HAZELCAST

Case Study

Accelerating Mobile Apps with In-Memory Technologies

Introduction

Mobile app deployment should be a core component of your online strategy, regardless of your company's industry. But since the stakes have been continually raised, simply having a mobile presence is not nearly enough. As more consumers turn to mobile access and increasingly have higher expectations, apps that deliver a great customer experience are a must. This requirement is especially true for banks, as they face significant competition and must have customer retention as a top business priority. If a bank's mobile apps do not provide the ease-of-use and responsiveness that customers expect, the customer could choose to bank elsewhere. Never mind that customers might also use social media to express their complaints about the user experience, thus discouraging prospective customers from working with the bank.

Certainly, the mobile app experience does not rest entirely on the app developers' shoulders, as much of the core processing is done on backend servers. This means that these backend, data processing systems must be fast, scalable, and resilient to drive the data that enables the apps.

Background

Swedbank is a large banking group based in Stockholm that offers retail banking, asset management, and other services. Its nearly 15,000 employees are based around the world, with a focus on Sweden and the Baltic countries. It has over 7 million private customers, 5 million of which are digitally active, and over 600,000 corporate customers.

In the Digital Banking organization at Swedbank, the Tech Stream team is responsible for the backend systems that drive online channels such as Internet banking and mobile banking. Mostly made up of software engineers, the team also includes an enterprise architect and continuous integration / continuous delivery (CI/CD) engineers. The team provides the infrastructure to deliver data used by customers and bank clerks, so they work with the application developers that create the user interfaces. They also work with the core banking team to provide a data layer that delivers data to the end-user apps. In their highly competitive industry, they continually look for ways to build out their architecture to ensure performance, scalability, reliability, and agility.

Challenge

As the bank's application development teams were building more solutions, the Tech Stream team was facing significant overhead in setting up the supporting backend systems. Since all apps required data storage, they used a traditional relational database management system (RDBMS). But since they had to create a schema for every use case, even the most basic ones, the work became more involved than they felt it needed to be. This inhibited their ability to move quickly for any new solution the market demanded.

They soon learned that with one of their implementations using an RDBMS, the data storage was actually being handled entirely in memory. The non-volatile, disk-based storage component of their RDBMS was actually not used under the covers. This was another data point that suggested an RDBMS was overkill and not the right technology choice for their needs.

They were also using other technologies, but they sought a much lighter weight solution that would eliminate the time-consuming schema definition process while providing higher levels of performance. They simply needed a data layer that could store data and return it quickly.

Reliability was also a huge issue for them. Their RDBMS was a single point of failure, as they weren't running it in a clustered configuration, and so they faced many problems along the way. They looked for a system that had the business continuity features to let them achieve extremely high uptime.

Security was also obviously a pivotal issue, considering all the sensitive data they maintained. They would not consider any new technology that did not have a comprehensive set of built-in security capabilities.

Solution

They did some research and looked at a few in-memory technologies. Hazelcast was a strong candidate, especially since they knew it was being used in other software technologies. They had several vendor apps that bundled in Hazelcast. Hazelcast has a solid track record of providing a foundation for in-memory and distributed computing to other popular data platforms, so that gave it a lot of credibility.

Their evaluation of Hazelcast was quick, and they decided it was the right technology to use after a few weeks. During the research phase, they learned from published performance benchmarks that Hazelcast could deliver the speeds they needed. They also learned more about the security and business continuity capabilities and needed to leverage those in the enterprise version of the software.

The fact that it had a native Java API made it very easy to use for their Java software engineers, so the learning curve to get started was low. The ability to store data in object form meant that Hazelcast was a more natural fit for the types of data they were storing. Since they didn't have to build a schema that might have to be changed later, Hazelcast gave them the flexibility that allowed them to move more quickly.

Getting Hazelcast installed and running was very simple, and they could integrate their first application with Hazelcast in just a few hours. They started with the open source version of Hazelcast, and soon after doing some testing, they reached out to Hazelcast to work with the enterprise version. As part of Hazelcast Enterprise, the WAN Replication, 24/7 support, and security features were critical to their deployment.

Their first use case was to store single sign-on (SSO) security tokens for both the customer-facing and employee-facing systems. This implementation replaced the aforementioned RDBMS-based solution that was not storing data to disk. Their second use case was a security token cache for their open banking systems. Both of these use cases were great fits for the in-memory storage that Hazelcast provided.

The team offers Hazelcast as a service to other development teams, and there are four use cases from other teams that leverage Hazelcast at Swedbank. For example, the savings and investment team uses Hazelcast as a caching data layer. For them, fund details that are retrieved from an external system can be cached in Hazelcast, and any app that needs access to that data can retrieve it directly from the Hazelcast cluster rather than making another call to the external system.

As a distributed system, Hazelcast creates replicas (in Hazelcast parlance, these are called “backups”) across the cluster to ensure high availability despite node failures at a single data center. So Swedbank was covered in terms of safeguarding against common hardware failures. To round out their business continuity strategy, they implemented a disaster recovery topology using the WAN Replication feature. They had two separate clusters, one for customer use and the other for employee use, at each of the two data centers in Stockholm for a total of four clusters. The two clusters in

the primary data center replicate their data to the secondary data center in an active-passive configuration. In this configuration, if a site-wide failure occurs at the primary data center, the secondary data center can take over. Swedbank could potentially deploy this as an active-active topology in which all four clusters are in production use, but this has not been necessary so far.

Results

With the new Hazelcast implementation, their key concerns were fully addressed. It is used in their biggest app, which hosts the Internet banking and mobile banking systems, and multiple mobile apps are in use. Even reliability no longer was an issue, as they had no issues with Hazelcast in the two years it has been running. They did face one problem that impacted the system, but it was actually due to a hardware switch failure unrelated to Hazelcast.

Their successful production deployment of Hazelcast is a solid testimonial of their choice of technology. They now have the capabilities they sought with the lightweight solution that makes sense for their environment. They manage and upgrade the Hazelcast cluster themselves and can do so without a separate operations team. They expect many more use cases to be implemented with Hazelcast in the future.

"Our previous database was a single point of failure. After switching to Hazelcast, we not only decreased latency, but we also haven't had any problems with Hazelcast since the day we started to use it. Hazelcast is very easy to host, so we can do that ourselves."

— John Robert Hoglund, Sr. Software Engineer, Swedbank

Next Steps

Swedbank is undergoing a migration to a containers- and Kubernetes-based architecture based on Red Hat OpenShift to implement a private cloud in Swedbank data centers. As IBM is both a big technology partner with Hazelcast and a big technology vendor for Swedbank, there is significant potential in the use of Hazelcast within OpenShift. Hazelcast is certified to run on IBM Cloud Paks, which run on OpenShift. In fact, Hazelcast is the primary in-memory technology option for Cloud Paks, so Swedbank benefits from that joint technology effort. Swedbank would like to see more dynamic configuration capabilities within Hazelcast, and the joint work between IBM and Hazelcast should address that soon.

They are currently using the public cloud to some extent, with the expectation to make a bigger push soon. Since they are big users of IBM mainframes, they expect to use an ongoing mix of on-premises and public cloud deployments. With Hazelcast in the cloud (with Hazelcast Cloud Enterprise) and its close partnership with IBM, there are more opportunities for the two vendors to work with Swedbank in the coming years.