唯品会峰值系统架构演变

姚仁捷 程序员日志 2015-05-15

↑↑↑

当你决定关注「日志君」，你已然超越了99%的程序员

日志君导读：

在唯品会，用户来得越早，越能买到又便宜又好的东西，所以在大促一开始会涌入大量用户，形成系统流量峰值。唯品会数据平台与应用部门研发工程师姚仁捷在本文总结了唯品会419时日志平台遇到的问题和解决方案，同时根据实践经验，整理了在面对峰值前要做的准备。

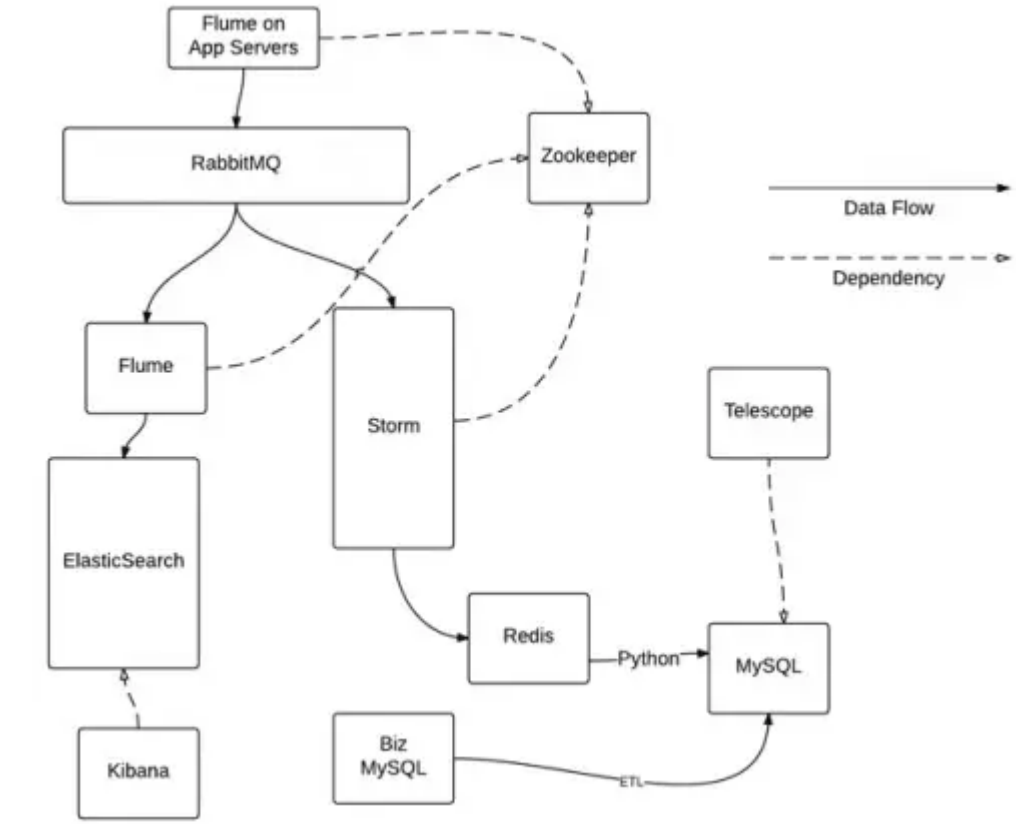
本文转自 程序员杂志 ，点击 **阅读原文** 可查看文章原网页。

唯品会每年最大力度的促销活动在4月19日，就是419（For One Night），意在告诉唯品会用户只有这一晚有这么大的折扣力度（本文中用"大促"就指代419）。唯品会是一个闪购网站，用户来得越早，越能买到又便宜又好的东西，所以在大促的一开始，会涌入大量用户，形成系统流量峰值。

本文总结了唯品会419时日志平台遇到的问题和解决方案，同时根据实践经验，整理了在面对峰值前要做的准备。

2013年419

唯品会的日志平台，包括消息中间件、计算和数据可视化。前两者和峰值系统相关度更大一些。在2013年419时，我们使用Flume来收集日志，RabbitMQ作为传输日志的消息中间件，用Storm和Redis进行计算，然后通过脚本将Redis的数据写入MySQL，前端从MySQL中获取数据做数据可视化。架构如图1所示。



在这次419之前，我们对这个系统并不是很有信心。一个原因是刚开始做这块内容，很多工具都不够成熟。另一个原因是在大促之前，我们还在开发新功能，既没有稳定运行一段时间，也没有做容量规划之类的事情。

最后的结果确实如此，4月19日0点，大量用户进入唯品会购物，系统计算开始出现延迟，最初是1分钟，后面逐渐扩大到10分钟，最后由于雪崩效应，整个集群垮了。在分布式系统中，“雪崩效应”指的是系统中一个小问题会逐渐扩大，最后造成整个集群宕机。前面这个例子，一开始的计算延迟是1分钟，这在可以接受的范围内，但因为这个延迟，系统需要付出更多的代价去计算，如此恶性循环，数据延迟会越来越大，最后导致整个集群宕机。

在大促之后，我们进行了全面分析，发现这个系统的瓶颈在于RabbitMQ和Storm。

作为整个平台输送数据的管道，RabbitMQ的性能直接决定了后端消费数据系统的消费能力。整个平台就像是大炮，大炮发射再快，输送炮弹的速度跟不上都没用。这次大促中，RabbitMQ的性能出了问题。我们需要处理的日志量是每秒15万条左右，而我们使用RabbitMQ的环境下，每一台RabbitMQ服务器大约能达1.2万条日志每秒，由4台机器组成RabbitMQ集群，所以当流量暴涨时，RabbitMQ服务器负载会变得很高，而produce/consume速度变慢。在当时的情况下，我们并不能判断这个Queue的堵塞是由于下游的Storm消费得慢，还是RabbitMQ本身慢造成。

再看Storm。在分析Storm出问题的原因之前，先先介绍一下使用Storm计算的内容：一是根据用户访问日志计算PV/UV；二是根据Nginx日志计算各个域的访问量、响应时间和

4xx/5xx数。由于Storm在各个计算节点之间无法共享数据（不像Spark有broadcast），我们只能借助Redis来做一个类似MapReduce中的Reduce功能。为了让大家能深入了解，下面详细介绍一下如何使用Storm计算这些数据。

PV

在Redis中有不同的key，如b2c_pv和mobile_pv，对Storm中得到的每条日志进行逻辑判断决定是b2c还是mobile访问，再使用Redis的incr操作（incr[key]，表示将key对应的value加1，如果key不存在，则在这个操作前，会先置为0）。

UV

我们计算的是每5分钟的UV，方法很简单。在Redis中有一个DB专门用来计算UV，Storm将每个用户的cid（标识用户唯一身份的id）incr到DB中，这样能保证一个cid对应一个key。最后汇总通过Redis的“keys *”来获取DB中key的数目，这样就获取到了cid的数目。又因为我们计算的是5分钟的UV，所以还需要一个crontab，每5分钟将DB中的内容truncate掉。

计算Nginx日志

实际上，计算Nginx日志非常简单，就是分割和计算。将一条Nginx日志分割后，就能知道这次访问的状态码是什么，响应时间是多少。然后DB中会有不同的key，如domain是cart，那么cart域的响应时间在Redis DB里的key就是cart_resp_time，访问量就是cart_count，2xx数量就是cart_2xx_count。根据从日志获取的值，我们会使用Redis的incrby来操作（incrby和incr类似，incr是value加1，incrby可以指定增加的数字）。然后在计算metrics时，脚本先获取当前的cart_count，然后sleep 1秒，再获取一次cart_count，这两个的差值是这1秒钟内cart域的访问量。同样的方法，可以获取这1秒的响应时间，与访问量相除，就可以计算出这1秒的平均响应时间。

介绍完计算逻辑，可以了解到，Storm的处理逻辑非常简单，主要工作就是“分割日志”和“操作Redis计数”。

为了判断到底是RabbitMQ慢还是Storm慢，我们先将Storm停了，然后用一个Python脚本向Queue发送数据和消费Queue里的数据，这样来减小Producer和Consumer性能对RabbitMQ的性能影响。这样测试后发现，每台RabbitMQ的吞吐大概是1w条数据每秒，而且负载很高。后来，我们使用Erlang的HiPE特性（即High Performance Erlang），将性能提升20%，大概达到1.2w条数据每秒。但仍然不能满足我们的要求。我们要求达到15w msg/s，加上25%的冗余，此时需要15×(1+25%)/1.2≈16台服务器，比较多。再考虑

到唯品会正处于快速增长期，目前是15w msg/s，很有可能明年就翻几番。使用RabbitMQ似乎不太符合我们的需求，因为在可预见的将来，需要大量服务器来支撑。此外，RabbitMQ对服务器的CPU消耗非常大。

RabbitMQ的消费者除了Storm外，还有Elastic-Search（ES）。使用ES来做日志的全文检索，包括Nginx日志和PHP错误日志，因为Nginx日志的计算只能帮助运维人员和开发人员定位到某个域出问题，再深入地分析，就要从出错时的日志入手了。我们的日志还会有一份全量流入HDFS，原本用日志的搜索直接从HDFS来获取，但发现用Hive查询速度非常慢，大约需要几分钟。ES是基于Solr的一个全文检索引擎，有一个很好用的前端Kibana。在这次大促中，由于前端的RabbitMQ挂了，所以ES没有受到很大的压力。

……

作者：姚仁捷，唯品会数据平台与应用部门研发工程师，从事日志平台相关工作，关注Flume、Kafka、Storm和ElasticSearch等技术。对Zabbix监控系统有深入研究，出版了《Zabbix监控系统深度实践》。新浪微博：@超大杯摩卡星冰乐

阅读原文