QingCloud Technologies

NoSQL 技术实践及未来展望

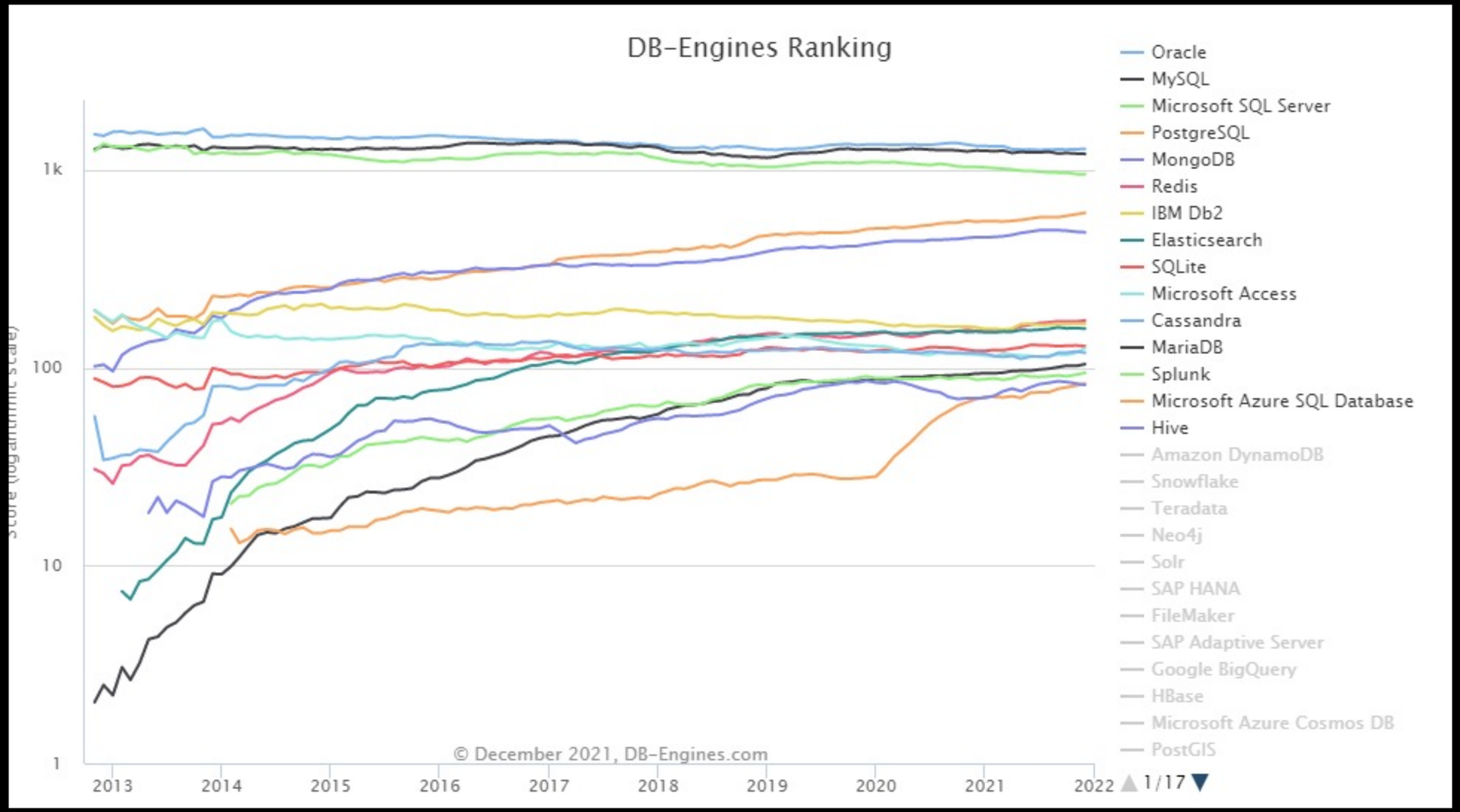
Wang Qi
青云QingCloud 数据库团队负责人

引言

随着工业互联网时代的来临，NoSQL 数据库技术的兴起从 2009 年到现在已经有十来年发展，在数字化转型的当下，NoSQL 数据库扮演着重要角色，企业级的应用也出现了越来越多的机遇与挑战。

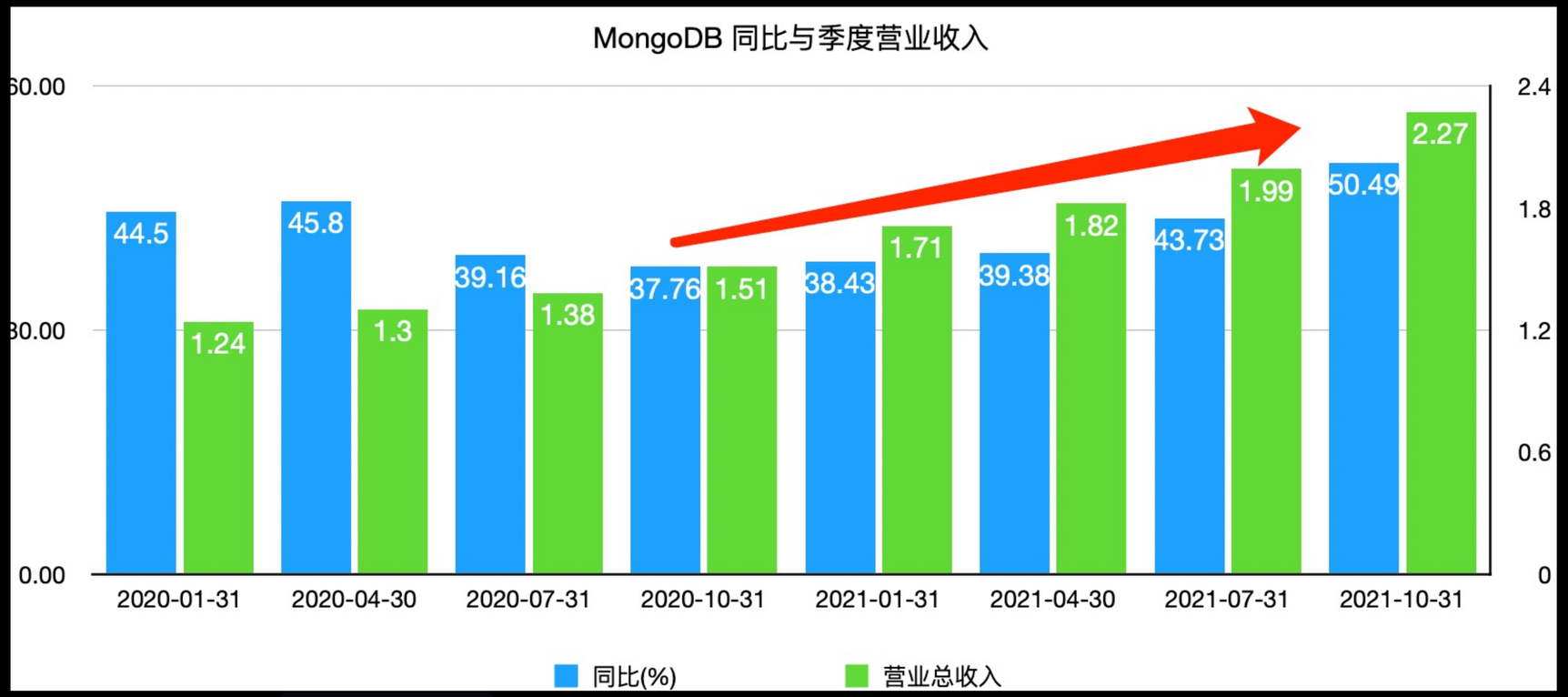
目录

NoSQL 趋势分析


DB-Engines Ranking
Score (logarithmic scale)
1k
100
10
1
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
© December 2021, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
Redis
IBM Db2
Elasticsearch
SQLite
Microsoft Access
Cassandra
MariaDB
Splunk
Microsoft Azure SQL Database
Hive
Amazon DynamoDB
Snowflake
Teradata
Neo4j
Solr
SAP HANA
FileMaker
SAP Adaptive Server
Google BigQuery
HBase
Microsoft Azure Cosmos DB
PostGIS
1/17

MongoDB 财报情况

MongoDB 2022 财年第三季度财务，截止 2021 年 10 月 31日



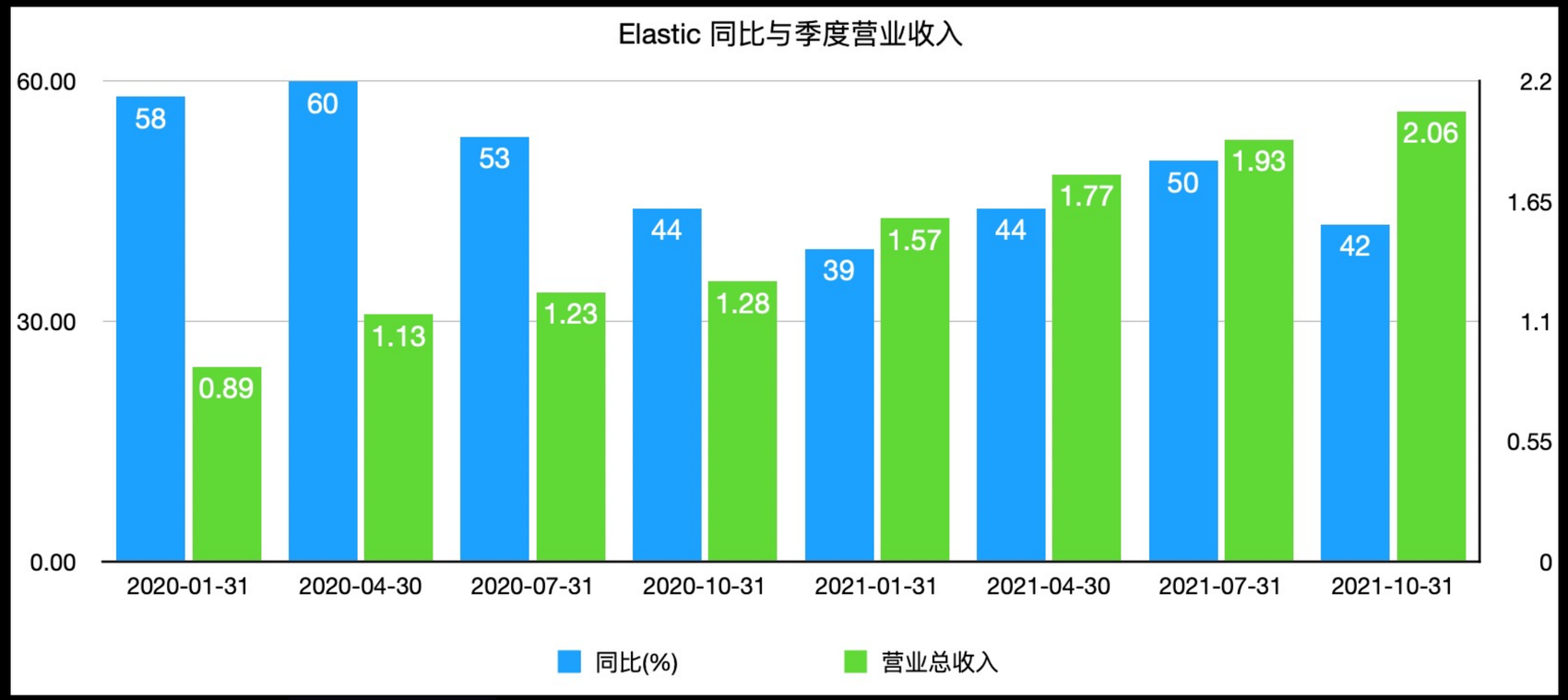
收入：2022 财年第三季度的总收入为 2.27 亿美元，同比增长 50%。订阅收入为 2.179 亿美元，同比增长 51%，服务收入为 900 万美元，同比增长 35%。

其中 Atlas 收入增长了 84%，公司的客户数量增加到 31,000 多个。

MongoDB 成唯一市值超300亿美元的上市开源公司。

Elastic 财报情况

Elastic 2022 财年第二季度财务，截止 2021 年 10 月 31日



总收入为 2.060 亿美元，同比增长 42%，按固定汇率计算增长 41%。订阅客户总数超过 17,000 人。

Elastic Cloud 收入为 6900 万美元，按报告和固定汇率计算，同比增长 84%。

预计 Elastic Cloud 收入在未来三年内将超过总收入的 50%。

总结

无论是 MongoDB Atlas 的 DBasS 还是 Elastic Cloud 的 SaaS 服务，他们依托于自身强大的背书结合产品优势来降低复杂性、自动化运营并更快地做出数据驱动的决策。

从 Elasticsearch 到 OpenSearch 的转变

各大云厂商提供开源版本的 ElasticSearch 已经很长时间了，很多用户都是基于开源版本并开始构建使用的，其实客户是需要用到某些特性的，例如：安全认证，跨集群同步，告警，性能分析等

现状

客户继续使用开源版本，还有其他代替方案吗？

新的选择？

OpenSearch 是一个社区驱动的开源搜索和分析套件，

源自 Apache 2.0 许可的 Elasticsearch 7.10.2 和 Kibana 7.10.2

OpenSearch 正在从开源 Elasticsearch 停止的地方开始，代码中任何有 Elasticsearch 或 Kibana 引用的地方，最后都会更改为 OpenSearch

可替代性

两者都提供了大数据解决方案，且底层实现一致

OpenSearch 覆盖了开源版 Elasticsearch 的所有功能，并为其提供媲美 X-Pack 的商业能力

总结

许可协议限制了在云厂商的使用，开源则提供了更多的可能。当云厂商无法使用 Elasticsearch 后续版本的时候，或许可以考虑 OpenSearch

展望

Elasticsearch 仍然会继续引领潮流，占据该领域的霸主地位。而 OpenSearch 依托于 AWS ，相信也会成为一个优秀的搜索引擎解决方案。

目录

背景

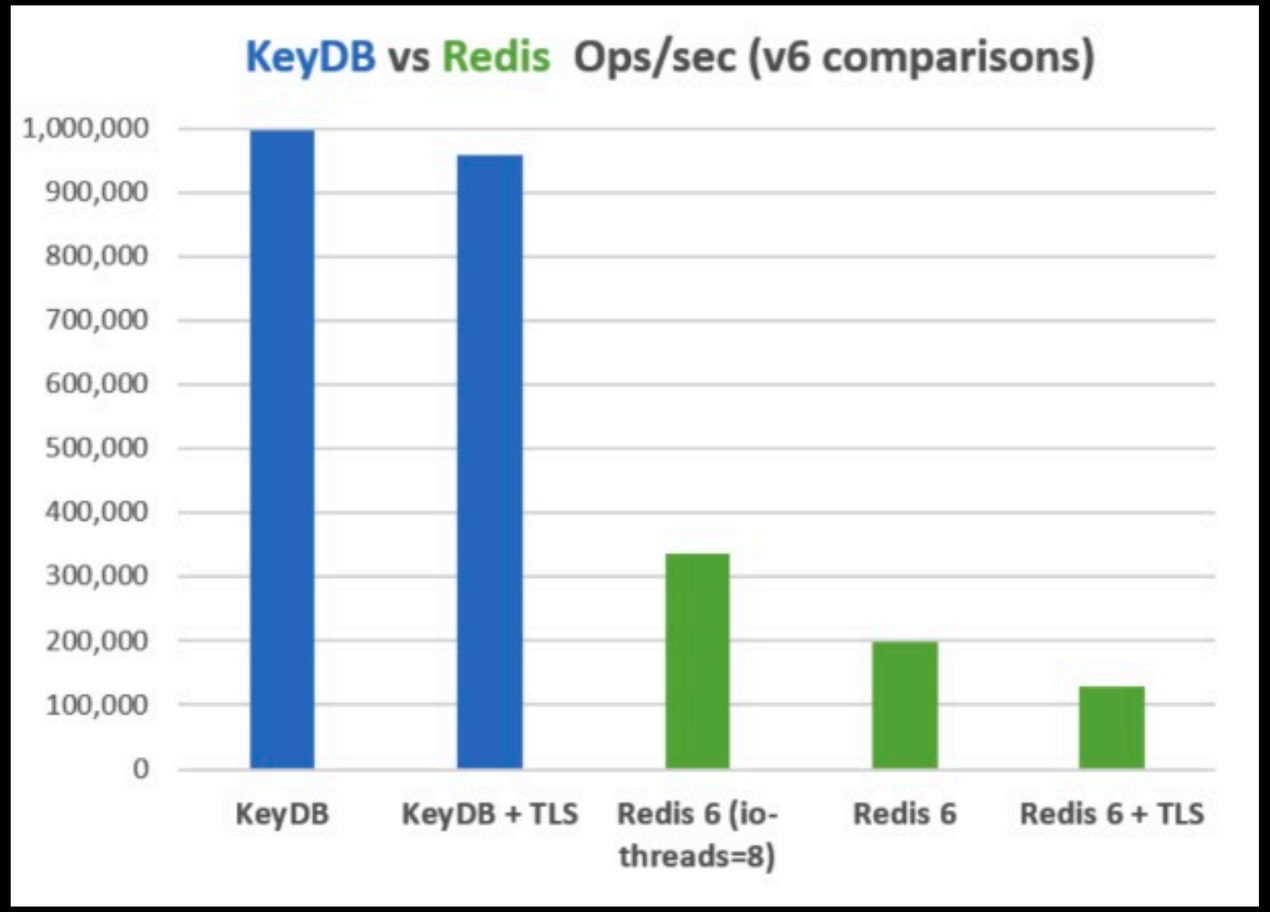
Redis 的 QPS 能否在资源较少的场景下能够取得更好的效果？

有没有合适的解决方案?

什么是 KeyDB?

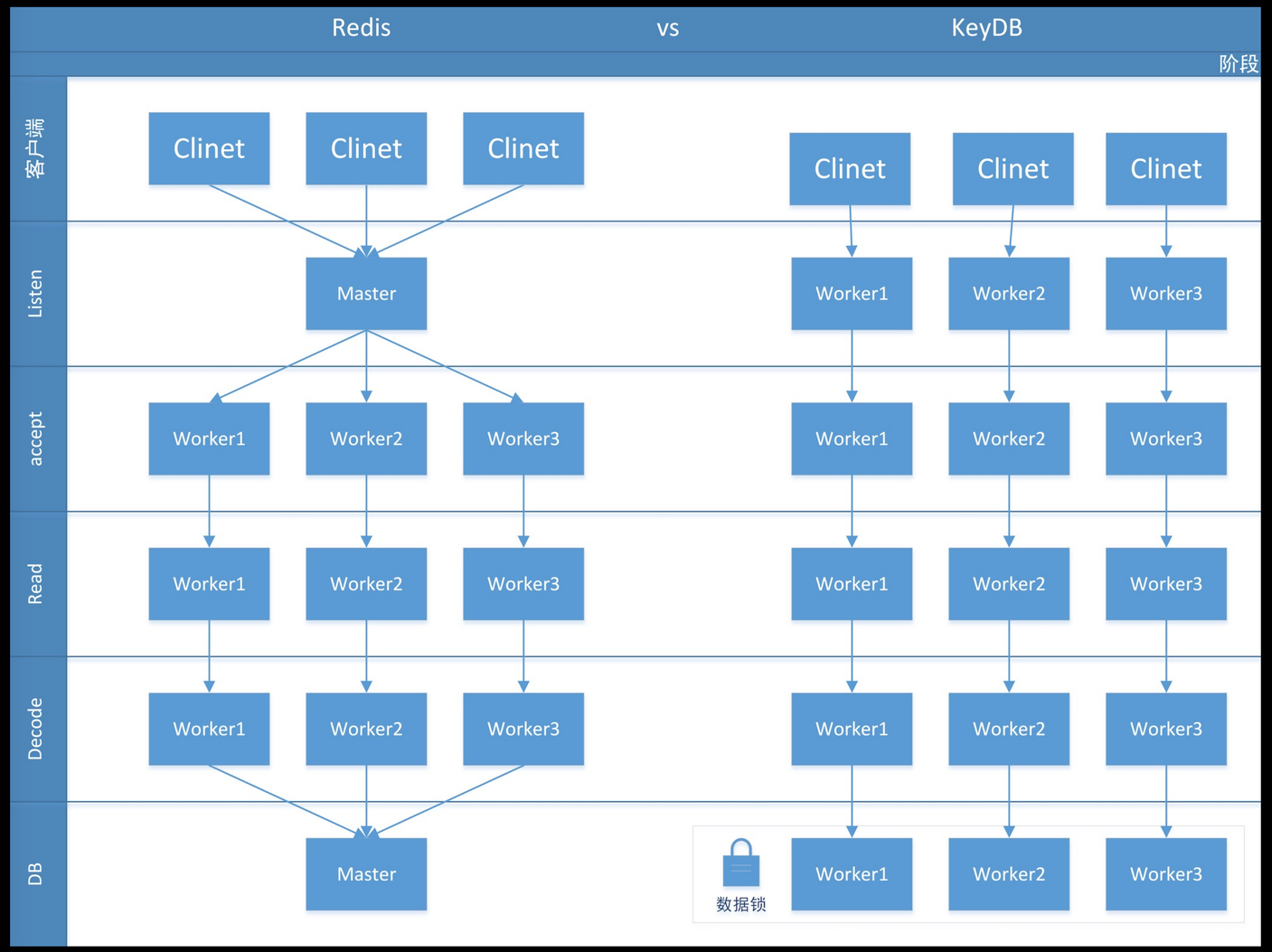
KeyDB 是 Redis 的高性能分支，专注于多线程、内存效率和高吞吐量。除了多线程之外，KeyDB 还具有仅在 Redis Enterprise 中可用的功能，例如 Active Replication、FLASH 存储支持，并支持直接备份到 S3，且完全兼容 Redis。

KeyDB vs Redis 性能对比



在相同的硬件上，KeyDB 每秒可以执行两倍于 Redis 的查询，延迟降低 60%

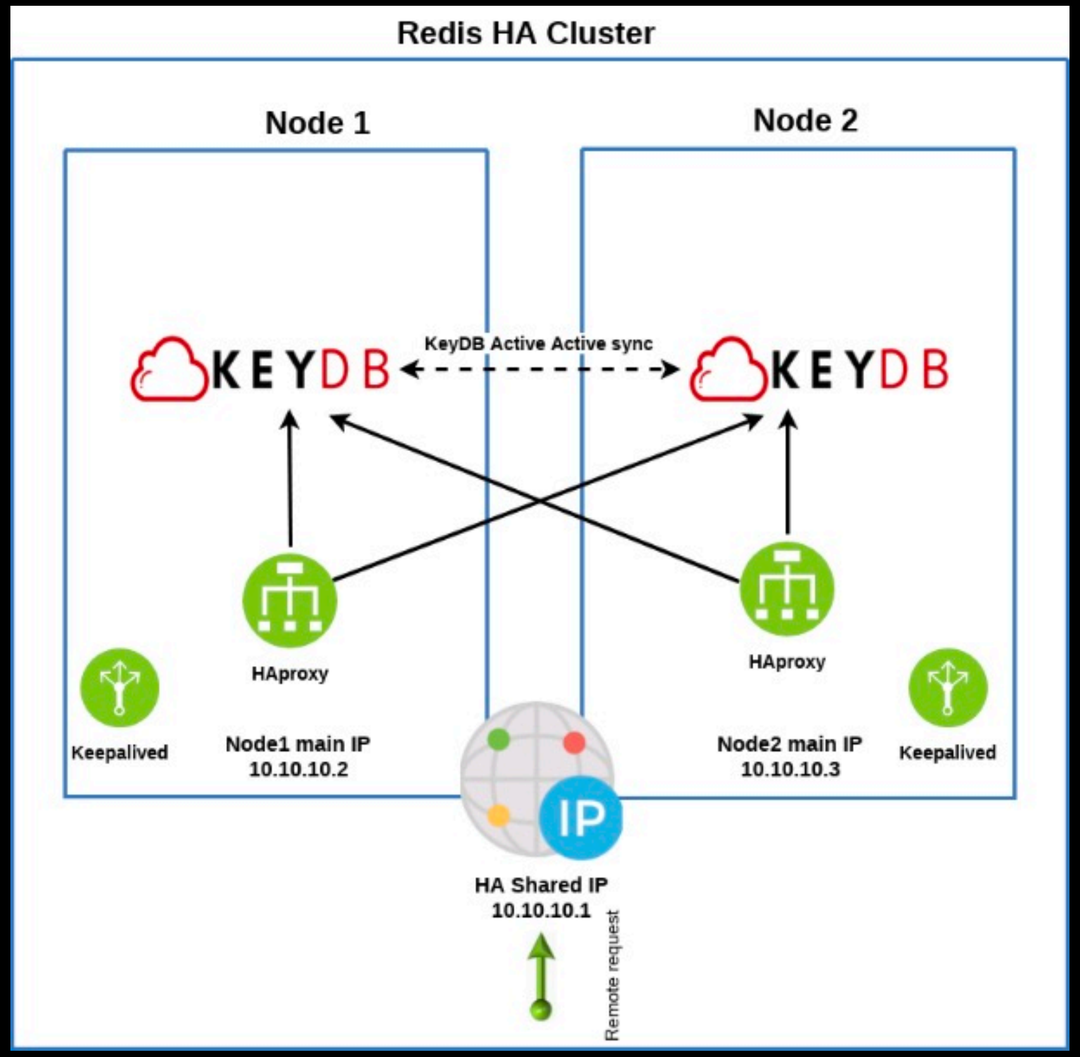
KeyDB vs Redis 原理比较



KeyDB 将 Redis 原来的主线程拆分成了主线程和 worker 线程. 每个 worker 线程都是 IO 线程, 负责监听端口, accept 请求，优化了读取数据和解析协议，这样性能有了较大的提升。

注：更多的测试报告请关注近期RadonDB开源社区公众号。

一个具有完全同步且具备高可用性的 Redis HA 架构图



Keepalived 将负责其中两个节点之间的一个共享 IP，该 IP 将进行故障转移，如果一个节点出现故障，第二个节点将使用该 IP 并继续处理请求。在下一个级别， HAProxy 将侦听此故障转移 IP（对于此示例为 10.10.10.1）。

活动的 Haproxy 将使用简单的循环负载平衡在两个节点上的两个 KeyDB 服务之间传播请求。最后，KeyDB 将使用“ active-replica yes ”参数启动，这会将 KeyDB 服务配置为彼此的活动副本。

优势

KeyDB 为我们提供了一个完全一致的集群，并在节点出现问题后为我们提供自动修复。我们所有的数据现在都在保存中，并且将具有高可用性和容错性，而且我们现在有一个负载平衡和两个节点，处理请求的速度方面比一个节点更好且 QPS 是 Redis sentinel 的两倍，成本却低于它。

为什么我会关注 KeyDB

青云一直关注 NoSQL 生态，我们发现客户绝大部分的资源（副本）处于闲置的状态，只有遇到故障主从切换时，副本节点才能起到作用。我们需要思考是，如何提高资源利用率，如何提高性能，这时我认为你至少可以去关注它。

目录

来自客户的困惑

```
public static LotteryPeriod getCurrentAwardNumberPeriod(String gameId) {
    List<LotteryPeriod> periodList = getPeriodsByGameId(gameId);
    if (periodList == null) {
        return null;
    }
    Timestamp now = DateUtil.getCurrentTimestamp();
    LotteryPeriod lotteryPeriod = null;
    for (LotteryPeriod period : periodList) {
        if (period.getEndTime().before(now) && period.getAwardTime().before(now)) {
            if (LotteryAwardCache.getLotteryAward(gameId, period.getPeriod(), false) != null) {
                lotteryPeriod = period;
                break;
            }
        }
    }
    return lotteryPeriod;
}
```

问题

A. 3000+ 行代码

B. 接近 20 个 getXXXPeriod(String)

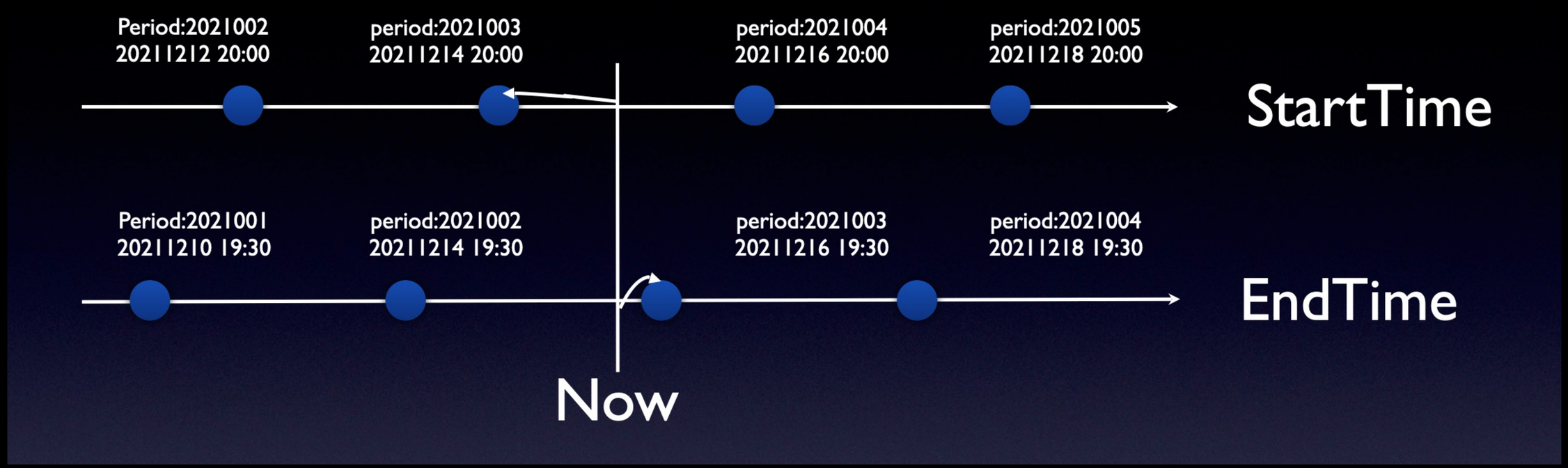
C. 运行后加载几万个 Period 实例

改进思路

Timestamp - long

startTime,endTime....betDeadline迭代比较 -> Timeline

Cache



将每一个时间维度按照 Redis 缓存的 SortedSet 结构保存，Timestamp 转化为 Long 作为 SortedSet 的 score 存储，member 则为 periodId ；为了提高用户访问效率，将每个特定需求的期次信息直接使用 Redis 缓存 String 结构保存，即将对象转化为字节数组保存；为保证这些数据时效性，设置合理的过期时间并且定时刷新这些数据。

```java
@Override
public List<GamePeriod> getCurrentPeriods(Long gameId) {
    String key = RedisConstant.getPeriodDetailKey(gameId, RedisConstant.CURRENT_PERIOD);
    List<GamePeriod> list = redisService.hessian2Get(key);
    if (GameCache.getGame(gameId).getGameType() != Game.GAME_TYPE_TRADITIONAL) {
        list = resetRedisTimeline(gameId, RedisConstant.CURRENT_PERIOD, list);
    }
    return list;
}
```

A. 3000+ 行代码 ——————-> 500+ 行代码

B. 接近 20 个 getXXXPeriod(String) ——————-> 6 个 通用的 getXXXPeriod(String)

C. 运行后加载几万个 Period 实例 ——————-> 运行后加载几个 Period 实例

目录

NoSQL 未来展望

NoSQL 数据库市场仍保持高速增长，2021-2025年均复合增速高达13%

在数字化转型下，容器化、服务化会成为未来趋势

青云 NoSQL 开源宗旨

求新创新，持之以恒，共创价值，解决用户的痛点，与用户一起成长

开源项目

Xenon

2017

高可用组件 Xenon开源。

基于 Xenon 构建的，金融级强一致数据库 MySQL Plus 发布。

基于 Raft 协议，无中心化选主；
支持一主多从，弹性扩容伸缩。

Radon

2018

分布式组件 Radon 开源。

兼容 MySQL 的分布式数据库RadonDB 发布。

ChronusDB

2019

基于 ClickHouse，分析型时序数据库。

MySQL HTAP

2020

研发 MaterializeMySQL 引擎，并贡献 ClickHouse 开源社区。

数据库容器化

2021

基于 Heml、Operator 数据库容器化产品发布。

已将容器编排方案开源。

仓库地址：https://github.com/radondb/

开源社区

群助手微信号：radondb

技术文章

RadonDB 开源社区

高可用 | repmgr 构建 PostgreSQL 高可用集群部署文档【建议收藏】
原创:颜博

工具 | pg_recovery 设计原理与源码解读
原创:张连壮

工具 | PG 集群复制管理工具 repmgr
原创:颜博

工具 | 一条 SQL 实现 PostgreSQL 数据找回
原创:张连壮

视野 | OpenSearch，云厂商的新选择?
原创:王奇

MySQL Operator 02 | 脚手架选型 & 工程创建（文末抽奖）
原创:高日耀

MySQL Operator 01 | 架构设计概

开源项目

RadonDB 开源社区

RadonDB MySQL on K8s 2.1.0 发布!

容器化　高可用

容器化 | ClickHouse Operator 原理解析
原创:苏厚镇

容器化 | ClickHouse on K8s 部署篇【建议收藏】
原创:苏厚镇

容器化 | ClickHouse on K8s 基础篇
原创:苏厚镇

容器化 | 基于 K8s 的新一代 MySQL 高可用架构实现方案

Thank you.