# 论系统性能优化在图书馆数据中台的实践

## 一、摘要

2019年11月，本人所在信息技术部承接了馆内数据中台的建设，该项目为期半年。该项目于 2020 年 5 月正式上线， 2020 年 6 月通过最终验收。通过数据中台，实现了我馆数据的分层与水平解耦，沉淀了公共基础数据。通过数据建模实现跨域数据整合和知识沉淀，通过数据服务实现数据的封装和开放，通过数据开发工具满足个性化数据需求，为全市文旅数据融合工作提供了有力的数据支撑。该项目时间紧任务重，涉及人员组织多，涉及馆内 10 余个部门，外部配合协作 10 多个厂商团队。本人在此项目中担任系统架构设计师，主要完成技术方案评估与实现，项目立项论证等工作。本文以此数据中台为例，讨论系统性能优化在该项目中的具体应用，包括系统性能设计的概念、性能评估的具体实施以及在实际场景中遇到的问题。

## 二、正文

近年来，随着我馆服务质量的不断提升，大量软硬件系统在我馆得到部署。硬件设备包括自助借还机、交互机器人、电子书借阅机、人脸识别门禁、人流量计数器，软件系统包括图书管理系统、微信小程序、移动端APP。软硬件系统的增加带来了大量用户数据的流入，如读者借阅数据、读者信息、阅读偏好、入馆行为等海量数据，但是这些数据缺乏统一的接入和分析。因此，需要通过数据技术，对海量数据进行采集、计算、存储、加工，同时统一标准和口径，并在此基础之上，对数据进行归纳分析，对外提供统一的、科学的、精准的馆情数据服务。

笔者带领团队于2019年11月承接馆内数据中台的建设，并担任系统架构设计师。主要负责项目整体技术方案评估实现、立项论证以及项目管理工作。该项目的架构工作于同年12月完成，整个项目耗时6个月，于2020年5月上线测试。于2020年6月完成验收。

在实际开发阶段，笔者意识到架构数据中台的性能是架构设计必须要首先考虑的属性。对于数据中台这种以数据为核心的系统来说，由于其定位是一种要开放接口服务多方其他系统的系统，性能的设计更是十分重要。在理论上，系统的性能一般用响应时间来衡量，即一个请求发出开始到收到请求结束为间隔得到的时间。相同的业务场景，响应时间越短，那么可以任务系统的性能越好。从这个定义出发可以看到，在整个请求的生命周期中，从请求传输、处理、到落库的任何一个环节消耗时间的长短都会对系统性能产生巨大的影响。从实践角度来看，作为系统架构师主要关注的是服务的业务逻辑优化、中间件优化、以及数据库优化。针对数据中台的应用场景，笔者从数据接入层、数据模型层、数据服务层，数据开发层几个角度来阐述性能设计优化在各个层次上的作用。

在数据接入方面。在实际接入过程中，笔者除了考虑常规的WEB类应用比如APP、小程序、Web等，还要众多硬件设备的数据接入。这些硬件设备由于某种原因往往只支持基于TCP的自定义应用协议。由于工期进度等等原因，在实际项目推进中，让对方再提供一套HTTP的协议进行接入往往不太现实，因此再数据接入支持方面，协议也不局限于HTTP，也有基于TCP 的图书馆自有协议SIP2和厂家自定义协议。为此在实现过程中，笔者将接入服务集群分为两大类,一类是常规的HTTP服务集群,另一类是走TCP自定义协议的服务集群。这两个集群由同一个NGINX服务器做负载均衡与反向代理，用端口加以区分。由于NGINX1.9以上才支持四层的反向代理，因此在实际部署中特地使用第三方模块重新编译NGINX 来对反代TCP协议进行支持。

在数据模型方面，笔者这里概括为三层。分别是基础模型、融合模型和挖掘模型。其中基础模型一般是关系建模，主要实现数据的标准化，这也叫作“书同文、车同轨”。通过设计好的数据标准将接入的数据进行入库管理。其次是融合模型。融合模型一般是维度建模，主要实现跨域数据的整合，整合的形式可以是汇总、关联，也包括解析。这些融合模型表是通过各个业务部门梳理而得到的常用的宽表，比如读者终端使用行为表、读者阅读偏好表等20余个宽表。最后是挖掘模型。常规理解，挖掘模型其实是偏应用的，但是在本馆的馆情分析中，例如读者常驻区域模型这类常用的挖掘模型，也把它规整到中台模型，以开放给其它人使用。在数据模型这个层次上的性能优化主要是梳理常用的数据查询方法，对于需要筛选和排序的**字**段加上适当的索引，如唯一性索引、非唯一性索引和联合索引。但是需要注意，虽然索引大大提高了查询速度，但同时却会降低更新表的速度，如insert、update和delete。因为更新表时，不仅要更新数据，还要更新索引文件。另外，建立索引会占用一定的磁盘空间，但是在一般情况下，这个问题不太严重。笔者在实践中发现，如果在一个大表上创建了多种组合索引，索引文件的体积才会增长很快。最后，虽然索引能够解决大部分问题，但它也不是万能的。对于一些加上索引仍然有性能问题的表，笔者通过开启 MYSQL 慢查询日志检查有问题的SQL日志，然后并对 SQL 执行计划进行分析，这个要具体案例具体分析，这里不再展开。

在数据服务方面，笔者将数据模型按照应用要求做服务封装，就构成了数据服务。这个跟业务中台中的服务概念是完全相同的，只是数据封装比一般的功能封装要难一点，毕竟OLTP功能的变化有限，而数据分析受业务因素的影响很大，变化更快，导致服务封装的难度变大。为了避免单点接入服务导致的单点故障和性能问题，对于所有数据服务接口，笔者都采用多实例的方式启动接入服务。在实践中，笔者采用借助业内著名的中间件Consul 对微服务进行治理，和 Nginx 相互配合可以负载均衡与高可用。Consul 提供健康检测功能，可以在服务注册的同时，配置相关的检测条件,并指定在故障出现多久后自动将服务从注册中心注销。对于需要暴露在外网的服务，笔者通过 Consul-Template 实例自动监测 Consul 集群的变化，若发现服务实例被注销，则按照事先定义好的Nginx 配置模板重新生成nginx.conf并reload本节点的Nginx，使得失效的服务从Nginx中删除，避免开放接口出现故障。在nginx.conf 中指定轮询算法为 fair，此算法按后端服务器的响应时间来分配请求，响应时间短的优先分配。对于处于系统内部的微服务，利用Consul的DNS发现层，每当应用程序或内核解析这个DNS条目时，它将收到一组IP地址的随机round-robin响应，这些地址对应于集群中的健康服务。DNS接口基本上提供了与任何应用程序的零接触服务发现集成。后端服务直接通过 Consul 提供的服务域名进行互相访问，实现了负载均衡。在数据服务方面，笔者在做了两方面的性能优化工作。一方面是数据缓存的设计，目的是减少数据库资源的争夺，提升系统的响应时间。该数据中台使用 Redis Cluster实现热点数据的缓存，并采用主从复制技术实现缓存层的高可用与读写分离。当数据请求到达时，首先判断数据是否在缓存中，若命中则直接返回。若涉及到数据库写入操作，则先将数据写入数据库，再删除缓存。这种先更新数据库，再删除缓存的方法，比先删除后更新来讲触发脏读的概率更低，因此目前在业界被广泛采用。另外在实践中还需要考虑缓存穿透的场景。当一个数据请求访问一个不存在的数据（可能是恶意），那么每次都会去访问缓存，导致数据库压力增大。实践中笔者通过缓存空对象来避免缓存穿透。。另一方面是MYSQL数据库读写分离的设计。对于MYSQL数据库实现一主多从的设计，所有读操作全部在副本上完成，主库与从库通过传递 binlog 实现数据同步。读写分离通常来说需要引入MYCAT之类的中间件进行读写的自动代理，或者在业务逻辑层上分离读写的对象实现读写分离。通过仔细分析，笔者从代码维护的角度考虑，选择引入中间件MYCAT来实现读写分离。通过读写分离技术，实现了数据库层面的负载均衡，避免了单机系统的故障与IO资源的争夺，极大提升了数仓操作的效率，确保了开放接口业务的稳定性。

但有数据模型和数据服务还是远远不够的，因为再好的现成数据和服务也往往无法满足前端个性化的要求。这时候就得授人以鱼。数据中台的最后一层就是数据开发，其按照开发难度也分为三个层次，最简单的是提供标签库（DMP），用户可以基于标签的组装快速形成分析报表，一般面向普通馆员。这里的标签库实际使用非常简单，馆员以拖拽的形式选择感兴趣的标签字段，比如性别、常驻区域、阅读偏好（分类号），通过添加逻辑运算符组合，并点击确定就可以快速生成目标读者群体。其次是提供数据开发平台，用户可以基于该平台访问到所有的数据并进行可视化开发，一般面向技术部SQL开发人员。最后就是提供应用环境和组件，让技术人员可以自主打造个性化数据产品，以上层层递进，满足不同层次人员的要求。由于引入了前端技术，因此这部分主要侧重于前端的优化。前端优化的主要目的是减少HTTP请求和缓存数据来提高页面的响应速度。一般在前端工具链中可以使用 Webpack 合并和压缩JS和CSS，还有使用专门的雪碧图制作工具来合并图片，减少图片请求。在数据方面，习惯将缓存分为强缓存和协商缓存两种。两者的主要区别是使用本地缓存的时候，是否需要向服务器验证本地缓存是否依旧有效。顾名思义，协商缓存，就是需要和服务器进行协商，最终确定是否使用本地缓存。般，我们会设置Cache-Control的值为“public, max-age=xxx”，表示在xxx秒内再次访问该资源，均使用本地的缓存，不再向服务器发起请求。显而易见，如果在xxx秒内，服务器上面的资源更新了，客户端在没有强制刷新的情况下，看到的内容还是旧的。因此笔者采用协商缓存。协商缓存意味着，当浏览器对某个资源的请求没有命中强缓存，就会发一个请求到服务器，验证协商缓存是否命中，如果协商缓存命中，请求响应返回的http状态为304并且会显示一个Not Modified的字符串。主要使用ETAG 和 Last-Modified 标志位在后端进行判断请求的资源是否过期。

通过引入性能优化来提升数据中台的建设质量，满足了日常的业务需求，部署实施后运行一直比较稳定。目前数据中台中的数据服务作为单独出口，对外提供了统一的馆情数据。服务运行稳定。数据开发层承载了馆员日常80%的取数工作，极大提升了报表生成的效率。但是从实践角度来看，随着接入数据的不断增长，传统的基于单机的关系型数据库会逐渐显现出瓶颈。因此项目二期的主要工作会放在基础数据库的改造，暂定使用 Hive 来实现分布式数据存储，使用 SPARK 进行基于分布式数据的内存计算，以及在此之上强化机器学习和数据挖掘的作用，进一步挖掘基础数据的内在规律，为更广泛的业务需求提供数据支持。