Яндекс.Практикум

Работа с горутинами

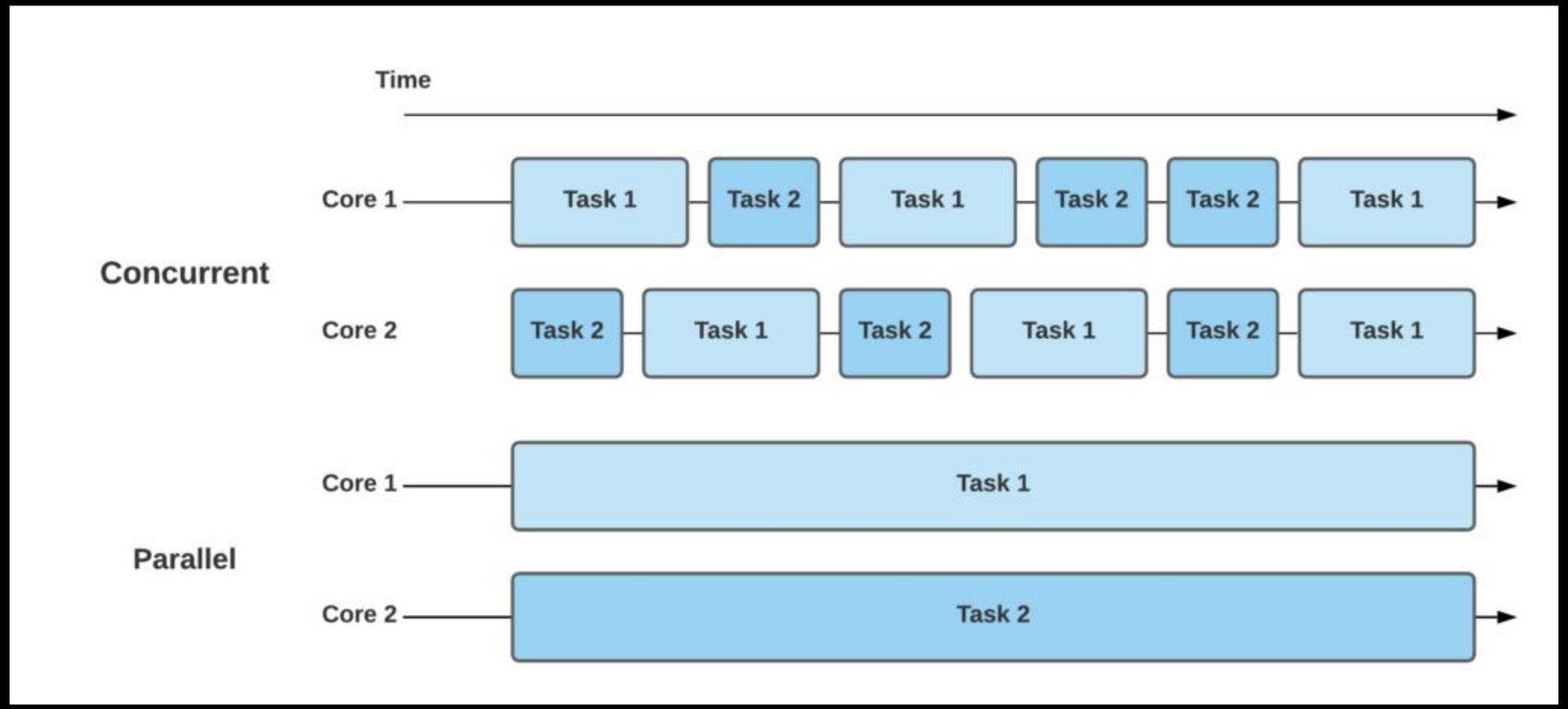
Что нас сегодня ждет

- Обсудим,чем конкурентность отличается от параллелизма
- Поговорим про каналы, контекст, graceful shutdown
- Напишем простой генератор нагрузки
- Научимся детектировать race conditions и предотвращать deadlock’и

Конкурентность vs параллелизм

Параллелизм = одновременное выполнение нескольких операций. Требует нескольких CPU

Конкурентность = возможность выполнения нескольких операций в один и тот же временной интервал



Конкурентность в Go

Планированием запуска Go-рутин на тредах ОС занимается runtime.

Программист может опосредованно влиять на работу планировщика:

- GOMAXPROCS
- Gosched()
- LockOSThread()

Основной механизм реализующий конкурентность - каналы: с помощью них go-рутины обмениваются сообщениями.

Хороший цикл статей про планировщик:

https://www.ardanlabs.com/blog/2018/08/scheduling-in-go-part1.html

Каналы

Какие из этих операций приведут к панике:

1. Чтение из закрытого канала
2. Запись в закрытый канал
3. Закрытие закрытого канала
4. Чтение из nil-канала
5. Запись в nil-канал
6. Закрытие nil-канала

Каналы

Какие из этих операций приведут к панике:

1. Чтение из закрытого канала
2. Запись в закрытый канал
3. Закрытие закрытого канала
4. Чтение из nil-канала
5. Запись в nil-канал
6. Закрытие nil-канала



Именно поэтому важно, чтобы только одна горутина управляла каналом (могла его закрыть).

Буферизованные каналы

Обычные каналы = синхронное взаимодействие между горутинами.

Буферизованные каналы = асинхронное взаимодействие.

Как выбрать размер буферизованного канала?

Буферизованные каналы

Обычные каналы = синхронное взаимодействие между горутинами.

Буферизованные каналы = асинхронное взаимодействие.

Как выбрать размер буферизованного канала?

- В общем случае лучше начать с размера 1
- Если буферизованный канал используется для передачи данных набору горутин, то его длина может равняться количеству этих горутин
- Функциональность приложения не должна зависеть от размера буферизованного канала

Контекст и graceful shutdown

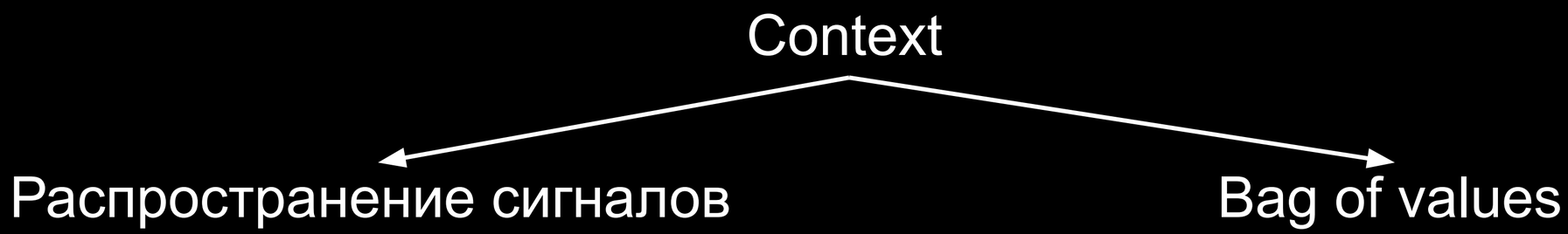
Load-генератор

Напишем небольшой load-генератор

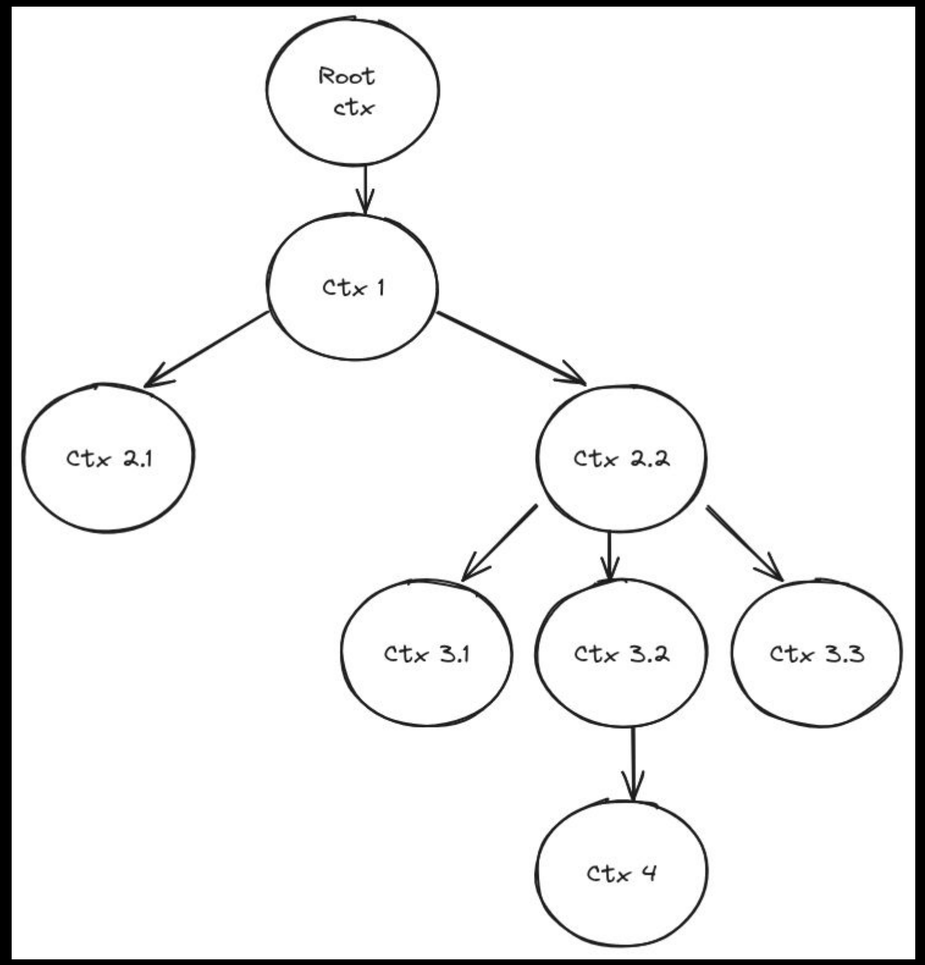
Инструменты для нагрузочного тестирования:

- Yandextank (https://github.com/yandex/yandex-tank) + Pandora (https://github.com/yandex/pandora)
- gatling: https://github.com/gatling/gatling

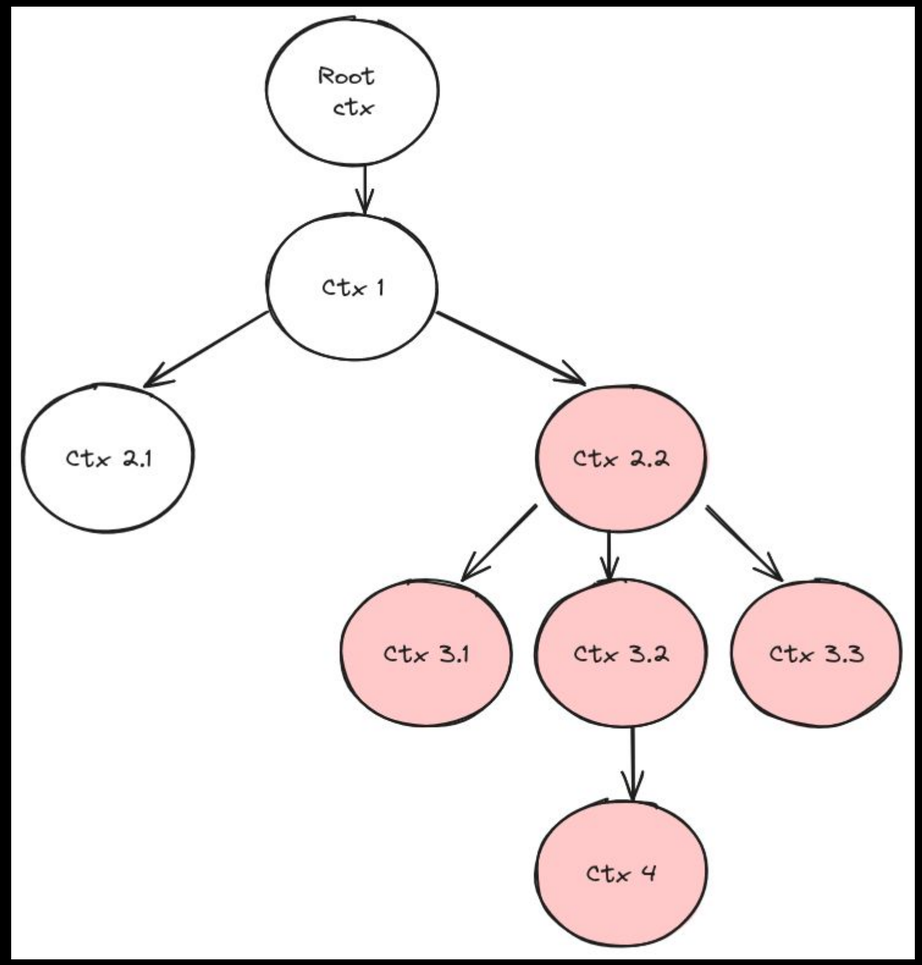
Context



Context. Распространение сигналов



Отмена Ctx 2.2



Context. Bag of value

В контекст можно поместить любой объект. Извлечь этот объект можно по ключу. Этот объект доступен всем дочерним контекстам.

При этом:

1. объекты и ключи не типизированы
2. создается множество неявных зависимостей

Context стоит использовать как последнее средство передачи объектов.

Race-детектор

В Go есть утилита для детектирования data-race’ов. Для ее активации добавьте флаг *-race* к командам:

- *go test*
- *go run*
- *go build*
- *go install*

Подключение race-детектора приводит к снижению быстродействия приложения, отключайте его в production-окружении

https://go.dev/doc/articles/race_detector

Утечки горутин

Утечки горутин удобнее всего детектировать при помощи утилиты *perf*, но ее мы не будем разбирать сегодня

Одно из удобных решений для детектирования утечек:

https://github.com/uber-go/goleak

Спасибо за внимание!

Вопросы?