WOT
World Of Tech

2016年8月26日-27日　北京珠三角JW万豪酒店

World Of Tech 2016
移动互联网技术峰会
THE BEST MOBILE TECH IN HERE

主办：51CTO

日志漫谈
不同规模下的日志运维与优化

于炳哲

纲要

小企业日志运维

大企业日志运维

手机微博日志系统架构及相关调优

结语&反思

小企业在不同规模下日志架构

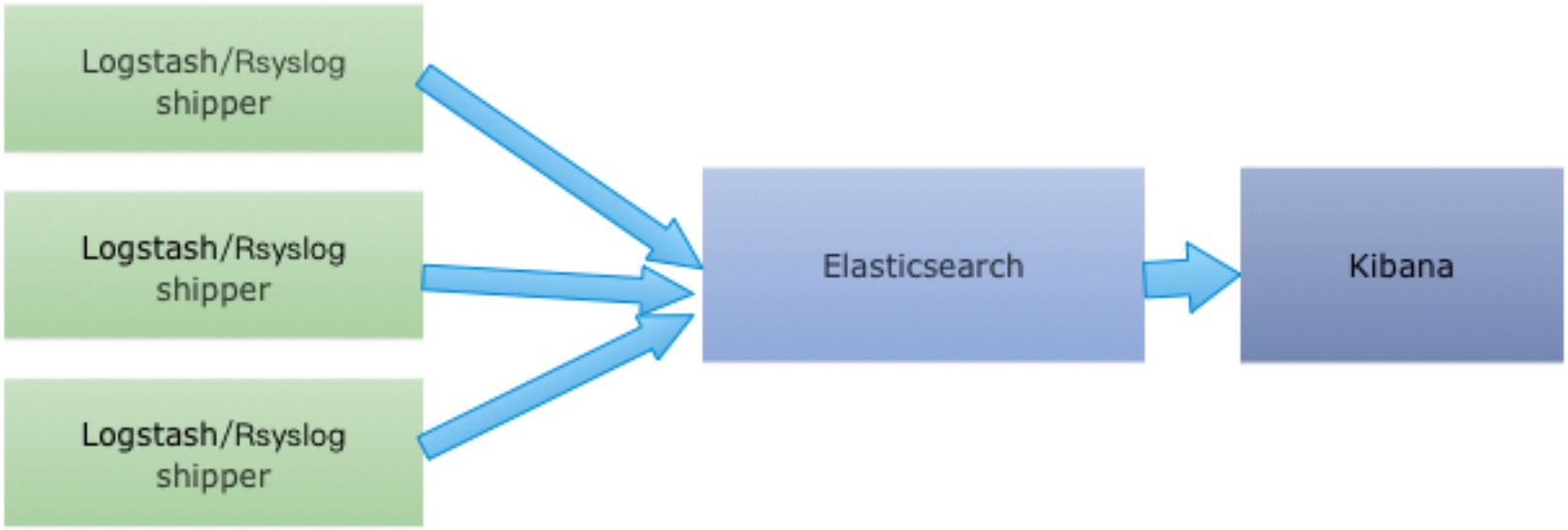
开荒阶段

- 纯文本 + grep + awk + sed
- 传统数据库 重点数据直接写入数据库
- 使用第三方的监控

数据量增长阶段

- 架构分布式，需要分布式收集
- 业务可能已经上线，需要实时反馈运行情况
- 已经有专门的运维人员，或者有偏向运维的开发人员

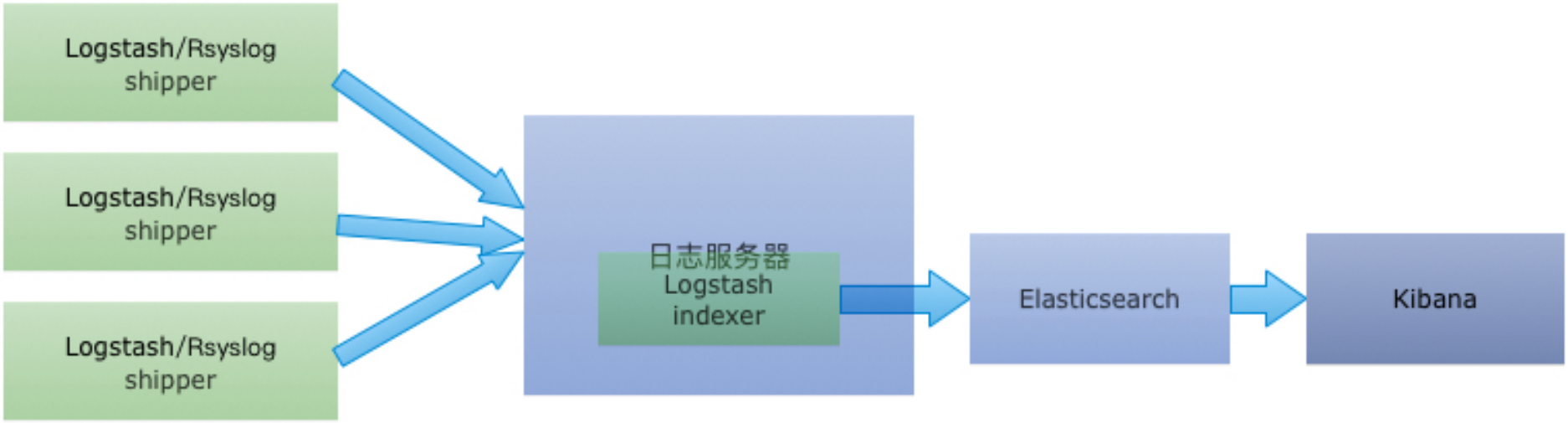
分布式架构，需要分布式收集



日志中转阶段

- 对日志数据的安全性不是有非常高的要求。
- 考虑单点问题。
- 需要对数据进行归档保存。

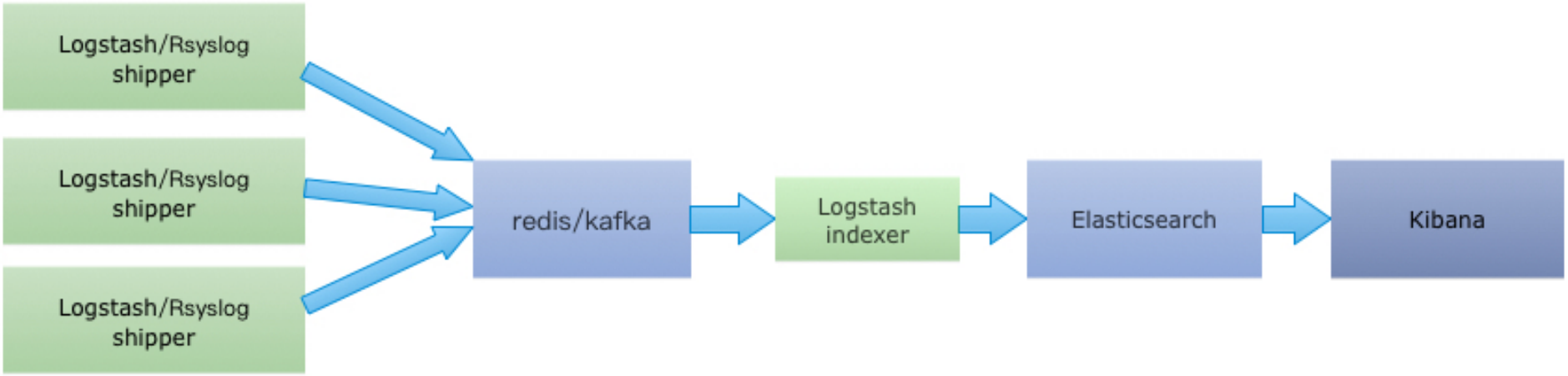
日志中转阶段


Logstash/Rsyslog
shipper
Logstash/Rsyslog
shipper
Logstash/Rsyslog
shipper
日志服务器
Logstash
indexer
Elasticsearch
Kibana

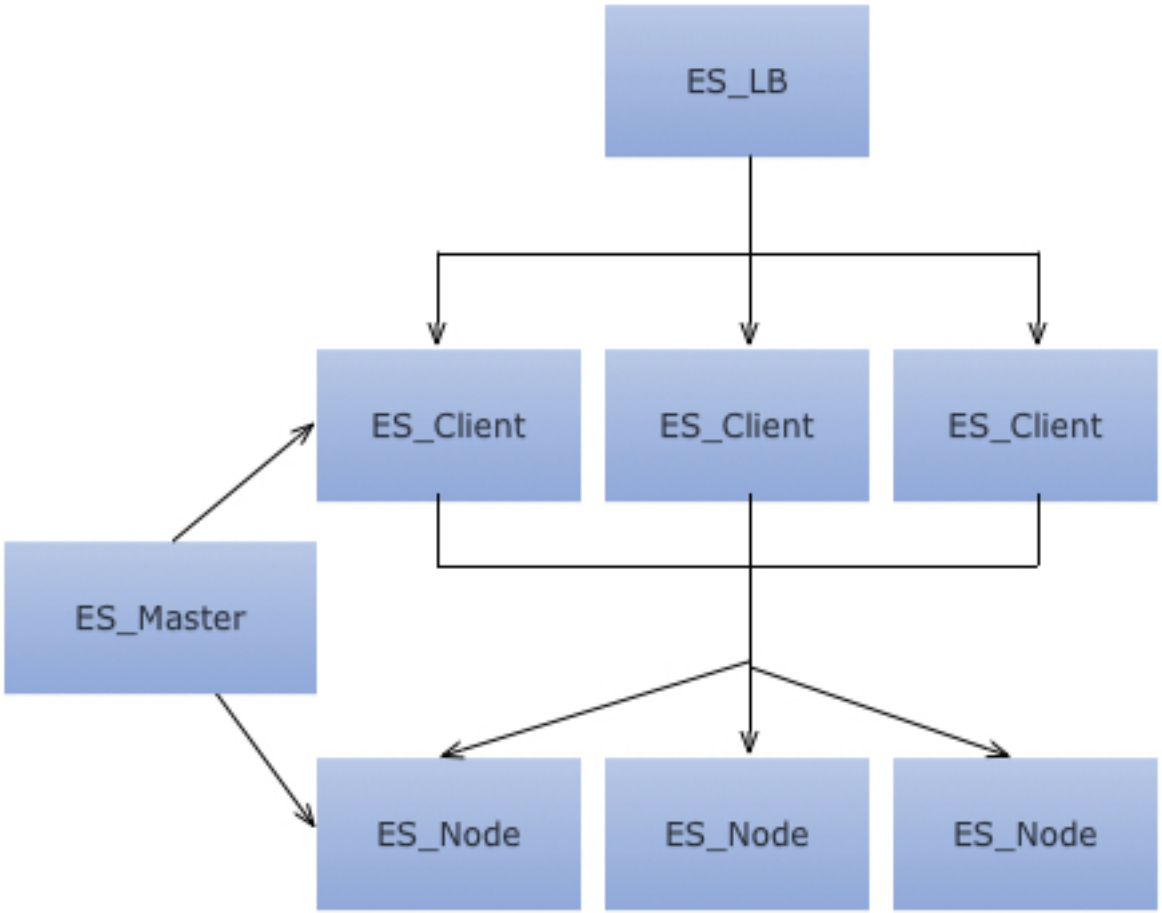
队列阶段

- 日志需要统一归档。
- 要考虑数据安全问题。
- 需要一写多读。

日志服务器磁盘写入成为瓶颈



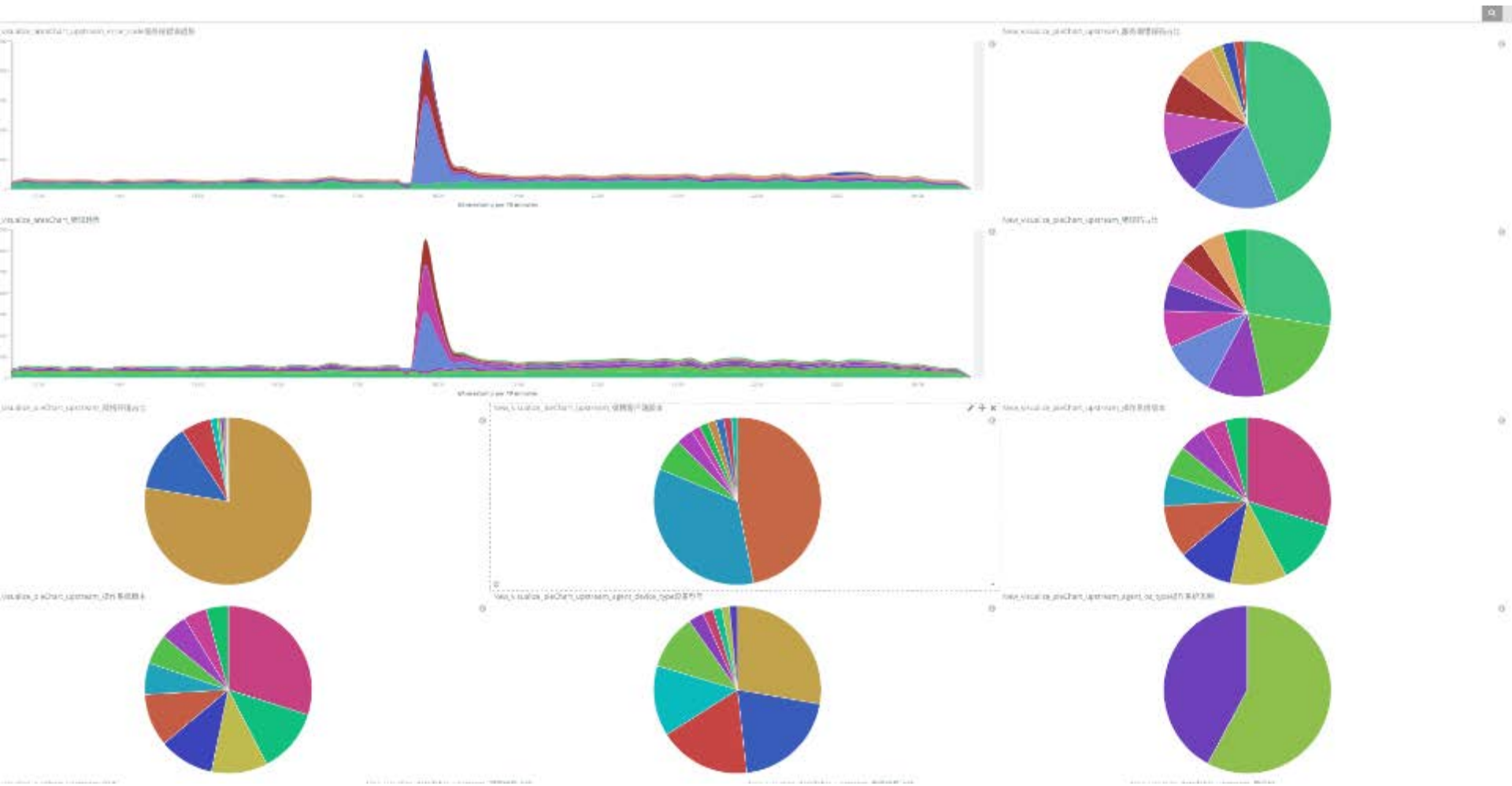
Elasticsearch常规扩展架构



小企业日志的一些应用

- 1.运维监控
- 2.基于Elasticsearch的报警
- 3.数据统计&广告平台

运维监控



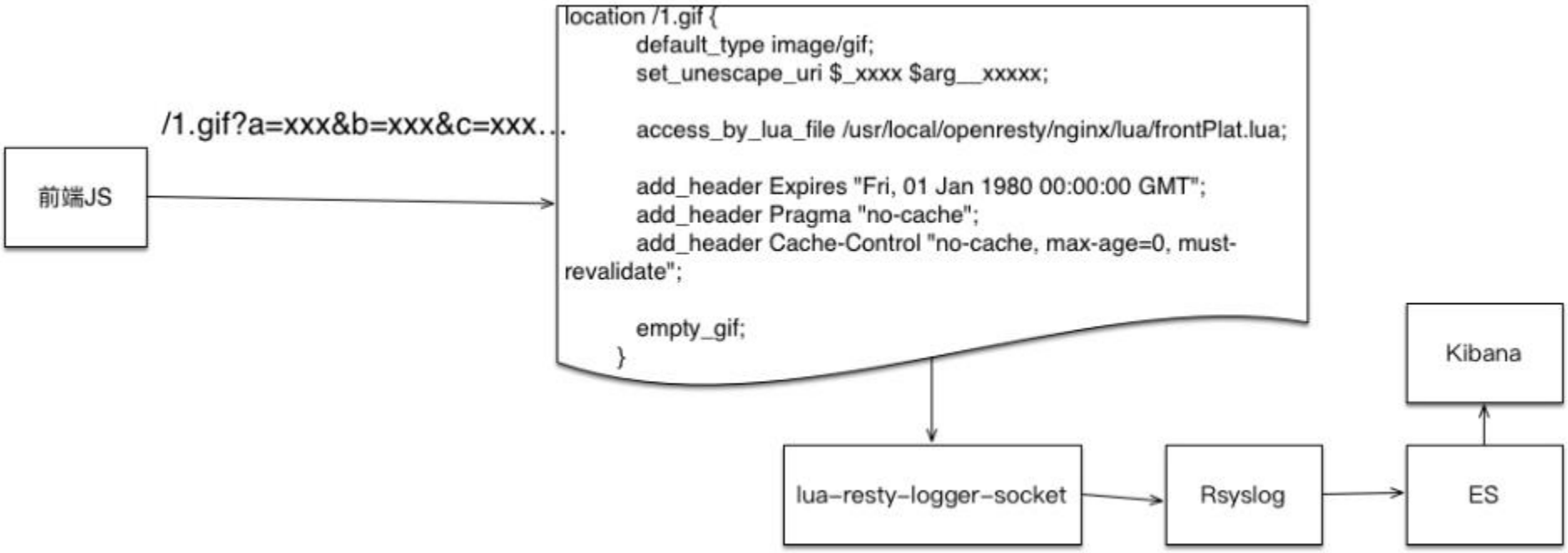
基于Elasticsearch的报警

- 原理：使用ES相关的API。
- ES官网提供了开源的组件watcher。
- 也可以自己实现，使用ESAPI + crontab。

基于Elasticsearch的报警

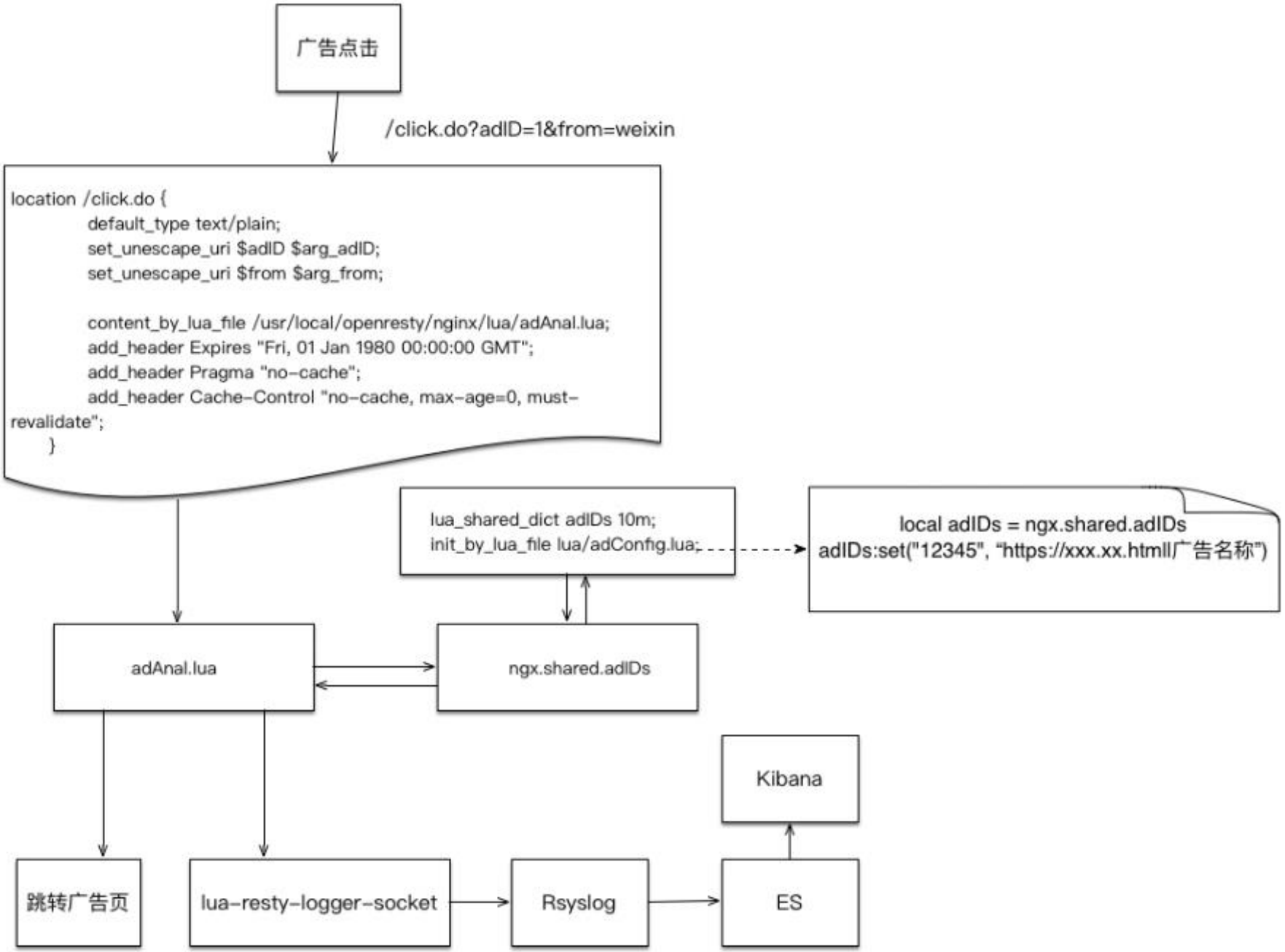
```
    param = ({"facets": {"terms": {"terms": {"field": "interface","size": 50,"order": "count","exclude
{"term": {"interface": "[redacted]"}},{"term": {"interface": "[redacted]
{"@timestamp": {"gt": "now-5m"}}}]}}}}}}}},"size": 0})
    infoDict = {
    'ES_IP':'[redacted]',
    'ES_PORT':9200,
    'requestSuffix':'search',
    'requestMethod':'POST',
    'param':param,
    'indexName':'logstash-tomcat-',
    'indexSuffixFormat':'day',
    'alertType':'search',
    'threshold':50,
    'mailUserFileName':'/usr/local/pythontools/python_ops/email_group/tomcat_error_mail_group.txt',
    'mailSubject':'LOGSTASH_TOMCAT_ERROR_EVERY_TIME',
    'mailTitle':'the error count > 50 in the past 10min',
    'mailContent':'',
    'kibanaUrl':'tomcat_url_v1',
    'kibanaUrlInterval':5
    }
```

前端点击数据统计

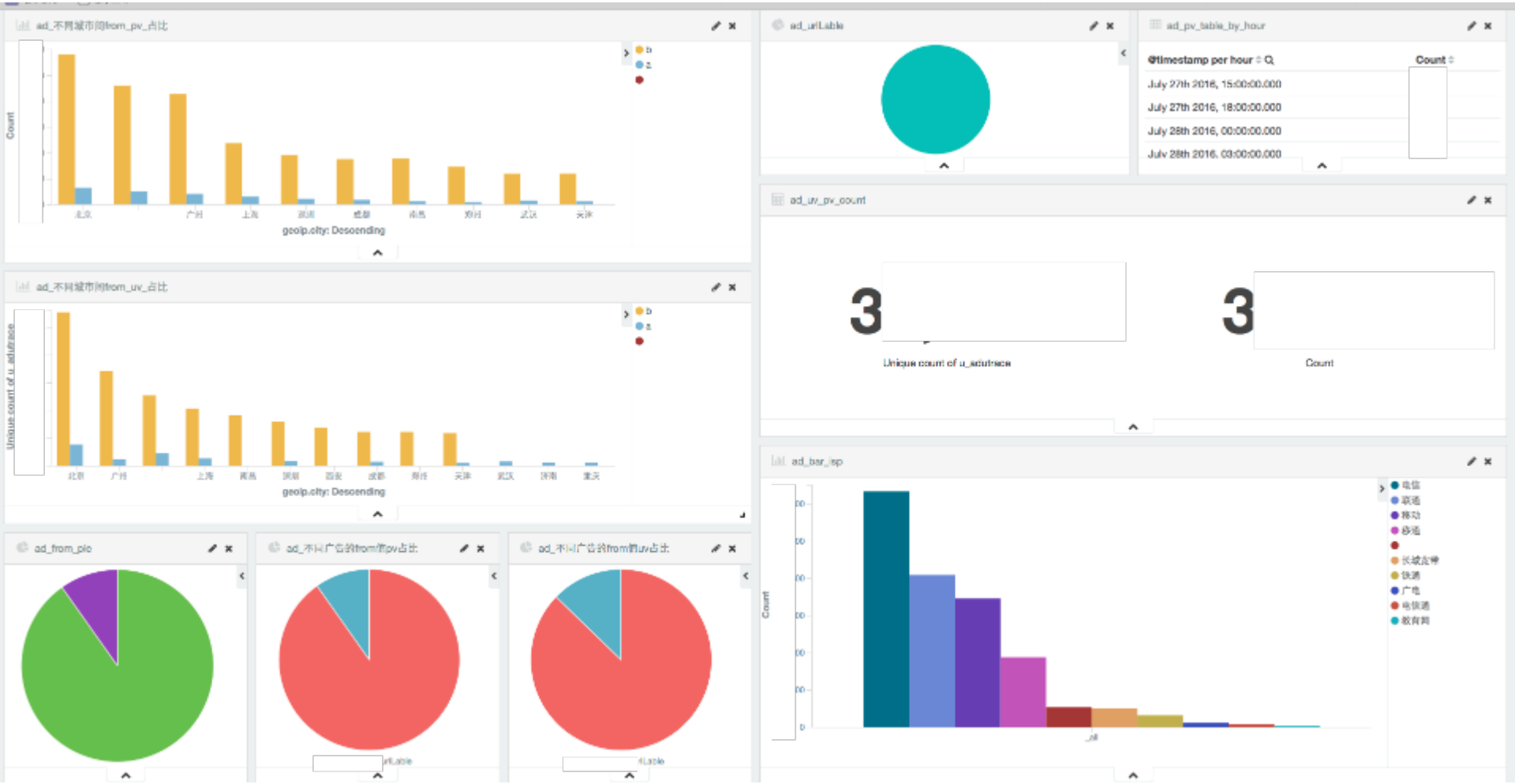

/1.gif?a=xxx&b=xxx&c=xxx...
前端JS
location /1.gif {
default_type image/gif;
set_unescape_uri $_xxxx $arg__xxxxx;
access_by_lua_file /usr/local/openresty/nginx/lua/frontPlat.lua;
add_header Expires "Fri, 01 Jan 1980 00:00:00 GMT";
add_header Pragma "no-cache";
add_header Cache-Control "no-cache, max-age=0, must-revalidate";
empty_gif;
}
lua-resty-logger-socket
Rsyslog
ES
Kibana

广告统计

- 设计的目标：
  - 业务方只需提供广告页链接及广告ID、广告来源（from），平台自动跳转广告页并统计各个维度数据。
  - 可以通过配置进行设置。
  - 不需要修改前端代码。
  - 实时数据。


广告点击
/click.do?adID=1&from=weixin
location /click.do {
default_type text/plain;
set_unescape_uri $adID $arg_adID;
set_unescape_uri $from $arg_from;
content_by_lua_file /usr/local/openresty/nginx/lua/adAnal.lua;
add_header Expires "Fri, 01 Jan 1980 00:00:00 GMT";
add_header Pragma "no-cache";
add_header Cache-Control "no-cache, max-age=0, must-revalidate";
}
lua_shared_dict adIDs 10m;
init_by_lua_file lua/adConfig.lua;
local adIDs = ngx.shared.adIDs
adIDs:set("12345", "https://xxx.xx.html/广告名称")
adAnal.lua
ngx.shared.adIDs
Kibana
跳转广告页
lua-resty-logger-socket
Rsyslog
ES

广告统计


ad_不同城市间from_pv_占比
Count
geoip.city: Descending
ad_不同城市间from_uv_占比
Unique count of u_adutrace
geoip.city: Descending
ad_urlLable
ad_pv_table_by_hour
@timestamp per hour
Count
July 27th 2016, 15:00:00.000
July 27th 2016, 18:00:00.000
July 28th 2016, 00:00:00.000
July 28th 2016, 03:00:00.000
ad_uv_pv_count
3
Unique count of u_adutrace
3
Count
ad_bar_isp
Count
ad_from_pie
ad_不同广告的from的pv占比
ad_不同广告的from的uv占比

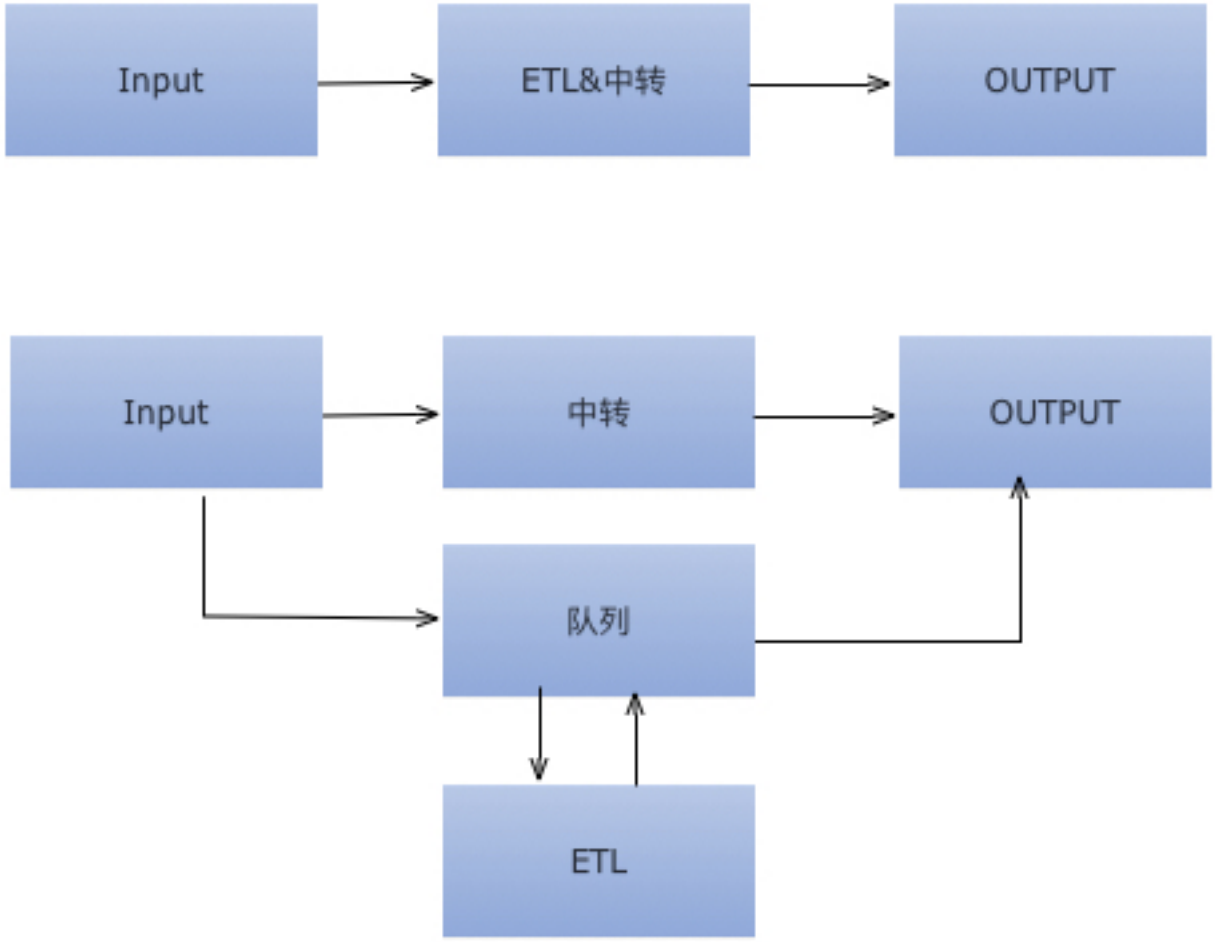
几点补充

- 一切架构设计都要测试为先，同时做好监控。
- Logstash的日志收集相对不可控。
- Rsyslog提供impstats监控模块。
- Elasticsearch的监控可以使用自带的插件。

大企业日志系统的特点&关注点

- 成本。
- 运营与运维数据复杂性。
- 历史遗留问题较多。
- 各部门相互依赖。
- 应对业务突增情况的能力。
- 丢失率监控&SLA。

同步与异步架构



同步与异步架构

- 传输的归传输，解析的归解析。
- 应对整体服务出现日志暴增的情况。
- 分池之后便于计算资源容量。

成本控制&业务优化

一个问题：日志量大是可以自豪的事情吗？

日志分类
格式协定
日志合并
日志分级传输
运维侵入开发

日志分类

- 系统监控日志（可以全量，但要做权衡）
- 运营打点计数日志（必须全量，关系到财报）
- 性能监控采样日志
  - 全量UID 全量状态码 部分详细信息

格式协定

- 带宽
- 计算量
- 兼容

日志合并

- 具有相同意义的日志不要写多份。
- 避免多传，一写多读。

日志分级传输

- 高保通道，传输业务数据相关。
- 通用通道，运维监控数据相关。
- 降级通道，紧急情况按比例丢弃。
- 开发调适通道，不转发，只落磁盘，定时丢弃。

以Rsyslog为例的实现

1. 用日志tag进行区分。

2. 高保的tag调用高保RuleSet，其他的日志tag以syslogfacility-text进行区分，通过syslogfacility-text来写入普通的通道。

3. 使用rsyslog的action中的属性：action.execOnlyEveryNthTime实现按比例丢弃。

日志降级的处理

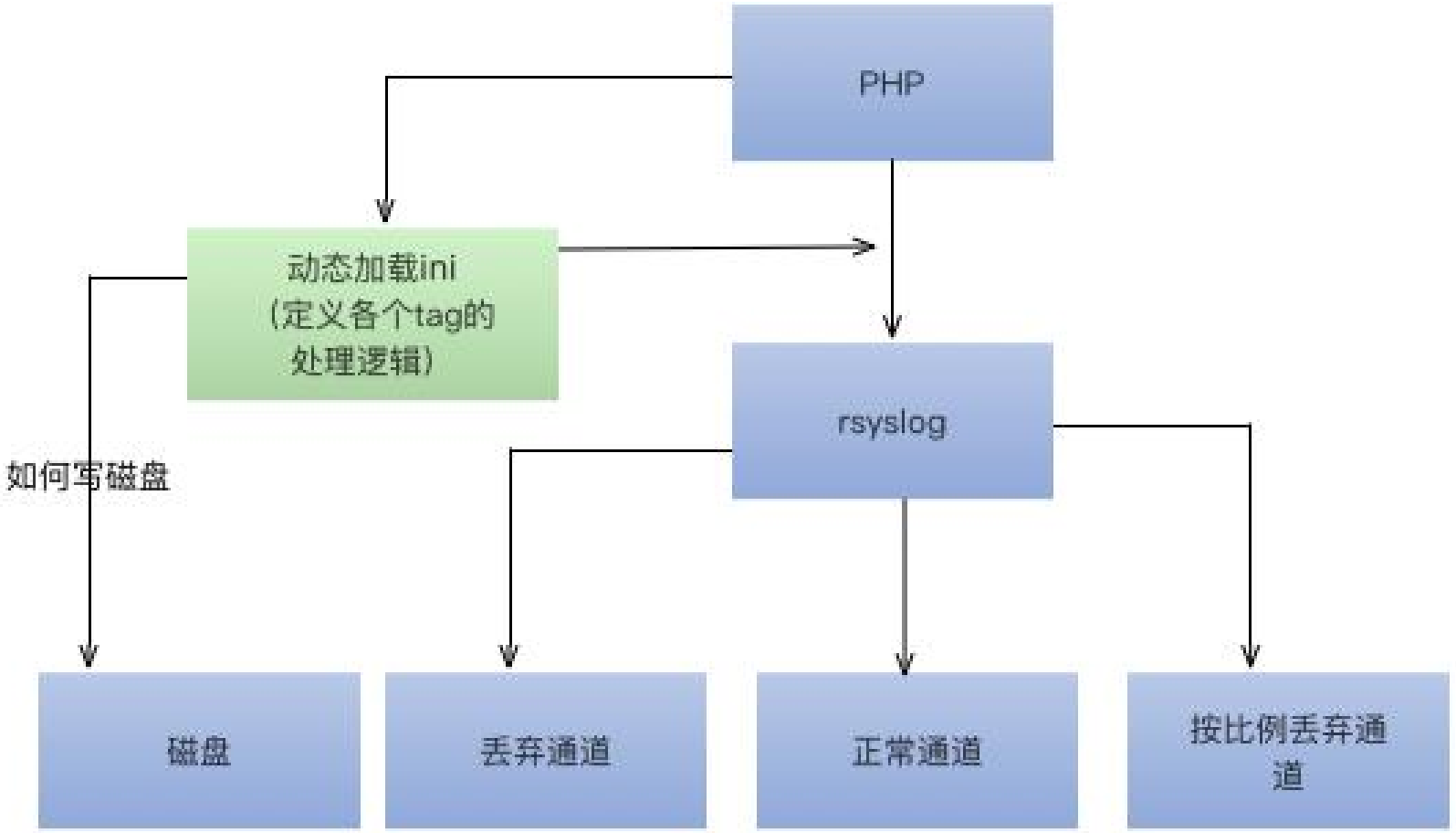
- 日志业务背景：日志在正常情况下传输量恒定，在业务暴增情况下错误日志与业务日志量同步增加。某些情况下超出平时上百倍！

- 瓶颈（降级标准）：CPU计算，带宽，磁盘。

以微博为例的日志降级逻辑

- 核心：动态调整日志tag的级别
- 是否转发
  - 全量转发
  - 按比例丢弃
  - 不转发
- 是否落本地磁盘
  - 全量落磁盘
  - 按比例丢弃
  - 不写磁盘

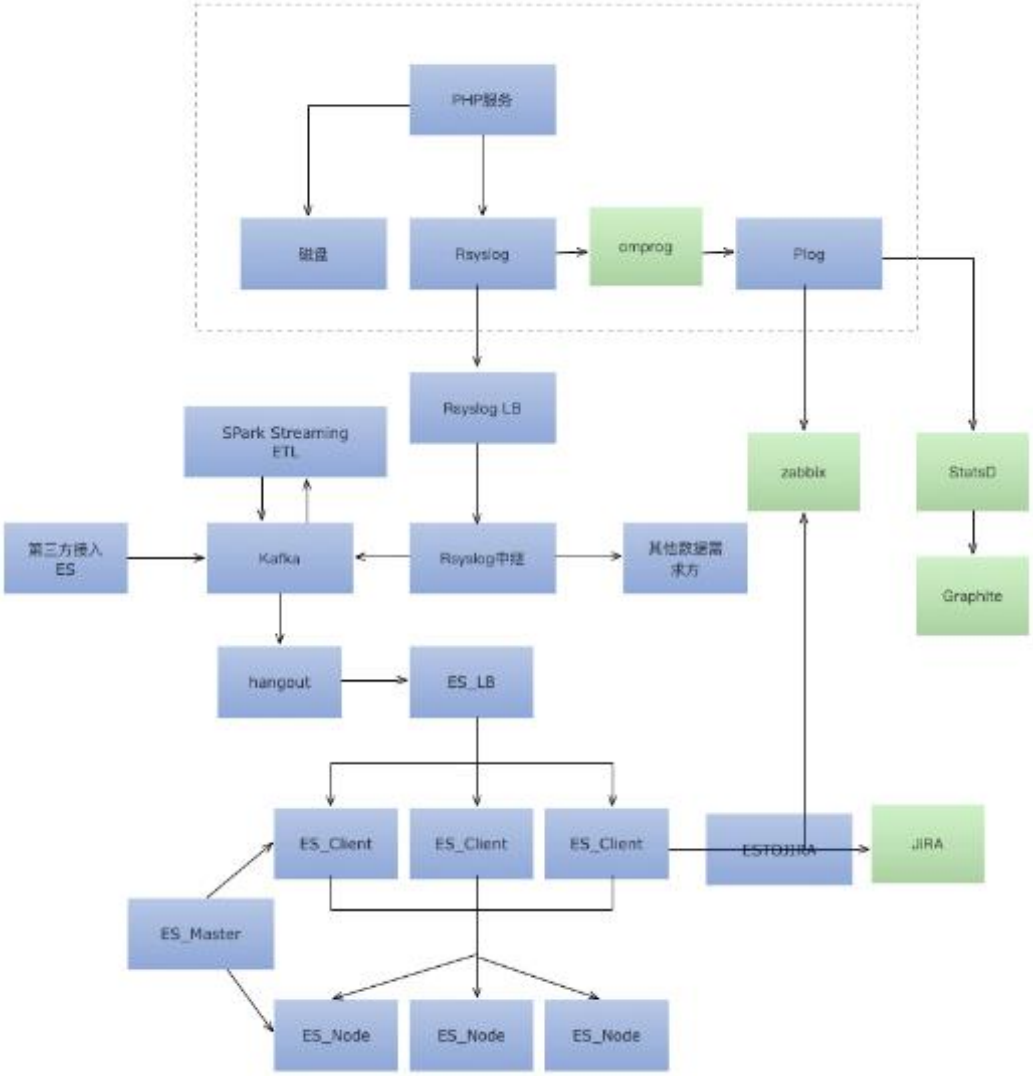
以微博为例的日志降级逻辑



几点补充

- 日志传输的监控。
- 测试。
- 通过队列解藕各个部门的依赖。（很难）
- 运维侵入开发。

手机微博日志系统架构图



各个组件简介

- Rsyslog：前端日志收集，后端日志中继分发。
- Plog：轻量级日志格式处理框架。
- Hangout：模仿Logstash写的日志处理框架，主要做Kafka与Elasticsearch之间的数据传输。
- Kafka：做日志队列缓冲。

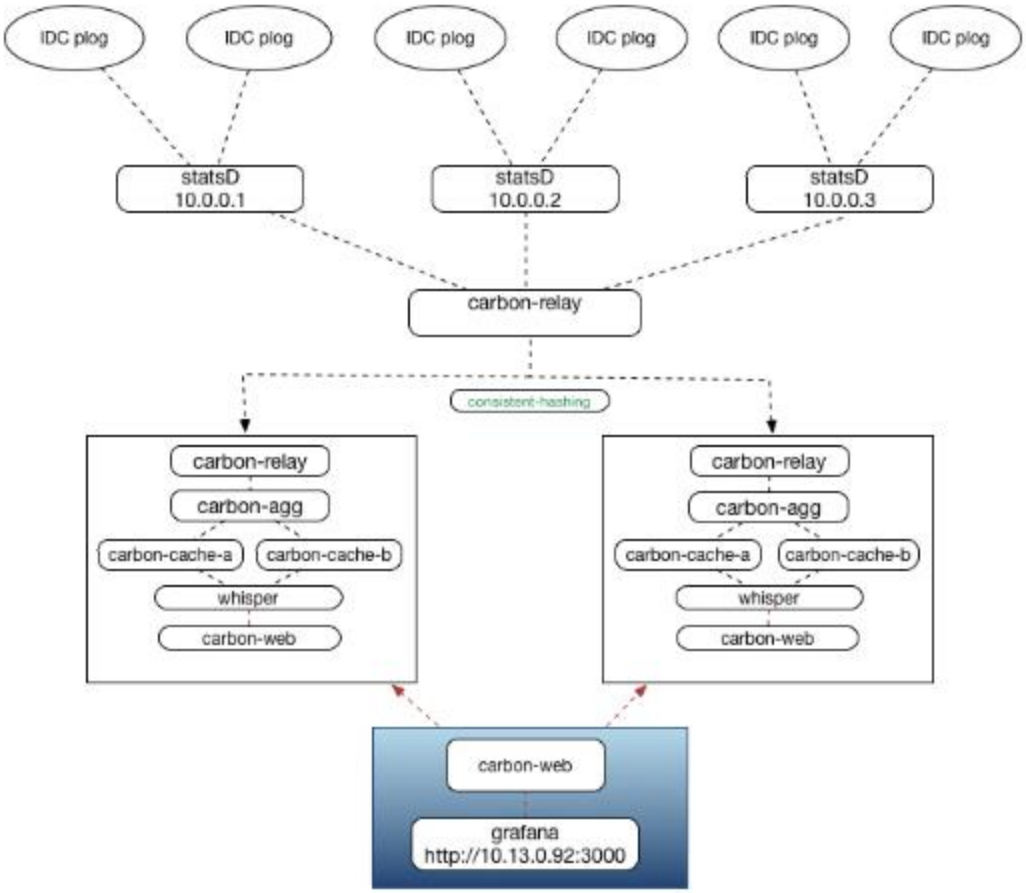
各个组件简介：ESTOJIRA

- ESTOJIRA：自动从ES取数据分析客户端Crash日志，服务于QA。

各个组件简介：ESRouteUtils

- 背景：集群内服务器配置（CPU，内存等）存在差异。
- 功能：根据ES集群内不同服务器配置及历史负载情况分配索引分片数。
- 基于历史监控数据分析生成分配规则。

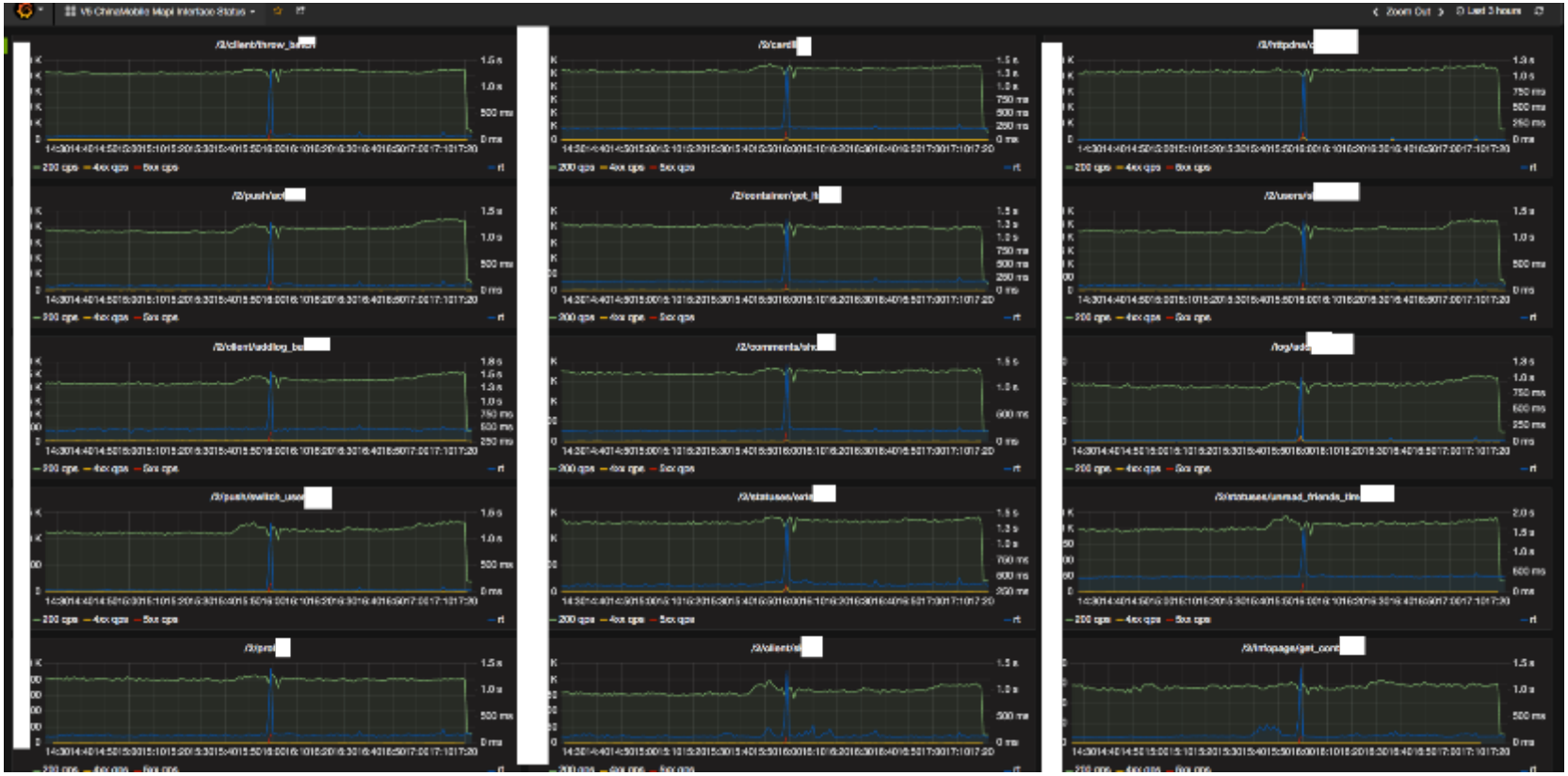
各个组件简介：StatsD&Grahite



各个组件简介：StatsD&Grahite

- StatsD：汇聚不同时间的同一key的值。
- Carban-relay：负载均衡（一致性hash）。
- Carbon-agg：汇聚同一时间不同key的值。

各个组件简介：StatsD&Grahite

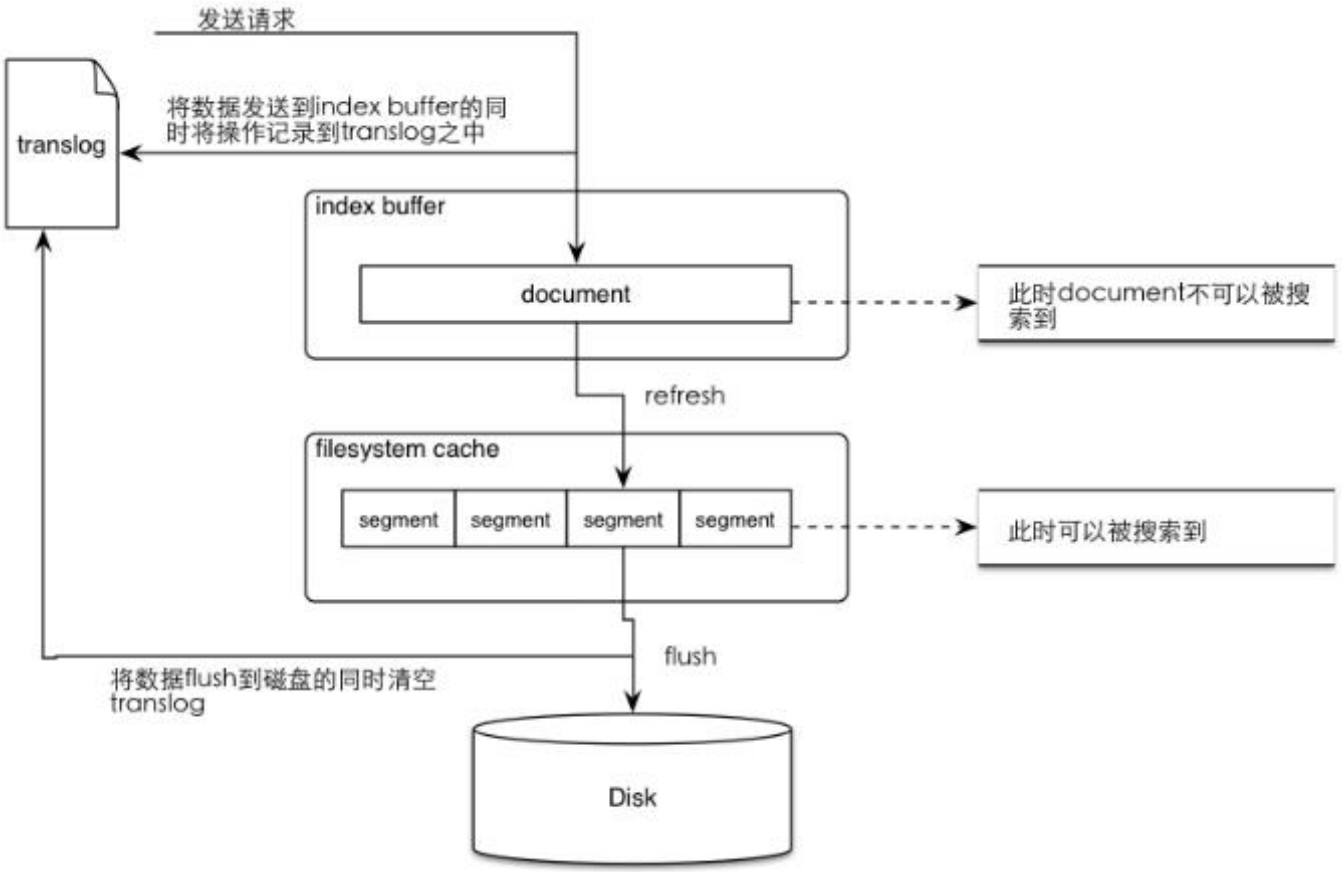


各个组件简介：StatsD&Grahite

调优漫谈

- 从来没有标准答案。
- 一切从了解其内部开始。
- 测试与监控。

Elasticsearch refresh flush



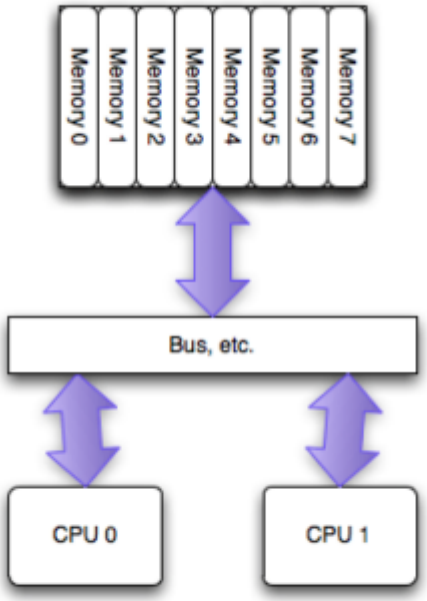
调优方向

- 内存：
- CPU：
- IO：
- 网络限流：

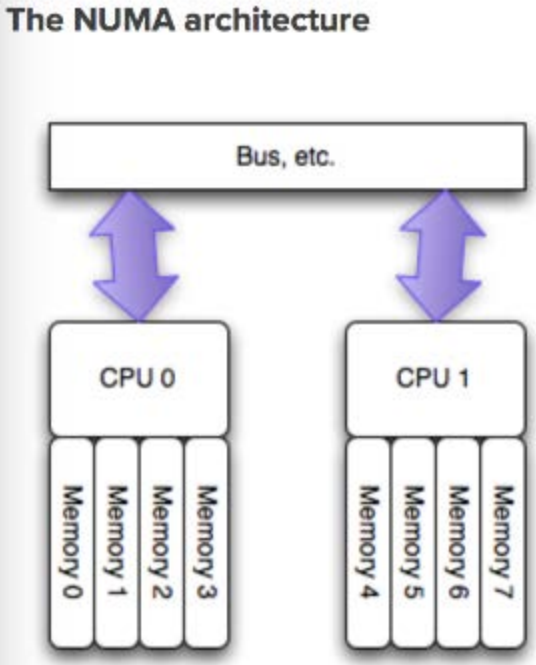
内核系统层面

- vm.zone_reclaim_mode=0

SMP/UMA&NUMA架构

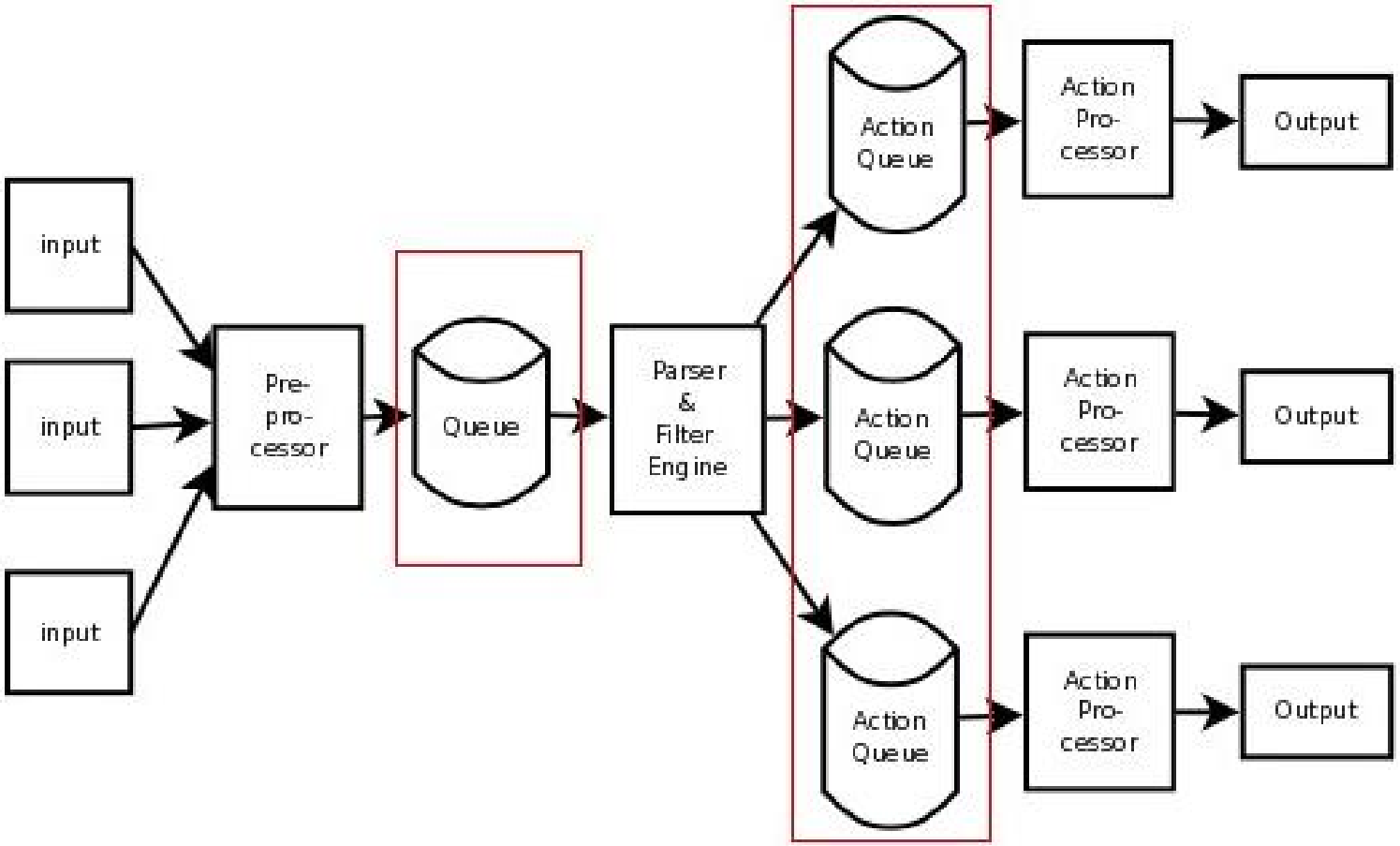


The SMP, or UMA architecture, simplified



The NUMA architecture, simplified

Rsyslog


input
input
input
Pre-
pro-
cessor
Queue
Parser
&
Filter
Engine
Action
Queue
Action
Queue
Action
Queue
Action
Pro-
cessor
Action
Pro-
cessor
Action
Pro-
cessor
Output
Output
Output

Rsyslog队列详解

- 两种类型队列:
  - Main queue
  - Action queue
- 四种队列设置:
  - Direct queue
  - Disk queue
  - In-memory queue
    - LinkedList
    - FixdArray
  - Disk-assisted memory queue

调优方向

- 虽然官网写的很牛，但自身存在一些问题。
- 尽量外部队列（kafka）替换自身队列。
- 应对特殊情况，提高主队列大小。
- 核心还是不要挂太多逻辑，减少计算提高吞吐。
- TCP backlog调优。
  - MaxSessions(10%+5)

Rsyslog监控&impstats

- Actions：
  - processed：rsyslog处理的日志数量。
  - failed：在处理中失败的次数。
  - suspended：action被挂起的次数。
- Queue：
  - size：当前在队列内的日志数量
  - enqueued：经队列流转的日志数量。
  - maxsize：在队列中日志数量最大数。
  - full：发生队列满的次数。
  - discarded.full：因队列满而丢弃的日志数量。
  - discarded.nf：因接近队列满时丢弃的日志数量。

结语&总结

- 日志量就是成本，所以不要以花了多少钱而自豪，而是要以做了多少事作为目标。要传输有价值的数据！
- 要多注意业务优化。
- 选择哪一种架构由具体的场景决定，不要过度设计，但要尽量快速迭代。

Thank you