NoSQL 数据库存取 yago 数据

大数据管理技术第二次实习作业

张文杰 1500011394

2018 年 4 月 30 日

1 介绍

本次上机作业使用 Redis、Mongodb、Cassandra 三种数据库，利用上一次实验装好的数据库和数据库驱动程序，在我自己的电脑上，完成了对于 yago 数据的储存和查询操作。使用的操作系统为 Ubuntu 18.04，8G 内存，编程语言为 Python 3.5。报告中仅展示了部分源代码，详细源代码见 Github https://github.com/myxxxsquared/db_yago。

2 实验内容

实验的内容分为两部分，第一部分为将数据导入到相应的数据库中，第二部分为完成相应的查询，这里规定了四个查询操作。

Exp01 查询 S 为 'Mel_Thompson' 的所有的 P 和 O。

Exp02 查询 O 为 'England' 的所有的 S 和 P。

Exp03 查询含有 'graduatedFrom' 和 'isLeaderOf' 的 S。

Exp04 查询含有 'England' 最多的 S。

2.1 Redis

2.1.1 储存模式

Redis 数据库不能建立除 KEY 之外的额外索引信息，为了完成指定的查询任务，需要对于 S，O，V 都有查询，因此需要在 Redis 数据库中使用冗余的数据，才能完成指定的操作。在这里，我使用了如下的数据存储模式。

1. 建立一个名为 ‘ ’(空格) 的 SET，用于保存所有的 S 的列表。
2. 建立名为 ‘S 名称 P 名称’(S 名称 + 空格 +P 名称) 的 SET，用于保存 S 在 P 下的所有 O。
3. 建立名为 ‘O 名称’ 的 SET，用于保存所有含有 O 的 S 和 P。

以如下的数据为例

```
Richard_Stallman isLeaderOf Free_Software_Foundation
Richard_Stallman isLeaderOf Cambridge_Bay
Andranik isLeaderOf Armenian_fedayi
```

其储存在 Redis 数据库中格式为

```
' ':
    ['Richard_Stallman', 'Andranik']
'Richard_Stallman isLeaderOf':
    ['Free_Software_Foundation', 'Cambridge_Bay']
'Andranik isLeaderOf':
    ['Armenian_fedayi']
'Free_Software_Foundation':
    ['Richard_Stallman isLeaderOf']
'Cambridge_Bay':
    ['Richard_Stallman isLeaderOf']
'Armenian_fedayi':
    ['Andranik']
```

2.1.2 查询方法

对于 Redis 的查询只能使用 KEY 来进行，通过 si 查找 P 和 O，只需要找到所有的形式为 'si *' 的 KEY，即使用命令 'KEY "si *"'，然后使用 SMEMBERS 命令查询 KEY 的值即可。

通过 oi 来查找 S 和 P，只需使用数据库中已有的数据，利用 SMEMBERS 命令直接查询即可。

通过 p1、p2 查找所有同时拥有 p1、p2 的 S，可以通过查询所有拥有 p1 的 S 和所有拥有 p2 的 S，再求交集即可。查询拥有 p1 的 S 可以通过命令 'KEY "* p1"' 来完成，同理可以查询到所有拥有 p2 的 S，之后通过 intersection 求交集，可以得到同时拥有两者的 S。

通过 oi 查询拥有这样 oi 最多的 S，可以直接使用 SMEMBERS 命令查询到所有拥有这样 oi 的 si 和 pi，经过计数即可找到最多的一项。

2.1.3 遇到的问题

我测试使用的电脑内存只有 8G，由于储存的数据量较大，而 Redis 的所有数据均储存在内存中，占用大量的内存。

Redis 的数据储存到硬盘的过程直接使用了 fork 系统调用，在占用大量内存时就会导致内存的 overcommit，fork 操作失败，失败提示为

```
Can't save in background: fork: Cannot allocate memory
```

这里采用的解决方法是，始终允许系统内核 overcommit 内存，即写入一个 '1' 到 '/proc/sys/vm/overcommit_memory' 文件内，这样 fork 系统调用就不会失败。

2.2 MongoBD

2.2.1 储存模式

在 MongoDB 中，为了实现以上四种查询，在这里我使用了一种简单类似二维表的存储模式，对于每一条记录，将其储存为一个文档，例如

```
Richard_Stallman isLeaderOf Free_Software_Foundation
```

被储存为

```
{
    s: `Richard_Stallman',
    v: `isLeaderOf',
    o: `Free_Software_Foundation'
}
```

在 MongoDB 中可以使用索引来加速查询过程，在本次实验中，分别测试了没有索引和有索引的查询效率。

2.2.2 查询方法

已知 si 或 oi 查询 PO 和 SP，只需使用 MongoDB 的查询文档

```
{ s: `si' }
```

或

```
{ o: `oi' }
```

即可完成查询

通过 p1、p2 查找所有同时拥有 p1、p2 的 S，类似 Redis，可以直接查询出所有含有 p1 的 S 和所有 p2 的 S，再取交集。通过 oi 查询拥有这样 oi 最多的 S，可以通过找出所有的再进行计数。

2.3 Cassandra

2.3.1 储存模式

这里采用的也是类似二维表的储存方法，建立一个表。由于 Cassandra 必须有一个无重复的主键，因此多引入一项用于存储编号作为主键。

```
create table if not exists yago (
    i int primary key,
    s varchar,
    v varchar,
    o varchar
);
```

由于 Cassandra 的应用场景的要求，Cassandra 对数据的查询有很多 time-out 的情形，因此如果不对于数据库建立索引，会导致查询操作超时报错，因此为了满足以上的查询需求，本次实验中仅进行了建立索引后的查询实验，建立索引的操作为

```
create index if not exists sindex on yago (s);
create index if not exists vindex on yago (v);
create index if not exists oindex on yago (o);
```

2.3.2 查询方法

由于数据存储模式与 Cassandra 基本相同，查询方法也基本相同，只不过是把查询文档改为 CQL 语句。

```
select s, v, o from yago where s = %s;
select s, v, o from yago where v = %s;
select s, v, o from yago where o = %s;
```

3 实验结果和分析

四种操作分别在三个不同的数据库上进行操作，得到的操作时间如表1所示。

比较 MongoDB 的有无索引，有索引的查询速度总是大大快于无索引的查询速度。

表 1: 查询操作时间（单位秒）

	Redis	MongoDB (有索引)	MongoDB (无索引)	Cassandra
Exp01	1.351	**0.022**	10.41	0.040
Exp02	**0.779**	1.210	5.799	30.79
Exp03	8.378	**0.982**	10.25	14.63
Exp04	**1.024**	1.046	6.110	24.28

Exp02 和 Exp04 中涉及了大量的数据查询，这是由于以 'England' 为 O 的记录很多，在这样的实验中，Redis 表现出较高的效率，说明 Redis 适用于大量简单的查询操作。在这两个测试上，MongoDB 的效率略低于 Redis，但是也远远高于 Cassandra。

在 Exp01 和 Exp03，MongoDB 的速度快于 Redis，这里 Redis 需要进行 KEY 查询，一定程度上影响了速度，使得 MongoDB 速度相对较快。

Cassandra 数据库的表现除了数据量较小的 Exp01 外，表现普遍较差，可能是由于 Cassandra 数据库的设计中有冗余储存等技术，Cassandra 查询数据速度慢与另外两个数据库。

4 个人体会

在本次的上机实验中，三个数据库各有各的特点。Redis 全部使用内存存储，占用大量内存空间。MongoDB 支持一定的数据结构，使得操作和查询变得更加容易。Cassandra 更适用于分布式存储，对于本实验中单一结点的存储体验较差。

个人感觉而言，还是更喜欢 MongoDB，因为 MongoDB 操作简单，支持比较灵活的数据结构操作和查询，并且不容易发生问题。