# 三一集团统一会话共享平台论文稿

## 概述

**作 用：**为应用提供**统一会话管理**功能，**避免**集群部署场景下负载切换**session丢失**问题

**存 储：**采用**mongodb**存储会话数据，采用**增量模式**修改会话属性，简单高效

**序列化：**采用自主研发的序列化机制，序列化存储会话数据，提供**序列化插件**，扩展性强

**规 范：遵循servlet 2/3规范，兼容**遗留系统会话数据的管理,**可无缝与现有应用系统集成，无需修改应用代码或者只需修改少数代码。Session监听器需遵循会话共享规范，需使用会话共享实现替换原来容器的session监听器。**

**兼容性：跨容器，**兼容业界主流的应用服务器（tomcat,weblogic,webspere,jetty，jboss等等）

**约 束：无约束，**无需session sticking，访问请求可以平均分派给各集群节点，支持lvs,haproxy，ngix 4,7层负载。如果修改session中属性对象中的属性，必须调用session.setAttribute重新将对象到设置到session中，以便将修改存储到mongodb中。

**安全性：**客户端基于cookie机制存储sessionid，通过设置cookie **httponly**属性阻止XSS窃 取sessionid，通过设置**secure**属性并结合https阻止传输过程中sessionid被窃取

**高 阶：除支持集群应用会话共享，还可**跨应用，跨域共享会话；

**监 管：**session信息统计查询, 应用在线用户数统计查询，应用会话管理功能（包括删除会话、 查看会话属性数据）

## 架构设计

会话共享整体架构：

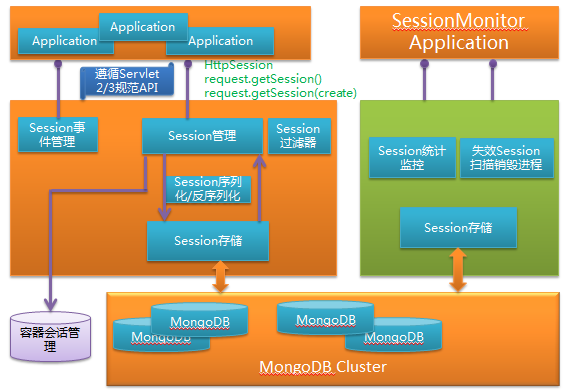


图1.统一会话管理平台体系架构

会话共享存储结构

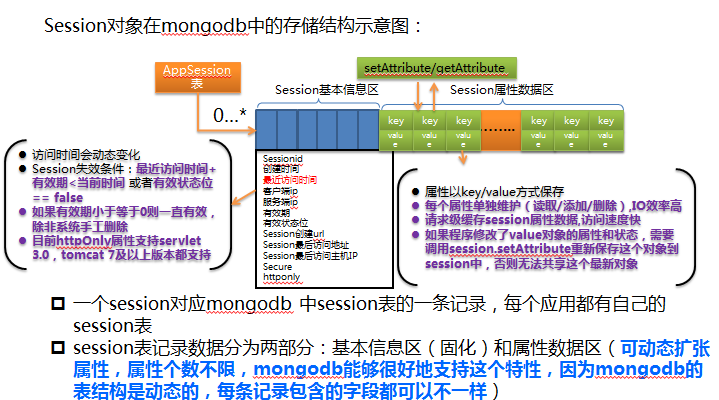
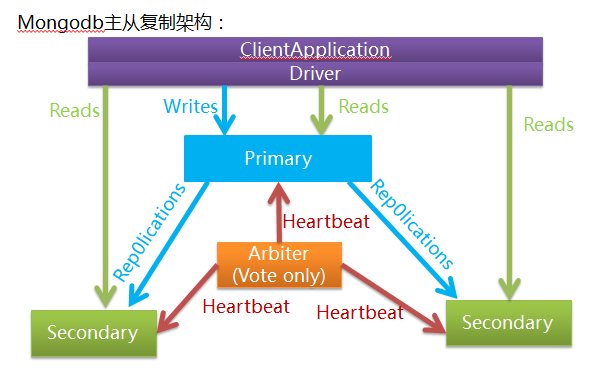
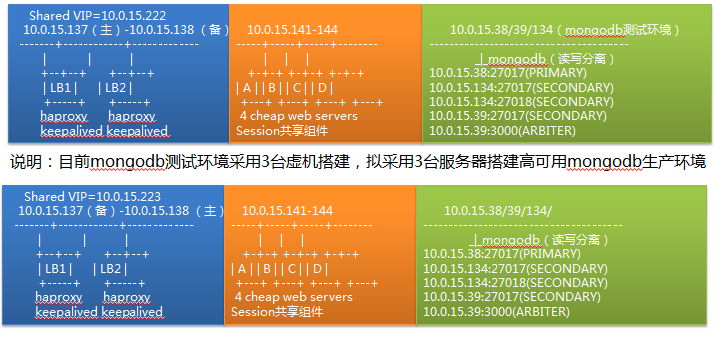


图2.统一会话管理平台mongodb存储结构设计

高可用mongodb服务器部署架构



会话共享在集群环境中地位：



## 三、性能测试

针对测试环境登录和登出场景进行测试，分别测试不同并发下的性能情况，各场景均达到性能标准，简述如下，具体参考各场景的详细测试报告：

1、50并发登录的平均响应时间:1.35s,登出的平均响应时间：0.177s

2、100并发登录的平均响应时间 2.29s，登出的平均响应时间0.295s

3、200并发登录的平均响应时间5.2s，登出的平均响应时间0.358s

4、300并发登录的平均响应时间 7.78s，登出的平均响应时间0.473s

测试结果分析：

随着并发量的增加登录响应时间呈线性增长，主要的影响因素为loadrunner服务器资源受限，在各个级别并发场景下各个应用服务器的cpu使用率保持在20%左右，各个mongodb服务的cpu使用率保持在20%左右

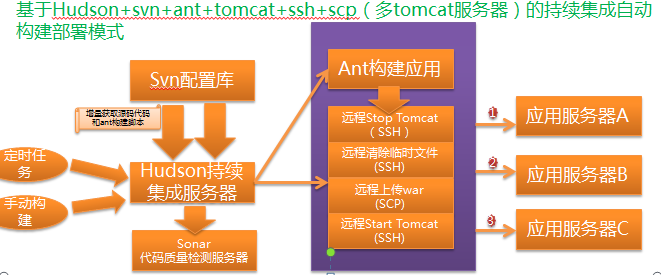
在执行压力测试的同时，不断人工登录，通过监控工具检测的人工登录时间基本为500毫秒到800毫秒之间，少于1秒，登出时间为60毫秒到200毫秒之间，没有因为压力的加大而出现响应时间的波动

基于mongodb的会话共享机制，对登录，登出的影响不大，可以在实际项目中推广使用。

## 四、应用集成

## 五、应用部署

采用会话共享的应用系统一般都会采用集群模式进行部署，商业的应用服务器部署集群应用非常麻烦，代价也非常高昂（采购商业套件，部署成本高，难度大），我们经过实践摸索，形成了一套高效廉价部署方案：



说明：按先后顺序将应用部署到应用服务器，只停用正在部署应用的tomcat服务器，部署完毕后立即启动tomcat，完成后再依次将应用部署其他应用服务器，这样保证部署过程中其他服务器仍然保持工作状态，达到无停机部署和升级应用的目标。

## 六、场景演示

花了近1个小时搭建g4studio开发环境、整合bboss会话共享框架并调整部分g4studio源码程序，终于在g4studio中搞定会话共享功能:lol: ，

至于sarkara测试过程中碰到的问题并不是序列化导致的，具体原因跟我之前回复中提到的内容有关：

[size=x-large][color=blue][b]开启会话共享机制后，如果修改了sessionContainer中的数据，需要将修改后的sessionContainer对象重新设置到session中，否则无法将最新的sessionContainer存储到mongodb中 [/b][/color][/size]

以下是问题分析和解决过程：

g4studio对session的操作不是很规范，登录时，先创建一个sessionContainer对象，然后把这个空的对象设置到共享session中，再完成登录操作，得到一个UserInfoVo对象并将UserInfoVo对象设置到sessionContainer对象中，然后登录就结束了；这个做法在非session共享情况下是没有问题的，但是在session共享情况下就有问题了，最后必须将sessionContainer重新设置到共享session对象中，针对sessionContainer的修改才会生效 :lol:

另外，g4studio登录到首页后有个选择主题的功能，其中也会修改sessionContainer中UserInfoVo对象的一个layout属性，这里也需要重新设置sessionContainer对象到session中；还有g4studio种将一些报表对象也设置到sessionContainer中，每次设置后也需要将sessionContainer重新设置到共享session对象中。由此可以看出sessionContainer是一个非常重量级的session数据对象，建议g4studio 的作者将一些报表对象单独设置到session中，从sessionContainer中剥离出来，这样可以避免每次将一些无关的报表对象重新写入mongodb中。

我已经将所有的修改以及集成了bboss会话共享的commit给g4studio作者，看看作者是否愿意讲这个commit合并到g4studio master主干版本中。

修改的版本可以从以下地址下载：

[url][https://github.com/yin-bp/g4studio[/url](https://github.com/yin-bp/g4studio%5b/url)]

对g4studio详细修改说明参考地址：

<https://github.com/yin-bp/g4studio/commit/d7f802f8f48c05e096ebde4cf1a878217cddd7e7>