






大型券商的ELK日志运维实践

李凯平
国信证券



战略级赞助商  HUAWEI | 钻石级赞助商  普翔 | 白金级赞助商  华夏博格 | 神州数码  Digital China | 金牌级赞助商  iDataAPI

合作伙伴  开源中国  掘金  知乎  IT大咖说  otpub  Broadview  百格活动  MAXHUB
oschina.net www.broadview.com.cn bagevent.com 高效会议平台

大型券商的ELK日志运维实践

李凯平

11-10,2018



议题

01 背景和目标

02 技术方案

03 应用效果

04 实践中的问题及解决方案

05 后续工作

1、背景和目标 | 背景

01

指标类监控工具多，但缺乏对日志的利用，完整的运维数据包括日志、指标和配置

02

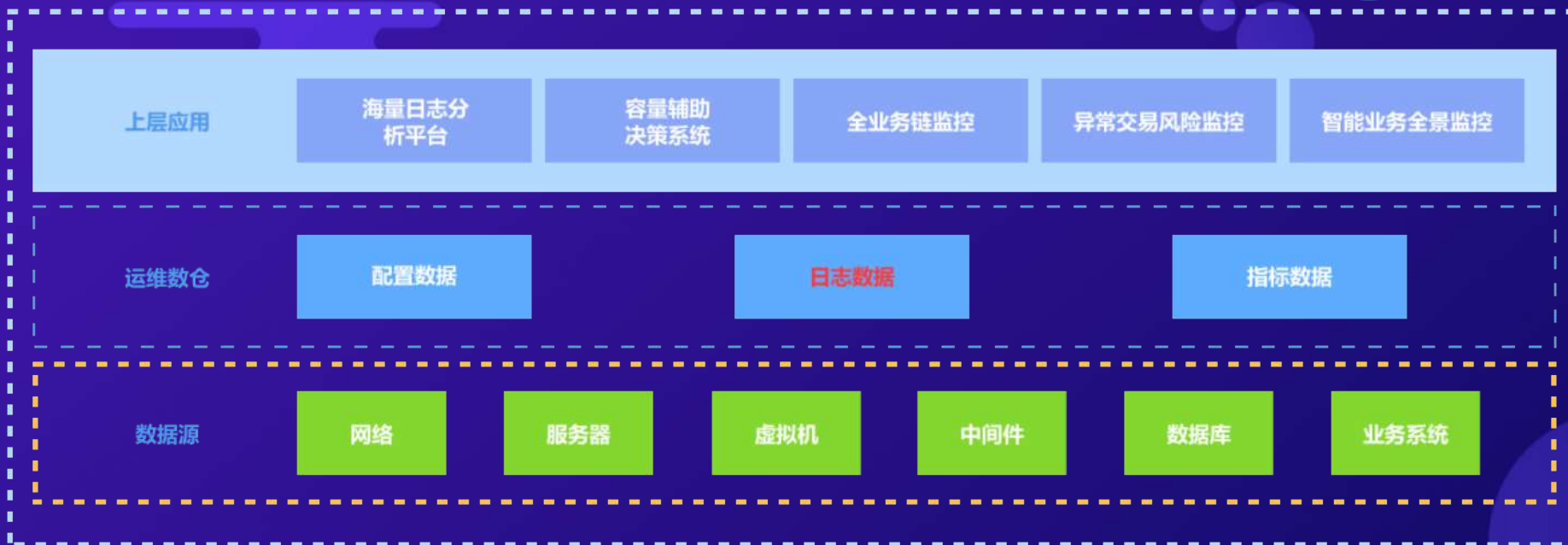
日志数据分散，保存时间短，无法满足运维即席检索及监管审计要求

03

日志中的业务价值有待挖掘，如用户行为、应用性能、安全攻击等

1、背景和目标 | 目标

- 增加日志处理能力，结合配置数据，增强现有的运维能力
- 业务监控，应用监控，智能告警，容量辅助决策
- 建立可持续扩展的运维数据仓库，对接更多的数据源，提供数据消费能力



2、技术方案 | 选型

Hadoop

批量统计、事后分析

存储、处理为主

超低成本存储

ELK

实时搜索

实时统计、在线分析

采集、解析、处理、
展示一体化

低成本存储



ELK为主
实时热数据

Hadoop为辅
冷数据

2、技术方案 | 系统架构

管控平台



管控平台



2、技术方案 | 其他关注点

企业级应用还需要关注以下几点：

1

数据规范：建立元数据体系

2

数据标签：与CMDB结合，数据经过治理才能有效应用

3

容量管理：了解重要组件的容量性能基线才能高效运维

4

管控与自动化：具有集中化的管控和自动化能力，才能有效管理

5

存储规划：区分冷热数据，做好数据的存储和使用规划

2、技术方案 | 更多思考



3、应用效果 | 被采集系统



50+个系统
4500+主机



采集了全部网络设备日志、重要系统应用日志、Linux及windows系统日志

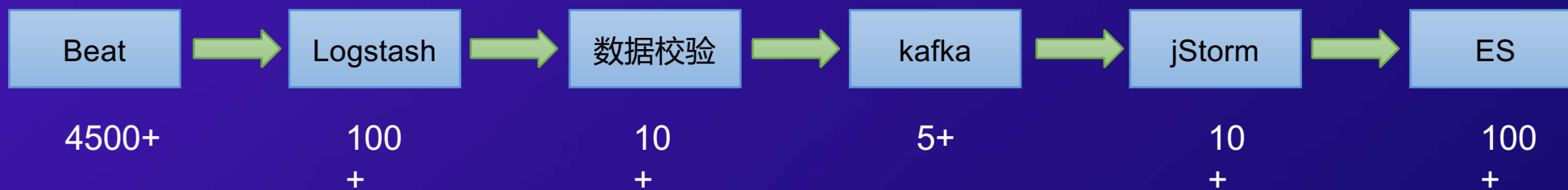


集中化管理

3、应用效果 | 集群情况

- 100+解析节点、20个ES服务器节点，日采集3.5T日志
- 秒级查询，秒级告警

3.5T
每日

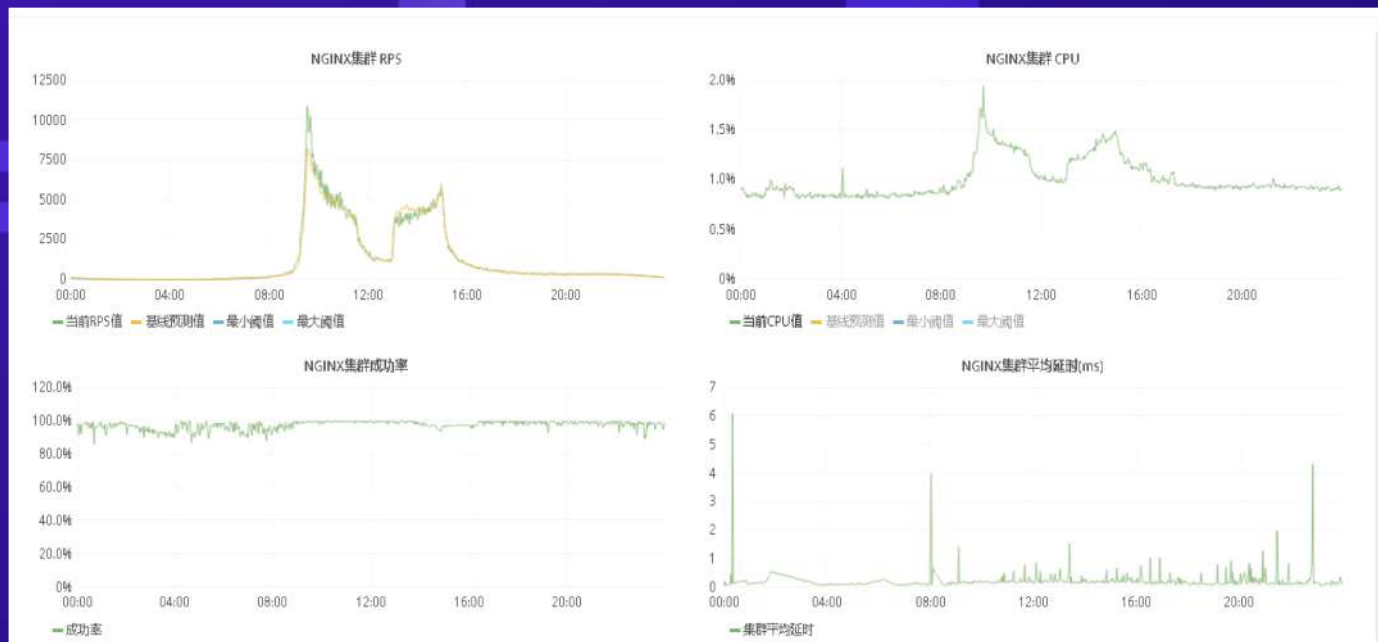


3、应用效果 | 应用情况

- 日志检索

- 业务指标分析

- 调用链



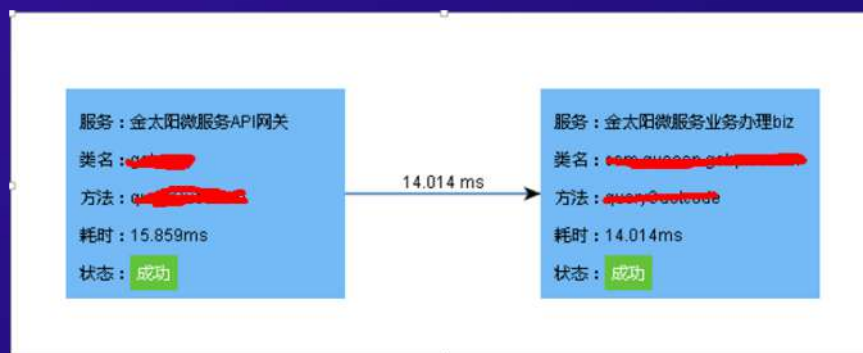
3、应用效果 | 应用情况

- 日志检索
- 业务指标分析
- 调用链

调用链列表 / 列表详情 拓竹显示

开始时间: 2018-11-06 09:41:27.882 结束时间: 2018-11-06 09:41:27.898 调用链耗时: 15.859 ms 调用失败 所有

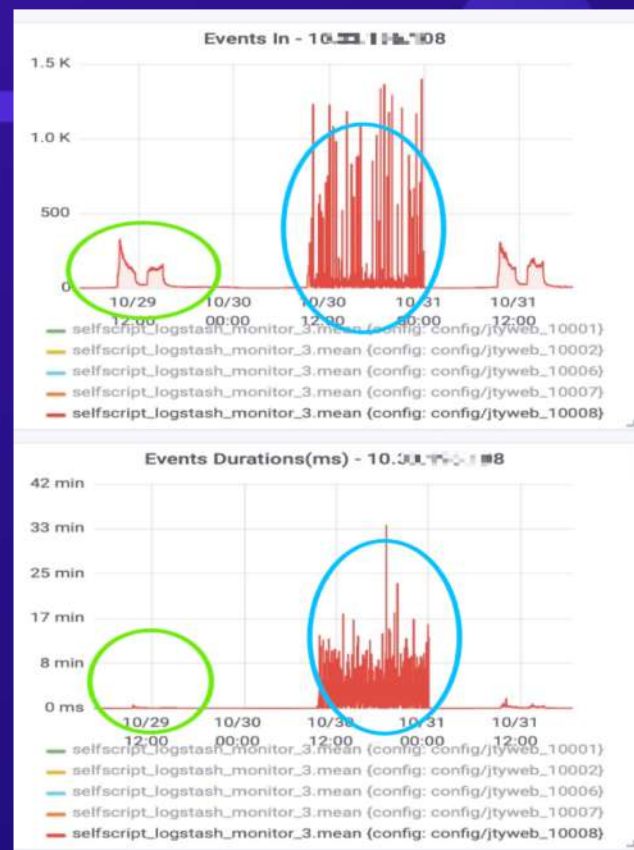
系统	主机名称	服务	类_方法	开始时间	结束时间	耗时	状态
金太阳4.0	ZWJTY4-WAPI089	金太阳微服务API网关	gateway_queryCustcode	2018-11-06 09:41:27.882	2018-11-06 09:41:27.898	15.859ms	成功
金太阳4.0	ZWJTY4-WBIZ056	金太阳微服务业务办理biz	com.guosen.gsbp.service.Gsb...	2018-11-06 09:41:27.883	2018-11-06 09:41:27.897	14.014ms	成功



4、实践中的问题及解决方案 | 问题

1.时延

2.数据的漏采和非标



4、实践中的问题及解决方案 | 方案

做好自监控

建立好基线

做好自监控

&

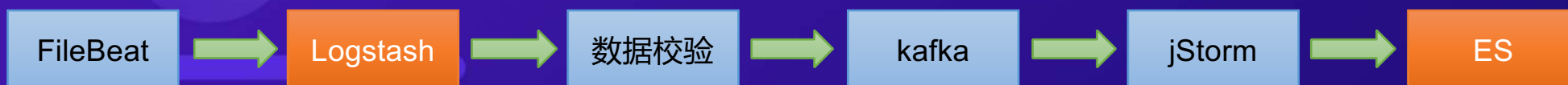
建立好基线

做好自监控

建立好基线

重要的事情说三遍!!!

4、实践中的问题及解决方案 | 方案



1. 时间标记，时延监控
2. 日志自监控
3. 漏文件和文件解析失败
4. 文件级别采集监控

name ↕	lang ↕	script ↕
delay_log_to_filebeat	expression	doc['normalFields.collecttime']-doc['timestamp']
delay_filebeat_to_prekafka	expression	doc['normalFields.logchecktime']-doc['normalFields.collecttime']
delay_jstorm_to_index	expression	doc['indexTime']-doc['deserializerTime']
delay_prekafka_to_jstorm	expression	doc['deserializerTime']-doc['normalFields.logchecktime']
delay_log_to_index	expression	doc['indexTime']-doc['timestamp']

找出Filebeat、Logstash能力基线 – 示例

实例数	pipeline workers	output workers	old heap(%)	CPU(%)	CPU(总%)	TPS	备注	时延
1	4	1	~	54%	54%	3200		
1	8	1	~	86%	86%	4900		
1	12	1	~	90%	90%	5400		
1	16	1	~	90%	90%	5500		
2	4	1	~	43%	86%	2700		
2	4	1	~	43%	86%	2700	内存为 2G	1.5ms
2	4	1	18%	35%	70%	3700	grok + geoip	1ms
2	8	1	25%	45%	90%	4800	grok + geoip	2ms
2	8	1	70%	47%	94%	2900		
2	12	1	70%	48%	96%	2900		
4	4	1	70%	24%	96%	1530		
4	8	1	70%	24%	96%	1580		6ms

Logstash 过滤器为空。									
Logstash	~	512	2s	8294	5266944	18%	0.7%	1	
Logstash	~	1024	2s	8806	5592064	17.5%	0.8%	1	
Logstash	~	2048	2s	8601	5462016	16%	0.9%	1	
Logstash	~	1024	4s	8601	5462016	16%	0.8%	1	
Logstash	~	1024	8s	8396	5331968	16%	0.8%	1	
Logstash	1	1024	2s	8601	5462016	17.4%	0.8%	1	
Logstash	2	1024	2s	8601	5462016	17%	0.8%	1	
Logstash	4	1024	2s	8601	5462016	16.8%	0.8%	1	
Logstash	8	1024	2s	8192	5201920	16.6%	0.8%	1	
Logstash 过滤器为空, filebeat 位于不同机器。									
Logstash	~	1024	2s	3072	1950720	6.2%	0.6%	4	11763
Logstash	~	1024	2s	4096	2600960	8.1%	0.5%	3	11353
Logstash	~	1024	2s	6144	3901440	11.9%	0.6%	2	11112
Logstash	~	1024	2s	9420	5982208	19.9%	0.8%	1	9117
Logstash 过滤器为空, 单个 filebeat 进程, 多个文件源, 一个文件源 6G, 一个文件源是空文件, 2 个文件源的文件不存在。									
Logstash	~	1024	2s	总: 9011	文件源:4	19%	0.4%	1	8700
Logstash 过滤器: grok, geoip, file。									
Logstash	~	1024	2s	4300	2731008	8.8%	0.5%	1	
Logstash 过滤器: grok, geoip, file; filebeat 有 4 个实例位于不同机器。									
Logstash	~	1024	2s	1228	780288	2.4%	0.4%		

- 综上, filebeat -> logstash。
 1. logstash 的处理能力: 1.2w。
 2. filebeat 单个进程处理能力: 9000。
- 基于以上 2 个限制。
 1. n 个 filebeat 对接到一个 logstash, 则每个 filebeat 的处理能力为 $\min\{1.2w / n, 9000\}$ 。
 2. 1 个 filebeat 接 m 个文件源, 则每个文件的读取速度为 $\min\{1.2w/n, 9000\} / m$ 。

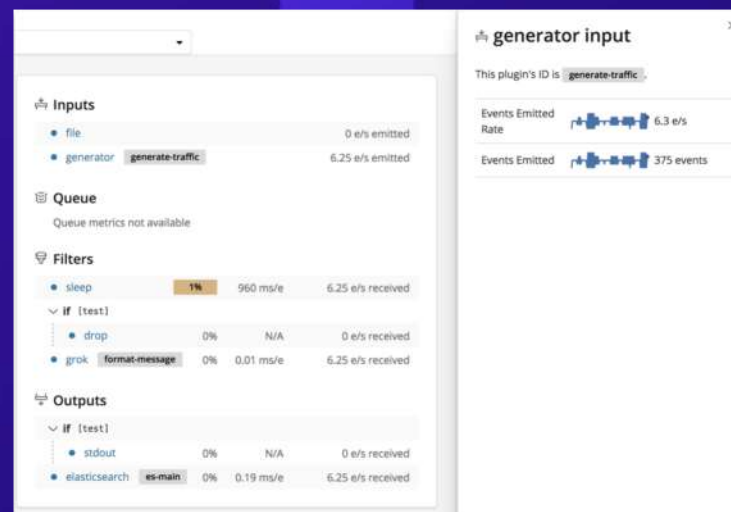
Logstash性能优化



问：Logstash性能受到很多因素的影响，除了做好参数调优，通过实验得到基线，还需要做什么工作？



答：利用pipeline工具发现性能不足的原因，修改代码实现方式，持续的观察性能瓶颈点，经过持续改进，直到达到要求。优化的代码可以提升性能好多倍。



Pipeline工具可以看到代码级耗时情况，是性能调优利器。

ES性能优化



问：服务器异构，如何让shard更多的分布在SAS盘服务器？如何提高磁盘的利用率？

问：同一个index的shard数远小于es实例数，并且统一零点创建index，导致index实例只会分布在部分磁盘



答：配置zone，进行shard个数优化



问：在提升了磁盘利用率之后，所有磁盘iostat util都非常高，写入频次很高，吞吐量很低，如何提高磁盘吞吐量？



答：调整压缩方式、refresh时间间隔、merge线程数、translog sync模式、translog flush阈值

5、后续工作

Zabbix 4.0的原生
Elasticsearch解决方
案



全业务链监控



平台的集中管控和
自动化能力



数据服务API的梳理

5、公司招聘



诚聘英才！



谢谢聆听！



elastic 中文社区

专业、垂直、纯粹的 Elastic 开源技术交流社区

<https://elasticsearch.cn/>