

极客时间 Java 进阶训练营

第 23 课



分布式缓存-Redis 高可用/Redisson/Hazelcast

KimmKing

Apache Dubbo/ShardingSphere PMC

个人介绍

Apache Dubbo/ShardingSphere PMC

前某集团高级技术总监/阿里架构师/某银行北京研发中心负责人

阿里云 MVP、腾讯 TVP、TGO 会员

10 多年研发管理和架构经验

熟悉海量并发低延迟交易系统的设计实现

目录

1. Redis 集群与高可用

2. Redisson 介绍

3. Hazelcast 介绍

4. 总结回顾与作业实践

1. Redis 的集群与高可用

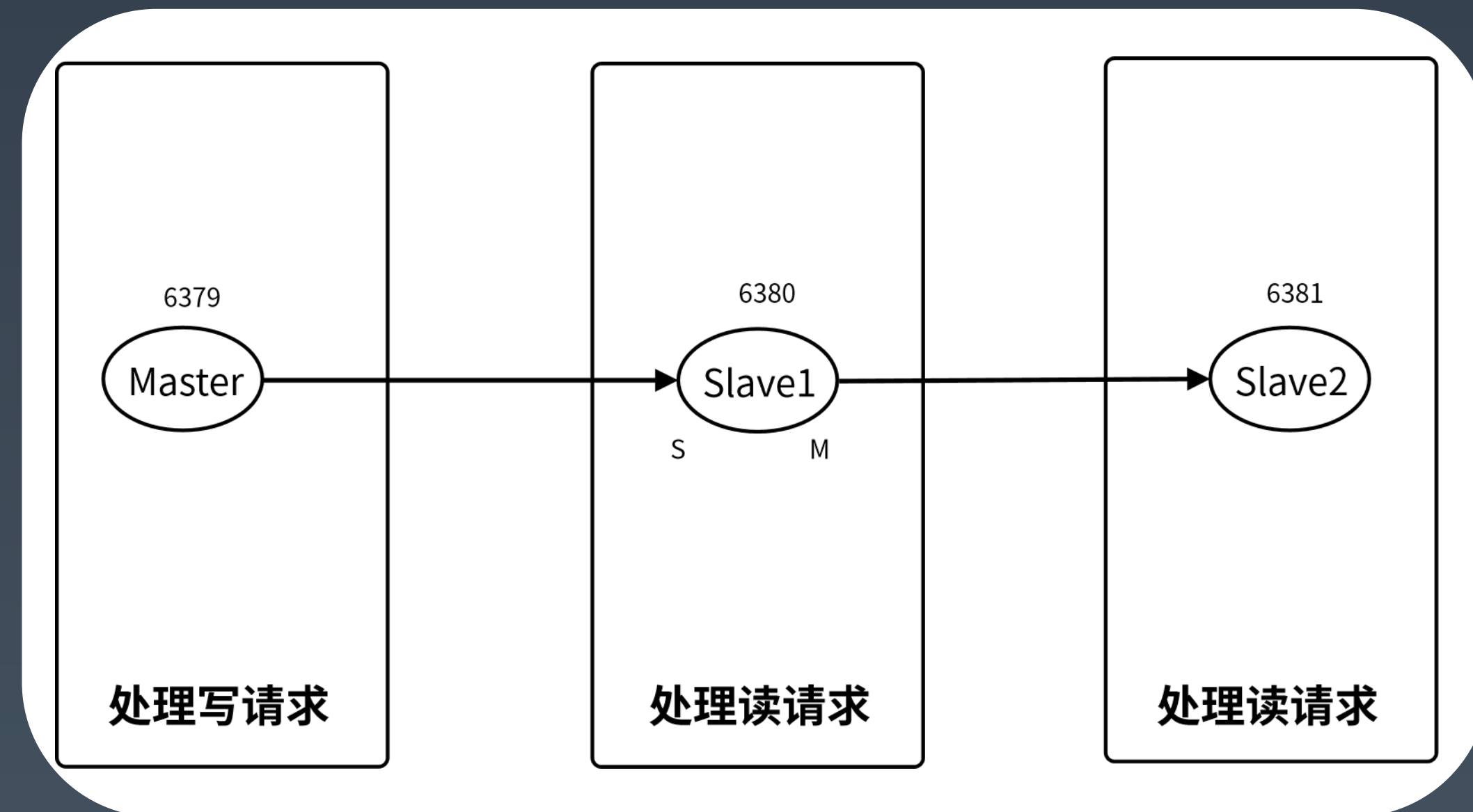
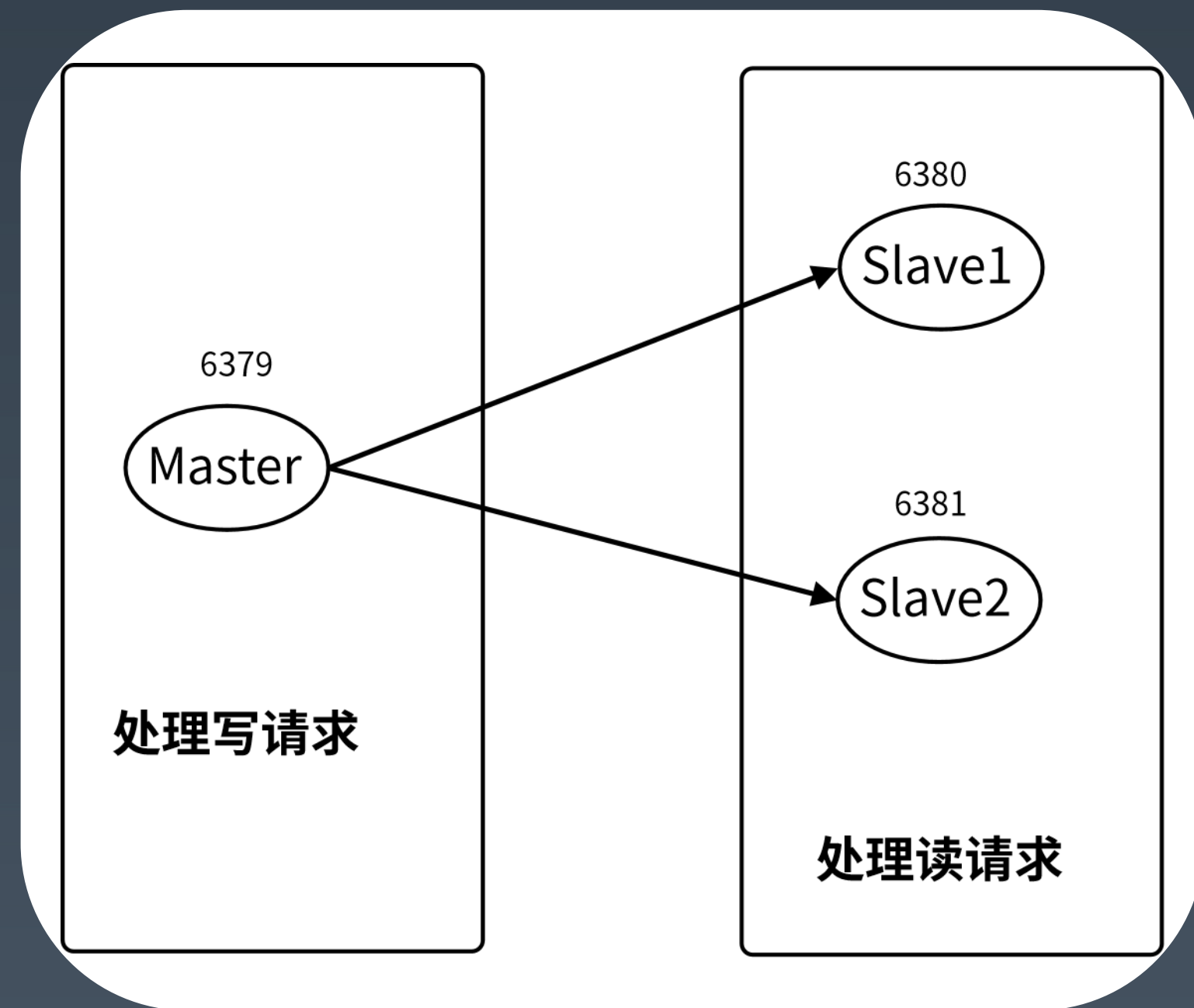
Redis 主从复制：从单机到多节点 ~ mysql 主从

极简的风格, 从节点执行:

> SLAVEOF 127.0.0.1 6379

也可以在配置文件中设置。

注意：从节点只读、异步复制。



Redis Sentinel 主从切换：走向高可用-MHA

可以做到监控主从节点的在线状态，并做切换（基于 raft 协议）。

两种启动方式：

```
> redis-sentinel sentinel.conf
```

```
> redis-server redis.conf --sentinel
```

sentinel.conf配置：

```
sentinel monitor mymaster 127.0.0.1 6379 2
```

```
sentinel down-after-milliseconds mymaster 60000
```

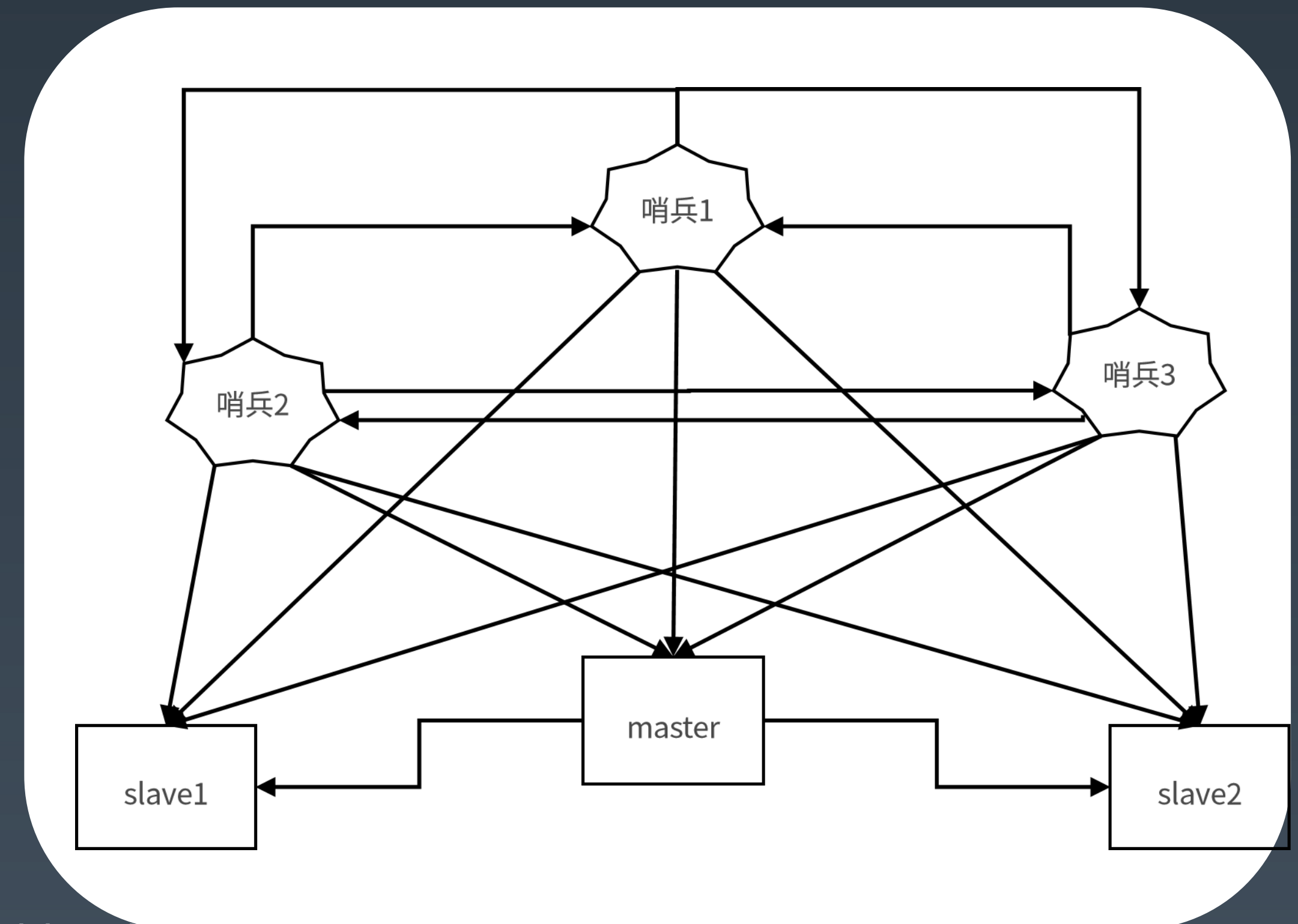
```
sentinel failover-timeout mymaster 180000
```

```
sentinel parallel-syncs mymaster 1
```

不需要配置从节点，也不需要配置其他sentinel信息

redis sentinel 原理介绍：<http://www.redis.cn/topics/sentinel.html>

Redis 复制与高可用配置：<https://www.cnblogs.com/itzhouq/p/redis5.html>



Redis Cluster: 走向分片 ~ 全自动分库分表

主从复制从容量角度来说，还是单机。

Redis Cluster 通过一致性 hash 的方式，将数据分散到多个服务器节点：先设计 16384 个哈希槽，分配到多台 redis-server。当需要在 Redis Cluster 中存取一个 key 时，Redis 客户端先对 key 使用 crc16 算法计算一个数值，然后对 16384 取模，这样每个 key 都会对应一个编号在 0-16383 之间的哈希槽，然后在此槽对应的节点上操作。

> cluster-enabled yes

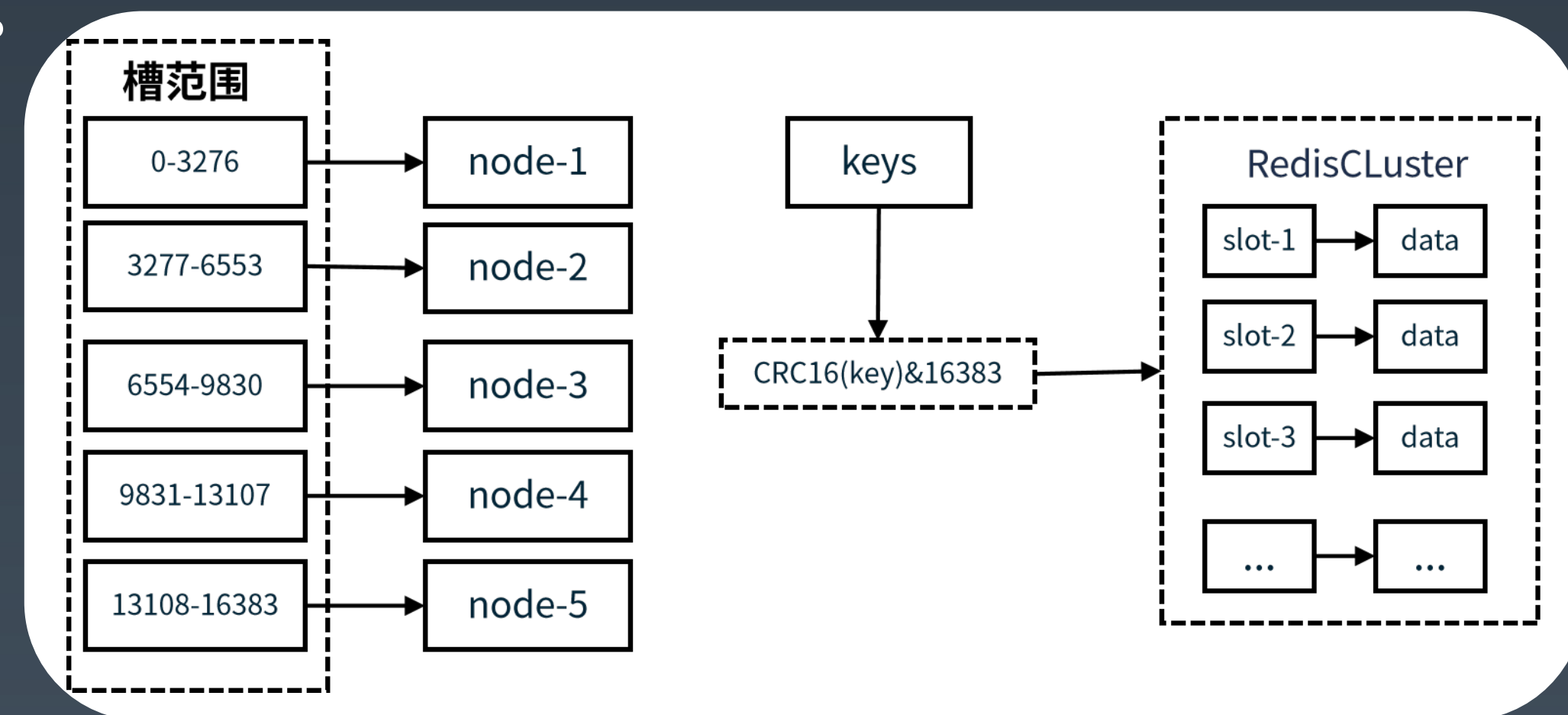
注意：

- 1、节点间使用 gossip 通信，规模<1000
- 2、默认所有槽位可用，才提供服务
- 3、一般会配合主从模式使用

redis cluster 介绍: <http://redisdoc.com/topic/cluster-spec.html>

redis cluster 原理: <https://www.cnblogs.com/williamjie/p/11132211.html>

redis cluster 详细配置: <https://www.cnblogs.com/renpingsheng/p/9813959.html>



Java 中配置使用 Redis Sentinel *

代码示例 && 作业

Java 中配置使用 Redis Cluster *

代码示例 && 作业

2. Redisson 介绍

Redis 的 Java 分布式组件库-Redisson

基于 Netty NIO, API 线程安全。

亮点：大量丰富的分布式功能特性，比如 JUC 的线程安全集合和工具的分布式版本，分布式的基本数据类型和锁等。

官网：<https://github.com/redisson/redisson>

Redisson

示例1:

分布式锁, RLock ==> 能实现跨节点的锁状态

示例2:

分布式的 Map, RMap ==> 全集群共享的, 一个机器改了, 其他都会自动同步

代码演示。

3. Hazelcast 介绍



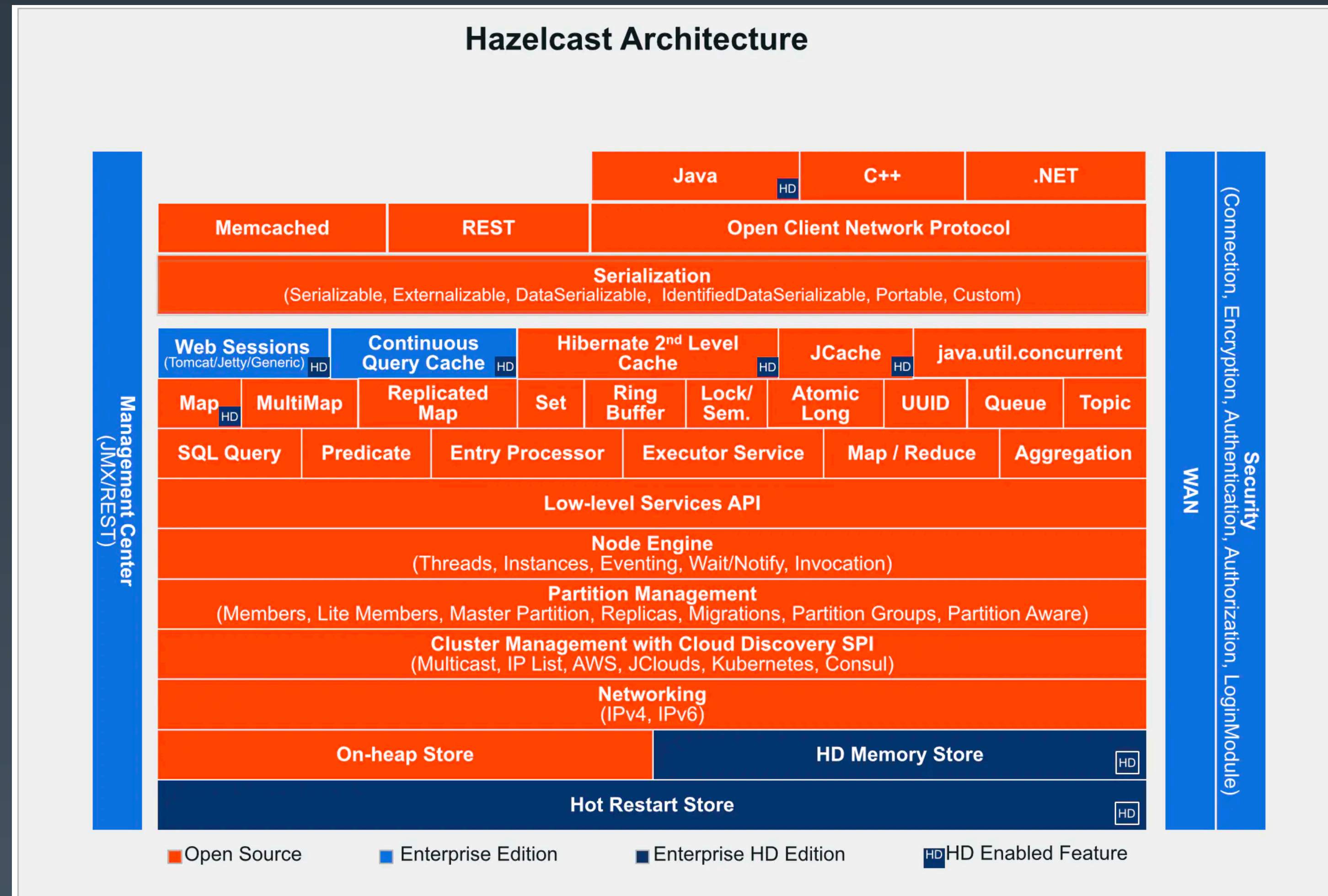
内存网格 - Hazelcast

Hazelcast IMGD (in-memory data grid) 是一个标准的内存网格系统；它具有以下的一些基本特性：

1. **分布式的**：数据按照某种策略尽可能均匀的分布在集群的所有节点上。
2. **高可用**：集群的每个节点都是 active 模式，可以提供业务查询和数据修改事务；部分节点不可用，集群依然可以提供业务服务。
3. **可扩展的**：能按照业务需求增加或者减少服务节点。
4. **面向对象的**：数据模型是面向对象和非关系型的。在 Java 语言应用程序中引入 hazelcast client api 是相当简单的。
5. **低延迟**：基于内存的，可以使用堆外内存。

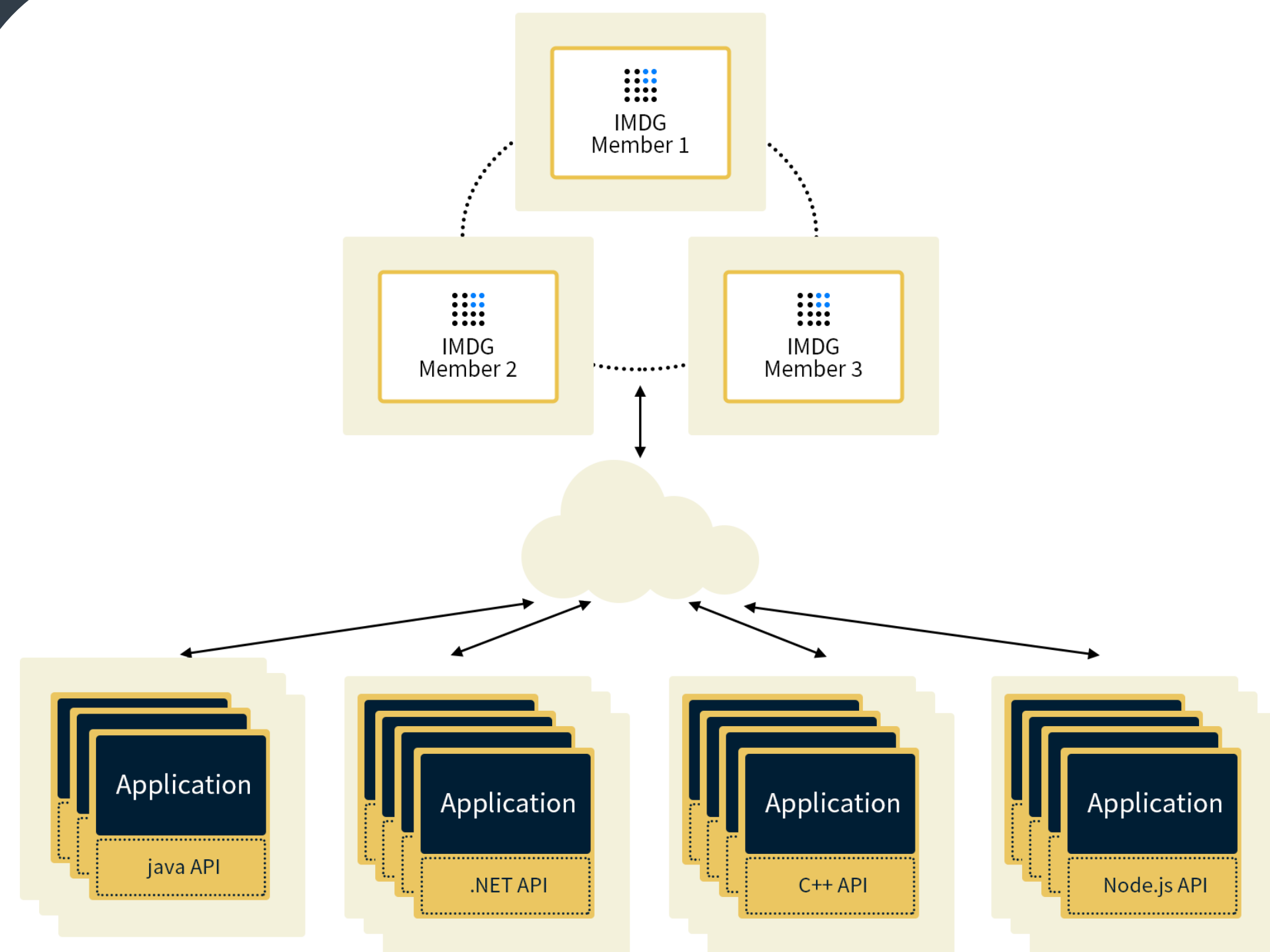
文档：<https://docs.hazelcast.org/docs/4.1.1/manual/html-single/index.html>

内存网格 - Hazelcast 架构

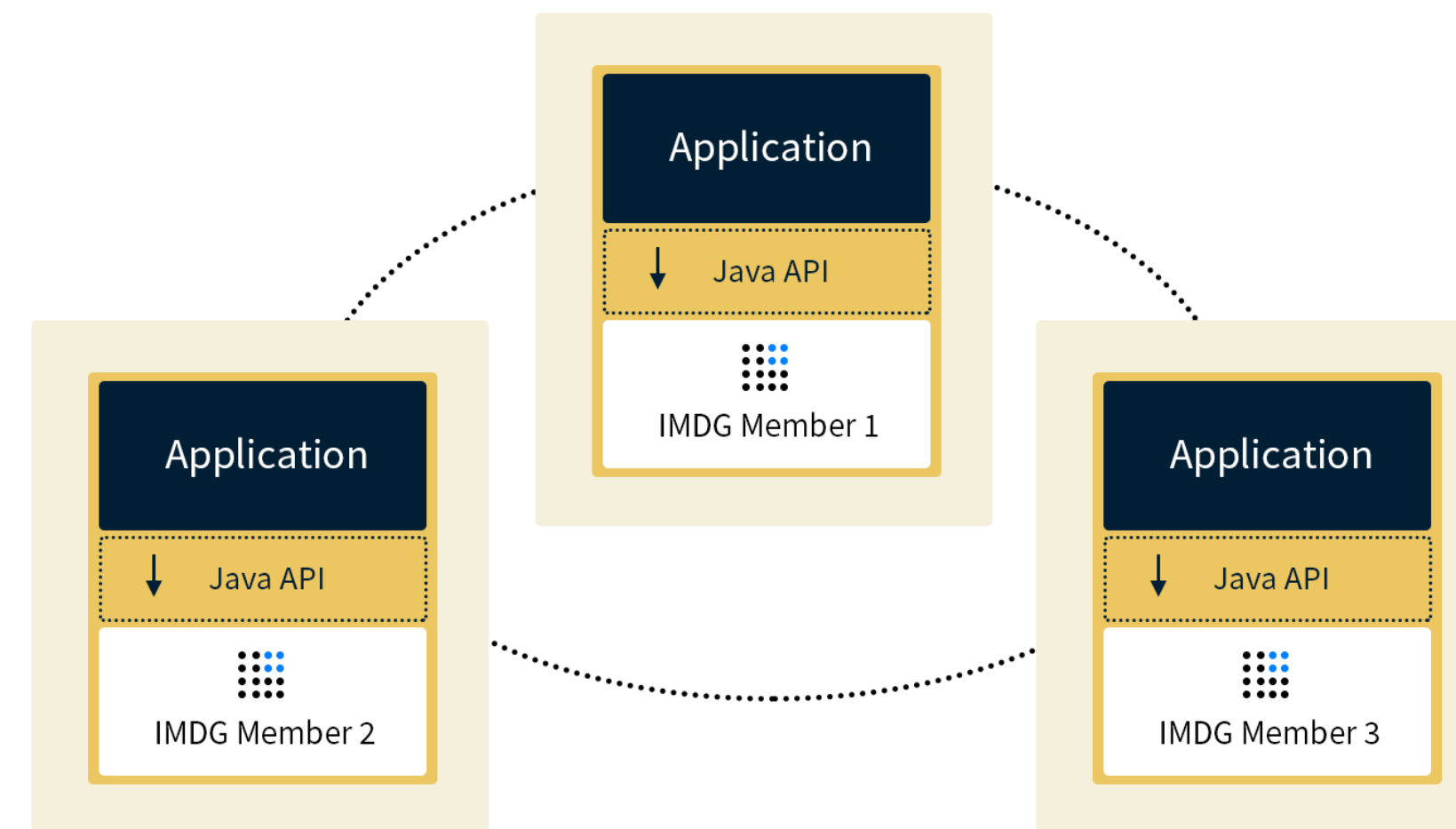


内存网格 - Hazelcast 部署模式

Client-Server 模式



嵌入 (Embedded) 模式



内存网格 - Hazelcast 数据分区

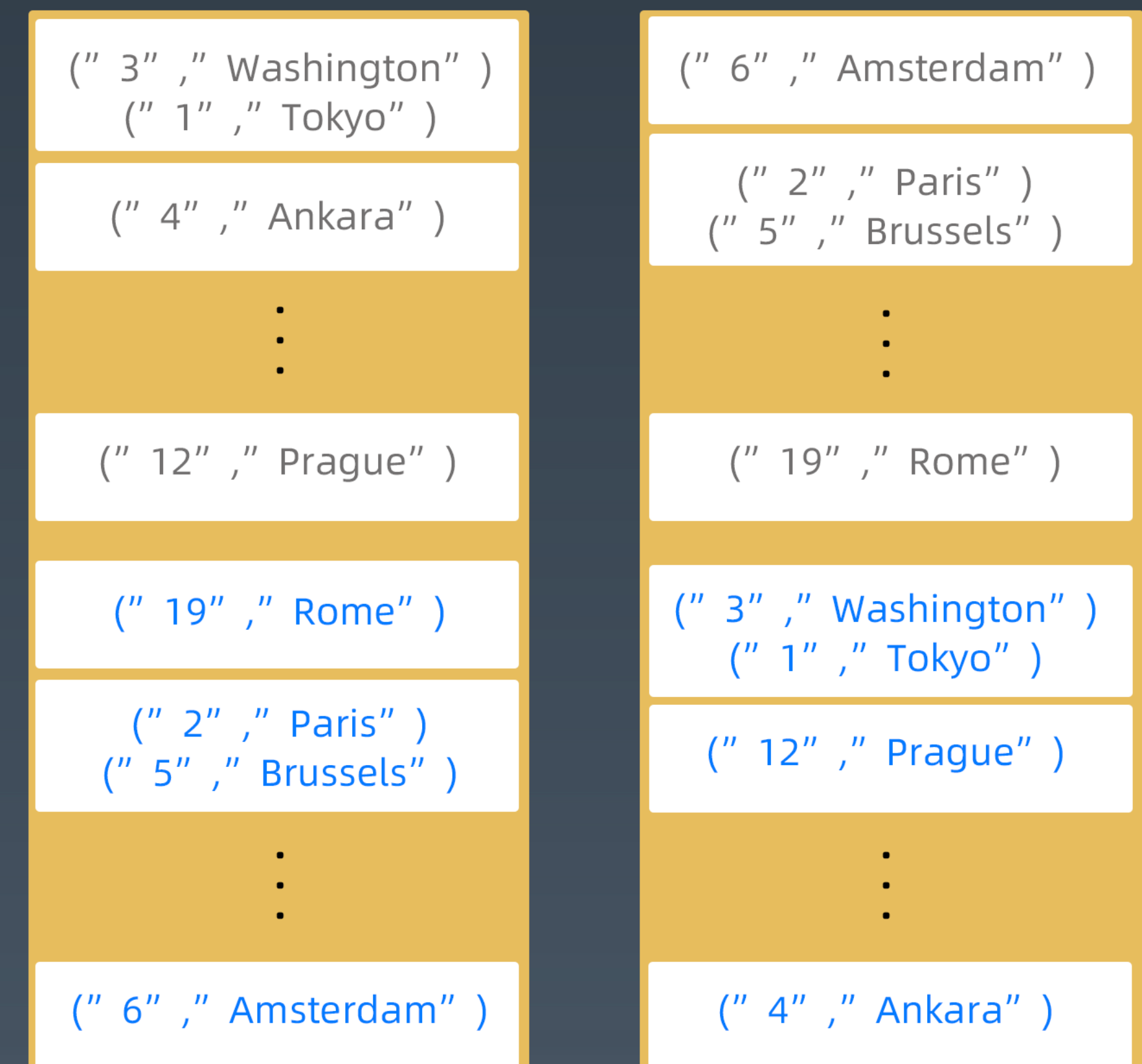
以 Map 结构说明如下：

数据集默认分为 271 个分区；可以通过 `hazelcast.partition.count` 配置修改。

所有分区均匀分布在集群的所有节点上；同一个节点不会同时包含一个分区的多个副本(副本总是分散的以保证高可用)。

副本配置：

```
<hazelcast>
  <map name="default">
    <backup-count>0</backup-count>
    <async-backup-count>1</async-backup-count>
  </map>
</hazelcast>
```



内存网格 - Hazelcast 集群与高可用

- 1、AP，集群自动管理，
- 2、扩容和弹性，分区自动 rebalance，业务无感知，
- 3、相关问题：

内存网格 - Hazelcast 事务支持

支持事务操作：

```
TransactionContext context = hazelcastInstance.newTransactionContext(options);  
    context.beginTransaction();  
    try {  
        // do other things  
        context.commitTransaction();  
    } catch (Throwable t) {  
        context.rollbackTransaction();  
    }
```

支持两种事务类型：

ONE_PHASE: 只有一个提交阶段；在节点宕机等情况下可能导致系统不一致；

TWO_PHASE: 在提交前增减一个 prepare 阶段；该阶段检查提价冲突，然后将 commit log 拷贝到一个本分节点；如果本节点宕机，备份节点会完成事务提交动作；

内存网格 - Hazelcast 数据亲密性

确保业务相关的数据在同一个集群节点上，避免操作多个数据的业务事务在执行中通过网络请求数据，从而实现更低的事务延迟。

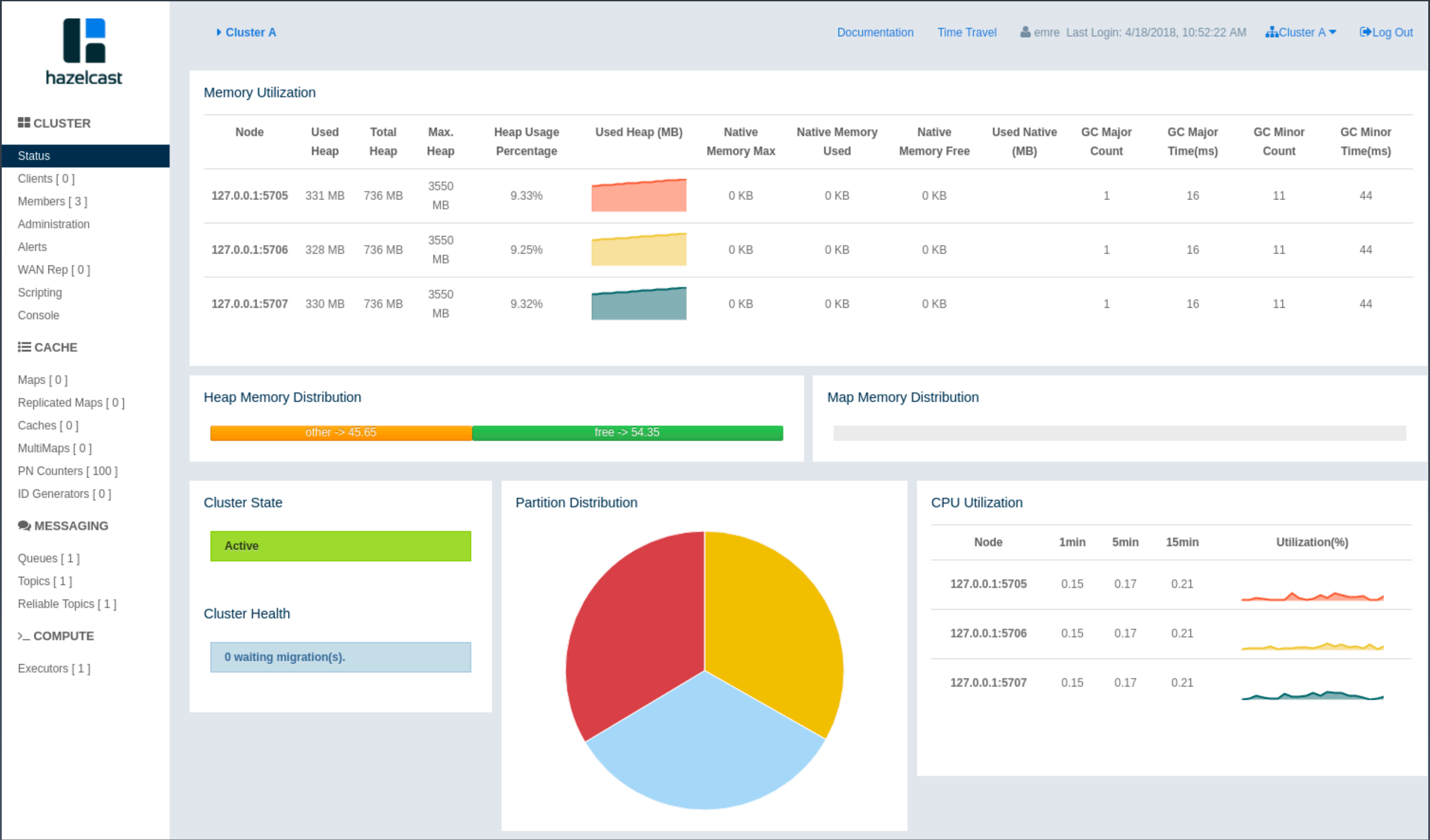
1. 通过 PartitionAware 接口，可以将相关数据定位在相同的节点上；

```
public interface PartitionAware<T> {  
    T getPartitionKey();  
}
```

2. 自定义：PartitioningStrategy

```
<map name="name-of-the-map">  
    <partition-strategy>  
        com.hazelcast.partition.strategy.StringAndPartitionAwarePartitioningStrategy  
    </partition-strategy>  
</map>
```

内存网格 - Hazelcast 控制台



4.总结回顾与作业实践

第 23 课总结回顾

Redis 的集群与高可用

Redisson

Hazelcast

第 23 课作业实践

1、（**必做**）配置 redis 的主从复制，sentinel 高可用，Cluster 集群。

提交如下内容到 GitHub：

- 1) config 配置文件，
- 2) 启动和操作、验证集群下数据读写的命令步骤。

2、（选做）练习示例代码里下列类中的作业题：

`08cache/redis/src/main/java/io/kimmking/cache/RedisApplication.java`

3、（挑战☆）练习 redission 的各种功能；

4、（挑战☆☆）练习 hazelcast 的各种功能；

5、（挑战☆☆☆）搭建 hazelcast 3节点集群，写入100万数据到一个 map，模拟和演示高可用，测试一下性能；