**Redis для .Net разработчиков.**



С 2014 года .net стал совсем другим (не тем, каким мы его знаем). Открытие части .net framework, новый компилятор C#, новый jit компилятор, .net native, активное использование нетипичных для windows технологий в Azure (из-за чего даже переименовали Windows Azure в Microsoft Azure), все большее движение Asp.net в сторону не только windows.

**На DevCon в 2012 году мы (я) с непониманием, слушал доклады по использованию Redis в .net приложениях. В нынешнем 2015 году не обращать внимания на Redis невозможно, даже живя на другой планете.**

Я для себя точкой смены вех вижу 7 октября 2014 года, когда Скотт Гаттри анонсировал общую доступность Redis Cache в Azure. <http://weblogs.asp.net/scottgu/azure-redis-cache-disaster-recovery-to-azure-tagging-support-elastic-scale-for-sqldb-docdb> (Слово Cache не надо воспринимать буквально как временное хранилище! Понимать надо в том смысле, что сам Redis называют кэшэм).

А дальше стали обнаруживаться места, где в платформе Microsoft торчит Redis:

А в Asp.Net5 (до декабря 2014 - vnext) появятся возможности:

* Команда Entity Framework анонсировала поддержку NoSQL хранилищ в Entity Framework 7 (правда в первом релизе 7.0 будет по старинке только SQL Server). Моя статья об этом <http://habrahabr.ru/post/111542/>
* хранить Session в Redis <http://msdn.microsoft.com/en-us/library/azure/dn690522.aspx> <http://blogs.msdn.com/b/webdev/archive/2014/05/12/announcing-asp-net-session-state-provider-for-redis-preview-release.aspx>
* И кэшировать данные в Redis <https://github.com/aspnet/Caching/tree/dev/src/Microsoft.Framework.Cache.Redis>
* Для SignalR можно использовать Redis в качестве scaleout backplane Моя статья <http://habrahabr.ru/post/108929/> <http://www.asp.net/signalr/overview/performance/scaleout-with-redis>

И последним ударом стало <http://msdn.microsoft.com/en-us/library/azure/dn766201.aspx> - **Microsoft теперь официально рекомендует использовать Redis для кэша.** Всю жизнь говорили про SQL Server (либо распределенный AppFabric Caching), а тут бац и Redis.

А когда-то давно, во времена .net 4.0- 4.5, был только Output cache <http://msdn.microsoft.com/en-us/library/azure/dn798898.aspx> и различные умельцы допиливали сами - один из авторов на хабре недавно писал про использование Redis вместе со своим CacheDependency <http://habrahabr.ru/post/240269/>

**Краткая справка про Redis**

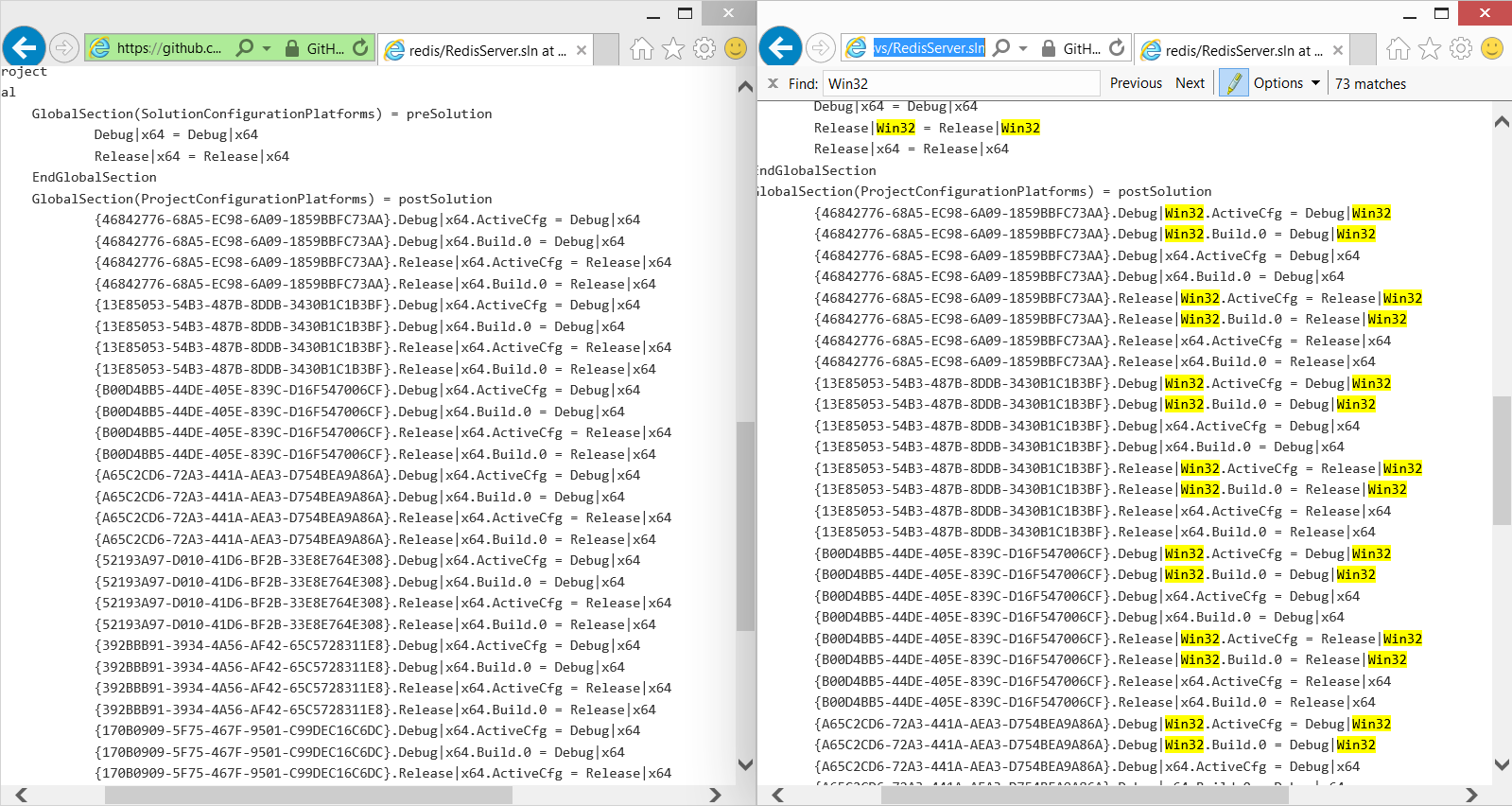
Изначально Redis писался под nix системы <https://github.com/antirez/redis> . Через какое-то время Microsoft занялся поддержкой Redis для Windows через Microsoft Open Technologies <https://github.com/MSOpenTech/redis> . И уже потом, от проекта под Windows, Azure форкнула свою версию <https://github.com/Azure/redis> .

Чтобы понимать разницу между проектами:

* antirez/redis – 4836 комитов
* MSOpenTech/redis – 4399 комитов
* Azure/redis – 3813 комитов

Команде Azure проще всего, т.к.

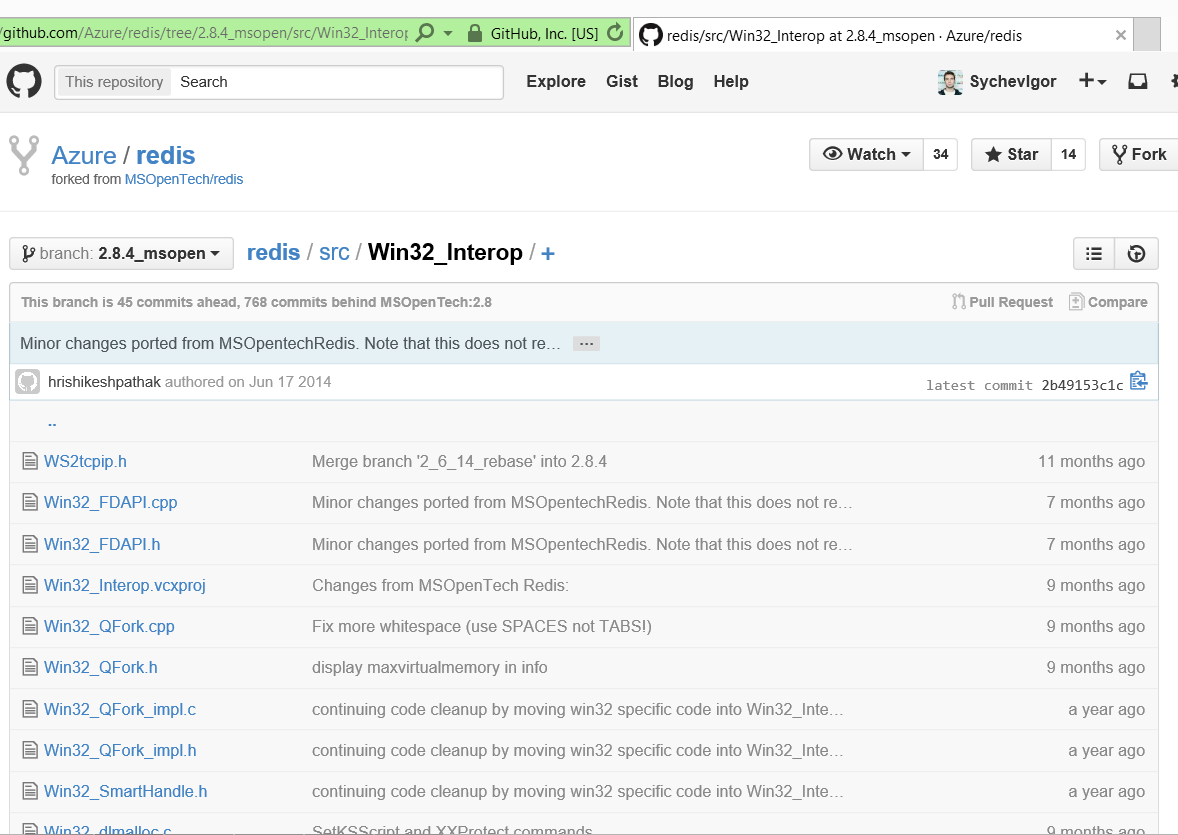
* они не тащат кучу старых веток типа 2.2, 2.4. У них есть prod версия 2.8 и на этом все.
* Они не используют 32-битную версию сборки - только 64.



* Им не нужны инструкции инсталляции, т.к. в Azure ты просто берешь инстанс, а не занимаешься сам его настройкой.

Основную работу по поддержке Windows версии несет команда MS Open Technologies.

Она пишет Win32\_Interop, для работы Redis под Windows, сохраняя все интерфейсы/header файлы для совместимости с не windows версией; при этом ответственность на себя за windows версию берет только в Azure варианте. На локальном сервере/кластере – это на ваш страх и риск.



**Часто задаваемые вопросы**

Коллеги, которые очень хорошо/плотно разбирались/ковырялись с SQL Server лет по 10, но вообще не ковырялись с Redis спрашивали:

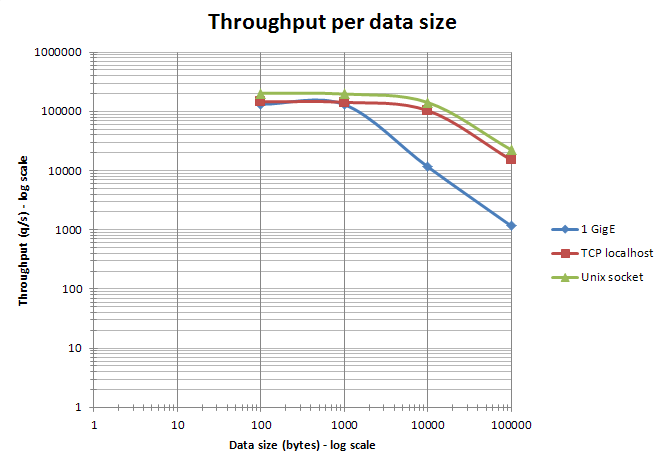
А насколько он быстрый? А выдержит ли он нашу нагрузку, причем с запасом, мы ведь все-таки enterprise, а не детская площадка?

Простой ответ - Redis очень быстрый. На бытовом уровне я бы объяснил так:

* Использовать данные в памяти сильно быстрее, чем сначала читать их с диска.
* Внутри себя Redis хранит данные в разных структурах данных в зависимости от типа. Быстрее чем поиск по hash таблице, тяжело что-то придумать (без углубления в вопросы распределения hash и качества hash функций) => если вы храните данные в ней, у вас все очень хорошо. Если храните данные в списках, то могут быть проблемы. <http://stackoverflow.com/questions/9625246/what-are-the-underlying-data-structures-used-for-redis?answertab=votes#tab-top>
* Т.к. в Redis нет такого сложного конвейера обработки запроса, как в SQL, накладные расходы ниже (план запроса, к примеру, надо составить, если его нет. Он составляется на основе статистики, статистика может быть некорректной и т.д., и т.п.).

В документации есть статья про производительность <http://redis.io/topics/benchmarks> (очень рекомендую потратить время и прочесть), сравнение производительности с memcache <http://oldblog.antirez.com/post/redis-memcached-benchmark.html> <http://dormando.livejournal.com/525147.html> <http://oldblog.antirez.com/post/update-on-memcached-redis-benchmark.html>

Если коротко, то 100 тысяч запросов в секунду на одном узле - это абсолютно нормальный результат, без какого-то фантастического железа или плясок с бубном при настройке.

Мне нравится вот этот график, 

который объясняет большую часть вещей, а именно: что при росте объема данных сеть становится узким местом, а не процессор или память. Сеть - это уже вопрос не к производительности самого Redis.

**Второй вопрос был: кластерную конфигурацию поддерживает? Мы ведь серьезную систему пишем!**

Ответ: да, причем из коробки, и кластер - это практически базовая конфигурация (design for cluster).

В этих статьях это описано подробно <http://redis.io/topics/cluster-spec> <http://redis.io/topics/cluster-tutorial>, и пусть Вас не смущает фразы про альфа/бета - тут как с gmail (которая была beta когда уже сотни миллионов ящиков на ней были по 5-6 лет в эксплуатации). Люди годами работают и не жалуются.

Кластеры поддерживают и распределение нагрузки, и отказоустойчивость. Master-узлы разделяют между собой нагрузку (диапазон ключей) + у каждого узла может быть любое число shard-ов (на которых хранится реплика мастера, готовая его заменить в случае недоступности.)

**А какие хорошие библиотеки доступа из .net есть?** Не самим же писать.

Года 2 назад, на devcon 2012, ребята показывали библиотеку, написанную на коленках за неделю. С тех пор мир изменился кардинально. Есть множество библиотек <http://redis.io/clients>, но для .net де факто стандартом стала <https://github.com/StackExchange/StackExchange.Redis> (те, кто использовал другие библиотеки в своих проектах, все чаще выпиливают и заменяют на эту версию).

Чтобы начать пользоваться этой библиотекой, долго вникать в нее не нужно. Принцип ее работы можно разобрать, открыв всего 3 класса.

* <https://github.com/StackExchange/StackExchange.Redis/blob/master/StackExchange.Redis/StackExchange/Redis/RedisCommand.cs>
* <https://github.com/StackExchange/StackExchange.Redis/blob/master/StackExchange.Redis/StackExchange/Redis/ConnectionMultiplexer.cs>
* <https://github.com/StackExchange/StackExchange.Redis/blob/master/StackExchange.Redis/StackExchange/Redis/RedisDatabase.cs>

**Если данные хранятся в памяти, то как долговременно хранить их. Сервер же перегружается иногда?!**

В определении что такое Redis есть слово, которое всех сбивает с толку – InMemory, но оно не означает In Memory Only. В Redis есть механизм сброса данных на диск, точнее 2 механизма. Инкрементальный - это когда каждые n секунд (можно сделать каждые 5, 30, 600 секунд) идет сброс данных, и Полный - это когда сбрасывается все содержимое на диск. (Как с бэкапами баз данных - инкрементальный и полный бэкап). Эти варианты друг другу не противоречат, можно оба включить. Это не бесплатно в плане производительности, как и любая запись на диск.

**Вместо заключения**

.Net сильно меняется, и придется меняться .net разработчикам. Использование Redis – это очередной шаг эволюции платформы, и его надо принять. Это не страшно.

**Ссылки**

* <http://redis.io>
* <http://msdn.microsoft.com/en-us/library/azure/dn690523.aspx>
* <http://azure.microsoft.com/en-us/services/cache/>