

Устройство map в Golang

// Failover bar

Филипп Кулин

20 июня 2022

Preface

- Презентация сделана на \LaTeX
- Презентация размещена на github
<https://github.com/schors/gmi2022-fob>

Preface

- Презентация сделана на \LaTeX
- Презентация размещена на github
<https://github.com/schors/gmi2022-fob>
- О, Великий Один, я торопился

map в Golang — хэш таблица

Хэш таблица

Таблица

- Ключу `key` сопоставляется значение `value`
- Операции:
 - Вставки
 - Поиска
 - Удаления

Хэш таблица

- Поиск по хэшу ключа
 - Константное или линейное время поиска $O(1)$ или $O(n)$

Что такое "хэш"

- Результат выполнения хэш функции
 - Преобразование данных произвольной длины в данные установленной длины
- Изменение данных приводит к изменению хэша
- Разные данные могут иметь один хэш — коллизии
 - Вероятность коллизий — мера качества хэш-функции

Тип данных map

Тип `map[KeyType]ValueType`

- `KeyType`: логические (`bool`), числовые, строки, указатели, каналы, интерфейсы, структуры и массивы
те, что могут сравниваться (==)
- **НЕ могут** быть ключами: срезы, `map`, функции

`map` — ссылочный тип, как срез и указатель

```
var m map[string]int
```

Залезем под капот

Что иль кто есть map'a?

// <https://go.dev/src/runtime/map.go>

Функция создания возвращает указатель на структуру

```
// makemap implements a Go map creation make(map[k]v, hint)
```

```
func makemap(t *maptype, hint int, h *hmap) *hmap
```

Компилятор заменяет конструкции функциями

```
v := m["ключ"]           // → runtime.mapaccess1(m, "ключ")
```

```
v, ok := m["ключ"]       // → runtime.mapaccess2(m, "ключ")
```

```
m["ключ"] = "значение"   // → runtime.mapassign(m, "ключ")
```

```
delete(m, "ключ")        // → runtime.mapdelete(m, "ключ")
```

Конкурентная работа с map

- Всегда `sync.RWLock` при конкурентной записи
 - Все операции состоят из набора функций
 - Планирование в прологах функций
- Только конкурентное чтение блокировки не требует

Заголовок тар'ы

```
type hmap struct {  
    count      int      // # кол-во элементов, len()  
    flags      uint8  
    B          uint8    // log_2 числа корзин, размер  
    noverflow  uint16   // примерное кол-во допкорзин  
    hash0      uint32   // hash seed  
  
    buckets    unsafe.Pointer // массив корзин  
    oldbuckets  unsafe.Pointer // устаревшие корзины (при росте)  
    nevacuate   uintptr      // прогресс эвакуации  
  
    extra *mapextra // дополнительные данные  
}
```

Папа, ты с кем сейчас разговаривал?

Корзины

- Данные хранятся в корзинах
- В корзине 8 элементов
- `bucketSize * (1+maxKeySize+maxValSize) + ptrSize`
// <https://go.dev/src/reflect/type.go>
- `TOP[N]` — первый байт хэша ключа
 - < 5 : флаги (пусто, конец, эвакуация)
 - реальные данные: приводится к ≥ 5

TOP1	TOP2	TOP3	TOP4	TOP5	TOP6	TOP7	TOP8
КЛЮЧ 1				ЗНАЧЕНИЕ 1			
КЛЮЧ 2				ЗНАЧЕНИЕ 2			
КЛЮЧ 3				ЗНАЧЕНИЕ 3			
КЛЮЧ 4				ЗНАЧЕНИЕ 4			
КЛЮЧ 5				ЗНАЧЕНИЕ 5			
КЛЮЧ 6				ЗНАЧЕНИЕ 6			
КЛЮЧ 7				ЗНАЧЕНИЕ 7			
КЛЮЧ 8				ЗНАЧЕНИЕ 8			
Указатель на переполнение							

↑
Фиксированный размер
↓

Поиск в корзине

- TOP — первый байт хэша искомого ключа
- TOP[N] — первый байт хэша записанного ключа
- TOP[N] == 0: поиск останавливается
- TOP == TOP[N]: сравниваются значения ключей
- Ключи не равны: ищем дальше

TOP1	TOP2	TOP3	TOP4	TOP5	TOP6	TOP7	TOP8
КЛЮЧ 1				ЗНАЧЕНИЕ 1			
КЛЮЧ 2				ЗНАЧЕНИЕ 2			
КЛЮЧ 3				ЗНАЧЕНИЕ 3			
КЛЮЧ 4				ЗНАЧЕНИЕ 4			
КЛЮЧ 5				ЗНАЧЕНИЕ 5			
КЛЮЧ 6				ЗНАЧЕНИЕ 6			
КЛЮЧ 7				ЗНАЧЕНИЕ 7			
КЛЮЧ 8				ЗНАЧЕНИЕ 8			
Указатель на переполнение							

↑
Фиксированный размер
↓

Организация корзин. Адресация

Размер map'ы hash.B

- $2^{hash.B}$ — количество корзин
- hash.B — маска значимых битов хэша ключа
- Значимые биты — индекс в массиве корзин

Устройство тар



- Количество корзин == маска хэша
- Внутри корзины поиск перебором

Выбор хэш-функции

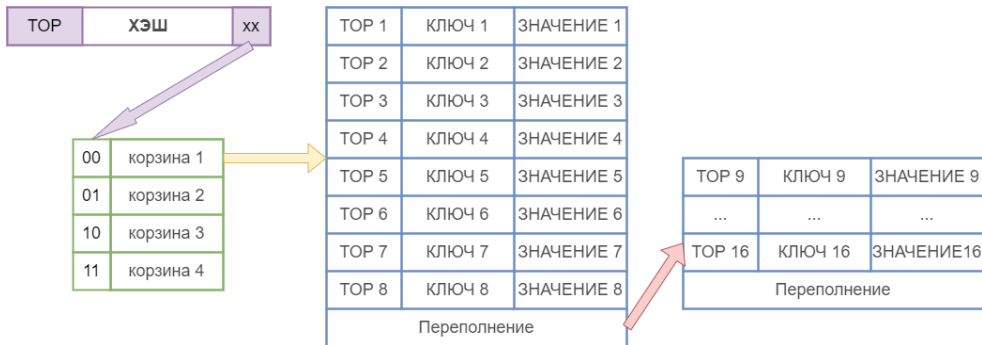
Условия

- Быстрая
- Низкая вероятность коллизий
- Хороший разброс (для уменьшения переполнений)
- Каждая таблица имеет свою уникальную "затравку"
хэш одних и тех же данных будет разным у разных экземпляров таблиц

Реализация

- Отдельные функции для ключей размера 32,64 бита и строк
- Генерируемые компилятором функции для остальных
- Меняются от версии к версии и на разных платформах
например на amd64 пытается использовать AES процессора

Устройство map



- Количество корзин == маска хэша
- Внутри корзины поиск перебором

Рост map

Условия

- Или среднее количество значений в корзине **6.5**
- Или слишком много переполнений (формула)

Что происходит

- Создаётся в два раза больше корзин
 - Если корзин 16 и больше, резервируется место для переполнений
- На записи и удалении данные мигрируют (эвакуируются)

Таблицы только растут

Особые случаи

- Если размер ключа или значение больше 128 байт, они размещаются отдельно (аллокация)
- Маленькая таблица до 8 значений будет размещена в стеке
- Пустая структура `struct` имеет нулевой размер

Всё — тлен

- Алгоритмы внутренних механизмов все время меняются
- 4 года назад все было не так
- Через год — всё будет иначе

Вопросы?

Ссылки

- [1] Beamer - Overleaf, Online LaTeX Editor. <https://www.overleaf.com/learn/latex/Beamer>.
- [2] Uri Nativ. How to present code. 2016. <https://www.slideshare.net/LookAtMySlides/codeware>.
- [3] Филипп Кулин. Пишем презентации в LaTeX. 14 окт. 2019. <https://habr.com/ru/post/471352/>.
- [4] The Go Programming Language Specification. <https://go.dev/ref/spec>.
- [5] Effective Go. https://go.dev/doc/effective_go.
- [6] Go maps in action. 6 февр. 2013. <https://go.dev/blog/maps>.
- [7] Dave Cheney. How the Go runtime implements maps efficiently (without generics). 29 мая 2018.
<https://dave.cheney.net/2018/05/29/how-the-go-runtime-implements-maps-efficiently-without-generics>.
- [8] Dave Cheney. If a map isn't a reference variable, what is it? 30 апр. 2017.
<https://dave.cheney.net/2017/04/30/if-a-map-isnt-a-reference-variable-what-is-it>.