目前的web应用集群，使用Nginx做负载均衡，将upstream配置成ip\_hash的方式，这种模式下，Nginx会根据来源IP和后端配置来做hash分配，确保固定IP只访问一个后端。

upstream YOUR\_NAME {

ip\_hash;

server 192.168.8.15:8080;

server 192.168.8.17:8080;

}

但是，由于固定某个IP只能访问单独的一个后端，如果宕机或者需要升级程序时做停机重启，正在操作的用户就会退出到登录页面，不仅用户体验很差，而且正在做的操作不能保证成功，容易产生脏数据等。

**从Nginx upstream配置说起**

首先，来看一下Nginx upstream的几种负载均衡策略：

1）轮询

upstream YOUR\_NAME {

server 192.168.8.15:8080;

server 192.168.8.17:8080;

}

2）权重：该策略可解决服务器性能不等的情况下轮询比率的调配

upstream YOUR\_NAME {

server 192.168.8.15:8080 weight=2;

server 192.168.8.17:8080 weight=3;

}

3）ip\_hash

upstream YOUR\_NAME {

ip\_hash;

server 192.168.8.15:8080;

server 192.168.8.17:8080;

}

4）fair：需要安装[Upstream Fair Balancer](http://wiki.nginx.org/HttpUpstreamFairModule) Module。该策略根据后端服务的响应时间来分配，响应时间短的后端优先分配

upstream YOUR\_NAME {

server 192.168.8.15:8080;

server 192.168.8.17:8080;

fair;

}

5）一致性Hash：需要安装[Upstream Consistent Hash](https://www.nginx.com/resources/wiki/modules/consistent_hash/) Module，该策略可以根据给定的字符串进行Hash分配，具体参见官方Wiki。

由此可见，我们迫切的需要使用轮训的方式来做负载均衡，那对于大规模集群部署的web应用来讲，轮训的方式就要Session必须进行共享。

**Session共享机制**

在集群系统下实现Session共享机制一般有如下两种方案：

* 应用服务器间的Session复制共享（如Tomcat自带的Session共享）
* 基于缓存数据库的Session共享（如使用Memcached、Redis）

**应用服务器间的Session复制共享**

Session复制共享，主要是指集群环境下，多台应用服务器之间同步Session，使其保持一致，对外透明。如果其中一台服务器发生故障，根据负载均衡的原理，Web服务器（Apache/Nginx）会遍历寻找可用节点，分发请求，由于Session已同步，故能保证用户的Session信息不会丢失。

此方案的不足之处：

* 技术复杂,必须在同一种中间件之间完成(如Tomcat-Tomcat之间).
* Session复制带来的性能损失会快速增加.特别是当Session中保存了较大的对象,而且对象变化较快时, 性能下降更加显著. 这种特性使得Web应用的水平扩展受到了限制。
* Session内容序列化（Serialize），会消耗系统性能。
* Session内容通过广播同步给成员，会造成网络流量瓶颈，即便是内网瓶颈。

**基于缓存数据库的Session共享**

基于缓存数据库的Session共享是指使用如Memcached、Redis等Cache DB来存取Session信息：应用服务器接受新请求将Session信息保存到Cache DB中，当应用服务器发生故障，Web服务器（Apache/Nginx）会遍历寻找可用节点，分发请求，当应用服务器发现Session不在本机内存，则会去Cache DB中查找，如果找到，则复制到本机，这样就实现了Session的共享和高可用。

我选用的是Redis而不是Memcached，是因为Redis具有更丰富的数据结构，比如可以为Key指定过期时间，从而不需要我们定期的刷新缓存。而Memcached做不到，所有就有了这样一个合理的方案——

在GitHub有这样一个开源工具[tomcat-redis-session-manager](https://github.com/jcoleman/tomcat-redis-session-manager)，可以帮我们实现基于Redis的Session共享，然而直接拿来用的话，Session的key直接就是SessionID，没有一个统一的前缀，所以经过一些小改造，代码已托管到[这里](https://github.com/2hf/customized-tomcat-redis-session-manager)，可以通过Tomcat/conf/server.xml的最下面的中增加sessionCookieName配置你想要的Redis中key的前缀，如下所示：

<Context docBase="/root/YOUR\_WEB\_APP"

path=""

reloadable="true" />

闲话少说，下面开始讲解如何使用：  
1）下载源码编译成Jar包，讲 tomcat-redis-session-manager-1.2.jar 、jedis-2.6.1.jar、commons-pool2-2.2.jar拷贝到Tomcat目录下的lib中（Jedis、commons-pool2版本任意）  
2）在Tomcat的conf目录下，编辑context.xml。如果你是用Redis单点，则可以仿照如下配置：

<Context sessionCookieName="ECSCSessionID">

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

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

host="192.168.8.38"

port="6379"

database="0"

maxInactiveInterval="60" />

如果是Redis集群环境，则可仿照如下配置：

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

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

database="0"

maxInactiveInterval="60"

sentinelMaster="mymaster"

sentinels="192.168.8.43:26379,192.168.8.45:26379,192.168.8.47:26379"/>

参数均可选，详见上面tomcat-redis-session-managerGithub中的说明。

**关于maxInactiveInterval，即失效时间，这里做一些说明：**

即使在这里配置的maxInactiveInterval是60s，如果web.xml配置了session的失效时间，则以web.xml为准。  
另，  
如果有一下三处配置了Session的失效时间，则下面的配置覆盖上面的配置:

* TOMCAT\_HOME/conf/web.xml
* WebApplication/webapp/WEB-INF/web.xml
* 写在代码中的值 : HttpSession.setMaxInactiveInterval(int)

即实际生效顺序是:  
HttpSession.setMaxInactiveInterval(int) > $WebApplication/webapp/WEB-INF/web.xml > $TOMCAT\_HOME/conf/web.xml

**关于需要密码的Redis环境，这里做一些说明：**

只需要在context.xml中增加password=”XXXX”即可（单点和集群同理），如下所示：

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

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

database="1"

maxInactiveInterval="60"

sentinelMaster="mymaster"

sentinels="192.168.8.43:26379,192.168.8.45:26379,192.168.8.47:26379"

password=”Your\_Redis\_Password”/>

启动Tomcat，访问应用，即可在Redis中看到效果。

关于测试，可以将Nginx Upstream配置为轮询后，仅留一台应用服务器启动，登陆操作，然后启动另外一台，停止第一台服务，继续操作，发现并未受任何影响，即可。

**参考**

nginx upstream的几种配置方式：<http://alwaysyunwei.blog.51cto.com/3224143/1239182>  
Load Balancing via Nginx Upstream :<http://nginx.org/en/docs/http/load_balancing.html>  
Tomcat7基于Redis的Session共享：<https://yq.aliyun.com/articles/1298>  
Tomcat Session Timeout Web.xml: <http://stackoverflow.com/questions/13463036/tomcat-session-timeout-web-xml>