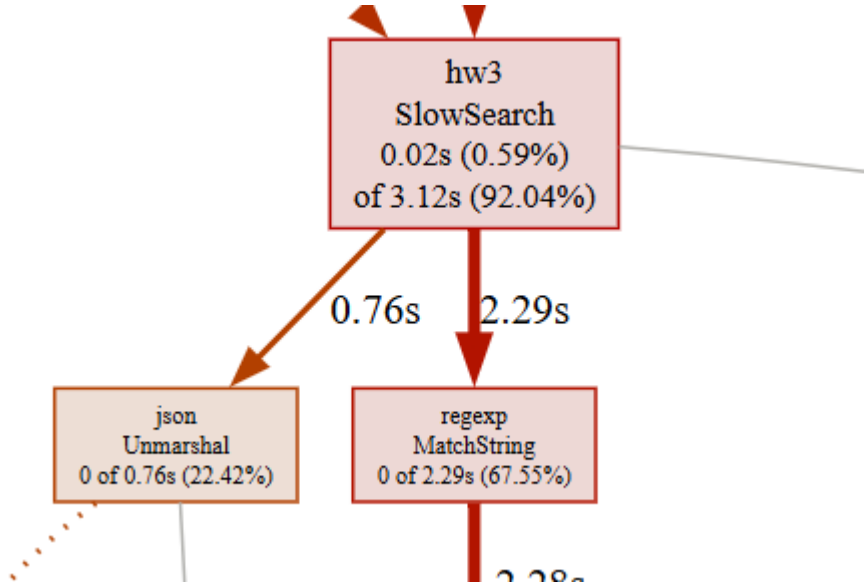
Наша цель оптимизировать работу функции SlowSearch (написав его новую версию в *FastSearch*). Метрики, которые используется для сравнения версий это ns/op, B/op, allocs/op. Начнем с того, что сгенерируем файлы mem и cpu с помощью команды `go test -bench . -cpuprofile=cpu -memprofile=mem .`

I. Скорость программы

```
PS C:\GoPractice\3> go tool pprof .\hw3.test.exe .\cpu
File: hw3.test.exe
Build ID: C:\Users\ACCEPT~1\AppData\Local\Temp\go-build2000210849\b001\hw3.test.exe2025-07-11 03:57:36.8331731 +0300 MSK
Type: cpu
Time: Jul 11, 2025 at 3:57am (MSK)
Duration: 3.45s, Total samples = 3.39s (98.35%)
Entering interactive mode (type "help" for commands, "o" for options)
(pprof) web
```

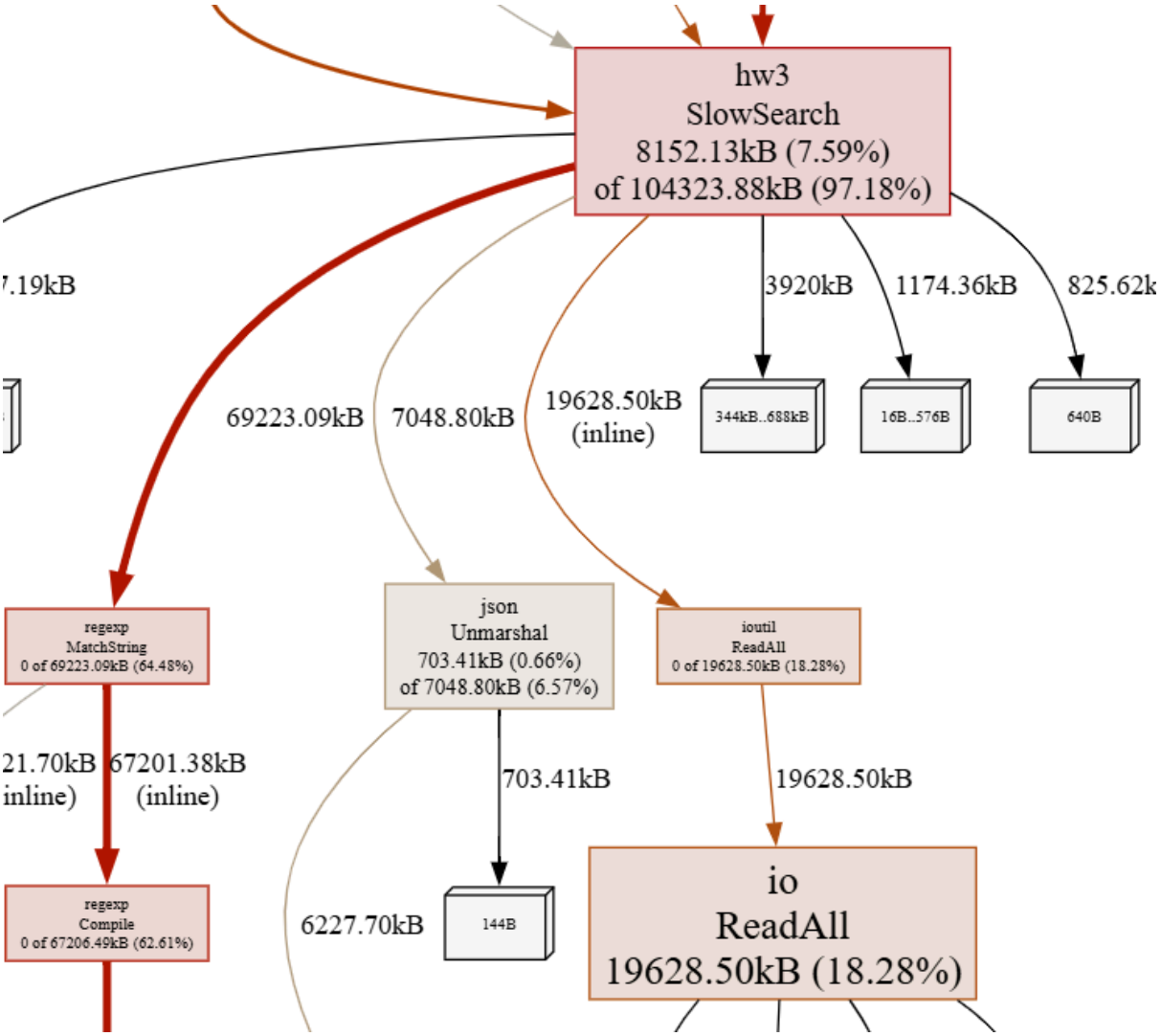
Смотрим на граф:



И обращаем внимание на этот кусок графа, в котором мы видим, что основные функции, которые съедают процессорное время это MatchString и Unmarshal.

1. Начнем с MatchString. Если внимательно посмотреть на код, то эта функция используется лишь для того, чтобы проверить есть данная строка в строке. Если немного изучить работу этой функции (либо посмотреть на граф), то можно понять, что эта функция для начала компилирует регулярное выражение, чтобы эффективно искать совпадения. Можно заменить эту функцию на strings.Contains и тогда скорость заметно повыситься
2. Если говорить про Unmarshal, то можно воспользоваться библиотекой easyjson (кодогенерация), правда для этого нужно немного поглядеть на код, чтобы понять какую структуру нужно создать, чтобы в нее анмаршалить.

II. Работа с памятью



На графе можно увидеть, что самые проблемные функции с точки зрения памяти - это MatchString и ReadAll

1. MatchString. Мы уже о ней говорили. Тут проблема в компиляции регулярного выражения, которая создает структуру, которая видимо много занимает памяти
2. ReadAll. Здесь интересный момент. Программа никогда не использует весь файл целиком, а обрабатывает его построчно. Стало быть стоит изменить программу так, чтобы читать до ‘\n’.

Результат:

```
BenchmarkSlow-16              44          33064211 ns/op        20472684 B/op        182874 allocs/op
BenchmarkFast-16             379           3628274 ns/op         2095975 B/op          9799 allocs/op
```