Alachisoft®

AppFabric

vs.

NCache

Feature Level Comparison

AppFabric v1.1 vs. NCache 5.3 SP1

This document compares AppFabric v1.1 and NCache 5.3 SP1. Read this comparison to:

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

Disclaimer

The comparison provided in this document is to help you get a better understanding of AppFabric versus NCache. The information about AppFabric comes from freely available downloads, product documents, and forums.

We did not conduct any scientific benchmarks for the performance and scalability of AppFabric, so our assessment of it may be different from yours. NCache benchmarks are already published on our website (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. However, do keep in mind any information about AppFabric could be unintentionally incorrect or missing, and we do not take any responsibility for it.

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

Table of Content

1. Executive Summary

This document compares AppFabric v1.1 with NCache 5.3 SP1, and explains their significant differences. This comparison focuses on all the major areas that a good distributed cache should provide.

Feature	AppFabric	NCache
NCache Client Support		
- .NET	Supported	Supported
- Java	Not Supported	Supported
- Scala	Not Supported	Supported
- Node.js	Not Supported	Supported
- Python	Not Supported	Supported
.NET Platform Support		
- Cache Client (.NET 4.x)	Supported	Supported
- Cache Client (.NET 6)	Not Supported	Supported
- NuGet Packages (.NET 4.x)	Supported	Supported
- Cache Server (.NET 4.x)	Supported	Supported
- Cache Server (.NET 6)	Not Supported	Supported
- Server-Side Code (.NET 4.x)	Supported	Supported
- Server-Side Code (.NET 6)	Not Supported	Supported
Operating System Support		
- Windows (Cache Server)	Supported	Supported
- Windows (Cache Client)	Supported	Supported
- Linux (Cache Server)	Not Supported	Supported
- Linux (Cache Client)	Not Supported	Supported
Containers & Docker Support		
- Docker Image (Windows)	Not Supported	Supported

- Docker Image (Linux)	Not Supported	Supported
- Azure Service Fabric	Not Supported	Supported
- Azure Kubernetes Service (AKS)	Not Supported	Supported
- AWS Elastic Kubernetes Service (EKS)	Not Supported	Supported
- Red Hat OpenShift Kubernetes	Not Supported	Supported
Cloud Support		
- Azure Virtual Machine	Not Supported	Supported
- Azure Managed Service	Not Supported	Supported
- AWS Virtual Machine	Not Supported	Supported
- AWS Managed Service	Not Supported	Supported
- Other Leading Clouds	Not Supported	Supported
Performance and Scalability		
- Cache Performance	*Please verify yourself*	Super-Fast
- Cache Scalability	*Please verify yourself*	Extremely Scalable
- Bulk Operations	Partial Support	Supported
- Async Operations	Not Supported	Supported
- Compression	Partial Support	Supported
- Fast Dynamic Compact Serialization	Not Supported	Supported
- Indexes	Not Supported	Supported
- Multiple NIC Binding	Not Supported	Supported
- Pipelining	Not Supported	Supported
Cache Elasticity (High Availability)		
- Dynamic Cache Cluster	Partial Support	Supported
- Peer-to-Peer Architecture	Not Supported	Supported
- Connection Failover	Partial Support	Supported

- Dynamic Configuration	Not Supported	Supported
- Multiple Clusters	Not Supported	Supported
- Named Caches	Supported	Supported
- Cluster Specific Events	Partial Support	Supported
- Split Brain Detection & Auto-Recovery	Not Supported	Supported
Cache Topologies		
- Local Cache	Partial Support	Supported
- Client Cache (Near Cache)	Partial Support	Supported
- Mirrored Cache	Not Supported	Supported
- Replicated Cache	Not Supported	Supported
- Partitioned Cache	Supported	Supported
- Partitioned-Replica Cache	Supported	Supported
- Partitioned Data Balancing	Supported	Supported
- Load Balancing	Not Supported	Supported
- Partitioned Data Affinity	Not Supported	Supported
- Persistence	Not Supported	Supported
WAN Replication (Multi-Datacenter)		
- Active – Passive	Not Supported	Supported
- Active – Active (2 Datacenters)	Not Supported	Supported
- Active – Active (3+ Datacentres)	Not Supported	Supported
- Conflict Resolution	Not Supported	Supported
- De-duplication	Not Supported	Supported
- Data Security	Not Supported	Supported
ASP.NET & ASP.NET Core Support		
- ASP.NET Core Sessions (basic)	Not Supported	Supported
- ASP.NET Core Sessions (advanced)	Not Supported	Supported

- ASP.NET Core Sessions (multi-datacentre)	Not Supported	Supported
- ASP.NET Core Response Cache	Not Supported	Supported
- ASP.NET Session State Caching (basic)	Partial Support	Supported
- ASP.NET Session State Caching (advanced)	Not Supported	Supported
- ASP.NET Sessions State (multi-datacentre)	Not Supported	Supported
- ASP.NET SignalR Backplane	Not Supported	Supported
- ASP.NET View State Cache	Partial Support	Supported
- ASP.NET Output Cache	Supported	Supported
Object Caching Features		
- Get, Add, Insert, Remove, Exists, Clear Cache	Supported	Supported
- Expirations	Partial Support	Supported
- Lock & Unlock	Supported	Supported
- Streaming API	Not Supported	Supported
- Transactions	Not Supported	Partial Support
- Data Portability	Not Supported	Supported
- Item Versioning	Supported	Supported
- Multiple Object Versions	Not Supported	Supported
Managing Data Relationships in Cache		
- Key Based Relationships	Not Supported	Supported
- Key Based Relationships Across Caches	Not Supported	Supported
Cache Synchronization with Database		
- SQL Dependency (SQL Server)	Not Supported	Supported
- Oracle Dependency (Oracle)	Not Supported	Supported
- Db Dependency (Any DB)	Not Supported	Supported
- File Dependency	Not Supported	Supported
- Aggregate Dependency	Not Supported	Supported

- Custom Dependency (polling)	Not Supported	Supported
- Custom Dependency (events)	Not Supported	Supported
Event Driven Data Sharing		
- Item Level Events (OnInsert/OnRemove)	Supported	Supported
- Cache Level Events (Add/Insert/Remove)	Partial Support	Supported
- Custom Events (Fired by Apps)	Not Supported	Supported
- Continuous Query	Not Supported	Supported
- Pub/Sub Messaging (Topic)	Not Supported	Supported
- Pub/Sub Messaging (Queue)	Not Supported	Not Supported
- Pub/Sub Messaging (Multiple or Wildcard Subscriptions)	Not Supported	Supported
- Pub/Sub Messaging (Durable Subscriptions)	Not Supported	Supported
- Pub/Sub Messaging (Non-Durable Subscriptions)	Not Supported	Supported
SQL-Like Cache Search		
- SQL Search	Not Supported	Supported
- LINQ Queries	Not Supported	Supported
- SQL & LINQ on Tags, Named Tags & Groups	Not Supported	Supported
Data Grouping		
- Groups	Not Supported	Supported
- Tags	Supported	Supported
- Named Tags	Not Supported	Supported
Read-through, Write-through, Cache Loader & Refresher		
- Read-through	Supported	Supported
- Write-through	Not Supported	Supported
- Write Behind	Supported	Supported

- Auto Reload at Expiration & Database Sync	Not Supported	Supported
- Cache Start-up Loader & Refresher	Not Supported	Supported
Big Data Processing		
- MapReduce Query	Not Supported	Supported
- Aggregators	Not Supported	Supported
- Entry Processor	Not Supported	Supported
Third Party Integrations & Extensions		
- Entity Framework Core Cache (Extension Methods)	Not Supported	Supported
- Entity Framework 6 Cache	Not Supported	Supported
- NHibernate 2nd Level Cache	Not Supported	Supported
- Server-side Extensible Modules	Not Supported	Supported
- Full Text Search Extensible Module	Not Supported	Supported
- IdentityServer4 Cache and Data Store	Not Supported	Supported
- Memcached Protocol Server	Not Supported	Supported
- Memcached Smart Wrapper	Not Supported	Supported
Security & Encryption		
- Authentication	Supported	Supported
- Authorization	Supported	Supported
- Data Encryption	Partial Support	Supported
- Secure Communication	Supported	Supported
Cache Size Management (Evictions Policies)		
- Max Cache Size (in MBs)	Supported	Supported
- LRU Evictions (Least Recently Used)	Supported	Supported
- LFU Evictions (Least Frequently Used)	Not Supported	Supported
- Priority Evictions	Not Supported	Supported

- Do Not Evict Option	Not Supported	Supported
Distributed Data Structures		
- String	Not Supported	Supported
- List	Not Supported	Supported
- Sorted List	Not Supported	Not Supported
- Set	Not Supported	Supported
- Sorted Set	Not Supported	Not Supported
- Queue	Not Supported	Supported
- Dictionary	Not Supported	Supported
- Counter	Not Supported	Supported
- Hyper Log	Not Supported	Not Supported
- SQL Search on Data Structures	Not Supported	Supported
Cache Administration		
- Admin Tool (Web based GUI)	Partial Support	Supported
- Monitoring Tool (Web based GUI)	Not Supported	Supported
- Monitoring Tool (Prometheus)	Not Supported	Supported
- Monitoring Tool (Grafana)	Not Supported	Supported
- SNMP Counters	Not Supported	Supported
- PerfMon Counters	Supported	Supported
- Admin Tools (PowerShell)	Supported	Supported
- Admin Tools (Command Line)	Supported	Supported
- Administration and Monitoring (API)	Partial Support	Supported
Java Support		
- Java API Support	Not Supported	Supported
- JCache API Support	Not Supported	Supported
- Spring Caching	Not Supported	Supported

- Java Web Sessions	Not Supported	Supported
- Java Web Sessions (multi-datacentre)	Not Supported	Supported

2. Qualitative Differences Explained

2.1. NCache Supported Clients

To allow for user flexibility when employing NCache features, it supports a variety of client applications along with the associated namespaces/packages/classes as demonstrated below.

Feature Area	AppFabric	NCache
.NET	Supported	Supported NCache supports working with .NET applications.
Java	Not Supported	Supported NCache supports working with Java applications.
Scala	Not Supported	Supported NCache supports working with Scala applications.
Python	Not Supported	Supported NCache supports working with Python applications.
Node.js	Not Supported	Supported NCache supports working with Node.js applications.

2.2. .NET Platform Support

For .NET applications, it is important that your distributed cache is also native to .NET, so your entire application stack is .NET. Otherwise, it unnecessarily complicates things for your development, testing, and deployment. This section describes how AppFabric and NCache support .NET platform.

Feature Area	AppFabric	NCache
Cache Client (.NET 4.x)	Supported	Supported Cache Client present as a NuGet

		package and separate installation.
Cache Client (.NET 6)	Not Supported	Supported .NET Core Client is officially supported.
NuGet Packages (.NET 4.x)	Supported	Supported Full set of NuGet Packages Provided.
Cache Server (.NET 4.x)	Supported	Supported NCache server is native .NET.
Cache Server (.NET 6)	Not Supported	Supported NCache server is native .NET Core.
Server-Side Code (.NET 4.x)	Supported	Supported Develop all server-side code like Read-through, Write-through, Write-behind, Cache Loader & Refresher, Custom Dependency, and more in .NET.
Server-Side Code (.NET 6)	Not Supported	Supported NCache server supports .NET Core based server-side code like Read- through, Write-through, Write- behind, Cache Loader & Refresher, Custom Dependency, and more.

2.3. Operating System Support

Most .NET applications run on Windows and now .NET Core applications can run on both Windows and Linux. Therefore, it is important that your distributed cache also provides support for these operating systems. This section describes how AppFabric and NCache support different OS.

Feature Area	AppFabric	NCache
Windows (Cache Server)	Supported	Supported Windows officially supported

		for Cache Server.
Windows (Cache Client)	Supported	Supported Windows officially supported for Cache Client.
Linux (Cache Server)	Not Supported	Supported Linux officially supported for Cache Server.
Linux (Cache Client)	Not Supported	Supported Linux officially supported for Cache Client using .Net Core support.

2.4. Containers & Docker Support

Containers are becoming very popular for deploying applications in the cloud and elsewhere.

Feature Area	AppFabric	NCache
Docker Image (Windows)	Not Supported	Supported Windows Docker Image officially supported for Cache Server.
Docker Image (Linux)	Not Supported	Supported Linux Docker Image officially supported for Cache Server.
Azure Service Fabric	Not Supported	Supported NCache can be deployed inside Azure Service Fabric and accessed.You can use the same BYOL licenses within Azure Service Fabric.
Azure Kubernetes Service (AKS)	Not Supported	Supported NCache can be deployed inside Azure Kubernetes Service (AKS) and accessed. You can use the same BYOL licenses within AKS.

AWS Elastic Kubernetes Service (EKS)	Not Supported	Supported NCache can be deployed inside AWS Elastic Kubernetes Service (EKS) and accessed. You can use the same BYOL licenses within AKS that you used in Azure.
Red Hat OpenShift Kubernetes	Not Supported	Supported NCache can be deployed inside Red Hat OpenShift Kubernetes and accessed. You can use the same BYOL licenses within Red Hat OpenShift that you used in Azure.

2.5. Cloud Support

See how each product compares with each other when it comes to providing support for leading cloud platforms.

Feature Area	AppFabric	NCache
Azure Virtual Machine	Not Supported	Supported Preconfigured NCache Server VMs are available in Azure Marketplace. You can take BYOL licenses to another cloud.
Azure Managed Service	Not Supported	Supported With NCache managed service, you will get provisioned, installed and licensed images on Azure with billing configured through Azure portal.
AWS Virtual Machine	Not Supported	Supported Preconfigured NCache Server VMs are available in AWS Marketplace.
AWS Managed Service	Not Supported	Supported With NCache managed service, you will get provisioned, installed and licensed images on

		AWS with billing configured through AWS.
Other Leading Clouds	Not Supported	Supported You can install NCache in all leading cloud platform VMs. And, you can take BYOL licenses from one cloud to another.

2.6. 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	AppFabric	NCache
Cache Performance	*Please verify yourself* AppFabric uses WCF for client/server and server/server communication. WCF is quite heavy because it is a general-purpose communication protocol. Please test the benchmarks yourself and make sure your seeing performance based on a real-life usage.	Super-Fast NCache is extremely fast. Please see its performance benchmarks showing 2 million ops/sec that can scale further. You can do benchmarking of NCache in your own environment by using stress-testing tools provided with NCache.
Cache Scalability	*Please verify yourself* In non-scientific testing, we've seen AppFabric to not scale very nicely and as you increase load, the overall performance drops.	Extremely Scalable 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. You can do benchmarking of NCache in your own environment by using stress-testing tools provided with NCache.
Bulk Operations	Partial Support	Supported

	Only Bulk Get provided. No Bulk Add, Update or Delete.	Bulk Get, Add, Insert, and Remove.This covers most of the major cache operations and provides a great performance boost.
Async Operations	Not Supported	Supported 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	Partial Support You cannot specify a threshold on object size. As a result, even small cached items are compressed which slows down the cache.	Supported Specify this along with item size threshold and only items larger than the threshold are compressed. Rest are cached uncompressed. This is provided because compressing smaller items often slows things down. And, you can configure “compression” at runtime through “Hot Apply”.
Fast Dynamic Compact Serialization	Not Supported	Supported NCache lets you register your classes with the cache through a Web Based GUI tool (NCache Web Manager). Then, NCache generates serialization code and compiles it in-memory when your application connects to the cache. This code isthen used to serialize objects and it is almost 10 times faster than regular .NET and Java serialization (especially for larger objects).
Indexes	Not Supported	Supported You can use NCache Manager (GUI tool) to create indexes on any attributes of .NET or Java objects. NCache also creates

		indexes automatically on Tags, Named Tags, and Groups. Expiration and eviction policies also use indexes. NCache generates data extraction code at connection time, compiles it in-memory, and uses it for all data extraction instead of .NET and Java Reflection. This is much faster. 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.
Multiple NIC Binding	Not Supported	Supported You can assign two NICs to a cache server. One can be used to talk to the cache server and second for 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.
Pipelining	Not Supported	Supported NCache uses System.IO.Pipelines for high performance IO operations between clients and servers. With pipelining, you dramatically increase scalability. It is enabled by default but can be disabled through config.

2.7 Cache Elasticity (High Availability)

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

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

This is an area where AppFabric 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.

Feature Area	AppFabric	NCache
Dynamic Cache Cluster	Partial Support Not fully dynamic. Dependency on "lead hosts majority rule" means cluster can go down very easily if even one lead host goes down.	Supported NCache is highly dynamic and simple to manage. A shard in NCache is a partition. And, a partition can also have a replica but always on separate server. And, similar to AppFabric, if a cache server goes down, its replica automatically takes over. But, unlike AppFabric, NCache replica automatically rebalances and merges itself into other partitions and all the partitions ensure they have corresponding replicas. This way, NCache is not vulnerable to data loss except during the rebalancing (state transfer) which is quite fast.
Peer-to Peer-Architecture	Not Supported Cache cluster contains regular cache nodes and lead host nodes resembling "master" and "slave"architecture and is therefore not fully peer-to-peer.	Supported NCache cache cluster has a peer to peer 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. This means if any server goes down, the cluster always remains functional and correct (even if there is data loss due to not having replicas).
Connection Failover	Partial Support Client to server failover supported. However, server to server connection failure not fully supported due to lead host majority rule.	Supported 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. The cluster auto-manages itself by rebalancing its data and recreating replicas where needed.
Dynamic Configuration	Not Supported All configurations must be defined before cache starts or the cache is restarted for changes to take effect.	Supported 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 lot of the configuration at runtime without stopping anything.
Multiple Clusters	Not Supported Each cache server can participate in only one cache cluster.	Supported NCache allows you to create multiple cache clusters of either the same or different topologies onthe same set of cache servers.
Named Caches	Supported	Supported NCache allows you to create

		multiple named caches on the same set of servers.
Cluster Specific Events	Partial Support Polling is required by cache clients to receive event notifications.	Supported 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 Detection & Auto-Recovery	Not Supported	Supported Split brain detection is provided and you're notified through NCache events when that happens and auto recovery is provided.

2.8. Cache Topologies

Cache Topologies determine data storage, data replication, and client connection strategy. There are different topologies for different type of use cases. So, it is best to have a cache that offers a rich variety of cache topologies.

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

Feature Area	AppFabric	NCache
Local Cache	Partial Support AppFabric supports InProc Local Caches but not OutProc Local Caches.	Supported 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 copy per machine.
Client Cache (Near Cache)	Partial Support Only supports InProc client cache. Additionally, it does not support the data	Supported Client Cache is a local cache on the cache client machine but one that is connected and

	synchronization with cluster cache (client cache polling mechanism) available in NCache.	synchronized with the cache cluster. Client Cache gives a local cache performance (specially InProc) butwith the scalability of a distributed cache. NCache allows you to configure Client Cache without any code changes to the client application.
Mirrored Cache	Not Supported	Supported Mirrored Cache is a 2-node Active-Passive cache. All clients connect to the Active node and data mirroring is done asynchronously. In case the Active node goes down,Passive node automatically becomes Active and all clients connect to it automatically.
Replicated Cache	Not Supported	Supported In Replicated Cache, the entire cache is replicated on all nodes inthe 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 onenode. You can enable load-balancing or specify an ordered server list for the clients to use.
Partitioned Cache	Supported Does not allow changes to be made on runtime.	Supported Full failover support if any server goes down (although

		there is data loss). The Partitioned Cache is a very powerful topology. You can partition without replication to speed up the cache and also use less memory because you can always reload some data if lost in the cache. In Partitioned Cache, the entire cache is partitioned and each cache server gets one partition. All partitions are created or deleted and their buckets reassigned automatically at runtime when you add/remove nodes. Data re-balancing feature is provided even if no partition is added or removed but when any partition gets overwhelmed with too much data. Each client is connected to all cache nodes. This allows it to directly go where the data is (single hop).
Partitioned-Replica	Supported	Supported Similar to partitioned but contains replicas in different nodes as backups.
Partitioned Data Balancing	Supported	Supported 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. You can turn off auto rebalancing and manually do it if you wish.
Load Balancing	Not Supported	Supported NCache supports a load

		balancing option as well and clients are balanced among server nodes in case of Replicated Cache topology.
Partitioned Data Affinity	Not Supported	Supported NCache supports storing cache items with identical keys at the same node to save on matching cost.
Persistence	Not Supported	Supported NCache has the equivalent of RDB persistence though Dump/Reload tools that take a snapshot of the cache and persist them to a disk or reload the cache from a previous dump.

2.9. 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 for propagating the data replication. NCache provides Bridge Topology to handle all of this.

Feature Area	AppFabric	NCache
Active – Passive (2 datacenters)	Not Supported	Supported You can create a Bridge between Active and Passive sites. The Active site submits all updates to the Bridge which then replicates them to the Passive site.
Active – Active(2 datacenters)	Not Supported	Supported You can create a Bridge between two active sites. Both submit updates to the Bridge which handles conflicts on “last update wins” rule or through custom conflict resolution handler provided by you. Then,

		the Bridge ensures that both sites have the same update.
Active – Active (3+ datacenters)	Not Supported	Supported You can create a Bridge between three or more active sites. All submit their updates to the Bridgewhich handles conflicts on “last update wins” rule or through a custom conflict resolution handler provided by you. Then, the Bridge ensures that all sites have the same update.
Conflict Resolution	Not Supported	Supported 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 contentof both objects and deciding.
De-duplication	Not Supported	Supported 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	Supported You can encrypt data with 3DES and AES algorithms before transportation. Otherwise, you can use a VPN between data centers for security.

2.10. ASP.NET & ASP.NET Core Support

Given ASP.NET Core applications can persist their Sessions in a distributed cache and ASP.NET applications need three things from a good in-memory distributed cache; ASP.NET Session State storage, ASP.NET View State caching, and ASP.NET Output Cache. ASP.NET, it is essential for a

complete distributed cache to provide these features. The 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, State Server, and SQL Server 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.

Additionally, 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	AppFabric	NCache
ASP.NET Core Sessions (basic)	Not Supported	Supported NCache has implemented an ASP.NET Core Sessions Provider. NCache provides intelligent session replication and is much faster than any database storage for sessions.
ASP.NET Core Sessions (advanced)	Not Supported	Supported 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 Core Sessions (multi-datacenter)	Not Supported	Supported NCache allows you to share ASP.NET Core 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 ASP.NET Core sessions. The session moves from one datacenter to the next as the user moves.
ASP.Net Core Response Cache	Not Supported	Supported NCache implementation of IDistributedCache utilizes Distributed Cache Tag Helper that provides the ability to dramatically improve the performance of your ASP.NET Core app by caching its responses.
ASP.NET Session Caching (basic)	Partial Support ASP.NET Session replication is not fast and scalable because of WCF being the transport layer.	Supported 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	Supported 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-datacenter)	Not Supported	Supported 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 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 SignalR Backplane	Not Supported	Supported NCache supports the SignalR Backplane. It allows users to develop real-time ASP.NET web apps with servers that broadcast updates to all registered clients.
ASP.NET View State Cache	Supported AppFabric supports ASP.NET View State Cache but does not allow for any advanced capabilities.	Supported (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, etc.
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.

2.11. 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	AppFabric	NCache
Get, Add, Insert, Remove, Exists, ClearCache	Supported	Supported NCache provides more

		variations and more control to the user.
Expirations	Partial Support Only absolute expiration provided. No sliding expiration is available.	Supported 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 aftera 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	Supported 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	Not Supported	Supported 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	Not Supported	Partial Support - Explicit locking - Implicit locking (item versioning) - Entry Processor (are

		atomic)
Data Portability	Not Supported	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	Supported	Supported 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.	Supported 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.

2.12. Managing Data Relationships in Cache

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	AppFabric	NCache
Key Based Relation	Not Supported	Supported 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 Relationships Across Caches	Not Supported	Supported This is an extension of Key Based Cache Dependency except it allows you to create this dependency across multiple caches.

2.13. Cache Synchronization with Database

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 very powerful database synchronization features.

Feature Area	AppFabric	NCache
SQL Dependency (Sync with SQLServer) (Event based)	Not Supported	Supported NCache provides SqlDependency support for SQL Server. You can associate a cached item with a SQL statement-based dataset in SQL Server database. Then whenever that dataset changes (addition, updates, or removal), SQL Server sends a 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 SQL Server 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.
Oracle Dependency (Sync with Oracle) (Event based)	Not Supported	Supported NCache provides OracleDependency support for Oracle. You can associate a cached item with a SQL statement-based 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 ensures cache freshness.
Db Dependency (Sync with OLEDB) (Polling based)	Not Supported	Supported NCache provides support for you to synchronize the cache with any OLEDB database. This synchronization is based on polling. And, although it isnot 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 incase of OracleDependency.
File Dependency (Sync with Non-Relational Source)	Not Supported	Supported 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.
Aggregate Dependency	Not Supported	Supported NCache also supports using different strategies on the same cache data in the form of Aggregate Cache Dependency. For example, you can associate key dependency and file dependency with an item using aggregate dependency and data will be invalidated based on the first dependency triggered.
Custom Dependency (Sync with any DB) (Polling based)	Not Supported	Supported NCache allows you to implement a custom dependency and register your 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.
Custom Dependency (Sync with Any DB) (Event Based)	Not Supported	Supported NCache allows you to implement an event based custom dependency called NotifyExtensibleDependency to receive notifications from your data source whenever data changes so you can update the cache. With this, you can write custom code to sync cache through event notification against SQL Server, Oracle, CosmosDB, MongoDB, and others.

2.14. Event Driven Data Sharing

Event Driven 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 asynchronously.

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.

NCache provides very powerful features to facilitate Event Driven Data Sharing. They are discussed below and compared with AppFabric.

Feature Area	AppFabric	NCache
Item Level Events (onInsert onRemove)	Supported	Supported 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.
Cache Level Events (Add/Insert/Remove)	Partial Support Cache Region event notifications. These events require polling by the cache clients which has a negative performance impact.	Supported 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 (Fired by Apps)	Not Supported	Supported 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.
Continuous Query	Not Supported	Supported 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.
Pub/Sub Messaging (Topic)	Not Supported	Supported NCache allows your applications to do Pub/Sub style messaging through Topic.
Pub/Sub Messaging (Multiple or Wildcard)	Not Supported	Supported You can specify the topic names through a wildcard pattern. And,

		it not only maps you to all the existing topics that match this pattern but also looks for any new topics added at runtime matching the same pattern and automatically subscribes to them. Pattern can be identified wildcards along with multiple delivery options.
Pub/Sub Messaging (Durable Subscription)	Not Supported	Supported Durable subscription allows clients who disconnect with NCache for any reason to not lose their messages. When they reconnect, they see all the message sent while they were disconnected.
Pub/Sub Messaging (Non-Durable Subscription)	Not Supported	Supported This method allows subscribers to register to only receive messages intended for it for as long as it stays connected.

2.15. SQL-Like Cache Search

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 SQL-like searching capability of the cache.

Feature Area	AppFabric	NCache
SQL Search	Not Supported	Supported 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, Tags, and Named Tags in your SQL query.
LINQ Queries	Not Supported	Supported 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 & LINQ Search on Tags, Named Tags & Groups	Not Supported	Supported NCache allows you to include Tags, Named Tags, and Group names as part of you SQL search criteria.

2.16. Data Grouping

An in-memory distributed cache should be much more than a Hash table 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 SQL 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	AppFabric	NCache
Groups	Not Supported	Supported NCache provides the ability to fetch or remove all items belonging to a group. You can also fetch just the keys and then only fetch subset of them.
Tags	Supported Tags in AppFabric are specific to regions and not at cache level.	Supported 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	Supported NCache provides Named Tags feature. Each Named Tag has a “key” and a “tag value” portion. So, you can search for items basedon tag key and tag value and not just tag value. Named Tag works the same as a regular Tag.

2.17. Read-through, Write-through, Cache Loader & Refresher

Many people use the 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 “read 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 returns 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	AppFabric	NCache
Read-through	Supported	Supported 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	Not Supported	Supported 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.
Write-behind	Supported	Supported If you’ve enabled write-behind, then NCache updates the cache immediately and queues up the database update and then a background thread processes it and call your write-through handler.
Auto Reload at Expiration & Database Sync	Not Supported	Supported 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 & Refresher	Not Supported	Supported 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 the code that reads data from your data source/database. However, this approach is likely to result in stale data. As the user loads the relevant data at cache startup, and any change occurring in the data source outdates it. To prevent this invalidation, NCache provides another feature called cache refresher. Which synchronizes the cache with the database based on a specified refresh interval.

2.18. Big Data Processing

In-memory procedures are ideal when analyzing and processing significant data amounts. A distributed cache is a scalable in-memory data store. And, if it can support the popular Map/Reduce style processing, then you're able to speed up your work greatly.

Feature Area	AppFabric	NCache
MapReduce Query	Not Supported	Supported NCache provides a MapReduce framework where your program can run on cache servers for parallel processing of Big Data.
Aggregators	Not Supported	Supported NCache provides Aggregator that works with MapReduce framework and provides you statistical data.
Entry Processor	Not Supported	Supported NCache fully supports Entry Processor execution on cache nodes in parallel.

2.19. Third-Party Integrations & Extensions

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.

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. See NHibernate Second Level Cache for details.

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.

Feature Area	AppFabric	NCache
Entity Framework Core Cache	Not Supported	Supported NCache has implemented EF Core Extension Methods for caching to make it really simple for EF applications to use caching. It also gives full control to the application about how to cache data.
Entity Framework 6 Cache	Not Supported	Supported 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.
NHibernate 2nd Level Cache	No Official Support This feature is not officially supported. Only opensource projects support them.	Supported NCache provides a 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.
Server-side Extensible Modules	Not Supported	Supported NCache allows you to dynamically register server-side modules that can use NCache's distributed architecture. Lucene.NET support is one such module.
Full Text Search Extensible Module	Not Supported	Supported NCache provides Full Text Search through the industry standard Lucene.
IdentityServer4 Cache and Data Store	Not Supported	Supported NCache supports the IdentityServer4 authentication server to provide a collective way to authenticate requests to all of your applications, whether they are web based, native, mobile based or API endpoints.
Memcached Protocol Server	Not Supported	Supported 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 Opensource .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	Supported 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 youto go through a Memcached Protocol Server which is an extra hop.

2.20. 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 cache 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.

Feature Area	AppFabric	NCache
Authentication	Supported Through Windows Identity Foundation.	Supported You can authenticate users against Active Directory or LDAP.
Authorization	Supported AppFabric allows authorization through 3 users groups; "administrators", "Observers", and "Users".	Supported 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	Partial Support AppFabric only provides encryption for transporting data between the client cache and cache cluster. No encryption is	Supported You can enable encryption and NCache automatically encrypts all items one client-side before sending them to the cache.

	available for items stored in 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	Supported	Supported NCache provides SSL support for client/server communication. Additionally, strong encryptions are provided by NCache so you can encrypt data over an unsecured connection.

2.21. Cache Size Management (Evictions Policies)

An 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. ASP.NET 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.

Feature Area	AppFabric	NCache
Max Cache Size (in MBs)	Supported	Supported NCache supports setting limits on cache sizes.
LRU Evictions (Least Recently Used)	Supported	Supported NCache supports least recently used eviction.
LFU Evictions (Least Frequently Used)	Not Supported	Supported

		NCache supports least frequently used eviction policy using which you will be able to keep that data in cache for longer which is being used more frequently.
Priority Evictions	Not Supported	Supported This eviction policy lets the cache evict lesser important data first.
Do Not Evict Option	Not Supported	Supported 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.

2.22. Distributed Data Structures

NCache provides an extensive set of distributed data structures with .NET interfaces.

Feature Area	AppFabric	NCache
String	Not Supported	Supported NCache provides string data structures.
List	Not Supported	Supported NCache provides support for distributed list.
Sorted List	Not Supported	Not Supported
Set	Not Supported	Supported NCache provides the .NET HashSet class implementation in a distributed manner.
Sorted Set	Not Supported	Not Supported

Queue	Not Supported	Supported NCache provides a Distributed Queue.
Dictionary	Not Supported	Supported NCache provides an IDictionary implementation but in a distributed manner.
Counter	Not Supported	Supported
Bitmap	Not Supported	Not Supported
Hyper Log	Not Supported	Not Supported
SQL Search on Data Structure	Not Supported	Supported NCache allows you to use SQL/LINQ to search items inside collection type of data structures like List, Queue, Dictionary, and Set.

2.23. Cache Administration

Cache administration is a very important aspect of any distributed cache and support the following:

1. GUI based and command line tools for cache administration including cache creation and editing/updates.
2. GUI based tools to monitor 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.

Feature Area	AppFabric	NCache
Admin Tool	Not Supported	Supported NCache Web 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 (Web-Based GUI)	Partial Support	Supported

	AppFabric provides a very basic monitoring through Azure Portal.	NCache Web Monitor provides rich set of performance counters and other stats. NCache Web 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 with a lot of details. And, you can incorporate non-NCache PerfMon counters in it for comparison with NCache stats. This real-time comparison is often very important.
Monitoring Tool (Prometheus)	Not Supported	Supported NCache allows you to monitor Distributed Caches, Distributed Cache with Persistence, the Pub/Sub Message Store, Distributed Lucene, Clients and Bridges through the extensive counters published by NCache on a single platform.
Monitoring Tool (Grafana)	Not Supported	Supported NCache provides a Grafana Application Plugin that uses Prometheus as a data source to display cluster metrics.
SNMP Counters	Not Supported	Supported NCache supports Simple Network Management Protocol counters to see how the network devices of the associated cache are communicating and sharing information with one another.
PerfMon Counters	Supported AppFabric provides a range of PerfMon counters for monitoring and troubleshooting the cache.	Supported NCache provides a rich set of PerfMon counters that can be seenfrom NCache Web Manager, NCache Web Monitor,

		or any third-party tool that supports PerfMon monitoring.
Admin Tools (PowerShell)	Supported PowerShell Cmdlets provided. You have to use PowerShell Console or write scripts to use these Cmdlets.	Supported NCache provides a rich set of PowerShell admin tools. You can create a cache, add remote clientsto it, add server nodes to it, start/stop the cache, and muchmore. Use these tools from your PowerShell scripts and automate various cache admin operations.
Admin Tools (Command Line)	Supported	Supported 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 REST API. No .NET or Java events provided for cluster changes at runtime.	Supported NCache provides Java and .NET APIto manage and monitor the caches & client. Using this API, you can stop/start the cache, receive connected client statistics or the health of the cluster.

2.24. Java Support

NCache provides strong support in all of these areas. See how AppFabric compares with NCache.

Feature Area	AppFabric	NCache
Java Client	Not Supported	Supported NCache provides a Java Client with full support.

JCache API	Not Supported	Supported NCache provides JCache API as its primary API for Java applications and only provides extended JCache API for features that NCache provides but that are not supported by JCache. As a result, you can plug-in NCache to any JCache application without any code changes.
Spring Caching	Not Supported	Supported NCache fully supports integration with Spring Framework version 3.1 and further.
Java Web Sessions	Not Supported	Supported 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 session storage.
Java Web Sessions (Multi-site)	Not Supported	Supported NCache allows you to share Java Web sessions across multiple datacenters. This serves situations where you don't want to replicate all sessionsto each data center but want the ability to overflow traffic from one datacenter to another without losing your Java Web sessions. The session moves from one datacenter to the next as the user moves.

3. Conclusion

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

Please note that the true cost of ownership for a 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. Please read more about NCache and also feel free to download a fully working 60-day trial of NCache from:

NCache Details Download NCache