LOGSTASH+ELASTICSEARCH+KIBANA- 实现相对通用的数据收集分析

ElasticSearch + Logstash + Kibana, 目前应该算是一套完整的日志收集/存储/统计解决方案.

在上一篇 Logstash+ElasticSearch+Kibana处理nginx访问日志 中, 介绍了如何统一处理`nginx`日志

最近正好在做应用日志及上报日志的汇聚和统计工作, 分享下处理方式.

目标是: 一次搭建, 后面只需要关心输入(日志记录)以及输出(Kibana统计展示)

日志

日志的来源:

1. 服务日志: 服务端记录下来的日志, 例如搜索日志等, 内容较为详尽
2. 上报日志: 来自于前端/android/ios/桌面端等, 根据用户操作行为, 上报一些数据, 例如按钮点击量, 转化率等, 也可以上报崩溃日志

其中, 上报日志, 可以制定一套协议, 不同端统一走上报服务接口. 这个服务对性能有要求, 具体协议需要足够灵活, 支持各类统计分析需求. 使用golang写了一个 http_json_logger

日志在不同应用/机器记录后, 可以通过`rsync`/`nfs`/`scp`等, 汇总到一个地方进行统一处理, 也可以通过多个`logstash shipper`进行汇聚

关于日志格式, 统一使用`json`格式, 落地过程中包含平台`{platform}`, 以及 项目`{project}`, 模块`{module}`, 落地时间`ts`等

为什么要落地: 落地成文件, 定时压缩备份存档, 不论日志处理系统是否有问题, 都能保证数据已经存下来了.

当然, 也可以考虑使用logstash监听端口, 分别落地到文件及转入es, 没具体实践过.

在这一步, 我的处理方案是:

1. 使用统一上报接口, 日志落地到上报服务的数据盘
2. 服务端服务日志, 同一台机器在相同数据盘, 使用同一个logstash shipper进行汇聚

对日志文件名等, 不强求一致性, 你可以认为, 不同项目/不同模块的json都可以直接记录到同一个日志文件(虽然不鼓励这么做), 通过日志body内容而不是日志文件名来处理

汇聚: logstash shipper

不得不说, logstash的确是神器

上一步, 日志中强制日志中每一行是一条json记录. 同时json body中记录了时间戳(timestamp, `ts`字段)

这一步, 配置 `logstash`将某些目录下的所有日志文件进行汇聚

配置示例:

```
input {
  file {
    path => [ "/data/collect/ios/*.log", "/data/collect/android/*.log", "/data/collect/web/*.log",
"/data/collect/wap/*.log" ]
    start_position => "beginning"
    codec => json
  }
}
```

```
# make ts to @timestamp
filter {
  date {
    match => [ "ts" , "dd/MMM/YYYY:HH:mm:ss Z", "UNIX" ]
  }
}


output {
    redis { host => "127.0.0.1" data_type => "list" key => "logstash:collect:log" }
}
```

这时候, 所有日志集中到了一个地方

处理并存储到**es**

首先, 从redis中读取消息体, 检查并丢弃一些信息, 然后, 根据消息体内`platform`/ `project`/`module`,

分配到es不同的`index`

可以根据需要控制粒度

配置示例

```
input {
  redis {
    host => "127.0.0.1"
    port => "6379"
    key => "logstash:collect:log"
    data_type => "list"
    codec  => "json"
    type => "logstash-collect-log"
    tags => ["collect"]
  }
}


# drop invalid record
filter {
    if ![platform] {
        drop {}
    }
    if ![project] {
        drop {}
    }
    if ![module] {
```

```
        drop {}
    }
}

output {
    elasticsearch {
      host => "127.0.0.1"
      index => "%{platform}-%{project}-%{module}-%{+YYYY.MM.dd}"
    }
}
```

ES

你会发现, es已经的index中已经有了具体的数据. json中的每个字段都有.....

然后呢?

整套调通之后, 接下来的工作呢?

假设来了个统计需求

1. 分析需求, 拆解, 确定统计维度, 需要上报的字段等, 根据协议, 确定`{platform}/{project}/{module}`
2. 前端/客户端等, 根据协议, 调用上报接口, 执行数据上报
3. 到`kibana`, 找到对应`index`, 根据需求配置对应的展现

所有的一般性统计需求, 都可以通过**三板斧**直接搞定, 只需处理输入以及输出, 没有任何额外工作.

最后

目前量不大, 完美解决了快速迭代中各类原先处理起来十分困难的统计需求和日志分析(原来要自己上报汇聚数据, 自己拷贝到同一台机器, 自己写统计脚本, 存库, 还得自己撸前端, 搞完之后还被黑说: **花了那么多时间, 只搞出个这么反人类的/丑/不是我想要的.....界面**)

当然, 随着业务发展, 各类日志的量都会逐渐上来, 对性能/存储的要求会越来越高, 但是`elk`本身对横向扩容只是非常完美, 在相当长一段时间内, 应该可以`hold`住.(老大, 我要加机器/硬盘/内存/CPU)

ok, 先这些

wklken

2015-05-08 于深圳

如果我的文章对你有所帮助，可以扫码进行小额捐赠 **or** 分享给更多人。多谢支持**:)**

上一篇: k-vim 更新9.0版本

下一篇: 分享一份 Vim 简介PPT