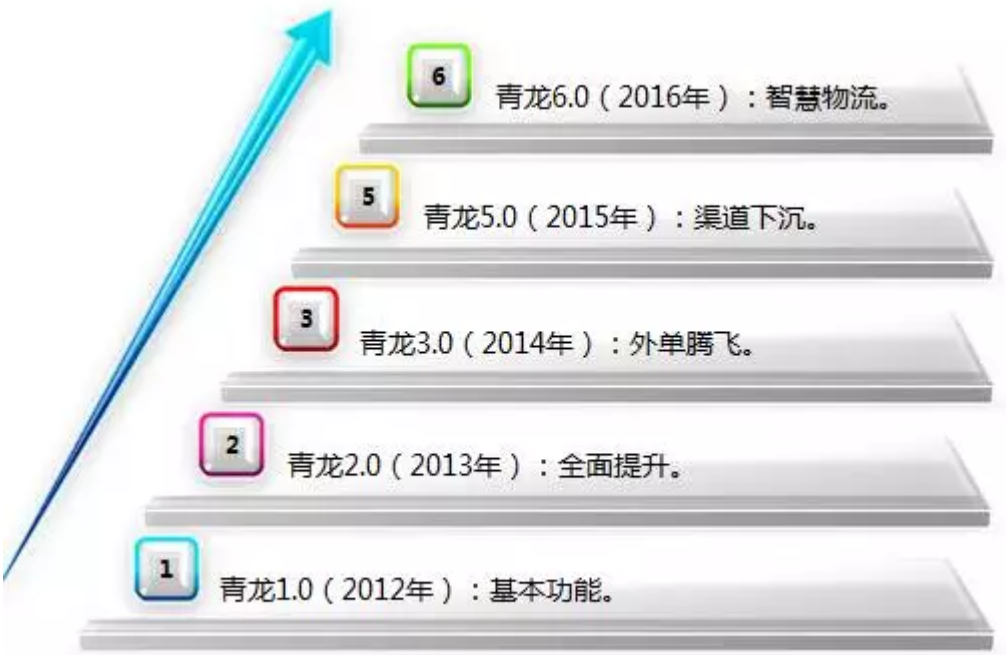
京东物流系统架构演进中的最佳实践

李鹏涛 架构文摘 2017-01-21

> 来源：彗星计划

青龙系统发展历程



青龙系统从2012年开始1.0的封闭开发，到2016年规划的6.0智慧物流，基本打造了一套完善的电商物流系统。

青龙系统1.0，主要实现了电商物流基础功能，满足了但是的核心业务诉求。青龙系统1.0上线，被京东当年评为优秀项目，成功之处就是比较好的遵循了MVP原则，也就是最初的版本只实现最有价值的部分。在著名的《人月神话》中，多次提到第二次开发系统失败的案例，最重要的原因就是，大家对于第二次开发的系统寄予了太多的希望，而项目负责人也给予了太多的承诺，时间等诸多问题，让项目走向失败，这个在中国互联网系统重构的案例中也可以找到。当时，青龙项目和业务负责人，都还是有比较清醒的认识，基本没有增加特别多的需求。项目组利用半年封闭完成开发，再用半年完成全国推广上线，完全替代了老系统。青龙系统是基于Java的SOA理念来开发，而老的系统是.net ，这也就是造成是完全重新开发，而不是平滑升级，付出代价也是很大的。

青龙系统2.0，起于2013年，主要是追赶功能。因为2012年开发新系统，业务方的新需求较少支持，也积压了非常多的需求，也非常感谢业务方的支持。团队利用一年的时间，完成了两年业务开发，也是非常给力，

和业务方也构建了非常信任的伙伴关系，为后续系统健康发展，奠定了很好的基础。到2.0完成开发时，青龙系统已经成为完善的自营电商物流系统。

2014年，我们规划3.0时，也有些迷茫，因为系统已经比较完善了，该向何处去是个问题。这时候，公司推动了物流开放的战略，希望利用京东物流的优势，来带动POP平台体验的提升，因此，我们抓住机会，确立了以外单开放为主题。我们开发了青龙开放平台，接单系统，和主流的ISV软件完成对接，以及改造现有分拣，运输，配送等环节，来支持外单。因为符合公司战略，业绩还是不错，当年双十一的时候，外单量也突破了60万单。这给我们很大的经验就是研发的方向，要符合公司战略，获得运营团队支持，才能形成合力，共同推动业务成功，否则，即使做出很炫的系统，也会因为没有用户而夭折。

2015年，团队也比较成熟，公司的战略是渠道下沉，3F等，我们也据此确定5.0主题是渠道下沉，配合公司战略，从零开始，构建了京东乡村推广员系统和校园派系统。年底，乡村和校园业务，都成为年会的明星，我们也获得了很大的认可。

从2015年开始，互联网＋提升到国家战略层面，物流也越来越受到重视，互联网＋物流，能做的事情越来越多。因此，到2016年，我们规划青龙系统6.0的时候，把主题确定为智慧物流，也就顺理成章了。

青龙系统发展到今天，已经包含了分拣中心，运输路由，终端，对外拓展，运营支持等五十余个核心子系统，构建了完善的电商物流体系。

青龙系统架构最佳实践

青龙系统架构演进过程中，从高可用，高性能，数据一致性，用户体验四个方面，积累了丰富的经验，确保了青龙系统在发展过程赢得了公司内外的口碑。

高可用

每年“双十一”都是网购狂欢节，假设当天哪个电商系统出现系统不可用，那几乎是灾难性的，不仅会导致用户快速流失，而且，公司将承受重大损失，甚至在未来竞争中失败。即使对于创业公司，在当前获取用户如此昂贵和竞争如此激烈的情况下，系统不可用的代价也是非常大的，会遭到用户的抛弃而失败。

青龙系统作为京东后台物流系统，系统高可用也同样重要，因为，即使在平时，物流系统出现不可用的情况，会造成订单时效履约失败，极大影响用户体验，这也是无法接受的；同时，系统不可用也会导致数十万员工无法正常工作，对于效率极大影响，公司损失也非常大。我们在研发过程中，对于系统高可用，也积累了丰富经验，主要包括：合适的架构方案；大系统小做，服务拆分；并发控制，服务隔离；灰度发布；全方位监控报警；核心服务，平滑降级。

首先是选择合适的架构方案。互联网系统一般可以分为前端应用系统和后端数据库系统，前端应用系统实施分布式集群部署技术上是比较成熟的，后端数据库系统实现异地多活技术难度很大，目前也只有阿里，京东这样的公司才真正实现。因此，对于大多数应用，前端应用双机房集群部署，后端数据库系统采取成熟的主备从的模式，也就是单个机房作为写入，备库在另外机房，可以快速进行切换，读库双机房部署，是优选的方案。对于这个架构方案，存在跨机房写延长的问题，可以根据场景利用异步的方式进行解决，一般也是没有问题的。对于青龙系统来讲，也有些特别，利用分拣中心的本地服务器和操作人员的设备，实现离线生产，进一步提高可用性。

大系统小做，服务拆分，是互联网应用的特点，也符合敏捷交付的理念。对于传统软件，如Windows，Office等，都要经过一个漫长的需求，研发，测试，发布周期，在“唯快不破”的互联网时代，这显然是无法满足业务要求的，即使最后上线，也可能因为周期太长而不再适用了。因此，对一个互联网服务，一般会首先完成最核心的功能，快速进行上线，不断进行迭代，后续再进行辅助功能跟进。对于核心功能，随着用户数的增加，会不断进行服务拆分，如何进行拆分，拆分到什么样的粒度，是不是微服务是解决问题的银弹？这些都要根据实际的应用场景来评估，绝不是越细越好，而是要达到一个优雅的平衡。

并发控制，服务隔离。并发控制，现在已经成为互联网服务基本要求，在应用程序端和数据库端，也都有成熟的方案，如果忽略，可能造成灾难性的后果。对于重要的服务，还要进行隔离，例如同一个服务，要提供给内部调用，公司级调用和公司外开放服务调用，开放服务调用者我们一般认为是不可靠的，甚至有可能是恶意的，如果不进行隔离，开放服务调用有可能使得服务资源占满，对内也无法提供服务。从技术上，可以是硬件级隔离，全部隔离，也可以是前端应用的隔离。

灰度发布也是互联网服务的一大利器，有了灰度发布，才使得快速迭代成为可能，并且，很多服务因为各种原因线下也是很难测试的，只能在线上测试。如果没有灰度发布，只能全量发布，就存在较长测试周期问

题，如果没有重复勉强上线，就存在很大的系统崩溃的风险。按照用户，区域进行灰度发布是比较常用的方法。

全方位监控报警，可以分为技术层面和业务层面，技术层面包括对CPU，内存，磁盘，网络等的监控，业务层面，包括对处理积压量，正常的业务量等。做到全方位监控，才有可能在影响用户之前，提前解决问题，提升系统可用性。否则，等用户发现问题，在很大的压力下，技术团队更难处理，导致系统不可用时间加长。

最后就是，核心服务，平滑降级。任何技术手段，都不可能保障100%可用，并且，即使能够做到，其代价也是巨大，不经济的，因此，对于核心服务来讲，能够平滑进行降级，提供基础的服务，也是非常重要的。对于青龙系统来讲，就利用分拣中心本地服务器和操作人员的设备，研发了离线生产系统，来应对集中服务万一不可用的情况。

举两个案例来做简单说明。

第一个是电子签收的案例。电子签收对于京东物流意义重大，不仅提升效率降低成本，而且实现了所有业务数据化，为智慧物流奠定基础。我们最早做的时候，也是遵循了MVP原则，开发了一个很简单的原型系统，内部测试可行后，才进行第二步，也就是和京东云合作，接入京东云图片处理系统，支持百亿张图片存储，找到安全认证伙伴，接入安全认证服务。这些完成后，我们就开始线上测试，不断提升用户体验。第三部，在大规模推广之前，我们完成了系统降级服务，也就是确保京东云，安全认证服务，甚至，我们自身的电子签收管理系统不可用的情况下，如何保证配送员的业务不受到影响。

第二个案例是去IOE的案例。青龙系统最初也是基于IOE开发，后来，随着系统规模的扩大，去IOE成为必然，在整个过程中，我们也是遵循了很多架构实践。例如，第一步，我们是先去IE（IBM小型机和EMC存储），将核心业务系统库拆到Oracle PC服务器上，并且，将数据进行同步，在验证服务完成后，把读服务迁到小型机，稳定后，再迁全部服务，并且，保证服务可以随时迁回。第二步，我们开发了双写服务，逐步将Oracle，迁移到MySQL集群。第三步，完成MySQL的数据聚合，确保数据服务。在整个过程中，都确保了系统在任何情况下，都可以会退，确保业务正常。

高性能

对应互联网服务来说，高性能是必须的，用户的响应一般都要求是秒级，而一个用户操作都包含多个服务调用，对应服务接口响应的要求都是毫秒级。对应青龙系统来讲，支持物流操作人员有十万余人，每个操作提升一秒，那么就能节约三个人员，意义是非常大的。

如何提高性能，接口数据缓存化是非常重要的手段。青龙系统属于后台操作型系统，业务逻辑复杂，如果不能缓存，完全依靠数据库操作，那么，响应会超过数十秒。如何进行缓存，需要设计缓存系统进行支撑，青

龙系统在演进过程中，依托公司的缓存服务，并且结合应用内存，包括Redis消息通知体系，构建了具有自己特色的缓存体系，很好的支撑了业务发展，案例结构如下图。

大型互联网服务，一般都微服务化了，这样意味着一个用户操作，都是由多个服务接口支持，如果按照传统的同步接口设计，那么，不仅面临性能问题，而且，QPS也是无法满足的，因此，需要将同步接口调用异步化。在2012年左右，eBay就提出所有系统调用异步化，后面，几乎所有大型互联网公司，都对自身系统进行了异步化改造，并且，取得了很好的效果，在和腾讯CTO Tony交流中，他就提出即使支付这种服务，也是有办法进行异步化设计的。同步接口异步化，也是需要系统工具支持的，青龙系统在发展过程中，也发展了基于Redis的分布式调度系统，架构参考下图：

做大了缓存和异步化，系统性能会有很大的提升。对应青龙系统这样的大型互联网服务，对应核心服务要求是非常高的，同时，又有数量非常多的非核心服务，如果不能进行主次服务分离化，那么意味着如果要提升核心服务水平，增加服务器，那么，就需要为所有的服务进行扩容，这样是不经济的。因此，系统需要精心设计，做到核心服务和非核心服务分离，给核心服务提供充足的资源，确保核心服务的性能。

数据一致性

数据服务，对于大型互联网应用，已经变为非常核心，称为系统的大脑也不为过。

我们一般需要考虑实时性和一致性，这两个最重要的维度，当然，数据量也是一个维度，一般我们认为是大数据的应用场景。

这样，我们就能分为四个基本的场景：高实时性/高一致性，高实时性/低一致性，低实时性/高一致性，低实时性/低一致性。针对具体的业务，我们可以匹配到具体的数据场景，这样，我们就能找到对应的解决方案。在这个过程中，客观的进行业务分析非常重要，并不是，选择高实时性/高一致性是最优方案，因为这个方案的实现成本是最昂贵的，可能是不经济，也没有必要的。

实时&强一致场景：这个在大数据技术成熟之前，是非常棘手的，但是，现在解决方案已经比较成熟了。典型应用是生产系统的实时监控，例如实时生产量，各个生产环节差异量等，其实是作为生产系统的一部分。利用当前主流的大数据处理架构是可以解决的，例如线上生产库binlog实时读取，Kafaka进行数据传输，Spark进行流式计算，ES进行数据存储等。如果利用传统的ETL抽取方案来解决，频繁对生产数据库进行抽取，并不是可行的方案，因为，这样会极大的影响线上OLTP系统的性能。还可以举一个生产系统实时监控案例，架构方案是应用系统完成写数据库的同时，把内容通过消息发送，后面的大数据处理系统接收消息来进行处理，这个架构方案，对于实时性某种程度上可以保障，但是，也存在效率问题，但是，对于强一致性就非常不合适了，因为消息系统如ActiveMQ等不仅无法保障消息数据不能丢失，而且对应消息顺序也是无法保障，项目实施后，虽然采取了很多补救措施，也无法满足强一致性需求，不得不重起炉灶。

实时&弱一致性场景：典型的应用场景是消息通知，例如电商的全程跟踪消息，如果个别数据出现丢失，对于用户的影响并不大，也是可以接受的，因此，可以采用更加廉价的解决方案，应用完成对应的动作后，将消息发出即可，使用方订阅对应的消息，按照主键，如订单号，存储即可。

离线&强一致场景：这是典型的大数据分析场景，也就是众多的离线报表模式。从技术上，传统的ETL抽取技术也能满足要求，数据仓库对应的技术也能够解决。

离线&弱一致场景：对于抓取互联网数据，日志分析等进行统计系统，用于统计趋势类的应用，可以归为此

类，这类应用主要是看能够有足够廉价的方案来解决，是不是可以巧妙的利用空闲的计算资源。这个在很多公司，利用晚上空闲的计算资源，来处理此类的需求。

以上讨论的都是大数据应用，也就是从数据量大的应用场景。但是，对应现实中很多数据处理系统来讲，例如很多B2B业务系统，或者传统行业，其实是数据量并大，那么采用更加廉价的OLTP的方法，例如复制读库等，也是可以完成对应的工作的。

因此，架构设计应当针对具体应用场景的，满足当前业务的发展需求，可以考虑两年的需求，最合适的架构就是最好的，而不存在放之四海都是最好的架构设计。不分析清楚自己的应用场景，盲目照抄大公司的技术架构，显然也是不合适的。当然，如果选择的架构本身，不能满足应用场景的需求，后续，不论进行多少补救，依然无法满足需求，并且，架构会变得异常复杂，替换的成本也将是非常高昂，不得不慎重。

用户体验

京东是非常重视用户体验的公司，老刘就明确指出任何人不能对用户体验提升的意见说No。青龙系统在研发过程中，我们认为MVP原则和动态运营是非常重要的。

MVP原则，也就是敏捷开发中的迭代思路。对应一个大的项目，按照传统的瀑布模型，一般经历设计，研发，测试到最后上线阶段，这对于互联网应用来说，很多情况下是不能接受的，因为业务需求变化太快，如果上线周期太长，也许上线后发现情况已经变化了，或者，上线后发现不能落地推广。因此，对应一个大项目，一般会进行迭代分解，最核心的需求，会优先开发，并完成上线，上线验证后，继续开发优先级低的需求。

动态运营，其实也和MVP原则有很强的联系，也就是功能上线后，要真正运营起来，看具体数据，如果发现和设计不符合，那么，就要进行调整，到符合用户需求。这也是互联网服务的用户体验，要优于传统的软件开发系统，传统软件开发基本上上线后就不在优化了，而对于互联网服务来说，上线只是开始，只有将这个功能运营好，才叫好，并且，这个过程一直是持续的。

小结

对于如何打造一个高可用的互联网系统，上面很多点大家都知道，包括高可用，高性能，数据一致性和用户体验，关键是如何落实和做到极致，就如大家都学习乔布斯，但是，能够真正把产品做到极致的还是凤毛麟角。

版权申明：内容来源网络，版权归原创者所有。除非无法确认，我们都会标明作者及出处，如有侵权烦请告知，我们会立即删除并表示歉意。谢谢。

-END-

架构文摘

ID：ArchDigest

互联网应用架构丨架构技术丨大型网站丨大数据丨机器学习

更多精彩文章，请点击下方：阅读原文

阅读原文