Hazelcast 开源项目架构分析报告

需求分析 • 核心流程 • 高级设计 • 扩展点建模

2023K8009937008 王俊峰

2025 年 10 月 30 日

目录

1 需求分析与建模

Hazelcast 是一个**分布式内存数据网格（In-Memory Data Grid, IMDG）**和**实时数据处理平台**，其核心需求源于现代分布式应用对**低延迟、高可用、弹性扩展**的数据存储与计算能力的需求。通过对 GitHub 仓库 `https://github.com/hazelcast/hazelcast` 分析，其需求可划分为功能性与非功能性两大类，并通过领域建模进行结构化抽象。

1.1 功能需求

- **分布式数据结构:** 支持 `IMap`（键值存储）、`ICache`（JCache 兼容）、`IQueue/IList/ISet`（队列/列表/集合）、`Ringbuffer`（环形缓冲）、`Topic`（发布订阅消息）。
- **实时流处理：**集成 Hazelcast Jet 引擎，支持通过 SQL 或 Java API 构建数据处理管道（`source` → `transform` → `sink`），支持流式与批处理统一，单集群可聚合超过 10M+ 事件/秒。
- **查询与计算：**内置 SQL 服务支持对静态与流式数据的查询；`EntryProcessor` 允许在数据所在节点直接执行计算逻辑。
- **集群协调：**CP 子系统（基于 Raft 协议）提供分布式锁、`Semaphore`、领导选举等强一致性原语。
- **系统集成：**`MapStore/MapLoader` 实现与外部存储的读写穿透；WAN 复制支持跨数据中心异步复制。

1.2 非功能需求

表 1: 非功能需求指标

类别	需求描述	指标示例
性能	低延迟读写、弹性扩展	$<$ 10ms 延迟，亿级事件/秒
可用性	自动故障转移、备份复制	N 备份，零数据丢失
可扩展性	动态加减节点，自动再平衡	水平扩展至数千节点
一致性	AP/CP 模式可选	AP：最终一致；CP：线性一致
安全性	传输加密、身份认证	TLS、JWT、LDAP 等

1.3 领域建模

1.3.1 实体模型（类图抽象）

`Cluster`

```
Node (HazelcastInstance)
    PartitionTable (271 分区)
    Services (PartitionService, JetEngine, SQLService)
DataStructure (Proxy: IMap, IQueue...)
    Partition (Primary + N Backups)
JetPipeline (Source → Operators → Sink)
```

1.3.2 用例模型

表 2: 核心用例模型

参与者	用例	前置/后置条件
Developer	创建集群实例	配置 YAML/XML → 自动发现节点
Application	put/get 数据	哈希键 → 分区路由 → 备份同步
Operator	部署 Jet Job	SQL/API → 弹性执行 → Sink 输出
Administrator	监控/再平衡	Heartbeat → 故障检测 → 迁移

该建模确保系统具备**解耦**（代理模式）与**分区负载均衡**两大核心能力。

2 核心流程设计分析

Hazelcast 采用 **P2P 无中心架构**，所有节点平等，通过**分区表**（Partition Table）实现请求路由。核心代码模块位于 `hazelcast/src/main/java/com/hazelcast/`，主要包括：

- `internal/partition`：分区管理
- `map`：IMap 实现
- `spi`：服务提供者接口

2.1 关键流程详解

2.1.1 1. 集群启动与节点加入（Node Join Flow）

```
Hazelcast.newInstance()
    → Discovery SPI (Multicast/TCP/K8s)
    → Heartbeat Gossip 协议
    → Sync PartitionTable
    → Rebalance Data
    → Ready
```

- 完成时长：5–10 秒
- 核心类：`internal/cluster/ClusterService`

2.1.2 2. 数据读写流程（Get/Put Flow）

```
IMap.put(key, value)
    → Proxy → hash(key) % 271 → Find Owner Node
    → Local: Direct Write + Backup Async
    → Remote: Send Operation → Ack from Backups → Return
```

- 优化手段：Near Cache（本地缓存）、EntryProcessor（原地计算）
- 核心类：`map/impl/MapService`、`internal/partition/operation/`

2.1.3 3. Jet 流处理管道（Pipeline Flow）

```
Pipeline p = Pipeline.from(...)
    → addSource → groupAndAggregate → writeTo(Sink)
    → instance.getJet().newJob(p)
    → Elastic Execute → Fault-Tolerant Snapshot
```

- 核心模块：`jet/`

2.1.4 4. 故障恢复流程（Failover Flow）

```
Heartbeat Timeout
    → FailureDetector
    → Promote Backup
    → Migrate Partitions
    → WAN Replication（可选）
    → Consistency Repair
```

- 实现零 RTO（Recovery Time Objective）
- 核心类：`internal/partition/impl/PartitionStateManager`

所有核心流程均采用**异步非阻塞**模型（基于 `CompletableFuture`），确保高吞吐与低延迟。

3 高级设计意图分析

Hazelcast 的设计目标是构建一个**统一的实时数据平台**，解决传统 Lambda/Kappa 架构的复杂性，强调**弹性**、**可观测性**与**零运维**。其高级设计意图体现在以下方面：

- **固定 271 分区 + 备份机制:**实现 O(1) 路由与自动容错,核心类 `PartitionService`。
- **代理模式（Proxy Pattern）:** `IMap` 等数据结构为轻量级代理，委托至 `MapService`，实现用户 API 与内部实现的解耦。
- **SPI 驱动的插件化扩展：** `spi/` 包定义服务接口，通过 `META-INF/services` 自动加载工厂。
- **AP/CP 能力分离：**
  - 默认 AP 模式（最终一致，适用于缓存）
  - CP 子系统（Raft 实现，适用于锁、领导选举）
- **无状态服务 + 有状态 Jet 执行：**分布式执行服务 + 快照恢复机制，支持精确一次（exactly-once）语义。
- **模块化架构：**
  - `hazelcast`：核心 IMDG
  - `jet`：流处理引擎
  - `client`：客户端协议

  支持 Maven 多模块构建，便于按需裁剪部署。

总体意图：实现**云原生**、**自动运维**的实时数据平台，支持从缓存到流处理的完整场景。

4 扩展点的需求分析与建模

Hazelcast 通过 **SPI（Service Provider Interface）**机制提供强大的扩展能力，所有扩展点通过 `META-INF/services/` 注册工厂实现热插拔。本节聚焦**非功能优化**领域的典型扩展点：**自定义驱逐策略（Eviction Policy）**。

4.1 非功能优化：自定义驱逐策略（Eviction Policy）

4.1.1 需求描述

- **场景**：大对象缓存场景，内存紧张，需防止 OOM。
- **目标**：实现基于 LRU/LFU + 业务权重的混合驱逐策略，提升缓存命中率。
- **非功能指标**：命中率提升 20% 以上，支持动态调整策略。

4.1.2 建模设计

```
EvictionPolicyProvider
    → newEvictionPolicy(key, entry)
    → return CustomPolicy(LFUCount + Weight + TTL)
```

配置示例：

```
<map name="large-cache">
    <eviction policy-provider="com.my.MyEvictionProvider"/>
    <max-size policy="PER_NODE">10000</max-size>
</map>
```

图 1: 自定义驱逐策略建模

4.1.3 扩展收益

- **零侵入**：无需修改 Hazelcast 主项目代码
- **热插拔**：运行时动态加载新策略
- **业务适配**：支持按业务权重（如用户等级、对象热度）定制驱逐逻辑

5 总结

Hazelcast 作为一个成熟的分布式内存计算平台，通过清晰的**分区 + 代理 + SPI** 架构，实现了高性能、高可用与强扩展性的统一。以上内容从需求建模到核心流程、高级设计意图，再到关键扩展点，分析了其技术内核。

核心价值：

- 统一实时处理平台（IMDG + Stream）
- 云原生零运维能力
- 强大的 SPI 插件生态