1、面试题

集群部署时的分布式session如何实现？

2、面试官心里分析

面试官问了你一堆dubbo是怎么玩儿的，你会玩儿dubbo就可以把单块系统弄成分布式系统，然后分布式之后接踵而来的就是一堆问题，最大的问题就是分布式事务、接口幂等性、分布式锁，还有最后一个就是分布式session。

当然了，分布式系统中的问题何止这么一点，非常之多，复杂度很高，但是这里就是说下常见的几个，也是面试的时候常问的几个。

3、面试题剖析

session是啥？浏览器有个cookie，在一段时间内这个cookie都存在，然后每次发请求过来都带上一个特殊的jsessionid cookie，就根据这个东西，在服务端可以维护一个对应的session域，里面可以放点儿数据。

一般只要你没关掉浏览器，cookie还在，那么对应的那个session就在，但是cookie没了，session就没了。常见于什么购物车之类的东西，还有登录状态保存之类的。

这个不多说了，懂java的都该知道这个。

但是你单块系统的时候这么玩儿session没问题啊，但是你要是分布式系统了呢，那么多的服务，session状态在哪儿维护啊？

其实方法很多，但是常见常用的是两种：

（1）tomcat + redis

这个其实还挺方便的，就是使用session的代码跟以前一样，还是基于tomcat原生的session支持即可，然后就是用一个叫做Tomcat RedisSessionManager的东西，让所有我们部署的tomcat都将session数据存储到redis即可。

在tomcat的配置文件中，配置一下

<Valve className="com.orangefunction.tomcat.redissessions.RedisSessionHandlerValve" />

<Manager className="com.orangefunction.tomcat.redissessions.RedisSessionManager"

host="{redis.host}"

port="{redis.port}"

database="{redis.dbnum}"

maxInactiveInterval="60"/>

搞一个类似上面的配置即可，你看是不是就是用了RedisSessionManager，然后指定了redis的host和 port就ok了。

<Valve className="com.orangefunction.tomcat.redissessions.RedisSessionHandlerValve" />

<Manager className="com.orangefunction.tomcat.redissessions.RedisSessionManager"

sentinelMaster="mymaster"

sentinels="<sentinel1-ip>:26379,<sentinel2-ip>:26379,<sentinel3-ip>:26379"

maxInactiveInterval="60"/>

还可以用上面这种方式基于redis哨兵支持的redis高可用集群来保存session数据，都是ok的

（2）spring session + redis

分布式会话的这个东西重耦合在tomcat中，如果我要将web容器迁移成jetty，难道你重新把jetty都配置一遍吗？

因为上面那种tomcat + redis的方式好用，但是会严重依赖于web容器，不好将代码移植到其他web容器上去，尤其是你要是换了技术栈咋整？比如换成了spring cloud或者是spring boot之类的。还得好好思忖思忖。

所以现在比较好的还是基于java一站式解决方案，spring了。人家spring基本上包掉了大部分的我们需要使用的框架了，spirng cloud做微服务了，spring boot做脚手架了，所以用sping session是一个很好的选择。

pom.xml

<dependency>

<groupId>org.springframework.session</groupId>

<artifactId>spring-session-data-redis</artifactId>

<version>1.2.1.RELEASE</version>

</dependency>

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.8.1</version>

</dependency>

spring配置文件中

<bean id="redisHttpSessionConfiguration"

class="org.springframework.session.data.redis.config.annotation.web.http.RedisHttpSessionConfiguration">

<property name="maxInactiveIntervalInSeconds" value="600"/>

</bean>

<bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig">

<property name="maxTotal" value="100" />

<property name="maxIdle" value="10" />

</bean>

<bean id="jedisConnectionFactory"

class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory" destroy-method="destroy">

<property name="hostName" value="${redis\_hostname}"/>

<property name="port" value="${redis\_port}"/>

<property name="password" value="${redis\_pwd}" />

<property name="timeout" value="3000"/>

<property name="usePool" value="true"/>

<property name="poolConfig" ref="jedisPoolConfig"/>

</bean>

web.xml

<filter>

<filter-name>springSessionRepositoryFilter</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSessionRepositoryFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

示例代码

@Controller

@RequestMapping("/test")

public class TestController {

@RequestMapping("/putIntoSession")

@ResponseBody

public String putIntoSession(HttpServletRequest request, String username){

request.getSession().setAttribute("name", “leo”);

return "ok";

}

@RequestMapping("/getFromSession")

@ResponseBody

public String getFromSession(HttpServletRequest request, Model model){

String name = request.getSession().getAttribute("name");

return name;

}

}

上面的代码就是ok的，给sping session配置基于redis来存储session数据，然后配置了一个spring session的过滤器，这样的话，session相关操作都会交给spring session来管了。接着在代码中，就用原生的session操作，就是直接基于spring sesion从redis中获取数据了。

实现分布式的会话，有很多种很多种方式，我说的只不过比较常见的两种方式，tomcat + redis早期比较常用；近些年，重耦合到tomcat中去，通过spring session来实现。