ELK实践

2019年4月

詹玉林

ELK实践

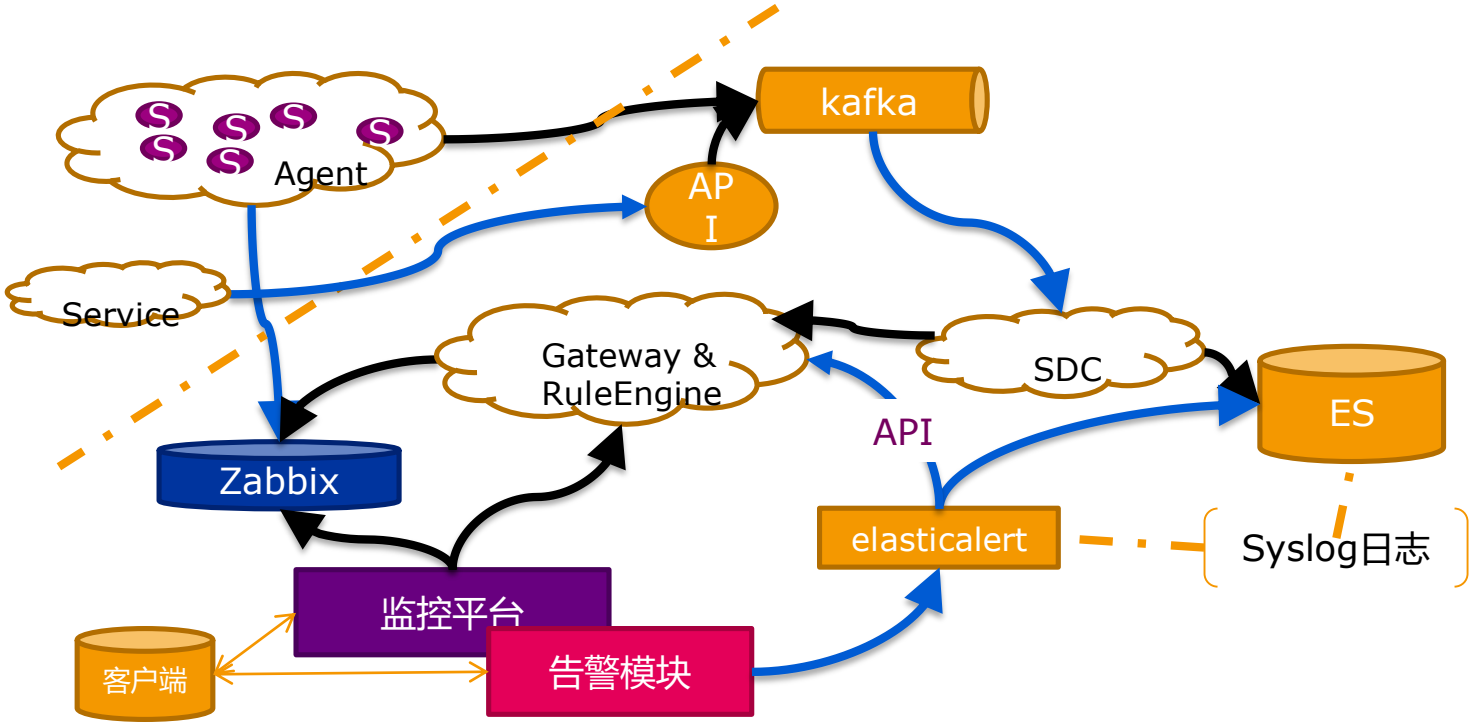
- ELK+应用场景
- 集群自动化实践
- Flume工作

应用场景

Part 1

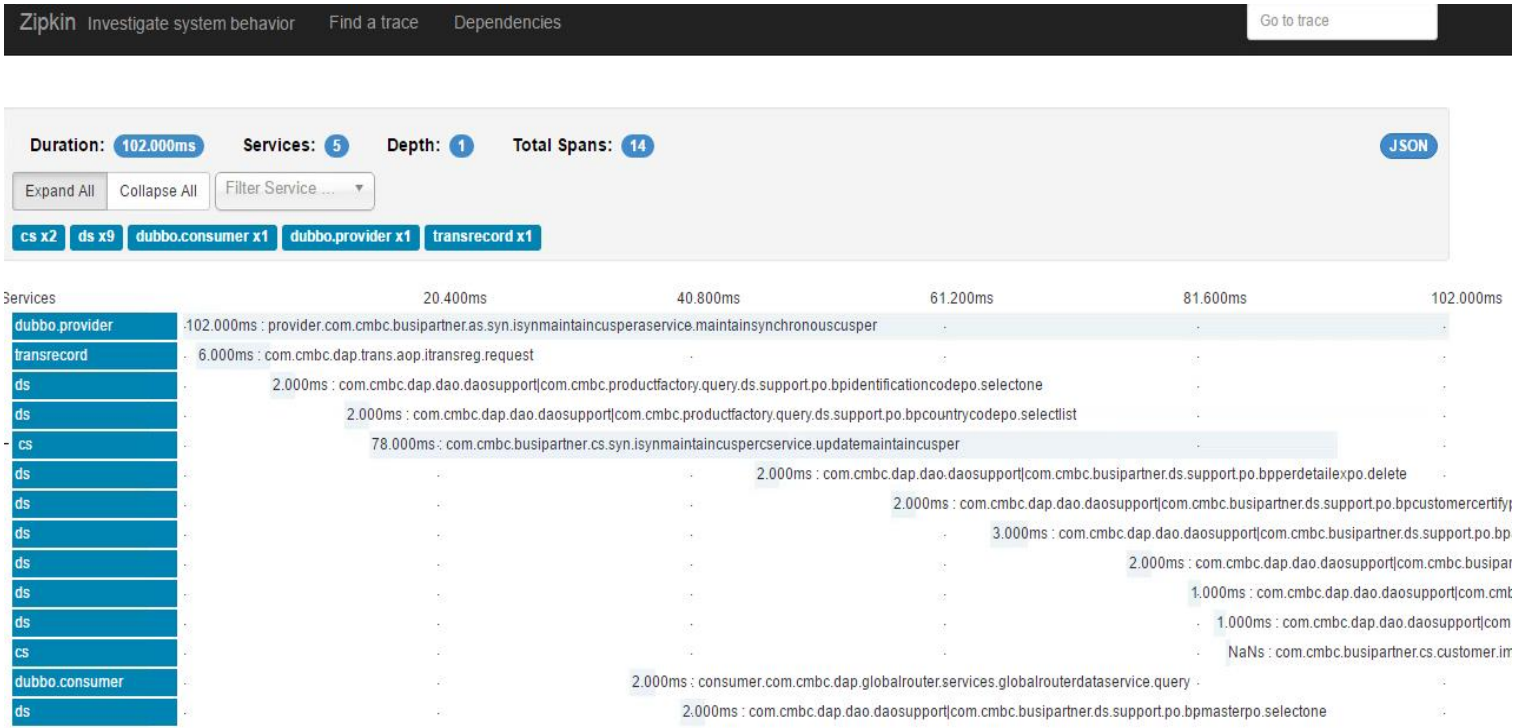
成果及应用——日志平台应用场景1：日志监控告警，解决运维痛点

日志监控：实现日志复杂条件内容检测告警



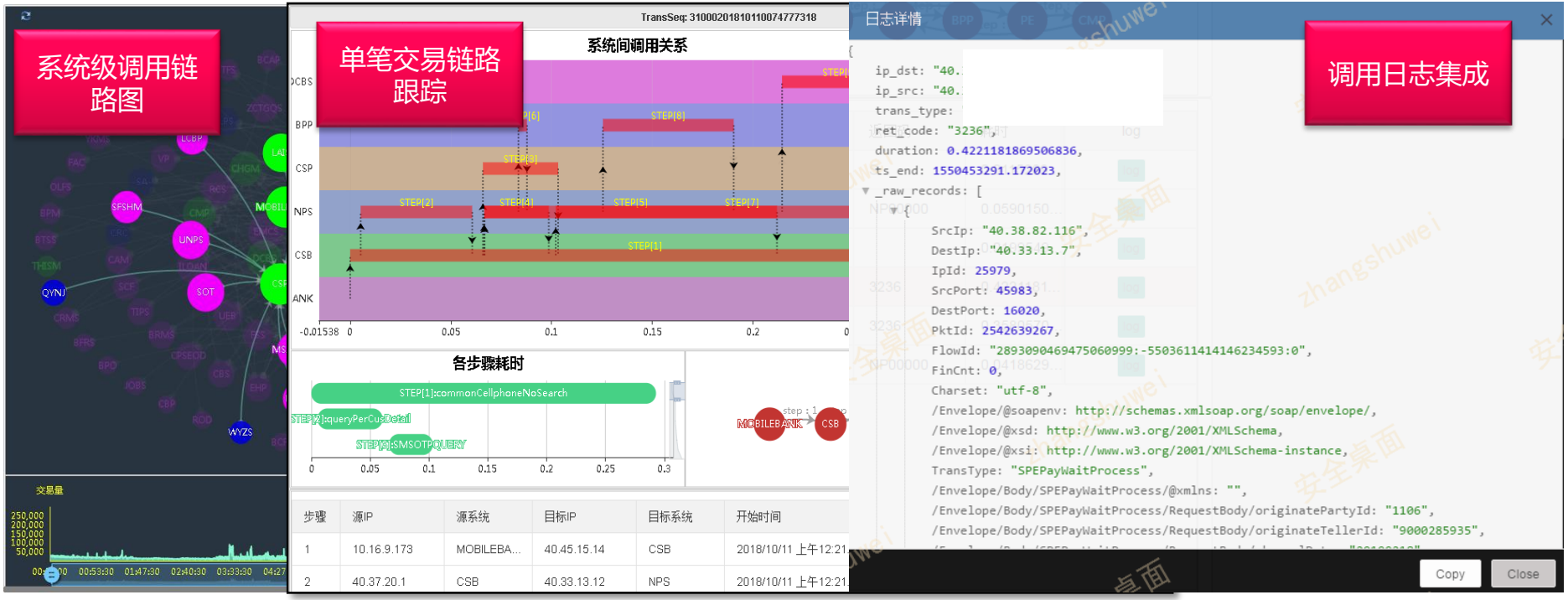
成果及应用——日志平台应用场景2：日志平台服务，平台能力输出

- 基于平台提供的日志收集和存储能力，分布式核心系统采用ZipKin组件实现了服务交易链路监控(APM)



成果及应用——日志平台应用场景3：日志平台服务，平台能力输出

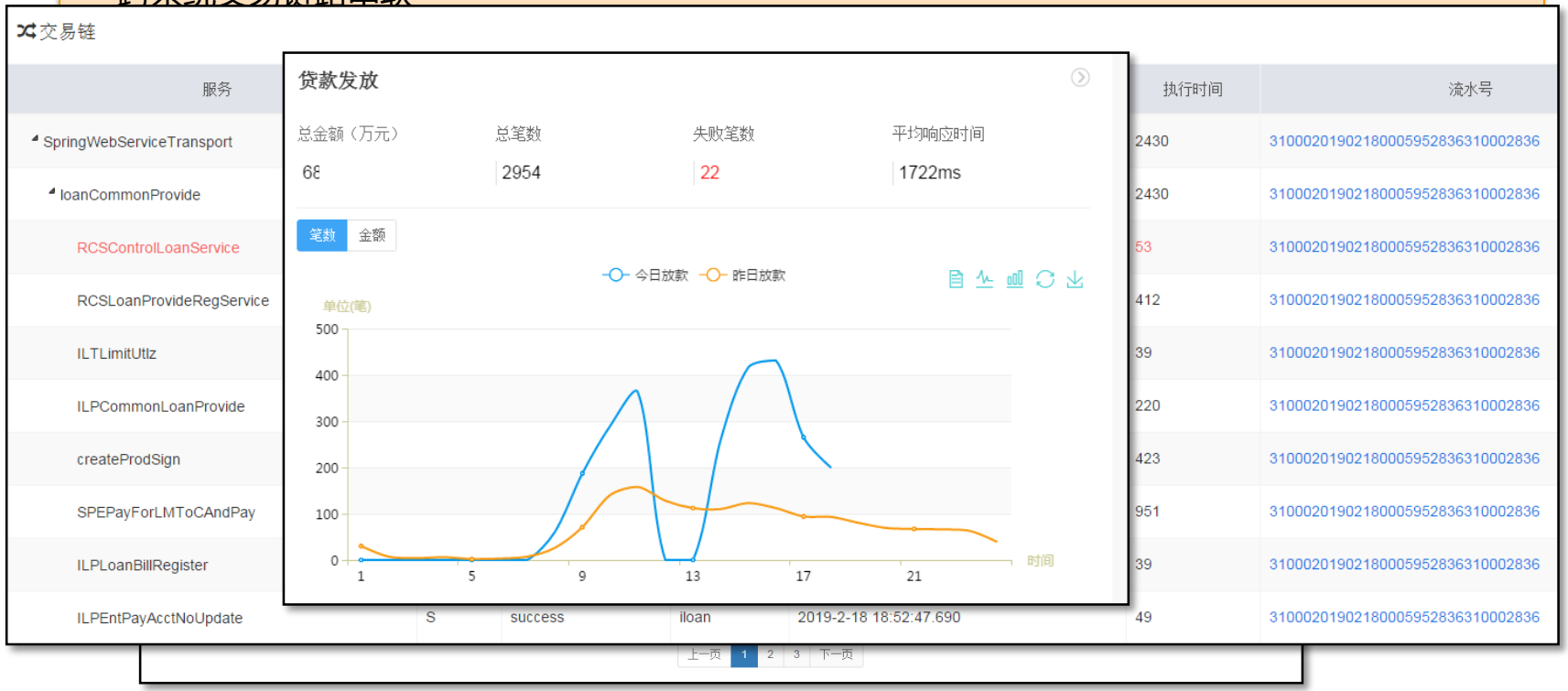
OnPlat全景运维视图系统，基于平台的日志能力，实现了系统链路交易量和入访交易量拓补图展现，并可以随时给予交易流水号完成上下游系统的交易查询，为运维人员了解系统和查询问题提供了便利



成果及应用——日志平台应用场景4：日志平台服务，平台能力输出

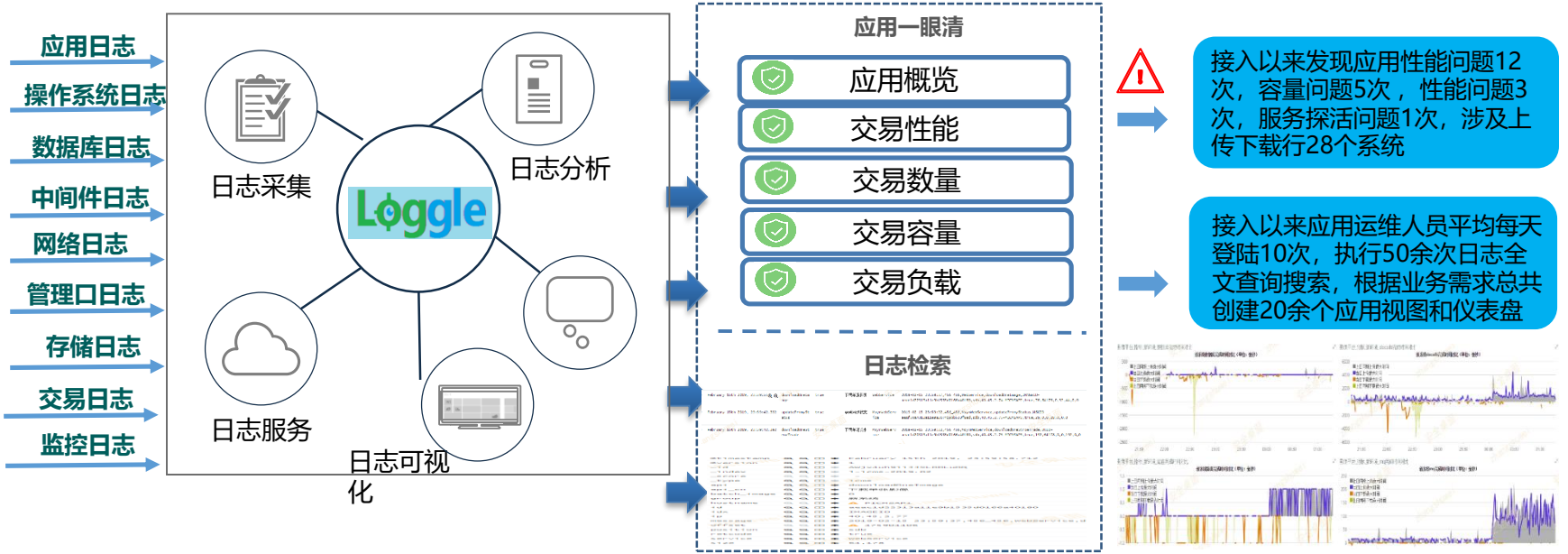
基于日志平台，新零售信贷建设了一体化全景监控平台，通过日志旁路的方式，无需应用埋点，在对应用零侵入的情况下，实现了系统监控和业务数据两方面的场景，取得了良好的应用前景

- 跨系统交易链路串联



成果及应用——日志平台典型使用案例

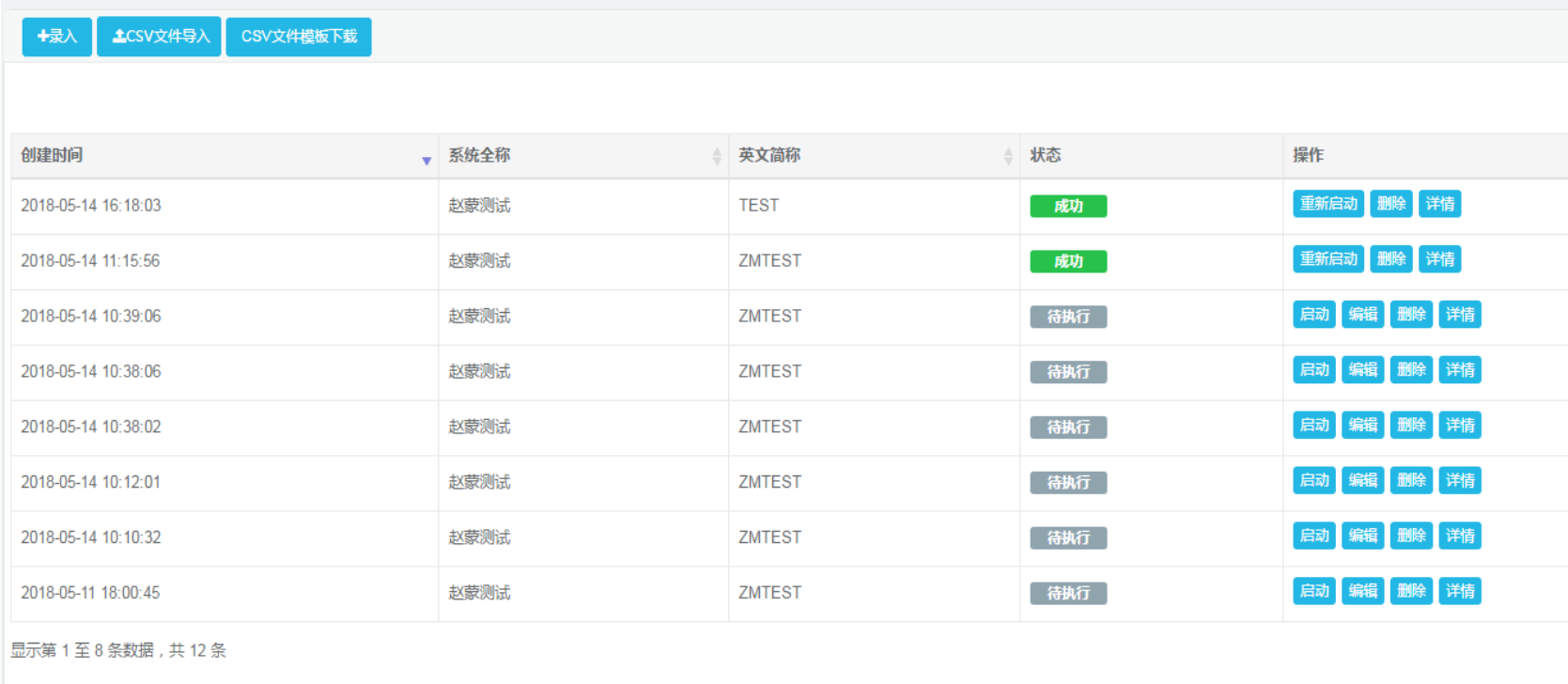
为全行业务系统提供影像存取服务的影像平台于2017年4月接入天眼实时智能日志管理分析平台，根据业务需求对日志进行了标准化并针对主要业务指标进行了解析和抽取，基于应用一眼清形成了一套完整的包含包含业务交易、操作系统、中间件、数据库等日志统一信息的监控运营框架，应用人员每天定时登陆平台进行相关视图监控，并在系统故障时协助定位，日志平台成为其系统运维工作中的重要工具之一。



成果及应用——日志平台典型使用案例

PaaS平台ELK Agent的自动化部署模块

完成Agent自动化部署平台的开发工作，实现了批量倒入、流程编排、步骤内嵌、断点重提、过程展现等一系列配套功能，需要使用PaaS平台ssh互信推送脚本执行、抓取文件、推送文件、CSV文件读取等技术。



成果及应用——日志平台典型使用案例

PaaS平台ELK Agent管控模块

丰富Agent管理场景，提供一键化管控日志平台Agent功能，需要使用Storm流式处理、PaaS平台ssh互信推送脚本执行、抓取文件、推送文件等技术

退库 更新 清理 日志分析 日志下载 配置文件更新 启动 强启 停止 强杀

10秒后自动刷新 10s 停止 刷新

显示 10 条数据

IP	所属应用	进程数量	进程号	OS类型	OS版本	设备类型	主机名	VM/PM	CPU使用率(单核)	内存使用量(M)	内存使用率	文件系统使用量	状态
197.3.84.202	agentinstall	1	14935	SLES	11.4	x86_64	BIGL1TMP	pm	0	12.94	0	29	正常
197.3.64.249	oracle	1	60555660	AIX	6.1	PowerPC_POWER7	AMLDPPD	lpar	0	0	0	138.37	正常
197.3.176.211	zmtest	1	37355562	AIX	7.1	PowerPC_POWER7	NAPS2PSQ	lpar	0	0	0	138.3	正常
197.3.176.139	cmp	1	59899932	AIX	7.1	PowerPC_POWER7	CMP2PSQ	lpar	0	0	0	571.29	已停止
197.3.16.5	kafka	1	29188	SLES	12.2	x86_64	BIGLGTMP	pm	0	9.53	0	47	已停止
197.3.16.4	kafka	1	80837	SLES	12.2	x86_64	BIGLFTMP	pm	0	24.38	0	91	已停止
197.3.16.3	kafka	1	160579	SLES	12.2	x86_64	bigletmp	pm	0	21.33	0	97	正常
197.3.16.2	kafka	1	58992	SLES	12.2	x86_64	bigldtmp	pm	0	19.98	0	94	正常
197.3.16.1	log	1	192663	SLES	12.2	x86_64	biglctmp	pm	0	26.89	0	97	已停止
197.3.153.60	weblogic	1	24276	SLES	12.2	x86_64	ZBXSrv1	pm	0	10.84	0.1	94	正常

集群自动化

Part 2

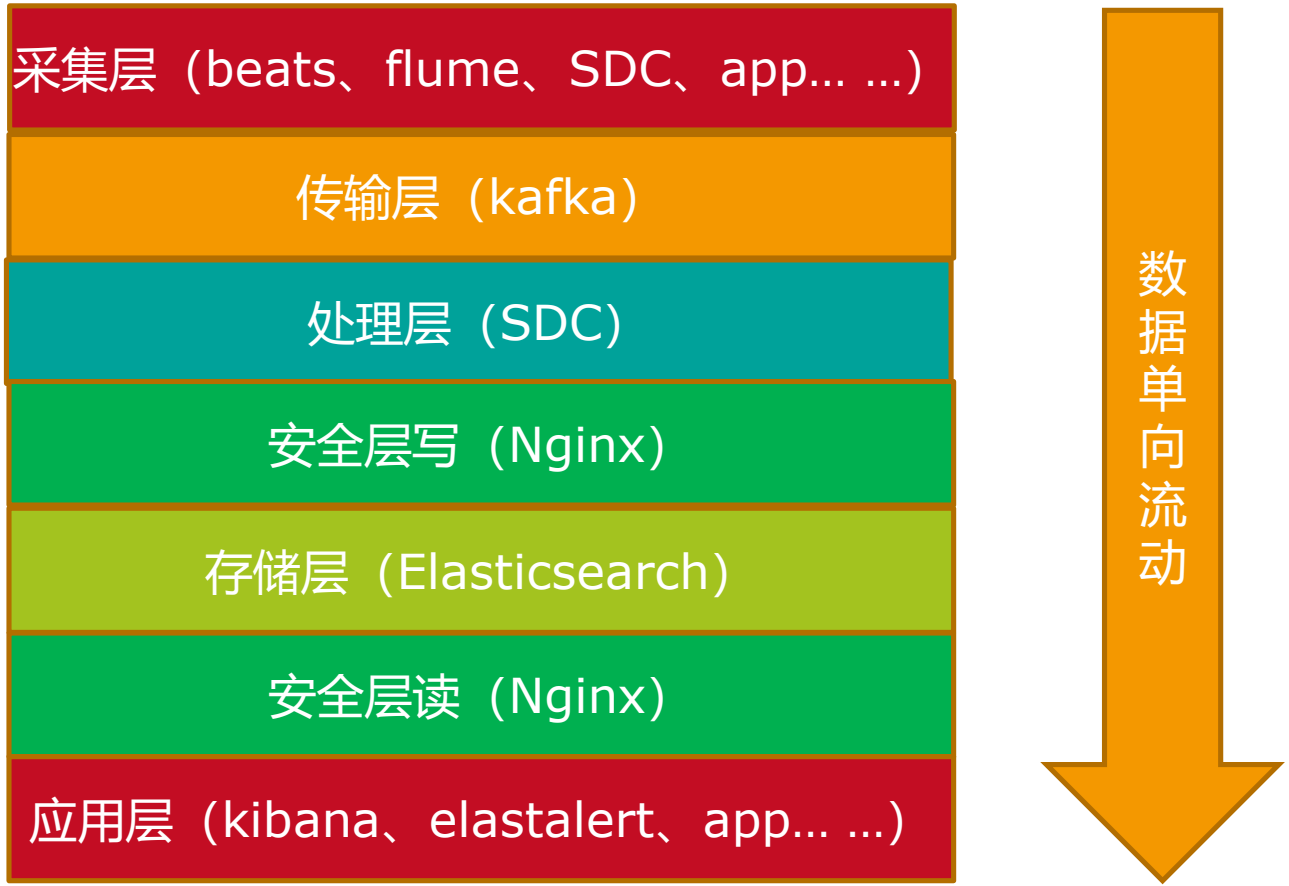
生产环境的挑战

- 分布式集群维护困难：搭建、集群节点间配置同步、日常维护（节点启停、服务启停、状态查看）
- 升级风险大：升级过程中、升级过程后、数据量大、持续时间长、影响范围大、业务影响大
- 故障定位复杂：大量服务状态需要检查、日志信息四散分布
- 故障恢复代价高：重新搭建故障节点或模块、重新恢复耗时、费力

理想的目标

- 稳定：集群内不存在单点、数据保持完全同步（数据、配置、程序等）
- 高效：自动化部署、升级、恢复，快速故障定位（状态查看、日志定位）
- 安全：避免各个节点人工修改风险，验证后再发布到集群其他节点（灰度发布）
- 简单：不依赖于过多的外部资源，使用成熟工具避免引入外部故障
- 灵活：适用于多种分布式、非分布式软件，易于扩展，易于与其他产品结合

框架结构-数据处理流程

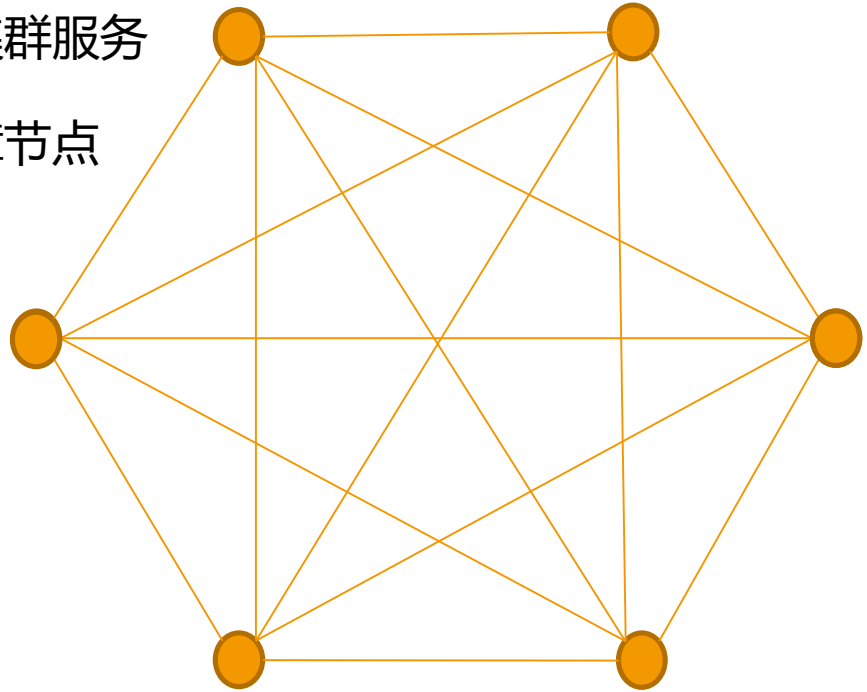

采集层（beats、flume、SDC、app… …）
传输层（kafka）
处理层（SDC）
安全层写（Nginx）
存储层（Elasticsearch）
安全层读（Nginx）
应用层（kibana、elastalert、app… …）
数据单向流动

框架结构-服务节点间关系

对等节点、Decentralized design

少数任意节点故障不会影响集群服务

从剩余任一节点快速恢复故障节点

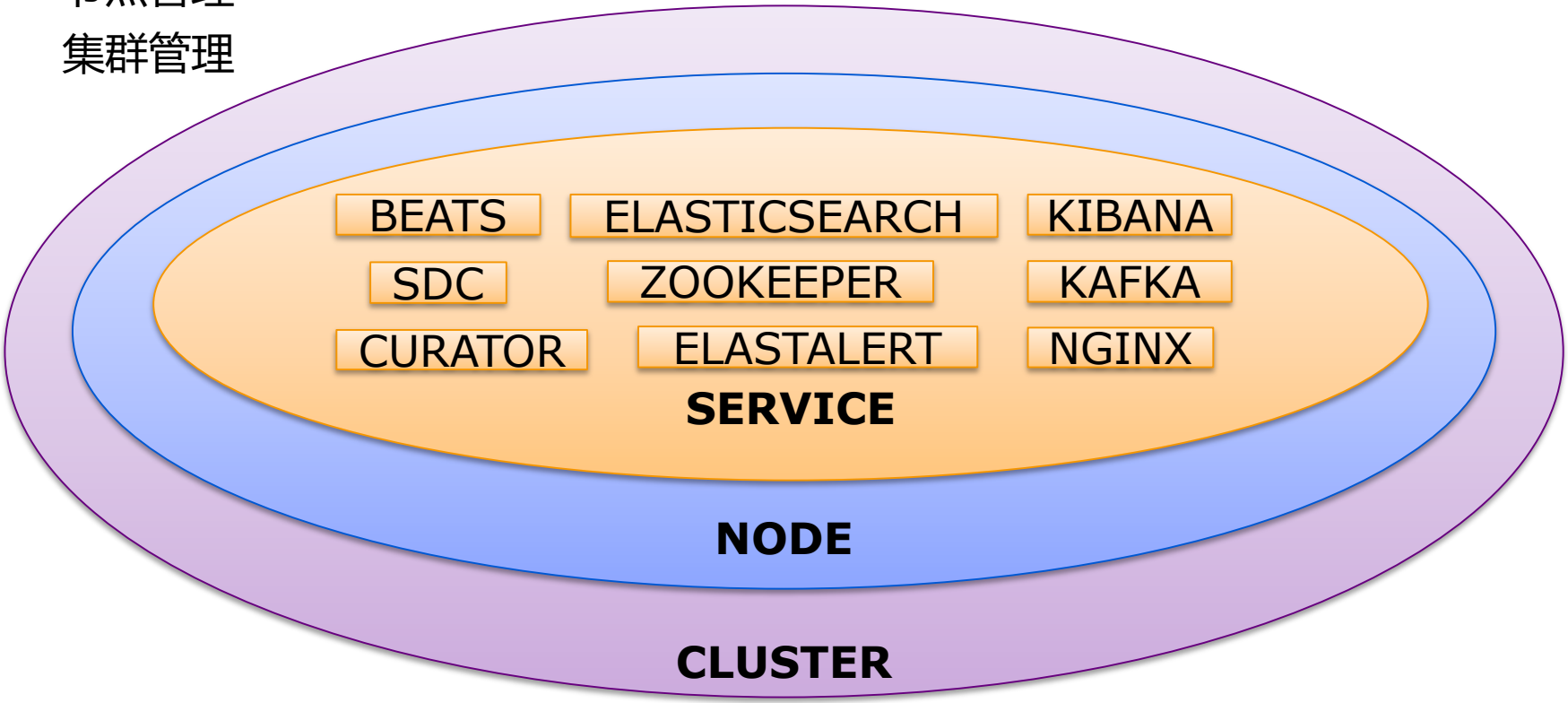


框架结构-服务管理关系

服务管理

节点管理

集群管理



实现原理

- 本地服务拆分（程序、配置、日志、运行数据等），避免升级导致的覆盖，同时规范数据存储位置易于信息查找
- 本地服务管理（启动、停止、状态），脚本进行封装
- 节点服务管理（启动、停止、状态）调用本地服务管理，脚本进行封装，supervisor
- 集群服务管理（启动、停止、状态）远程调用本地服务管理，脚本进行封装，pssh
- 数据同步（rsync），所有节点数据一致，节点间关系都是对等的，在服务启动时再确定正确的运行环境（广义上的一种docker）

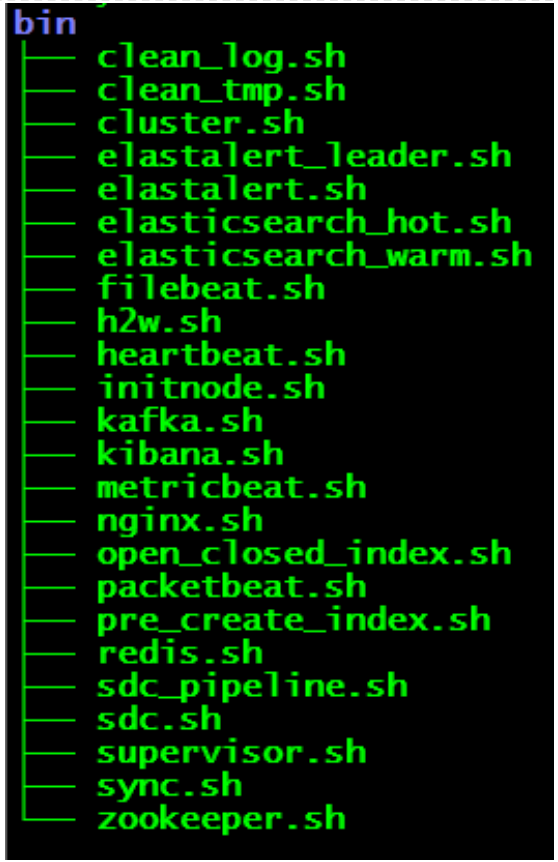
目录结构

- 命令（bin）：存放在整个集群环境运行所需的各种命令脚本
- 配置(conf)：各服务、组件的配置文件；按服务、组件分子目录存放，事先准备好；该目录下的*.list文件定义了哪些节点启动哪些服务以及运行角色
- 数据(data)：各服务、组件的中间运行态数据
- 日志(log)：各服务、组件的运行日志信息
- 安装(install)：各服务、组件的标准安装包，从各产品的官方网站下载
- 程序(software)：各服务、组件的运行程序，由各安装包安装后产生

```
.
├── bin
├── conf
│   ├── elastalert
│   ├── elastalert-server
│   ├── elasticsearch
│   ├── filebeat
│   ├── heartbeat
│   ├── kafka
│   ├── kibana
│   ├── metricbeat
│   ├── nginx
│   ├── packetbeat
│   ├── redis
│   ├── sdc
│   └── supervisor
├── data
│   ├── curator
│   ├── elasticsearch
│   ├── filebeat
│   ├── heartbeat
│   ├── kafka
│   ├── kibana
│   ├── metricbeat
│   ├── nginx
│   ├── sdc
│   ├── supervisor
│   └── zookeeper
├── install
│   ├── openresty-1.13.6.2
│   ├── openssl-1.1.1
│   └── pcre-8.42
├── log
│   ├── curator
│   ├── elasticsearch
│   ├── filebeat
│   ├── heartbeat
│   ├── kafka
│   ├── kibana
│   ├── metricbeat
│   ├── nginx
│   ├── sdc
│   ├── supervisor
│   └── zookeeper
└── software
    ├── curator
    ├── elastalert
    ├── elastalert-server
    ├── elasticsearch
    ├── filebeat
    ├── heartbeat
    ├── jdk
    ├── kafka
    ├── kibana
    ├── metricbeat
    ├── nginx
    ├── packetbeat
    ├── redis
    └── streamsets-datacollector
```

重要命令

- initnode.sh：初始安装或升级安装一个节点上所有服务、组件
- 各服务脚本：服务层面控制，负责自身服务的启停、状态监控
- supervisor.sh：节点层面的服务控制，对节点范围内对服务进行启停、状态监控，节点服务退出自动拉起
- cluster.sh：集群层面的服务控制，在整个集群范围内对服务进行启停、状态监控
- sync.sh：将当前节点的数据完全同步到集群中的其他节点，保证节点间信息完全同步。用于扩容、升级、信息同步等场景



操作流程

- 选取集群中任意节点为首节点
- 操作场景：
  - 部署场景：执行initnode.sh对该节点进行安装
  - 升级场景：执行initnode.sh对该节点进行升级
  - 变更场景：不执行initnode.sh，仅更改配置
  - 扩容场景：不执行initnode.sh
- 完成上一步处理后，执行sync.sh，采用轮转的方式将首节点的信息完全同步到集群中其他节点
- **通过initnode.sh+sync.sh就可以实现所有服务的自动化部署、升级、变更、扩容等工作**

架构优势

- 实现去中心化结构，无单点
- 节点间数据完全同步，具体运行态依赖于环境配置
- 可以由任何节点的数据自动生成其他节点
- 任何修改在小范围内通过验证后才同步到整个集群
- 结构简单易维护
- 该架构适用于多种软件、服务，易于快速扩展

高可用架构

基于前面的部署架构可以很容易地扩展为多集群的高可用架构

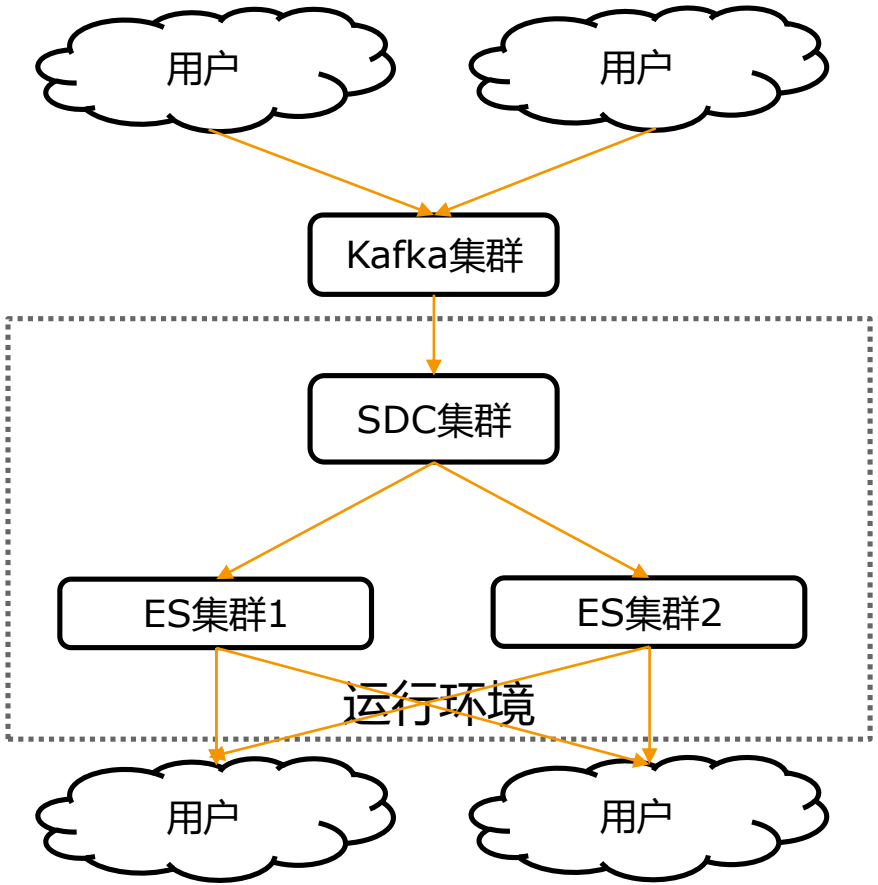
通过该架构，可提供：

1、集群间的高可用；

2、故障后自动恢复；

3、无风险的版本升级；

4、跨版本升级；



后续计划

- 微服务化
- Docker/Kubernete化

开源项目

- **详情参考：https://github.com/zhan-yl/es_cluster**

Flume工作

Part 3

TaildirSource使用中遇到的坑

- 用正则表达式匹配文件，却无法获取文件完整路径（FLUME-2955）
- 不支持多行（FLUME-2960）
- 不支持目录正则匹配（FLUME-2961）
- 无法同时保留主机名和IP（FLUME-3181）
- 文件句柄未释放（FLUME-3255）

坑一：无法获取文件完整路径

- **问题描述**
- 在/app/logs中存在a.log，b.log，c.log三个文件，为了采集这些后缀为log的日志文件，使用正则表达式配置filegroup，在Flume1.6中，发现无法获取文件的完整路径，导致日志数据入到 Elasticsearch 后，用户从 Kibana 从查询时无法定位到数据所在的日志文件路径。

```
1 agent.sources.s1.type = org.apache.flume.source.taildir.TaildirSource
2 agent.sources.s1.filegroups = f1
3 agent.sources.s1.filegroups.f1 = /app/logs/.*.log
```

- **解决方案（FLUME-2955代码已被社区Merge，并发布在1.7中）**
- 增加 fileHeader 和 fileHeaderKey 两个参数，两个参数含义分别是：

参数	默认值	含义
fileHeader	false	是否在event的header里包含文件的绝对路径
fileHeaderKey	file	指定header里fileHeader的key值

坑二：不支持多行

- **问题描述**
- Taildir Source按行读取日志，把每一行的内容放入event的body中，无法合并多行内容，Java应用程序输出等典型日志无法处理。
- **解决方案（FLUME-2961代码在review，本行生产环境已使用两年）**
- 设计一个 buffer缓存多行内容，仿照 Logstash 的 codec/mulitline 插件配置，设计参数：

参数	默认值	含义
multiline	false	是否使用多行合并
multilinePattern	\n	指定正则表达式，匹配指定的正则表达式来确定是前一个event的内容还是下一个event的内容
multilinePatternBelong	next	值可选"next"或"previous"。若multilineMatched=true,"previous" 是指匹配正则表达式的行是前一event的内容，"next"是指匹配正则表达式的行是下一event的内容；若multilineMatched=false，"previous"是指不匹配正则表达式的行是前一event的内容，"next"是指不匹配正则表达式的行是后一event的内容
multilineMatched	true	是否匹配正则表达式
multilineEventTimeoutSeconds	0	缓冲区buffer event的超时时间，0表示永远不超时
multilineMaxBytes	10485760	单位字节数，默认值是10MB，若缓冲的内容长度超过该参数，则缓冲区中的数据作为一个event被flush输出
multilineMaxLines	500	行数，默认值是500行，若缓冲的内容行数超过该参数，则缓冲区中的数据作为一个event被flush输出

坑三：不支持目录正则匹配

- **问题描述**
- 在实际应用写日志时，很多系统是根据日期生成日期目录，每个日期目录中包含一个或多个日志文件，类似： /app/logs/20170101/、/app/logs/20170102/、/app/logs/20170103，但是Taildir Source只支持文件名带正则，不能支持目录带正则。
- **解决方案（FLUME-2960代码在review，本行生产环境已使用两年）**
- 增加 filegroups.\.parentDir 和 filegroups.\.filePattern 两个参数，两个参数含义分别是

参数	含义
filegroups.<filegroupName>.parentDir	filegroup的目录，不能包含通配符或正则表达式
filegroups.<filegroupName>.filePattern	相对于parentDir的路径，filePattern可包含目录，可使用正则

坑四：无法同时保留主机名和IP

- **问题描述**
- 为了获取Flume agent所在机器的主机名或IP，我们使用了主机名拦截器(Host Interceptor)，但是根据主机名拦截器的定义，只能保留主机名和IP中的一种，无法同时保留主机名和IP。
- **解决方案（FLUME-3187代码在review，本行生产环境已使用两年）**
- 将原来的useIP参数扩展，增加一个参数useHostname，若同时设置为true，可同时保留主机名和IP；另外支持自定义主机名和IP地址在event header里的key，参数如下：

参数	默认值	含义
useIP	true	是否保留IP地址
useHostname	true	是否保留主机名
ip	ip	指定event header里IP地址的key值
hostname	hostname	指定event header里主机名的key值

坑五：文件句柄未释放

- **问题描述**
- 在部分生产环境中，某些日志文件会被归档到其他目录，且一轮处理后仍然处于打开状态，若此时遇到下一轮处理，该文件因为不再匹配，所以仍然是打开状态，随着时间累计这样的文件句柄会越来越多，导致无法删除这些文件，或者是超过文件描述符的最大限制。
- **解决方案（FLUME-3255，本行生产环境已使用近一年）**
- 在taildirsource的每次处理之前，将前一次处理过的文件中，不再匹配且处于open状态的文件关闭。

Flume Issue总结

IssueID	相关组件	简要介绍	JIRA地址
FLUME-2955	Taildir Source	根据参数fileHeader的值可将文件的完整路径放在event的头部	https://issues.apache.org/jira/browse/FLUME-2955
FLUME-2960	Taildir Source	filegroup的目录中支持正则表达式	https://issues.apache.org/jira/browse/FLUME-2960
FLUME-2961	Taildir Source	实现多行合并处理日志	https://issues.apache.org/jira/browse/FLUME-2961
FLUME-3187	Host Interceptor	支持同时保留主机名和IP	https://issues.apache.org/jira/browse/FLUME-3187
FLUME-3255	Taildir Source	关闭不再匹配的文件句柄	https://issues.apache.org/jira/browse/FLUME-3255

FLUME-2960/2961/3187/3255四个Patch合并到Flume 1.7上，欢迎大家下载使用，Github地址：https://github.com/tinawenqiao/flume，分支名trunk-cmbc。

我行生产环境Flume TaildirSource配置实例

```
agent.sources.r1.type = org.apache.flume.source.taildir.TaildirSource
agent.sources.r1.filegroups = f1 f2
agent.sources.r1.filegroups.f1.parentDir = /weblogic/olfsdoms/1213/olfsdoms/logs
agent.sources.r1.filegroups.f1.filePattern = /\\weblogic_.*.log
agent.sources.r1.headers.f1.type = olfs

agent.sources.r1.filegroups.f2.parentDir = /weblogic/olfsdoms/1213/olfsdoms/logs
agent.sources.r1.filegroups.f2.filePattern = /\\weblogic.*.log
agent.sources.r1.headers.f2.type = olfs

agent.sources.r1.positionFile = /home/logger/flumeAgent/apache-flume-1.7.0-bin/conf/olfs_r1_tailPosition.json
agent.sources.r1.skipToEnd = true

agent.sources.r1.multiline = true
agent.sources.r1.multilinePattern = ^.*\\d{4}-\\d{2}-\\d{2}\\s\\d{2}:\\d{2}:\\d{2}.\\d{3}
agent.sources.r1.multilinePatternBelong = previous
agent.sources.r1.multilineMatched = false
agent.sources.r1.multilineEventTimeoutSeconds = 120
agent.sources.r1.multilineMaxBytes = 50457280
agent.sources.r1.multilineMaxLines = 30000
agent.sources.r1.cachePatternMatching = false
agent.sources.r1.fileHeader = true
agent.sources.r1.fileHeaderKey = path

agent.sources.r1.interceptors = i1
agent.sources.r1.interceptors.i1.type = host
```

谢谢

elastic
中文社区

专业、垂直、纯粹的 Elastic 开源技术交流社区
https://elasticsearch.cn/