日志分析平台设计文档

目录

概述

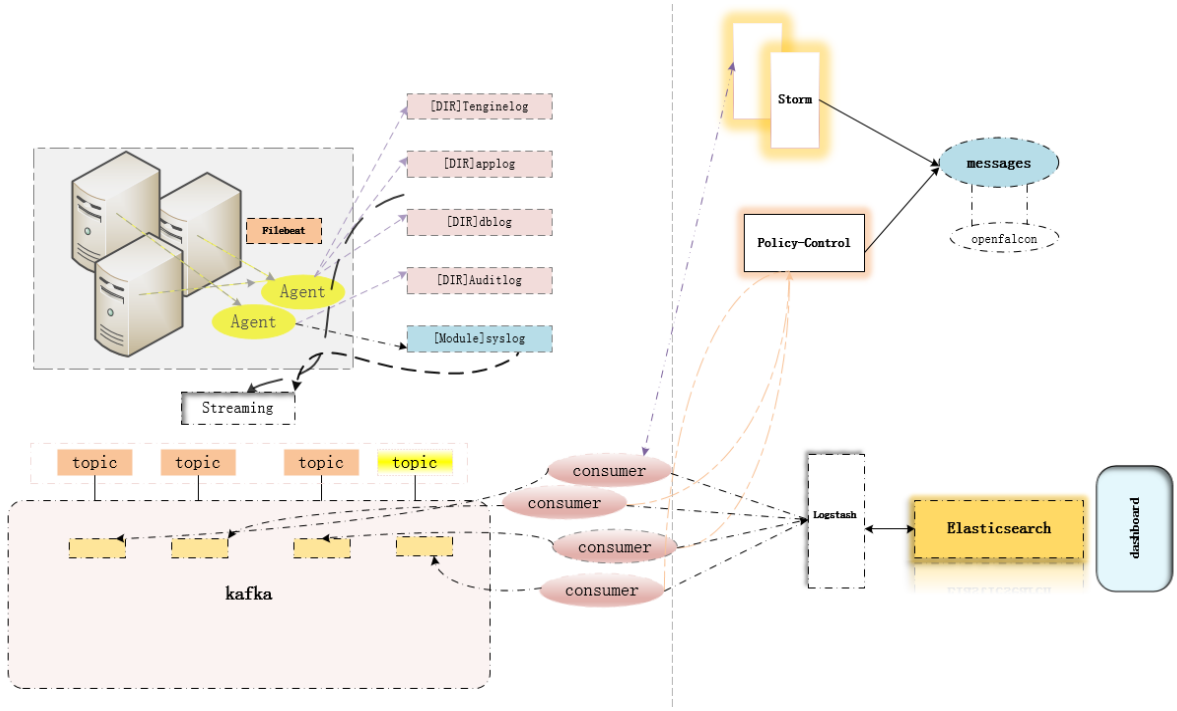
日志分析平台对收集到的日志进行分析和展示，并对隐含故障、危险操作、服务情况进行预警。

介绍

日志分析平台涉及到日志收集、拆解、存储、分析、检索、展现、报警等几个重要部分。

日志收集器（Sniper）收集系统、服务级日志并作为生产者将每条日志作为一条消息写入消息中间件（Broker），提供给日志索引器（Indexer）、日志分析器（analyzer）等应用消费并进行字段拆分、分析、报警、落库、展现等操作。

架构和基本数据流



日志收集器 - Filebeat

日志分析平台主要收集以下日志：

- 系统日志
- /var/log/message
- /var/log/secure.log
- cmd_log
- 网络日志
- Tengine 日志
- access log
- error log
- DB 日志

目前所有的系统日志和网络日志均实时上传至 rsyslog-ng 服务器（syslog.pajkdc.com）上存储，因此日志只需要在 rsyslog 服务器 、DB 服务器、LB 服务器上进行收集即可，不需要大面积部署日志收集器，减少日志收集器的维护成本。

日志收集器作用

- 指定日志收集文件（支持通配符匹配、包括、排除）；
- 能够自动发现符合要求的日志并收集；
- 能够准确跟踪日志文件；
- 添加额外字段标识日志类型；
- 将类似 DB 操作日志、Java 日志等多行日志形成一条完整日志进行收集或转储；
- 支持输出到消息中间件进行缓冲消息。
- 支持不同类型日志分发到不同的目标上，如转发到 kafka 不同的 topic 上。

日志收集器要求

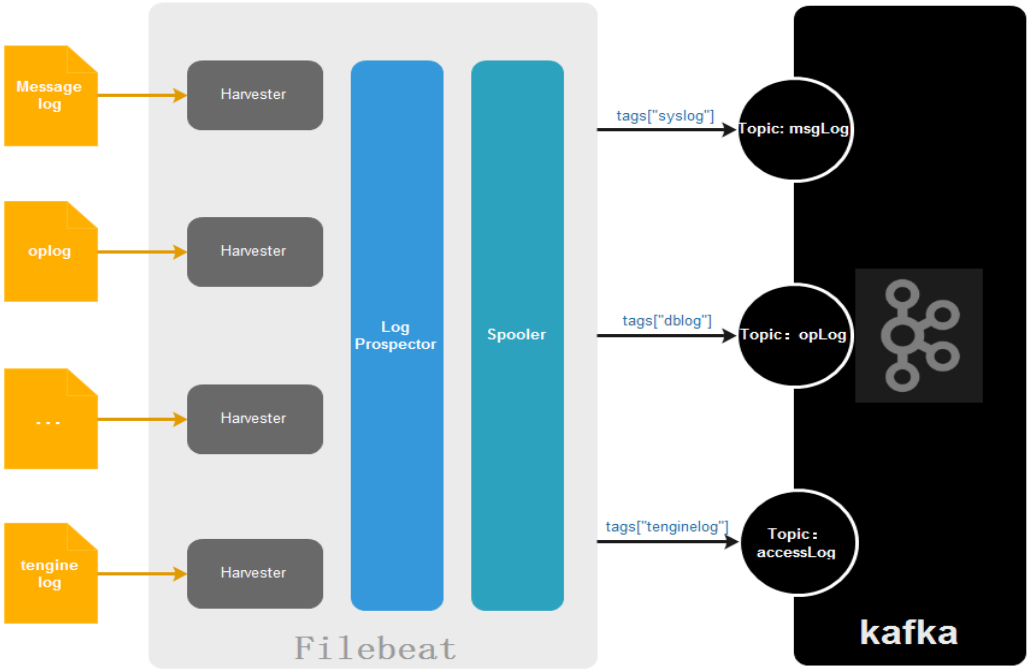
日志收集器需要运行在 Tengine、MySQL、rsyslog 等高负载服务器上，所以日志分析器满足**轻量级**、**高效**才能够保证日志收集器在收集日志期间不影响服务运行，同时能够及时收集日志。**易部署**、**易配置性**也是日志收集器选型的标准。

Filebeat

Filebeat 是 Elastic stack 提供的一个负责日志收集和转储的日志收集器。Filebeat 是基于 logstash-forwarder 源代码重新使用 GO 语言进行改造和实现的产物，用于替代 logstash-forwarder，换句话说 Filebeat 是 logstash-forwarder 新版。支持使用通配符指定收集的日志目录或文件，并将日志输出到 logstash、Elasticsearch、console、file、kafka、redis。

Filebeat 和 logstash 对比，logstash 由 ruby 实现，在运行过程中还需要依赖 JDK，在收集大量日志时，logstash 会占用较高的内存并消耗 CPU，可能会影响服务响应。而 Filebeat 在同等情况下，CPU 和内存消耗率极低。同时 Filebeat 满足现日志收集器作用中列出的功能，所以选用 Filebeat 做为日志平台的日志收集器。

Filebeat 收集日志流



Filebeat 对收集到的日志根据其大类打上一个 tags，再将其类型作为 type 和对应 Topic，之后，Filebeat 根据日志 type 将其输出到 kafka 对应的 Topic 中。例如 /var/log/messages 日志属于系统日志这一大类，所以 tag 为“syslog”。其本身类型为 msg，则将 type 设置为 msgLog，并输出到 Topic msgLog。

Broker - kafka

Broker 作用

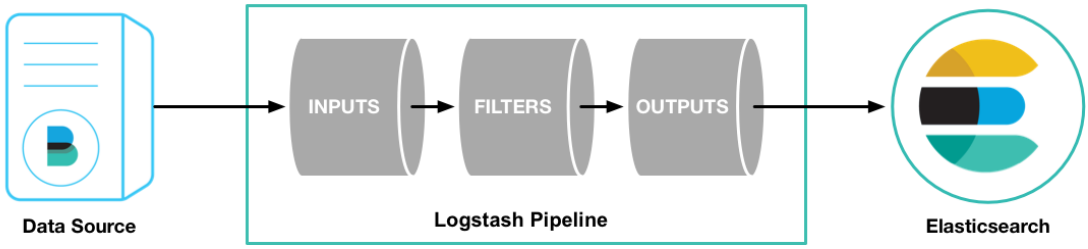
使用 kafka 作为 Broker 可以保证在后端无法即使消费的情况下，不会导致日志缺失。Kafka 还可以提高整个系统的可拓展性、一致性和多样性。

- **可拓展性：**例如当日志索引器集群负载较高、处理性能较差时，可以通过横向扩展日志索引器实例数来提高集群的处理能力。
- **一致性：**当消费者是一个集群时，kafka 可以保证同一个集群中的消费者不会重复消费数据。
- **多样性：**日志拆解器、storm、日志分析器等应用可以各自成组（group）同时订阅相同 Topic 中的信息。

日志索引器（Indexer）- Logstash

Logstash 是日志整合的一大利器。在大部分日志平台架构中，logstash 均作为 Indexer 负责对数据进行修改、过滤、拆解、分析等日志整合工作。通常作为 Indexer 的 logstash 的数据来源于 redis 或 kafka，而输出可以为任何形式的存储或消息中间件。Logstash 还支持多种多样的插件，使其功能也具有多样化。

Logstash 处理流



Logstash 所有的日志整合操作均在 FILTERS 中由 FILTERS Plugin 来完成。

Elasticsearch

Elasticsearch 负责存储经过 Logstash 整合后的日志信息，提供给展示平台用于日志检索、查询、以及报表生成。

Elasticsearch 直通多目录数据盘和 RAID0 单目录数据盘压测

软件环境

名称	版本
CentOS	6.5
Elasticsearch	5.2.1
压测工具 RALLY	0.5
数据采集 dstat	

硬件环境

服务器类型	配置	作用	数量
物理机	S3	Elasticsearch 压测机	2
物理机	A6	RALLY 压测工具	1

压测环境部署

机器 IP	节点数	节点类型	磁盘挂载类型	目录
10.255.36.19	1	Master	直通	/data/esdata[1-6]
10.255.36.19	1	Data	直通	/data/esdata[7-12]

机器 IP	节点数	节点类型	磁盘挂载类型	目录
10.255.40.26	1	Master	直通	/data/esdata[1-6]
10.255.40.26	1	Data	直通	/data/esdata[7-12]
10.255.36.19	1	Master	RAID0	/data/M
10.255.36.19	1	Data	RAID0	/data/D
10.255.40.26	1	Master	RAID0	/data/M
10.255.40.26	1	Data	RAID0	/data/D

压测命令

index-only

1admin$ /usr/local/python3/bin/esrally --distribution-version=5.2.1 --target-hosts=10.255.40.26:9200,10.255.36.19:92

index-one-replica

1admin$ /usr/local/python3/bin/esrally --distribution-version=5.2.1 --target-hosts=10.255.40.26:9200,10.255.36.19:92

压测结果

			multipath					HARDWARE RAID 0				
Metric	Operation	Unit	bulk 5000 shards 6 replica 0 aclient 8	bulk 5000 shards 6 replica 1 aclient 8	bulk 10000 shards 6 replica 0 aclient 8	bulk 10000 shards 6 replica 1 aclient 8	bulk 5000 shards 6 replica 0 aclient 16	bulk 5000 shards 6 replica 0 aclient 8	bulk 5000 shards 6 replica 1 aclient 8	bulk 10000 shards 6 replica 0 aclient 8	bulk 10000 shards 6 replica 1 aclient 8	bulk 5000 shards 6 replica 0 aclient 16
Min Throughput	index-append	docs/s	62656.4	45020.3	0	53944.5	8876.03	12472.9	9128.9	12662.3	8130.92	2224.99
Median Throughput	index-append	docs/s	63382.3	45767.4	0	54471.1	75947.8	92769.5	59658.5	120867	59128.2	94977
Max Throughput	index-append	docs/s	63594.5	46459.1	0	55439	76969.3	94204.3	60933.3	122877	61822.2	98738.5
Min Throughput	force-merge	ops/s	0.111095	0.0669718	0.0960337	0.06165	0.0814889	0.601039	0.317752	0.482941	0.243244	0.363038
Median Throughput	force-merge	ops/s	0.111095	0.0669718	0.0960337	0.06165	0.0814889	0.601039	0.317752	0.482941	0.243244	0.363038
Max Throughput	force-merge	ops/s	0.111095	0.0669718	0.0960337	0.06165	0.0814889	0.601039	0.317752	0.482941	0.243244	0.363038
took time		sec	152	213	127	188	133	106	156	81	157	96
node19	最低写入速率	MB/s	6.8	2.5	7.6	13	5	24.7	25	18.7	17.9	30.7
node19	最高写入速率	MB/s	18.7	492.5	151	377	100	538.9	504	498	552	545.7
node19	平均写入速率	MB/s	15	33.8	23	39	23.1	56.8	61.8	44.4	49.8	65
node26	最低写入速率	MB/s	4.7	3.3	10	12	8.8	20	19	7	22	21
node26	最高写入速率	MB/s	17.8	344	250	335	214.6	462	551	463	507	498
node26	平均写入速率	MB/s	15.1	33.8	35	39	35	42.9	61.5	59.8	68	46.5

压测总结

1. RAID0 单目录对比直通多目录上性能更好，写入效率前者是后者的 2 倍左右。
2. ES 直通多目录会导致出现热盘现象，当有多个索引同时写入时，可能会导致出现不同索引的部分分片写入到相同的磁盘上，造成单块磁盘出现瓶颈。而 RAID0 则是分散到各个磁盘。
3. RAID0 写入是异步，而直通多目录几乎是同步写入，因此性能上前者更好。
4. 压测过程中，当分片数和客户端数增加后，直通多目录方式部署的 ES 较 RAID0 方式，CPU 负载增加 20%。

综上，使用硬 RAID0 作为 ES 的存储是更好的选择。