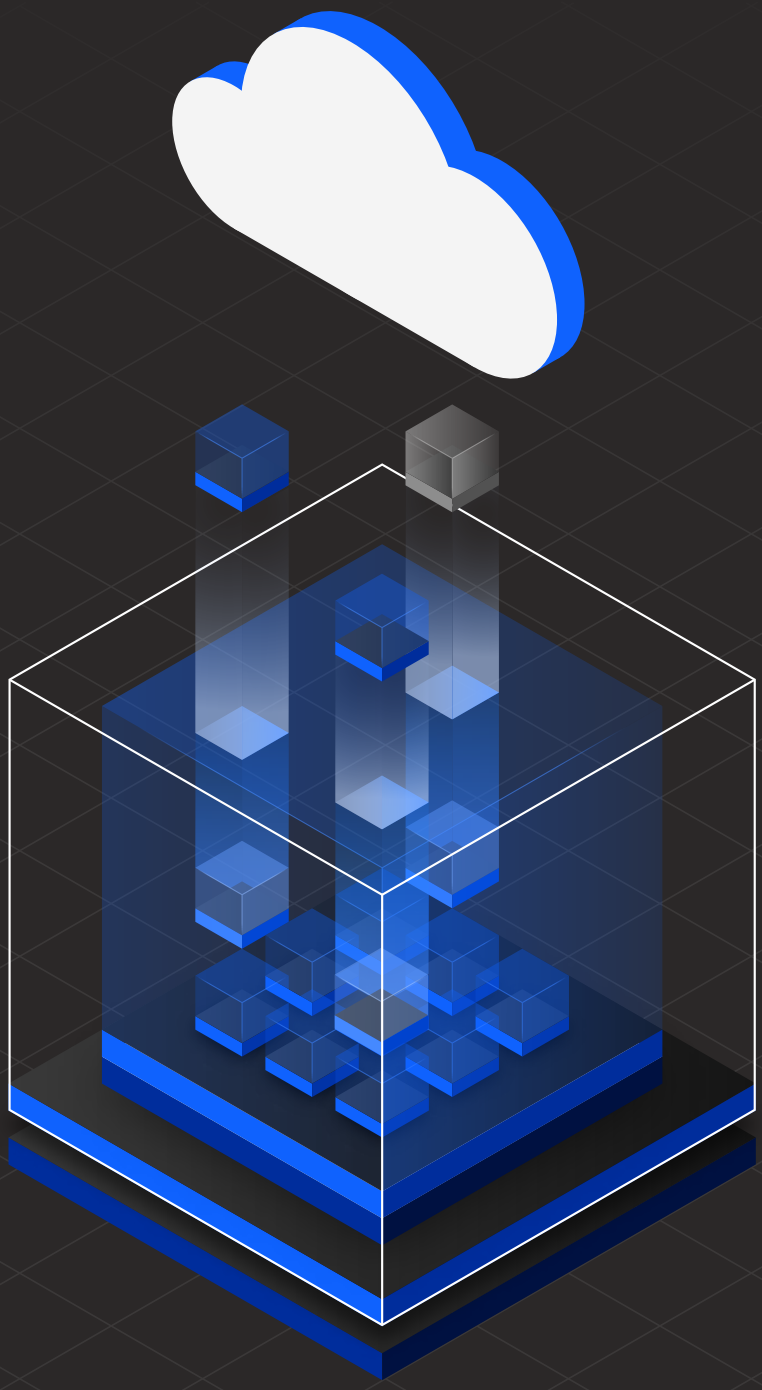
IBM Cloud

Migrating to in-memory computing for modern, cloud-native workloads

WebSphere eXtreme Scale and Hazelcast In-Memory Computing Platform for IBM Cloud® Paks



Introduction

Historically, single instance, multi-user applications were sufficient to meet most commercial workloads. With data access latency being a small fraction of the overall execution time, it was not a primary concern. Over time, new requirements from applications like e-commerce, global supply chains and automated business processes required a much higher level of computation and drove the development of distributed processing using clustered groups of applications.

A distributed application architecture scales horizontally, taking advantage of ever-faster processors and networking technology, but with it, data synchronization and coordinated access become a separate, complex system. Centralized storage area network (SAN) systems became common, but as computational speeds continued to advance, the latencies of disk-based storage and retrieval quickly became a significant bottleneck. In-memory data storage systems were developed to provide distributed, fault-tolerant, all-in-memory data access, which is orders of magnitude faster than disk-based systems.

Moving to cloud

Mobile technology, ubiquitous networks, automated "smart" devices and myriad other new technologies have changed the way users and applications interact with each other. These technologies are constantly connected and always on, sending infinite streams of unique data events and expecting a real-time response. These massively parallel, high-speed data interaction requirements are fueling the migration of applications to cloud-native architectures.

Cloud-based application installations, using virtualization technology, offer the ability to scale environments dynamically to meet demand peaks and valleys, optimizing computation costs to workload requirements. These dynamic environments further drive the need for in-memory data storage with extremely fast, distributed, scalable data sharing. Cloud instances come and go as load dictates, so immediate access to data upon instance creation and no data loss upon instance removal are paramount. Independently scaled, in-memory data storage meets all of these needs and enables additional processing capabilities as well.

Data replication and synchronization between cloud locations enable globally distributed cloud environments to function in active/active configurations for load balancing and fault tolerance. These capabilities require the fastest transfer speeds possible. With advanced networking technology and memory-based data storage, previously impossible transfer throughput with extremely low latency can be achieved.

As organizations begin application modernization projects and migrations from on-premises installations to cloud architectures, the infrastructure must also evolve to support cloud-native capabilities. IBM Cloud Pak® solutions are software stacks that help businesses deploy enterprise systems into any cloud environment. IBM Cloud Paks are built on Red Hat® OpenShift®, Kubernetes, containers and supporting software to run enterprise applications.

Migrating to cloud-native in-memory computing

IBM WebSphere® eXtreme Scale is an elastic, scalable in-memory data grid designed for on-premises environments and does not support cloud environments using Kubernetes and containers. To support the migration of applications to the cloud, organizations will need to migrate to a solution that supports cloud-native environments to meet application performance expectations.

Hazelcast In-Memory Computing Platform

Hazelcast In-Memory Computing Platform for IBM Cloud Paks is a fast, flexible and cloud-native distributed application platform that is certified for deployment with IBM Cloud Paks and is comprised of two projects/components:

- Hazelcast In-Memory Data Grid (IMDG) for in-memory data storage and data-local distributed processing
- Hazelcast Jet for high throughput, low latency stream processing

The platform supports application modernization projects with capabilities designed for cloud deployments. Organizations can expand innovation for new applications through advanced features and capabilities not found in other in-memory storage technology. Containerization and orchestration management are fully supported on many platforms, including Red Hat OpenShift.

At a high level, Hazelcast and eXtreme Scale both use clusters of servers with partitions (shards) of memory distributed across all the cluster members (nodes). With both approaches, each node is the primary server for the data contained in partitions owned by that node, with backup replicas created on other nodes for fault tolerance. Failure detection and recovery are automatic. Each system scales dynamically by adding or removing nodes to the cluster.

The Hazelcast solution from IBM starts with the distributed data grid, Hazelcast IMDG, and adds Hazelcast Jet, the component that is a state-of-the-art, real-time stream processing engine. Jet natively leverages all of the capabilities found in the Hazelcast in-memory storage layer, adding advanced stream processing capabilities to meet today's newest workloads. For example, built-in time window management correlates event streams for continuous value aggregations and calculations. Streams can be merged with other streams and enriched with data from other sources. And events can be transformed from one form to another, among other capabilities, all in real time.

Machine learning (ML) has also brought capabilities that were previously impossible for automated computer processing systems. Operationalizing this technology in production, however, can be extremely complex and time consuming. Hazelcast solves this problem by enabling ML models to be run directly within the stream processing engine with "inference runners." These runners allow Java, Python and C/C++ ML models to run in real time, taking advantage of all the in-memory, distributed capabilities of the Hazelcast solution. New versions of ML models can be loaded to replace older versions without downtime.

In summary, the Hazelcast In-Memory Computing Platform for IBM Cloud Paks combines an innovative in-memory data storage layer with its third-generation stream processing engine to create an advanced in-memory processing platform.

Technology comparison: Hazelcast and eXtreme Scale

Operational features

Table 1 lists supported cluster-wide operational features that contribute to ease of use, fault tolerance, scaling and flexibility for both platforms.

	Hazelcast 4.2	eXtreme Scale 8.6
Topology	App-embedded client/server	WebSphere client/server
Installation platform	Bare metal, virtualized containers, Kubernetes, cloud	Bare metal, virtualized
Cluster replication	•	•
Elastic	•	•
100 GB per Instance	•	•
Elastic	•	•
Fault tolerance	•	•
Disk persistence	•	•
DB persistence	•	•
Certified for IBM Cloud Paks	•	•
Docker	•	
Kubernetes	•	
Eureka	•	
Apache jclouds	•	
Cloud native	•	
OpenShift	•	
Split brain protection	•	
Quorum	•	•
Serialization	Multiple built-in and third party	Java and XDF
JSON Support	•	

Table 1. Operational features comparison for Hazelcast 4.2 and eXtreme Scale 8.6

Distributed structures

The CAP theorem states simply that a distributed system must be able to tolerate a network partition (P) and can be either available (A) or consistent (C), but not both (i.e., AP or CP). eXtreme Scale and traditional Hazelcast structures are all of type AP. Hazelcast recently implemented a CP subsystem using the RAFT protocol to offer optional CP distributed data structures. These are noted in the Distributed Structures (CP) table.

	Hazelcast 4.2	eXtreme Scale 8.6
Map	•	•
Multi-Map	•	Bare metal, virtualized
Replicated Map	•	•
Set	•	•
List	•	•
Queue	•	•
Reliable Topic	•	•
Topic	•	•
Ring Buffer	•	•
Flake ID Generator	•	•
CRDT PN Counter	•	

Table 2. AP-style distributed data structures supported by Hazelcast 4.2 and eXtreme Scale 8.6

Distributed structures (CP)

CP-style distributed data structures, as noted earlier, are guaranteed to maintain data consistency across their copies on a configurable number of grid nodes. This approach is necessary for some use cases that require certain guarantees, such as uniqueness or monotonic progression, among others. See Table 3 for supported CP-style data structures.

Distributed structures (CP)

CP-style distributed data structures, as noted earlier, are guaranteed to maintain data consistency across their copies on a configurable number of grid nodes. This approach is necessary for some use cases that require certain guarantees, such as uniqueness or monotonic progression, among others. See Table 3 for supported CP-style data structures.

	Hazelcast 4.2	eXtreme Scale 8.6
Fenced Lock / Sem	•	
Atomic Long	•	
Count Down Latch	•	
Atomic Reference	•	

Table 3. CP-style distributed data structures supported by Hazelcast 4.2 and eXtreme Scale 8.6

Distributed computation

Distributed, in-memory data storage allows various calculations to be performed across data sets held in the storage layer. For example, aggregating values held in particular stored entities produces a total of the values. This takes advantage of the parallel, distributed nature of the storage layer for performance. Additionally, event-driven computations, such as continuous query and entry processors, can be executed automatically when data is modified. See Table 4 for supported calculations.

	Hazelcast 4.2	eXtreme Scale 8.6
Continuous Query	•	•
HyperLogLog	•	
SQL Query	•	•
Predicate Query	•	
Entry Processor	•	
Executor Service	•	
Aggregation	•	

Table 4. Types of distributed computation supported by Hazelcast 4.2 and eXtreme Scale 8.6

Clients

Hazelcast and eXtreme Scale are both built on Java. They can both run embedded within another Java application or in client/server mode, where the data layer runs independently and applications access the data via clients. Clients are built in various programming languages to provide access to a wide variety of applications. This allows applications built on different language platforms to easily share data. Table 5 presents the programming languages supported by Hazelcast and eXtreme Scale.

	Hazelcast 4.2	eXtreme Scale 8.6
Java	•	•
Scala	•	
C++	•	
C#/.Net	•	•
Python	•	
Node.js	•	
Go	•	
REST	•	• (Deprecated)
Memcached	•	
Clojure	•	
Smart Client Routing	•	

Table 5. Client language platforms supported by Hazelcast 4.2 and eXtreme Scale 8.6

API providers

Standard APIs have been defined for generic use in common applications such as web servers and application servers. Platforms that support those APIs, shown in Table 6, can seamlessly implement underlying storage for applications using those APIs.

	Hazelcast 4.2	eXtreme Scale 8.6
Web sessions	•	•
Hibernate	•	
JCache	•	•

Table 6. APIs supported by Hazelcast 4.2 and eXtreme Scale 8.6

Security

Authentication and authorization must be enforced by any data storage layer to protect against unauthorized access. Ease-of-use, the flexibility of infrastructure integration, and adhering to current standards significantly improve security manageability and overall strength. This is especially important when using multiple cloud vendors' security implementations. Table 7 defines security standards supported by Hazelcast 4.2 and eXtreme Scale 8.6.

	Hazelcast 4.2	eXtreme Scale 8.6
JAAS	•	•
LDAP	•	• (Using WebSphere)
Kerberos	•	• (Using WebSphere)
Grid security	Sym, Cert, Kerb	Shared password
Structure-level security	•	Static group access (restart required)
Field-level security	•	
FIPS-140-2	•	

Table 7. Security standards supported by Hazelcast 4.2 and eXtreme Scale 8.6

Hazelcast in-memory stream processing

The Hazelcast solution from IBM also provides an advanced stream processing engine running directly on the in-memory data grid technology. Event-stream in-memory processing provides real-time event processing with extremely high throughput and low latency. In a recent benchmark, Hazelcast processed 1 billion events per second with a 99th percentile of 26 milliseconds on only 45 AWS cloud instances (720 cores). This benchmark demonstrates a level of efficiency that reduces operational costs due to fewer hardware resources.

Stream processing requires capabilities beyond storage and distributed processing. Multiple event sources must be received and correlated, enrichment data must be added "in flight," aggregation, correlation and computation must be carried out within defined time windows, and processing guarantees must be enforced. eXtreme Scale does not offer stream processing capabilities. Hazelcast stream processing features follow.

Stream processing

Real-time stream processing requires important additional capabilities beyond simply receiving events. The platform must automatically implement various time-window frameworks to correctly perform calculations such as aggregation, value deviation and threshold violations. Processing guarantees, such as exactly-once processing, must be upheld, along with enterprise operational features like job snapshots for error correction and zero-downtime upgrades. As Table 8 demonstrates, the Hazelcast Platform implements all necessary functionality for enterprise-grade, real-time stream processing.

Installation platform	Bare metal, virtualized containers, Kubernetes, cloud
OpenShift	Full support for OpenShift, operators
Kubernetes	Full support for Kubernetes
Streaming SQL	Real-time SQL queries on streaming data including the ability to join with reference data stored in IMDG
Time-window management	Sliding, tumbling, session
Event-time processing	Native, inserted or calculated
Back pressure flow control	Automatic back pressure propagation
Event-ID correlation	Hash-join multiple streams or events on ID
Elastic, scalable	Fully elastic scaling
API support	Java Pipeline, DAG, Apache Beam

Table 8. Functionality requirements for enterprise-grade, real-time stream processing

Event processing guarantees

Hazelcast is an enterprise-grade platform with support features for DevOps teams managing the platform. Examples (see Table 9) include fault tolerance with automatic error detection and correction, running job snapshots for resilience, intelligent load balancing of workloads across the cluster, and live in-place job upgrades.

Exactly once or at least once	Jet supports distributed state snapshots. Snapshots are periodically created to back up the running state. Periodic snapshots are used as a consistent point of recovery for failures. Snapshots are also taken and used for upscaling.
Exactly once	Jet ensures exactly-once semantics when a replayable source (e.g., Kafka) is used with an idempotent sink (e.g., any store with upsert functionality).
Two-phase commit for exactly once	Jet supports distributed transactions to enable exactly once guarantees on sources and sinks to participate in transaction-based streaming. If the source/sink supports the two-phase commit protocol, Jet will leverage it. Otherwise, Jet tracks the transaction state to guarantee exactly-once semantics even if the source is not replayable or the sink is not idempotent.
Fault tolerance	If there is a fault, Jet uses the latest state snapshot and automatically restarts all jobs that contain the failed member as a job participant from this snapshot.
Resilient snapshot storage	Jet uses the distributed in-memory storage to store snapshots.
Replay capable	When a replayable source is used, Jet can rewind and reprocess events from the source.

Table 9. Built-in capabilities that make Jet easily manageable by DevOps teams.

Connectors

The Hazelcast solution has a library of connectors through which the system can send and receive events. This catalog is continuously updated with new connectors to meet user community needs.

Events can be received through one connector and sent through another. Event streams from various connectors can also be combined. Table 10 lists the connectors supported at the time of writing.

Hazelcast IMDG	Jet is integrated with IMDG elastic in-memory storage, providing a highly optimized read and write memory channel.
Kafka	Jet comes with a Kafka connector for reading from and writing to the Kafka topics.
Java Message Service (JMS)	The Jet JMS connector allows you to stream messages from/to a JMS queue or a JMS topic using a JMS client on a classpath (such as ActiveMQ or RabbitMQ).
Java Database Connectivity (JDBC)	Jet JDBC Connector can be used to read or write the data from/to relational databases or another source that supports the standard JDBC API.
Hadoop Distributed File Systems (HDFS)	Hazelcast Jet can use HDFS as either a data source or data sink. If Jet and HDFS clusters are co-located, then Jet benefits from the data locality and processes the data from the same node without incurring a network transit latency penalty.
Avro	Jet can read and write Avro-serialized data from the self-contained files (Avro Object Container format), HDFS and Kafka. A Kafka connector can be configured to use the schema registry.
Local files	Jet comes with batch and streaming file readers to process local data (e.g., CSVs or logs). The batch reader processes lines from a file or directory. The streamer watches the file or directory for changes, streaming the new lines to Jet.
Sockets	The socket connector allows Jet jobs to read text data streams from the socket. Every line is processed as one record.
Custom source/sink	Jet provides a flexible API that makes it easy to implement your own custom sources and sinks.
Kafka Connect modules	Jet supports the use of any Kafka Connect module without the presence of a Kafka cluster. This adds more sources and sinks to the Jet ecosystem. This feature includes full support for fault tolerance and replaying.

Table 10. Hazelcast Jet supports a wide range of connectors for in-stream processing

Security

The Hazelcast solution offered by IBM is deployed in banking, government, healthcare and many other high-security environments and meets stringent certification requirements across these systems. Leveraging the strength of the data storage layer security, Jet adds additional security to the communication layer for end-to-end encryption and access control (see Table 11).

SSL/TLS 1.2 asymmetric encryption	Hazelcast provides encryption based on TLS certificates between members, between clients and members, and between members and the management center.
SSL/TLS 1.2 asymmetric encryption with OpenSSL	Hazelcast adds some performance enhancements to the SSLEngine built into the Java Development Kit (JDK).
Secure connectors	Connectors are used to connect the Jet job with data sources and sinks. Secure connections to external systems combined with security within the Jet cluster make the data pipeline secure end to end.
Allowed connection IP ranges	Whitelist specific IP ranges to limit connections to known hosts.
Authentication	The authentication mechanism for Hazelcast client security works the same as cluster member authentication. Hazelcast includes out-of-the-box integration with Lightweight Directory Access Protocol (LDAP) and Kerberos.
Authorization	Role-based access control (RBAC) is supported internally by default or can use external mechanisms such as LDAP.
Symmetric encryption	Symmetric encryption uses a single pre-shared key for simple installations.
Java Authentication and Authorization Service (JAAS) module	Hazelcast has an extensible, JAAS-based security feature used to authenticate both cluster members and clients and to perform access control checks on client operations.
Pluggable socket interceptor	Hazelcast allows you to intercept socket connections before a nodejoins to a cluster or a client connects to a node. This provides the ability to add custom hooks to join and perform connection procedures such as identity checking using Kerberos.
Security interceptor	IMDG allows you to intercept every remote operation executed by the client. This lets you add a flexible custom security logic.

Table 11.

Summary

Hazelcast In-Memory Computing Platform for IBM Cloud Paks runs in virtually any environment: bare metal, containers, VM, mainframe and cloud. The data storage layer, IMDG, enables high-performance data access and scalability for e-commerce, elastic workload management and distributed calculations. For data in motion, Jet enables real-time event stream analysis for mobile, IoT, edge and machine learning applications. Certified for IBM Cloud Pak deployment, the Hazelcast solution from IBM supports the performance needs for application modernization projects moving to the cloud.

To learn more about the Hazelcast In-Memory Platform for IBM Cloud Paks, visit ibm.com/cloud/hazelcast.

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
May 2021