

为什么要采用分布式

- 用户访问量大(万级别),并发量大
- 数据量大(千万级别) ,不可能直接查询数据库
- 如果用户量极少,并不需要使用分布式

系统架构的演变

架构	特点	优点	缺点
单体架构	一只猫部署所有项目	开发快速	维护性低,耦合度高,安全性低
前后端分离	不同项目不同猫	安全性高	
项目集群	同一个项目多只猫	性能,可靠性,灵活性高	
分布式/微服务	一群猫部署一个项目的不同模块	负载均衡 ,灵活调度	

为什么使用redis作为缓存

- 因为采用了分布式集群,不同的服务间无法共享session中的数据
- 而且并发量大,不适合直接查询sql数据库
- 需要一个公共的区域来供各个服务存储/读取相关的信息
- 这个公共区域就是redis缓存

redis设计细节

前缀设计

- 实现序列化接口(需要在服务间传输,用到流)
- 创建BasePrefix(String prefix,long time)
- 子类调用其构造器并创建静态的前缀对象供调用

token:

	key	value
cookie	"TOKEN_COOKIE_NAME"	"user_info_token"+token(uuid)
redis	"user_info_token"+token(uuid)	JSONString(userInfo)

有区别的请求控制:

- 例如登录控制@RequiredLogin
- 创建一个注解,贴在需要区别的请求方法上
- 创建一个拦截器,拦截贴有注解的方法

- 在启动类中配置添加拦截器

spring参数注入

- 无法直接从session获取userInfo对象,于是自定义参数解析器,从缓存中取出userInfo对象
- UserInfoArgumentResolver //自定义参数解析器
 - supportsParameter //判断字段上注解和参数类型是否对应
 - resolveArgument //若上面的方法返回true,返回自定义的注入对象(cookie->redis->userInfo)
- 启动类中添加自定义参数解析器

第三方api调用
















- 本次项目仅仅是调用短信api的调用,可以拓展到其他api的调用
- 其本质其实就是java代码模拟浏览器发送请求,让接口进行某些操作,再解析接口返回的响应
- 也可以使用HttpClient https://blog.csdn.net/w_372426096/article/details/82713315
- 还可以延伸到爬虫,结合正则表达式,爬取网络资源

mongoDB

- domain对应document文档
- MongoTemplate mongoDB自带的模板
- MongoRepository JPA规范处理,底层还是在使用模板

mongoDB es redis mysql

- 对数据的读写要求极高,并且你的数据规模不大,也不需要长期存储,选redis;
- 数据规模较大,对数据的读性能要求很高,数据表的结构需要经常变,有时还需要做一些聚合查询,选MongoDB;
- 需要构造一个搜索引擎或者你想搞一个看着高大上的数据可视化平台,并且你的数据有一定的分析价值或者你的老板是土豪,选ElasticSearch;
- mysql是关系型数据库,数据存储在磁盘中,nosql数据初始化需要读取mysql,而数据落地需要将数据同步到mysql

支持情况	差	一般	好	极好
数据规模	 redis	 elastic	 mongoDB	 APACHE HBASE
查询性能	 APACHE HBASE	 elastic	 mongoDB	 redis
写入性能	 elastic	 mongoDB	 APACHE HBASE	 redis
复杂查询、检索功能	 redis	 APACHE HBASE	 mongoDB	 elastic

知乎 @麦田里的老农

<https://blog.csdn.net/huailiang01>

数据统计缓存

- 对象设计 实现
- 方案对比 优缺点

项目	前缀	key	value
统计	"strategy_statis_vo"	sid	vo
收藏	"strategy_statis_favor"	sid	list<uid>
点赞	"strategy_statis_thumbsup"	sid:uid	"yyyy-MM-dd"

Spring事件监听器

- 缓存数据初始化
- spring容器启动后,会执行onApplicationEvent方法
- 将mysql中的数据, 同步到redis缓存中

Spring定时器

- 缓存数据同步到数据库/数据落地
- RedisDataPersistenceJob
- `@Scheduled(cron = "0/10 * * * * ? ")` //定时任务执行计划

ZSET

- zset的每一个成员都有一个分数与之对应,可用于排序操作

key	value
"strategy_statis_commend_sort"	zset<"vo:sid">
"strategy_statis_commend_hot"	zset<"vo:sid">

ElasticSearch

- 原理:倒排索引

正排:

文档编号	文档内容
1	谷歌地图之父跳槽Facebook
2	谷歌地图之父加盟Facebook
3	谷歌地图创始人拉斯离开谷歌加盟Facebook
4	谷歌地图之父跳槽Facebook 与Wave项目取消有关
5	谷歌地图之父拉斯加盟社交网站Facebook

倒排:

单词ID	单词	倒排列表 (DocID)
1	谷歌	1,2,3,4,5
2	地图	1,2,3,4,5
3	之父	1,2,4,5
4	跳槽	1,4
5	Facebook	1,2,3,4,5
6	加盟	2,3,5
7	创始人	3
8	拉斯	3,5
9	离开	3
10	与	4
11	Wave	4
12	项目	4
13	取消	4
14	有关	4
15	社交	5
16	网站	5



单词ID	单词	倒排列表 (DocID;TF)
1	谷歌	(1;1),(2;1),(3;2),(4;1),(5;1)
2	地图	(1;1),(2;1),(3;1),(4;1),(5;1)
3	之父	(1;1),(2;1),(4;1),(5;1)
4	跳槽	(1;1),(4;1)
5	Facebook	(1;1),(2;1),(3;1),(4;1),(5;1)
6	加盟	(2;1),(3;1),(5;1)
7	创始人	(3;1)
8	拉斯	(3;1),(5;1)
9	离开	(3;1)
10	与	(4;1)
11	Wave	(4;1)
12	项目	(4;1)
13	取消	(4;1)
14	有关	(4;1)
15	社交	(5;1)
16	网站	(5;1)

图6 带有单词频率、文档频率和出现位置信息的倒排索引

- 应用场景:项目/站内搜索
- 精确搜索
- 全文搜索
- 高亮显示

参考文章

<http://bridgeforyou.cn/>

<https://zhuanlan.zhihu.com/p/37964096>

拓展

- 分布式事务
- 消息队列
- JPA/MyBatis+