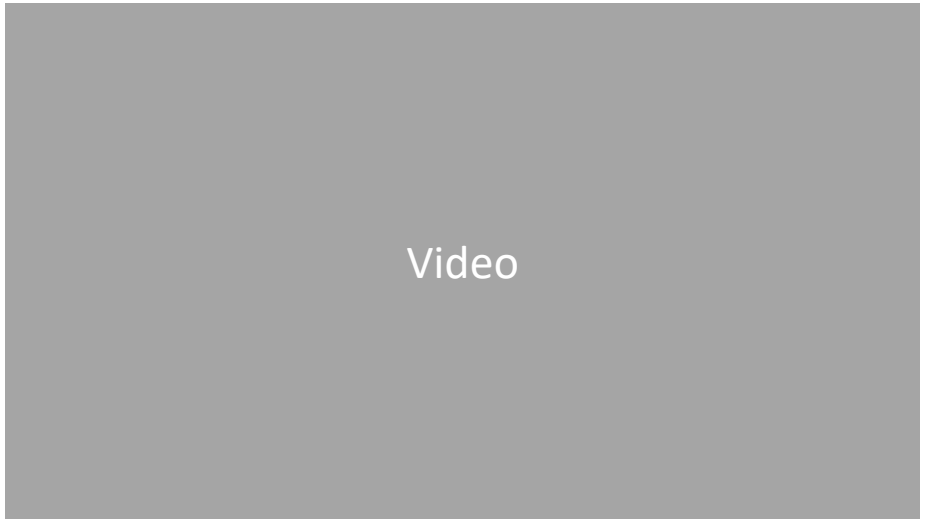
GOPAS

Novinky v .NET Internals

Robert Haken | TechEd 2021



TechEd

GOPAS

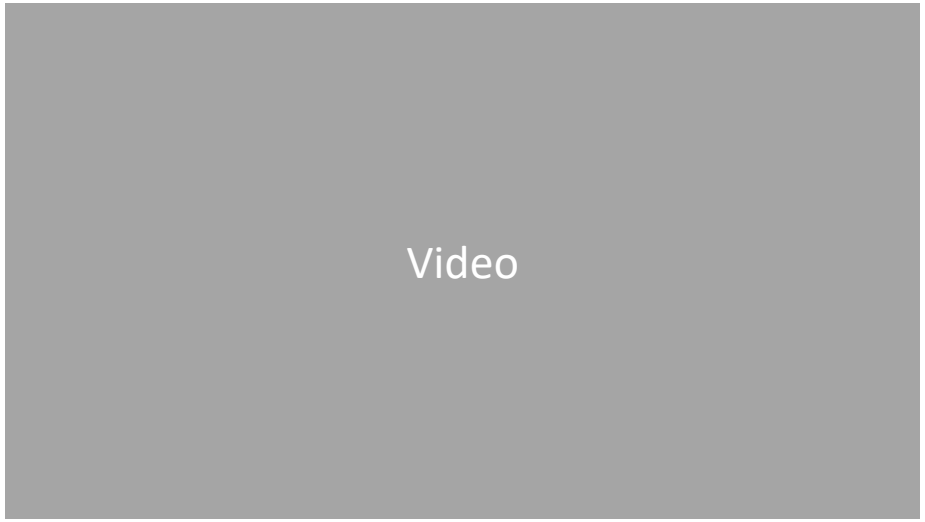
Agenda

JIT, compilation

Diagnostic CLI tools

Memory Management

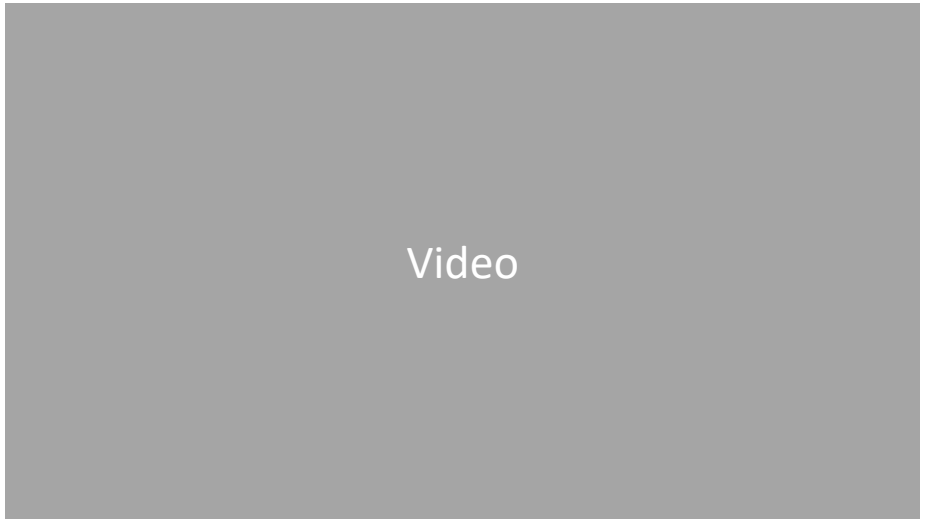
Perf



TechEd

GOPAS

JIT, compilation, publishing



TechEd

JIT: Tiered Compilation

Tier 0 – *Quick JIT / ReadyToRun*

Tier 1 – *optimizing JIT*

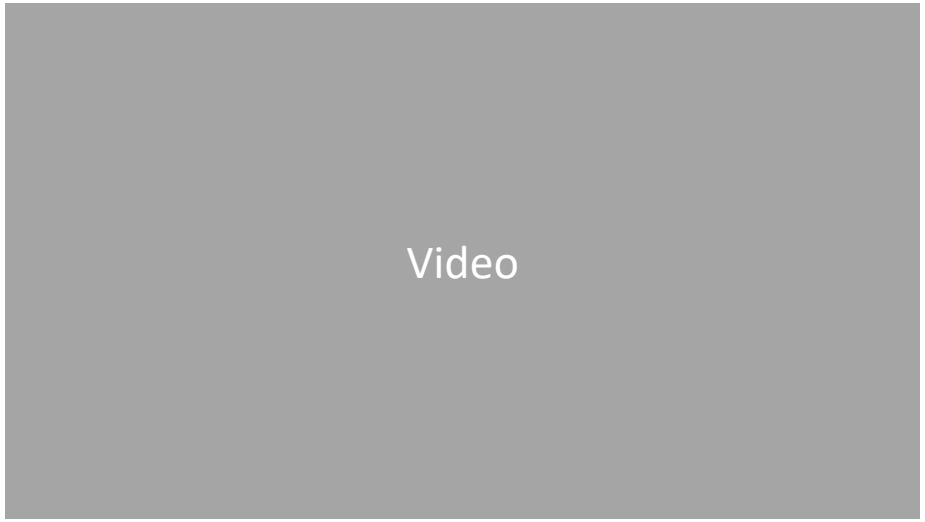
promotion

- 30+ calls, 100ms „startup timer“

.NET Core 3.0+ enabled by default

On Stack Replacement (in-place)

- while method has active stack frames!



ReadyToRun

ahead-of-time (AOT) compilation

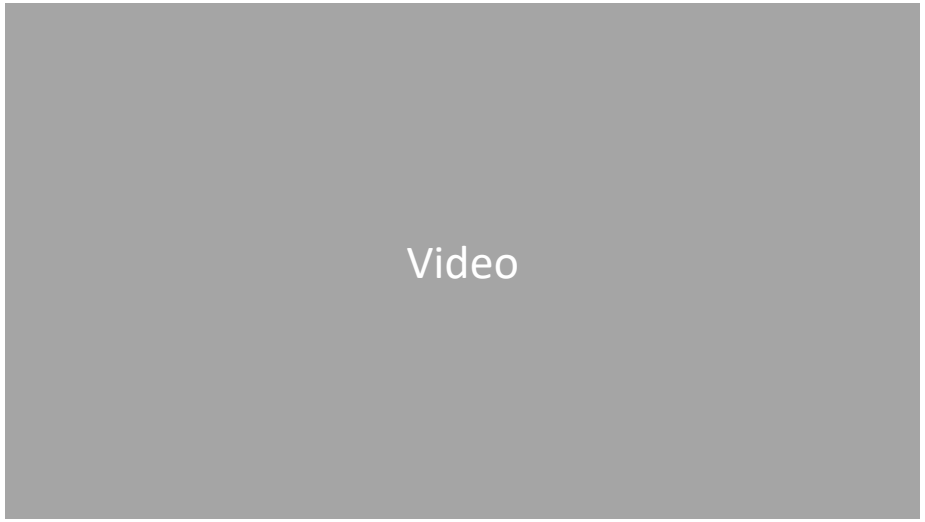
both IL + native version of the same code (bigger, slower to load)

part of publish

```
<PropertyGroup>
        <PublishReadyToRun>true</PublishReadyToRun>
</PropertyGroup>
```

enabled for .NET runtime libraries

still replaced by tiered compilation



JIT: Escape Analysis

instance used only in local variable, does not escape method

=> allocate on stack instead of heap

```
public class Calculator
{
    public int X;
    public int Y;
    public int Add()
    {
        return x + y;
    }
}

public class EscapeAnalysis
{
    private int _x;
    private int _y;
    public int UseCalculator()
    {
        Calculator calc = new Calculator();
        calc.X = _x;
        calc.Y = _y;
        return calc.Add();
    }
}
```

Video

GOPAS

App Trimming

.NET3 – Assembly-level trimming

```
<TrimMode>CopyUsed</TrimMode>
```

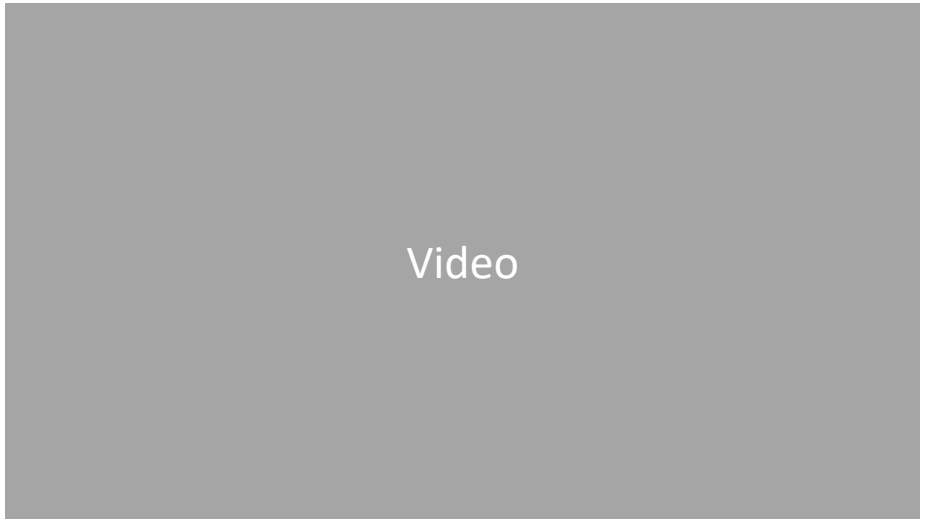
.NET5 – Member-level trimming

```
<TrimMode>Link</TrimMode>
```

removes ReadyToRun data

Blazor WASM defaults

- fw/runtime member-level trimming
- Microsoft.Extensions.* assemblies type-level
- Microsoft.AspNetCore.* hand-generated XML
- other assemblies are not trimmed



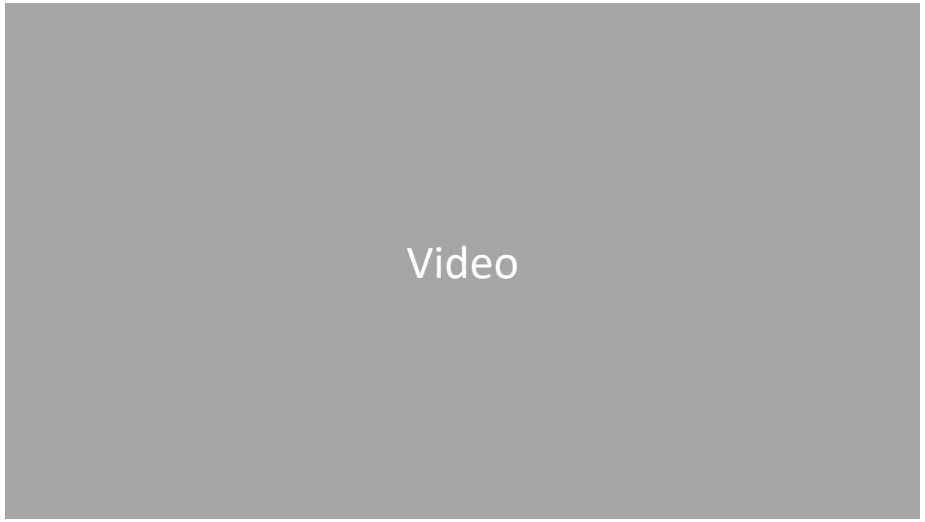
GOPAS

Other improvements

Profile Guided Optimization (PGO)

Loop alignment

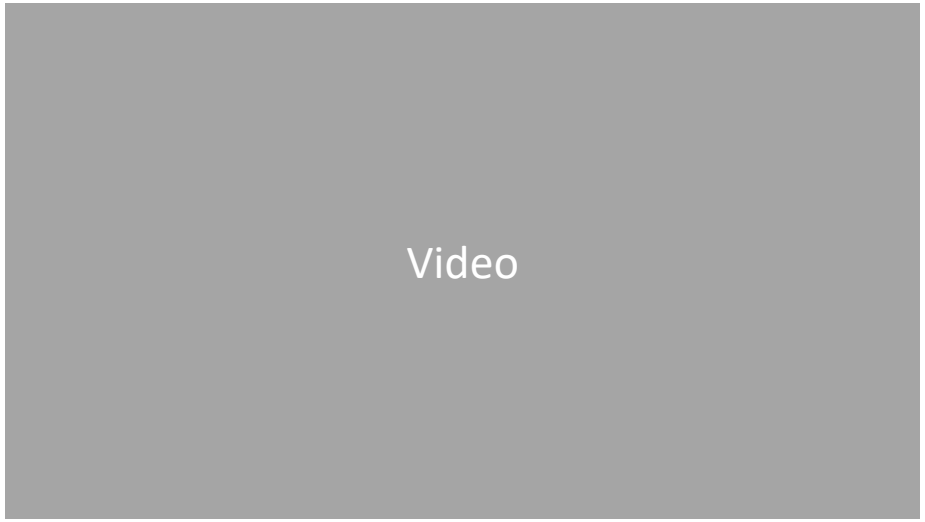
- aligning methods to 32B boundary [NET5]
- adaptive loop alignment with NOPs [NET6]



TechEd

GOPAS

Diagnostic CLI tools

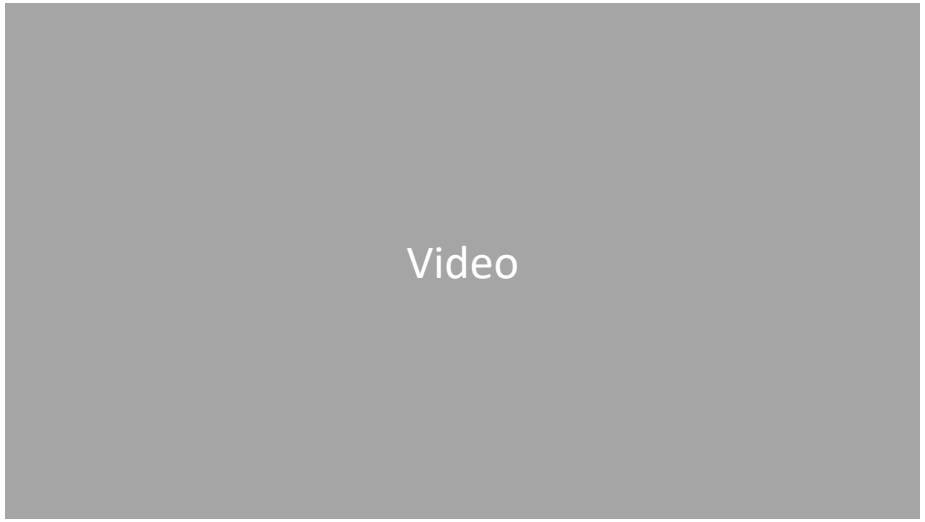


TechEd

dotnet-dump

SOS.dll command-line incl. Linux

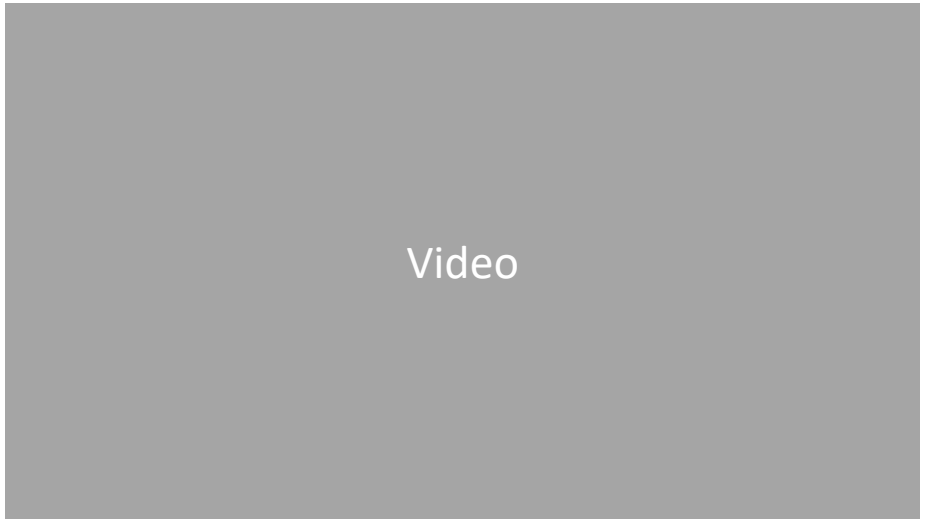
```
dotnet tool install -g dotnet-dump
dotnet-dump collect –p <pid>
dotnet-dump analyze <dumpfile>

setsymbolserver –directory <path>
                –ms
```



GOPAS

demo

dotnet-dump



TechEd

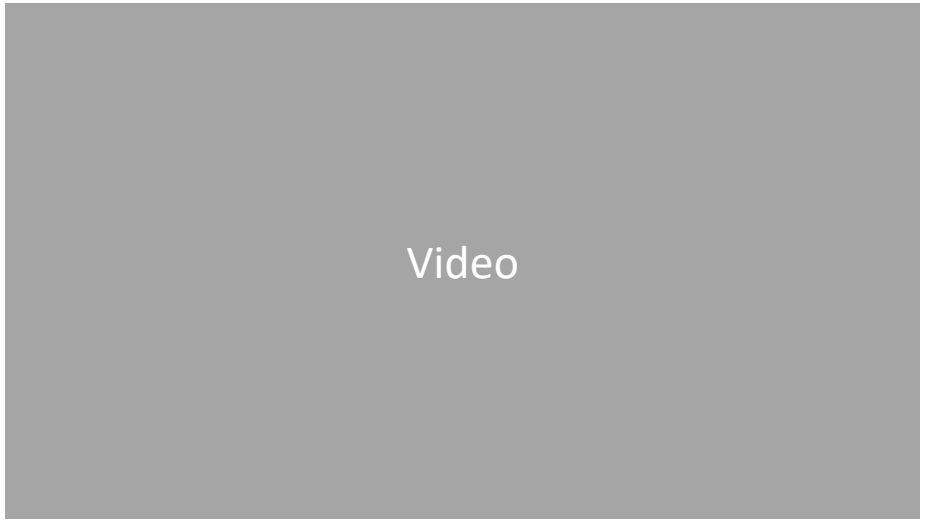
GOPAS

dotnet-dump [vNext]

packing (binaries, symbols)

triggers (exception, perf metrics)

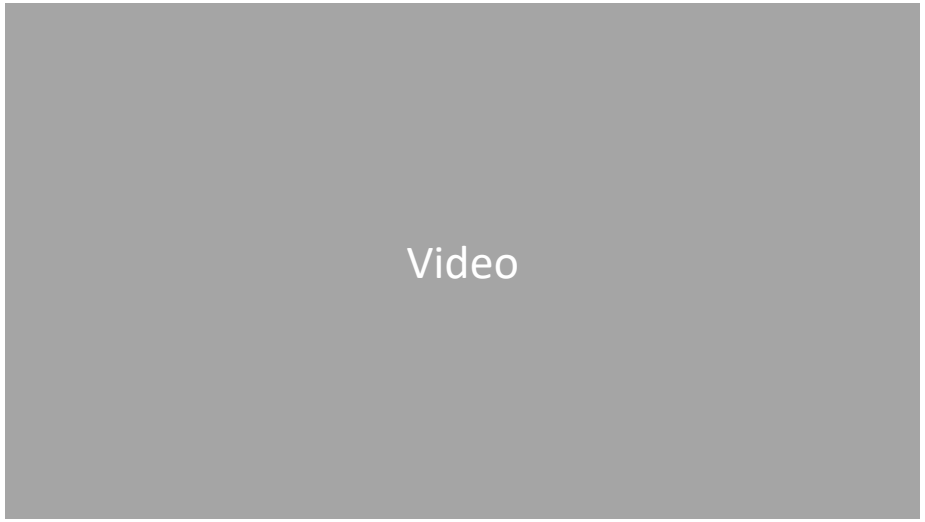
extensible commands (ála sosex, NetExt, Psscor4, ...)



dotnet-stack

„.NET Stack Printing Tool“

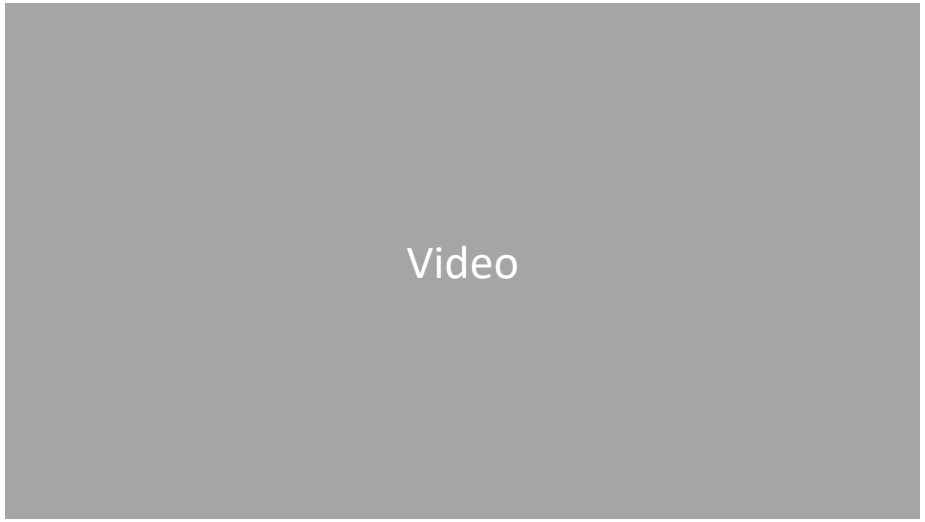
```
dotnet tool install -g dotnet-stack
dotnet-stack ps
dotnet-stack report -p <id>
```



GOPAS

demo

dotnet-stack



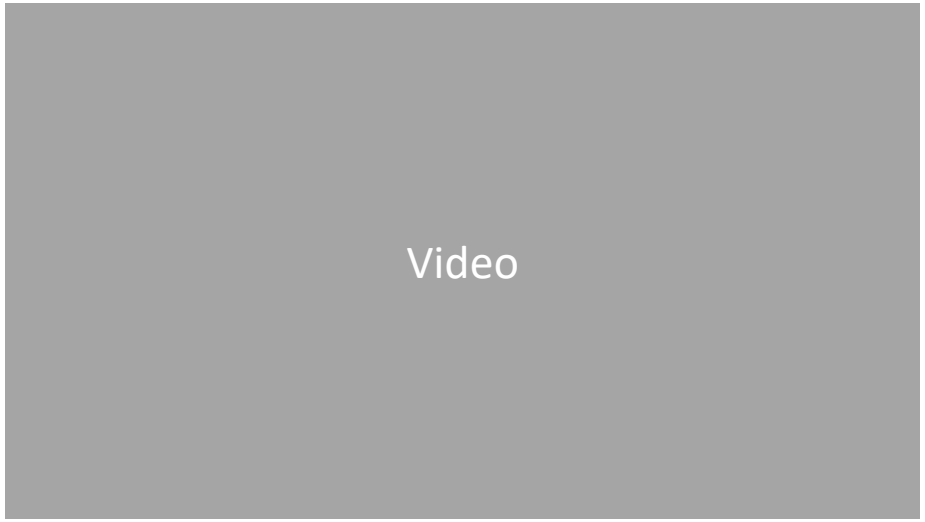
TechEd

GOPAS

dotnet-counters

based on `EventCounter` API (x-plat)

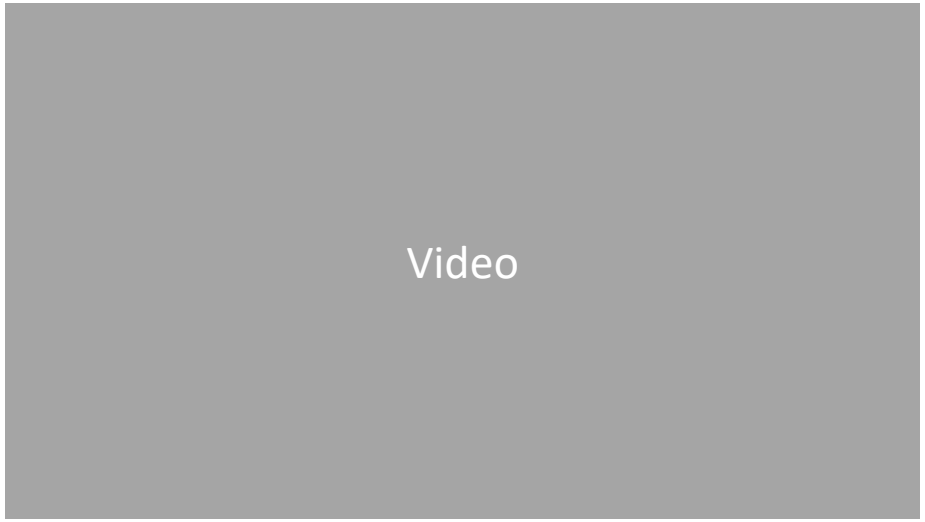
```
dotnet tool install –g dotnet-counters
dotnet-counters list
dotnet-counters monitor –p <id> [Microsoft.AspNetCore.Hosting]
```



GOPAS

demo

dotnet-counters



TechEd

GOPAS

`dotnet-counters` [vNext]

profiles (named set of counters accross multiple providers)

`list -p <pid>` (process-specific list)

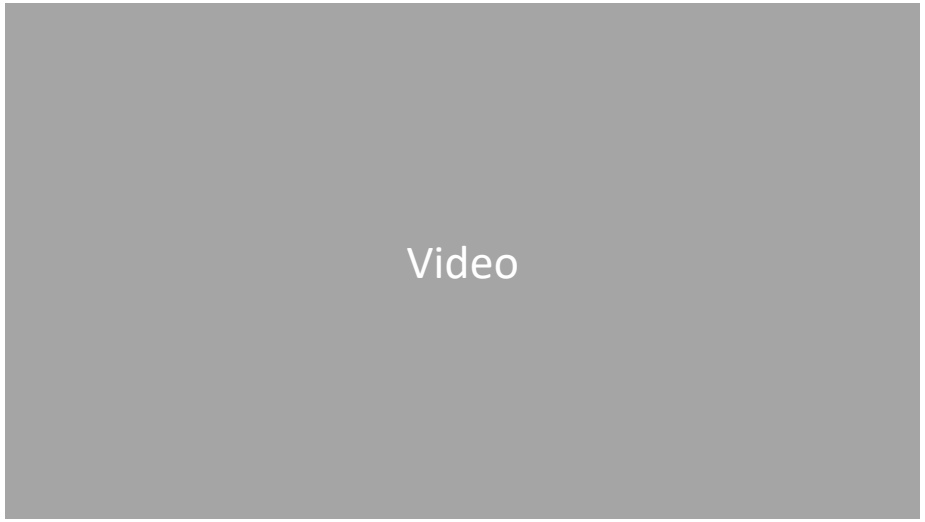
`view` (one-off dump of values, no wait for next counter)

Video

TechEd

GOPAS

Garbage Collector



TechEd

GOPAS

GC: Memory-constrained scenario (containers)

NET Core 3.0+

limits taken from Linux CGroups / Windows Job objects

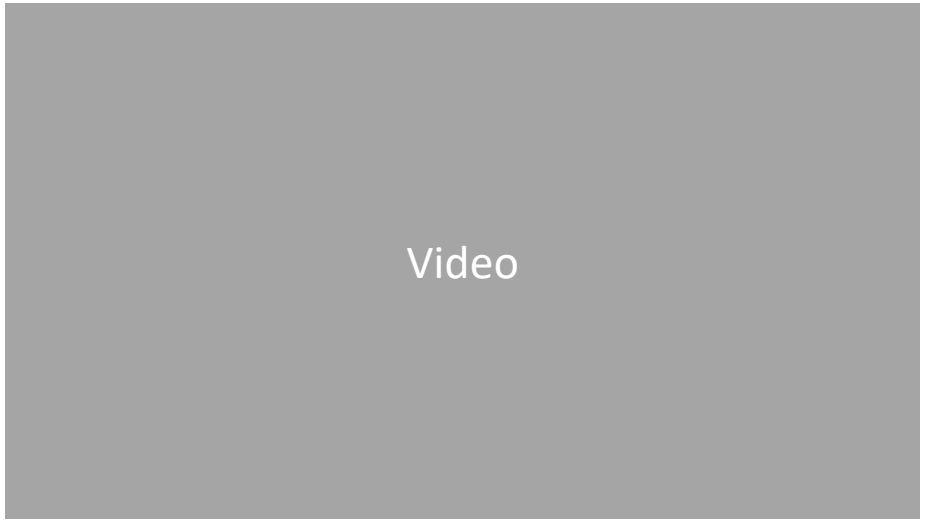
reduced minimal generation 0 GC alloc. budget (% last-level CPU cache)

default heap size in container = max(20 MB, 75% of memory limit)

minimal heap segment size 16 MB

- heap count = min(cpu_count, mem_limit /16 MB)
- Workstation GC no longer needed

LOH compaction if needed

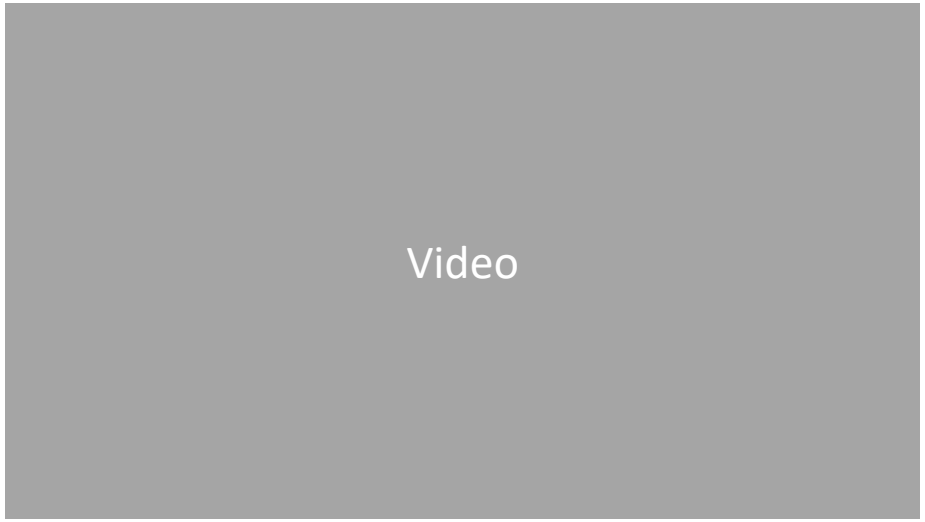


GOPAS

GC: Memory-constrained scenario (containers)

NET Core 3.0 configs

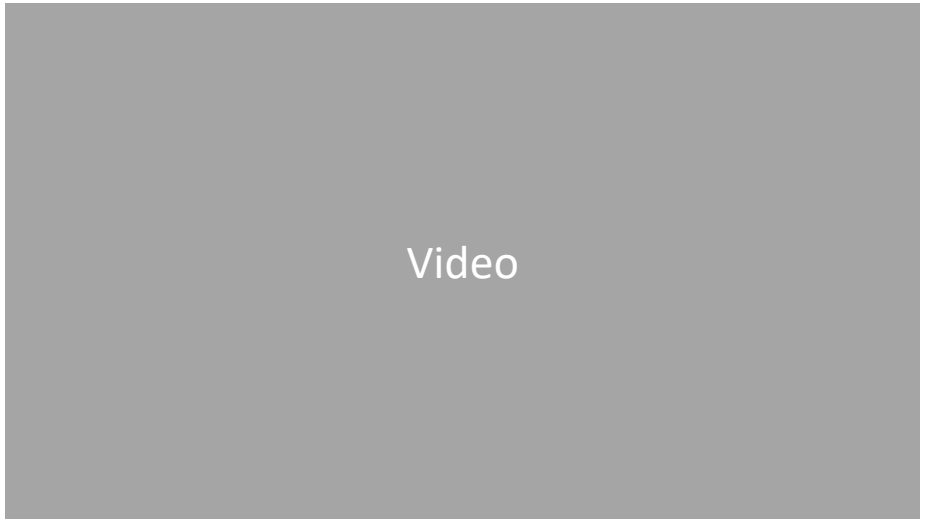
- `GCHeapHardLimit` – limit for GC heap (commit)
- `GCHeapHardLimitPercent`
- `GCHighMemPercent` – default is 90%



GOPAS

demo

ContainerResourceLimits



TechEd

GOPAS

Docker CPU Limits

prior: `cpu_count = Math.Round(container_cpu_limit)`

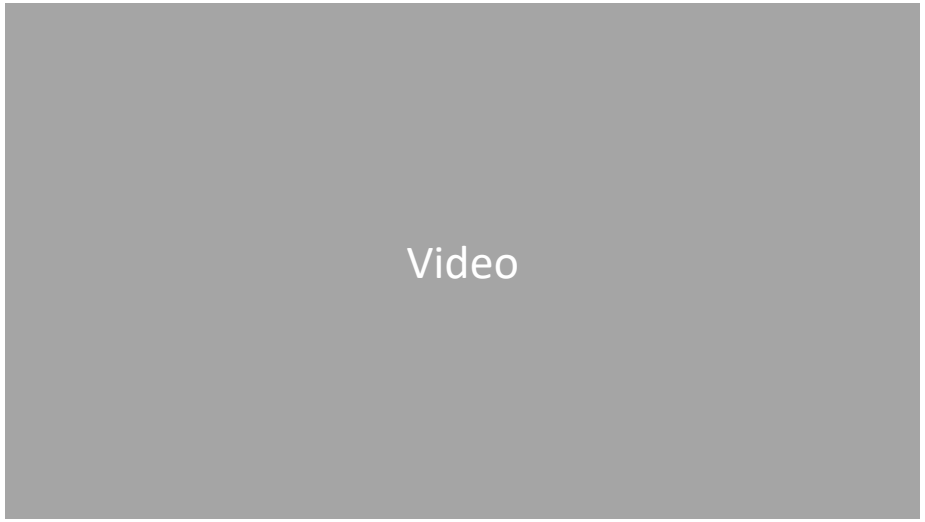
- CPU underutilization

now: `cpu_count = Math.Ceiling(container_cpu_limit)`

- no performance degradation even for 1.000001 limit (rounded to 2)

for single-core environment, Workstation GC is ALLWAYS used

- no matter you explicitly specify Server GC



TechEd

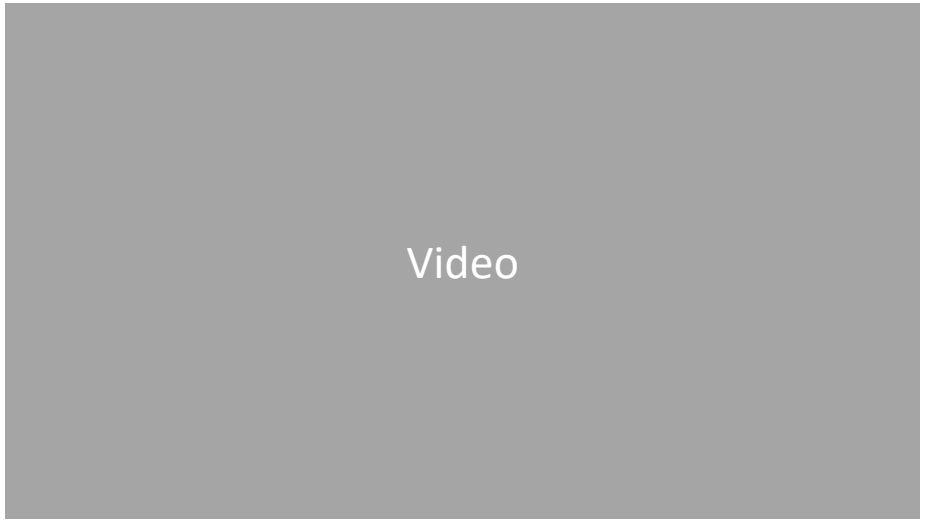
GCLOHThreshold [NET30]

default = 85 000 bytes

new config

- `GCLOHThreshold` configuration knob
- `COMPlus_GCLOHThreshold` environment variable (HEX!)

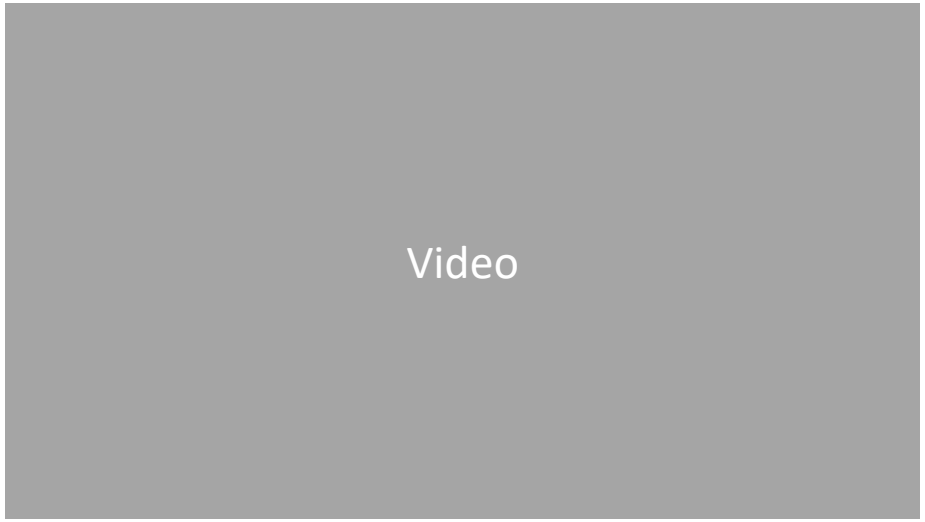
lower values ignored



GOPAS

demo

LOH Threshold



TechEd

GC: Pinned Object Heap [NET5]

Scenario: Pin object when you allocate them

- `GC.AllocateArray(int length, bool pinned = false)`
- `GC.AllocateUninitializedArray()` – no zeroing
- blittable types only (no references)

own segment

swept in gen2 GCs

Video

GOPAS

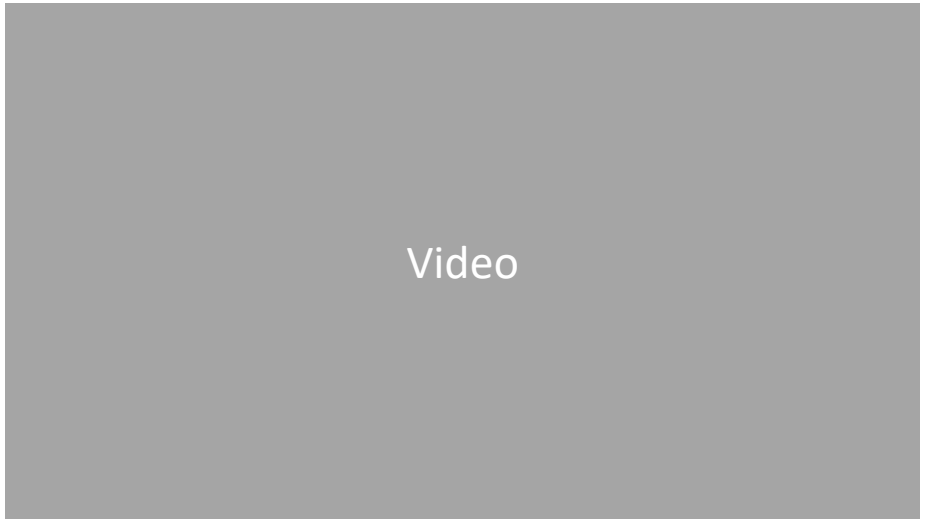
GC: Pinned Object Heap [NET5]

UOH – User Old Heap

- LOH – Large Object Heap (phys. gen3)
- POH – Pinned Object Heap (phys. gen4)

StringBuilder, Convert, Sockets, static, ...

```
COMPlus_GCHeapHardLimitPOH[Percent]
COMPlus_GCHeapHardLimitLOH[Percent]
COMPlus_GCHeapHardLimitSOH[Percent]
```

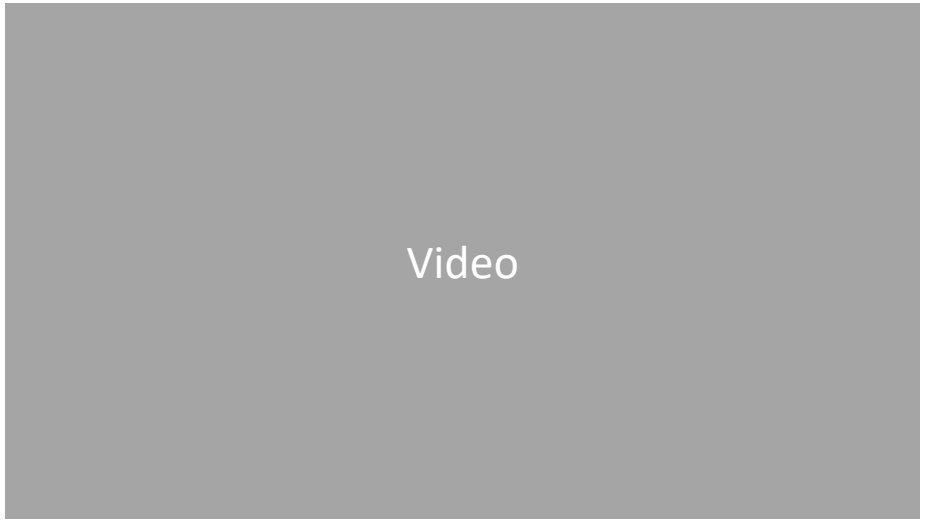


TechEd

GOPAS

demo

Pinned Object Heap



TechEd

GOPAS

GC.GetGCMemoryInfo() [NET3, NET5]

metrics from last GC

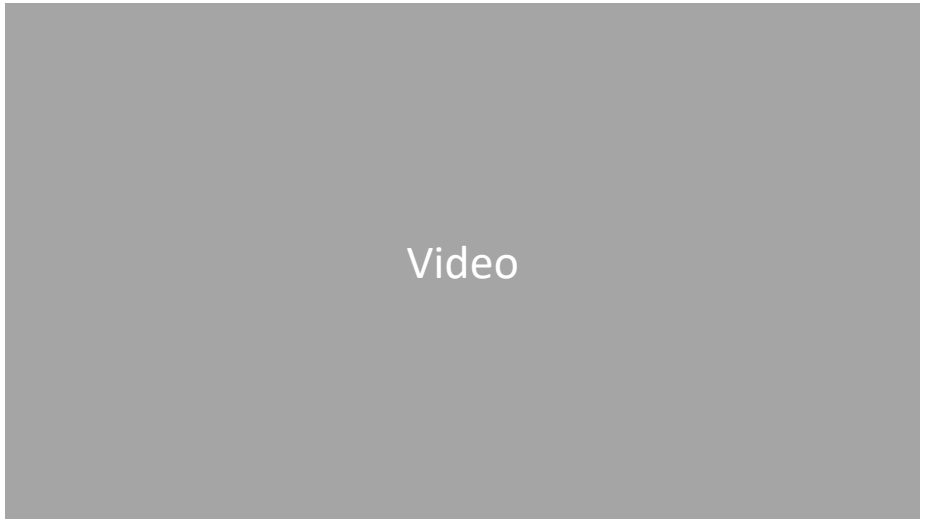
Video

TechEd

GOPAS

demo

GC.GetGCMemoryInfo()



TechEd

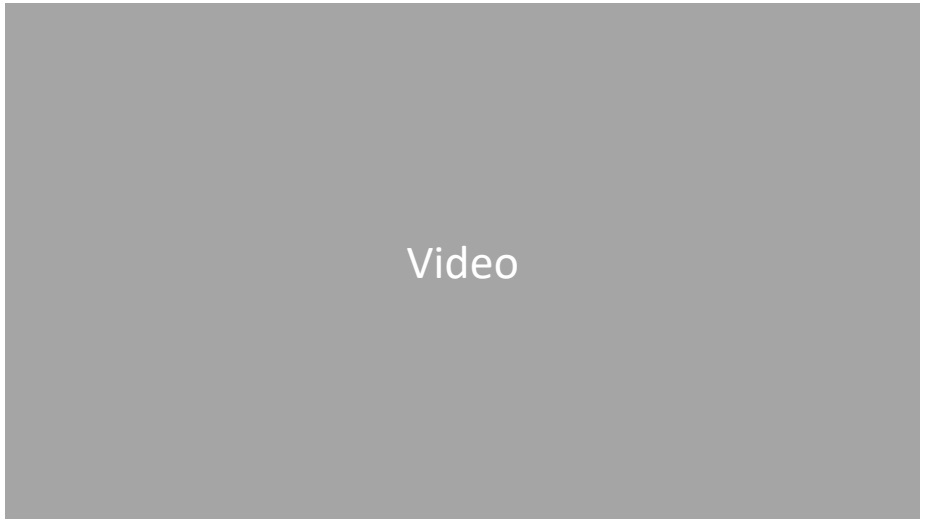
GOPAS

Other GC improvements in .NET5

Better decommit logic, ephemeral decommit moved out of STW

Card mark stealing

Vectorized mark list sorting (vctd. quicksort + vctd. bitonic) – AWX2+

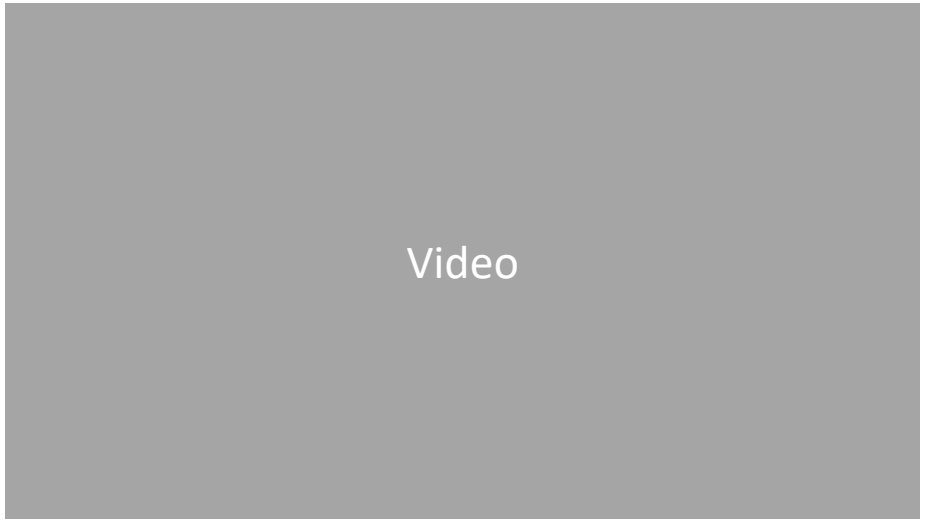


GOPAS

GC: Segements => Regions [NET6]

change of granularity

allows transitions of whole regions to another gen

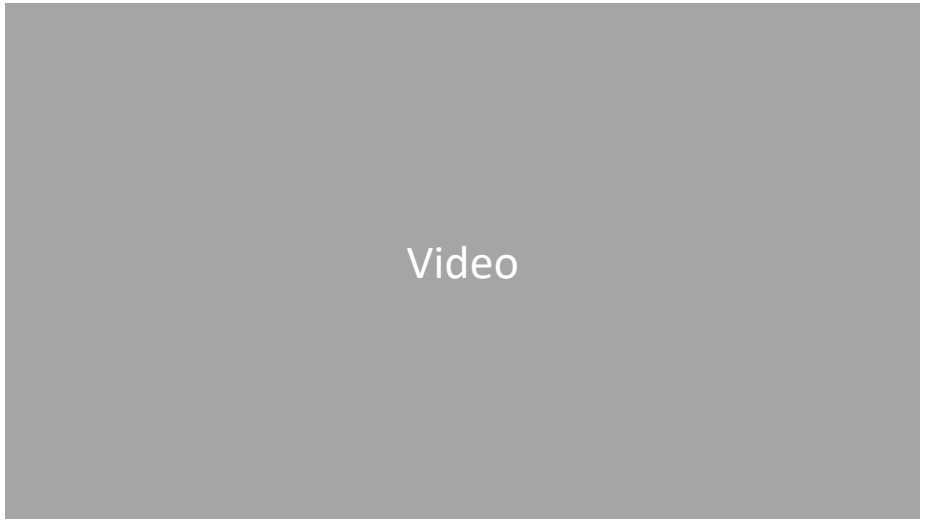


GOPAS

GC: Concurrent Compacting GC [NET-Future]

new flavor

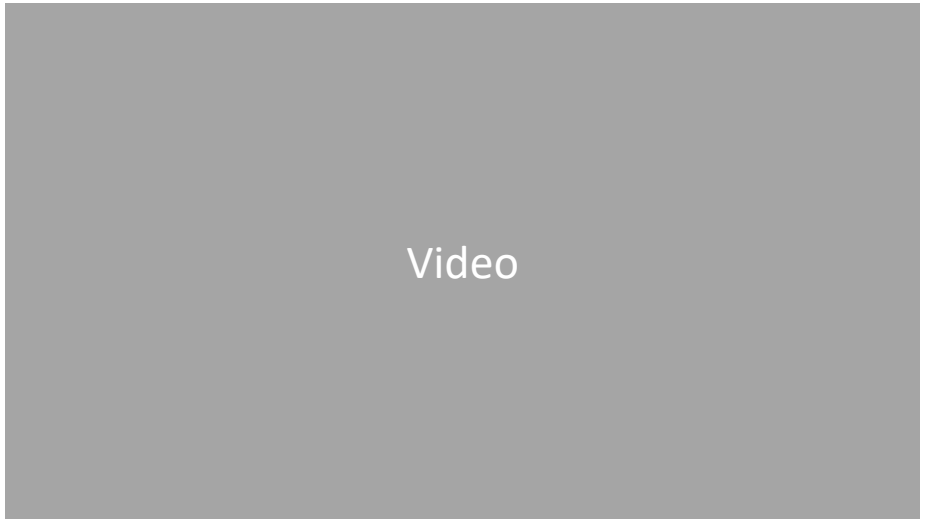
Read Barrier needed (currently only Write Barrier)



TechEd

GOPAS

Perf



TechEd

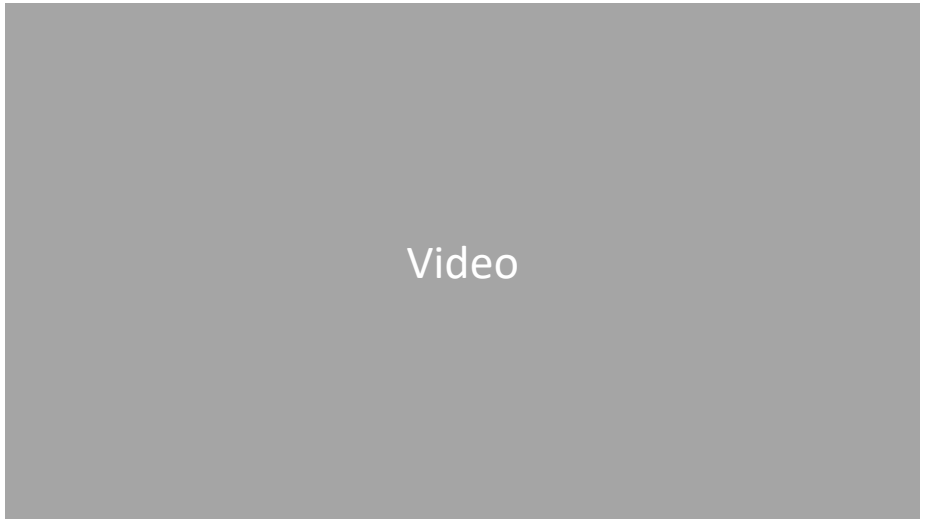
GOPAS

[SkipLocalsInit]

skip zeroing of stack locals

method/type/module scope

used on every assembly in netcoreapp



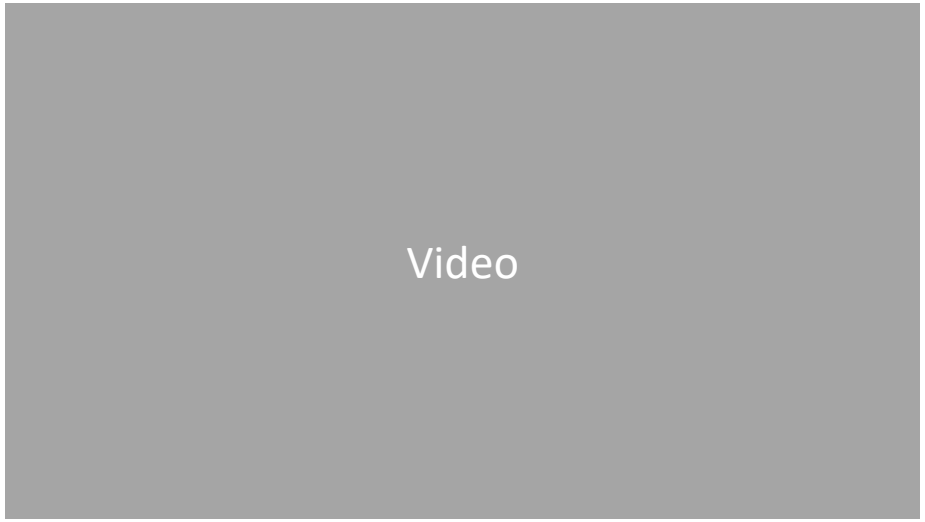
TechEd

GOPAS

`Sort()` – no longer native

fast C# implementation

allows GC to pause whenever needed



TechEd

GOPAS

Other

Video

TechEd

GOPAS

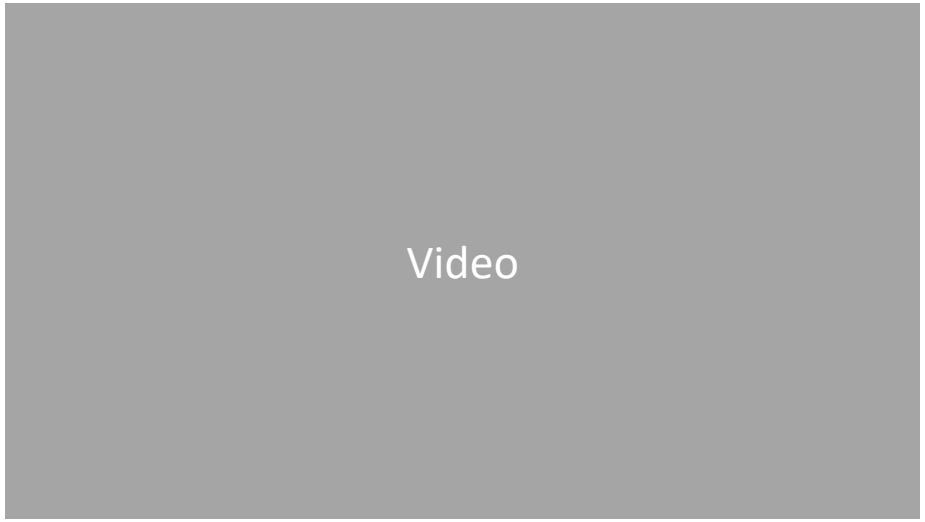
unloadable AssemblyLoadContext

replacement for AppDomains

no security, just loading isolation!

selective assembly sharing

unloading (.NET3+, GC backed)



TechEd

TechEd

Děkuji Vám za pozornost

Přednášející je Vám k dispozici na chatu ještě 15 minut
po skončení přednášky.


TechEd
ONLINE | 19. – 20. 5. 2021