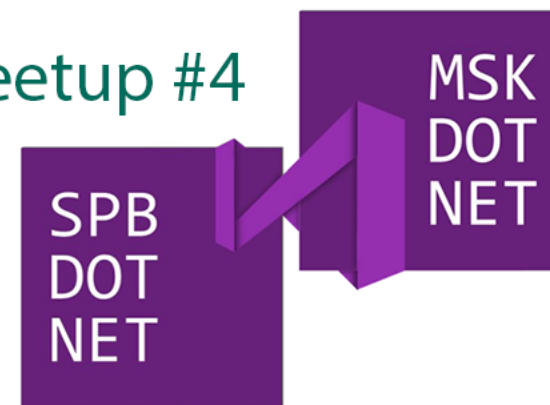




MSK .NET Meetup #4
14.11.2016



18:00 Регистрация участников

19:00 Вступительное слово

19:10 Приемы оптимизации Desktop приложений
на платформе .NET

20:20 Кофе-брейк

20:40 Что полезного в разборе дампов
для .NET- разработчиков?

Юлия Цисык, Никита Цуканов

Сергей Сенцов

Александр Рахманов

MSK DOT NET

 <http://mskdotnet.org>

 <https://vk.com/mskdotnet>

 <https://www.facebook.com/mskdotnet>

 <https://twitter.com/mskdotnet>

 yulia@tsisyk.com

SPB DOT NET

 <http://spbdotnet.org>

 <https://vk.com/spbdotnet>

CoLaboratory: MSK .NET Meetup #4

18:30 Регистрация участников

19:00 Вступительное слово

19:10 Приемы оптимизации Desktop приложений
на платформе .NET

20:20 Перерыв

20:40 Что полезного в разборе дампов для .NET
разработчиков?

Приемы оптимизации Desktop приложений на платформе .NET

Сергей Сенцов

План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

Идеальные требования

- Должно работать на системах от Windows XP до Windows 10
- На системах от Intel Atom до Intel Core I7
- Старт должен быть максимально быстрый
- Поднятие из SWAP должно быть максимально быстрым
- Переходы между страницами приложения не более 500 мс

Т.е. в идеале мы должны занимать 0 Мб в памяти и тратить 0 CPU time.

План

- Под какие требования оптимизируем
- **Используемый инструментарий для анализа performance**
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

Основные способы анализа

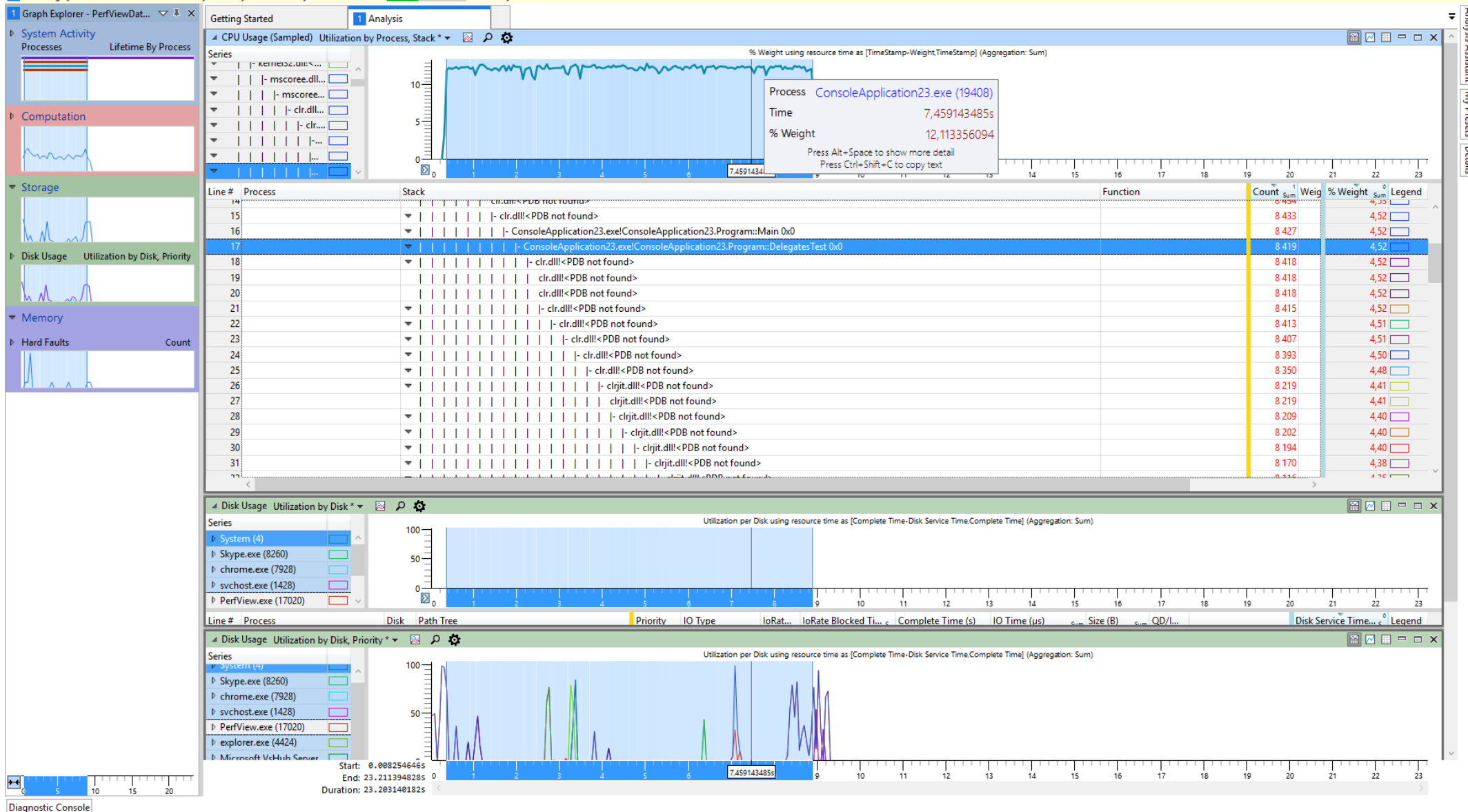
- Profilers (dotTrace, ANTS...)
- Performance Counters
- Benchmarking (StopWatch, BenchmarkDotNet ...)
- Aspect Oriented Programming AOP (PostSharp, Aspect.Net...)
- **Event Tracing for Windows (ETW)**
- Windows Management Instrumentation (WMI)
- Profiling API
- Debug API
- WinDbg
- Camera
- ...

Windows Performance Analyzer

PerfViewData.etl - Windows Performance Analyzer

File Trace Profiles Window Help

1 Loading symbols - You can continue with your analysis while the symbols are loaded 627 symbols found



Common ETW for .NET

The CLR runtime provider GUID is e13c0d23-ccbc-4e12-931b-d9cc2eee27e4.

- Runtime Information ETW Events
- Exception Thrown_V1 ETW Event
- Contention ETW Events
- Thread Pool ETW Events
- **Loader ETW Events**
- Method ETW Events
- Garbage Collection ETW Events
- **JIT Tracing ETW Events**
- Interop ETW Events
- Application Domain Resource Monitoring (ARM) ETW Events
- Security ETW Events
- Stack ETW Event

Capture CLR ETW events using PrefView

Collecting ETW Data while running a command

This dialog give displays options for collecting ETW profile data. The only required field the 'Command' field and this is only necessary when using the 'Run' command.

If you wish to analyze on another machine use the Zip option when collecting data. See [Collecting ETW Profile Data](#), for more.

Command:

Data File: ...

Current Dir:

Zip: ☐ Circular MB: Merge: ☐ Thread Time: ☐ Mark Text: Mark

Status:

Advanced Options

Kernel Base: <input checked="" type="checkbox"/>	Cpu Samples: <input checked="" type="checkbox"/>	Page Faults: <input type="checkbox"/>	File I/O: <input type="checkbox"/>	Registry: <input type="checkbox"/>	VirtAlloc: <input type="checkbox"/>	MemInfo: <input type="checkbox"/>
Handle: <input type="checkbox"/>	RefSet: <input type="checkbox"/>	IIS: <input type="checkbox"/>	NetMon: <input type="checkbox"/>	Net Capture: <input type="checkbox"/>		
.NET: <input checked="" type="checkbox"/>	.NET Stress: <input type="checkbox"/>	Background JIT: <input type="checkbox"/>	.NET Calls: <input type="checkbox"/>	JIT Inlining: <input type="checkbox"/>		
GC Collect Only: <input type="checkbox"/>	GC Only: <input type="checkbox"/>	.NET Alloc: <input type="checkbox"/>	.NET SampAlloc: <input type="checkbox"/>	ETW .NET Alloc: <input type="checkbox"/>	Dump Heap: <input type="checkbox"/>	Finalizers: <input type="checkbox"/>

Additional Providers: ?

CPU Sample Interval Msec: Cpu Ctrs: OS Heap Exe: OS Heap Process:

.NET Symbol Collection: ☐ No V3.X NGEN Symbols: ☐ Symbol TimeOut:

Max Collect Sec: Stop Trigger:

PerfView

Managed Load Stacks(3 metric) PerfViewData.etl.zip in Xperf (C:\ForVMWare\Xperf\PerfViewData.etl.zip)

File Diff Regression Help Stack View Help (F1) Understanding Perf Data Starting an Analysis Troubleshooting Tips

Update Back Forward Totals Metric: 3,0 Count: 3,0 First: 342,308 Last: 352,087 Last-First: 9,779 Metric/Interval: 0,31 TimeBucket: 11,0

Start: 0 End: 352,087 Find:

GroupPats: [group module entries] {%}!=>modul Fold%: FoldPats: ntoskrnl!%ServiceCopyEnd IncPats: Process% ConsoleApplicati ExcPats:

By Name ? Caller-Callee ? CallTree ? Callers ? Calleees ? Notes ?

Name ?	Inc % ?	Inc ?	Inc C	Exc %	Exc ?	Exc C	Fold	Fold	Whe	First	Last
<input checked="" type="checkbox"/> ROOT	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> Process32 ConsoleApplication23 (19408)	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> Thread (4240) CPU=8532ms (Startup Thread)	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> module ntdll <<ntdll!>>	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> module kernel32 <<kernel32!>>	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> module mscoree <<mscorlib!>>	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> module mscoreei <<mscorlib!>>	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> module clr <<clr!>>	100,0	3,0	3	0,0	0	0	0	0	0	342,30	352,08
+ <input checked="" type="checkbox"/> Load C:\WINDOWS\Microsoft.Net\assembly\GAC_32\mscorlib\v4.0_4.0.0.0_b77a5c561934e089\mscorlib.dll	33,3	1,0	1	33,3	1	1	0	0	0	342,30	342,30
+ <input checked="" type="checkbox"/> module <<Load C:\InWork\ReflectionVsDelegates\ConsoleApplication23\bin\Release\ConsoleApplication23.dll>>	33,3	1,0	1	33,3	1	1	0	0	0	343,96	343,96
+ <input checked="" type="checkbox"/> module <<consoleapplication23!ConsoleApplication23.Program.Main(class System.String[])>>	33,3	1,0	1	0,0	0	0	0	0	0	352,08	352,08
+ <input checked="" type="checkbox"/> module clr <<clr!>>	33,3	1,0	1	0,0	0	0	0	0	0	352,08	352,08
+ <input checked="" type="checkbox"/> module clrjit <<clrjit!>>	33,3	1,0	1	0,0	0	0	0	0	0	352,08	352,08
+ <input checked="" type="checkbox"/> module clr <<clr!>>	33,3	1,0	1	0,0	0	0	0	0	0	352,08	352,08
+ <input checked="" type="checkbox"/> module <<Load C:\InWork\ReflectionVsDelegates\ConsoleApplication23\bin\Release\ClassLibrary1.dll>>	33,3	1,0	1	33,3	1	1	0	0	0	352,08	352,08

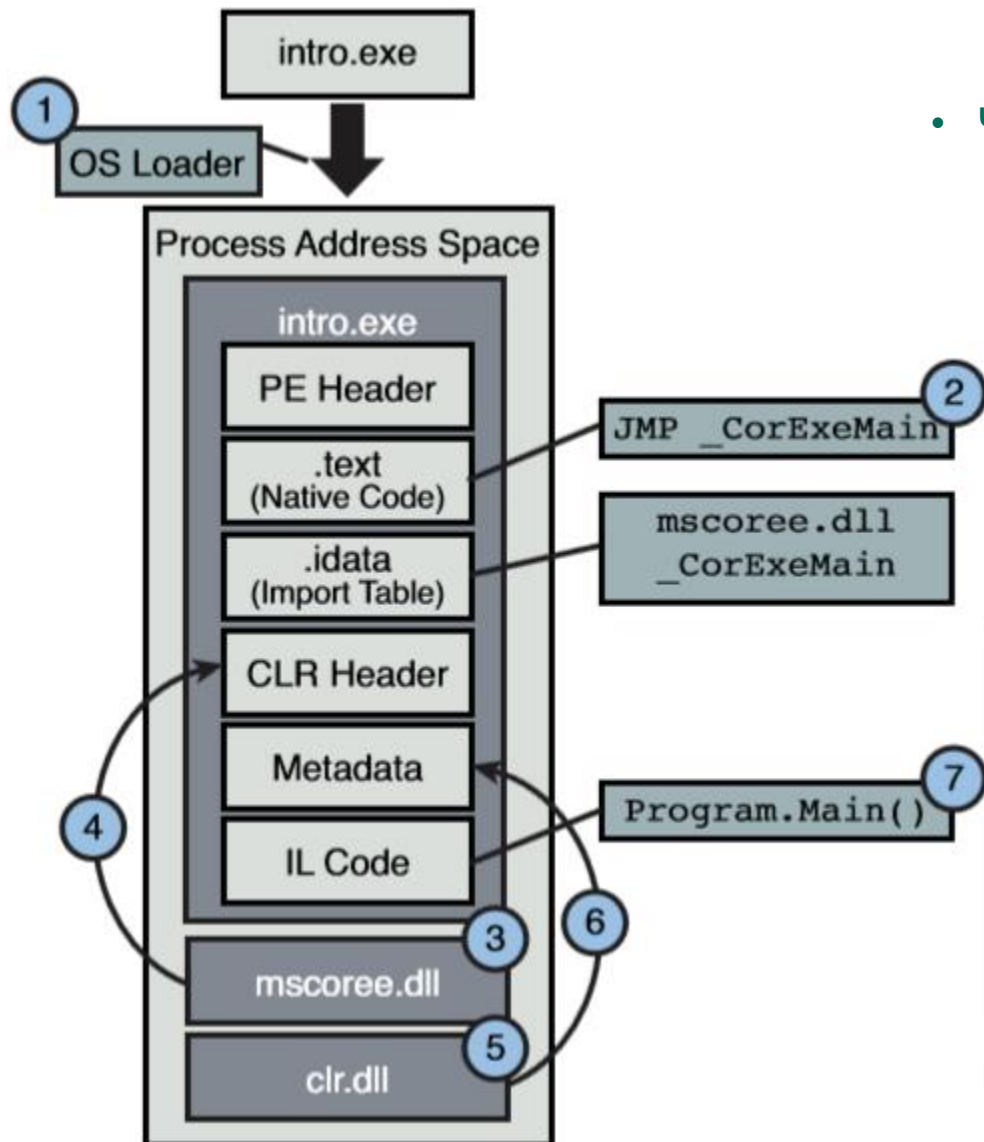
Cell Contents: module <<Load C:\InWork\ReflectionVsDelegates\ConsoleApplication23\bin\Release\ClassLibrary1.dll>>

Ready Log Cancel

План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

Оптимизация запуска .NET-приложения

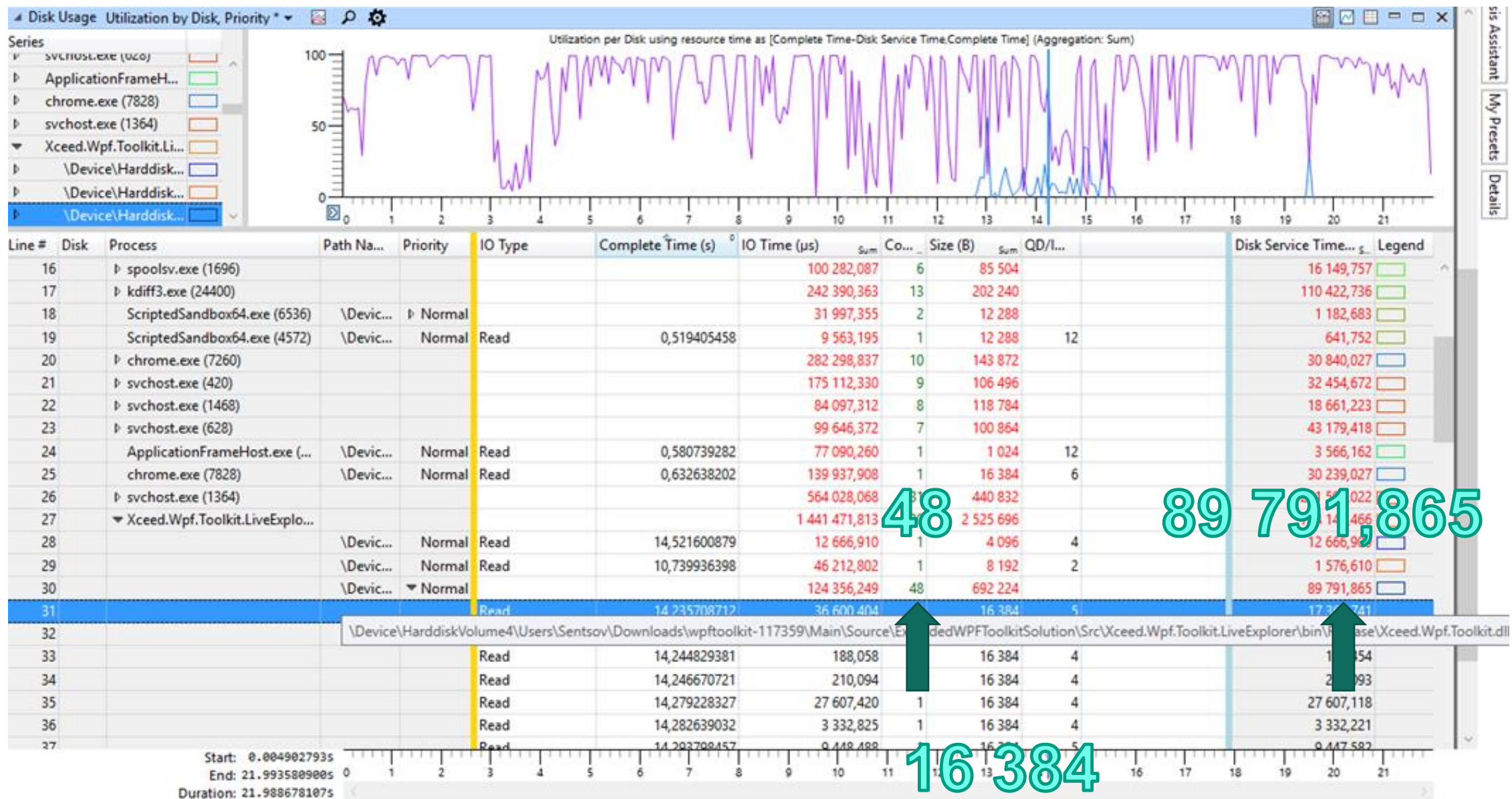


- Что из себя представляет запуск .Net приложения
 - Инфраструктура CLR
 - Старт CLR
 - Загрузка основных CLR и native сборок
 - Вызов EntryPoint
 - Пользовательский код
 - Подгрузка зависимых сборок по мере необходимости
 - Jitting

Loading the CLR

1. Executable loaded by OS
2. Loader stub to CLR entry point
3. CLR shim initialization
4. Inspect headers and policy
5. Shim loads specific CLR version
6. Finding managed entry point
7. Control transfer to managed code

Как CLR грузит сборки



Как можно ускорить загрузку сборок

1) Мы знаем все зависимости, нужно найти конкретные сборки на диске

- Сбоки приложения ищем в

```
AppDomain.CurrentDomain.SetupInformation.ApplicationBase;
```

- Сборки .NetFramework ищем с помощью fusion.dll получаем IAssemblyCache и с помощью метода QueryAssemblyInfo получаем структуру GacAssemblyInfo

```
IAssemblyCache assemblyCache;
```

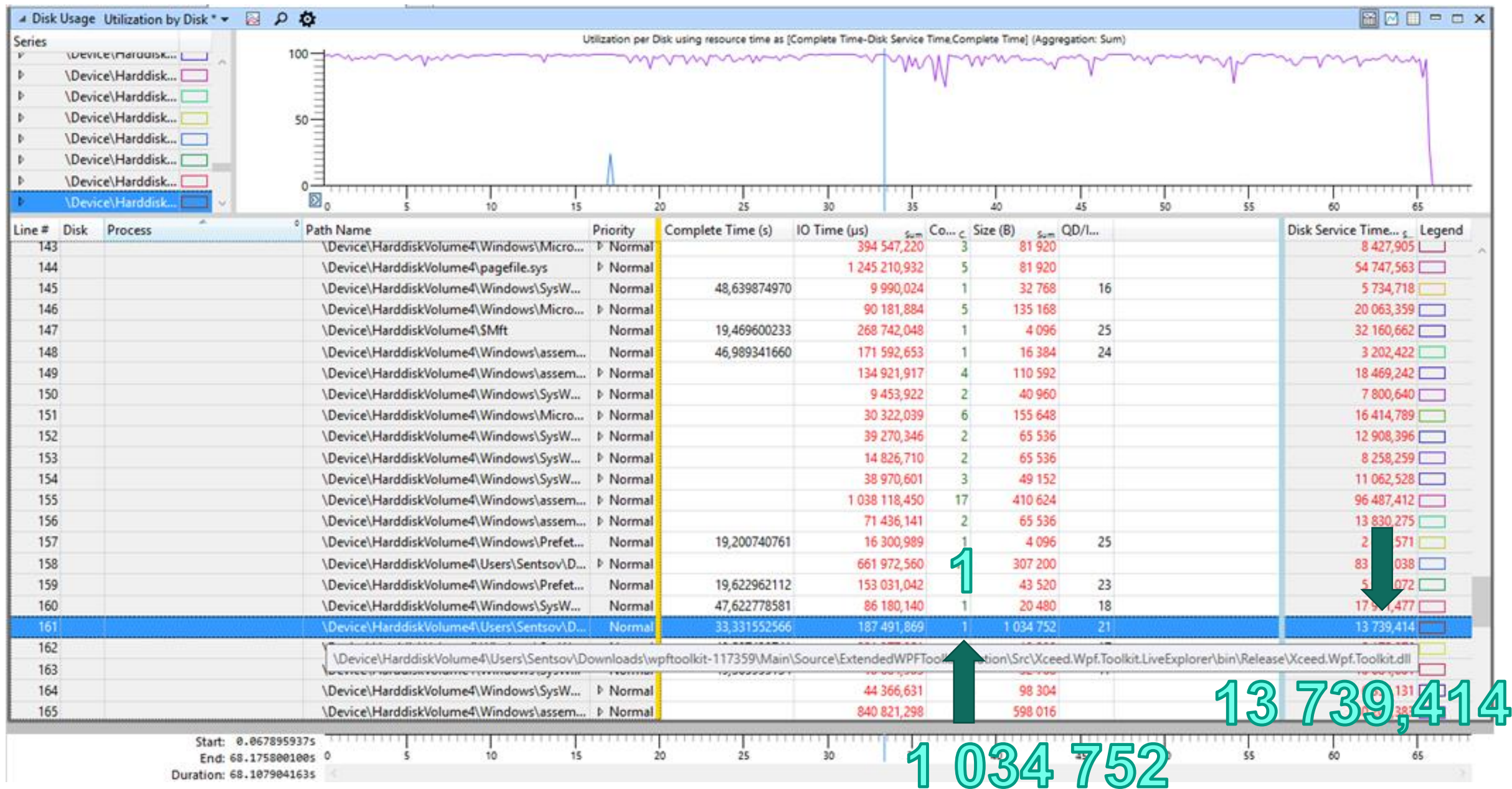
```
hr1 = FusionDll.CreateAssemblyCache(out assemblyCache, 0);
```

```
hr2 = assemblyCache.QueryAssemblyInfo(0, assemblyName, ref info);
```

2) Зачитаем сборки большими блоками

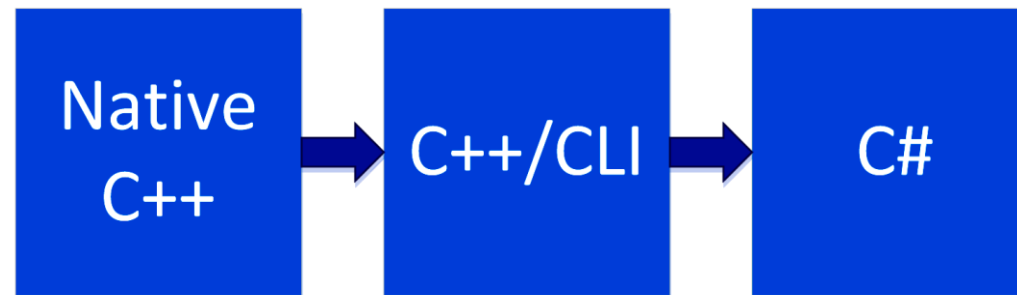
```
var handle = stream.SafeFileHandle.DangerousGetHandle();
```

```
ReadFile(handle, buffer, bytesToRead, out read, IntPtr.Zero);
```

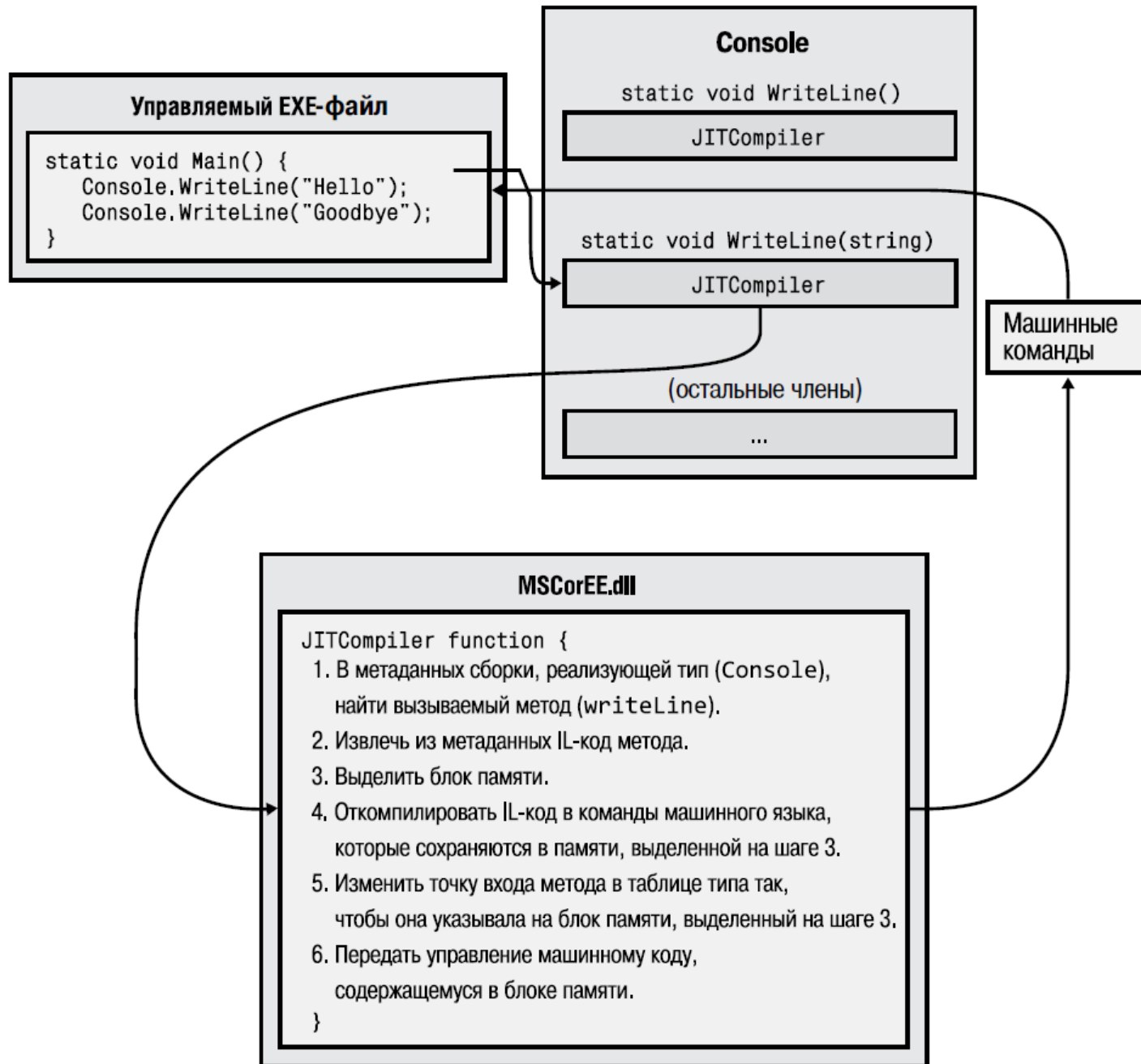
Сравнение разных видов загрузки

	Sec	Disk Service Time	IO Time	IO Count	Size	IOPS
WPF (W/O Native Images)	7,8	5477	6338	1504	25354	193
WPF (Native Images)	6,1	5845	6923	1210	25202	198
WPF (Native Launcher)	4,5	2140	3459	437	41585	97
Console (Main Loop Only)	2,2	1849	1995	462	9989	210
Native	0,2	72	124	23	514	115



План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI



Multicore JIT

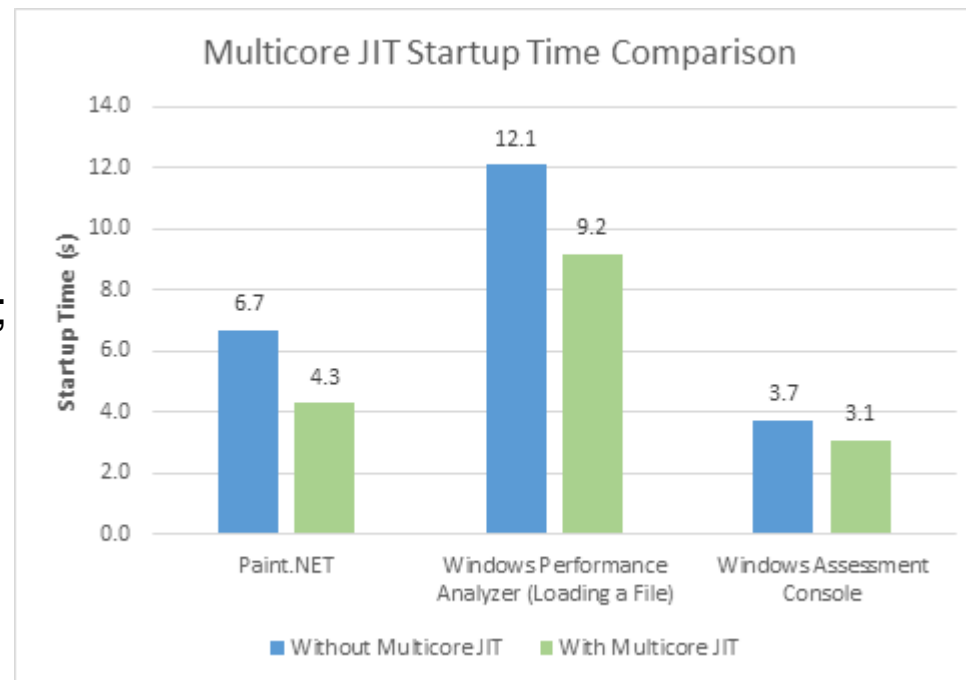
- **Плюсы**

- Работает из коробки
- API

```
ProfileOptimization.SetProfileRoot(@"C:\MyAppFolder");  
ProfileOptimization.StartProfile("Startup.Profile");
```

- **Минусы**

- .Net 4.5+
- Начинает работать со второго запуска
- Нельзя распространять вместе с приложением
- Нельзя управлять
- Если приложение завершается некорректно, то профиль не создается



NGen

- Плюсы
 - Не тратится время на Jit при работе
- Минусы
 - Требуется инсталляция
 - Требуются права администратора
 - После обновления сборок приложения или .NetFramework требуется перезапустить NGen для всех сборок
 - Jit может генерировать более эффективный код чем Ngen
 - У NI сборок отсутствует подпись

Ручной Jitting

```
public class MyClass { public MyClass() { } }
```

```
RuntimeHelpers.PrepareMethod(  
    typeof(MyClass).GetConstructor().MethodHandle);
```

```
public class MyClass<T>  
{ public T Method<TT>(){return default(T);} }
```

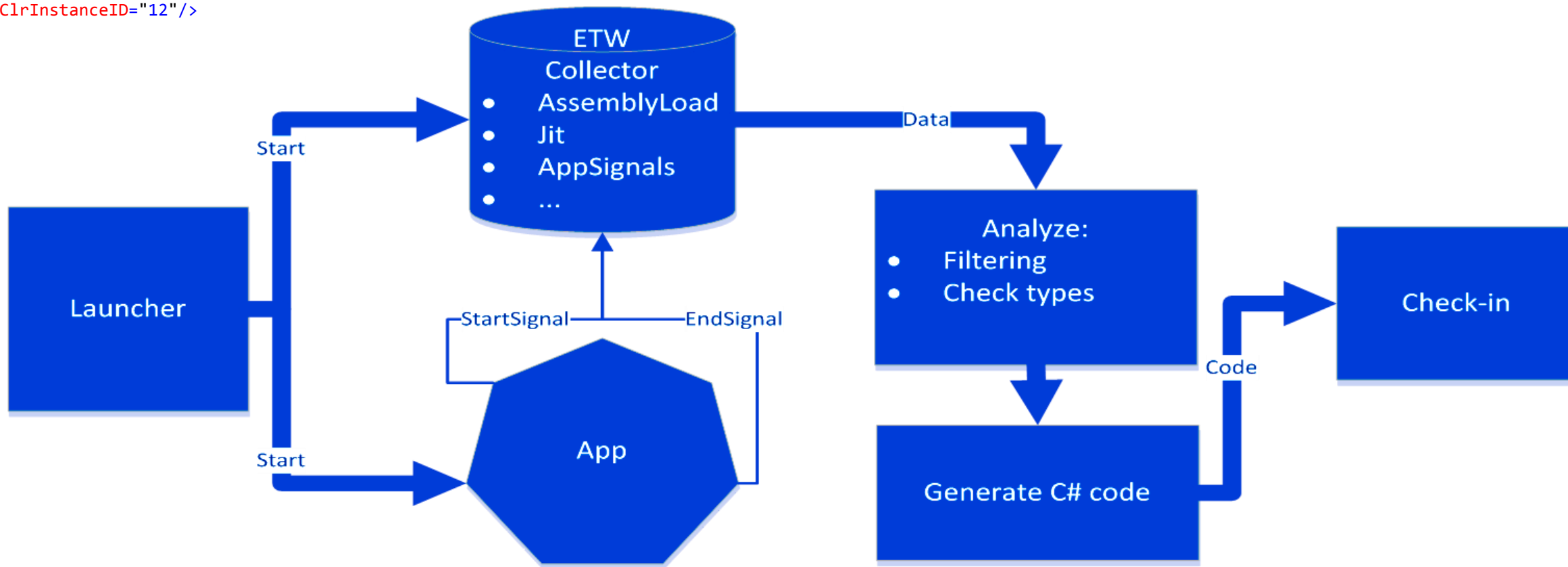
```
RuntimeHelpers.PrepareMethod(  
    typeof(MyClass<>).GetMethods().First().MethodHandle,  
    new RuntimeTypeHandle[]  
        {typeof(int).TypeHandle, typeof(long).TypeHandle});
```

```
typeof(int).Assembly.GetType("System.__Canon");
```

Автогенерация профиля

```
<Event MSec= "5781,2778" PID="8840" PName="" TID="11268" EventName="Loader/ModuleLoad" ModuleID="0x1B263068" AssemblyID="0x1138F620" ModuleFlags="Manifest"
ModuleILPath="C:\Windows\Microsoft.Net\assembly\GAC_MSIL\Accessibility\v4.0.0.0__b03f5f7f11d50a3a\Accessibility.dll" ModuleNativePath=""
ManagedPdbSignature="f9f3e4d9-2685-4e4b-a9e9-96657ae3425d" ManagedPdbAge="1" ManagedPdbBuildPath="Accessibility.pdb"/>
```

```
<Event MSec= "159,8282" PID="8840" PName="" TID="1192" EventName="Method/JittingStarted" MethodID="0x00D695AC" ModuleID="0x00A24010" MethodToken="0x06000281"
MethodILSize="0x00000034" MethodNamespace="System.Array" MethodName="Copy" MethodSignature="void (class System.Array,class System.Array,int32)"
ClrInstanceID="12"/>
```



```
Add(typeof(Microsoft.Win32.FileDialogCustomPlace).Assembly.GetType("System.Windows.Data.BindingBase"), new string[] { "get_BindingGroupName" });
Add(typeof(Microsoft.Win32.RegistryHive).Assembly.GetType("System.Threading.WaitHandle"), new string[] { "WaitOne", "Close", "Dispose", "WaitAny" });
```

Результаты оптимизации на примере нашего продукта

	Mode	Result	Difference
High Performance PC	None	26s	8s (31%)
High Performance PC	Warmup + Jit	18s	
Average Performance PC	None	45s	13s (29%)
Average Performance PC	Warmup + Jit	32s	
Low Performance PC	None	168s	51s (30%)
Low Performance PC	Warmup + Jit	117s	

План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

IoC (DI) для быстрого старта

- Быстрая регистрация
- Поддержка потокобезопасной регистрации
- Быстрый **первый Resolve**

```
public Func<object[], object> BuildConstructor(ConstructorInfo ctorInfo)
{
    var ctorParams = ctorInfo.GetParameters();
    var dynamicMethod = new DynamicMethod("Create_" + ctorInfo.Name, ctorInfo.DeclaringType, new[] { typeof(object[]) });
    var ilgen = dynamicMethod.GetILGenerator();

    for (int i = 0; i < ctorParams.Length; i++)
    {
        ilgen.Emit(OpCodes.Ldarg_0);
        ilgen.Emit(OpCodes.Ldc_I4, i);
        ilgen.Emit(OpCodes.Ldelem_Ref);
        var type = ctorParams[i].ParameterType;
        ilgen.Emit(type.IsValueType ? OpCodes.Box : OpCodes.Castclass, type);
    }

    ilgen.Emit(OpCodes.Newobj, ctorInfo);
    ilgen.Emit(OpCodes.Ret);

    return (Func<object[], object>)dynamicMethod.CreateDelegate(typeof(Func<object[], object>));
}
```

BenchmarkDotNet.Core=v0.9.9.0

Processor=Intel(R) Core(TM) i7-3770 CPU 3.40GHz, ProcessorCount=8

CLR=MS.NET 4.0.30319.42000,

Arch=32-bit RELEASE \ Arch=64-bit RELEASE [RyuJIT]

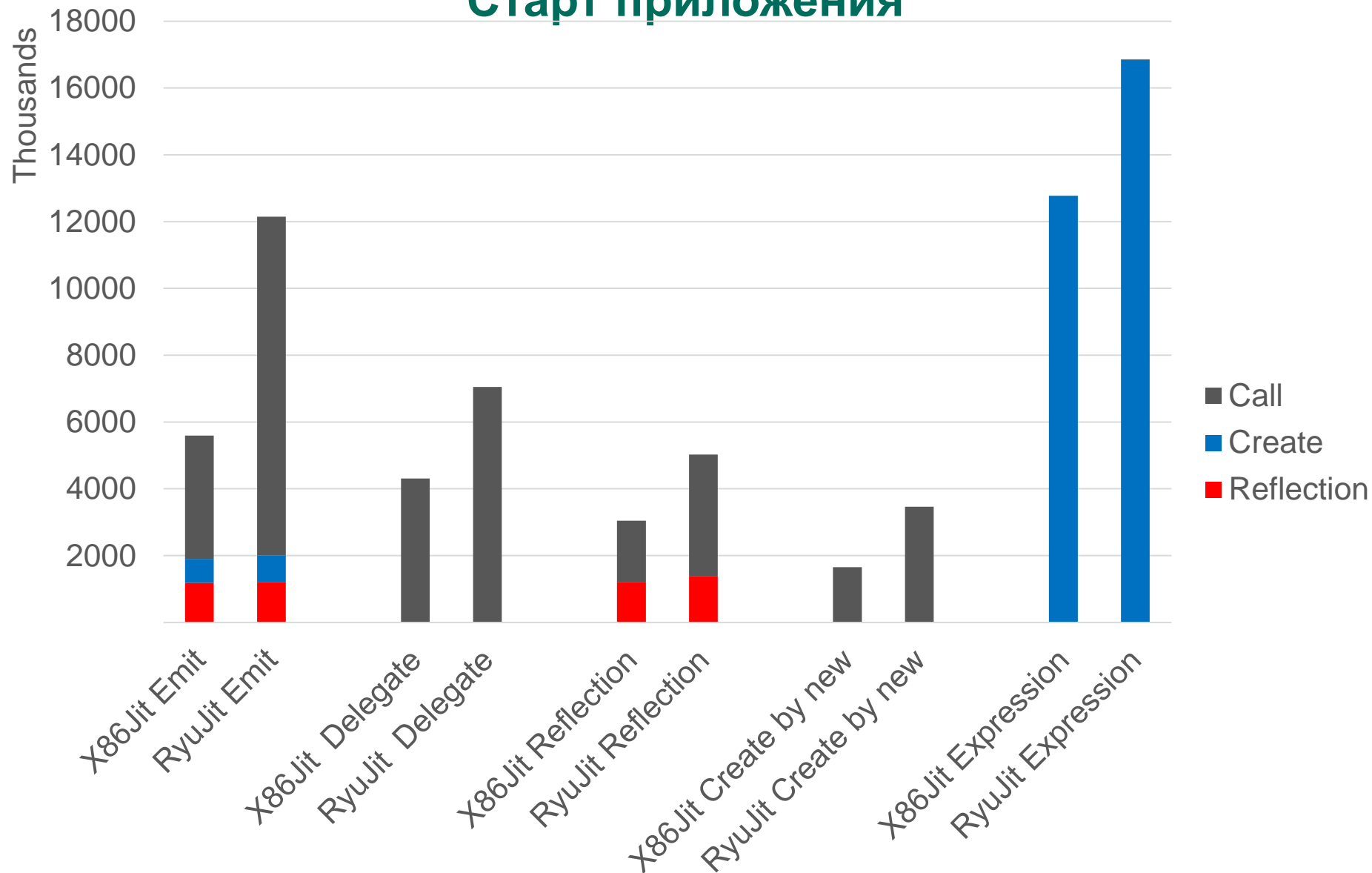
	X86Jit	RyuJit
Delegates	646 us	1 041 us
Reflection	26 640 us	36 719 us
New	366 us	497 us
Emit	422 198 us	798 021 us
Expression	724 744 us	1 029 723 us

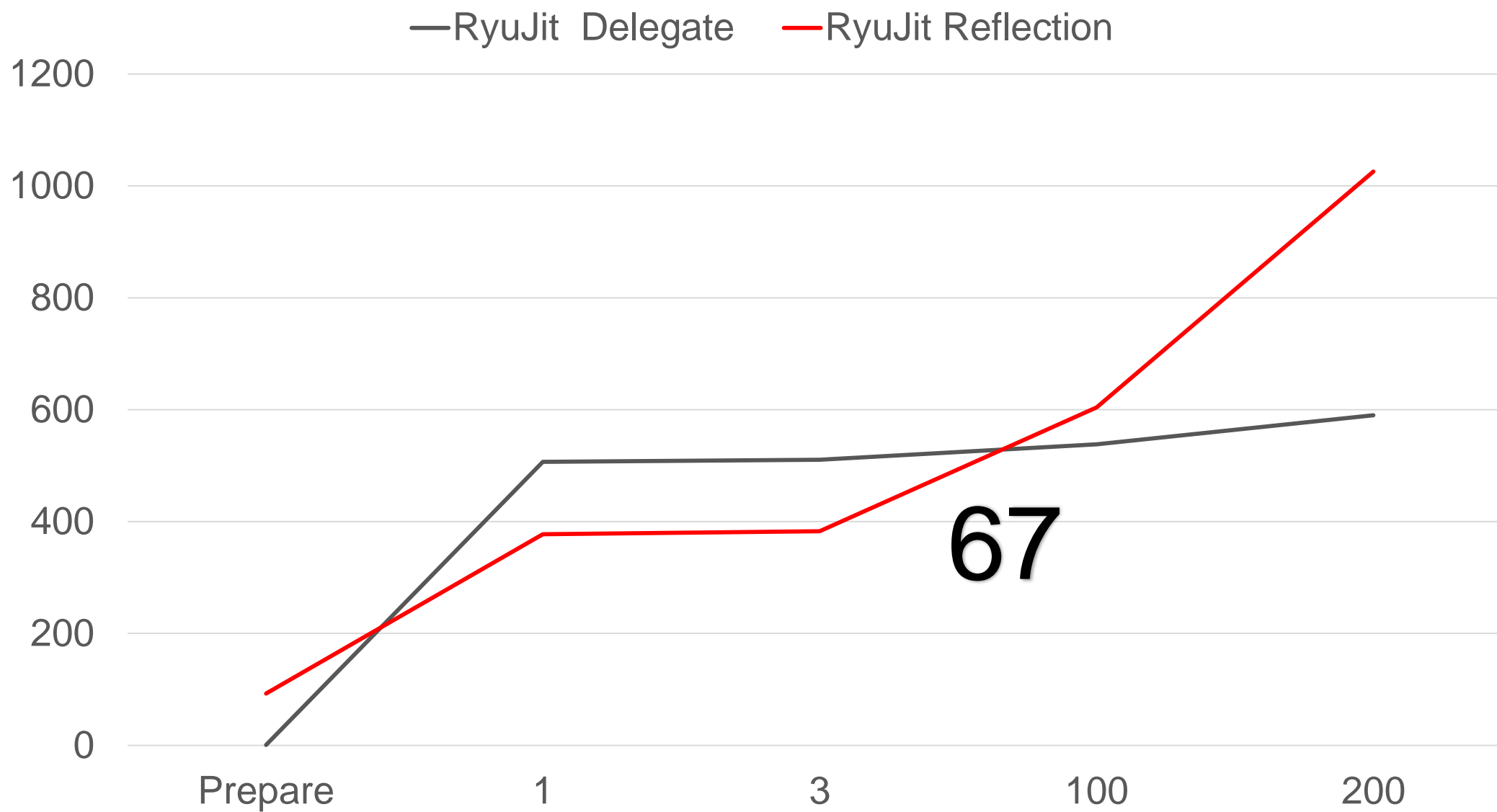
Код

```
public class TestClass8164{public TestClass8164(){}}

public Func<object>[] GetDelegates()
{
    return new Func<object>[]
    {
        CreateDefaultConstructor<TestClass0>(),
        ...
        CreateDefaultConstructor<TestClass10000>()};
}
```

Старт приложения

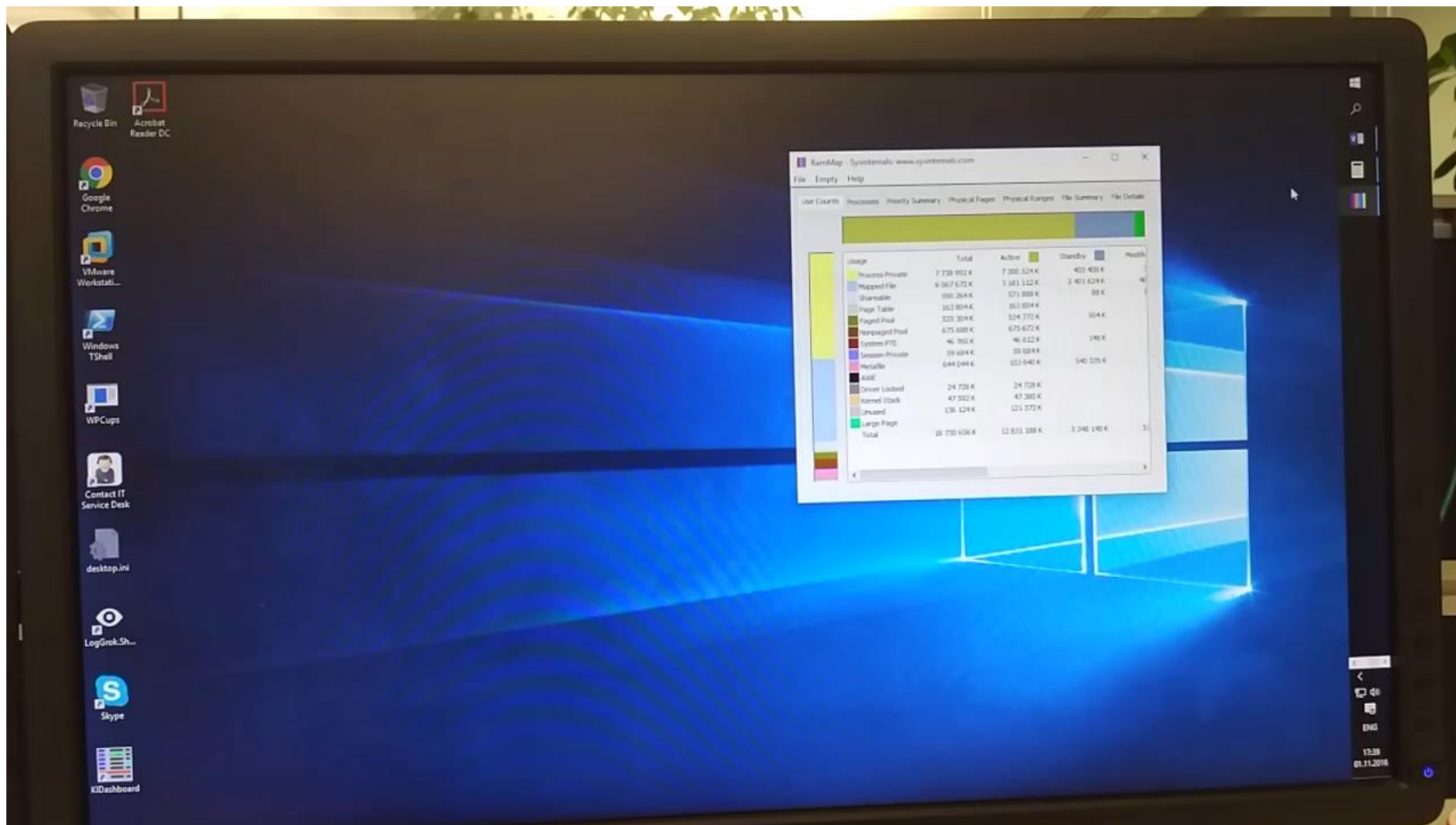




План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

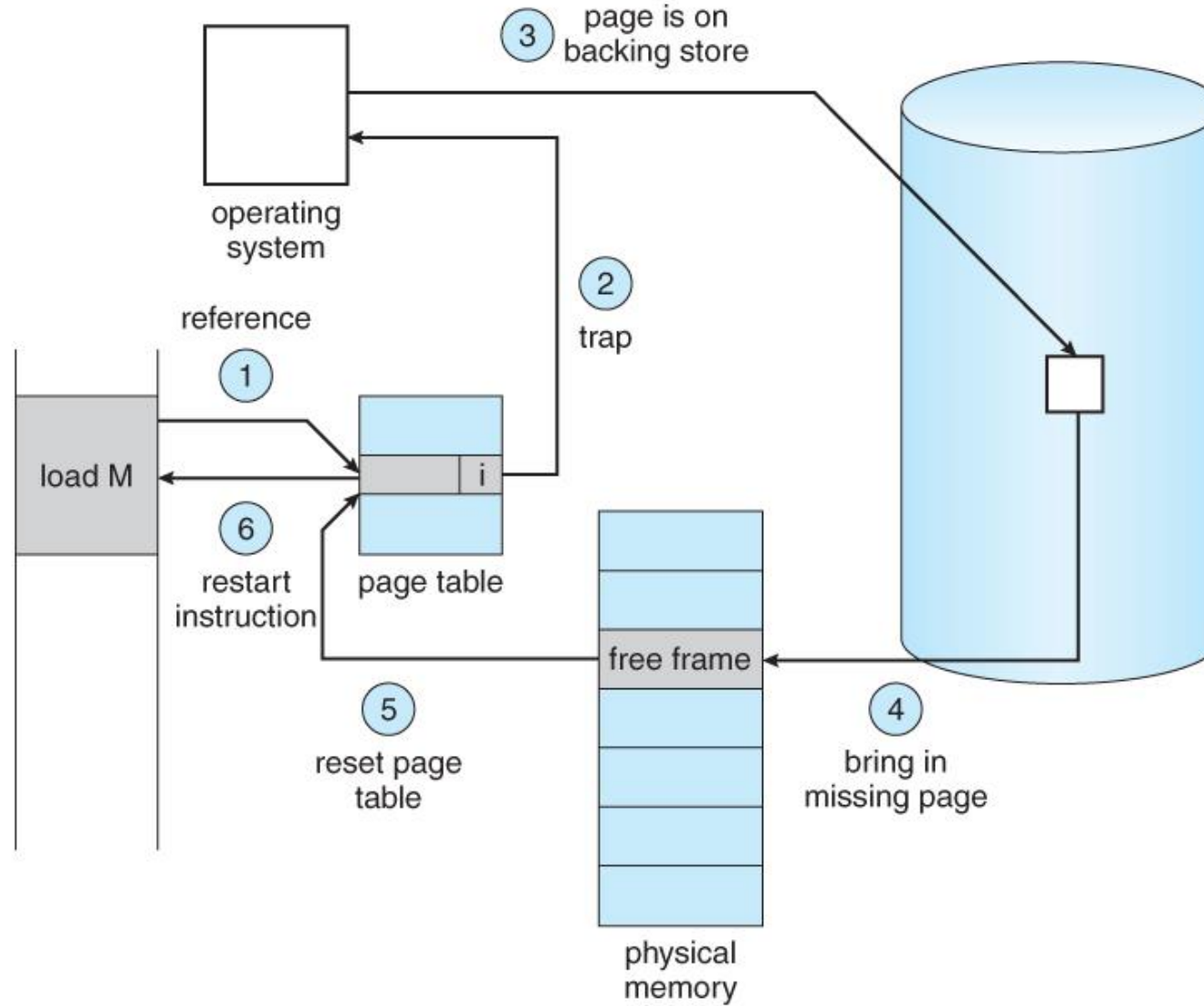
Поднимем калькулятор из SWAP



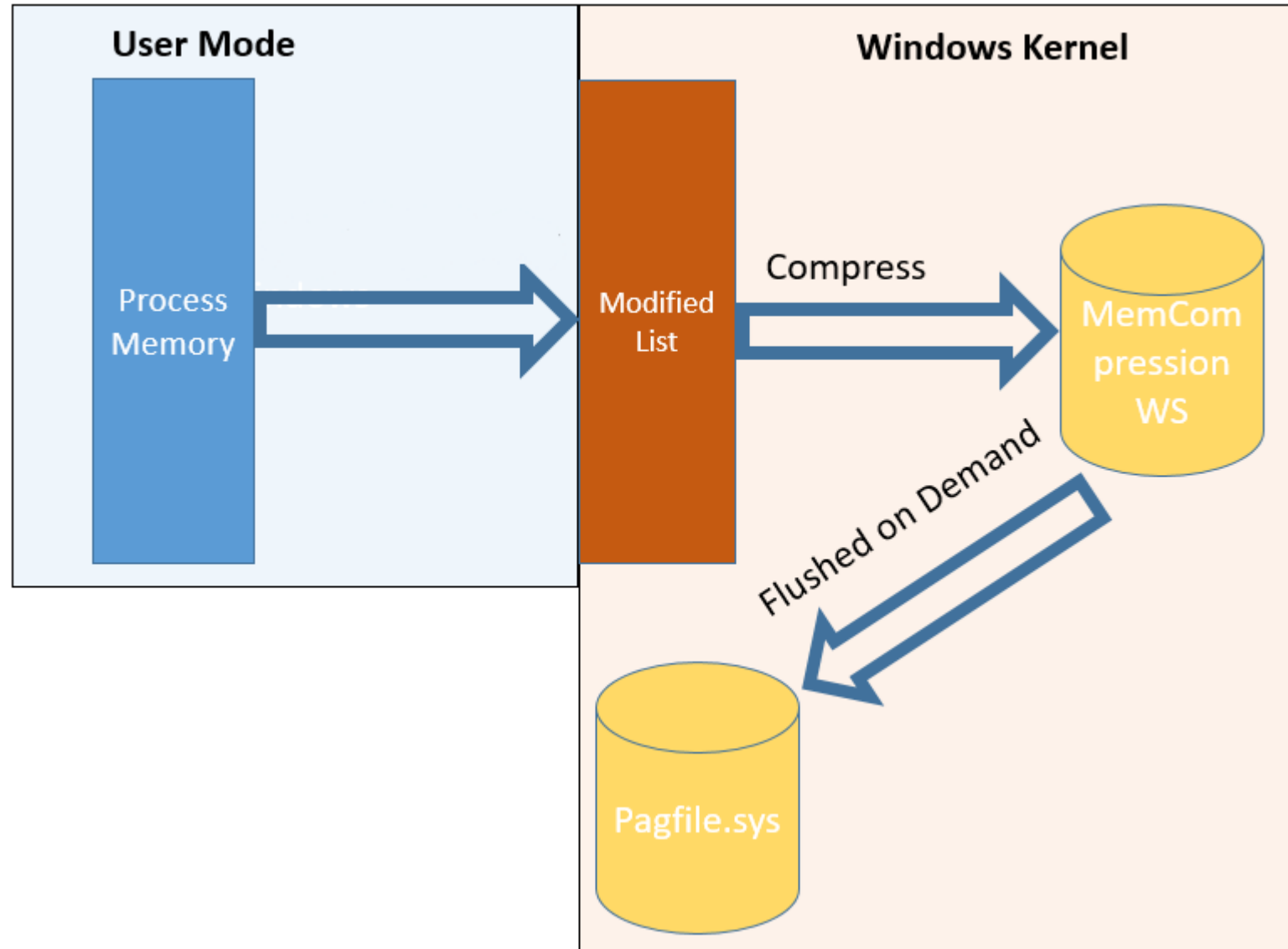
План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

SWAP



Windows



План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - **Prefetch API**
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

Prefetch API

```
BOOL WINAPI PrefetchVirtualMemory(  
    _In_ HANDLE hProcess,  
    _In_ ULONG_PTR NumberOfEntries,  
    _In_reads_(NumberOfEntries) PWIN32_MEMORY_RANGE_ENTRY VirtualAddresses,  
    _In_ ULONG Flags  
);
```

```
typedef struct _WIN32_MEMORY_RANGE_ENTRY {  
    PVOID VirtualAddress;  
    SIZE_T NumberOfBytes;  
} WIN32_MEMORY_RANGE_ENTRY, *PWIN32_MEMORY_RANGE_ENTRY;
```

Minimum supported client	Windows 8 [desktop apps only]
Minimum supported server	Windows Server 2012 [desktop apps only]
Header	WinBase.h (include Windows.h)

План

- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - **Как использовать в реальных проектах**
- Как автоматически ловить залипания GUI

Three Domains in a Managed Process

System Domain (singleton)

High-Frequency Heap
Low-Frequency Heap
Stub Heap
Handle Table
LOH Handle Table
Interface Vtable Map
Assembly Cache
Context
Security Descriptor
Global Interface Vtable Map
Global String Literal Map
Default Domain
System Domain
Global Interface ID Table

Shared Domain (singleton)

High-Frequency Heap
Low-Frequency Heap
Stub Heap
Handle Table
LOH Handle Table
Interface Vtable Map
Assembly Cache
Context
Security Descriptor
Assembly Map
DLSRecords

Default AppDomain

High-Frequency Heap
Low-Frequency Heap
Stub Heap
Handle Table
LOH Handle Table
Interface Vtable Map
Assembly Cache
Context
Security Descriptor
Interface Vtable Map
String Literal Map

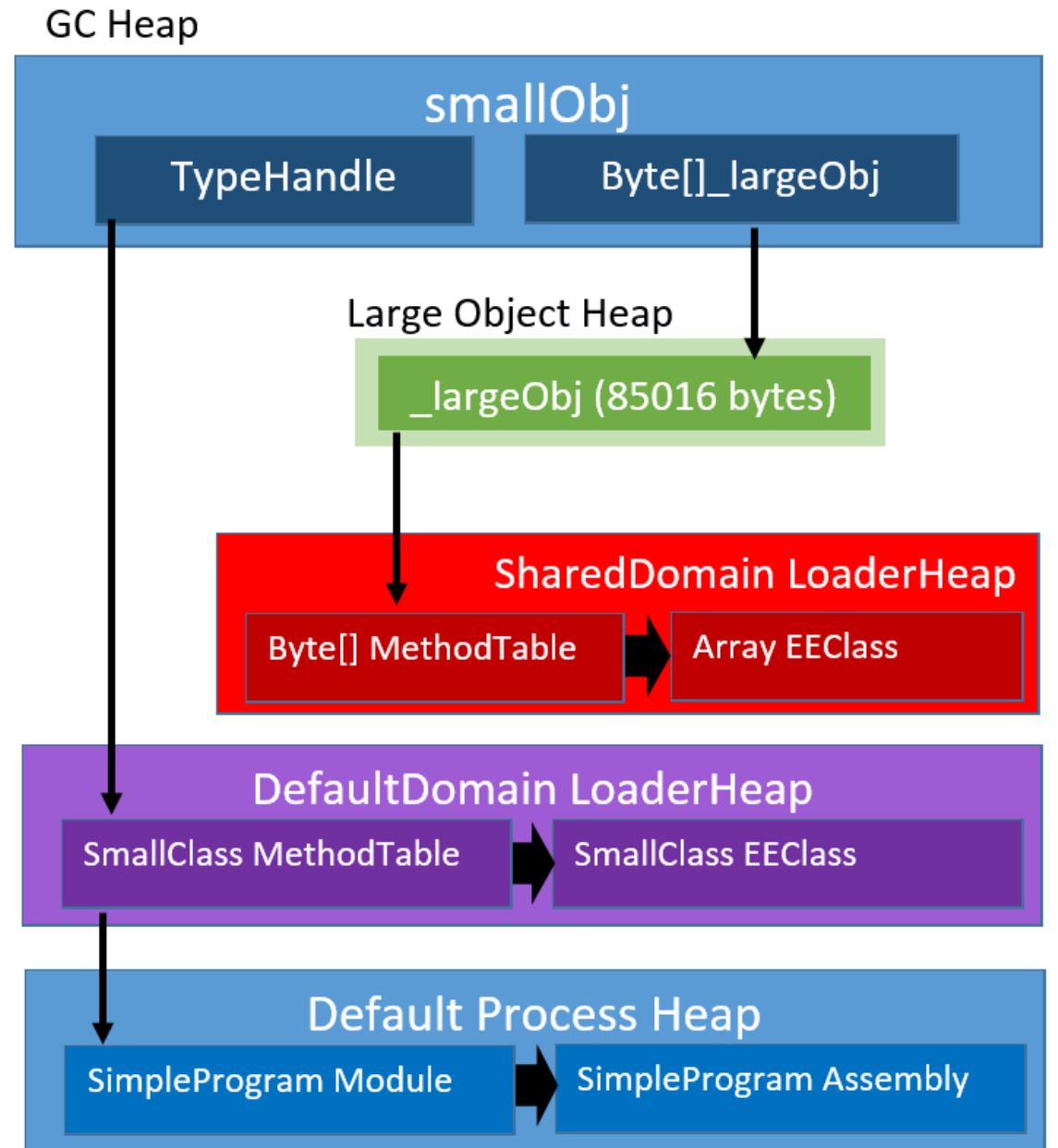
Process Heap	JIT Code Heap	GC Heap	Large Object Heap
--------------	---------------	---------	-------------------

```

public class SmallObjClass
{
    public Byte[] _largeObj;
}

var smallObj = new SmallObjClass();

```



!EEHeap

0:000> !eeheap

Loader Heap:

System Domain: [73eb1200](#)
LowFrequencyHeap: 00e20000(3000:1000) Size:
HighFrequencyHeap: 00e24000(9000:1000) Size:
StubHeap: 00e2d000(3000:1000) Size:
Virtual Call Stub Heap:
 IndcellHeap: 01310000(2000:1000) Size:
 LookupHeap: 01315000(2000:1000) Size:
 ResolveHeap: 0131b000(5000:1000) Size:
 DispatchHeap: 01317000(4000:1000) Size:
 CacheEntryHeap: 01312000(3000:1000) Size:
Total size: Size: 0x8000 (32768) bytes

Jit code heap:

LoaderCodeHeap: 00000000(0:0) Size: 0x0 (0) bytes.
LoaderCodeHeap: 00000000(0:0) Size: 0x0 (0) bytes.
LoaderCodeHeap: 00000000(0:0) Size: 0x0 (0) bytes.
LoaderCodeHeap: 00000000(0:0) Size: 0x0 (0) bytes.
Total size: Size: 0x0 (0) bytes.

Module Thunk heaps:

Module [717d1000](#): Size: 0x0 (0) bytes.
Module [00e33fdc](#): Size: 0x0 (0) bytes.
Module [70b61000](#): Size: 0x0 (0) bytes.
Total size: Size: 0x0 (0) bytes.

Module Lookup Table heaps:

Module [717d1000](#): Size: 0x0 (0) bytes.
Module [00e33fdc](#): Size: 0x0 (0) bytes.
Module [70b61000](#): Size: 0x0 (0) bytes.
Total size: Size: 0x0 (0) bytes.

Total LoaderHeap size: Size: 0x254000 (2441216) bytes

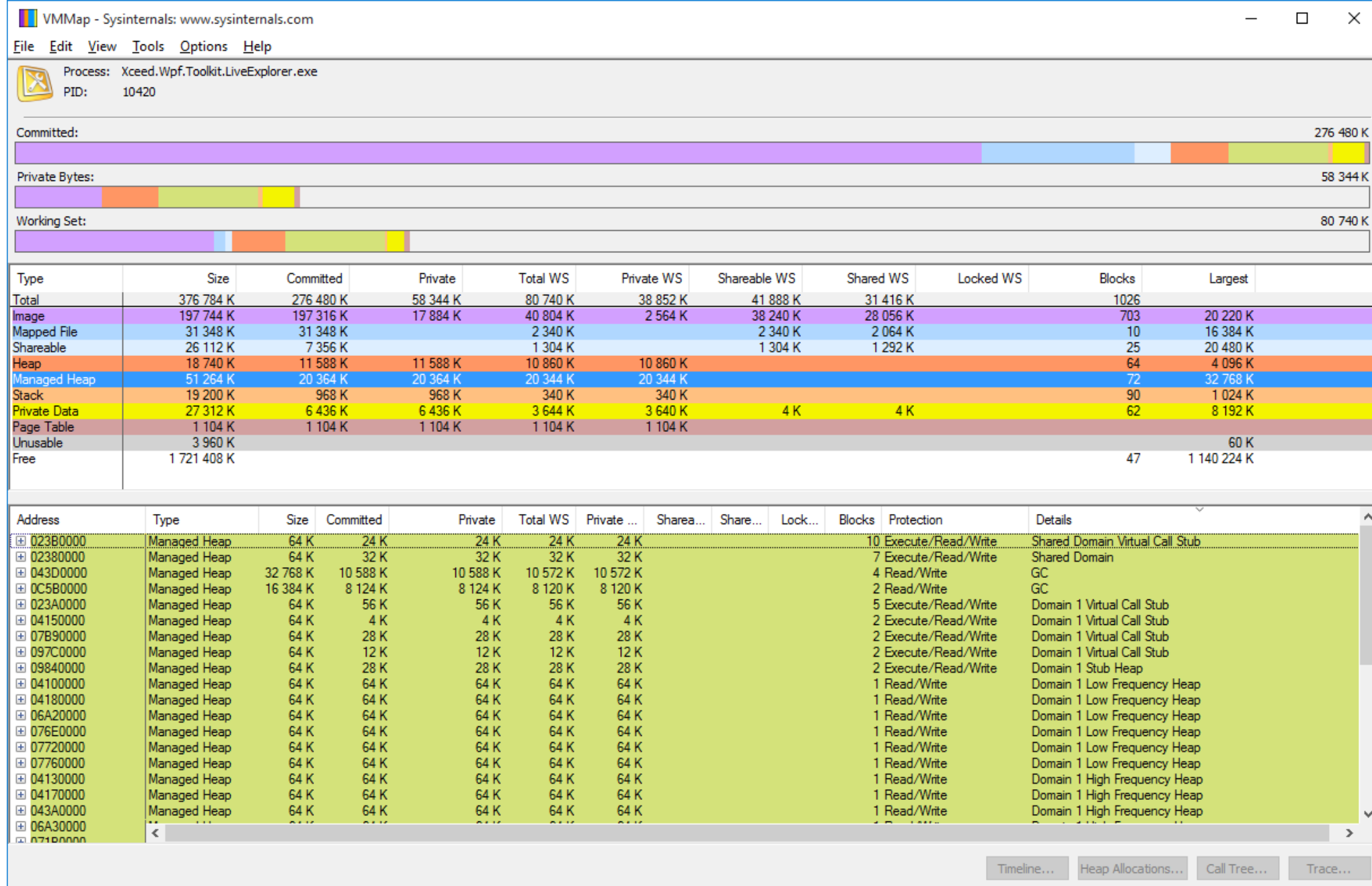
=====

Number of GC Heaps: 1

generation 0 starts at 0x02dd1018

generation 1 starts at 0x02dd100c

generation 2 starts at 0x02dd1000



Сразу после запуска



Process: Xceed.Wpf.Toolkit.LiveEx

PID: 2488

Committed:



Private Bytes:



Working Set:



Type	Size	Committed	Private	Total WS
Total	382 444 K	302 092 K	76 892 K	96 420 K
Image	188 320 K	186 276 K	22 088 K	58 412 K
Mapped File	41 212 K	41 212 K		1 596 K
Shareable	37 804 K	15 292 K		996 K
Heap	17 472 K	12 152 K	12 152 K	11 740 K
Managed Heap	34 496 K	16 992 K	16 992 K	16 984 K
Stack	14 080 K	524 K	524 K	252 K
Private Data	43 720 K	24 304 K	24 304 K	5 608 K
Page Table	832 K	832 K	832 K	832 K
Unusable	4 508 K	4 508 K		
Free	1 715 476 K			

SWAP



Process: Xceed.Wpf.Toolkit.LiveEx

PID: 2488

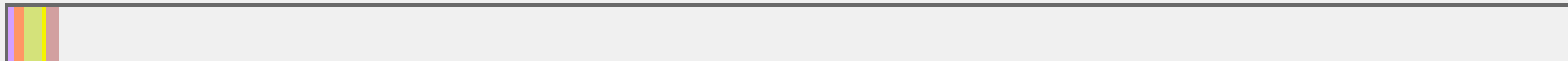
Committed:



Private Bytes:



Working Set:



Type	Size	Committed	Private	Total WS
Total	376 032 K	301 916 K	76 624 K	3 872 K
Image	188 320 K	186 276 K	22 088 K	868 K
Mapped File	41 212 K	41 212 K		
Shareable	37 804 K	15 336 K		
Heap	17 472 K	12 152 K	12 152 K	560 K
Managed Heap	34 496 K	16 992 K	16 992 K	1 368 K
Stack	7 680 K	312 K	312 K	24 K
Private Data	43 648 K	24 236 K	24 236 K	208 K
Page Table	844 K	844 K	844 K	844 K
Unusable	4 556 K	4 556 K		
Free	1 721 900 K			

Open from SWAP



Process: Xceed.Wpf.Toolkit.LiveEx

PID: 2488

Committed:



Private Bytes:



Working Set:



Type	Size	Committed	Private	Total WS
Total	382 440 K	302 648 K	77 308 K	42 440 K
Image	188 320 K	186 276 K	22 088 K	13 800 K
Mapped File	41 212 K	41 212 K		192 K
Shareable	41 680 K	15 444 K		212 K
Heap	17 472 K	12 696 K	12 696 K	10 348 K
Managed Heap	34 496 K	16 996 K	16 996 K	11 604 K
Stack	10 240 K	400 K	400 K	184 K
Private Data	43 672 K	24 276 K	24 276 K	5 248 K
Page Table	852 K	852 K	852 K	852 K
Unusable	4 496 K	4 496 K		
Free	1 715 500 K			

Modules

Managed modules

```
const int peHeaderSize = 0x2000;  
var currentAppDomainModulesHandles = AppDomain.CurrentDomain.GetAssemblies()  
    .SelectMany(o => o.GetLoadedModules(false))  
    .Select(m => new IntPtr(Marshal.GetHINSTANCE(m).ToInt32() + eHeaderSize));
```

Native modules

```
Process.GetCurrentProcess().Modules.OfType<ProcessModule>()  
    .Select(module => new {  
        Address = module.BaseAddress,  
        SizeInBytes = module.ModuleMemorySize  
    });
```

Code and Data

Low-Frequency Heap & Jitted code

```
IntPtr metaHandle = method.MethodHandle.Value;  
IntPtr implHandle = method.MethodHandle.GetFunctionPointer();
```

GC heap objects

```
IntPtr handle = GCHandle.Alloc(managedObject, GCHandleType.Pinned)  
                .AddrOfPinnedObject();
```

Как можно это использовать в реальности

```
SIZE_T WINAPI VirtualQuery(  
    __in_opt LPCVOID lpAddress,  
    __out_bcount_part(dwLength, return) PMEMORY_BASIC_INFORMATION lpBuffer,  
    __in SIZE_T dwLength  
);
```

```
typedef struct _MEMORY_BASIC_INFORMATION {  
    PVOID BaseAddress;  
    PVOID AllocationBase;  
    DWORD AllocationProtect;  
    SIZE_T RegionSize;  
    DWORD State; //MEM_COMMIT, MEM_FREE, MEM_RESERVE  
  
    DWORD Protect;  
    DWORD Type;  
} MEMORY_BASIC_INFORMATION, *PMEMORY_BASIC_INFORMATION;
```

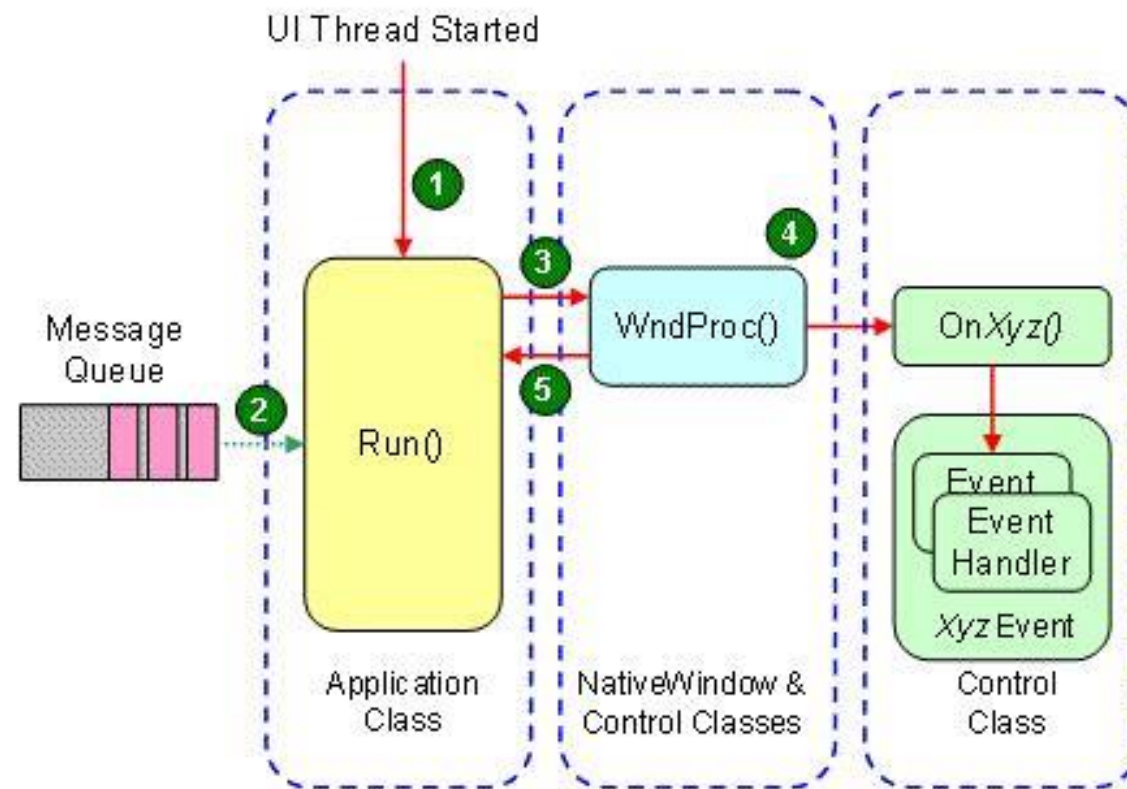
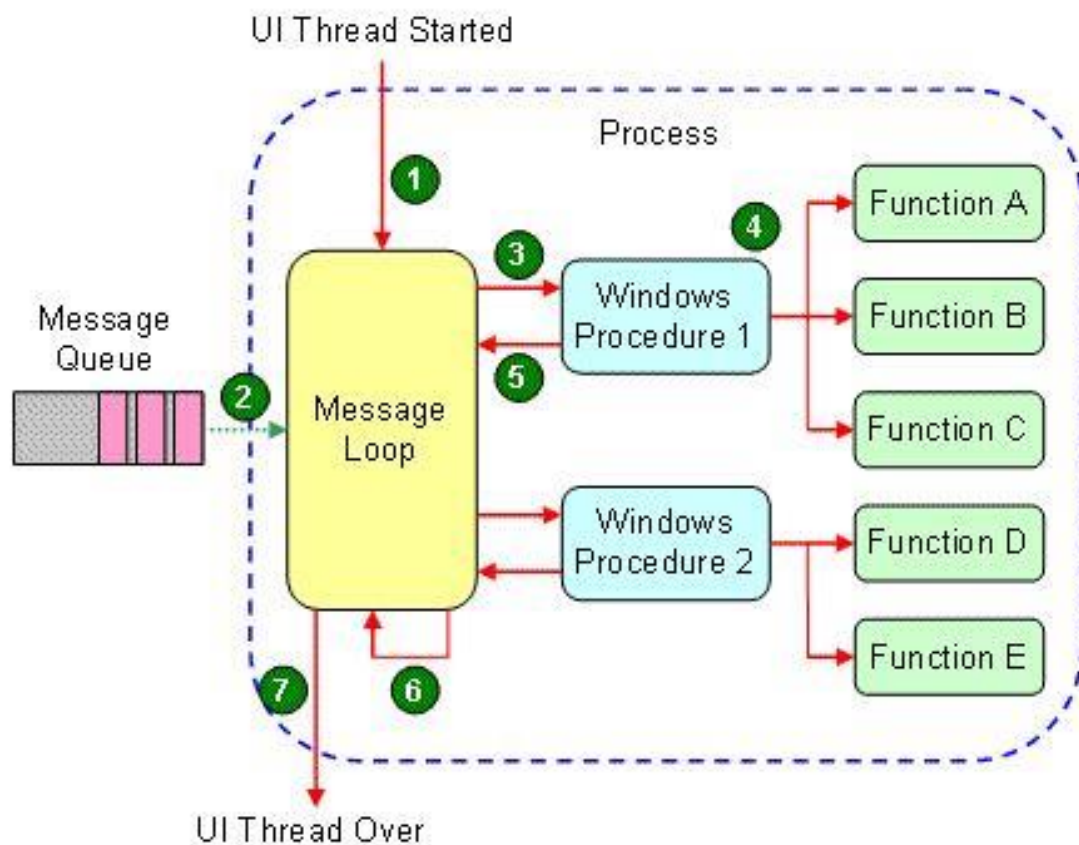

Результаты

	Default ms	With prefetch ms	Difference ms
Open after SWAP	5500	3600	1900
Window1	1667	1607	60
Window2	800	367	433
Window3	1033	500	533
Window4	334	167	167
Window5	700	267	433
Window6	834	567	267
Window7	433	267	166
Window8	634	333	301
Window9	100	33	67
Window10	534	434	100
Window11	666	367	299
Window12	867	400	467
Window13	566	433	133
Window14	533	234	299

План

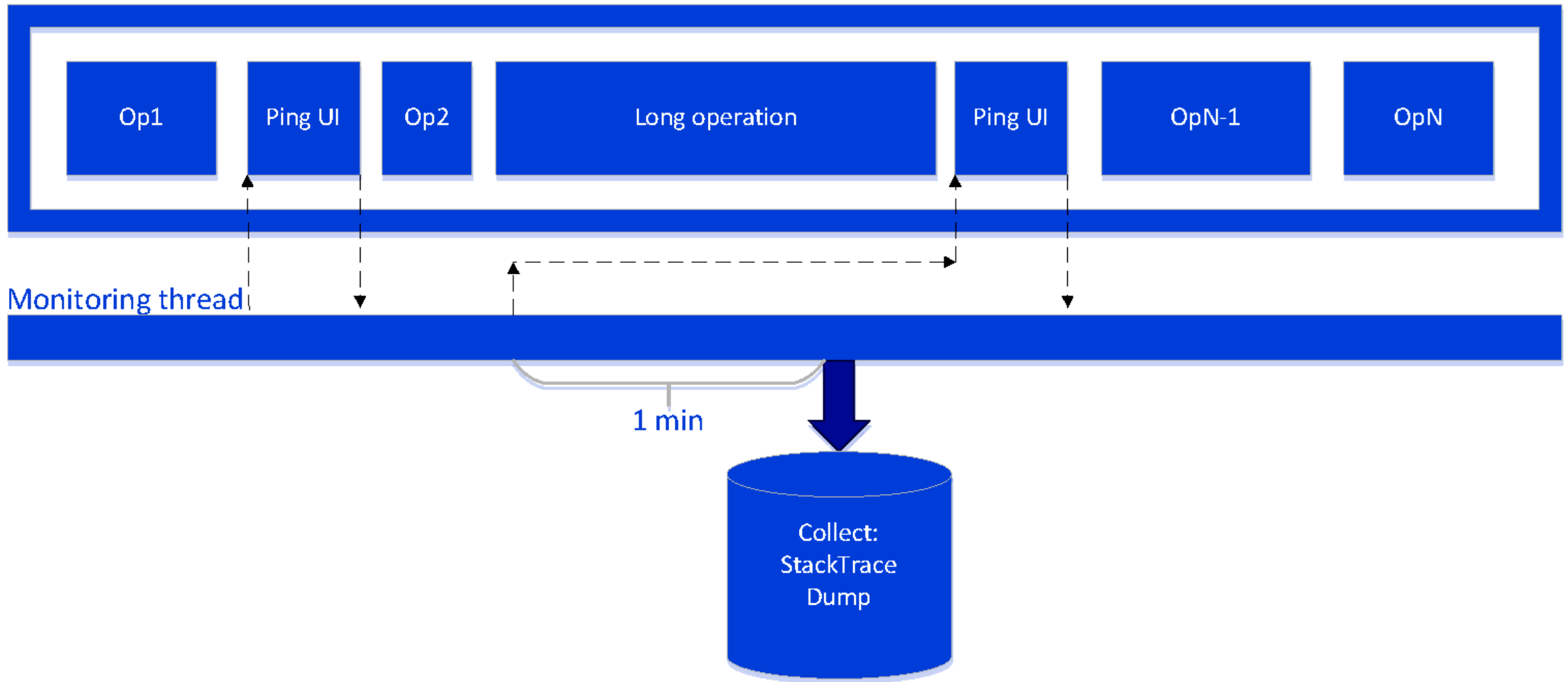
- Под какие требования оптимизируем
- Используемый инструментарий для анализа performance
- Оптимизация запуска .NET-приложения
 - Assemblies Warmup
 - Jit
 - IoC (DI) – контейнер
- Ускорение поднятия из SWAP
 - Поднимем калькулятор из SWAP
 - Как работает SWAP
 - Prefetch API
 - Как использовать в реальных проектах
- Как автоматически ловить залипания GUI

Как автоматически ловить залипания GUI

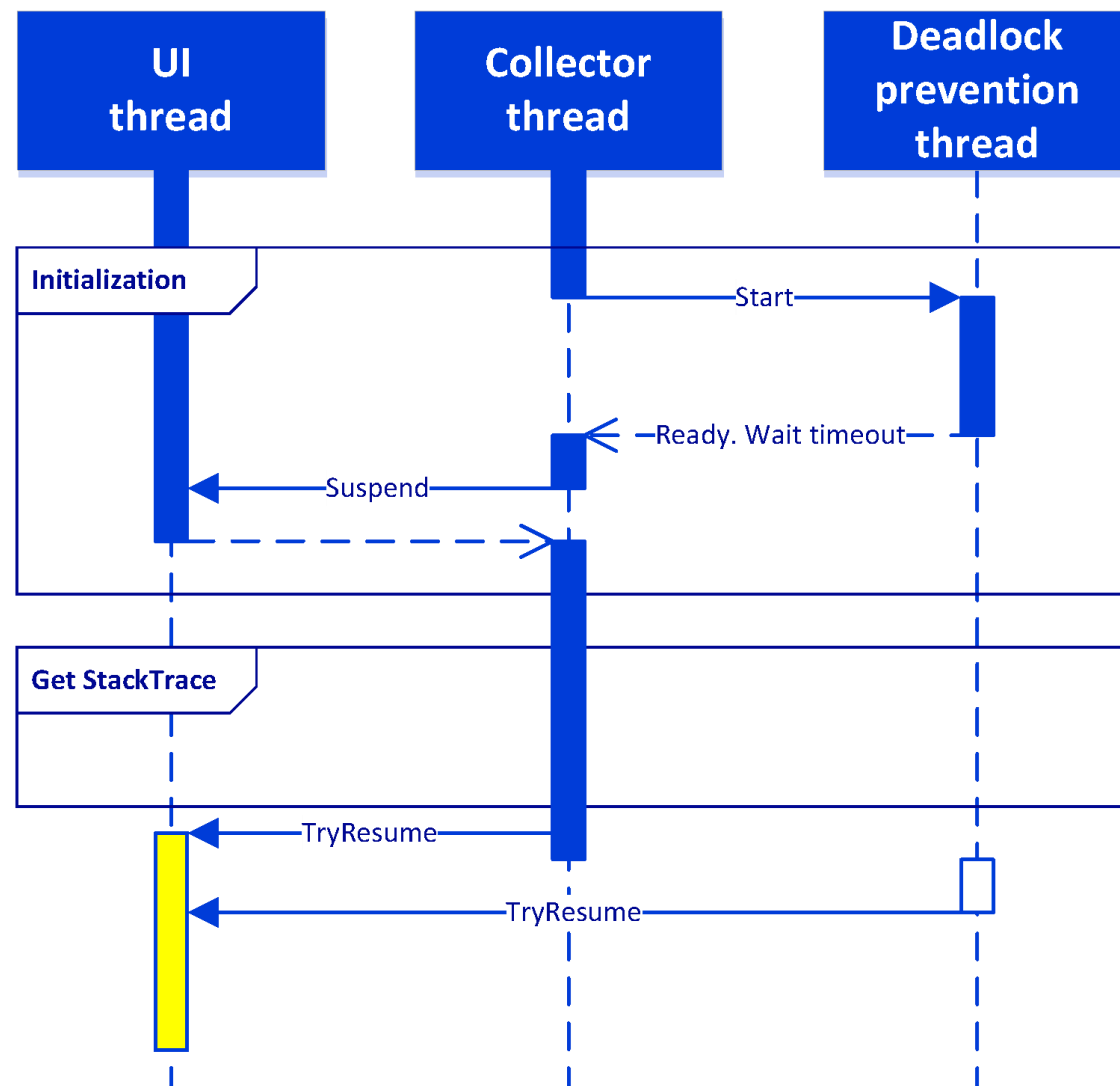


Принцип работы FreezeDetector

UI thread



Получение StackTrace UI потока



```
new StackTrace(Application.Current.Dispatcher.Thread, false);
```

Links

- <http://www.uadnan.com/dot-net-framework/diving-into-clr-runtime/>
- https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html
- <https://aloiskraus.wordpress.com/>
- <http://blog.csdn.net/phiger/article/details/5566988>
- <https://xavierantony.wordpress.com/2010/08/24/windows-messaging-architecture/>

LET'S TALK?

Сергей Сенцов
zikyrat@gmail.com

Kaspersky Lab HQ
39A/3 Leningradskoe Shosse
Moscow, 125212, Russian Federation
Tel: +7 (495) 797-8700
www.kaspersky.com

KASPERSKY 

Join our team:

<https://hh.ru/vacancy/16771928>

<https://hh.ru/vacancy/17641618>

kaspersky.ru/job

KASPERSKY 