Senparc 盛派®

盛派网络

微信公众号+小程序快速开发

2018 年 1 月

Senparc 盛派®

第7节：缓存策略及并发场景下的分布式锁

所需课时：1

延伸参考：《微信开发深度解析》第8-9章

Senparc 盛派®

学习目标：

1、通过对 SDK 缓存策略的学习，了解本地缓存、分布式缓存的运行机制，掌握在不同缓存环境下的数据处理；
2、掌握分布式锁的运用场景和用法。

SDK 源代码：
https://github.com/JeffreySu/WeiXinMPSDK

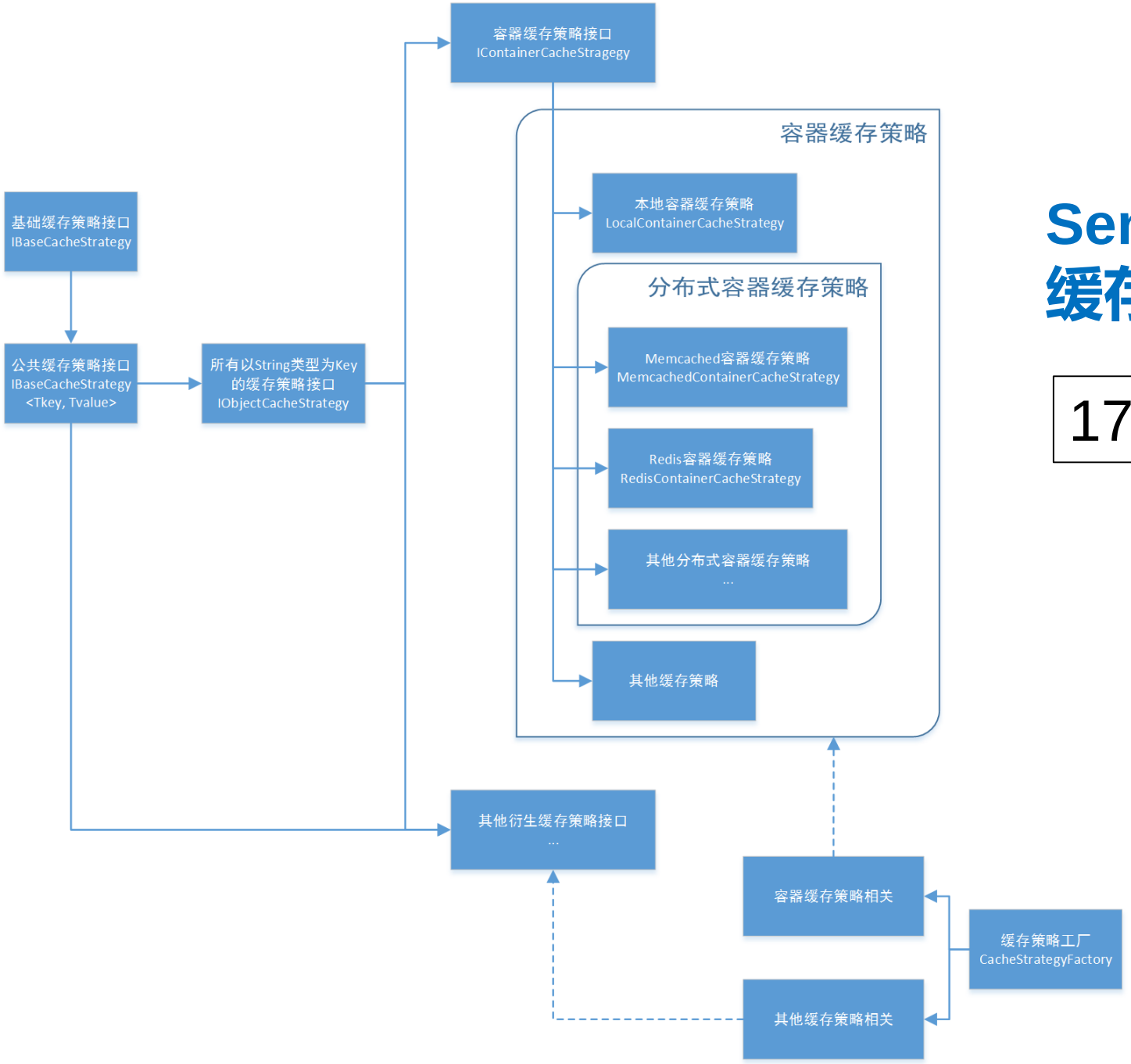
参考 Sample 源代码：
Senparc.Weixin.MP.Sample

课堂案例源代码下载：
https://github.com/JeffreySu/WechatVideoCourse

Senparc 盛派®

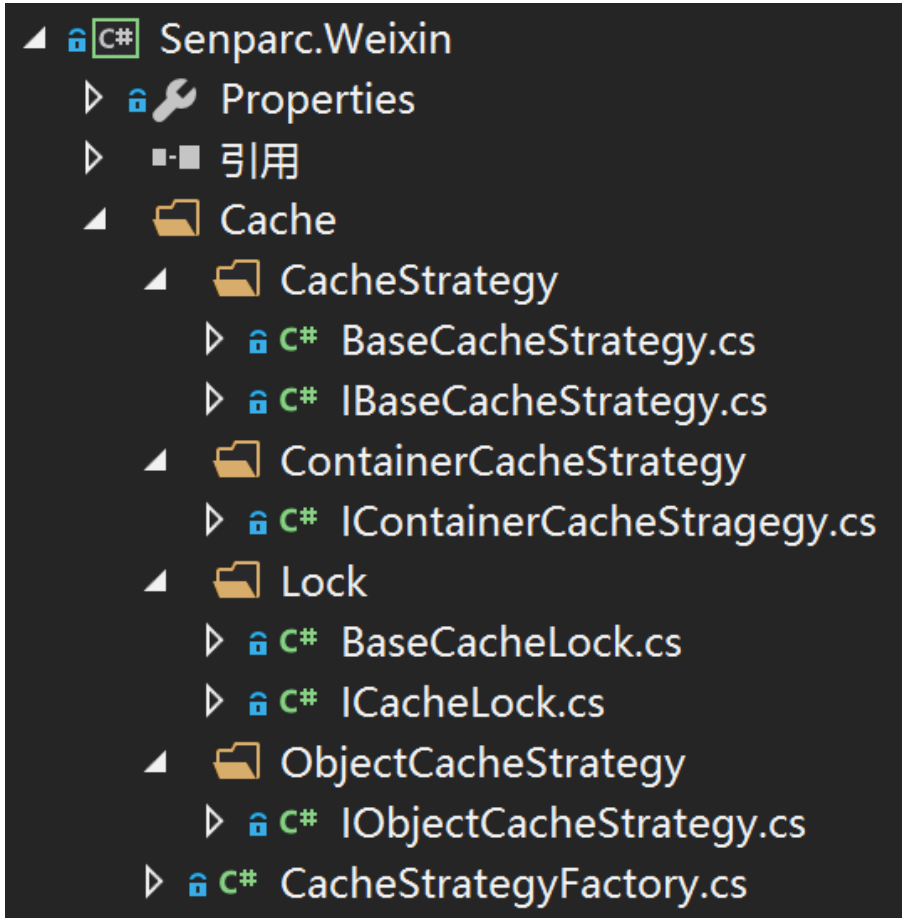
Senparc.Weixin SDK 缓存策略架构

175#356



Senparc 盛派®

核心缓存策略代码



Senparc 盛派®

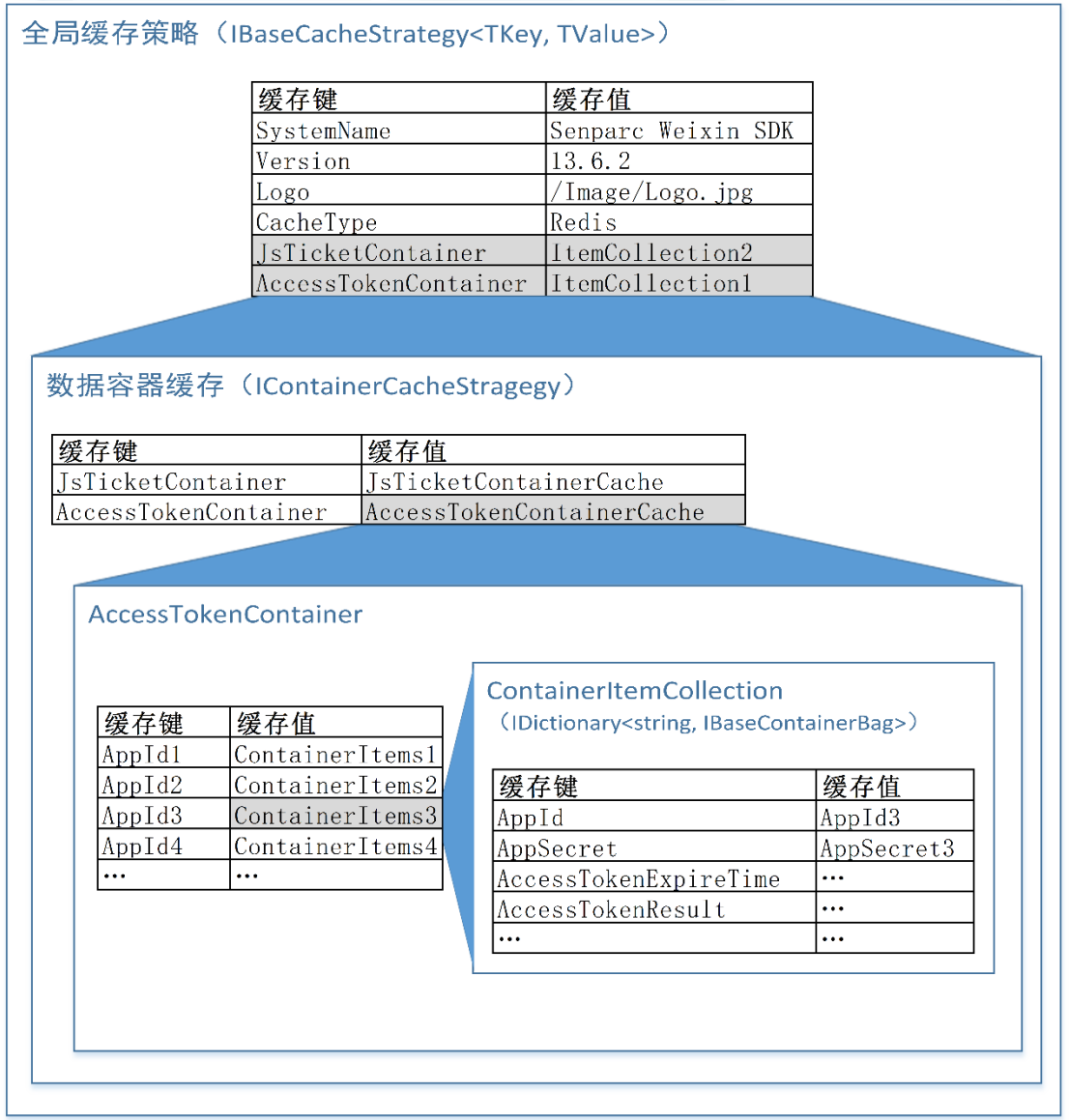
基础缓存策略

176#439

Senparc 盛派®

数据容器缓存策略接口：IContainerCacheStrategy

填充缓存数据之后的缓存结构和状态



180#358

缓存策略扩展

1. 本地缓存
2. 分布式缓存
   ① Redis 缓存
   ② Memcached 缓存
   ③ ……

定义数据源

介质	方案类型	优点	缺点
硬盘	**数据库** 如： SQL Server MySQL SQLite Hadoop Mongodb	1.可持久化储存 2.扩展方便 3.支持多维查询 4.支持切片查询	1.读写速度慢 2.可能造成数据冗余
	文本 如： TXT XML CSV	1.可持久化储存 2.支持查询 3.存储、备份方便	1.读写速度慢 2.不益处理过多数据 3.不益处理太复杂的查询条件 4.属性一旦确定，不太容易扩展
	其他文件 如： 序列化后的数据文件	1.可持久化储存实体数据 2.比文本储存略安全	1.读写速度慢 2.反序列化效率较低 3.不太容易实现高效的查询
内存	**系统缓存** 如： System.Web.Caching.Cache	1.读写速度快 2..NET 框架集成，比较完善 3.性能比较优秀 4.不支持泛型 5.容易扩展	1.通常不依赖持久化储存方案，数据容易丢失 2.遍历效率略低 3.可控性略低 4.和系统其他缓存公用
	静态变量 如： IDictionary<TKey,TValue>	1.读写速度快 2.轻巧 1.可控性高 2.支持泛型 3.容易扩展 4.独立于系统缓存	1.无法使用持久化方案，数据容易丢失 1.构造简单，默认状态下没有针对非常庞大缓存的处理方案 2.遍历效率略低

对于常规的单机环境，假设我们对容器缓存追求的指标依次为：

1) 安全性
2) 读写效率
3) 运行稳定性、抗干扰性
4) 可控性
5) 持久化（可选）

结合上述的假设，我们可以认为 **IDictionary** 可能是最好的选择。

Senparc 盛派®

缓存策略中的单例模式

Senparc 盛派®

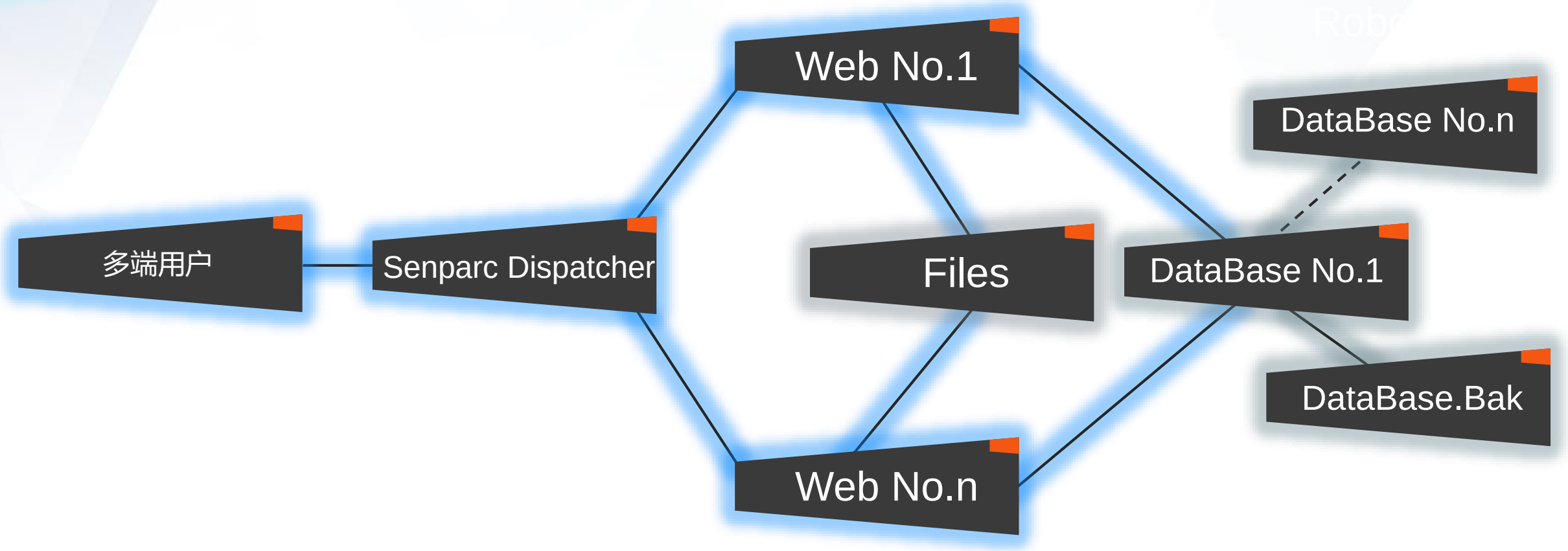
缓存测试

Senparc 盛派®

负载均衡（Load Balance）

Senparc 盛派®

负载均衡服务器架构



分布式使用的注意点

1.内存位置不同
2.响应速度不同
3.连接方式不同

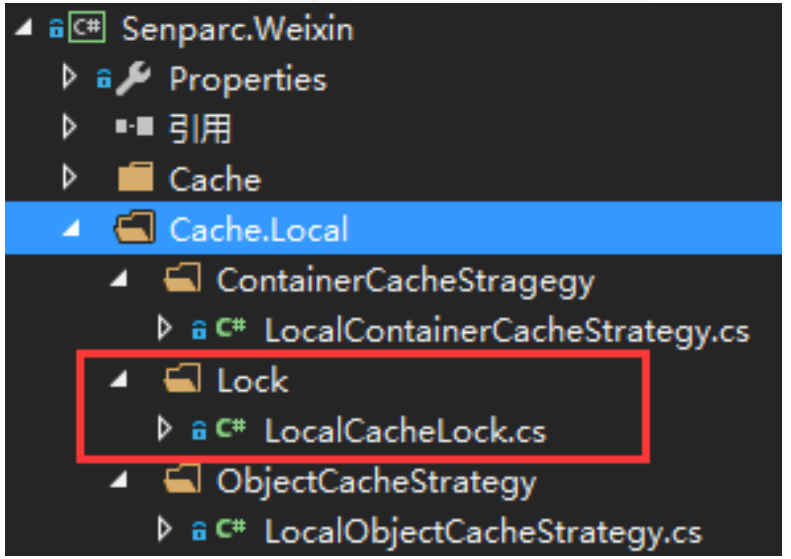
Senparc 盛派®

分布式缓存：Redis

Senparc 盛派®

并发场景下的分布式缓存锁

并发场景下的分布式缓存锁




```
//实际上strategy在LocalCacheLock内部暂时没有用到，
//这里给一个实例是因为还有一个基类，需要微程序提供良好的弹性
var strategy =
CacheStrategyFactory.GetObjectCacheStrategyInstance();

using (new LocalCacheLock(strategy, "Test", "LocalCacheLock"))
{
    //操作公共资源
}
```