用大数据思维做运维监控

今天一大早就看到了一篇文章，叫【大数据对于运维的意义】。该文章基本上是从三个层面阐述的：

1. 工程数据，譬如工单数量，SLA可用性，基础资源，故障率，报警统计
2. 业务数据，譬如业务DashBoard,Trace调用链，业务拓扑切换，业务指标，业务基准数据，业务日志挖掘
3. 数据可视化

当然，这篇文章谈的是运维都有哪些数据，哪些指标，以及数据呈现。并没有谈及如何和大数据相关的架构做整合，从而能让这些数据真的变得活起来。

比较凑巧的是，原先百度的桑文峰的分享也讲到日志的多维度分析，吃完饭的时候，一位优酷的朋友也和我探讨了关于业务监控的的问题。而我之前发表在肉饼铺子里的一篇文章【大数据给公司带来了什么】 也特地提到了大数据对于整个运维的帮助，当时因为这篇内容的主旨是罗列大数据的用处，自然没法细讲运维和大数据的整合这一块。

上面的文字算引子，在步入正式的探讨前，有一点我觉得值得强调：

虽然这里讲的是如何将大数据思维/架构应用于运维，平台化运维工作，但是和大数据本质上没有关系，我们只是将大数据处理的方式和思想应用在运维工作上。所以，即使你现在所在的公司没有数据团队支撑，也是完全可以通过现有团队完成这件事情的。

运维监控现状

很多公司的运维的监控具有如下特质：

1. 只能监控基础运维层次，通过zabbit等工具提供服务器,CPU,内存等相关的监控。这部分重要，但确实不是运维的核心。
2. 对业务的监控是最复杂的，而现在很多公司的要么还处于Shell脚本的刀耕火种阶段，要么开发能力较强，但是还是东一榔头西一棒子，不同的业务需要不同的监控系统，人人都可以根据的自己的想法开发一个监控的工具也好，系统也好，平台也好。总之是比较凌乱的。
3. 使用第三方的监控平台。这个似乎在Rails/NodeJS/Pythone相关语系开发的产品中比较常见。我不做过多评价，使用后冷暖自知。

当然也有抽象的很好的，比如点评网的运维监控据说就做的相当好，运维很闲，天天没事就根据自己的监控找开发的搽，让开发持续改进。不过他们的指导思想主要有两个：

1. 运维自动化。怎么能够实现这个目标就怎么搞，这严重依赖于搞的人的规划能力和经验。
2. 抽象化，根据实际面临的问题做出抽象，得到对应的系统，比如需要发布，于是又发布系统，需要管理配置文件，所以有配管系统，需要日志分析所以有了有日志分析系统。然而这样是比较零散的。

有点扯远，我们还是focus在监控上。

如果以大数据的思维去思考，我们应该如何做好监控这件事情?

罗列出你的数据源

【大数据对于运维的意义】 这篇文章也讲了，主要有工程数据，业务数据。所有的数据源都有一个共性，就是日志。无论文本的也好，二进制的也好。也就是我们的数据源就是日志：所有业务的，所有服务器自身的系统日志。直观的感受是，一旦出了问题，你的第一反应是查日志。所以日志是整个信息的源头。

从日志我们可以挖掘出什么?

我觉得抽象起来就一个： 指标。 指标可以再进行分类，

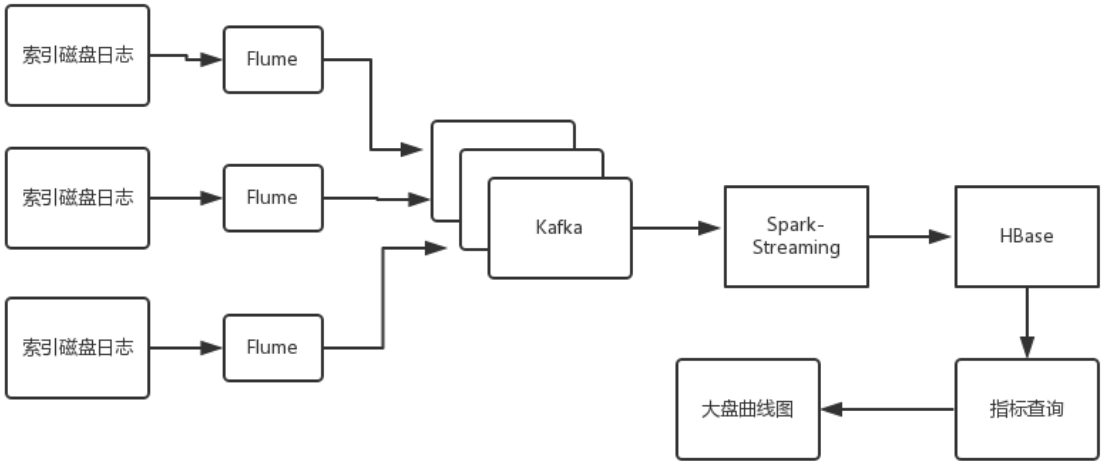
1. 业务层面，如团购业务每秒访问数，团购券每秒验券数，每分钟支付、创建订单等
2. 应用层面，每个应用的错误数，调用过程，访问的平均耗时，最大耗时，95线等
3. 系统资源层面：如cpu、内存、swap、磁盘、load、主进程存活等
4. 网络层面： 如丢包、ping存活、流量、tcp连接数等

每个分类里的每个小点其实都是一个指标。

如何统一实现

千万不要针对具体问题进行解决，大数据架构上的一个思维就是：我能够提供一个平台让大家方便解决这些问题么？ 而不是，这个问题我能解决么?

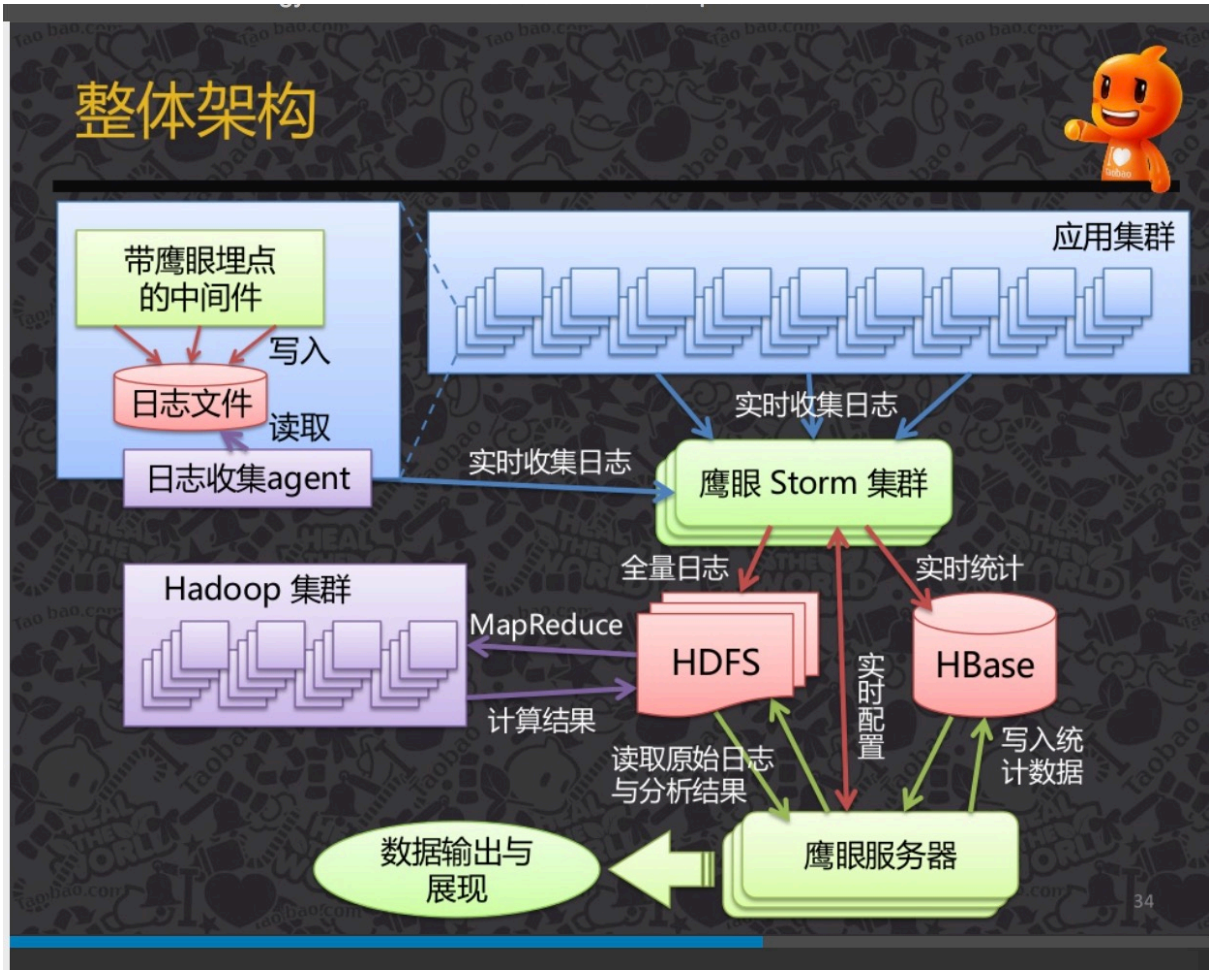
先来看看架构图：



因为目前我负责应用层的研发，业务还比较少，主要就需要监控三个系统：

1. 推荐
2. 搜索
3. 统一查询引擎

所以监控的架构设计略简单些。如果你希望进行日志存储以及事后批量分析，则可以采用淘宝的这套架构方式：



稍微说明下，日志收集Agent可以使用Flume,鹰眼Storm集群，其实就是Storm集群，当然有可能是淘宝内部Java版的， Storm(或第一幅图的SparkStreaming)做两件事情

1. 将日志过滤，格式化，或存储起来
2. 进行实时计算，将指标数据存储到HBase里去

到目前为止，我们没有做任何的开发，全部使用大数据里通用的一些组件。至于这些组件需要多少服务器，就看对应的日志量规模了，三五台到几百台都是可以的。

需要开发的地方只有两个点，有一个是一次性的，有一个则是长期。

先说说一次性的，其实就是大盘展示系统。这个就是从HBase里取出数据做展示。这个貌似也有开源的一套，ELK。不过底层不是用的HBase存储，而是ES。这里就不详细讨论。

长期的则是SparkStreaming(淘宝是使用Storm，我建议用SparkStreaming,因为

SparkStreaming可以按时间窗口，也可以按量统一做计算)，这里你需要定义日志的处理逻辑，生成我上面提到的各项指标。

这里有一个什么好处呢，就是平台化了，对新的监控需求响应更快了，开发到上线可能只要几个小时的功夫。如果某个系统某天需要一个新的监控指标，我们只要开发个SparkStreaming程序，丢到平台里去，这事就算完了。

第一幅图的平台我是已经实现了的。我目前在SparkStreaming上只做了三个方面比较基础的监控，不过应该够用了。

1. 状态码大盘。HTTP响应码的URL(去掉query参数)排行榜。比如你打开页面就可以看到发生500错误的top100的URL，以及该URL所归属的系统。
2. 响应耗时大盘。URL请求耗时排行榜。比如你打开页面就可以看到5分钟内平均响应耗时top100的URL(去掉query参数).
3. 还有就是Trace系统。类似Google的Dapper,淘宝的EagleEye。给出一个唯一的UUID,可以追踪到特定一个Request的请求链路。每个依赖服务的响应情况，比如响应时间。对于一个由几个甚至几百个服务组成的大系统，意义非常大，可以方便的定位出到底是那个系统的哪个API的问题。这个最大的难点是需要统一底层的RPC/HTTP调用框架，进行埋点。因为我使用的是自研的ServiceFramework框架，通讯埋点就比较简单。如果是在一个业务线复杂，各个系统使用不同技术开发，想要做这块就要做好心理准备了。

现在，如果你想要监控一个系统是不是存活，你不在需要取写脚本去找他的pid看进程是不是存在，系统发现在一定的周期内没有日志，就可以认为它死了。而系统如果有异常，比如有大量的慢查询，大盘一定能展示出来。

描述到这，我们可以看到，这套架构的优势在哪：

1. 基本上没有需要自己开发的系统。从日志收集，到日志存储，到结果存储等，统统都是现成的组件。
2. 可扩展性好。每个组件都是集群模式的，没有单点故障。每个组件都是可水平扩展的，日志量大了，加机器就好。
3. 开发更集中了。你只要关注日志实际的分析处理，提炼指标即可。

大数据思维

对于运维的监控，利用大数据思维，需要分三步走：

1. 找到数据
2. 分析定义从数据里中我能得到什么
3. 从大数据平台中挑选你要的组件完成搭积木式开发

所有系统最可靠的就是日志输出，系统是不是正常，发生了什么情况，我们以前是出了问题去查日志，或者自己写个脚本定时去分析。现在这些事情都可以整合到一个已有的平台上，我们唯一要做的就是定义处理日志的的逻辑。

这里有几点注意的：

1. 如果你拥有复杂的产品线，那么日志格式会是一个很痛苦的事情。以为这中间Storm(或者SparkStreaming)的处理环节你需要做大量的兼容适配。我个人的意见是，第一，没有其他更好的办理，去兼容适配吧，第二，推动大家统一日志格式。两件事情一起做。我一个月做不完，那我用两年时间行么？总有一天大家都会有统一的日志格式的。
2. 如果你的研发能力有富余,或者有大数据团队支撑，那么可以将进入到SparkStreaming中的数据存储起来，然后通过SparkSQL等做即席查询。这样，有的时候原先没有考虑的指标，你可以直接基于日志做多维度分析。分析完了，你觉得好了，需要固化下来，那再去更新你的SparkStreaming程序。

后话

我做上面第一幅图架构实现时，从搭建到完成SparkStreaming程序开发，到数据最后进入HBase存储，大概只花了一天多的时间。当然为了完成那个Trace的指标分析，我修改ServiceFramework框架大约改了两三天。因为Trace分析确实比较复杂。当然还有一个比较消耗工作量的，是页面可视化，我这块自己还没有能力做，等招个Web开发工程师再说了。