哈尔滨工业大学

**计算学部**

**2024年秋季学期**

**《软件架构与中间件》课程**

**实验报告**

**Lab 2：计算层与数据层软件架构实验**

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **联系方式** |
| XXX | 22NNNNNNN | Email/手机号码 |
| XXX | 22NNNNNNN | Email/手机号码 |

目 录

[1 实验概述 1](#_Toc177484343)

[1.1 实验目的 1](#_Toc177484344)

[1.2 实验要求 1](#_Toc177484345)

[2 实验内容与过程 1](#_Toc177484346)

[2.1 nginx集群负载均衡实验 1](#_Toc177484347)

[2.2 hadoop分布式计算实验 2](#_Toc177484348)

[2.3 Mycat数据库分库分表实验 2](#_Toc177484349)

[2.4 Sharding-JDBC数据库分库分表实验 2](#_Toc177484350)

[2.5 Redis数据缓存实验 3](#_Toc177484351)

[3 结对开发过程记录 3](#_Toc177484352)

[4 实验总结 4](#_Toc177484353)

[5 教师评语 4](#_Toc177484354)

[文档全部完成之后，请在上述区域点击右键，选择“更新域”，在打开的对话框中选择“更新整个目录”]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学号：** |  | **姓名：** |  | |
| **学号：** |  | **姓名** |  | |
| 实验概述 | | | |  |
| 实验目的 1）学习构建集群的方法  2）学会使用典型的负载均衡框架  3）学会使用Hadoop 进行map-reduce编程  4）学会使用MyCat 或 Sharding-JDBC进行分库分表等数据架构技术  5）学会使用 Redis进行数据缓存优化  6）能够灵活应用计算/数据密集型中间件到实际系统 实验要求 1）2人结对成组  2）实验2.1必做，2.2选做，2.3和2.4两选一，2.5必做  3）结合前序课程中“进销存”管理类系统(或其他实际软件系统)进行计算层架构重构，支持海量用户的在线高并发请求场景  4）应给出关键过程的细节 | | | | |
| 实验内容与过程 | | | |  |
| nginx集群负载均衡实验 1）请给出搭建单虚拟机nginx的关键过程，并修改nginx主机的首页内容从英文变为中文。  2）请详析配置文件中涉及的upstream、location和server块。  3）请描述所搭建的虚拟集群环境，验证虚拟集群可以互相访问。  4）配置集群的负载均衡系统，给出关键过程，验证负载均衡系统是否工作。  5）配置并验证使用不同集群负载均衡算法（至少三种），给出过程、观测的情况，给出自己对负载均衡算法的理解和认识。  6） 应用负载均衡技术改造遗留的“进销存”系统，赋予支持海量用户的在线高并发请求的能力，请给出设计细节并分析负载均衡前后的区别。  7）迁移到真实集群环境重做负载均衡实验(选做)。请给出实验过程中需要注意的问题，比较并分析虚拟集群和真实集群下的异同。 hadoop分布式计算实验 1）配置集群环境，设置主节点主机名为manager，从节点主机名为workerX(X为数字编号)，给出主要过程；同时给出配置SSH免密登录的关键步骤。  2）请配置hadoop集群计算环境，要求数据块副本数为3。给出主要过程，并验证正确性。  3）请使用hadoop集群环境计算英文版[Stray Birds](https://en.wikisource.org/wiki/Stray_Birds)正文中每个单词出现的次数。  4）请使用hadoop集群环境计算遗留的“进销存”系统中海量用户的log日志中的访问统计。请模拟输入数据，从1万个用户共100万次操作记录中计算每个用户的访问次数。 Mycat数据库分库分表实验 1）请给出Mycat配置安装过程中遇到的问题和解决方案。  2）请详析Mycat的分库分表原理和操作方法。  3）请模拟具有复杂表结构和含有较大数据量的数据库表， 并基于此库表描述分库分表的结果，且验证分库分表的效果。 Sharding-JDBC数据库分库分表实验 1）请给出Sharding-JDBC配置安装过程中遇到的问题和解决方案。 1. 首先，打开IDEA，创建Springboot项目    然后，修改pom.xml，加载maven依赖    以docker部署MySQL服务    使用Springboot构建时，出现问题，需要在application.yaml中进一步修改配置项。因为Sharding-JDBC已经修改了配置方法，在yaml中定义分配方法。通过在新版Sharding-JDBC中查阅文档获取配置方案。    2）请详析Sharding-JDBC的分库分表原理和操作方法。  原理：  Sharding-JDBC在应用层实现水平拆分，无需改变业务代码。它基于用户设定的分片键(Sharding Key)，将数据分配到不同的数据节点上。具体是解析SQL请求，判断是否涉及到用户设定的分库分表逻辑，然后根据用户设定的路由策略，将SQL请求路由到合适的库和表。  例如：SELECT \* FROM orders WHERE order\_id = 123  假设订单号按order\_id % 4拆分到4个表，则 Sharding-JDBC 会根据123 % 4的结果确定该查询应路由到orders\_3表。  当涉及多库多表的查询时，Sharding-JDBC 会在不同的数据节点上并行执行查询，然后执行结果归并处理。对于聚合查询，会合并各节点的查询结果。如果是分页查询，按分页条件对合并后的结果再次排序截取。（即都需要先合并，再选择要不要截取）  操作方法：  现在的Sharding-JDBC支持Springboo框架，通过在application.yaml中设定分片策略来进行分库分表。常见的分片策略如下：  范围分片（Range Sharding）：根据数据范围进行拆分，如用户 ID 小于 10000 的存储在数据库 1，大于 10000 的存储在数据库 2。  哈希分片（Hash Sharding）：将分片键的哈希值取模，用于确定数据存储的位置，如order\_id % 4 分散到 4 张订单表。  复合分片（Complex Sharding）：基于多个字段的组合拆分数据，如按用户 ID 和订单时间共同决定数据的存储位置。  按日期分片（Date Sharding）：数据按日期区间拆分，如按月份生成一张表，orders\_2024\_10 表示 2024 年 10 月的订单数据。  3）请模拟具有复杂表结构和含有较大数据量的数据库表， 并基于此库表描述分库分表的结果，且验证分库分表的效果。  本实验基于一个简单的电商订单查询环境，综合了多种分库分表方案，分片规则如下： 用户表 (users) 和 订单表 (orders) ：根据 user\_id 进行分库分表，表分为 users\_0 和 users\_1，orders\_0 和 orders\_1  订单项表 (order\_items) ：根据 order\_id 进行分表，分为 order\_items\_0 和 order\_items\_1，但不分库  商品表 (products) ：不分库分表  分库规则：根据user\_id % 2进行分库。 Redis数据缓存实验 1）请给出Redis配置安装过程中遇到的问题和解决方案。  1. 使用docker安装  docker run –name redis -p 6379:6379 redis:latest    使用redis-cli    2）请详析Redis的缓存清洗策略，数据迁移及扩容策略，面向缓存雪崩、穿透等问题的策略。   1. 缓存过期命令   EXPIRE key seconds [ NX | XX | GT | LT]   1. NX 当key没有过期时间时，更新key为seconds 2. XX 当key已过过期时间时，更新key为seconds 3. GT 当新设定的过期时间大于当前剩余时间时，更新key为seconds 4. LT 当新设定的过期时间小于当前剩余时间时，更新key为seconds 5. 过期缓存删除方式 1. 惰性删除：客户端请求查询key时，根据设置的过期时间检查，过期了就删除，重新从数据库获取。没过期就返回 2. 定期删除：设定定时任务，每次随机抽取一些设定过期时间的key，过期了就删除。如果key很多，那么就根据hash桶的维度去扫描key，扫到20个key为止，如果第一个桶不够，那就接着扫第二个桶。最后删除已选中的key，进行删除，如果过期的key超过25%，接着删 6. 内存淘汰策略  1. Volatile-random：随机删除，使用allkeys-random策略，适用于常规时段商品浏览等不具有明显业务特征的数据 2. Volatile-ttl：删除过期时间内剩余时间最短的key，根据过期时间与当前时间的差额进行短到长的排序，适用于不具有明显业务特征的数据 3. Volatile-lru：删除最近最少使用的key，也就是最近最不常用数据筛选出来，适用于具有冷热数据隔离需求的数据进行清理。比如电商的受季节影响的数据。 4. Volatile-lfu：删除访问次数最少的key。在LRU中，假如一个数据只是最近才被访问一次，就被作为热点保留，显然不合理。因此根据key最近的访问频率进行淘汰 5. 剩下的是针对全部的数据，也就是可以与不需要强制配置过期时间的key的数据分开。 7. 数据迁移策略 1. 主从架构：有一个主数据库实例（master）和多个从数据库实例（slave），可通过slaveof命令来使服务器成为新的从服务器。然后进行复制初始化，通过SYNC命令可以使master向slave发送快照文件，其中包含这段时间执行的写命令，slave执行快照后，master执行写操作都会同步给slave。如果需要将新从服务器变为主服务器，可以先对原master停止写操作，打开新redis的读写操作，修改业务为新的redis配置，然后断开数据同步，这样新redis就从slave升为master。或者使用aof，追加key到新redis服务器，不会删除原有的key。 2. 哨兵模式：基于主从模式，只不过引入哨兵来监控与自动处理故障，监控主从是否正常运行，master故障能自动将slave转换为master。数据迁移上同主从复制 3. 集群模式：通过分片进行数据管理，将哈希槽分配给节点，集群之间的信息通过Gossip协议进行交互。对于新插入的redis服务器，可以将插槽中的一些移动到新服务器，然后会根据CRC16算法得到结果，对结果进行16384取余数，得到插槽值，进而会有一部分请求分配到新服务器中。整个过程无需停止任何业务。 8. 数据扩容策略（上一部分提到的集群模式基础上） 1. 垂直扩容：增大redis服务器的硬件配置。对于更新的服务器，先按照主从模式，通过slaveof进行复制，作为从节点，然后转换业务配置升为主节点 2. 水平扩容：向集群中增加新的Redis实例，然后更新哈希槽，使其一部分由新的Redis进行管理，并将原节点属于该部分插槽的数据进行迁移，迁移后通知客户端新的节点信息。可采用aof或rdb进行迁移。 9. 缓存雪崩：缓存雪崩是指缓存中数据大批量到过期时间，而查询数据量巨大，请求直接落到数据库上，引起数据库压力过大甚至宕机。 解决方案： 1. 均匀过期：设置不同的过期时间，让缓存失效的时间点尽量均匀。可以为有效期增加随机值或者统一规划有效期。 2. 互斥锁：当需要回写时，只允许第一个线程获取缓存，访问数据库，然后加载到缓存中，剩下的线程需要等到该回写线程执行完，再获取缓存。 3. 数据永不过期：对于大量访问的数据，不设置过期时间，而采用异步的方式更新缓存中的数据。 4. 双层缓存策略：使用主备两层缓存，主缓存有效期正常设置，主缓存失效时从数据库加载最新值。而备份缓存中的数据有效期更长，当获取数据失败的时候读取该缓存，当主缓存更新时，同步通过主缓存更新缓存备份，防止失效时直接访问数据库。 10. 缓存穿透：缓存穿透指用户请求的数据不命中缓存，且数据库也没有，每次都需要查询，且返回空值。如果有攻击者不断请求不存在的数据，会导致数据库大量请求，甚至数据库宕机。 解决方案： 1. 布隆过滤器：是一种概率型数据结构，检测集合中是否存在一个特定的元素。通过k个哈希函数对元素计算产生k个哈希值，并以哈希值作为m位的位数组的下标（取余数），将所有k个对应的比特值由0设为1。当查询一个元素时，计算k个哈希值，并查询，如果有某个比特位为0，则该元素一定不在集合中。如果都为1，则可能存在集合中。对于判断不存在的元素，布隆过滤器时间复杂度低，节省空间不需要存储数据。但缺点加入比特位后是不能删除元素。最适用于不需要删除的场景，如黑名单和爬虫Url去重。 2. 返回空对象。如果缓存未命中，查询数据库、也为空，先将空对象写到该key中，请求不会落到数据库，并且给空对象设置过期时间。这样缺点是，在这段时间内可能会存在缓存和持久层数据不一致。 11. 缓存击穿：一个热点key，在失效的瞬间，并发穿破缓存请求数据库，导致数据库压力骤增。 解决方案： 1. 互斥锁：当需要回写时，只允许第一个线程获取缓存，访问数据库，然后加载到缓存中，剩下的线程需要等到该回写线程执行完，再获取缓存。 2. 热点数据永不过期：对于热点数据，不设置过期时间，而采用异步的方式更新缓存中的数据。   3）请模拟一个简单场景，实现缓存读写操作，缓存更新操作，给出缓存的效果，分析2问题中相关策略的效果。  最终：要求所有实验均应用到实际系统中，进行专门讨论。 | | | | |
| 结对开发过程记录 | | | |  |
| **（1）角色切换与任务分工**  表1-1结对开发角色与任务分工   | 日期 | 时间(HH:MM - HH:MM) | 驾驶员角色 | 领航员角色 | 本段时间的任务 | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  |   【注意】该表格可自行增加更多的行。  **（2）工作日志**  由领航员负责记录，记录结对开发期间的遇到的问题、两人如何通过交流合作解决每个问题的。  表1-2 结对开发工作日志   | 日期/时间 | 问题描述 | 最终解决方法 | 交流过程 | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  |   【注意】该表格可自行增加更多的行  **（3）结对开发工作现场照片、或视频及文件沟通截图**  请其他同学帮助拍摄结对开发现场照片至少2张。  13071045_wLhN 1  结对开发现场照片1 结对开发现场照片2 | | | | |
| 实验总结 | | | |  |
| 【结合前期课程项目系统或其他实际软件系统，解决其中的消息订阅/分发问题等】 | | | | |
| 教师评语 | | | |  |
|  | | | | |