MAP

EXAMPLE

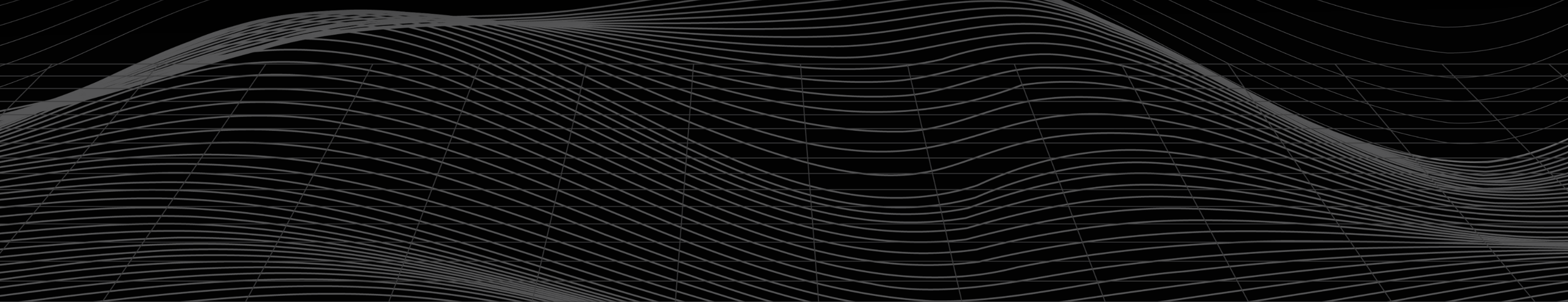
Concurrent map writes

GO 1.8

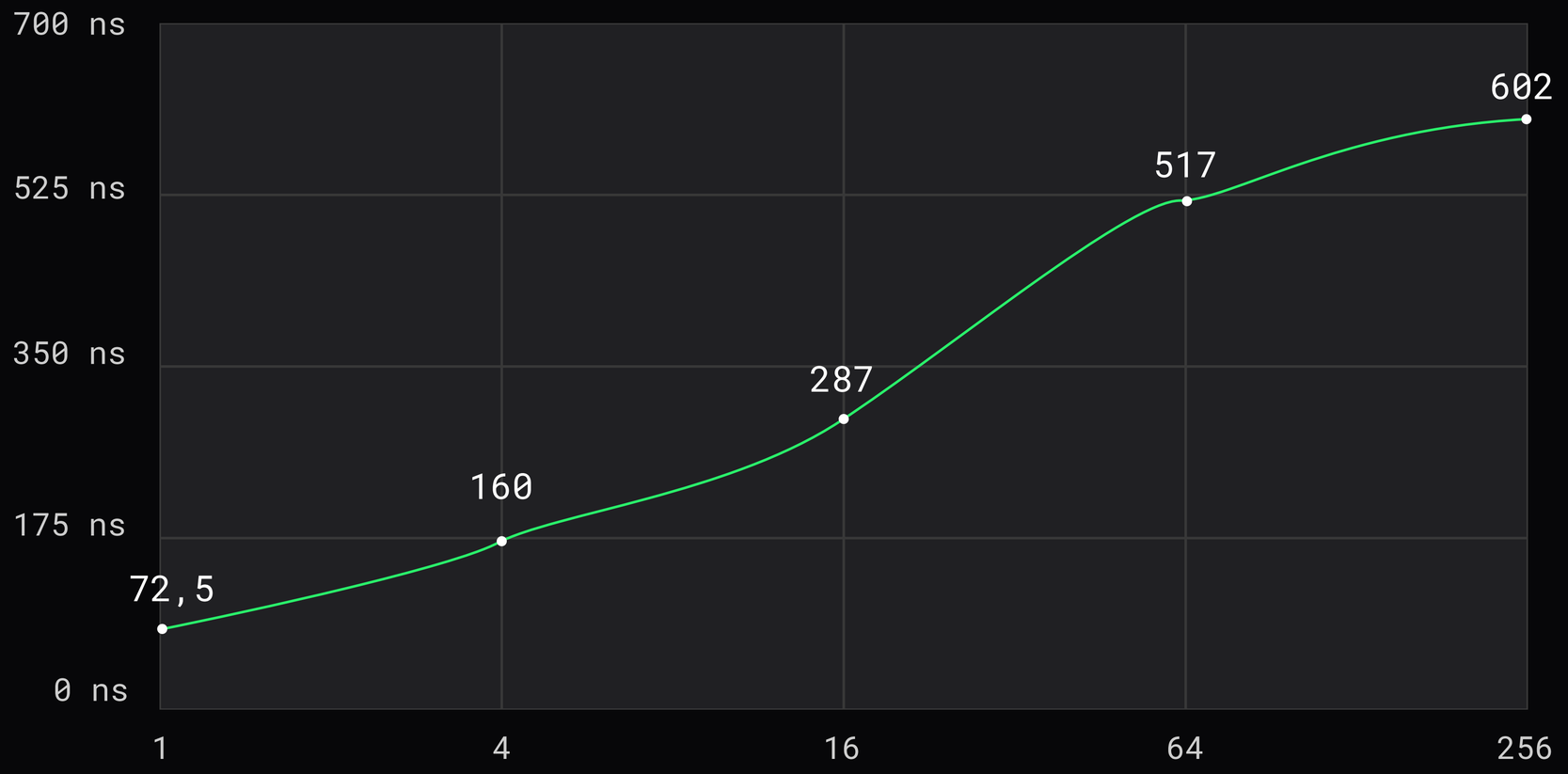
Maps are not safe for concurrent use: it’s not defined what happens when you read and write to them simultaneously. If you need to read from and write to a map from concurrently executing goroutines, the accesses must be mediated by some kind of synchronization mechanism

EXAMPLE

RWMutex with map



Бенчмарк показывает почти **8**-кратное увеличение времени на **RLock/RUnlock** на **64**-ядерной машине по мере увеличения количества горутин



Terminal: Concurrency

CACHE CONTENTION

Конфликт в кэше – ситуация, когда несколько потоков (горутин) пытаются читать или записывать данные, находящиеся в одной и той же кэш-линии, это может привести к конфликтам в кэше (то есть ядра начинают мешать друг другу)

SYNCHRONIZED MAP

The Map type is optimized for two common use cases:

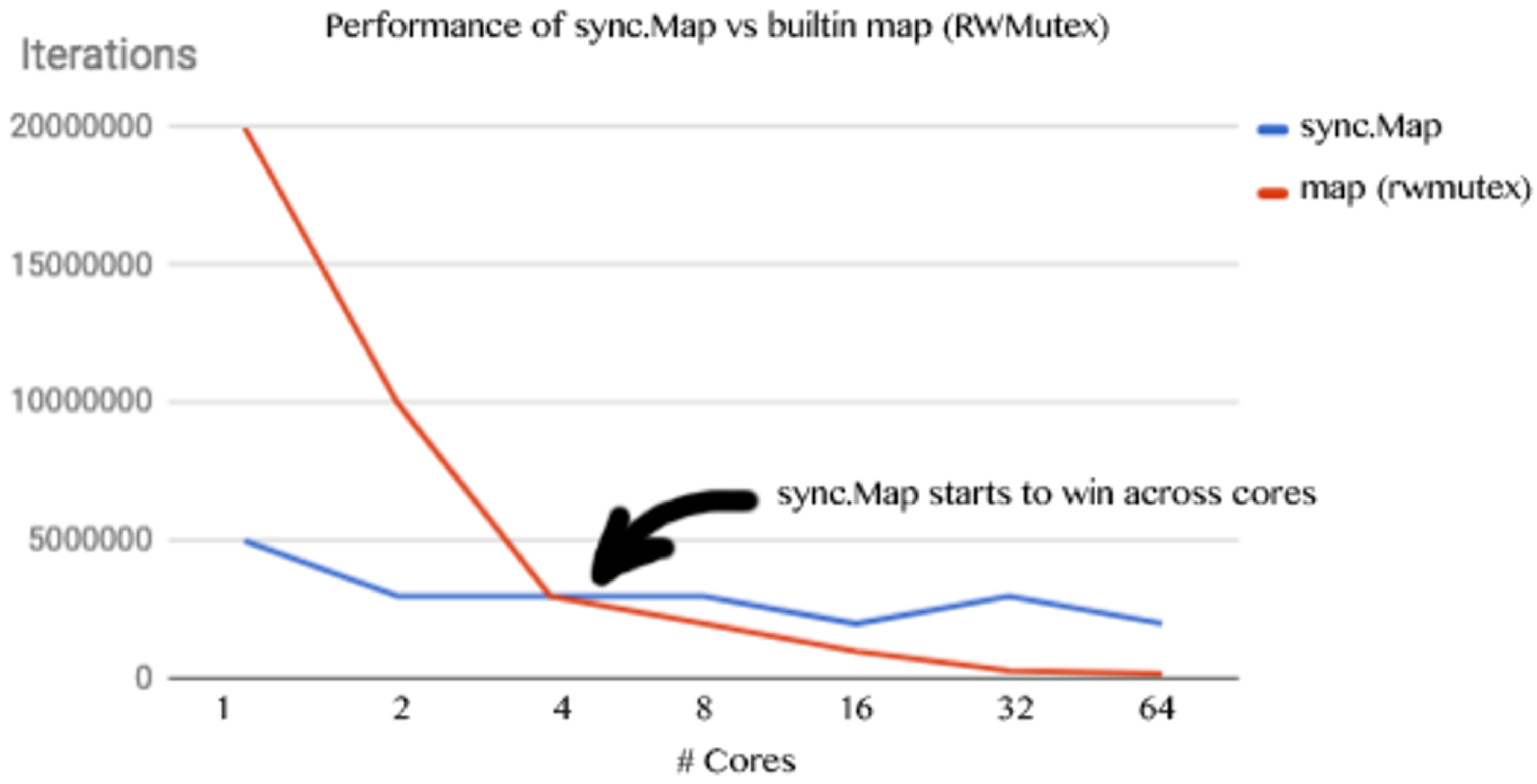
- when the entry for a given key is only ever written once but read many times
- when multiple goroutines read, write and overwrite entries for disjoint sets of keys

API

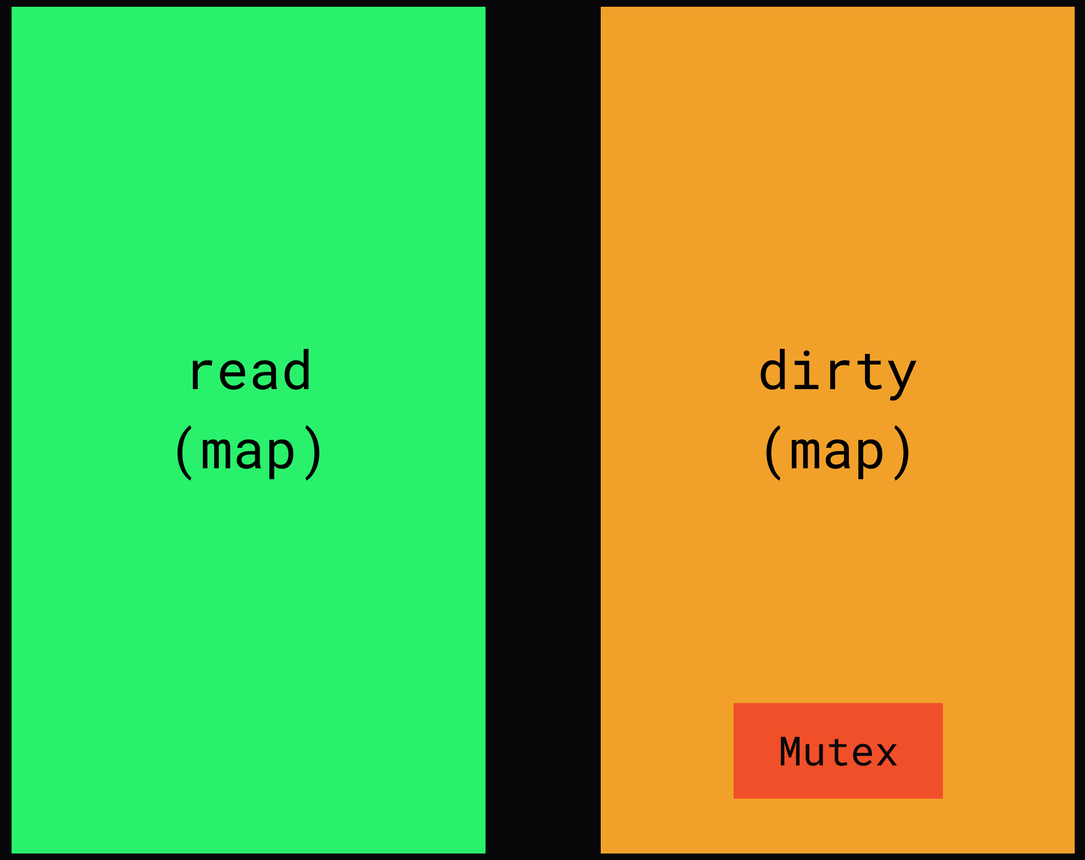
```go
type Map struct { ... }

func (m *Map) CompareAndDelete(key, old any) bool                    // удаляет, если по ключу key значение равно old
func (m *Map) CompareAndSwap(key, old, new any) bool                 // меняет значение по ключу на new, если значение равно old
func (m *Map) Delete(key any)                                        // удаляет значение по ключу
func (m *Map) Load(key any) (value any, ok bool)                     // возвращает значение по ключу
func (m *Map) LoadAndDelete(key any) (value any, loaded bool)        // удаляет значение по ключу и возвращает предыдущее значение
func (m *Map) LoadOrStore(key, value any) (actual any, loaded bool) // возвращает значение по ключу, если есть - иначе добавляет
func (m *Map) Range(f func(key, value any) bool)                     // проход по всем ключам и значениям
func (m *Map) Store(key, value any)                                  // добавление элемента
func (m *Map) Swap(key, value any) (previous any, loaded bool)       // меняет значение на value и возвращает предыдущее значение
```

EXAMPLE

Sync map


Performance of sync.Map vs builtin map (RWMutex)
Iterations
20000000
15000000
10000000
5000000
0
1
2
4
8
16
32
64
Cores
sync.Map
map (rwmutex)
sync.Map starts to win across cores

SYNCHRONIZED MAP



EXAMPLE

Sync map implementation