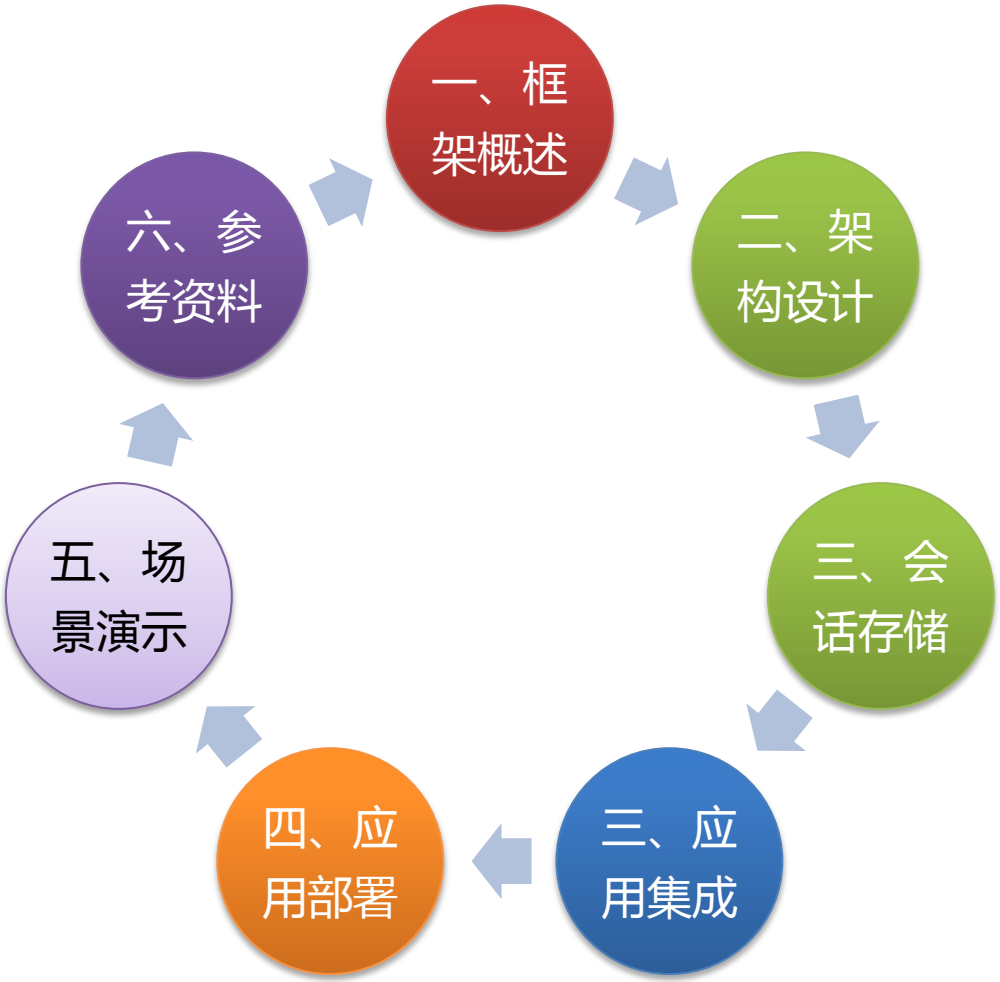
bboss会话共享培训文档

2015 中国·长沙

大纲



会话共享概述

大纲 | **概述** | 架构 | 会话存储 | 应用集成 | 应用部署 | 场景演示

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；跨域跨应用共享会话并实现SSO功能；解决了会话共享五大技术难题：session数据序列化问题，session sticking问题，跨域跨应用session共享问题，跨容器(tomcat,jetty,weblogic)共享session问题，sso单点登入单点登出一致性问题。

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bboss序列化机制以xml格式序列化会话数据，可读性好，易于监控，提供**序列化插件**，扩展性强

规　范：**遵循servlet 2/3规范，可无缝与现有应用系统集成，**无需或者少量修改应用代码。Session监听器需遵循bboss会话共享规范，需将原来容器session监听器迁移到bboss会话共享实现。如修改session中对象数据，必须调用session.setAttribute方法将对象数据更新到mongodb中，以便将更新后的数据共享给其他应用。

兼容性：**跨容器，**兼容业界主流的应用服务器（tomcat,weblogic,webspere,jetty），支持容器会话管理和bboss会话管理两种机制，可根据实际需要自由切换应用会话管理机制。

约　束：**无约束，**无需session sticking，客户端请求可以平均分派给各集群节点，支持lvs,haproxy，ngix 4,7层负载。

会话共享概述

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

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）,每个应用只能管理本应用的会话数据，监控中心可以管理所有的会话数据

高　阶：提供两种会话共享模式

模式一 集群间会话共享模式，实现同一个应用集群各节点之间的会话共享 ，通过这种模式可以避免因故障导致访问请求切换服务器时session丢失问题，同时也可以让用户请求无差别地平均分派到各个服务器上，达到真正的负载均衡。

模式二 跨域跨应用模式，实现同一域名或者同一根域（不同的子域名）下不同应用之间的会话共享 ，实现他们之间的单点登录功能（SSO）

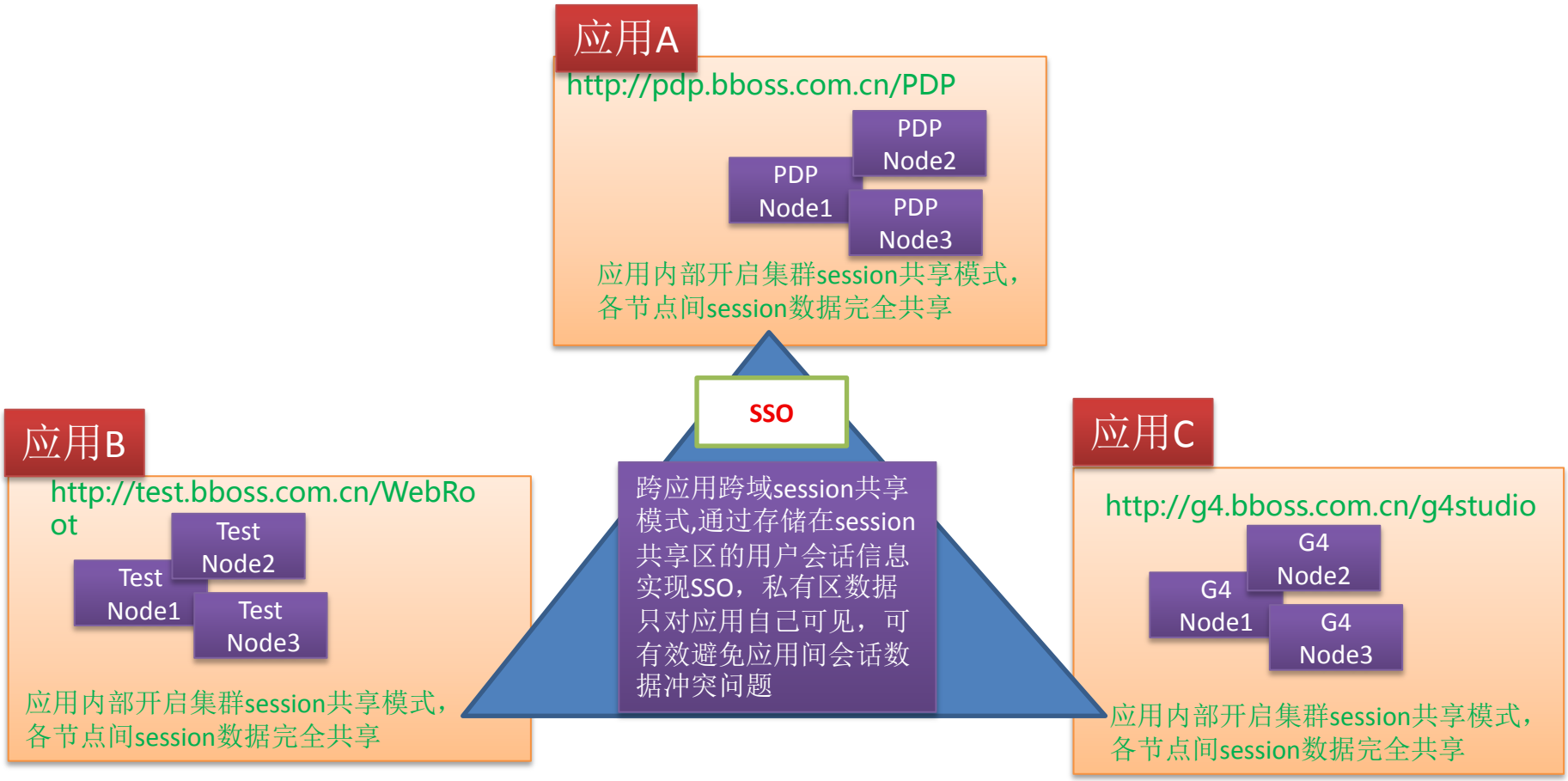
第一种模式相对简单；第二种模式在配置方面比模式一稍微复杂一些，通过模式二可以灵活定义哪些会话数据需要在应用之间进行共享，哪些数据作为应用私有会话数据不对其他应用共享（这个在实际情况下很有用），默认情况下共享应用间的所有会话数据。

实际的应用环境中，模式一和模式二经常组合一起使用，每个应用本身采用集群部署模式（开启集群间会话共享模式），同时利用**跨域跨应用模式实现不同应用间的单点登录功能（前提是这些应用必须使用同一个域名或者都拥有相同的根域名）。**

会话共享概述

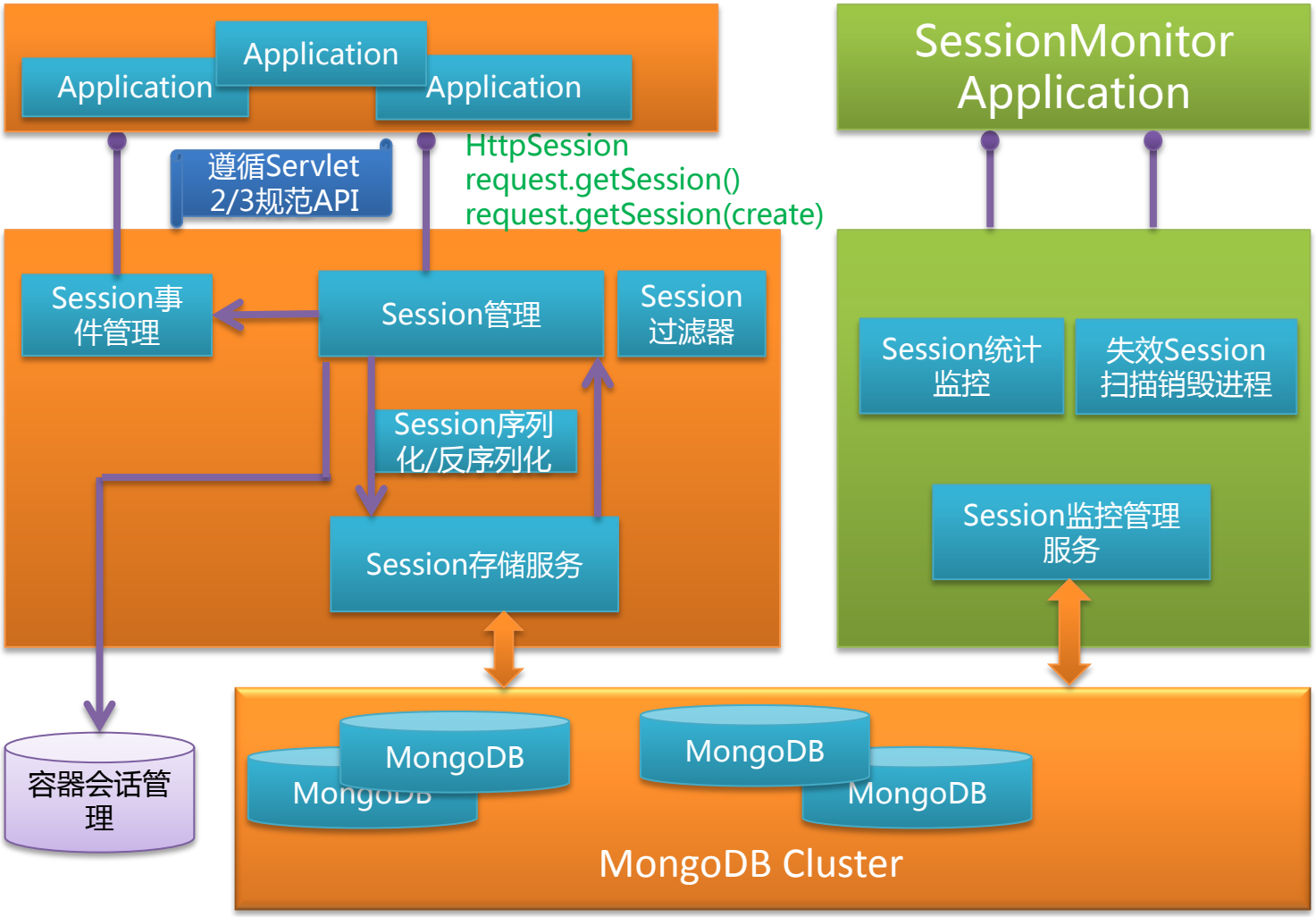
大纲 | 概述 | 架构 | 会话存储 | 应用集成 | 应用部署 | 场景演示

高　阶：两种会话共享模式一起使用示意图(前提:应用必须使用同一个域名或者都拥有相同的根域名，根域名不同的话可以使用bboss统一令牌系统实现跨根域系统之间的SSO)



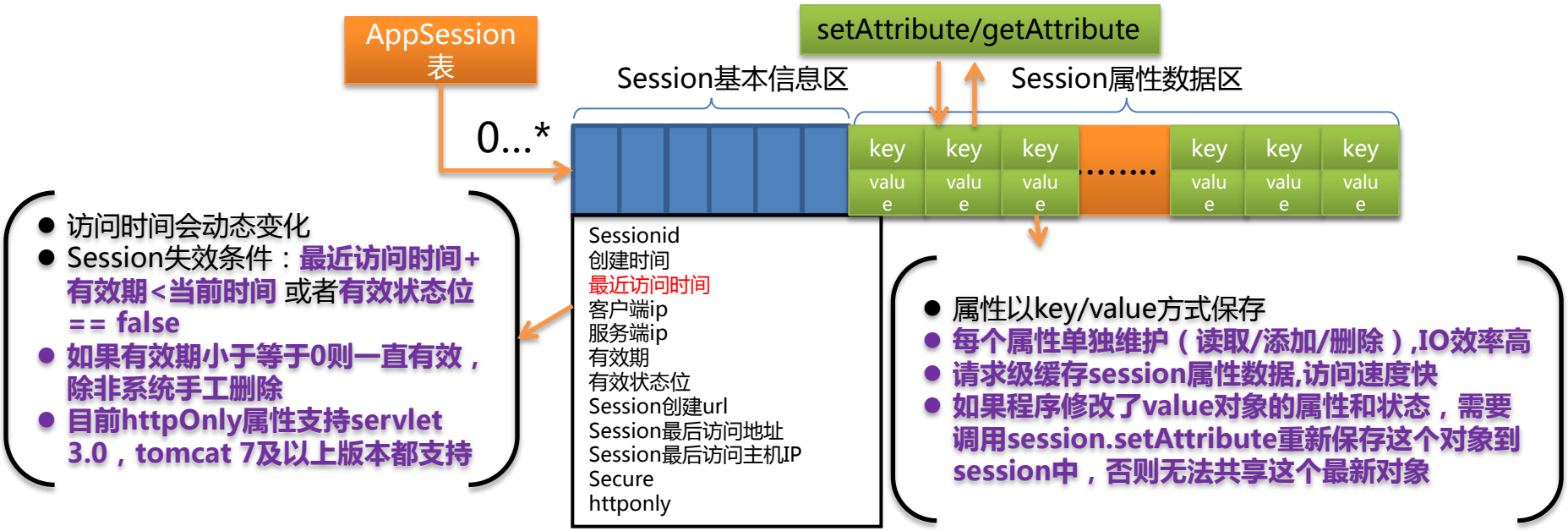
会话共享-逻辑架构

大纲 | 概述 | **架构** | 会话存储 | 应用集成 | 应用部署 | 场景演示



会话共享-session存储结构

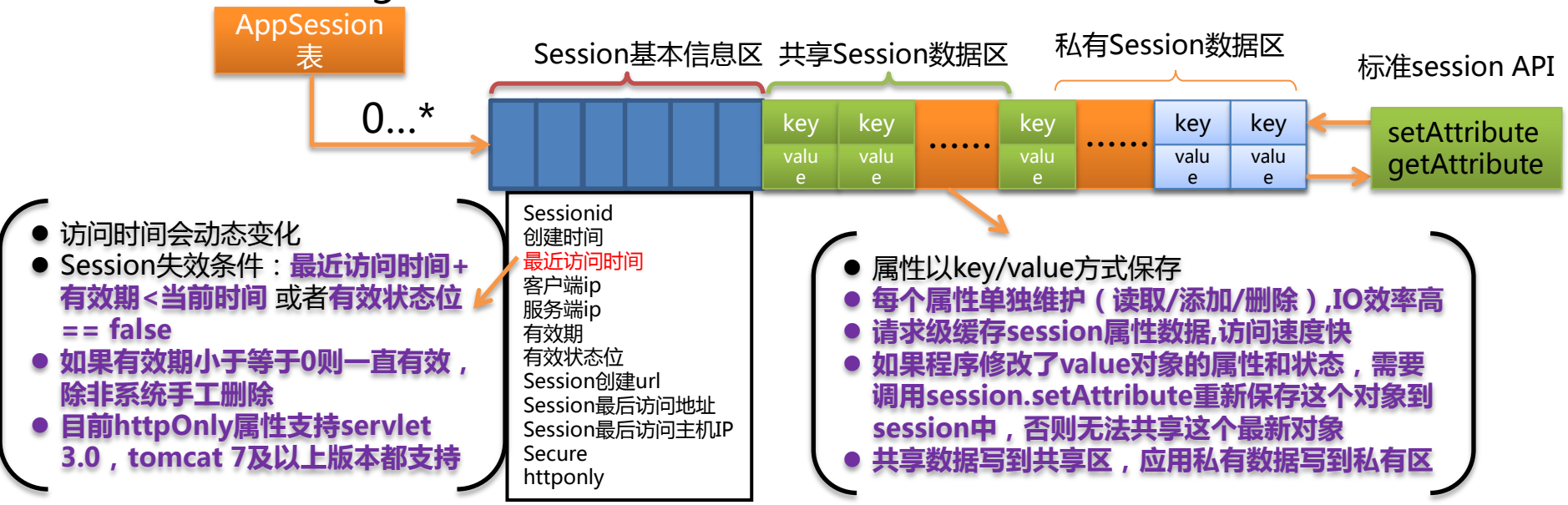
Session对象在mongodb中的存储结构示意图：单应用集群节点之间共享会话存储结构



- 一个session对应mongodb 中session表的一条记录，每个应用都有自己的session表
- session表记录数据分为两部分：基本信息区（固化）和属性数据区（**可动态扩张属性，属性个数不限，mongodb能够很好地支持这个特性，因为mongodb的表结构是动态的，每条记录包含的字段都可以不一样**）

会话共享-session存储结构

Session对象在mongodb中的存储结构示意图：跨域不同应用之间共享会话存储结构

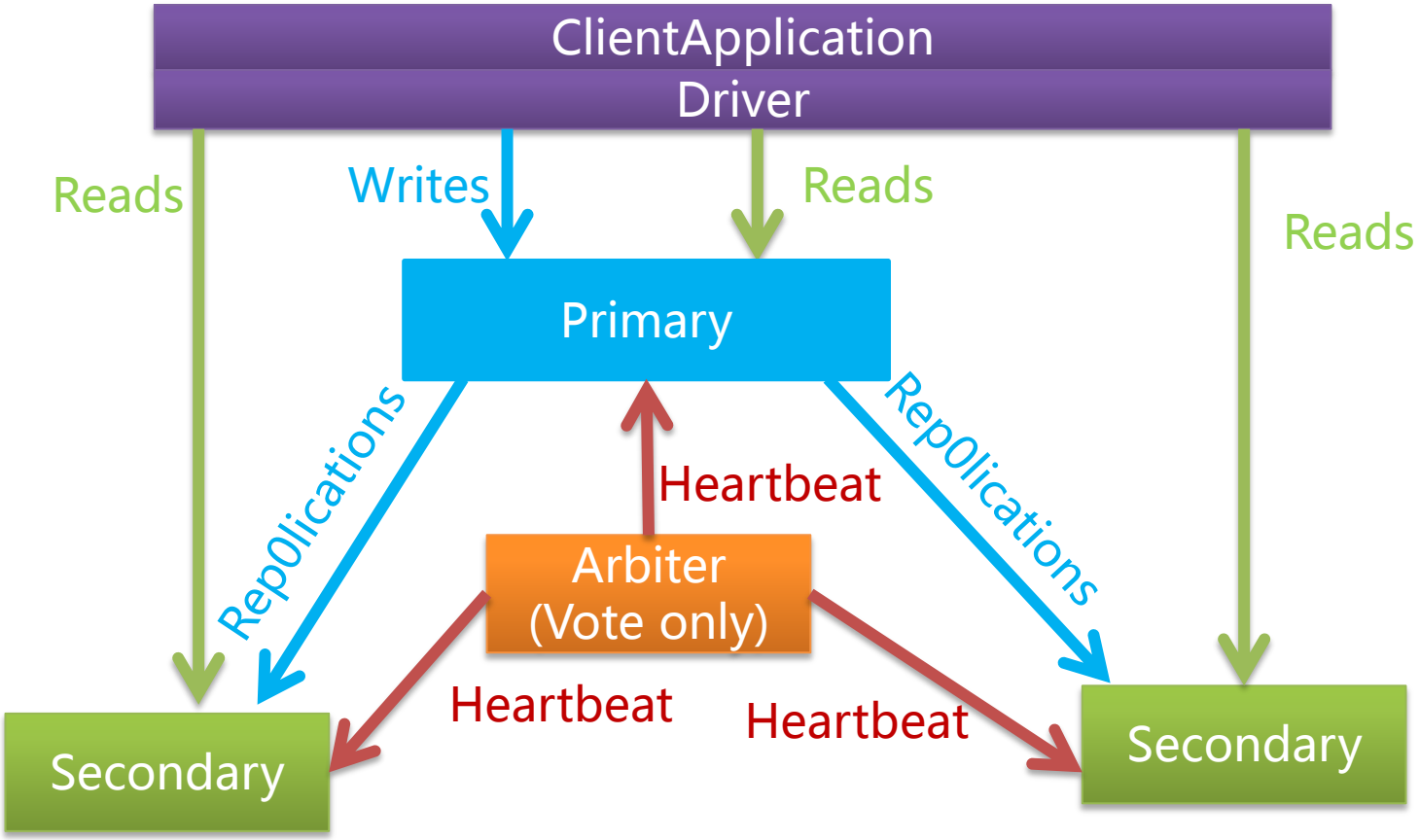


- 一个session对应mongodb 中session表的一条记录，跨域各应用公用一个session表，表名为**appcode+_sessions（appcode在sessionconf.xml文件中指定）**
- session记录分为三部分：基本信息区（固化）、共享属性数据区和私有属性区（**可动态扩张共享属性和私有属性，属性个数不限，mongodb能够很好地支持这个特性，因为mongodb的表结构是动态的，每条记录包含的字段都可以不一样**）

会话共享- Mongodb主从复制架构

大纲 | 概述 | 架构 | **会话存储** | 应用集成 | 应用部署 | 场景演示

Mongodb主从复制架构（读写分离）：



会话共享- Mongodb主从复制时延

Mongodb-读写分离场景下，主从复制存在时延问题，为了解决这个问题，在配置客户驱动程序的writeConcern参数时，需要采用REPLICA_ACKNOWLEDGED(n,2000)模式,其中n代表mongodb的数据节点数目（最多为所有mongodb数据节点数，可以小于数据节点数目），2000表示主节点数据写操作阻塞等待从节点数据复制完成的最长时间，单位为毫秒，这里是2000毫秒（2秒），如果超过2000毫秒，直接返回继续执行后续的程序操作。一般情况下复制过程等待不会超过2000毫秒，复制一完成立马返回写操作主程序继续执行后续程序。

会话共享- Mongodb客户端配置

Mongodb-单节点配置示例- resources/mongodb.xml

```
<properties>
        <!-- 增加mongodb数据源配置和client工厂类 -->
        <property name="default" factory-class="org.frameworkset.nosql.mongodb.MongoDB"
                init-method="init" destroy-method="close" factory-method="getMongoClient">

                <property name="serverAddresses" >
                        192.168.1.100:27016
                </property>
                <property name="option" ></property>

                <property name="writeConcern" value="JOURNAL_SAFE"/>

                <property name="readPreference" value=""/>


        </property>
</properties>
```

会话共享- Mongodb客户端配置

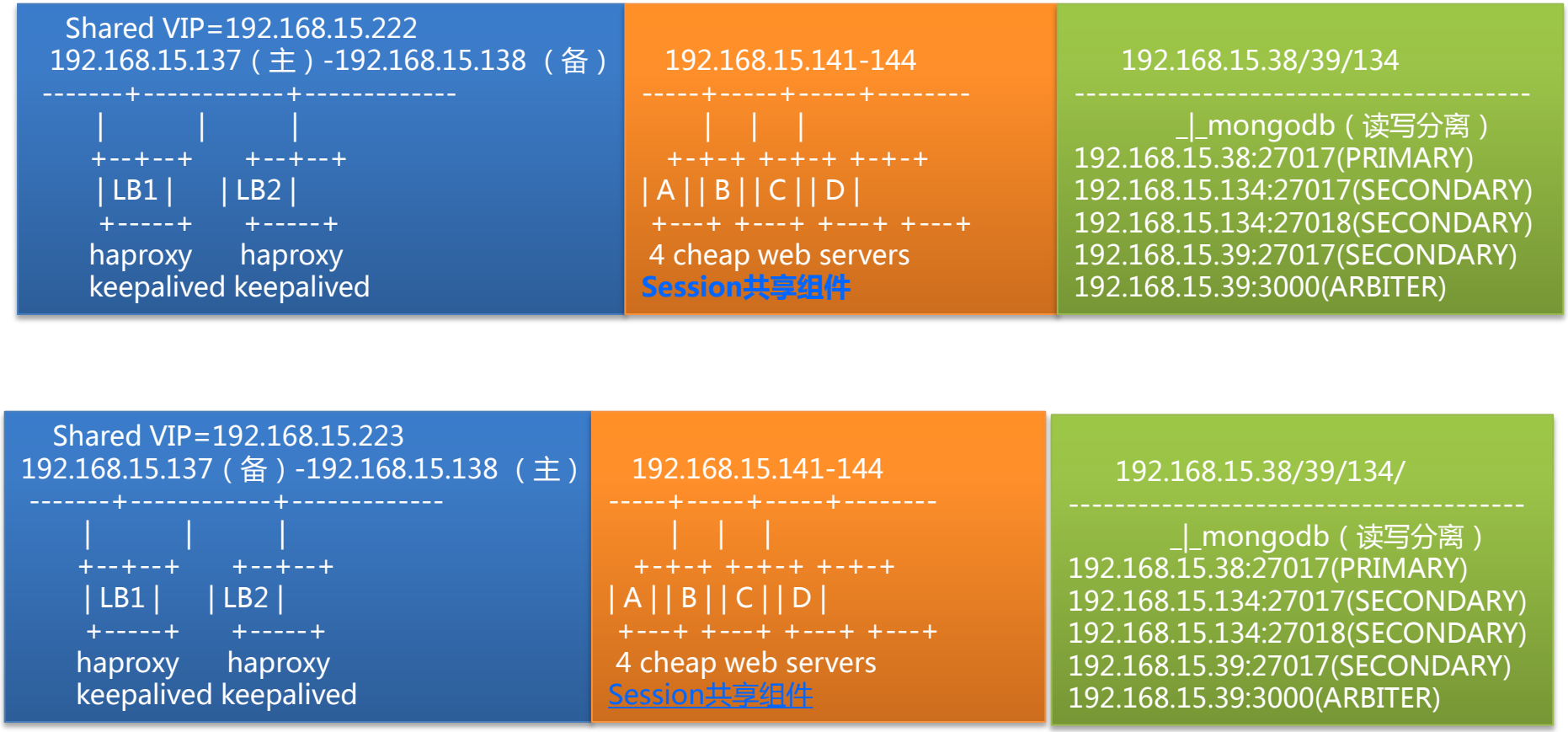
Mongodb-集群复制节点配置示例- resources/mongodb.xml

```
<properties>
        <!-- 增加mongodb数据源配置和client工厂类 -->
        <property name="default" factory-class="org.frameworkset.nosql.mongodb.MongoDB"
                init-method="init" destroy-method="close" factory-method="getMongoClient">

                <property name="serverAddresses" >
                        192.168.134:27017
                        192.168.15.134:27018
                        192.168.15.38:27017
                        192.168.15.39:27017
                </property>
                <property name="option" >
                        QUERYOPTION_SLAVEOK
                </property>
                <property name="writeConcern" value=" REPLICA_ACKNOWLEDGED(4,2000) "/>
                <property name="readPreference" value="NEAREST"/>
        </property>
</properties>
```

会话共享-应用集成-双主备模式

大纲 | 概述 | 架构 | 会话存储 | **应用集成** | 应用部署 | 场景演示



会话共享-应用集成

平台集成：
会话共享功能内置在bboss开发平台中，所以基于bboss开发平台新开发的项目只需开启会话共享机制并修改mongodb的配置即可；基于老版本的PDP开发的项目只需升级平台框架即可集成会话共享功能。

第三方J2ee项目集成：
第三方J2ee项目集成会话共享，只需要将会话共享的相关jar和配置文件整合到项目工程中，然后进行相应的功能验证测试，可能需要编写序列化插件。

集成准备工作：
目前已经整理好(1) **会话共享最小依赖jar和资源文件工程**，(2)**会话共享及监控最小依赖jar和资源文件**工程

会话共享-应用集成-注意事项

遗留系统集成会话共享组件哪些情况下需要进行程序调整：

a).遗留系统修改存储在session中数据对象的属性和状态，但是没有重新set到session中

修改：调用session.setAttribute方法将修改后的对象再次存储到mongodb中的session表中，以便将修改共享给其他域应用或者集群应用节点。

b). 遗留系统存储在session中的对象数据无法采用bboss进行序列化或者反序列化

修改：调整对象定义或者为对象编写序列化插件，采取哪种方式视实际情况来定

c).遗留系统定义并使用Session Event Listener，bboss提供了特有的session监听器

修改：需要将原来的session监听器迁移到bboss规范的session监听器

d.)遗留系统使用了重量级的session数据对象（对象中引用了很多大对象，这些大对象彼此没有关联，如果只需要获取或者修改其中的一部分数据，每次都要完整地将从mongodb中获取/写入这个完整的大对象，导致IO和性能低下）

修改：将这些彼此独立的数据从大对象中剥离出来，作为独立属性存储到session中，从而避免不必要的开销。

会话共享-应用集成-会话共享配置

大纲 | 概述 | 架构 | 会话存储 | **应用集成** | 应用部署 | 场景演示

会话共享组件集成到应用程序只需修改配置文件resources/sessionconf.xml：

```
<properties>
<property name="sessionManager" class="org.frameworkset.security.session.impl.SessionManager"
init-method="init" destroy-method="destroy">
<property name="sessionTimeout" value="3600000"/>
<property name="sessionstore" refid="attr:sessionstore"/>
<!-- <property name="sessionstore" value="session"/>-->
<property name="cookiename" value="b_sessionid"/>
<property name="httpOnly" value="true"/>
<property name="secure" value="false"/>
<property name="sessionlisteners"
value="org.frameworkset.security.session.impl.NullSessionListener"/>
<!--<property name="appcode" value="10_25_192_142_pdp"/>-->
<property name="startLifeScan" value="false"/>
</property>
```

集群应用会话共享模式

```
<!-- f:monitorScope  all|self  监控中心为all，本地设置为self -->
<property name="sessionStaticManager"  f:monitorScope="all"
class="org.frameworkset.security.session.statics.
MongoSessionStaticManagerImpl"/>
<property name="sessionstore" class="org.frameworkset.security.session.impl.MongDBSessionStore"/>
</properties>
```

sessionManager配置说明：
org.frameworkset.security.session.impl.SessionManager是bboss提供的session会话管理组件。
sessionTimeout-属性指定会话超时时间，单位：毫秒，-1或0标识session永不超时。
sessionstore-指定session的存储机制，session表示会话使用容器默认的会话管理机制，配置为MongDBSessionStore时则启用bboss的会话管理机制
cookiename-指定会话cookie名称
httpOnly-指定会话cookie是否使用httponly模式，true和false两个值，默认为true
secure –结合https阻止传输过程中sessionid被窃取
Appcode-应用编码，如果没有指定appcode值默认为应用上下文
appcode的作用：当所有的应用上下文为“/”时，用来区分后台统计的会话信息
如果应用上下文为“/”时,appcode为ROOT
startLifeScan:控制是否启用失效session扫描进程，监控中心开启时，本地务必关闭

会话共享-应用集成-会话共享配置-跨域跨应用配置

会话共享组件集成到跨域应用时，只需在之前的基础上为组件 *org.frameworkset.security.session.impl.SessionManager* “添加appcode 和 crossDomain 两个属性即可：

```
<property name="appcode" value="pdp"/>
<property name="crossDomain" class="org.frameworkset.security.session.domain.CrossDomain"
   f:domain="bboss.com.cn"
   f:shareSessionAttrs="CREDENTIAL_INDEXS,PRINCIPAL_INDEXS"    init-method="init">
   <property name="domainApps">
         <list componentType="bean">
           <property class="org.frameworkset.security.session.domain.App"
             f:path="/PDP"  f:attributeNamespace="pdp_bboss_com_cn"
             f:currentApp="true"
             init-method="init" />
           <property class="org.frameworkset.security.session.domain.App"
             f:currentApp="false"
             f:path="/g4studio"   f:attributeNamespace="g4_bboss_com_cn"   init-method="init" />
           <property class="org.frameworkset.security.session.domain.App"
             f:currentApp="false"
             f:path="/WebRoot"    f:attributeNamespace="testpdp_bboss_com_cn"   init-method="init" />
         </list>
   </property>
</property>
```

跨域不同应用间会话共享模式

f:currentApp 如果应用时当前应用，需要把这个属性设置为true，反之为false
跨域共享应用时，同样可以通过私有ip或者域名单独访问应用。

会话共享-应用集成-会话共享配置-跨域跨应用配置

会话共享组件集成到跨域应用相关属性说明：

跨域不同应用间会话共享模式

domain：指定跨域共享的根域，基于该域名的子域名都可以共享session,也可以是ip地址
shareSessionAttrs:配置需要在应用间共享的会话数据属性名称，以逗号分隔；如果没有配置shareSessionAttrs属性，则所有的属性都是共享数据
domainApps：指定需要session共享的应用列表，每个应用必须指定path属性（对应应用上下文路径），如果应用指定了attributeNamespace属性，则用attributeNamespace对应的值来限定应用私有的会话数据名称，每个app的attributeNamespace属性只有在CrossDomain上指定了shareSessionAttrs属性才有意义
path:共享session的应用上下文名称
attributeNamespace:应用私有session属性名称命名空间，用来限定应用私有session数据的存储空间 ，如果指定了shareSessionAttrs则必须指定每个应用的attributeNamespace

样例配置说明：假设有以下三个应用，访问地址分别为
http://pdp.bboss.com.cn:8080/PDP
http://g4.bboss.com.cn:169/g4studio
http://test.bboss.com.cn:8080/WebRoot

通过样例配置，三个应用的用户会话信息存储在session的CREDENTIAL_INDEXS,PRINCIPAL_INDEXS两个共享属性中，可以实现三个应用之间的会话共享和单点登录功能。如果需要共享更多的会话数据，可以将对应的属性追加到shareSessionAttrs中（以逗号分割），没有出现在shareSessionAttrs中的属性都是私有会话数据（对其他应用不可见）。如果没有指定shareSessionAttrs，则会话数据全部在三个应用间共享。

会话共享-应用集成-session监听器

session监听器： session监听器用来监听并处理session创建和销毁事件，会话属性创建和删除事件, 通过SessionManager组件的sessionlisteners属性来指定多个session监听器，多个listener按照顺序以逗号分隔，listener只有在PDP**会话共享**管理会话的时候才有用，每一个listener必须实现接口：

```
package org.frameworkset.security.session;
public interface SessionListener {
        public void createSession(SessionEvent event);
        public void destroySession(SessionEvent event);
        public void addAttribute(SessionEvent event);
        public void removeAttribute(SessionEvent event);
}
```

session监听器是可选配置项，可以不配置。

会话共享-应用集成-过滤器配置

Session过滤器配置-web.xml：

```
<filter>
        <filter-name>sessionFilter</filter-name>
        <filter-class>org.frameworkset.security.session.impl.SessionFilter</filter-class>
 </filter>
<filter-mapping>
    <filter-name>CharsetEncoding</filter-name>
    <url-pattern>*.jsp</url-pattern>
 </filter-mapping>
 <filter-mapping>
    <filter-name>CharsetEncoding</filter-name>
    <url-pattern>*.do</url-pattern>
 </filter-mapping>
 <filter-mapping>
    <filter-name>CharsetEncoding</filter-name>
    <url-pattern>*.frame</url-pattern>
 </filter-mapping>
 <filter-mapping>
    <filter-name>CharsetEncoding</filter-name>
    <url-pattern>*.page</url-pattern>
 </filter-mapping>
 <filter-mapping>
    <filter-name>CharsetEncoding</filter-name>
    <url-pattern>*.freepage</url-pattern>
 </filter-mapping>
 <filter-mapping>
    <filter-name>CharsetEncoding</filter-name>
    <url-pattern>/cxfservices/*</url-pattern>
 </filter-mapping>
 <filter-mapping>
    <filter-name>CharsetEncoding</filter-name>
    <url-pattern>/jasperreport/*</url-pattern>
 </filter-mapping>
```

样例配置

会话共享-应用集成-序列化插件配置

大纲 | 概述 | 架构 | 会话存储 | **应用集成** | 应用部署 | 场景演示

序列化插件配置- /resources/org/frameworkset/soa/serialconf.xml ：

```
<property name= "java.util.Locale" magic= "1" serial= "org.frameworkset.soa.LocaleSerial"/>
 <property name= "net.sf.jasperreports.engine.JasperPrint" magic= "2" serial= "org.frameworkset.soa.JDKSerial"/>
 <property name= "com.frameworkset.platform.security.authentication.CheckCallBack$Attribute" magic= "3"/>
 <property name= "com.frameworkset.platform.security.authentication.CheckCallBack" magic= "4"/>
 <property name= "com.frameworkset.platform.security.authentication.Credential" magic= "5"/>
 <property name= "com.frameworkset.platform.security.authorization.AuthPrincipal" magic= "6"/>
```

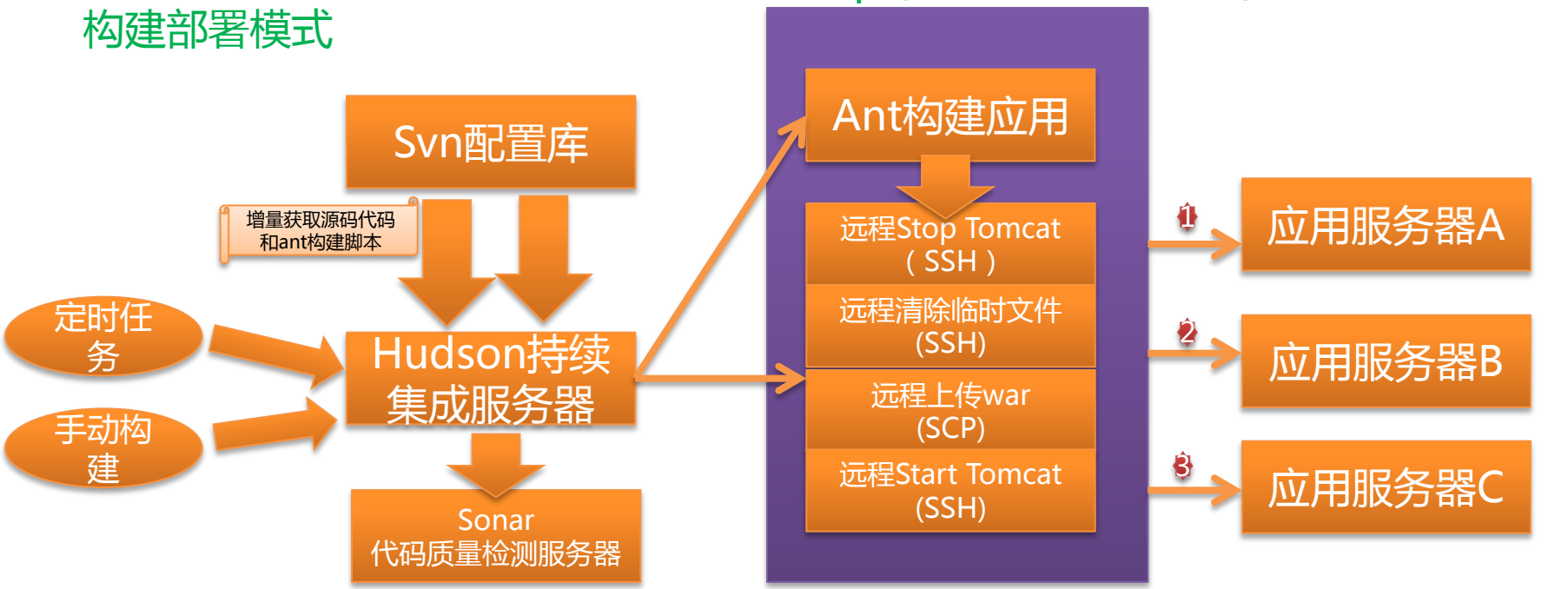
序列化插件机制是序列化功能的一个辅助功能，通过序列化插件机制，可以达到以下目标：
1.自定义对象的序列化和反序列化行为
2.对序列化对象进行预处理
3.为类路径定义magic数字别名，降低序列化报文大小

会话共享-应用构建部署

大纲 | 概述 | 架构 | 会话存储 | 应用集成 | **应用部署** | 场景演示

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模式：

基于Hudson+svn+ant+tomcat+ssh+scp（多tomcat服务器）的持续集成自动构建部署模式



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。

会话共享-场景演示

大纲 | 概述 | 架构 | 会话存储 | 应用集成 | 应用部署 | **场景演示**

场景1：Session统计监控管理

明细查看-SanyPDP

基本信息

SessionID: b605bd50-3d5e-49fd-9792-3b052b33d03d　　创建时间: 2014-06-10 16:47:38　　最后访问时间: 2014-06-10 16:58:16　　客户端: 10.0.15.137

服务端: 127.0.0.1-sany.mdss7　　有效期: 1小时　　失效时间: 2014-06-10 17:58:16　　状态: 有效

属性列表

MACHINENAME_CACHE_KEY

```
view plain  copy to clipboard  print  ?
01. <ps><p n="_dflt_" s:t="String"><![CDATA[]]></p></ps>
```

CREDENTIAL_INDEXS

```
view plain  copy to clipboard  print  ?
01. <ps><p n="_dflt_" s:t="HashMap"><m cmt="bean">
<p n="console" cs="com.frameworkset.platform.security.authentication.Credential">
<p n="checkCallBack" cs="com.frameworkset.platform.security.authentication.CheckCallBack">
<p n="callBacks" s:t="HashMap"><m cmt="bean">
<p n="userPinyin" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userPinyin]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="userOicq" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userOicq]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="userPostalcode" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userPostalcode]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="userSn" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userSn]]></p><p n="value" s:t="String"><![CDATA[1]]></p></p>
<p n="remark5" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[remark5]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="remark4" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[remark4]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="userEmail" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userEmail]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="remark1" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[remark1]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="remark3" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[remark3]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="remark2" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[remark2]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="userType" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userType]]></p><p n="value" s:t="String"><![CDATA[0]]></p></p>
<p n="userAddress" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userAddress]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="userFax" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userFax]]></p><p n="value" s:t="String"><![CDATA[ ]]></p></p>
<p n="userName" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userName]]></p><p n="value" s:t="String"><![CDATA[系統管理员]]></p></p>
<p n="userWorknumber" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
<p n="name" s:t="String"><![CDATA[userWorknumber]]></p><p n="value" s:t="String"><![CDATA[1]]></p></p>
<p n="CHARGEORGID" cs="com.frameworkset.platform.security.authentication.CheckCallBack$Attribute">
```

会话共享-场景演示

场景2：正常切换集群节点演示，用户访问请求均衡负载到各集群节点，admin用户登录系统，先后访问两次系统主页，对比haproxy监控会话统计结果，可以看出在线用户请求被平均分发到各集群节点，没有发生因切换节点而导致用户退出系统，Session共享机制工作正常。

第一次访问，三台服务器接收请求数基本持平

webserver

		Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
	s1	0	0	-	0	2		0	4	-	10	10	1h31m	88 833	303 600		0		0	0	0	0	23h11m UP	L4OK in 0ms	1	Y	-	4	0	0s	-
	s2	0	0	-	0	2		0	3	-	10	10	1h31m	89 500	725 411		0		0	0	0	0	23h11m UP	L4OK in 0ms	1	Y	-	4	0	0s	-
	s3	0	0	-	0	2		0	3	-	9	9	1h31m	74 295	675 484		0		0	0	0	0	19h7m UP	L4OK in 0ms	1	Y	-	4	1	10s	-
	Backend	0	0		0	5		0	10	300	29	29	1h31m	252 628	1 704 495	0	0		0	0	0	0	23h11m UP		3	3	0		0	0s	

第二次访问，三台服务器的请求数都发生了增长，而且基本持平，请求被平均分发。

webserver

		Queue			Ses																		Server								
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
	s1	0	0	-	0	2		0	4	-	11	11	51s	94 133	353 489		0		0	0	0	0	23h14m UP	L4OK in 0ms	1	Y	-	4	0	0s	-
	s2	0	0	-	0	2		1	3	-	11	11	17s	89 500	725 411		0		0	0	0	0	23h14m UP	L4OK in 0ms	1	Y	-	4	0	0s	-
	s3	0	0	-	0	2		1	3	-	10	10	18s	74 295	675 484		0		0	0	0	0	19h10m UP	L4OK in 0ms	1	Y	-	4	1	10s	-
	Backend	0	0		0	5		2	10	300	32	32	17s	257 928	1 754 384	0	0		0	0	0	0	23h14m UP		3	3	0		0	0s	

会话共享-场景演示

大纲 | 概述 | 架构 | 会话存储 | 应用集成 | 应用部署 | **场景演示**

场景3：故障切换集群节点演示，用户访问请求均衡负载到各有效集群节点，admin用户登录系统，先后访问两次系统主页，第一次三台服务都正常，第二次停用其中一台，对比haproxy监控会话统计结果，可以看出在线用户请求被平均分发到有效集群节点，没有发生因故障切换节点而导致用户退出系统，宕机情况下，Session共享机制工作正常。

webserver

第一次访问，各集群节点工作正常

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
s1	0	0	-	0	2		1	4	-	16	16	5s	111 043	441 364		0		0	0	0	0	23h36m UP	L4OK in 0ms	1	Y	-	4	0	0s	-
s2	0	0	-	0	2		0	3	-	15	15	1m44s	105 184	738 696		0		0	0	0	0	23h36m UP	L4OK in 0ms	1	Y	-	4	0	0s	-
s3	0	0	-	0	2		0	3	-	14	14	1m44s	91 518	733 810		0		0	0	0	0	19h32m UP	L4OK in 0ms	1	Y	-	4	1	10s	-
Backend	0	0		0	5		1	10	300	45	45	5s	307 745	1 913 870	0	0		0	0	0	0	23h36m UP		3	3	0		0	0s	

Choose the action to perform on the checked servers : Apply

webserver

第二次访问：停用s1节点，请求全部被转发到s2,s3

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
s1	0	0	-	0	2		0	4	-	24	24	12m	157 051	566 326		0		0	0	0	0	7m55s DOWN	L4CON in 0ms	1	Y	-	16	6	15m15s	-
s2	0	0	-	0	3		0	3	-	31	31	7m10s	216 416	1 359 500		0		0	0	0	0	1d39m UP	L4OK in 0ms	1	Y	-	4	0	0s	-
s3	0	0	-	0	3		0	3	-	31	31	7m10s	192 544	1 098 681		0		0	0	0	0	20h35m UP	L4OK in 0ms	1	Y	-	4	1	10s	-
Backend	0	0		0	6		0	10	300	86	86	7m10s	566 011	3 024 507	0	0		0	0	0	0	1d39m UP		2	2	0		0	0s	

参考资料

更多bboss会话共享参考资料请访问博客文章：

http://yin-bp.iteye.com/category/327553

Bboss介绍参考资料请访问博客文章：

http://yin-bp.iteye.com/blog/1080824

Thanks for your attention!

谢 谢 ！

2015 中国·长沙