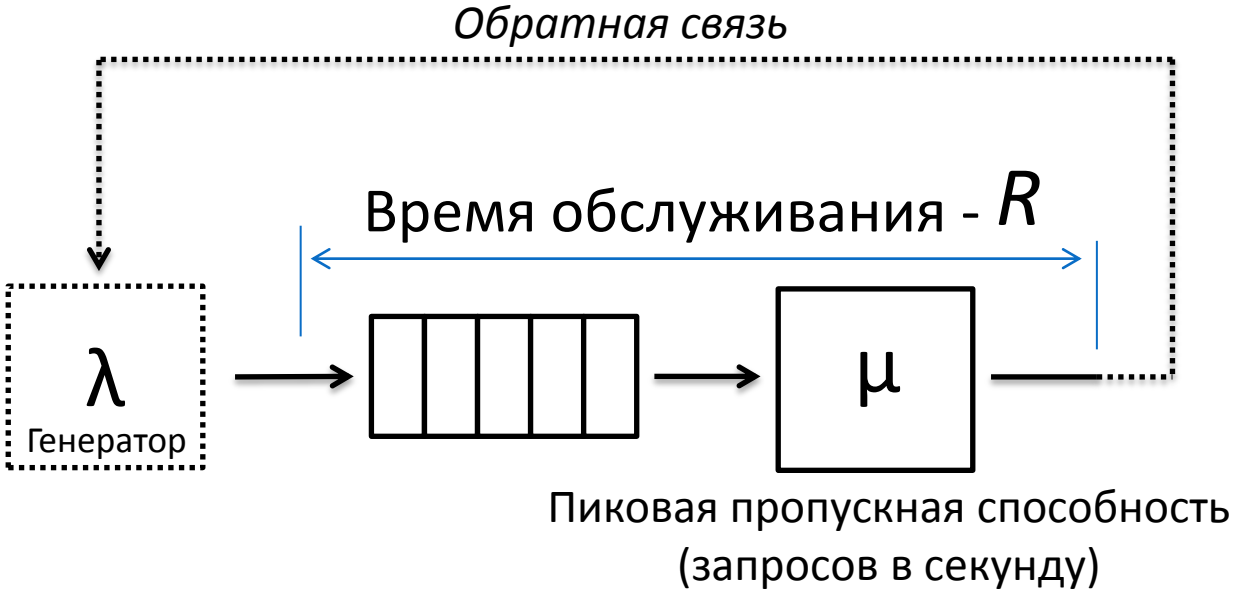


μ = 100

ТО = 200мс

Нагрузка
20 - потоков

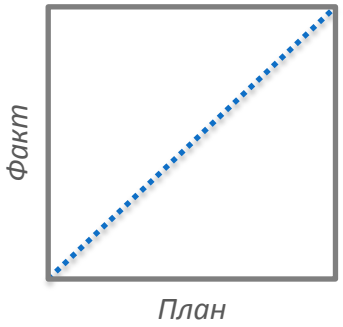
Закрытая модель нагрузки



Модели нагрузки

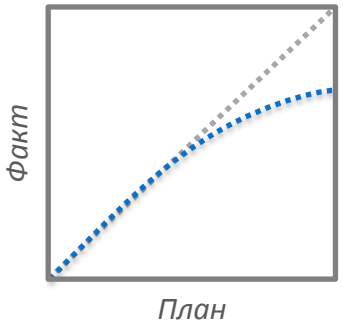
Открытая модель нагрузки

- Поток запросов не зависит от реакции системы

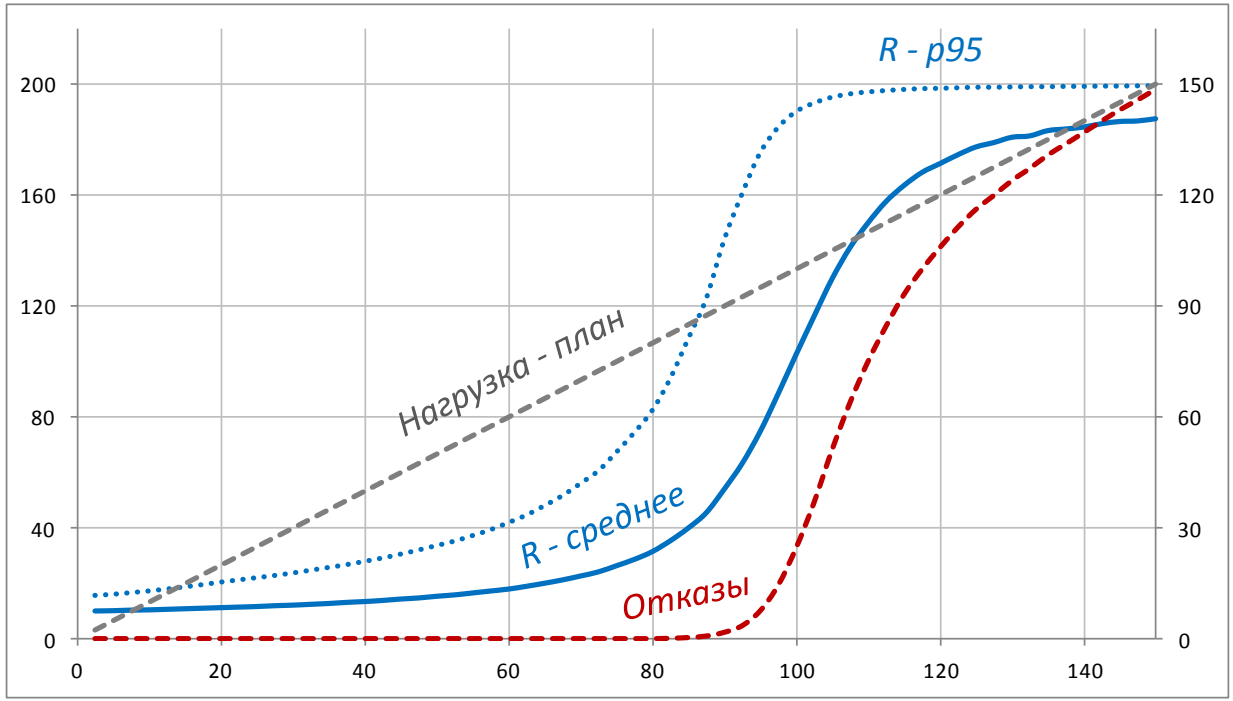


Закрытая модель нагрузки

- Поток запросов реагирует на поведение системы



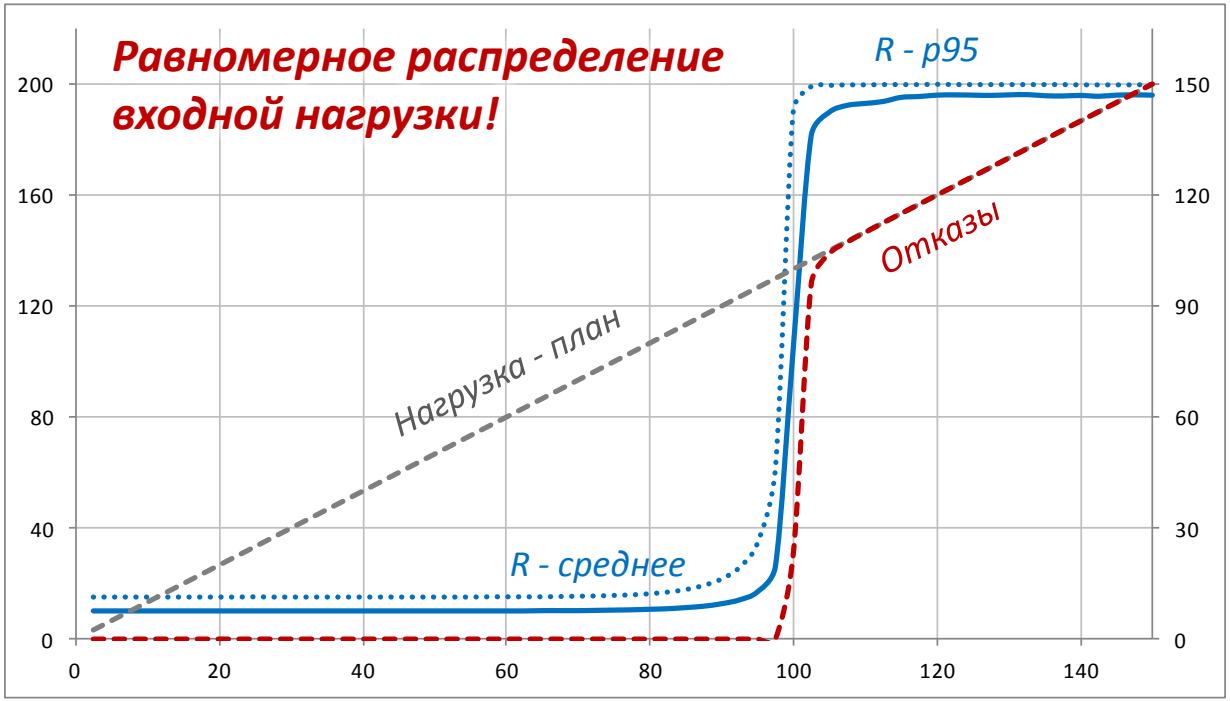
СМО с очередью



μ = 100

ТО = 200мс

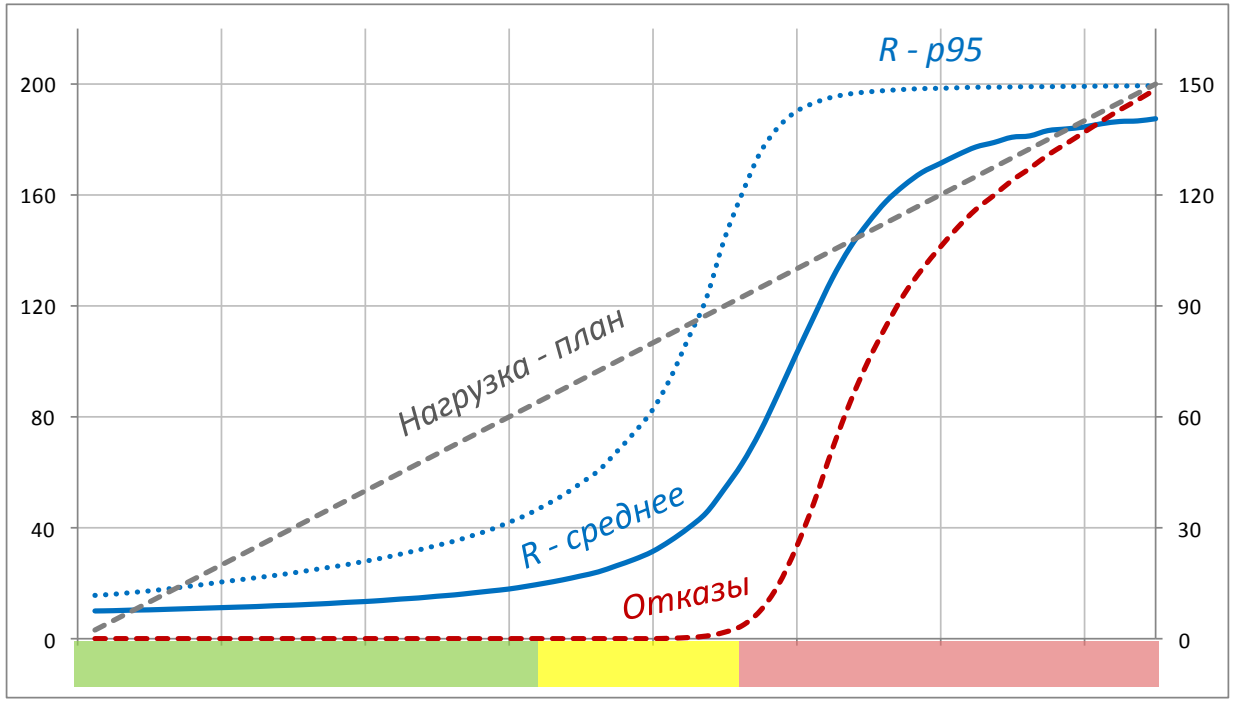
СМО с очередью



μ = 100

ТО = 200мс

СМО с очередью



μ = 100

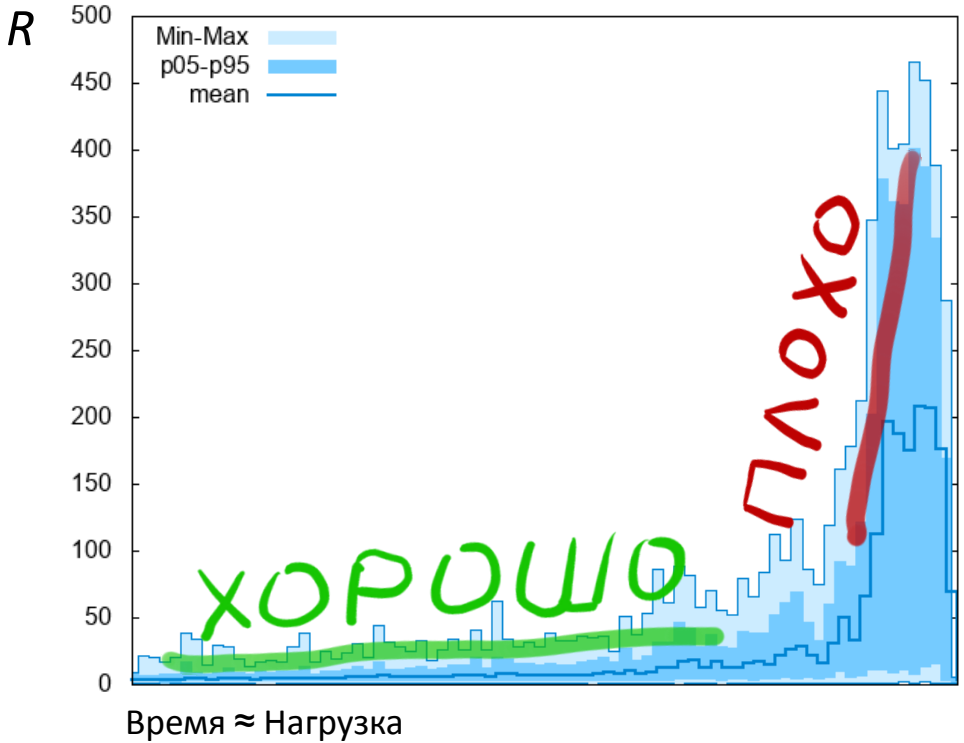
ТО = 200мс

Тестирование с линейной нагрузкой

Тестирование с линейной нагрузкой

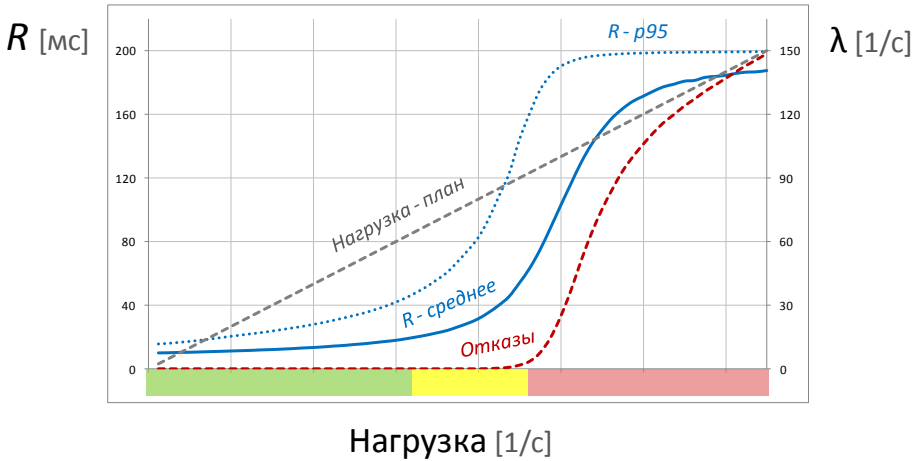
- Нагрузка линейно возрастает со временем
- Нарушение SLA по времени отклика –> **точка деградации**
- Нарушение SLA по ошибкам –> **предельно допустимая нагрузка**

Точка деградации



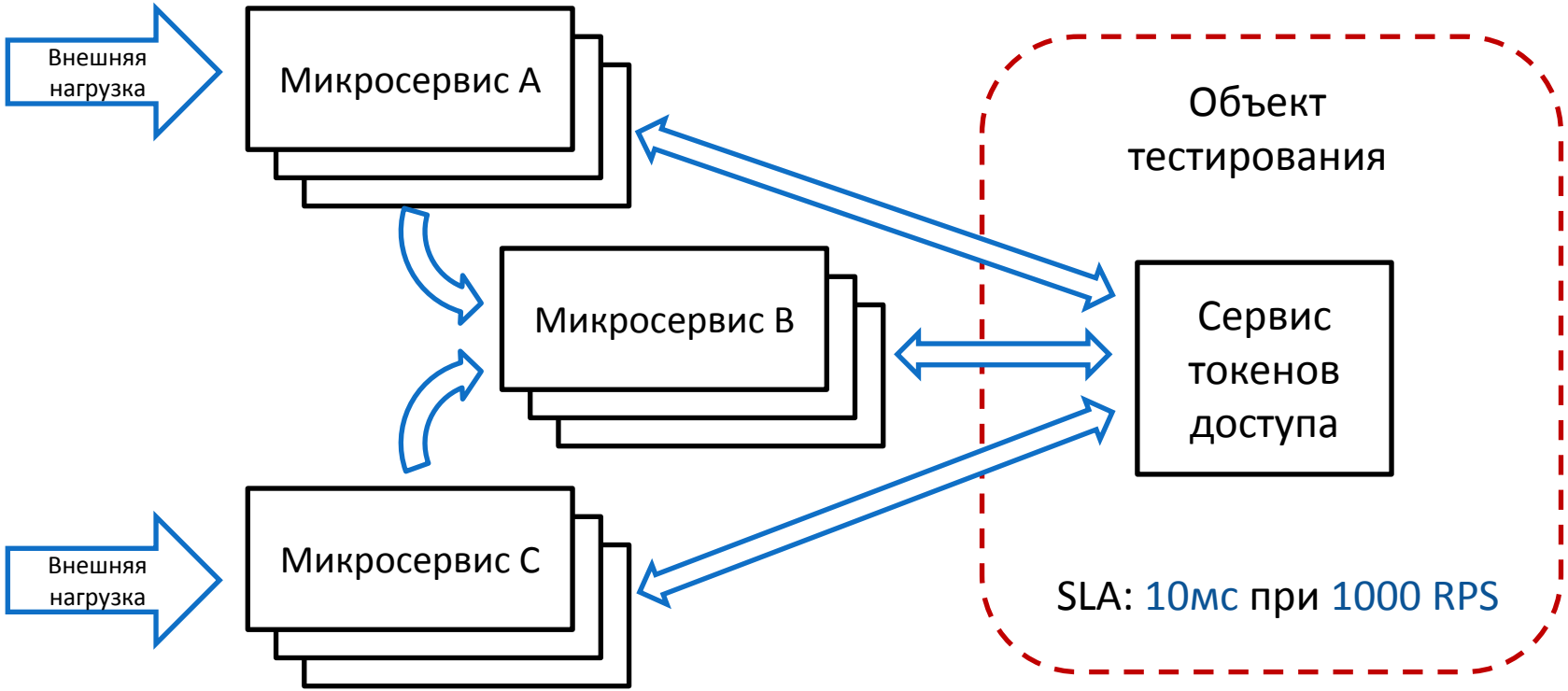
СМО с очередью - выводы

- СМО **не может работать на 100%** пропускной способности
- В зелёной зоне качество обслуживания определяется логикой обработки запроса
- В жёлтой зоне качество обслуживания определяется поведение очереди
- Изменение пропускной способности **непропорционально** влияет на время отклика



Реальная система
не сферический квадрат
в вакууме!

Пример системы



Насколько сферичен наш сервис?

Сервис токенов доступа

- “Идеальный” микросервес
- Выписывает токены на доступ между сервисами
- Интерфейс HTTP
- Основная нагрузка – криптографические преобразования
- Находится на критическом пути

Насколько сферичен наш сервис?

Сервис токенов доступа

- “Идеальный” микросервес
- Выписывает токены на доступ между сервисами
- Интерфейс HTTP
- Основная нагрузка – криптографические преобразования
- Находится на критическом пути

А где же тут очередь?

Где здесь очередь?

Ты видишь очередь? – А она есть.

- *Сетевые буфера*
- *Очередь соединений TCP стека*
- *Пул потоков HTTP обработки*
- *Очередь задач планировщика ЦПУ ядра*

Очереди повсюду!

Какой у нас тип нагрузки?

Исходные данные

- **2000 RPS** - максимальная тестовая нагрузка
- **0.5 сек** – таймаут на стороне клиента
- 2000 * 0.5 = **1000** – достаточное число потоков для гарантированного обеспечения уровня нагрузки

Какой у нас тип нагрузки?

Что у нас с политикой повторных запросов при ошибке?

Предположим

- **800 RPS** - текущая тестовая нагрузка
- **5%** – отказов по таймауту, которые будут повторены
- $\frac{800}{1 - 0.05}$ = **842 RPS** – реальная нагрузка на сервис

Возможно логику повторных запросов нужно включить в тест

Чем тестировать?

Взять готовый

LOCUST

APACHE JMeter™

k6

Grafana Labs

или свой велосипед?

- Вы знаете что делаете?
- Вы работаете с нестандартной нагрузкой

Gatling

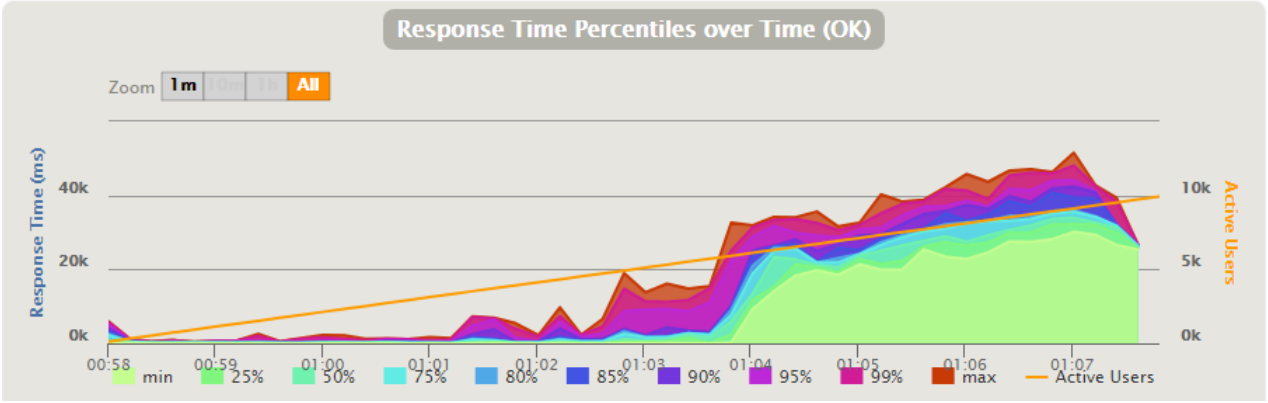
Гибкая модель нагрузки

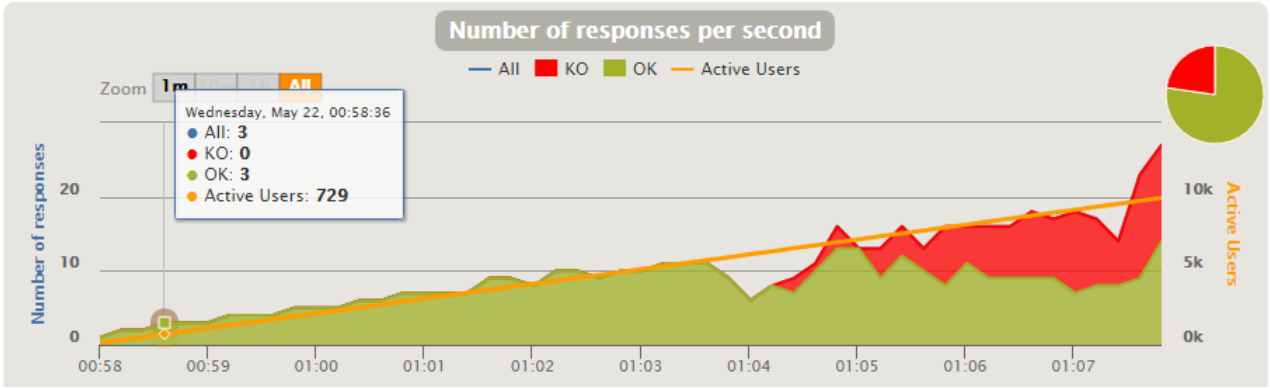
- открытая / закрытая
- фаза разогрева
- линейно возрастающая нагрузка
- произвольные профили нагрузки

Масштабируемость

Классные отчёты

Gatling


Response Time Percentiles over Time (OK)
Zoom
1m
All
Response Time (ms)
40k
20k
0k
10k
5k
0k
Active Users
00:58
00:59
01:00
01:01
01:02
01:03
01:04
01:05
01:06
01:07
min
25%
50%
75%
80%
85%
90%
95%
99%
max
Active Users


Number of responses per second
All
KO
OK
Active Users
Zoom
1m
All
Wednesday, May 22, 00:58:36
All: 3
KO: 0
OK: 3
Active Users: 729
Number of responses
20
10
0
10k
5k
0k
Active Users
00:58
00:59
01:00
01:01
01:02
01:03
01:04
01:05
01:06
01:07

Нагрузочное тестирование

Формулируем требования

- *Проектную нагрузку и SLA*

Нагрузочное тестирование

Формулируем требования

Определяем методику

Тип нагрузки, сценарии

Нагрузочное тестирование

Формулируем требования

Определяем методику

Выбираем инструмент

JMeter, Gatling, Locust, K6, ...

Нагрузочное тестирование

Формулируем требования

Определяем методику

Выбираем инструмент

Проводим тестирование

Нагрузочное тестирование

Формулируем требования

Определяем методику

Выбираем инструмент

Проводим тестирование

Собираем отчеты

- Не забываем включить **ошибки** и **реальную нагрузку** в отчёт

Нагрузочное тестирование

Формулируем требования

Определяем методику

Выбираем инструмент

Проводим тестирование

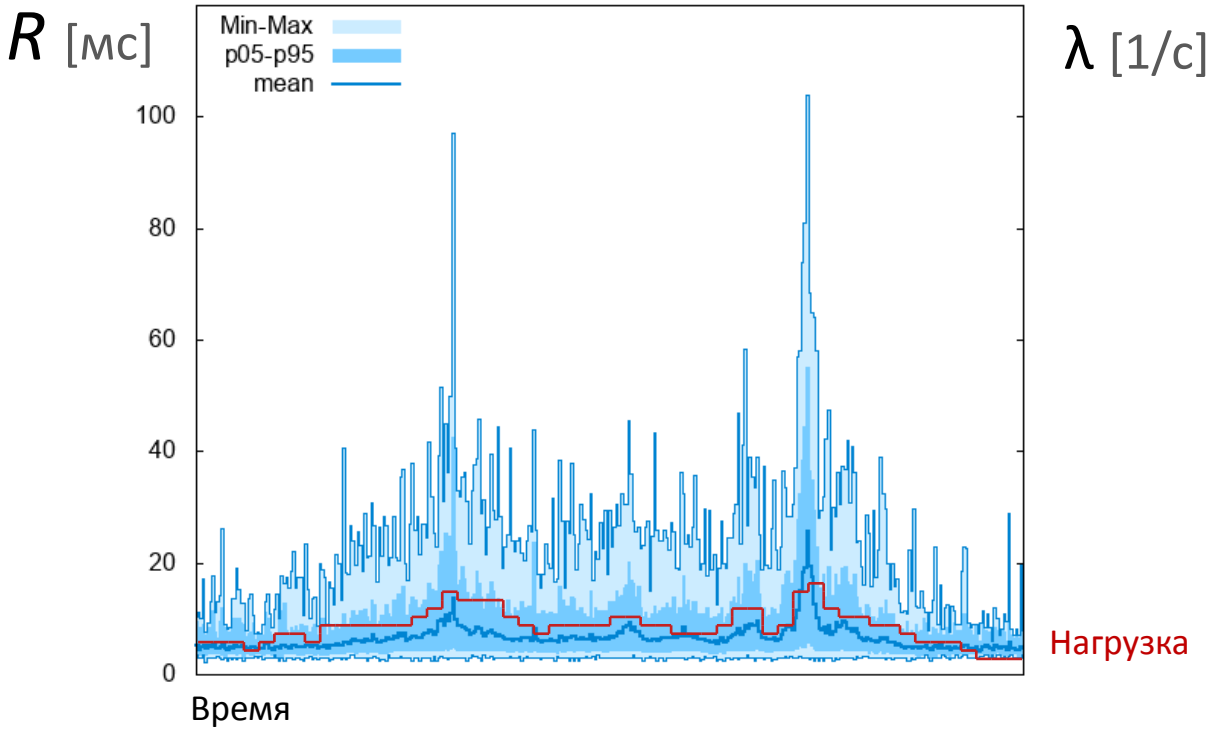
Собираем отчеты

- Не забываем включить **ошибки** и **реальную нагрузку** в отчёт

Можно ли сравнить результаты теста с боевой системой?

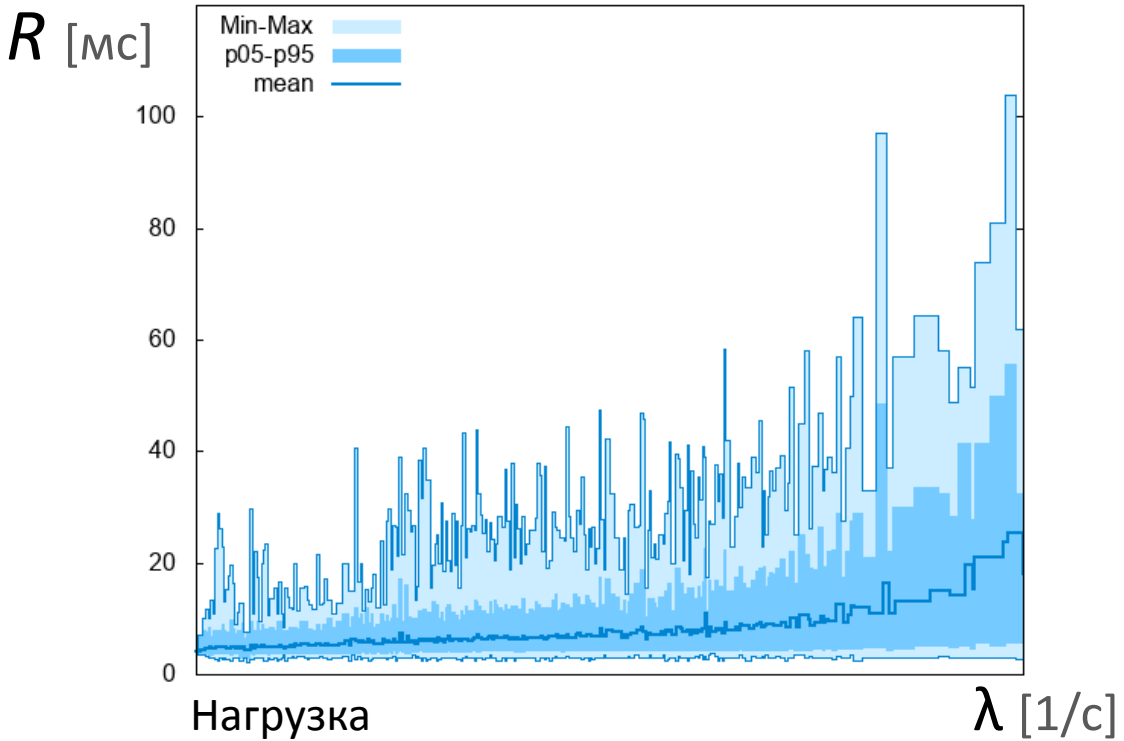
Сравнение с реальностью

Время отклика под реальной нагрузкой за сутки

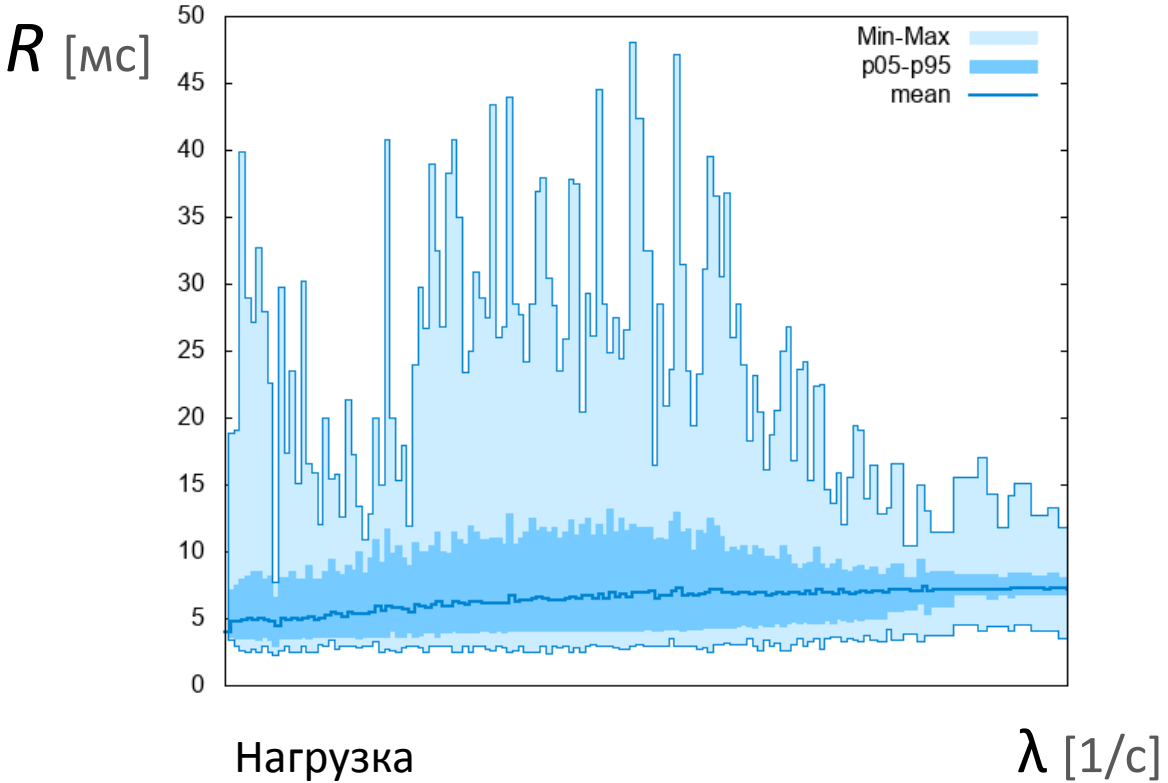


Сравнение с реальностью

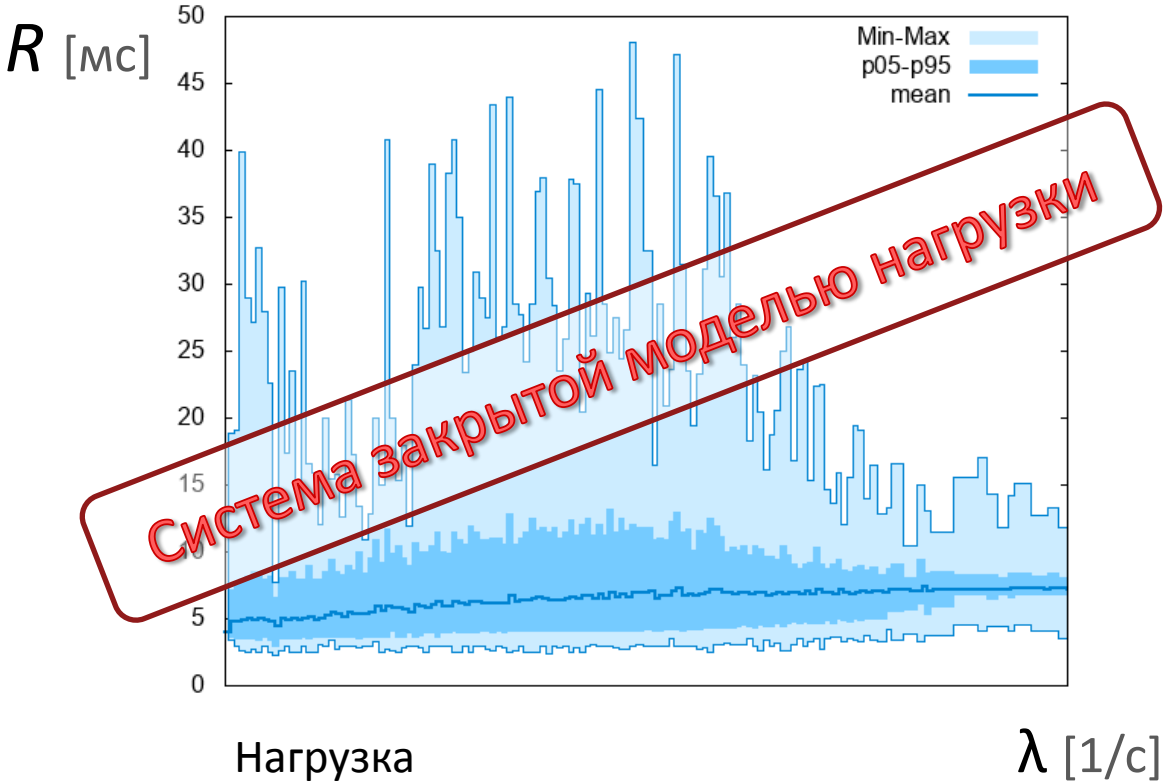
Время отклика от нагрузки по “боевым” данным



Аномальная картина



Аномальная картина



Системы массового обслуживания

(определение из учебника http://mathhelpplanet.com/static.php?p=sistema-massovogo-obsluzhivaniya)

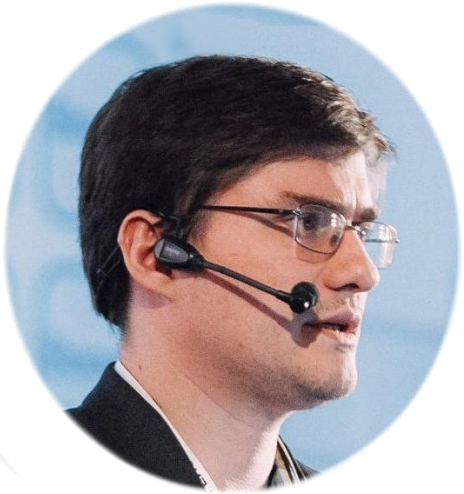
Математическая модель системы массового обслуживания (СМО) включает три основных элемента:

- ***поток поступающих сообщений,***
  интенсивность, статистическое распределение
- ***систему обслуживания,***
  время обработки запроса
- ***характеристики качества и дисциплину обслуживания.***
  FIFO/LIFO/…, таймауты

Заключение

- Системы массового обслуживания подчиняются фундаментальным законам
- Тестирование производительности – эмпирический процесс, его не заменить формулами
- Теория для выбора методики и контроля результата

О докладчике



Занимаюсь высоконагруженными системами на Java с 2006.
Разрабатывал софт для торговли на фондовых рынках, телекоме, e-commerce, RTB, здравоохранения.
Выступаю на конференциях, организую митапы, провожу тренинги.

Email: alexey.ragozin@gmail.com
Blog: blog.ragozin.info
Github: https://github.com/aragozin
Митапы и вебинары: https://aragozin.timepad.ru