ОНЛАЙН-ОБРАЗОВАНИЕ

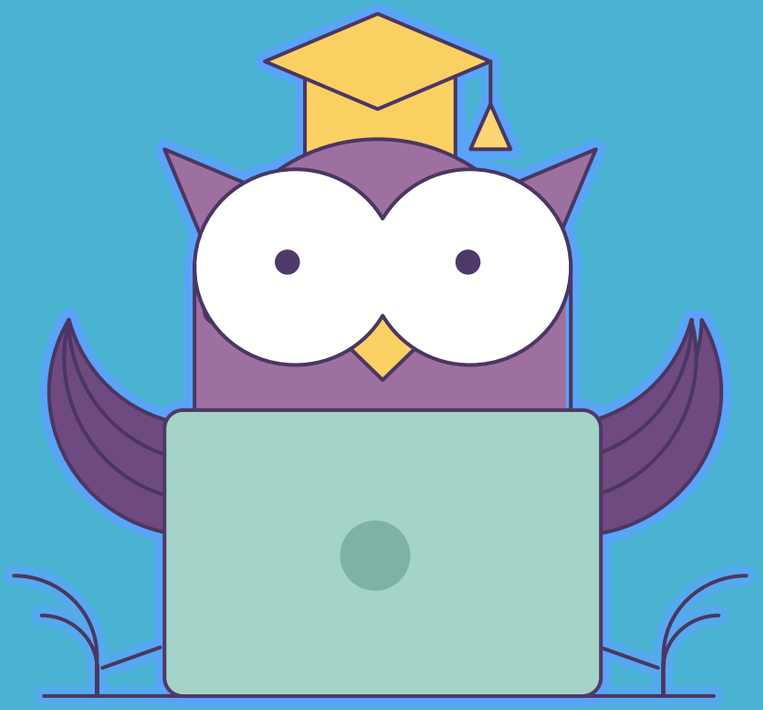
OTUS

Проблема высоких нагрузок

Иван Ремень

OTUS

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

– если есть проблемы со звуком или с видео

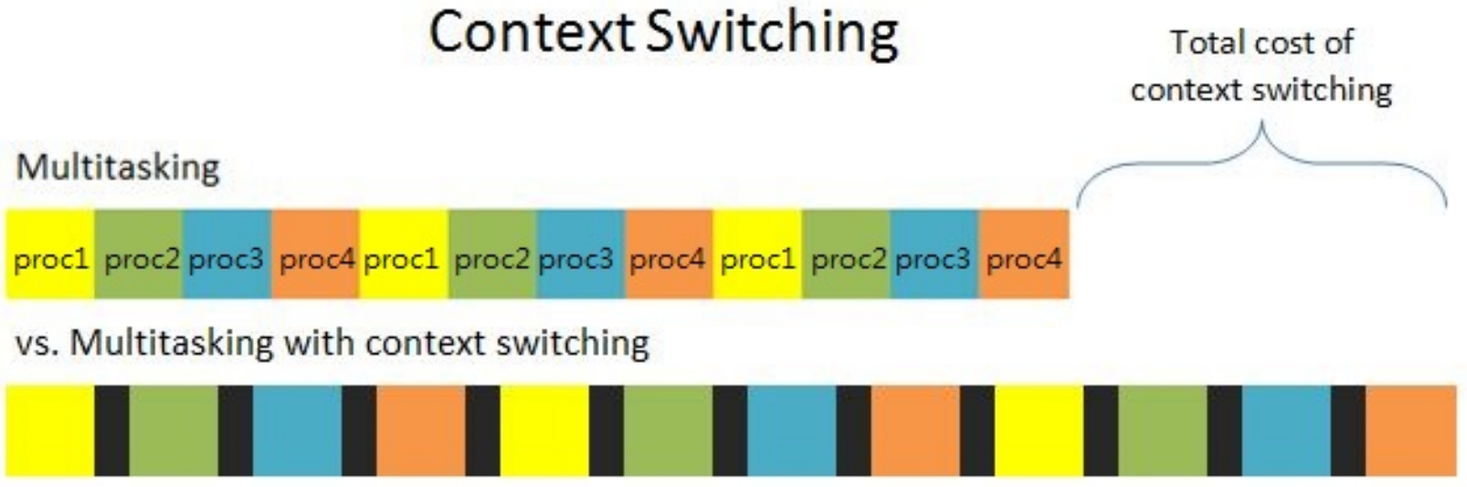
Цель занятия

- Понять, что такое "Переключение контекста".
- Узнать модели языков программирования.
- Узнать модели веб-серверов.
- Узнать, что такое трехзвенная архитектура и зачем она нужна.

Переключение контекста (context switch)

- Сохранение регистров процессора
- Сохранение общей информации pid, tid, uid, gid, euid, egid и т. д.
- Сохранение состояния процесса/потока
- Сохранение прав доступа
- Сохранение используемых потоком ресурсы и блокировки
- Сохранение счетчики использования ресурсов (например, таймеры использованного процессорного времени)
- Сохранение регионов памяти, выделенных процессу
- Очистка конвейера команд и данных процессора
- Очистка TLB, отвечающий за страничное отображение линейных адресов на физические.

Переключение контекста (context switch)



Потоки и процессы

- В linux процессы и потоки почти одно и тоже
- Потоки имеют общую память

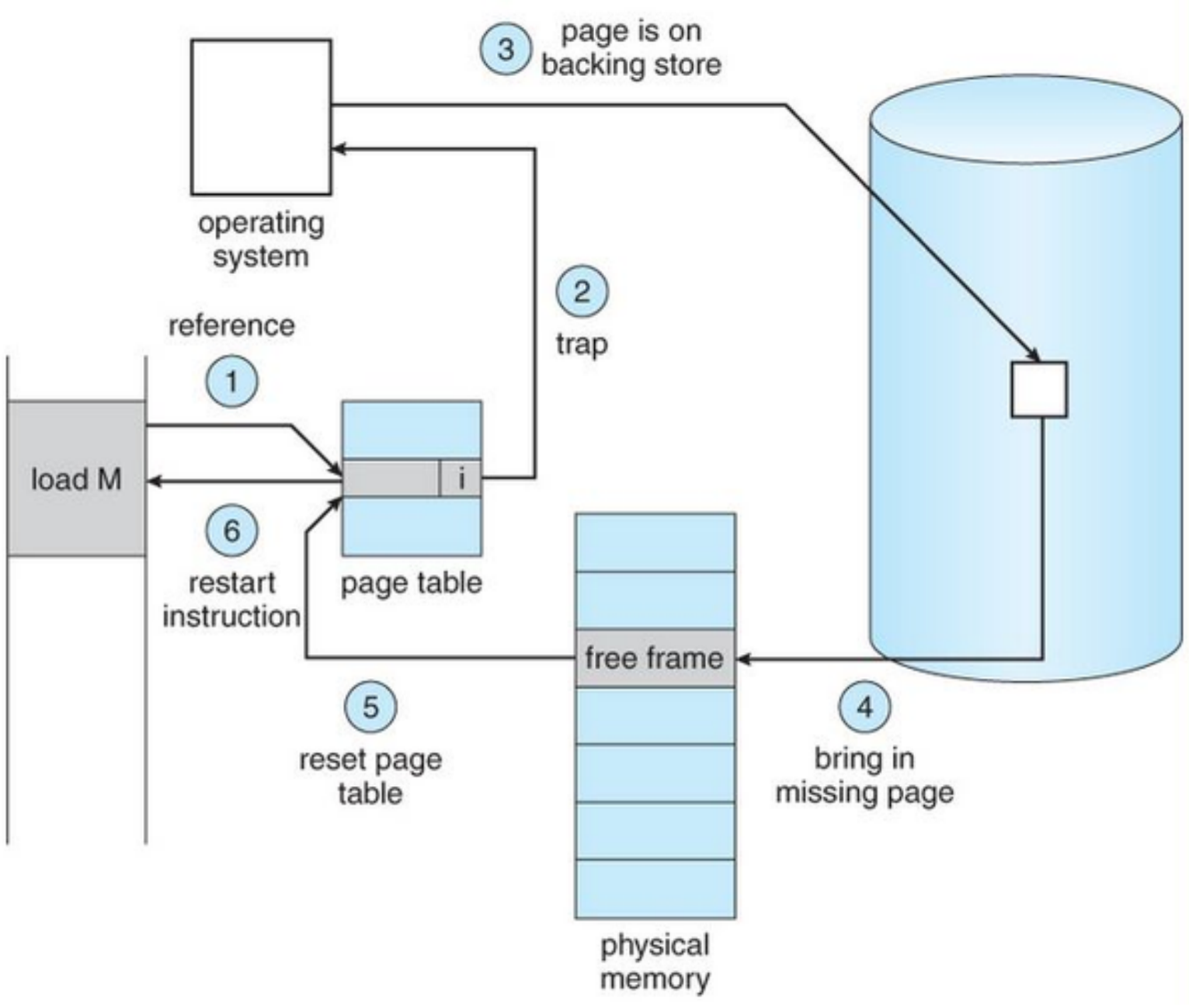
Потоки и процессы

Какой системный вызов создает процессы?
Какой системный вызов создает потоки?

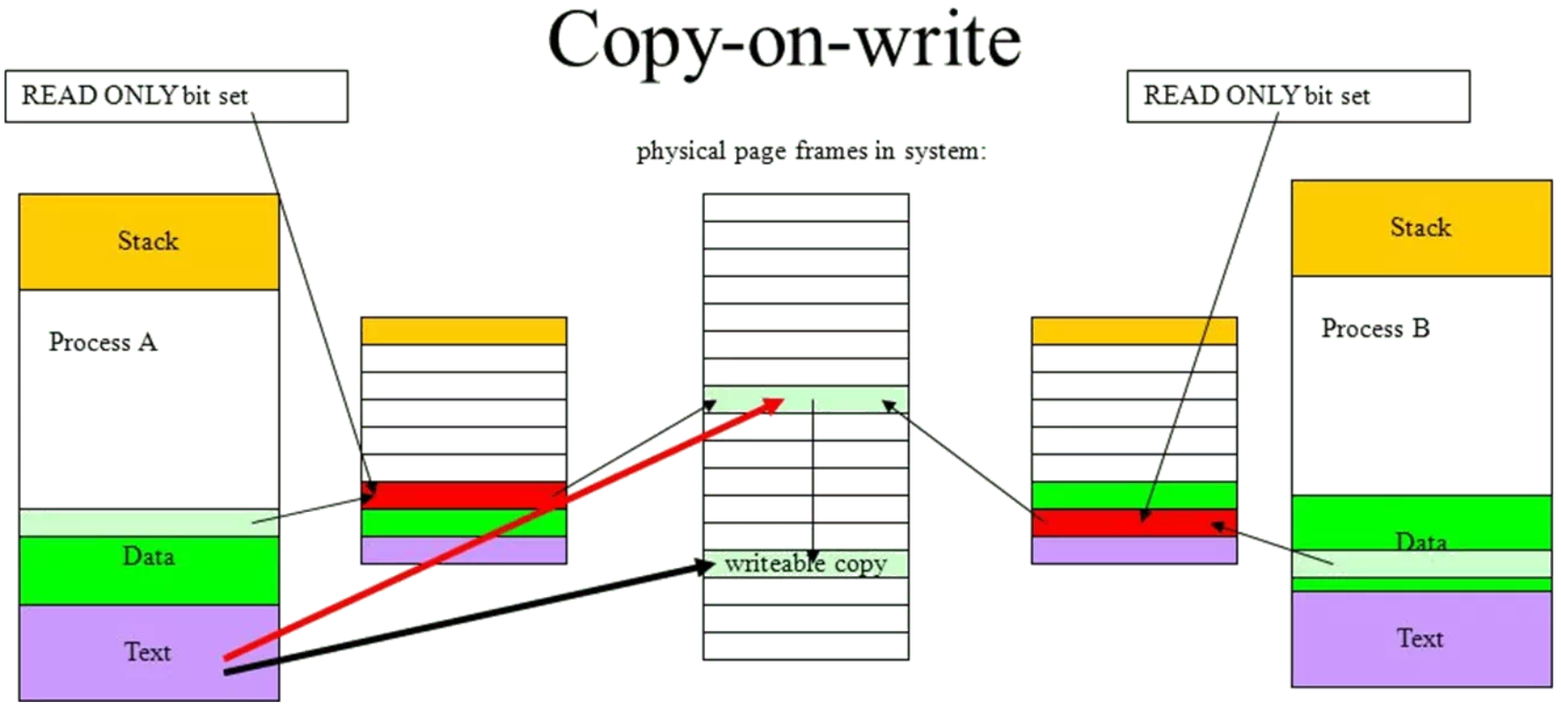
Системные вызовы

```
fork()
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fa001a93a10) = 6916
pthread_create()
clone(child_stack=0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SYSVSEM|CLONE_SIGHAND|CLONE_THREAD|CLONE_SETTLS|CLONE_PARENT_
child_tidptr=0x7fa001a93a10) = 6916
```

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

Copy-on-write



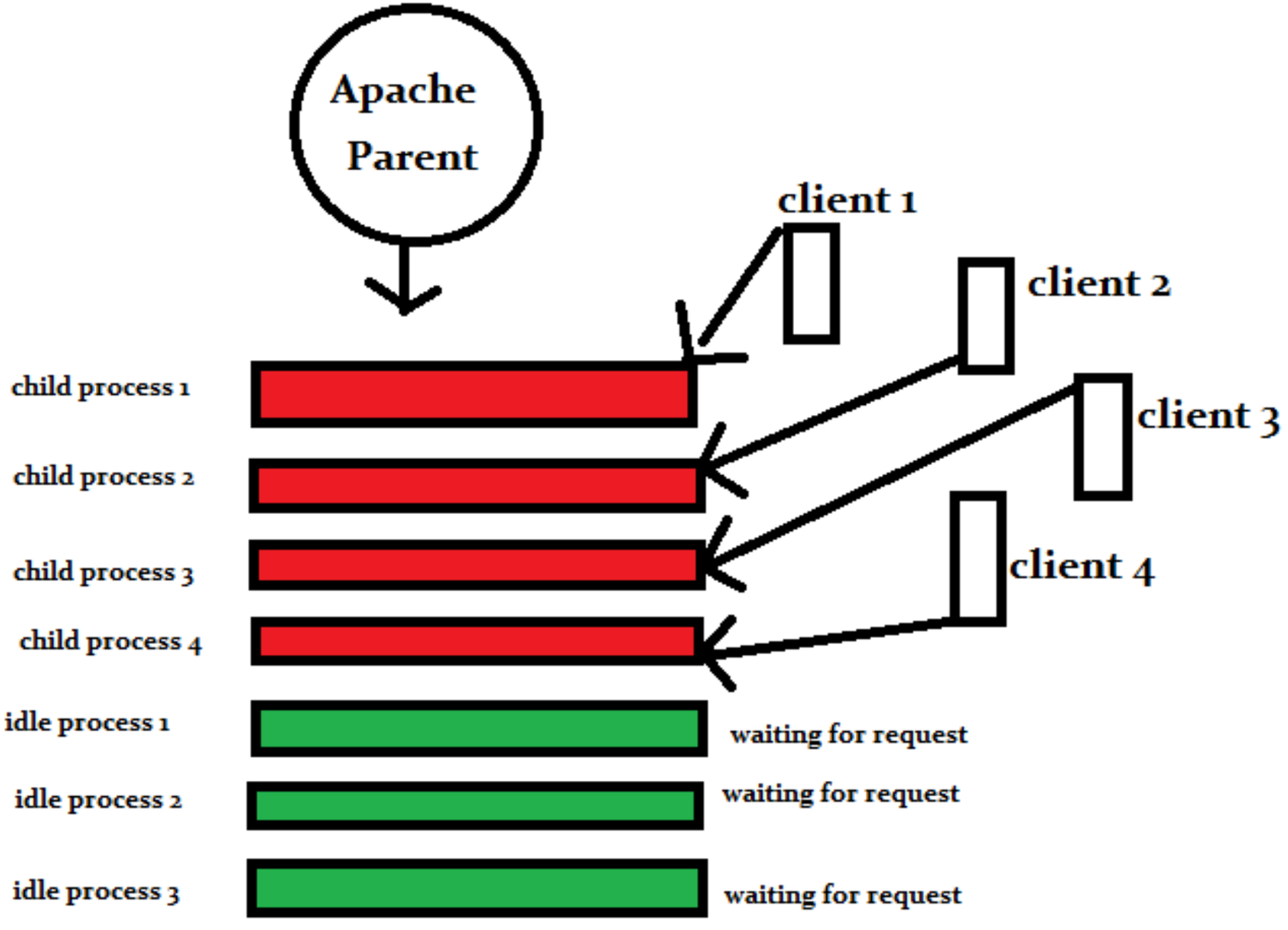
Потоки и процессы

Какой системный вызов создает процессы?
Какой системный вызов создает потоки?

Модели веб-серверов.

- worker (многопоточный)
- prefork (многопроцессный)
- async
- Комбинированные варианты

worker и prefork



worker vs prefork

- Потоки экономят память.
- context switch между потоками дешевле
- Потоки сложнее синхронизировать.

Блокирующие и неблокирующие системные вызовы

Блокирующие системные вызовы блокируют процесс (или поток) до того, как будут получены данные (часть данных).
Во время блокировки процесс не потребляет процессорное время, но потребляет память.
Чтение из сети может быть неблокирующим. В неблокирующем режиме чтение из сокета возвращает или ответ (хотя бы его часть), или сообщение о том, что данные еще не готовы.
Чтение с диска может быть только блокирующим(*).

Вопрос

Почему запись в сеть может заблокировать?

async модель

- Асинхронные сервера писать сложнее
- Необходимо избегать блокировок, иначе сильно растет latency
- Потребляют не более одного ядра (но на помощь приходит комбинированная модель)

Исходники nginx

```
switch (s->protocol) {

case NGX_MAIL_POP3_PROTOCOL:
    p->upstream.connection->read->handler = ngx_mail_proxy_pop3_handler;
    s->mail_state = ngx_pop3_start;
    break;

case NGX_MAIL_IMAP_PROTOCOL:
    p->upstream.connection->read->handler = ngx_mail_proxy_imap_handler;
    s->mail_state = ngx_imap_start;
    break;

default: /* NGX_MAIL_SMTP_PROTOCOL */
    p->upstream.connection->read->handler = ngx_mail_proxy_smtp_handler;
    s->mail_state = ngx_smtp_start;
    break;
}
```

Языки программирования

Как вы выбираете язык?

Концепция fiber

- Асинхронный код - быстрый, потребляет мало памяти.
- Асинхронный код - сложно писать.
- Синхронный код - медленее, потребляет больше памяти.
- Синхронный код - просто писать.

Попытка объединить - fibers.

Perl

- Обычно работает внутри apache.
- Синхнронный язык с поддержкой тредов (редко).
- Есть библиотека AnyEvent (еще реже тредов)
- Есть библиотека для fiber
- И да, он все еще жив ;-)

- CLI SAPI - в качестве консольной команды **php** для запуска наших кронов и других cli-программ (Command Line Interface)
- apxs2 SAPI - в качестве модуля к apache2
- CGI SAPI - в качестве запускаемого на каждом запросе CGI (сейчас так почти никто не делает)
- FPM SAPI - Fast Process Manager, написанный для PHP разработчиками из комании Badoo и теперь поддерживаемый сообществом

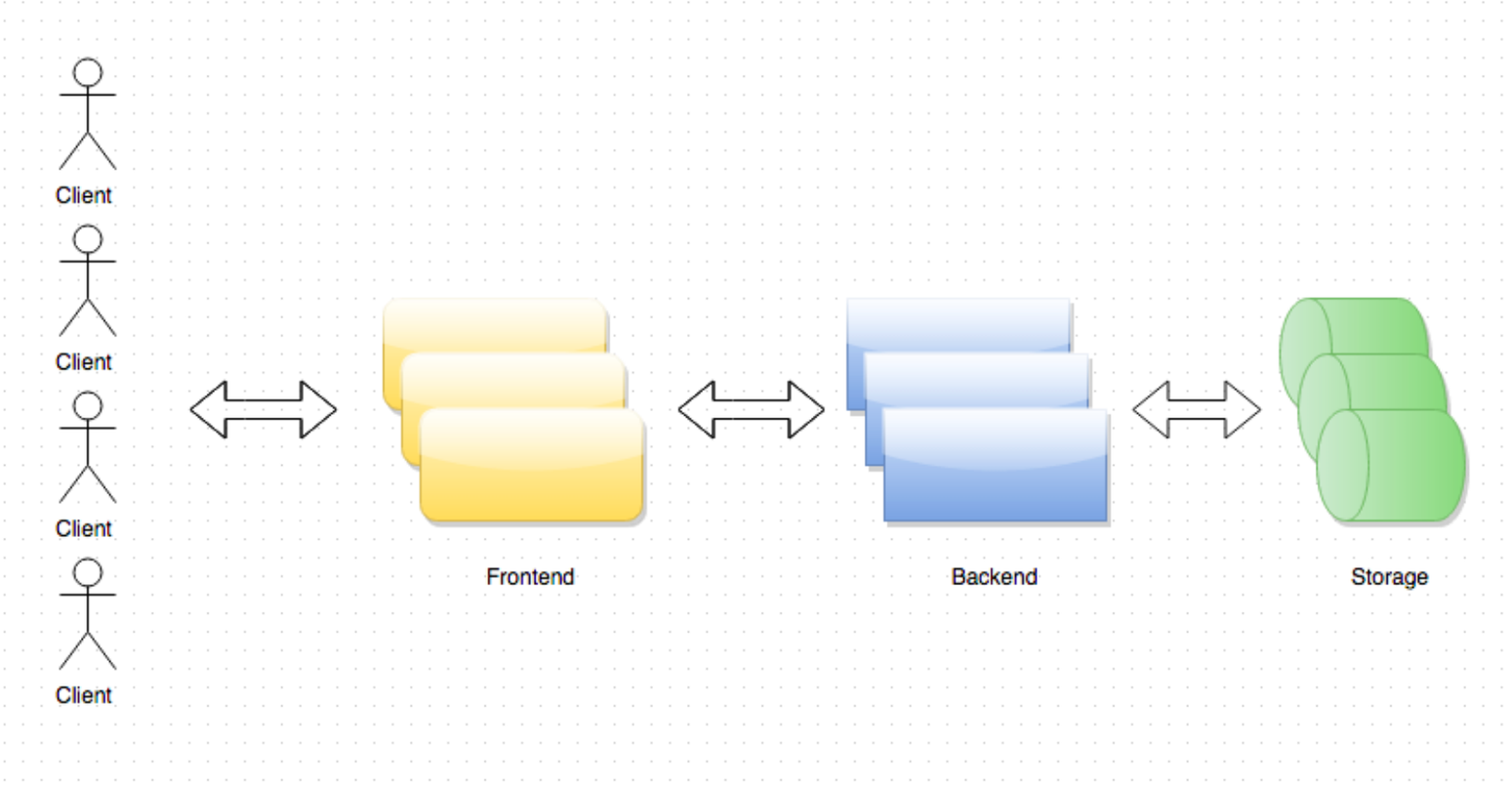
Можно делать потоки, но редко используется.

- Есть потоки.
- Есть fiber.
- А сколько ядер может использовать процесс с потоками в Python?

- Асинхронный код.
- Строго однопоточный.

- Концепция зеленых тредов.
- Умеет использовать столько ядер ЦПУ, сколько нужно.
- Ориентирован на микросервисы.
- Быстрый.

Трехзвенная архитектура



Задачи frontend (reverse proxy)

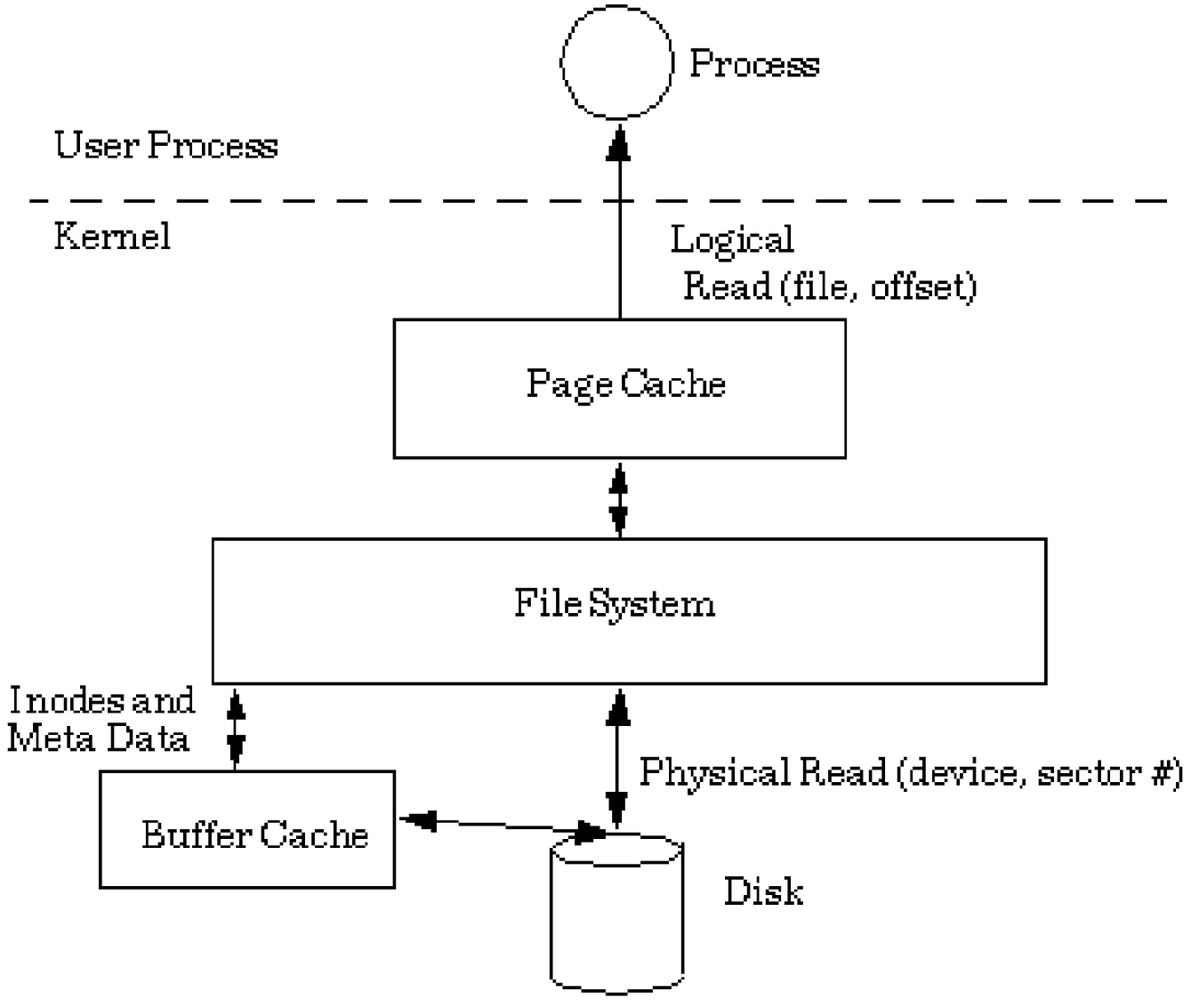
- Терминировать ssl-соединения.
- Обработка медленных клиентов.
- Отдача статики.
- Keep-Alive.
- Кэширование.
- Балансировка.
- Роутинг по бэкендам.

Идея используется не только для HTTP.

Вопрос: отдача статики.

Факт №1: чтение с диска в linux всегда блокирующая операция.
Факт №2: nginx нельзя блокировать.
Факт №3: статические файлы лежат на диске.
Почему же мы раздаем статику nginx'ом?

Page cache


Process
User Process
Kernel
Logical
Read (file, offset)
Page Cache
File System
Inodes and
Meta Data
Physical Read (device, sector #)
Buffer Cache
Disk

Задачи backend

- Бизнес-логика
- Ожидание ответов от хранилищ

Задачи хранилищ

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

Итоги занятия

- Поняли, что такое "Переключение контекста".
- Узнали модели языков программирования.
- Узнали модели веб-серверов.
- Узнали, что такое трехзвенная архитектура и зачем она нужна.

Вопросы?

Спасибо за внимание!