Contents

1. Abstract

The objective of this assignment is creation of Data lakes on HDFS.User should be able to manage i.e upload,download,find a file,search in a data lakes.

2. Dataset Information

We have tested on two sizes of datasets 1GB and 10GB. TPC-H benchmark has provided a script to download the datasets by varying the Scale Factor(SF) . All the tables in TPC-H scale from a given size at Scale Factor 1 (SF = 1) to 10 times as large at SF = 10 (i.e. Tables have cardinalities that are multiples of SF).

3. TPC-H Schema diagram

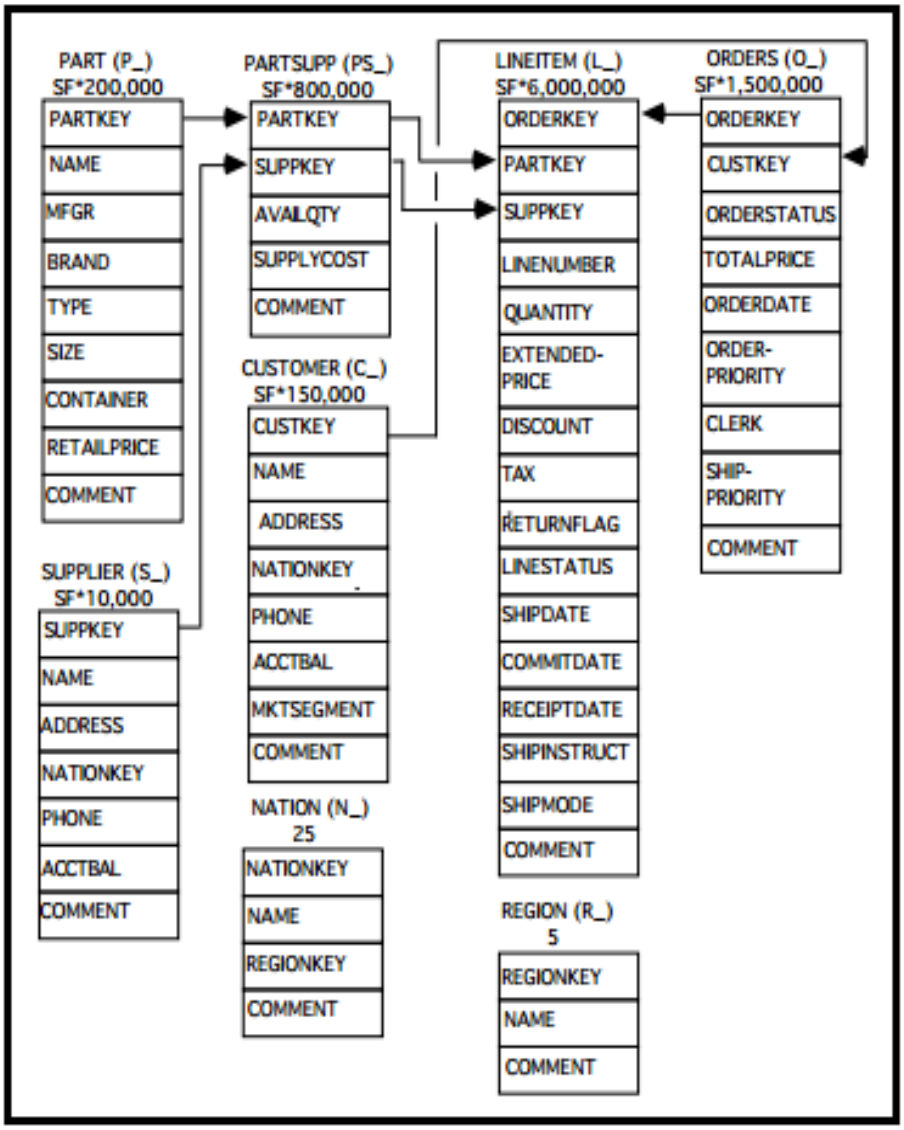


Figure 1 TPC-H benchmarking Schema

4. TPC-H database Creation and Loading

4.1. Queries on single node(SN) & 2-node hadoop(MN) cluster

Sr. No.	Query	Time taken SN (1 GB)	Time taken SN (10 GB)	Time taken MN (1 GB)	Time taken MN (10 GB)
1	create table nation(nationkey int,name varchar(64),regionkey int,comment varchar(64))row format delimited fields terminated by '|' stored as textfile;	0.556	1.248	0.402	0.347
2	load data local inpath '/home/akanshahduser/Pictures/10gb/nation.tbl' into table nation;	0.983	2.298	1.106	0.543
3	create table region(regionkey int,name varchar(64),comment varchar(64))row format delimited fields terminated by '|' stored as textfile;	0.062	0.058	0.063	0.056
4	load data local inpath '/home/akanshahduser/Pictures/10gb/region.tbl' into table region;	0.186	2.119	0.233	0.164
5	create table supplier(suppkey int,name varchar(64),address varchar(64),nationkey int,phone varchar(18),acctbal decimal(13,2),comment varchar(105)) row format delimited fields terminated by '|' stored as textfile;	0.09	0.059	0.096	0.06
6	load data local inpath '/home/akanshahduser/Pictures/10gb/supplier.tbl' into table supplier;	0.222	4.156	0.211	0.384
7	create table partsupp(partkey int,suppkey int,availqty int,supplycost decimal(13,2),comment varchar(64))row format delimited fields terminated by '|' stored as textfile;	0.261	1.04	0.081	0.058
8	load data local inpath '/home/akanshahduser/Pictures/10gb/partsupp.tbl' into table partsupp;	0.466	94.643	1.24	17.595
9	create table part(partkey int,name varchar(64),mfgr varchar(64),brand varchar(64),type varchar(64),size int,container varchar(64),retailprice decimal(13,2),comment varchar(64))row format delimited fields terminated by '|' stored as textfile;	0.06	1.378	0.08	0.051
10	load data local inpath '/home/akanshahduser/Pictures/10gb/part.tbl' into table part;	0.244	2.094	0.393	8.607
11	create table orders(orderkey int,custkey int,orderstatus varchar(64),totalprice decimal(13,2),orderdate date,orderpriority varchar(64),clerk	0.066	0.127	0.121	0.108

	varchar(64),shippriority int,comment varchar(64))row format delimited fields terminated by '|' stored as textfile;				
12	load data local inpath '/home/akanshahduser/Pictures/10gb/orders.tbl' into table orders;	1.556	27.698	1.735	27.256
13	create table lineitem(orderkey int,partkey int,suppkey int,linenumber int,quantity int,extendedprice decimal(13,2),discount decimal(13,2),tax decimal(13,2),returnflag varchar(64),linestatus varchar(64),shipdate date,commitdate date,receiptdate date,shipinstruct varchar(64),shipmode varchar(64),comment varchar(64))row format delimited fields terminated by '|' stored as textfile;	0.076	0.108	0.108	0.058
14	load data local inpath '/home/akanshahduser/Pictures/10gb/lineitem.tbl' into table lineitem;	12.198	139.09	8.132	148.513
15	create table customer(custkey int,name varchar(64),address varchar(64),nationkey int,phone varchar(64),acctbal decimal(13,2),mktsegment varchar(64),comment varchar(64))row format delimited fields terminated by '|' stored as textfile;	0.069	0.114	0.066	0.039
16	load data local inpath '/home/akanshahduser/Pictures/10gb/customer.tbl' into table customer;	0.458	2.158	0.377	2.927

5. Conversion of Snowflake to Star Schema

Here are some of the schema changes that we made to change the Normalized TPC-H schema (Figure 1) to the efficient star schema form (Figure 2) as suggested in reference [1].

1. We combine the TPC-H LINEITEM and ORDERS tables into one sales fact table that we name LINEORDER.
2. 2. We drop the PARTSUPP table since it would belong to a different data mart than the ORDERS and LINEITEM information. This is because PARTSUPP has different temporal granularity.
3. We drop the comment attribute of a LINEITEM (27 chars), the comment for an order (49 chars), and the shipping instructions for a LINEITEM (25 chars), because a warehouse does not store such information in a fact table.
4. We add the DATE dimension table, as is standard for a warehouse on sales.

5.2. TPC –H star schema diagram

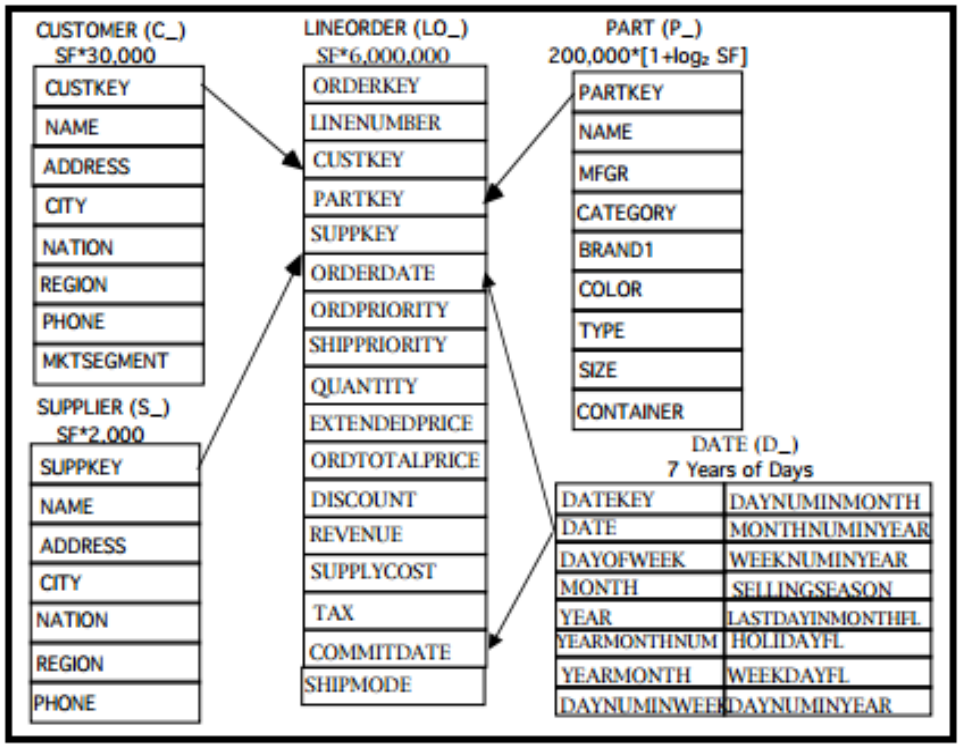


Figure 2 Star Schema

5.3. Transformation Queries

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
1	create table lineorder as select l.orderkey as orderkey,l.linenumber as linenumber,o.custkey as custkey,l.partkey as partkey,l.suppkey as suppkey, o.orderdate as orderdate,o.orderpriority as orderpriority,o.shippriority as shippriority,l.quantity as quantity,l.extendedprice as extendedprice,o.totalprice as ordertotalprice,l.discount as discount,l.tax as revenue,l.extendedprice as supplycost,l.tax as tax,l.commitdate as commitdate,l.shipmode as shipmode from lineitem l join orders o on (l.orderkey=o.orderkey);	96.609	948.74	47.95	261.40
2	create table N_R as select n.nationkey as nationkey,r.regionkey as regionkey,n.name as nation,r.name as city from nation n join region r on (n.regionkey==r.regionkey);	5.021	8.855	14.58	14.258
3	create table suppl_n as select * from supplier	1.386	1.485	11.06	11.091
4	drop table supplier;	0.127	3.917	0.734	0.492
5	create table supplier as select s.suppkey as suppkey,s.name as name,s.address as address,n.city as city,n.nation as nation,s.phone as phone from suppl_n s join N_R n on (s.nationkey == n.nationkey);	4.569	4.626	13.14	12.887
6	create table cust2 as select * from customer;	3.323	7.399	11.0	15.049
7	drop table customer;	0.664	0.207	0.171	0.126
8	create table customer as select c.custkey as custkey,c.name as name,c.address as address,n.city as city,n.nation as nation,c.phone as phone,c.mktsegment as mktsegment from cust2 c join N_R n on (c.nationkey==n.nationkey);	4.75	11.533	14.07	18.382
9	create table part_b as select * from part;	2.227	9.448	11.063	18.24
10	drop table part;	0.1	0.09	0.094	0.076
11	create table part as select p.partkey as partkey,p.name as name,p.mfgr as mfgr,p.brand as brand,p.type as type, p.size as size,p.container as container from part_b p;	1.388	7.448	11.23	15.138
12	drop table lineitem;	0.105	0.125	0.576	0.094
13	drop table part_b;	0.097	0.116	0.117	0.081
14	drop table cust2; drop table region; drop table nation; drop table orders;	0.097 0.036 0.091 0.088	0.139 0.086 0.125 0.122	0.116 0.113 0.082 0.164	0.083 0.083 0.083 0.084
15	create table datee as select distinct orderdate from lineorder union select distinct commitdate from lineorder;	18.845	142.15	47.63	117.47

16	create table ddate(datekey varchar(10),t_date varchar(18),month int,year int,todays_date int,yearmonth varchar(8),Quarters int,sellingseason varchar(12),weekofyear int,holiday varchar(1),weekday varchar(1));	0.079	0.097	0.084	0.055
17	create table ddate1 as select distinct orderdate from lineorder union select distinct commitdate from lineorder;	15.919	140.285	48.29	119.46
18	insert into ddate (datekey,t_date,month,year,todays_date, yearmonth, weekofyear) select orderdate,orderdate, month(orderdate),year (orderdate),day(orderdate), concat(year(orderdate), month(orderdate)), weekofyear(orderdate) from ddate1;	1.502	1.687	10.12	10.059
19	drop table ddate1; drop table datee; drop table n_r; drop table suppl_n;	0.638 0.094 0.086 0.102	0.106 0.097 0.083 0.081	0.099 0.118 0.143 0.1	0.517 0.075 0.075 0.083

Now by executing show tables; Following tables will be reflected.

- customer
- ddate
- lineorder
- part
- partsupp
- supplier

6. Performance comparison Queries

6.1. Aggregate Query

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
1	select max(tax) from lineorder;	7.218	70.322	15.07	31.839
2	select min(tax) from lineorder;	7.206	64.272	15.04	34.038
3	select avg(tax) from lineorder;	7.208	62.244	15.11	34.897
4	select sum(tax) from lineorder;	7.191	62.32	15.03	34.913
5	select count(*) from lineorder;	3.187	24.451	14.09	25.489

6.2. NVL Query

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
6	select NVL(discount,0) from lineorder;	0.043	76.458	0.08	39.005

6.3. Query with multiple constraint on the fact table

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
7	select tax,orderkey,linenumber,partkey,ordertotalprice from lineorder where shipmode in ('MAIL','AIR') and ordertotalprice in (select max(d.ordertotalprice) from lineorder d union select min(e.ordertotalprice) from lineorder e) and discount > 0.07;	32.853	249.0	63.10	134.02

6.4. Query on single dimension

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
8	select custkey,sum(revenue) from lineorder group by custkey;	10.214	118.93	27.92	329.93

Query Description:

1. To get the revenue earned from each customer

6.5. Query Constraining on 1 dimension

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
9	select sum(l.quantity) as revenue from lineorder l,ddate d where l.orderdate==d.datekey and t_date==2 and year > 1994;	13.865	107.765	0.852	47.094
10	select avg(l.supplycost) as cost from lineorder l,ddate d where l.orderdate==d.datekey and t_date>1993-01-12 group by month;	13.829	105.959	19.598	155.828
11	select avg(l.supplycost) as revenue from lineorder l,ddate d where l.orderdate==d.datekey and month between 1 AND 6 and todays_date<15 and year<1994 group by month;	15.895	119.07	23.56	114.645
12	select avg(l.tax) as revenue,l.custkey from lineorder l,ddate d where l.orderdate==d.datekey and month between 1 AND 6 and todays_date<15 and year<1994 group by month,l.custkey order by l.custkey;	17.974	140.06	39.77	158.93

Query Description:

1. To get the quantity of product sold in month of Feburary.
2. Average of supplycost on monthly basic after 1993.

6.6. Query Constraining on 2 dimensions

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
13	select c.name,c.city from lineorder l,customer c where l.custkey==c.custkey and c.name like 'A%' and l.discount in (select max(ll.discount) from lineorder ll) order by c.name,c.city desc;	24.582	359.51	60.05	263.89
14	select p.name,p.mfgr,p.brand from part p,lineorder l where l.orderdate>1992-01-01 and l.orderdate<1993-02-01 and p.container like '%iron%' and p.type like '%BOX%' order by p.name desc,p.brand asc;	15.334	104.66	33.98	92.846

Query Description:

1. select customer who got highest discount and whose name starts with A

2. select parts ordered between 1992-01-01 and 1992-02-02 which is having container like iron and part type as box.

6.7. Query Constraining on 3 dimensions

Sr. No	Query	Time taken SN (1 GB)	Time taken MN(1GB)
15	select p.name,p.mfgr,p.brand,s.name from part p,lineorder l,supplier s,ddate d where l.orderdate>1992-01-01 and l.orderdate<1993-02-01 and p.container like '%iron%' and p.type like '%BOX%' and s.nation='INDIA' or s.nation='INDONESIA' group by d.year, p.name,p.mfgr,p.brand,s.name order by p.name desc,p.brand asc;		

6.8. Query with not exist clause

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
16	select s.city,s.nation from customer s where NOT EXISTS (select c.city,c.nation from supplier c where s.city==c.city);	36.517		39.37	2383.0

6.9. Query with left outer join

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
17	select c.city,c.nation from customer c left outer join supplier s on c.city=s.city where s.city is null;	31.587		39.68	2312.8

Query Description:

1. select city, nation where we have customer but no supplier.

6.10. Top N query performance

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
18	select * from lineorder sort by orderdate desc Limit 10;	61.375	151.59	51.83	51.83
19	select * from lineorder where quantity>2 sort by orderdate desc Limit 10;	58.363	120.59	53.21	53.21

6.11. Query on View verses Query directly on fact table

Sr. No	Query	Time taken SN (1 GB)	Time taken SN (10GB)	Time taken MN (1GB)	Time taken MN (10GB)
20	create view detailss1 as select * from part where size>7 and MFGR="Manufacturer#1" and brand like "Brand#1%";	0.093	0.581	0.219	0.714
21	select * from detailss1 where container = "JUMBO PKG" ;	0.158	69.547	0.089	155.82
22	select * from part where size>7 and MFGR="Manufacturer#1" and brand like "Brand#1%" and container ="JUMBO PKG" ;	0.039	80.649	0.072	114.64

Query Description:

2. Select from view
3. Select from base table

6.12. Partitioning

Enabling partitions require following settings on hive:

SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;

Sr. No	Query	Time taken SN (1 GB)	Time taken MN(1GB)
23	create table cust2 as select * from customer;	1.435	10.05
24	select * from cust2 where mktsegment='BUILDING' and nation='JORDAN';	0.034	0.904
25	create table partitioned_cust(custkey int,name varchar(64),address varchar(64),city varchar(64),phone varchar(64)) PARTITIONED BY (mktsegment varchar(64),nation varchar(64)) stored as sequencefile;	0.148	0.186
26	insert into table partitioned_cust Partition(mktsegment,nation) select	18.64	32.581

	custkey,name,address,city,phone,mktsegment,nation from cust2;		
27	select * from partitioned_cust where mktsegment='BUILDING' and nation='JORDAN';	0.153	0.163

6.13. Index vs Partitioning

6.13.1. Point query

Sr. No	Query	Time taken SN (1 GB)	Time taken MN(1GB)
28	CREATE INDEX table01_index ON table customer(mktsegment,nation) AS 'COMPACT' WITH DEFERRED REBUILD;	0.262	0.187
29	alter index table01_index on customer rebuild;	2.577	15.256
30	select * from cust2 where mktsegment='BUILDING' and nation='JORDAN';	0.037	0.076
31	select * from customer where mktsegment='BUILDING' and nation='JORDAN';	0.047	0.059
32	select * from partitioned_cust where mktsegment='BUILDING' and nation='JORDAN';	0.046	0.103
33	DROP INDEX table01_index ON customer;	0.099	0.099
34	CREATE INDEX table01_index ON table customer(mktsegment,nation) AS 'BITMAP' WITH DEFERRED REBUILD;	0.121	0.069
35	alter index table01_index on customer rebuild;	3.553	17.355
36	select * from customer where mktsegment='BUILDING' and nation='JORDAN';	0.032	0.051
37	DROP INDEX table01_index ON customer;	0.09	0.066

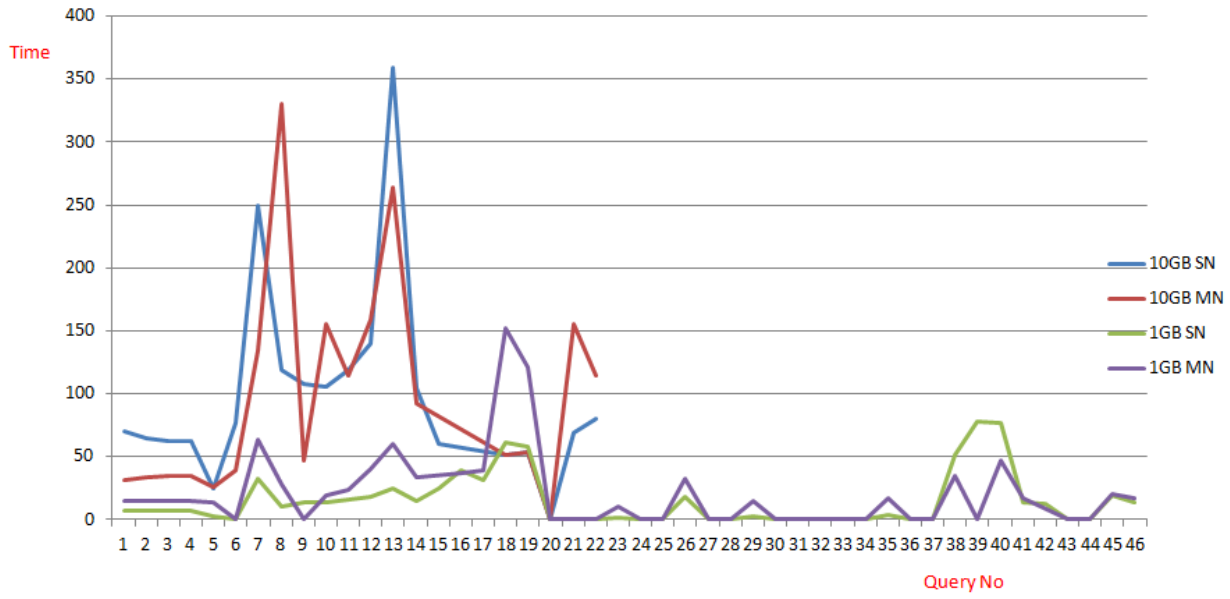
6.13.2. Range query

Sr. No	Query	Time taken SN (1 GB)	Time taken MN(1GB)
38	create table lineorder2 as select * from lineorder;	51.417	34.66
39	CREATE INDEX table01_index ON table lineorder(discount) AS 'BITMAP' WITH DEFERRED REBUILD;	78.171	0.093
40	alter index table01_index on lineorder rebuild;	76.675	46.71
41	select count(*) from lineorder2 where discount<0.08 and discount>0.05;	14.06	17.09
42	select count(*) from lineorder where discount<0.08 and	12.262	

	discount>0.05;		
43	DROP INDEX table01_index ON lineorder;	0.759	0.936
44	CREATE INDEX table01_index ON table lineorder(discount) AS 'COMPACT' WITH DEFERRED REBUILD;	0.279	0.334
45	alter index table01_index on lineorder rebuild;	19.717	19.88
46	select count(*) from lineorder where discount<0.08 and discount>0.05;	14.125	17.23

7. Analysis

As the data size is increasing, performance of multi-node cluster improves. After a certain data size threshold, hive overcomes the time it takes to communicate to nodes over network.



Observations on query:
Query on view run faster as compared to query on base tables.
Outer join runs faster than not exist to acquire same result.
Bitmap indexes outperforms compact index scheme and partitioning.

8. References

[1] Star Schema Benchmark by Pat O'Neil, Betty O'Neil, Xuedong Chen

[2] The Star Schema Benchmark (SSB) by Pat O'Neil, Betty O'Neil, Xuedong Chen