Comparing Redis Enterprise Real-Time Data Platform and Hazelcast In-Memory Data Grid

The digital economy calls for real-time data processing to provide the customer experience and data driven decision making needed by all industries. Applications and the associated data infrastructure are hindered by the poor performance and scalability of legacy databases. Speed is just the beginning, reliability, scalability, and support for multiple data models are also necessary capabilities for a true real-time data platform that can meet business-critical enterprise application needs.

Redis Enterprise, built on top of open source Redis, provides a scalable, in-memory multi-model data platform that keeps pace with these demands at a lower cost than In-Memory Data Grids (IMDGs). It does so with better performance, sub-millisecond latency, high availability, scalability, and more extensibility. Redis is one of the most popular databases, ranked 6th on DB-Engines rankings (second among NoSQL databases). Hazelcast is ranked 48th, as of September 2022.

Hazelcast Enterprise is the commercial version of open source Hazelcast. Hazelcast is primarily used as a compute platform to provide fast, stateful, and data intensive workloads for Java applications.

In-Memory Data Grids based on Java

In-memory data grids (IMDGs) came into existence in the 2000s as a way to implement faster data access and to co-locate compute with application logic. Primarily written in Java, IMDGs were positioned as a plugin or suite of Java middleware application servers. IMDGs store data in memory across many servers by automatically and dynamically partitioning data, in a highly distributed (i.e. parallelized) fashion. This increases performance by moving data closer to end users for efficient access.

Java programs require a Java Virtual Machine (JVM) environment on any platform that can then execute the compiled bytecode. JVMs' core strength is its 'Write Once Run Anywhere' principle; Java hides the specifics of the underlying platform, which enables a hosted application to move easily to another environment. However, the JVM overhead adds tradeoffs on deployment complexity and performance. In contrast, Redis is written in C, a native language that is compiled and optimized to match the specific platform.

Considerations:

- **License Cost:** Java requires a license which must be factored in the overall costs of IMDGs..
- **Performance:** JVM memory management impacts performance (especially during Garbage Collection). Redis is written from the ground up with optimized memory management.
- **Complexity**: Java-based implementations often come with serialization choices adding complexity, including additional overhead for the client API and server side engine to deserialize fields on queries to match on object fields. Serialization can also add additional complexity for object versioning. By contrast, Redis provides native core data structure, as well as native JSON, at lower serialization cost.
- **Memory management:** JVM-based IMDGs provide an "off heap" capability which add further costs to manage large memory systems. They handle "off-heap" as a generic RAM or storage problem. Redis on Flash tiered storage is purposely designed for NVMe architectures. With specific multi-thread Flash-aware access, Redis Enterprise is able to deal with large amounts of data, further reducing the amount of memory tuning.
- **Development Expertise:** Limited talent pool with knowledge and expertise in Hazelcast

With Redis Enterprise, you benefit from:

- **Simple to build and deploy**
- **Performance at scale**
- **Fault tolerance, resilience, and high availability** with 99.999% service level agreements (SLAs)
- **Global Active-Active distribution** that maintains real-time local speed and data consistency
- **Extensibility with multiple data structures and models** including search and JSON
- **Lower TCO** with intelligent data tiering and resource efficiency
- **Flexible deployment options** across on-premises, hybrid, and multi cloud including managed services offerings on AWS, Azure, and GCP

Which use cases fit better with Redis Enterprise?

- Speed application performance with enterprise caching
- Session management and user profile store to power customer 360
- Fraud detection requiring sub-millisecond latency and high throughput
- Polyglot microservices architecture that takes advantage of Redis multi-model capabilities (search, JSON, time series) and extensive language support
- Always on, globally distributed applications and data
- DevOps automation, Kubernetes deployments, and Infrastructure as Code requirements

Feature Comparison

		Hazelcast Enterprise	Redis Enterprise
Designed for performance and scalability	Throughput/Latency	190K ops/sec tput @ 0.9ms latency	715K ops/sec tput @ 0.3ms latency
	Development language	Java JVM can limit performance due to thread syncs, garbage collection, etc.	C Runs natively as a Linux process
	Multi-threaded architecture	○ Added instruction complexity and requires the use of (fenced) locks	● Multi-threaded for IO, single-threaded for data operations
	On-demand scalability		● Add more shards automatically without downtime.
Simplified developer and DevOps experience	Number of client libraries/ SDKs supported	7	>70
	Multi-use case extensibility		● Extensive set of data types plus Search, JSON, and more
	Supports advanced Kubernetes Operator		● Level III Full Lifecycle with automatic scalability. Available on OpenShift, VMware Tanzu, GKE, AKS, EKS, Rancher, and any k8s OSS distros
	Vibrant community support		● Most loved database on stackoverflow developer survey five years in a row
Enterprise-grade availability and resilience	High availability and disaster recovery	○ Reduced performance during node failure and migration	● Automatic cluster recovery, data persistence, and automatic backup
	Automated Active-Active geo replication using Conflict-Free Data Types (CRDT)		●
Total cost of ownership	Intelligent Enterprise tiered storage		● Redis on Flash
	Optimized storage density		● Stores significantly more data per gigabyte of storage. Hazelcast stores metadata about serialized objects
Deployment flexibility	Multi- and hybrid cloud support		● Managed DBaaS on AWS, GCP, and Azure. plus Kubernetes

Don’t take our word for it. Just ask our customers

“We have access to the highest in-memory performance available on the market today, flexible data structures for extreme efficiency across a wide variety of use cases, and fully-managed operations that speed up, rather than slow down, application delivery.”

–Naren Janakiraman, director of engineering, Freshworks

View Case Study

Ready to upgrade to a simple to deploy in-memory data platform that goes beyond what Hazelcast offers?

Try for free

Watch demo

Schedule a meeting with Redis today