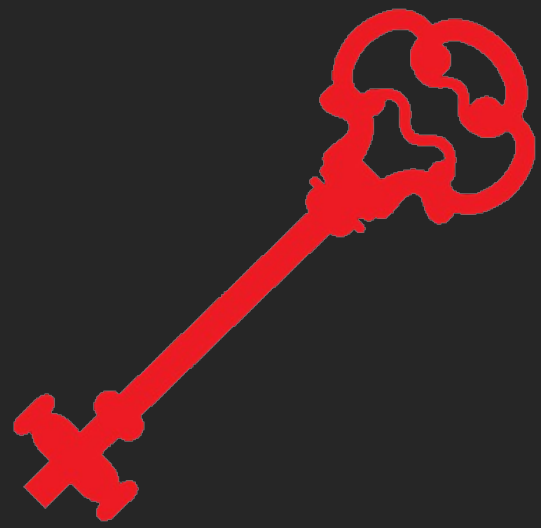
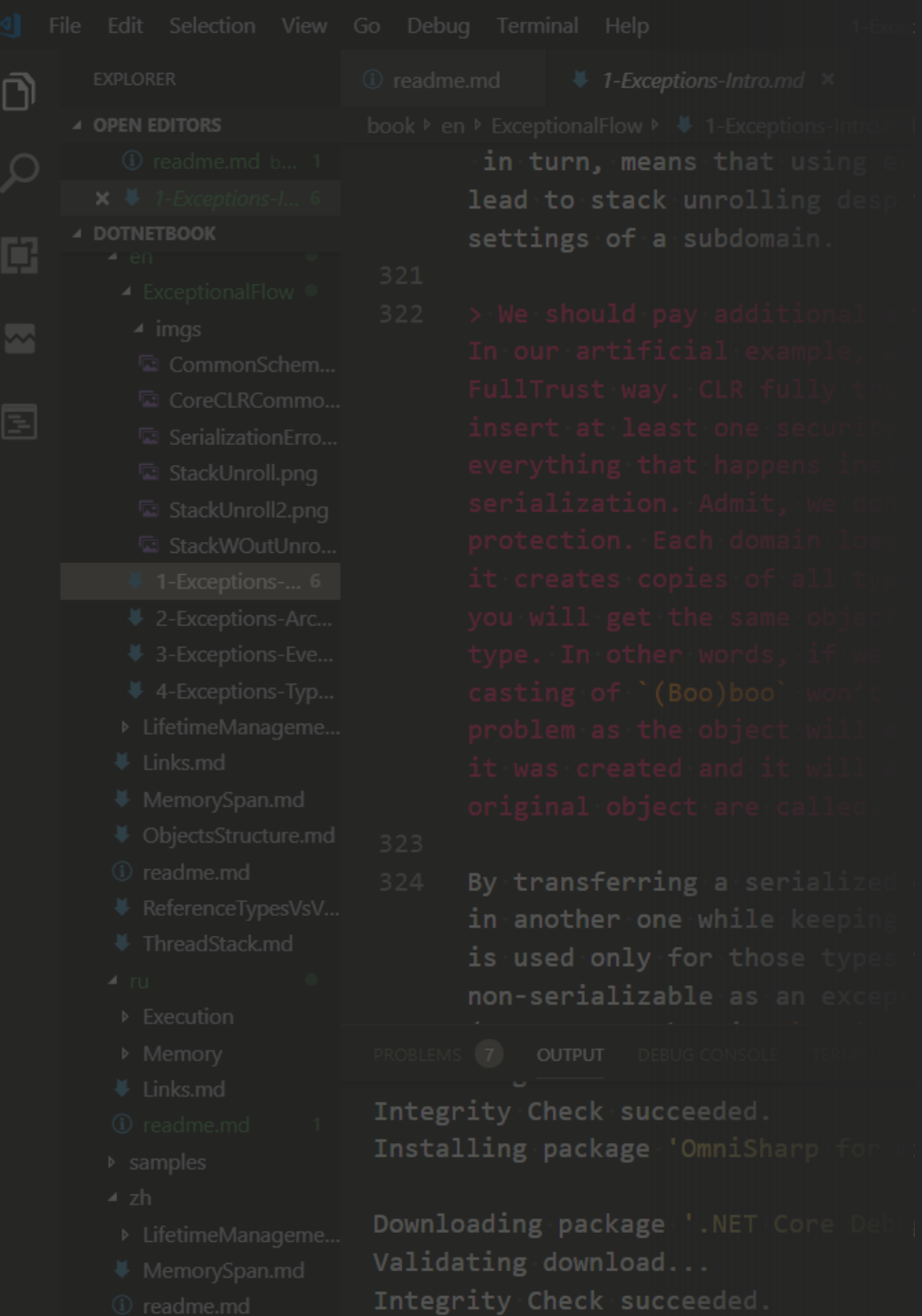
DOTNEXT

Делаем zero-allocation код



Сидристый Станислав



DOTNEXT

Станислав Сидристый

Стеки:

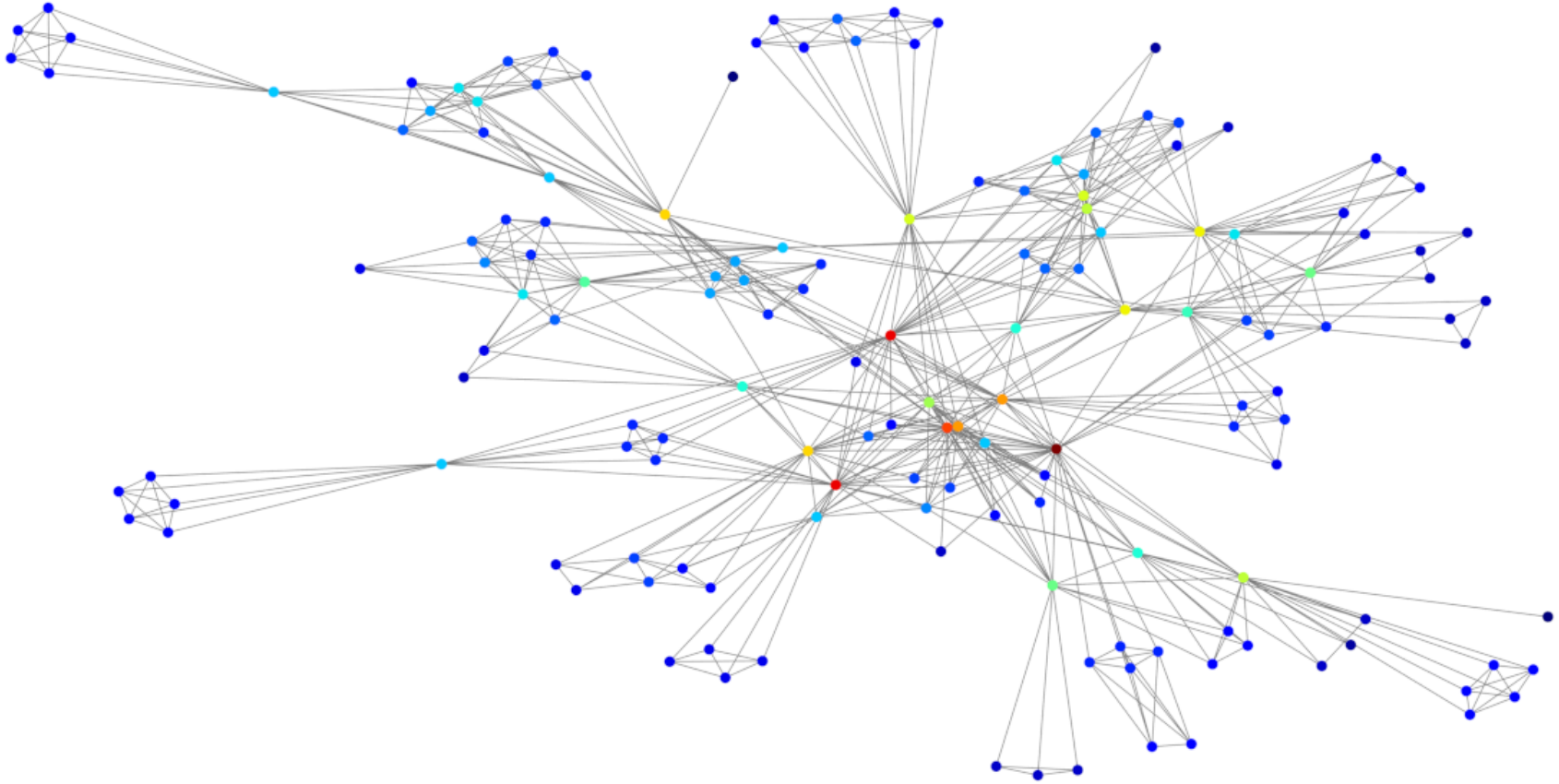
- WEB/WPF/WinForms/... стеки
- C/C++, C++/CLI когда необходимо
- Книга: https://github.com/sidristij/dotnetbook

Связь:

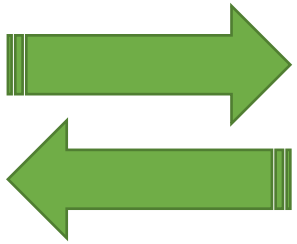
- telegram: @sidristij
- sunex.development@gmail.com

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



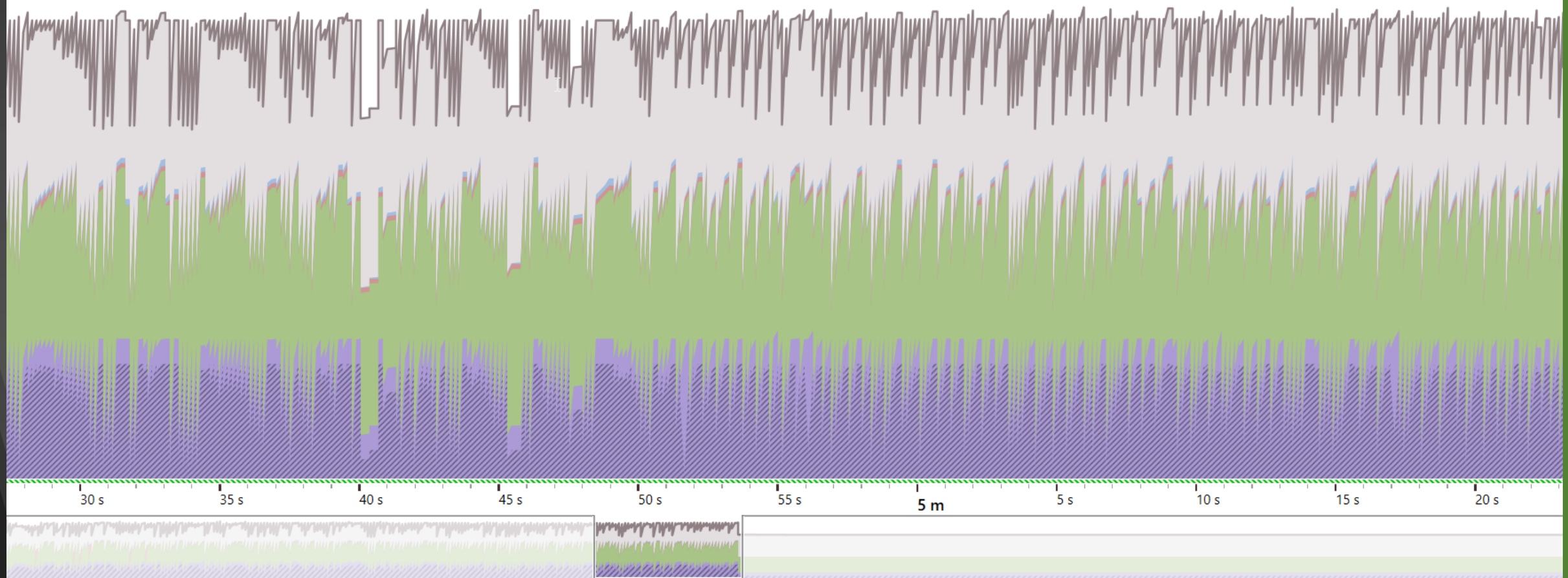
Бывает **много**серверное решение, а бывает – **одно**серверное



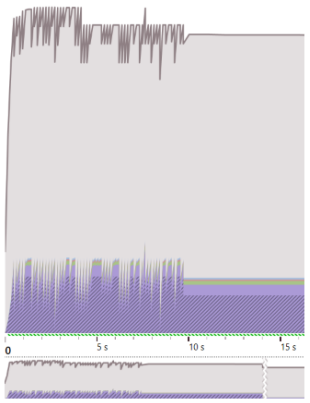
SMB2.0+

При высокой нагрузке на приложение – очень плотная работа с серверами

new T(), new T[N];



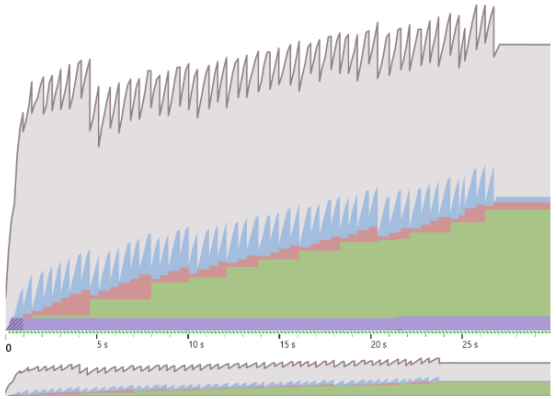
10,000
соединений



7,955 сек.
1,12 Gb

50+ GC_0

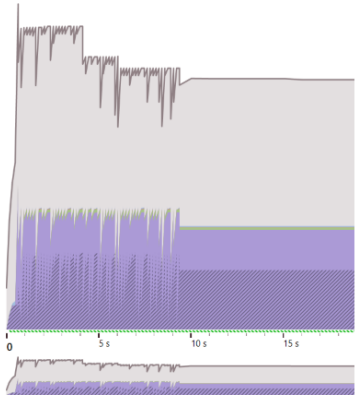
Перечисление 21,000
файлов в сложной структуре
каталогов



26,899 сек.
265,89 Mb

50+ GC_0

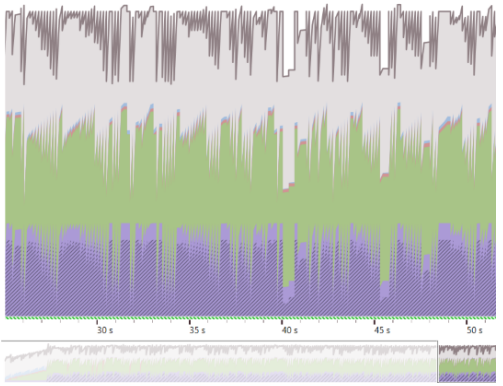
Скачать 200
файлов 16Мб каждый



7,325 сек.
6,49 Gb

50+ GC_0

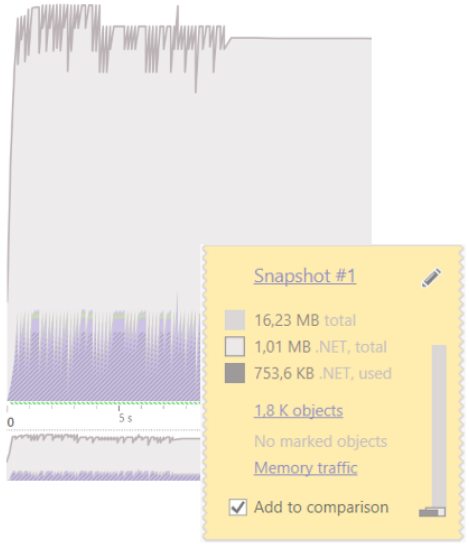
Скачать 21,000
файлов из сложной структуры
каталогов



255,222 сек.
25,12 Gb

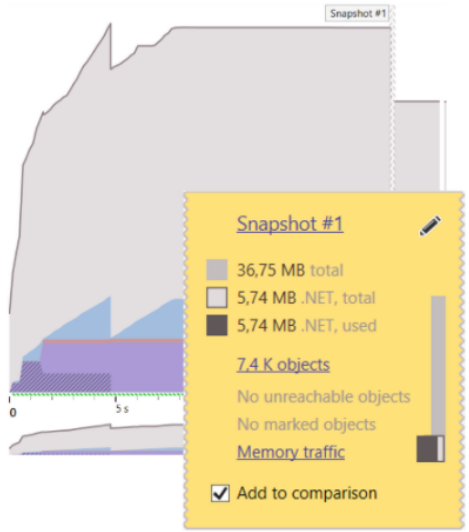
4000+ GC_0

10,000 соединений



7,96 сек.
1,12 Gb

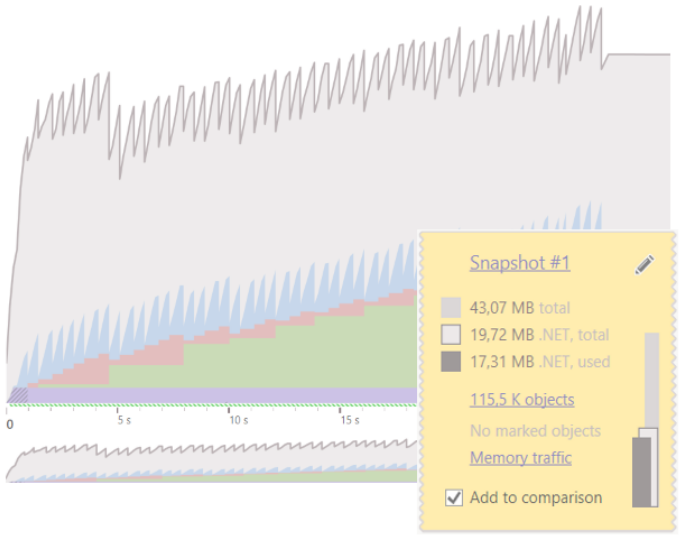
50+ GC_0

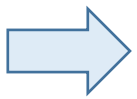


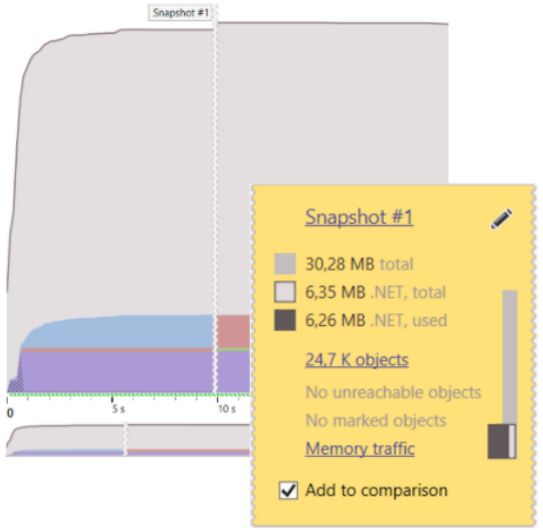
5,10 сек.
8,50 Mb
*System.Cryptography.**

2 GC_0

Перечисление **21,000** файлов в сложной структуре каталогов







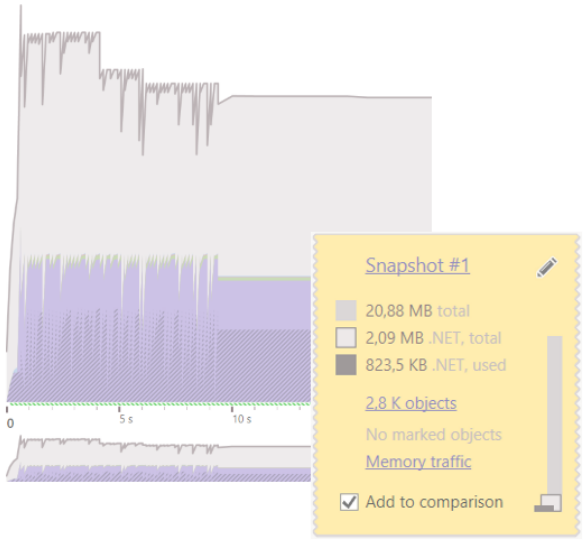
26,899 сек.
265,89 Mb

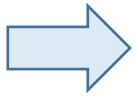
50+ GC_0

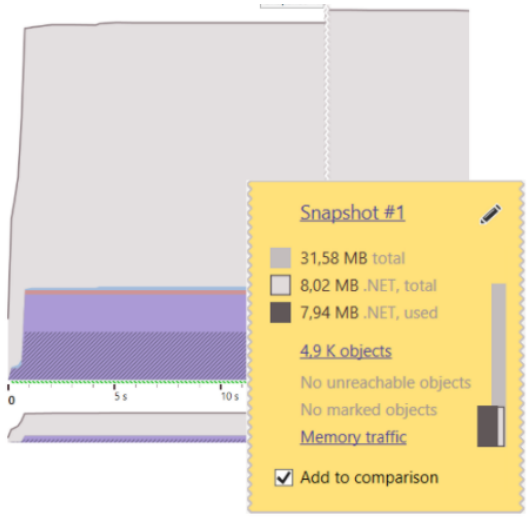
4 сек.
0,005 Mb

0 GC_0

Скачать 200 файлов 16Мб каждый







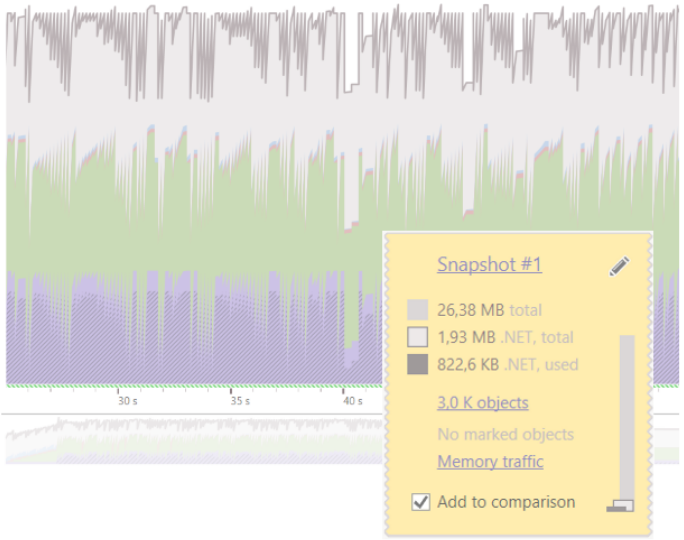
7,325 сек.
6,49 Gb

50+ GC_0

3 сек.
0,016 Mb

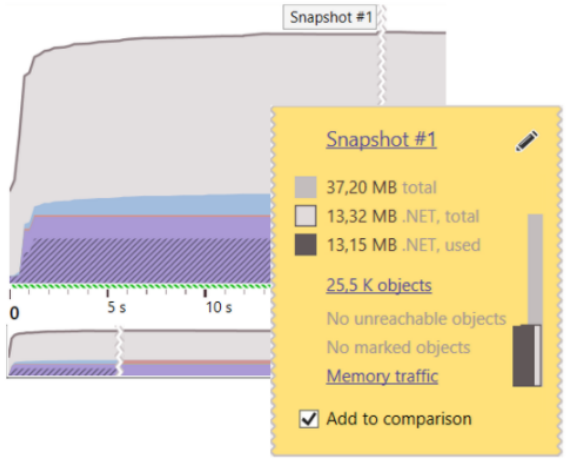
0 GC_0

Скачать **21,000** файлов со сложной структуре каталогов



255,222 сек.
25,15 Gb

4,000+ GC_0



11 сек.
0,021 Mb

0 GC_0

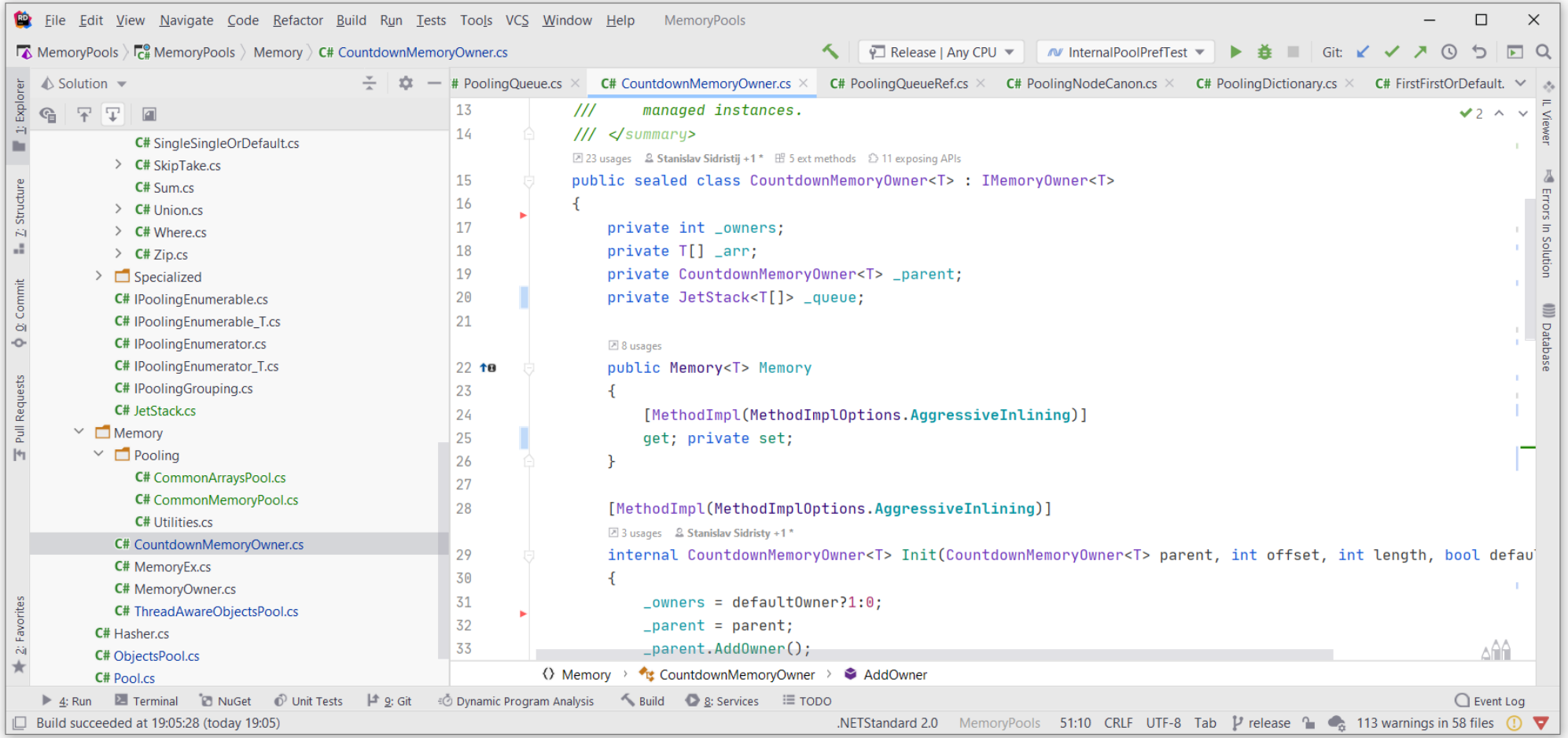
Так что же делать?

Пулинг!

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.*

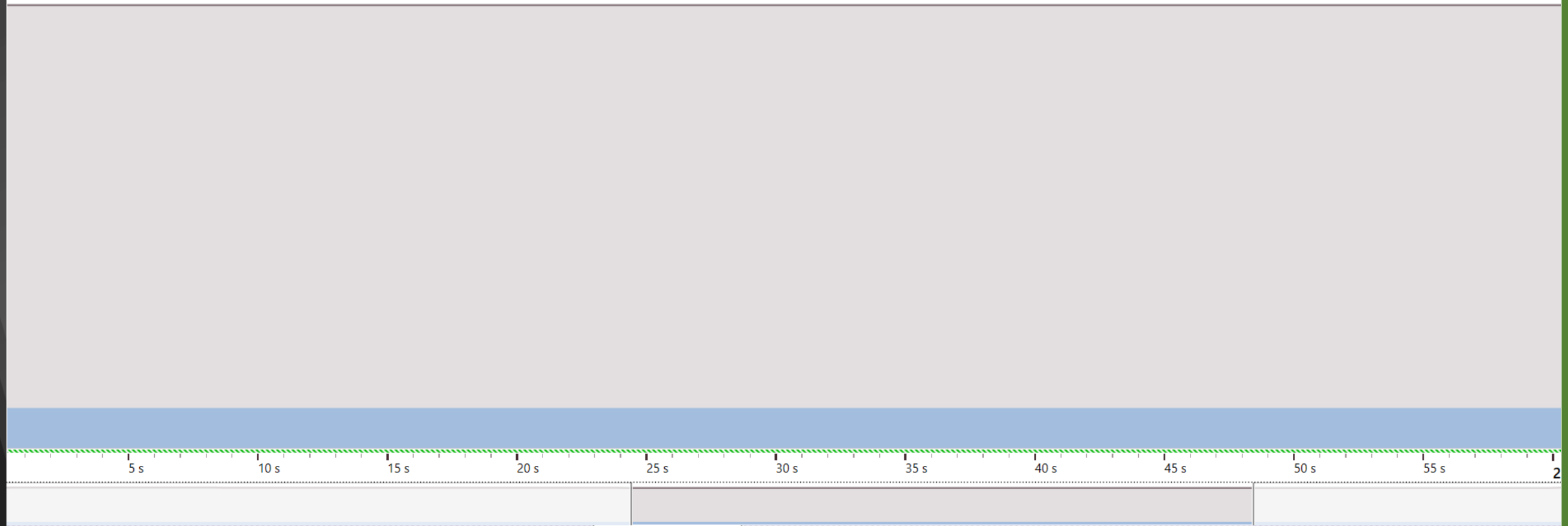
- Пулинг массивов
- IMemoryOwner<T>, Memory<T>, Span<T>
- Пулинг объектов
- Пулинг IAsyncEnumerator
- К чёрту async/await!
- Ускоряем пулинг!!
- Profit!!!


File Edit View Navigate Code Refactor Build Run Tests Tools VCS Window Help MemoryPools
MemoryPools 〉 MemoryPools 〉 Memory 〉 C# CountdownMemoryOwner.cs
Release | Any CPU
InternalPoolPrefTest
Git:
Solution
C# SingleSingleOrDefault.cs
C# SkipTake.cs
C# Sum.cs
C# Union.cs
C# Where.cs
C# Zip.cs
Specialized
C# IPoolingEnumerable.cs
C# IPoolingEnumerable_T.cs
C# IPoolingEnumerator.cs
C# IPoolingEnumerator_T.cs
C# IPoolingGrouping.cs
C# JetStack.cs
Memory
Pooling
C# CommonArraysPool.cs
C# CommonMemoryPool.cs
C# Utilities.cs
C# CountdownMemoryOwner.cs
C# MemoryEx.cs
C# MemoryOwner.cs
C# ThreadAwareObjectsPool.cs
C# Hasher.cs
C# ObjectsPool.cs
C# Pool.cs
PoolingQueue.cs
C# CountdownMemoryOwner.cs
C# PoolingQueueRef.cs
C# PoolingNodeCanon.cs
C# PoolingDictionary.cs
C# FirstFirstOrDefault.
/// managed instances.
/// </summary>
23 usages Stanislav Sidristij +1 * 5 ext methods 11 exposing APIs
public sealed class CountdownMemoryOwner<T> : IMemoryOwner<T>
{
private int _owners;
private T[] _arr;
private CountdownMemoryOwner<T> _parent;
private JetStack<T[]> _queue;
8 usages
public Memory<T> Memory
{
[MethodImpl(MethodImplOptions.AggressiveInlining)]
get; private set;
}
[MethodImpl(MethodImplOptions.AggressiveInlining)]
3 usages Stanislav Sidristy +1 *
internal CountdownMemoryOwner<T> Init(CountdownMemoryOwner<T> parent, int offset, int length, bool defau
{
_owners = defaultOwner?1:0;
_parent = parent;
_parent.AddOwner();
Memory 〉 CountdownMemoryOwner 〉 AddOwner
4: Run Terminal NuGet Unit Tests 9: Git Dynamic Program Analysis Build 8: Services TODO Event Log
Build succeeded at 19:05:28 (today 19:05)
.NETStandard 2.0 MemoryPools 51:10 CRLF UTF-8 Tab release 113 warnings in 58 files
1: Explorer
7: Structure
0: Commit
Pull Requests
2: Favorites
IL Viewer
Errors In Solution
Database

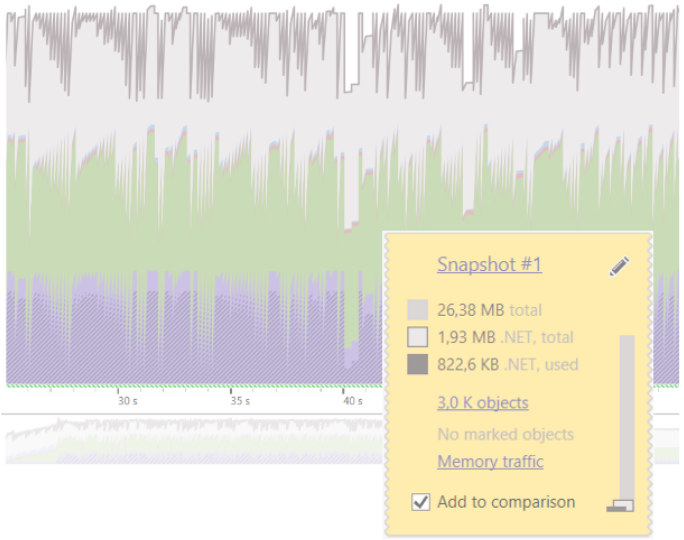
Путь до конца

- 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

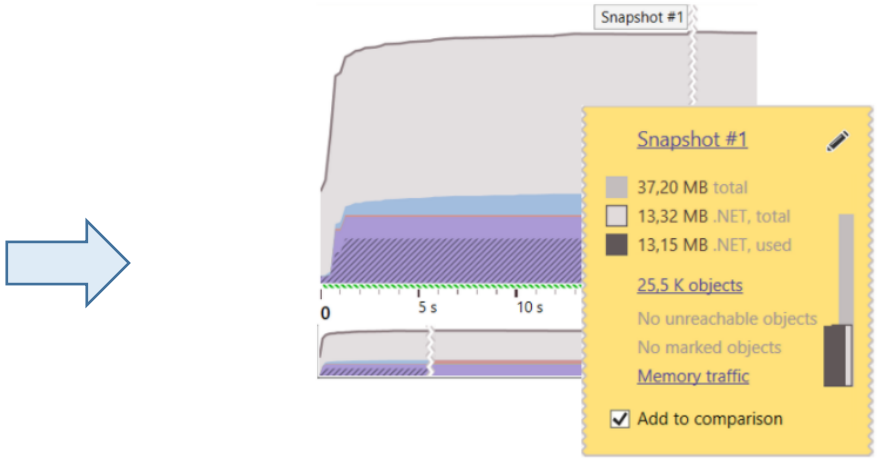


Скачать **21,000** файлов со сложной структуре каталогов



255,222 сек.
25,15 Gb

4,000+ GC_0



11 сек.
0,021 Mb

0 GC_0

Что бы ещё такого сделать?

Коллекции!

LINQ!!

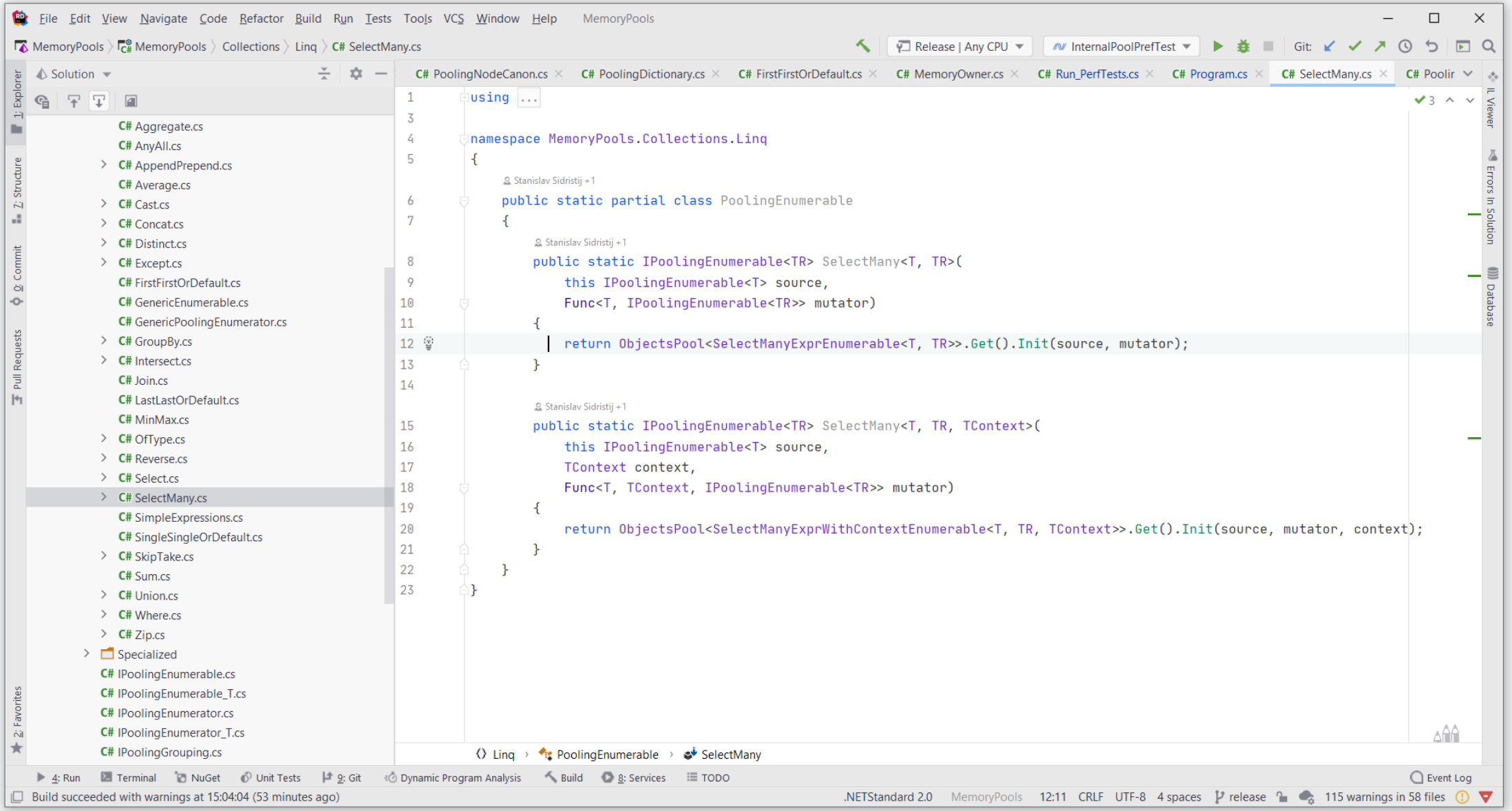

Всё по пулам!!

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?


File Edit View Navigate Code Refactor Build Run Tests Tools VCS Window Help MemoryPools
MemoryPools › MemoryPools › Collections › Linq › SelectMany.cs
Release | Any CPU
InternalPoolPrefTest
Git:
Solution
PoolingNodeCanon.cs
PoolingDictionary.cs
FirstFirstOrDefault.cs
MemoryOwner.cs
Run_PerfTests.cs
Program.cs
SelectMany.cs
Aggregate.cs
AnyAll.cs
AppendPrepend.cs
Average.cs
Cast.cs
Concat.cs
Distinct.cs
Except.cs
FirstFirstOrDefault.cs
GenericEnumerable.cs
GenericPoolingEnumerator.cs
GroupBy.cs
Intersect.cs
Join.cs
LastLastOrDefault.cs
MinMax.cs
OfType.cs
Reverse.cs
Select.cs
SelectMany.cs
SimpleExpressions.cs
SingleSingleOrDefault.cs
SkipTake.cs
Sum.cs
Union.cs
Where.cs
Zip.cs
Specialized
IPoolingEnumerable.cs
IPoolingEnumerable_T.cs
IPoolingEnumerator.cs
IPoolingEnumerator_T.cs
IPoolingGrouping.cs
using ...
namespace MemoryPools.Collections.Linq
{
Stanislav Sidristij +1
public static partial class PoolingEnumerable
{
Stanislav Sidristij +1
public static IPoolingEnumerable<TR> SelectMany<T, TR>(
this IPoolingEnumerable<T> source,
Func<T, IPoolingEnumerable<TR>> mutator)
{
return ObjectsPool<SelectManyExprEnumerable<T, TR>>.Get().Init(source, mutator);
}
Stanislav Sidristij +1
public static IPoolingEnumerable<TR> SelectMany<T, TR, TContext>(
this IPoolingEnumerable<T> source,
TContext context,
Func<T, TContext, IPoolingEnumerable<TR>> mutator)
{
return ObjectsPool<SelectManyExprWithContextEnumerable<T, TR, TContext>>.Get().Init(source, mutator, context);
}
}
}
Linq › PoolingEnumerable › SelectMany
4: Run
Terminal
NuGet
Unit Tests
9: Git
Dynamic Program Analysis
Build
8: Services
TODO
Event Log
Build succeeded with warnings at 15:04:04 (53 minutes ago)
.NETStandard 2.0
MemoryPools
12:11
CRLF
UTF-8
4 spaces
release
115 warnings in 58 files

MemoryPools by: StanislavSidristij

147 total downloads | last updated 18 days ago | Latest version: 1.1.0.1 | linq collections pool performance

Objects pooling, buffers pooling, 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 траффика;
- IAsyncEnumerable 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

147 total downloads last updated 18 days ago Latest version: 1.1.0.1 linq collections pool performance

Objects pooling, buffers pooling, 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.

DOTNEXT

QA

Станислав Сидристый

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