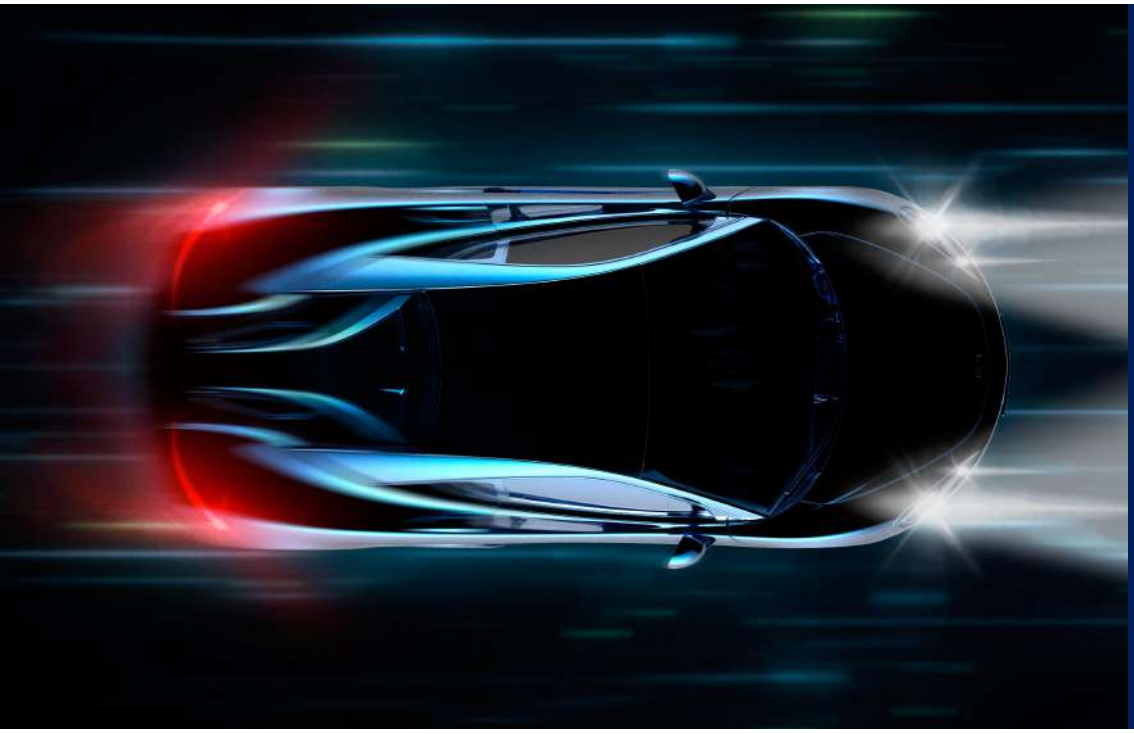
NCache

Case Study



> We focused on NCache because we like the way it improves application performance and how it meets all our caching needs.”
>
> Team Lead
> Systems Architect

Top 10 Auto Maker

NCache allows this automaker to maintain high-availablity and accelerate response time of their Automobile Financial Services Application.

"We focused on NCache because we like the way it improves application performance and how it meets all our caching needs."

Team Lead
Systems Architect

Country or Region:
Germany

Industry:
Automobile Finance

Company Profile:
One of the top ten premium manufacturers of automobiles was established in Germany.

They have been manufacturing their automobiles for a long time and have successfully delivered around millions of automobiles by the end of 2018, including motorcycles and cars.

Introduction

Automobile Finance industry is one of the largest industries in the world. With a massive customer base, they have their network spread all over the world. One of the top ten auto makers, with hundreds of dealers all across USA and tens of thousands of customers, needs high availability and scalability within their automobile finance applications for their everyday use. As a leading auto maker, their main goal is to provide a value service for their dealers and the end customers by providing a fast and reliable data access mechanism.

Their application is divided up into two parts. First is an interactive initiation system for customer facing self-service portal and second is a dealer facing portal for their financial service management. Financial services include loan management, payment plans, installment plans and contract management. The multiple systems handled by their application are:

- **Initiation System:** this serves the purpose of talking to the dealer & signing loan agreement for the purchase of an automobile.
- **Contract Management System:** an account management system that goes through the life span of the automobile.
- **Customer Self-Service:** a self-service portal for the customer which provides basic information about pricing and purchase details.

A diverse system with a large number of end users and multiple functionalities demand a high-speed data delivery along with consistency on their public facing site.

Challenges

With high transactional applications, the main challenge is to maintain fast speed and high availability of data. The application heavily relies on the backend relational database and it becomes a performance and scalability bottleneck under high transactional load. With hundreds of

thousands of end-users, there is a high request load causing slow performances. Additionally, the end-users are located in different time zones. The load shifts over regions thus causing increased transactional loads on the database. It results in a negative impact on the user and an associated business cost.

Other than that, with large data objects the data transmission over the network becomes very slow. The network trips are very costly for large data objects and it affects the overall performance of the application.

Static data can be kept in-memory to avoid the database trips every time a request is made. This reduces the network trips as they are expensive in nature. However, the in-memory data needs to be synchronized with the database as there are two separate copies of the data and they can fall out of sync. Thus, data synchronization is another challenge their application is facing.

Alachisoft Product:

NCache

Customer Needs Met:

- High Availability
- Consistency
- Stability
- Reliability

NCache is the Solid Fix!

NCache provides them with a flexible and scalable solution for their multi-tiered application along with its advanced features. Caching the frequently used data in memory using NCache not only speeds up the application response time but also saves the database trips every time a request is made. They are experiencing a faster data access and stability on their application with NCache as their caching tier. NCache provides them with the ease of adding more servers in the caching tier to increase the application's request handling threshold. It helps with the accommodation of increasing request load on the application.

There is no downtime and no user data loss with NCache always ON application setup for their mission critical system. This way, unplanned outages or planned maintenance do not affect the application performance.

Architectural Overview

The application architecture comprises of the following components:

- **Customer Self-Service Application (ASP.NET)**
  It is a customer facing portal designed in ASP.NET. It is used by the customers.

- **Dealer Facing Loan & Contract Management App (ASP.NET)**
  It is for the dealers and also designed in ASP.NET.

- **Middleware Services (.NET Web Services)**
  This layer comprises of a number of services that are used for the data accessing. The applications on the front-end talk to the middleware for services. This layer handles all the caching for the application.

For a closer look on the application, their web application includes a public facing web portal which is used by the customers and dealers in regards to the financial services for the automobiles. The self-service customer portal retains the information about the loans, payment plans, pricing etc. The dealers all across the country use the dealer facing portal which retains information about user's credit history, installment plans and contract management.

The middleware service tier has a number of WCF SOAP services responsible for accessing the data from the backend database. It comprises of 12-17 servers and this is where NCache is used. The front-end web application communicates with the middleware for requests such as user records, pricing details and loan management etc.

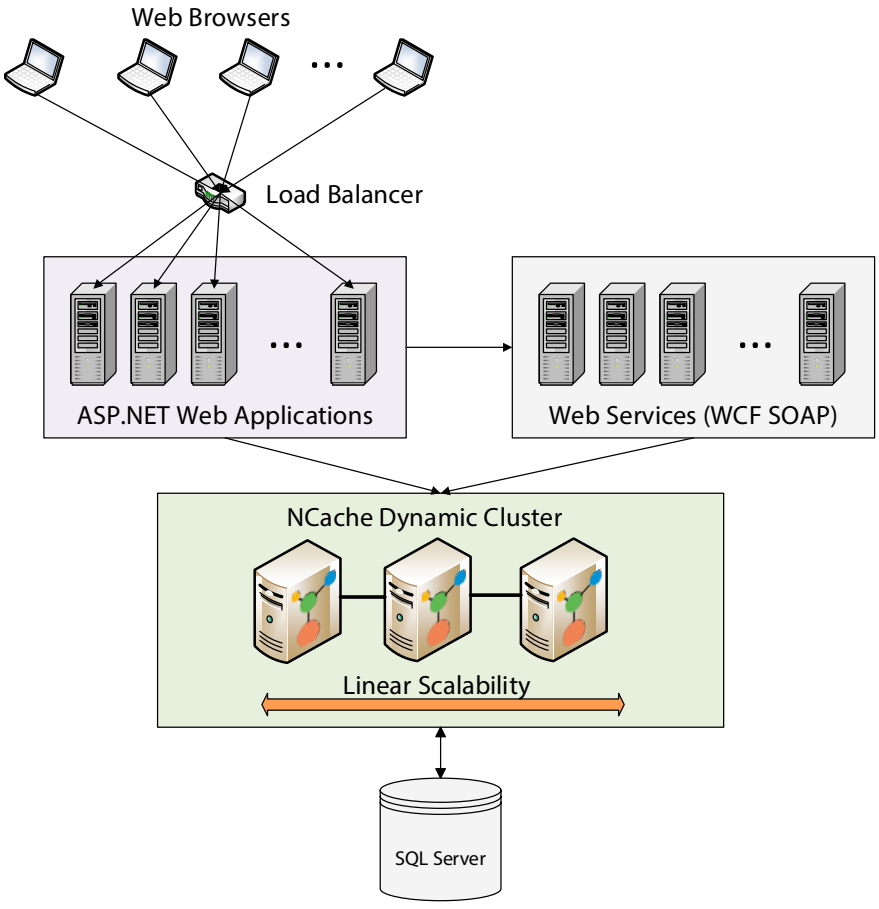


Figure 1 - Distributed cache reducing network latency and increasing response time

Caching tier is a dedicated tier of 4 cache servers that physically resides very close to the web servers so that they are on the same network. This reduces the network latency and improves the request response time.

One of the main information they are caching is their complex pricing information that contains a set of parameters such as year of manufacture, model, regions etc.

They are using SQL Server as their relational database at the backend. The following diagram depicts the application architecture visually. They have around 400-500 dealers located all over USA. Their network load thus shifts with respect to the active users. They further explained that their applications is very high transactional with 2-3 million transactions per second.

High Performance and Scalability

NCache is extremely fast and scalable with a distributed caching solution. The data residing in NCache saves the application from making costly trips to the database. They make complex pricing calculation on their pricing object and then cache this data in NCache. This has helped them improve application performance and also simplified the architecture where they don't have to perform these complex pricing calculations again if data is already in NCache. If the data is not available in the cache, then this pricing data is fetched from the database and is cached for future usage.

This has improved end user experience and has a positive business impact.

Client Cache for InProc Speed

The customer portal displays information about the prices of the cars and the details such as manufacturing details and model etc. The catalogue displays the pictures of the vehicles along with vehicle details. This data is mostly static and some of it is large in size as well which makes it expensive to fetch from database for individual requests. So, caching the static data for usage enhances the user experience as the response time is better along with the application performance.

NCache provides a specialized feature called Client Cache that is ideal for caching static data. Client Cache allows them to keep data locally on the application servers (closer to the application) and helps them save expensive network trips in addition to expensive database trips that NCache was already saving for them. Client Cache is also synchronized with their remote clustered cache that ensures data consistency while improving application performance.

Client cache also resolves the performance issues associated with large data objects. Client Cache originally resides these large objects locally on

the same client machine hence, saving the back and forth network trips. The application experiences a lot faster data retrieval with NCache client cache.

In addition to Client cache, it is also recommended that they turn on compression to reduce the size of the object to further tune the performance.

Cache Loader for Pre-Loading the Cache

With caching, the application first checks for the data in the cache for example, in order to view the price details the cache is checked first for pre-calculated pricing data. If the cache fails to provide with the required data, the data is looked for in the database. It is then saved in the cache for future usage. It improves the application performance but it costs a database trip for every data item that is not cached.

This problem is handled using the Cache Startup Loader. It pre-loads most of the data in the cache on startup as most of their data is static. It makes the data highly available and saves the network cost. In the above example, for pricing details which require fast data loading, the cache will always retain the data. It is of great help for further improving the application performance and handling the request throughput.

Synchronizing Cache with SQL Server

NCache retains the data from the database and keeps it for usage. It improves the throughput and performance of the application. However, for two separate copies of data, synchronization is a major necessity. If the data in the database is updated, the cache data becomes stale and the application keeps using the stale data. In order to keep the cache and database synchronized, NCache provides with a feature of database synchronization which keeps the data in the both the data sources consistent. For any change in the database, the data in cache automatically gets removed and fetches the latest copy of data, the next time it is asked for. This way, the application will always perform operations on the updated data set.

Moving Forward with NCache

The automobile finance company is benefitting from NCache in a lot of areas as discussed above and moving further, they are looking forward to use Microservices architecture with NCache. They have decided to enhance NCache usage further for their mission critical Microservices applications for a higher uptime across all user regions. "I am really hoping that we can move something fairly easy next and we start taking

About Alachisoft:

Alachisoft provides a popular high performance in-memory distributed cache called NCache. NCache is an Open Source middleware that runs in production environment and boosts performance and scalability of .NET web apps, SOA service apps, and general high traffic server apps. NCache has a 13 year proven track record with hundreds of customers all over the world and specially in US, UK, and Western Europe.

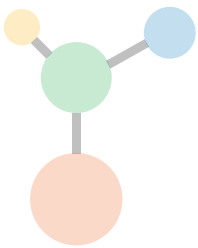
Visit our website at
www.alachisoft.com

You can download a free 60 days fully working trial of NCache from here:
www.alachisoft.com/ncache

Alachisoft
Corporate Headquarters
12005 Ford Road, Suite 520
Dallas, TX 75234

US: +1 (214) 764-6933
UK: +44 207 993-8327
Fax: +1 (925) 886 8361

Sales Email:
sales@alachisoft.com
Technical Support:
support@alachisoft.com



advantage of some advanced features of NCache", said one of the architects of the team.

They are planning to use .NET Core with Kubernetes for this application which NCache fully supports. They have also shown their interest in event driven Pub/Sub messaging feature of NCache to provide communication between their microservices. NCache is able to manage all of this for them (.NET Core, Containerization, Event Driven Pub/Sub Messaging) while managing super-fast performance and extreme scalability.