به نام خدا

گزارش پروژه پایگاه داده پیشرفته

استاد درس: دکتر امین غیبی

نام و نام خانوادگی: حسین نجاتی جوارمی

شماره دانشجویی: 9412057

آدرس گیت هاب:https://github.com/HosseinNejatiJavaremi/AdvanceDatabase

Contents

مقدمه:

در این گزارش ما ابتدا دو سیستم پردازش آنلاین OLTP و OLAP را با هم مقایسه می کرده سپس تعدادی از ابزارهای مورد استفاده در این دو محیط را با هم مقایسه می کرده و نتایج آن را برای هر سیستم اعلام می شود.

تمام کد های نوشته شده در طی انجام این پروژه در گیت هاب قرار گرفته و تمامی افراد حق دسترسی و استفاده از نتایج آن را دارند.

OLTP vs OLAP:

OLAP:	OLTP:
1. بیشتر خواندن	1. بیشتر نوشتن و بروزرسانی
2. کوئری های بزرگ و پیچیده	2. کوئری های کوچک و ساده
3. حجم داده زیاد	3. می تواند خیلی کم باشد
4. زمان زیادی صرف پردازش کوئری می شود	4. نسبتا کمتر است
5. ساختار دیتابیس معمولا نرمال نیست	5. معمولا نرمال هستند(3NF)

OLTP:

1. PostgreSQL:

در این قسمت ابتدا دیتابیس و جدول ها را ساختیم و با کمک دستور زیر داده ها را از روی فایل خوانده و به داخل دیتابیس ریختیم.

```
Copy table_name from file_addres.file_name.tbl (format csv, delimiter '|')
```

نکته: این دستور به صورت مستقیم پاسخگوی نیاز ما نبود و ما ابتدا باید به تمام جداول یک ستون موقتی اضافه می کردیم و پس از ذخیره کردن داده برروی دیتابیس، آن ستون ها را پاک می کردیم. دلیل این کار وجود علامت '|' در آخر تمامی رکورد ها بود.

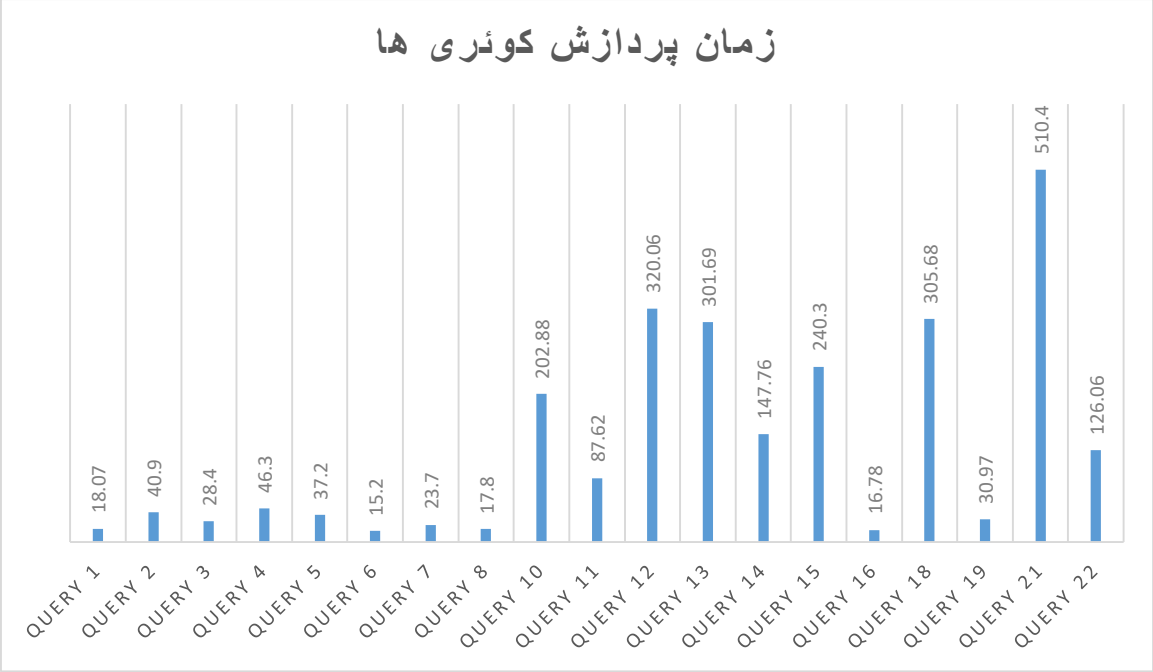
نکته: انجام عمل ذخیره سازی داده برروی دیتابیس زمان بر می باشد و با توجه به شلوغی سرور بین 6 تا 14 ساعت ممکن است طول بکشد.

نکته: حجم داده های ذخیره شده برروی دیتابیس بیشتر از داده ها خام می باشد و دلیل آن هم تغییرات اعمال شده در هنگام ذخیره سازی و ایندکس های ساخته شده می باشد.

در مرحله بعدی کوئری های بنچمارک TPC-H دانلود کرده و آن ها را با مقادیر مجاز کامل کردیم. به دلیل زمان بر بودن اجرای این کوئری ها ما از نرم افزار screen استفاده کردیم تا در صورت قطعی اینترنت یا هر دلیل دیگری اجرای کوئری متوقف نشود.

نکته: برای این که روند تصمیم گیری و اجرا کوئری ها را در دیتابیس ببینیم به اول تمامی کوئری ها دستور explain analyze اضافه شد. روند اجرای کوئری ها در گیت هاب موجود می باشد.

نکته: زمان پردازش کوئری های شماره 9، 17 و 20 بیش از 10 ساعت شد، به همین دلیل اجرای این کوئری ها به صورت دستی متوقف شد.



نکته: تمامی زمان های اعلام شده در جدول بالا به دقیقه می باشد.

نکته: مقادیر جدول بالا، زمان های پردازش می باشد و با زمان واقعی اجرای کوئری ها بسیار متفاوت می باشد و یکی از مهم ترین دلایل این تفاوت دیسک مورد استفاده می باشد.

نکته: در اجرای این کوئری ها مقدار رم و سی پی یو استفاده شده ناچیز می باشد و هزینه اصلی خواندن و نوشتن برروی دیسک می باشد.

2. Spark

در این قسمت ابتدا فایل ها از روی هارد خوانده شده و پس ازانجام تغییرات لازم در فرمت parquet داخل HDFS ذخیره شدند. پس از آن فایل در فرمت parquet خوانده شده و در فرمت های orc و avro داخل HDFS ذخیره شدند. تمامی کد های تبدیل فرمت و جابجایی داده داخل گیت ها قرار گرفته اند.

نکته: هنگام تبدیل داده خام به فرمت parquet ما باید ساختار جداول را مشخص کنیم اما در مراحل بعدی نیازی به این کار نیست.

نکته: حجم داده در هنگام ذخیره سازی در فرمت های parquet، orc و avro به دلیل فشرده سازی داده ها کاهش پیدا می کند.

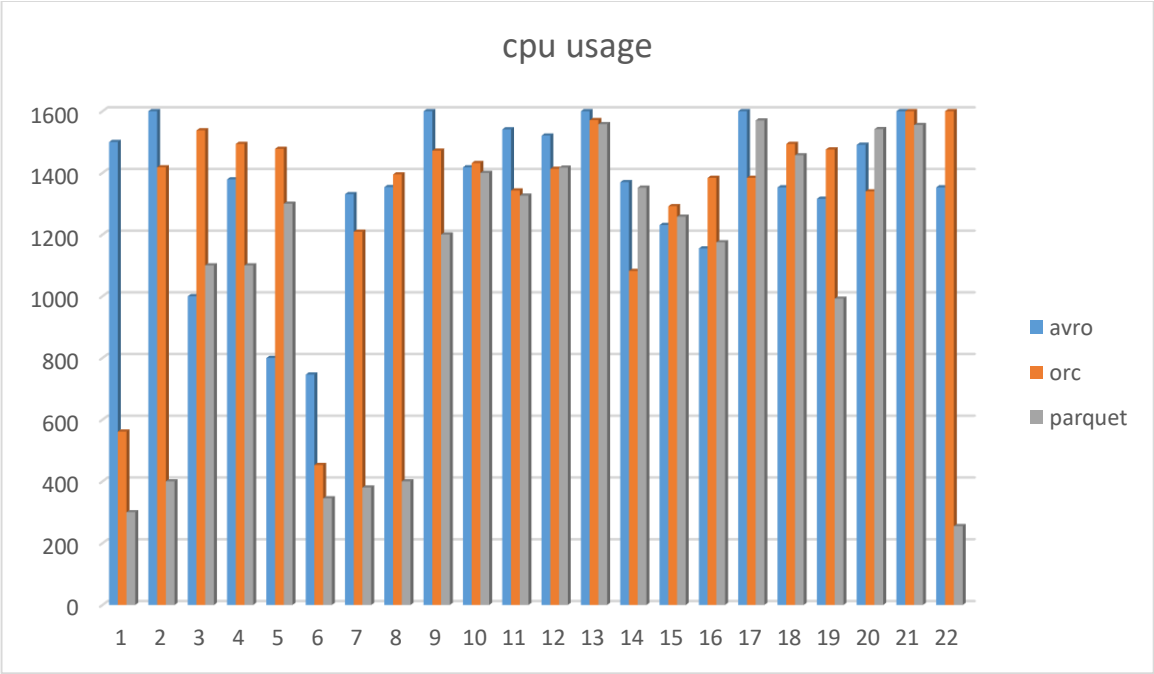
نکته: در هنگام استفاده از فرمت orc باید از تنظیم زیر استفاده کنید:

```
sqlContext.setConf('spark.sql.orc.iml', 'native')
```

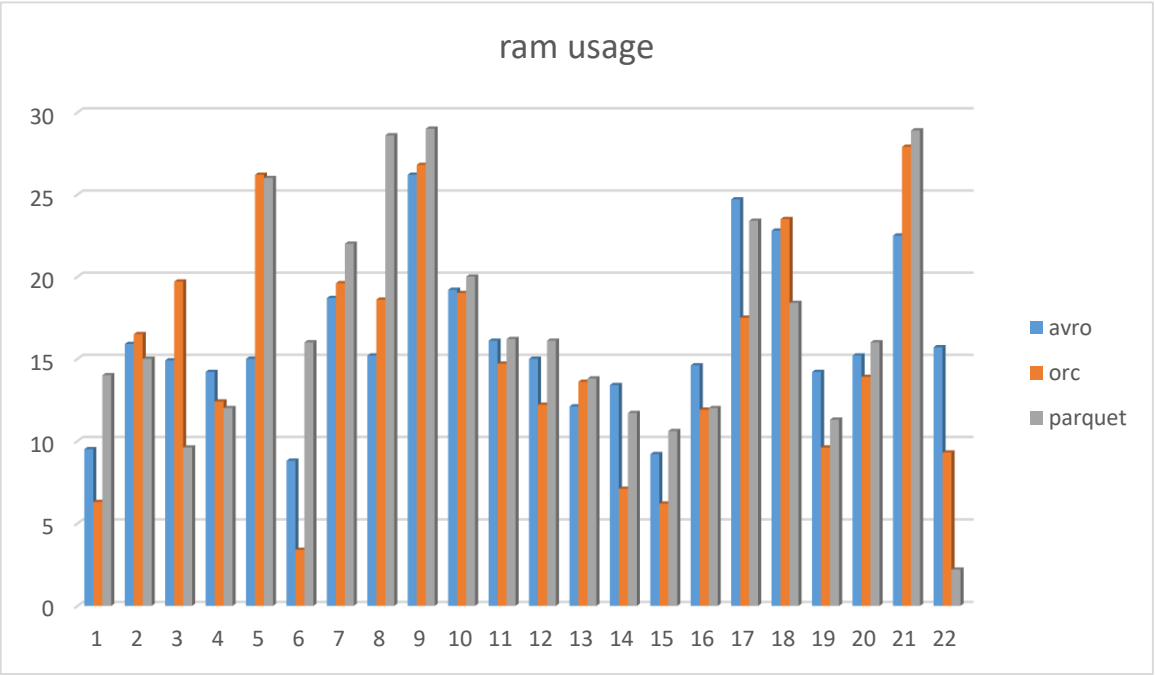
نکته: در هنگام استفاده از فرمت avro باید دو کتابخانه به فایل های jar اسپارک اضافه کنید. این کتابخانه ها در قسمت jar_files داخل گیت هاب قرار گرفته اند.

نکته: کد ها به زبان پایتون نوشته شده اند و از pyspark استفاده شده است.

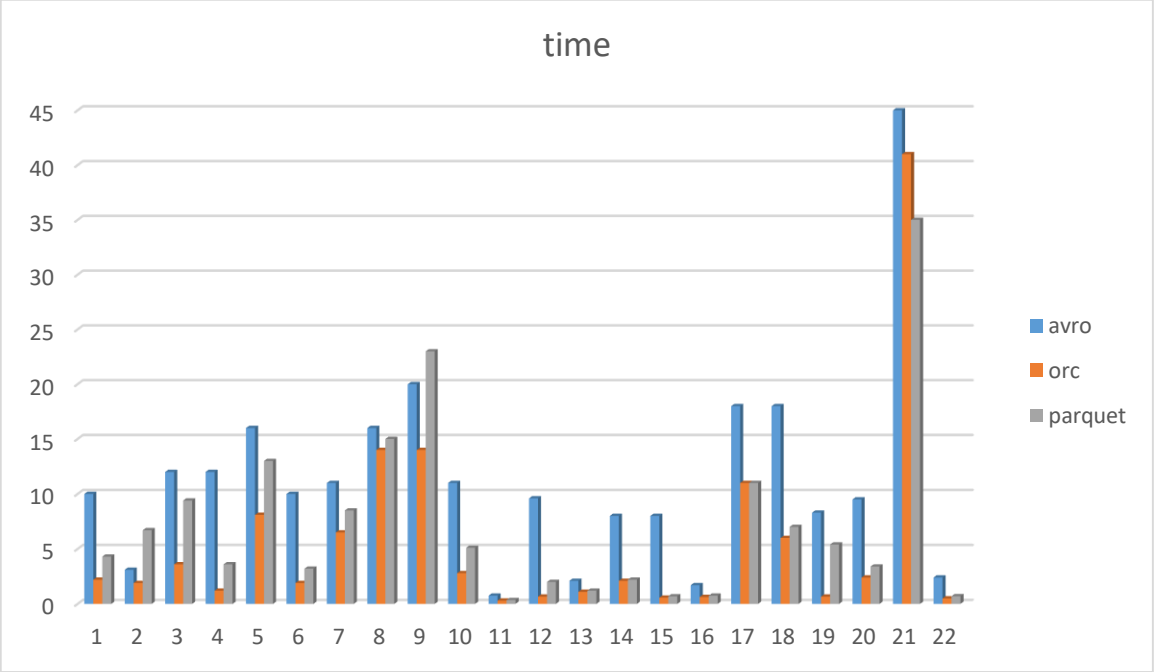
نکته: تمامی کوئری جداگانه اجرا شدند و برای هر کوئری یک سشن جداگانه ساخته شد. دلیل این کار کاهش مراحل اجرای کوئری ها و عدم وابستگی و تاثیر کوئری ها برروی یک دیگر می باشد. در طی انجام گزارش مشاهده شد که اگر بعد از اتمام کوئری ها، سشن ها بسته نشود و کوئری بعدی بر روی هم سشن اجرا شود، حجم زیادی داده از اجرای قبلی برای روی رم کش شده است و سرعت اجرای کوئری جدید کاهش پیدا می کند.



نکته: میزان مصرف سی پی یو براساس استفاده از تمام 16 هسته موجود می باشد.



نکته: بر روی اسپارک محدودیت سقف 30 گیگابایت رم گذاشته شده بود.



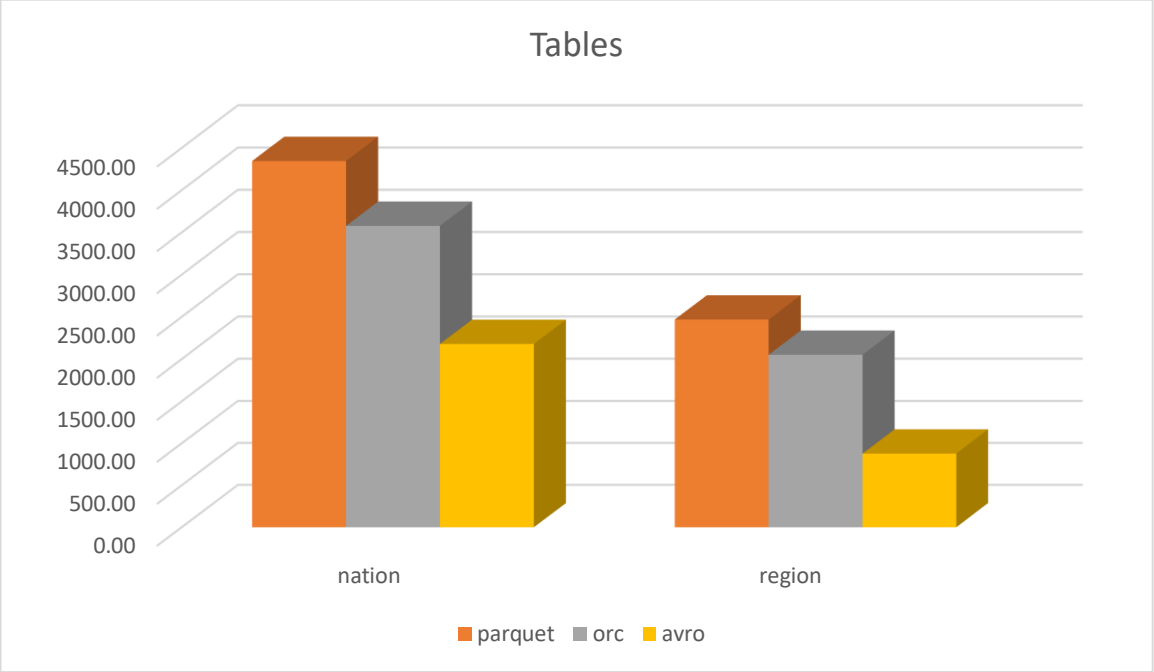
نکته: تمام زمان های بالا برحسب دقیقه می باشند.

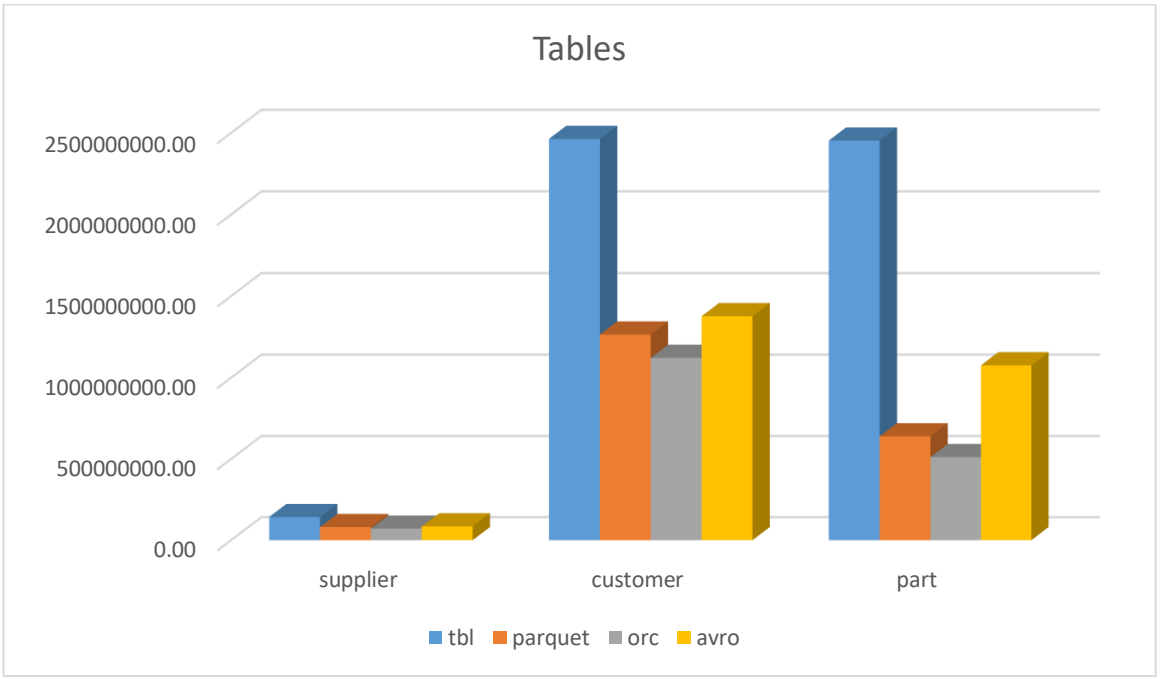
نکته: مقادیر بالا کل زمان اجرا می باشد و با زمان پردازش متفاوت است.

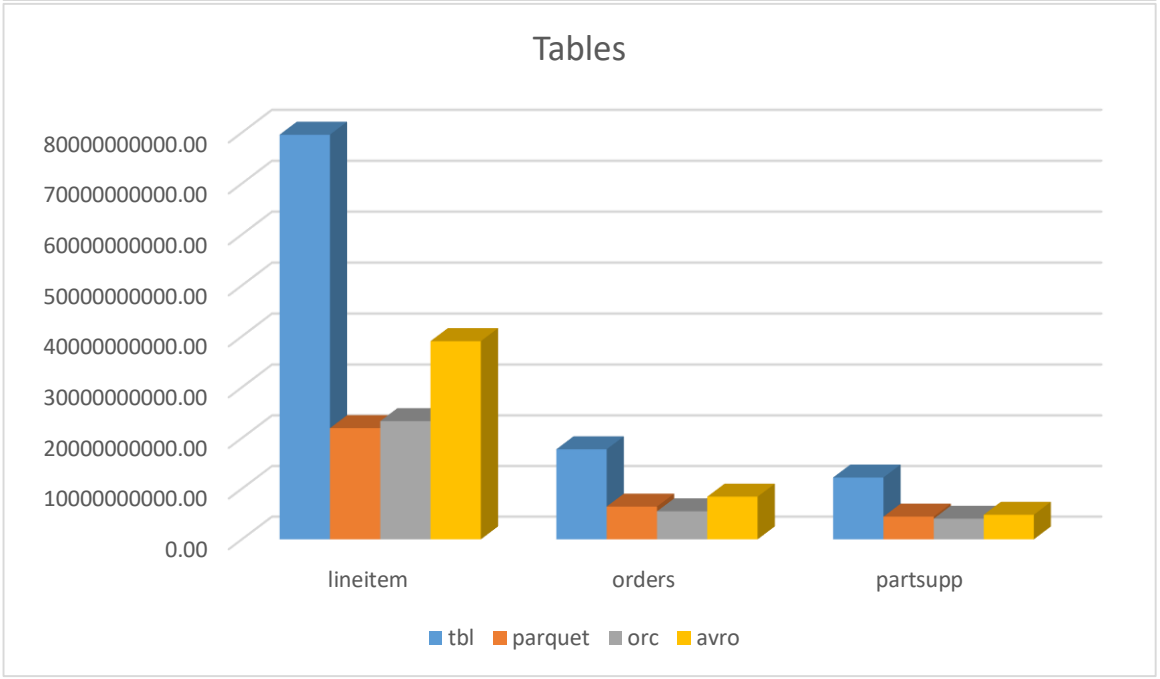
3. مقایسه حجم فایل ها در فرمت های مختلف

نمودار زیر حجم داده های جدول ها را در فرمت های tbl، parquet، orc و avro نشان می دهد.

نکته: حجم های زیر برحسب بایت محاسبه شده اند.



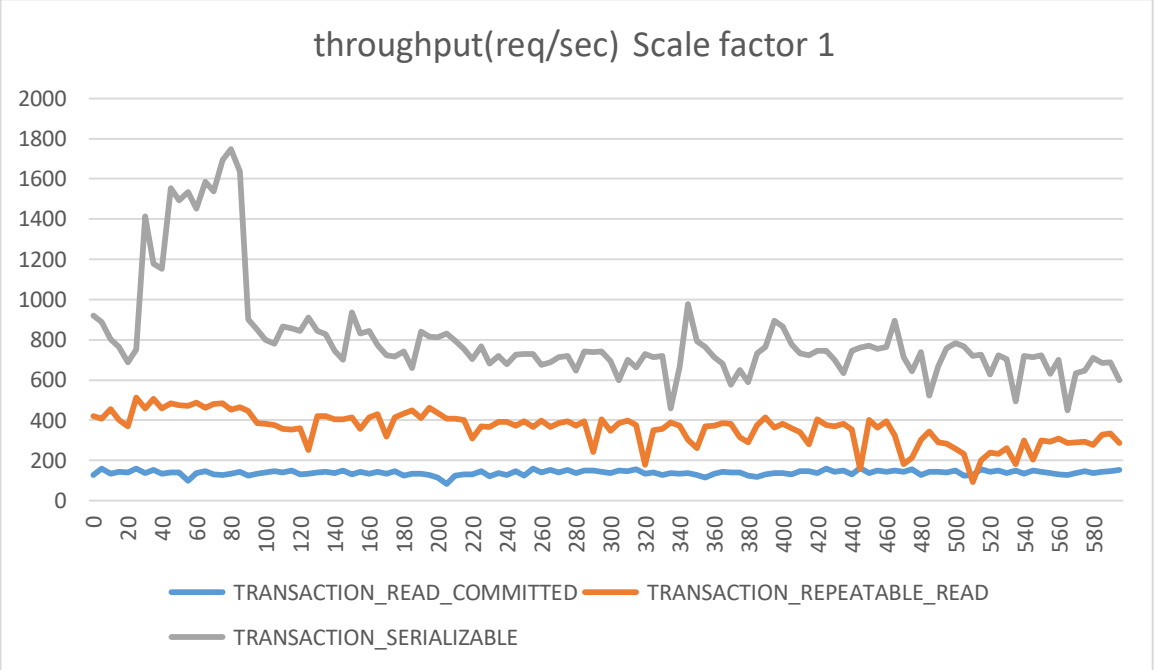

Tables
2500000000.00
2000000000.00
1500000000.00
1000000000.00
500000000.00
0.00
supplier
customer
part
tbl
parquet
orc
avro

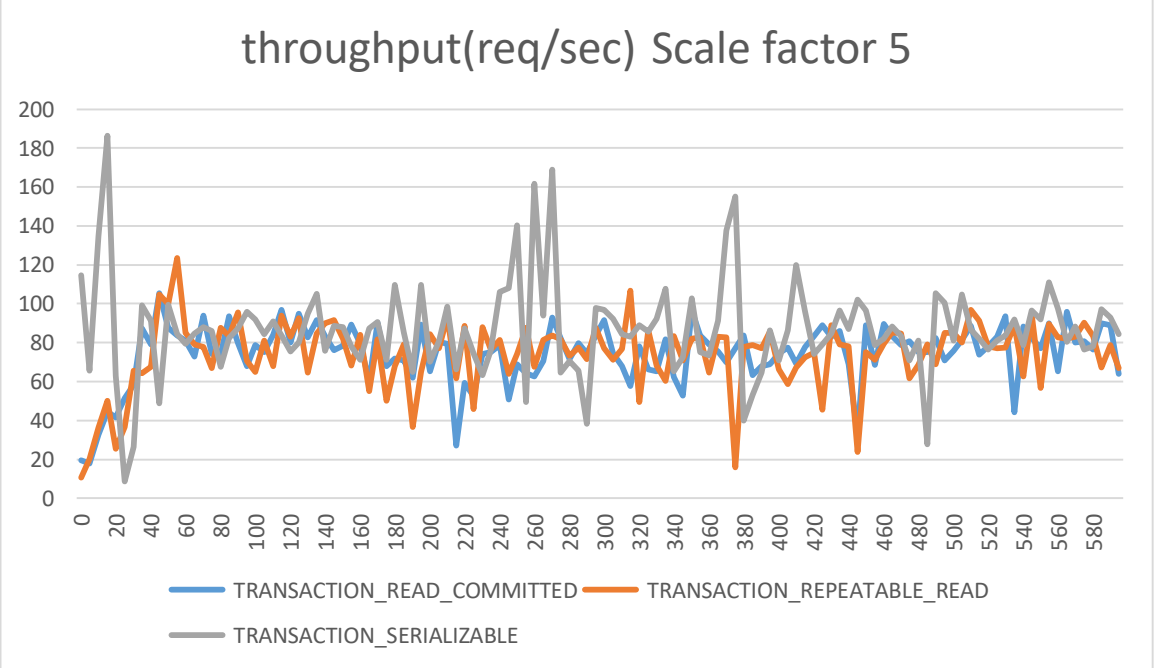

Tables
80000000000.00
70000000000.00
60000000000.00
50000000000.00
40000000000.00
30000000000.00
20000000000.00
10000000000.00
0.00
lineitem
orders
partsupp
tbl
parquet
orc
avro

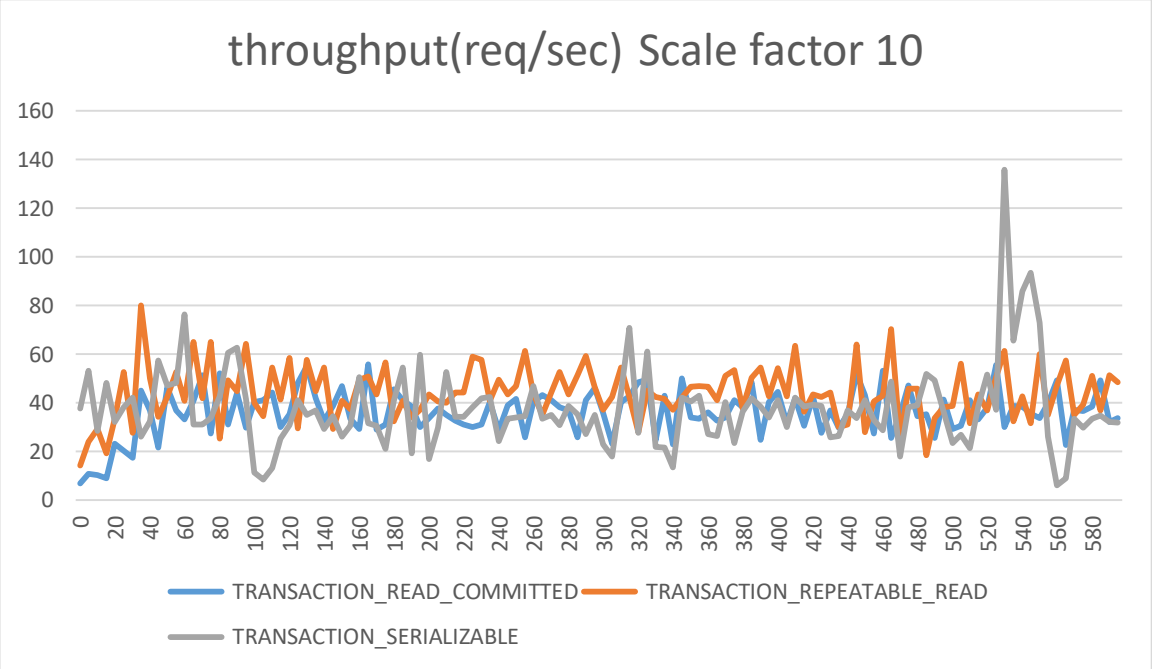
:OLAP

1. PostgreSQL:

در این قسمت با استفاده از ابزار OLTPBenchmark ابتدا به دیتابیس متصل می شویم و با توجه به مقدار ضریب بزرگنمایی داده های مورد آزمایش را می سازیم. در این قسمت ما از ضریب بزرگنمایی 1، 5 و 10 استفاده کردیم.هر چه مقدار ضریب بزرگنمایی ما بیشتر باشد، حجم داده ما بیشتر می شود. درنتیجه سرعت ما کاهش میابد. علاوه بر مقدار ضریب بزرگنمایی، ما از سه حالت کنترل همزمانی که دیتابیس پستگره پشتیبانی می کند هم استفاده کردیم و نتایج حاصل به صورت زیر بود.







تمامی فایل های کانفیگ و نتایج حاصل در گیت هاب موجود می باشد.

نکته: در این جا هم مشاهده شد که به دلیل سرعت کم هارد دیسک، میزان کمی سی پی یو و رم استفاده شد.

نکته: با کاهش ضریب بزرگنمایی دیتابیس، تعداد deadlock ها بیشتر می شود.

نکته: برای استفاده از این ابزار حتما باید از جاوا 7 یا 8 نصب کرد.

کامند های مورد استفاده در این قسمت به صورت زیر می باشند:

./oltpbenchmarkn-b tpcc –c config_address –create=true –load=true

./oltpbenchmarkn-b tpcc –c config_address –execute=true –s time_preiod –o output_file_address

2. CockroachDB

در این قسمت هم از ابزار OLTPBenchmark استفاده می کنیم که توسط Robert S Lee برای دیتابیس cockroachdb تغییر کرده است.این ابزار را می توانید با دستور زیر دانلود و نصب کنید:

```
git clone https://github.com/robert-s-lee/oltpbench.git --branch cockroachdb --single-branch oltpbenchmark; cd oltpbenchmark; ant
```

متاسفانه سرعت cockroachdb به شدت پایین بود و حتی بعد از گذشت یک روز نتوانست داده مورد نیاز برای تست را تولید کنند. زمان وارد کردن داده برحسب ثانیه برابر 0.15 الی 0.3 بود.

فایل های کانفیگ این نرم افزار در گیب هاب قرار گرفته اند.

کامند های مورد استفاده در این قسمت به صورت زیر می باشند:

```
./oltpbenchmarkn-b tpcc –c config_address –create=true –load=true
```

```
./oltpbenchmarkn-b tpcc –c config_address –execute=true –s time_preiod –o output_file_address
```

منابع مطالعاتی:

1. world wide web
2. کتاب docker in action
3. داکیومنت های اسپارک
4. دوره آموزشی Taming Big Data with Apache Spark and Python – Hands On!
5. دوره آنلاین آموزش لینوکس توسط مهندس جادی
6. منابع سایت https://developer.ibm.com/tutorials/l-lpic1-map/
7. کتاب learning Apache Flink