

Семинар 1. Мониторинг и нагрузочное тестирование

Принципы построения высоконагруженных систем

Георгий Семенов

Институт прикладных компьютерных наук
Университет ИТМО

осень 2025

Домашние задания и оценивание

- Задание выдается на 2 недели с двумя дедлайнами:
 - **Мягкий дедлайн** – в рамках этого периода можно заранее отправить решение и получить обратную связь, чтобы исправить замечания до наступления жесткого дедлайна.
 - **Жесткий дедлайн** – после этого новые посылки работы не принимаются, сдать работу больше нельзя.
- Планируется выдать 4 домашних задания, каждое из которых стоит ± 12 баллов. Правила выставления оценки за курс следующие:
 - **A – «5»** – $\geq 90\%$ от общей суммы обязательных баллов (предв. ≥ 43.2)
 - **B/C – «4»** – $\geq 75\%$ от общей суммы обязательных баллов (предв. ≥ 36)
 - **D/E – «3»** – $\geq 60\%$ от общей суммы обязательных баллов (предв. ≥ 28.8)
 - **F – «2»** – $< 60\%$ от общей суммы обязательных баллов (предв. < 28.8)

Как отправлять домашние задания?

- Выполненное задание рекомендуется оформить в \LaTeX (например, в Overleaf) и выслать файлом с именем вида DDIA25-HW1-IvanIvanov.pdf на почту georgii.v.semenov@mail.ru.
 - Пожалуйста, оформляйте ответы на задания в блоке Решение.

Содержание

- 1 Мониторинг
- 2 Временные ряды
- 3 Prometheus и PromQL
- 4 Демонстрация: Grafana
- 5 Нагрузочное тестирование
- 6 Демонстрация: wrk
- 7 Итоги

Мотивация

- **Backend** – сеть геораспределенных черных ящиков

Мотивация

- **Backend** – сеть геораспределенных черных ящиков
- Ящики расположены в разных data-центрах, у них разные нагрузки и поведение

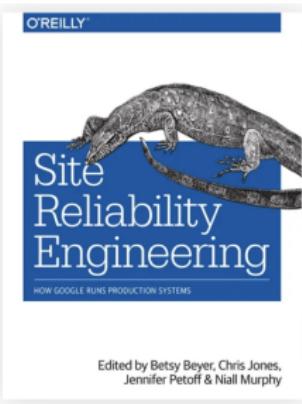
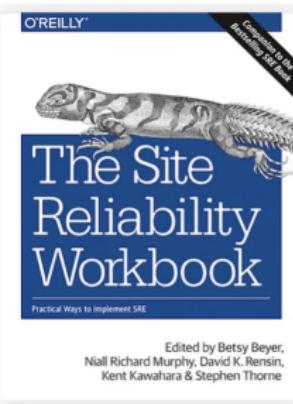
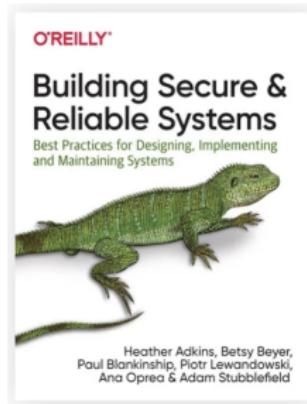
Мотивация

- **Backend** – сеть геораспределенных черных ящиков
- Ящики расположены в разных дата-центрах, у них разные нагрузки и поведение
- Хотим **observability** – представление о том, как дела у ящиков

Мотивация

- **Backend** – сеть геораспределенных черных ящиков
- Ящики расположены в разных дата-центрах, у них разные нагрузки и поведение
- Хотим **observability** – представление о том, как дела у ящиков
- С ней можем обеспечивать и оценивать **reliability**

SRE – Software Reliability Engineering



- SRE¹ как дисциплина определяет подходы к **reliability** и **observability** систем.
- SLA (Service Level Agreement) – гарантии и последствия их нарушения
- SLO (Service Level Objective) – метрики и целевые требования к ним
- SLI (Service Level Indicator) – измеряемые метрики

¹Ссылки на книги: <https://sre.google/books>

Что такое мониторинг?

- Собираем показатели с каждого пода каждого узла системы

Что такое мониторинг?

- Собираем показатели с каждого пода каждого узла системы
- Строим временные ряды по этим показателям

Что такое мониторинг?

- Собираем показатели с каждого пода каждого узла системы
- Строим временные ряды по этим показателям
- Оцениваем показатели SLI, стремимся обеспечить SLO, соблюдаем SLA

Что такое мониторинг?

- Собираем показатели с каждого пода каждого узла системы
- Строим временные ряды по этим показателям
- Оцениваем показатели SLI, стремимся обеспечить SLO, соблюдаем SLA
- Настраиваем **alerts**, **tickets**, **logging** и реагируем на инциденты дежурной сменой

Метрики системы: ресурсы

- **CPU usage:** утилизация, параллелизм, троттлинг...
- **Memory usage:** утилизация по типам памяти, подкачка...
- **Disk I/O:** объем дисков, рейт чтения/записи, буферы, задержки...
- **Network I/O:** рейт чтения/записи, задержки пакетов TCP/UDP...
- **GC, Heap Memory** – параметры для сред Java, Go и т.п.
- **Queues** – параметры очередей задачий
- **Connections** – количество открытых соединений
- **Threads** – количество активных процессов, потоков

Метрики системы: запросы

- QPS, или Throughput (requests per second)
- Timing, или Latency (response time)
- Error rate – RPS ошибок
- Uptime
- Downtime
- Availability SLO: Error Budget, Uptime

Метрики системы: бизнес-метрики

- Активность пользователей: Time Spent, Active Users (MAU/WAU/DAU/PAU)
- Конверсия: конвертируемость пользователей вдоль CJM
- Транзакции: успешно завершенные пользовательские сценарии
- Доход: выручка (ARPU, ARPPU), юниты

Базовый минимум для мониторинга системы:

- **Latency** – семейство измерений времени отклика
- **Traffic** – семейство измерений входящего трафика
- **Errors** – семейство измерений ошибок
- **Saturation** – семейство измерений загрузки системы

Содержание

- 1 Мониторинг
- 2 Временные ряды
- 3 Prometheus и PromQL
- 4 Демонстрация: Grafana
- 5 Нагрузочное тестирование
- 6 Демонстрация: wrk
- 7 Итоги

Понятие метрики

- Метрика — набор: timestamp, value, labels.
- У каждой метрики есть служебный label `__name__`
- Пример метрики:

```
node_manager_node_up{node="n1"}
```

- То же самое, но без синтаксического сахара:

```
{__name__="node_manager_node_up", node="n1"}
```

Типы метрик (в Prometheus)

- **Counter** — монотонно растёт (например, суммарное количество запросов).
- **Gauge** — текущее значение, оно может как увеличиваться, так и уменьшаться.
- **Histogram** — распределение величины по *buckets* (счетчики по бакетам).
- **Summary** — похож на histogram, даёт квантильные оценки, но с искажениями при агрегации.

PromQL: типы значений

- **Instant Vector** — множество временных рядов в конкретный момент (набор пар {labels, value}).
- **Range Vector** — для каждой временной серии набор samples за интервал (для функций over_time, rate, increase и т.д.).
- **Scalar** — единичное числовое значение (результат вычисления). Иногда используется в alert expressions.

Содержание

- 1 Мониторинг
- 2 Временные ряды
- 3 Prometheus и PromQL
- 4 Демонстрация: Grafana
- 5 Нагрузочное тестирование
- 6 Демонстрация: wrk
- 7 Итоги

СУБД для временных рядов

- Оптимизированы для данных вида $timestamp \rightarrow value$.
- Высокая скорость записи (до миллионов точек в секунду).
- Эффективная компрессия (TSDB, columnar storage).
- Поддержка downsampling (сужение), retention (TTL), агрегирования.
- Пример: Prometheus TSDB, VictoriaMetrics, M3, InfluxDB.

- Open-source TSDB и система мониторинга.
- Pull-модель: опрашивает endpoints (`/metrics`).
- Язык запросов: PromQL.
- **Job → Instance → Metric → Labels.**
- Хранение локально, без распределённости (по дизайну).
- Экспорт данных в long-term: Thanos / VictoriaMetrics.

Примеры Node Exporter

- CPU: `node_cpu_seconds_total{mode="user"}`
- Memory: `node_memory_MemAvailable_bytes`,
`node_memory_MemFree_bytes`.
- Disk: `node_filesystem_avail_bytes`.
- Network: `node_network_receive_bytes_total`.
- Load (avg for mins): `node_load1`, `node_load5`, `node_load15`.
- Filesystem saturation: `node_disk_io_time_seconds_total`.

Пример запроса: `rate(node_network_receive_bytes_total[5m])`

Link

- Совместима с Prometheus.
- Язык запросов: MetricsQL (обратно совместим с PromQL).
- Более эффективное хранение: до 7–10x компрессия.
- Дает горизонтальное масштабирование (кластерная версия).
- Может заменять Prometheus или быть хранилищем “за ним”.

- Универсальная система визуализации данных.
- Поддерживает множество источников: Prometheus, VM, PostgreSQL, ElasticSearch, CSV...
- Дашборды: графики, таблицы, heatmap, alert panels.
- Поддержка переменных, шаблонов, drilldown.
- Экспорт/импорт дашбордов JSON.

Алертинг в Prometheus

- Два компонента:
 - **Prometheus alerting rules** — проверка условий (alert:).
 - **Alertmanager** — маршрутизация уведомлений.
- Каналы: Email, Slack, PagerDuty, Telegram, Webhook.
- Поддерживает инциденты, группировку, подавление, тишину (silences).
- Пример правила:

```
alert: HighCpuUsage
expr: avg(rate(node_cpu_seconds_total{mode!="idle"}[5m])) > 0.8
for: 10m
labels: { severity="warning" }
annotations: {
    summary="High CPU usage on {{ $labels.instance }}"
}
```

Содержание

- 1 Мониторинг
- 2 Временные ряды
- 3 Prometheus и PromQL
- 4 Демонстрация: Grafana
- 5 Нагрузочное тестирование
- 6 Демонстрация: wrk
- 7 Итоги

Куда посмотреть?

- grafana/docker-compose.yml
- grafana/prometheus.yml
- docker-compose up
- http://localhost:3000
- http://localhost:3000/connections/datasources/new
- http://victriametrics:8428
- http://localhost:8428/targets

Содержание

- 1 Мониторинг
- 2 Временные ряды
- 3 Prometheus и PromQL
- 4 Демонстрация: Grafana
- 5 Нагрузочное тестирование
- 6 Демонстрация: wrk
- 7 Итоги

Мотивация

- Обеспечение стабильной работы системы под высокой нагрузкой
- Выявление узких мест производительности до выхода в прод
- Оценка поведения при пиковых значениях запросов
- Проверка корректности масштабирования
- Подтверждение соблюдения SLA/SLO
- Предотвращение деградаций и простоев

Некоторые идеи

- **Load Testing** — проверка работы при ожидаемой нагрузке
- **Stress Testing** — определение предельных возможностей
- **Spike Testing** — реакция на резкие скачки трафика

Содержание

- 1 Мониторинг
- 2 Временные ряды
- 3 Prometheus и PromQL
- 4 Демонстрация: Grafana
- 5 Нагрузочное тестирование
- 6 Демонстрация: wrk
- 7 Итоги

Подготовка

- На MacOS: brew install wrk
- На Ubuntu: sudo apt-get install wrk
- wrk -t4 -c50 -d20s http://localhost:8081/

Содержание

- 1 Мониторинг
- 2 Временные ряды
- 3 Prometheus и PromQL
- 4 Демонстрация: Grafana
- 5 Нагрузочное тестирование
- 6 Демонстрация: wrk
- 7 Итоги

Итоги

- Рассмотрели мониторинг и нагрузочное тестирование
- Рассмотрели примеры в Grafana и с wrk
- Выдано первое домашнее задание