中国民生银行天眼日志平台架构演进的平凡之路

原创：带你解密 AI前线 2017-12-07

作者 | 赵蒙、黄鹏程、文乔
编辑 | Emily

AI 前线导读: 随着中国民生银行的 IT 业务系统的迅速发展，主机、设备、系统、应用软件数量不断增多，业务资源访问、操作量不断增加，对于应用整体系统智能分析与处理的要求不断提高，急需建立包含所有应用、系统、存储、设备的统一的日志集中管理平台。本文分享了中国民生银行大数据基础产品团队如何基于 ELK 技术栈构建自己的天眼日志平台以及平台架构优化、升级和演进的过程。

更多干货内容请关注微信号“AI 前线”，ID：ai-front

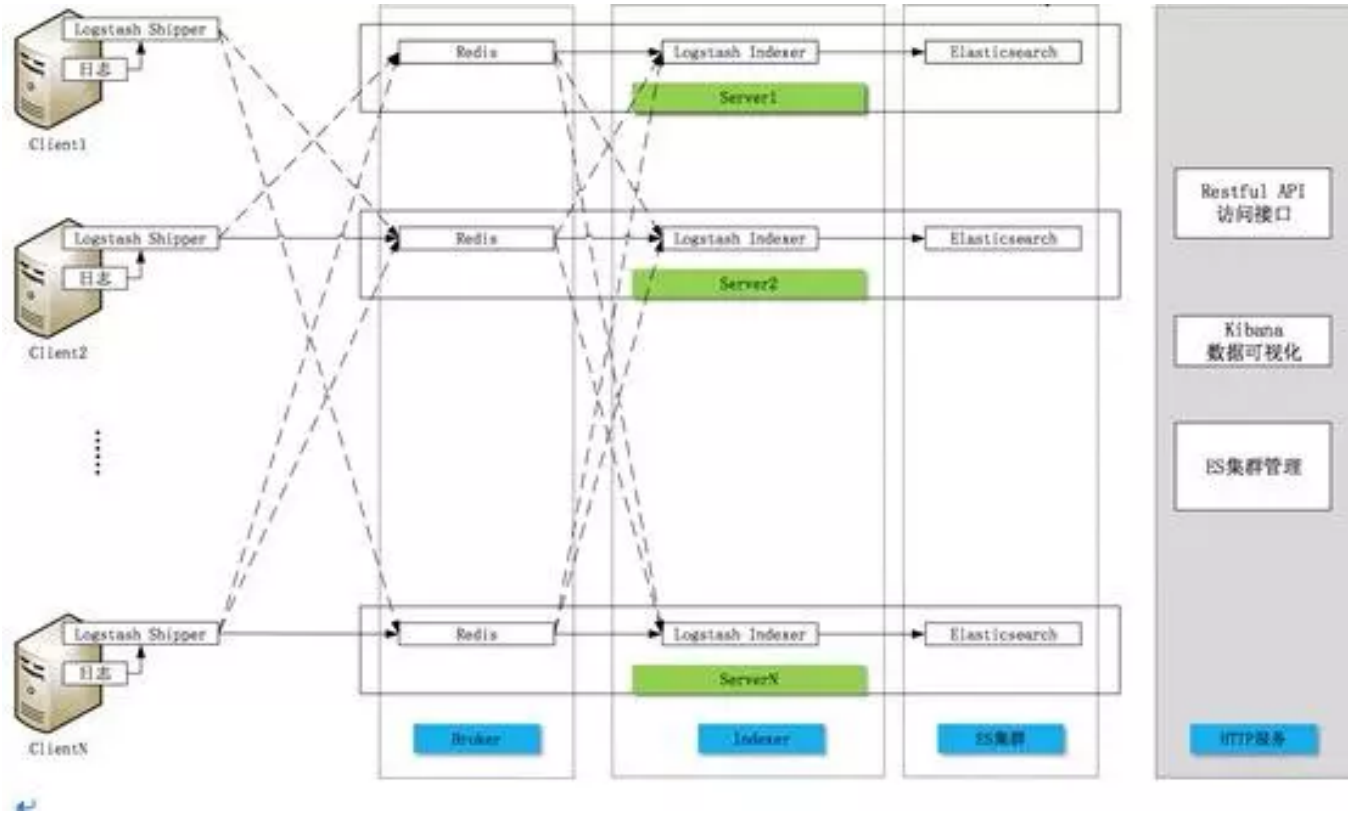
天眼日志平台功能定位

基于 ELK（Elasticsearch＋Logstash＋Kibana）建立一体化日志平台，提供日志收集、处理、存储、搜索、展示等全方位功能。通过日志的收集、传输、储存，对海量系统日志进行集中管理和准实时搜索分析，帮助运维人员进行业务准实时监控、故障定位及排除、业务趋势分析、安全与合规审计等工作，深度挖掘日志的大数据价值。

目前接入天眼日志平台的日志覆盖了应用、操作系统、数据库、中间件、存储和管理口日志数据，同时对于各个模块部分指标进行采集和存储。

早期的平台架构

早期的天眼日志平台使用了原始的 ELK 三层架构模式：多个独立的 Logstash agent(shipper) 负责收集不同来源的数据，一个中心 agent(Indexer) 负责汇总和分析数据，在中心 agent 前的 Broker 使用 Redis 作为缓冲区，中心 agent 后的 Elasticsearch（后简称 ES）用于存储和搜索数据，应用可以采用定制化的 Kibana 提供丰富的图表展示，也可以根据 ES 的 RESTful API 行开发定制。



- **采集层：** shipper 表示日志收集，使用 Logstash 收集各种来源的数据，启动时随机在 Redis 服务器列表中选择一个服务器进行对 Broker 的连接。
- **缓冲层：** Broker 作为远程 shipper 与中心 Indexer 之间的缓冲区，使用 Redis 实现，一是可以提高系统的性能，二是可以提高系统的可靠性，当中心 Indexer 提取数据失败时，数据保存在 Redis 中而不至于丢失。
- **处理层：** 中心 Indexer 也采用 Logstash，从 Broker 中提取数据，可以执行相关的分析和处理 (filter)；一个 Indexer 会轮询从多个 Redis 进行数据获取，这样防止了一个 Indexer 宕后对应的 Broker 无人处理。
- **存储层：** Elasticsearch 用于存储最终的数据，并提供搜索功能。
- **展示层：** Kibana 提供一个简单、丰富的 web 界面，数据来自于 Elasticsearch，支持各种查询、统计和展示。

经过一年多的时间，随着日志平台的发展，接入日志量呈几何级增长，日志写入请求给服务器带来很大的性能压力，同时应用运维人员对平台的需求越来越复杂和多样性，前期架构设计带来的各个层次的一系列问题都逐步暴露出来：

1. 采集层基于 Logstash 实现的 agent 平台类别受限，无法支持 AIX、HP_UNIX 等 Unix 操作系统，同时通用的开源产品 Flume 功能比较单一，无法满足我们常规的日志收集需求。
2. agent 日志解析对资源消耗较多，迫切需要将日志的分析与处理提升到后端处理层。
3. 缓冲层无法提供分布式消息队列服务，同时容量和效率亟待提高。
4. 存储层原始版本组件功能有缺陷，版本迭代较快，需要集中升级。
5. 展示层缺乏场景统一管理入口，各个应用场景相互独立不具备通用性。基于上述问题，我们设计了新的架构。

天眼日志平台目前已经接入 58 个应用系统，其中 A、B 类重要核心应用 44 个，覆盖了 500+ 的服务器，日均 5T 的数据写入量，可以很好地支持运维应用人员对日志文件进行智能分析与处理，达到了平台日志收集、处理、存储、搜索、展示等全方位功能要求。下面我们分别对这个架构中我们所做的一些工作和大家进行分享。

采集层（Agent）

适配 Unix 操作系统

在 Agent 层，为了更好地适配更多样的操作系统，我们主要采用 Logstash 和 Flume 进行日志和指标采集，在这个过程中对 Flume 进行了较多的定制，并进行了社区回馈。

首先，我们在 Linux 操作系统上采用 Logstash 进行日志的采集和初步解析。但是由于 Logstash 的 jvm 环境不是标准的 jdk，在 HP_UNIX 和 AIX 操作系统 Logstash 无法运行，所以这两类操作系统使用 Flume 进行日志采集。所有 agent 的解析文件都是通用的，Logstash 一类通用，Flume 一类通用。如果通用日志的系统上（主要是中间件）同时需要采集应用日志，Logstash 需要配置将应用日志和系统日志的采集逻辑都包含进去，原则就是一台机器采集日志的 Logstash/Flume agent 只启动一个进程。而对于操作系统的 syslog 和存储设备日志的采集方式是集中采集，使用单独的 Logstash agent 进行解析上送。

Flume 我们最初使用的版本是 Apache Flume 1.6，在使用 Taildir Source 组件和核心组件的过程中，发现其无法完全满足我们的需求，例如：

1. 若 filegroup 路径中包含正则表达式，则无法获取文件的完整路径，在日志入到 Elasticsearch 后无法定位日志的路径；
2. Taildir Source 不支持将多行合并为一个 event，只能一行一行读取文件；
3. filegroup 配置中不支持目录包含正则表达式，不便配置包含多个日期并且日期自动增长的目录，例如 /app/logs/yyyymmdd/appLog.log；
4. 在使用 Host Interceptor 时，发现只能保留主机名或者是 IP，二者无法同时保留。

在研究 Flume 源码之后，我们在源码上扩展开发。截至目前，我们为开源社区贡献了如下 4 个 Patch，其中 FLUME-2955 已被社区 Merge 并在 1.7 版本中发布。有关 4 个 Patch 的详细细节介绍请参见《拥抱开源，回馈社区：民生银行 Flume 源码贡献实践》。

另外我们在 Github 上开放了一个版本，将 FLUME-2960/2961/3187 三个 Patch 合并到 Flume 1.7 上，欢迎大家试用，地址：

https://github.com/tinawenqiao/flume，分支名 trunk-cmbc。

Flume 配置样例如下：

源端采集 Agent 轻量化

随着日志接入工作的不断推进，日志解析都是在 agent 端进行的弊病显露出来。由于 Logstash 是使用 Java 编写，插件是使用 jruby 编写，又需要 jvm 环境运行，对服务器的资源要求会比较高。同时 Logstash 在使用 filter/grok 插件处理复杂的日志字段抽取和预处理时，正则解析会耗费大量的 cpu 资源，在初始启动的一瞬间偶尔会冲破 cpu 监控报警阀值，有可能影响生产服务器。虽然目前还没发现对应用产生实际负面影响（agent 有监控脚本自杀机制），但是导致应用运维人员会非常紧张，在后来的架构演进中，我们正逐步取消源端解析而改为后台解析。

随着 Elastic 技术堆栈的发展，出现了 Filebeat。Filebeat 是 Beat 成员之一，基于 Go 语言编写，无任何依赖，比 Logstash 更加轻量，非常适合安装在生产机器上，不会带来过高的资源占用。对比测试中我们创建了 1 个 G 的日志数据，不做正则解析，在分配同样资源的情

况下，单线程 logtash 启动 cpu 瞬间飚高到 80%，后面基本上在 60% 左右，63s 处理完毕，Filebeat 启动时 cpu 瞬间在 40% 左右，之后在 20% 左右，15s 处理完毕。从结果上来看不管是性能还是资源占比 Filebeat 都比 Logstash 要好上不少。在后期演进中，我们逐步在新架构中将日志解析的工作放在了后端进行，只需要 Filebeat 将收集的日志整合原样上送即可。

Agent 统一管控和性能监控

由于上一代原始架构平台部署的 agent 缺乏统一管理功能，相关配置信息均需要手工实施，自动化程度较弱，也无法与其他系统关联。对此我们在新的日志平台架构下依赖大数据管控平台完成相关自动化需求，大数据管控平台是我们大数据基础产品团队自主开发的一套对大数据集群和天眼日志平台中 agent 统一运维管控的项目，具备智能服务发现和服务器管理功能。大数据管控平台上线实现了日志平台 agent 的自动化部署、点击触发启停和监控可视化，监控可视化通过 Filebeat 和 Flume 上送心跳信息到 Kafka 上，在 Storm 集群上开发拓扑程序消费 Kafka 心跳信息存入到大数据管控平台的 MySQL 数据库中进行展示。另外通过大数据管控平台与工单系统、服务目录、CMDB 系统进行联动，日志平台本身只充当基础框架，统一认证权限添加、元数据信息下发、工单数据流处理都通过管控平台页面 agent 配置文件的集群模块化录入和上传来实现。

在 Agent 资源消耗方面，除了进行必要的优化手段外，我们还在部署 agent 的同时配套配置 agent 进程监控，由集中监控平台进行统一部署，同时在 agent 端部署一个 crontab 脚本，发现监控报警后即刻将占用较高资源（cpu、内存、存储）的 agent 进程杀掉，避免采集 agent 对应用系统性能造成影响。

明确日志规范

在应用日志规范上，我们规定应用日志中要求必须带时间戳，这个时间戳在 agent 进行采集时会作为默认的 @timestamp。Logstash agent 需要增加应用英文简称，Flume agent 需要使用 avro 序列化方式在 head 头里增加 appname、hostname、path 构成数组传给 Indexer 进行反序列化。自采集的操作系统指标会带上操作系统类型的字段，操作系统和存储集中后的日志需要带上主机名或者 IP 标识。目前指标数据存入到 ES 中都使用小写，后续可以根据系统人员具体需求优化解析相关日志。

缓冲层（Broker）

使用 Kafka 替代 Redis

在这一层我们主要进行了使用 Kafka 来代替 Redis 的工作。从技术上看虽然在 Elastic 早期官方的指南中推荐使用 Redis，但是后来从产品角度上来看 Redis 毕竟不是专业的消息队列，Redis 做队列最大的问题在于容量受制于内存，且单节点大内存持久化过长，且没有复制情况下整机故障时存储在 Redis 中的数据容易丢失（有副本太浪费因此起初未使用复制）。在平均每天数据量 T 级，每秒接入文档数上亿的情况下，Kafka 的吞吐量和集群模式都比 Redis 更优秀，并且 Kafka 有比较完善的高可用机制，一个 Broker 下线不影响整个集群的运行。Elastic 官方在 blog 上后边也发布了使用 Kafka 作为 ELK broker 的一系列文字。Kafka 作为分布式消息队列，能充分的利用集群资源，每个应用上传日志分配一个 topic，不同系统日志使用自己单独的 topic，Flume 和 Logstash 上送日志和指标走自己单独的 topic，一个 topic 可拥有多个 partition，并且 partition 能合理分配到每个节点上面，采集上来的日志也会均匀的放到 partition 中。

进行 Kafka 整体管控

同时对 Kafka 的 Broker、Topic、ConsumerGroup 和其 Offset 我们自己开发了相应的管控系统提供配置服务、容量性能指标收集、展示和启停维护操作等功能。

进行 Zookeeper 整体管控

我们还针对 Kafka 使用的 Zookeeper 进行了相关配置管理和性能监控功能的开发：

上述 Kafka 和 Zookeeper 的核心微服务代码已经开源，欢迎大家试用：https://github.com/gnuhpc/Kafka-zk-restAPI/tree/0.10.x

处理层（Indexer）

Logstash Indexer 后端解析

上篇提到目前我们已经开始逐步开展源端轻量化的工作，因此我们专门申请了一批 cpu 能力较强的机器用作 Logstash Indexer 日志解析服务，agent 前端计划采用 Filebeat/Rsyslog 代替 Logstash 进行日志采集，Filebeat 只做日志采集和多行匹配，日志解析处理集中到 Kafka 后的 Logstash Indexer 来做。初步架构图如下：

相关规则如下：操作系统、标准中间件、数据库运行日志和指标等由于日志规范，Logstash 一个 Indexer 就能通用处理所有 agent 端上送的数据。而应用日志由于日志格式不统一，因此不管是 Flume 还是 Logstash 采集上来的数据，在 Indexer 层上均针对一个系统使用一个（或多个，视处理能力而定）Logstash Indexer 来处理，以达到日志字段解析都在后端执行的目标。在处理数据入 ES 时统一使用按天原则，在 ES 中以应用为单位创建索引，索引按照天来拆分，同一个应用的应用日志存放在同一个 index 模式下。

同时，由于 Logstash 作为后端 Indexer 进行日志解析效率和众多 Logstash 进程管理带来的复杂度，目前我们正积极调研 Hangout

（https://github.com/childe/hangout）

携程开源，我团队为该项目的第二作者）on Docker 以及 StreamSets 替代后端 Logstash 的可能性，以便更高效灵活的处理日志。前者是一个替代 Logstash 的方案，相对 Logstash 功能简单但处理更高效，后者 Streamsets 是一种专门针对传输中数据进行优化的数据处理平台，提供了可视化数据流创建模型，通过开源的方式发行，数据收集器的生命周期可通过管理控制台进行控制，提供了丰富的监视和管理界面。但是它本身也是不支持集群模式，而且目前国内外实际运用到生产环境的案例较少，学习成本较高。

Logstash Indexer 升级

我们针对 Logstash 进行了从 2.x 到 5.x 的升级。新版本 Logstash 性能根据一些测试结果有比较大的提升。另外新版的 Logstash 提供了监控 API 接口，同时 Kibana 的 x-pack montior 也是免费的，这个对管理和查看 Logstash 状态有极大帮助，尤其是对于我们目前多 Logstash Indexer 消费 Kafka 消息的架构下。Logstash 升级相对简单一些，软件层直接进行替换即可，但是由于最新版的有不少参数和配置文件发生变化，所以需要进行目录重规划和重新配置，具体如下：

1. 与老版本不同的是 Logstash 5.X 不同的解析配置文件需要单独放在不同的目录中，因为新版本的 jvm 参数和相关配置都是写在独立的配置文件中而不是像以前作为参数传入，这样可方便对不同的 Indexer 作不同的资源分配，如 cpu、内存等等。同时由于新版本 Logstash 提供了监控 API，需要分配 http 端口进行访问，相同目录配置也会端口冲突导致 Logstash 无法启动，启动命令需要增加 --path.settings 指定到对应的不同配置目录上。

2. 解析配置文件相关参数需要调整，Logstash 新的版本的 Kafka input 插件不再支持 zk_connect、white_list 这种旧 Kafka API 的参数，必须使用新的参数去连接 9092 端口而不是 zookeeper 端口去消费数据。

3. 针对消费不同的 Kafka topic 的数据量设置不同的资源参数，通过修改 jvm.options 分配内存，通过修改 Logstash.yml 设置 cpu 资源和 batch 参数。

4. 使用升级后的 Logstash 发现生产中非 UTF-8 编码的中文日志经过 avro 序列反序列化后有乱码的问题，我们最后修改了 Logstash-input-Kafka 插件的源代码，在

   vendor/bundle/jruby/1.9/gems/Logstash-input-Kafka-5.1.8/lib/Logstash/inputs/Kafka.rb

   中修改提供了两个新选项：

(1) charset 指定原先日志的编码，默认不设置为 UTF-8

(2) charset_field 是一个数组，指定哪些 field 需要转码，默认为空，也就是都不转换。

具体代码参见

https://github.com/logstash-plugins/logstash-input-kafka/pull/222

下篇我们将介绍我们的存储层（Storage）和展示层（Presentation）的技术架构以及我们的工作，最后我们将展示目前的应用场景并做出总结。另外中国民生银行数据呈井喷式增长，ELK、Hadoop、Spark 大数据相关技术人员急缺，我行官网正诚招有志于投身到银行大数据行业并专注于技术的小伙伴，也欢迎联系关注。

作者介绍：

赵蒙， 工作于中国民生银行总行信息技术部大数据基础技术平台和产品组，天眼日志平台负责人，专注于全行分布式日志平台的建设以及 Elasticsearch 在银行的应用方案落地实施。

黄鹏程， 工作于中国民生银行总行信息技术部大数据基础技术平台和产品组，团队负责人，负责 Hadoop 平台的规划建设和维护工作，参与天眼日志平台和大数据管控平台，微信 gnuhpc。

文乔， 工作于中国民生银行总行信息技术部大数据基础技术平台和产品组，负责行内大数据管控平台的设计开发，同时参与 Elasticsearch 技术工作。她在 Flume 上亦有所深入。

中国民生银行天眼日志平台架构演进的平凡之路（下篇）

存储层（Storage）

我们的存储层采用 Elasticsearch。采用热数据进入 SSD、温数据进入 SATA 盘，冷数据 snapshot 至 HDFS 的层次化存储结构。最早天眼日志平台 Elasticsearch 和 Logstash 使用的是 2.3.5 版本，基于 Kibana4 作可视化展现。由于 ES 社区非常活跃，ELK 版本发布特别频繁，在不到一年的时间里已经从 2.X 跳到了 5.X。5.X 版本的 lucene 更新到了 6.2，搜索方面应该会有比较大的性能提升。同时官方推荐新的 ES 性能更加稳定，ELK 整个技术栈产品都有很大的变化和功能的扩展。升级时官方资料显示 ES 5.X 已经比较稳定，我们观察一段时间后最后我们使用的版本是 5.5.0 版本，另外 5.X ES 优化了 indexing 也是我们很期待的。

ES 集群升级

采用官方提供的离线升级方式，直接用新版本软件替换掉老版本并保证原数据可用。升级前使用 ES migration 迁移插件检查升级前潜在的问题

A. node 属性

新版本一些 node 属性名称发生了变化，如 node.box_type 改成 node.attr.box_type。新版本没有 client node 了，分为 master node，data node，ingest node，coordinating only node，设置方法分别是：

B. index settings

ES 从 5.0 开始 index 相关参数均不在 config 文件里配置，而需要在集群索引级别中进行设置（除了 index.codec 等类似实例级别可以在 node 级别设置）。注意执行下面这条语句会报错，因为这几个参数都是不能动态修改的。

C. 重命名设置

5.X 参数名称变化较大，bootstrap.mlockall 改为 bootstrap.memory_lock，discovery.zen.initial_ping_timeout 改为 discovery.zen.ping_timeout，默认是 3s，新版本去掉了 discovery.zen.initial_ping_timeout 和 discovery.zen.ping.timeout 两个设置。

D. 参数变化

discovery.zen.ping.multicast 参数已经废弃，在 5.X 里去掉了多播，用单播或者 cloud discovery 插件，action.disable_delete_all_indices 改成 action.destructive_requires_name，需要用 cluster update API 修改：

5.X 里没有这个 path.work: 配置，ES 5.X 里 translog 的 flush 参数只有

index.translog.flush_threshold_size。

ES 升级步骤大致如下：

1. 暂停 Logstash Indexer 日志数据写入
2. 关闭集群 shard allocation
3. 手动执行 POST /_flush?wait_for_ongoing，等到操作结束再返回

4. 手动执行 POST /_flush/synced
5. 关闭所有 ES 集群节点
6. 执行新版本 Elasticsearch 软件替换
7. 启动所有 ES 集群节点
8. 重新开启集群 shard allocation
9. 等待 recovery 完成，集群 health status 变成 green
10. 重新开启 Logstash Indexer 日志数据写入

ES 集群升级以后一些相关插件和工具也需要捆绑升级，head 插件需要升级到 5.X，原来的 kopf 可视化监控插件不再可用，我们使用 cerebro 代替 kopf，cerebro 和老版本的 kopf 插件页面几乎一模一样，功能上可以完全替代。同时我们计划采用最新版本引入的 cross-cluster 来实现跨中心跨集群访问功能。

IK 兼容性

升级过程中当然不可能是一帆风顺的，期间遇到了很多问题，现分享一个比较有代表性的“坑”给大家分享。

ES 新版本程序替换重启后，状态一直是 red，查看日志，有大量关于 ik 的报错，找不到 ik 的分析器，如下所示：

在升级之前，我们已从 Github 的 ik 项目上得知，在 5.0 之后，ik 被改名，截图如下：

但是对于已有的索引，某些字段的分析器指定的是 ik，例如上图报错中索引 l-Kibana-2017.08 是升级前建立的索引，_all 字段指定的分析器是 ik，而升级后更名为 ik_smart，因此报找不到 ik 分析器的错误。虽然关闭索引之后修改索引的 analyzer 也可以实现分词器名称的修改（在不关闭索引的情况下直接修改分析器会报错），但由于索引众多，为了更快的实现兼容，我们修改 Elasticsearch-analysis-ik 源码解决，使得最新版 ik 插件也能支持名为 ik 的分析器。

(https://github.com/medcl/Elasticsearch-analysis-ik/pull/411/commits/63326ca322ccb8c1bd3662ab7c0bfd38a86a53cb)

存储层各个组件升级以后效果明显，最直观的感受就是之前 master 节点随着时间的推移 heap 内存使用率持续增长的问题没有了。实际上存储层不仅仅是对 ELK 进行了升级，还进行了调整底层索引组织结构，对冗余数据进行清理，开发常用工具脚本，小规模资源扩容等等一系列工作，所以经过测试和实际使用评估，升级后平台更加稳定，查询更加高效。

冷热数据控制

同时针对日志量数据量激增的情况，新架构引入了 SSD 来提升 ES 的读写性能。单台 ES 存储有 2 块 SD 盘和若干 SATA 盘，所以每台 ES server 都启动了 3 个 ES 节点，2 个 hot 节点和 1 个 warm 节点。Indexer 中指配置了 hot 节点的端口，通过 ES 中的模板定义保证实时数据只写入 hot 节点。

通过 ES 官方推荐的 curator 工具定时将数据从 hot 节点搬迁到 cold 节点，SSD 数据保留周期为一周。curator 的 action 配置文件如下：

生命周期管理上天眼日志平台实施日志数据定期导入到大数据平台 hdfs 策略，日志数据保留一个月，一个月前的数据根据定时任务定期导入到大数据平台中，ELK 中不再保存，大数据平台的日志、指标数据保存期限是一年。注意：数据冷热分离过程中使用到的 curator 也必须使用最新的版本才可以使用，在集群升级完毕以后也需要重新安装。

展示层（Presentation）

Kibana 升级

Kibana5 可视化方式更加灵活、丰富，提供了更加多的组件和监控可视化视图，更多的功能和更好的用户体验对于平台使用者有极大的吸引力。Kibana 升级比较简单，但是由于默认新版本无分词的 raw 字段已经变成了 keyword 字段，所以 Kibana 的 indices 属性需要刷新，同时之前建立的 visualize 也需要进行字段属性调整。新版本的 Kibana 确实增加了丰富的视图和展现功能，页面显示也更加美观。

多租户访问控制和数据脱敏显示

由于 X-pack security 模块是收费的，我们通过自主开发的方式使用 Kibana proxy（地址 https://github.com/gnuhpc/Kibana-multitenant-proxy）实现了 Kibana 权限控制和数据显示脱敏。目前权限控制到索引层，即一个用户只可以访问指定的 Index，如果访问其他 Index 则在 Kibana 上无法显示。同时由于银行业的数据敏感，我们还提供了在配置文件中设置脱敏关键字，日志入 ES 时不脱敏，在 Kibana 上查询时加密显示的功能。Kibana-proxy 脱敏显示的效果如下：

应用场景

日志定位搜索

能通过关键字和简单符号解决搜索问题，避免使用复杂正则，有比较好的用户搜索体验。当多台服务器输出日志的时候，需要能够快速发现这个究竟请求落到哪台服务器上，报了什么

样的错误，为什么其他服务器上不会报错等问题。有时也需要同时能够平行关联查询某些服务器日志才能做出问题判断，比如同时查询主备库日志来判断某一点时刻数据库是否同步。

运营分析支持

对日志数据进行挖掘，提取有价值的数据形成图表进行展示。Kibana 提供了丰富的可视化图表展现，方便从应用角度对于业务系统的总体日均访问量情况、重要功能的访问进行统计；从交易角度显示系统的总体交易量、交易成功率和延迟，多维度支持业务运营工作。

监控统计分析

定期对告警数据进行分类统计分析，为监控应用程序性能预估、流程监控、任务流检测、推广部署等提供数据支撑。对于生产环境的相关技术组件覆盖情况进行统计、趋势等分析，多维度了解各种技术组件生产实际使用情况。

应用统一分析视图

在 Kibana 升级后我们在 dashboard 之上封装了一层作业总体概况显示，将 A、B、C、D 类系统接入情况进行统一展示，形成以应用为维度的视图分析模板，新接入应用只需要直接套用模板即可无需进行单独配置，模板包含业务交易、操作系统、中间件、数据库等日志统一信息，扩展成为应用系统一分析视图。

总结与展望

经过不到两年的建设，通过不同的架构调整和设计开发，我们基于 ELK 的日志平台对于如下功能目标均按预期实现：

- **数据定位准确：**通过对日志进行细颗粒度字段解析尽可能满足实现不同场景下日志集中存储和管理的业务需求，方便进行查询。
- **写入和查询高效：**通过对 ELK 平台升级和集群内存调优、合理的分片配置、冷热数据分离最大程度地提高日志的写入和查询效率。
- **高可用部署：**通过 ELK 平台集群高可用设计部署实现故障切换时日志平台系统自身可用，服务不中断。
- **安全可靠：**通过自主开发权限控制和数据脱敏，保证了平台日志数据的安全性和可靠性。

架构持续在演进，技术永远在前行。客观地讲，目前该平台的每一次架构变迁并非是最正确的选择或者是最优的解决方案，如何让“平凡之路”走得不平凡，我们民生银行大数据基础技术平台和产品团队一直在孜孜不倦地探索中。我们的终极目标就如平台的名字一样，可以让开发和运维工程师随时主动通过“天眼”实时查看系统状况，对系统情况了如指掌，对事故隐患明察秋毫，对性能容量成竹在胸，促使通过天眼日志平台进行日志接入和管理成为生产运维的重要组成部分。另外中国民生银行数据呈井喷式增长，ELK、Hadoop、Spark 大数据相关技术人员急缺，我行正在官网上诚招有志于投身到银行大数据行业并专注于技术的小伙伴，也欢迎联系第二作者微信交流关注。

作者介绍：

赵蒙， 工作于中国民生银行总行信息技术部大数据基础技术平台和产品组，天眼日志平台负责人，专注于全行分布式日志平台的建设以及 Elasticsearch 在银行的应用方案落地实施。

黄鹏程， 工作于中国民生银行总行信息技术部大数据基础技术平台和产品组，团队负责人，负责 Hadoop 平台的规划建设和维护工作，参与天眼日志平台和大数据管控平台，微信 gnuhpc。

文乔， 工作于中国民生银行总行信息技术部大数据基础技术平台和产品组，负责行内大数据管控平台的设计开发，同时参与 Elasticsearch 技术工作。她在 Flume 上亦有所深入。

今日荐文

点击下方图片即可阅读

AI 培训班有存在的必要吗?

活动推荐

12 月 8 日 -9 日 ArchSummit 全球架构师峰会将在北京国家会议中心举办，极客时间将同步免费直播大会主题演讲和大前端技术与管理专题，识别下图二维码，下载极客时间 App 还可免费获取 ArchSummit 全球架构师峰会全部 PPT。

关注极客时间服务号（ID: jikeshijian），回复“AS”，抢先加入直播交流群。