微软智能云MySQL数据库和.NET Core开源技术栈的联通

宋青见
C+E CCIC
Principle PM

Microsoft

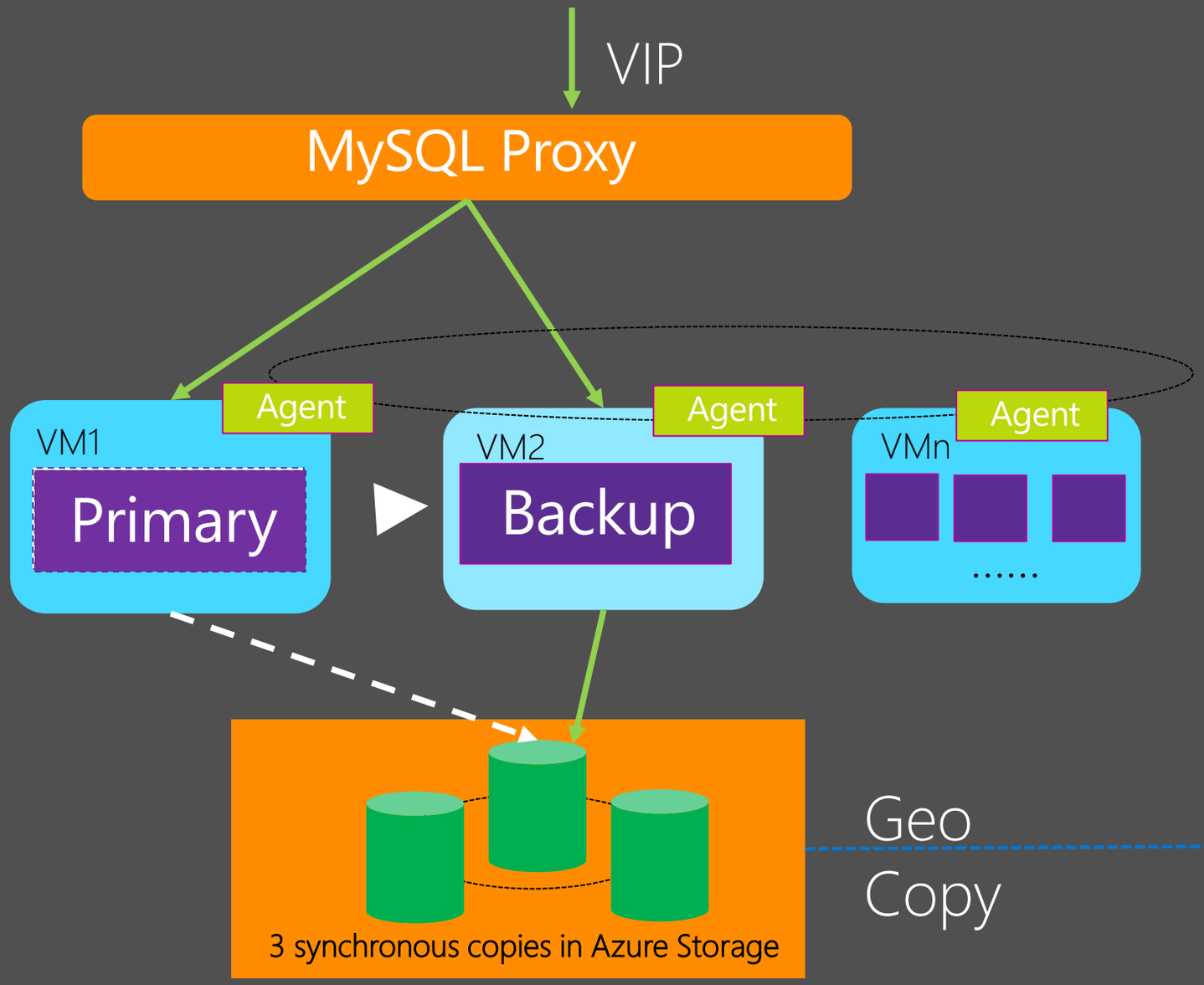
议题

- MySQL on Azure 托管服务简介
- .NET Core 简介
- Pomelo：Entity Framework MySQL Provider
- 应用案例: Senparc 微信SDK (最好的C#开源项目之一)

Microsoft

MySQL Database on Azure 简介

云原生的 MySQl PaaS服务： 高可用高可靠



基于Azure存储提供数据的高可用性和高可靠性

本地3份同步数据拷贝
异地3份异步数据拷贝

数据库服务的高可用

设计和运维以99.9%高可用为标准
Enable-Secondary 启用备用库
达到99.99%

支持异地灾备恢复

自建MySQL服务器的痛点

- 需要有基础架构部署运维专业人员搭建和维护
- 需要专业DBA知识经验
- 自己设计、搭建及测试故障切换转移费时费力，人力和资源成本较高

自建云端或本地MySQL 服务器
1. 安装主从集群
2. 设计及搭建故障切换转移(failover)
3. 设计及搭建备份系统
4. 维护OS和MySQL服务器
5. 系统监控
6. 系统调优
需要一周的搭建时间和持续的维护

MySQL PaaS 的好处

简单快速部署，高性价比，高可用性、以及高安全性的全托管的服务

MySQL PaaS
快速创建MySQL服务器
2 分钟

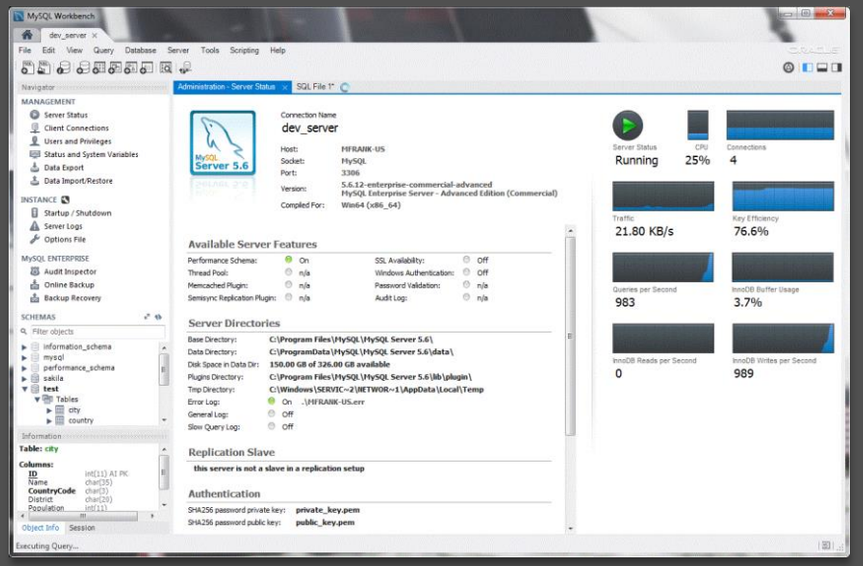
+

额外的好处
1. 高可用性、高可靠性和服务运行保证99.9%协议
2. 提供本地异地多重备份，保障数据高度可靠和安全
3. 每天自动备份 + 任意时间点的回滚
4. 支持replica
5. 支持灾备恢复
6. 提供丰富的监测指标

支持熟悉的平台和工具

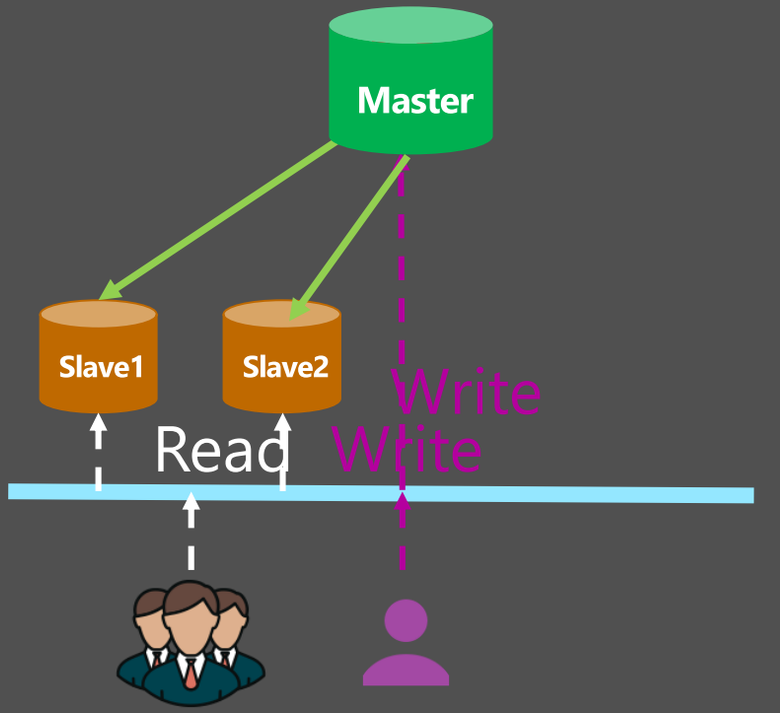
后台用的DB Engine是MySQL 社区版本（Community Edition）

支持现有的MySQL客户端和工具 (例如phpMyAdmin, MySQL workbench, navicat 等）



新功能: 启动备用库

- 可读备用库功能可以帮助用户进一步增强MySQL实例的可用性。 99.99% SLA
- 当用户为MySQL实例启动可读备用库功能后，Azure会自动创建一个备用实例。 主备实例之间通过复制技术同步数据。
- 当主实例发生故障，Azure会自动提升备用实例为新的主实例，并且创建新的备用实例。
- 故障恢复时间通常在分钟级（一般在60秒内）。故障转移后无数据丢失。
- 用户可以将备用库作为只读实例使用，以满足应用程序的大量读需求。
- 2017年5月1日正式上线



新功能：Premium版本计划 MP1/MP2 SKU

- 基于Azure高级存储（更高的IO吞吐量和更小的IO延迟）
- 更高的计算资源配置
- 包含更大的免费存储容量
- 包含更长时间的备份和任意时间点的回滚
- 2017年5月1日正式上线

.NET Core 简介

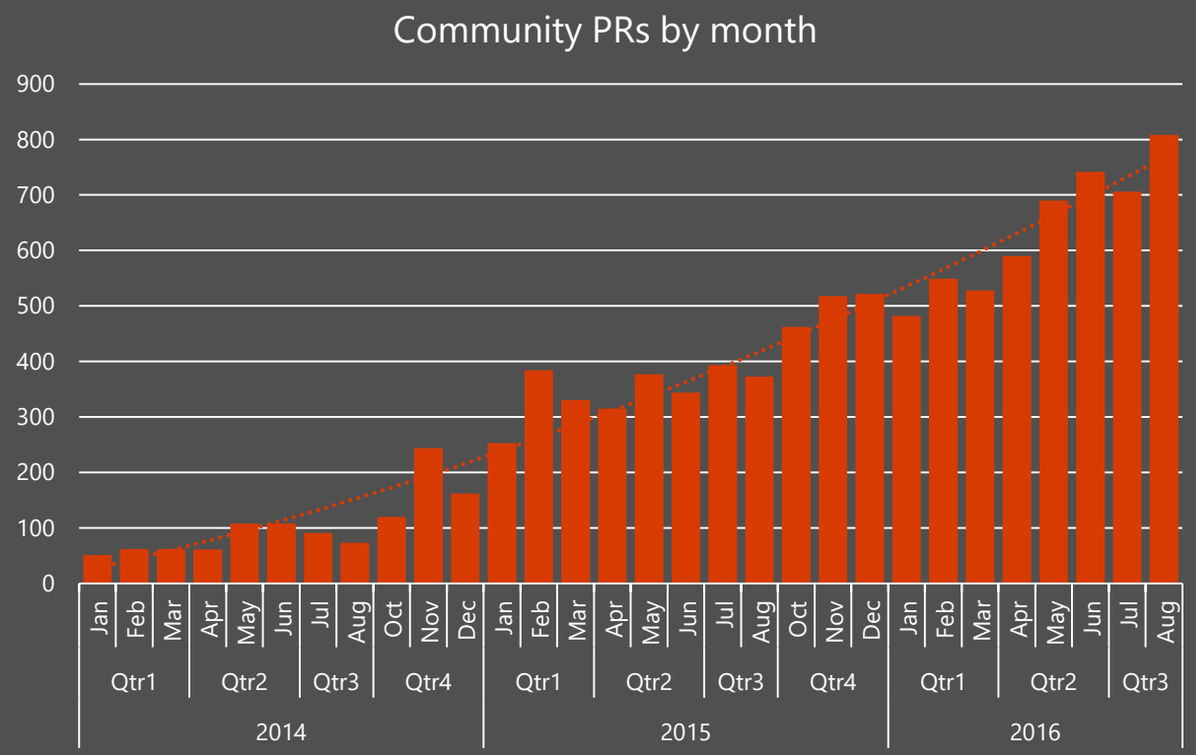
Microsoft

61%

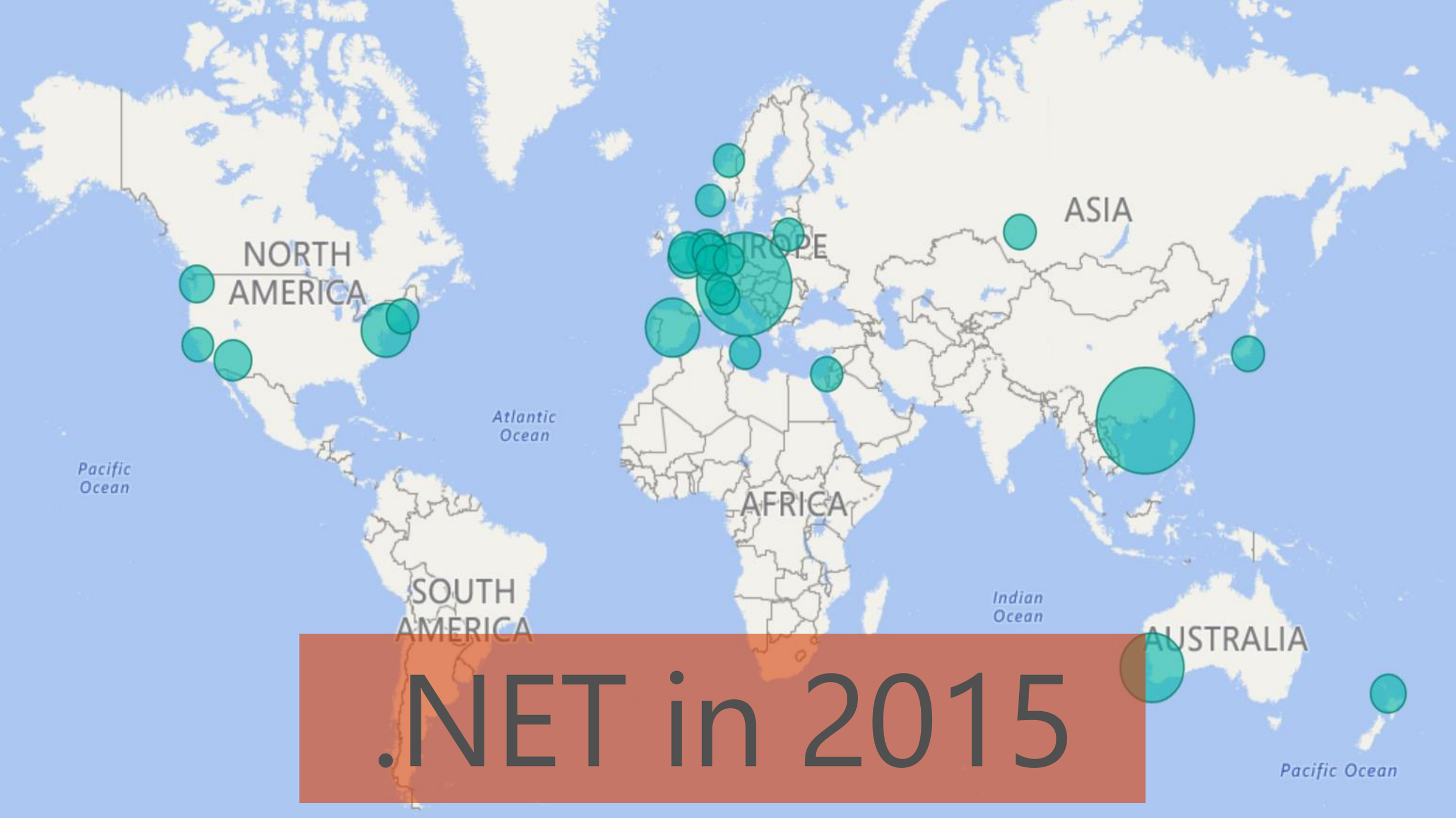
2016年比2015年 新增 .NET 活跃开发者
(VS 2012+)

40%

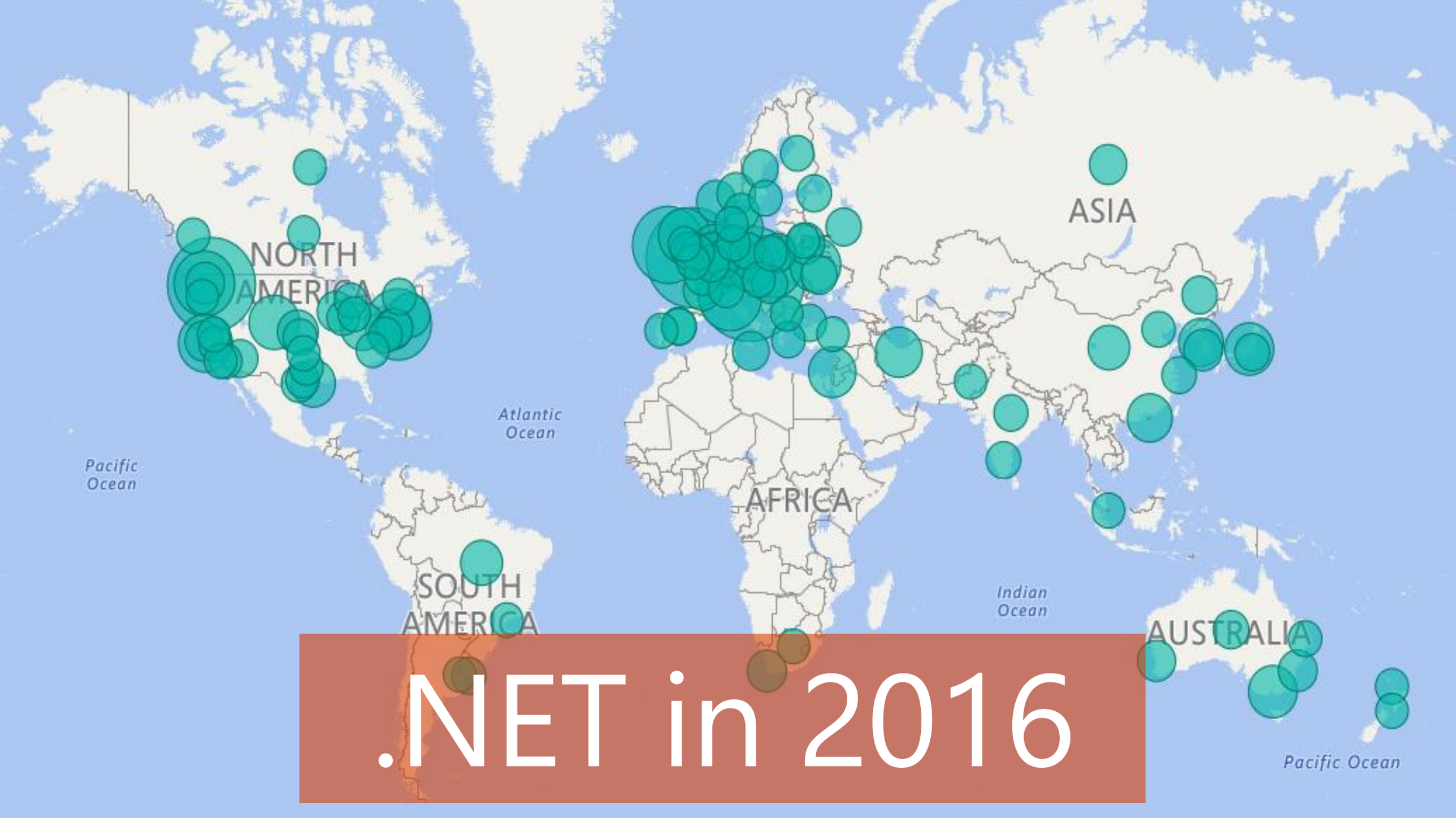
.NET Core 2016年新的开发者的下载量



62% GitHub 贡献者来自微软之外
(corefx / coreclr repos)


NORTH AMERICA
EUROPE
ASIA
Atlantic Ocean
Pacific Ocean
AFRICA
SOUTH AMERICA
Indian Ocean
AUSTRALIA
Pacific Ocean

.NET in 2015

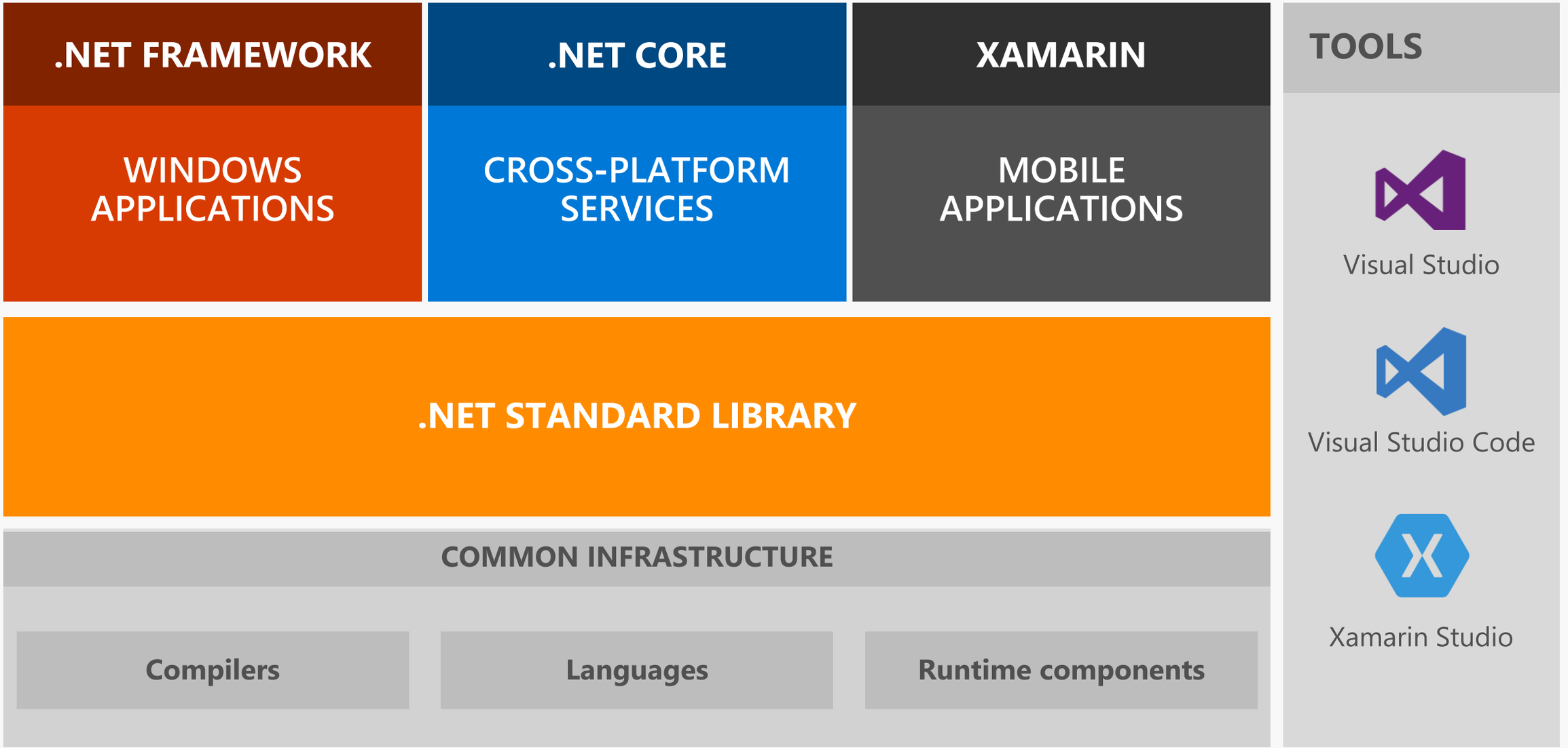

NORTH AMERICA
ASIA
Atlantic Ocean
Pacific Ocean
AFRICA
SOUTH AMERICA
Indian Ocean
AUSTRALIA
Pacific Ocean

.NET in 2016

.NET 的全新世界


WINDOWS APPLICATIONS
（Windows 应用程序）
CROSS-PLATFORM SERVICES
（跨平台服务端应用）
MOBILE APPLICATIONS
（移动应用程序）
.NET FRAMEWORK
.NET CORE
XAMARIN

统一的平台架构


.NET FRAMEWORK
.NET CORE
XAMARIN
TOOLS
WINDOWS APPLICATIONS
CROSS-PLATFORM SERVICES
MOBILE APPLICATIONS
.NET STANDARD LIBRARY
COMMON INFRASTRUCTURE
Compilers
Languages
Runtime components
Visual Studio
Visual Studio Code
Xamarin Studio

.NET Core

跨平台

Windows, Linux and macOS.

快速

快于Node.js8倍， 快于 Go 3倍

轻量

灵活的部署方式，非常适合容器的模块化开发方式

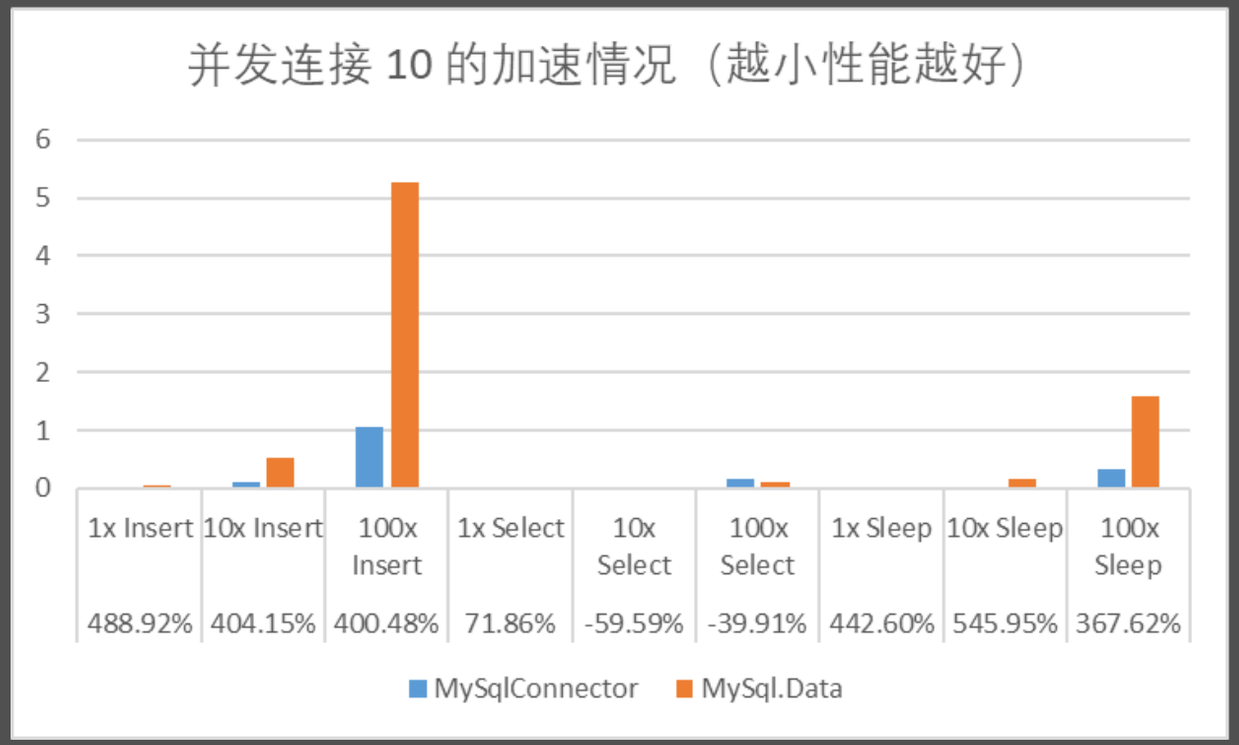
开源

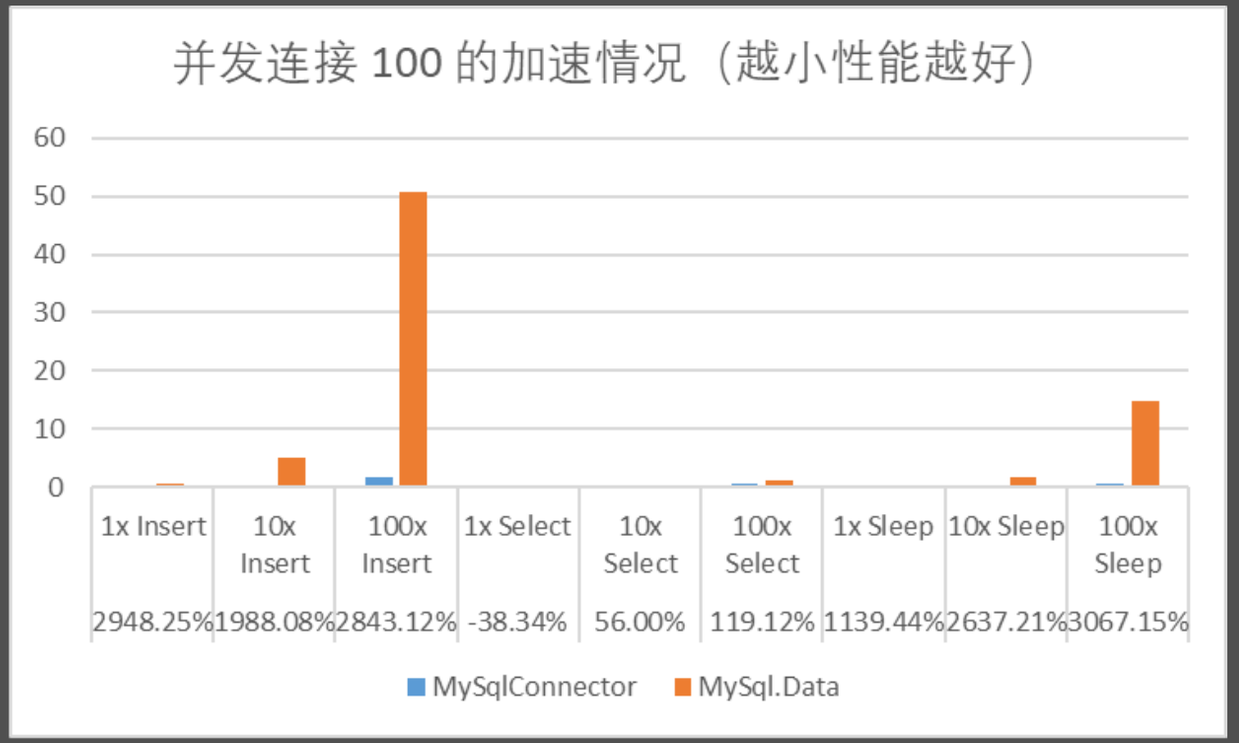
运行引擎, 框架链接库, 编译器，语言和开发工具都开放在GitHub上

Pomelo – EF MySQL Provider

比官方性能更好的 ADO.NET/.NET Core MySQL Connector

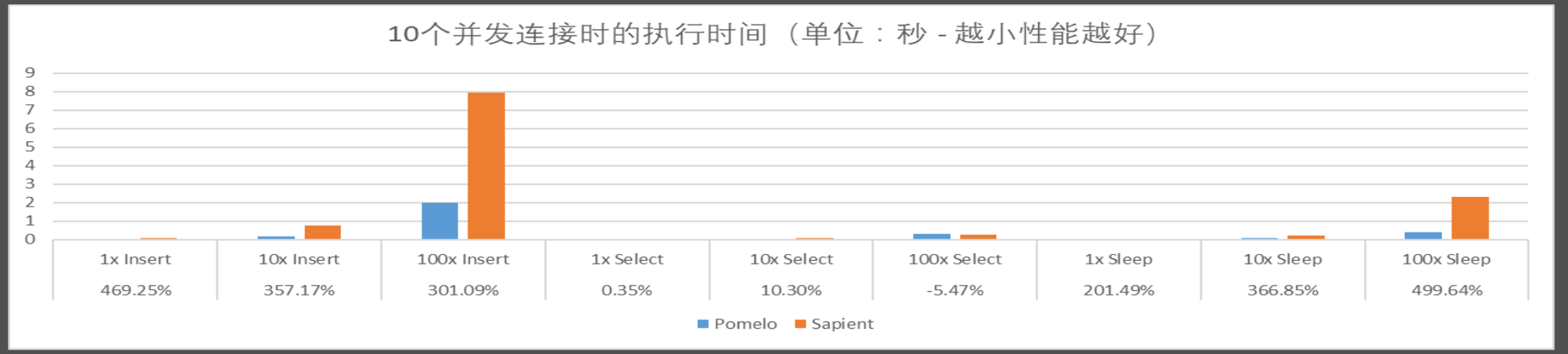
- https://github.com/mysql-net/MySqlConnector
- 采用了LibUV, 完全异步的方式大幅提升性能

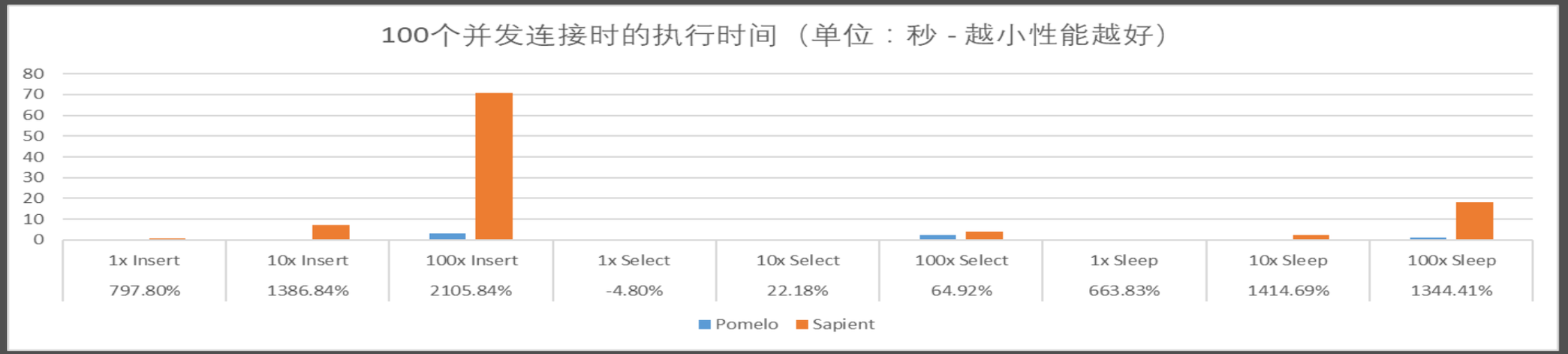


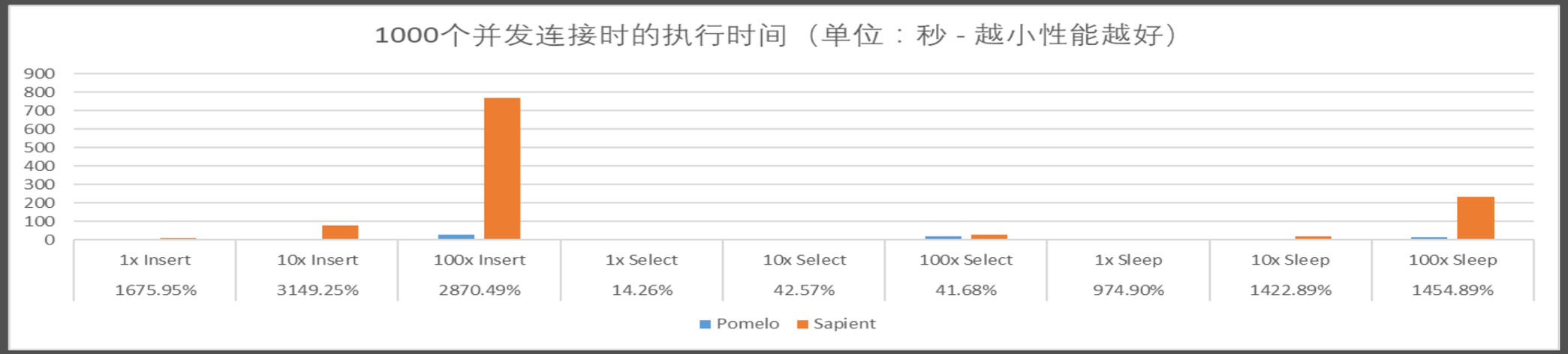


弥补了.NET应用和MySQL数据库之间的ORM一环

- https://github.com/PomeloFoundation/Pomelo.EntityFrameworkCore.MySql
- 在 MySqlConnector之上开发的 Entity Framework Core provider
- 下载量最大，超过官网同类插件
- 为MySQL Database on Azure进行了专门的测试和优化
- License: MIT


10个并发连接时的执行时间（单位：秒 - 越小性能越好）
9
8
7
6
5
4
3
2
1
0
1x Insert
469.25%
10x Insert
357.17%
100x Insert
301.09%
1x Select
0.35%
10x Select
10.30%
100x Select
-5.47%
1x Sleep
201.49%
10x Sleep
366.85%
100x Sleep
499.64%
Pomelo
Sapient


100个并发连接时的执行时间（单位：秒 - 越小性能越好）
80
70
60
50
40
30
20
10
0
1x Insert
797.80%
10x Insert
1386.84%
100x Insert
2105.84%
1x Select
-4.80%
10x Select
22.18%
100x Select
64.92%
1x Sleep
663.83%
10x Sleep
1414.69%
100x Sleep
1344.41%
Pomelo
Sapient


1000个并发连接时的执行时间（单位：秒 - 越小性能越好）
900
800
700
600
500
400
300
200
100
0
1x Insert
1675.95%
10x Insert
3149.25%
100x Insert
2870.49%
1x Select
14.26%
10x Select
42.57%
100x Select
41.68%
1x Sleep
974.90%
10x Sleep
1422.89%
100x Sleep
1454.89%
Pomelo
Sapient

Senparc 微信 SDK

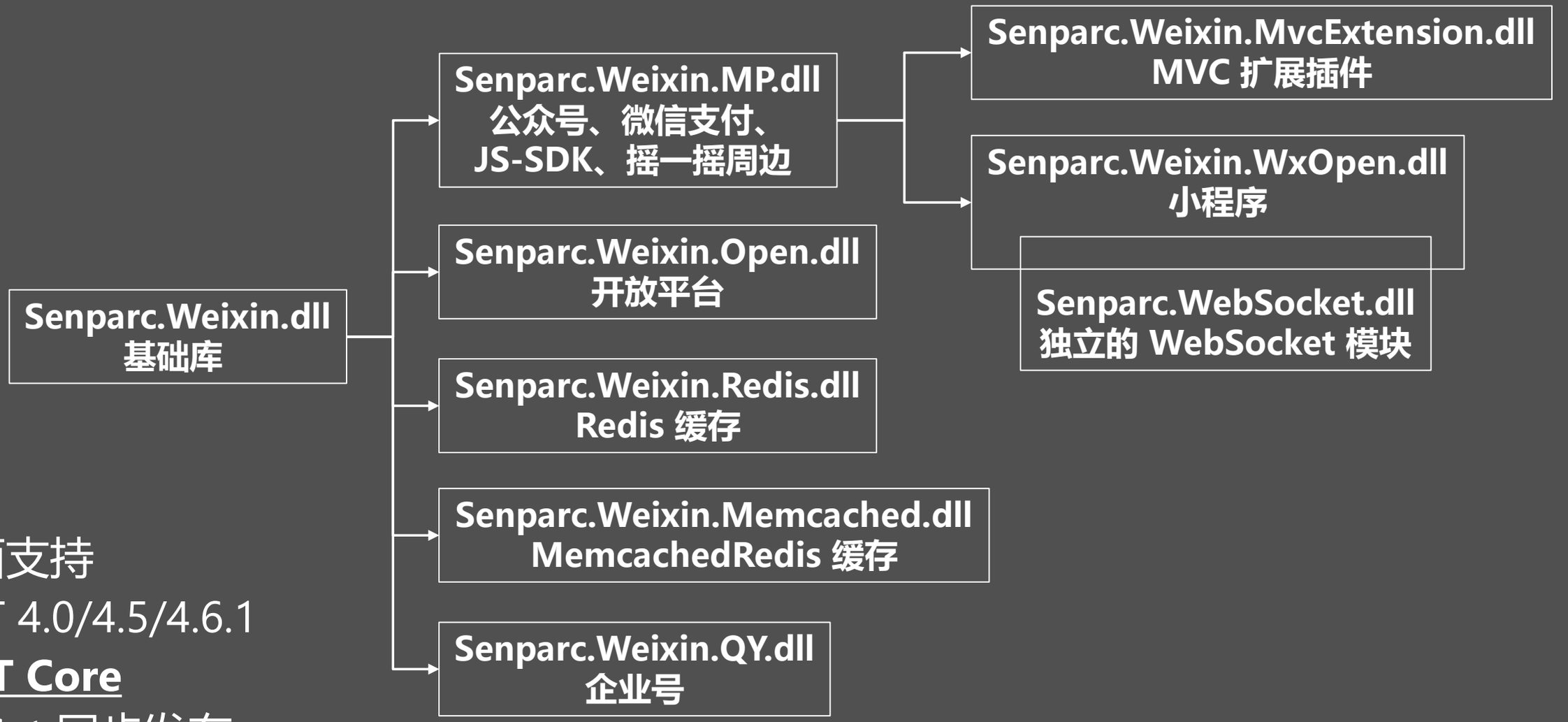
国内最好的C#开源项目之一

Senparc.Weixin SDK

- Senparc.Weixin SDK 是目前 GitHub 上 Star 和 Fork 数最多的国内 C# 项目，也是使用率最高的微信 C# SDK。
- Senparc.Weixin SDK 提供了全面的微信开发套件，帮助开发者快速创建稳定、高效、高扩展能力的微信项目。
- 开源项目地址：https://github.com/JeffreySu/WeiXinMPSDK

Senparc.Weixin SDK

模块架构



全面支持

.NET 4.0/4.5/4.6.1

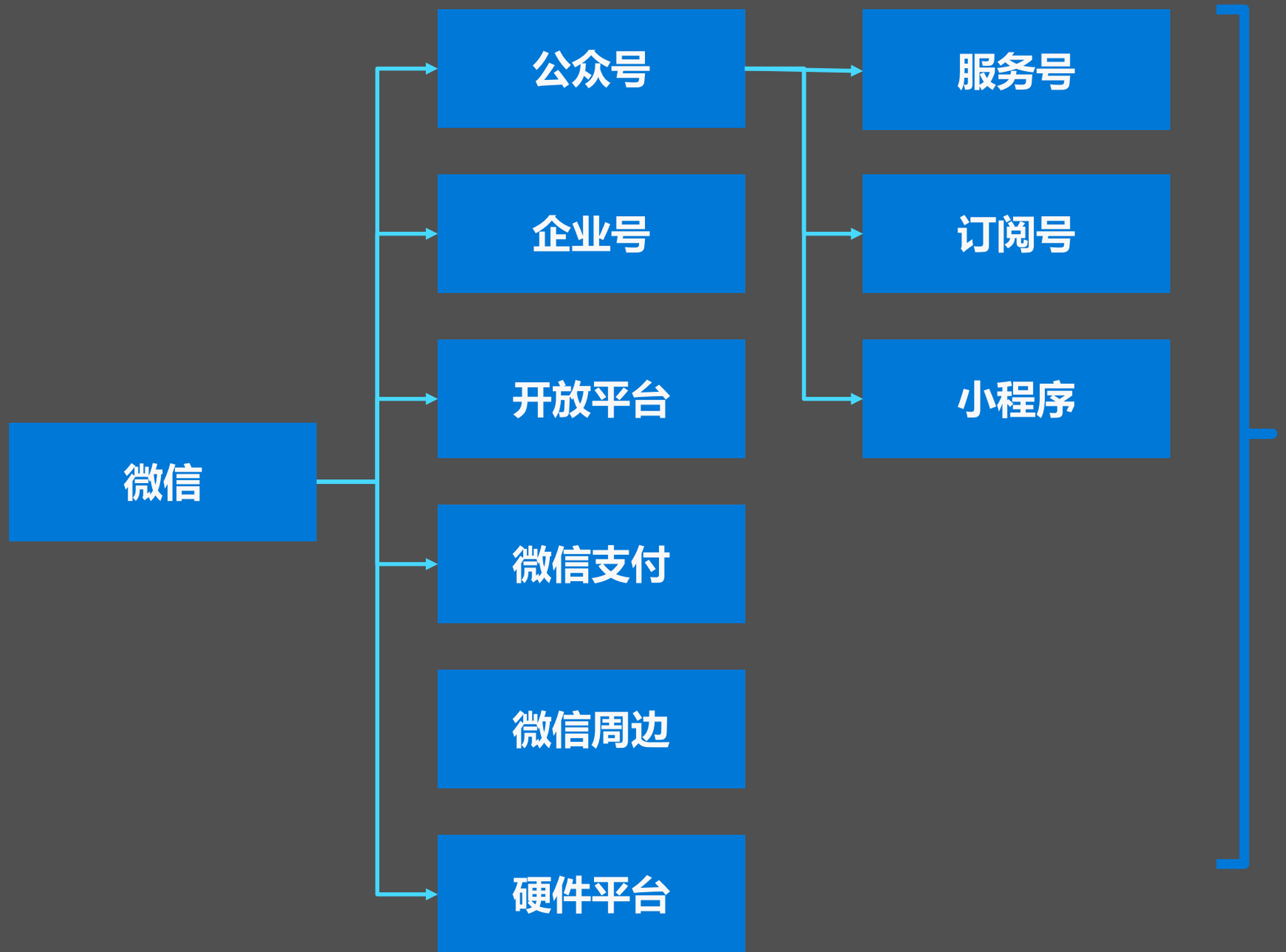
.NET Core

NuGet 同步发布

开源地址：https://github.com/JeffreySu/WeiXinMPSDK

#	功能模块	Nuget 文件名	Nuget 版本	当前站点运行版本	.NET 4.0	.NET 4.5	.NET Core
1	SDK 公共基础库	Senparc.Weixin.dll	nuget v4.12.2	v4.11.11	4.0 Y	4.5 Y	core Y
2	公众号 微信支付 JSSDK 摇一摇周边	Senparc.Weixin.MP.dll	nuget v14.4.4	v14.4.1	4.0 Y	4.5 Y	core Y
3	开放平台 SDK	Senparc.Weixin.Open.dll	nuget v2.4.0	v2.3.8	4.0 Y	4.5 Y	core Y
4	企业号 SDK	Senparc.Weixin.QY.dll	nuget v4.3.0	v4.2.3	4.0 Y	4.5 Y	core Y
5	企业微信 SDK	Senparc.Weixin.Work.dll	nuget inaccessible	即将发布	即将发布	即将发布	即将发布
6	MVC 扩展插件	Senparc.Weixin.MP.MvcExtentsion.dll	nuget v4.4.0	v4.3.0	4.0 Y	4.5 Y 4.6.1 Y	core N
7	Core MVC 扩展插件	Senparc.Weixin.MP.CoreMvcExtentsion.dll	nuget v1.0.0	未部署	4.0 N	4.5 N 4.6.1 Y	core Y
8	Redis 缓存	Senparc.Weixin.Cache.Redis.dll	nuget v1.3.0	v1.2.0	4.0 N	4.5 Y	core Y
9	Memcached 缓存	Senparc.Weixin.Cache.Memcached.dll	nuget v0.3.0	v0.2.0	4.0 N	4.5 Y	core Y
10	小程序	Senparc.Weixin.WxOpen.dll	nuget v1.3.0	v1.2.1	4.0 N	4.5 Y	core Y
11	WebSocket 模块	Senparc.WebSocket.dll	nuget v0.2.0	v0.1.3	4.0 N	4.5 Y	core Y

微信平台生态基本全部覆盖



Senparc.Weixin SDK

+

Azure

+

My SQL

代码实现：只需三行代码（.NET Framework）

```
[HttpPost]
public ActionResult Index(PostModel postModel)
{
    // 1. 定义MessageHandler
    var messageHandler = new CustomMessageHandler(Request.InputStream, postModel);
    // 2. 执行微信处理过程
    messageHandler.Execute();
    // 3. 返回结果
    return new WeixinResult(messageHandler);
}
```

代码实现：只需三行代码（.NET Core）

```
[HttpPost]
public ActionResult Index(PostModel postModel)
{
    // 1. 定义MessageHandler
    var messageHandler = new CustomMessageHandler(Request.Body, postModel);
    // 2. 执行微信处理过程
    messageHandler.Execute();
    // 3. 返回结果
    return new WeixinResult(messageHandler);
}
```

.Net Core 版本充分考虑了系统从 .NET Framework 移植过来的可能性，多个版本之间的命名空间及方法保持了高度的一致，同时针对.NET Core的特性增加了DI等参数设置方法。总之怎么玩都可以。

设置消息处理逻辑

```
public class CustomMessageHandler : MessageHandler<MessageContext>
{
    public public CustomMessageHandler(Stream inputStream, PostModel postModel)
        : base(inputStream, postModel, maxRecordCount) {}

    //处理文字类型消息
    public override IResponseMessageBase OnTextRequest(RequestMessageText requestMessage)
    {
1       var responseMessage = CreateResponseMessage<ResponseMessageText>();//定义响应消息类型
2       responseMessage.Content = “您刚才发送了：“ + requestMessage.Content;//设置响应参数
3       return responseMessage;//返回响应对象
    }

    //要处理更多消息类型只需重写对应的OnXxRequst方法。
}
```

微信高级接口调用

第一步：注册（全局一次）

```
AccessTokenContainer.Register(appId, secret);
```

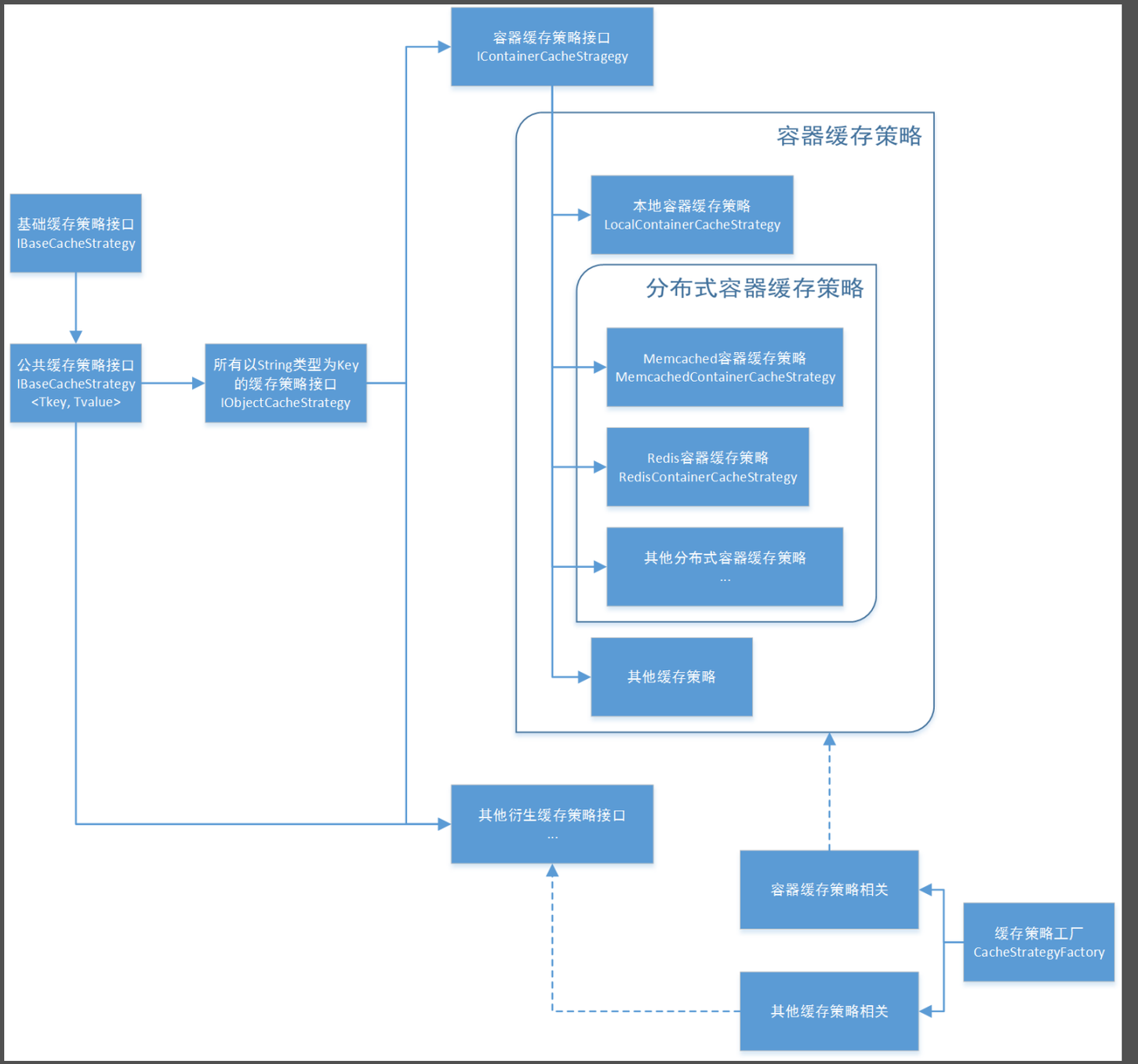
第二步：使用（任意地方）

```
var userInfo = UserApi.Info(appId, openId);
```

开发者可以完全忘记 AccessToken 及其过期时间、缓存序列化等细节问题，SDK全部自动处理，开发者只需要专注业务逻辑。

缓存架构

- 缓存策略全部面向接口，高度规范可扩展；
- 支持缓存框架热切换；
- SDK默认已经提供三种缓存方式：
  - 本地缓存
  - Redis（支持分布式）
  - Memcached（支持分布式）
- 可自由重写和扩展任意缓存框架（包括数据库）；
- 专为微信数据制作了"容器"中间层缓存；
- 充分解耦，底层使用工厂模式管理和调用缓存服务，开发者只需开发业务代码（最简单的 Key-Value 读写），无需关注缓存服务本身。



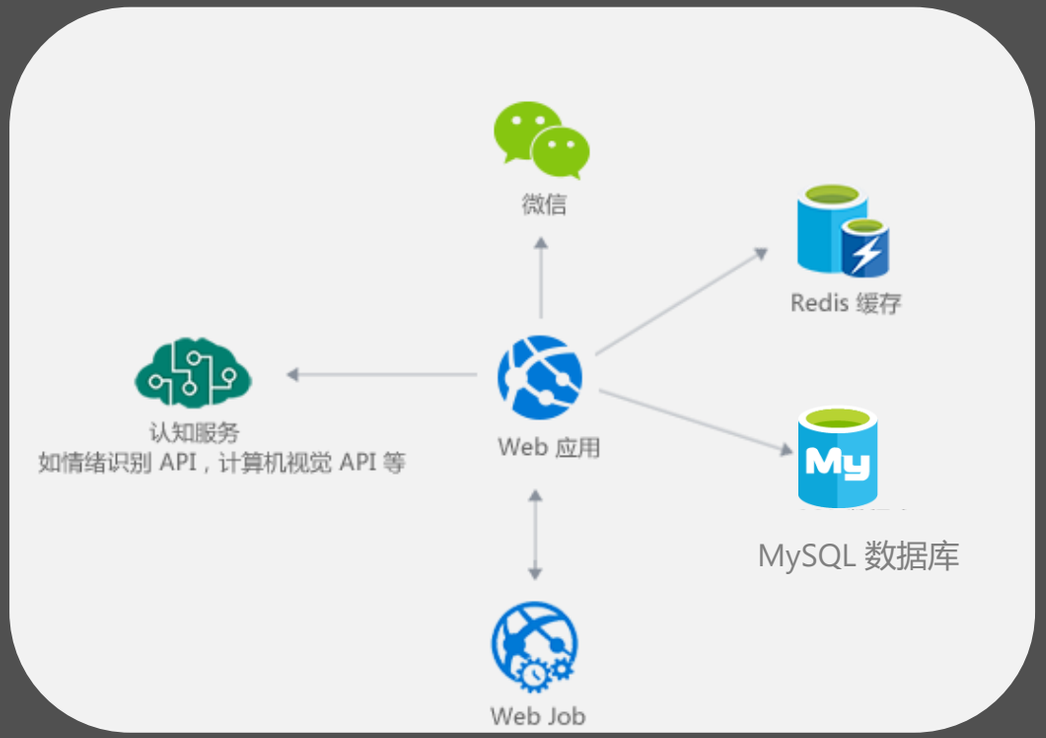
演示：

1. 扫码进入
微信入口

2. 上传照片
Web App

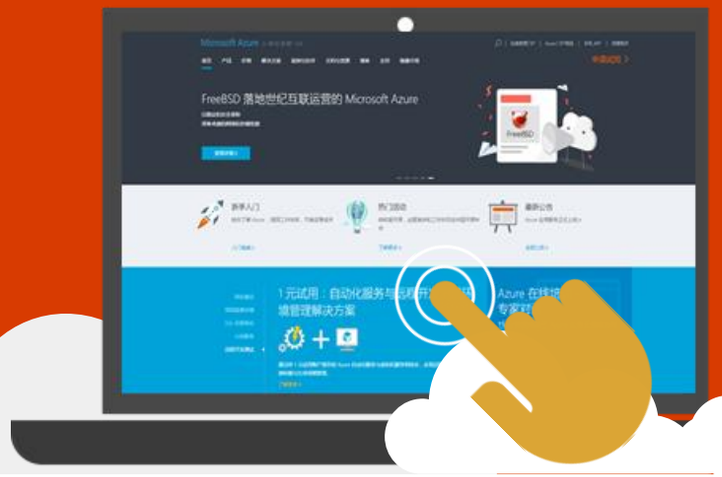
3. 表情识别
认知服务

4. 存储
Redis+MySQL



Senparc
小助手

更多信息与资源



- Azure 中国官网 https://www.azure.cn/ 提供最新产品与解决方案信息， 技术文档，以及SDKs下载
- Azure 应用程序开发说明 https://www.azure.cn/dev-notes/ 概述了海外与中国区服务开发人员需要注意的区别
- 申请一元试用，即刻体验 Azure 服务：https://www.azure.cn/pricing/1rmb-trial-full/
- Azure 镜像市场： https://market.azure.cn/

Microsoft 云科技公众号

Azure 云助手手机 App

Thank you!

Appendix - 附录

功能介绍

快速创建，无需维护基础架构

无需DBA经验

只需简单的输入几个参数，

兼容MySQL协议
(支持5.5，5.6， **5.7**)

2分钟内创建有高可用的MySQL服务器

无需维护VM和网络

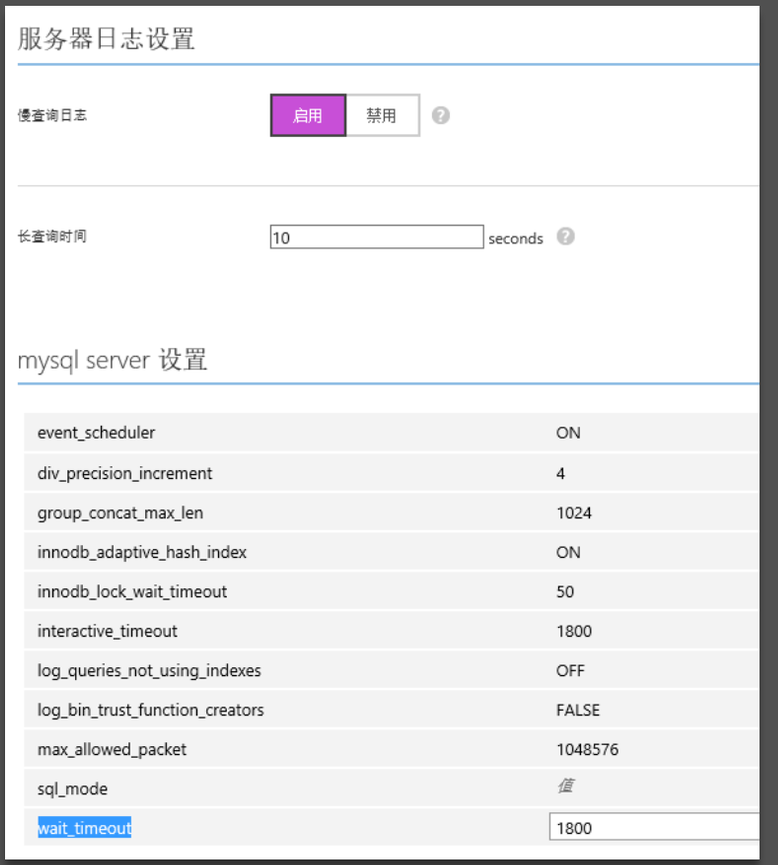
无需维护MySQL服务器本身

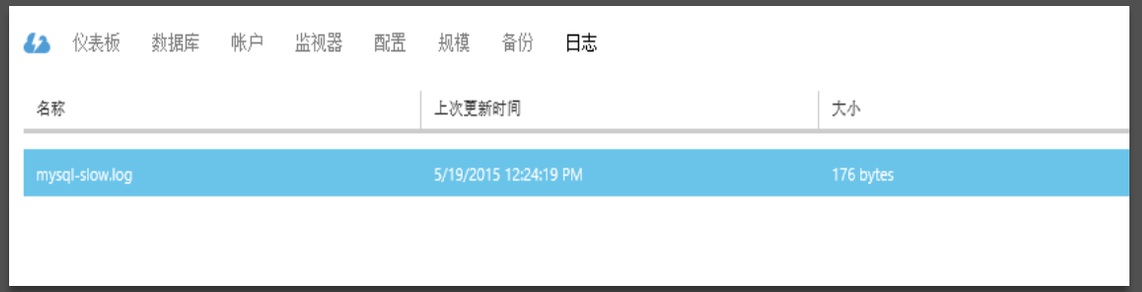
支持参数定制和日志

支持开启慢查询日志

支持定制常用的MySQL服务器参数

- event_scheduler
- div_precision_increment
- group_concat_max_len
- innodb_adaptive_hash_index
- innodb_lock_wait_timeout
- wait_timeout
- long_query_log
- max_allowed_packet
- sql_mode
- time_zone
- …





提供完善的监控体验

提供20+监控参数

- CPU %
- 峰值/平均的总QPS及子查询类别QPS(Select, Update, 等等)
- 成功/失败/节流的数据库链接数
- 慢查询
- …

提供多种视图 (1小时 / 24小时 / 7天)

支持第三方的监控工具如Zabbix

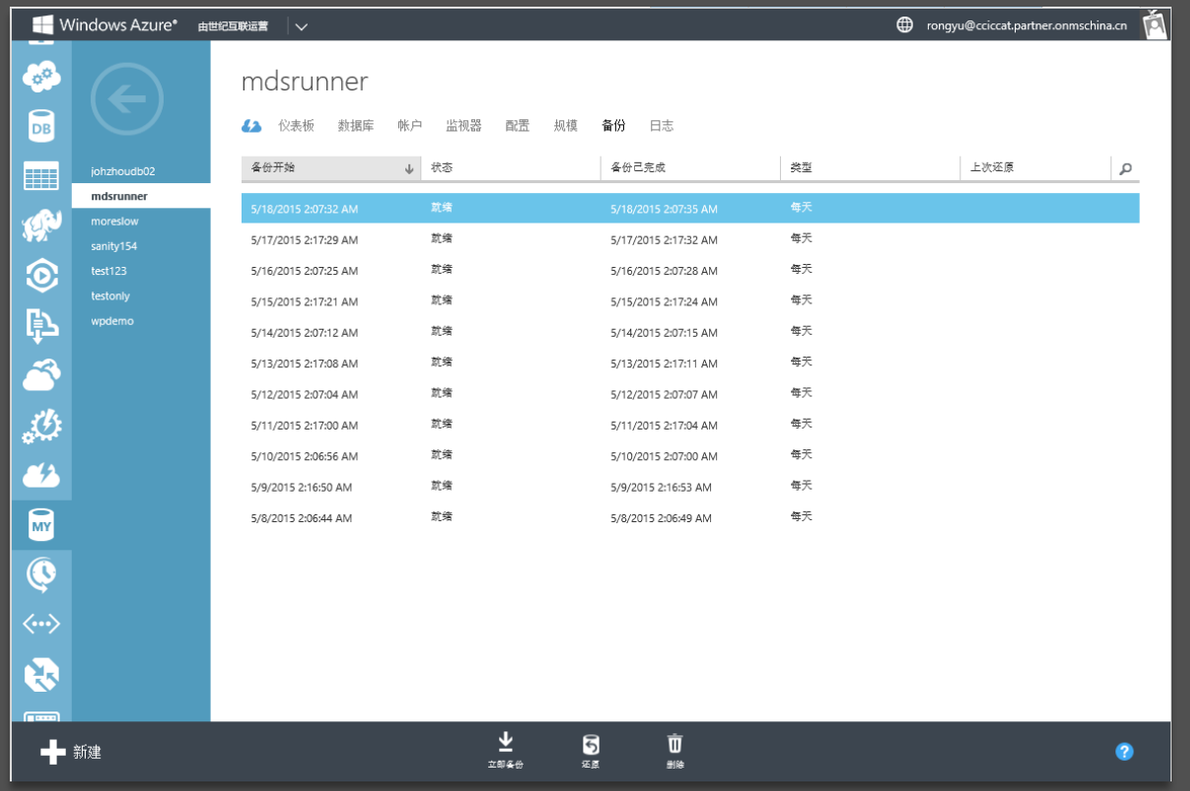
数据库备份和恢复

支持全量快照备份和增量备份

- 系统自动做日备，日备保留长达30天
- 支持即时快照备份
- 日备和即时快照备份一般在1分钟以内完成
- **备份所用存储不计费!**

灵活的数据库恢复选择

- 基于系统日备和即时快照备份的恢复(e.g.应用升级失败时的快速恢复）
- 支持恢复到任意一个时间点
- 支持异地恢复（可用于数据库异地迁移或者在不同环境快速拷贝数据库）



任意时间点的回滚 (Point In Time Restore)

用户场景

- 数据库数据损坏或数据丢失
- 检查历史数据

功能

- 支持回滚到过去7天的任意时间点
- 时间点精确到分钟
- 支持恢复到新的服务器实例
- 支持恢复到另一个数据中心

参考文档：

https://www.azure.cn/documentation/articles/mysql-database-point-in-time-restore

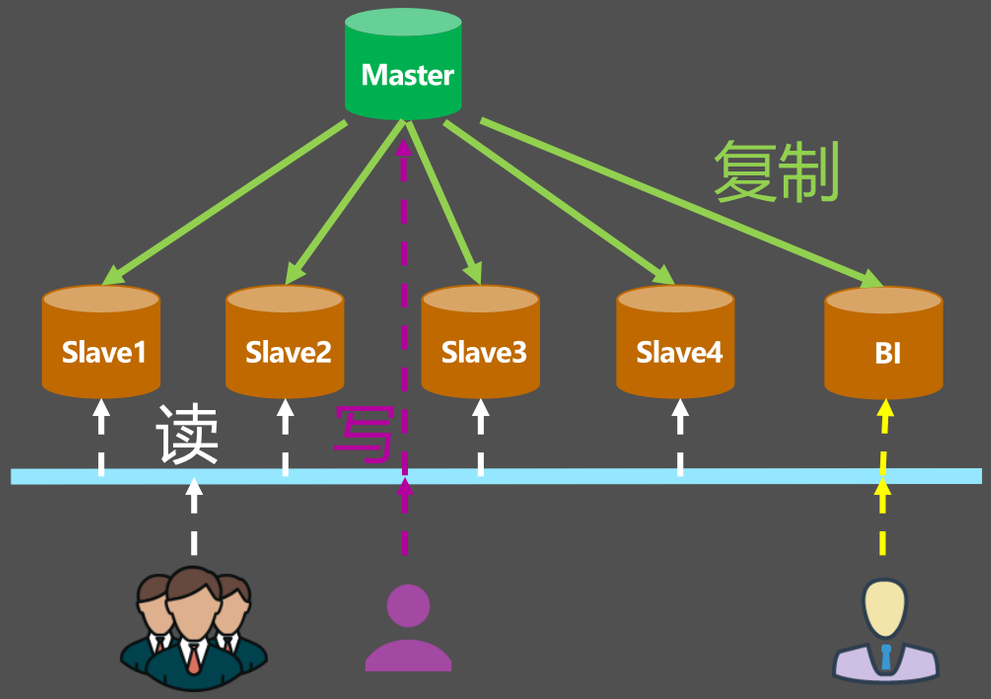
支持弹性扩展与收缩

简化购买体验，按客户性能需求付费而不是基础架构的配置

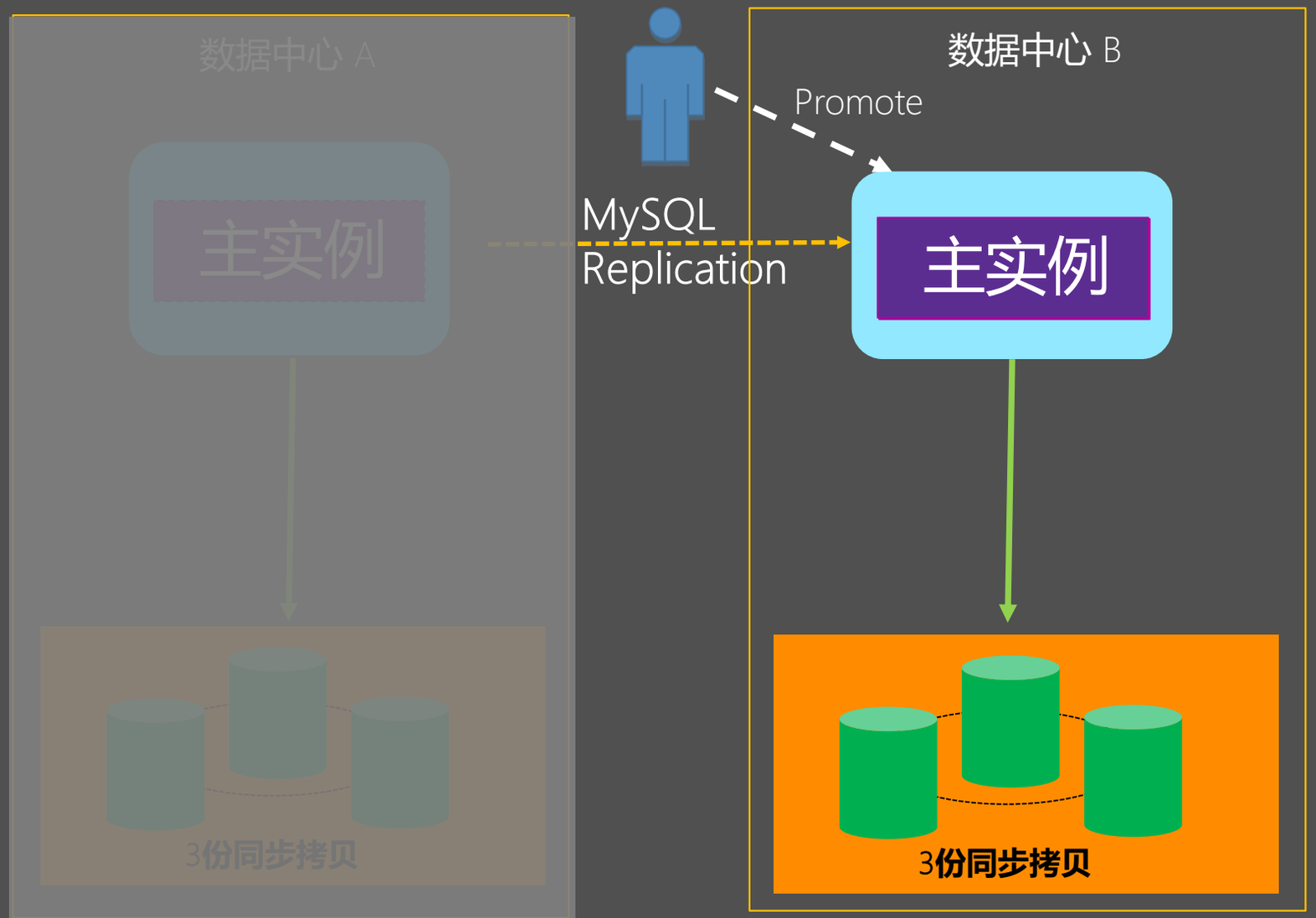
- 提供6个性能不同层级的版本, 成倍数提高
- 支持在不同层级之间快速切换
- 按小时付费

支持基于副本(Replica)的横向扩展

- 轻松实现弹性扩展，增加运行能力
  - 适用于读取压力较大，或者读远大于写的业务
  - 最多可添加5个从属实例
  - 在应用程序中实现读写分离
- 只读实例其他适用场景
  - 商业智能报告(BI)
  - 不同部门之间的数据共享



灾备恢复 – 基于异地副本（replica）



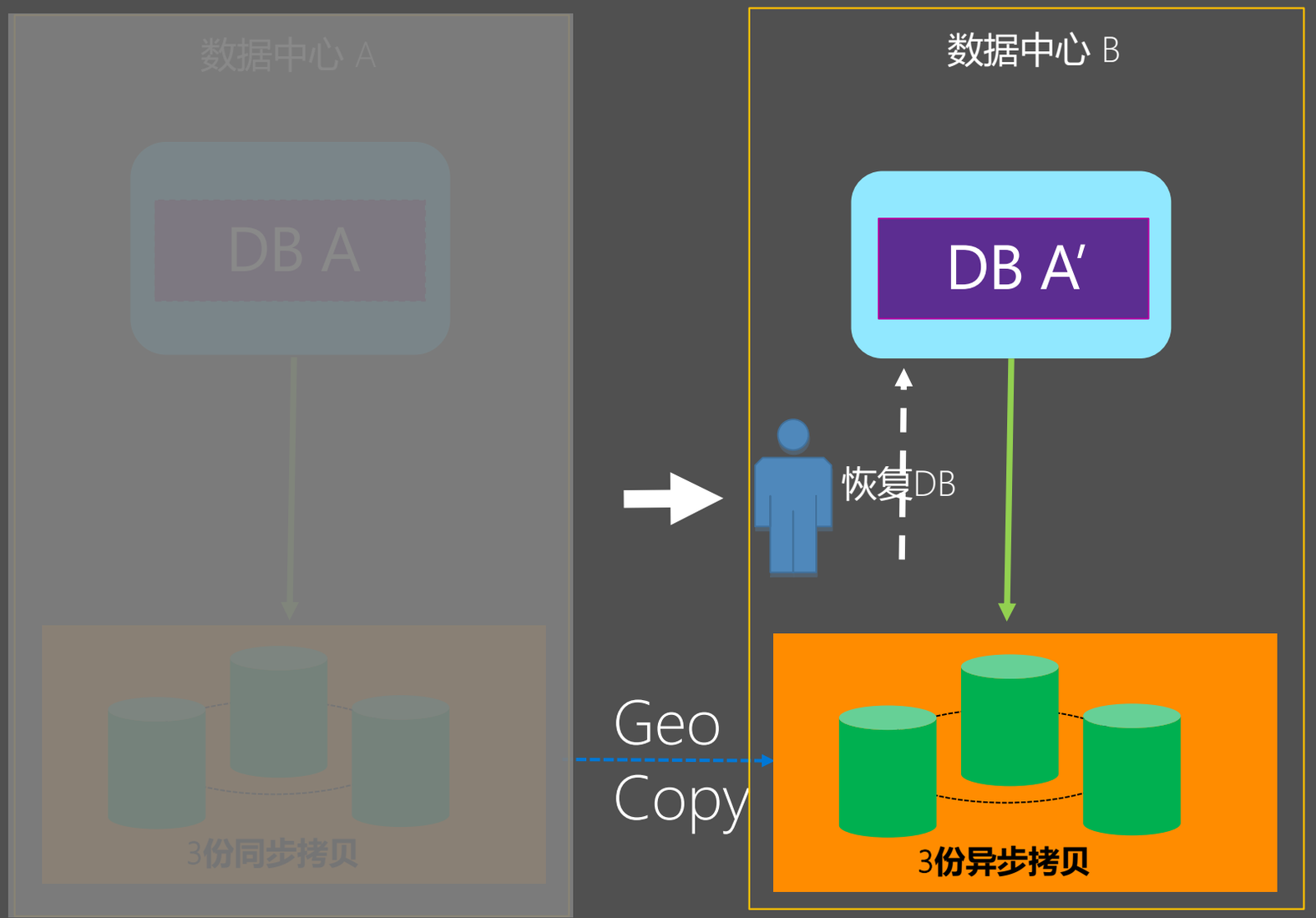
基于异地副本的恢复

灾备恢复指标
RPO < 30秒， ERT < 10秒

运维人员通过Portal或Powershell升级副本为主实例

需要支付异地副本的费用

灾备恢复 – 基于异地数据库恢复



基于异地数据拷贝的恢复

灾备恢复指标

RPO < 1小时， ERT < 3小时

运维人员通过Powershell进行恢复

所有版本具备这个功能，没有额外费用

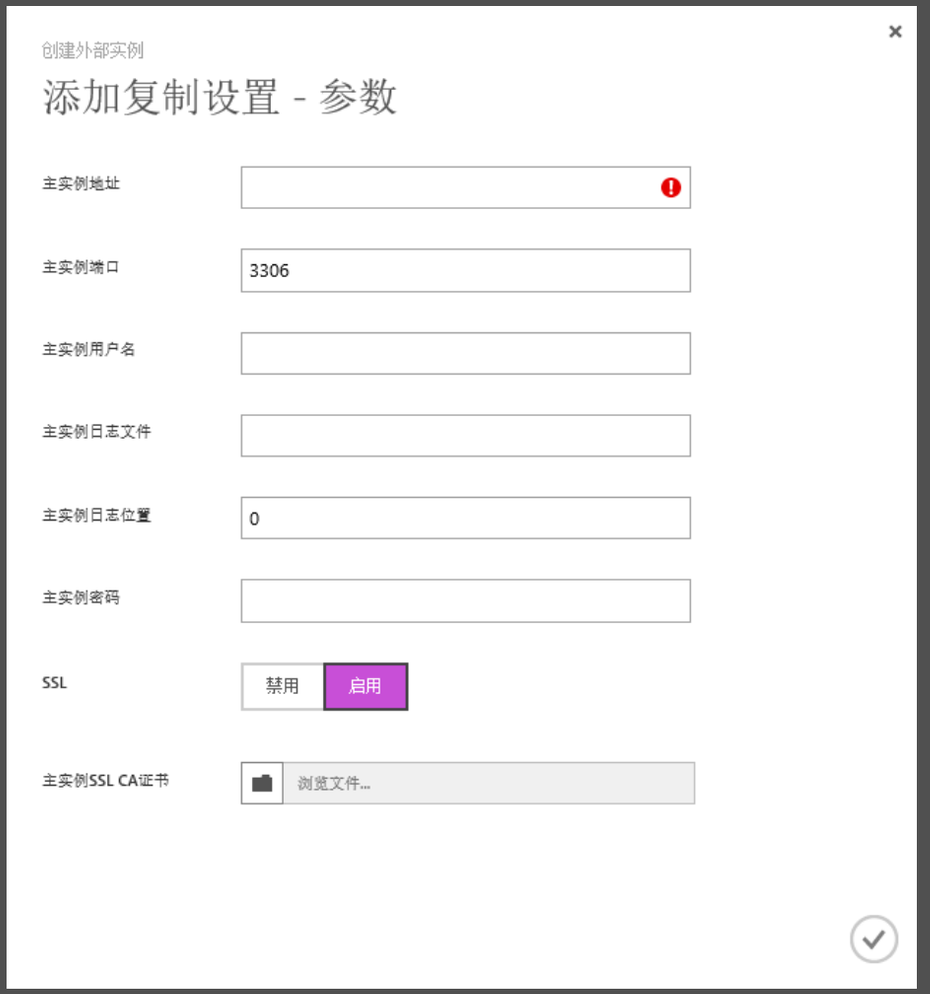
支持混合云的数据库同步

支持标准的MySQL Slave模式

常见混合云场景

- 从本地同步数据库到Azure上以满足Azure上的应用需要
- 支持应用平滑迁移到Azure

通过管理门户配置同步和查看同步状态



阿里云MySQL主，Azure MySQL从*

- 目前需要工程师团队后台协助+DevOps工作， 5.7（单基础机版）比较容易，5.6（双机高可用）有一定工作量，5.5（双机高可用）比较困难

配置文档

- http://www.windowsazure.cn/documentation/articles/mysql-database-data-replication

版本和价格

MySQL Database on Azure性能版本

每个版本包括**免费**数据库容量100GB，超过的部分按0.61元/GB/月计费。

针对不同性能层级进行调优，每个性能版本都有保证的最低性能指标。

	MS1	MS2	MS3	MS4	MS5	MS6	MP1	MP2
性能层级	每分钟处理几百个事务	提供约2倍于MS 1的性能，每分钟处理几百个到上千个事务	提供约3倍于MS 2的性能，每分钟处理几千个事务	提供约3倍于MS 3的性能，每分钟处理几千到上万个事务	提供约1.5倍于MS4的性能	提供约1.5倍于MS5的性能	提供1.5倍 – 3 倍性能于MS6	提供几乎两倍的性能于MP1
数据备份	每日自动备份，备份保留30天， 支持7天内任意时间点回滚	每日自动备份，备份保留30天, 支持7天内任意时间点回滚	每日自动备份，备份保留30天, 支持7天内任意时间点回滚	每日自动备份，备份保留30天, 支持7天内任意时间点回滚	每日自动备份，备份保留30天, 支持7天内任意时间点回滚	每日自动备份，备份保留30天, 支持7天内任意时间点回滚	每日自动备份，备份保留35天, 支持35天内任意时间点回滚	每日自动备份，备份保留35天, 支持35天内任意时间点回滚
最大支持同时连接数	100	200	500	1000	1500	2000	4000	6000
正式商用版价格	¥0.09/小时（~¥67/月）	¥0.18/小时（~¥134/月）	¥0.48/小时（~¥357/月）	¥1.44/小时（~¥1071/月）	¥2.16/小时（~¥1607/月）	¥3.24/小时（~¥2410/月）	¥7.64/hr (~¥5685/月mon)	¥12.02/hr (~¥8943/mon)

推荐配置

开发，测试，staging 环境， 或日均PV在10万以下的应用。

推荐配置：
MS1（~67元/月）

我的应用属于轻量级应用，日均PV在50万以下，数据库日常每秒钟查询率达到700左右。

推荐配置：
MS2（~134元/月）

适用场景：
企业官网，个人博客等

我的应用具备一定规模，日均PV达到100万，数据库日常每秒钟查询率达到4500左右。

推荐配置：
MS4（~1071元/月）

适用场景：
在线社区，营销网站，微信公众号等

我的应用有高并发需求，数据库规模可能超过100GB, 数据库日常每秒钟查询率达到10000左右。

推荐配置：
MS6（~2410元/月）

适用场景：
微信推广活动，抢红包，手游等

建议搭配只读实例进行读写分离。

典型应用场景和案例

MySQL广泛应用的场景

网站
公司官网，营销网站，博客，论坛，wiki，CMS， 在线教育，等等

微信平台应用
产品公众号 +信息推送+客户管理，等等

数字营销
抢红包，微信抽奖，等等

IOT
存储采集数据/分析后的结构数据，等等

移动应用
用户管理，数据属性管理，内容推送，用户行为日志，等等

电商
用户管理，订单管理，...

企业应用
BI报表，CRM，ERP, 监控系统（e.g. Zabbix), 等等

手游
用户信息，装备，日志，等等

光明网用户调查网站（光明云投平台）

- **场景**
  光明网创建问卷调查让用户投票。

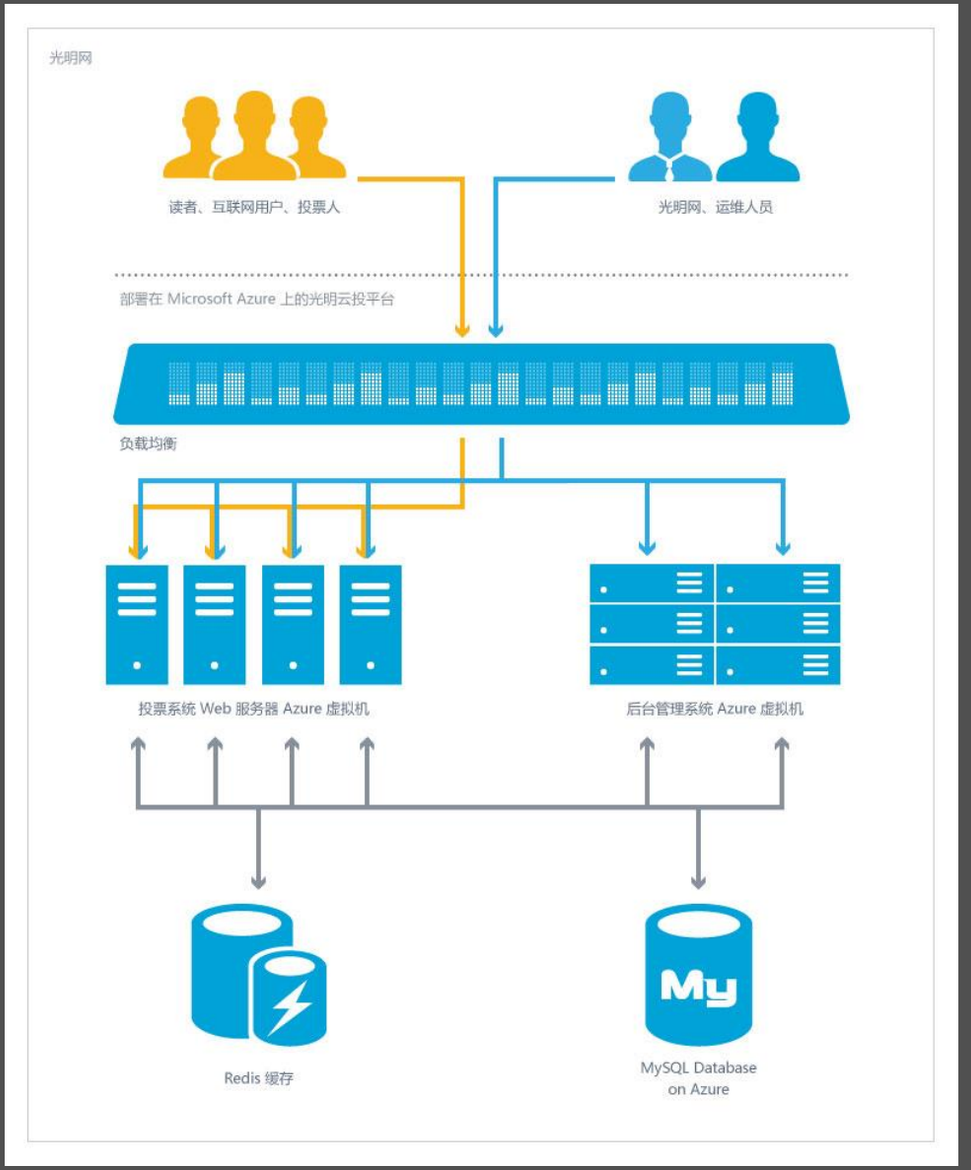
  并支持让用户可以利用此投票系统自行创建问卷。

- **实现方案**
  使用基于 Linux 的虚拟机服务、MySQL Database on Azure 数据库服务（PaaS）以及 PHP 语言、Redis 构建网络投票平台。

  其中Redis中存放用户投票数据，MySQL Database on Azure 数据库中存储清洗过后有效票数据。

  运维人员可以通过 Azure 管理门户内置的监控特性实时了解资源用量，根据负载饱和度来随时对服务器进行缩放，保证投票系统的正常运行。

- **链接**: https://www.azure.cn/partnerancasestudy/case-studies/guangmingwang/



移动应用案例-云图微动

- **场景**

玩图、拼立得、画中画相机、美丽拍、自拍相机，口袋应用等移动产品。（长期占据摄影类APP榜单前十位，产品总安装量过5亿，月活跃用户超过1亿。）

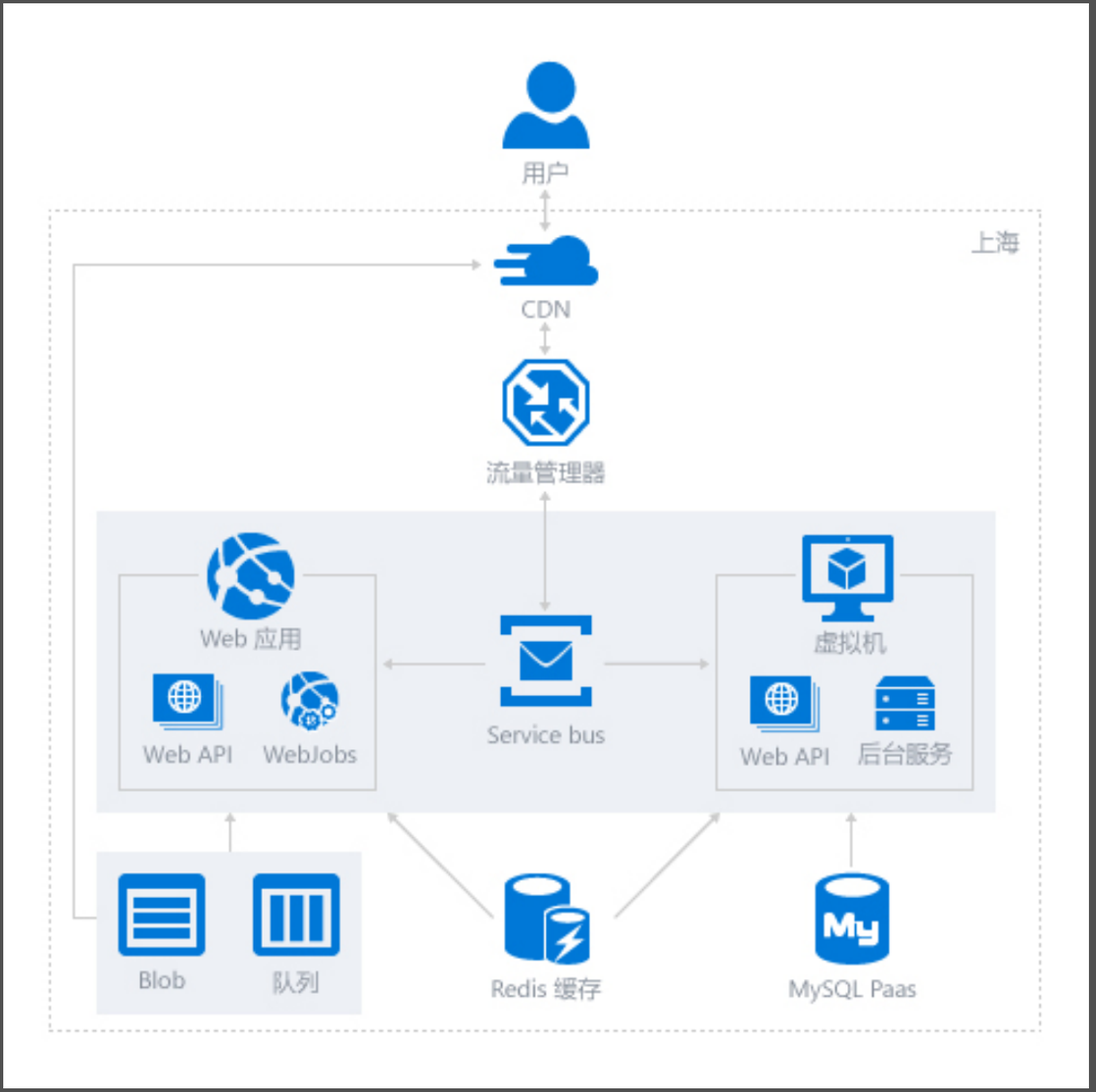
- **实现方案**

采用Web app应用以及虚拟机搭建前端服务器，将大量图片和视频存储在Azure存储中，并用CDN加速。将用户数据，日志，meta data等数据存在MySQL Database on Azure中，并利用Redis做缓存。

基于MySQL Database on Azure给用户带来的良好体验，用户逐步从IaaS服务向PaaS服务进行迁移。

- **链接**:

https://www.azure.cn/partnerancasestudy/case-studies/yuntu/



观致汽车connected cars

- **场景**

观致汽车内置一台 3G 联网的 8 英寸触控设备，Azure 将来自高德、新浪微博等互联网合作伙伴的信息，和行车讯息以及车辆的健康状态通过 3G 网络推送到车载设备和其他移动设备如手机上。

- **实现方案**

QorosCloud CRM系统将行车讯息和阈值规则写入 MySQL Database on Azure 数据库。

后台服务检查当前行车讯息是否超过设定的阈值，超过即向车主用多种方式推送消息。

MySQL Database on Azure 大大减少了 Qoros 自己搭建高可用DB 的运维成本。

- **链接**: https://www.azure.cn/partnerancasestudy/case-studies/qorosqloud/

风向标IoT后台服务

- **场景**

智能家居物联网在云端的后台服务，包括数据采集，数据存储，大数据分析，机器学习等等。

- **实现方案**

智能家居设备信息，状态，用户信息等通过前端的web服务器上传到Azure。

设备信息和用户信息存放在MySQL Database on Azure 的数据库中，设备的状态更新存放在其他storage中。

MySQL Database on Azure 的数据库中的数据会导入到HDInsight以用于大数据分析和机器学习。

- **链接**: https://www.azure.cn/partnerancasestudy/case-studies/fengxiangbiao/

最佳实践

提升应用的容错性

- 增加数据库重连逻辑
- 采用只读实例
- 采用cache(比如Redis)
- 把写请求暂时放在persistent的队列里
- Promote只读实例

良好的数据库设计

- 一定要有主键！！！
- 主键设计的不好性能可能差10倍
  - 主键建议用自增主键
  - 如果不是也要用数值类型的，以递增顺序插入
  - 不能用CHAR或者UUID等类型做主键
- 建索引很重要
  - 避免扫描全表
  - InnoDB的行锁是锁索引的，没有索引会导致锁表

如何得到稳定高效的性能

- 采用连接池或Persistent Connection
- 设置连接池自动检查连接的有效性
- 连接数不是越高越好，是够了就好
- 不同应用的DB放在不同的实例上，减少互相影响
- 建议新写的应用考虑用MySQL5.7
- 适当提高MySQL5.7里的参数binlog_group_commit_sync_delay（比如1000）对高并发下写的性能会有明显提升

连接池的配置示例

```
public class SimpleTestOnBorrowExample {
    public static void main(String[] args) throws Exception {
        PoolProperties p = new PoolProperties();
        p.setUrl("jdbc:mysql://localhost:3306/mysql");
        p.setDriverClassName("com.mysql.jdbc.Driver");
        p.setUsername("root");
        p.setPassword("password");
           // The indication of whether objects will be validated by the idle object evictor (if any).
           // If an object fails to validate, it will be dropped from the pool.
        p.setTestOnBorrow(true);
           // The SQL query that will be used to validate connections from this pool before returning them to the caller.
           // If specified, this query does not have to return any data, it just can't throw a SQLException.
        p.setValidationQuery("SELECT 1");
           // The timeout in seconds before a connection validation queries fail.
           // This works by calling java.sql.Statement.setQueryTimeout(seconds) on the statement that executes the validationQuery.
        p.setValidationQueryTimeout(1);
           // set other usefull pool properties.
        DataSource datasource = new DataSource();
        datasource.setPoolProperties(p);
         Connection con = null;
        try {
           con = datasource.getConnection();
           // execute your query here
        } finally {
         if (con!=null) try {con.close();}catch (Exception ignore) {}
        }
    }
}
```

应用设计具备良好的弹性扩展性

- 应用支持读写分离

```
ReplicationDriver driver = new ReplicationDriver();
String url = "jdbc:mysql:replication://address=(protocol=tcp)(type=master)(host=masterhost)(port=3306)(user=masteruser),address=(protocol=tcp)(type=slave)(host=slavehost)(port=3306)(user=slaveuser)/yourdb";
Properties props = new Properties();
props.put("password", "yourpassword");
try (Connection conn = driver.connect(url, props))
{
    // Perform read/write work on the master
    conn.setReadOnly(false);
    conn.createStatement().executeUpdate("update t1 set id = id+1;");

    // Set up connection to slave;
    conn.setReadOnly(true);
    try (Statement statement = conn.createStatement())
    {
        ResultSet res = statement.executeQuery("show tables");
        while (res.next()) {
            String tblName = res.getString(1);
            System.out.println(tblName);
        }
    }
}
```

- 对大容量高并发写的DB可以考虑在应用层做分库分表（sharding）
  - 在MySQL PaaS上运行1个和10个的运维成本是差不多的。

Thanks.