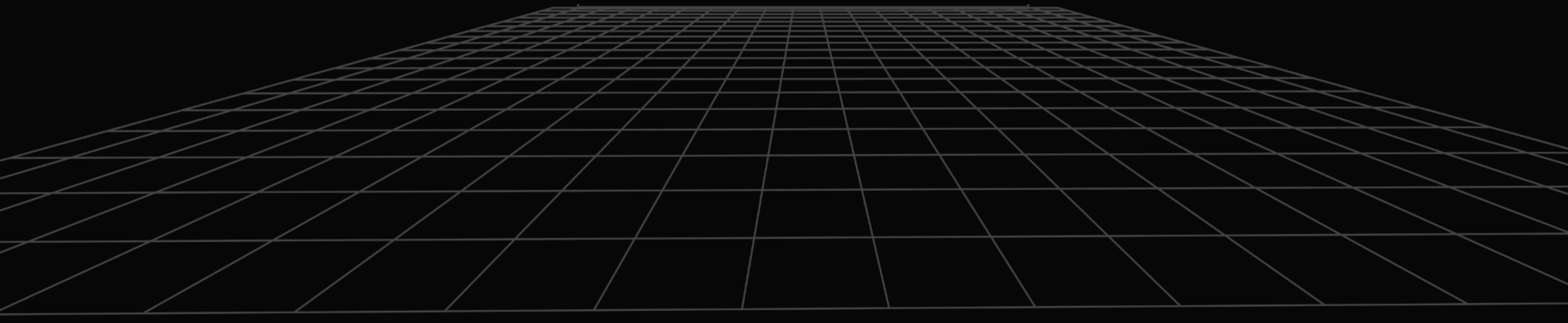


ГОРУТИНЫ И ПЛАНИРОВЩИК GO

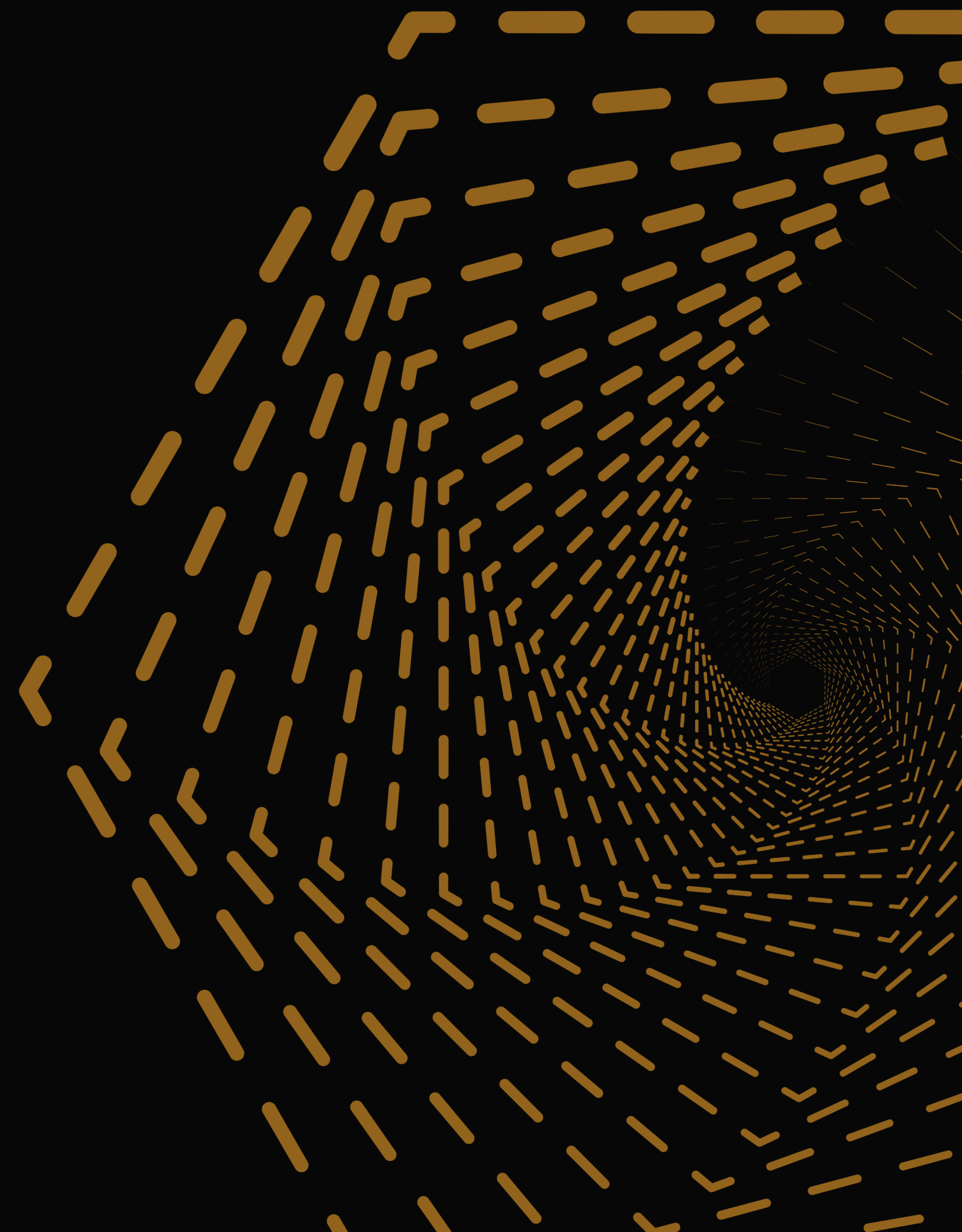


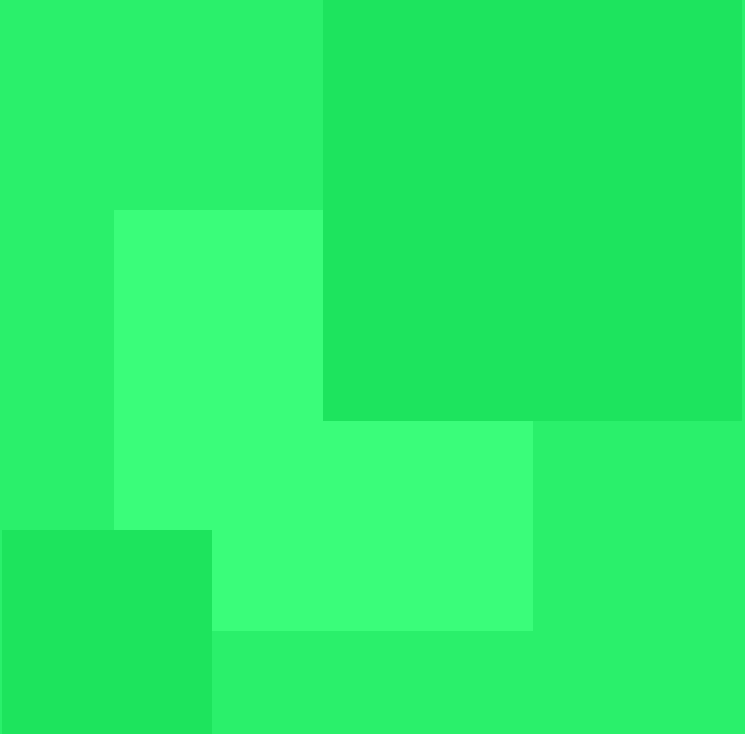
ПРОВЕРЬ ЗАПИСЬ



ПРАВИЛА ЗАНЯТИЯ

1. вопросы в чате можно задавать
в любое время
2. вопросы голосом задаем
по поднятой руке в Zoom
3. ответы на вопросы будут
в запланированных местах





ГОРУТИНЫ

CONTEXT SWITCHING

THREAD

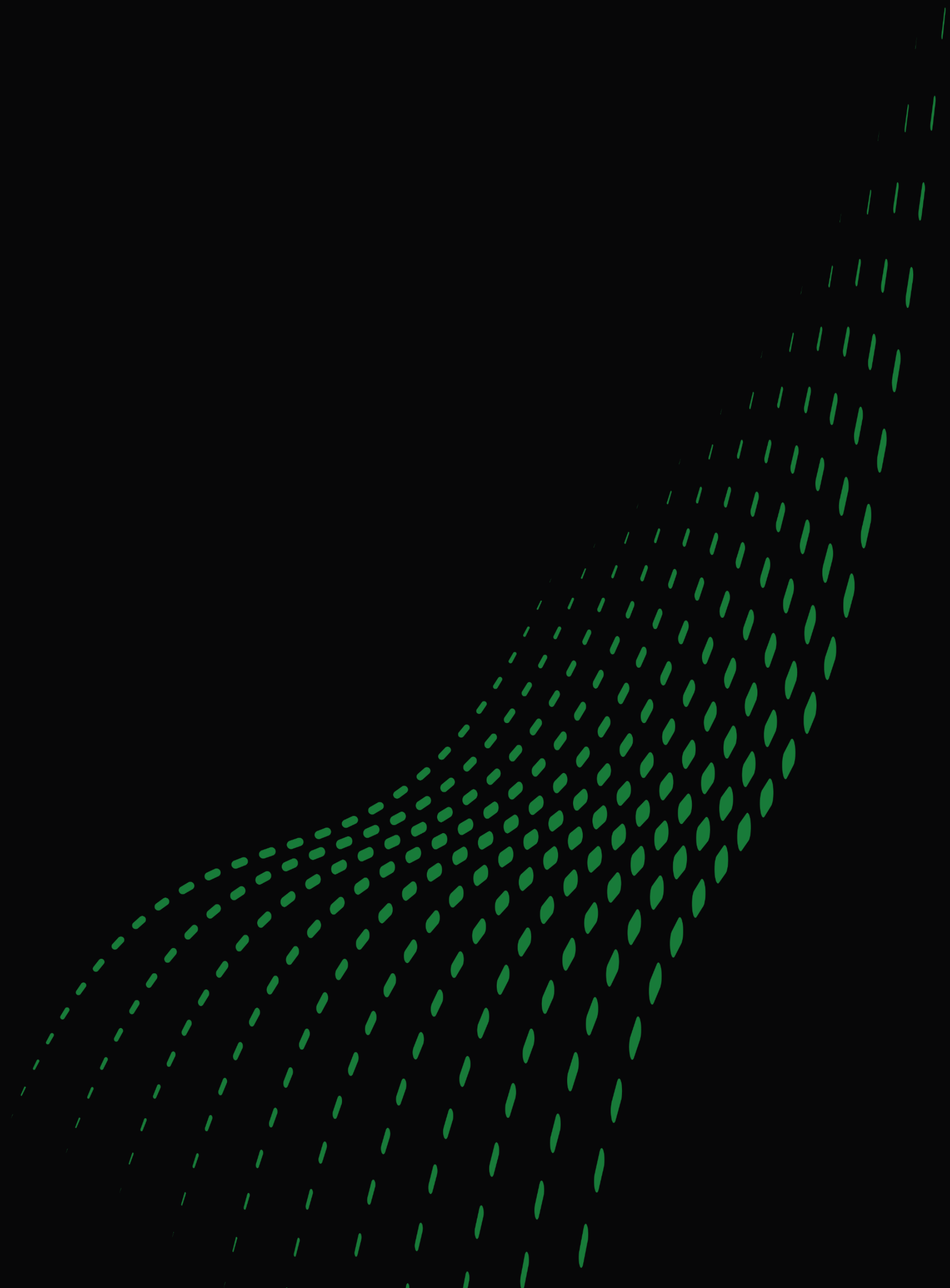
Goroutine #1
(registers + stack)

Goroutine #2
(registers + stack)

Goroutine #3
(registers + stack)

КЛЮЧЕВАЯ ИДЕЯ

1. Не даем в руки программисту поток
2. Делаем вид, что есть только горутины
3. Всю сложность распределения горутин абстрагируем



EXAMPLE

Goroutines number

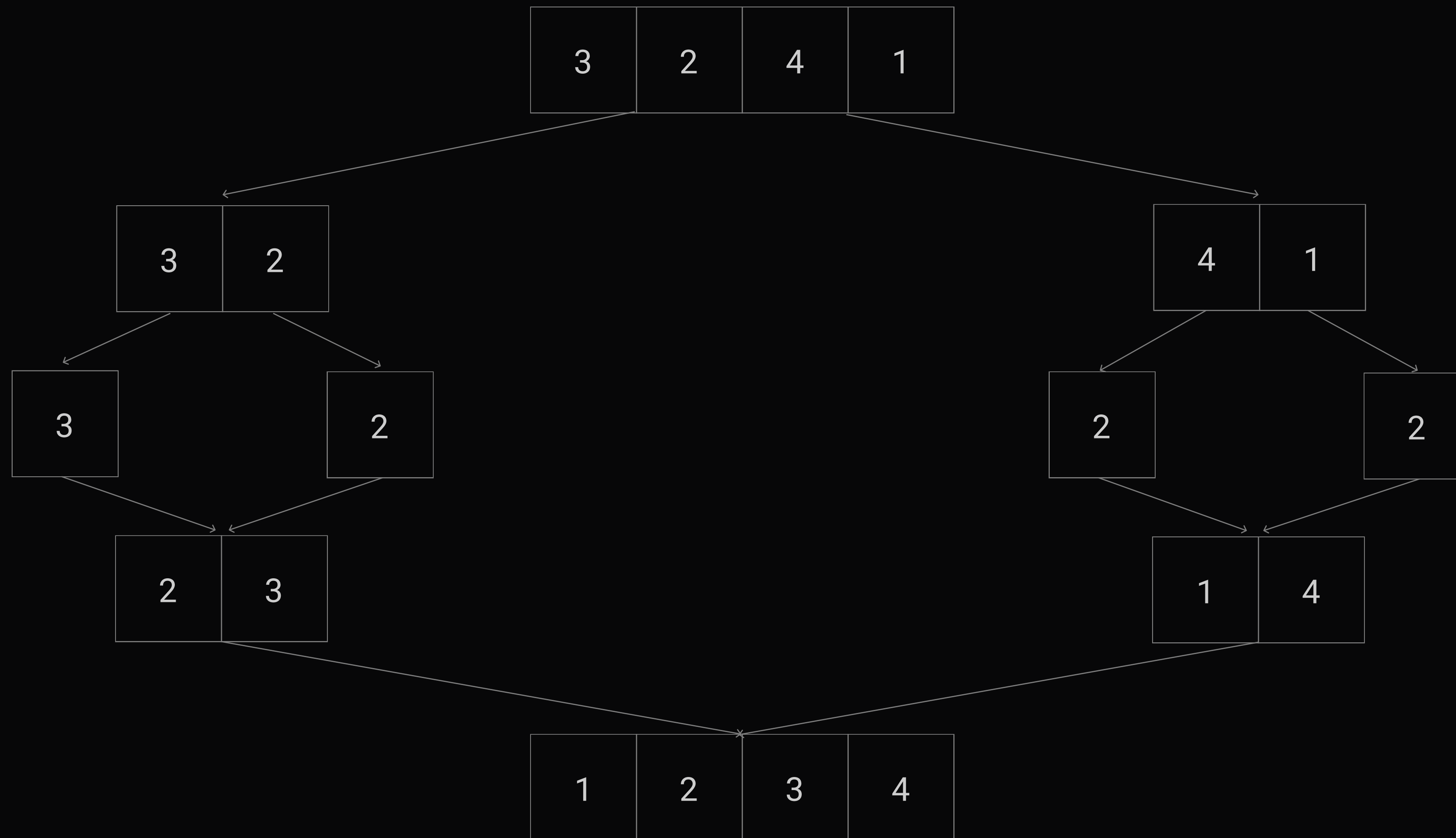
API



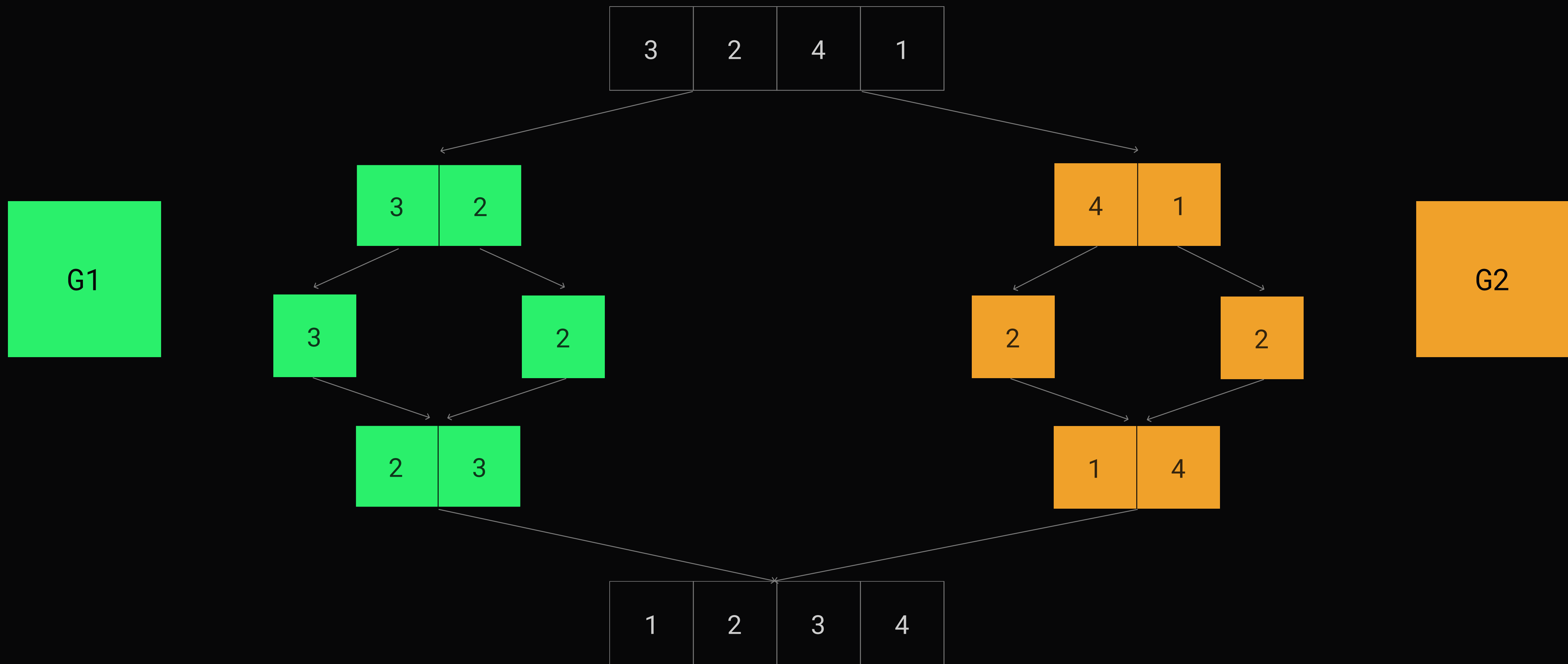
```
1 go func() {  
2     ...  
3 } ()  
4  
5 go fn()  
6 go obj.fn()
```


**ЗАЧЕМ НУЖНО
СОЗДАВАТЬ ГОРУТИНЫ?**

ПОСЛЕДОВАТЕЛЬНАЯ РЕАЛИЗАЦИЯ



ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ



Еще можно обрабатывать конкурентно пользовательские
запросы с использованием горутин

EXAMPLE

TCP server with panic

EXAMPLE

TCP server without panic

EXAMPLE

Never exit

EXAMPLE

Panic from other goroutine

ПОЧЕМУ Я ИСПОЛЬЗОВАЛ
LOG.PRINTLN(), А НЕ FMT.PRINTLN()?

ПОЧЕМУ Я ИСПОЛЬЗОВАЛ LOG.PRINTLN(), А НЕ FMT.PRINTLN()?

Операции в пакете `log` синхронизируются, поэтому сообщения, напечатанные двумя горутинами, не будут перепутаны/перемешаны в одной строке

**ПОСМОТРИМ НА НЮАНСЫ,
КОТОРЫЕ СЛЕДУЕТ УЧИТЫВАТЬ
ПРИ ЗАПУСКЕ ГОРУТИН**

EXAMPLE

Goroutine index

EXAMPLE

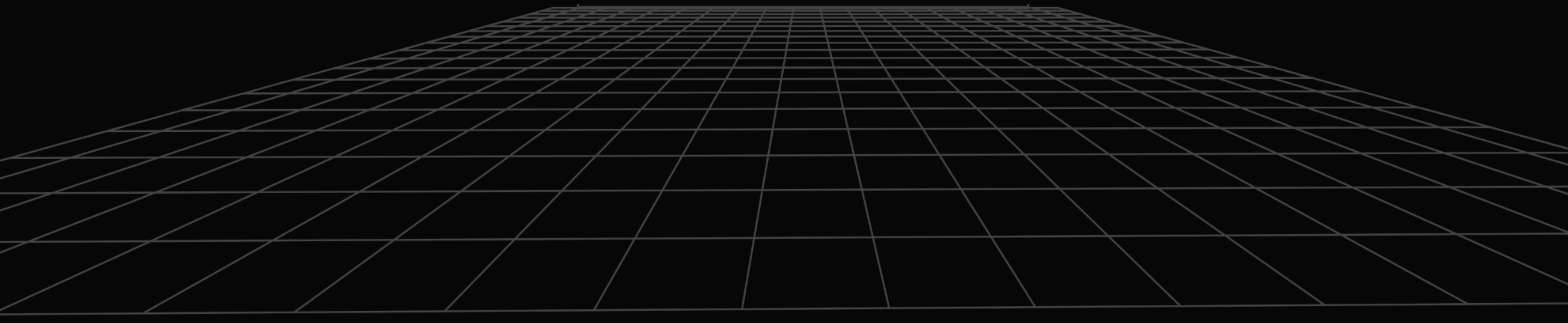
Endless loop

«So, it is not a problem for a Go program to maintain
tens of thousands goroutines at the same time, as long
as the system memory is sufficient»

**КТО ЗАНИМАЕТСЯ
ЗАПУСКОМ ГОРУТИН?**

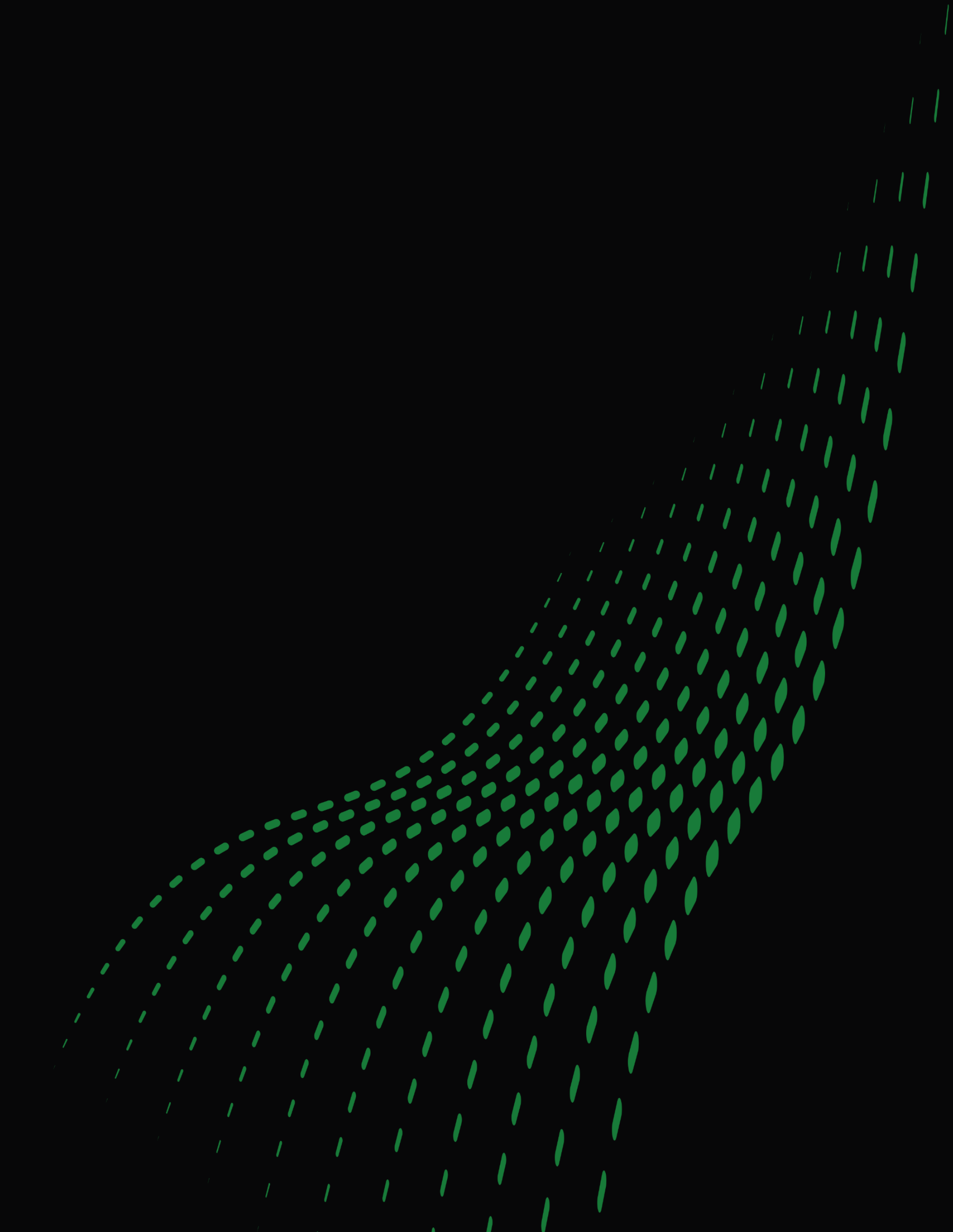
FAQ

Горутины



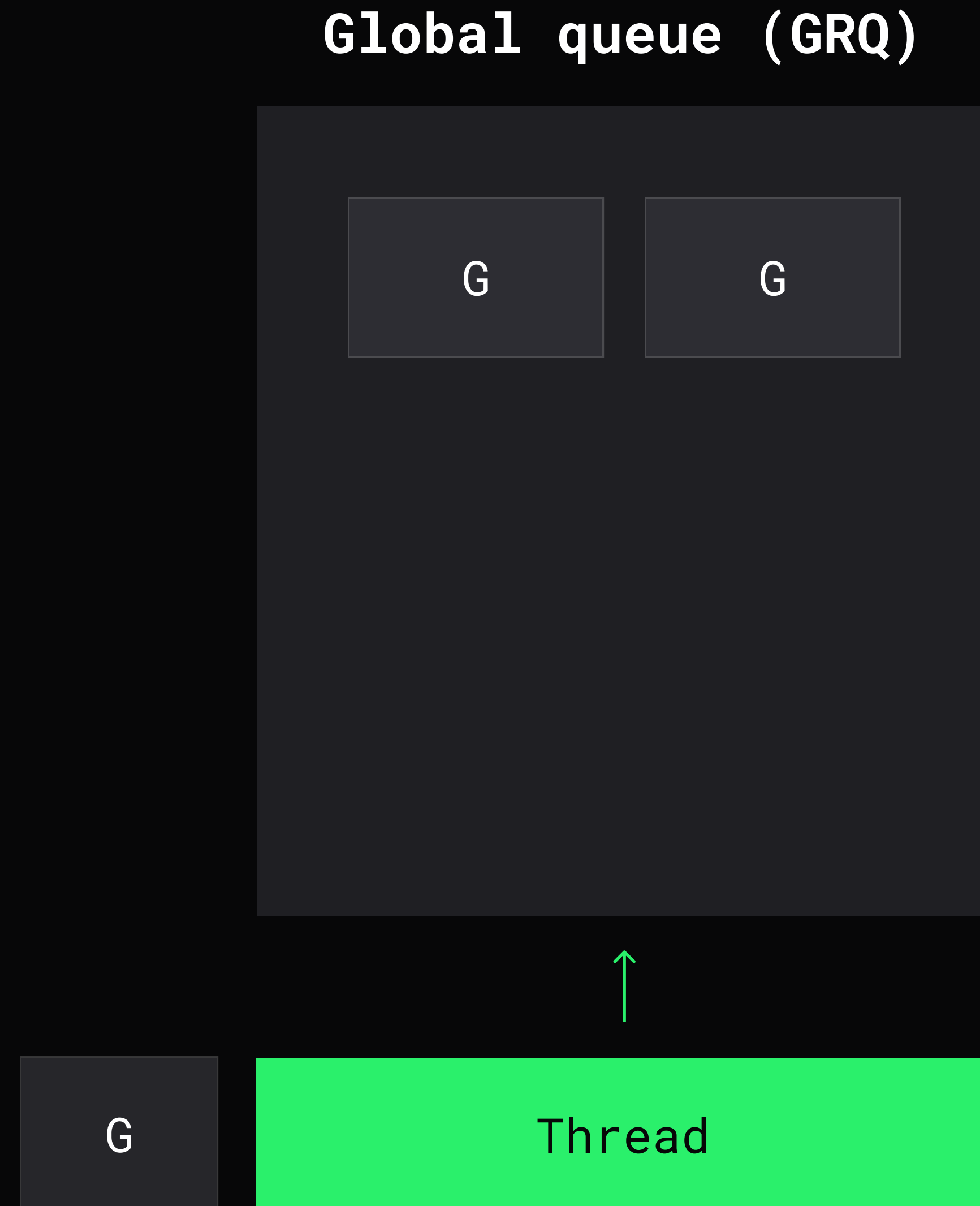
ПЛАНИРОВЩИК GO

СОСТОЯНИЕ ГОРУТИНЫ

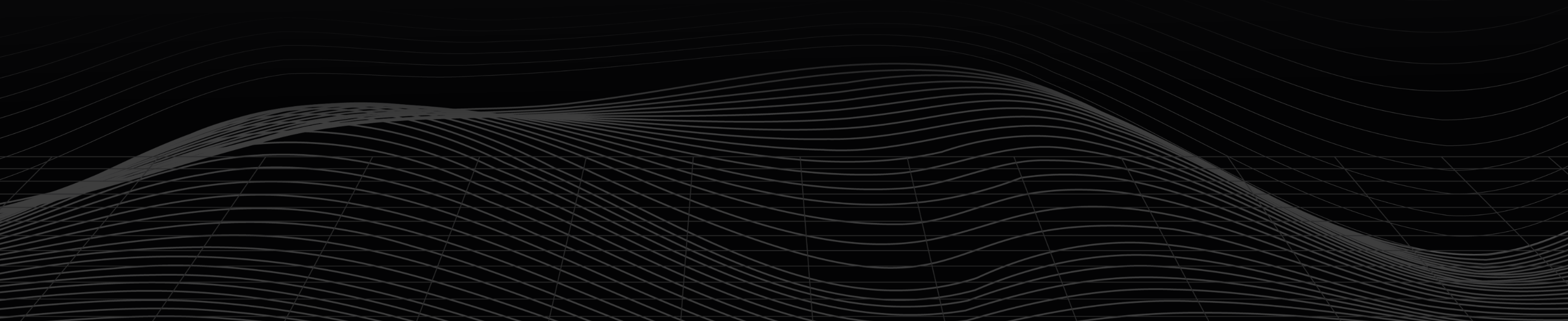
1. **Running** — выполняется
 2. **Runnable** — готова к выполнению
 3. **Waiting** — остановлена и ожидает чего-то
- 

N:1 МОДЕЛЬ

Выполняем N горутин
на одном потоке



**КАК И КОГДА БУДЕМ
ВЫТЕСНЯТЬ ГОРУТИНЫ?**



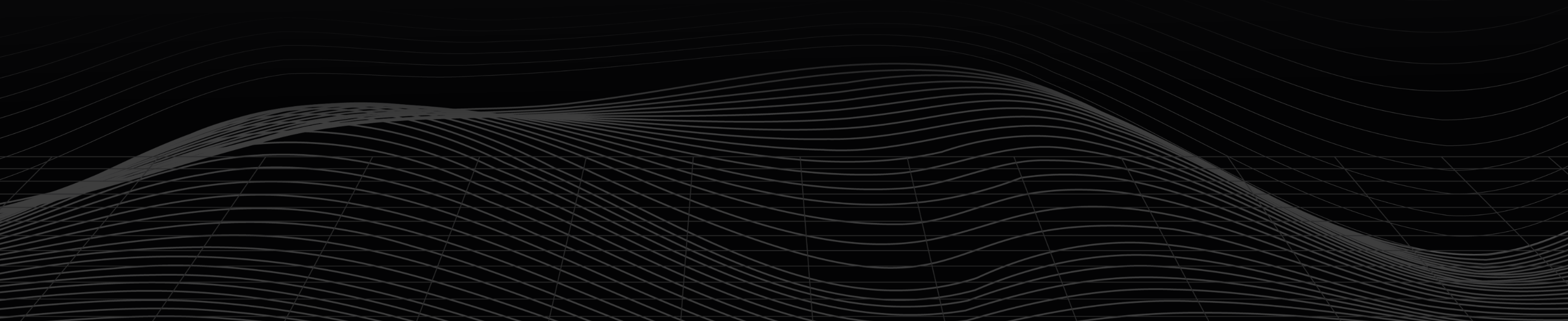
Компилятор добавит в исходный код места для переключения контекста

- і например в момент проверки необходимого пространства для стека горутин (кооперативная многозадачность)

подтема №1

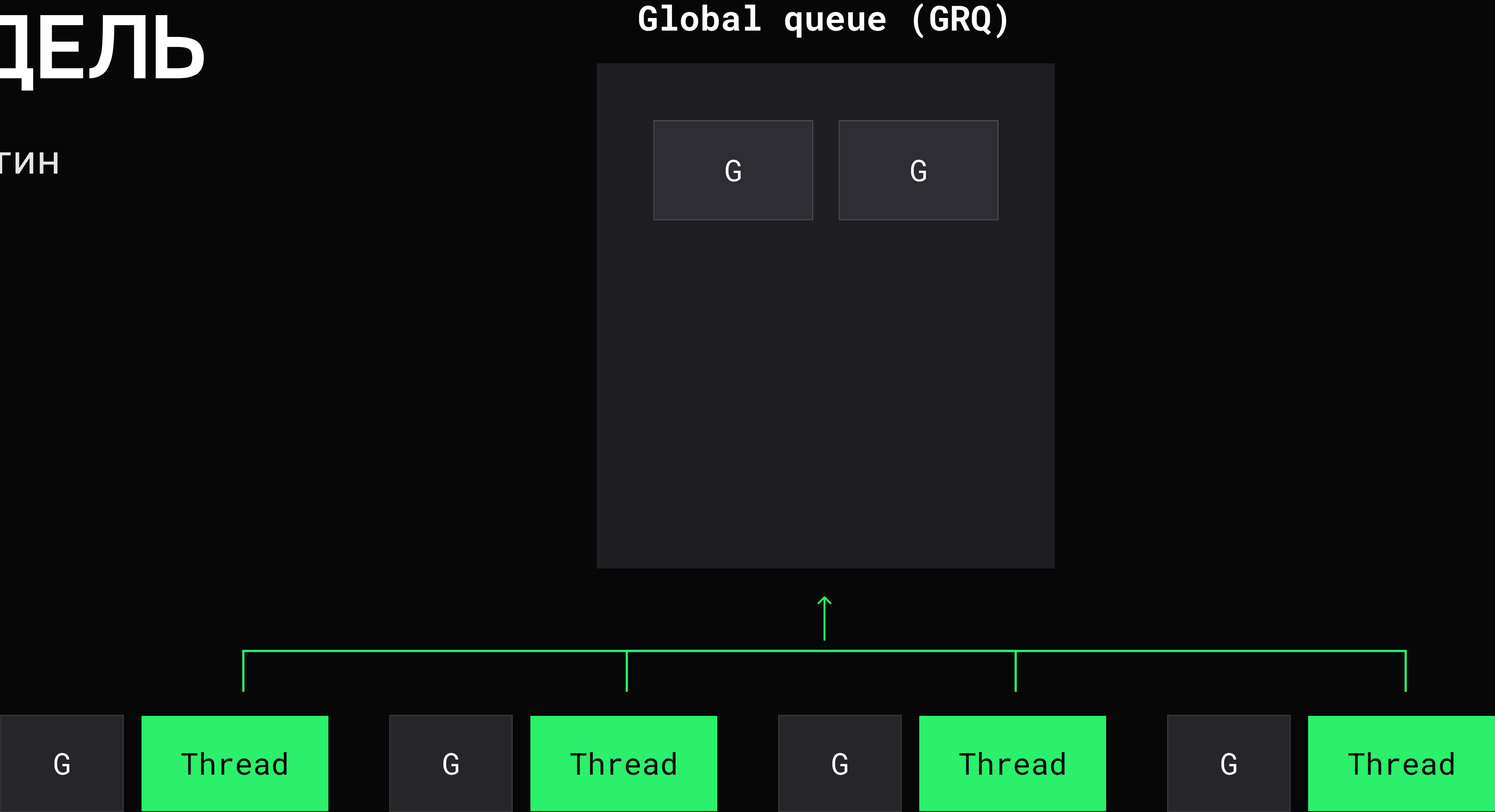
МАСШТАБИРОВАНИЕ

**КАК БЫТЬ СЕРВЕРАМ
С МНОГОЯДЕРНЫМИ
ПРОЦЕССОРАМИ?**

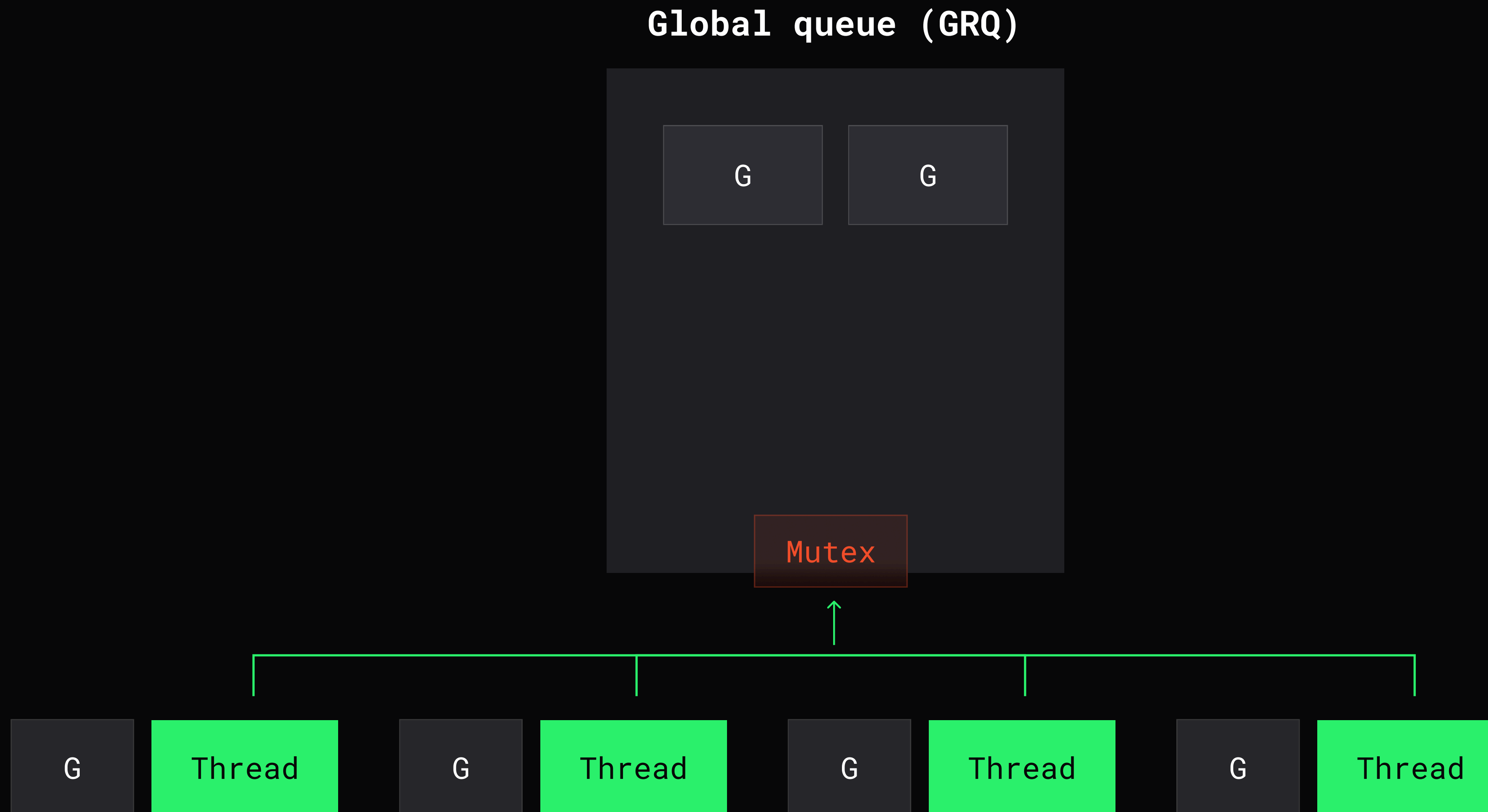


N:M МОДЕЛЬ

Выполняем **N** горутин
на **M** потоках

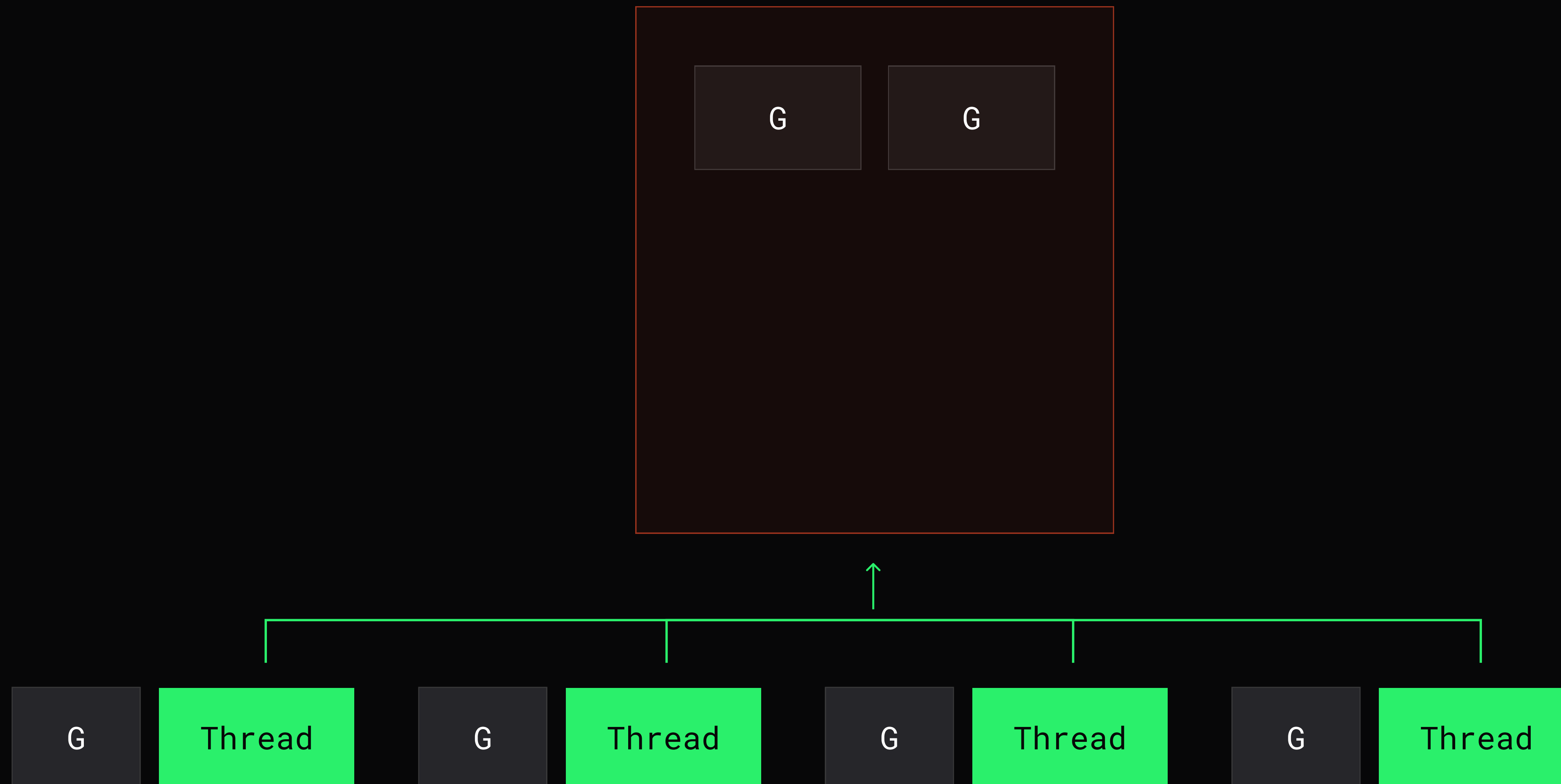


NOT SCALABLE



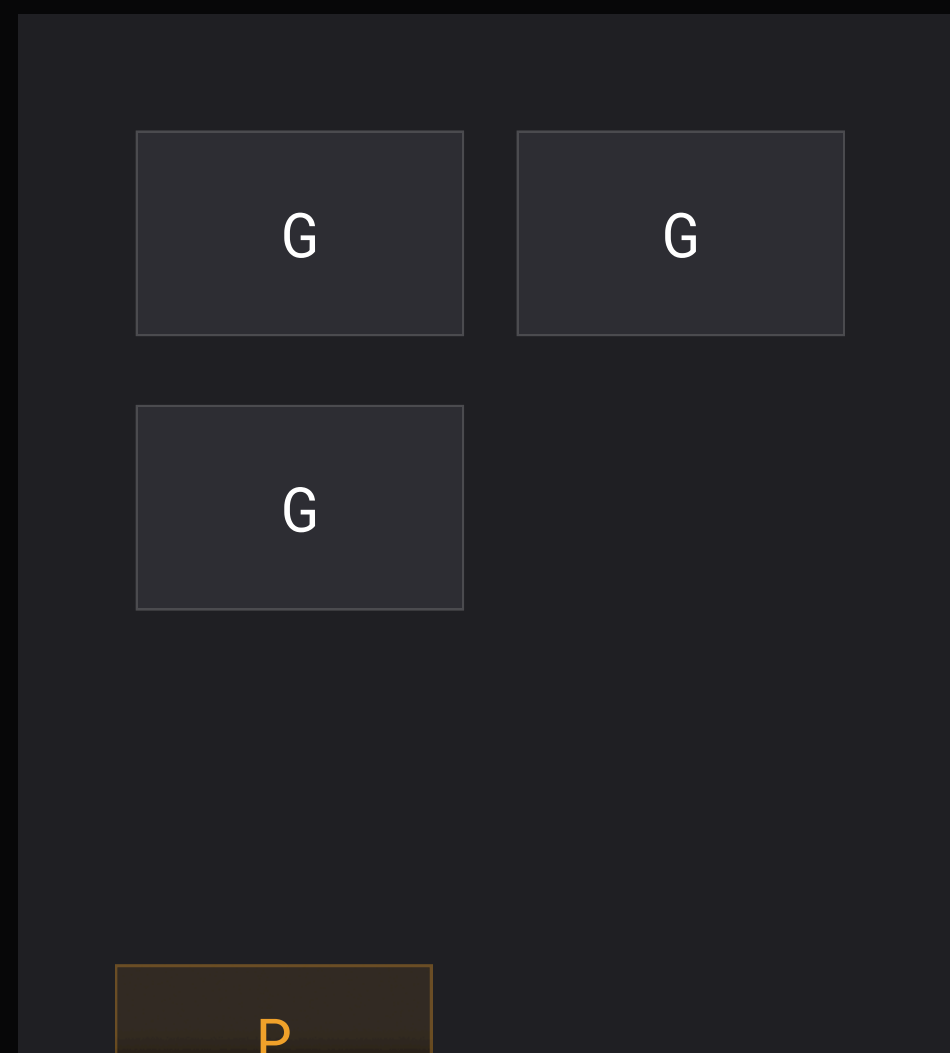
NOT SCALABLE

Global **lock-free** queue (GRQ)

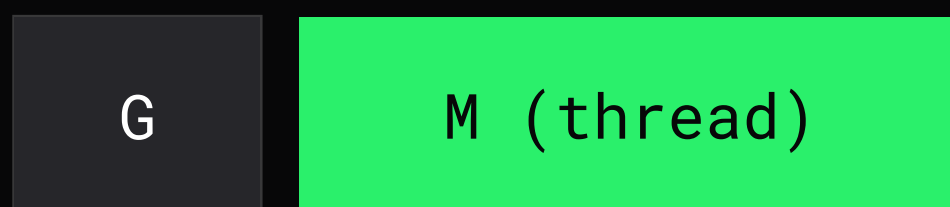
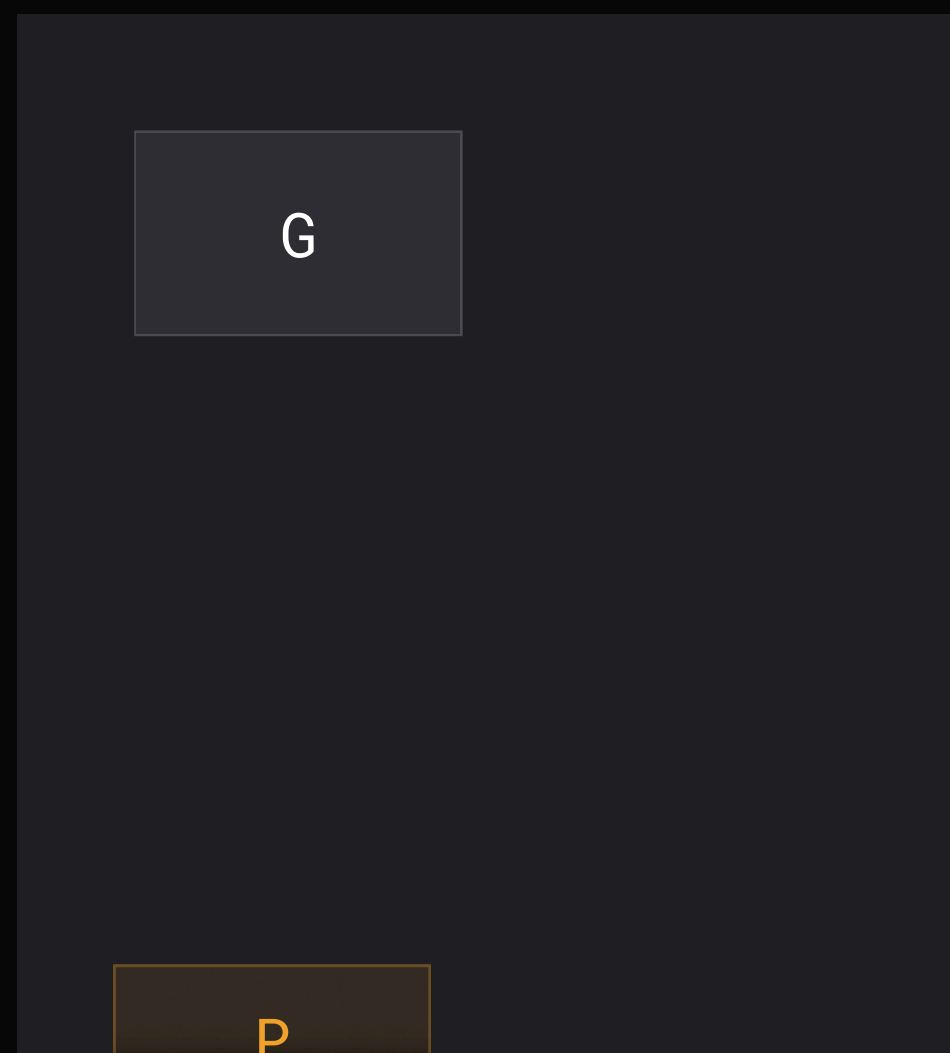


ЛОКАЛЬНЫЕ ОЧЕРЕДИ

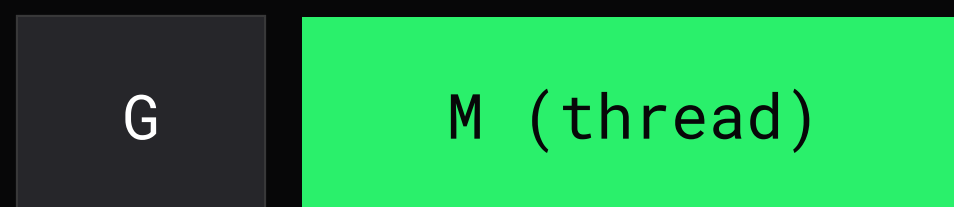
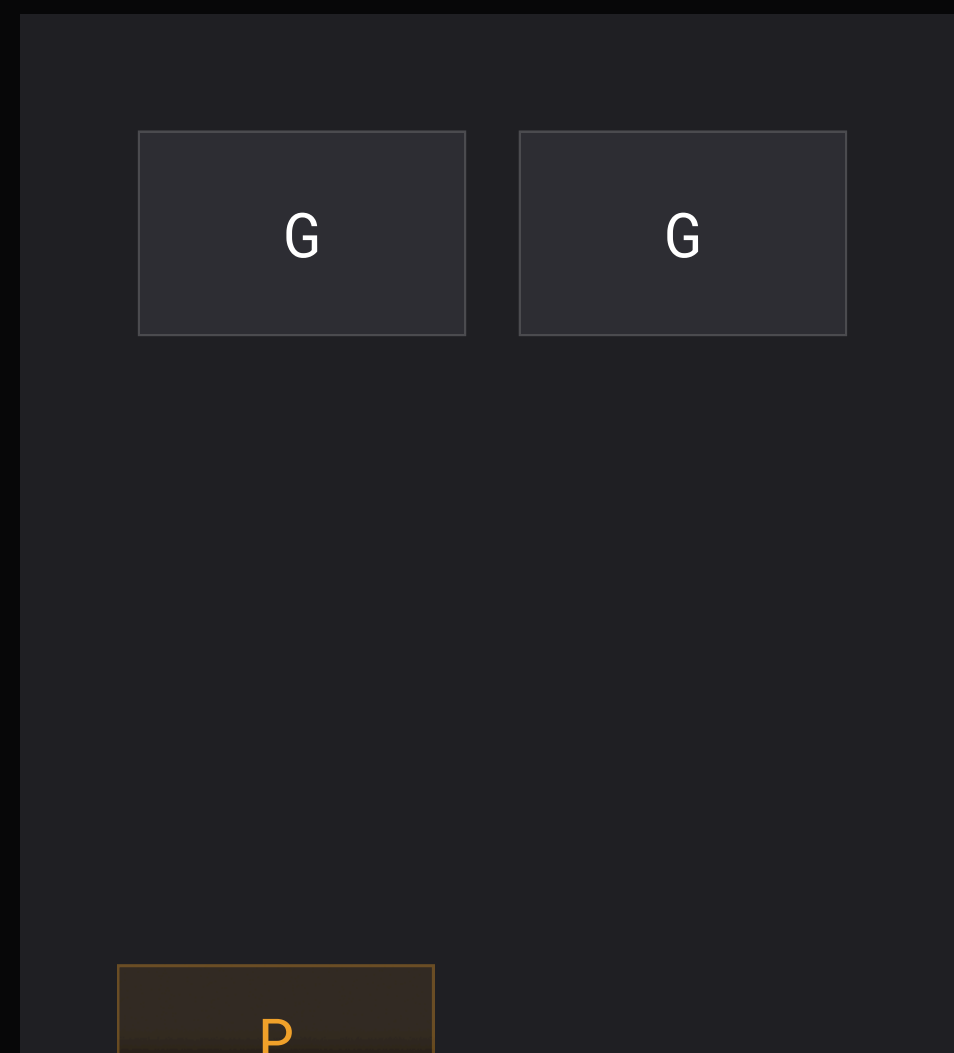
Local queue (LRQ=256)



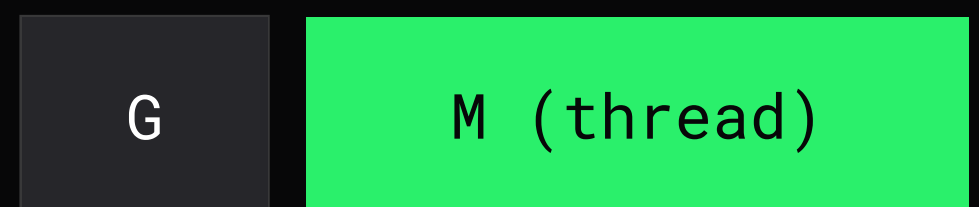
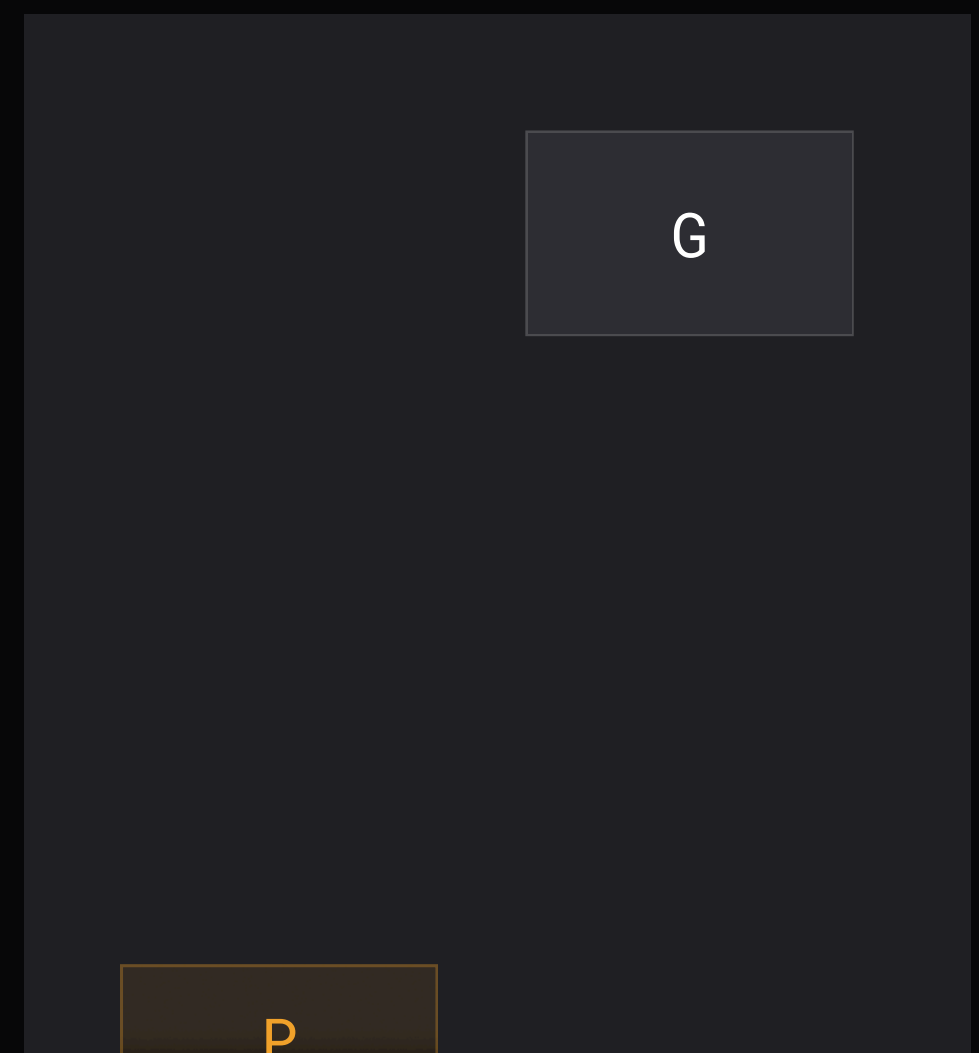
Local queue (LRQ=256)



Local queue (LRQ=256)



Local queue (LRQ=256)



GMP МОДЕЛЬ

G - Goroutine (что исполняем)

M - Machine (где исполняем)

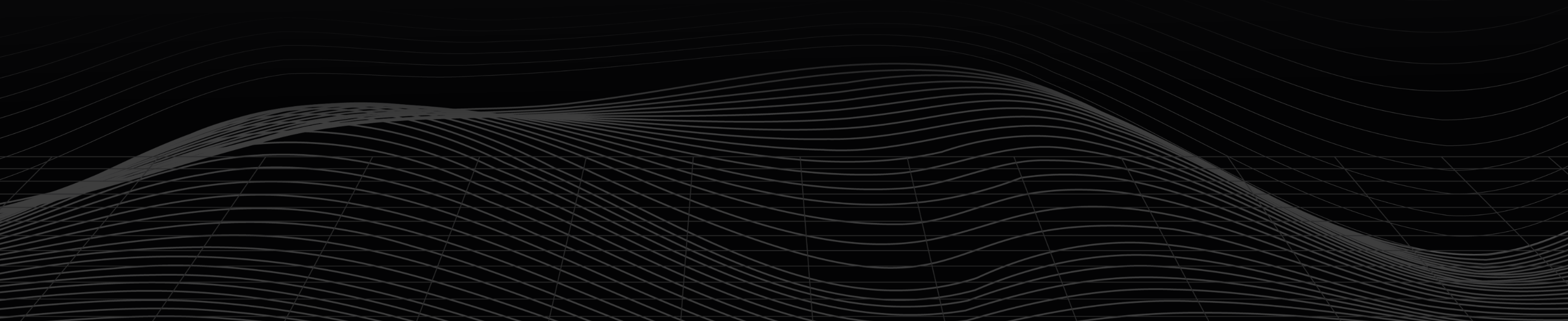
P - Processor (права и ресурсы для исполнения)

EXAMPLE

GMP model

По умолчанию количество P берётся из значения
переменной среды **GOMAXPROCS** и равно количеству
логических ядер компьютера (стоит быть внимательным)

**В КАКУЮ ИЗ ОЧЕРЕДЕЙ
ДОБАВЛЯТЬ НОВУЮ ГОРУТИНУ?**





**А ЧТО ДЕЛАТЬ, ЕСЛИ
В ОЧЕРЕДИ ЗАКОНЧАТСЯ
ГОРУТИНЫ?**

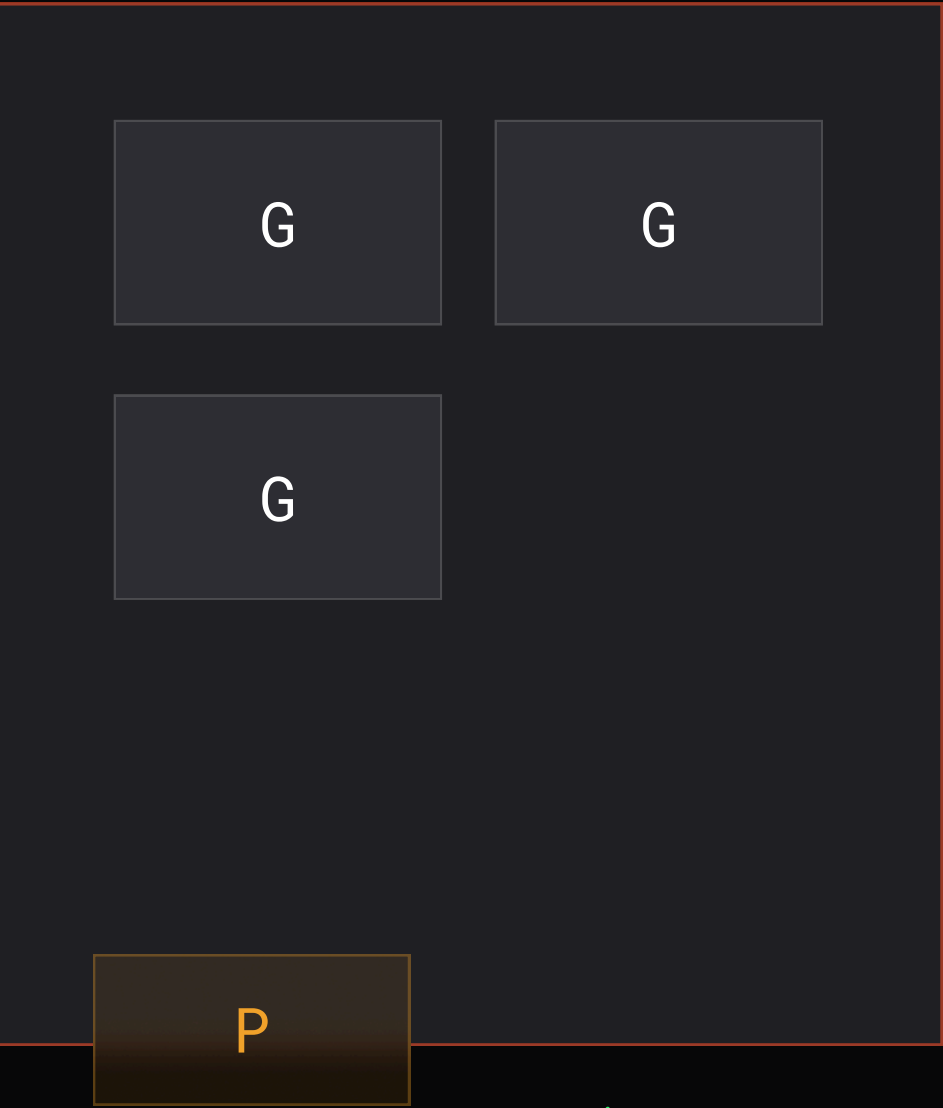
WORK SHARING



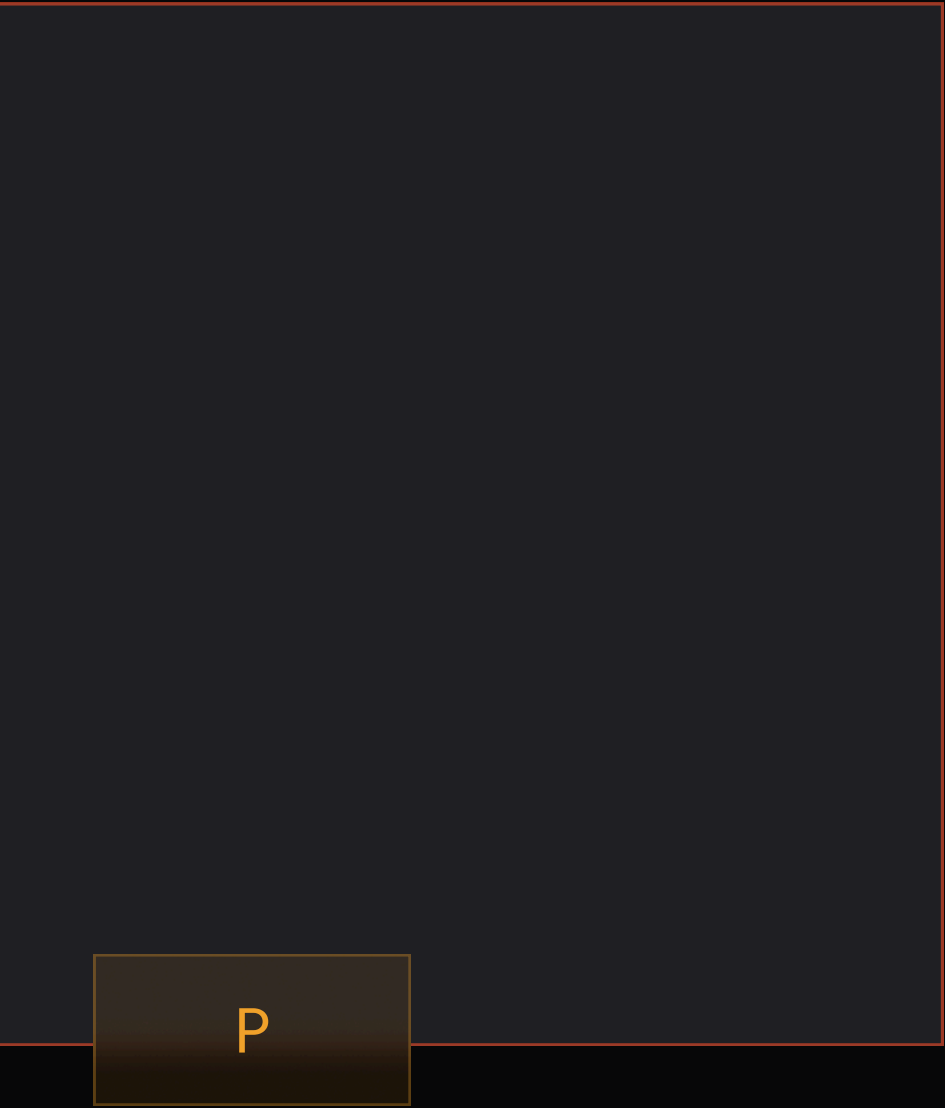
WORK STEALING



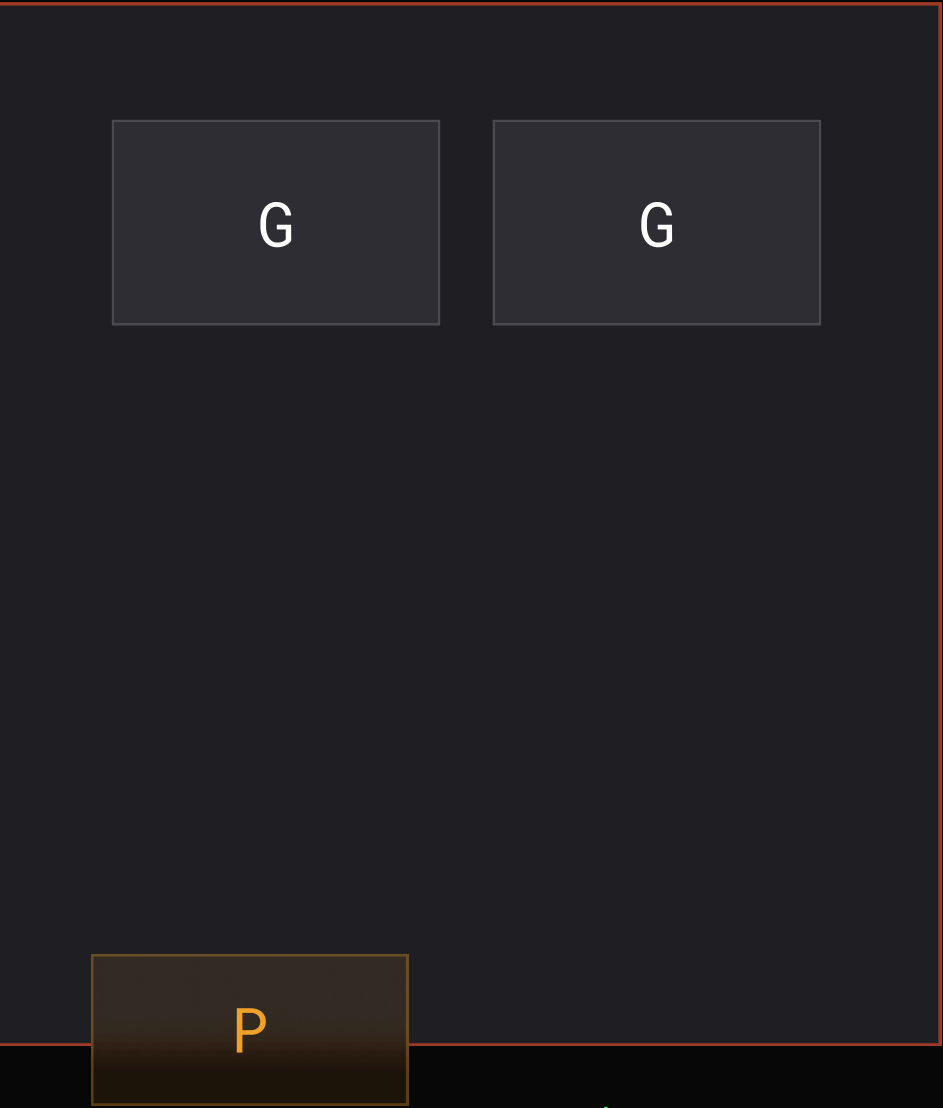
Local **lock-free** queue (LRQ)



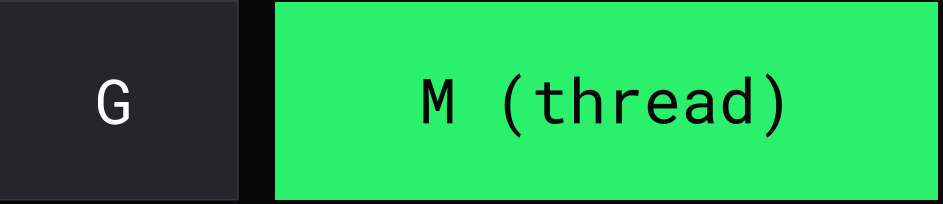
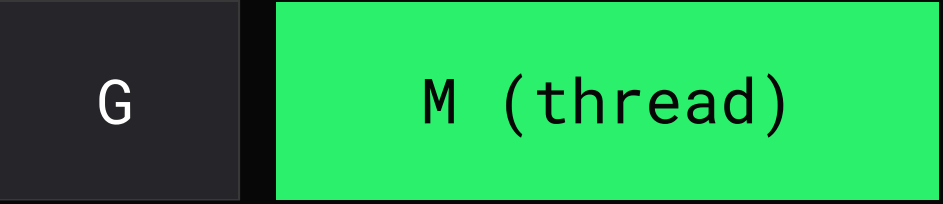
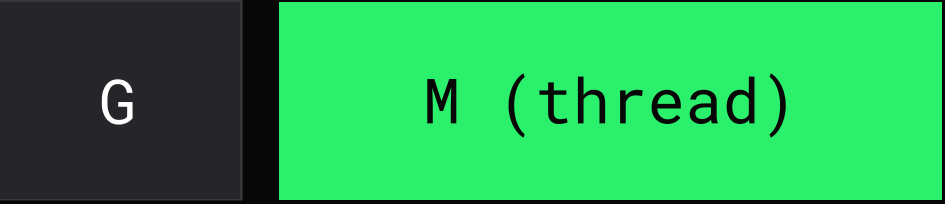
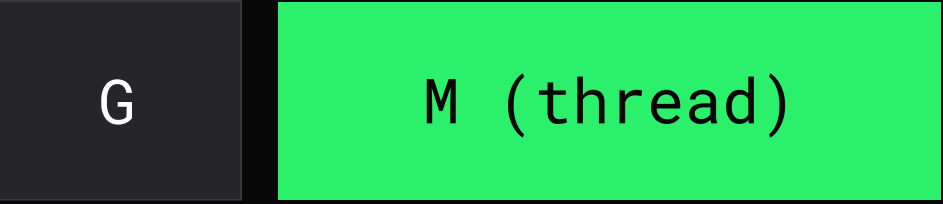
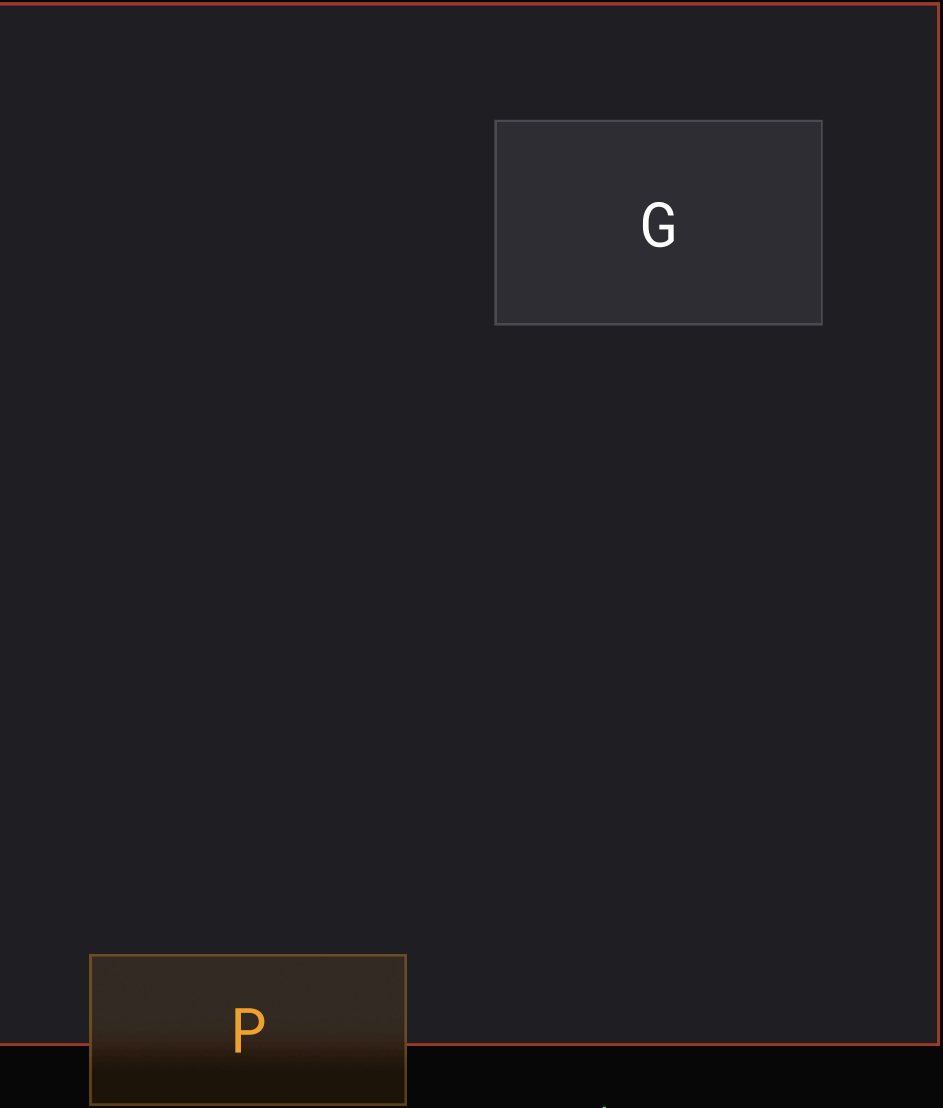
Local **lock-free** queue (LRQ)



Local **lock-free** queue (LRQ)



Local **lock-free** queue (LRQ)

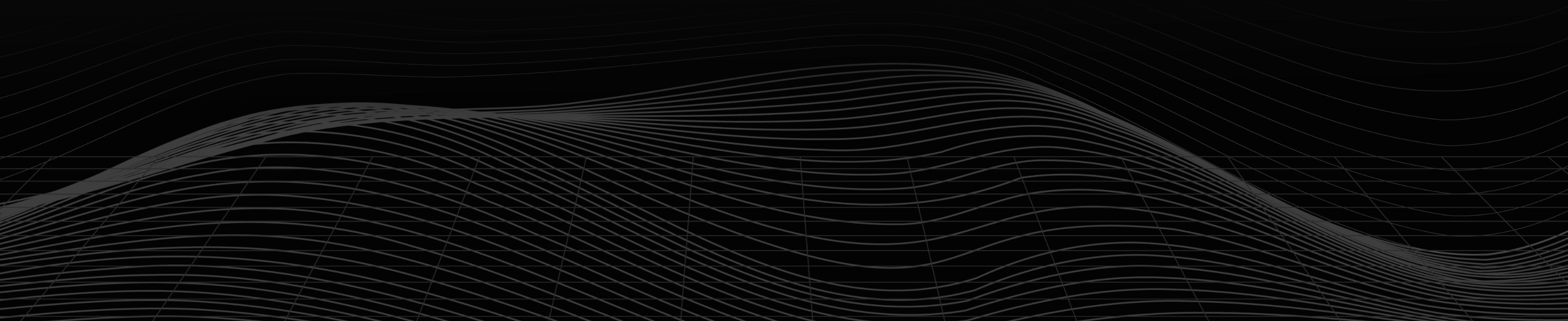


**У КАКОЇ ІМЕННО
ОЧЕРЕДІ КРАСТЬ?**

У КАКОЙ ИМЕННО ОЧЕРЕДИ КРАСТЬ?

Выбираем очередь **рандомно**

**ЧТО ДЕЛАТЬ, ЕСЛИ В ДРУГОЙ
ОЧЕРЕДИ ТОЖЕ НЕТ ГОРУТИН?**



ЧТО ДЕЛАТЬ, ЕСЛИ В ДРУГОЙ ОЧЕРЕДИ ТОЖЕ НЕТ ГОРУТИН?

Пробуем украсть **четыре** раза

**СКОЛЬКО КРАСТЬ ИЗ ДРУГОЙ
ЛОКАЛЬНОЙ ОЧЕРЕДИ?**

СКОЛЬКО КРАСТЬ ИЗ ДРУГОЙ ЛОКАЛЬНОЙ ОЧЕРЕДИ?

Крадем **половину горутин**
из другой локальной очереди

ПЕРЕРЫВ 5 МИНУТ

подтема №2

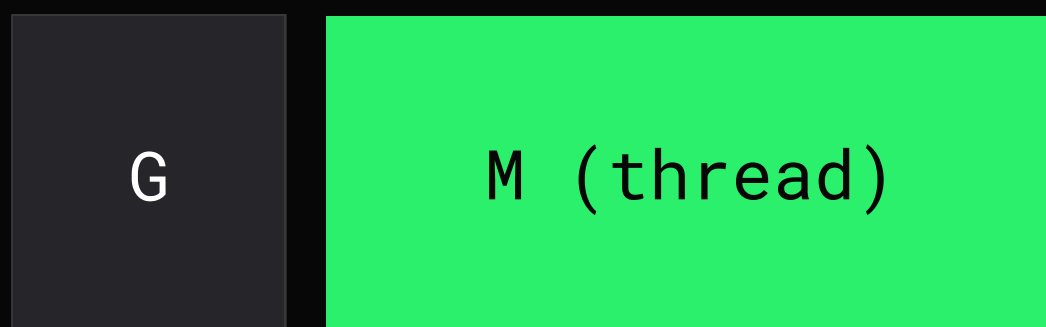
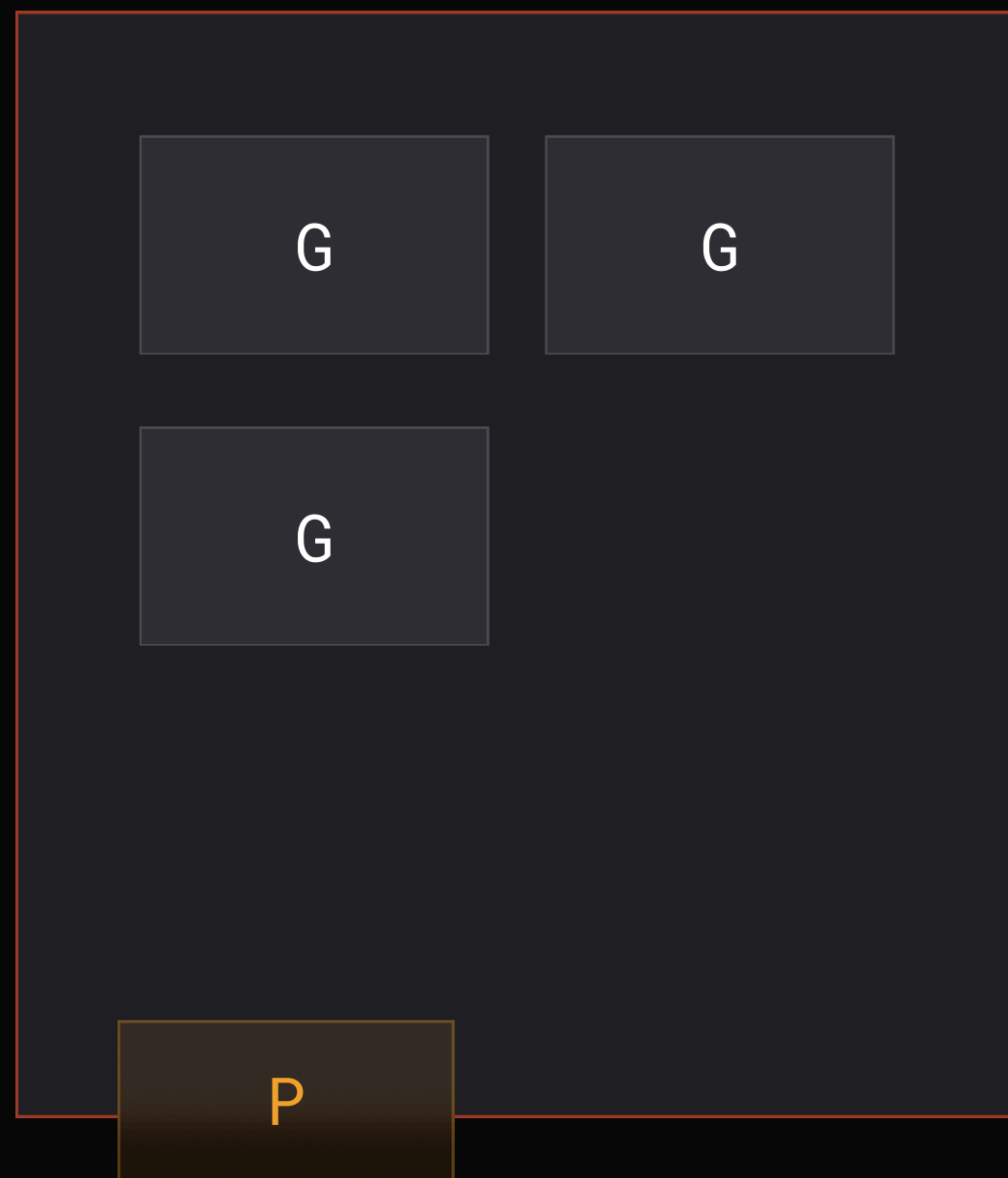
SYSCALLS

КАК БЫТЬ С SYSCALL-АМИ?

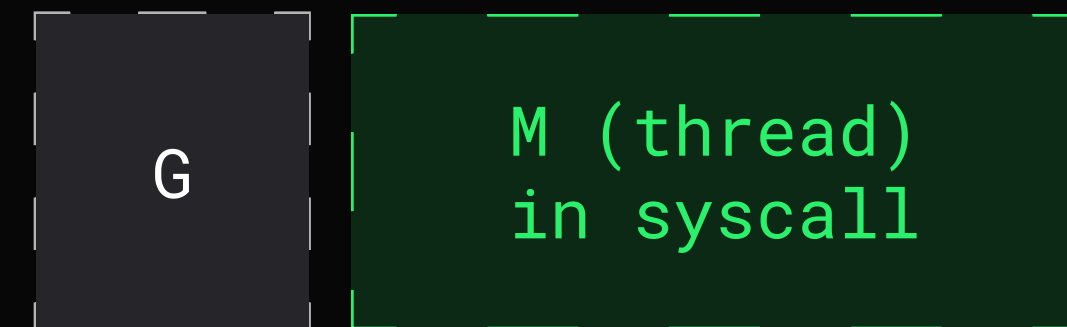
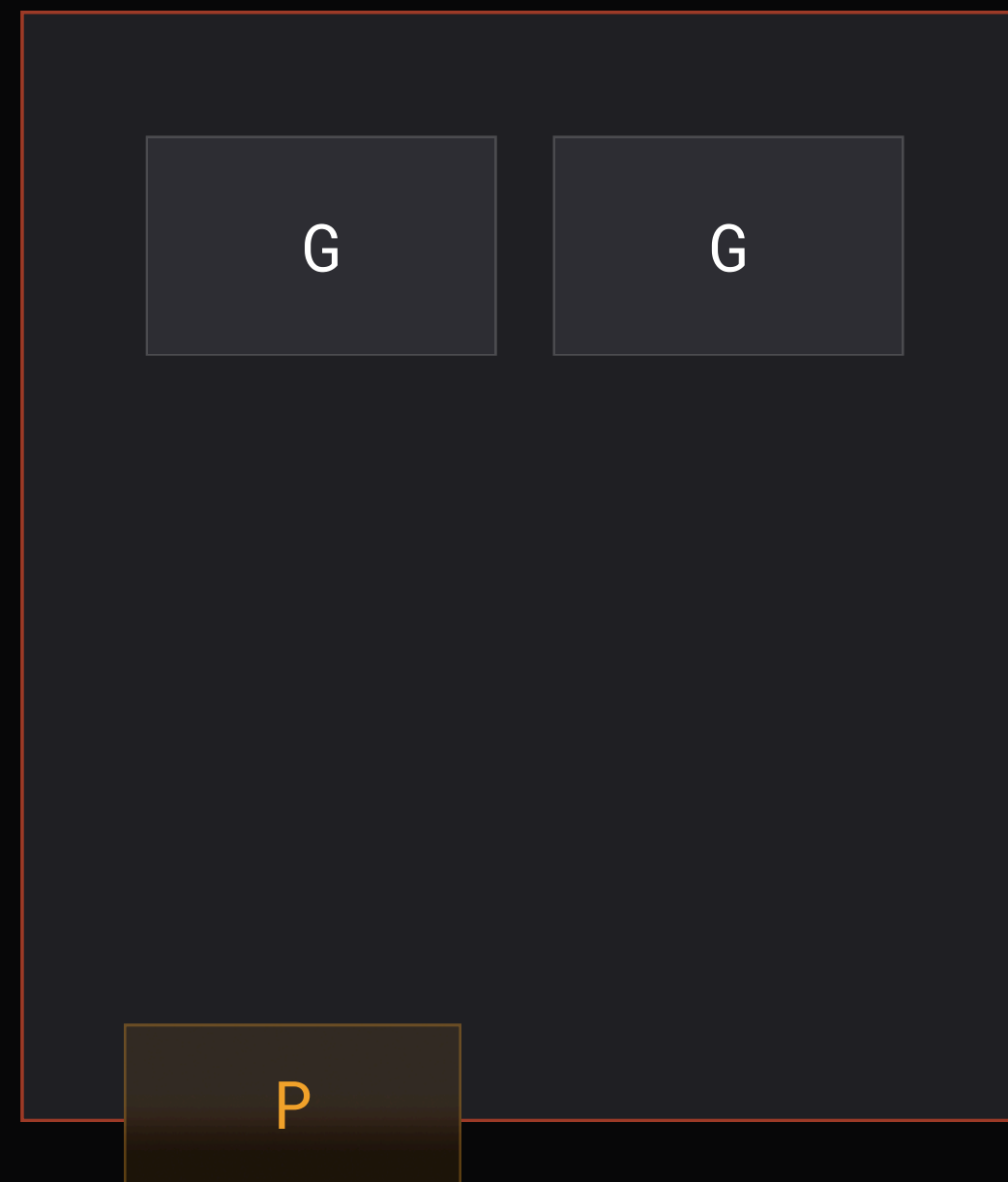


HANDOFF

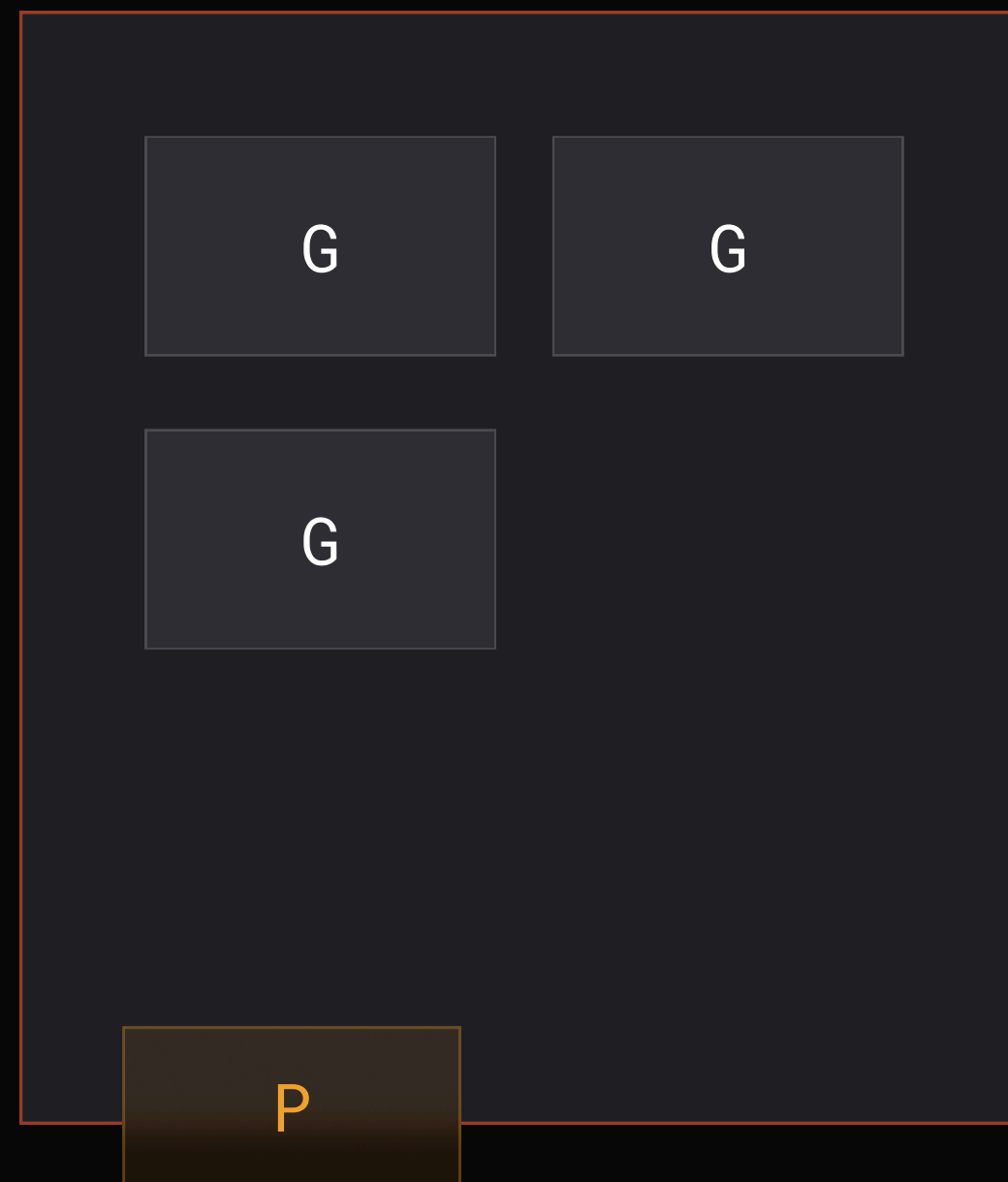
Local **lock-free** queue (LRQ)



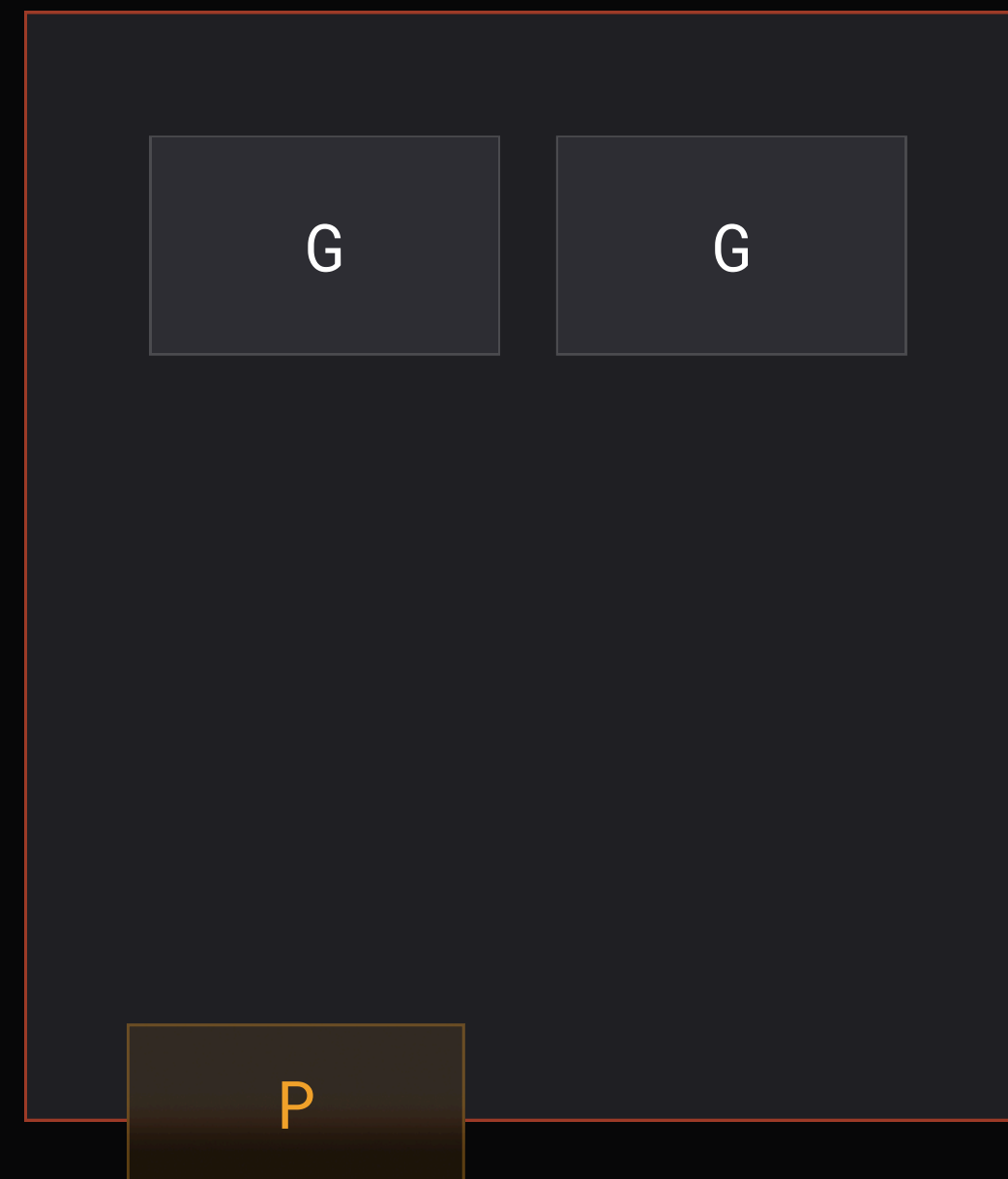
Local **lock-free** queue (LRQ)



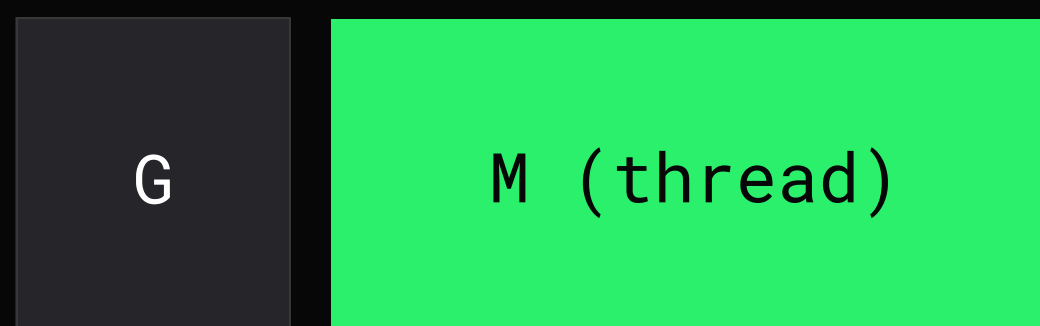
Local **lock-free** queue (LRQ)



Local **lock-free** queue (LRQ)

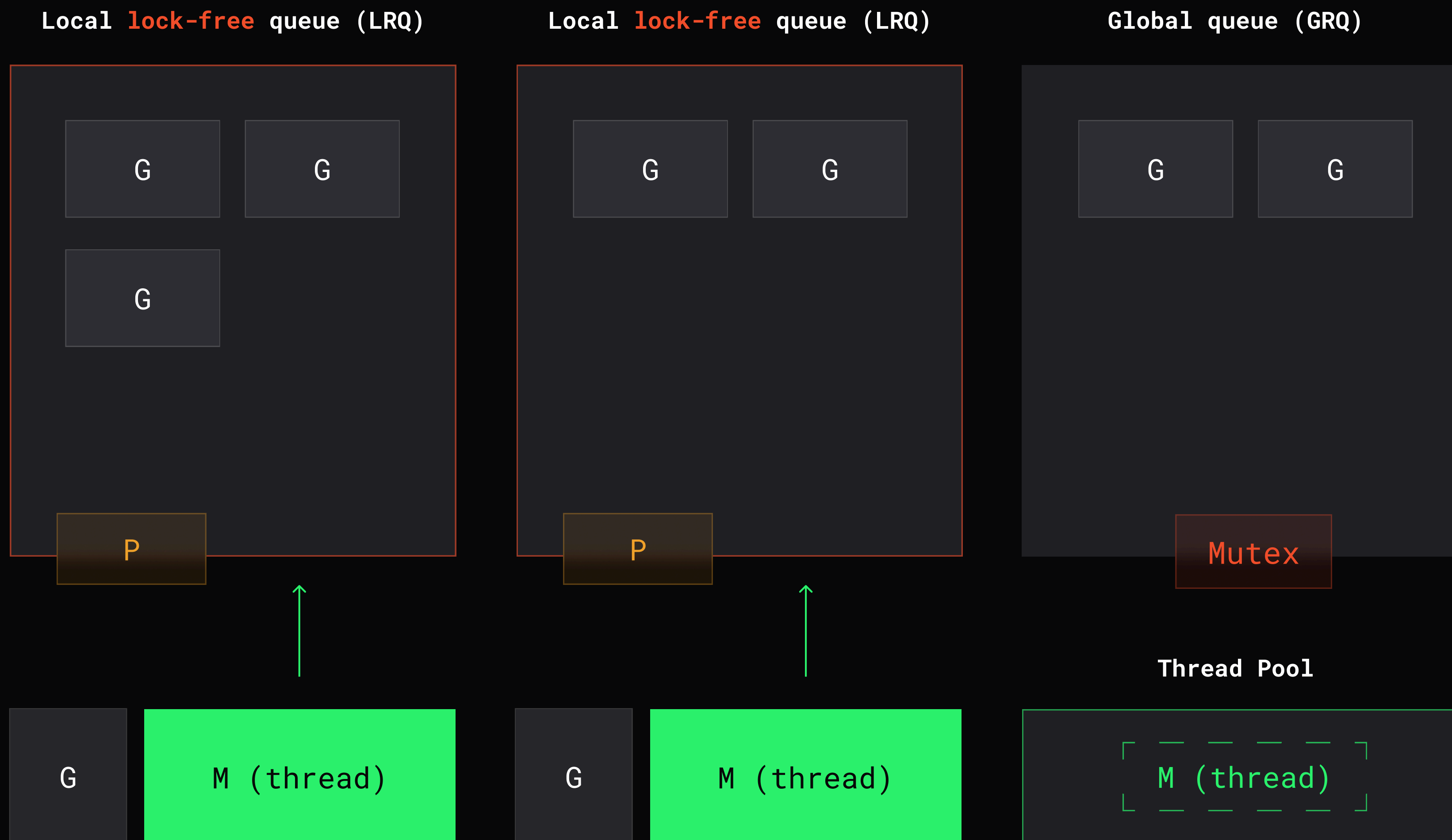


Thread Pool



**КУДА ДЕВАТЬ ГОРУТИНУ ПОСЛЕ
ВЫПОЛНЕНИЯ SYSCALL-A?**

ГЛОБАЛЬНАЯ ОЧЕРЕДЬ



Если в моем web сервисе открыто
10 000 соединений, то потенциально в моем приложении
может быть создано более 10 000 потоков?

И если размер стека потока в среднем 8 МБ, то тогда
потребуется примерно 80 ГБ оперативной памяти для этого?

NETPOLLER

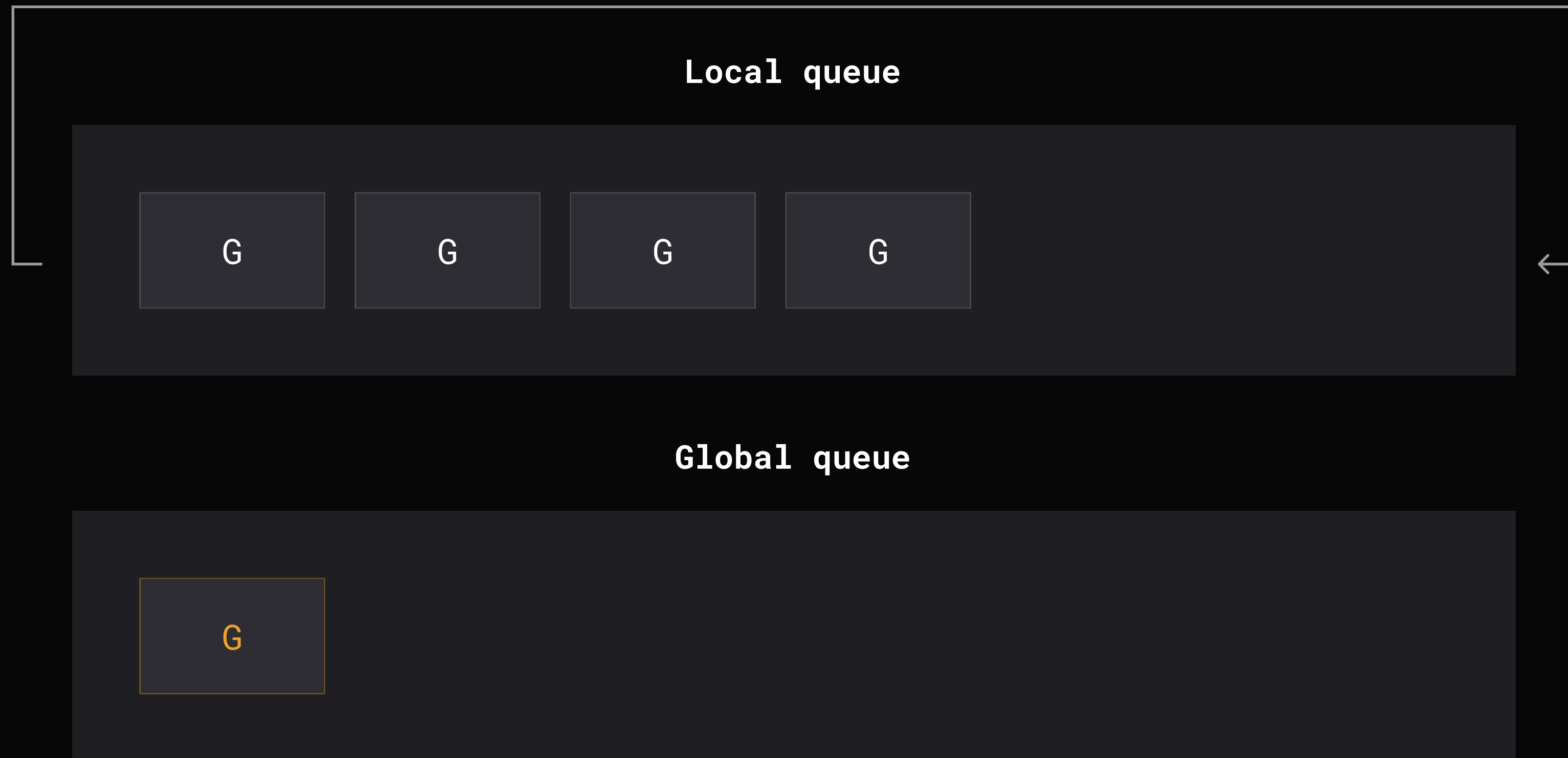


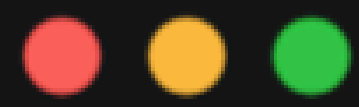
подтема №3

ОЧЕРЕДИ

КАК БЫТЬ С ГОРУТИНАМИ В ГЛОБАЛЬНОЙ ОЧЕРЕДИ?

Ведь они там будут голодать





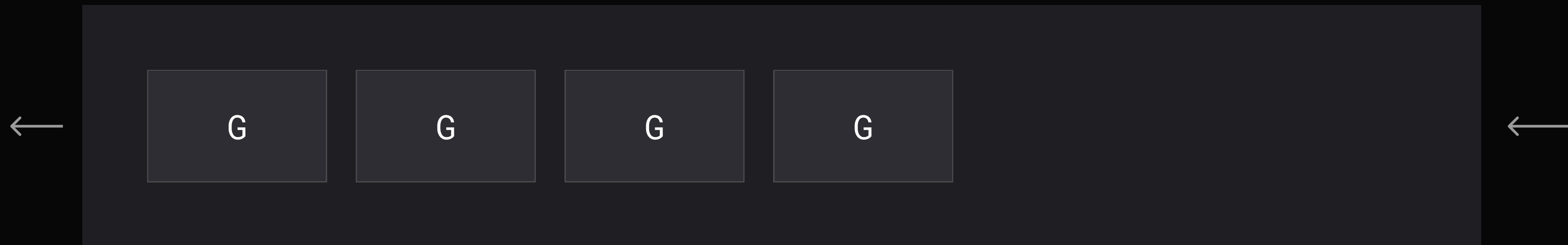
```
1 runtime.schedule() {  
2 // only 1/61 of the time, check the global runnable queue for a G.  
3 // if not found, check the local queue.  
4 // if not found,  
5 // try to steal from other Ps.  
6 // if not, check the global runnable queue.  
7 // if not found, poll network.  
8 }
```


А КАК БЫТЬ С КЕШАМИ, КОГДА НЕСКОЛЬКО ГОРУТИН БЕРУТ ОБЩУЮ БЛОКИРОВКУ

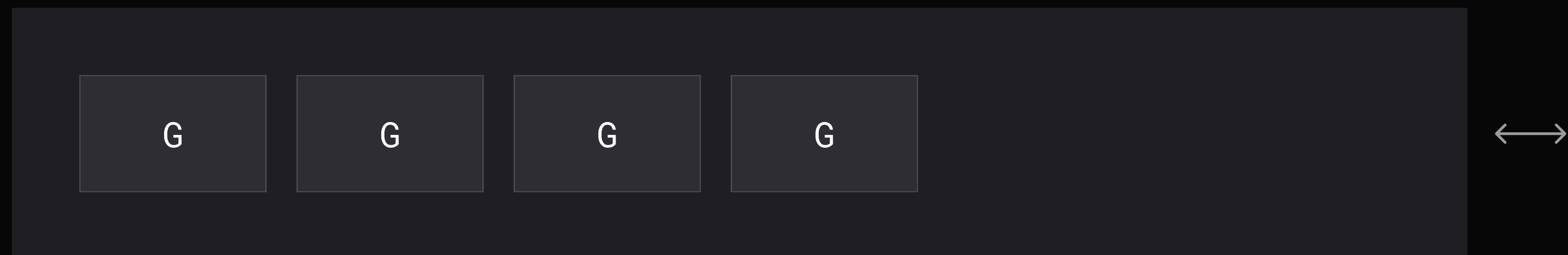
или отправляют данные
друг другу через канал?

FAIRNESS

FIFO



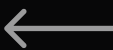
LIFO



FIFO



LIFO



КАК БЫТЬ С ЦИКЛАМИ ПРИ ОБЩЕНИИ ГОРУТИН ЧЕРЕЗ НЕСКОЛЬКО КАНАЛОВ?

Когда две горуты ставят
друг друга в LIFO?

Считаем время непрерывной работы и в случае превышения
временного лимита – вытесняем из LIFO

подтема №4

ЦИКЛЫ

**КАК БЫТЬ С БЕСКОНЕЧНЫМИ
ЦИКЛАМИ ИЛИ CPU-BOUND
ЗАДАЧАМИ?**

Terminal: Concurrency × + ∨



ASYNCHRONOUS PREEMPTION

Как только обнаруживается горутина, работающая более 10мс, в поток передается сигнал для ее вытеснения (SIGURG). **Этим всем занимается `sysmon`**

EXAMPLE

Async preemptible

**КАК ЗАКРЕПИТЬ ГОРУТИНУ
ЗА КОНКРЕТНЫМ ПОТОКОМ?**



```
1 runtime.LockOSThread( )  
2 runtime.UnlockOSThread( )
```

**КАК ПРИНУДИТЕЛЬНО
ЗАВЕРШИТЬ ГОРУТИНУ?**



```
1 runtime.Goexit()
```


**КОГДА ЗАВЕРШАЕТСЯ
ВЫПОЛНЕНИЕ ПРОГРАММЫ НА GO?**

**ЧТО БУДЕТ С ГОРУТИНАМИ, КОГДА
ПРОГРАММА ЗАВЕРШИТСЯ?**

EXAMPLE

Runtime exit

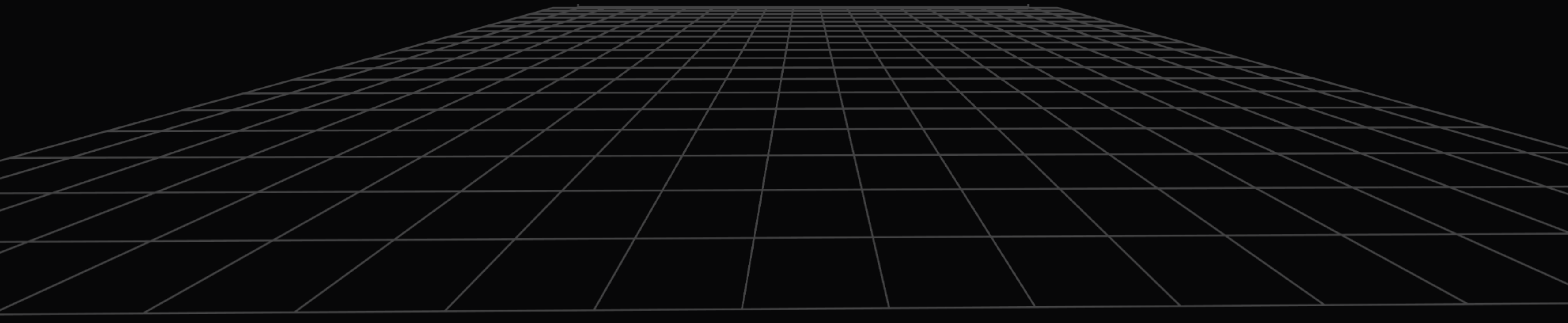
Calling Goexit from the main goroutine terminates that goroutine without func main returning

Since func main has not returned, the program continues execution of other goroutines

-  If all other goroutines exit, the program crashes. It crashes if called from a thread not created by the Go runtime

FAQ

Планировщик Go



ПОЖАЛУЙСТА, ЗАПОЛНИ ОПРОС О ЗАНЯТИИ

Ссылка в чате и в группе участников

