



# Асинхронное программирование

# Разработка веб-сервисов на Golang

# Цель занятия

1. Научиться работать с ошибками
2. Узнать инструментарий для асинхронного и параллельного программирования

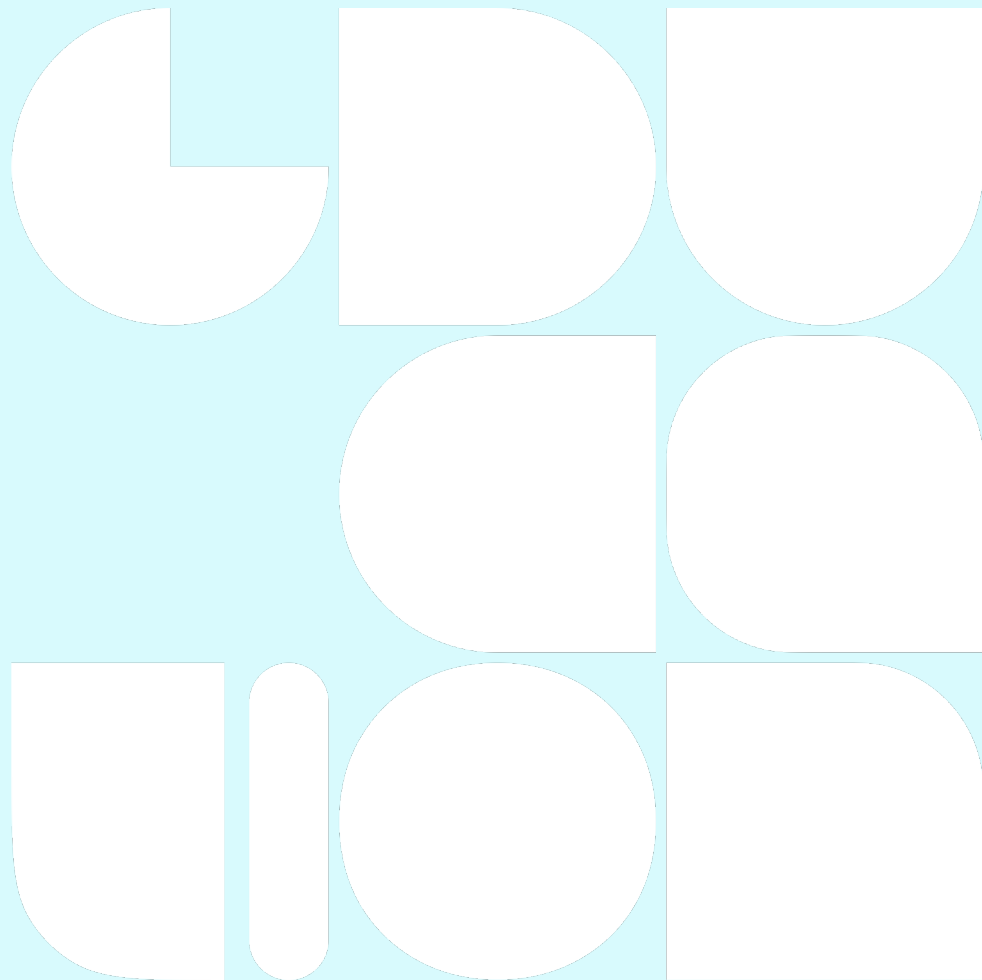


# О чём поговорим?

- Основы работы с ошибками в языке
- Подходы к эффективной обработке веб запросов
- Основные инструменты языка для написания конкурентного/параллельного кода
- Шаблоны конкурентного программирования в Go

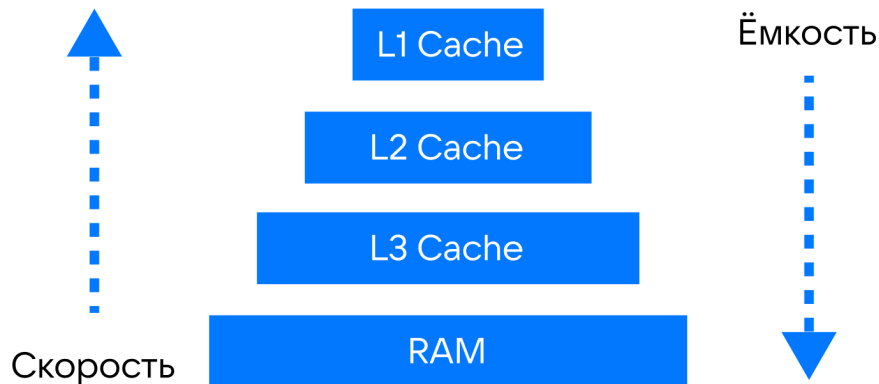
# Асинхронное программирование

Утилизируем сервер



# Memory

- Скорость ЦПУ растёт быстрее скорости памяти
- Чтобы это скомпенсировать есть кеш процессора



# Context Switch

- Процессор выполняет только 1 задачу одновременно
- Планировщик задач переключает их
- Для этого надо выгрузить одну и загрузить другую
- Для этого может потребоваться обращение к основной памяти



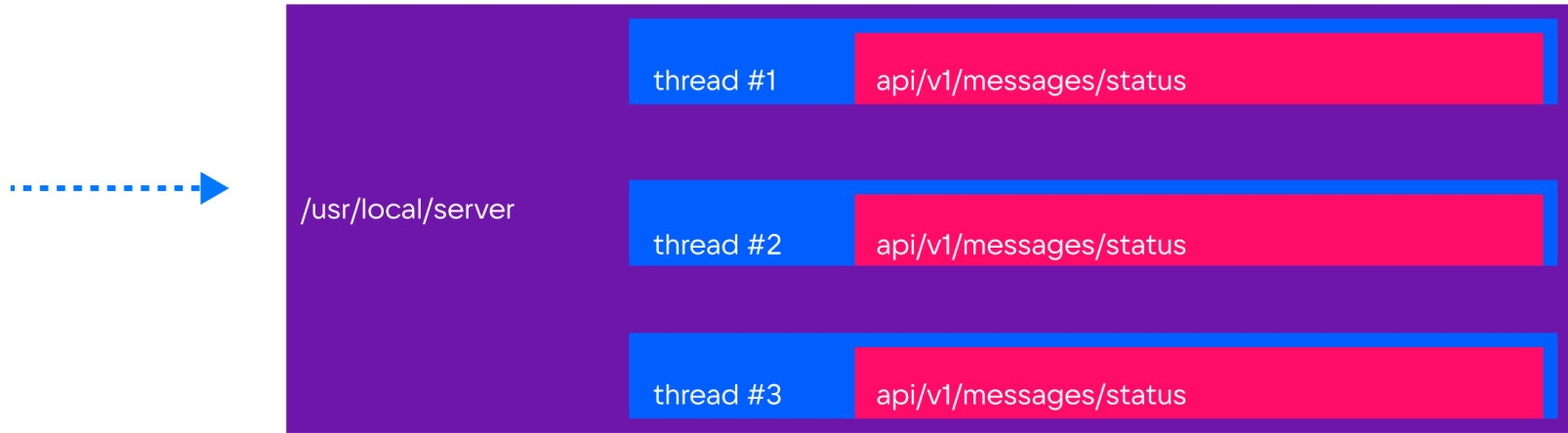
# cgi-bin

- Одно соединение — один процесс
- Если много запросов — упрёмся в память



# Multithreading

- Одно соединение — один тред
- Если много запросов — упрёмся в память + дорогое переключение





# А что внутри запроса?



Работа программы (операции на ЦПУ сервера)



Ожидание внешних сервисов (например, ответа от БД)

Запрос #1

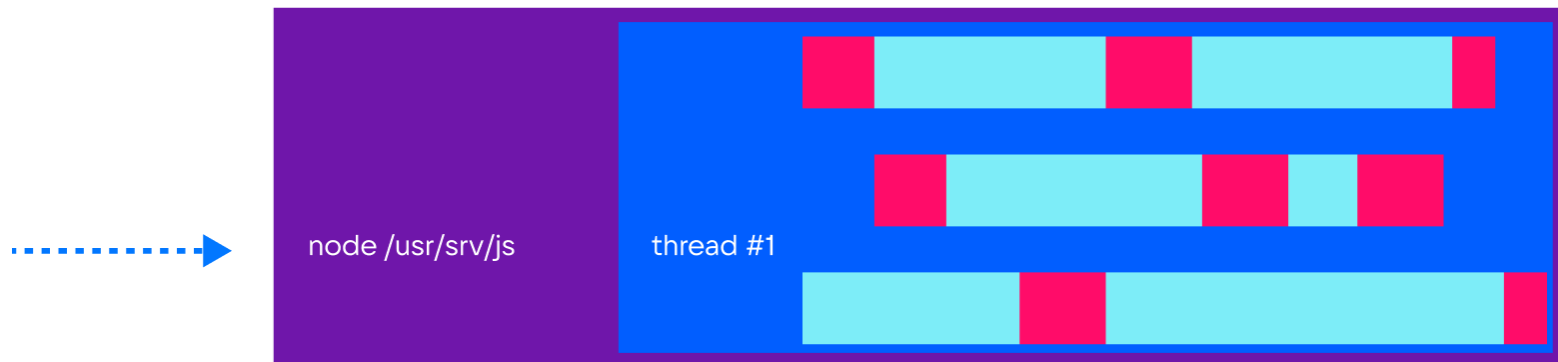


Запрос #2



# non-blocking IO на 1 ядре

- Событийная модель, кооперативная многозадачность
- Одновременно работает только 1 запрос, но I/O не блокирующий
- Если много запросов — упрямся в ЦПУ



# non-blocking IO на 1 ядре



Работа программы (операции на ЦПУ сервера)



Ожидание внешних сервисов (например, ответа от БД)



Ожидание другого запроса

Запрос #1



Запрос #2

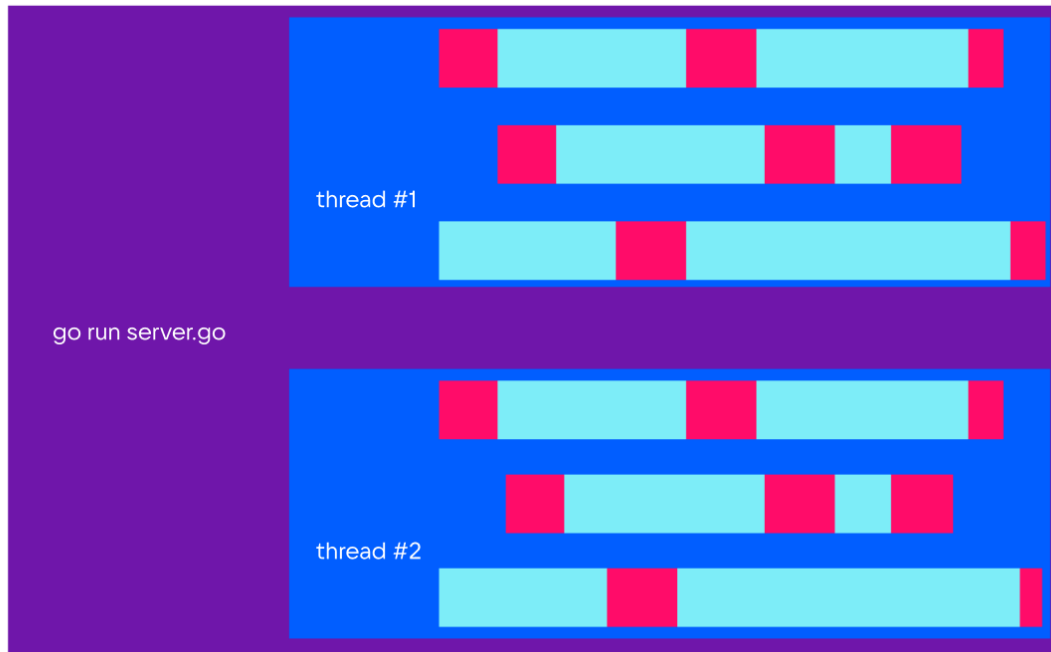


Запрос #2 — реальность



# non-blocking IO на все ядре (multithreading)

- Реализовано в Golang
- На основе CSP\* Тони Хоара
- Оперирует легковесными потоками — горутинами



\*CSP - communicating sequential processes, взаимодействующие последовательные процессы



# Вопросы?

