**Eagyele详细设计文档**

**最后更新: 2014/8/12**

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 作者 | 版本 | 备注 |
| 2014/8/12 | 刘宏伟 | 1.0 | 创建 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

1 总述 3

2 词汇表 3

3 需求 4

3.1 异常预警管理 4

3.1.1 用户组管理 4

3.1.2 应用管理 5

3.1.3 预警规则管理 6

3.2 应用预警日志管理 7

3.3 Dubbo监控管理 8

3.3.1 跟踪日志检索 8

3.3.2 异常跟踪日志检索 9

3.3.3 跟踪统计 10

3.4 异常日志预警 10

3.5 Dubbo跟踪预警 10

4 Eagleye概要设计 11

4.1 项目结构 11

4.2 系统架构 12

5 Eagleye详细设计 13

5.1 Dubbo跟踪详细设计 13

5.1.1 设计前的考虑 13

5.1.2 参考资料介绍 13

5.1.3 Dubbo的关键领域模型 14

5.1.4 跟踪领域模型 15

5.1.5 主要类图 16

5.2 异常预警详细设计 16

6 部署 16

6.1 配置文件详细说明 16

6.2 测试环境说明 19

6.3 生产环境说明 20

6.4 eagleye部署 20

7 遇到的坑 21

7.1 曾经已解决的坑 21

7.2 曾经到现在未解决的坑 22

# 总述

XX电子商务平台是由很多个应用集群组成的复杂的分布式系统.这里面主要由处理用户请求的前端系统和提供服务的后端系统组成.这么多的系统同时运行,怎样可以快速的知道系统产生的问题,就需要我们可以随时收集各个系统之间的异常,并进行预警.各个系统之间一般有RPC调用和异步消息通信等手段.由于我们采用的是Dubbo进行服务的治理, 所以RPC调用链的跟踪,主要是对Dubbo服务的调用跟踪.实时的跟踪与异常预警帮助我们了解我们系统的健康程度,在复杂的业务织网中快速定位系统已经发生或潜在的危险,减小运营风险.

为了便于文档的维护, 我将详细设计文档的内容丰富为: 需求文档; 概要设计; 详细设计; 部署文档,等等这些文档的集合, 并通过不同的章节进行详细描述. 这样随着项目的推进, 我们只维护这一份文档.

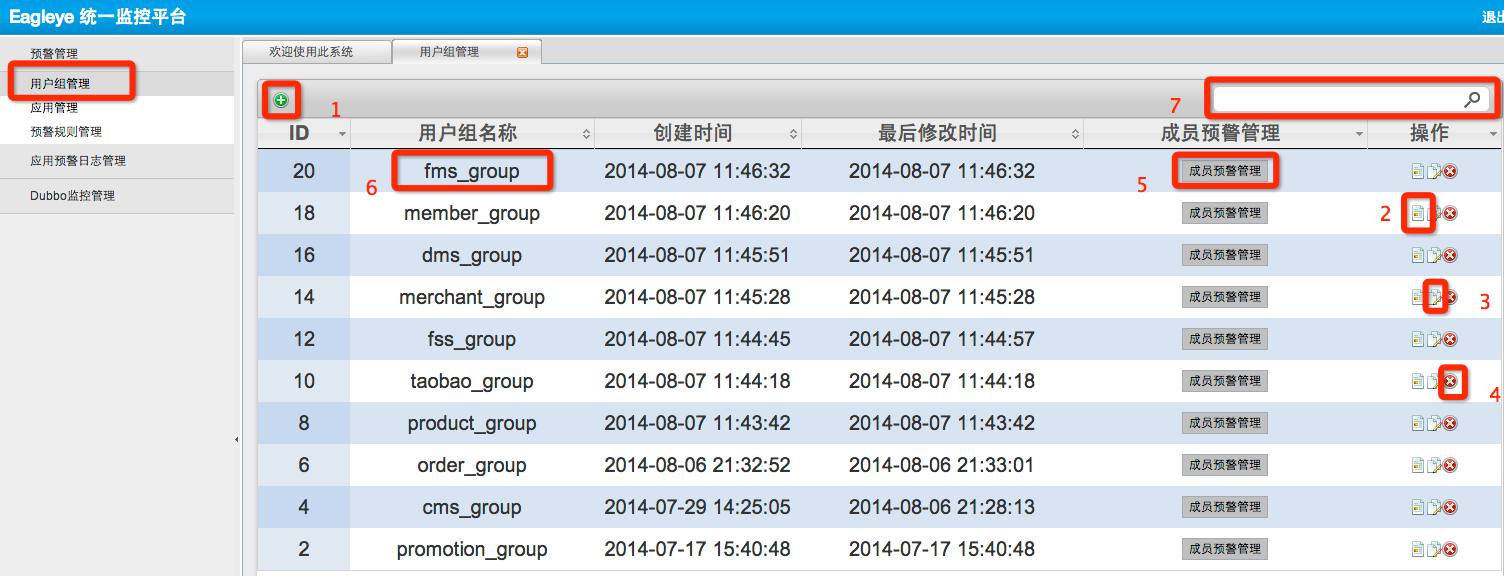
# 词汇表

|  |  |
| --- | --- |
| 词汇 | 含义（原词组） |
| Dubbo | 公司的RPC调用框架, 用它来进行各个系统的服务进行治理. |
| TraceId | 某一个服务调用链的唯一标识, 通过它可以检索到具体的调用链. |
| SpanId | 一个服务调用链中的某一个具体调用的唯一标识. |
| 子调用 | 比如: A调用B调用C, 其中B调用C的过程可以称为一个子调用过程. |
|  |  |
|  |  |
|  |  |

# 需求

## 异常预警管理

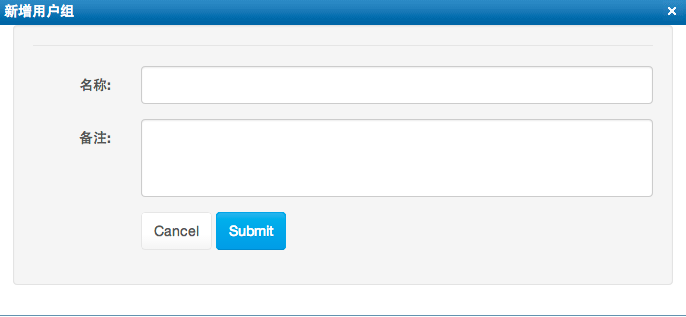
### 用户组管理



(图3-1)

用户组管理主要是将各个开发组的成员进行聚合分组, 便于将来异常日志预警可以更精准的推送个各个应用对应的项目组,如图3-1,为用户组的管理界面. 这其中包括的功能如下:

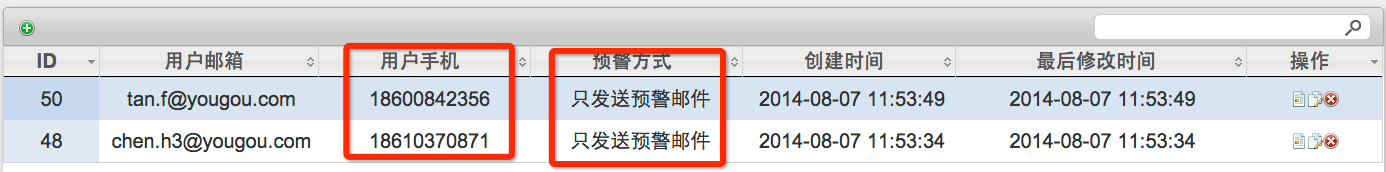
1. 用户组的添加.



(图3-2)

点击添加图标, 弹出用户组添加页面, 这里只需要填写用户组名称, 如: cms用户组,可以命名为: cms\_group, 如图:3-2.

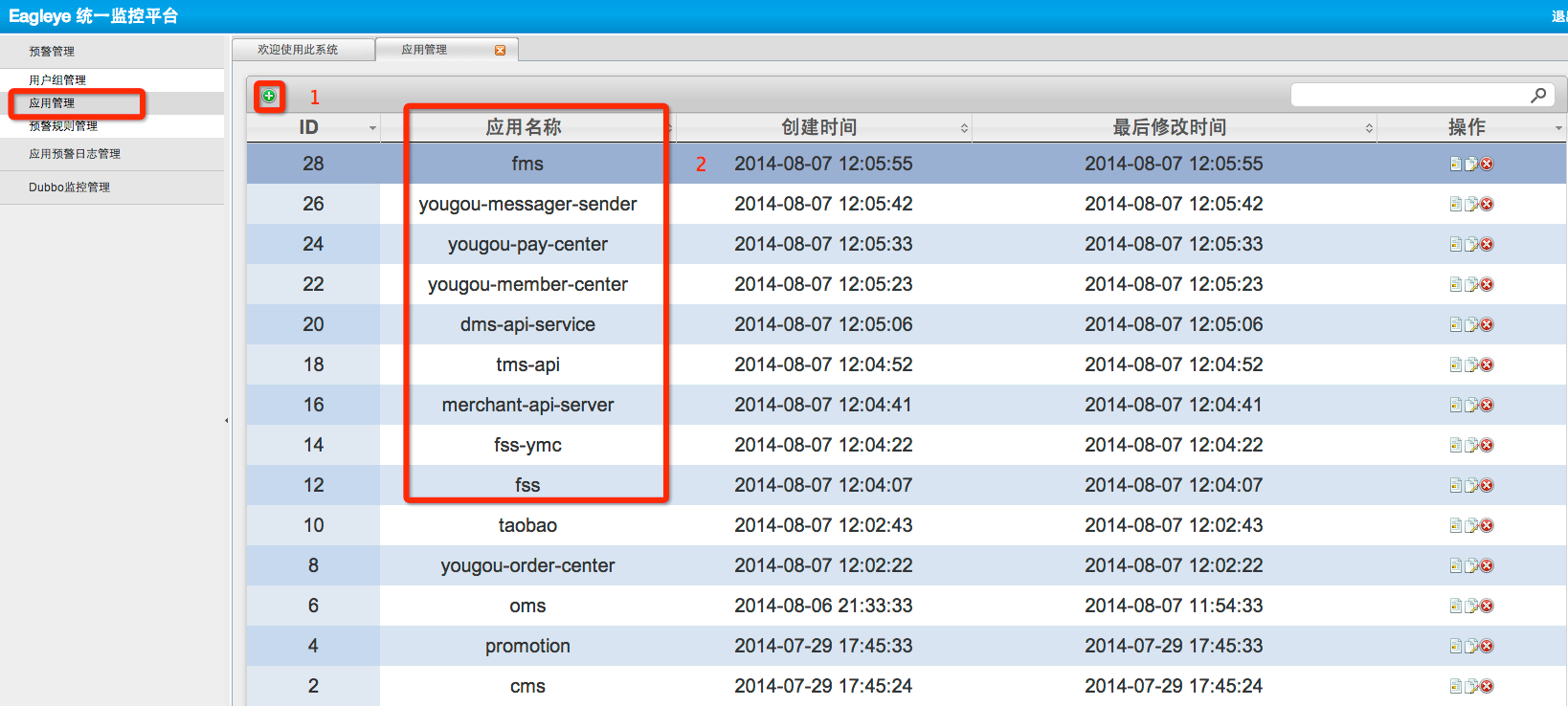
1. 用户组详情页, 可以看到具体的信息, 这个在之后的功能中都是以此类推, 不在做详细的描述.
2. 用户组编辑页面, 这里可以通过该功能对用户组的名称, 备注进行修改, 该功能在之后的模块也不再做详细描述.
3. 删除用户组功能, 该功能只是对数据进行逻辑删除, 并非进行物理删除, 之后的模块中的删除功能, 将都是按照逻辑删除进行处理.
4. 用户组成员管理功能



(图3-3)

用户组成员管理中的用户手机是必填项, 由于现在还没有对接公司的统一权限管理(还未开发完成, 这边也并不着急对接), 所以用户的选择暂时是写死在页面上, 将来获取了用户列表可以通过将这部分进行替换. 已经方式有四种可以选择, 分别是: 只发送短信; 只发送邮件; 短信和邮件都发送; 不进行任何预警. 用户手机信息是必填项, 如图3-3.

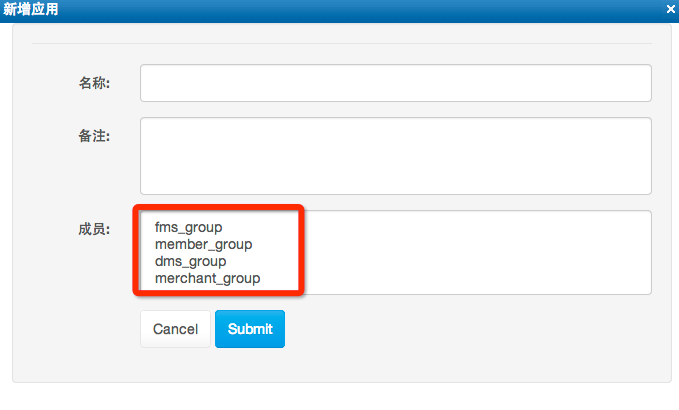
### 应用管理



(图3-4)

应用管理主要是在eagleye监控平台注册各个系统应用, 如图3-4.

1. 注册应用.

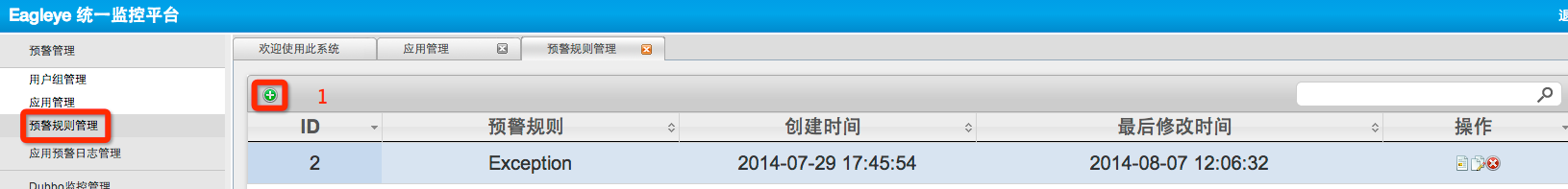


(图3-5)

如图3-5所示, 注册应用是可以选择应用所在的项目组, 一个应用可以匹配多个项目组, 也就是说一个应用产生的异常预警可以推送给多个用户组成员.

1. 这里的应用名称必须要和收集预警信息的名称保持一致, 该名称是匹配预警信息精准推送的重要标识.

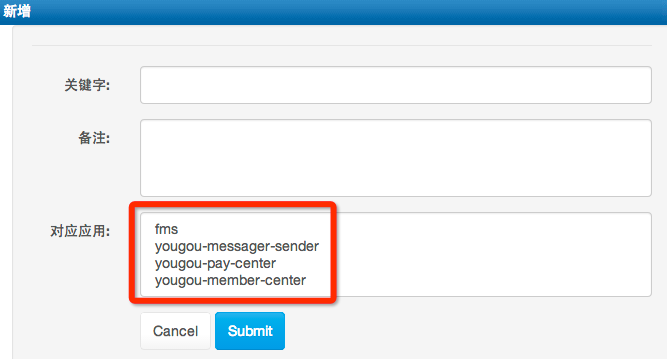
### 预警规则管理



(图3-6)

预警规则是设定哪些异常信息满足预警规则才进行预警, 如图3-6.

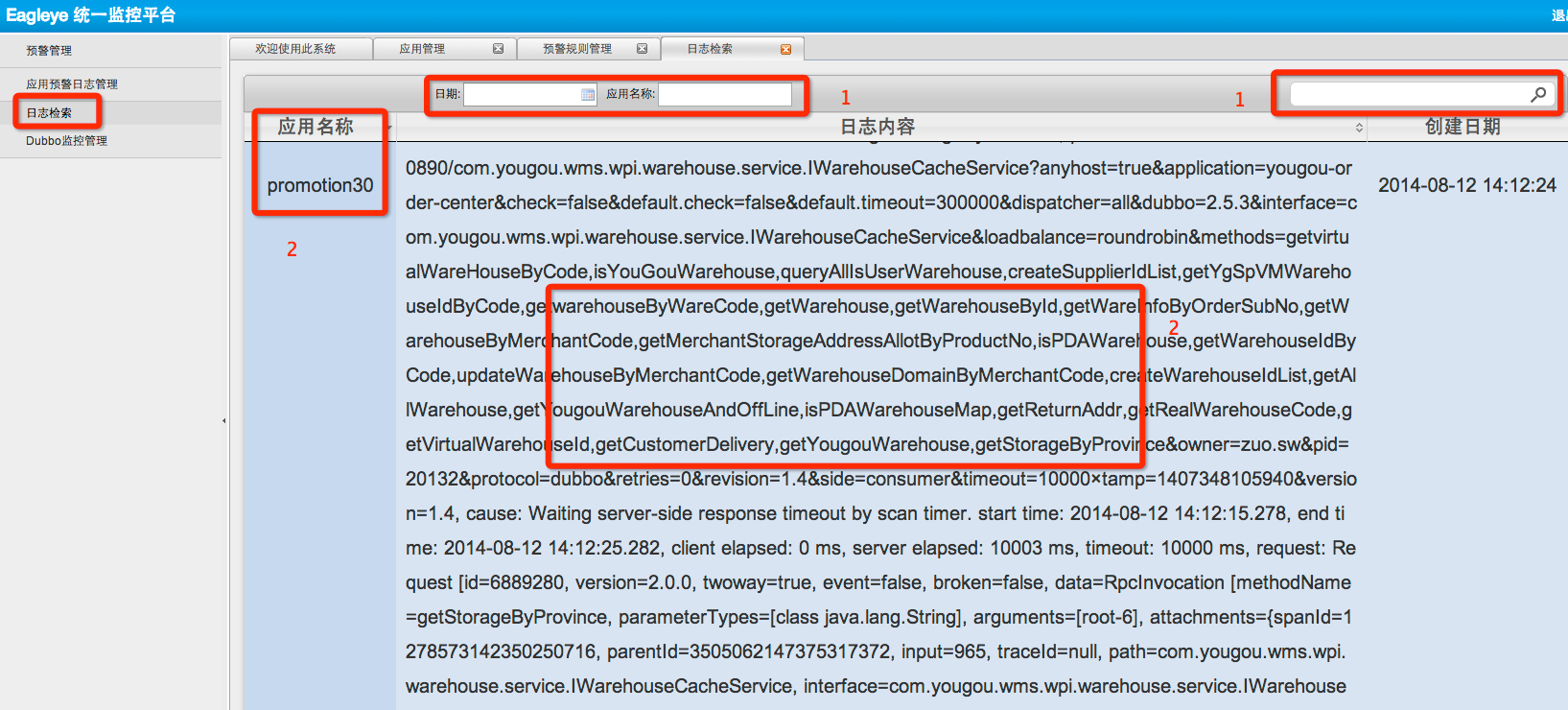
1. 添加预警规则



(图3-7)

预警规则中的关键字可以是正则表达式, 也可以是个字符, 只有收集的异常信息满足该正则表达式, 或者包含该字符, 才满足预警条件. 每个预警规则可以同时适用多个应用, 不用每个应用进行重复设置.如图3-7.

## 应用预警日志管理



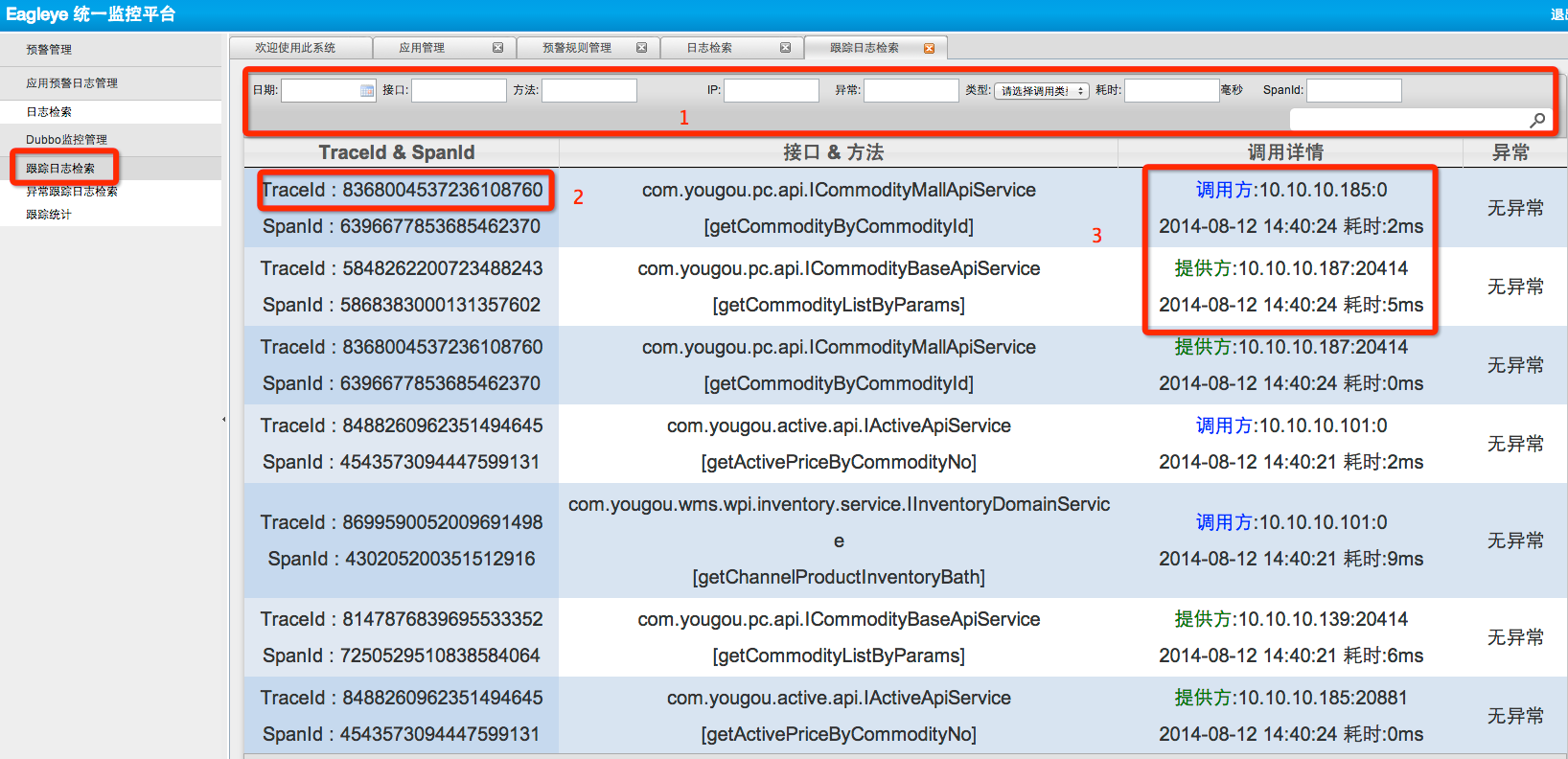
(图3-8)

这里可以对各个业务系统收集到的异常进行检索, 预警短信只会发送截取的异常信息, 具体的异常详情可以通过该功能进行检索, 如图3-8.

1. 可以针对某一天, 某个应用机器, 和具体的异常关键字进行检索.
2. 检索列表中显示的应用名称是之前应用注册时的名称和应用所在机器的IP最后一位, 日志内容则是应用报的具体异常详情.

## Dubbo监控管理

### 跟踪日志检索



(图3-9)

这里可以查询到七天之内的所有dubbo调用日志,如图3-9.

1. 可以七日之内具体某一天的跟踪日志, 也可以通过接口名称, 方法名进行精确检索. 也可以通过IP地址定位到某一台具体的dubbo应用服务器. 我们可以通过设置耗时值来检索超过该时间的日志. SpanId可以唯一标识一条跟踪日志.
2. 查看调用链详情.

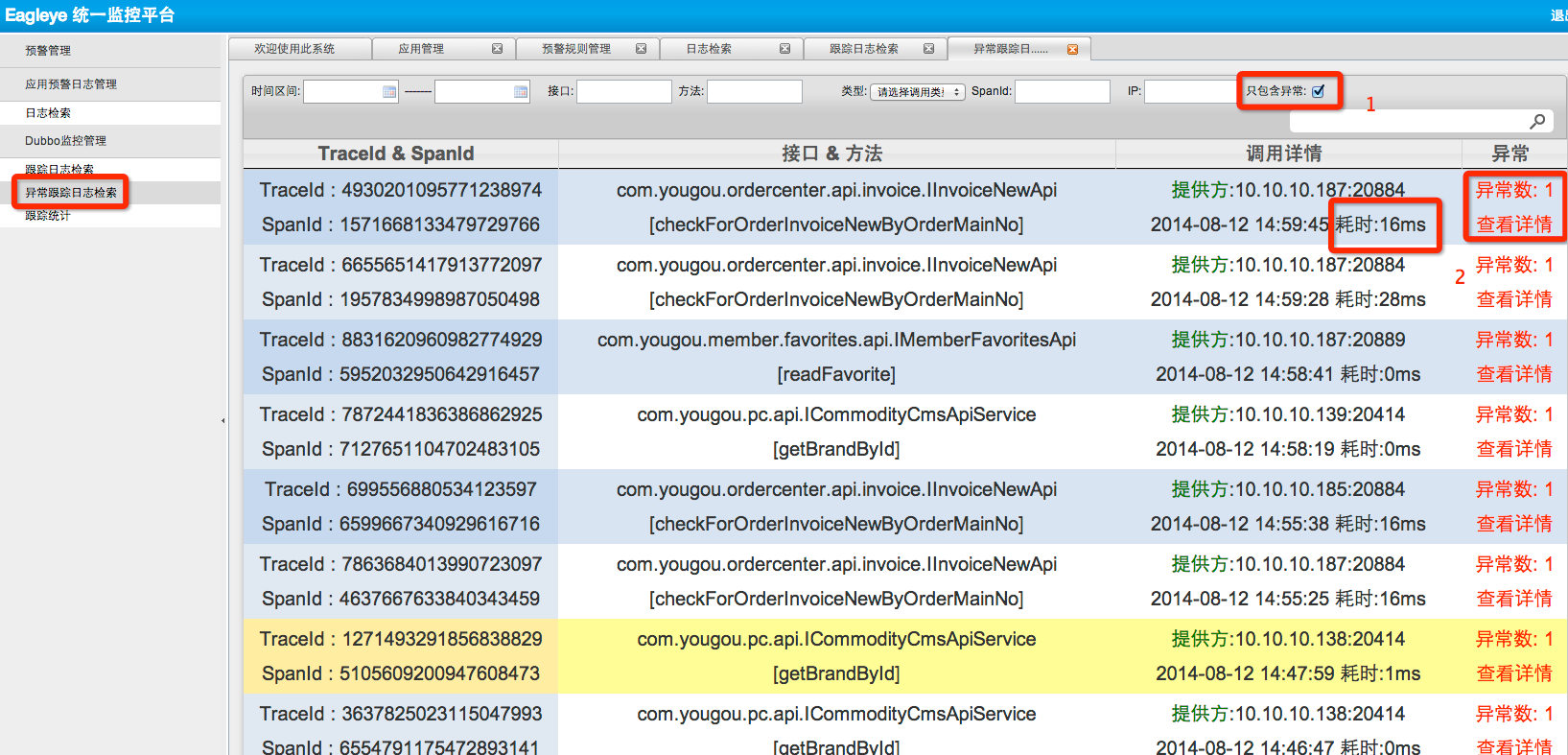


(图3-10)

如果想查看某一条跟踪日志所在调用链的详情, 请点击TraceId,即可进入调用链详情, 如图3-10, 该链条是一个树形结构, 可以看到一个调用链从顶层消费者开始到最底层的提供者的详细调用路径, 每一步所消耗的时间, 我们可以通过调用链查看到一个调用链中具体是什么环境出现了问题.

1. 某一个调用日志可以通过详情了解调用的起始时间, 是调用方还是提供方产生的轨迹,具体消耗了多长时间, 每一次完整的子调用(比如: A调用B调用C, 其中B调用C的过程可以称为一个子调用过程)是由两天日志组成的, 一个是调用方产生的, 一个是提供方产生的. 如果我们将调用方产生的耗时减去提供方产生的耗时,则是一次子调用在网络中消耗的时间.

### 异常跟踪日志检索



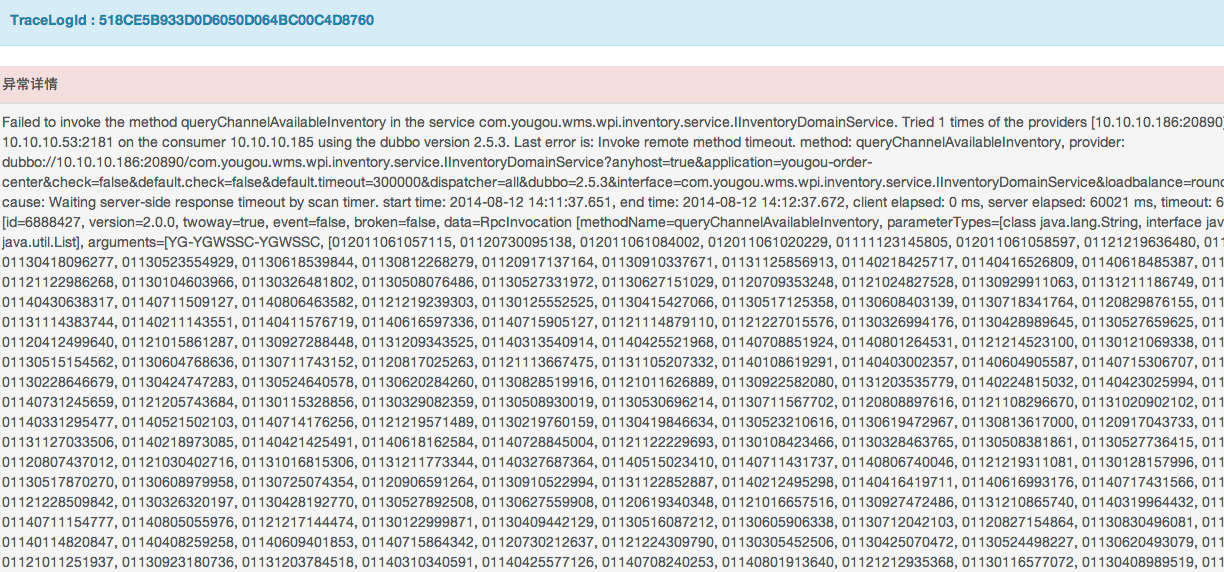
(图3-11)

这里主要是针对所有跟踪日志做了一次筛选, 默认进入查询出来调用耗时超过30秒的所有调用. 如果在调用过程中产生了**业务异常也需要进行归类**. 但凡产生了这样的异常,我们会根据预先设定的异常规则**给相应的人员发送报警邮件或短信**.如图3-11.

1. 查询所有包含异常信息的调用

可以通过选择查看异常调用来检索所有包含异常信息的调用.

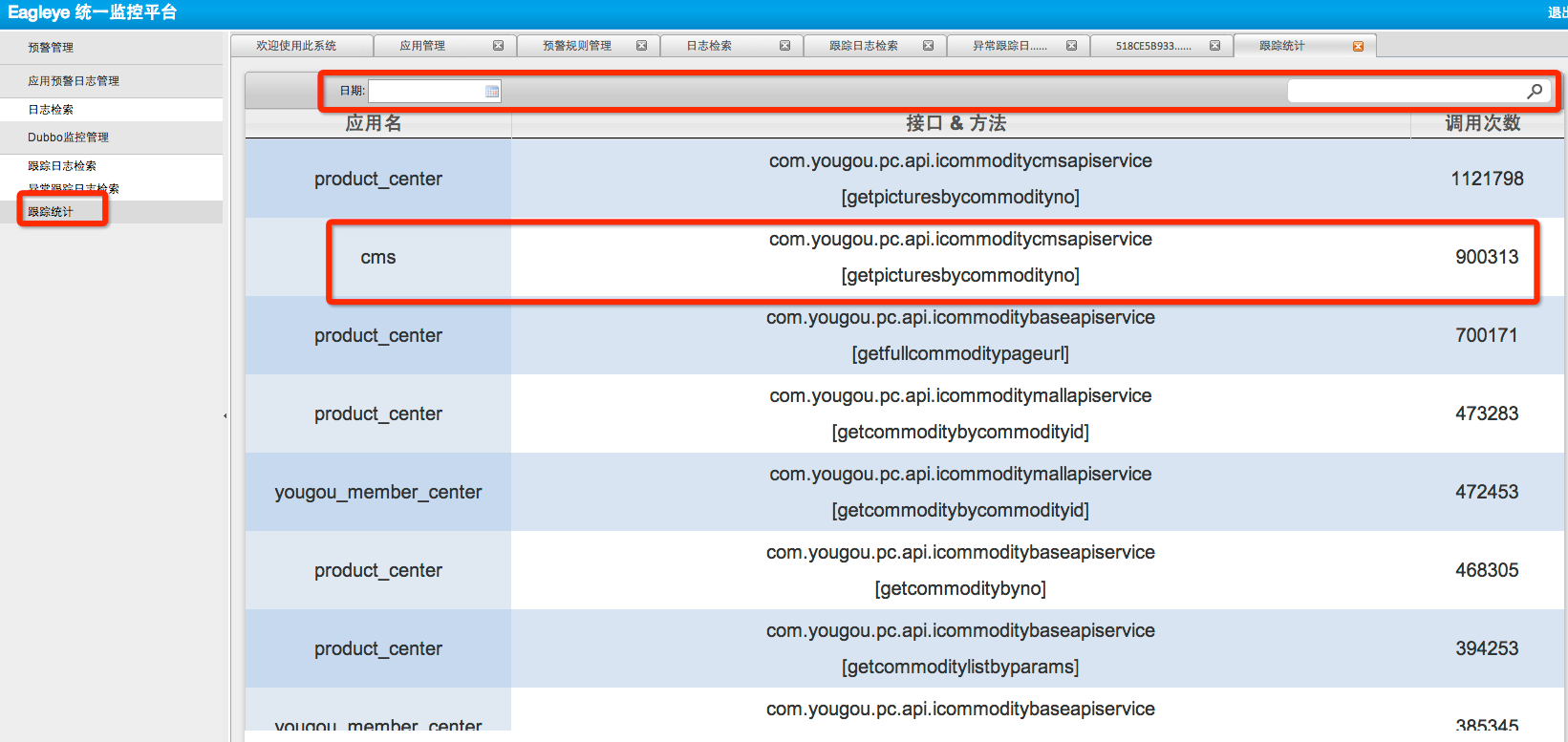
1. 查看异常详情



(图3-12)

点击查看详情可以进入到具体的异常详情页面, 如图3-12.

### 跟踪统计



(图3-13)

该统计功能主要是针对所有Dubbo接口被哪些应用调用了多少次的一个统计, 如果调用次数太多, 可能需要相关负责人进行关注. 我们可以统计七日之内某一天的调用情况,也可以根据某个关键字只统计包含这个关键字的Dubbo接口被各个应用调用了多少次.

## 异常日志预警

我们在上面设置的各个用户组, 用户组成员, 应用, 预警规则, 都是为了实现对各个应用系统的异常进行实时预警. 只要有新的预警信息被Eagleye收集到则会根据异常内容与预警规则进行匹配, 如果匹配成功, 则进行预警. 相同的异常针对某一个规则和用户组,在10分钟之内不会重复预警. 这样可以避免因为系统大量报出异常信息而大量发送预警的问题.

## Dubbo跟踪预警

Dubbo跟踪预警现在是每隔半个小时会对这半个小时之内产生的跟踪日志进行分析,如果跟踪耗时超过30秒, 或者跟踪日志有包含异常信息的则进行汇总. 然后统一发送给 [eagleye@yougou.com](mailto:eagleye@yougou.com)的邮件组, 该邮件组成员是各个Dubbo应用的相关负责人. 因为现在Dubbo预警需要一些统计操作, 所以暂时没有做成实时预警, 后续有时间,可以持续改进为实时预警.

# Eagleye概要设计

## 项目结构

(图4-1)

Eagleye统一监控平台项目是一个Java的Maven项目, 我们通过Maven将项目划分为如图4-1所示的几个子项目,下面针对每一个子项目进行详细的描述.

* eagleye-admin , eagleye管理控制台, 通过该项目包含对预警规则进行设置, 检索异常日志, 检索dubbo跟踪信息, 跟踪信息的统计等功能. 为了部署方便, 将收集的日志进行持久化也是有eagleye-admin来完成, 可以进行集群部署, 通过参数设置开启或关闭日志收集工作.
* eagleye-core , eagley 的核心依赖包, 里面存放了一些公关方法, 持久层的封装.
* eagleye-dubbo-client , 该项目会放置在各个应用端, eagleye通过该客户端进行dubbo调研日志的收集.
* eagleye-trace-domain , 针对RPC跟踪的通用模型, 不止可以应用在dubbo的跟踪, 可以应用在各种RPC跟踪场景. 该项目同时被eagleye-admin复用.
* eagleye-log4j-appender , 针对各个应用服务的异常信息进行收集, 不止局限于各个dubbo应用, 公司的各个应用都可以进行接入.

## 系统架构

eagleye-admin

eagleye-admin

Elasticsearch cluster

图4-2

各个应用在只需要Maven引入eagleye-dubbo-client, eagleye-log4j-appender 之后就可以收集相应的日志信息, 并通过redis的pub/sub方式传递给接收端的eagleye-admin, 然后通过多线程进行日志信息的处理,并最终持久化到elasticsearch中, 之后上亿级别的日志检索, 分析统计都是基于elasticsearch来展开. eagleye-admin 被赋予了日志收集持久化的能力是为了降低部署难度, 由于eagleye-admin即可以进行日志收集, 也可以对外提供eagleye的配置管理功能, 所以eagleye-admin内置了收集持久化开关, 可以达到某些节点提供检索, 配置,预警功能, 有点节点可以单一做持久化,日志收集工作.

根据图4-2中所描述的, redis要想做到高可用, 持续对外提供服务, 需要做HA方案, 目前生产场景还没有做HA方案(认为日志收集并不会影响具体的业务, 损失一些日志也是可以接受的). eagleye-admin也可以做成集群部署, 目前生产上是其中一台开启了日志收集, 持久化功能, 而预警, 检索, 统计, 配置管理等服务则是由另外一台服务器提供. Elasticsearch目前是两台服务器做集群, 每个服务器存储的数据是一样的. 每天大概的日志量级应该是上亿级别. 我们对日志的检索统计, 可以控制在秒级.

# Eagleye详细设计

本章节重点介绍一下RPC跟踪(我们的具体实现是Dubbo跟踪)设计的核心思路, 异常日志收集的思路, 以及eagleye-admin软件结构.

## Dubbo跟踪详细设计

### 设计前的考虑

我们在设计Dubbo跟踪的过程中考虑过使用spring-aop的方式或者是自定义annotation的方式进行跟踪日志的收集, 但是这样对系统的侵入性比较大. 我们如果在原有系统中嵌入跟踪会带来一些工作量, 这些工作量会随着嵌入的应用服务数量增多而增多. 这在有几十个应用和上千个服务接口的大优购来说, 工作量还是蛮大的.

如何才能对原有系统的侵入性降到最低, 目前来看只有通过Dubbo对外提供的SPI接口来打印日志, 然后通过对日志的收集来完成对Dubbo调用的跟踪, 预警. 在翻看了Dubbo的文档和部分源码之后, 认为实现Dubbo的Filter接口是比较靠谱的实现. 一个好的框架都会对外部提供很多可供拓展的SPI接口, 而责任链模式则是比较常用的拓展应用. 实现Dubbo的Filter可以从中获取到调用链的上下文信息, 当然,这些信息传递的原理也是通过ThreadLocal进行传递的. 在这里面进行跟踪日志的输出, 这样应用在嵌入跟踪时,只需要在pom.xml文件中引入eagleye-dubbo-client.jar即可,不会有任何代码的侵入. 即使日志发送失误也不会影响正常的业务. 在开发eagleye-dubbo-client子项目时,还需要考虑尽可能少的引入第三方jar包, 降低依赖包与Dubbo应用中的jar包冲突的可能性.

当然, 在设计领域模型时尽量避免对dubbo的依赖, 换句话说, 如果RPC调用换成了Thrift等其他方式, 只要可以输出的日志格式相同, 同样可以被收集跟踪.

### 参考资料介绍

eagleye最后定型为基于日志收集,分析的方式进行跟踪预警, 其中dubbo跟踪日志的生产原理和领域模型主要参考了Google的分布式系统跟踪(Dapper)论文, 淘宝的服务跟踪框架eagle-eye, 京东基于Dubbo的跟踪项目Hydra. 因为京东的Hydra也是基于Dubbo的跟踪项目, 我们最初是想直接将该项目应用到我们的生产环境, 但是在搭建环境的过程中, 出现了下面几个问题: 开源出来的部分已经是很久以前的了, 开源的版本当时在京东生产环境应用时也出现了很多问题(存在很多bug, 和作者联系确认过); 部署略显复杂; 没有统计功能. 基于以上几点, 我们还是决定参考多方面的资料自己实现一个简单, 不过度设计的跟踪客户端, 在开发过程中的一个基本原则是基于现有的生产环境场景, 不做过度的考虑和设计. 如果碰到功能上和性能上需要提升的地方, 再进行迭代优化. 下面基于dubbo跟踪的详细设计内容, 我会引入一些Dapper的论文和Hydra的设计文档, 毕竟他们是一个团队在做这个事情, 而大优购只有我一个人在做.

### Dubbo的关键领域模型

methodA-1

methodA-2

methodA-3

promotionA

methodB-1

methodB-2

methodB-3

commodityB

methodC-1

methodC-2

methodC-3

wpiC

Invoker

(图5-1)

在图5-1中, 描述了一个优购网从用户🡪促销, 促销🡪货品, 货品🡪wms的一个经典流程. 其中, promotionA可以作为一个Application, 一个Application会包含多个InterfaceService, 一个InterfaceService会包含多个Method, 而每一次调用的调用链信息都需要RpcContext进行传递. 下面会分别对着几个领域模型进行说明.

1. Application

在一个业务系统中的一系列服务, 比如: 货品系统对外提供服务的应用, 这个dubbo的application可以命名为:commodity-api, 其中包含了多个服务接口.

1. InterfaceService

接口服务, 一个接口服务可以提供多个业务处理方法.

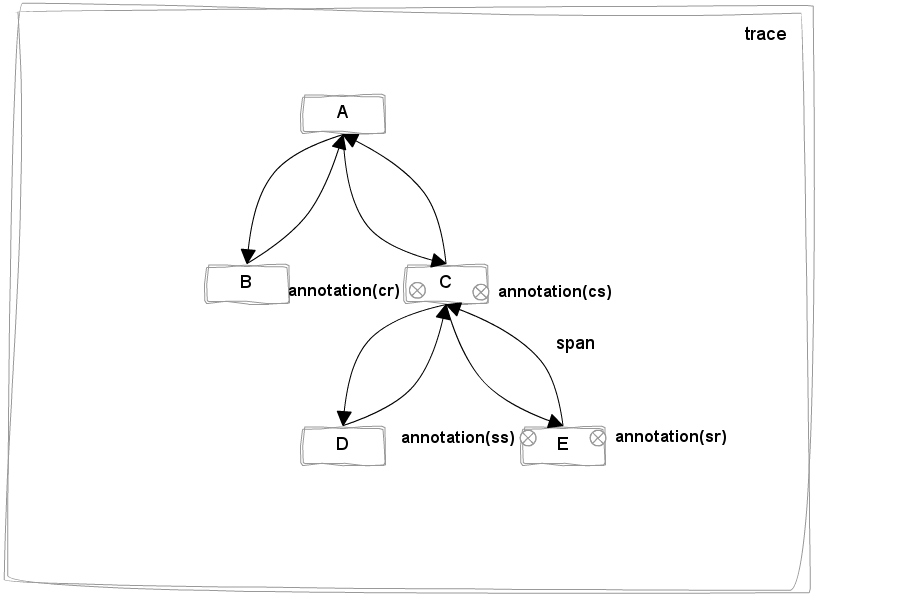
1. Method

接口服务中提供具体服务的方法.

1. RpcContext

Rpc调用的上下文对象, 我们可以通过该对象传递TraceId, 采样标识等重要信息.

### 跟踪领域模型



(图5-2)

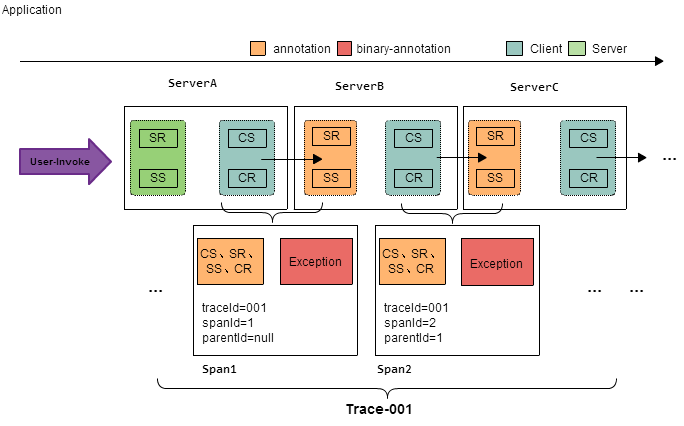
跟踪Client的主要领域模型如下:

1. Trace:一次服务调用追踪链路.
2. Span:追踪服务调基本结构，多span形成树形结构组合成一次Trace追踪记录.
3. Annotation:在span中的标注点，记录整个span时间段内发生的事件.
4. BinaryAnnotation:属于Annotation一种类型和普通Annotation区别，这键值对形式标注在span中发生的事件，和一些其他相关的信息.

Annotation在整个跟踪数据模型中最灵活的，灵活运用annotation基本能表达你所想到的跟踪场景.在跟踪模型中定义4种不同value的annotation用来表达记录span 4个最基本的事件.通过这4个annotation能计算出链路中业务消耗和网络消耗时间.

如图5-2所示的应用场景, 对A服务的调用.A服务在被调用的过程中会继续调用服务B和服务C,而服务C被调用之后又会继续调用服务D和服务E. 在我们的领域模型中，服务A被调用到调用完成的过程，就是一次trace.而每一个服务被调用并返回的过程（一去一回的箭头）为一个span.可以看到这个示例中包含5个span，client-A，A-B，A-C，C-D，C-E. span本身以树形结构展开，A-C是C-D和C-E的父span，而client-A是整个树形结构的root span.之后要提到的一个概念就是annotation，annotation代表在服务调用过程中发生的一些我们感兴趣的事情，如图所示C-E上标出来的那四个点，就是四个annotation，来记录事件时间戳，分别是C服务的cs（client send），E服务的ss（server receive）,E服务的ss（server send）, C服务的cr（client receive）.如果有一些自定义的annotation我们会把它作为BinaryAnnotation，其实就是一个k-v对，记录任何跟踪系统想记录的信息，比如服务调用中的异常信息，重要的业务信息等等.

跟踪模型和dubbo模型之间关系, 如图5-3:



(图5-3)

### 主要类图

暂时看项目代码

## 异常预警详细设计

暂时略, 以后补充

# 部署

## 配置文件详细说明

整个eagleye项目主要分为两类配置文件, 详细介绍如下:

1. eagleye-admin的配置文件

该配置文件在部署时的路径为服务器的/etc/yougouconf/eagleye-admin/config.properties.

主要项的解释如下:

#mysql 这是以前基于dbcp的数据库连接配置, 现在已经废弃

eagleye.admin.username=root

eagleye.admin.password=root

eagleye.admin.driverClassName=com.mysql.jdbc.Driver

eagleye.admin.hibernate.dialect=org.hibernate.dialect.MySQLDialect

eagleye.admin.url=jdbc\:mysql\://192.168.211.248\:3306/eagleye\_admin\_test?useUnicode\=true&characterEncoding\=utf-8

eagleye.admin.maxActive=100

eagleye.admin.maxIdle=10

eagleye.admin.maxWait=3600000

eagleye.admin.hibernate.show\_sql=true

eagleye.admin.hibernate.use\_sql\_comments=true

eagleye.admin.hibernate.default\_batch\_fetch\_size=16

######C3P0 MySQL config 这里是基于c3p0的数据库配置#######

c3p0.url=jdbc\:mysql\://192.168.211.248\:3306/eagleye\_admin\_test?useUnicode\=true&characterEncoding\=utf-8

c3p0.user=root

c3p0.password=root

c3p0.driverClass=com.mysql.jdbc.Driver

c3p0.acquireIncrement=1

c3p0.maxIdleTime=60

c3p0.maxPoolSize=200

c3p0.minPoolSize=50

c3p0.initialPoolSize=60

#elasticsearch 这里是我们的elasticsearch的集群配置,如果是多个集群用逗号分隔

#eagleye.admin.es.address=10.10.10.220:9300

eagleye.admin.es.address=192.168.211.250:9300

eagleye.admin.es.clustername=eagleye\_elasticsearch

# trace redis 这里是收集跟踪日志的redis配置, 考虑将来把跟踪和异常收集分开, 所以他们分别有不同的redis配置, 现在配置项都是一个.

eagleye.admin.trace.redis.pool.maxActive=300

eagleye.admin.trace.redis.pool.maxIdle=50

eagleye.admin.trace.redis.pool.maxWait=1000

eagleye.admin.trace.redis.pool.testOnBorrow=false

#eagleye.admin.trace.redis.ip=10.10.10.220

eagleye.admin.trace.redis.ip=192.168.211.247

eagleye.admin.trace.redis.port=6379

# alert redis 这是专门收集异常日志并预警使用的redis

eagleye.admin.alert.redis.pool.maxActive=300

eagleye.admin.alert.redis.pool.maxIdle=50

eagleye.admin.alert.redis.pool.maxWait=1000

eagleye.admin.alert.redis.pool.testOnBorrow=false

#eagleye.admin.redis.ip=10.10.10.220

eagleye.admin.alert.redis.ip=192.168.211.247

eagleye.admin.alert.redis.port=6379

#send email and sms interface 这是我们发送预警邮件和短信使用的地址

eagleye.admin.sendEmailUrl=http://10.10.10.30:8849/messenger/sendEmailForSystem.do

eagleye.admin.sendSmsUrl=http://10.10.10.30:8849/messenger/sendSms.do

#alert frequence default 10 min, 这是我们需要对重复异常进行预警的时间间隔, 因为异常预警是实时进行了,为了避免同一条异常在单位时间内多次预警

eagleye.admin.alert.frequence=1200

# open or close alert, 是否开启异常日志预警

eagleye.admin.alert.status=false

# trace alert frequence default 600 second, dubbo跟踪预警,该配置暂时没有用, 在将来做成实时以后可以使用, 目前只是用quartz进行定时预警,预警间隔为30分钟.

eagleye.admin.trace.alert.frequence=600

# open or close trace alert, 是否开启dubbo跟踪预警

eagleye.admin.trace.alert.status=false

# open or close storage, 是否开启dubbo跟踪日志收集, 为了方便部署, 我们的跟踪信息收集, 持久化也是由eagleye-admin项目进行的, 如果关闭该配置, 那么eagleye-admin只提供检索统计服务

eagleye.admin.storageStatus=true

1. eagleye-client的配置文件

该配置文件在生产上的放置路径为: /etc/yougouconf/eagleye-client/config.properties

主要配置说明如下:

#alert redis 异常预警配置,这里主要是redis的配置

eagleye.client.alert.redis.pool.maxActive=300

eagleye.client.alert.redis.pool.maxIdle=50

eagleye.client.alert.redis.pool.maxWait=1000

eagleye.client.alert.redis.pool.testOnBorrow=false

eagleye.client.alert.redis.pool.timeout=2000

eagleye.client.alert.redis.retryNum=0

eagleye.client.alert.redis.ip=192.168.211.247

eagleye.client.alert.redis.port=6379

#是否开启异常日志收集

eagleye.client.alert.isOpen=false

#trace redis dubbo跟踪信息收集配置

eagleye.client.trace.redis.pool.maxActive=300

eagleye.client.trace.redis.pool.maxIdle=50

eagleye.client.trace.redis.pool.maxWait=1000

eagleye.client.trace.redis.pool.testOnBorrow=false

eagleye.client.trace.redis.pool.timeout=2000

eagleye.client.trace.redis.retryNum=0

eagleye.client.trace.redis.ip=192.168.211.247

eagleye.client.trace.redis.port=6379

#是否开启dubbo跟踪日志收集

eagleye.client.trace.isOpen=false

**因为现在eagleye的client还处在测试阶段, 如果发现生产环境因为eagleye的client引起了业务异常, 请将配置中的isOpen关闭, 然后重启服务, 不过到目前在测试,生产试运行的快一个月时间, 还没有发现对业务的影响.**

## 测试环境说明

目前测试环境只部署在了250这台服务器, 上面有elasticsearch, tomcat. Redis我们用的是测试环境247的服务. Elasticsearch在之前测试时还和212的机器做了集群测试, 不过将来打算升级新的版本(jdk升级到1.7), 升级后212则不提供参与集群(上面还有其他业务服务, 用的是jdk1.6), 所有软件的部署路径都是/usr/local/下.

## 生产环境说明

生产环境主要有220, 20这两台服务器, 每台服务器都装有elasticsearch, tomcat. 现在20主要只进行日志的收集和持久化, 220主要提供检索,统计和预警功能. 我们的redis使用的是220机器提供的服务, 端口6379. 我们可以通过 <http://10.10.10.220:9200/_plugin/bigdesk> 去查看elasticsearch的集群情况. 通过<http://10.10.10.220:8080/eagleye-admin> 登录生产的eagleye-admin控制台, 管理员权限为: administrator/ administrator, 客户权限: guest/guest. 因为将来要对接公司的权限系统, 所以登录临时写死在程序中.

我们的elasticsearch安装路径为: /usr/local/elasticsearch-1.1.1

tomcat 安装路径为: /usr/local/tomcat…..

redis安装路径为: /usr/local/redis

## eagleye部署

部署分为两个部分, eagleye-admin的部署和eagleye-dubbo-client部署,eagleye-log4j-appender部署.

eagleye-admin部署主要是从gitlab中获取编译打包, 可以直接部署在tomcat中, 不需要任何改动. 需要将之前说的eagleye-admin的配置文件放在约定的地方, 并修改相应的配置.

eagleye-dubbo-client的部署需要用户应用的pom.xml文件中引入如下配置, 并将指定的配置文件放在在指定的位置(上面已经描述):

<dependency>

<groupId>com.yougou</groupId>

<artifactId>eagleye-dubbo-client</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

eagleye-log4j-appender的部署需要用户的pom.xml文件引入如下配置:

<dependency>

<groupId>com.yougou</groupId>

<artifactId>eagleye-log4j-appender</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

然后需要修改log4j配置,如果是properties格式的,做出修改如下:

log4j.appender.eagleye=com.yougou.eagleye.client.log4jappender.EagleyeAppender

    log4j.appender.eagleye.appGroup=cms

    log4j.appender.eagleye.layout=org.apache.log4j.PatternLayout

    不用忘记在rootLogger中添加eagleye的appender

    log4j.rootLogger=INFO,console,file,eagleye

如果是xml格式的做出修改如下:

**其中以cms为例, appGroup为所在应用的简称, 统一用小写字母**

<appender name="agleye" class="com.yougou.eagleye.client.log4jappender.EagleyeAppender">

        <param name="appGroup" value="**cms**" /> **<!— 这里的名称请按照上面的对号入座, eagleye会按照这个进行约定处理 -->**

         <layout class="org.apache.log4j.PatternLayout" />

 </appender>

在 root中加入 eagleye 的appender, 这步一定别忘记做

<root>

        <level value="INFO"/>

        <appender-ref ref="console"/>

        <appender-ref ref="rollingFile”/>

<appender-ref ref="eagleye”/>

    </root>

# 遇到的坑

## 曾经已解决的坑

最开始我们采用的日志收集方案在应用端部署flume进行收集, flume还是比较灵活的, 但是随着应用越来越多, 需要在服务器上开启多个flume实例用来监听需要收集的日志文件, 这样的部署是非常恐怖的, 如果需要修改什么配置,或者升级flume版本, 那么所有服务器都需要进行变动. 后来为了降低对应用服务的侵入, 我们还是采用了pub/sub的方式, 首选redis做这项工作. 在做压力测试是, 单台redis可以达到50000/s的处理效果. 由于日志收集,我们所要求的可靠性不是第一位的, 即使丢失那个几个日志对跟踪预警的影响也不会很大.

## 曾经到现在未解决的坑

我们在eagleye-admin中操作elasticsearch时,选用了开源的spring-data-elasticsearch, 在使用的过程中,有它的便利性, 但是也存在一些未修复的bug, 其中最坑的是不能对数字进行排序, 如果需要排序走的是字符串排序(9大于100), 还有其中的Or检索不起作用.只有and操作好使. 最后考虑升级新的版本, 但是新版本用的是jdk1.7, 这些又悲催了, 我们在将来还是打算将elasticsearch升级到1.2版本, 将spring-data-elasticsearch升级到最新版本.

在做elasticsearch的统计时,或者频繁检索时, 会出现存储数据异常, 查看相关文档, 说是jdk1.6版本可能会有这个问题 “Failed to deserialize exception response from stream”, 现在还不确定这个问题的所在, 还需要持续跟进. 毕竟接触elasticsearch的时间还很短.