论大规模分布式系统缓存设计策略

【摘要】

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了大规模分布式系统缓存设计策略在项目中的具体应用。系统缓存基于 Redis 内存数据库实现，工作模式的选择上根据不同数据类型，采用了主从模式与集群模式结合的设计；通过数据持久化、数据备份计划、冗余机制和监控平台等方法实现高可用性；通过数据访问层封装同步操作实现缓存一致性，通过哈希环实现分布式算法。最终系统顺利上线，获得用户一致好评。

【正文】

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为“OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

系统需要支撑全校学生编程课程的学习考试，以及大量校外用户的使用，为保证系统的性能和用户体验，以缓存技术进行优化尤为重要。常见的缓存分为三种工作模式：单点模式、主从模式和集群模式。单点模式各个应用服务共享同一个缓存实例，主要瓶颈在于缓存服务器的内存大小、处理能力和网络带宽，适合缓存要求简单、数据量和吞吐量小、性能要求低的场景；主从模式将缓存的数据复制到多台机器上，数据彼此同步，可分散压力、获得更高的吞吐能力，适合数据量不大、改动频率不大、性能要求高的场景；集群模式节点间共享数据，部分冗余保证一定程度的可用性，可存储超过单个服务内存容量数据，通过增加服务器缓解数据增长和压力增加，适合总体数据量大、可伸缩性需求高、客户端数量庞大的场景。

OJ 系统使用 Spring Cloud 微服务架构开发，基于 Redis 内存数据库实现缓存，运用了多种缓存设计策略，本文着重从工作模式的选择、高可用性的设计、缓存一致性与分布式算法三个方面的问题展开论述。

1. 工作模式的选择

OJ 系统需要支撑全校学生编程课程的学习考试，以及大量校外用户的使用，对可靠性和性能要求高，因此单点模式不予考虑。经过对 OJ 系统业务需求的分析，我们归纳了三种需要缓存的数据类型。第一种是系统配置项、角色权限分配等元数据信息。这部分信息使用频率很高但数据量小，不会经常改动，我们采用主从模式来缓存，主缓存节点供系统配置模块写入数据，从缓存节点分别供具体业务服务调用。在系统启动时，自动将相关数据装入缓存。信息修改时，先写入主缓存节点，再同步更新至从缓存节点。第二种是用户登录的会话状态，这部分信息使用频率高、更新频繁，且几乎所有的业务服务都需要依赖此类数据，为保证性能与可靠性，我们采用集群方式来缓存，集中管理用户会话，各个业务服务通过数据访问层来调用具体的缓存节点。第三种是业务逻辑中一定时间内不会变化，但数据量大的信息，如试题信息、实验作业信息等，以及统计机制自动识别的高频访问信息，也采用集群方式缓存。

2. 高可用性的设计

为实现 OJ 系统缓存的高可用性，一方面我们开启了 Redis 数据持久化功能，并配置了相应的备份策略，能够有效地解决数据误操作和数据异常丢失的问题，另一方面，我们设计了一些冗余机制。在主从模式中，采用多机主备架构实现冗余，在主节点故障时，自动进行主备切换，将从节点提升为主节点继续服务，保证服务的平稳运行。如果主节点和从节点之间连接断开，重新连接时从节点会进行数据的部分重同步，当无法进行部分重同步时，会进行全量重同步。在集群模式中，采用 Cluster 技术实现冗余，每个节点保存数据和整个集群状态，负责一部分哈希槽，每个节点都和其他所有节点连接并共享数据。为了使在部分节点失效或者大部分节点无法通信的情况下集群依然可用，在集群内部使用了主从复制模型，每个节点都会相应地有 1~N 个从节点，当某个节点不可用时，集群便会将它的从节点提升作为新的主节点继续服务。缓存服务发生异常时，可通过 OJ 系统的服务监控平台产生报警，提醒运维人员及时处理。

3. 缓存一致性与分布式算法

为保证 Redis 缓存与原数据库的数据一致性，当读取数据时，会先读取 Redis 缓存中的数据，如果 Redis 缓存中不存在所要的数据，则从原数据库中读取，并同步写入至 Redis 缓存中，当写回/删除数据时，写入到原数据库中，并同步淘汰 Redis 缓存中的数据。业务服务通过使用专门的数据访问层来调用增加缓存后的数据库，使数据缓存机制对应用透明。在缓存内部实现一致性，通过分布式哈希算法来实现，为考虑到日后缓存集群的扩展需要，因此不能使用简单的模 N 哈希法，我们在 OJ 系统中采用了哈希环的算法。该算法构造一个长度为 2^32 的整数环，将 Redis 节点放置于环上，当业务服务调用缓存时，首先以服务的应用 ID 作为键值计算哈希在环上定位，然后沿顺时针方向找到最近的 Redis 节点，完成映射。当缓存服务集群中有节点故障，以及添加新节点时，只会影响上一个节点的数据，避免了发生缓存雪崩的情况，提高了容错性和扩展性。哈希环中缓存节点过少时易发生缓存倾斜，我们通过增加虚拟节点的方式解决了该问题。

【总结】

我们在这次系统设计中，还使用了很多其他的缓存设计策略，这里不再一一赘述。系统在经过性能测试、负载测试、压力测试、稳定性测试后，自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 名以上的学生用户、评测了 70000 份以上的程序代码，获得

了单位同事领导和学校教师们的一致好评。日常使用过程中最高出现过 600 余用户同时在线进行实验作业提交、评测的情况，基本未出现页面加载缓慢、请求超时的问题，满足了高校编程课平台的基本性能需求。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合理的缓存设计密不可分。经过这次大规模分布式系统缓存设计的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，使整个系统能够更加好用，更有效地服务于高校师生。

———————————————————