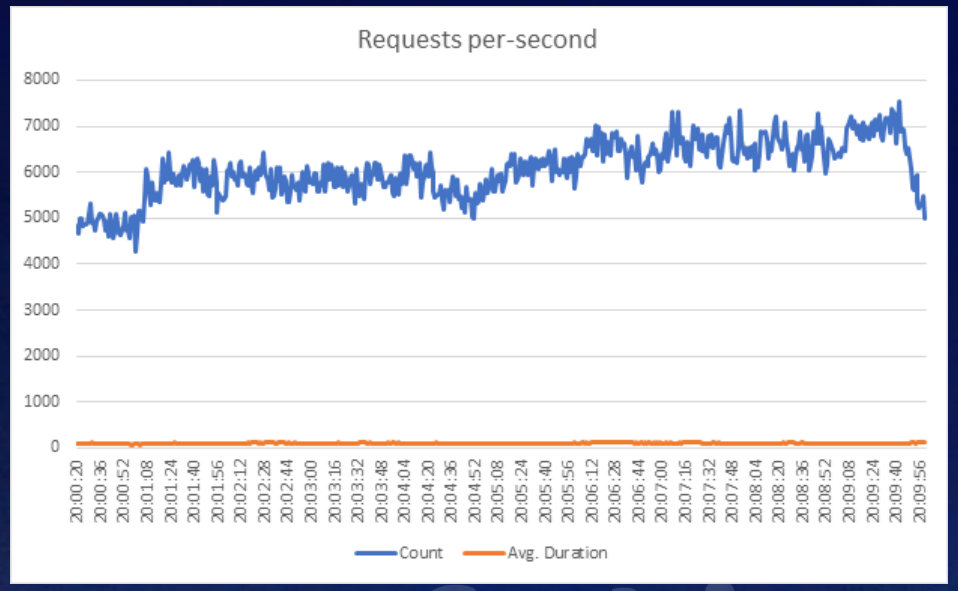
.NET Conf China
2022

从.NET Framework 到.NET 6
--- 微软广告平台的.NET 演进之旅

Mike Treit 微软广告平台首席工程师
刘盈 微软广告平台资深工程师

我们是谁？

- 微软广告平台在.NET 领域拥有10年以上的深耕经验。
- 微软广告平台由若干个技术团队构成，负责不同的业务领域，其中大部分团队都是C#开发为主。
- Campaign 平台是微软广告平台的核心组成部分之一。
- 为百万级客户提供广告投放服务，帮助用户更快、更精确的创建广告内容，让广告投入创造更大的价值。



.NET Conf China

Campaign 平台的技术背景

Campaign 平台的代码库数量庞大。

- Git repo里面包含600+ C# 项目 (*.csproj)
- 近8百万行代码。
- 直接引用近500个独立的nuget 包。

Language	Files	Lines	Code
Batch	269	5216	3627
C#	30556	7436049	6106337
F#	9	2205	1940
Java	3721	375448	219982
JavaScript	174	184997	122044
Perl	1	142	95
PowerShell	157	15624	11602
Python	354	57813	49269
TypeScript	1	138	108
Total	35242	8077632	6515004

.NET Conf China

Campaign 平台的技术背景

随着Campaign 平台十多年的发展，其底层托管服务也历经了一番演变。

1. On-Premise => Cloud
2. 我们最早的上云尝试仅仅是做”lift and shift”, 即只从自有硬件迁移到Azure Windows VMs，这是受原先的技术架构约束，主要体现在以下几点：
   - 基于.NET Framework开发，必须运行在Windows Server 上
   - 使用IIS 托管我们的 web server
   - 我们重度使用WCF来构建SOAP 服务。这对.NET迁移产生了重大的影响。
3. 截止到2018年，我们的应用是运行在Windows VM + .NET framework 4.6上。
4. 此后，我们经历了2年左右的时间，最终完成了从.NET framework 到.NET 的转换

.NET Conf China

为什么要迁移到.NET?

- .NET framework, .NET core, .NET?
- .NET 的优势
  - 跨平台
    - .NET为了跨平台而设计的， 这样我们就可以以此为契机去探索在Linux上运行的可能性。
  - .NET代表未来趋势
    - 应用运行时的新特性，仅在.NET 上实现
    - C#语言的新特性，仅在.NET上实现
    - .NET framework4.8是最后一个版本
  - 更快速的迭代
    - 独立、开源项目，没有一些特定的历史包袱，可以快速更新
  - 开源管理使问题的反馈、沟通效率大幅度提高
  - 更好的工具链支持， .NET CLI， 更简单的proj定义等

.NET Conf China

我们的演进路径

针对于公用类库，我们的演进路径是

- .NET Framework 4.6 -> .NET Framework 4.7 -> .NET Standard 2.0

针对于服务、应用类型的项目，我们的演进路径是

- .NET Framework 4.6 -> .NET Framework 4.7 -> .NET Core 3.1 -> .NET 5 -> .NET 6

.NET 提供了一系列工具帮助进行.NET迁移的分析和辅助更新:

- GitHub - dotnet/upgrade-assistant: A tool to assist developers in upgrading .NET Framework applications to .NET 6 and beyond

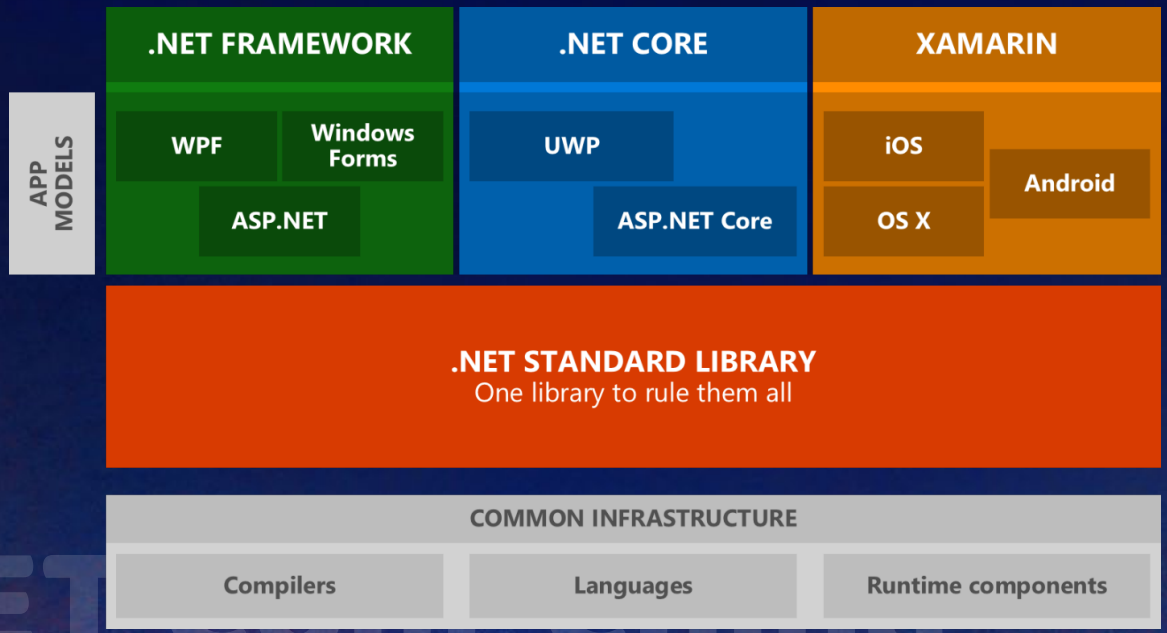
.NET Conf China

关键一环

.NET Standard

.NET standard

- 一套所有.NET 平台都可以实现的API规范
- .NET Standard2.0 被.NET framework, .NET core等实现，是从.NET framework迁移到.NET的唯一路径
- 根据我们的经验，.NET framework 4.7是对.NET Standard 2.0的完整实现的最低版本要求。
- .NET Standard2.1没有被.NET framework 实现



道阻且长

我们面临的挑战

缺失匹配的Nuget 包

- 我们正在使用的Nuget包中指定了targetframework是.NET 4.6, 无法与.NET standard2.0的project兼容。

```
<TargetFramework>net461</TargetFramework>
<TargetFramework>netstandard2.0</TargetFramework>
```

- Nuget包的新版本无法向后兼容，比如Unity (依赖注入框架，发布于2008年)
- 有些Nuget包不再更新，缺失了支持.NET standard 2.0的版本

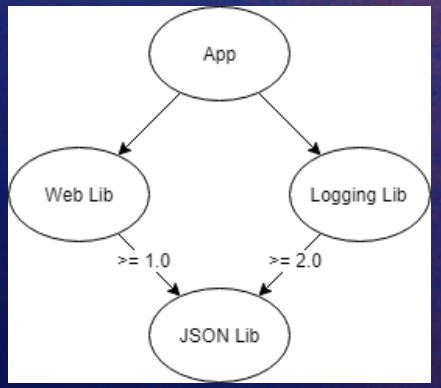
.NET Conf China

道阻且长

我们面临的挑战

版本管理问题

- Diamond Dependency ➔
- 你需要添加binding redirect element去解决




```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
     <assemblyIdentity name="....">
     <bindingRedirect
              oldVersion="0.0.0.0-4.0.6.0"
              newVersion="4.0.6.0"/>
    </dependentAssembly>
    <!-- ...and maybe some more... -->
  </assemblyBinding>
```

- 鉴于各种历史原因, 我们的项目存在大量binding redirect，且<AutoGenerateBindingRedirects>无法完全解决版本冲突的问题
- 持续地手工地在配置文件中添加binding redirect，既是一项繁重的工作，也让.NET迁移工作变得更加繁琐

.NET Conf China

道阻且长

我们面临的挑战

Windows Communication Foundation (aka: WCF)

- 45+ 服务构建于WCF之上。
- 已有客户群对WCF/SOAP API的重度使用

vs

.NET社区无意投入资源去支持WCF， 甚至考虑放弃对WCF的支持

.NET Conf China

道阻且长

我们面临的挑战

Windows Communication Foundation (aka: WCF)

- 45+ 服务构建于WCF之上。
- 已有客户群对WCF/SOAP API的重度使用

vs

.NET社区无意投入资源去支持WCF， 甚至考虑放弃对WCF的支持

.NET Conf China

行则将至

我们的解决方案

解决不兼容的Nuget包

- 建立私有的nuget 源 => Azure Artifacts
- 重新打包发布
  - ✓ 下载 *.nupkg文件，解压缩
  - ✓ 修改你需要的文件，并更新Package.nuspec 中的信息
  - ✓ 执行 “nuget pack” 命令生成新的nupkg 文件
  - ✓ 执行“nuget push” 推送到内部nuget站点
- Fork源码，修改后重新发布
- 反编译源码，修改后重新发布

.NET Conf China

行则将至

我们的解决方案

版本管理问题

从.NET framework的packages.config 到.NET的PackageReference , 让proj文件更小、更易于管理

- 我们使用了Converting C# projects to the new SDK format 来转换600+ proj文件。

```
dotnet tool install --global Project2015To2017.Migrate2017.Tool
```

- .NET 团队的try-convert 也可以完成类似功能

使用版本集中管理的方式在Packages.props文件中统一管理nuget包的版本信息。

- Packages.props 文件统一声明版本信息

```
<PackageReference Update="Newtonsoft.Json" Version="10.0.1" />
```

- Proj文件中尽可能只声明依赖关系, 尽可能减少versionoverride

```
<PackageReference Include="Newtonsoft.Json" />
<PackageReference Include="Newtonsoft.Json" VersionOverride="9.0.1" />
```

.NET Conf China

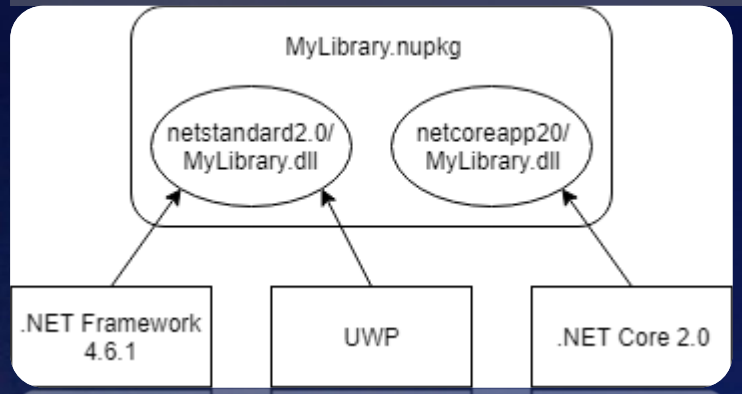
行则将至

我们的解决方案

WCF

- 微软最终决定在.NET中提供对WCF API 的支持，仅限于SOAP相关的子集，即CoreWCF 项目。
- 新的问题: CoreWCF 使用了全新的namespace, 原先的code无法平滑迁移
- 集中抽取所有WCF相关的code，为迁移做准备
- 采用multi-targeting的方式， 在迁移过程中同时支持.NET framework WCF与CoreWCF两套API

```
<TargetFrameworks>netstandard2.0;net461</TargetFrameworks>
```



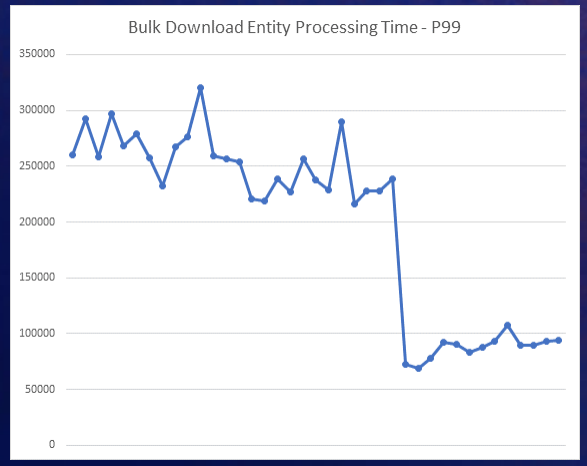

```
    public static bool IsSupported
    {
        get
        {
#if NET461 || WINDOWS_UWP
            return true;
#else
            return false;
#endif
        }
    }
```

.NET Conf China

行则将至

.NET 迁移后性能提升

- 右图是我们某一个service的latency曲线。
- 当我们完成了.NET迁移、在没有任何其他performance修改的前提下，仅仅是针对.NET进行了重新编译， 我们获得了更高的性能。



- 下图展示了迁移到.NET后， WCF服务的内存消耗大幅度下降

ServerName	WCF Service Mem Usage (KB)	Core WCF Service Mem Usage (KB)	Diff	% Reduction
CH01EAP00000109	1,280,764	682,236	598,528	47%
CH01EAP0000010A	1,286,864	646,360	640,504	50%
CH01EAP00000107	1,357,676	900,964	456,712	34%
CH01EAP00000108	1,074,220	637,052	437,168	41%
CH01EAP0000010B	1,190,948	700,104	490,844	41%
Avg	1,238,094	713,343	524,751	42%

.NET Conf China

行而不辍

我们在持续演进

与业界最佳实践接轨

- .NET的迁移更重要我们为日后的发展奠定了新的基础。
- 从 IIS -> Kestrel
- 从Windows VM -> Linux Container
- 从Azure Auto Pilot -> Azure Kubernetes

.NET Conf China

行而不辍

我们在持续演进

迁移到Linux Container

- 去掉Windows相关Runtime/ Assembly依赖项
- 处理路径分隔符的不同
- .NET上的优雅关闭

资源利用率提高 & 更高效的CICD流程

	Before (EAP)	After (AKS)	Improvement
CPU Cores	14,610	9,984	-32%
Deployment Time	45m	12m	-73%

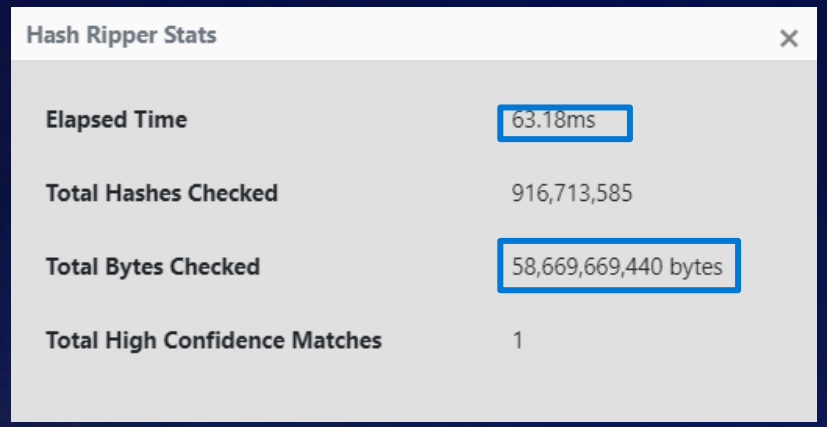
.NET Conf China

行而不辍

我们在持续演进

- 跟进.NET新特性、持续地提高性能
- 右图是我们做的一些探索性项目
  - .NET 6上，两两对比一百万条64字节数据，耗时大约60ms
  - 采用Span<T> 方式优化heap使用
  - 采用SequenceEqual 进一步提高性能

```
return bufferA.AsSpan().SequenceEqual(bufferB);
```



Method	Count	Mean	Error
CompareNormal	10000	4,635.0 ns	91.83 ns
CompareUnsafeHandRolled	10000	572.5 ns	9.33 ns
CompareSpanSequenceEquals	10000	198.3 ns	3.65 ns

.NET Conf China

未来可期

拥抱云原生

- 基于Kubernetes的高可用配置，提高平台可靠性。
- 基于SLA的自动化伸缩，提升平台的可扩展性
- 增强可观测性平台
  - Metrics： Prometheus + Grafana
  - Logging: Azure Data Explorer
- 开发、测试、生产环境的一致性，提升DevOps流程的效率
- 全自动版本更新流程、缩短发布周期
  - 基于Argo-Rollout 的金丝雀发布
- 基于云原生网关/Service Mesh技术，细粒度的流量治理

.NET Conf China

未来可期

迈向.NET 7

- 缩短startup time:
  - Faster, Lighter Apps with Native AOT
  - Startup time improvements with Write-Xor-Execute enabled
- 内置的容器化支持 built-in container support for the .NET SDK
  - 当前的方法： msbuild + docker build
  - .NET7：可以直接输出为container image

```
# create a new project and move to its directory
dotnet new mvc -n my-awesome-container-app
cd my-awesome-container-app

# add a reference to a (temporary) package that creates the container
dotnet add package Microsoft.NET.Build.Containers

# publish your project for linux-x64
dotnet publish --os linux --arch x64 -c Release -p:PublishProfile=DefaultContainer

# run your app using the new container
docker run -it --rm -p 5010:80 my-awesome-container-app:1.0.0
```

.NET Conf China

总结回顾

迁移到.NET6对我们来说是一个耗时巨大的工程， 但我们的收获也证明了这一切的努力都是值得的，既大幅度提升了我们当前服务质量，也有利于日后的可持续性发展。

如果您也在规划从.NET framework到.NET6的迁移，也许可以参考我们的步骤：

1. 当前.NET framework版本升级到.NET framework 4.7 或者 4.8
2. 先更改proj文件格式，把使用PackageReference作为第一步
3. 使用集中化的包管理机制，减少版本冲突。
4. 以.NET standard作为桥梁,连接.NET framework和 .NET项目

.NET Conf China

获取英文版博客原文

谢谢大家

Thank you!

Let's build amazing apps with .NET 7

get.dot.net/7