链滴

标签服务实现漫谈

作者：88250
原文链接：https://ld246.com/article/1448958541321
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

标签服务是一个较为通用的基础业务服务，比如博客系统对文章加标签、社交网络中为好友添加印象收藏的歌曲贴标签方便整理等等。

其主要提供两类接口：

- 标签实体的管理/查询：负责标签实体的 CRUD
- 标签关联的管理/查询：将外部业务实体与标签建立/删除关联，根据外部业务实体 id 查询标签集

RDB 实现

基于关系型数据库的实现是最容易的，并且上大多数应用也是这样做的。

1. 建立 tag 表，其中包含了 tag 的基础属性，例如 name、description 等
2. 建立 tag_rel 关联表，其中主要包含了 object_id、tag_id

管理服务（创建/更新/删除）的实现非常容易；根据 object_id 查询其对应的标签集也很容易实现：

```
SELECT
  tag_rel.tag_id,
  tag_rel.object_id,
  tag.name
FROM
  tag_rel
LEFT JOIN tag ON tag_rel.tag_id = tag.tag_id
WHERE
  tag_rel.object_id = '2db775c1d2174a8c67fc39b86c3fc168'
```

问题

RDB 实现标签服务时最主要会面临的是性能问题，随着 tag_rel 数据量的不断增长，上面查询的性能不断下降。

优化

数据库

- 给列 tag_rel.object_id 加普通索引
- 升级数据库服务器硬件配置
- 使用数据库产品的特性，比如表分区/内存表

在应用层面，可以缓存查询结果，如果实时性要求不高，缓存机制是比较容易实现的，但大多数业务景可能并不适用。

分表

前面提到的数据库产品的表分区特性（Oracle）我们也可以在应用层代码通过分表来实现。

- 建立多张标签关联表 tag_rel_0、tag_rel_1、...、tag_rel_31

- 将查询条件 object_id 通过某种散列算法（比如 UUID 字符串的话可以通过 FNV Hash 后取模 3）得到 [0, 31] 的结果
- SQL 落到具体的某张具体的分表上 FROM tag_rel_16

大部分应用应该通过分表就能解决性能问题，还解决不了的话可以通过这个思路进行分库、分实例。

NoSQL 实现

基于前文的背景可知，标签服务的定位是一个独立的服务，并不维护业务主体（object），所以将标嵌套到业务主体中的设计在这里并不适合。

排除了这一条，我们还是只能用 RDB 的思想来进行设计：

- 数据结构沿用 RDB 设计，即同样分为两个“表”： tag、tag_rel
- 将 SQL 中的 join 分成两步进行：1. 根据 object_id 查询出 tag_id 集合；2. 根据 tag_id 集合查出 tag 集合

Redis

Redis 介绍和“为什么选用 Redis”略过。下面我们介绍一下如何使用 Redis 实现上述思路。

数据类型

我们可以使用 Redis Hash 类型来保存标签，使用 Redis Set 来保存标签关联（这也是 Redis 官方文的示例）。

给新闻（id: 1000）添加 4 个标签（通过 id 关联）：

```
> sadd news:1000:tags 1 2 5 77
(integer) 4
```

添加反向关联：

```
> sadd tag:1:news 1000
(integer) 1
> sadd tag:2:news 1000
(integer) 1
> sadd tag:5:news 1000
(integer) 1
> sadd tag:77:news 1000
(integer) 1
```

查询该新闻的标签 id 集合：

```
> smembers news:1000:tags
1. 5
2. 1
3. 77
4. 2
```

然后根据返回的标签 id 集合查询标签。虽然分了两步进行查询，但 Redis 的高性能将为我们带来很的耗时。

结论

- 数据库分表能够解决标签服务的性能问题
  - 实现简单，在现有实现的基础上改造很小
  - 依然是基于关系型数据库，不用调整技术架构，开发模式、运维等保持不变
- 使用 Redis 同样可以实现标签服务
  - 使用 Hash 保存标签、Set 保存关联关系
  - 分两步进行查询，Redis 自身的实现解决性能问题