Université Libre de Bruxelles

INFO-H-419: Data Warehouses

Project Assignment

Part I: Implementation of the TPC-DS benchmark

Authors:
Gonçalo Moreira
Nazrin Najafzade
Rémy Detobel
Shafagh Kashef
Group: 4

Supervisor:
Prof. Esteban Zimányi

November 1, 2018

Contents

1 Introduction

Databases have been around for a long time. However, in recent years the number of data has exploded. Although the cost of storage has dropped drastically, all this data must be saved and above all recovered.

There are therefore many databases available and it is not easy to find the database best suited to our needs. TPC-DS is a decision support benchmark. Indeed, this tool offers the possibility to test all databases in the same way.

Usually this tool is used for "Big Data". We will see in this paper results that we can have with MariaDB (which is a fork of MySQL). In the second part of the project we will compare this result with other "SQL" database (like PostgreSQL or MySQL).

2 Objective

The main goal of this project is to manage Big Data with SQL language and use the TPC benchmarks to evaluate and analyze the performance.

As mention in the introduction, we will run the TPC-DS tools on MariaDB. We first search if someone have already done this task. And indeed, there is a github repository (see [1]) where some parameters have been adapted to MariaDB syntax. But this repository is more than one year old.

We also find a repository where some issues in TPC-DS tool have been fixed (see reference [2]). Indeed, TPC-DS is actually at the version 2 but the project is not very open and thus a part of the community update this repository.

So, after all this research, the first goal was to install MariaDB. On Ubuntu (and Debian) the default version of MariaDB is the 10.1. The version 10.2 is totally supported by Ubuntu and the version 10.3 is stable but still occasionally bugs from time to time on Ubuntu. But we will detail the choice of the MariaDB version in the next point.

The second objective was to build, launch and extract result with the TPC-DS Tool. Build is easy (the generator doesn't depend of the database). For the launch we had to adapt some requests or even exclude some of them. Results are presented at the end of this document.

3 TPC

TPC is a corporation that defines transaction processing and database benchmarks and disseminates objective, verifiable TPC performance data to the industry. Regarding data warehouses, two TPC benchmarks are relevant:

- TPC-DS, the Decision Support Benchmark, which models the decision support functions of a retail product supplier.

- TPC-DI, the Data Integration Support Benchmark, which models a typical ETL process that loads a data warehouse.

3.1 TPC-DS

TPC-DS is the de-facto industry standard benchmark for measuring the performance of decision support solutions including, but not limited to, Big Data systems. The current version is v2. It models several generally applicable aspects of a decision support system, including queries and data maintenance. Although the underlying business model of TPC-DS is a retail product supplier, the database schema, data population, queries, data maintenance model and implementation rules have been designed to be broadly representative of modern decision support systems.

This benchmark illustrates decision support systems that:

- Examine large volumes of data
- Give answers to real-world business questions
- Are characterized by high CPU and IO load
- Are periodically synchronized with source OLTP databases through database maintenance functions
- Run on “Big Data” solutions, such as RDBMS as well as Hadoop/Spark based systems

4 MariaDB

We choose MariaDB as our software to implement the TPC-DS benchmark because we wanted an open-source, free solution, easy-to-use and with the standard SQL capabilities.

MariaDB has united transactional and analytic processing with a single front end to deliver an enterprise-grade solution that simplifies high-performance, big data analytics. It is a fork of the MySQL database management system.

MariaDB is often update. The last version is the 10.3 (released on 4.10.2018). In this last version (10.3), keywords `EXCEPT` and `INTERSECTION` have been implemented. Notice that we use MariaDB 10.2 because 10.3 is not all the time good supported by Ubuntu (indeed, by default, version 10.1 is used).

4.1 Update

We had some problem with some computer to update MariaDB. The easy way is to install default MariaDB version (`apt-get install mariadb-server`) and then update the source list. To make that you can directly use a scrip of MariaDB:

```
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup
    sudo bash -s -- --mariadb-server-version=mariadb-10.2
```

Then just execute agin `apt-get install mariadb-server`

5 Implementation

The project is published on Github[1]. To simplify implementation, the project contains a Makefile to automatically execute all commands.

This Make file is composed of two big part: initialize and launch test. The initialize part download all required software (`curl`, `git`...), clone the TPC-DS repository (see point 2). After this clone we replace some file (to adapt queries to MariaDB) and finally we build the TPC-DS tool.
The test part generate data, load the data into the database, generate the queries (template are transform to real queries) and then execute them.
To execute this two big part you need to execute:

```
make
make test_all
```

Notice that the Makefile have also other option like clear/remove but also a command to install MariaDB or to generate report (with the last test results).

The following points will detail the different steps.

5.1 Initialize

The initialize part begin to install some component: wget, git, byacc, bison, gcc and flex. Useful for the following steps.
The TPC-DS github (see [2]) (define in the Makefile configuration) is clone. Some file of this github are replaced by our files (updated for MariaDB).
After that the TPC-DS Tool is build (all the build is already configure in the TPC-DS Tool).

5.2 Test

To test MariaDB we need to have data. So a first step is to generate data. The scale of the data generated could be chose in the Makefile configuration (the scale factor is in Gigabyte). The TPC-DS Tool generate file, but we need data into the MariaDB. So the next step is to load all this data inside the database. We also directly execute indexing on this database. Then the template (present natively in TPC-DS) are transform to real SQL queries. We notice that some template are transform to two queries. Thus the query 80, is not all the time the template 80. But all the information are specified in the results. When all the queries are generated

[1] `https://github.com/detobel36/tpcds-mariadb`

(from template to query) we can execute them. We record the time required to execute each of them.

6 Problems

The installation of MariaDB is easy on Linux environment. Indeed, a lot of distribution (Ubuntu, Debian...) support natively MariaDB (just execute "`apt-get install mariadb-server`"). But it is not so easy on windows and automating queries is not as simple as in a UNIX environment. Note that some test have been made on "Ubuntu on Windows" (which is a new feature of Windows 10).
After the installation of MariaDB we see that some queries aren't support. Indeed, TPC-DS tools have 99 queries and some keywords in these queries aren't support by MariaDB 10.1.
In the next subsections we will describe each keyword problem.

The whole error list has been added to the following link: `https://github.com/detobel36/tpcds-mariadb/blob/master/ChangelogTemplates.md`|

6.1 MariaDB: "WITH" keyword

The first problem was the keyword `WITH`. A lot of queries are based on this keywords. It is possible to adapt the query to not use them but it would have increased the compute time and therefore distorted the results. We see that in the version 10.2 of MariaDB[2] this keyword is supported. We choose to update from MariaDB 10.1 to 10.2.

6.2 MariaDB: "ROLLUP" keyword

The template in the TPC-DS tool use the syntax `group by rollup (columns)` to make rollup. On MariaDB, the syntax is a little bit different. We update thus the templates 5, 18, 22, 27, 67 and 77.

6.3 MariaDB: "+ x days"

A lot of databases allow to directly add some days to a date. In TPC-DS we can find queries with the following syntax: `date + <x> days`. For instance in the template 77:

```
d_date between cast('[SALES_DATE]' as date)
    and (cast('[SALES_DATE]' as date) +  30 days)
```

This syntax is not recognize by MariaDB. To make this we use the operator `ADDDATE`. The query 77 have been, for example, transform to:

[2] `https://mariadb.com/kb/en/library/with/`

```
d_date between cast('[SALES_DATE]' as date)
    and ADDDATE(cast('[SALES_DATE]' as date), 30)
```

Note that it is the same with the subtraction, where we change `- <x> days` by `SUBDATE`.

6.4 MariaDB: Table alias

The syntax of MariaDB require to give name of all derived table (even if we don't use directly this table). Thus to fix this problem we just add `AS test` (so we give the name "test" to the derived table) to some templates. This is the case for the templates: 2, 23 and 49.

6.5 MariaDB: Concatenation

In the default TPC-DS request, the symbol `||` is used to make a concatenation. But in MariaDB we use the keyword `CONCATENATION`. We thus updated the template 5 to make it working with MariaDB.

6.6 MariaDB: "FULL OUTER JOIN" keywords

The keyword "FULL" doesn't exist in MariaDB. To make an `OUTER JOIN`, we must to use the keyword `LEFT`. Concretely we just replaced by: `LEFT OUTER JOIN` in the templates 51 and 97.

6.7 MariaDB: "ORDER BY" and "GROUP BY"

It is not possible in MariaDB to have an `ORDER BY` and a `GROUP BY <col> WITH ROLLUP`. We need to chose, so we remove this order constrain. But it is possible to add order option directly in the `GROUP BY`. The syntax is not complicated: `GROUP BY <column> [ASC/DESC]` (to chose between ascending or descending order).

6.8 MariaDB: "INTERSECTION", "EXCEPT" keywords

This two keywords `"INTERSECTION"`[3] and `"EXCEPT"`[4] doesn't exist on MariaDB 10.2. But are implemented on MariaDB 10.3. This version is not all the time stable on Ubuntu. Thus we don't test all queries with this version. For this reason we don't have complete results for the templates: 8, 14, 38 and 87.

[3]https://mariadb.com/kb/en/library/intersect/
[4]https://mariadb.com/kb/en/library/except/

6.9 MariaDB: "GROUPING" keywords

The keyword `GROUPING` is used with `ROLLUP` to see if the current row is the resulting of the aggregation (where SQL sum some columns) or just a normal row. This keyword is not yet implemented in MariaDB.
Thus templates: 36, 70, 86 could not be executed (for the 27 we just remove the "grouping" keyword).

6.10 MariaDB: tipo problem

Some little tipo have been a problem. For instance in the template 48 there was a space between `sum` and the parenthesis. We just fix the problem by removing it. We have the same problem with the keyword `cast` (in the template 40 for instance).

7 Results

We made two tests: one with 1Go and another with 2Go of data. The scale if the data is not huge because we don't have computer/server with enough powerful and/or memory to test more data. Notice also that MariaDB is not made to manage a big size of data. All the results are in appendix.

7.1 Server information

- CPU: Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
- RAM: 2Go
- Hard disk: SSD 20Go
- Database: 10.2.18-MariaDB-10.2.18+maria stretch
- OS: Debian 9 Stretch (64 bits)
- Notice: other process running during the test

7.2 Discussion

First of all we can see that some result haven't any template. It is because (like mention in the point 5.2) some template are split in two queries. For instance query 22 is group with the query 21 and correspond to the template 23. Note also that some template are remove (like mention in point 6) because the syntax could not be fix for MariaDB.
We can logically notice that the execution time is longer for 2 Go than for 1 Go. But it's interesting to note this evolution. We can look this evolution globally (if we sum all the queries). But to make that we first need to mention that a query doesn't return any result

(in fact, the execution time was more than 5 hours, so we had to manually stop this query). Thus we can only compare 95 request. For the 1Go we have 1745 seconds (which is equals to 29 minutes). For the 2Go we have two results. One for the first request (which is quite long) and one other for the second request (which is faster).
The first one take 25893 seconds (equals 431.5534 minutes, which is equals to approximately six hours). The second request take 5743 seconds (equals to 95 minutes which correspond to one hours and half). So we see that there is a factor 0,30 of difference between 1Go and 2Go. It means that execute 2 Go take "only" 30% time more.

For all the results, we see that the template 72 take more time. If we check the query, we see that this query is composed of an association of four times a temporary table which is composed (itself) of three tables. This could explain the long process time. The difference between the first and the second test for the 2Go data (see chart 2) is probably due to two things: cache and analytic. Indeed, MariaDB use a cache system. Here we execute the two queries with only a few hours delay and no other queries to the database. We can imagine that some results are still in cache. Like other system, MariaDB is based on analytic. The database try to predict some result and optimized the order of the query on the basis of the data that it know (so its history).

8 Conclusion

TPC-DS Tools aim to compare the different database. In this report we use them to evaluate MariaDB's performances in handling data. First of all we see that all the queries aren't supported by MariaDB. Some of them could be fix but some other needs to be ignore.
Then we test the ability of MariaDB to manage data. We first test with one gigabyte. The result are quite positive. But when we increase the size of the data, the compute time increase more.

Thus we see that MariaDB is not made to manage a lot of data. But it could be interesting to see if PostgreSQL (or MySQL maybe) have the same results.

More generally, with this first part of the project, we were able to test a powerful benchmark tool and evaluate the performance of a database software to perform various complex functions important to Decision Support Systems.

References

[1] sjp38. tools for tpc-ds benchmark on mariadb, 2017. https://github.com/sjp38/tpcds-mariadb.

[2] gregrahn. Tpc-ds benchmark kit with some modifications/fixes, 2018. https://github.com/gregrahn/tpcds-kit.

[3] MariaDB. Mariadb, 2018. https://mariadb.org/.

[4] Meikel Pöss, Raghunath Othayoth Nambiar, and David Walrath. Why you should run tpc-ds: A workload analysis. In Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold, editors, *VLDB*, pages 1138–1149. ACM, 2007.

[5] TPC. The transaction processing performance council defines transaction processing and database benchmarks and delivers trusted results to the industry., 2001-2018. http://www.tpc.org/tpcds.

[6] Esteban Zimanyi. Cours: Info-h-419: Data warehouses, 2018. https://cs.ulb.ac.be/public/teaching/infoh419.

A Results for 1Go

Query	Template	Time (secs)	Note
1	1	0.024	
2	2	0.029	
3	3	0.023	
4	4	0.023	
5	5	0.024	
6	6	0.024	
7	7	0.030	
8	9	0.024	
9	10	0.025	
10	11	0.023	
11	12	0.026	
12	13	0.027	
13	15	0.023	
14	16	0.024	
15	17	0.024	
16	18	0.025	
17	19	0.024	
18	20	0.026	
19	21	0.024	
20	22	0.023	
21	23	0.024	
22		0.025	
23	24	0.025	
24		0.025	
25	25	0.023	
26	26	0.024	
27	27	0.025	
28	28	0.024	
29	29	0.027	
30	30	0.025	
31	31	0.025	
32	32	0.025	
33	33	0.026	
34	34	0.025	
35	35	0.024	
36	37	0.024	
37	39	0.025	
38		0.023	
39	40	0.024	
40	41	0.024	
41	42	0.024	
42	43	0.029	
43	44	0.025	
44	45	0.033	

45	46	0.042	
46	47	0.029	
47	48	0.025	
48	49	0.030	
49	50	0.023	
50	51	0.028	
51	52	0.023	
52	53	0.026	
53	54	0.031	
54	55	0.024	
55	56	0.026	
56	57	0.026	
57	58	0.031	
58	59	0.026	
59	60	0.026	
60	61	0.027	
61	62	0.031	
62	63	0.029	
63	64	0.027	
64	65	0.025	
65	66	48.738	
66	67	0.031	
67	68	0.024	
68	69	0.024	
69	71	0.027	
70	72	1686.772	
71	73	0.316	
72	74	0.254	
73	75	0.066	
74	76	0.249	
75	77	0.202	
76	78	0.156	
77	79	0.061	
78	80	0.057	
79	81	0.043	
80	82	0.039	
81	83	0.033	
82	84	5.304	
83	85	0.363	
84	88	0.087	
85	89	0.084	
86	90	0.052	
87	91	0.175	
88	92	0.040	

89	93	0.030	
90	94	0.031	
91	95	0.043	
92	96	0.023	
93	97	0.023	
94	98	0.061	
95	99	0.025	
96		0.023	

B Results for 2Go

B.1 First request

Query	Template	Time (secs)	Note
1	1	0.023	
2	2	496.649	
3	3	0.025	
4	4	761.118	
5	5	688.435	
6	6	206.613	
7	7	49.996	
8	9	36.762	
9	10	35.558	
10	11	425.260	
11	12	2.486	
12	13	61.702	
13	15	113.300	
14	16	152.764	
15	17	59.149	
16	18	72.128	
17	19	201.948	
18	20	36.559	
19	21	791.030	
20	22	3.412	
21	23	833.346	
22		966.455	
23	24	2.128	
24		1.335	
25	25	64.018	
26	26	61.876	
27	27	100.777	
28	28	24.634	
29	29	31.092	
30	30	9.293	
31	31	542.878	
32	32	1.502	
33	33	110.639	
34	34	49.037	
35	35	5728.619	
36	37	1.754	
37	39	1502.559	
38		1324.643	
39	40	14.857	
40	41	2.753	
41	42	4.253	
42	43	80.216	
43	44	88.351	
44	45	65.475	

45	46	75.930	
46	47	196.331	
47	48	59.956	
48	49	21.148	
49	50	2.089	
50	51	162.721	
51	52	0.281	
52	53	0.723	
53	54	223.171	
54	55	0.092	
55	56	5.946	
56	57	142.479	
57	58	152.931	
58	59	124.929	
59	60	51.668	
60	61	170.540	
61	62	61.905	
62	63	2.055	
63	64	66.171	
64	65	105.060	
65	66	151.927	
66	67	240.142	
67	68	77.350	
68	69	162.541	
69	71	146.609	
70	72	5658.055	
71	73	116.473	
72	74	343.850	
73	75	40.537	
74	76	77.241	
75	77	152.189	
76	78	550.900	
77	79	65.870	
78	80	19.237	
79	81	7.975	
80	82	0.898	
81	83	9.323	
82	84	1.801	
83	85	15.895	
84	88	313.104	
85	89	5.561	
86	90	25.906	
87	91	10.059	
88	92	0.591	

89	93	13.699	
90	94	11.561	
91	95	-	Time out
92	96	24.516	
93	97	114.717	
94	98	9.368	
95	99	91.498	
96		0.278	

B.2 Second request

Query	Template	Time (secs)	Note
1	1	0.026	
2	2	0.025	
3	3	0.021	
4	4	0.029	
5	5	0.023	
6	6	0.032	
7	7	0.024	
8	9	0.022	
9	10	0.024	
10	11	0.026	
11	12	0.023	
12	13	0.023	
13	15	0.039	
14	16	0.023	
15	17	0.022	
16	18	0.022	
17	19	0.024	
18	20	0.025	
19	21	0.023	
20	22	0.029	
21	23	0.022	
22		0.023	
23	24	0.023	
24		0.023	
25	25	0.024	
26	26	0.024	
27	27	0.023	
28	28	0.021	
29	29	0.021	
30	30	0.023	
31	31	0.025	
32	32	0.022	
33	33	0.022	
34	34	0.023	
35	35	0.022	
36	37	0.030	
37	39	0.040	
38		0.042	
39	40	0.036	
40	41	0.022	
41	42	0.026	
42	43	0.027	
43	44	0.029	
44	45	0.022	

45	46	0.024	
46	47	0.023	
47	48	0.023	
48	49	0.021	
49	50	0.032	
50	51	0.025	
51	52	0.024	
52	53	0.026	
53	54	0.024	
54	55	0.022	
55	56	0.023	
56	57	0.025	
57	58	0.026	
58	59	0.023	
59	60	0.023	
60	61	0.023	
61	62	0.023	
62	63	0.021	
63	64	0.022	
64	65	0.023	
65	66	91.527	
66	67	0.317	
67	68	0.146	
68	69	0.101	
69	71	0.036	
70	72	5642.131	
71	73	0.431	
72	74	0.191	
73	75	0.138	
74	76	0.044	
75	77	0.036	
76	78	0.051	
77	79	0.048	
78	80	0.051	
79	81	0.032	
80	82	0.035	
81	83	0.036	
82	84	5.131	
83	85	0.394	
84	88	0.213	
85	89	0.036	
86	90	0.077	
87	91	0.032	
88	92	0.022	

89	93	0.031	
90	94	0.034	
91	95	-	Time Out
92	96	0.047	
93	97	0.027	
94	98	0.053	
95	99	0.024	
96		0.021	

C Charts

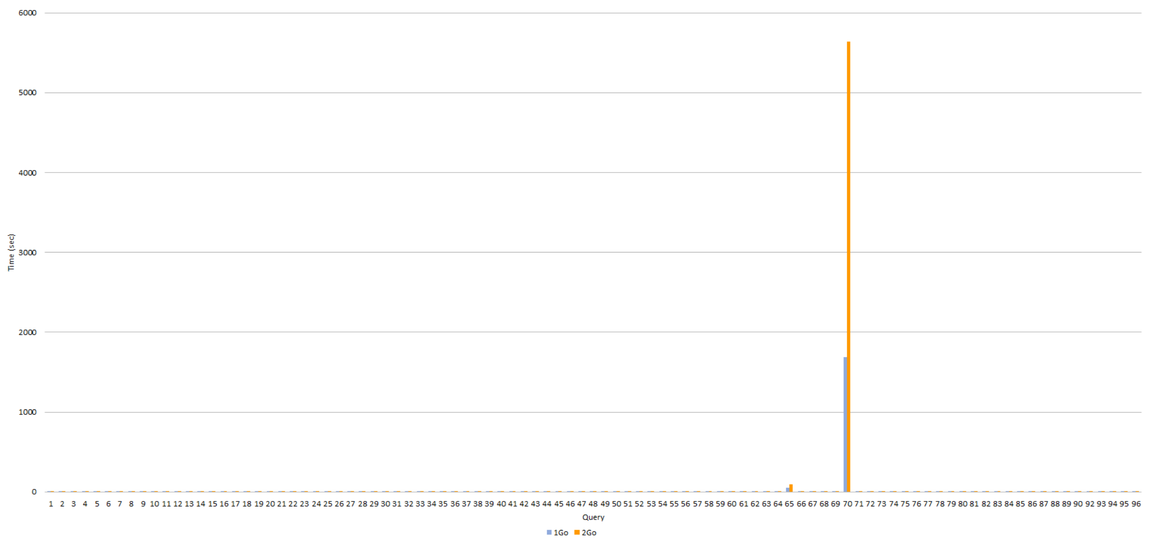

Figure 1: Results between query for 1Go and the second query for 2Go

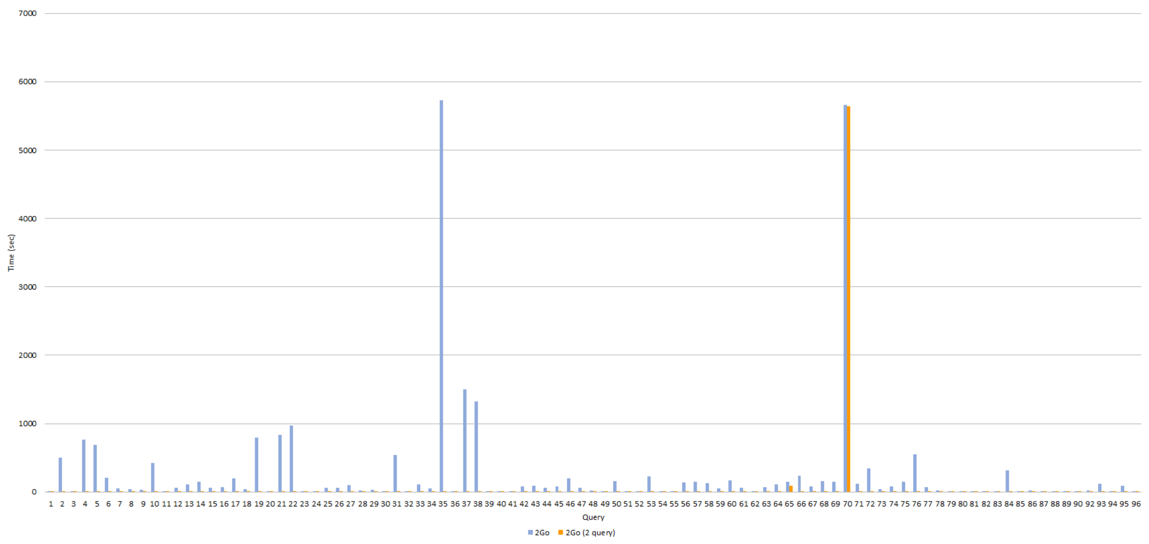

Figure 2: Results between first and second query for 2Go