**session共享**

我们还用nginx解决session共享的问题，就是我们访问nginx的时候，可能会分配到不同的服务器上，造成session失效，这块的话有三种解决方案：第一种是session或凭据缓存到独立的服务器中，第二种是session或凭据保存到数据库中，第三种是使用nginx的ip\_hash，使得同一ip的请求都指定到同一个服务器上。保存到数据库的这种方式虽然可行，但是需要对数据库进行一些维护等等，比较麻烦，还会涉及到数据的增删改以及同步的问题，使用nginx的ip\_hash，同一ip指定同一个服务器，如果这台服务器出现故障或者问题，就会导致请求失败，此外我们一些文件的上传下载也会遇到这样的问题，可能我们上传的服务器在一个服务器上，下载的请求发送到另外的服务器上，就会出现类似session共享的问题，解决办法有两种：搭建独立的文件服务器或者使用文件压缩数据库。

因此我们选择是nginx+redis+tomcat的方式，使用redis的缓存进行存储session，解决session共享的问题。

### Redis知识点

在我们前台商品展示界面，我们使用redis缓存，将一些前台页面展示的数据放到redis中，减少访问数据库的次数，减缓服务器的压力。当初在进行技术选型的时候，我们主要对redis和memcache进行了一个对比。

1. redis与memcache的对比

Memcached是多线程，而Redis使用单线程.

1)存储方式

memecache 把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小

redis支持数据的持久化可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。用于数据恢复

持久化方式：有快照（rdb）和AOF日志两种持久化方式，在实际应用的时候，要特别注意配置文件快照参数，要不就很有可能服务器频繁满载做dump）

2)数据支持类型：

redis在数据支持上要比memecache多的多。不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。

3)数据备份

Redis支持数据的备份，即master-slave模式的数据备份，防止单点故障

4)底层数据模型

使用底层模型不同： redis直接自己构建了VM 机制

对于两者的选择还是要看具体的应用场景，你需要关注关注数据持久化和主从复制，排序等一系列复杂的操作时，毫无疑问选择redis，如果你的目标是构建一个缓存在升级或者重启后之前的数据不会丢失的话，那也只能选择redis。如果需要缓存的数据只是key-value这样简单的结构时，我在项目里还是采用memcache它也足够的稳定可靠。

Ehcache是进程内缓存，memcached和redis是分布式缓存，分布式缓存会比较方便你以后项目的横向扩展；如果应用比较简单Ehcache会更加适合

Redis的应用

哨兵模式也就是在主从上加了监控，所以主从模式相对更简单，但无借助外力的话不是高可用的架构。

2、Redis两种持久化方式

持久化，redis的数据会存储到硬盘上；（rdb，aof）

优点

性能极高 – Redis能支持超过 100K+ 每秒的读写频率。

丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。

原子 – Redis的所有操作都是原子性的，同时Redis还支持对几个操作全并后的原子性执行。

丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。

1.数据经常被读取的地方，如电商购物车中的价格、数量等；

2.它应用在一些能大大减轻数据库压力的地方。

3、Redis分区方式

两种方式：一种是顺序分区，一种是hash分区

4、Redis的缓存穿透

我们在项目中使用缓存通常都是APP先检查缓存中是否存在，如果存在直接返回缓存内容，如果不存在就直接查询数据库然后再缓存查询结果返回。这个时候如果我们查询的某一个数据在缓存中一直不存在，就会造成每一次请求都查询DB，这样缓存就失去了意义，在流量大时，可能DB就挂掉了。

解决办法：果碰到这样的问题可以在封装的缓存SET和GET部分增加个步骤，如果查询一个KEY不存在，就已这个KEY为前缀设定一个标识KEY；以后再查询该KEY的时候，先查询标识KEY，如果标识KEY存在，就返回一个协定好的非FALSH或者NULL值，然后APP做相应的处理，这样缓存层就不会被穿透。当然这个验证KEY的失效时间不能太长。

针对缓存失效或者缓存并发，使用的方法是：设置缓存失效的时间是1~5分钟内的一个随机值，避免高并发情况下缓存同时失效。

5、Redis的缓存雪崩

缓存层宕掉后，流量会像奔逃的野牛一样，打向后端存储

6、Redis主从同步

开发阶段测试时候采用1主3从。真正上线不清楚，外包过去的项目结束，我们就撤回了。

原理：

第一次Slave向Master同步的实现是：

Slave向Master发出同步请求，Master先dump出rdb文件，然后将rdb文件全量传输给slave，然后Master把缓存的命令转发给Slave，初次同步完成。

第二次以及以后的同步实现是：

Master将变量的快照直接实时依次发送给各个Slave。但不管什么原因导致Slave和Master断开重连都会重复以上两个步骤的过程。

Redis的主从复制是建立在内存快照的持久化基础上的，只要有Slave就一定会有内存快照发生。

Redis主从同步比重新插入速度快、效率高。

同步是服务器和服务器之间通讯的功能，事务已经结束，只需要赋值。

插入必须开启事务。

### **Springmvc工作流程：**

1. 用户请求发送到前端控制器DispatcherServlet
2. DispatcherServlet收到请求调用HandlerMapping处理器映射器
3. 处理器映射器找到具体的处理器（根据xml配置、注解进行查找），生成处理器对象及处理器拦截器一并返回给DispatcherServlet
4. DispatcherServlet调用HandlerAdapter处理器适配器
5. HandlerAdapter经过适配调用具体的处理器
6. Controller执行完成之后返回ModelAndView
7. HandlerAdapter将Controller执行结果ModelAndView返回给DispatcherServlet
8. DispatcherServlet将ModelAndView传给ViewReslover试图解析器
9. ViewReslover解析后返回具体的视图view
10. DispatcherServlet根据view进行渲染视图
11. DispatcherServlet响应用户