几周前，有个盆友问老王，说现在有多台服务器，怎么样来解决这些服务器间的session同步问题？老王一下就来精神了，因为在n年以前，老王还在学校和几个同学一起所谓创业的时候，也遇到了类似的问题。当时查了很多资料，没有解决，于是后来投身百度，终于学到了“葵花宝典”，方才大彻大悟。所以，今天想跟大家分享一下关于session同步的那些事儿。

秉着问题驱动的原则，老王先提几个问题：

1、什么是session？什么又是cookie？他俩有啥联系和区别？

2、为什么要在多台服务器间进行session的共享同步？

3、以及有哪些方法来实现这个同步？

大家快搬板凳，老王开始扯淡咯~

**1、session和cookie的缠绵与悱恻**

相信有盆友跟老王一样，曾经为session和cookie纠结过，或者现在正在为他们纠结。session在英文里的意思是[会议](javascript:;)，而cookie则是饼干。你说这个会议和饼干怎么就关联上了呢？（开会的时候可以吃饼干）

我们先来看看百度百科的解释吧：

A、cookie：

Cookie，有时也用其复数形式Cookies，指某些网站为了辨别用户身份、进行session跟踪而储存在用户本地终端上的数据（通常经过加密）。Cookie是由服务器端生成，发送给User-Agent（一般是浏览器），浏览器会将Cookie的key/value保存到某个目录下的文本文件内，下次请求同一网站时就发送该Cookie给服务器（前提是浏览器设置为启用cookie）

B、session：

在计算机中，尤其是在网络应用中，称为“会话控制”。Session对象存储特定用户会话所需的属性及配置信息。这样，当用户在应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。当用户请求来自应用程序的 Web 页时，如果该用户还没有会话，则 Web 服务器将自动创建一个 Session 对象。当会话过期或被放弃后，服务器将终止该会话。

大家看懂了嘛？我打包票，肯定还是有盆友没看懂。老王自己是这么来理解他们的：

A、cookie：

浏览器请求服务器，服务器为了区别不同的用户请求，就需要给他们打上标签，比如：发放一个访问令牌(access\_token)给客户端。发放的过程是通过在HTTP请求返回的时候，通过设置HTTP的header：Set-Cookie来实现的。



以上就是我请求百度，他给我发放的cookie们。每一个Set-Cookie里一般会含有设置的key=value、过期时间，以及域和路径。

当浏览器接收到这样的返回头以后，就把他稳稳当当的存起来，以后每次发送请求的时候，就会把他带上（具体还要看过期时间、作用的域和路径）。

这个cookie看起来像个什么东东呢？像不像有关部门给我们发放的身份证？你去有关部门申请，他就把你的ID、性别、年龄等等信息给你打到一张叫做身份证的东东上，然后发给你。以后你每次去办点啥关键的事情，就需要带上这些cookie们。

一般服务器会在浏览器里种上一些类似于访问令牌（access\_token）、用户ID（user\_id）等等的cookie，这样你一去访问对应的网站，他就把你认出来了。特别，像java的服务器，还会种一些类似jsession\_id的cookie，服务器采用一定的算法（比如随机算法），生成一个一定长度（比如10字节）的字符串" angOwberup "，然后发放给浏览器： Set-Cookie：jssesion\_id= angOwberup，当浏览器收到这个cookie以后，就跟拿到宝一样，好好的把这个key和value收藏了起来，以后每次去服务器请求都带上。

B、session：

与此同时，服务器把这个字符串"angOwberup"作为key，把一个叫做User的类的一个实例user，设置好id、nickname等等信息以后，放入了一个类似于map的容器里：map.put("angOwberup", user)。当浏览器请求来的时候，服务器就会getCookie("jsession\_id")，把这个种在浏览器里的字符串取出来，然后用这个字符串去map里找找，看看有没有对应的User对象：map.get(sessionId)。如果取到了，说明就找到了这个用户的id、nickname等等信息，直接就可以在网页上显示：“老王你好，欢迎回来！”。如果没有找到，有可能就跳到登录页面，让用户做登录。

我们把用户在一定时间内访问某个网站时，请求不同页面的过程叫做一个会话，也就是session。在同一个session里，我们可以记录用户访问的状态和信息。这样，那个类似于map的容器就是session管理器。

打个形象的比喻，如果cookie是身份证，那session就是你的档案。你的所有信息都存放在档案里，有关部门（server）管理着你的档案。当你要办重要的事情时，就需要拿着身份证去有关部门提取档案，有关部门查阅档案后，再看要不要给你办事儿。如果你做了坏事，他们就会往你的档案（session）里写一些不好的东西；当然，如果你得了什么奖，也会往里面放。

这下，是不是有点清楚cookie和session有什么联系和区别了呢？再简要的总结一下：

A、cookie就是服务器发放给客户端的一些标识，让客户端记住每次请求的时候带上，以区分不同的用户；

B、session是服务器存放在自己那里的用户相关的数据，用每次用户带来的cookie去提取出来，恢复一个之前访问的历史或者相关环境。

好了，有了上面的内容，接下来，我们就需要讨论一下那个类似于map的session管理器了。

**2、session的管理**

上面说了，服务器用了一个类似于map的容器来管理session。那具体来看，这个map是怎么样来实现的呢？

不同的服务器、不同的语言框架都有不同的实现。比如java的服务器，有的是用文件方式来存储的、有的是用内存cache的方式来存储的。老王还听说有的语言的服务器将数据做加密，然后设置成cookie，存到了客户端（浏览器）。那这些实现方式都有哪些优缺点呢？我们逐个来分析。（当然，有可能还有其他的实现方法，老王可能不了解，不过大体思路相似，如有遗漏请指正）

A、文件方式：这种方式，将文件作为一个map，当新增一个数据的时候，就在文件中增加类似这样的一条数据：

angOwberup =>

data={"user":{"id":1,"nickname":"老王"}};

expiry="2016-10-0100:00:00"

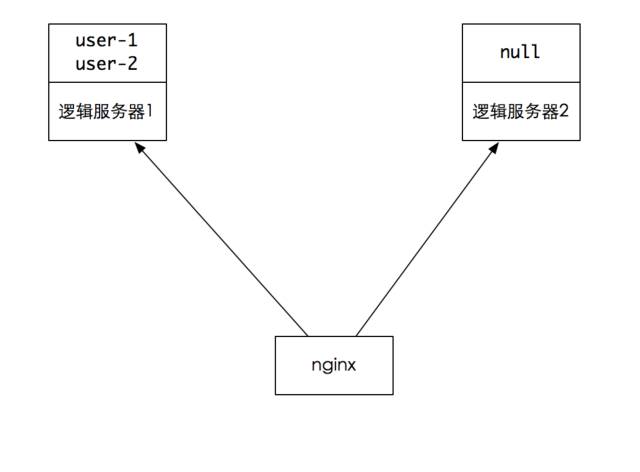
(当然，具体实现的时候有可能是用的二进制方式，而不是字符串)

这种方式的好处，就是能够存储大量的用户session，使得这个session有效期可以比较长（比如：三个月用户不用登录）。不过这个方式也有对应的问题，就是文件操作比较麻烦。比如，有一个用户的session过期了，需要删掉这条记录，那这个文件就需要挪动或重写。

B、cache方式：有好多web端的逻辑服务器都采用这种方式。这种方式好处非常明显，就是实现起来非常简单。将所有数据放入到内存cache中。如果有失效，直接内存删除就可以了。不过带来的问题也很明显，当服务器重启以后，所有session都丢失了。或者当有大量用户登录（也有可能是遭受攻击），就会很快让cache被充满，然后大量session被LRU算法淘汰，造成session的大量失效，使得用户需要反复登录等操作。

C、cookie方式：这种方式是最偷懒的方式。就是我服务器任何数据都不存，我把你们所有的客户端当做我的存储器，我就需要做一个加密和解密操作。当然这种方式最大的好处就是实现极其简单（还有其他的好处，稍后再说），不过问题也是很明显的，就是客户端要记录大量信息，同时还要保证加密信息的安全。如果session里要存放大数据，这种方式就不是很适合了。

除了上述说到的优缺点以外，A、B两种方式还有另外一个问题，就是当我有不止一台服务器的时候，不同服务器间的session数据共享就成问题了。



比如，最初我只有一台服务器1，他的session里记录了user-1和user-2的数据。这个时候，我需要增加一台服务器2。当nginx把用户的请求转发到服务器2的时候，他就傻眼了：用户带了一个jsession\_id=angOwberup这个的cookie过来，而在他的session管理器里却找不到这样一个session数据。那该怎么办？！（苦！恼！啊！）

因此，就出现了我们文章一开始提到的问题：在分布式系统里，用户session如何才能实现同步？

**3、session的同步**

有了上面的情况，我们就必须要去考虑，如何在多个服务器之间实现session同步这个操作。常见的做法有以下几种，我们逐个来看看：

A、进程间通信传递session数据。

这是最容易想到的一个方法。我们在不同的server服务里开一个socket，然后用socket来将相互拥有的session数据进行传递。我记得多年以前tomcat就是采用这样的方式来做的（已经很久没用过tomcat了，不知道现在是否还在这样使用）。

这种方式的好处很明显，就是原理简单明了；坏处也很明显，就是同步合并过程复杂，还容易造成同步延迟。比如，某个用户在server-1登录了，server-1存储了这个用户的session，当正准备将数据同步给server-2的时候，由于用户访问实在是太快（飞一般的速度），server-2还没收到server-1传来的session数据，用户访问就已经来了。这个时候，server-2就不能识别这个用户，造成用户需要再次登录。

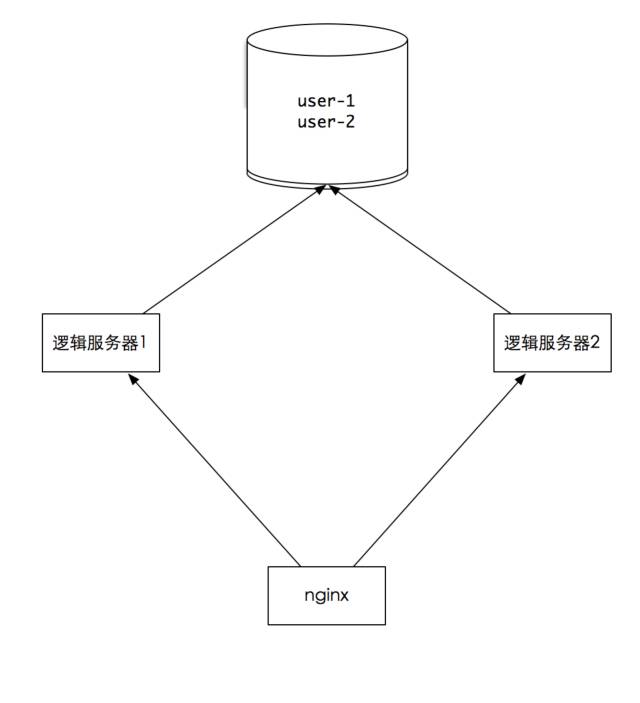
而且，当有成千上万台服务器的时候，session同步就是一个噩梦：每一个服务器都要将自己拥有的session广播给其他所有机器，而且还要随时进行，不能停歇…… （最后这些机器估计都是累死的）

B、cookie存储方式。我们在上面讲到了一个很偷懒的方式，就是把session数据做加密，然后存储到cookie中。用户请求到了，就直接从cookie读取，然后做解密。这种方式真是把分布式思想发挥到了一个相当的高度。他把用户也当做分布式的一员，你要访问数据，那你就自己携带着他，每次到服务器的时候，我们的服务器就只负责解密……

对于session里只存放小数据，并且加密做的比较好（防止碰撞做暴力破解）的系统来讲，这是一个比较好的选择。他实现超级简单，而且不用考虑数据的同步。

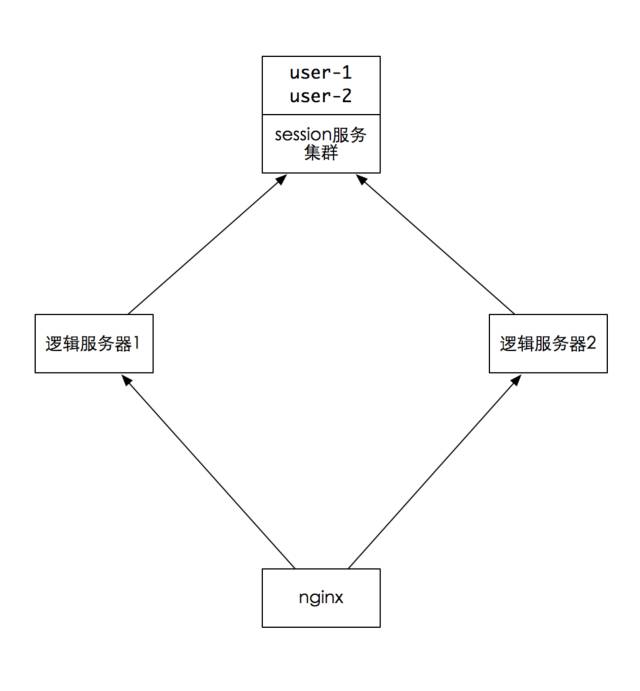
不过如果要往session里存放大数据的情况就不是太好处理。或者安全性要求很高的系统，也不是太好的一个方式（数据有被破解的风险）。

C、cache集群或者数据库做session管理。我们也可以采用另外一种架构来解决session同步问题，那就是引入统一session接入点。



我们session放入到cache集群或者数据库中，每次请求的时候，都从他们中来获取。这样，所有的机器都能获取到最新的session数据。这种方案也是很多中大型网站采用的解决方案。他实现起来相对简单（利用cache集群或者主从数据库自身的管理来实现多机的互备），而且效率很高，安全性也不错。

D、还有一种方式是从上面这种方式延展出来的，就是提供session服务。这个服务负责管理session，其他服务器每次从这个服务处获取session数据，从而达到数据的共享。



大家如果仔细观察一下baidu或者google，你做登录的时候，他们可能会让你跳到[passport.baidu.com](http://passport.baidu.com/) 或者[accounts.google.com](http://accounts.google.com/" \t "_blank)这两个域名之下。这两个就是他们用来做用户登录和类似session管理的一个地方（由于之前只呆过baidu，所以google并不是非常清楚）。当一个访问请求来的时候，server就从cookie里取类似session\_id的东东，然后用这个东东去passport服务去请求用户的session数据。

这种方式的好处就在于：

A、可以非常方便的扩展用户登录的数量以及存储数据的大小。当时在x度的时候，N亿用户的session都在这个系统里进行管理；

B、方便做性能优化。如果用cache集群的方案，如果cache有机器坏掉，那么就会造成一部分用户session失效；如果用数据库方案，如果量太大，有可能会出现性能问题。而这种方案在实现的时候，可以用cache和数据库结合的实现方式，保证高效和稳定。同时，针对一些接口，可以做性能的优化，提升查询效率；

C、对外封闭，保证数据安全。这种方式还有一个好处，就是可以将加密算法、密钥等封闭在系统内部，对外只暴露接口，使得数据安全性更有保障。（涉及到用户信息的，都是隐私！）

不过，这种方式也有自己的问题，就是运维相对更复杂，有可能需要专门的团队去管理这些系统。

当然，除了上述的一些方式以外，还有其他的手段（比如，在入口nginx处对用户cookie做一致Hash，将某一用户分配到固定机器）。鉴于老王知识有限，且码字速度有限，就先介绍这些了，不知道你是否看懂了呢？

总结一下：

关于session同步，其实方案有很多，没有哪个方案是最好的，只有某一种方案是最适合你现在架构的。所以，老王抛了几了解决方案，供大家了解。如果在业务中能够使用到，老王就很开心了~

好了，老王今天就先写这么多吧，如果想听老王继续扯淡，请每周日下午带上你的小板凳，来老王的微信公众号：simplemain，老王跟你不见不散~

今天有个号外：大家看到的文章头图，是家里领导做的曲奇饼干，因为涉及到知识产权同时为了讨好领导，在此郑重声明—— Image By Pure！ ^\_^



