OTUS

Highload Architect

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Проблемы высоких нагрузок

Преподаватель

Никита Сапогов

Telegram: @AmsTaFFix

Руководитель курса “Архитектор высоких нагрузок”

Руководитель мониторинга в OZON

5 лет развивал MSA (Go, Tarantool, PostgreSQL)

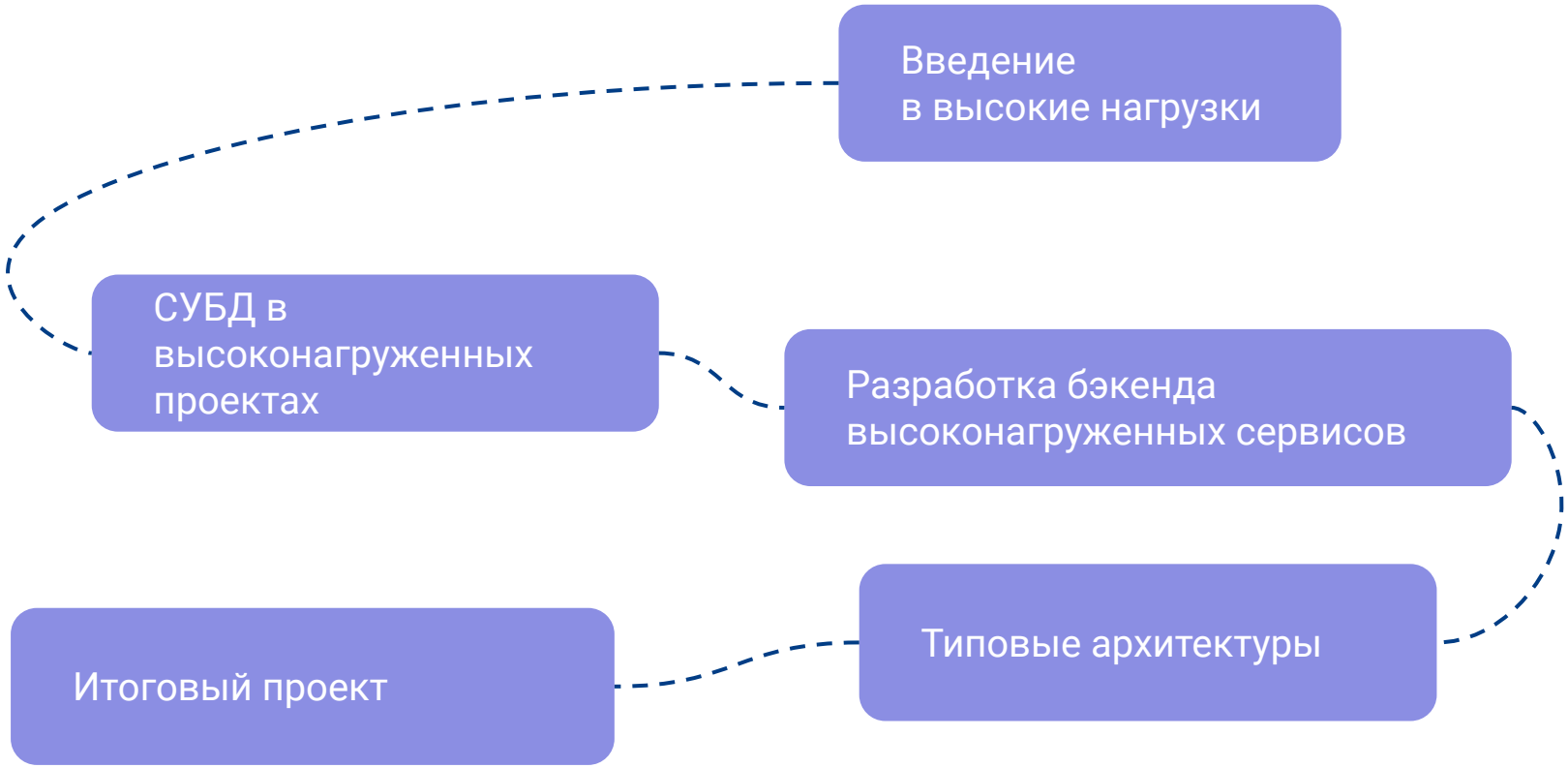
Ментор, консультант на других площадках

Результат от курса

1) **Что получите от курса**
   - создавать веб-приложения, которые легко масштабировать
   - обеспечивать отказоустойчивость веб-приложений даже при падении серверов
   - правильно использовать шаблоны (кеширование, реплицирование, шардирование, индексирование)

2) **Что создадите в качестве проекта**
   - Проект собранный на основе домашних заданий
   - Проект в виде проектной работы

Карта курса


Введение
в высокие нагрузки
СУБД в
высоконагруженных
проектах
Разработка бэкенда
высоконагруженных сервисов
Типовые архитектуры
Итоговый проект

Как будем учиться?

Вебинары

Вторник, четверг, 20:00.
(запись и материалы выкладывают, как правило, на следующий день после вебинара)

Домашние задания

1 дз в неделю.
Типовой срок проверки: 2-3 дня

Чат в telegram

Задавайте вопросы, обменивайтесь инсайтами.

Лайфхаки

Сделайте упор на тему, которая вам важна

Старайтесь полученные знания применять на практике

Задавайте вопросы, так материал лучше усваивается

Регулярное выполняйте ДЗ (наверстать пропуски тяжело)

Мотивация

Приготовьтесь к волнообразной реакции

от «ух ты как интересно», до «куда я попал?»

* в такие моменты вспоминайте
про цели обучения, которые вас мотивируют)

Это нормально :)

Отзывы

Вебинары во многом адаптивны.
Мы следим за результатами опросов, обсуждениями в telegram’е и ДЗ.

Всегда рады вашим
конструктивным отзывам :)

Расскажите о себе

Напишите, пожалуйста, в чат или скажите голосом

Как вас зовут?

Какой опыт в IT?

Какие ожидания
от курса?

Заполните информацию
в разделе «О себе»
в личном кабинете

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Telegram

Задаем вопрос
в чат

Вопросы в чате вижу, но
могу ответить не сразу

Есть ли вопросы?

Маршрут вебинара

- Знакомство
- Особенности работы ОС
- Модели веб-сервисов
- Возможности языковых платформ
- Трехзвенная архитектура
- Рефлексия

Цели вебинара

После занятия вы будете

1. знать основы работы операционных систем, существенные для высоких нагрузок;
2. знать принципы работы, преимущества и недостатки различных моделей веб-серверов;
3. проанализировать причину возникновения и принципы работы концепций fibers/green threads/goroutines;
4. проанализировать возможности языковых платформ с точки зрения поддержки асинхронного программирования.

Смысл

Зачем вам это уметь

1. определять причину торможения
2. проектировать приложения с высокой пропускной способностью

Особенности работы операционной системы

Стандартный сценарий

Дано: имеется компьютер с одним 4-х ядерным процессором, на котором запущены: chrome, skype, zoom, telegram, slack, …

Как это вяжется с представлением о том, что на одном ядре выполняется 1 программа?

Пишем в чат

Отвечаем устно

Передача управления

- Операционная система выделяет для выполнения программы квант времени (~50 мкс), затем происходит прерывание и переключение контекста.
- Переключение контекста происходит и при любом системном вызове.

Переключение контекста (context switch)

Сохранение

- Регистров и процессов
- Общей информации: pid, tid, uid, gid, euid, egid, …
- Состояния процесса/потока
- Прав доступа
- Используемых потоком ресурсов и блокировок
- Счетчиков использованных ресурсов (например, таймеров процессорного времени)
- Регионов памяти, выделенных процессу
- и т.п.

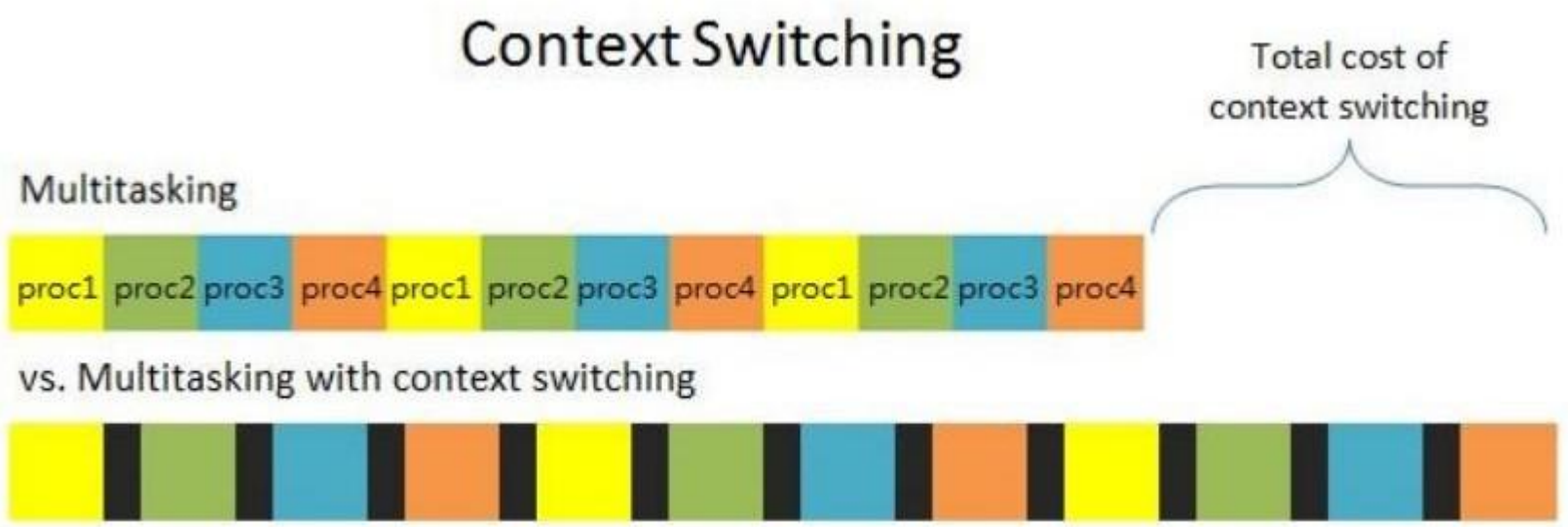
Переключение контекста (context switch)

Очистка

- Конвейера команд и данных процессора
- TLB, отвечающего за постраничное отображение линейных адресов на физические

Вывод: переключение контекста – дорогая операция (~1 мкс (10^{-6}с), тогда как выполнение простой строчки кода занимает ~0.5 нс (10^{-9}с))

Переключение контекста (context switch)



Опишите, что будет если увеличить кол-во процессов?

Пишем в чат

Отвечаем устно

Есть ли вопросы?

Процессы и потоки

- В Linux практически одно и тоже
- Потоки имеют общую память

Вопрос

Какими системными вызовами создаются потоки и процессы?

Пишем в чат

Отвечаем устно

Процессы и потоки

```
clone(child_stack=0,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fa001a93a10) = 6916 [fork()]

clone(child_stack=0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SYSVSEM|CLONE
_SIGHAND|CLONE_THREAD|CLONE_SETTLS|CLONE_PARENT_S
child_tidptr=0x7fa001a93a10) = 6916 [pthread_create()]
```

Память

- Память разбивается на однотипные куски одинакового размера 4 кБ (иногда 4 МБ)
- Аллокация памяти происходит при первом обращении
- Со включенным режимом swap возможна подкачка памяти с HDD (на проде желательно выключать)

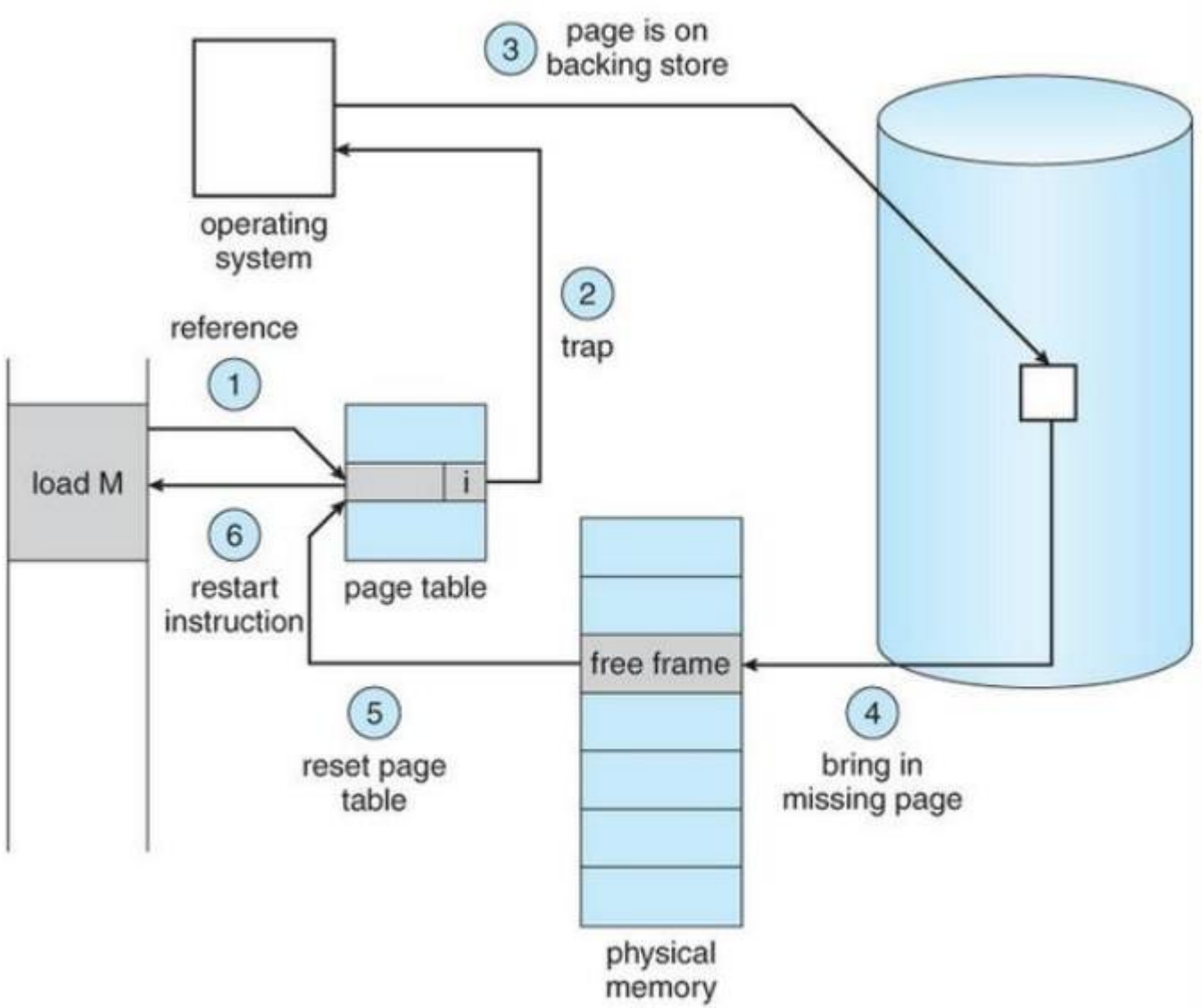
Вопрос

Что такое Page Fault?

Пишем в чат

Отвечаем устно

Page fault


3
page is on
backing store
operating
system
2
trap
reference
1
load M
i
6
restart
instruction
page table
free frame
5
reset page
table
4
bring in
missing page
physical
memory

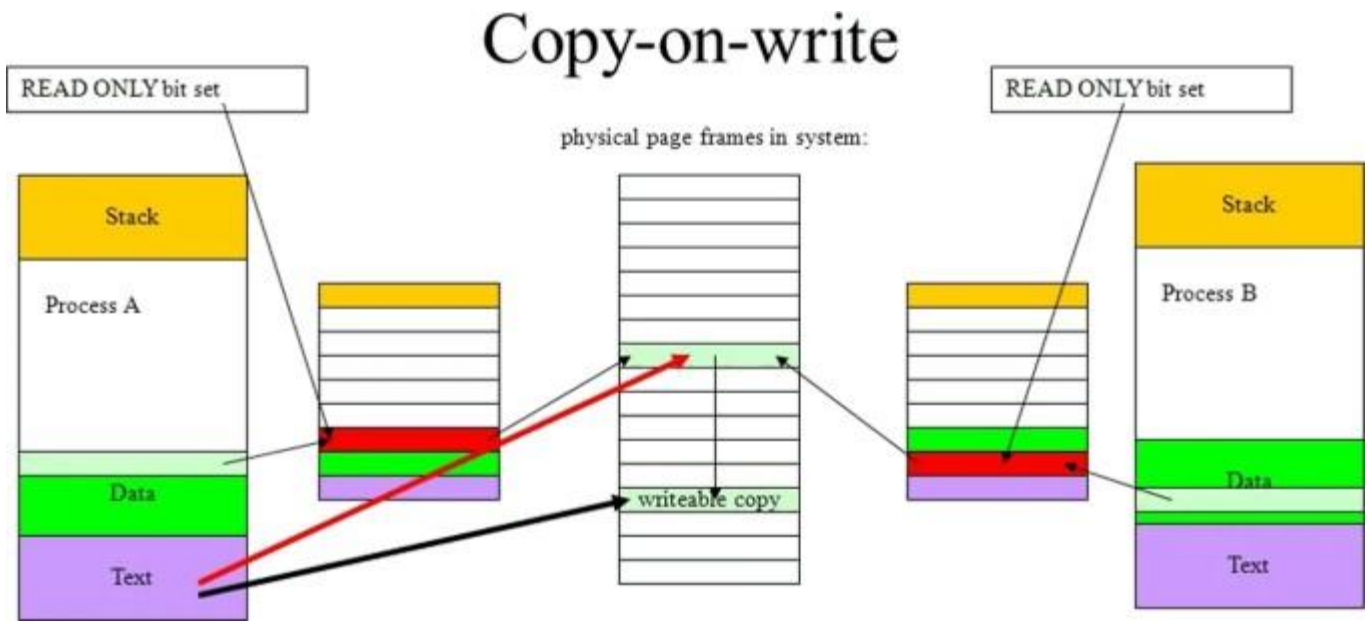
Вопрос

Что такое copy-on-write?

Пишем в чат

Отвечаем устно

Copy-on-write



Page fault

Причины возникновения

- Первое обращение к памяти
- ОС выгрузила страницу на HDD
- Copy-on-write

Почему плохо

- Происходит context switch
- В худших случаях еще и поход на HDD

Вопросы для проверки

По пройденному материалу

Код занимает 1 Мб, данные 1 Мб. Сколько нужно памяти после fork’а? А после изменения данных?

Пишем в чат

Отвечаем устно

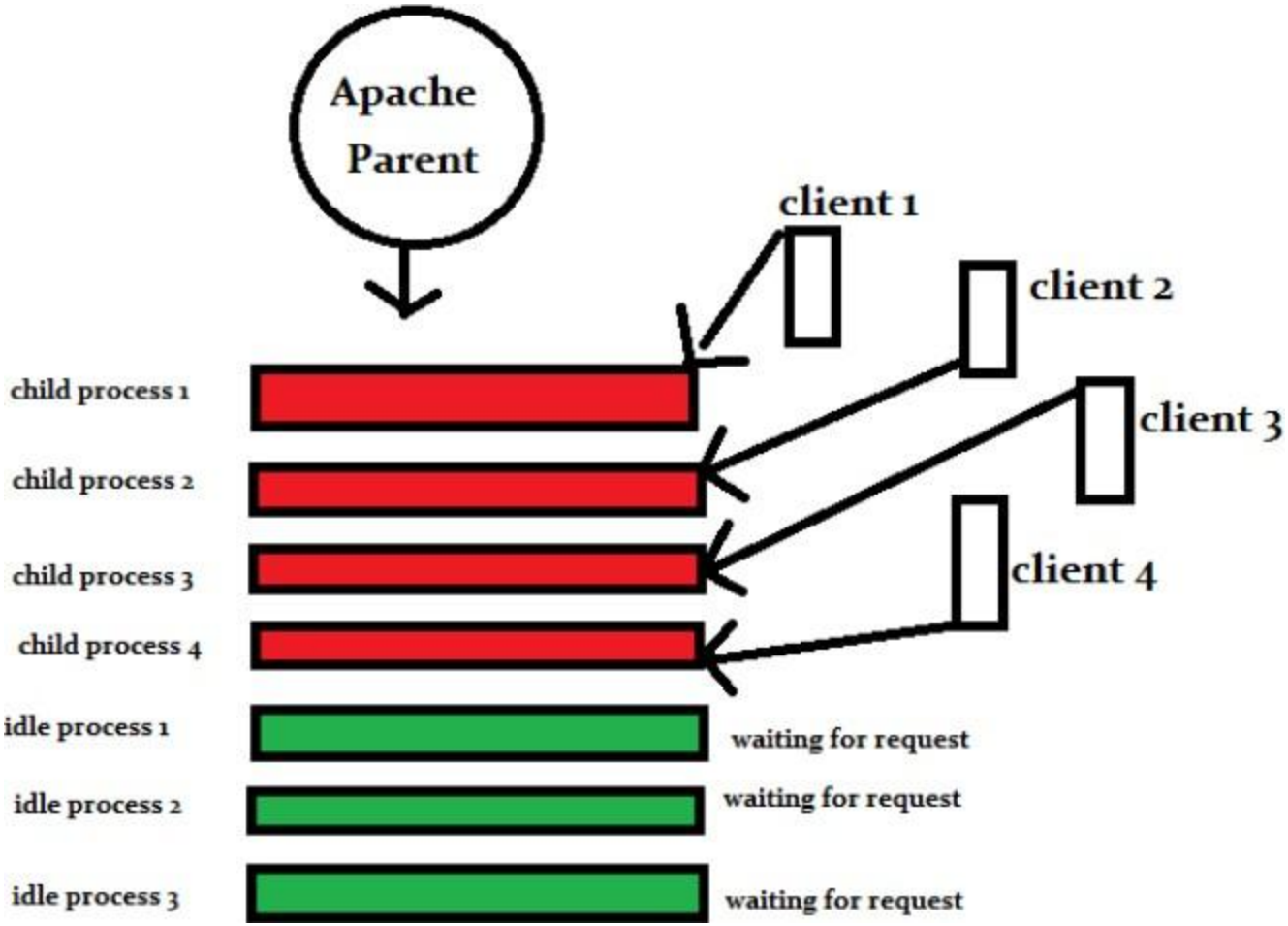
Есть ли вопросы?

Модели веб-серверов

Модели веб-серверов

- Worker (многопоточный)
- Prefork (многопроцессный)
- Асинхронный
- Комбинированный

Worker и prefork



Worker vs prefork

- Потоки экономят память
- Переключать потоки дешевле
- Потоки сложнее синхронизировать

Worker/prefork

Преимущества

- Простота

Недостатки

- Неэффективность при интенсивной работе с I/O (увеличение числа обработчиков – не вариант)
- Ограниченность размером пула числа клиентов, обслуживаемых одновременно
- Дороговизна выделения по обработчику на каждого клиента

Асинхронный веб-сервер

- Основан на однопоточном событийном цикле и шаблоне «reactor»
- Цикл событий читает события, представленные в виде callback’ов, из очереди с приоритетом (приоритет – время готовности)
- Клиентский код, инициируя операцию I/O, регистрирует callback в очереди
- Цикл событий опрашивает дескрипторы, ожидающие I/O, обновляет приоритеты в очереди
- В конце callback’а возвращаем управление циклу событий

Асинхронный веб-сервер

Преимущества

- Можно обрабатывать одновременно большое количество клиентов
- Отсутствует переключение контекста

Недостатки

- Потребляет не больше одного процессорного ядра (лечится применением комбинированной модели)
- Клиенты связаны одним потоком (утечки памяти, блокировки, ошибки, действия, нагружающие процессор недопустимы)
- Код становится сложнее

Комбинированный веб-сервер

- Имеет пул процессов, каждый из которых запускает пул потоков, каждый из которых использует модель обработки на основе асинхронного ввода-вывода
- Применяется в реальных серверах (nginx и apache http server в режиме MPM event)

Есть ли вопросы?

Возможности языковых платформ

Дальнейшее развитие

Асинхронный код	Синхронный код
• Быстрый	• Медленный
• Потребляет мало памяти	• Потребляет много памяти
• Сложно писать	• Просто писать

Соединим концепции на уровне языковых платформ или библиотек (fibers, green threads, goroutines).

Концепция fiber’ов

- Совмещаем плюсы синхронной и асинхронной модели
- Пишем свой более легковесный планировщик
(программа будет иметь линейный вид)
- Оставляем машину событий (получаем быстродействие)

Вопрос

А как вы делаете выбор языка программирования?

Пишем в чат

Отвечаем устно

Java

- Треды
- Future’ы
- Куча библиотек
- Не так все просто

Java

- Треды
- Future’ы
- Куча библиотек
- Не так все просто

Python

- Есть потоки
- Есть fiber
- Процесс с потоками в python может использовать 1 ядро (GIL)

Java

- Треды
- Future’ы
- Куча библиотек
- Не так все просто

Python

- Есть потоки
- Есть fiber
- Процесс с потоками в python может использовать 1 ядро (GIL)

NodeJS

- Синхронный код
- Строго однопоточный
- Надо поднимать много процессов
- Сложно писать*

Java

- Треды
- Future’ы
- Куча библиотек
- Не так все просто

Python

- Есть потоки
- Есть fiber
- Процесс с потоками в python может использовать 1 ядро (GIL)

NodeJS

- Синхронный код
- Строго однопоточный
- Надо поднимать много процессов
- Сложно писать

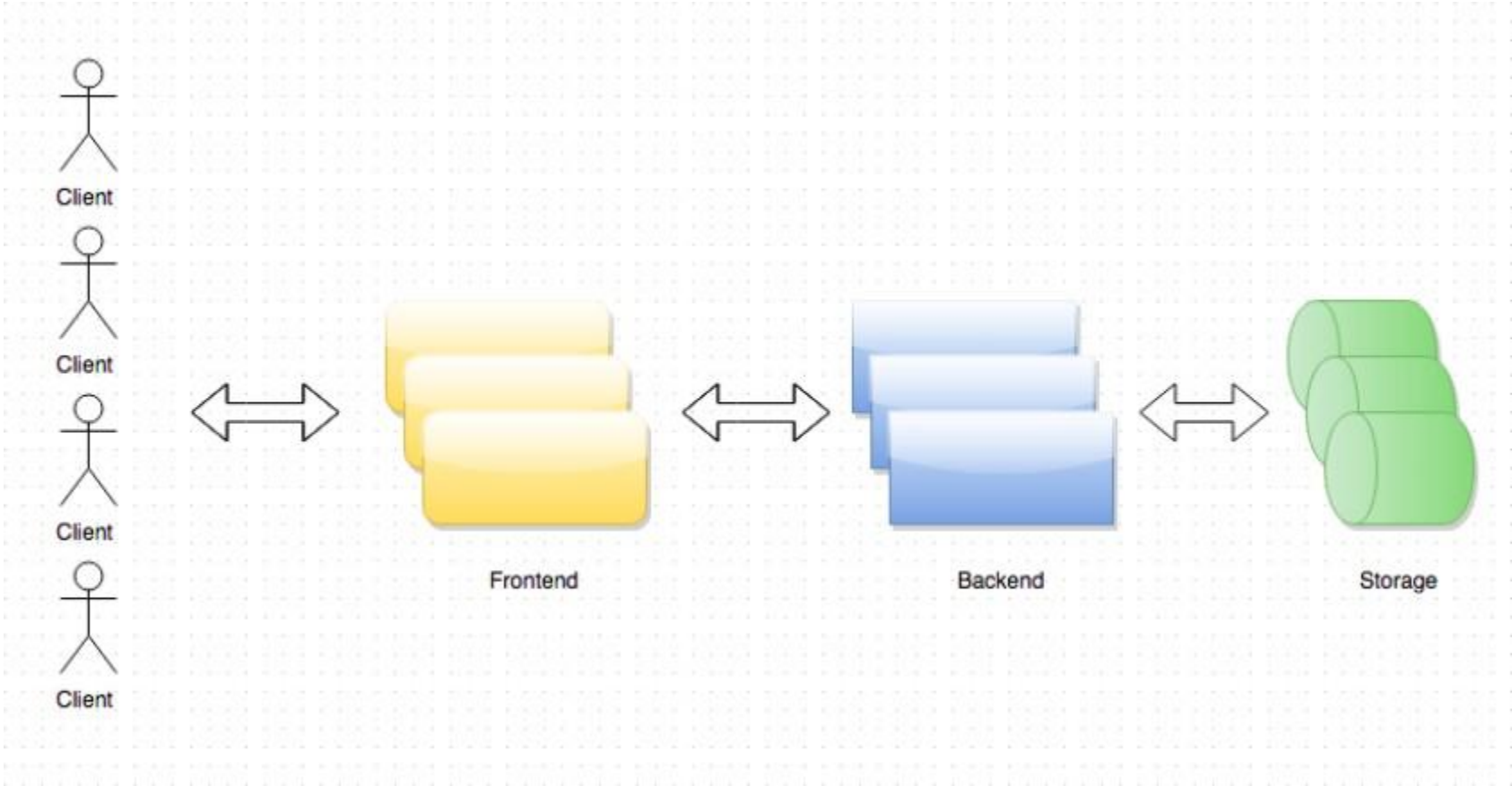
Go

- Концепция зеленых потоков
- Умеет использовать столько ядер CPU, сколько нужно
- Ориентирован на микросервисы
- Быстрый

Есть ли вопросы?

Трехзвенная архитектура

Трехзвенная архитектура



Задачи frontend (reverse proxy)

- Терминировать SSL- соединения
- Обработка медленных клиентов (spoon-feeding)
- Отдача статики
- Keep-alive
- Кеширование
- Балансировка
- Роутинг по backend’ам

Задачи backend

- Бизнес-логика
- Ожидание ответов от хранилищ/сервисов

Задачи хранилищ

- Хранение информации
- Быстрый поиск (индексы)
- Обеспечение транзакционности (ACID)

Вопрос

Какое из звеньев чаще всего является “бутылочным горлышком” и почему?

Пишем в чат

Отвечаем устно

Ваши вопросы?

Домашнее задание

Написать заготовку для социальной сети.

- без ORM
- фронт не нужен
- можно воспользоваться готовой спецификацией
- при возможности докеризировать
- БД - MySQL/PostgreSQL

Лучше делать сразу после занятия

Список материалов для изучения

1. strace
2. https://habr.com/ru/post/423049/
3. https://habr.com/ru/post/40227/
4. https://www.youtube.com/watch?v=4rLW7zg21gI
5. https://www.youtube.com/watch?v=TJzltwv7jJs
6. https://en.wikipedia.org/wiki/Reactor_pattern
7. https://java-design-patterns.com/patterns/reactor/

https://en.wikipedia.org/wiki/Reactor_pattern

Рефлексия

Цели вебинара

Проверка достижения целей

1. повторить/ознакомиться с основами работы операционных систем, существенные для высоких нагрузок;
2. рассмотреть принципы работы, преимущества и недостатки различных моделей веб-серверов;
3. проанализировать причину возникновения и принципы работы концепций fibers/green threads/goroutines;
4. проанализировать возможности языковых платформ с точки зрения поддержки асинхронного программирования.

Вопросы для проверки

Вопросы для проверки по всему вебинару

- Что такое context switch?

Пишем в чат

Отвечаем устно

Рефлексия

С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Пишем в чат

Отвечаем устно

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Следующий вебинар

M1: Введение в высокие нагрузки
Введение в docker. Обзор docker-compose

Ссылка на вебинар будет в ЛК за 15 минут

Материалы к занятию в ЛК — можно изучать

Обязательный материал обозначен красной лентой

Спасибо за внимание!

Приходите на следующие вебинары

Сапогов Никита

руководитель курса “Архитектор высоких нагрузок”

Telegram: @AmsTaFFix