NCache™
Distributed Cache for .NET

Whitepaper

NCache vs Redis

Summary of Differences for .NET Applications

By

Iqbal Khan

February 8, 2019

Table of Content

Introduction

Today's ASP.NET / ASP.NET Core web applications, .NET Web Services, .NET Microservices, and any other .NET server applications need to handle extreme transaction loads. And, they need to use a distributed cache to achieve this goal because data storage and databases quickly become a bottleneck.

A distributed cache allows you to cache data and reduce those expensive database trips. And, a distributed cache scales linearly because it is distributed over multiple cache servers.

Over the last 13 years since 2005, NCache has provided such capabilities for .NET applications. NCache is a popular Open Source (Apache 2.0 License) distributed cache for .NET. NCache is 100% native .NET and allows you to cache application data as well as ASP.NET / ASP.NET Sessions, ASP.NET Core Response Cache, ASP.NET View State, and more.

Redis has been around since 2009 positioning itself as an Open Source (BSD License) key-value store with persistence. Redis has mainly been on the Linux platform and used by applications developed in PHP, Python, Java, and other similar languages.

Redis was unknown to the .NET community until 2012 when Microsoft decided to use it in Azure when they decided to discontinue their own offering of AppFabric. Microsoft chose the Redis option because this product supported a variety of languages enabling other non-.NET applications

In this whitepaper, I am going to mainly focus on areas where Redis and NCache are different. You can read a detailed featured comparison of Redis vs NCache from:

> NCache vs Redis – Detailed Feature Comparison

Difference 1: Native .NET Cache

NCache is .NET / .NET Core

NCache is 100% native .NET (and .NET Core). And, this fits very nicely with your .NET application stack. That is why, it is the favorite in .NET developer community and a .NET market leader for the last 13 Years.

As a result of this, NCache allows you to have .NET server-side code like Read-through/Write-through, Cache Loader, Custom Dependency, and Entry Processor.

NCache supports the following platforms:

- **Windows Server:** officially supported
- **Windows Nano Server:** officially supported
- **Linux (thru .NET Core):** officially supported

Read more about NCache On-Premises Deployment Options.

Redis is C++ / Linux

Redis is developed in C++ on Linux. And, it is not officially supported on Windows. Below are the Redis platform support options:

- **Linux:** supported by Redis Labs. Azure Redis Cache by Microsoft also runs on Linux.
- **Windows:**
  - Ported by Microsoft but usage without any support.
  - Redis Labs does not support Windows in production.

Difference 2: Cache Performance

NCache is an extremely fast and scalable distributed cache.

NCache has Client Cache (InProc Speed)

On top of this, NCache offers Client Cache (Near Cache) feature that is basically a local cache very close to your application (InProc/OutProc). And it keeps a subset of the cache either very close to your application (OutProc) or inside your application process (InProc) while keeping it synchronized with the entire cache cluster. So, if another user modifies the cache in the cluster, the Client Cache is automatically updated.

Client Cache gives your application InProc speed, meaning as fast as fetching objects from your application process memory/heap. This is extremely fast and no OutProc cache (like Redis) can match this.

Redis has No Client Cache (Remote Cache Speed)

Redis does not provide any Client Cache. As a result, you must always go to the caching tier across the network to fetch any data from the cache or update it.

And, this is no match to the InProc speed in NCache thru Client Cache.

Difference 3: High Availability

NCache has High Availability (Self-Healing)

NCache has a highly dynamic architecture and there are two things that sets it apart from Redis when it comes to high availability. They are:

1. Peer-to-Peer Clustering architecture
2. Partition-Replica Cache topology with dynamic cache partitioning

Peer-to-peer clustering means there is no Master/Slave concept and you can add or remove ANY cache server from the cluster without stopping or restricting anything. NCache automatically

adjusts the cluster membership when this happens. This is extremely useful because at runtime you don't know which server might go down but you want your cache to always be running.

Partition-Replica Cache topology is the second reason for high availability because it automatically adjusts cache partitioning whenever you add or remove a cache server from the cluster. This makes NCache self-healing which means NCache automatically adjusts itself to the "new reality" when a cache server is added or goes down.

Redis has Availability Issues

Unlike NCache's Peer-to-Peer dynamic clustering, Redis has a Master/Slave clustering architecture with shards that are rigid at runtime. For example, if a shard goes down or is removed at runtime, the cluster becomes unavailable until hash-slots of the removed shard are manually reassigned to remaining shards in the cluster. And, this manual intervention while cache is down is very painful and problematic.

Difference 4: Azure & AWS Cloud

NCache Officially Supports Cloud Platforms

NCache officially supports both Azure and AWS cloud platforms both from their marketplace and through NCache Cloud Portal for Azure & AWS. And, NCache in cloud automatically provides email and phone based technical support during office hours. And, you can optionally purchase our 24x7 support as well.

Following cloud options are provided by NCache:

- **Virtual Machine (Azure & AWS Marketplace)**
  - Free 60-day Trial (BYOL licensing)
  - Tech support included
  - 24x7 support provided (optional)
- **Managed Cache Service (NCache Portal for Azure & AWS)**
  - Free 60-day Trial (BYOL licensing)
  - Tech support included
  - 24x7 support included
  - 24x7 monitoring included

NCache Virtual Machines are available from Azure and AWS Marketplace. In this option, the caching infrastructure is managed and monitored by your own IT staff and they have full access to these VMs. And, you can contact NCache tech support for both regular support and optionally 24x7 support if purchased.

NCache Managed Cache Service is available from NCache Cloud Portal (at alachisoft.com website). This option comes with built-in regular support and 24x7 support. And, it also includes 24x7 monitoring provided by Alachisoft staff. Your IT staff can also have full access to the cache cluster VMs if you wish but you do not have to have this access.

Read more about NCache Cloud Deployment Options.

Redis has Mixture of Weak Cloud Support for .NET Apps

Unlike NCache that is supported fully by Alachisoft everywhere including 24x7 support, Redis comes in multiple flavors that are supported by different companies that are unrelated to each other. And, most of the places, Redis Open Source is being supported by companies that do not own its source code and neither do they control it. These flavors of Redis are:

- **Azure Marketplace: Azure Redis Cache (Redis Open Source)**
  - Supported by Microsoft
  - Fewer features than NCache Enterprise
  - Source code not controlled by Microsoft
  - No bug fixes or feature enhancements provided with any SLA
  - No 24x7 support provided
  - Only Linux platform supported (no Windows support)
- **Azure & AWS Marketplaces: Redis (Redis Open Source)**
  - Supported by Bitnami and Jetware
  - Fewer features than NCache Enterprise
  - Source code not controlled by Bitnami and Jetware
  - No bug fixes or feature enhancements provided with any SLA
  - Only Linux platform supported (no Windows support)
- **AWS Marketplace: Redis Enterprise**
  - Supported by Redis Labs
  - Fewer features than NCache Enterprise
  - Bug fixes and feature enhancements with SLA.
  - Only Linux platform supported (no Windows support)

For mission-critical .NET applications, you need to make sure that the distributed cache you're integrating into your application is fully supported for such sensitive business needs. Otherwise, your business suffers when something goes wrong.

Difference 5: Keeping Cache Fresh

The most common use case for a distributed cache is to cache application data that also resides in the database and could be updated by other applications in the database. And, if this data becomes stale in the cache then you lose all confidence on the data integrity.

As a result, you end up restricting your caching to read-only data which is only 10% of the total data in most cases. And, because of this, your application does not benefit completely from a distributed cache and does not scale its performance fully.

See how NCache and Redis compare in this very important feature area.

NCache Provides Rich Features

NCache provides a rich set of features to let you make sure your cache is always fresh and you can confidently rely on its data integrity. As a result, you'll cache practically all data.

Here are some of the features NCache provides:

- **Expirations:** You can use Absolute Expirations and Sliding Expirations to remove older data from the cache and keep only fresh data in it.

- **Synchronize Cache with DB:** You can sync cache with SQL Server and Oracle DB thru database notifications. You can also use OLEDB for polling-based cache synchronization.
- **Sync Cache with Non-Relational:** You can monitor any custom data source thru a Custom Dependency feature of NCache and sync the cache with external data sources whenever data changes there.
- **Handle Relational Data:** You can create one-to-many and one-to-one data relationships in the cache between cached items and auto-invalidate an item from the cache when a related item is updated or removed. This helps keep only fresh data in the cache.

Redis Provides Very Limited Features

Redis does not have the ability to do most of what NCache does when it comes to keeping your cache fresh. It relies on your application to keep track of all this which makes your life difficult and, in many cases, it is not even possible.

Here is what Redis provides:

- **Expirations:** You can use Absolute Expirations. You have to implement Sliding Expiration yourself by changing Absolute Expiration every time the item is fetched.

Difference 6: Search Cache with SQL & LINQ

Once you have a distributed cache that allows you to keep it fresh, you end up caching a lot of data. And, the more data you cache, the more you need to be able to "search" it based on something other than keys.

NCache lets you search the cache based on object attributes, groups/subgroups, tags, and named tags. This makes it really easy for you to find the data you've stored in the cache.

Redis has no such support.

NCache SQL & LINQ Search

NCache provides a rich set of features to let you search the cache with SQL/LINQ. Here are some of the features NCache provides:

- **Groups/Subgroups:** You can assign group/subgroup to cached items and later find them based on this logical grouping.
- **Tags / Named Tags:** You can assign Tags and Named Tags to cached items and later find them based on this.
- **SQL & LINQ Queries:** You can search the cache with SQL queries and LINQ queries. And, you can include object attributes, Groups/Subgroups, Tags, and Named Tags as part of the search criteria.
- **Indexing for Speed:** NCache lets you created indexes on object attributes. NCache automatically creates indexes on Groups/subgroups, Tags, and Named Tags. As a result, your search queries are super-fast.

Redis Has No Support

Redis has no such support for SQL/LINQ searching.

Difference 7: Server-Side Code (in .NET)

A really powerful distributed cache allows you to deploy server-side code to run on the cache servers in the cluster. NCache provides strong support server-side code and also lets you develop it in .NET since NCache itself is a native .NET solution.

Redis has no such support.

NCache Server-Side Code

NCache is the only .NET solution to provide you server-side code capability. NCache provides a rich set of features to let you develop and deploy server-side code in .NET to run on the cache servers in the cluster. Here are some of the features NCache provides:

- **Read-through:** You can implement a Read-through handler and deploy it on cache servers. NCache then calls your custom Read-through handler to fetch data from your database if it is not found in the cache.
- **Read-through with Auto-Reload:** When you combine Read-through with auto-reload feature, NCache automatically reloads data from the database upon expiration or database synchronization
- **Write-through:** You can also implement Write-through to let the cache update your database for you when you update the cache. This simplifies your application by shifting persistence code into the caching tier.
- **Write-behind:** Write-behind uses the Write-through handler to update the database asynchronously so your application doesn't have to wait for the database to be updated. This improves your application performance.
- **Cache Loader:** You can implement a Cache Loader and deploy it to the cache cluster. NCache then calls it to pre-load the cache automatically in a distributed fashion from all the cache servers in the cluster.
- **Custom Dependency:** You can write and deploy Custom Dependency code to monitor your own data source for any updates. When an update occurs, NCache removes the corresponding cached items from the cache.
- **Entry Processor:** You can write and deploy Entry Processor that NCache calls on all the cache servers to process data in the cache close by. Entry Processor can run InProc to the cache server where cached data is available from within process.

Redis Has No Support

Redis has no such support for server-side code.

Difference 8: WAN Replication for Multi-Datacenter

More and more high traffic applications are now being deployed in multiple datacenters (Azure Regions or AWS Availability Zones) either for geographic affinity and load balancing or for high availability and disaster recovery.

When this happens, then just like your application database, your distributed cache needs to handle this distribution across the WAN.

NCache Provides WAN Replication

NCache provides a rich set of features for WAN Replication of the cache. Here are some of the features NCache provides:

- **Active-Passive:** One datacenter is Active and the other one is Passive. NCache handles all updates from Active to Passive asynchronously.
- **Active-Active:** Both datacenters are active and processing user transactions simultaneously. NCache handles two-way synchronization asynchronously and also does conflict resolution if the same data is updated on both sides.
- **Multi-Datacenter ASP.NET / ASP.NET Core Sessions:** In some cases, you have regional load balancing in multi-datacenter where you don't want to replicate all user sessions across datacenters as it is unnecessary. But you want to overflow users from one datacenter to another or bring one datacenter down and migrate all its users to the other datacenter seamlessly. NCache allows you to do this for ASP.NET and ASP.NET Core.

Redis Has Limited Support

Redis has the following limited support.

- **Azure Redis Cache:** No WAN replication provided.
- **Azure & AWS Redis (Bitnami & Jetware):** No WAN replication provided.
- **Redis Open Source (On-Premises):** No WAN replication provided.
- **Redis Enterprise (Redis Labs):** WAN replication provided for active-active. Runs on Linux.

Conclusion

Below are some useful links for NCache. Please download NCache Open Source for free and NCache Enterprise fully-working 60-day trial. And, contact our tech support for technical questions and our sales team for other questions.

NCache Details Edition Comparison Download NCache

NCache vs Redis – Detailed Feature Comparison