数据库原理（1）期末报告

小组成员	
姓名	学号
汪江豪	22121630
董晓鸥	22121397

一、 题目选择：

实验 4:Spark-SQL 分布式数据库查询-推荐系统（大数据集）

二、 实验思路：

使用两个中间表 ref_count 记录每个网站的外部标签数；common 记录任意两个网站的公共标签数量。并集|A ∪ B| = |A| + |B| - |A ∩ B|,最后交集÷并集即为两个网站之间的相似度。

三、 实验步骤：

1. 准备工作：
   与实验 4 步骤基本相同，在腾讯云上租用 3 台云服务器，先搭建好基于 hadoop 的分布式文件系统，在 hadoop 目录下运行 sbin/start-all.sh，启动 hadoop，三台节点 master,slave01,slave02，查询 jps，确保 datanode 等信息出现。
2. 设置工作节点：
   (1)将 slave01,slave02 设为工作节点，master 默认作为任务调度和资源分配节点。以下给出方法：在 spark 目录下,cd conf,打开 spark-env.sh，在头部新增
   export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
   将 hadoop 下的配置文件保存到 spark 配置文件的环境变量中，以便启动 spark-sql 时使用资源分配命令。
3. 启动 spark-sql，将 slave01,slave02 设为工作节点：
   在 spark 目录下运行命令：
   bin/spark-sql --master yarn --num-executors 2
   该命令指明启动 spark-sql 时，选择 salve01,slave02 作为工作节点。
4. 将大数据集上床到 hdfs 上后，使用如下命令导入大数据集，并创建为 relation 表，列属性名为 referrer,referree，数据类型为 int。

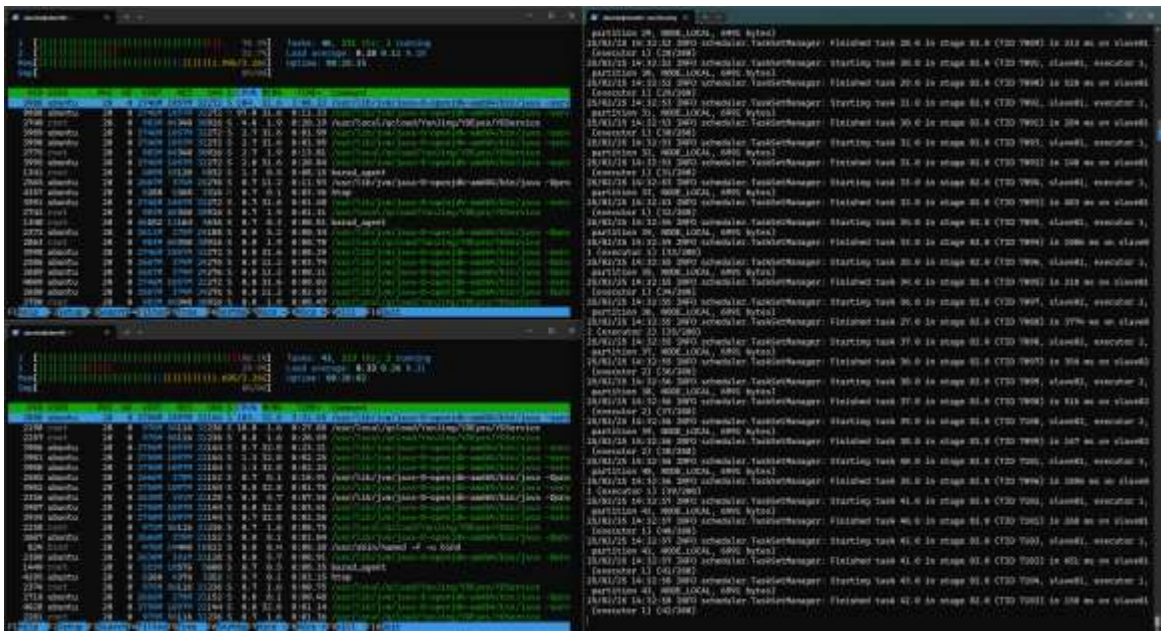
```
create table relation(
    referrer int,
    referree int
)
using csv
options(
    path "./large_relation",
    delimiter ' ',
    header 'false'
);
```

5. 使用连接方式进行查询：

语句如下，由于结果有 6400 万多行，为了节省数据打印时间，这里限制返回最多 500 行：

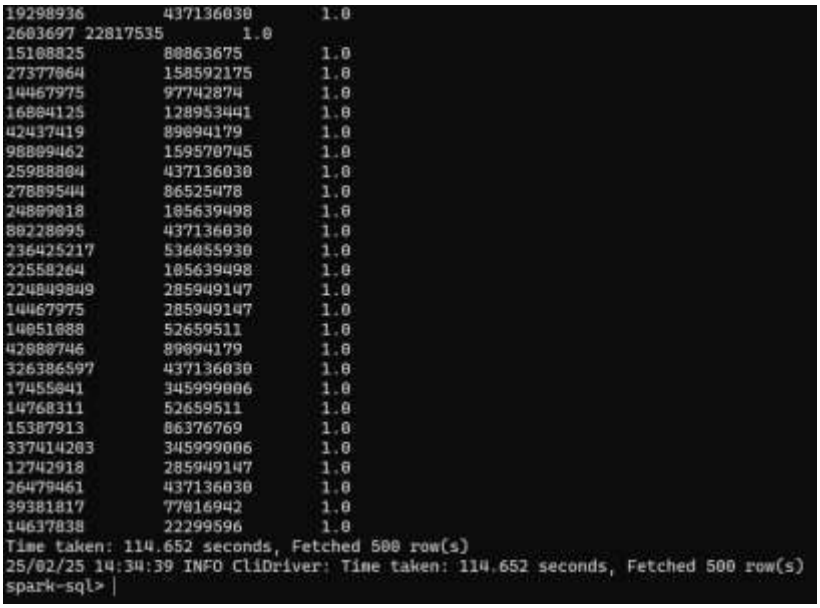
```
with ref_count as (select referrer, count(distinct referree) as
web_count
       from relation group by referrer),
common as (select a.referrer as web1, b.referrer as web2, count(*) as
com_cnt
           from relation a join relation b
           on a.referree = b.referree
           where a.referrer < b.referrer
           group by a.referrer, b.referrer)
select common.web1 as web1, common.web2 as web2,
case when r1.web_count + r2.web_count - common.com_cnt > 0
then common.com_cnt / (r1.web_count + r2.web_count - common.com_cnt)
else 0
end as similarity
from common join ref_count r1 on common.web1 = r1.referrer
join ref_count r2 on common.web2 = r2.referrer
order by similarity desc limit 500;
```

运行状况截图：



截图显示：在执行查询过程中，slave01, slave02 共 4 个 CPU 核心均处于工作状态，表明查询计算成功分布在两台机器上。

结果展示：



耗时 114.652 秒完成。

四、 更多方式尝试

为了比较不同方式速度差异，我将实验 4 的其余几种查询语句也进行了测试，sql 语句以及结果如下：

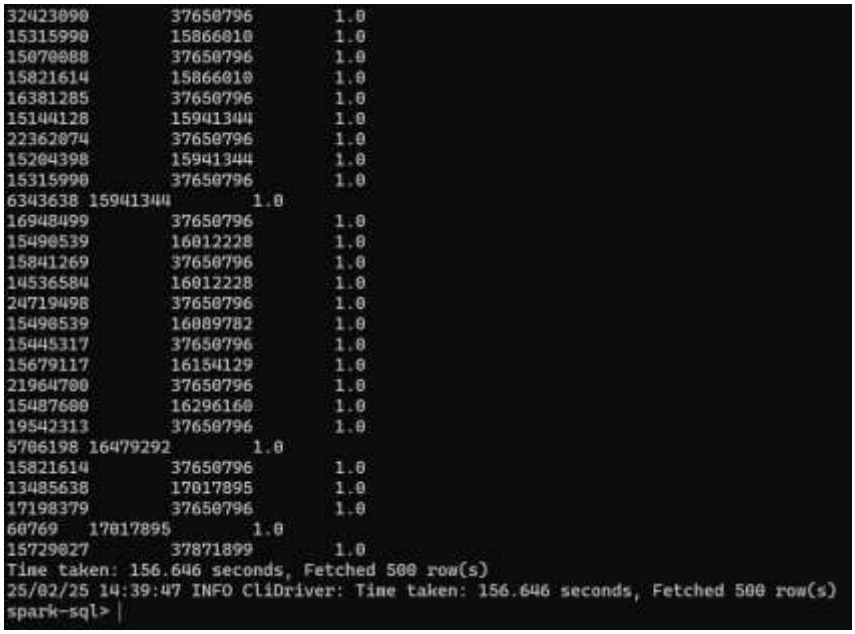
- 嵌套相关子查询

查询语句：

```
WITH ref_count AS (
    SELECT referrer, COUNT(DISTINCT referree) AS web_count
    FROM relation
    GROUP BY referrer
),
common AS (
    SELECT
        a.referrer AS web1,
        b.referrer AS web2,
        COUNT(*) AS com_cnt,
        min((SELECT web_count FROM ref_count WHERE referrer =
a.referrer)) AS web1_count,
        min((SELECT web_count FROM ref_count WHERE referrer =
b.referrer)) AS web2_count
    FROM relation a
    JOIN relation b ON a.referree = b.referree
    WHERE a.referrer < b.referrer
    GROUP BY a.referrer, b.referrer
)
SELECT
    common.web1,
    common.web2,
    CASE
```

```
      WHEN common.web1_count + common.web2_count - common.com_cnt >
0
      THEN common.com_cnt / (common.web1_count + common.web2_count -
common.com_cnt)
      ELSE 0
   END AS similarity
FROM common ORDER BY similarity DESC LIMIT 500;
```

查询结果：



耗时 156.646s 完成。

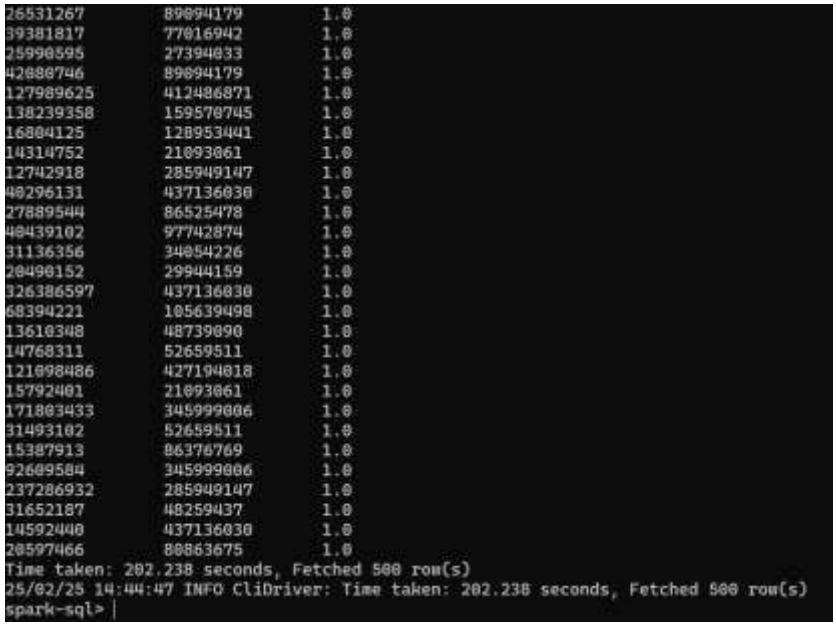
- 嵌套不相关子查询：

  查询语句：

```
WITH ref_count AS (
   SELECT referrer, COUNT(DISTINCT referree) AS web_count
   FROM relation
   GROUP BY referrer
),
common AS (
   SELECT a.referrer AS web1, b.referrer AS web2, COUNT(*) AS
com_cnt
   FROM relation a
   JOIN relation b ON a.referree = b.referree
   WHERE a.referrer < b.referrer
   GROUP BY a.referrer, b.referrer
)
SELECT common.web1, common.web2,
CASE WHEN r1.web_count + r2.web_count - common.com_cnt > 0
THEN common.com_cnt / (r1.web_count + r2.web_count - common.com_cnt)
ELSE 0 END AS similarity
```

```
FROM common JOIN (
    SELECT referrer, web_count
    FROM ref_count
    WHERE referrer IN (SELECT web1 FROM common)
) r1 ON common.web1 = r1.referrer
JOIN (
    SELECT referrer, web_count
    FROM ref_count
    WHERE referrer IN (SELECT web2 FROM common)
) r2 ON common.web2 = r2.referrer order by similarity desc limit 500;
```

查询结果：



耗时 202.238s 完成

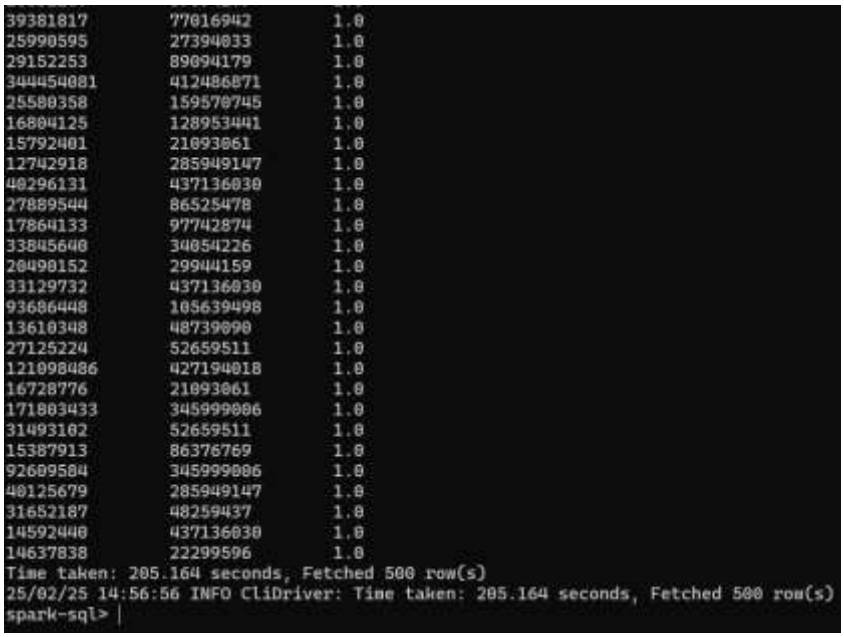
• intersect 操作：

查询语句：

```
WITH ref_count AS (
    SELECT referrer, COUNT(DISTINCT referree) AS web_count
    FROM relation GROUP BY referrer),
 common as (select a.referrer as web1, b.referrer as web2, count(*)
as com_cnt
           from relation a join relation b
           on a.referree = b.referree
           where a.referrer < b.referrer
           group by a.referrer, b.referrer)
SELECT common.web1, common.web2,
CASE WHEN r1.web_count + r2.web_count - common.com_cnt > 0
THEN common.com_cnt / (r1.web_count + r2.web_count - common.com_cnt)
ELSE 0 END AS similarity
FROM common JOIN (
    SELECT referrer, web_count FROM ref_count
```

```
    WHERE referrer IN (
        SELECT web1 FROM common
        INTERSECT
        SELECT referrer FROM ref_count)
) r1 ON common.web1 = r1.referrer
JOIN (
    SELECT referrer, web_count
    FROM ref_count
    WHERE referrer IN (
        SELECT web2 FROM common
        INTERSECT
        SELECT referrer FROM ref_count)
) r2 ON common.web2 = r2.referrer order by similarity desc limit 500;
```

查询结果：



耗时 205.164s 完成

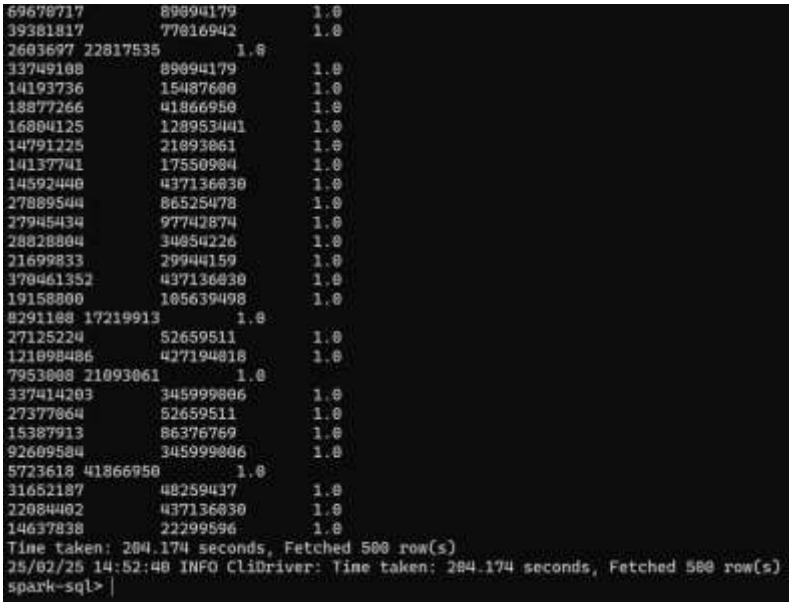
- exists 方式：

  查询语句：

```
WITH ref_count AS (
    SELECT referrer, COUNT(DISTINCT referree) AS web_count
    FROM relation
    GROUP BY referrer),
common AS (
    SELECT a.referrer AS web1, b.referrer AS web2, COUNT(*) AS
com_cnt
    FROM relation a
    JOIN relation b ON a.referree = b.referree
    WHERE a.referrer < b.referrer
    GROUP BY a.referrer, b.referrer)
```

```sql
SELECT
    common.web1,
    common.web2,
    CASE
        WHEN r1.web_count + r2.web_count - common.com_cnt > 0
        THEN common.com_cnt / (r1.web_count + r2.web_count -
common.com_cnt)
        ELSE 0
    END AS similarity
FROM common JOIN (
    SELECT referrer, web_count
    FROM ref_count r
    WHERE EXISTS (SELECT 1 FROM common c WHERE c.web1 = r.referrer)
) r1 ON common.web1 = r1.referrer
JOIN (
    SELECT referrer, web_count
    FROM ref_count r
    WHERE EXISTS (SELECT 1 FROM common c WHERE c.web2 = r.referrer)
) r2 ON common.web2 = r2.referrer
ORDER BY similarity DESC LIMIT 500;
```

查询结果：



耗时 204.174s 完成。

五、 查询验证

为了方便后续查询验证，我先前将临时表 ref_count,common 保存为了中间表 tmp_ref_count,tmp_common，相似度计算结果表保存到了 result。

查询 result 表数据行数：

```
25/02/25 14:59:20 INFO scheduler.DAGScheduler: Job 43 finished: processCmd at CliDriver.java:376, took 6.931805 s
64195979
Time taken: 7.678 seconds, Fetched 1 row(s)
25/02/25 14:59:20 INFO CliDriver: Time taken: 7.678 seconds, Fetched 1 row(s)
spark-sql>
```

耗时 7678s 完成，显示结果 64195979 行，可见表及其庞大。

1. 查询相似度位于 0.72~0.73 的记录，返回最多 30 条。

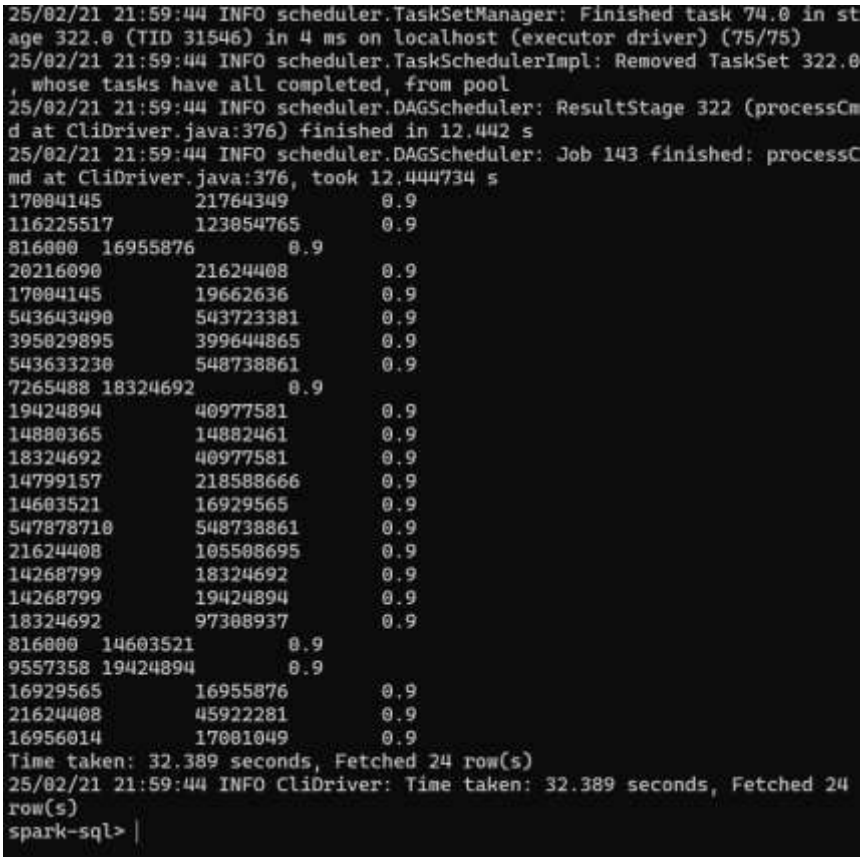
```
25/02/21 21:55:42 INFO CliDriver: Time taken: 19.932 seconds, Fetched 30
row(s)
spark-sql> select * from result where similarity between 0.72 and 0.73 li
mit 30;
```

```
25/02/21 21:57:46 INFO scheduler.TaskSetManager: Finished task 99.0 in st
age 317.0 (TID 31346) in 227 ms on localhost (executor driver) (100/100)
25/02/21 21:57:46 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 317.0
, whose tasks have all completed, from pool
25/02/21 21:57:46 INFO scheduler.DAGScheduler: ResultStage 317 (processCm
d at CliDriver.java:376) finished in 15.950 s
25/02/21 21:57:46 INFO scheduler.DAGScheduler: Job 138 finished: processC
md at CliDriver.java:376, took 15.953909 s
291245333       296171249       0.7297297297297297
25227543        334462096       0.7297297297297297
99786878        122113661       0.7297297297297297
28539307        437857716       0.7297297297297297
15594319        29725736        0.7297297297297297
14667720        500021610       0.7297297297297297
42098840        148230772       0.7297297297297297
19040457        46970699        0.7297297297297297
131785632       202186645       0.7297297297297297
24033843        64058908        0.7297297297297297
18888315        28393099        0.7297297297297297
16987309        30211029        0.7295918367346939
17677642        18559386        0.7295597484276729
57046183        69306513        0.7295081967213115
16809676        16929288        0.7295081967213115
79378664        90433572        0.7295081967213115
15594319        35415483        0.7294117647058823
18359358        19040457        0.7294117647058823
291245333       370352656       0.7294117647058823
286481558       338614684       0.7294117647058823
16541224        260975690       0.7294117647058823
18359358        177761713       0.7294117647058823
128496489       273186922       0.7294117647058823
395472459       523832662       0.7294117647058823
1976847 45869579        0.7293233082706767
1976847 17023350        0.7293233082706767
1976847 16956336        0.7293233082706767
294854141       368829932       0.7292817679558011
397464137       554003477       0.7292817679558011
327129  9659998 0.7291666666666666
Time taken: 19.945 seconds, Fetched 30 row(s)
25/02/21 21:57:46 INFO CliDriver: Time taken: 19.945 seconds, Fetched 30
row(s)
spark-sql>
```

耗时 19.945s 完成查询，返回了 30 行数据。

2. 查询相似度为 0.9 的记录，最多 30 条。

```
25/02/21 21:57:46 INFO CliDriver: Time taken: 19.945 seconds, Fetched 30
row(s)
spark-sql> select * from result where similarity=0.9 limit 30;
```



耗时 32.389s 完成查询，返回了 24 项相似度为 0.9 的数据。

至此，实验完成。

六、 实验总结

本次实验基于 Hadoop HDFS 和 Spark SQL 构建分布式数据库查询系统，我深入理解了大数据存储与处理的分布式架构。通过实验，掌握了 HDFS 的文件存储机制及 Spark SQL 的分布式查询优化技术，体验了其在海量数据下的高效查询性能。实验过程中，数据分区、并行计算和内存优化等策略显著提升了查询效率，同时也认识到集群资源调度与数据倾斜问题的挑战。总体而言，我熟悉了手动搭建 hadoop 和 spark-sql 的基本流程，学会了对大数据集查询的 sql 语句进行一定程度优化，本学期的数据库实验让我受益匪浅！