Touchstone—Github版本使用说明

> 编撰人：李宇明，王清帅
>
> 单位：华东师范大学 数据科学与工程学院

在Github仓库的running examples中，我们提供了可直接运行的jar打包程序，运行环境为：**Linux & Java 1.8+**，本篇文档将对程序的使用做出详细说明。

概述

程序文件概述

在running examples文件夹中，包含3个可执行文件，分别是Touchstone.jar，RunController.jar和RunController.jar。其中Touchstone.jar负责在集群中自动化部署运行环境并启动生成任务，RunController.jar和RunDataGenenrator.jar负责在集群中执行数据生成任务，概要说明如下：

1. Touchstone.jar，部署和启动程序。程序运行时会根据配置内容，将执行文件和配置文件拷贝到集群的运行节点中，拷贝完成后启动对应节点上的程序执行生成任务。
2. RunController.jar，由Touchstone.jar在部署之后自动启动，只有一个节点运行该程序，是集群的管理节点，负责管理负载生成任务的中各个RunDataGenenrator.jar生成任务，通过netty网络框架进行通信（发送数据生成任务和Join Information Table）。
3. RunDataGenenrator.jar，由Touchstone.jar在部署之后自动启动，可以运行在多个集群节点中，分布式并行执行数据生成任务，由集群中的RunController.jar程序文件分配运行时信息。

配置文件概述

集群启动前需要编写集群配置文件和负载生成任务的配置文件。前者配置集群运行时需要的节点，并发度，运行路径等信息，后者包含生成Table的Schema信息和负载语句信息两个主要配置文件，下面在配置集群运行文件和配置负载生成任务两个章节对相关配置参数做了具体说明。说明配置文件格式之后，在章节集群标准配置文件样例中，我们给出了TPC-H和SSB的配置样例以供参考。

运行方式

Touchstone的启动程序是Touchstone.jar，在编写完成配置文件之后，使用如下命令即可启动程序

```
java -jar Touchstone.jar XXX.conf
```

运行结果

- 实例化的查询参数写在了Touchstone controller的日志中，在日志文件中搜索"Final instantiated parameters"进行定位，这里的参数顺序与输入基数约束中的符号参数顺序相同。
- 生成的表数据文件在data generator配置的生成路径下。

配置集群运行文件

注意： 所有集群配置文件需要编写在一个文件中，样例如running examples文件夹下的 touchstone.conf

1. Touchstone.jar的配置文件

   由于该程序文件需要拷贝运行文件到集群中，并且需要启动集群任务，因此需要配置集群节点的IP，用户名和密码，配置项为IPs of servers，user names of servers和passwords of servers，每个节点的配置顺序需要保持一致。程序在运行时默认会清空所有节点的操作系统缓存，确保程序在运行过程中不会因为内存不足而出现JVM GC，若没有内存不足的情况可以不配置root密码，样例配置文件如下。

   ```
   ## configurations of servers

   IPs of servers: 10.11.1.190; 10.11.1.191; 10.11.1.192
   password of root user: w@ngl5i
   user names of servers: touchstone; touchstone; touchstone
   passwords of servers: 123456; 123456; 123456
   ```

2. RunController.jar的配置文件

   对于该程序文件，需要配置在集群中用作controller的节点ip和程序运行的文件路径，以及和dataGenerator交互时的ip端口号。例如下面的配置项，配置controller的运行节点为10.11.1.190，端口号为32100，执行文件路径为~//icde_test

   ```
   ## configurations of controller

   IP of controller: 10.11.1.190
   port of controller: 32100
   running directory of controller: ~//icde_test
   ```

   在controller运行时，需要加载数据生成任务的配置文件，包括待生成数据库的schema和目标测试Query上的基数约束，这两项配置文件的具体形式将在下面做出具体介绍，配置controller的配置文件输入项如下：

   ```
   ## input files

   database schema: .//input//tpch_schema_sf_1.txt
   cardinality constraints: .//input//tpch_cardinality_constraints_sf_1.txt
   ```

   针对复杂的过滤谓词以及非等值连接谓词，Touchstone需要利用Mathematica提供的数值积分运算功能（对应论文中的random sampling算法），但是如果输入负载中仅包含简单的过滤谓词（如col > para）和等值连接谓词，则无需配置，配置Mathematica的方式为

   ```
   ## configuration of Mathematica

   path of JLink: C://Mathematica//10.0//SystemFiles//Links//JLink
   ```

运行时日志记录方式采用log4j框架，可以加载自定义配置文件，配置项为

```
## configuration of log4j

path of log4j.properties: .//lib//log4j.properties
```

3. RunDataGenerator.jar

   该程序文件需要配置五个参数，分别为：

   1. 在集群中用作data generator节点的ip
   2. 程序运行的文件路径
   3. 和controller交互的端口号
   4. 当前程序用于数据生成的线程数
   5. 数据输出的文件夹

   下面给出了一组配置实例，在191，192两台机器上配置了data generator程序，每台机器上运行3个实例，每个实例运行2个生成线程，每台机器上的端口号为32101; 32102; 32103，输出文件夹为.//data。

```
## configurations of data generators

IPs of data generators: 10.11.1.191; 10.11.1.191; 10.11.1.191;
10.11.1.192; 10.11.1.192; 10.11.1.192
running directories of data generators: ~//icde_test//dg1;
~//icde_test//dg2; ~//icde_test//dg3; ~//icde_test//dg1;
~//icde_test//dg2; ~//icde_test//dg3
ports of data generators: 32101; 32102; 32103; 32101; 32102; 32103
thread numbers of data generators: 2; 2; 2; 2; 2; 2
data output path: .//data
```

   由于一个JVM中启动多个数据生成线程的性能往往没有多个JVM中启动相同数量数据生成线程的性能好，所以建议在一个节点上根据CPU物理核数启动多个JVM。上面的示例配置在每个物理节点上启动了3个JVM，每个JVM中启动了2个数据生成线程。所有运行目录会自动创建，无需人工创建。

4. Touchstone运行过程中所需的一些参数

   这部分参数一般不需更改，可直接使用默认值，详细含义可查看论文。

```
## running parameters

thread numbers of query instantiation: 2
maximum iterations of query instantiation: 20
global relative error of query instantiation: 0.0001
maximum iterations of parameter instantiation: 20
relative error of parameter instantiation: 0.0001
maximum number of shuffle: 1000
maximum size of PKVs: 10000
```

配置负载生成任务

Touchstone有两个输入数据文件，分别包含了数据库Schema信息（含数据特征信息）和基数约束信息。

数据库Schema信息

1. Schema信息

```
## 输入格式

T[table_name; table_size; column_name, data_type; ...; P(primary_key);
F(foreign_key, referenced_table.referenced_primary_key); ...]

## 输入样例，以TPC-H的表PARTSUPP为例

T[PARTSUPP; 8000000; PS_PARTKEY, INTEGER; PS_SUPPKEY, INTEGER;
PS_AVAILQTY, INTEGER; PS_SUPPLYCOST, DECIMAL; PS_COMMENT, VARCHAR;
P(PS_PARTKEY, PS_SUPPKEY); F(PS_PARTKEY, PART.P_PARTKEY); F(PS_SUPPKEY,
SUPPLIER.S_SUPPKEY)]
```

2. 基本数据特征

```
# Integer:
D[table_name.column_name; null_ratio; cardinality; min_value; max_value]
# 示例:
D[T1.c1; 0.05; 100; 1; 9999]

# Real & Decimal:
D[table_name.column_name; null_ratio; min_value; max_value]
# 示例:
D[T1.c2; 0; 0.1; 1000.5]

# Varchar
D[table_name.column_name; null_ratio; avg_length; max_length]
# 示例
D[T1.c3; 0; 123.5; 199]

# Bool:
D[table_name.column_name; null_ratio; true_ratio]
# 示例:
D[T1.c4; 0.2; 0.6]

# DateTime:
D[table_name.column_name; null_ratio; begin_time; end_time]
示例:
D[T1.c5; 0; 1992-01-02-00:00:00; 1998-12-01-00:00:00]
```

```
# Date:
D[table_name.column_name; null_ratio; begin_time; end_time]
# 示例:
D[T1.c6; 0; 1992-01-02; 1998-12-01]
```

负载特征（约束链中的基数约束）

约束链中含有三类基数约束，基本数据结构如下：

1. 选择节点：

   ```
   Filter node: [0, exp1@op1#exp2@op2 ... #and|or, probability]
   ```

   说明：

   - "0"标示了这是一个选择节点；
   - "exp1@op1#exp2@op2 ... #and|or"描述的是当前选择操作中的所有选择条件；
   - "probability"是选择率。
2. 主键节点（当前数据表在等值连接操作中的属性是主键属性）

   ```
   PKJoin node: [1, pk1#pk2 ..., num1, num2, ...]
   ```

   说明：

   - "1"标示了这是一个主键节点；
   - "pk1#pk2 ..."是主键（若主键是复合主键则这里为一个属性集合）；
   - "num1" & "num2"是维护主键连接信息时的编码。
3. 外键节点（当前数据表在等值连接操作中的属性是外键属性）

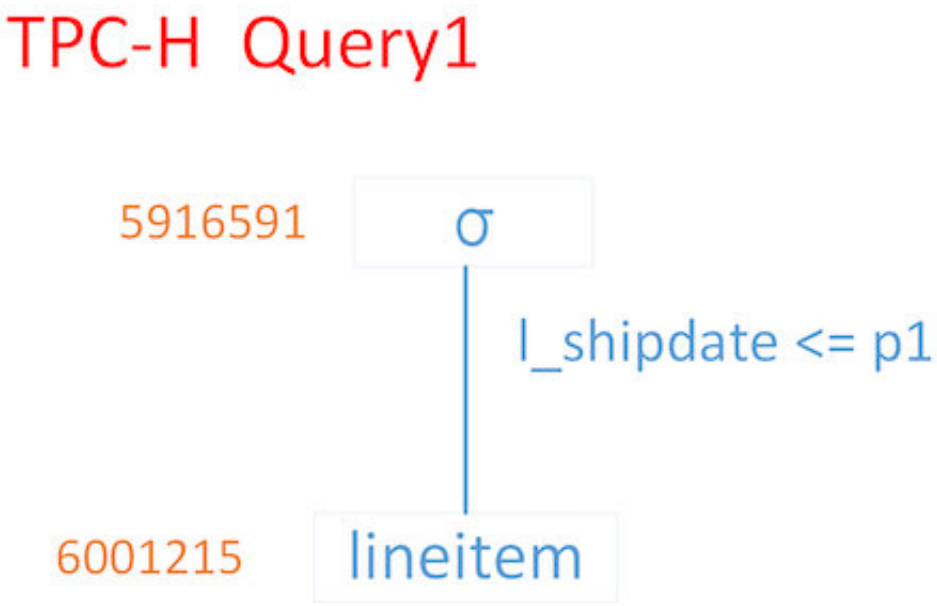
   ```
   FKJoin node: [2, fk1#fk2 ..., probability, pk1#pk2 ..., num1, num2]
   ```

   说明：

   - "2"标示了这是一个外键节点；
   - "fk1#fk2 ..."是外键属性集合；
   - "probability"是连接率；
   - "pk1#pk2 ..."为参照的主键（若参照的是复合主键则这里为一个属性集合）；
   - "num1" & "num2"是相应主键连接信息的编码。

下面会根据TPC-H的一些几个Query示例输入来介绍输入数据的具体格式。

1. TPC-H的Query 1在MySQL上的物理查询树



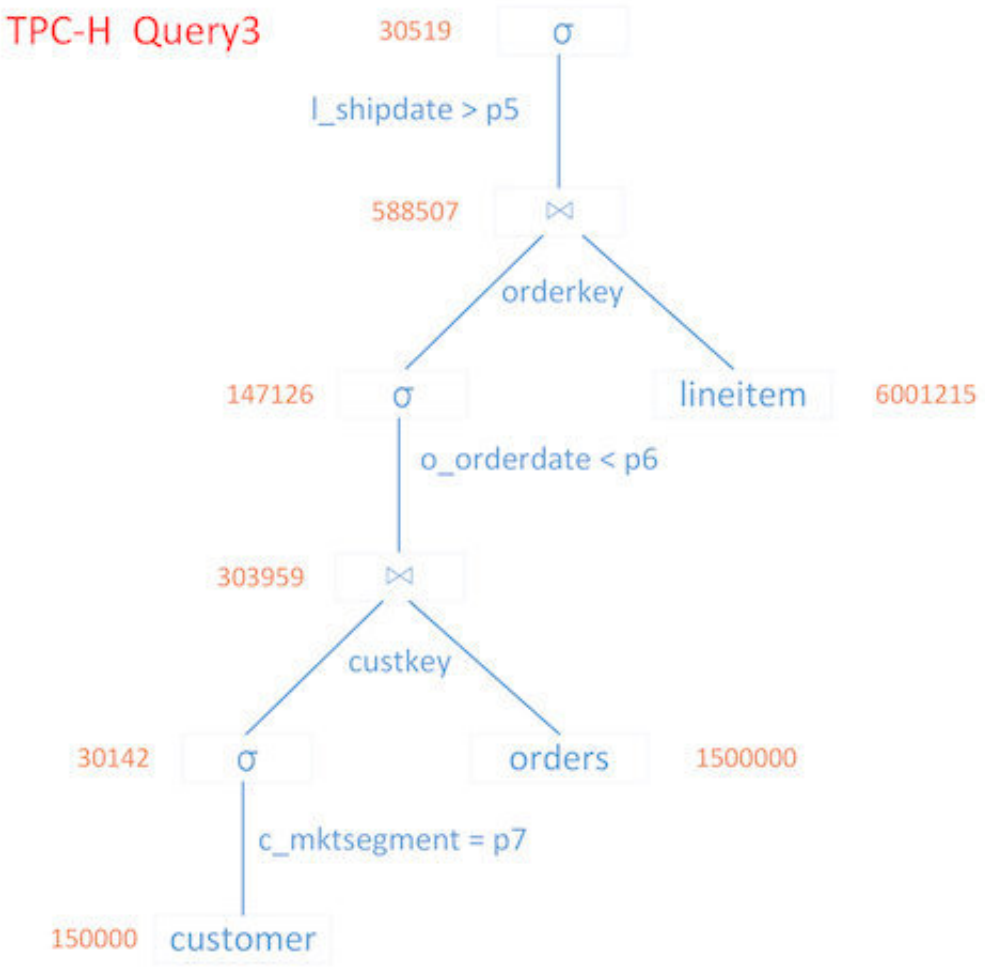
对应的约束链为：

```
[lineitem]; [0, l_shipdate@<=, 0.985899]
```

说明：

- 一条约束链是针对某个特定数据表的，这里[lineitem]标示了该条约束链是针对数据表lineitem的。
- [0, l_shipdate@<=, 0.985899]
  - "0"标示了这是一个Filter约束节点；
  - "l_shipdate@<="用于描述选择谓词"l_shipdate <= p1"；
  - "0.985899"是选择率（=5916591/6001215）。

2. TPC-H的Query 3在MySQL上的物理查询树



对应的约束链为：

```
[customer]; [0, c_mktsegment@=, 0.20095]; [1, c_custkey, 1, 2]
[orders]; [2, o_custkey, 0.20264, customer.c\_custkey, 1, 2]; [0,
o_orderdate@<, 0.48403]; [1, o_orderkey, 1, 2]
[lineitem]; [2, l_orderkey, 0.09806464, orders.o_orderkey, 1, 2]; [0,
l_shipdate@>, 0.05185835]
```

说明：

因Query 3涉及3个数据表，所以这里有三条约束链。

第一条约束链中的"[1, c_custkey, 1, 2]"，第一个"1"标示了这是一个PK约束节点（customer表在这个等值连接操作中的属性是主键属性），后面的"1, 2"是维护主键连接信息时用的编码。

第二条约束链中的"[2, o_custkey, 0.20264, customer.c_custkey, 1, 2] "，第一个"2"标示了这是一个FK约束节点（orders表在这个等值连接操作中的属性是外键属性）。"o_custkey"是外键属性名，"0.20264"是连接率（=303959/1500000），"customer.c_custkey"是参照的主键，"1, 2"是该FK约束节点所在连接操作相对应的那个PK约束节点（即"[1, c_custkey, 1, 2]"）中的编码。

所有基数约束中的probability要么是选择率，要么是连接率，都是根据实际查询树中的中间结果集大小计算而来的。

> **关于编码的解释：**
>
> 当customer表的某个tuple满足了"c_mktsegment = p7"选择操作，那么这个tuple的主键c_custkey就可能与接下来的orders表连接上，对于这样的c_custkey我们就给它打上标签"1"；对于那些不符合"c_mktsegment = p7"选择操作的c_custkey就打上标签"2"以标示其肯定与后面的orders表连接不上。编码都是2^n，目前由输入保证，其实可以由程序自动生成这些编码，而不需要人工输入，后续会提供额外工具实现。
>
> **为什么编码是2^n呢?**
>
> 因为需要用当前tuple的主键在所有约束链上的编码之和来表示其连接状态（相当于有n个位，每两个位对应一个连接操作，并且这两个位中只能有一个为1，以标示这个主键是否可以连接上）！

集群标准配置文件样例

系统运行程序的标准配置文件请见：touchstone.conf

- sf=1时的tpch标准配置文件

  shema配置文件：tpch_schema_sf_1

  前16个语句的约束配置文件：tpch_cardinality_constraints_sf_1.txt

  前16个语句的约束配置示例图：TPC-H Query1-16 SF=1.png

- sf=1时的ssb标准配置文件

  shema配置文件：ssb_schema_sf_1

  约束配置文件：ssb_cardinality_constraints_Q1-Q4.txt