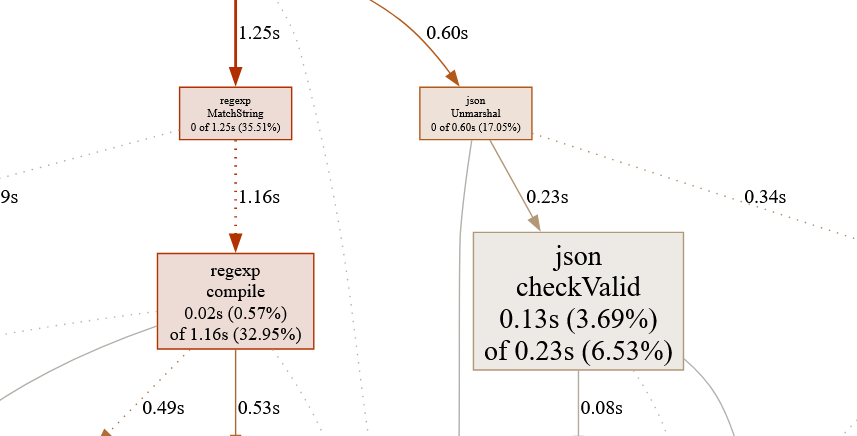
Рассмотрим древо



Как можно заметить, почти 50% времени CPU занимает regexp и и распаковка json’a

Рассмотрим, как используется функция MatchString

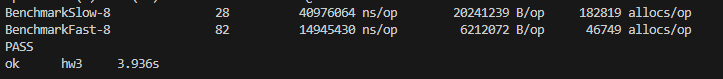
ok, err := regexp.MatchString("Android", browser); ok && err == nil {

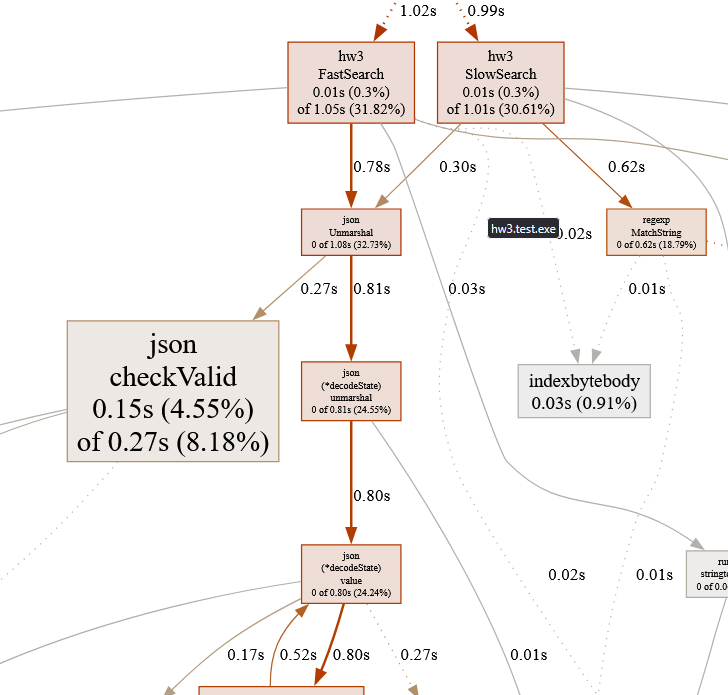
ok, err := regexp.MatchString("MSIE", browser);

Можно заметить, что происходит каждый раз компиляция ради простого слова Android , MSIE , а не какого-то шаблона слов, из-за чего можно попробовать заменить простым string.Contains

Вместо json unmarshal можно попробовать использовать потоковую обработку.

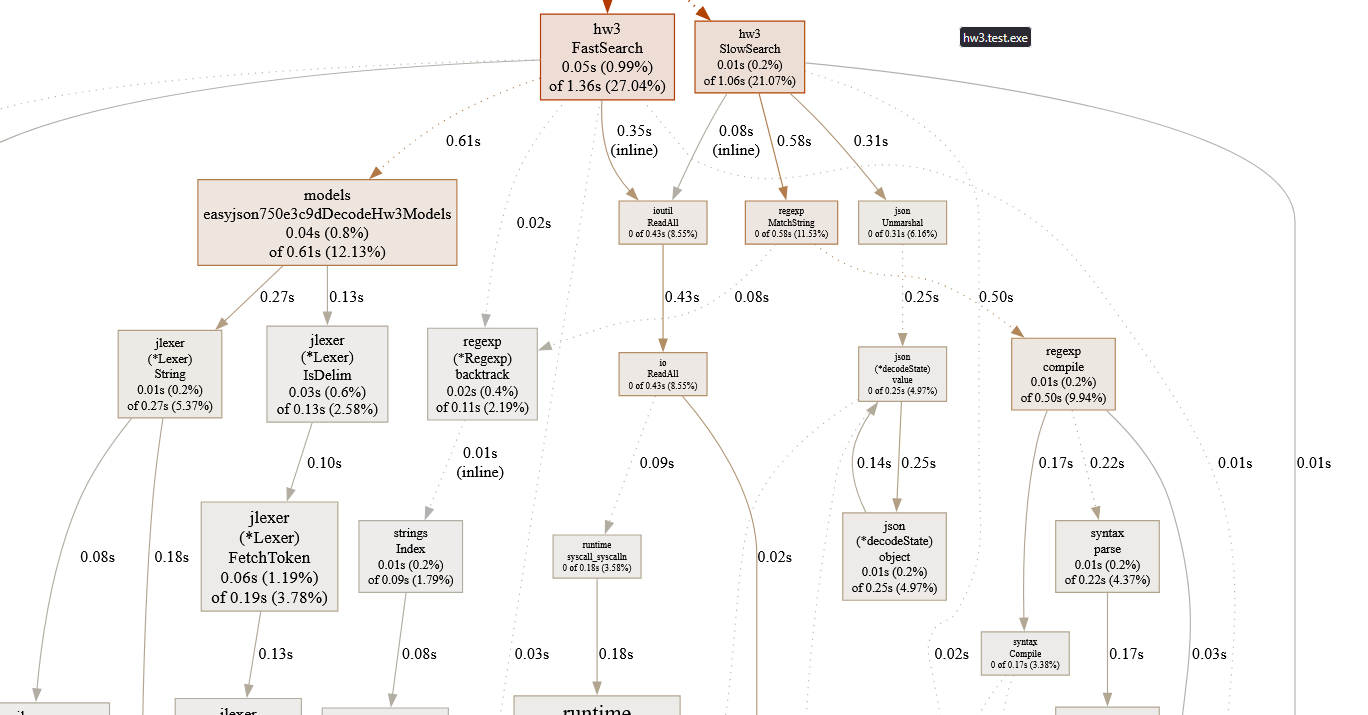
Попробуем заменить MatchString



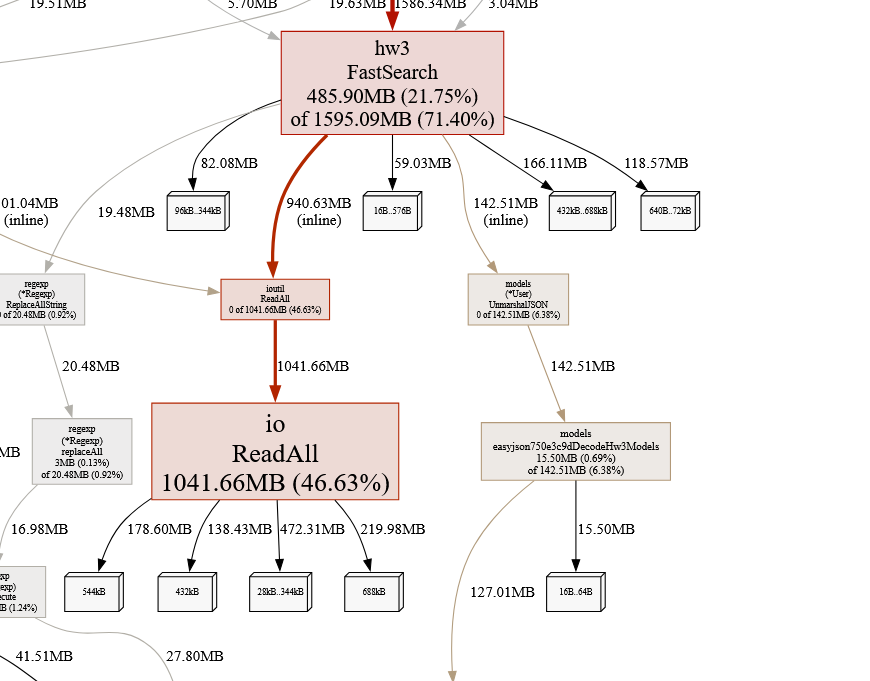


Как видим, большую часть у нас занимает ещё обработку Json, добавим easyjson, что убрало нам многие проверки и преобразования типов.

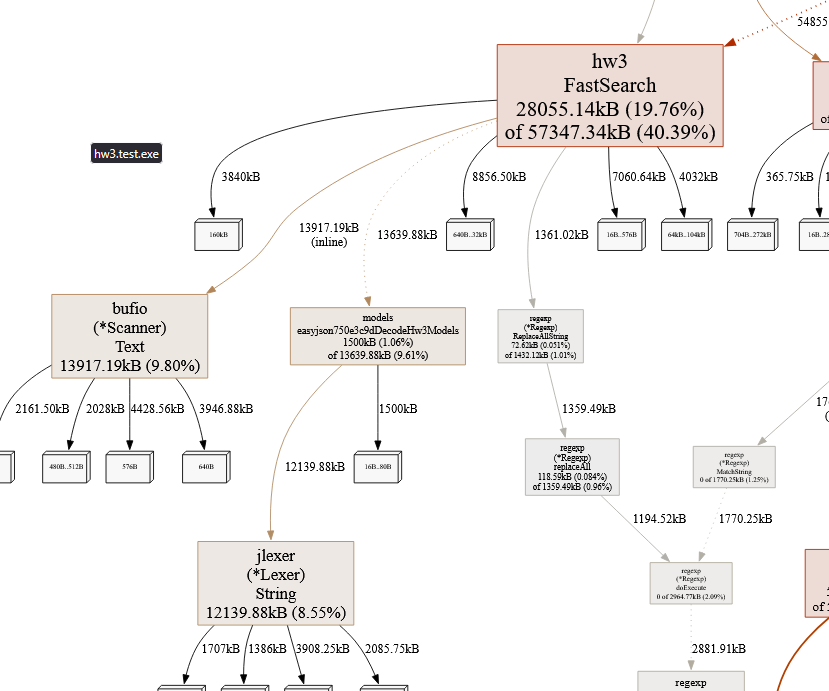




Как можно заметить, явных «тормозов» при обработки файла на стороне CPU нет, давайте рассмотрим память

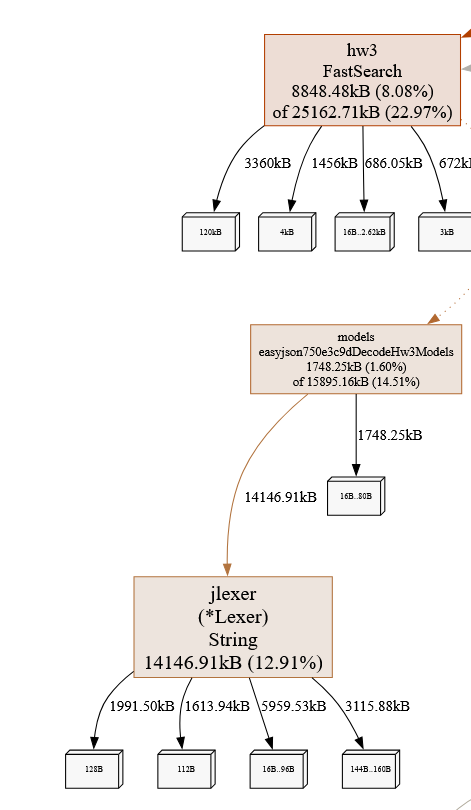


Как видно, считывание всего файла требует много ресурсов, однако, нам нужна только одна строка. Добавим Scanner





Как минимум, количество потребляемой памяти в 2 раза уменьшилось



Посмотрим текстово  
  
5.11MB 5.27MB 101: foundUsers += fmt.Sprintf("[%d] %s <%s>\n", i, user.Name, email)  
  
Заменим на strings.Builder. fmt.Fprintf(&sb, "[%d] %s <%s>\n", i, user.Name, email)

fmt.Fprintln(out, "found users:\n"+sb.String())

До:



После:



Займёмся парсингом json’a , и вместо всех структур сократим до тех, которые нам нужны  
  
type User struct {

    Browsers []string `json:"browsers"`

    Email    string   `json:"email"`

    Name     string   `json:"name"`

}

Проверям



Что по B/op и allocs/op ниже BenchmarkSolution

Можно чисто по алгоритму обратить внимание, что нам не надо дальше обрабатывать массив с браузерами, если мы уже нашли дубликат, т.е. добавим везде break

for \_, item := range seenBrowsers {

                    if item == browser {

                        notSeenBefore = false

                        break

                    }

                }

                if notSeenBefore {

                    seenBrowsers = append(seenBrowsers, browser)

                    uniqueBrowsers++

                    break

                }

Также и в поиске MSIE

Запустим ещё раз



Что слегка улучшило скорость выполнения функции.