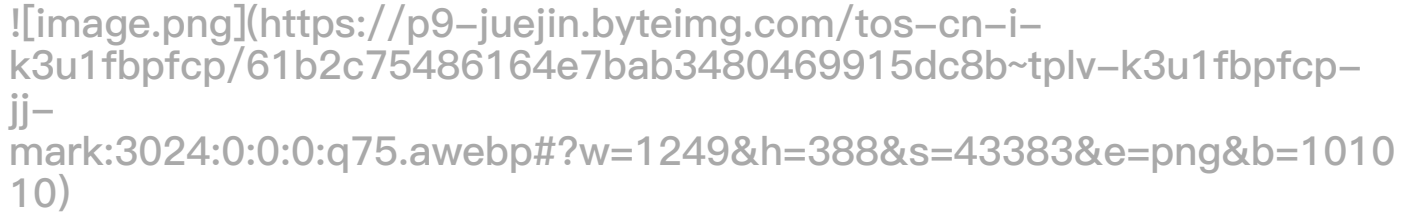
```
conds} ms");

    // Test with Garnet
    var garnetStartTime = DateTime.Now;
    for( int i = 0; i < 1000; i++ ) {
        garnet.StringSet(key + i, value + i, null);
    }
    var garnetEndTime = DateTime.Now;
    Console.WriteLine($"StringSet Time taken by Garnet: {(garnetEndTime
- garnetStartTime).TotalMilliseconds} ms");

    // Test with Redis
    redisStartTime = DateTime.Now;
    for( int i = 0; i < 1000; i++ ) {
        string a = redisDb.StringGet(key + i).ToString();
    }
    redisEndTime = DateTime.Now;
    Console.WriteLine($"StringGet Time taken by Redis: {(redisEndTime -
redisStartTime).TotalMilliseconds} ms");

    // Test with Garnet
    garnetStartTime = DateTime.Now;
    for( int i = 0; i < 1000; i++ ) {
        garnet.StringGet(key + i, null);
    }
    garnetEndTime = DateTime.Now;
    Console.WriteLine($"StringGet Time taken by Garnet: {(garnetEndTime
- garnetStartTime).TotalMilliseconds} ms");
}
```

最后运行的结果还是挺振奋人心的!



兼容性
===

当然在进行中间件选择的时候，切换的成本也是重点要纳入考虑的。“白嫖一时爽，重构火葬场！”这种事情显然是我们不愿意看到的。

这个时候我们在官网看到这样一句话

> Garnet 并不是要成为 Redis 100% 完美的替代品，而是应该将其视为一个足够接近的起点，以确保对您重要的功能的兼容性。 Garnet 确实可以在未经修改的情况下与许多 Redis 客户端一起使用（我们特别对 Garnet 进行了StackExchange.Redis很好的测试），因此入门非常容易。

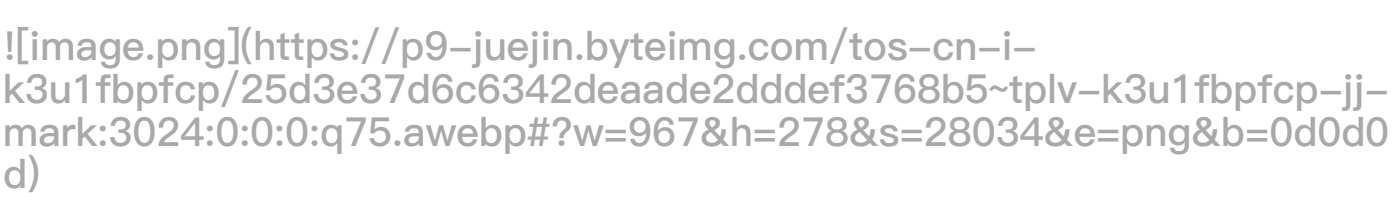
于是，我们再整个demo

```
[HttpGet]
[ActionTitle(Name = ”测试Garnet连接”)]
[Route(”connect.svc”)]
public void Connect()
{
    // Redis connection
    var redis = ConnectionMultiplexer.Connect(”localhost”);
    var redisDb = redis.GetDatabase();

    // Garnet connection
    var garne wo keys (non-transactionally) and return their values as an array of bulk strings
    /// </summary>
    public override void Finalize<TGarnetApi>(TGarnetApi api, ArgSlice input, ref MemoryResult<byte> output)
    {
        WriteSimpleString(ref output, ”Hello Garnet!”);
    }
}
```

同时还是因为Garnet遵循 RESP，因此客户端不是专用于 C#，也不是 Garnet 客户端独有的。 所以我们可以直接在redis客户端调用Garnet的自定义命令

```
 [HttpGet]
 [ActionTitle(Name = "测试Garnet 自定义命令")]
 [Route("custom-command.svc")]
 public async void Test()
 {
     // Garnet connection
     var garnet = new GarnetClient("localhost", 3278);
     garnet.Connect();
     // Test data
     string result = await garnet.ExecuteForStringResultAsync("TEST");

     Console.WriteLine($"Garnet Custom Command Result: {result}");

     // 采用redis客户端连接
     var client = ConnectionMultiplexer.Connect("localhost:3278");
     var db = client.GetDatabase();

     RedisResult redisResult = db.Execute("TEST");

     Console.WriteLine($"Redis Client Custom Command Result:
{((string?)redisResult)}");
 }
```

最后我们可以看到用两种客户端执行自定义命令都输出了同样的结果



结语

Garnet作为微软最新推出的远程缓存存储系统，为开发者提供了一种全新的选择。它具有高性能、可靠性和可伸缩性的特性，适用于各种规模和类型的应用场景。通过使用Garnet，开发者可以更好地提升应用的性能和用户体验，实现业务的快速发展和持续创新。

参考文档

* [Garnet 开发入门](http://cxyroad.com/
”https://microsoft.github.io/garnet/docs”)

更多一手讯息，可公众号：[ITProHub](http://cxyroad.com/
”https://myom-dev.oss-cn-
hangzhou.aliyuncs.com/WechatPublicPlatformQrCode.jpg”)

原文链接: https://juejin.cn/post/7356044171244159002