HANDSON 2

Big Data Systems

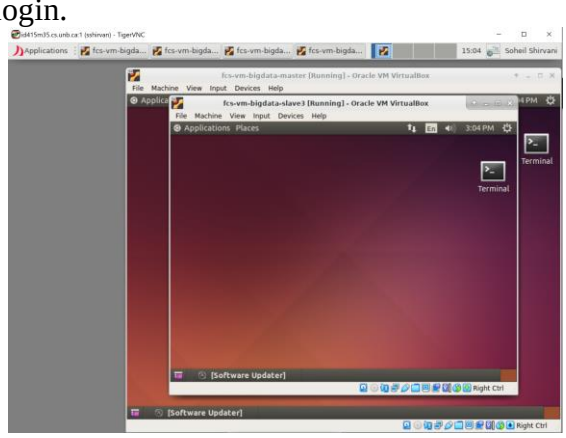
Soheil Shirvani - 3720505

Contents

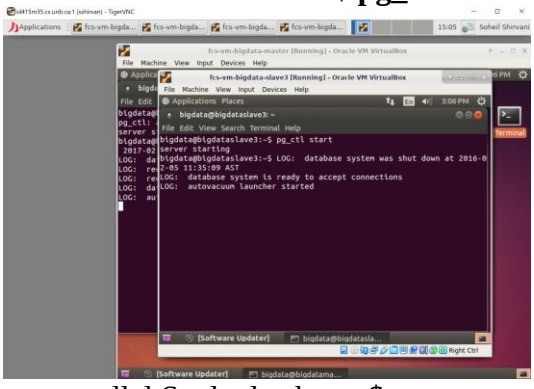
Lunching Stado Parallel database

Steps for lunching Stado Parallel:

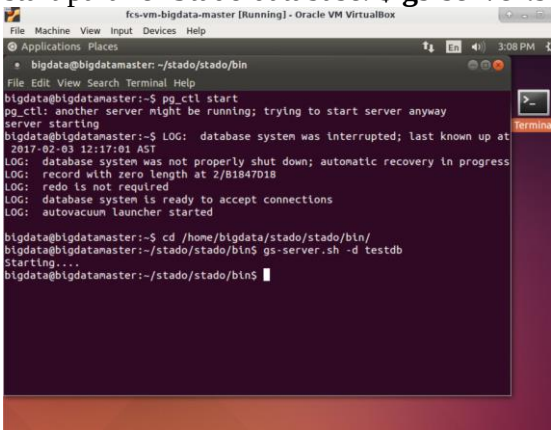
1. Here first we login into 4 different VMs. 1 is master VM and 3 Slave VMs. To do so we open all the VMs and login.



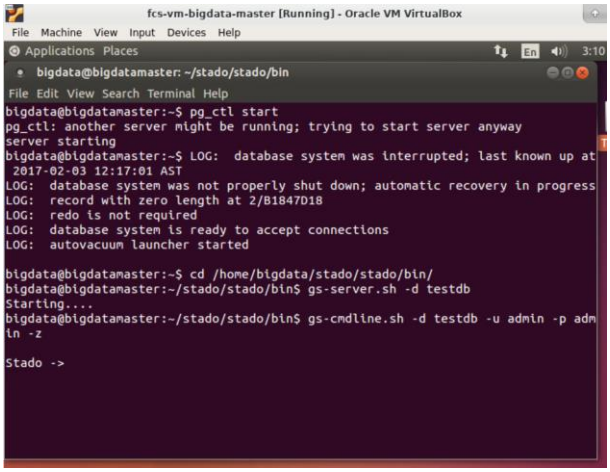
2. Start single PostgreSQL database on all the VMs. **$ pg_ctl start**



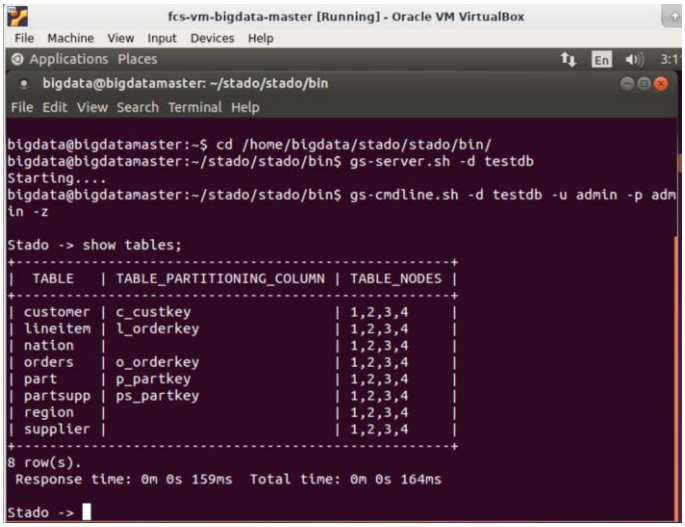
3. On master VM we will start parallel Stado database. **$ gs-server.sh -d testdb**



4. We will start Stado SQL client to connect to the parallel database. **$ gs-cmdline.sh -d testdb -u admin -p admin -z**



5. Then we are abale to see the tables in the database or create and insert in it.



Task 1

1.a Create a partitioned table partab

To Create a partitioned table named partab with 2 column including (col1, Integer) and (col2, Char) and partition key of (col1) we use the query below:

$$CREATE\ TABLE\ partab\ (col1\ INTEGER, col2\ CHAR(15))\ PARTITIONING\ KEY\ col1\ ON\ ALL;$$

And response is:

```
Stado -> CREATE TABLE partab (col1 INTEGER, col CHAR(15)) PARTITIONING KEY col1
ON ALL;
OK
 Response time: 0m 1s 275ms  Total time: 0m 1s 276ms
```

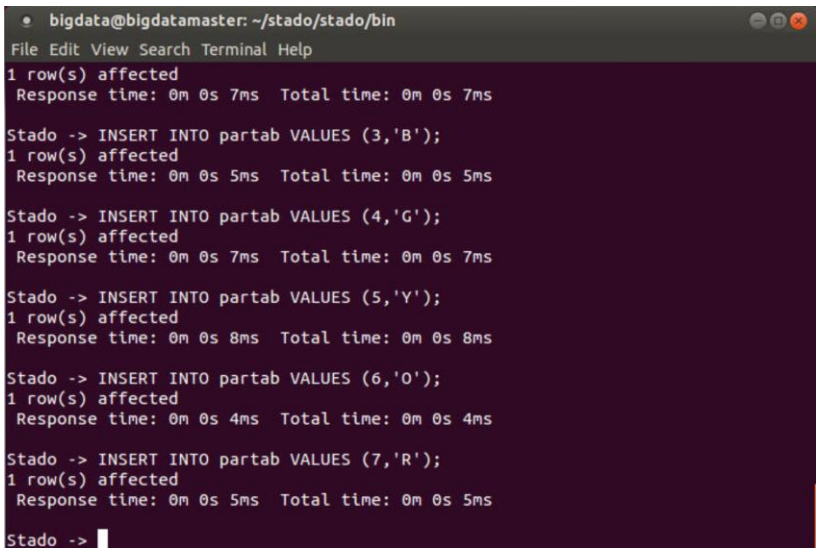
So the table is created. And Show tables here include the new table:

```
Stado -> show tables;
+-----------------------------------------------------+
|  TABLE   | TABLE_PARTITIONING_COLUMN | TABLE_NODES |
+-----------------------------------------------------+
| customer | c_custkey                 | 1,2,3,4     |
| lineitem | l_orderkey                | 1,2,3,4     |
| nation   |                           | 1,2,3,4     |
| orders   | o_orderkey                | 1,2,3,4     |
| part     | p_partkey                 | 1,2,3,4     |
| partab   | col1                      | 1,2,3,4     |
| partsupp | ps_partkey                | 1,2,3,4     |
| region   |                           | 1,2,3,4     |
| supplier |                           | 1,2,3,4     |
+-----------------------------------------------------+
9 row(s).
 Response time: 0m 0s 5ms  Total time: 0m 0s 7ms
```

1.b Insert the tuples (with given values) into partab

We will Insert 6 predefined tuples into our partab table that we created before. We use the command Below:

$$INSERT\ INTO\ partab\ VALUES\ (\text{'2'},\text{'I'}),\ \ldots$$



After Inserting all the tuples we can query our table and check if the tuples are in the table using the command below and see the result:

$$SELECT\ *\ FROM\ partab$$

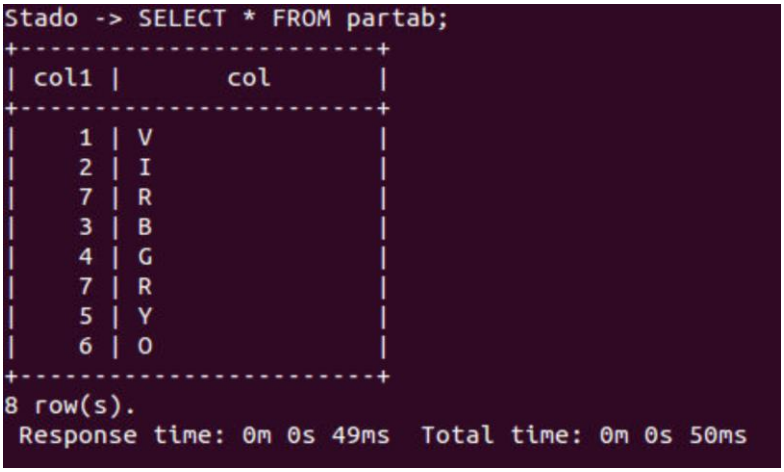


Table Actually have all the rows (7,'R') is not displayed and it was inserted twice. So all the insertion worked perfectly fine.

2.a Create a replicated table reptab.

To Create a replicated table named reptab with 2 column including (fld1, Integer) and (fld2, Char(2)) we use the query below:

$$CREATE\ TABLE\ reptab\ (fld1\ INTEGER, fld2\ CHAR(2))\ REPLICATED;$$

And response is:

```
Stado -> CREATE TABLE reptab (fld1 INTEGER, fld2 CHAR(2)) REPLICATED;
OK
 Response time: 0m 0s 106ms  Total time: 0m 0s 106ms
```

So the table is created. And Show tables here include the new table:

```
Stado -> show tables;
+-----------------------------------------------------+
|  TABLE   | TABLE_PARTITIONING_COLUMN | TABLE_NODES |
+-----------------------------------------------------+
| customer | c_custkey                 | 1,2,3,4     |
| lineitem | l_orderkey                | 1,2,3,4     |
| nation   |                           | 1,2,3,4     |
| orders   | o_orderkey                | 1,2,3,4     |
| part     | p_partkey                 | 1,2,3,4     |
| partab   | col1                      | 1,2,3,4     |
| partsupp | ps_partkey                | 1,2,3,4     |
| region   |                           | 1,2,3,4     |
| reptab   |                           | 1,2,3,4     |
| supplier |                           | 1,2,3,4     |
+-----------------------------------------------------+
10 row(s).
 Response time: 0m 0s 16ms  Total time: 0m 0s 17ms
```

2.b Insert the tuples (with given values) into reptab

We will Insert 6 predefined tuples into our reptab table that we created before. We use the command Below:

$$INSERT\ INTO\ reptab\ VALUES\ (\text{'2'},\text{'I'}),\ \ldots$$

```
Stado -> INSERT INTO reptab VALUES (7,'NB');
1 row(s) affected
 Response time: 0m 0s 18ms  Total time: 0m 0s 19ms

Stado -> INSERT INTO reptab VALUES (8,'NL');
1 row(s) affected
 Response time: 0m 0s 12ms  Total time: 0m 0s 12ms

Stado -> INSERT INTO reptab VALUES (9,'NS');
1 row(s) affected
 Response time: 0m 0s 13ms  Total time: 0m 0s 15ms

Stado -> INSERT INTO reptab VALUES (10, 'PE');
1 row(s) affected
 Response time: 0m 0s 11ms  Total time: 0m 0s 11ms

Stado ->
```

After Inserting all the tuples we can query our table and check if the tuples are in the table using the command below and see the result:

$$SELECT\ *\ FROM\ reptab$$

```
Stado -> SELECT * FROM reptab;
+-------------+
| fld1 | fld2 |
+-------------+
|    7 | NB   |
|    8 | NL   |
|    9 | NS   |
|   10 | PE   |
+-------------+
4 row(s).
 Response time: 0m 0s 11ms  Total time: 0m 0s 12ms

Stado ->
```

As we can see all the tuples were inserted into the reptab successfully.

Results Only:

1. Partitioned

$$CREATE\ TABLE\ partab\ (col1\ INTEGER, col2\ CHAR)\ PARTITIONING\ KEY\ col1\ ON\ ALL;$$

Schema of your table partab (output of: show table partab)

```
+-------------------------------------------------------------------------+
| COLUMN_NAME | SQL_DATA_TYPE | TYPE_NAME | IS_NULLABLE | KEY | DEFAULT |
+-------------------------------------------------------------------------+
| col1        |             4 | INTEGER   | YES         | NO  |         |
| col         |             1 | CHAR(15)  | YES         | NO  |         |
+-------------------------------------------------------------------------+
2 row(s).
 Response time: 0m 0s 5ms  Total time: 0m 0s 5ms
```

Output of the query: SELECT * FROM partab

```
Stado -> SELECT * FROM partab;
+----------------------+
| col1 |       col     |
+----------------------+
|    1 | V             |
|    2 | I             |
|    7 | R             |
|    3 | B             |
|    4 | G             |
|    7 | R             |
|    5 | Y             |
|    6 | O             |
+----------------------+
8 row(s).
 Response time: 0m 0s 49ms  Total time: 0m 0s 50ms
```

2. Replicated

Schema of your table reptab (output of: show table reptab)

```
Stado -> show table reptab
Stado -> ;
+-------------------------------------------------------------------------+
| COLUMN_NAME | SQL_DATA_TYPE | TYPE_NAME | IS_NULLABLE | KEY | DEFAULT |
+-------------------------------------------------------------------------+
| fld1        |             4 | INTEGER   | YES         | NO  |         |
| fld2        |             1 | CHAR(2)   | YES         | NO  |         |
+-------------------------------------------------------------------------+
2 row(s).
 Response time: 0m 0s 3ms  Total time: 0m 0s 4ms
```

Output of the query: SELECT * FROM reptab

```
Stado -> SELECT * FROM reptab;
+-------------+
| fld1 | fld2 |
+-------------+
|    7 | NB   |
|    8 | NL   |
|    9 | NS   |
|   10 | PE   |
+-------------+
4 row(s).
 Response time: 0m 0s 11ms  Total time: 0m 0s 12ms

Stado ->
```

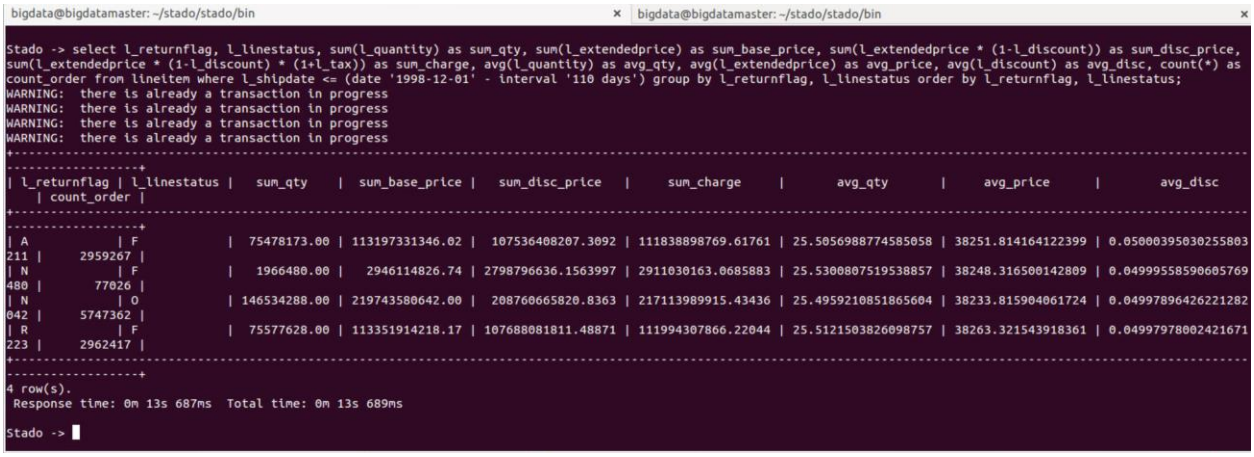
Task 2

Here we are going to run 3 queries from TPC-H benchmark and find the response time of database in 2 modes Parallel and Single. I ignored the first run of each query (cold) and made a note of the response time of 2 next query execution for both modes.

Run 3 query on Stado as parallel database.

Query 1:

After running 1 time (cold run) the Second result is: **Time: 13s 689ms**

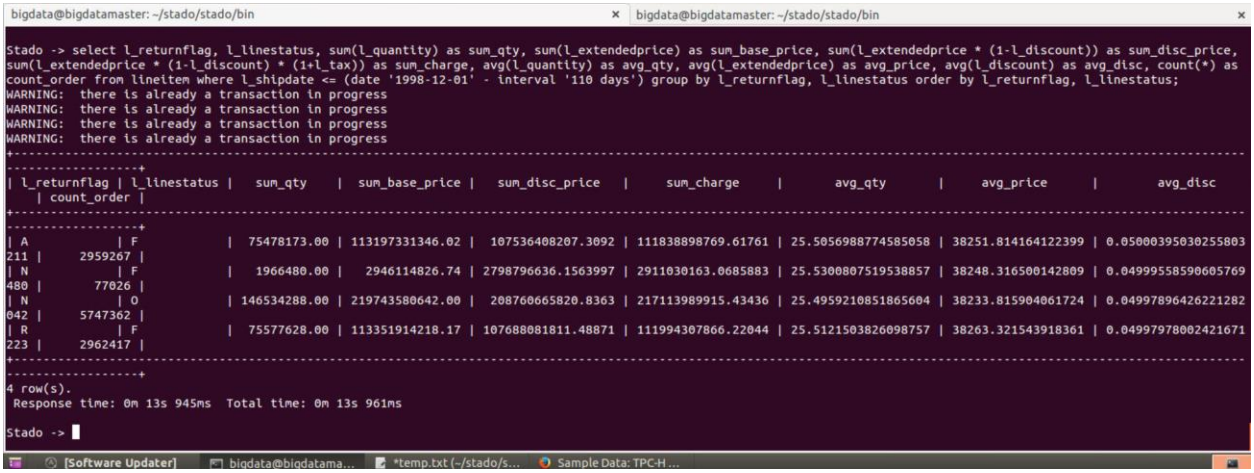


```
bigdata@bigdatamaster: ~/stado/stado/bin
bigdata@bigdatamaster: ~/stado/stado/bin

Stado -> select l_returnflag, l_linestatus, sum(l_quantity) as sum_qty, sum(l_extendedprice) as sum_base_price, sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge, avg(l_quantity) as avg_qty, avg(l_extendedprice) as avg_price, avg(l_discount) as avg_disc, count(*) as
count_order from lineitem where l_shipdate <= (date '1998-12-01' - interval '110 days') group by l_returnflag, l_linestatus order by l_returnflag, l_linestatus;
WARNING:  there is already a transaction in progress
WARNING:  there is already a transaction in progress
WARNING:  there is already a transaction in progress
WARNING:  there is already a transaction in progress
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------+
| l_returnflag | l_linestatus |   sum_qty    |  sum_base_price |   sum_disc_price   |     sum_charge     |       avg_qty       |     avg_price      |        avg_disc
    | count_order |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------+
| A            | F            |  75478173.00 | 113197331346.02 |  107536408207.3092 | 111838898769.61761 | 25.5056988774585058 | 38251.814164122399 | 0.05000395030255803
211 |     2959267 |
| N            | F            |   1966480.00 |   2946114826.74 | 2798796636.1563997 | 2911030163.0685883 | 25.5300807519538857 | 38248.316500142809 | 0.04999558590605769
480 |       77026 |
| N            | O            | 146534288.00 | 219743580642.00 |  208760665820.8363 | 217113989915.43436 | 25.4959210851865604 | 38233.815904061724 | 0.04997896426221282
042 |     5747362 |
| R            | F            |  75577628.00 | 113351914218.17 | 107688081811.48871 | 111994307866.22044 | 25.5121503826098757 | 38263.321543918361 | 0.04997978002421671
223 |     2962417 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------+
4 row(s).
 Response time: 0m 13s 687ms  Total time: 0m 13s 689ms

Stado ->
```

Third run is: **Time: 13s 961ms**



```
bigdata@bigdatamaster: ~/stado/stado/bin
bigdata@bigdatamaster: ~/stado/stado/bin

Stado -> select l_returnflag, l_linestatus, sum(l_quantity) as sum_qty, sum(l_extendedprice) as sum_base_price, sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge, avg(l_quantity) as avg_qty, avg(l_extendedprice) as avg_price, avg(l_discount) as avg_disc, count(*) as
count_order from lineitem where l_shipdate <= (date '1998-12-01' - interval '110 days') group by l_returnflag, l_linestatus order by l_returnflag, l_linestatus;
WARNING:  there is already a transaction in progress
WARNING:  there is already a transaction in progress
WARNING:  there is already a transaction in progress
WARNING:  there is already a transaction in progress
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------+
| l_returnflag | l_linestatus |   sum_qty    |  sum_base_price |   sum_disc_price   |     sum_charge     |       avg_qty       |     avg_price      |        avg_disc
    | count_order |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------+
| A            | F            |  75478173.00 | 113197331346.02 |  107536408207.3092 | 111838898769.61761 | 25.5056988774585058 | 38251.814164122399 | 0.05000395030255803
211 |     2959267 |
| N            | F            |   1966480.00 |   2946114826.74 | 2798796636.1563997 | 2911030163.0685883 | 25.5300807519538857 | 38248.316500142809 | 0.04999558590605769
480 |       77026 |
| N            | O            | 146534288.00 | 219743580642.00 |  208760665820.8363 | 217113989915.43436 | 25.4959210851865604 | 38233.815904061724 | 0.04997896426221282
042 |     5747362 |
| R            | F            |  75577628.00 | 113351914218.17 | 107688081811.48871 | 111994307866.22044 | 25.5121503826098757 | 38263.321543918361 | 0.04997978002421671
223 |     2962417 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------+
4 row(s).
 Response time: 0m 13s 945ms  Total time: 0m 13s 961ms

Stado ->
[Software Updater]  bigdata@bigdatama...  *temp.txt (~/stado/s...  Sample Data: TPC-H ...
```

Query 3:

After running 1 time (cold run) the Second result is: **Time: 10s 451ms**

```
tado -> SELECT L_ORDERKEY, SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS REVENUE, O_ORDERDATE, O_SHIPPRIORITY FROM CUSTOMER, ORDERS, LINEITEM WHERE C_MKTSEGMENT IN ('AUTOMOBIL
E') AND C_CUSTKEY = O_CUSTKEY AND L_ORDERKEY = O_ORDERKEY AND O_ORDERDATE < date '1995-03-19' AND L_SHIPDATE > date '1995-03-19' GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIP
PRIORITY ORDER BY REVENUE DESC, O_ORDERDATE;
|    3589860 |   1095.4752 | 1994-12-11  |              0 |
|    4302085 |   1091.9898 | 1995-02-22  |              0 |
|    3452033 |   1076.0937 | 1994-12-01  |              0 |
|    7557798 |   1066.9282 | 1995-01-28  |              0 |
|    9988964 |   1066.7136 | 1995-01-01  |              0 |
|     442437 |   1059.9072 | 1994-12-01  |              0 |
|    6456420 |   1059.9072 | 1995-02-19  |              0 |
|    8080036 |   1050.6652 | 1995-03-12  |              0 |
|    3445601 |   1044.6432 | 1995-01-06  |              0 |
|    6834311 |   1037.9440 | 1995-03-06  |              0 |
|   10804000 |   1029.6891 | 1995-01-20  |              0 |
|    5012578 |    989.7888 | 1994-12-18  |              0 |
|    7674112 |    977.7034 | 1995-02-02  |              0 |
|    5613284 |    963.3096 | 1994-12-30  |              0 |
|    4182179 |    924.4992 | 1994-11-25  |              0 |
|    7670117 |    915.8496 | 1995-02-27  |              0 |
|    2204743 |    906.7872 | 1995-03-15  |              0 |
|    7231653 |    900.9637 | 1994-12-21  |              0 |
|    9097634 |    883.9502 | 1995-01-14  |              0 |
|    9418727 |    873.9905 | 1994-12-01  |              0 |
+------------+-------------+-------------+----------------+
22451 row(s).
 Response time: 0m 7s 734ms  Total time: 0m 10s 451ms
```

Third run is: **Time: 10s 459ms**

```
|    9988964 |   1066.7136 | 1995-01-01  |              0 |
|     442437 |   1059.9072 | 1994-12-01  |              0 |
|    6456420 |   1059.9072 | 1995-02-19  |              0 |
|    8080036 |   1050.6652 | 1995-03-12  |              0 |
|    3445601 |   1044.6432 | 1995-01-06  |              0 |
|    6834311 |   1037.9440 | 1995-03-06  |              0 |
|   10804000 |   1029.6891 | 1995-01-20  |              0 |
|    5012578 |    989.7888 | 1994-12-18  |              0 |
|    7674112 |    977.7034 | 1995-02-02  |              0 |
|    5613284 |    963.3096 | 1994-12-30  |              0 |
|    4182179 |    924.4992 | 1994-11-25  |              0 |
|    7670117 |    915.8496 | 1995-02-27  |              0 |
|    2204743 |    906.7872 | 1995-03-15  |              0 |
|    7231653 |    900.9637 | 1994-12-21  |              0 |
|    9097634 |    883.9502 | 1995-01-14  |              0 |
|    9418727 |    873.9905 | 1994-12-01  |              0 |
+------------+-------------+-------------+----------------+
22451 row(s).
 Response time: 0m 7s 815ms  Total time: 0m 10s 459ms
```

Query 6:

After running 1 time (cold run) the Second result is: **Time: 2s 219ms**

```
Stado -> SELECT SUM(L_EXTENDEDPRICE*L_DISCOUNT) AS REVENUE FROM LINEITEM WHERE L_SHIPDATE >= date '1995-01-01' AND L_SHIPDATE < (date '1995-01-01' + interval '1 year') A
ND L_DISCOUNT BETWEEN .04 - 0.01 AND .04 + 0.01 AND L_QUANTITY < 24;
+----------------+
|    revenue     |
+----------------+
| 164347732.3953 |
+----------------+
1 row(s).
 Response time: 0m 2s 219ms  Total time: 0m 2s 219ms
```

Third run is: **Time: 1s 909ms**

```
+----------------+
|    revenue     |
+----------------+
| 164347732.3953 |
+----------------+
1 row(s).
 Response time: 0m 1s 909ms  Total time: 0m 1s 909ms

Stado ->
```

Run 3 query on PostgreSQL as Single database.

First we stop the Stado and go to PostgreSQL command line to do the queries.

```
Stado -> ^C
bigdata@bigdatamaster:~/stado/stado/bin$ cd
bigdata@bigdatamaster:~$ psql testdb
psql (9.4.0)
Type "help" for help.

testdb=#
```

Query 1:

After running 1 time (cold run) the Second result is: **Time: 40s 248ms**

```
testdb=# select l_returnflag, l_linestatus, sum(l_quantity) as sum_qty, sum(l_extendedprice) as sum_base_price, sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge, avg(l_quantity) as avg_qty, avg(l_extendedprice) as avg_price, avg(l_discount) as avg_disc, count(*) as
count_order from lineitem where l_shipdate <= (date '1998-12-01' - interval '110 days') group by l_returnflag, l_linestatus order by l_returnflag, l_linestatus;
```

```
 l_returnflag | l_linestatus |   sum_qty    | sum_base_price |  sum_disc_price   |     sum_charge      |        avg_qty        |      avg_price       |         avg_disc         |
   | count_order
--------------+--------------+--------------+----------------+-------------------+---------------------+-----------------------+----------------------+--------------------------+
---+-------------
 A            | F            |  75478173.00 | 113197331346.02 | 107536408207.3092 | 111838898769.617614 | 25.5056988774585058 | 38251.814164122399 | 0.050003950302558032
11 |     2959267
 N            | F            |   1966480.00 |   2946114826.74 |   2798796636.1564 |   2911030163.068588 | 25.5300807519538857 | 38248.316500142809 | 0.049995585906057694
80 |       77026
 N            | O            | 146534288.00 | 219743580642.00 | 208760665820.8363 | 217113989915.434356 | 25.4959210851865604 | 38233.815904061724 | 0.049978964262212820
42 |     5747362
 R            | F            |  75577628.00 | 113351914218.17 | 107688081811.4887 | 111994307866.220439 | 25.5121503826098757 | 38263.321543918361 | 0.049979780024216712
23 |     2962417
(4 rows)

Time: 40248.856 ms
```

Third run is: **Time: 40s 788ms**

```
 l_returnflag | l_linestatus |   sum_qty    | sum_base_price |  sum_disc_price   |     sum_charge      |        avg_qty        |      avg_price       |         avg_disc         |
   | count_order
--------------+--------------+--------------+----------------+-------------------+---------------------+-----------------------+----------------------+--------------------------+
---+-------------
 A            | F            |  75478173.00 | 113197331346.02 | 107536408207.3092 | 111838898769.617614 | 25.5056988774585058 | 38251.814164122399 | 0.050003950302558032
11 |     2959267
 N            | F            |   1966480.00 |   2946114826.74 |   2798796636.1564 |   2911030163.068588 | 25.5300807519538857 | 38248.316500142809 | 0.049995585906057694
80 |       77026
 N            | O            | 146534288.00 | 219743580642.00 | 208760665820.8363 | 217113989915.434356 | 25.4959210851865604 | 38233.815904061724 | 0.049978964262212820
42 |     5747362
 R            | F            |  75577628.00 | 113351914218.17 | 107688081811.4887 | 111994307866.220439 | 25.5121503826098757 | 38263.321543918361 | 0.049979780024216712
23 |     2962417
(4 rows)

Time: 40788.093 ms
```

Query 3:

After running 1 time (cold run) the Second result is: **Time: 16s 948ms**

```
testdb=# SELECT L_ORDERKEY, SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS REVENUE, O_ORDERDATE, O_SHIPPRIORITY FROM CUSTOMER, ORDERS, LINEITEM WHERE C_MKTSEGMENT IN ('AUTOMOBIL
E') AND C_CUSTKEY = O_CUSTKEY AND L_ORDERKEY = O_ORDERKEY AND O_ORDERDATE < date '1995-03-19' AND L_SHIPDATE > date '1995-03-19' GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIP
PRIORITY ORDER BY REVENUE DESC, O_ORDERDATE;
```

```
   1824643 | 380013.3687 | 1995-03-06 |              0
   1088995 | 375667.8475 | 1995-03-11 |              0
   6669831 | 375492.0278 | 1995-02-07 |              0
   4710272 | 375188.6517 | 1995-03-17 |              0
   3778628 | 373654.2087 | 1995-02-25 |              0
   4736678 | 372807.1864 | 1995-03-03 |              0
   1062786 | 372672.5509 | 1995-02-10 |              0
   6680197 | 369266.7219 | 1995-02-13 |              0
  10350439 | 369187.7220 | 1995-02-16 |              0
   8257025 | 368258.2766 | 1995-03-16 |              0
   4496835 | 367877.7452 | 1995-03-18 |              0
   7205063 | 366929.7907 | 1995-02-26 |              0
   1102534 | 366089.2778 | 1995-03-15 |              0
   7166048 | 365379.9582 | 1995-03-02 |              0
Time: 16948.201 ms
```

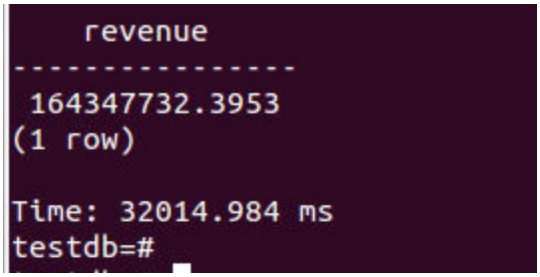
Third run is: **Time: 16s 187ms**

```
   4710272 | 375188.6517 | 1995-03-17 |              0
   3778628 | 373654.2087 | 1995-02-25 |              0
   4736678 | 372807.1864 | 1995-03-03 |              0
   1062786 | 372672.5509 | 1995-02-10 |              0
   6680197 | 369266.7219 | 1995-02-13 |              0
  10350439 | 369187.7220 | 1995-02-16 |              0
   8257025 | 368258.2766 | 1995-03-16 |              0
   4496835 | 367877.7452 | 1995-03-18 |              0
   7205063 | 366929.7907 | 1995-02-26 |              0
   1102534 | 366089.2778 | 1995-03-15 |              0
   7166048 | 365379.9582 | 1995-03-02 |              0
Time: 16187.535 ms
```

Query 6:

After running 1 time (cold run) the Second result is: **Time: 32s 14ms**

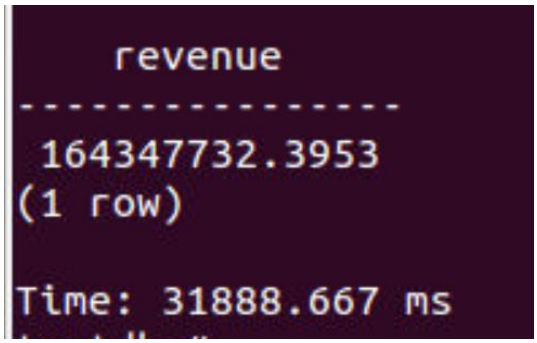
```
testdb=# SELECT SUM(L_EXTENDEDPRICE*L_DISCOUNT) AS REVENUE FROM LINEITEM WHERE L_SHIPDATE >= date '1995-01-01' AND L_SHIPDATE < (date '1995-01-01' + interval '1 year') A
ND L_DISCOUNT BETWEEN .04 - 0.01 AND .04 + 0.01 AND L_QUANTITY < 24;
```



```
    revenue
---------------
 164347732.3953
(1 row)

Time: 32014.984 ms
testdb=#
```

Third run is: **Time: 31s 888ms**



```
    revenue
---------------
 164347732.3953
(1 row)

Time: 31888.667 ms
```

Results Only

Average execution time of the warm runs of Q1, Q3 and Q6 with Stado

$$Q1 \begin{cases} warn\ run\ 1: \ 13\text{s}\ 689\text{ms} : 13689ms \\ warn\ run\ 2: \ 13\text{s}\ 961\text{ms} : 13961\text{ms} \end{cases} \rightarrow Avg = 13825ms$$

$$Q3 \begin{cases} warn\ run\ 1: \ 10\text{s}\ 451\text{ms} : 10451ms \\ warn\ run\ 2: \ 10\text{s}\ 459\text{ms} : 10459\text{ms} \end{cases} \rightarrow Avg = 10455ms$$

$$Q6 \begin{cases} warn\ run\ 1: \ 2\text{s}\ 219\text{ms} : 2219ms \\ warn\ run\ 2: \ 1\text{s}\ 909\text{ms} : 1909\text{ms} \end{cases} \rightarrow Avg = 2064ms$$

Average execution time of the warm runs of Q1, Q3 and Q6 with Single DB

$$Q1 \begin{cases} warn\ run\ 1: \ 40\text{s}\ 248\text{ms} : 40248ms \\ warn\ run\ 2: \ 40\text{s}\ 788\text{ms} : 40788\text{ms} \end{cases} \rightarrow Avg = 40518ms$$

$$Q3 \begin{cases} warn\ run\ 1: \ 16\text{s}\ 948\text{ms} : 16948ms \\ warn\ run\ 2: \ 16\text{s}\ 187\text{ms} : 16187\text{ms} \end{cases} \rightarrow Avg = 16568ms$$

$$Q6 \begin{cases} warn\ run\ 1: \ 32\text{s}\ 14\text{ms} : 32014ms \\ warn\ run\ 2: \ 31\text{s}\ 888\text{ms} : 31888\text{ms} \end{cases} \rightarrow Avg = 31951ms$$

Speedup of Q1, Q3 and Q6 achieved with Stado against single instance PostgreSQL

$$Q1\ Speedup = \frac{40518}{13825} = 2.93$$

$$Q3\ Speedup = \frac{16568}{10455} = 1.584$$

$$Q6\ Speedup = \frac{31951}{2064} = 15.48$$