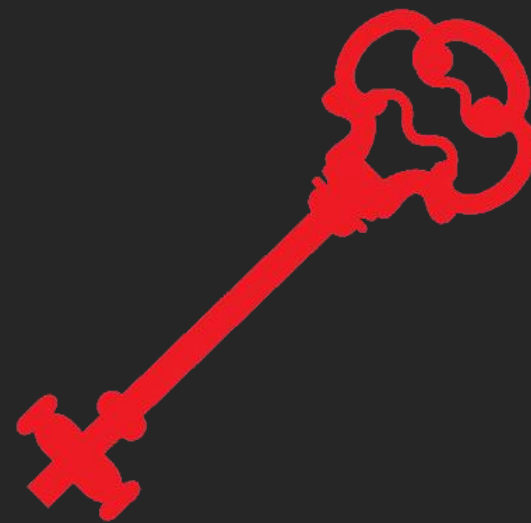
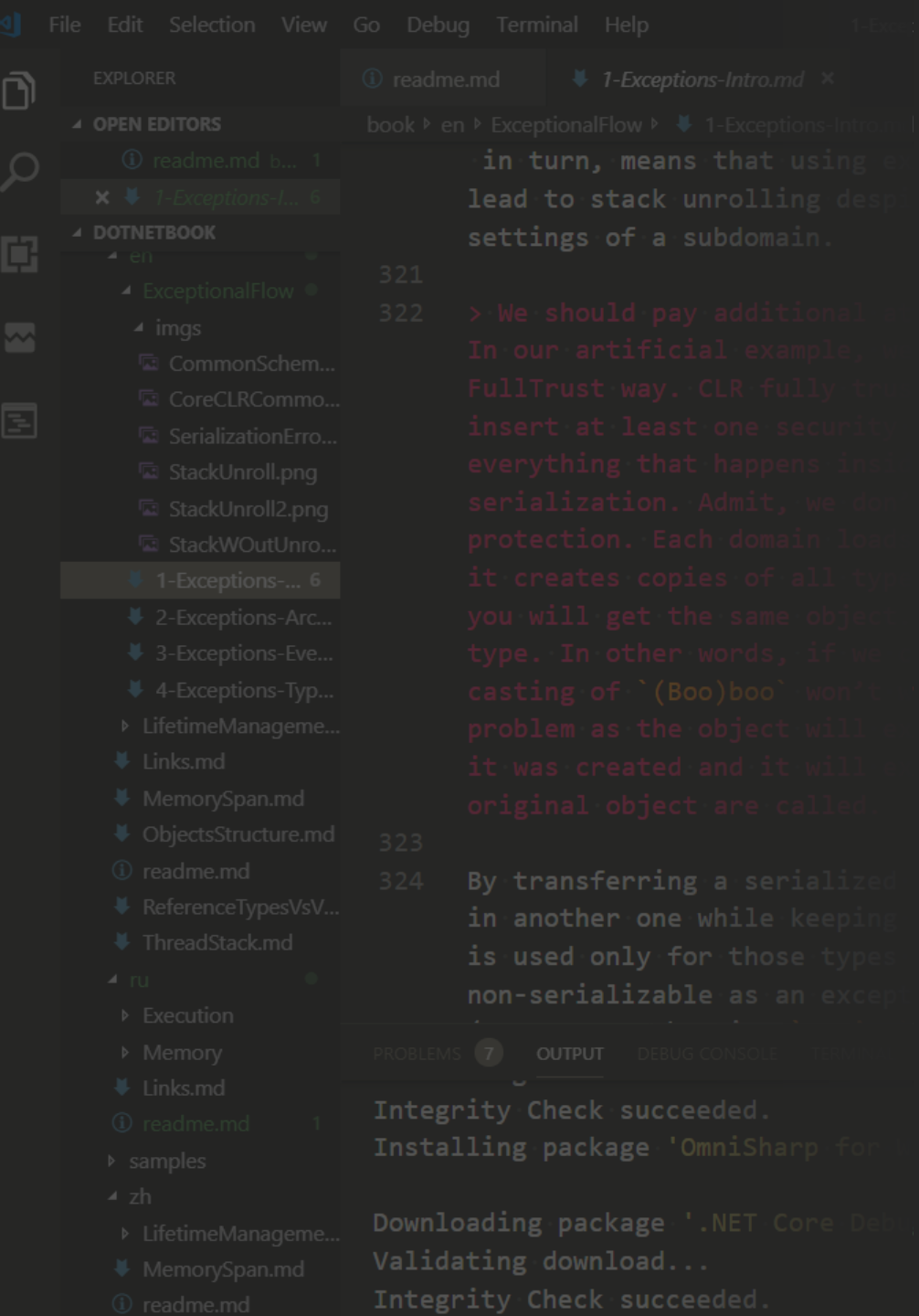


Делаем zero-allocation код






Станислав Сидристый

Стеки:

- WEB/WPF/WinForms/... стеки
- C/C++, C++/CLI когда необходимо
- Семинары *CLRium*
- Книга: <https://github.com/sidristij/dotnetbook>

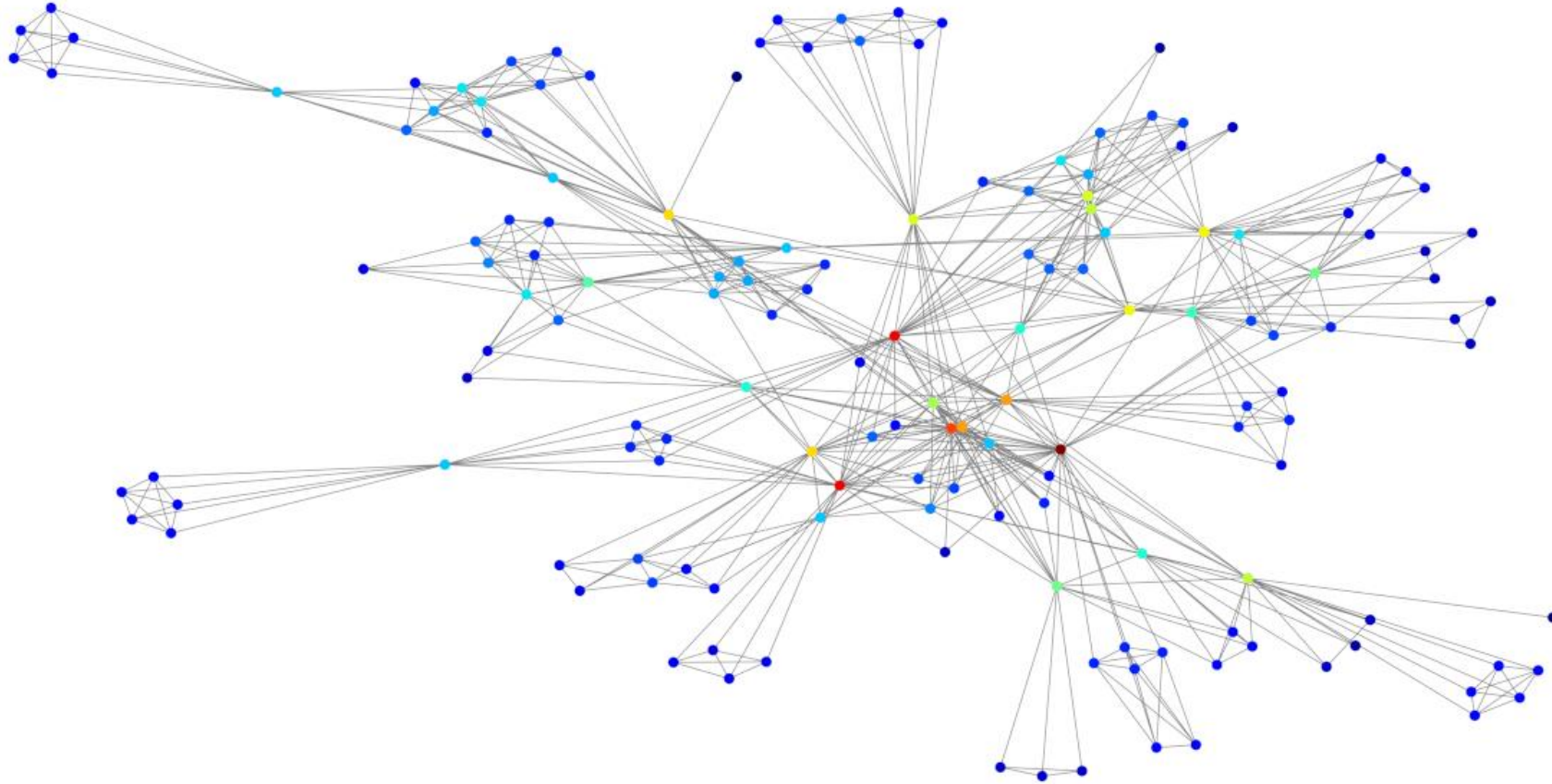
Связь:

- telegram: @sidristij
- sunex.development@gmail.com

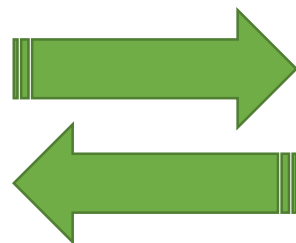
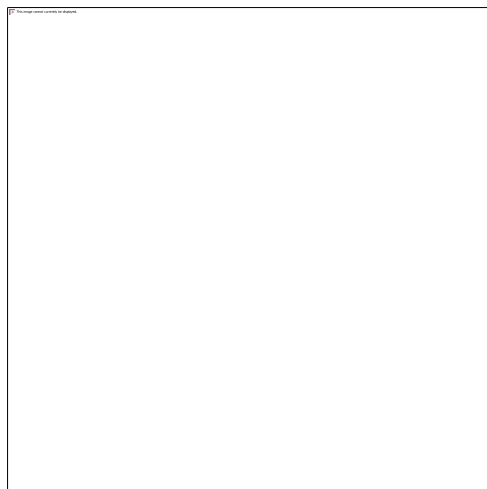


всё было хорошо,
пока не узнали, что всё плохо

*Первый акт — это презентация основных действующих лиц, мира,
в который нас приглашает автор и, собственно, завязка истории*



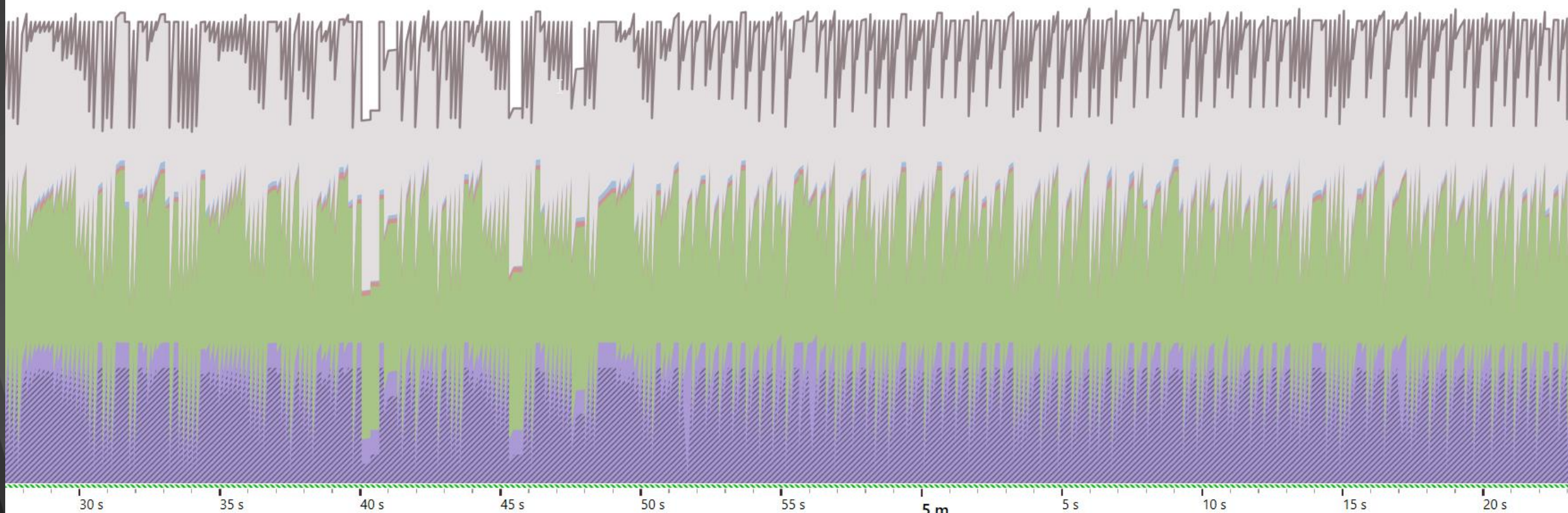
Бывает **многосерверное** решение, а бывает – **односерверное**



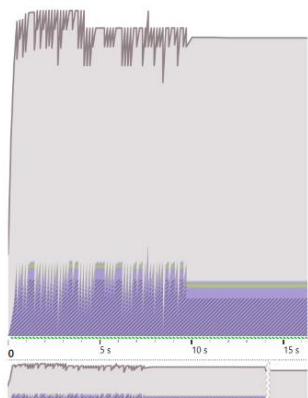
SMB_{2.0+}

При высокой нагрузке на приложение – очень плотная работа с серверами

new T(), new T[N];



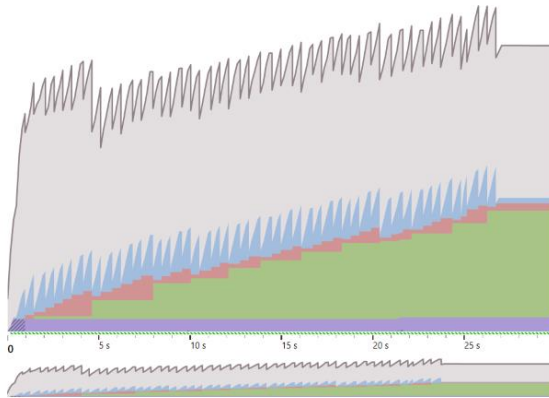
10,000
соединений



7,955 сек.
1,12 Gb

50+ GC₀

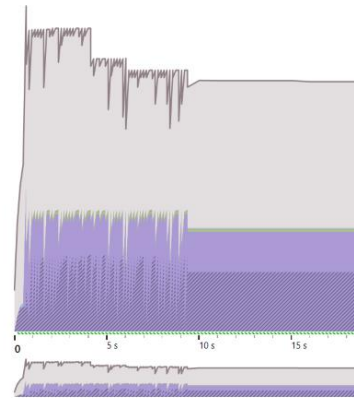
Перечисление 21,000
файлов в сложной структуре
каталогов



26,899 сек.
265,89 Mb

50+ GC₀

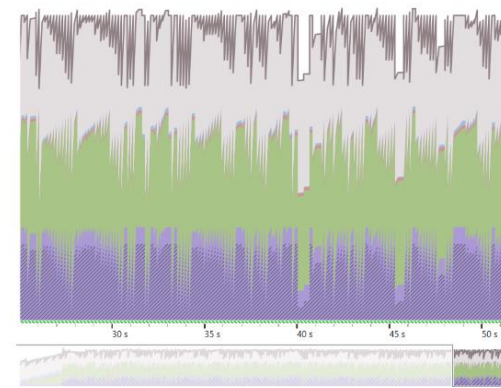
Скачать 200
файлов 16Мб каждый



7,325 сек.
6,49 Gb

50+ GC₀

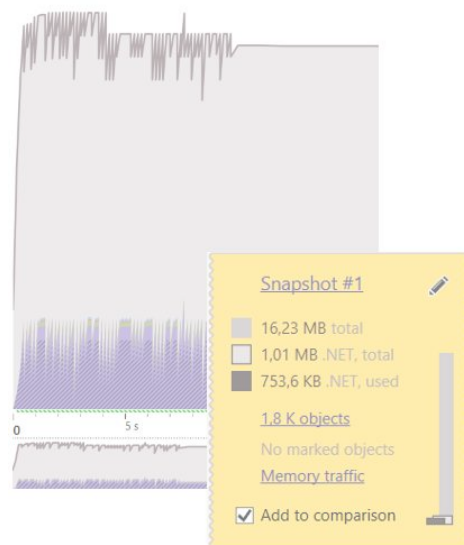
Скачать 21,000
файлов из сложной структуры
каталогов



255,222 сек.
25,12 Gb

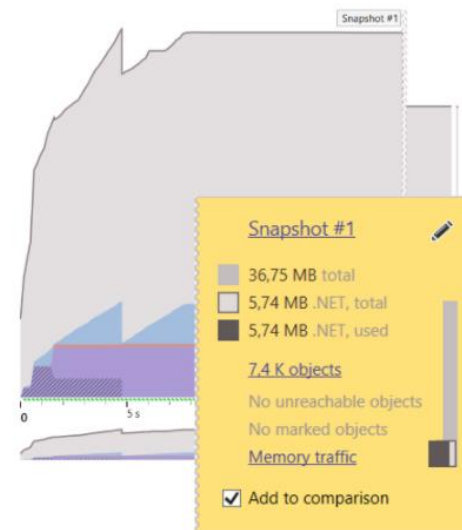
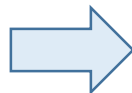
4000+ GC₀

10,000 соединений



7,96 сек.
1,12 Gb

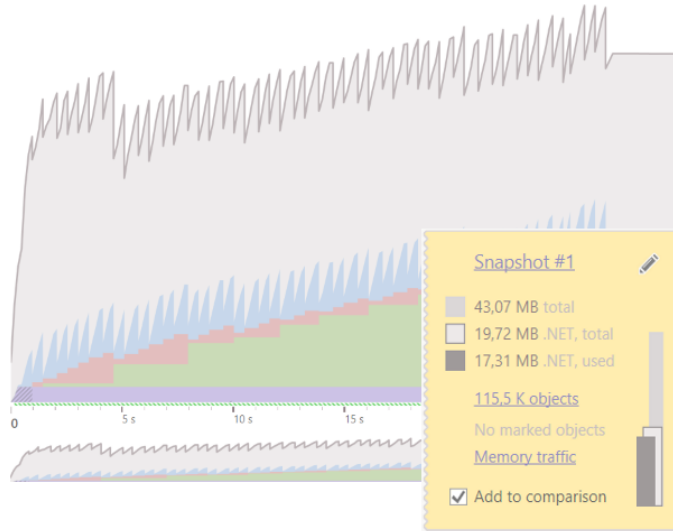
50+ GC₀



5,10 сек.
8,50 Mb
*System.Cryptography.**

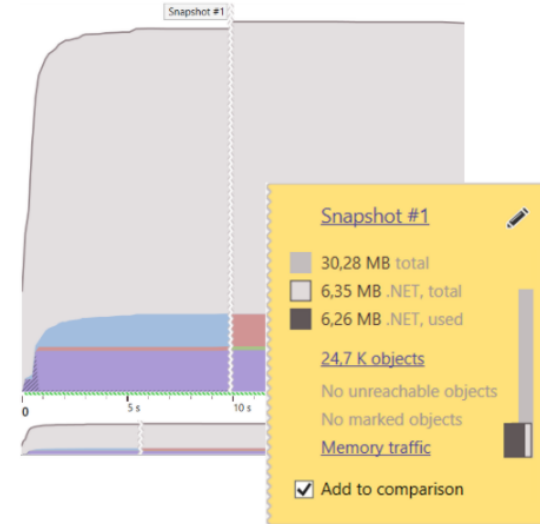
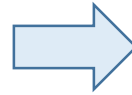
2 GC₀

Перечисление 21,000 файлов в сложной структуре каталогов



26,899 сек.
265,89 Mb

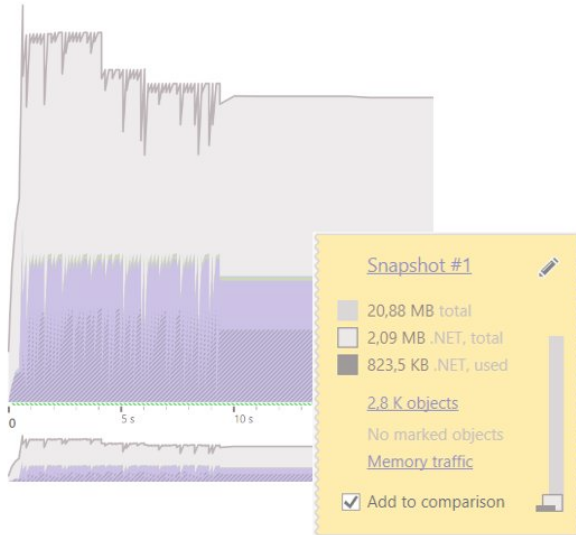
50+ GC₀



4 сек.
0,005 Mb

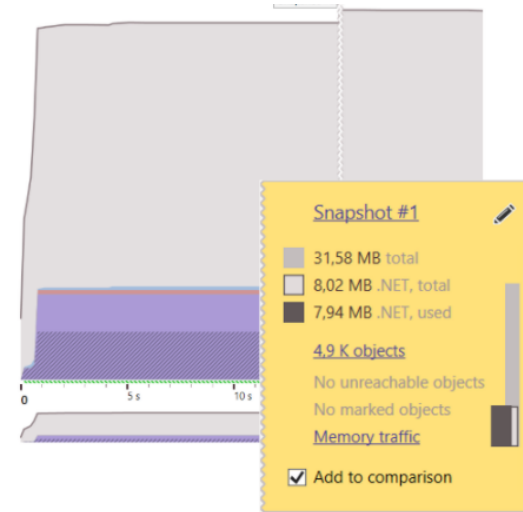
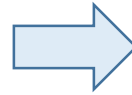
0 GC₀

Скачать 200 файлов 16Мб каждый



7,325 сек.
6,49 Gb

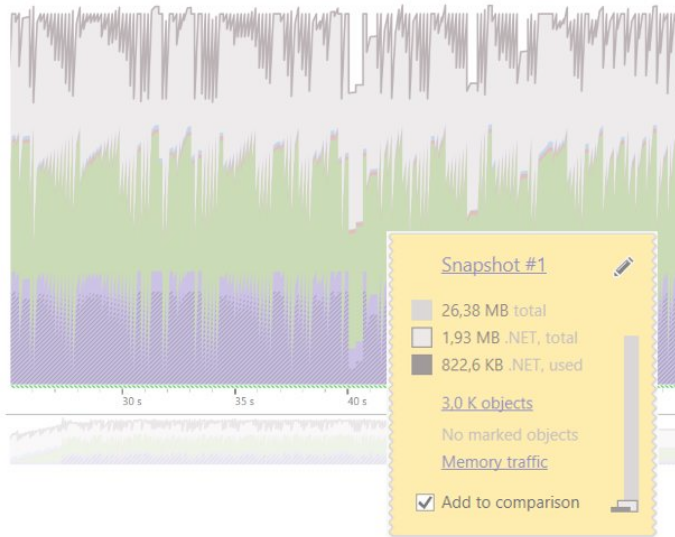
50+ GC₀



3 сек.
0,016 Mb

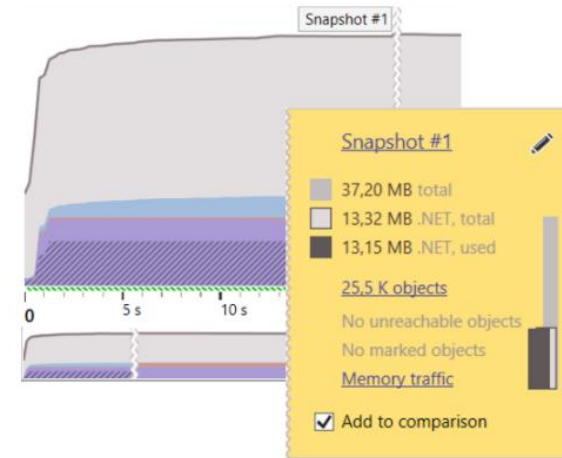
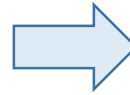
0 GC₀

Скачать 21,000 файлов со сложной структуре каталогов



255,222 сек.
25,15 Gb

4,000+ GC₀



11 сек.
0,021 Mb

0 GC₀

Так что же делать?

Пулинг!

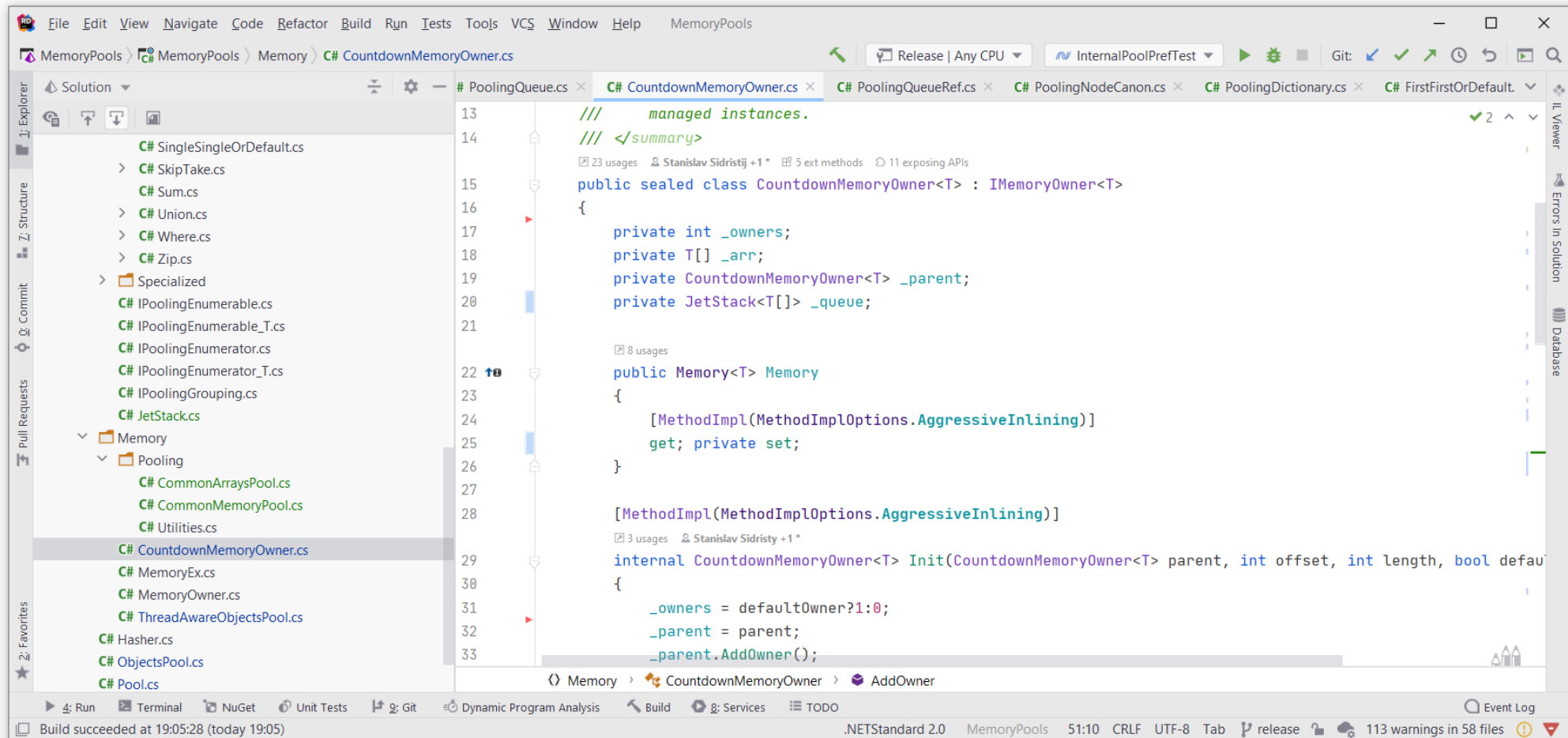


Three crossed keys, likely a symbol of memory or a key to a story, are positioned behind the title text.

С ГОЛОВОЙ В ПАМЯТЬ

Второй акт — это самая большая, основная часть истории. Здесь вызов, брошенный антагонистом, заставляет героя действовать. К середине второго акта он уже не может повернуть назад, как бы ему этого ни хотелось. Во второй половине второго акта многократно возрастают ставки и риски. И к концу второго акта герой терпит большое поражение, оказываясь в максимальной опасности, практически в безвыходной ситуации.

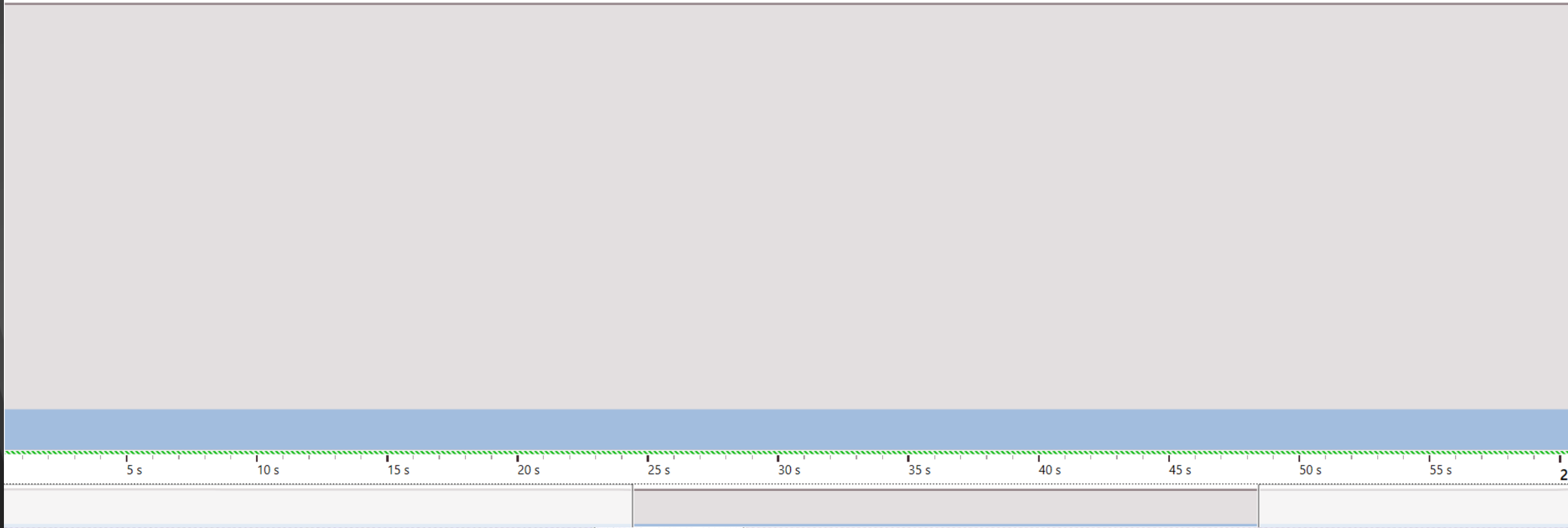
- Пулинг массивов
- `IMemoryOwner<T>`, `Memory<T>`, `Span<T>`
- Пулинг объектов
- Пулинг `IEnumerator`
- К чёрту `async/await`!
- Ускоряем пулинг!!
- Profit!!!



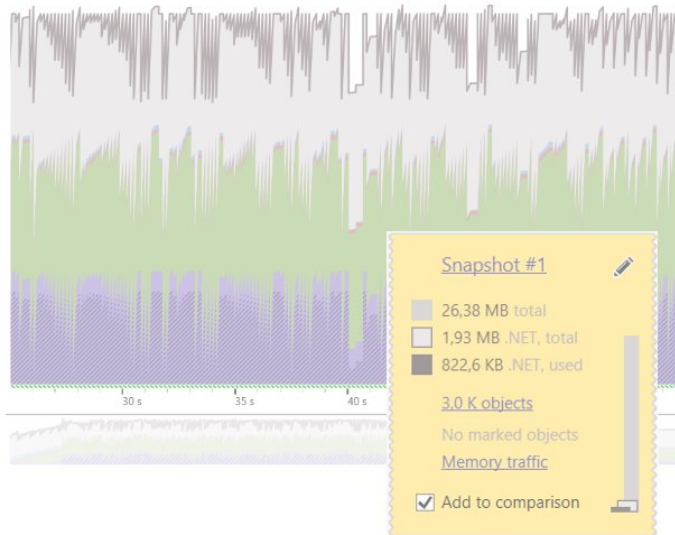
Путь до конца

- Самая большая сложность: мало точек, когда всё работает;
- Много изменений, которые «не покрыть мозгом». Есть только надежда;
- Но после прохождения каждой и них – ощущение победы;
- С каждой следующей точкой всё более явственно: «слишком много пройдено и назад пути нет»;
- Сроки давно пройдены;
- Надо отвечать на вопросы, почему всё затянулось;

- 99.9999% - работа Garbage Collector

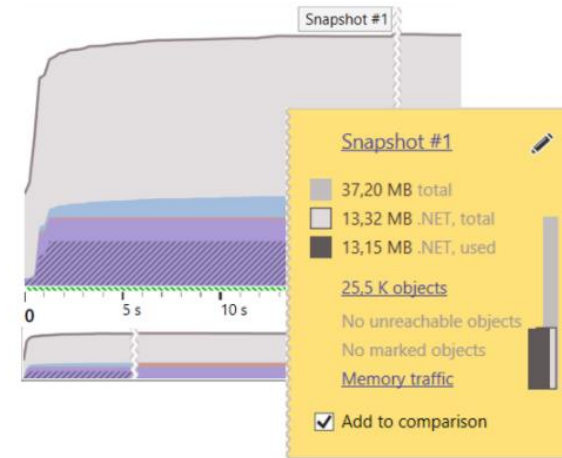
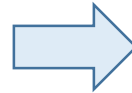


Скачать 21,000 файлов со сложной структуре каталогов



255,222 сек.
25,15 Gb

4,000+ GC₀



11 сек.
0,021 Mb

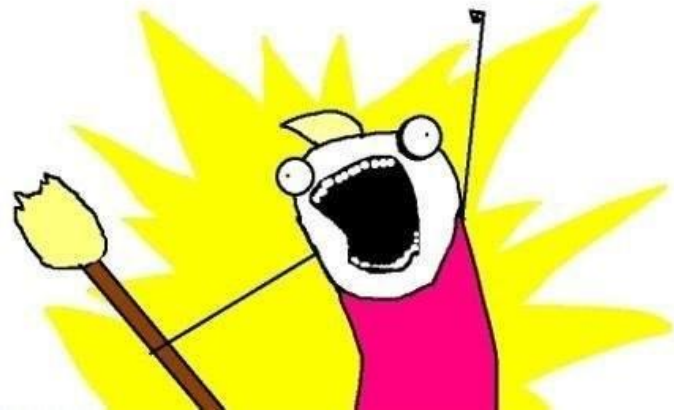
0 GC₀

Что бы ещё такого сделать?

Коллекции!



LINQ!!



Всё по пулам!!



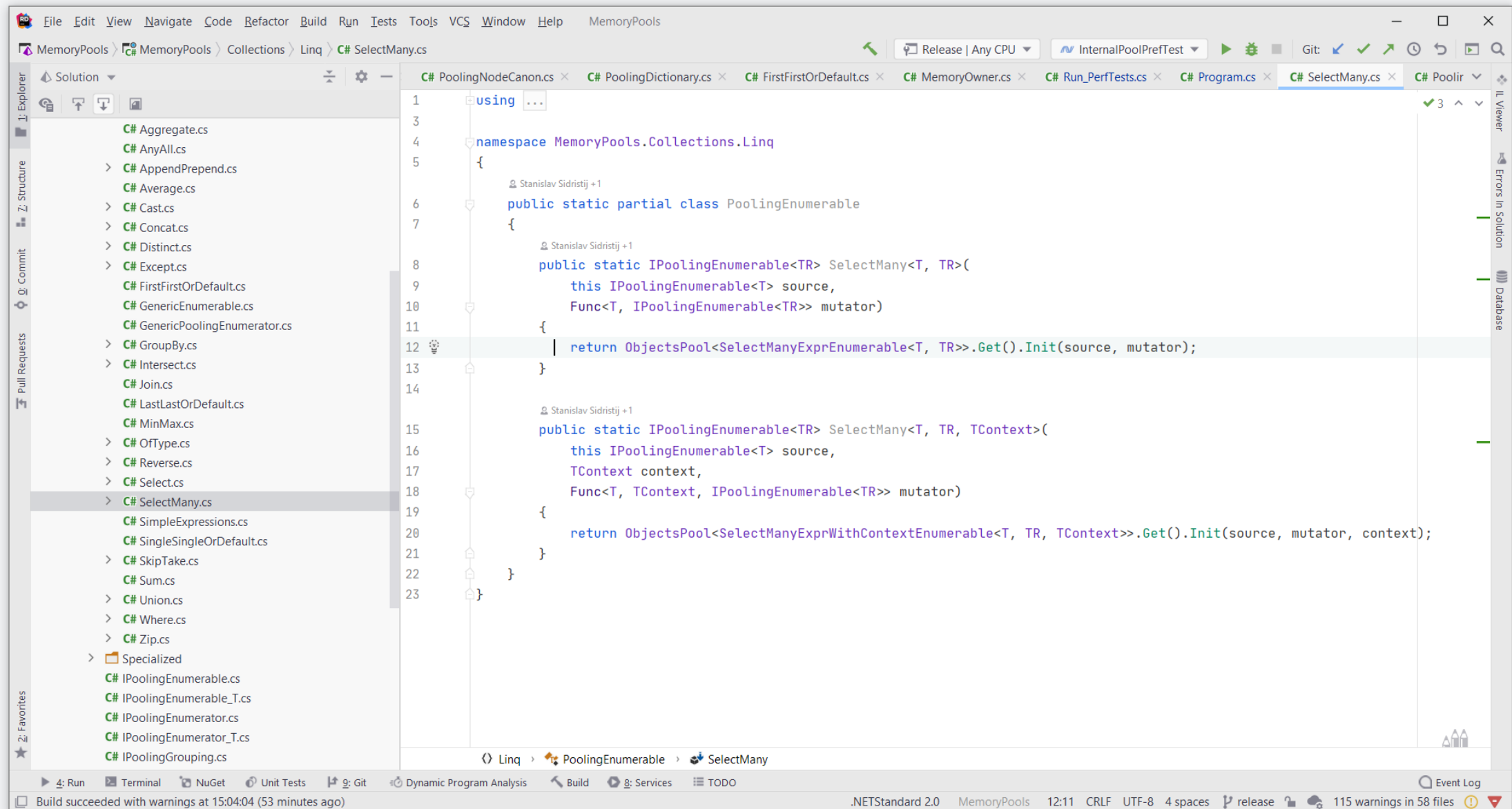


ВДОХНОВЛЕНИЕ

Третий акт — это момент осознания героем себя. Как птица Феникс, он восстает из пепла и стремится к кульминации. В ней герой вступает в отчаянную схватку с антагонистом, достигает (или не достигает) своей цели и далее происходит финал — т.е. развязка истории.

Стандартизация подхода

- Из повседневного трафик идёт от коллекций и LINQ
- Коллекции можно строить сегментами по 128 элементов
- LINQ -> можно пуллить функциональные блоки (Select, SelectMany, GroupBy)
- А потому – почему бы и да?





MemoryPools by: StanislavSidristij

↓ 587 total downloads ⌚ last updated a month ago 📦 Latest version: 1.1.3.2 ↗ linq collections pool performance

Objects pooling, buffers pooling, traffic-free collections (PoolingList, PoolingDictionary, PoolingQueue, PoolingStack), non-allocating LINQ



MemoryPools.Collections by: StanislavSidristij

↓ 249 total downloads ⌚ last updated a month ago 📦 Latest version: 1.1.3.3 ↗ linq collections pool performance

Traffic-free collections (PoolingList, PoolingDictionary, PoolingQueue, PoolingStack), non-allocating LINQ



Выводы

Выводы

- Частые выделения памяти приводят к серьёзным проседаниям производительности;
- Работать с массивами (о, чудо) – надо через Span/Memory;
- Массивы лучше всего гнать через пулинг (не обязательно ArrayPool);
- На нагруженном участке async/await порождает сотни мегабайт трафика;
- IEnumerable можно реализовывать самостоятельно, отдавая его через пул;
- Счётчик ссылок – хорошо для переиспользования буфера;
- Работа через интерфейсы на нагруженном участке ведёт к проседаниям.



MemoryPools by: [StanislavSidristij](#)

↓ 587 total downloads ⌚ last updated a month ago 📦 Latest version: 1.1.3.2 🔗 [linq collections pool performance](#)

Objects pooling, buffers pooling, traffic-free collections (PoolingList, PoolingDictionary, PoolingQueue, PoolingStack), non-allocating LINQ



MemoryPools.Collections by: [StanislavSidristij](#)

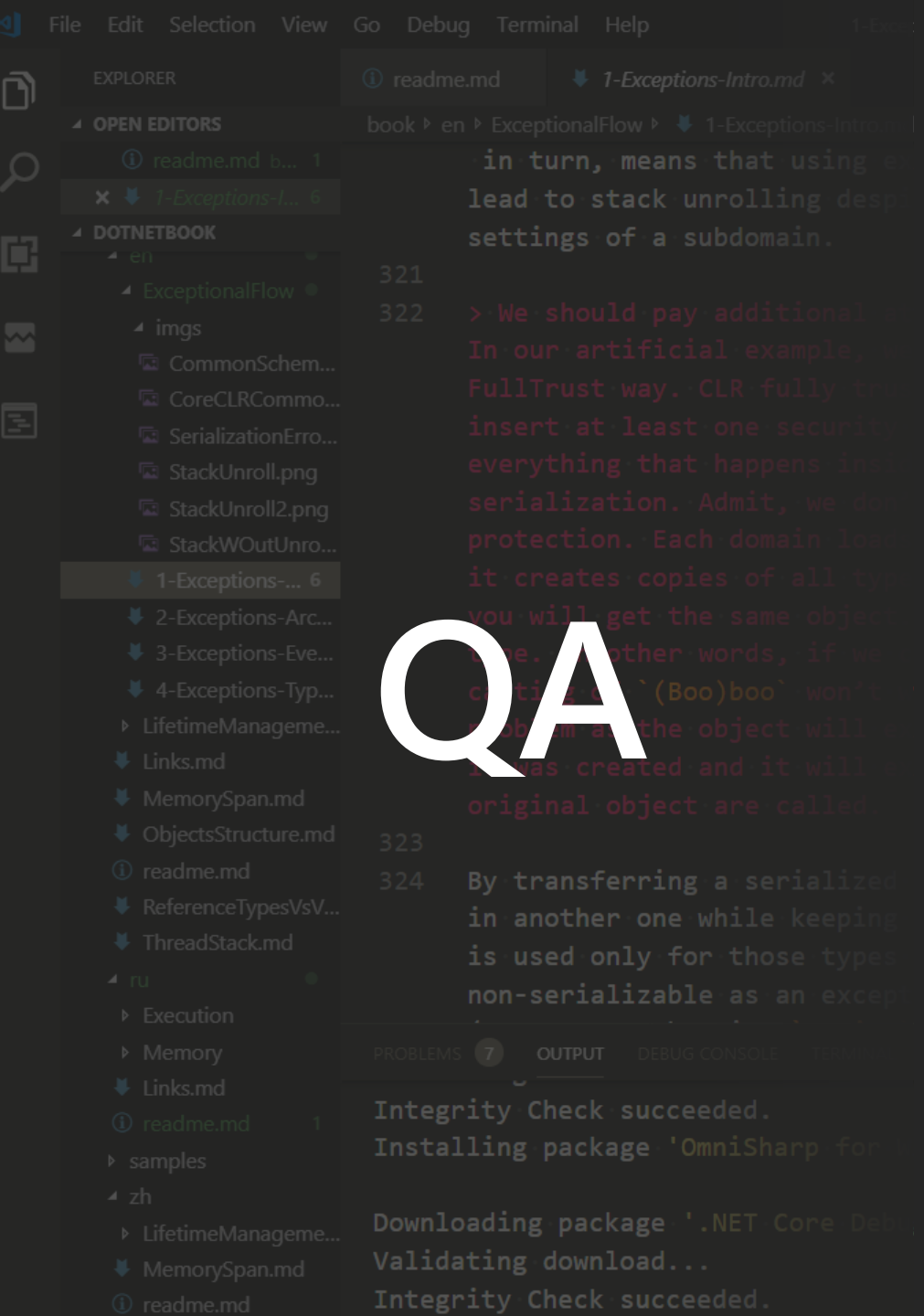
↓ 249 total downloads ⌚ last updated a month ago 📦 Latest version: 1.1.3.3 🔗 [linq collections pool performance](#)

Traffic-free collections (PoolingList, PoolingDictionary, PoolingQueue, PoolingStack), non-allocating LINQ

Как к этому прийти?

- Для начала – попросите немного времени на анализ в dotMemory/dotTrace;
- Далее – ещё немного времени на PoC;
- После этого вы получите доказательства и сможете презентовать решение;
- Иначе ваш внутренний мир не совпадёт с внутренним миром руководства.





Станислав Сидристый

- telegram: @sidristij
- sunex.development@gmail.com