# 谈谈架构和开源

架构经验和开源技术分享

# 关于我

**徐少敏**

**深圳市九指天下科技有限公司 CTO**

**崇尚技术，对技术很热衷**

**QQ：746167**

**QQ群：57224414**

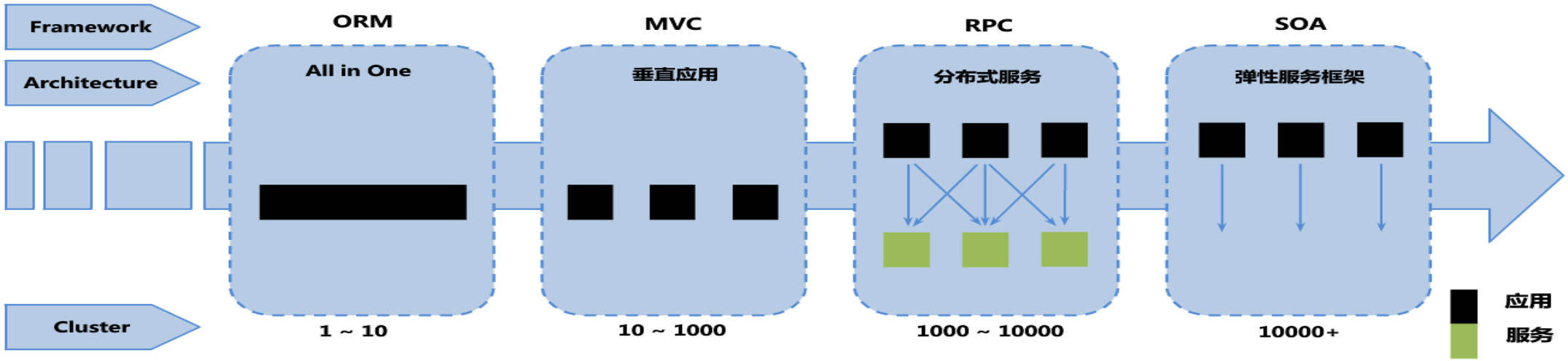
**Github ： https://www.github.com/xushaomin**

# 目录

- SOA和微服务
- APP服务端架构设计
- 配置管理
- 自动化运维
- 自动化监控
- 分布式日志管理
- 分布式跟踪系统

# SOA和微服务

## 网站应用架构的演进




**单一应用架构**

当网站流量很小时，只需一个应用，将所有功能都部署在一起，以减少部署节点和成本。此时，用于简化增删改查工作量的 数据访问框架(ORM) 是关键。

**垂直应用架构**

当访问量逐渐增大，单一应用增加机器带来的加速度越来越小，将应用拆成互不相干的几个应用，以提升效率。此时，用于加速前端页面开发的 Web框架(MVC) 是关键。

**分布式服务架构**

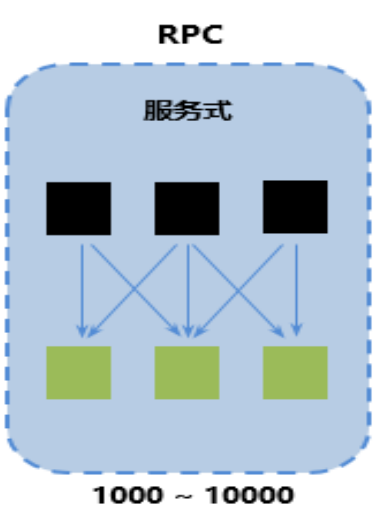
当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来作为独立的可以复用的服务，使前端应用能更快速的响应多变的市场需求。此时，RPC技术是关键。

**流动计算架构**

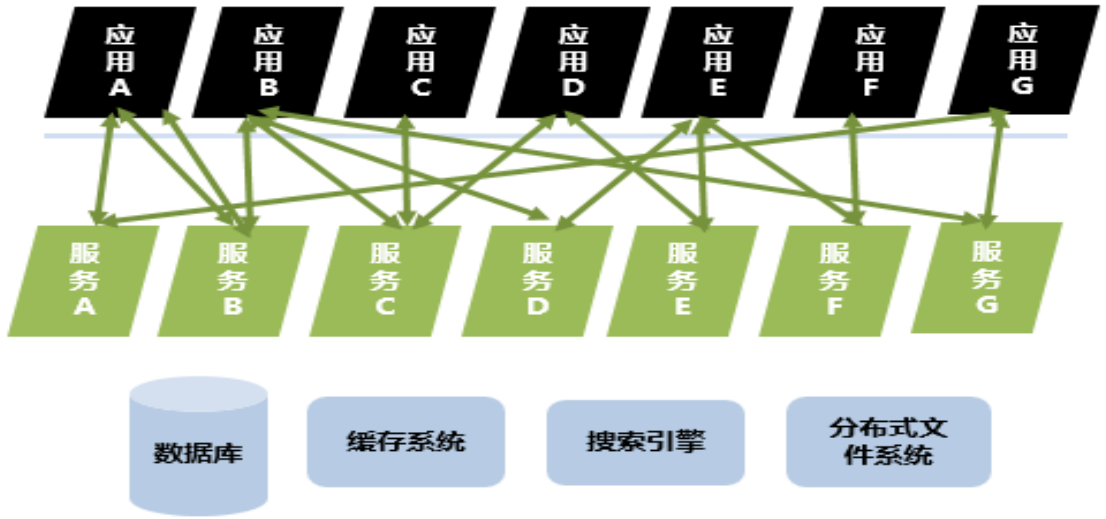
当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率。此时，用于提高设施利用率的资源调度和治理中心(SOA) 是关键。

# SOA和微服务

## 服务式应用架构







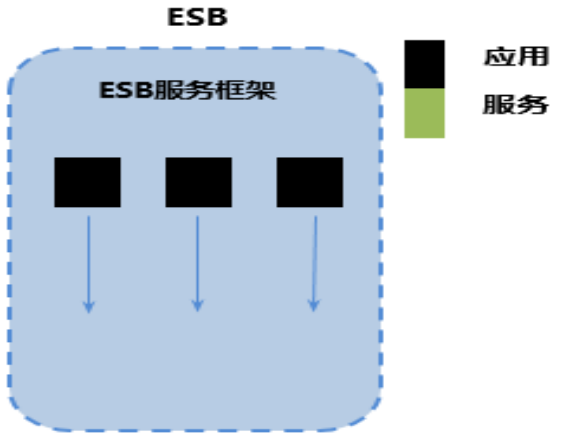

**服务式应用架构**

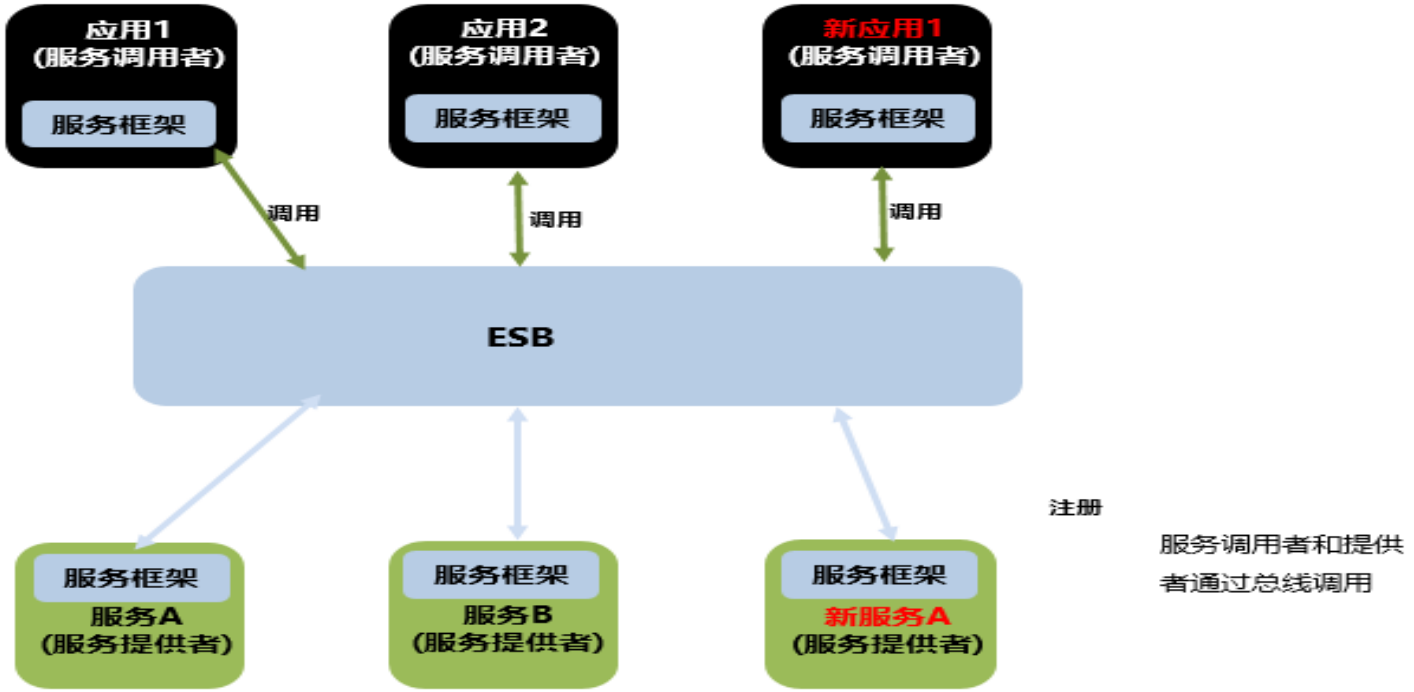
当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来作为独立的可以复用的服务，使前端应用能更快速的响应多变的市场需求。此时，RPC技术是关键。

- 增加服务层，把冗余的代码和可以复用的业务应用进行拆分提取，封装成服务
- 系统架构更加清晰，代码质量提高，利于升级和维护，稳定性高
- 应用层可以更专注在与前端用户如何交互，业务处理放在服务层来进行
- 服务和应用的管理不是自动化，服务层能够实现HA的功能
- 适用中大型网站系统的场景中

# SOA和微服务

## ESB服务框架




支持传输层：

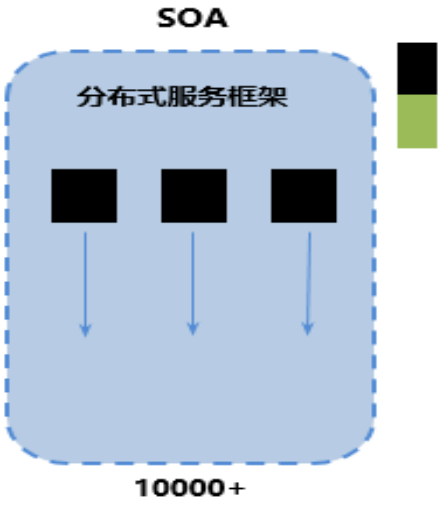
- HTTP/S
- JMS
- Email (POP3/IMAP/SMTP)
- AMQP
- File/SFTP/FTP/FTPS/Samba
- Timer (Scheduled Job)
- TCP/S
- MLLP/S

支持协议

- REST
- SOAP
- Hessian
- Protocol Buffers

# SOA和微服务

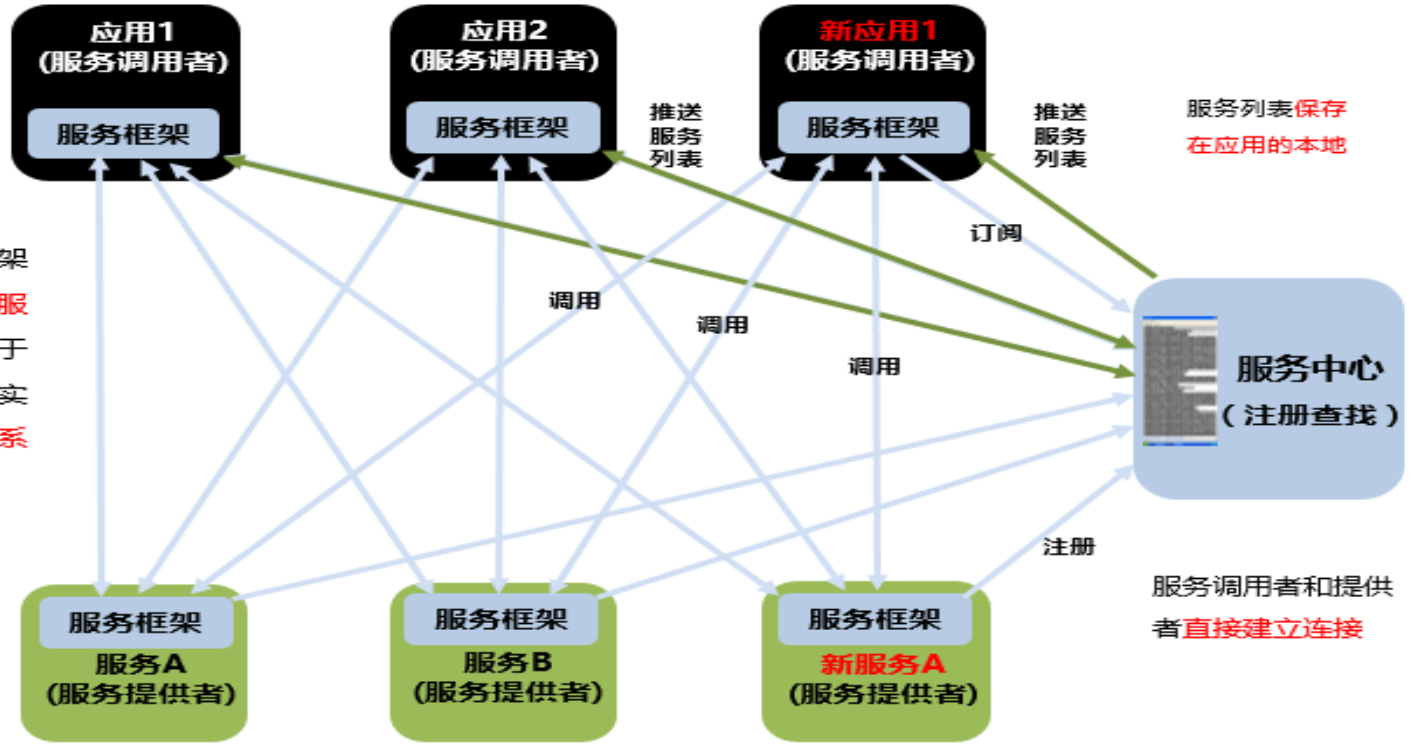
## 分布式服务框架




应用

服务

基于服务式应用架构基础上，引入服务注册中心，用于保存服务列表；实现自动化服务体系框架




**流动计算架构**

当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率。此时，用于提高设施利用率的资源调度和治理中心(SOA) 是关键。

- 分布式架构，应用层和服务层可根据需求进行动态水平扩展，应用与服务实现负载均衡，通过随机、轮询、权重等策略
- 开放式、标准化的框架，满足接口调用的服务都可以接入服务框架（RPC）
- 监控服务调用情况，可进一步对服务层再分层，根据业务需求和对服务运行情况对服务进行编排和梳理，以及服务治理
- 适用大型及超大型网站应用架构

# SOA和微服务

传统的SOA 使用ESB 或者Webservice 这种重量级的解决方案，微服务推荐使用一些更轻的解决方案，要通用性，可以用Restful 架构，走HTTP 通道，支持Json 序列化协议

阿里的框架dubbo 以及淘宝内部的HSF，Navi-rpc 都可以看做微服务化框架的雏形，加上服务治理中心的管理、基础交付设施的保障就可以构成完整的一套微服务框架。

- 推荐SOA和微服务开源项目：

1. Dubbo：http://dubbo.io/

2. Netflix-Hystrix: https://github.com/Netflix/Hystrix

3. spring-cloud： https://github.com/spring-cloud

# APP服务端的架构设计

| | | |
|---|---|---|
| 代理层 | LVS | HA-Proxy |
| | NGINX | Apache |
| 接入层 | WEB(REST) | API GW(REST) |
| 服务层 | 服务提供 | 服务提供 |
| 缓存层 | Redis | Codis或Redis集群 |
| 存储层 | MySQL/ Mycat | Hadoop/Hbase |

# APP服务端的架构设计

## 从API开始

一个App，最核心的东西，其实就是数据，而数据的主要来源，就是API。我之前负责的项目，因为API的坑已经受过了不少苦，因此，之后对App项目的架构设计我都会先从API开始。

### 制定安全机制

设计API第一个需要考虑的是API的安全机制。我负责的上一个项目，因为API的安全问题，就被人攻击了两次。之后经过分析，主要存在两个漏洞：一是因为缺少对调用者进行安全验证的方式，二是因为数据传输不够安全。那么，制定API的安全机制，主要就是为了解决这两个问题：

1. 保证API的调用者是经过自己授权的App；
2. 保证数据传输的安全。

# APP服务端的架构设计

第一个问题的解决方案，我主要采用设计签名的方式。对每个客户端，Android、iOS、WeChat，分别分配一个AppKey和AppSecret。需要调用API时，将AppKey加入请求参数列表，并将AppSecret和所有参数一起，根据某种签名算法生成一个签名字符串，然后调用API时把该签名字符串也一起带上。服务端收到请求之后，根据请求中的AppKey查询相应的AppSecret，按照同样的签名算法，也生成一个签名字符串，当服务端生成的签名和请求带过来的签名一致的时候，那就表示这个请求的调用者是经过自己授权的，证明这个请求是安全的。而且，每个端都有一个Key，也方便不同端的标识和统计。为了防止AppSecret被别人获取，这个AppSecret一般写死在代码里面。另外，签名算法也需要有一定的复杂度，不能轻易被别人破解，最好是采用自己规定的一套签名算法，而不是采用外部公开的签名算法。另外，在参数列表中再加入一个时间戳，还可以防止部分重放攻击。

# APP服务端的架构设计

| | | | |
|---|---|---|---|
| method | String | 是 | API接口名称。 |
| app_key | String | 是 | TOP分配给应用的AppKey。 |
| session | String | 否 | 用户登录授权成功后，TOP颁发给应用的授权信息，详细介绍请点击这里。当此API的标签上注明：“需要授权”，则此参数必传；“不需要授权”，则此参数不需要传；“可选授权”，则此参数为可选。 |
| timestamp | String | 是 | 时间戳，格式为yyyy-MM-dd HH:mm:ss，时区为GMT+8，例如：2015-01-01 12:00:00。淘宝API服务端允许客户端请求最大时间误差为10分钟。 |
| format | String | 否 | 响应格式。默认为xml格式，可选值：xml，json。 |
| v | String | 是 | API协议版本，可选值：2.0。 |
| partner_id | String | 否 | 合作伙伴身份标识。 |
| target_app_key | String | 否 | 被调用的目标AppKey，仅当被调用的API为第三方ISV提供时有效。 |
| simplify | Boolean | 否 | 是否采用精简JSON返回格式，仅当format=json时有效，默认值为：false。 |
| sign_method | String | 是 | 签名的摘要算法，可选值为：hmac，md5。 |
| sign | String | 是 | API输入参数签名结果，签名算法介绍请点击这里。 |

# APP服务端的架构设计

第二个问题的解决方案，主要就是采用HTTPS了。HTTPS因为添加了SSL安全协议，自动对请求数据进行了压缩加密，在一定程序可以防止监听、防止劫持、防止重发，主要就是防止中间人攻击。苹果从iOS9开始，默认就采用HTTPS了。而关于在Android中如何使用HTTPS，Google官方也给出了很多安全建议。不过，大部分App并没有按照安全建议去实现，主要就是没有对SSL证书进行安全性检查，这就成为了一个很大的漏洞，中间人利用此漏洞用假证书就可以通过检查，从而可以劫持到所有数据了。因此，为了安全考虑，建议对SSL证书进行强校验，包括签名CA是否合法、域名是否匹配、是不是自签名证书、证书是否过期等。

# APP服务端的架构设计

**接口版本控制**

我们已经不止一次因为接口发生变动而导致旧版本的App出错的问题，而且变动不一定是修改了接口本身，有可能是底层增加了一种新的数据结构，接口把新数据也返回给客户端了，但客户端旧版本是解析不了的，从而就导致出错了。

为了解决接口的兼容性问题，需要做好接口版本控制。实现上，一般有两种做法：

1. 每个接口有各自的版本，一般为接口添加个version的参数；
2. 整个接口系统有统一的版本，一般在URL中添加版本号，比如http://api.domain.com/v2。

平时小版本的更新，就采用第一种方式，我们的做法是根据不同版本号做不同分支处理。大版本的更新，则用第二种方式，这时候，基本就是一套全新的接口系统了，跟旧版本是相对独立的。

当版本越来越多时，维护就会成为一个大问题，我们没那么多精力去维护所有版本，因此，太旧的版本一般就不会再维护了。这时候，如果有用户还在使用即将废弃的旧版本，需要提醒用户升级到新版本。

# APP服务端的架构设计

推荐开源项目ROP：https://github.com/itstamen/rop
作者：陈熊华

ROP是采用Spring MVC 3.0框架，实现的模拟TOP的轻量级Web Service框架，完全开源，使用ROP可以非常快速地构建您自己的开放平台。

ROP采用如下实现技术：

- Spring MVC 3.0：整个ROP构建于Spring MVC 3.0基础上；
- JSR 303：采用JSR 303校验注解对请求参数进行合法性校验，ROP会自动将校验结果转换成错误报文输出；
- JAXB：虽然Spring MVC 3.0可以支持将POJO流化成XML及JSON输出，不过控制上不太便利，因此ROP采用JAXB对响应的对象进行注解，并通过JAXB+Jackson将响应对象流化等价XML或JSON；
- 国际化支持：错误信息支持国际化。

# APP服务端的架构设计

**Rop 功能架构**

CXF 和 Jersey 是纯技术纯的 Web Service 框架，而在 Rop 中，Web Service 只是核心，它提供了开发服务平台的诸多领域问题的解决方案：如应用认证、会话管理、安全控制、错误模型、版本管理、超时限制等。

下面通过图 1 了解一下 Rop 框架的整体结构：

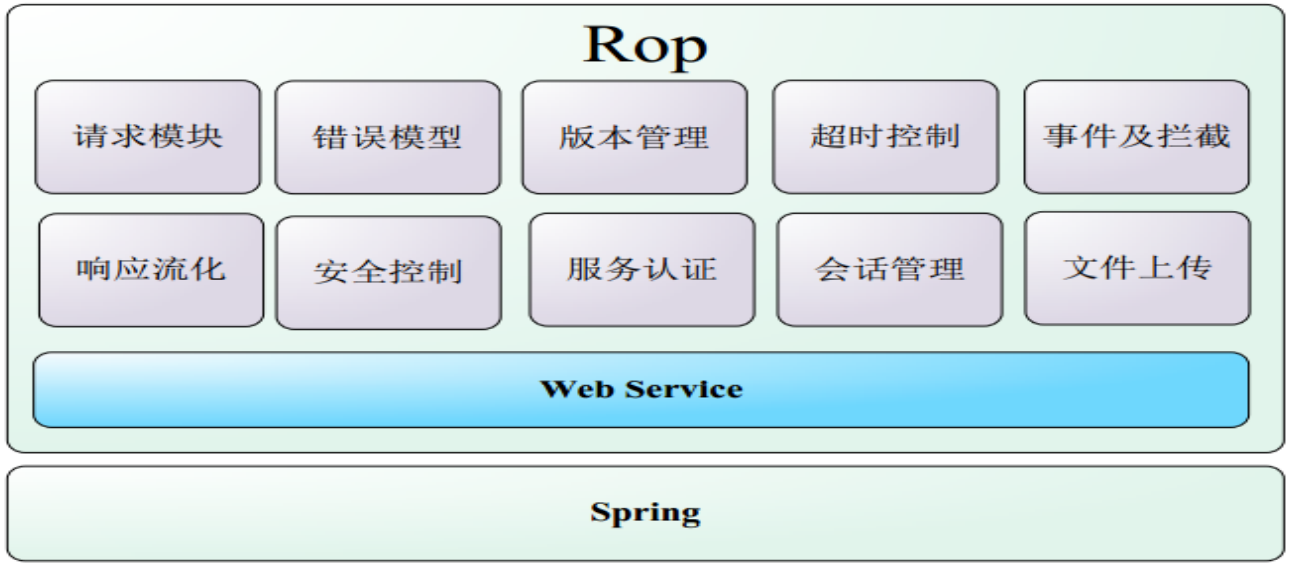



图 1 Rop 框架

从图 1 中，可以看到 Rop 所提供的大部分功能都是偏“应用层”的，传统技术型的 Web Service 框架是不会僭越到这些“应用层”的问题的。但是，在实际开发中，这些应用层的问题不但不可避免，而且非常考验开发者的设计经验，此外，这些工作还会占用较大的开发工作量。Rop 力图让开发者从这些复杂的工作中解脱出来，让他们可以真正聚焦服务平台业务逻辑的实现上。

# APP服务端的架构设计

推荐会话安全开源项目：

1. spring-oauth-server：http://git.oschina.net/shengzhao/spring-oauth-server

2. spring-security-oauth2.0: https://github.com/spring-projects/spring-security-oauth

3. apple-security: https://github.com/xushaomin/apple-security

# 配置管理

**配置中心的应用场景：**

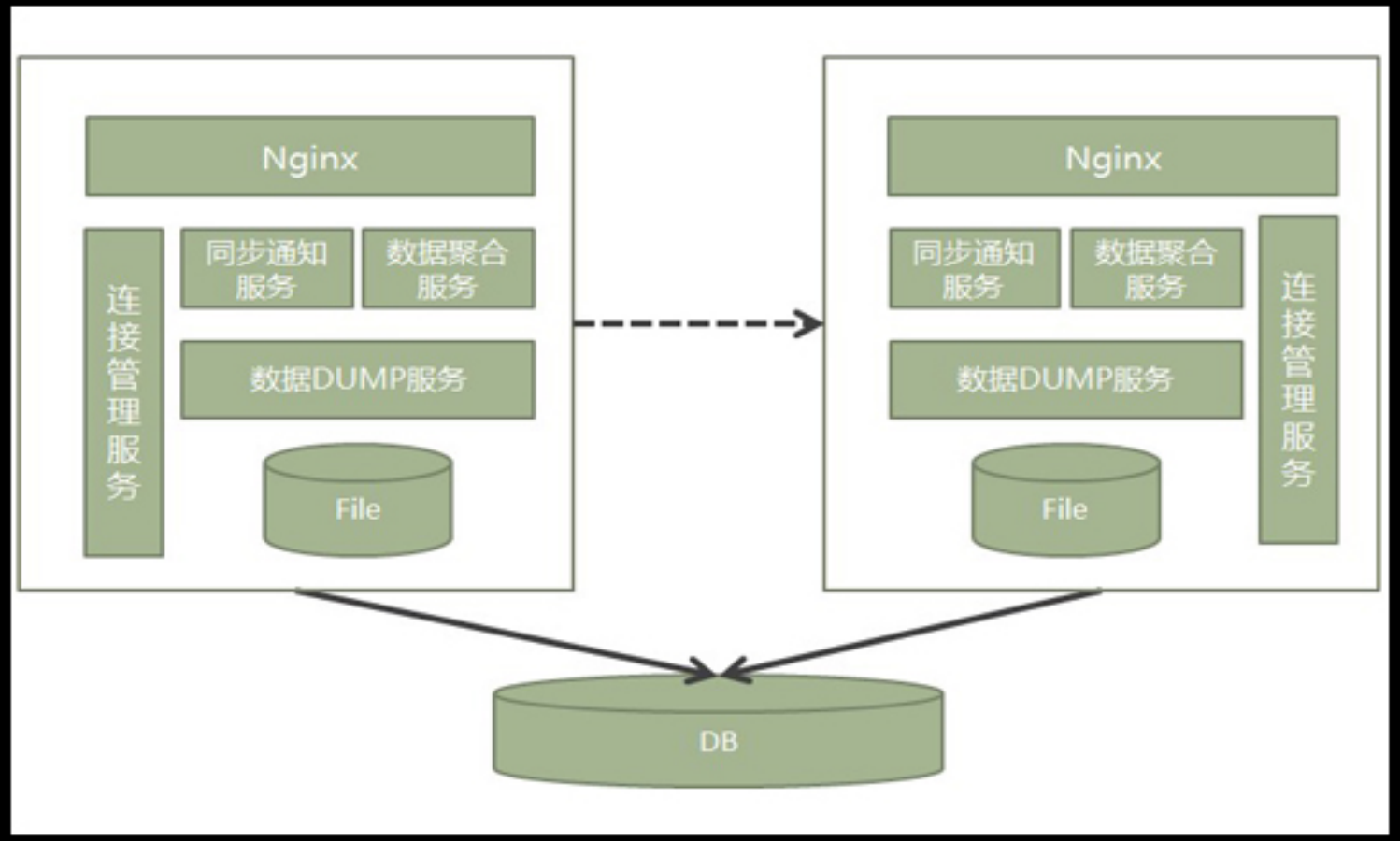
公司内存在多个系统，比如我们的web站点外加dubbo服务总超过200个，且系统之间的技术架构基本相同并且有一定的联系性

一套系统需要配置多个环境，我们有开发环境，测试环境，预发布环境，压力测试环境，线上环境

# 配置管理

配置中心开源项目：

1. 淘宝的diamond：http://code.taobao.org/svn/diamond/trunk



# 配置管理

配置中心开源项目：

1. 淘宝的diamond：http://code.taobao.org/svn/diamond/trunk

```
DiamondManager manager = new DefaultDiamondManager(group, dat
aId, new ManagerListener() {
    public void receiveConfigInfo(String configInfo) {
            // 客户端处理数据的逻辑
    }
});
```

```
import com.taobao.diamond.manager.impl.DefaultDiamondManager;

public class ExtendedPropertyPlaceholderConfigurer extends PropertyPlaceholderConfigurer {

        private static Logger logger = Logger.getLogger(ExtendedPropertyPlaceholderConfigurer.class);

        private Properties props;

        private String eventListenerClass;
```

# 配置管理

配置中心开源项目：

1. 淘宝的diamond：http://code.taobao.org/svn/diamond/trunk

```
                }
            } catch (Exception e) {
                logger.error(e);
            }

            DiamondManager manager = new DefaultDiamondManager(group, dataId, managerListeners);

            try {
                String configInfo = manager.getAvailableConfigureInfomation(30000);
                logger.warn("配置项内容: \n" + configInfo);
                if(!StringUtils.isEmpty(configInfo)) {
                    StringReader reader = new StringReader(configInfo);
                    props.load(reader);
                    PropertyConfigurer.load(props);
                }
                else {
                    logger.error("在配置管理中心找不到配置信息");
                }
            } catch (IOException e) {
                logger.error(e);
            }
        } else {
            PropertyConfigurer.load(props);
        }
        super.processProperties(beanFactory, props);
        this.props = props;
```

# 配置管理

配置中心开源项目：

1. 淘宝的diamond：http://code.taobao.org/svn/diamond/trunk

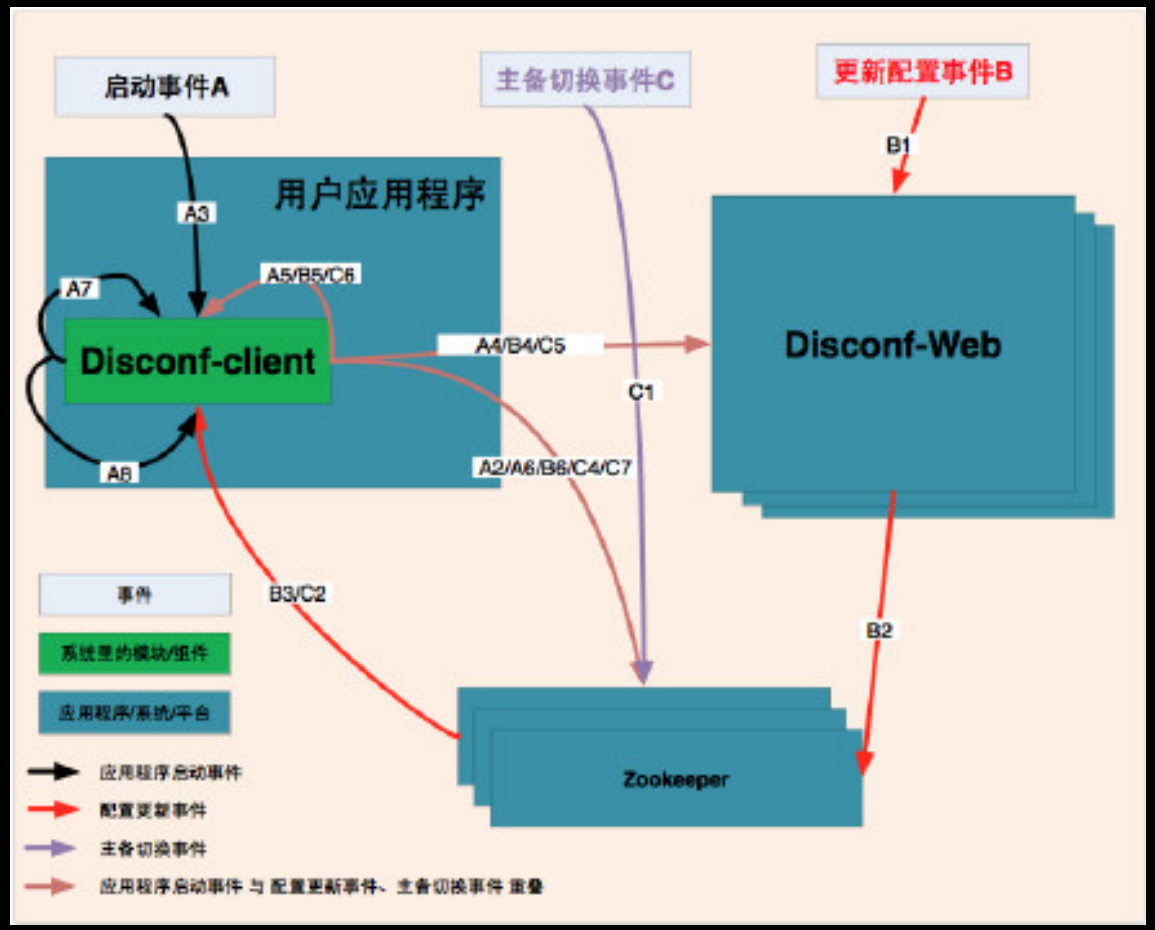
```
<bean id="propertyConfigurer"
    class="com.appleframework.config.ExtendedPropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
            <value>classpath*:system.properties</value>
        </list>
    </property>
    <property name="loadRemote" value="true" />
</bean>
```

ExtendedPropertyPlaceholderConfigurer代码：
https://github.com/xushaomin/apple-config

# 配置管理

配置中心开源项目：

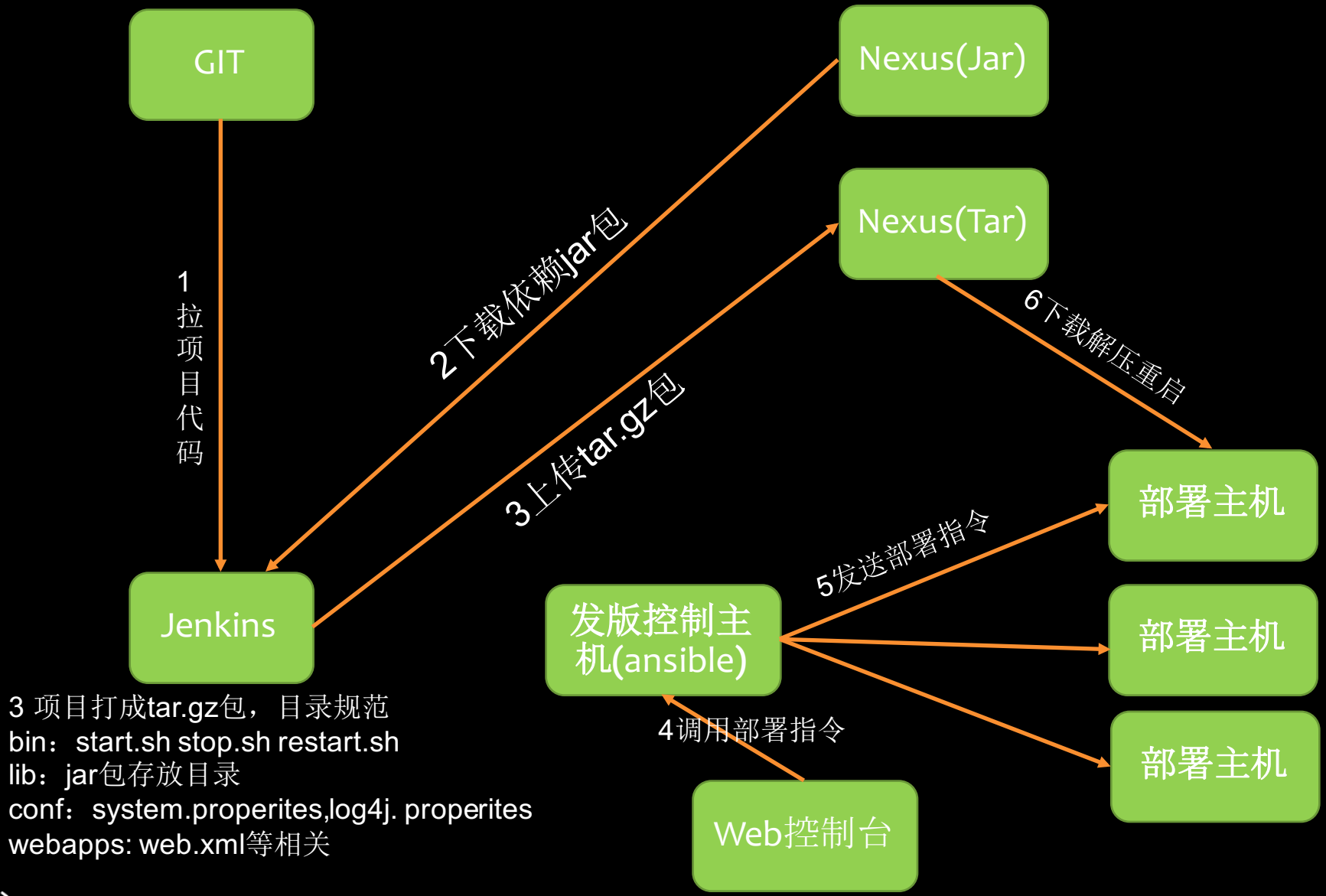
2. 百度的disconf：https://github.com/knightliao/disconf



# 自动化部署

提问：您们已经解决部署自动化了嘛？

# 自动化部署

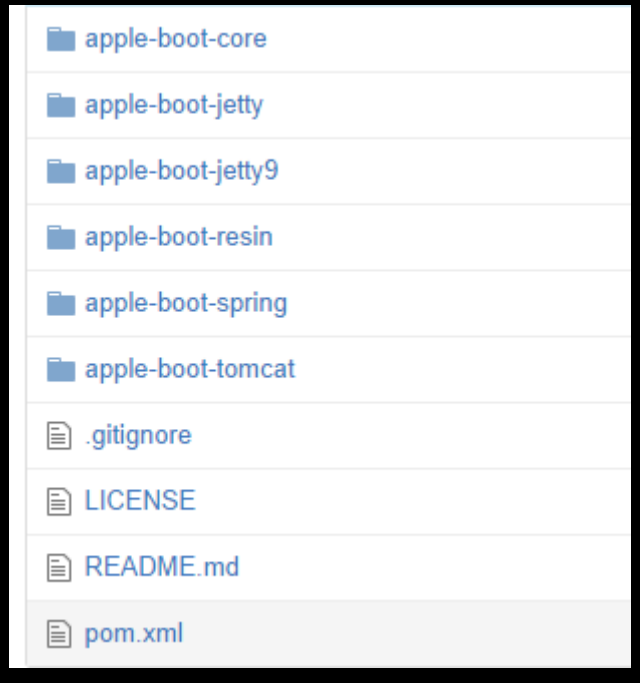



3 项目打成tar.gz包，目录规范
bin：start.sh stop.sh restart.sh
lib：jar包存放目录
conf：system.properites,log4j. properites
webapps: web.xml等相关

# 自动化部署

自动化开源项目：

1. 容器嵌入式：https://github.com/xushaomin/apple-boot




```
<bean id="threadPool" class="org.eclipse.jetty.util.thread.QueuedThreadPool">
        <property name="minThreads" value="${web.min.threads:100}" />
        <property name="maxThreads" value="${web.max.threads:1000}" />
        <property name="maxQueued" value="${web.max.queued:-1}" />
</bean>
```

# 自动化部署

自动化开源项目：

1. 容器嵌入式：https://github.com/xushaomin/apple-boot
   通过maven的assembly插件打成 tar.gz包，tar.gz的规范
   bin
   start.sh
   stop.sh
   restart.sh
   server.sh
   dump.sh
   lib
   apple-boot-spring-0.2.8.RELEASE.jar
   apple-boot-jetty-0.2.8.RELEASE.jar
   ……
   conf
   log4j. properties
   system. Properties
   webapps
   web.xml
   ……

# 自动化部署

自动化开源项目：

1. 容器嵌入式：https://github.com/xushaomin/apple-boot

```
    fi
    JAVA_JMX_OPTS=" -Dcom.sun.management.jmxremote.port=$JMX_PORT -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authe
fi

JAVA_MEM_OPTS=" -server -Xmx4g -Xms2g -Xmn512m -XX:PermSize=128m -Xss256k -XX:+DisableExplicitGC -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemark


echo -e "Starting the $SERVER_NAME ...\c"
nohup java $JAVA_OPTS $JAVA_MEM_OPTS $JAVA_DEBUG_OPTS $JAVA_JMX_OPTS -classpath $CONF_DIR:$LIB_JARS com.appleframework.boot.Main env=$ENV > $ST

COUNT=0
while [ $COUNT -lt 1 ]; do
    echo -e ".\c"
    sleep 1
    if [ -n "$SERVER_PORT" ]; then
        if [ "$SERVER_PROTOCOL" == "dubbo" ]; then
```

# 自动化部署

自动化开源项目：

1. 容器嵌入式：https://github.com/xushaomin/apple-boot

```
#!/bin/bash
VERSION=$1
NEXUS_URL="http://mvnrepo.appleframework.com:8081/nexus"
INSTALL_PATH="/work/www/biz-user-provider"
GROUP_ID="com.appleframework.biz"
ARTIFACT_ID="biz-user-provider"
if [ -z "$VERSION" ]; then
    echo "ERROR: THE PROJECT'S VERSION MUST TO INPUT!"
    echo "ERROR: THE PROJECT'S DEPLOY EXIT!"
    exit 1
fi
if [ ! -d $INSTALL_PATH ];then
    mkdir -p $INSTALL_PATH
fi
cd $INSTALL_PATH
FILE_PATH=${INSTALL_PATH}"/"${ARTIFACT_ID}"-"${VERSION}
FILE_NAME=${ARTIFACT_ID}"-"${VERSION}"-assembly.tar.gz"
FILE_NAME_BAK=${ARTIFACT_ID}"-"${VERSION}"-assembly-bak.tar.gz"
WORK_PATH=${INSTALL_PATH}"/work"
rm -rf $FILE_NAME_BAK
mv -f $FILE_NAME $FILE_NAME_BAK
DOWNLOAD_URL=${NEXUS_URL}"/service/local/repositories/releases/content/"${GROUP_ID//./\/}"/"${ARTIFACT_ID}"/"${VERSION}"/"${FILE_NAME
rm -rf $FILE_PATH
rm -rf $WORK_PATH
wget -c -q $DOWNLOAD_URL
tar -zxf $FILE_NAME
mv $FILE_PATH $WORK_PATH
BIN_PATH=$WORK_PATH"/bin"
cd $BIN_PATH
./restart.sh release jmx
echo "Start Over!"
```

# 自动化部署

自动化开源项目：

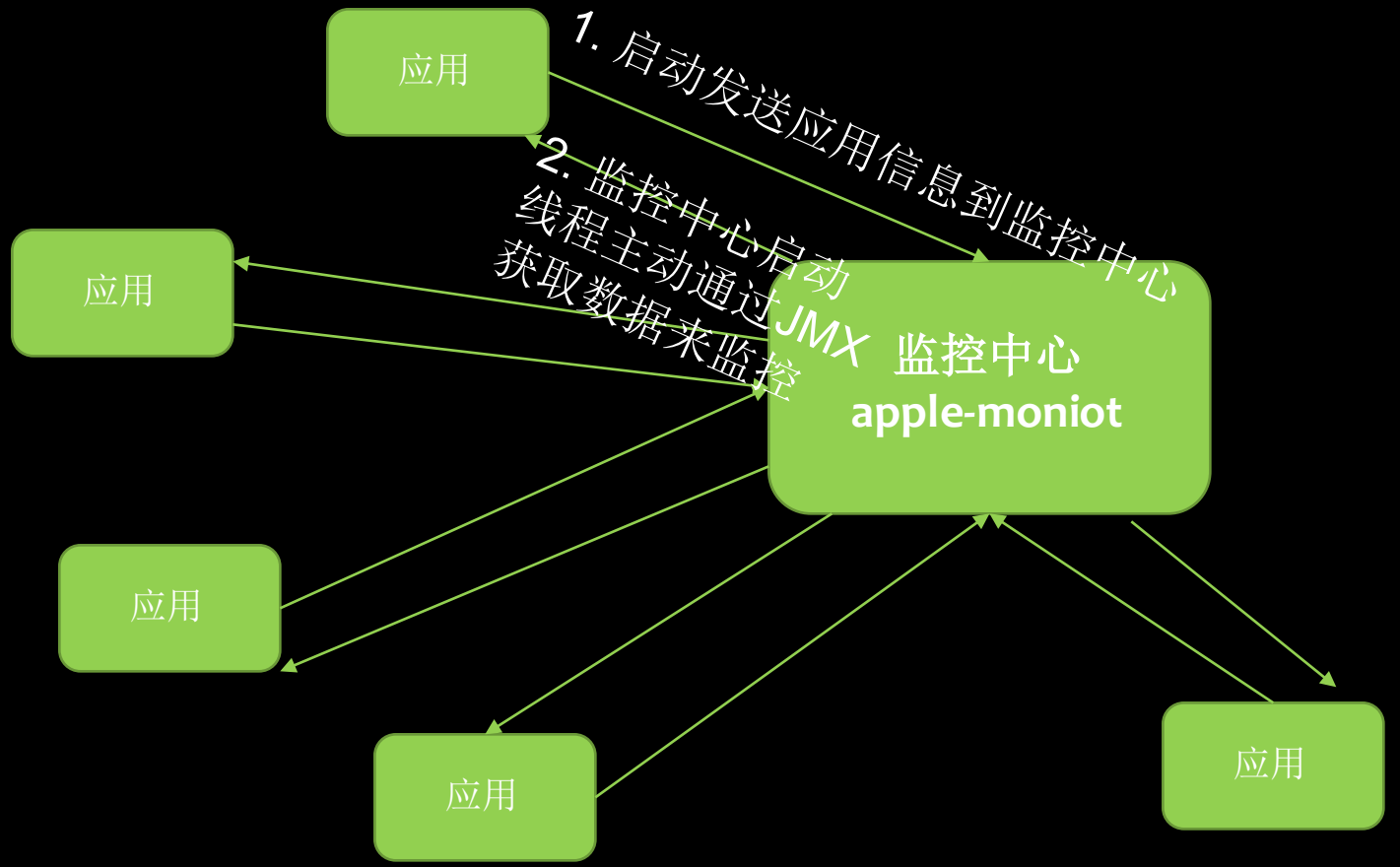
apple-deploy: https://github.com/xushaomin/apple-deploy
Rundeck: http://rundeck.org/
Ansible: http://docs.ansible.com/

# 自动化监控

JMX（Java Management Extensions，即Java管理扩展）是一个为应用程序、设备、系统等植入管理功能的框架。JMX可以跨越一系列异构操作系统平台、系统体系结构和网络传输协议，灵活的开发无缝集成的系统、网络和服务管理应用。

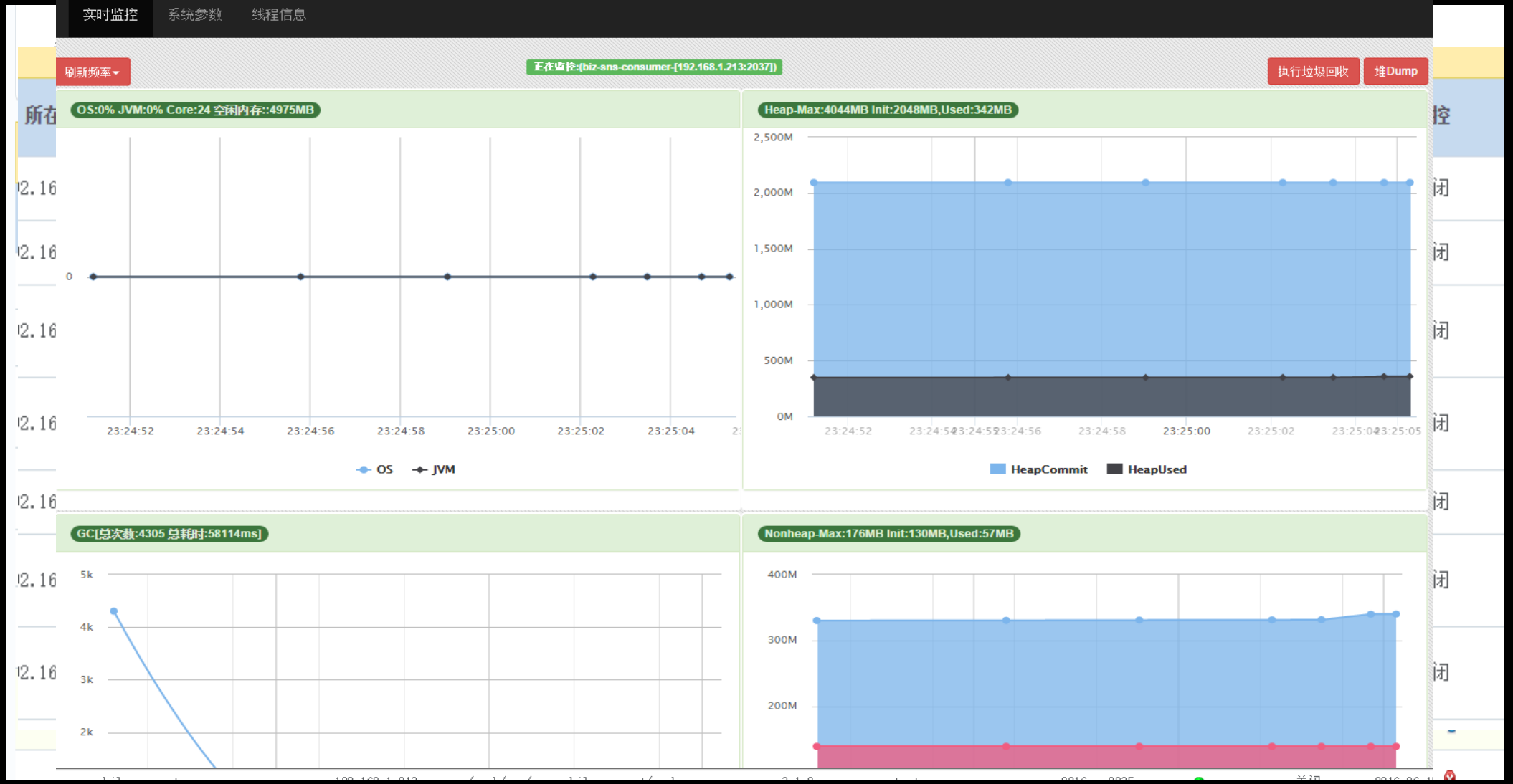
```
-Dcom.sun.management.jmxremote.port=$JMX_PORT
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

# 自动化监控


应用
1. 启动发送应用信息到监控中心
2. 监控中心启动
线程主动通过JMX
获取数据来监控
监控中心
apple-moniot
应用
应用
应用
应用

# 自动化监控

apple-monitor：https://github.com/xushaomin/apple-monitor



# 分布式日志

分布式日志开源项目：

闪电狗: https://github.com/flash-dog/flash-dog

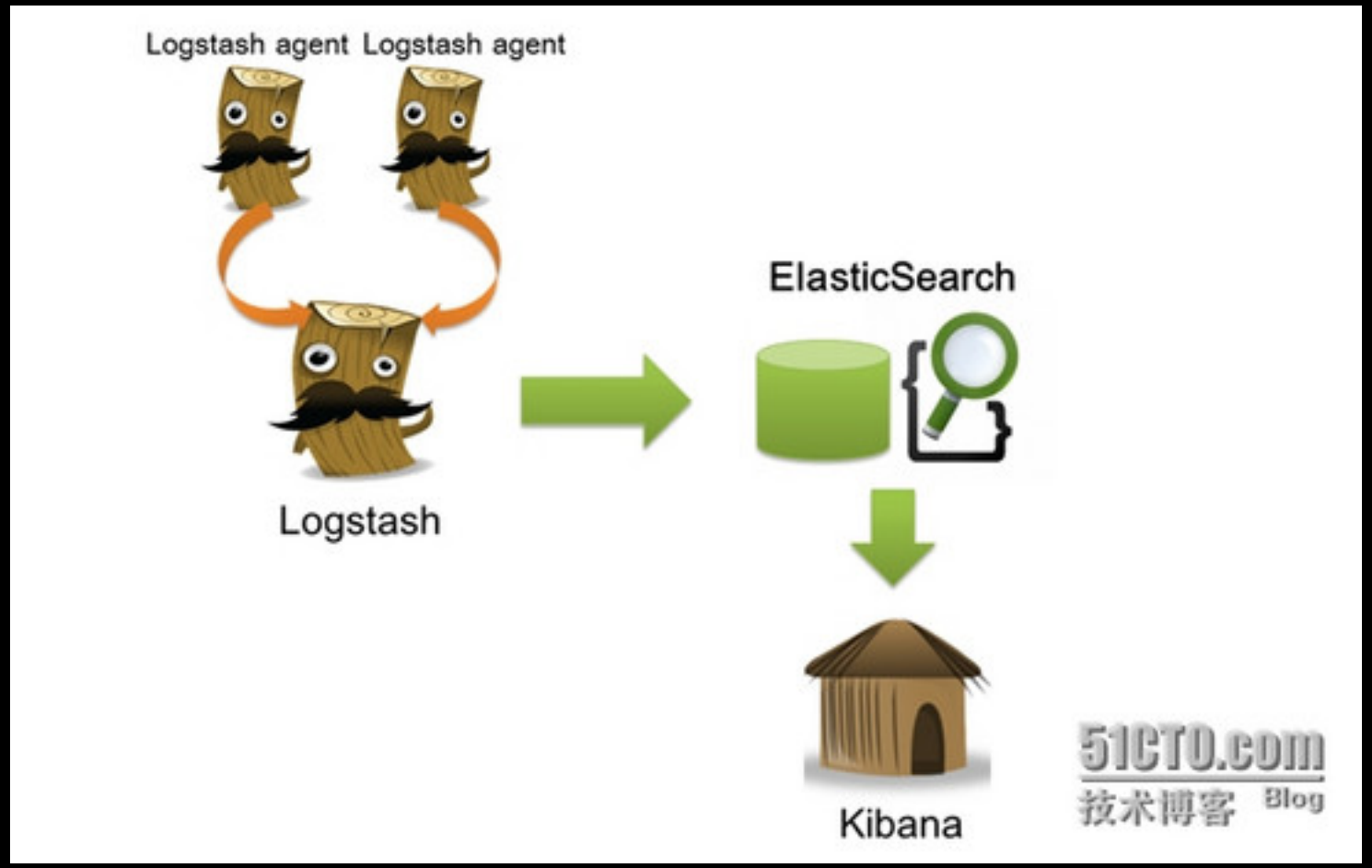
```
log4j.appender.MongoDB=org.log4mongo.AsynMongoURILayoutAppender
log4j.appender.MongoDB.layout=org.log4mongo.layout.MongoDbDefaultLayout
log4j.appender.MongoDB.layout.ConversionPattern={"timestamp":"%d","level":"%p","className":"%c","message":"%m","pid":"%V","ip":"%I"
log4j.appender.MongoDB.threadCount=2
log4j.appender.MongoDB.jvmMonitor=true
log4j.appender.MongoDB.mongoURI=mongodb://localhost:27017/monitor_test?slaveOk=true
log4j.appender.MongoDB.collectionName=flash_dog_log
```

# 分布式日志

分布式日志开源项目：

ELK: ElasticSearch 、 Logstash 和 Kiabana

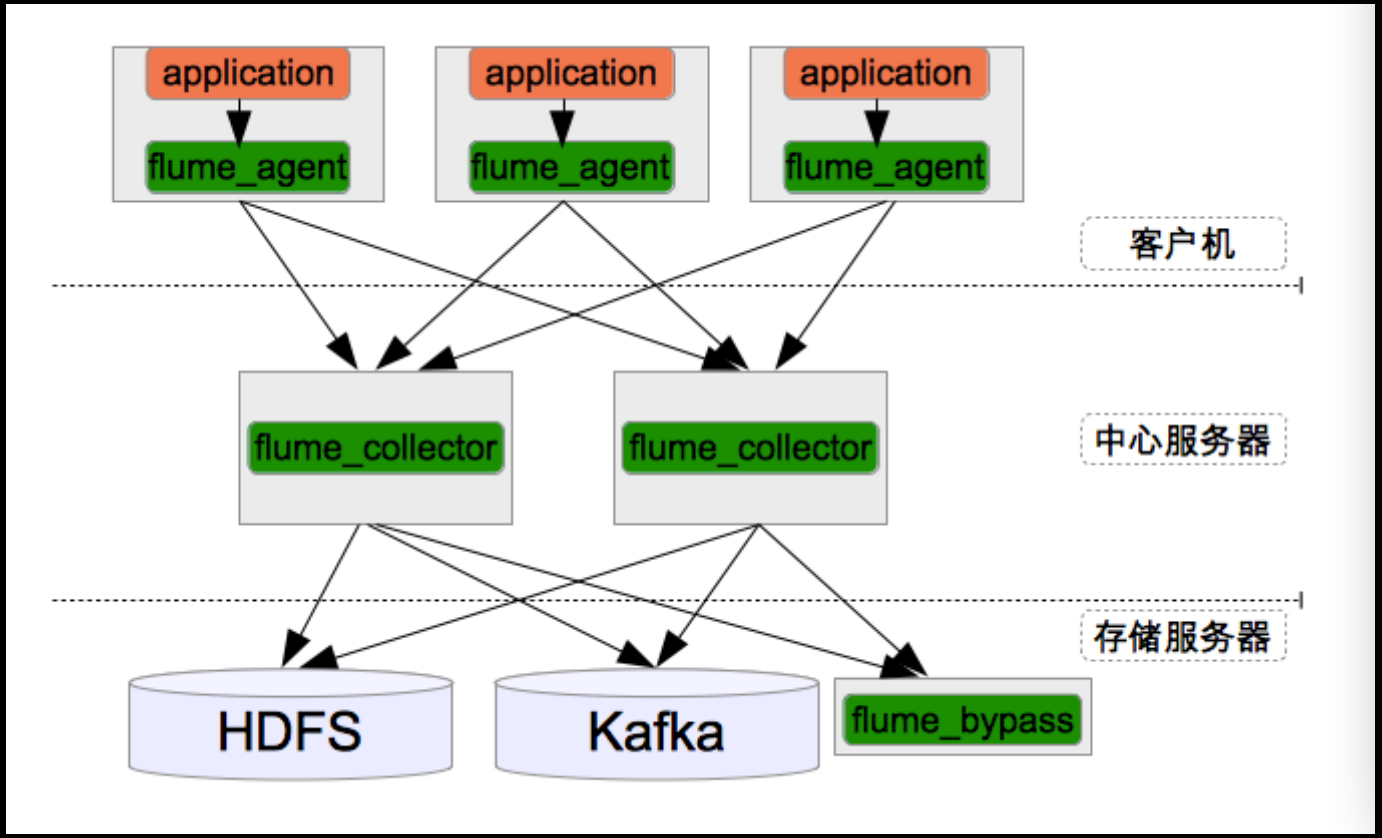
https://www.elastic.co/products



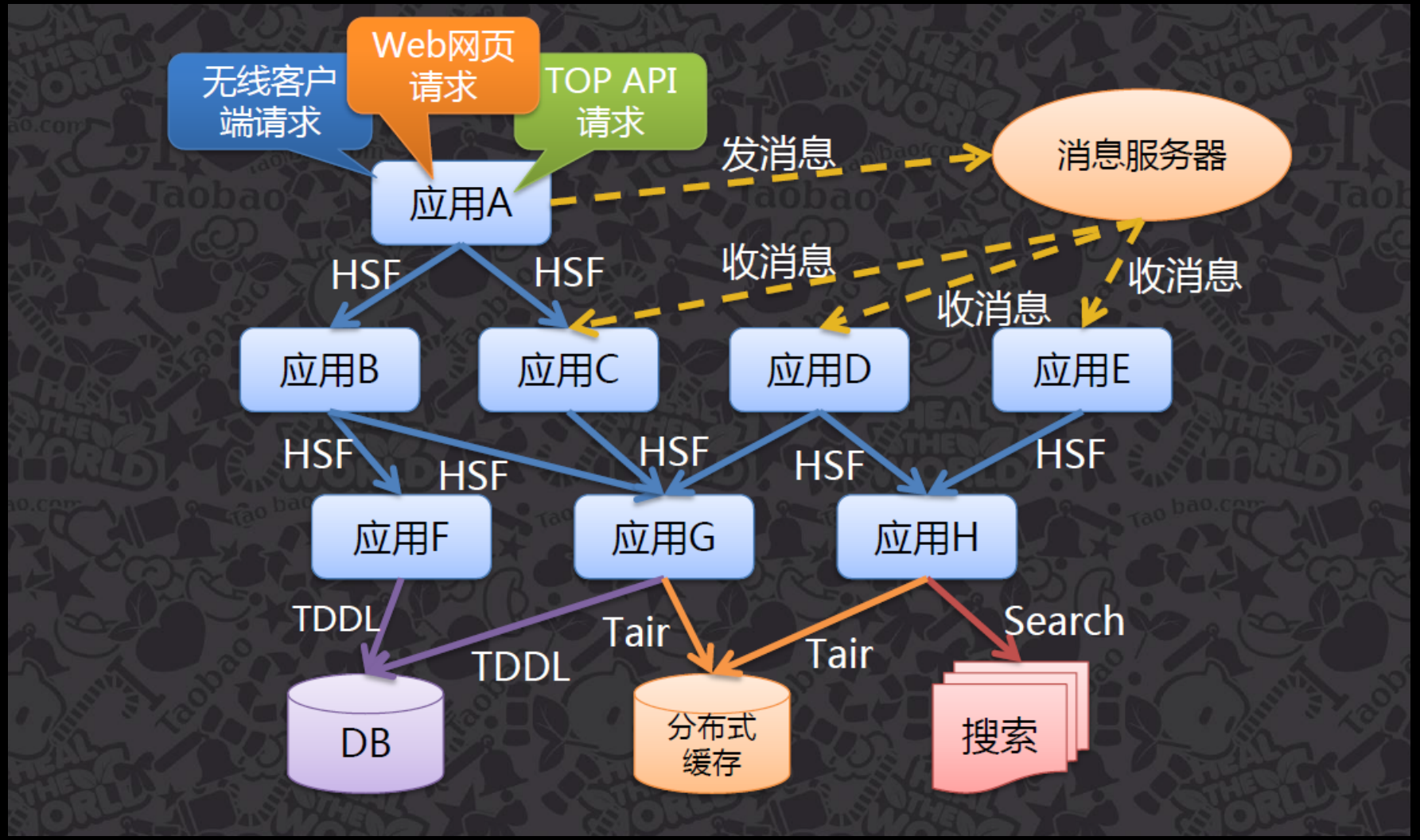
# 分布式日志

分布式日志开源项目：

大数据方案: Flume（NG） 、Kafka 、 Hadoop 、Hive 、 Storm、 Spark



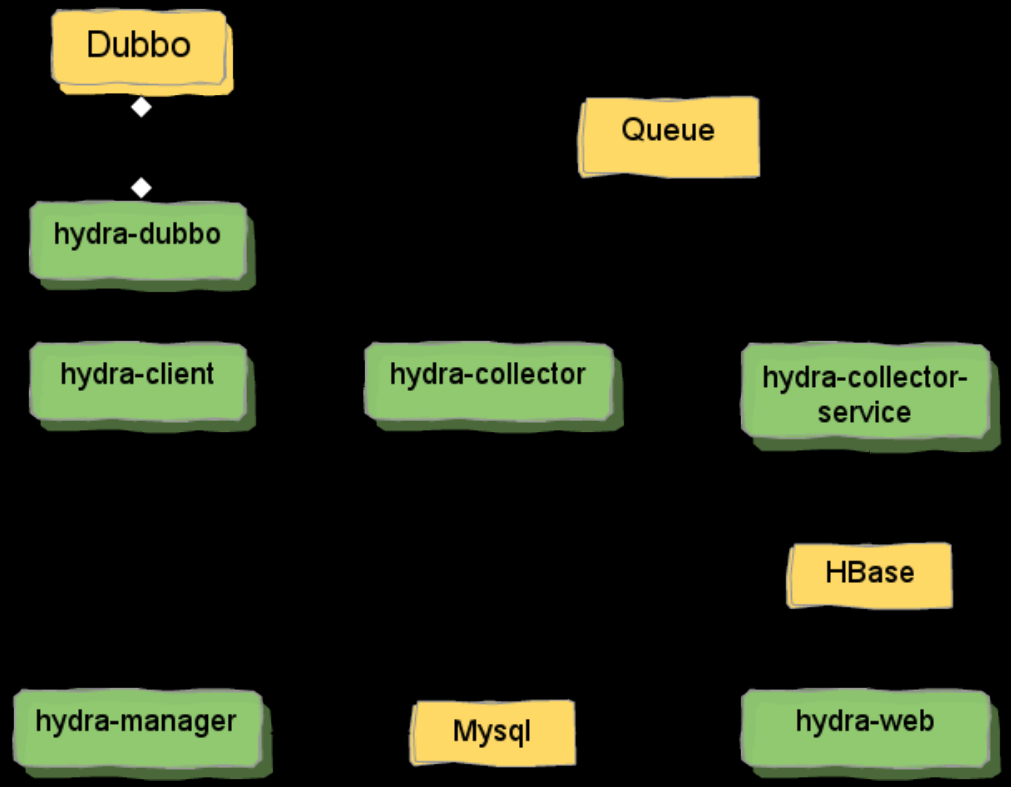
# 分布式跟踪


无线客户端请求
Web网页请求
TOP API请求
应用A
发消息
消息服务器
收消息
收消息
收消息
HSF
HSF
应用B
应用C
应用D
应用E
HSF
HSF
HSF
HSF
HSF
应用F
应用G
应用H
TDDL
TDDL
Tair
Tair
Search
DB
分布式缓存
搜索

# 分布式跟踪

分布式跟踪系统开源项目：

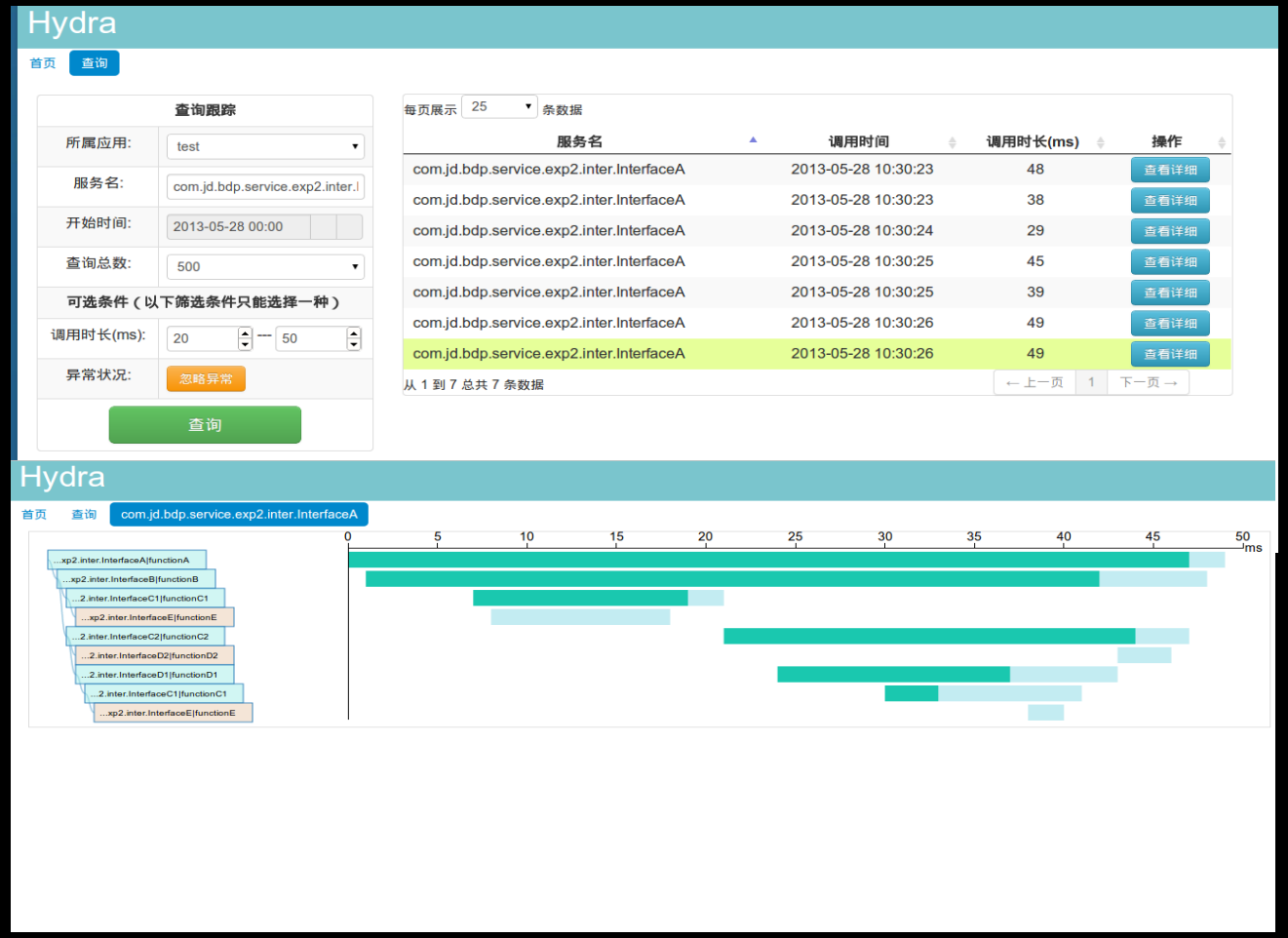
**Hydra：https://github.com/odenny/hydra**



# 分布式跟踪

分布式跟踪系统开源项目：

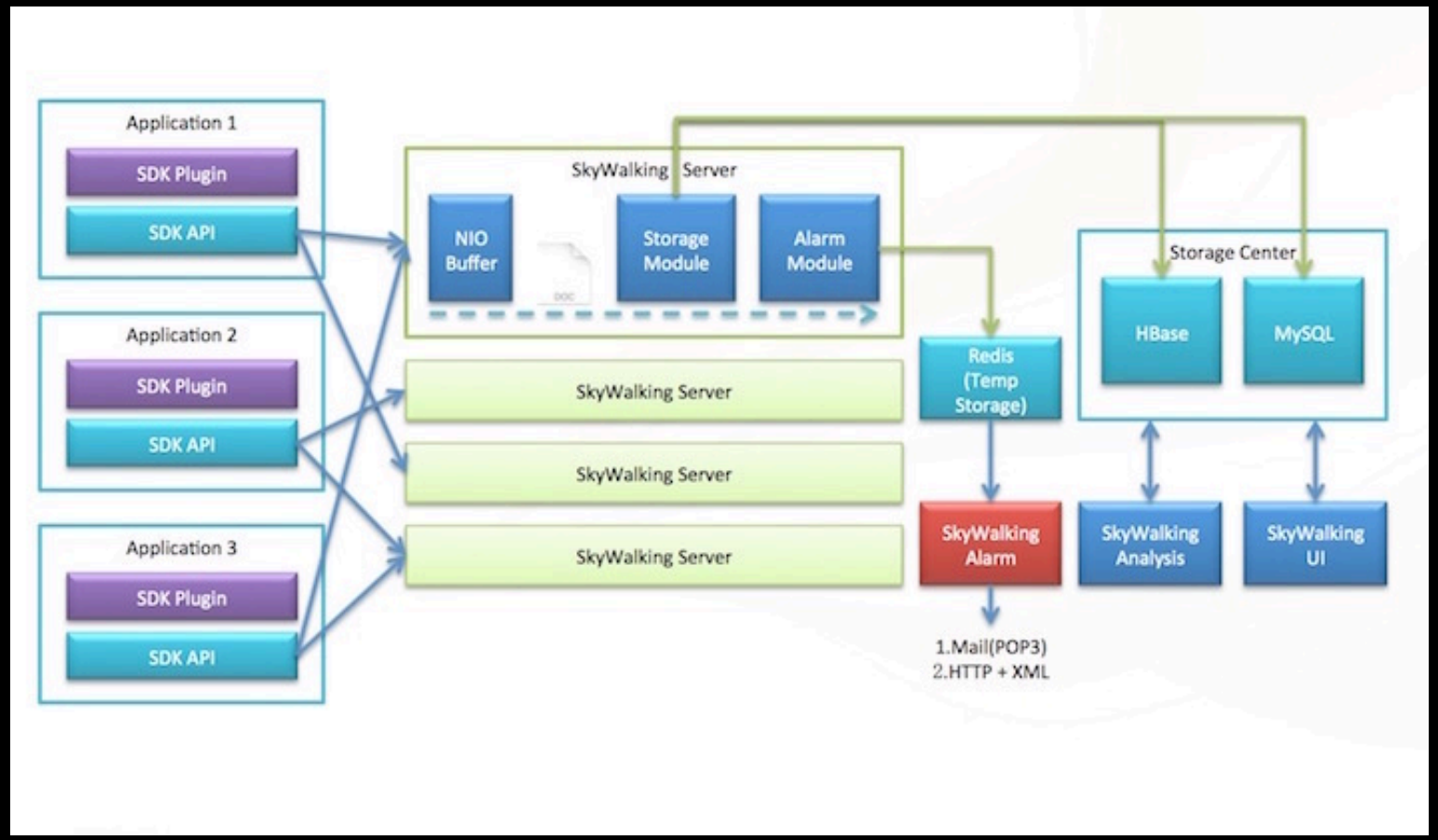
**Hydra：https://github.com/odenny/hydra**



# 分布式跟踪

分布式跟踪系统开源项目：

**sky-walking：https://github.com/wu-sheng/sky-walking**



# 分布式跟踪

分布式跟踪系统开源项目：

**sky-walking：https://github.com/wu-sheng/sky-walking**

# 分布式跟踪

分布式跟踪系统开源项目：

**sky-walking**：https://github.com/wu-sheng/sky-walking

# Thanks!