Alachisoft

Redis vs. NCache

Comparison

For .NET and Java Applications

Redis v3.0.7 vs. NCache 4.6

This document compares Redis and NCache. Read this comparison to:

- Understand Redis and NCache major feature differences
- See how Redis and NCache compare on qualitative aspects such as performance, scalability, high availability, data reliability, and administration.

Table of Content

Disclaimer

The comparison provided in this document is for the purpose of helping you get a better understanding of Redis versus NCache. Information obtained about Redis is from the freely available downloads, documents, and forums.

We did not conduct any scientific benchmarks for performance and scalability of Redis so our assessment about it may be different from yours. NCache benchmarks are already published on our website (http://www.alachisoft.com) for you to see.

Additionally, we have made a conscious effort to be objective, honest, and accurate in our assessments in this document. But, any information about Redis could be unintentionally incorrect or missing, and we do not take any responsibility for it.

Instead, we strongly recommend that you do your own comparison of NCache with Redis and arrive at your own conclusions. We also encourage you to do performance benchmarks of Redis and NCache both in your environment for the same purpose.

1 Executive Summary

This document compares NCache with Redis, and contrasts their significant differences. This comparison focuses on all the major areas that a good In-Memory Distributed Cache should provide.

Feature	Redis	NCache
Performance and Scalability		
- Cache Performance	*Please verify yourself*	Excellent
- Cache Scalability	*Please verify yourself*	Excellent
- Bulk Operations	Partial support	Full support
- Async Operations	Supported	Full support
- Compression	Not Supported	Full support
- Fast Compact Serialization	Not Supported	Full support
- Indexes	Not Supported	Full support
- Multiple NIC Binding	Not Supported	Full support
Cache Elasticity (High Availability)		
- Dynamic Cache Cluster	Partial support	Full support
- Peer to Peer Architecture	Supported	Full support
- Connection Failover	Partial support	Full support
- Dynamic Configuration	Partial support	Full support
- Multiple Clusters	Supported	Full support
- Named Caches	Supported	Full support
- Cluster Specific Events	Not Supported	Full support
- Split Brain	Not Supported	Partial support
Cache Topologies		
- Local Cache	Partial support	Full support
- Client Cache (Near Cache)	Not Supported	Full support
- Mirrored Cache	Supported	Full support
- Replicated Cache	Not Supported	Full support
- Partitioned Cache	Partial support	Full support
- Partitioned-Replica Cache	Partial support	Full support
- Partitioned Data Balancing	Not Supported	Full support
- Load Balancing	Supported	Full support
- Partitioned Data Affinity	Supported	No support
- Persistence	Supported	Partial support

Feature	Redis	NCache
Object Caching Features		
- Get, Add, Insert, Remove, Exists, Clear Cache	Supported	Full support
- Expirations	Supported	Full support
- Lock & Unlock	Supported	Full support
- Streaming API	Supported	Full support
- Transactions	Partial support	Partial support
- Data Portability	Not Supported	Supported
- Item Versioning	Not Supported	Full support
- Multiple Object Versions	Not Supported	Full support
WAN Replication		
- Active – Passive	Not Supported	Full support
- Active – Active	Not Supported	Full support
- Conflict Resolution	Not Supported	Full support
- De-duplication	Not Supported	Full support
- Data Security	Not Supported	Full support
Big Data Processing		
- Map-Reduce Query	Not Supported	Full support
- Aggregators	Not Supported	Full support
- Entry Processor	Not Supported	Full support
Cache Administration		
- Admin Tool (GUI)	Not Supported	Full support
- Monitoring Tool (GUI)	Not Supported	Full support
- PerfMon Counters	Not Supported	Full support
- JMX Counter	Not Supported	Full support
- Admin Tools (Command Line)	Supported	Full support
- Administration and Monitoring (API)	Supported	Full support
Security & Encryption		
- Authentication (Active Directory/LDAP)	Partial support	Full support
- Authorization	Supported	Full support
- Data Encryption	Not Supported	Full support
- Secure Communication	Partial support	Partial support
Managing Data Relationships		

Feature	Redis	NCache
- Key Based Cache Dependency	Not Supported	Full support
- Key Based Multi-Cache Dependency	Not Supported	Full support
Synchronization with Data Sources		
- Oracle Dependency (Oracle)	Not Supported	Full support
- Db Dependency	Not Supported	Full support
- File Dependency	Not Supported	Full support
- Custom Dependency	Not Supported	Full support
Runtime Data Sharing		
- Item Level Events (onInsert/onRemove)	Supported	Full support
- Cache Level Events (Add/Insert/Remove)	Supported	Full support
- Custom Events (Pub/Sub Fired by Apps)	Supported	Full support
Cache Search (SQL-Like)		
- SQL Search	Not Supported	Full support
- Continuous Query	Not Supported	Full support
- LINQ Queries	Not Supported	Full support
- SQL on Tags, Named Tags & Groups	Not Supported	Full support
Data Grouping		
- Groups/Subgroups	Not Supported	Full support
- Tags	Not Supported	Full support
- Named Tags	Not Supported	Full support
Read-through, Write-through & Cache Loader		
- Read-through	Not Supported	Full support
- Write-through & Write behind	Not Supported	Full support
- Auto Reload at Expiration & Database Synchronization	Not Supported	Full support
- Cache Startup Loader	Partial support	Full support
Cache Size Management (Evictions Policies)		
- Max Cache Size (in MBs)	Supported	Full support
- LRU Evictions (Least Recently Used)	Supported	Full support
- LFU Evictions (Least Frequently Used)	Not Supported	Full support
- Priority Evictions	Not Supported	Full support

Feature	Redis	NCache
- Do Not Evict Option	Supported	Full support
Session Persistence for Java Web Apps & ASP.NET		
- ASP.NET Session Caching (basic)	Supported	Full support
- ASP.NET Session Caching (advanced)	Not Supported	Full support
- ASP.NET Sessions (Multiple-site)	Not Supported	Full support
- ASP.NET View State Cache	Not Supported	Full support
- ASP.NET Output Cache	Supported	Full support
- Java Web Sessions	Partial support	Full support
- Java Web Sessions (Multi-site)	Not Supported	Full support
Third Party Integrations		
- NHibernate 2nd Level Cache	No Official Support	Full support
- Entity Framework 2nd Level Cache	Not Supported	Full support
- Memcached Protocol Server	Supported	Full support
- Memcached Smart Wrapper	Not Supported	Full support
- Hibernate 2nd Level Cache	No Official Support	Full support
- JCache API Support	Not Supported	Full support
- Spring Caching	Supported	Full support

2 Qualitative Differences Explained

2.1 Performance and Scalability

Performance is defined as how fast cache operations are performed at a normal transaction load. Scalability is defined as how fast the same cache operations are performed under higher and higher transaction loads. NCache is extremely fast and scalable.

See NCache benchmarks at Performance and Scalability Benchmarks.

Feature Area	Redis	NCache
Cache Performance	*Please verify it yourself* But, please note that Redis converts everything into string for storage. And, this can be very costly for images, binary data, or large objects.	Extremely good. NCache is extremely fast. Please see its performance benchmarks.
Cache Scalability	*Please verify it yourself*	Extremely good. NCache provides linear scalability, means as you add more nodes to the cluster your performance increases in a linear fashion. Please see its performance benchmarks.
Bulk Operations	Partial support Bulk operations are not distributed to all the nodes. Instead, they're all sent to one shard and all the keys must be present on that node for success.	Full support. Bulk Get, Add, Insert, and Remove. This covers most of the major cache operations and provides a great performance boost.
Async Operations	Supported	Full support. Async add, insert, and remove provided. Async operation returns control to the application and performs the cache operation in the background. Improves application response time greatly.
Compression	Not Supported	Full support. Specify this along with item size threshold and only items larger than the threshold are compressed. Rest are cached uncompressed.

Feature Area	Redis	NCache
		This is provided because compressing smaller items often slows things down. And, you can configure "compression" at runtime through "Hot Apply".
Fast Compact Serialization	Not Supported In fact, Redis stores all data as string. So, if you need to store images or objects, they get transformed into a string first and then stored. And, this is quite costly as compared to binary serialized objects that NCache stores.	Full support. Compact Serialization is extremely fast because it uses precompiled code to serialize and also because it stores type-ids instead of long type names in the serialized objects. This is almost 10 times faster. Once you register classes for Compact Serialization, NCache generates serialization code and compiles it in-memory all at runtime and uses this precompiled code for serialization. You can mix Compact Serialization with regular serialization on objects of your choice.
Indexes	Not Supported	Full support. NCache allows you to define indexes on object attributes. NCache then generates data extraction code for these indexes at connection time, compiles it in-memory, and uses it at client-side for all data extraction. This is much faster than using Reflection. NCache also creates indexes automatically on Tags, Named Tags, Groups, and Subgroups. Expiration and Eviction Policies.
Multiple NIC Binding	Not Supported You can bind multiple IPs but cannot define which to use for cluster replication. Multiple binding is only for accessing Redis through	Full support. You can assign two NICs to a cache server. One can be used for clients to talk to the cache server and second for

Feature Area	Redis	NCache
	different networks.	multiple cache servers in the cluster to talk to each other. This improves your bandwidth scalability greatly. You can also assign a specific NIC for a cache client to use for talking to the cache server.

2.2 Cache Elasticity (High Availability)

Cache elasticity means how flexible the cache is at runtime. Are you able to perform the following operations at runtime without stopping the cache or your application?

1. Add or remove cache servers at runtime without stopping the cache.
2. Make cache config changes without stopping the cache
3. Add or remove web/application servers without stopping the cache
4. Have failover support in case any server goes down (meaning are cache clients are able to continue working seamlessly).

This is an area where Redis is relatively weak. In fact, it doesn't provide support for some of these things. But, NCache is known for its strength in this area.

NCache provides a self-healing dynamic cache clustering that makes NCache highly elastic. Read more about it at Self-Healing Dynamic Clustering.

Feature Area	Redis	NCache
Dynamic Cache Cluster	Partial support Redis clusters with shards are rigid. To add a new Master/Slave pair, manual intervention is required to re-balance the data. Scaling out requires a fair amount of human intervention. If a shard goes down or is removed, the cluster becomes unavailable until hash-slots are reassigned to existing shards.	Full support. NCache is highly dynamic and lets you add or remove cache servers at runtime without any interruption to the cache or your application Data is automatically rebalanced at runtime (called state transfer) and without any interruption to the cache or its clients. There is also no performance degradation.
Peer to Peer Architecture	Supported	Full support. NCache cache cluster has a peer to peer

Feature Area	Redis	NCache
		architecture. This means there is no "master/slave" and no "majority rule" in the cluster. All nodes are equal. There is a "coordinator" node that is the senior most node. If it goes down, next senior most node takes over this role automatically.
Connection Failover	Partial support In Redis, if a shard with no replicas goes down, the entire cluster is halted and blocks any client requests.	Full support. NCache provides full connection failover support between cache clients and servers and also within the cache cluster. In case of cache server failure, NCache clients continue working with other servers in the cluster and without any interruption. Cluster auto-manages itself by rebalancing its data and recreating replicas where needed.
Dynamic Configuration	Partial support Although, Redis provides CONFIG SET command, you have to separately run it against every server whereas in NCache, you can do this against the entire cluster at once.	Full support. NCache cluster configuration is not hard coded and when you add or drop servers at runtime, all other servers in the cluster are made aware of it. NCache clients also learn about all the servers and a variety of other configuration at runtime from the cache cluster. Also, 'Hot Apply' feature allows you to change a lof of the configuration at runtime without stopping anything.
Multiple Clusters	Supported	Full support. NCache allows you to create multiple cache clusters of either the same or different topologies on the same set of cache servers.

Feature Area	Redis	NCache
Named Caches	Supported	Full support. NCache allows you to create multiple named caches on the same set of cache servers.
Cluster Specific Events	Not Supported Cluster Events are not supported. Redis only allows you to manually query about the cluster health.	Full support. NCache provides events about changes in the cluster like: MemberJoined, MemberLeft, CacheStopped, etc. These events can be delivered to both .NET and Java applications natively.
Split Brain	Not Supported	Partial support Split brain detection is provided and you're notified through NCache events when that happens. But no auto recovery is provided.

2.3 Cache Topologies

Cache Topologies determine data storage and client connection strategy. There are different topologies for different type of uses.

Read more details on NCache caching topologies at In-Memory Distributed Cache Topologies.

NOTE:

Feature Area	Redis	NCache
Local Cache	Partial support Only OutProc cache is supported	Full support. Both InProc and OutProc. InProc is much faster but your memory consumption is higher if you have multiple instances on the same machine. OutProc is slightly slower due to IPC and serialization cost but saves you memory consumption because there is only one

Feature Area	Redis	NCache
		copy per machine.
Client Cache (Near Cache)	Not Supported	Full support. Client Cache is simply a Local Cache (InProc/OutProc) on NCache client machine. But, it is also connected to and synchronized with the cache cluster. So, NCache clients get the benefits of a super-fast Local Cache without compromising on data integrity.
Mirrored Cache	Supported	Full support. Mirrored Cache is a 2-node Active-Passive cache. All clients connect to the Active node and data mirroring is done asynchronously. In case Active node goes down, Passive node automatically becomes Active and all clients connect to it automatically.
Replicated Cache	Not Supported	Full support. In Replicated Cache, the entire cache is replicated on all nodes in the cluster. You can have more than 2 nodes and all nodes are active meaning clients connect to them directly. Updates are done synchronously within the cluster and are therefore slower than other topologies. But, reads are super-fast. Each client connects to only one node. You can enable load-balancing or specify an ordered server list for the clients to use.
Partitioned Cache	Partial support If one shard or hash-slot becomes un-available, the whole cluster is halted and blocks any client requests. To resume, either manual intervention is	Full support. Entire cache is partitioned and each cache cluster node gets one partition. All partitions are created at runtime when you add/remove nodes.

Feature Area	Redis	NCache
	required to re-distribute hash-slots or the shard has to become available.	Each client is connected to all cache nodes. This allows it to directly go where the data is (single hop). Data balancing feature provided in case one partition gets overwhelmed with data. Partitioned Cache is extremely scalable. You grow both storage and transaction capacity as you add servers to the cluster.
Partitioned-Replica Cache	Partial support Only asynchronous replication is supported between master and slave nodes.	Full support. Same as Partitioned Cache (read above). Also provides a replica for each partition kept at another cache server. This provides reliability against data loss if a node goes down. Async and Sync, both replications are supported. Just like partitions, replicas are also created dynamically. Data balancing also updates replicas.
Partitioned Data Balancing	Not Supported	Full support. Data is automatically rebalanced when you add/remove cache servers from the cluster. Data is also rebalanced automatically when one cache server has a lot more data than other servers. You can configure the threshold of difference for this. You can turn off auto rebalancing in this case and manually do it if you wish. This applies to both Partitioned Cache and Partition-Replica Cache.
Load Balancing	Supported	Full support. Clients are balanced among server nodes in case of Replicated Cache topology.

Feature Area	Redis	NCache
		For Partitioned Cache topologies clients are connected to all nodes for single hop operation and therefore balanced as well.
Partitioned Data Affinity	Supported	No support
Persistence	Supported	Partial support. NCache allows you to persist data by using Read-thru/Write-thru and Write-behind. However, this requires programming on your part. The benefit is that you can pick your own storage which could be a database etc.

2.4 Object Caching Features

These are the most basic operations without which an In-Memory Distributed Cache becomes almost unusable. These by no means cover all the operations a good Cache should have.

Feature Area	Redis	NCache
Get, Add, Insert, Remove, Exists, Clear Cache	Supported	Full support. NCache provides more variations of these operations and therefore more control to the user.
Expirations	Supported	Full support. Absolute and Sliding expirations provided. Absolute expiration is good for data that is coming from the database and must be expired after a known time because it might become stale. Sliding expiration means expire after a period of inactivity and is good for session and other temporary data that must be removed once it is no longer needed.
Lock & Unlock	Supported	Full support.

Feature Area	Redis	NCache
		NCache provides both. Lock is used to exclusively lock a cached item so nobody else can read or write it. This item stays locked until either the lock expires or it is unlocked. NCache also provides "GetAndLock()", that locks the item before fetching it, and "InsertAndUnlock()" that updates the item and then unlocks it, all in one call.
Streaming API	Supported	Full support. For large objects, NCache allows the cache clients to fetch them in "GetChunk()" manner and update them in "AppendChunk()" manner. With this, NCache clients can stream in or out large objects from the cache.
Transactions	Partial support. Redis neither provides rollbacks nor supports the "all or nothing" proposition in a multi-command transaction.	Partial support. Explicit locking Implicit locking (item versioning) Entry Processor (are atomic)
Data Portability	Not Supported Redis does not provide any support for transforming C# objects into Java or vice versa. You have to do this yourself.	Supported. .NET to Java and Java to .NET object conversion supported without going through JSON/XML transformation. Configurable using a user friendly GUI.
Item Versioning	Not Supported You have to manage it yourself	Full support. This ensures that only one client can update an item and all future updates will fail unless cache clients first fetch the latest version and then update it.
Multiple Object Versions	Not Supported You have to manage it yourself	Full support. NCache allows two different versions of the same class to be stored in the cache by different apps. Each app retrieves its own version and the cache keeps a superset.

Feature Area	Redis	NCache

2.5 WAN Replication

WAN replication is an important feature for many customers whose applications are deployed in multiple data centers either for disaster recovery purpose or for load balancing of regional traffic.

The idea behind WAN replication is that it must not slow down the cache in each geographical location due to the high latency of WAN. NCache provides Bridge Topology to handle all of this.

Read more about it at WAN Replication of In-Memory Cache.

NOTE: Synchronization of clusters is named as WAN (Wide Area Network) Replication because it is mainly used for replicating different clusters running on WAN (more). WAN replication topology is totally user configurable, user can create any topology for WAN replication e.g. Active-Passive, Active-Active etc…

Feature Area	Redis	NCache
Active – Passive	Not Supported	Full support. Bridge Topology Active-Passive You can create a Bridge between the Active and Passive sites. The Active site submits all updates to the Bridge which then replicates them to the Passive site.
Active – Active	Not Supported	Full support. Bridge Topology Active-Active You can create a Bridge between two active sites. Both submit their updates to the Bridge which handles conflicts on “last update wins” rule or through a custom conflict resolution handler provided by you. Then, the Bridge ensures that both sites have the same update.
Conflict Resolution	Not Supported	Full support. By default, “last update wins” algorithm is used to resolve conflicts. But, you can specify a “custom conflict resolution handler” that is called to resolve conflict by comparing the content of both objects

Feature Area	Redis	NCache
		and deciding.
De-duplication	Not Supported	Full support. NCache Bridge optimizes replication queue by de-duplicating items. If the same key is updated multiple times, it only replicates the most recent update.
Data Security	Not Supported	Full support. You can encrypt data with 3DES and AES algorithms before transportation. Otherwise, you can use a VPN between data centers for security.

2.6 Big Data Processing

For analysis and processing large amount of cached data, fetching data on client side becomes too time consuming and effects application's performance. Processing data on grid is the ability to move processing logic near data on grid nodes, instead of moving data to client.

NCache provides a self-healing dynamic cache clustering that makes NCache highly elastic. Read more about it at Self-Healing Dynamic Clustering.

Feature Area	Redis	NCache
Map-Reduce Query	Not Supported	Fully supported. NCache provides a MapReduce framework where you program can run on cache servers for parallel processing of Big Data.
Aggregators	Not Supported	Fully supported. NCache provides Aggregator that works with MapReduce framework and provides you statistical data.
Entry Processor	Not Supported No entry processor like capability provided even though a very basic LUA scripting engine allows scripts to be executed on	Fully supported. NCache fully supports Entry Processor execution on cache nodes in parallel.

Feature Area	Redis	NCache
	servers.	

2.7 Cache Administration

Cache administration is a very important aspect of any In-Memory Cache. A good cache should provide the following:

1. GUI based and command line tools for cache administration including cache creation and editing/updates.
2. GUI based tools for monitor the cache activities at runtime.
3. Cache statistics based on PerfMon (since for Windows PerfMon is the standard)

NCache provides powerful support in all these areas.

Read more about it at Administration and Monitoring Tools.

Feature Area	Redis	NCache
Admin Tool (GUI)	Not Supported	Full support NCache Manager is a powerful GUI tool for NCache. It gives you an explorer style view and lets you quickly administer the cache cluster from a single place. This includes cache creation/editing and many other functions.
Monitoring Tool (GUI)	Not Supported	Full support NCache Monitor is a powerful GUI tool that lets you monitor NCache cluster wide activity from a single location. It also lets you monitor all of NCache clients from a single location. And, you can incorporate non-NCache PerfMon counters in it for comparison with NCache stats. This real-time comparison is often very important.
PerfMon Counters	Not Supported	Full support. NCache provides a rich set of PerfMon counters that can be seen from NCache Manager, NCache Monitor, or any third party tool that supports PerfMon monitoring.

Feature Area	Redis	NCache
JMX Counter	Not Supported	Full support. NCache provides a rich set of JMX counters for its Java clients that can be seen from any third party tool that supports JMX monitoring.
Admin Tools (Command Line)	Supported	Full support. NCache provides a rich set of command line tools. You can create a cache, add remote clients to it, add server nodes to it, start/stop the cache, and much more. Use these tools from your scripts and automate various cache admin operations.
Admin & Monitoring (API)	Supported	Full support. NCache provides Java and .NET API to manage and monitor the caches & client. Using this API you can stop/start the cache, get the statistics of the connected clients or get the health info of the cache cluster.

2.8 Security & Encryption

Many applications deal with sensitive data or are mission critical and cannot allow the cache to be open to everybody. Therefore, a good In-Memory Distributed Caches provides restricted access based on authentication and authorization to classify people in different groups of users. And, it should also allow data to be encrypted inside the client application process before it travels to the Distributed Cache.

NCache provides strong support in all of these areas.

Feature Area	Redis	NCache
Authentication (Active Directory / LDAP)	Partial support No support for Active Directory or LDAP authentication. Redis provides only text based	Full support. You can authenticate users against Active Directory or LDAP. If security is enabled, nobody can access the cache without authentication and authorization.

Feature Area	Redis	NCache
	authentication where administrator keeps a list of users and passwords in a configuration file.	
Authorization	Supported	Full support. You can authorize users to be either "users" or "admins". Users can only access the cache for read-write operations while "admins" can administer the cache.
Data Encryption	Not Supported	Full support. You can enable encryption and NCache automatically encrypts all items one client-side before sending them to the cache. And, this data is kept encrypted while in the cache. And decryption also happens automatically and transparently inside the client process. Currently, 3DES and AES128 / AES196 / AES256 encryptions are provided and more are being added.
Secure Communication	Partial support Doesn't provide secured encryption, Redis is meant to be used within a trusted network without outside access. User must install their own protection mechanism to secure their data.	Partial support NCache does not provide SSL support for client/server communication. But you can configure a VPN and it works fine there. Additionally, strong encryptions are provided by NCache so you can encrypt data over an unsecured connection.

2.9 Managing Data Relationships

Since most data being cached comes from relational databases, it has relationships among various data items. So, a good cache should allow you to specify these relationships in the cache and then keep the data integrity. It should allow you to handle one-to-one, one-to-many, and many-to-many data relationships in the cache automatically without burdening your application with this task.

Feature Area	Redis	NCache
Key Based Cache Dependency	Not Supported	Full support. You can specify that cached item A depends cached item B which then depends on cached item C. Then, if C is ever updated or removed, B is automatically removed from the cache and that triggers the removal of A from the cache as well. And, all of this is done automatically by the cache. With this feature, you can keep track of one-to-one, one-to-many, and many-to-many relationships in the cache and invalidate cached items if their related items are updated or removed.
Key Based Multi-Cache Dependency	Not Supported	Full support. This is an extension of Key Based Cache Dependency except it allows you to create this dependency across multiple caches.

2.10 Synchronization with Data Sources

Database synchronization is a very important feature for any good In-Memory Distributed Cache. Since most data being cached is coming from a relational database, there are always situations where other applications or users might change the data and cause the cached data to become stale.

To handle these situations, a good In-Memory Distributed Cache should allow you to specify dependencies between cached items and data in the database. Then, whenever that data in the database changes, the cache becomes aware of it and either invalidates its data or reloads a new copy.

Additionally, a good Distributed Cache should allow you to synchronize the cache with non-relational data sources since real life is full of those situations as well.

NCache provides a very powerful database synchronization feature.

Feature Area	Redis	NCache
Oracle Dependency (Oracle)	Not Supported	Full support. NCache provides OracleDependency support for Oracle. You can associate a cached item with a SQL statement based

Feature Area	Redis	NCache
		dataset in Oracle database. Then whenever that dataset changes (addition, updates, or removal), Oracle sends a data notification to NCache and NCache invalidates this cached item or reloads it if you have enabled it with ReadThrough. This feature allows you to synchronize the cache with Oracle database. If you have a situation where some applications or users are directly updating data in the database, you can enable this feature to ensure that the cache stays fresh.
Db Dependency	Not Supported	Full support. NCache provides support for you to synchronize the cache with any OLEDB database. This synchronization is based on polling. And, although it is not as real-time as a database notification, it is more efficient. It is more efficient because in one poll, NCache can synchronize thousands of cached items instead of receiving thousands of individual database notifications from Oracle in case of OracleDependency.
File Dependency	Not Supported	Full support. NCache allows you to specify a dependency on an external file. Then NCache monitors this file for any updates and when that happens, NCache invalidates the corresponding cached item. This allows you to keep the cached item synchronized with a non-relational data source.
Custom Dependency	Not Supported	Full support. NCache allows you to implement a custom dependency and register your

Feature Area	Redis	NCache
		code with the cache cluster. Then, NCache calls your code to monitor some custom data source for any changes. When changes happen, you fire a dependency update within NCache which causes the corresponding cached item to be removed from the cache. This feature is good when you need to synchronize the cached item with a non-relational data source that cannot be captured by a flat file. So, custom dependency handles this case.

2.11 Runtime Data Sharing

Runtime data sharing has become an important use for In-Memory Distributed Caches. More and more applications today need to share data with other applications at runtime in an asynchronous fashion.

Previously, relational databases were used to share data among multiple applications but that requires constant polling by the applications wanting to consume data. Then, message queues became popular because of their asynchronous features and their persistence of events. And although message queues are great, they lack performance and scalability requirements of today's applications.

As a result, more and more applications are using In-Memory Distributed Caches for event driven runtime data sharing. This data sharing should be between multiple .NET applications or between .NET and Java applications.

NCache provides very powerful features to facilitate runtime data sharing. They are discussed below and compared with Redis.

Feature Area	Redis	NCache
Item Level Events onInsert/onRemove	Supported	Full support. NCache can fire events to its clients whenever specific cached items are updated or removed based on client interest. You can register Java and .NET callbacks with NCache client and your callbacks are called in these cases. NCache uses its own socket-level protocol for this event propagation so it is super-fast.

Feature Area	Redis	NCache
Cache Level Events Add/Insert/Remove	Supported	Full support. If turned on, NCache sends event notifications to all clients whenever any item is added, updated, or removed from the cache. You can register Java and .NET callbacks with NCache client and your callbacks are called in these cases.
Custom Events (Pub/Sub Fired by Apps)	Supported	Full support. NCache allows your applications to fire custom events into the cache cluster. And, other applications can register to be notified for these events. This feature allows you to coordinate a pub/sub scenario with asynchronous event driven coordination between various clients.

2.12 Cache Search (SQL-Like)

In-Memory Distributed Cache is frequently used to cache objects that contain data coming from a relational database. This data may be individual objects or collections that are the result of some database query.

Either way, applications often want to fetch a subset of this data and if they have the ability to search the Distributed Cache with a SQL-like query language and specify object attributes as part of the criteria, it makes the In-Memory Distributed Cache much more useful for them.

NCache provides powerful Object Query Language (OQL) for searching the cache with a SQL-like query.

Feature Area	Redis	NCache
SQL Search	Not Supported	Full support. NCache provides a rich SQL based searching capability. You can search the cache based on object attributes instead of just keys. You can also include Group, Subgroup, Tags, and Named Tags in your SQL query.

Feature Area	Redis	NCache
Continuous Query	Not Supported	Full support. NCache provides a powerful Continuous Query (CQ) feature. CQ lets you specify a SQL query against which NCache monitors the cache for any additions, updates, or deletes. And, your application is notified whenever this happens. Think of this feature as being equivalent to OracleDependency but against the cache and not the database.
LINQ Queries	Not Supported	Full support. NCache allows you to search the cache with LINQ queries from .NET applications. LINQ is a popular object querying language in .NET and NCache has implemented a LINQ provider. So, if you're comfortable using LINQ, you can search the cache the same way you would with NCache SQL.
SQL Search on Tags, Named Tags & Groups	Not Supported	Full support. NCache allows you to include Tags, Named Tags, and Group names as part of you SQL search criteria.

2.13 Data Grouping

An In-Memory Distributed Cache should be much more than a Hashtable with a (key,value) pair interface. It needs to meet the needs of real life applications that expect to fetch and update data in groups and collections. In a relational database, SQL provides a very powerful way to do all of this.

We've already explained how to search an In-Memory Distributed Cache through OQL and LINQ. Now let's discuss Groups, Tags, and Named Tags. These features allow you to keep track of collections of data easily and even modify them.

Feature Area	Redis	NCache
Groups/Subgroups	Not Supported	Full support. NCache provides the ability for you to

Feature Area	Redis	NCache
		group cached items in a group-subgroup combination (or just group with no subgroup). You can later fetch or remove all items belonging to a group. You can also fetch just the keys and then only fetch subset of them.
Tags	Not Supported	Full support. NCache provides a concept called Tags. A Tag is a string that you can assign to one or more cached items. And one cached item can be assigned multiple Tags. And, later, you can fetch items belonging to one or more Tags in order to manipulate them. You can also include Tags in SQL or LINQ search as part of the criteria.
Named Tags	Not Supported	Full support. NCache provides Named Tags feature where you can assign a "key" and "tag" to one or more cached items. And, a single cached item can get multiple Named Tags. Later, you can fetch items belonging to one or more Named Tags. You can also use Named Tags in SQL and LINQ queries as part of the criteria.

2.14 Read-through, Write-through & Cache Loader

Many people use In-Memory Distributed Cache as "cache on the side" where they fetch data directly from the database and put it in the cache. Another approach is "cache through" where your application just asks the cache for the data. And, if the data isn't there, the In-Memory Distributed Cache gets it from your data source.

The same thing goes for write-through. Write-behind is nothing more than a write-through where the cache is updated immediately and the control returned to the client application. And, then the database or data source is updated asynchronously so the application doesn't have to wait for it.

NCache provides powerful capabilities in this area.

Feature Area	Redis	NCache
Read-through	Not Supported	Full support. NCache allows you to implement multiple read-through handlers and register with the cache as “named providers”. Then, the client can tell NCache to use a specific read-through upon a “cache miss”. NCache also allows you to add read-through handlers at runtime without stopping the cache.
Write-through & Write behind	Not Supported	Full support. NCache allows you to implement multiple write-through handlers and register with NCache as “named providers”. Then, whenever application updates a cached item and tells NCache to also call write-through, NCache server calls your write-through handler. If you’ve enabled write-behind, then NCache updates the cache immediately and queues up the database update and a background thread processes it and calls your write-through handler.
Auto Reload at Expiration & Database Synchronization	Not Supported	Full support. If you’ve implemented a read-through handler, NCache allows you to use it to specify that whenever a cached item expires, instead of removing it from the cache, NCache should call your read-through handler to read a new copy of that object and update the cache with it. You can specify the same when database synchronization is enabled and a row in the database is updated and a corresponding cached item would have been removed from the cache but is now reloaded with the help of your read-through.
Cache Startup Loader	Partial support	Full support.

Feature Area	Redis	NCache
	Redis persists everything in a file. Upon restarts, Redis re-loads the state of the cache. Some data loss is be expected since the replication is asynchronous. But, you can't write a custom cache loader.	NCache lets you implement a Cache Loader and register it with the cache cluster. NCache then calls it to prepopulate the cache upon startup. Cache Loader is your code that reads data from your data source/database.

2.15 Cache Size Management (Evictions Policies)

An in-memory In-Memory Distributed Cache always has less storage space than a relational database. So, by design, an In-Memory Distributed Cache is supposed to cache a subset of the data which is really the "moving window" of a data set that the applications are currently interested in.

This means that an In-Memory Distributed Cache should allow you to specify how much memory it should consume and once it reaches that size, the cache should evict some of the cached items. However, please keep in mind that if you're caching something that does not exist in the database (e.g. JSP Sessions) then you need to do proper capacity planning to ensure that these cached items (sessions in this case) are never evicted from the cache. Instead, they should be "expired" at appropriate time based on their usage.

NOTE: Find more details about eviction here.

Feature Area	Redis	NCache
Specify Cache Size (in MBs)	Supported	Supported.
LRU Evictions (Least Recently Used)	Supported	Full support.
LFU Evictions (Least Frequently Used)	Not Supported	Full support
Priority Evictions	Not Supported	Full support. NCache also lets you specify a "do not evict" priority for some cached items and then they are not evicted.
Do Not Evict Option	Supported	Full support. NCache lets you specify "do not evict" option for the entire cache. Then, nothing is evicted even when cache is full. Instead, the client applications receive an error stating that the cache is full when they try to add data to the cache.

2.16 Session Persistence for ASP.NET & Java Web Apps

ASP.NET applications need three things from a good In-Memory Distributed Cache. And, they are ASP.NET Session State storage, ASP.NET View State caching, and ASP.NET Output Cache.

ASP.NET Session State store must allow session replication in order to ensure that no session is lost even if a cache server goes down. And, it must be fast and scalable so it is a better option than InProc, StateServer, and SqlServer options that Microsoft provides out of the box. NCache has implemented a powerful ASP.NET Session State provider. Read more about it at NCache Product Features.

ASP.NET View State caching allows you to cache heavy View State on the web server so it is not sent as "hidden field" to the user browser for a round-trip. Instead, only a "key" is sent. This makes the payload much lighter, speeds up ASP.NET response time, and also reduces bandwidth pressure and cost for you. NCache provides a feature-rich View State cache. Read more about it at NCache Product Features.

Third is ASP.NET Output Cache. Since .NET 4.0, Microsoft has changed the ASP.NET Output Cache architecture and now allows third-party In-Memory Distributed Cache to be plug-in. ASP.NET Output Cache saves the output of an ASP.NET page so the page doesn't have to execute next time. And, you can either cache the entire page or portions of the page. NCache has implemented a provider for ASP.NET Output Cache.

Feature Area	Redis	NCache
ASP.NET Session Caching (basic)	Supported	Full support. NCache has implemented an ASP.NET Session State Provider (SSP) for .NET 2.0+. You can use it without any code changes. Just change web.config. NCache provides intelligent session replication and is much faster than any database storage for sessions.
ASP.NET Session Caching (advanced)	Not Supported	Full support. NCache also provides flexible session locking options to handle robots flooding an ASP.NET website. You can also related sessions with view state so when a session expires, all the corresponding view state is also removed.
ASP.NET Sessions (Multi-Site)	Not Supported	Full support. NCache allows you to share ASP.NET sessions across multiple data centers. This serves situations where you don't want to replicate all sessions to each data center but want the ability to overflow

Feature Area	Redis	NCache
		traffic from one data center to another without losing your ASP.NET sessions. The session moves from one data center to the next as the user moves.
ASP.NET View State Cache	Not Supported	Full support (advanced). Yes. NCache has an ASP.NET View State caching module. Use it without any code changes. Just modify config file. Here are some advanced features supported by NCache: - Group-level policy - Associate pages to groups - Link View State to sessions - Max View State count per user - More
ASP.NET Output Cache	Supported	Supported. NCache has an ASP.NET Output Cache provider implemented. It allows you to cache ASP.NET page output in an In-Memory Cache and share it in a web farm.
Java Web Sessions	Partial support No official support by Redis. But, provided by Spring Framework.	Full support. NCache has implemented a JSP Servlet Session Provider (Java Servlet 2.3+). You can use it without any code changes. Just change web.xml NCache provides intelligent session replication and is much faster than any database storage for sessions.
Java Web Sessions (Multi-site)	Not Supported	Full support. NCache allows you to share Java Web sessions across multiple data centers. This serves situations where you don't want to replicate all sessions to each data center but want the ability to overflow traffic from one data center to another without losing your Java Web sessions.

Feature Area	Redis	NCache
		The session moves from one data center to the next as the user moves.

2.17 Third Party Integrations

Memcached is an open-source in-memory distributed caching solution which helps speed up web applications by taking pressure off the database. Memcached is used by many of the internet's biggest websites and has been merged with other technologies. NCache implements Memcached protocol to enable users with existing Memcached implementations to easily migrate to NCache. No code required for this.

NHibernate is a very powerful and popular object-relational mapping engine. And, fortunately, it also has a Second Level Cache provider architecture that allows you to plug-in a third-party cache without making any code changes to the NHibernate application. NCache has implemented this NHibernate Second Level Cache provider.

Entity Framework from Microsoft is also a very popular object-relational mapping engine. And, although Entity Framework doesn't have a nice Second Level Cache provide architecture like NHibernate, NCache has nonetheless implemented a Second Level Cache for Entity Framework.

Hibernate is a very powerful and popular object-relational mapping engine. And, fortunately, it also has a Second Level Cache provider architecture that allows you to plug-in a third-party cache without making any code changes to the Hibernate application. NCache has implemented this Hibernate Second Level Cache provider.
See Hibernate Second Level Cache for details.

Similarly, Spring Framework is an open source application framework for the Java comprising of several modules that provide a range of services.
See Spring Data Integration for details.

Feature Area	Redis	NCache
NHibernate 2nd Level Cache	No Official Support Not officially supported. Open source projects only	Full support. NCache provides an NHibernate L2 Cache provider that you can plug-in through web.config or app.config changes. NCache has also implemented database synchronization feature in this so you can specify which classes should be synchronized with the database. NCache lets you specify SqlDependency or DbDependency for this.
Entity Framework 2nd Level Cache	Not Supported Not officially supported or even recommended by Redis. Small open	Full support. Custom ADO.NET Provider

Feature Area	Redis	NCache
	source projects are available without support.	NCache has implemented a behind-the-scene second level cache for Entity Framework. You can plug-in NCache to your EF application, run it in analysis mode, and quickly see all the queries being used by it. Then, you can decide which queries should be cached and which ones skipped.
Memcached Protocol Server	Supported	Full support. NCache has implemented Memcached protocol fully. This means you can plug-in NCache as an In-Memory Distributed Cache as a replacement of Memcached. Two ways are offered to use Memcached applications with NCache. Memcached Pug-In: All the popular Open Source .NET Memcached client libraries have been implemented for NCache. Memcached Gateway: Using this you can store your application data from any application that use the Memcached.
Memcached Smart Wrapper	Not Supported	Full support. NCache has implemented the popular .NET and Java Memcached client libraries which in-turn calls NCache. This allows you to plug-in Memcached client library to your application without any code change or recompilation. This wrapper does not require you to go through a Memcached Protocol Server which is an extra hop.
Hibernate 2nd Level Cache	No Official Support Not officially supported. Open source projects only	Full support. NCache provides Hibernate 2nd Level Cache provider that you can plug-in to your Java app without any code changes. NCache has also implemented database

Feature Area	Redis	NCache
		synchronization feature in this so you can specify which classes should be synchronized with the database. NCache lets you specify OracleDependency or DbDependency for this.
JCache API Support	Not Supported	Full support. In fact NCache uses JCache API as its primary API and only provides extended JCache API for features that NCache provides but that are not supported by JCache. So, you can plug-in or plug-out NCache to any JCache application without any code changes.
Spring Caching	Supported Maintained by Spring framework	Full support. NCache fully supports cache integration with Spring Framework version 3.1 and later.

3 Conclusion

As you can see in a very detailed fashion, we have outlined all of NCache features and all the corresponding Redis features or a lack thereof. I hope this document helps you get a better understanding of Redis versus NCache.

In summary, Redis is a popular free cache mainly on Unix/Linux platform with clients running on either Unix or Windows. And, the Windows port of Redis server done by Microsoft OpenTech group not very stable and therefore not as reliable. In fact, Microsoft itself is using the Linux version of Redis in Azure. So, if you want to use Redis, you'll probably want to run it on Unix and then access it from your Windows app servers.

But, the true cost of ownership for an In-Memory Distributed Cache is not just the price of it. It is the cost to your business. The most important thing for many customers is that they cannot afford unscheduled downtime (especially during peak hours). And, this is where an elastic cache like NCache truly shines.

Additionally, all those caching features that NCache provides are intended to give you total control over the cache and allow you to cache all types of data and not just simple data. This is something Redis cannot do.

Please read more about NCache and also feel free to download a fully working 60-day trial of NCache from:

- NCache details.
- Download NCache.

