沪江网日志平台化之路

——曹林华　沪江　资深架构师

大家好，我今天的分享是偏应用的，主题是沪江日志的平台化之路。我叫曹林华，来自沪江平台架构部，任资深架构师，主要负责沪江的搜索、日志、实验、全链路跟踪平台。比较擅长的是在分布式、高并发，以及大数据等技术难题上。大家有兴趣的话可以在我的个人博客 caolinhua.me 上交流这方面的经验。

沪江是做在线教育的，峰值的日志量达每秒数万以上，每天的索引量高于 10 亿条，每天的存储是 1TB 以上，接入的业务线有 20 多条，业务线分门别类，也是百花齐放的感觉。

首先介绍一下今天演讲的背景。沪江的日志平台是从 0 到 1，背景介绍会提及沪江为什么要做日志平台，要达到什么效果，以及对应的解决方案。第二是架构演进，是讲沪江的日志平台从 0 到 1 的演进过程。第三是最佳实践，沪江的日志平台有些地方和业界不一样，这跟业务特点有关。最后一个是业务价值，也就是我们的日志平台对业务产生了多少价值。

背景介绍

大家可以先想一下，做平台化的应用会遇到什么问题呢？

第一个问题是接入的业务线特别多，这会遇到什么问题呢？采集是技术层面的问题，有一个组织架构层面的问题更加严重。沪江的日志平台，只有我和另一个同事两个人负责。我们公司的开发人员有三四百人，如果一对一和他们沟通交流，要从早排到晚。其实这是一个组织架构的问题，平台架构是一个横向支撑的部门，做平台化的应用会涉及到地位的转变。

第二个是我们的日志标准不一致。业务线很多，每个团队使用的语言也不一样，前端是 JSON、Node.js，后端是 Java，DevOps 基本上是 Python，这就导致日志的格式不一样。更别说还有 Node.js 这种很多坑的语言 。

第三个是延时问题。前期日志量小时还好，后期各种业务线接入，延时就比较严重。有一天凌晨三四点，日志突然又延时了，只能起来一顿操作，操作完已经六点了，七点又起床上班去了。

讲了这么多，技术层面的解决方案是什么？很简单！业界主流的日志分析方案是 ELK，这里再加上 Beats。这些组件的功能不一样，Beats 是做数据搜集这一块，Logstash 是做数据的聚合和简单处理，Elasticsearch 是做索引和存储，最后是 Kibana 做一些界面的自动化，使图表看起来特别的炫。我们的日志平台是以这个为基础，基于这些做一些迭代和优化。

背景差不多了就是这些，重点是——两个平台维护人员，四五百号开发，怎么办？对，是有一个培训的！不过这其中还是有一个演进的过程。

下面说一下沪江日志架构平台从 0 到 1，再从 1 往后迭代的过程。

架构演进

简单版架构

开始是一个最简单的架构。没有搭建过日志平台的，也可以用这个做一下试手。

首先是日志采集，然后是 Logstash，包含 Input、Filter、Output，最终数据直接进入到 ES 里面，在用 Kibana 做展示。这个架构的问题之一是处理慢。我们有上千台物理机，物理机上做 KVM 虚拟化，虚拟出更多的机器。如果每一台都安装一个 Logstash

agent 发送数据到 ES，ES 服务器吃得消吗？

集群版架构

于是，基于简单版，我们做了一个集群版架构，加了一个缓存队列，也就是 Kafka。原来的日志搜集变成了 Shipper 集群，我们这边的上的比较早，用的是 Logstash 来做。日志搜集后，经 Kafka 到 Indexer，Indexer 也就是 Logstash，用来做一些数据的转化和切割。

是不是到了这里就没什么问题了呢？没错，Logstash 太重了！

引入 Filebeat

后来我们把 Logstash 换掉了，引入了 Filebeat。Filebeat 是纯 Go 语言写的，不需要像使用 JRuby 语言编写的 Logstash 一样运行在 JVM 上，而且它本身安装包也小。

开始我们不敢直接这么上，于是就做了一套压测方案。压测环境是 8 核 64G 的虚拟机，内存是 540G 的 SATA 盘，Logstash 版本是 2.3.1，Filebeat 版本是 5.5.0。压测方案是读取 350W 条日志数据，单行的数据大小为 0.5KB，有 8 个进程同时写入文件。压测结果显示，Logstash 的 CPU 占用达 57.7%，耗时 210s，并发是 1.6W，Filebeat 的 CPU 占用达 38.0%，耗时 30s，并发是 11W。这里的 Logstash 和 Filebeat 的压测没有做任何调优。当然，在实际环境中的平台型的项目，需要根据自己物理环境和业务来做一些调优。

整体架构

沪江整体架构和集群版架构的最大区别，是在日志搜集上。运维出身的，使用 ELK 做日志分析比较多，关注更多的可能是服务器的层面，用户的层面可能不是特别关注。我们其实是需要关注的，我是开发出身，对业务相对敏感一点。

我们做了一个 Log Center，也就是日志网关。我们通过日志网关搜集 IDC 外网的所有日志数据，包括 App 移动应用端、CDN、HTML。稍微大一点的互联网公司，所有的页面都会经过 CDN，CDN 用来优化链路、做静态化；搜集前端的数据，可以用来分析前端页面的快慢及错误。日志网关的数据、安全数据、MySQL 的数据、还有最重要的应用服务器的数据，也是通过 Filebeat 来搜集，然后传到 Kafka 里面。

Kafka 后面做冷热分离。为什么做冷热分离呢？一般 ES 会存大量的数据，这样 Kibana 展示的时候会特别累，用户看到了界面上功能众多，就可能会去点击，一点就导致 ES 集群挂了。所以后来超过一周的数据，我们就给放到 Hadoop 数据仓库里，查询时使用 Hive，不超过一周的数据还是使用之前的方案。一般情况下出现的问题，都是最近发生的，基于成本考虑，还是这套冷热分离比较合适。

ES 优化

说了整体的架构，下面我们着重说一下 ES 本身的优化。我这边主要是基于开源的 ES 做了一些调优，一个最重要的措施就是 Master 和 Data 分离。不混在一起用是为了责任单一，Master 只做路由转化，Data 只做数据存储。大数据量的场景，或使用 ES 做搜索时，都建议采用这种方式。因为搜索的场景里，词库的热加载、分词等占有大量资源，在线重启时对业务是无感知的。

这里我稍微介绍下 ES 优化的几个小建议。第一个就是冷热分离，**Hadoop 和 ES 分开存储**，也是沪江一个不一样的价值点；第二是 master node 和 data node 分离，各自责任单一；第三是增大 refresh interval 时间；第四是减少 mumber of replicas；第五是 ES 内存大小的设置，官网一直建议 ES heap size 小于 32G ；第六个是 filter 和 query 的区别，在日志平台 query 评分的用处不大，基本能用 filter 就用 filter；最后，有钱还是上SSD。

这些优化在我的博客中都有提及，都是可以直接配置的，大家可以修改下，看看效果怎么样。

最佳实践

第三部分是最佳实践，这里说下前端的最佳实践。前端做日志的话，一般会怎么做？埋什么点，怎么埋？其实前端埋点，大家可以看下 Windows.performance 这一款 API。我所了解的几乎所有的前端性能监控、所有的响应页面监控，都是基于 Windows.performance 来做。Windows.performance 把整个链路都给探出来了，网络这一块有对应的接口，Response 里有 ResponseStart 和 ResponseEnd 等信息，Client 这边有 domo、渲染时间等信息。如果有需求的话，大家可以直接使用 Windows.performance 获取相关数据。

在 APP 中，我们基于 connect、response 等数据做了一些响应时间的监控，关注比较多的是 Average load time 和 Average ready time 这两个指标。load time 渲染完成的时间，一般都是在两秒左右，ready time 都是在1秒以下。这实际上就可以监控用户打开网页或者 APP 的响应时间。

业务价值

做任何项目都要产生对应的业务价值，但业务价值要怎么样体现？这是一个比较难的问题。从技术的层面考虑，就是搜集所有的日志，保证延时很少。但这是针对开发层面的，怎样对业务有促进作用，怎样和老板汇报呢？这是需要思考的。

在我看来，日志分析对业务产生的价值分为几个方面。

首先是**移动 APP 服务质量**。从技术层面考虑，手机端哪一块的质量需要关注的多一点？crash 是一个非常重要的指标！我们基于 ES 自己做出了各应用模块错误占比分析，也就是 crash 率的监控页面，因为 Kibana 和业务指标没有任何耦合，我们只能自己来做页面，拉 ES 的数据进行展示。这个页面，开始我们使用外界的一些工具来做，后来是自建。对于中小型的公司，用外界的工具还是比较划算的。

其次是 **Web 端的服务质量**。关于 Web 端服务质量，ready time 和 load time 是比较重要的二个指标，监控的话需要粒度更细，一般要注意 ready time 大于 2 秒，load time 大于 4 秒的情况。这里的数据是从另外一个 API 调的，也是考虑到实际业务的情况

的。

然后是 **CDN 的服务质量**。我们用的 CDN 比较多，像腾讯、百度、网宿都用过，每一家的 CDN 都有自己的特点。大家有遇到 CDN 挂了，业务这边无可奈何的情况吗？我们遇到过了一个 CDN 的问题，一个用户的页面访问不了，当时我们服务器都是好的，让用户换手机、换浏览器都没用，因为用户所在区域都是这个情况。后来我们跟 CDN 厂商沟通，拉 CDN 本身的日志才排查出了是 CDN 的问题。得出的经验就是，做任何服务，无论是不是第三方的，都一定要有一个闭环，要有一个指标来判断这个产品运行的效果。

后来我们就把 CDN 的数据拉回来，放到了我们的日志平台来做。我们的服务以 HTTP 为主，CDN 错误类型基本上以 5XX 为主，监控也是以这些为主。

还有**后端服务质量**。后端会做哪些业务价值？一般是关注两个维度，第一是看请求是否成功，第二是看这个请求的快慢。通过这样呈现的报表，可以反推到业务线，就可以知道每天的业务数据如何，以及业务产出是否良好。

大家对这一块有什么问题吗？

Q&A：

Q：这些字段是怎么写入日志的？

A：其实我们所有的服务都有一个内部网关，是基于 OpenResty + Lua 做的内部路由网关，所有的基层转发都会经过内部路由网关，转发会记录日志，日志里面会有 response time、request time、http stats 等信息，我们将 OpenResty 日志拉到日志网关里来。这还是需要基于基础设施，而不是直接拉某个服务的业务日志来做。这个 OpenResty 日志是直接落到磁盘里面，然后通过 Filebeat 导出。

Q：这样会不会占有很多网络带宽？

A：我们是通过多机房、多 IDC 里部署的，多 IDC 里面的数据传输，会涉及到 VPN ，VPN 对带宽是有限制的。我们做了多 IDC 的数据闭环，是直接拿数据到每个 IDC 里面去查，如果是在 IDC1，就直接到 IDC1 数据仓库里去查询，每个 IDC 里都有一套

ES 集群，不是跨专线来进行数据传输的。机房里我们都是万兆网卡，在会话层或接入层基本上无压力。

提问：多 IDC 做了查询，这个查询结果怎么合并？

曹林华：我们是每天凌晨到机房批量拉 ES 数据，然后汇总数据到 MySQL 里面进行分析。我们也做告警，但告警是基于 OpenResty 日志的关键字，或者基于 5XX 的错误来做，ES 则是在每个机房内，是基于 API 来报警的。另外，我们是在界面上查询，通过日志 services 路由到对应的 IDC 机房里面的 ES 集群里去。统计上，我们基本是以接口的维度来定义，比如说最近十分钟内某个接口挂了，就可以基于 OpenResty 日志，统计下对多少用户有影响之类的。

提问：响应的时间，有没有一个 P99 或 P999 这样的指标？

曹林华：基本我们是 P99 为主，它是在批量 job ，也就是每天凌晨算出这块的数据报表。看实时的话，是通过 Kibana 到每一个界面上去看。

提问：Filebeat 搜集日志时，可能落盘速度比网络速度要快，这样会不会堵在哪里？

曹林华：上面 Filebeat 与 Logstash 压测的数据是 11W 左右，我们这里的并发量数据没这么大。你说的问题可能存在，但我们业务目前还没有这个问题。落磁盘有问题的话，OpenResty 本身就有问题了。单台 OpenResty 的 QPS 我们进行过压测，差不多是有四五万，使用的是内存 128G，64核的配置。

提问：ES 的权限问题是怎么解决的？

曹林华：这是一个痛点，我们现在用的还是原生的 ES 的权限，很多的人都问我要权限。这块我们还在进行规划，这个问题还没有解决。

提问：日志的采集经过这么多的环节，如果是在服务端感觉日志堵了，怎么判断堵在什么环节？

曹林华：我们可以切分粒度，比如通过 Kibana 查看最近的业务线，没有日志的话，是最近的五分钟，还是最近的一分钟没有日志，基于这个来报警，我们就可以知道是哪一个业务线，哪一台机器了。报警这块，我们是通过全自动来做的。

提问：我们的日志级别比较高，要求日志不能丢。我们要校验 Filebeat 采集的每一条日志是不是落到 ES 里面！我们是属于券商，需要进行日志审计。

曹林华：那可以用我们日志易的产品。

主持人：从根本来说，ES 从来没有讲过可以保证 100% 的数据完整性。要保证日志不能丢，监控的指标是可以做到很细的。我之前在微博的时候，当时一天有 2000 亿条的日志传输，在这个过程中，我们一条 ELK 传输系统的监控项指标就有20万个。为了保证一条日志处理系统的健康，花在监控上的成本，跟监控业务系统是相等的。要做好这个就要有投入，我当时的团队有 11 个人，才可以监控的这么细，只有两个人的话，肯定是没办法做到这种程度的。

所有的分布系统都是类似的，日志多发几遍，丢失率肯定就低。这对于 ELK 来说，是去重的问题。运维上很多问题都是，性能和实时很难两全，想保证数据完整，就会造成性能上的下降。我们上面做的那么细，ES 的性能却下降了 30% 左右。这也是一个取舍问题。

谢谢大家留到这么晚！