1万属性，100亿数据，每秒10万吞吐，架构如何设计？

Original KG沈剑 架构师之路 2022-09-01 07:46 Posted on 北京

收录于合集
#数据库 35 #架构 82 #架构师 11 #搜索引擎 2

有一类业务场景，没有固定的schema存储，却有着海量的数据行数，架构上如何来实现这类业务的存储与检索呢？

1万属性，100亿数据，10万吞吐，今天和大家聊一聊，这一类“分类信息业务”架构的设计实践。

一、背景描述及业务介绍

什么是分类信息平台最核心的数据？

一个分类信息平台，有很多垂直品类：招聘、房产、二手物品、二手车、黄页等等，每个品类又有很多子品类，不管哪个品类，最核心的数据都是“帖子信息”。

画外音：像不像一个大论坛？

各分类帖子的信息有什么特点？

逛过分类信息平台的朋友很容易了解到，这里的帖子信息：

（1）**各品类的属性千差万别**，招聘帖子和二手帖子属性完全不同，二手手机和二手家电的属性又完全不同，目前恐怕有近万个属性；

（2）**数据量巨大**，100亿级别；

（3）**每个属性上都有查询需求**，各组合属性上都可能有组合查询需求，招聘要查职位/经验/薪酬范围，二手手机要查颜色/价格/型号，二手要查冰箱/洗衣机/空调；

（4）**吞吐量很大**，每秒几10万吞吐；

如何解决100亿数据量，1万属性，多属性组合查询，10万并发查询的技术难题呢？一步步来。

二、最容易想到的方案

每个公司的发展都是一个从小到大的过程，撇开并发量和数据量不谈，先看看

（1）如何实现属性扩展性需求；

（2）多属性组合查询需求；

画外音：公司初期并发量和数据量都不大，必须先解决业务问题。

如何满足业务的存储需求呢？

最开始，业务只有一个招聘品类，那帖子表可能是这么设计的：

tiezi(tid, uid, c1, c2, c3);

那如何满足各属性之间的组合查询需求呢？

最容易想到的是通过组合索引满足查询需求：

index_1(c1, c2)
index_2(c2, c3)
index_3(c1, c3)

随着业务的发展，又新增了一个房产类别，存储问题又该如何解决呢？

可以新增若干属性满足存储需求，于是帖子表变成了：

tiezi(tid, uid, c1, c2, c3, c10, c11, c12, c13);

其中：

（1）c1,c2,c3是招聘类别属性；

（2）c10,c11,c12,c13是房产类别属性；

通过扩展属性，可以解决存储的问题。

查询需求，又该如何满足呢？

首先，跨业务属性一般没有组合查询需求。只能建立了若干组合索引，满足房产类别的查询需求。

画外音：不敢想有多少个索引能覆盖所有两属性查询，三属性查询。

当业务越来越多时，是不是发现玩不下去了？

三、垂直拆分是一个思路

新增属性是一种扩展方式，新增表也是一种方式，垂直拆分也是常见的存储扩展方案。

如何按照业务进行垂直拆分？

可以这么玩：

tiezi_zhaopin(tid, uid, c1, c2, c3);
tiezi_fangchan(tid, uid, c10, c11, c12, c13);

在业务各异，数据量和吞吐量都巨大的情况下，垂直拆分会遇到什么问题呢？

这些表，以及对应的服务维护在不同的部门，看上去各业务灵活性强，研发闭环，这恰恰是悲剧的开始：

（1）tid如何规范？

（2）属性如何规范？

（3）按照uid来查询怎么办（查询自己发布的所有帖子）？

（4）按照时间来查询怎么办（最新发布的帖子）？

（5）跨品类查询怎么办（例如首页搜索框）？

（6）技术范围的扩散，有的用mongo存储，有的用mysql存储，有的自研存储；

（7）重复开发了不少组件；

（8）维护成本过高；

（9）...

画外音：想想看，电商的商品表，不可能一个类目一个表的。

四、行业最佳实践：三大中心服务

第一：统一帖子中心服务

平台型创业型公司，可能有多个品类，各品类有很多异构数据的存储需求，到底是分还是合，无需纠结：**基础数据基础服务的统一**，是一个很好的实践。

画外音：这里说的是平台型业务。

如何将不同品类，异构的数据统一存储起来呢？

（1）全品类通用属性统一存储；

（2）单品类特有属性，品类类型与通用属性json来进行存储；

更具体的：

tiezi(tid, uid, time, title, cate, subcate, xxid, ext);

（1）一些通用的字段抽取出来单独存储；

（2）通过cate, subcate, xxid等来定义ext是何种含义；

tid	uid	time	cateid	ext
1	1	123	招聘	{"job":"driver","salary":8000,"location":"bj"}
2	1	456	二手	{"type":"iphone","money":500}

（3）通过ext来存储不同业务线的个性化需求

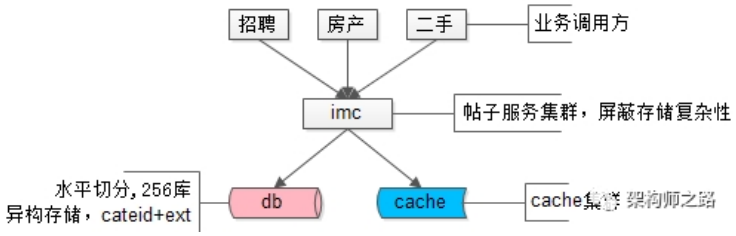
例如：

招聘的帖子，ext为：

{“job”:”driver”,”salary”:8000,”location”:”bj”}

而二手的帖子，ext为：

{”type”:”iphone”,”money”:3500}



帖子数据，100亿的数据量，分256库，通过ext存储异构业务数据，使用mysql存储，上层架了一个帖子中心服务，使用memcache做缓存，就是这样一个并不复杂的架构，解决了业务的大问题。

这是分类信息平台最核心的帖子中心服务IMC（Info Management Center）。

解决了海量异构数据的存储问题，遇到的**新问题**是：

（1）每条记录ext内key都需要重复存储，占据了大量的空间，能否压缩存储；

（2）cateid已经不足以描述ext内的内容，品类有层级，深度不确定，ext能否具备自描述性；

（3）随时可以增加属性，保证扩展性；

解决完海量异构数据的存储问题，接下来，要解决的是类目的扩展性问题。

第二：统一类目属性服务

每个业务有多少属性，这些属性是什么含义，值的约束等，耦合到帖子服务里显然是不合理的，那怎么办呢？

抽象出一个统一的类目、属性服务，单独来管理这些信息，而帖子库ext字段里json的key，统一由数字来表示，减少存储空间。

tid	uid	time	cateid	ext
1	1	123	招聘	{“1”:”driver”,”2”:8000,”3”:”bj”}
2	1	456	二手	{”4”:”iphone”,”5”:3500}

画外音：帖子表只存元信息，不管业务含义。

如上图所示，json里的key不再是”salary” ”location” ”money” 这样的长字符串了，取而代之的是数字1,2,3,4，这些数字是什么含义，属于哪个子分类，值的校验约束，统一都存储在类目、属性服务里。

key_id	cateid	sub_cateid	desc	show	正则校验	xx
1	招聘	100	job	职位	32字符	...
2	招聘	100	salary	薪水	uint类型	...
3	招聘	100	location	位置	8字符	...
4	二手	200	type	类型	short类型	...
5	二手	200	money	价格	uint类型	...

画外音：类目表存业务信息，以及约束信息，与帖子表解耦。

这个表里对帖子中心服务里ext字段里的数字key进行了解释：

（1）1代表job，属于招聘品类下100子品类，其value必须是一个小于32的[a-z]字符；

（2）4代表type，属于二手品类下200子品类，其value必须是一个short；

这样就对原来帖子表ext扩展属性：

{“1”:”driver”,”2”:8000,”3”:”bj”}

{”4”:”iphone”,”5”:3500}

key和value都做了统一约束。

除此之外，如果ext里某个key的value不是正则校验的值，而是枚举值时，需要有一个对值进行限定的枚举表来进行校验：

key_id	value_enum	show
4	1	三星
4	2	小米
4	3	魅族
4	4	vivo
4	5	iphone

这个枚举校验，说明key=4的属性（对应属性表里二手，手机类型字段），其值不只是要进行“short类型”校验，而是value必须是固定的枚举值。

{”4”:”iphone”,”5”:3500}

这个ext就是不合法的，key=4的value=iphone不合法，而应该是枚举属性，合法的应该为：

{”4”:”5”,”5”:3500}

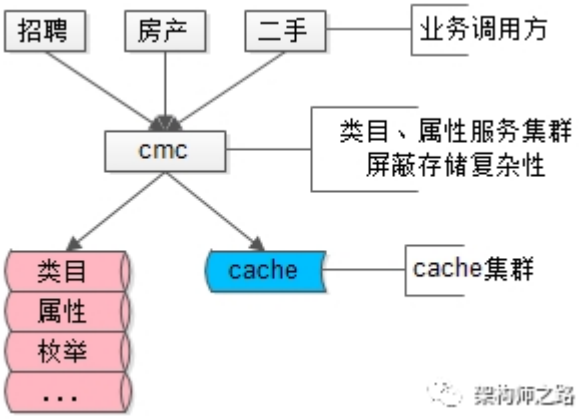
此外，类目属性服务还能记录类目之间的层级关系：

（1）一级类目是招聘、房产、二手…

（2）二手下有二级类目二手家具、二手手机…

（3）二手手机下有三级类目二手iphone，二手小米，二手三星…

（4）…



类目服务解释了帖子数据，描述品类层级关系，保证各类目属性扩展性，保证各属性值合理性校验，就是分类信息平台另一个统一的核心服务CMC（Category Management Center）。

画外音：类目、属性服务像不像电商系统里的SKU扩展服务？

（1）品类层级关系，对应电商里的类别层级体系；

（2）属性扩展，对应电商里各类别商品SKU的属性；

（3）枚举值校验，对应属性的枚举值，例如颜色：红，黄，蓝；

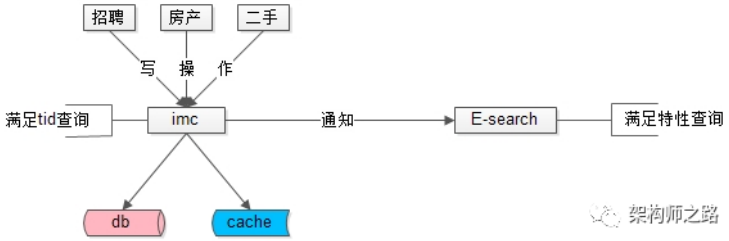
通过品类服务，解决了key压缩，key描述，key扩展，value校验，品类层级的问题，还有这样的一个问题没有解决：每个品类下帖子的属性各不相同，查询需求各不相同，**如何解决100亿数据量，1万属性的检索与联合检索需求呢？**

第三：统一检索服务

数据量很大的时候，不同属性上的查询需求，不可能通过组合索引来满足所有查询需求，“外置索引，统一检索服务”是一个很常用的实践：

（1）数据库提供“帖子id”的正排查询需求；

（2）所有非“帖子id”的个性化检索需求，统一走外置索引；



元数据与索引数据的操作遵循：

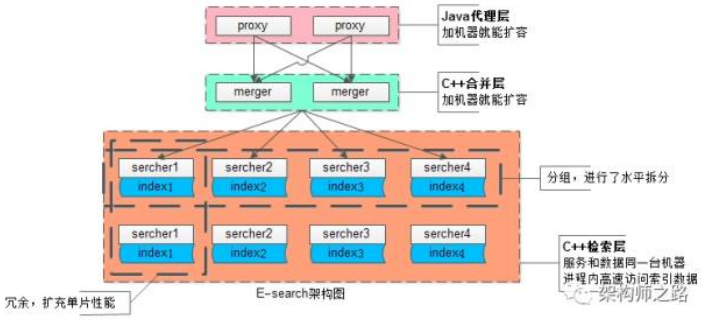
（1）对帖子进行tid正排查询，直接访问帖子服务；

（2）对帖子进行修改，帖子服务通知检索服务，同时对索引进行修改；

（3）对帖子进行复杂查询，通过检索服务满足需求；

画外音：这个检索服务，扛起了分类信息平台80%的请求（不管来自PC还是APP，不管是主页、城市页、分类页、列表页、详情页，最终都会转化为一个检索请求）。

对于这个搜索引擎架构，简单说明一下：



为应对100亿级别数据量、几十万级别的吞吐量，业务线各种复杂的复杂检索查询，扩展性是设计重点：

（1）统一的**代理层**，作为入口，其无状态性能够保证增加机器就能扩充系统性能；

（2）统一的**结果聚合层**，其无状态性也能够保证增加机器就能扩充系统性能；

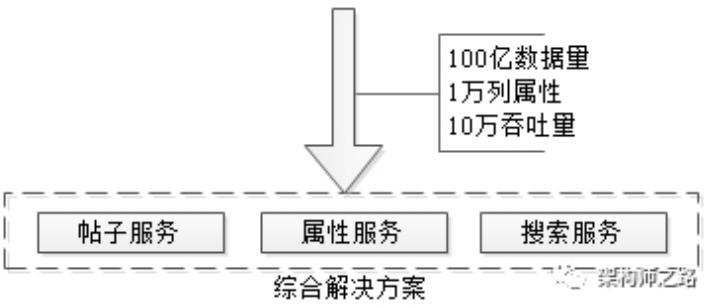
（3）搜索内核**检索层**，服务和索引数据部署在同一台机器上，服务启动时可以加载索引数据到内存，请求访问时从内存中load数据，访问速度很快：

- 为了满足数据容量的扩展性，索引数据进行了水平切分，增加切分份数，就能够无限扩展性能
- 为了满足一份数据的性能扩展性，同一份数据进行了冗余，理论上做到增加机器就无限扩展性能

系统时延，100亿级别帖子检索，包含请求分合，拉链求交集，从聚合层均可以做到10ms返回。

帖子业务，一致性不是主要矛盾，检索服务会定期全量重建索引，以保证即使数据不一致，也不会持续很长的时间。

五、总结



文章写了很长，最后做一个简单总结，面对100亿数据量，1万列属性，10万吞吐量的业务需求，可以采用了**元数据服务、属性服务、搜索服务来**解决：

（1）一个解决存储问题；

（2）一个解决品类解耦问题；

（3）一个解决检索问题；

任何复杂问题的解决，都是循序渐进的。

思路比结论重要，希望大家有收获。

推荐文章：

《被查询的列，为啥可以放到索引里？》

收录于合集 #数据库 35

下一篇 · 我被嘲笑了：被查询的列，为啥要放到索引里？（1分钟系列）

Read more