微博推荐架构的演进

架构师 2015-10-14

0引言

微博（Weibo）是一种通过关注机制分享简短实时信息的广播式社交网络平台。微博用户通过关注来订阅内容，在这种场景下，推荐系统可以很好地和订阅分发体系进行融合，相互促进。微博两个核心基础点：一是用户关系构建，二是内容传播，微博推荐一直致力于优化这两点，促进微博发展。如图1所示：

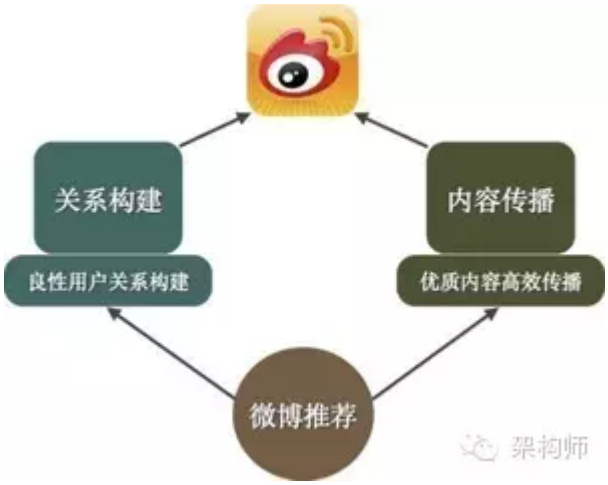


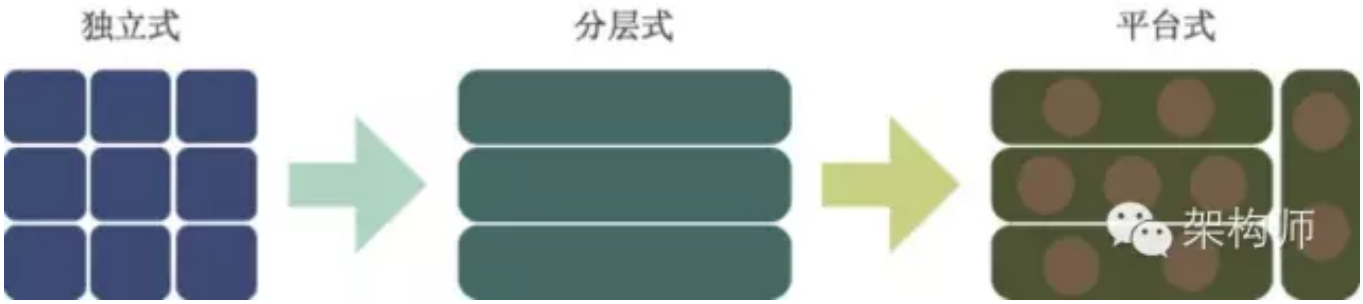
图1 微博推荐的使命

在微博推荐发展的过程中遇到体系方向的变化、业务的不断更迭、目标的重新树立，其产品思路、架构以及算法也随之进行变迁。本文主要阐述在这个过程中推荐架构的演进，从产品目标、算法需求以及技术发展等维度为读者呈现一个完整的发展脉络，同时也希望通过这个机会跟大家一起探讨业务与技术的相互关系。

为了便于理解微博推荐架构演进，在介绍之前需要陈述一下微博推荐在流程上的构成，其实这个和微博本身没有关系，理论上业内推荐所存在的流程基本都是相同的。如图2所示，推荐是为了解决用户与item之间的关系，将用户感兴趣的item推荐给他 / 她。那么，一个item被推荐出来会经过候选、排序、策略、展示、反馈到评估再改变候选等等形成一个完整的回路。

图2推荐的链路

在上述整体流程的基础上，微博推荐架构经历了如图3所示的三个阶段：



通常架构的产生都会来自于团队和业务环境，源于环境因素而致力于解决环境中的问题，架构形成会带着较为强烈的特点，在其实施中会产生交给针对性的效果。本文将从环境因素、架构组成与特点以及实施效果这三个方面进行阐述微博推荐的三个阶段。

1 独立式的1.0

1.1 环境

影响架构形成的环境因素可以分为内部环境因素以及外部环境因素。内部因素主要是团队及其成员相关内容，而外部因素主要来自于外部门、整个公司或者整个行业领域。

微博推荐1.0的这段时间是从2011年7月份到2013年2月份左右，其主要的目标就是实现当前的业务需求。对于独立式的解释：每一个业务项目都是一套完整架构流程，架构之间相对独立，甚至包括技术栈。之所以称之为独立式其内部因素有几点：

1) 当时团队是一个新团队，成员也相对较新，相互的合作不多，缺乏推荐领域整体性经验。

2) 团队成员对于推荐架构都有自己的一些或多或少的理解，但是对于在当前场景下的微博推荐架构，共识并没有形成。

当然起决定性因素的还是外部环境，是因为内部原因还是比较好协调和进化的。当时的外部环境因素包括：

1) 项目需求很多，在当时一个5人团队并行开发的项目平均在3-5个左右，当然最重要的因素是当时的微博产品正处于高速发展期，很多地方都需要微博推荐的支撑。同时，项目周期也很短，排期仓促，很难有时间进行细致的整理和抽象。典型产品包括：微吧、微群、微刊、微话题、用户以及内容排序等等。

2) 团队是一个支撑性的，绝大部分需求来自于外部团队，各个外部团队不同的产品方向也导致疲于应付需求。

3) 当时业内的推荐架构也有不同的发展方向，大家都在尝试摸索一些符合自身发展的架构思路。

由于上述的那些原因，通常我们面对一个接一个的项目时，都会根据自己的理解使用熟悉的技术栈来搭建流程，这样形成了一个又一个的独立架构。

1.2 架构组成与特点

上节中提到了独立架构形成的原因，大家可能觉得架构组成没有必要去描述了，这是不对的，事实上后来的分层以及平台架构的基础恰恰都来源于这个阶段，没有这个阶段团队不断踩坑总结就没有因地制宜产生的后续进化。因此，我们需要为大家剖析一下推荐1.0的架构组成与特点。

1) 技术目标

参考图2所示，以业务实现为主要目标的微博推荐1.0，没有建立起完整的反馈以及评估体系，同时排序也是被策略取代，那么讲主要的重点体现到了候选、策略以及展现上。上述推荐流程被转化为：候选à策略à展现简单形态。

2) 架构组成

如图4所示，我们试图将每个项目的架构能够在图中表达出来，在真正的实施过程中，每一个项目负责人会选择使用apache＋mod_python作为服务架构同时，使用redis作为存储选型。在一些特定的项目中，引入了复杂运算从而诞生了c/c++的服务框架woo；同时，对于数据的存储要求特型化的项目中又自己研发了一系列的db，比如早期存储静态数据的mapdb，存储key-list的keylistdb等等。当然，在部署中会比下图更加随意一些，一个项目几台服务器部署好微博服务提供http请求，然后再找几个服务器安装redis作为数据支撑，来源数据和业务方定好规则使用rsync传输就OK了，大部分策略在python中实现。

图中可以看到主要的技术栈：

web服务：apache＋mod_python，后来发展成为社区更为完善的mod_wsgi。使用python作为WEB开发语言主要是因为平时处理数据使用的都是python，同时上手快，学习曲线平缓。

运算服务：c/c++，形成woo内部服务框架

db：redis / mapdb / keylistdb等等，分为两种存储方法：redis以及自研型

数据来源：rsync文件传输，firehose作为微博相关内容来源［微博内部使用的一种数据队列］

图4 微博推荐1.0架构简图

3) 架构特点

将架构特点划分为优点和缺点进行描述。那么优点是：

简单，易于实现，不需要额外的基础支撑

利于业务的功能快速实现

利于多业务并行开展，相互不影响

而不足是：

推荐流程不完整，缺乏反馈、评估等等重要内容，对于数据方面也极度缺乏统一处理方法

没有提供给算法相关的支撑，很难将推荐做的深入

几乎无法进行专业运维

QA的测试仅仅能到功能层面，模块级别的测试几乎不可能，因为太过于分散

很难进行团队协作，不利于项目的分解

1.3 成果

尽管存在诸多的缺点，但是在其发展的过程中，也给后面的架构优化奠定了基础，其成果如下：

1) 在微博高速发展的过程中，满足了微博对于推荐的业务支撑要求，在这段时期里面共完成二十多个独立项目。

2) 诞生了woo的基础框架，后面的内部高效运算框架来源于此

3) 诞生了mapdb的静态存储，成为后期微博推荐静态存储的雏形

4) web应用层的不断需求的总结，组建形成推荐通用应用框架

2 分层式的2.0

上一节介绍完独立的1.0，按照架构发展的道路，我们到了分叉路口，一边是流行的LAMP架构，另一边是符合广告、搜索的CELL架构。LAMP架构数据策略分离，脚本语言作为业务开发主要语言，项目快速开发和迭代的首选。CELL结构强调本地流程处理，数据与业务耦合性强，自研的服务以及数据库较多出现，适用于高性能效果型产品。最终我们选择兼容两者，倾向于业务的架构体系。为何如此呢？让我们再来看看当时的环境。

2.1 环境

微博推荐2.0的时间段是2013年3月份到2014年年底，这段时间内部环境因素是：

1) 当前团队成员合作已经很长时间，彼此相互熟悉，同时对于技术选型有了一定的共识。

2) 团队产品进行了聚焦，针对内容 / 用户 / 垂直类三类推荐进行了整理，同时对于场景分别进行了重点划分：feed流内、正文页以及PC首页右侧。这种聚焦有利于进行架构统一，同时也为技术争取了时间。

而外部因素是：

1) 公司对于推荐有了比较明确的定位，提高关系达成以及内容传播效率，同时为推荐型广告打好技术探索、场景介入以及用户体验的基础。

2) 推荐领域里，各个公司都纷纷有了对于架构的产出，对于微博推荐有了很好的指导意义。

2.2 架构组成与特点

团队在执行核心业务实现的时候，不断演进工具以及框架，构建2.0的目标呼之欲出。

1) 技术目标

与1.0不同，仅仅实现业务需求已经不是2.0的技术目标了，针对完整的推荐流程，我们需要解决：

首先要实现完整的推荐流程，架构覆盖候选、排序、策略、展示、反馈和评估。

以数据为先，提炼出数据架构。实现数据对比，效果以数据为准；实现数据通道，体现反馈；实现数据落地，承接业务需求。

提供算法方便介入的方式。

既能保证业务的快速迭代和开发，又能支持高效运算。

2) 架构组成

微博推荐2.0的架构如图5所示，它不再是一个个独立的系统，也不是会让开发人员使用不同的技术解决相似的问题。这个架构图主要包括几个部分的内容：

应用层：主要承担推荐策略以及展现方面的工作，其特点在于充分发挥脚本语言的特点响应迭代需求。大部分的推荐内容经过排序之后已经可以展示了，但是由于前端产品策略的设定需要融合、删选以及重排操作，需要这一层来完成，在技术层面属于IO密集型的。在技术选型上，早期在原有apache＋mod_python基础上进行了框架开发产生了common_recom_frame。该框架面向的是二次开发者，基于此框架可以很好的实现推荐业务流程。该框架的核心思想是提炼出project、work以及data的三层interface，project针对每一个推荐项目，work针对每个推荐项目中不同推荐方法，而data则是管理下游数据的访问方法。同时，设定了两个规范：一个是统一了推荐接口，无论是用户、内容还是垂直业务；另一个是屏蔽了不同协议数据库访问方法，极

大提高了开发效率。common_recom_frame框架的诞生基本上解决了产品的各种推荐策略需求，走在了产品的前面。

图5 微博推荐2.0架构示意图

计算层：主要承担推荐的排序计算，主要消耗CPU，在这一层给算法提供介入方法，支持算法的模型迭代。在这一层的技术选型上，我们继承了原有的WOO协议框架，一种基于c/c++开发的内部高效通讯框架。当然也做了不少扩展，依然借用了上面提到的common_recom_frame的思想，在WOO框架基础上实现了对于project／work/data的管理，提供给二次开发者更为高效的开发工具。在团队的开源项目中包含这个工具：https://github.com/wbrecom/lab_common_so

数据层：主要承担推荐的数据流以及存储工作。数据层的工作主要是解决数据的IN/OUT/STORE问题。其中IN数据如何进入系统，OUT表示数据如何访问，STORE表示数据如何存。当在进行数据层规划的时候，又分析了微博推荐的数据特点，可以将其分为两类：静态和动态。静态数据的定义为： 更新需要全量同时频次较低的大规模数据；动态数据的定义为：动态更新同时频次较高的增量数据。这样在IN/OUT/SOTRE的大方向下，同时区分对待静态和动态数据，产生了RIN／R9-interface、redis/lushan、tmproxy／gout代理的工具或者框架。在这里展开讲一下，RIN支持数据动态数据的接入，通过web服务的方式接收数据，后端使用ckestrel进行队列管理，辅以多服务集群的消费框架，使用者只需要进行自己业务的开发即可快速上线消费动态数据。R9-interface处理静态数据的接入，推荐的大量静态数据来源于Hadoop集群的运算，r9-interface框架勇来解决静态运算［MR、HIVE

SQL以及SPARK 运算］的通知、管理以及数据载入。针对推荐数据的存储，动态数据大量使用了redis集群，静态数据则使用了lushan集群。对于lushan这个工具在团队开源项目中也包含了：https://github.com/wbrecom/lushan。tmproxy / gout用来解决数据的OUT问题，gout是一个代理中间件，用来处理推荐中对于数据的动静结合访问的需求，减少业务对于后端数据变化带来的影响。

基础服务：推荐系统的基础服务主要包括监控、报警以及评测系统，数据监控系统分为性能以及效果监控两类，评测系统主要用来进行下线评估，在上线之前对效果有一定预期以及减少无效上线。图6展示了基础服务的UI。

图6 基础服务系统的UI

3) 特点

优点是：

支撑完整的推荐流程，对于数据方面拥有统一的处理方法

在兼顾业务功能快速实现的同时保证了效果技术的不断深化

给算法提供了很好的支持

提出以数据为先的思想，可以全面对比效果，推荐效果不断得以提升

封层体系易于部署以及QA介入进行测试

而不足是：

和推荐核心有一定的距离，并没有完全为推荐量身定做

将推荐的策略算法完全交给了开发者，不利于推荐通用型

对于算法的训练并没有涉及，仅仅是一个线上投放系统，不足以构成完整的推荐体系

2.3 成果

微博推荐2.0的诞生产生很好的收益，其成果如下：

1) 微博推荐的核心业务均在该体系下完成：正文页推荐、趋势用户推荐、趋势内容推荐、各个场景下的用户推荐、粉丝经济的粉条、账号推荐等等产品

2) 诞生了lab_common_so的基础框架，并进行了开源

3) 诞生了静态存储集群解决方案lushan，并进行了开源

4) RUF框架的诞生极大提升了业务生产效率，同时也为openresty社区做出一定贡献

3 平台式的3.0

上节中描述2.0的时候提到了一个重要不足是“和推荐核心有一定的距离，并没有完全为推荐量身定做”，我们希望能够在推荐3.0中解决它，这个不足会带来什么问题，以及为何在已经满足业务需求的同时推荐的架构再次往前发展呢？那么接下来为各位展现微博推荐平台式的3.0设计，我们还是先看看所处的环境。

3.1 环境

微博推荐3.0的时间段是2014年底至今，当前的内部环境因素是：

1) 推荐产品不在扩张，对效果更为看重，将工作重点从业务开发和迭代转化为以效果为目标的技术迭代。

2) 新项目或者迭代推荐业务的时候发现重复的事情很多，而架构没有解决，工作存在冗余。

而外部因素是：

1) 公司也从业务扩展转变为效率为先，提升用户体验以及内容质量上来。

2) 微博推荐在推荐技术环节距离领域内有一定距离，当下有条件进行追赶。

3.2架构组成与特点

当前的环境也能体现出3.0的技术目标：

1) 技术目标

与2.0不同，全覆盖推荐流程已经不是3.0的目标，其目标是：

抽象出推荐流程中对于候选 / 排序 / 训练 / 反馈的通用方法来

推荐是一个算法数据问题，应该以一个算法的角度构建推荐系统，因此需要更为贴近算法策略

2) **架构组成**

如图7所示，是微博推荐3.0的架构，也是当前实行的架构体系，大家其实可以发现，这是基于2.0 发展起来的，既然还保留了大量2.0中使用的分层体系以及工具框架。在这里重点描述几个差异：

两个标准：一个是针对应用层，作为整体框架输出，应用层设定all in one 接口标准，其标准包含了输入以及输出参数；另外一个是针对动态输入rin，由于离线计算我们可以确定结构，因此一个输入层工具r9-interface不需要设定规范，但是rin是需要进行标准设定，从属性 / 交互数据 / 日志等等层面进行划分。

计算层增加对于候选的标准生成方法：Artemis内容候选模块，item-cands用户候选模块、……，在项目开发中只需要选择这些候选生成方法即可。

增加了策略平台EROS，解决算法模型的问题。EROS主要的几个功能是：1）训练模型 2）特征选取 3）上线对比测试。

数据层中的r9-interface以及rin增加对于候选的生成方法，在线以及离线使用推荐通用策略生成结果。

图7 微博推荐3.0的架构示意图

3) 特点

主要描述其优势：

继承了原有2.0的特点，保留了其优势

对于推荐理解更为深入，结合更为紧密

解决了推荐候选 / 排序 / 训练的算法最重要问题

3.3 成果

微博推荐3.0的诞生，其成果如下：

1) 微博推荐的核心业务会逐步迁移到该体系下，以算法数据作为驱动，提升效果

2) 诞生了EROS的训练流程，提出了训练的标准方法

3) 针对推荐设定了标准的输入输出方法

4) 针对候选，产生了具有抽象意义的推荐方法集合

4 总结

上文中对微博推荐架构演进做了较为详实的介绍，在这个演进的过程团队以及个人收益很大，技术与业务的关系在架构中得到了很好的体现。有几点可以跟大家分享的是：

1) 技术来源于业务同时提升业务发展，业务发展又反过来推动技术的前进，他们是一个相互影响相互促进的关系。和业务共同发展的技术才是有生命力的。

2) 技术架构的选型建议是寻找当前最短路径，然后进行不断优化迭代，一口气吃撑是不现实的，也是不合理的。

3) 推广某个框架和工具最好的方式不是行政命令也不是请客吃饭，而是的大家都是参与者，如同开源项目，每个人都是它的主人，这样人人维护，人人使用。

4) 团队崇尚简单可依靠，它说起来容易做起来难，不过有一个好方法就是懂得自己不应该做什么，而不是应该做什么。

5) 说到推荐这个特殊领域上来，设定目标，跟踪目标很重要，把数据和目标摆出来，产品、架构以及算法都会想办法去解决的。

最后，跟大家推荐一下微博推荐的官方博客：http://www.wbrecom.com/ 欢迎大家提出建议和建议。感谢大家对于微博推荐以及微博的关心和爱护，谢谢！

来源：http://www.wbrecom.com/?p=540

·END·

架构师

我们都是架构师！

架构师订阅号，关注获取更多技术分享

小编私人微信号：13511421494

工作邮箱：admin@137x.com

微信群请加小编微信号邀请

架构师技术交流QQ群