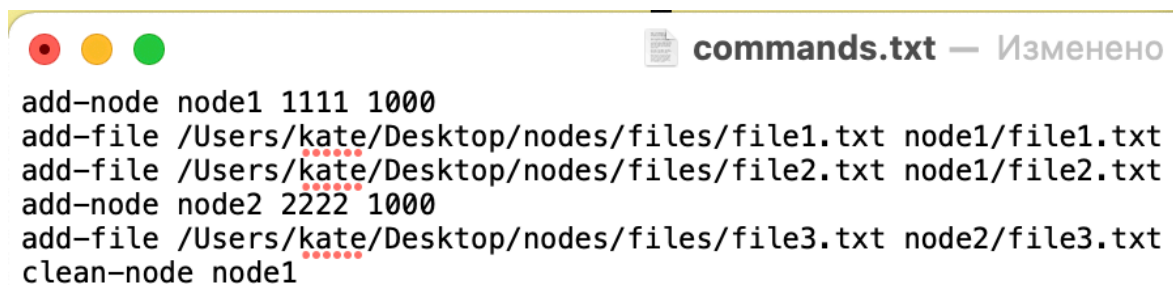


Отчет лаба 4, Жаркова Е.С.

Суть лабы:

У нас есть файл с командами, которые должны, как ни странно, выполняться:



```
add-node node1 1111 1000
add-file /Users/kate/Desktop/nodes/files/file1.txt node1/file1.txt
add-file /Users/kate/Desktop/nodes/files/file2.txt node1/file2.txt
add-node node2 2222 1000
add-file /Users/kate/Desktop/nodes/files/file3.txt node2/file3.txt
clean-node node1
```

Только их там не 3, а max = 10000

Запускаем клиенты на трех нодах с разными портами и запускаем сервак, который пробежится по командам и выполнит их.

В program.cs/server мы запускаем Exec(принимает путь), который в свою очередь выполняет команды, описанные в commands.txt

Команды делятся на два типа (по моему скромному мнению):

С которыми все просто, и не нужно писать алгоритм:

Add-node

Add-file

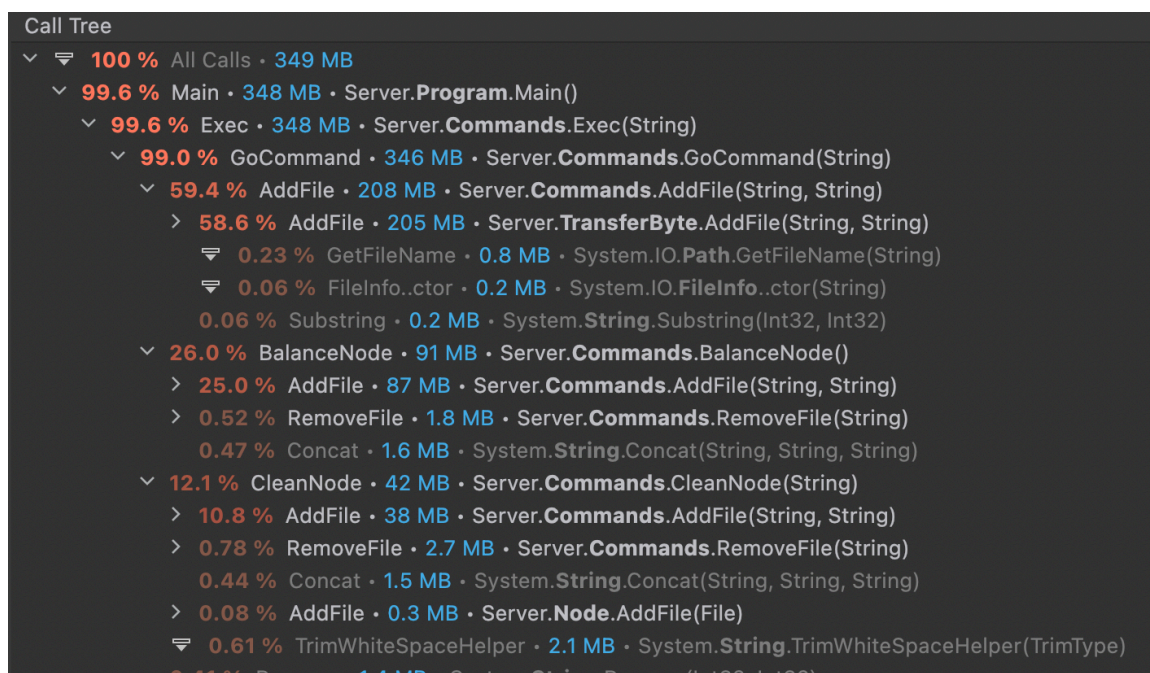
Remove-file

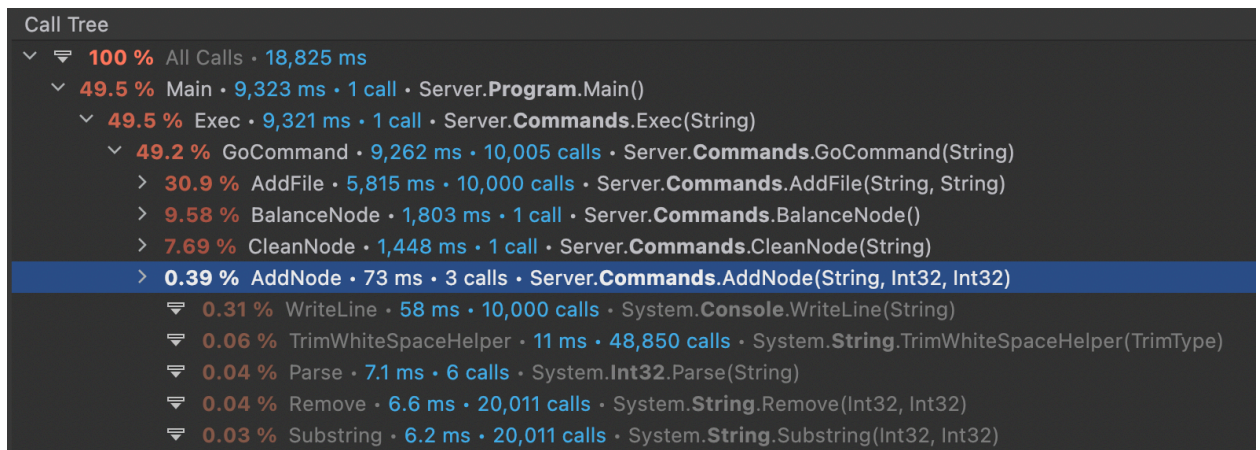
У которых нужно половить два дня ошибки:

Clean-node

Balance-node

А теперь посмотрим, насколько плохо это работает:



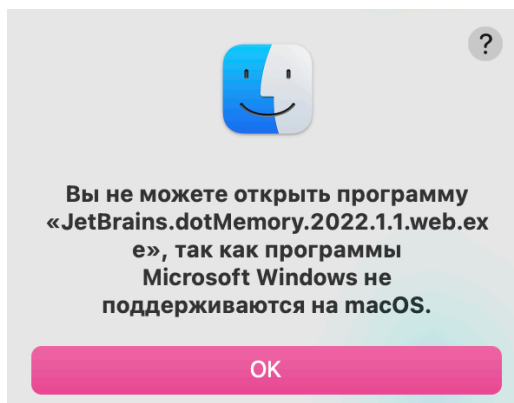


Посмотрели, теперь думаем, как это можно улучшить...

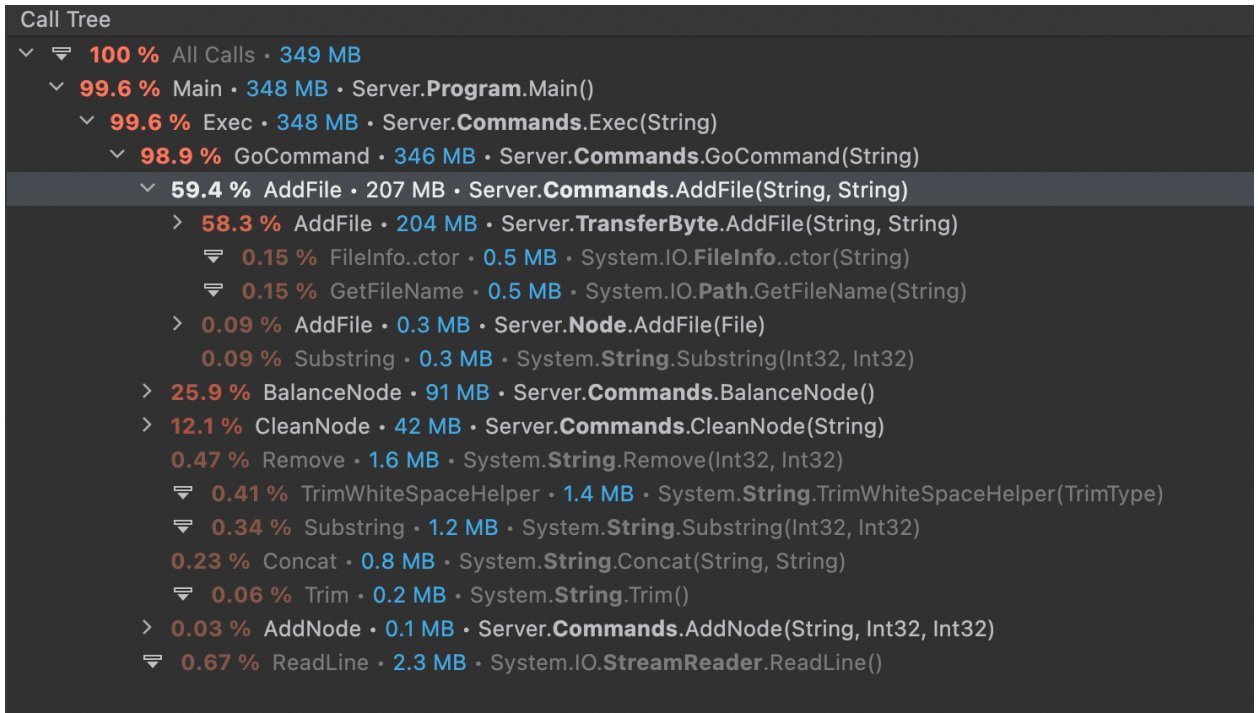
0) Для начала, как все нормальные люди, удаляем Скайп.

Теперь прекращаем шутить и открываем dotTrace

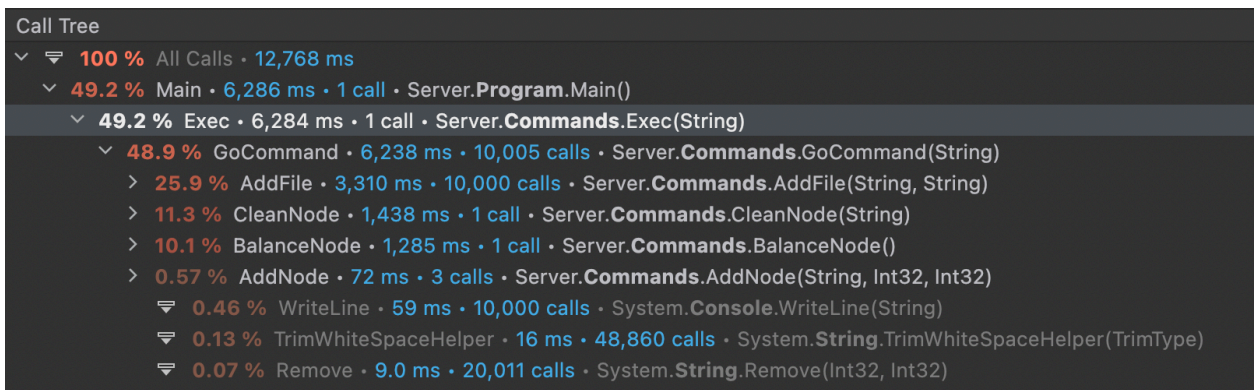
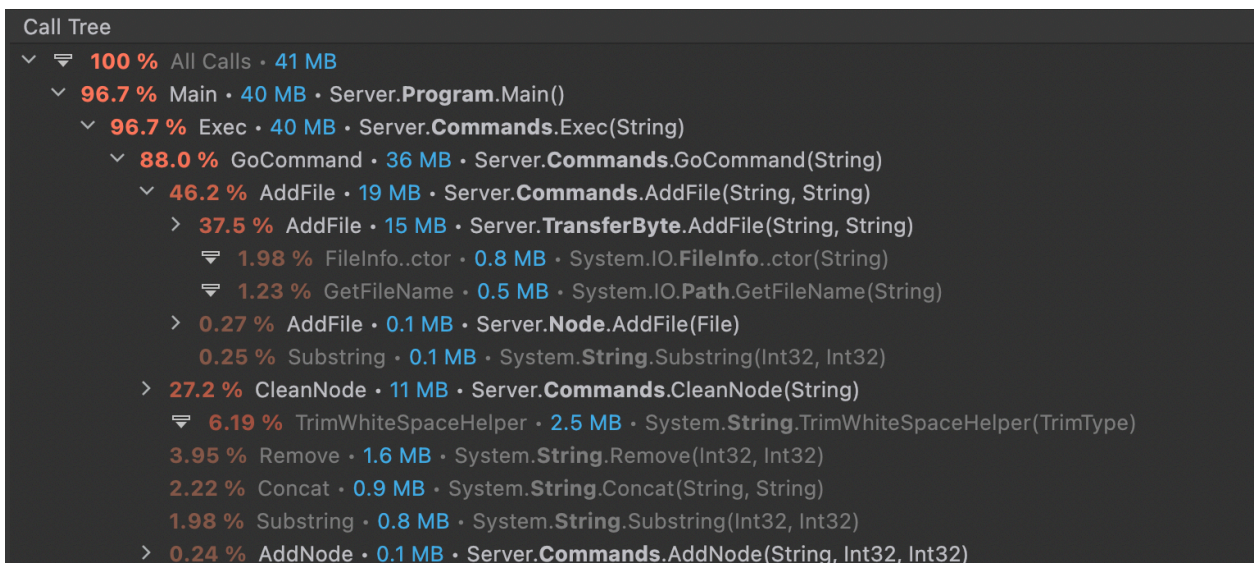
Еще пытаемся скачать dotMemory, вспоминаем, что работаем на маке и плачем в сторонке



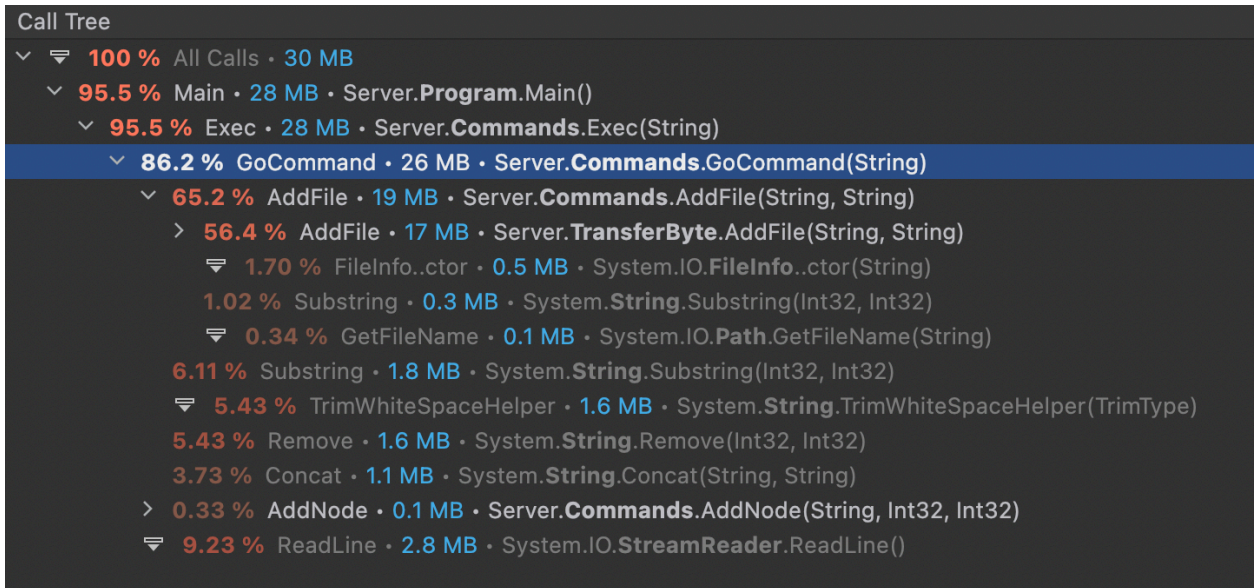
1) Заменяли class'ы на record'ы, но это дало мало...



2) Меняем на `arrayPool`, это значительноооооо помогло



- 3) Заменяем на сложные конструкции \$"{переменная}/{переменная}" на stringBuilder Улучшили!



- 4) Самое интересное! Переделываем парсинг команд с

```
public void GoCommand(string str)
{
    string command = str.Substring(startIndex: 0, length: str.IndexOf(' ')).Trim();
    str = str.Remove(startIndex: 0, count: str.IndexOf(' ')).Trim();
    string nodeName, path, partialPath;

    switch (command)
    {
        case "add-node":
            nodeName = str.Substring(startIndex: 0, length: str.IndexOf(' ')).Trim();
            str = str.Remove(startIndex: 0, count: str.IndexOf(' ')).Trim();
            string portStr = str.Substring(startIndex: 0, length: str.IndexOf(' ')).Trim();
            str = str.Remove(startIndex: 0, count: str.IndexOf(' ')).Trim();
            int port = Int32.Parse(portStr);
            string sizeStr = str.Trim();
            int size = int.Parse(sizeStr);
            AddNode(nodeName, port, size);
            break;
        case "add-file":
            path = str.Substring(startIndex: 0, length: str.IndexOf(' ')).Trim();
            str = str.Remove(startIndex: 0, count: str.IndexOf(' ')).Trim();
            partialPath = str.Trim();
            Console.WriteLine($"add file {partialPath}");
            AddFile(path, partP: partialPath);
            break;
    }
}
```


на

```
public void GoCommand(string str)
{
    var commands:string[] = str.Split();
    string command = commands[0];
    string nodeName, path;

    switch (command)
    {
        case "add-node":
            nodeName = commands[1];
            string portStr = commands[2];
            int port = Int32.Parse(portStr);
            string sizeStr = commands[3];
            int size = int.Parse(sizeStr);
            AddNode(nodeName, port, size);
            break;
        case "add-file":
            path = commands[1];
            var partialPath:string = commands[2];
            AddFile(path, partP: partialPath);
            break;
    }
}
```

И получаем:

Call Tree		
100 %	All Calls	58 MB
97.7 %	Main	57 MB • Server.Program.Main()
97.7 %	Exec	57 MB • Server.Commands.Exec(String)
91.8 %	GoCommand	53 MB • Server.Commands.GoCommand(String)
32.5 %	AddFile	19 MB • Server.Commands.AddFile(String, String)
32.4 %	CleanNode	19 MB • Server.Commands.CleanNode(String)
22.9 %	BalanceNode	13 MB • Server.Commands.BalanceNode()
3.84 %	SplitInternal	2.2 MB • System.String.SplitInternal(ReadOnlySpan, Int32, StringSplitOptions)
0.17 %	AddNode	0.1 MB • Server.Commands.AddNode(String, Int32, Int32)
5.91 %	ReadLine	3.4 MB • System.IO.StreamReader.ReadLine()