HAZELCAST

The Hazelcast and Apache Kafka® Transaction Processing Reference Architecture

White Paper

White paper

The Hazelcast and Apache Kafka® Transaction Processing Reference Architecture

Introduction

Most computerized systems today follow a "request-response" pattern where an end client (either a human end user or another software application) submits a request to a remote system, which then runs a task that results in a response. Popular examples of this type of interaction include e-commerce purchases, lookups in a corporate application, or even the swipe of a credit card. We can broadly refer to these interactions as "transactions." In this context, a transaction is simply a 1-to-1 exchange of information. These transactions are not necessarily financially based, nor are they necessarily associated with the transactions that make up the operations of a relational database management system (RDBMS). What we are referring to in this paper is simply about receiving a request from the client, running a computation on that request, and then returning an output back to the client.

Oftentimes in today's digital and high-tech world, these transactions involve RDBMSs, or more generally, with systems that store data. These systems typically store data to disk drives or solid-state drives (SSDs), which represent a bottleneck in the throughput of the transaction. And as the volume of transactions grows, the impact of the bottleneck gets more pronounced.

Hazelcast is used to accelerate the performance of transaction-based systems that have stringent requirements around high throughput and low latency. Such systems can encounter loads of millions of transactions per second, especially due to load spikes. This means the system must run as quickly and efficiently as possible to deliver consistent response times at any load. The system must also offer elasticity to grow dramatically in anticipation of large seasonal load spikes. This performance acceleration is achieved by storing and accessing data in random-access memory (RAM), thus reducing accesses to the relatively slow disk- or SSD-based storage systems. Since RAM accesses are orders of magnitude faster (with high predictability) than disk-based systems, applications can get significant performance gains.

The Reference Architecture

In a high-speed transactional system based on the Hazelcast Platform, there are seven main components that work together to accept requests and then return responses:

- Apache Kafka
- Transaction streaming data
- Online applications
- Connectivity
- Ingest/transform engine
- In-memory storage
- Transaction processing application

In Figure 1, you can see how these system components interconnect in the transaction workflow, as well as how Hazelcast fits in. We will briefly discuss each of the components.

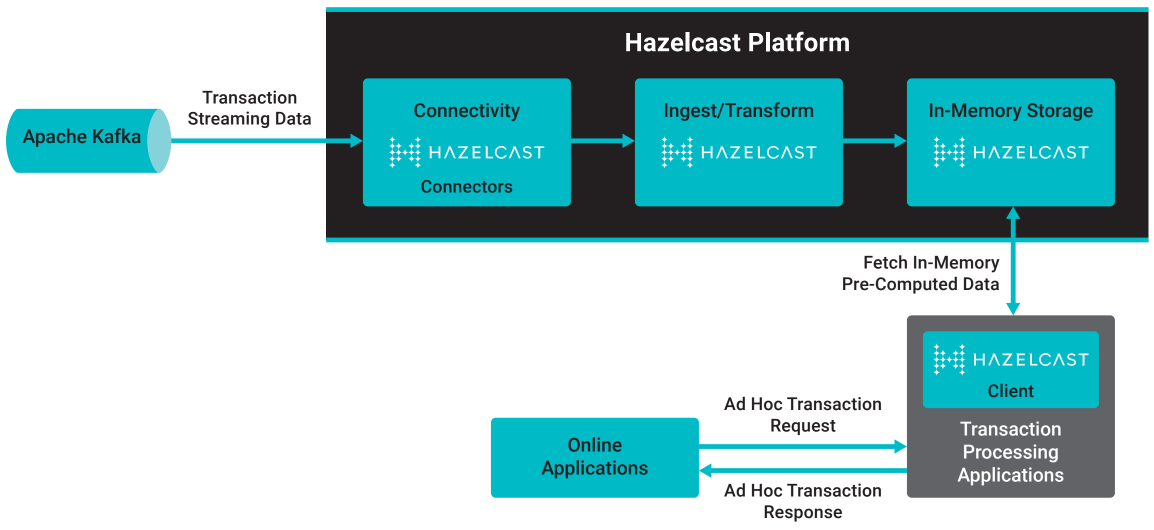
Hazelcast Platform
Apache Kafka
Transaction Streaming Data
Connectivity
HAZELCAST
Connectors
Ingest/Transform
HAZELCAST
In-Memory Storage
HAZELCAST
Fetch In-Memory Pre-Computed Data
HAZELCAST
Client
Transaction Processing Applications
Ad Hoc Transaction Request
Ad Hoc Transaction Response
Online Applications

Figure 1. Transactional System Reference Architecture

Apache Kafka®. In addition to Hazelcast, a key technology component in this transactional system is Apache Kafka, a popular messaging system that is ideal for storing incoming streams of data. Kafka is optimized to enable fast reads and writes of small blocks of data that capture details associated with updates or measurements, known as "events" in a streaming data context. Kafka is used as the event storage mechanism from the streaming data sources, to keep a record of all received data. In the architecture above, Hazelcast uses Kafka as the immediate data source to read in the transactional streaming data.

Hazelcast is especially useful as a transaction processing engine running alongside Kafka because of its speed advantages over similar technologies. The speed advantage lets you run more types of use cases, especially for real-time and near-real-time deployments, that require higher throughput and lower latency.

Kafka is deployed as a separate cluster from Hazelcast, and details on deploying Kafka are widely available online, especially at https://kafka.apache.org.

Transaction streaming data. This is the continuously incoming flow of data upon which action is taken on a real-time basis. In many cases, the transaction processing system (depicted by the Hazelcast Platform in the diagram above) takes automated action on this data like sending an alert, but it can also be aggregated and indexed into a data store to help generate responses to end client requests (in this context, the end client is synonymous with the "online applications" component in the diagram).

Examples of data sources include enterprise systems (RDBMSs, system log files, cloud applications, etc.), point-of-sale systems, Internet of Things devices/sensors, and social media feeds. These sources collectively make up an incoming data stream, which entails an ongoing and unbounded series of data points. In some sources like RDBMSs, a change notification system (i.e., "change data capture" or CDC) is used to deliver only the updates to a data set, essentially creating a data stream. In general, all streaming data should be modified and/or enriched by the "ingest/transform" component to get it into a format that can be used by the client-facing application.

Online applications. These are the end clients that submit the data that represents the requests, typically from an end user. As mentioned earlier, these could be e-commerce purchases, data lookups, or in-person purchases with a credit or debit card. These interactions are ad hoc transactions that the processing system serves (in addition to the continuous transactions described above).

Connectivity. Hazelcast connectors provide the interface to ingest data from the data sources. Many connectors are available out-of-the-box for connectivity to a wide variety of data sources, and custom connectors can be written using the connector API for other sources.

Ingest/Transform. This component is powered by Hazelcast to perform extract-transform-load (ETL) processing of the data stream from the transaction streaming data sources. The data can be cleansed, enriched, and then transformed into an aggregated format that can be used by the transaction processing application at very high speed. This pre-computation is critical in high-speed systems to reduce the computations that must be done on the request data coming from the online applications. Then the final data is saved to the in-memory store to enable high-speed lookups of that data. Hazelcast enables this component to be very fast and elastic, to handle the high volume and severe spikes seen in the streaming data sources.

In-memory storage. This is powered by the in-memory component of Hazelcast to provide fast data retrieval for other parts of the system. It stores all the data necessary for the transaction processing application to complete its work. The in-memory speed ensures the system can keep up with high transaction rates at extremely low latencies. Hazelcast leverages random-access memory (RAM) across multiple computers in a cluster, to accommodate data sets that normally would not fit into the RAM of a single computer.

Transaction processing application. The transaction processing application represents the core business logic of the system. It takes data in the form of requests from the online systems and calculates a response using data in the in-memory store. It uses Hazelcast client libraries to interface with the Hazelcast cluster to retrieve pre-computed calculations or aggregations or to retrieve data that enables other calculations that form the response. It then returns a response to the online applications to complete the transaction.

This application can also store metrics into the in-memory store. Other applications can then access the in-memory store to display those metrics in a dashboard or send alerts when specified thresholds are reached.

Deployment Factors

Hazelcast can be deployed on-premises or in the cloud. As a Java-based system, it can run on any hardware platform that supports a Java Virtual Machine (JVM), but is typically run on multi-core, x86-64-based Linux servers. With its “High Density” capability, Hazelcast can leverage much more RAM than is practical from allocated JVM memory. Specifically, this lets Hazelcast applications avoid performance-disrupting latencies of the heavy overhead of JVM garbage collection.

Hazelcast cluster sizes will vary depending on load and data size, and guidelines are available from Hazelcast in the sizing guides and deployment/operations guides. Some factors to consider for your deployment include:

Hazelcast Topology

The Hazelcast Platform integrates a compute engine with a storage engine, and both can be run together as a single cluster, sharing the hardware resources for both stream processing and in-memory storage. This makes deployment much simpler than maintaining two separate clusters. However, in environments where the processing load and the data volumes tend to grow independently, it sometimes makes sense to have separate compute and storage clusters (Figure 2). This decoupling of stream processing and storage lets you grow the separate clusters independently, to more efficiently add resources where they are most needed.

CPU Cores

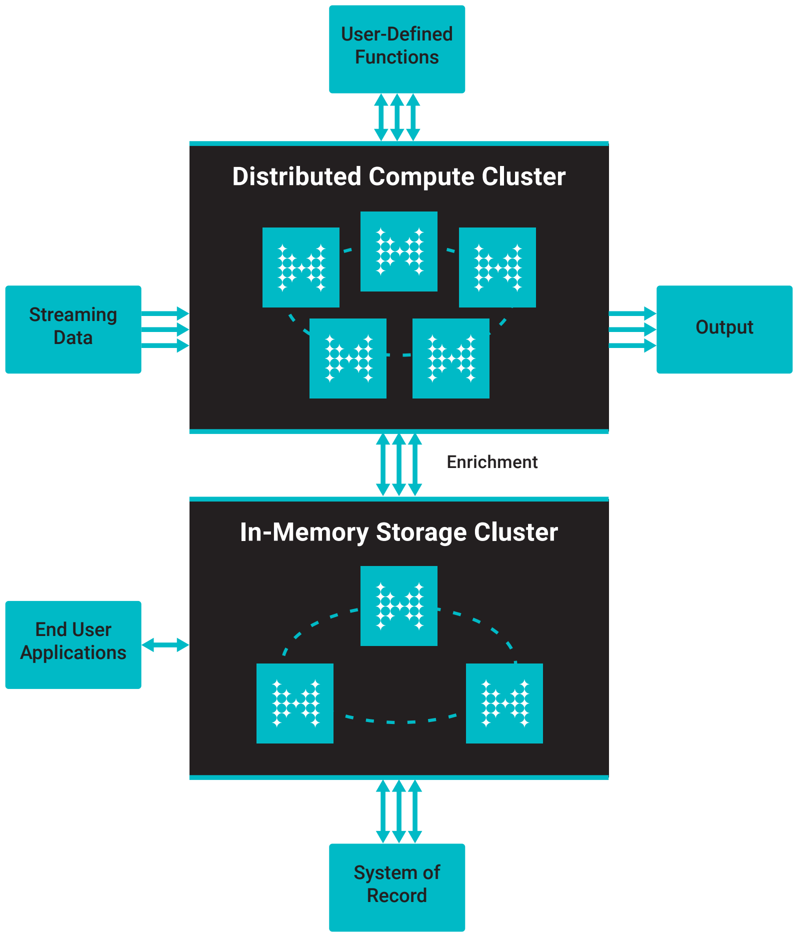

Figure 2. Separate compute and storage cluster topology

Hazelcast is architected to take advantage of multi-core servers, so you will gain more benefit with a higher core count in your cluster. The high core count is especially useful for compute-heavy workloads, which efficiently allocate tasks in a job across the many available cores. Cloud instances such as the AWS c4.8xlarge EC2 instances provide 36 virtual CPUs, which allow a high level of parallelism and significant throughput.

Memory

Hazelcast will take advantage of large pools of RAM in a cluster to store more data. The aforementioned AWS c4.8xlarge EC2 instances provide 60 GiB memory, which enables large in-memory data stores. Even larger (and more) servers/instances can be used with Hazelcast for larger stores. The amount of memory per server, and the total number of servers to be used by Hazelcast will depend on the amount of data you want instantly accessible. Factor in how much headroom you want to have in terms of extra memory above your data size. You should also consider the extra RAM needed to store backups, so if you have 200 GB of data, and you have configured two backups in your cluster, you should account for 600 GB of RAM, plus more RAM for your desired extra headroom.

As part of your deployment planning, you should consider the current trends in volatile memory technologies. For example, Intel® Optane™ persistent memory can be run in volatile memory mode, and supports larger capacities than RAM and at a far more economical price. Hazelcast is certified to use Intel Optane modules as an alternative to RAM. The combination of Hazelcast and Intel Optane enables a more cost-effective way to scale up to larger memory configurations.

Network

Having a fast network connection between cluster nodes will improve the overall performance by reducing the network bottleneck. Having a 25GbE or faster network connection can move your clusters to the next level of performance. Some optimizations in Hazelcast such as the near-cache feature will reduce network hops and thus mitigate the network bottleneck.

About the Hazelcast Platform

The Hazelcast Platform is an open source software technology that lets you build ultra-fast, zero-downtime applications. It was designed for data processing environments that require the highest throughput and lowest latency. The key Hazelcast advantages include:

Simplified development. Hazelcast was designed to enable easier development of applications that require both streaming and batch processing workloads, in a lightweight deployment. There are fewer moving parts compared to other processing environments, thus reducing the overhead of management and maintenance. It has no external dependencies to reduce the complexity of integration into your existing data architecture. Packaged as a small JAR file, it can be embedded in applications for extremely simple deployments. And it is deployable anywhere, including in small hardware footprints at the edge, as well as in on-premises data centers and the public cloud.

Microsecond performance. Hazelcast offers superior speed versus comparable data processing technologies as demonstrated in published benchmarks. It provides a scale-out architecture that enables you to add commodity hardware servers to your cluster to seamlessly scale to terabytes of in-memory data. It is useful for extremely demanding, data intensive environments, but is also valuable when infrastructural complexity is a concern, as its extreme efficiency allows more work with less hardware resources.

Mission-critical reliability. Hazelcast customers aim for zero downtime, and the Hazelcast Platform provides all the capabilities you need to reliably and securely run an enterprise deployment. It offers a complete business continuity system to ensure environmental failures do not disrupt operation. Its built-in replication ensures that a cluster remains operational, and no data is lost, even when hardware failures occur. The multi-site replication feature ensures another live cluster is available to safeguard against site-wide failures, and automatic failover built into client libraries provide seamless failover. Optimizations in the multi-site replication system allow active-active and active-passive topologies with minimized network traffic. Hazelcast also offers several security features including pluggable authentication, role-based access controls, encryption on data-at-rest and data-in-motion (between all clients and nodes), hooks to add custom security features, and optimized TLS to retain high speed in secure data transmissions.

About Hazelcast

Hazelcast is the fast, cloud application platform trusted by Global 2000 enterprises to deliver ultra-low latency for stateful and data-intensive applications. Featuring real-time streaming and memory-first database technologies, the Hazelcast platform simplifies the development and deployment of distributed applications on-premises, at the edge or in the cloud as a fully managed service.

Hazelcast is headquartered in San Mateo, CA, with offices across the globe. To learn more about Hazelcast, visit https://hazelcast.com.