بردیا حامدمحسنی

# گزارش پروژه

# درس پایگاه‌داده پیشرفته

دانشگاه صنعتی امیرکبیر

دی ماه ۹۷

# مقدمه

در این گزارش سعی شده تا به مستند کردن تمام مراحل انجام هر بخش از پروژه و ارائه لینک ها و تکه‌کدهای استفاده شده پرداخته شود.

برای برقراری ارتباط SSH با سرور، از نرم‌افزار MobaXtrem استفاده شد، این نرم-افزار علاوه بر مدیریت آسان کلیدهای SSH امکان مدیریت فایل‌های داخل سرور و کپی اطلاعات بین دو کامپیوتر را فراهم می کند.

پس از ساخت کلید Private و کلید Public از روی آن، کلید پابلیک به مسئول سرور تحویل داده شد و ارتباط با سرور برقرار شد.

یادگیری مقدماتی تکنولوژی Docker و نحوه استفاده از آن از چالش‌های اولیه این پروژه بود، دستورات پرکاربرد Docker در این پروژه عبارت اند از:

۱- Docker exec –it containerName bash که با آن به کانتینر مورد نظرمان وارد می‌شویم.
۲- Docker container ls که با آن لیست کانتینر های موجود و وضعیت آن‌ها را می‌توان بررسی کرد.
۳- Docker cp sourcePath destinationPath که با آن میتوان بین سیستم اصلی و کانتینر های داخلی یا بالعکس، فایل جابجا کرد.

همچنین، برای تضمین ارتباط برقرار شده با سرور در کانکشن‌های طولانی و ایجاد همزمانی بین چند عملیات طولانی، از قابلیت screen در لینوکس استفاده شده و صفحات بسته شده با دستور screen –x screenId مجدد قابل استفاده و ادامه اند.

# بخش اول : OLAP

در این بخش ابتدا به پیاده سازی PostgreSQL پرداخته شد. پس از ورود به کانتینر postgres با استفاده از دستور psql –U root وارد دیتابیس روت شده و با استفاده از کوئری های لینک زیر، دیتابیس ساخته شد:

https://github.com/dragansah/tpch-dbgen/blob/master/tpch-create.sql

پس از ساخت دیتابیس، نوبت به خواندن اطلاعات و پر کردن دیتابیس بود، فایل‌های OLAP در آدرس data/OLAP_Benchmark_data/ قرار دارند.با استفاده از دستور COPY در Postgres و مشخص کردن کارکتر جداکننده فایل های OLAPداخل پایگاه داده ریخته شدند. البته به دلیل کمبود یک جداکننده در انتهای سطرهای فایل csv تغییراتی در دستور کپی داده شد تا عملیات با موفقیت انجام شود. همچنین برای ثبت زمان اجرای هر کوئری و مراحل آن دستور EXPLAIN ANALYSE به ابتدای کوئری‌ها اضافه شده و با توجه به مقادیر ثابت وارد شده در کوئری، خروجی بعضی از داده‌ها نیز Limit شدند.

پس از کپی اطلاعات داخل دیتابیس، با استفاده از دو دستور زیر کوئری‌های آدرس data/OLAP_Benchmark_queries/ تک تک اجرا شده و نتایج آن در فایل ثبت شد:

/o filePath/outputFileName

/i filePath/inputFileName

فایل‌های نهایی نتایج اجرای این کوئری‌ها در گیت‌هاب قرار داده شده اند و مقدار عددی زمان صرف شده در نمودار نهایی بخش OLAP اضافه خواهد شد.

مرحله بعد بخش مرتبط با Spark و NoSQL است. در این بخش ابتدا با یک اسکریپت پایتون به اسم ParquetHDFS.py و با استفاده از کتابخانه Pyspark فایل‌های csv را به فایل Parquet تبدیل کردیم و روی یک name node مخصوص از HDFS ذخیره شدند. در این تبدیل حجم اطلاعات به مراتب کمتر شد. همچنین

برای تبدیل فایل در قسمت‌های بعد از اسکریپت مشابهی با نام OrcHDFS.py و دو فایل jar که در پوشه مخصوص قرار داده شده استفاده شد که فایل‌های فورمت Avro و Orc را تولید کردند. تفاوت اسکریپت OrcHDFS با اسکریپت اول این است که مستقیم اطلاعات Parquet را از روی HDFS می خواند و به فورمت جدید در می‌آورد. لازم به ذکر است که در فایل OrcHDFS تنها تبدیل مربوط به فایل region نوشته شده و تبدیل های فایل‌های دیگر مشابه هستند. فایل های JAR نیز از اینترنت توسط دیگر دانشجویان پیدا شده و مربوط به spark avro هستند.

در مرحله بعد، نوبت به تبدیل کوئری‌های SQL به فورمت Dataframe بود. با دو زبان Python و Scala می‌توان به Dataframe های Spark دسترسی داشت و با آن‌ها کار کرد که به دلیل آشنایی قبلی و سادگی، زبان Python مجددا انتخاب شد. کوئری‌های تبدیل شده همگی در فایل Spark DF Queries.py ذخیره شده و قسمت قسمت در محیط pyspark اجرا شدند. در ابتدای هر کوئری یک تایمر تعریف شده که با استفاده از آن، زمان دقیق اجرای کوئری‌ها ثبت و یادداشت شد، نتایج این کوئری‌ها نیز در فایل ثبت شده و در نمودار نمایش داده می‌شوند.

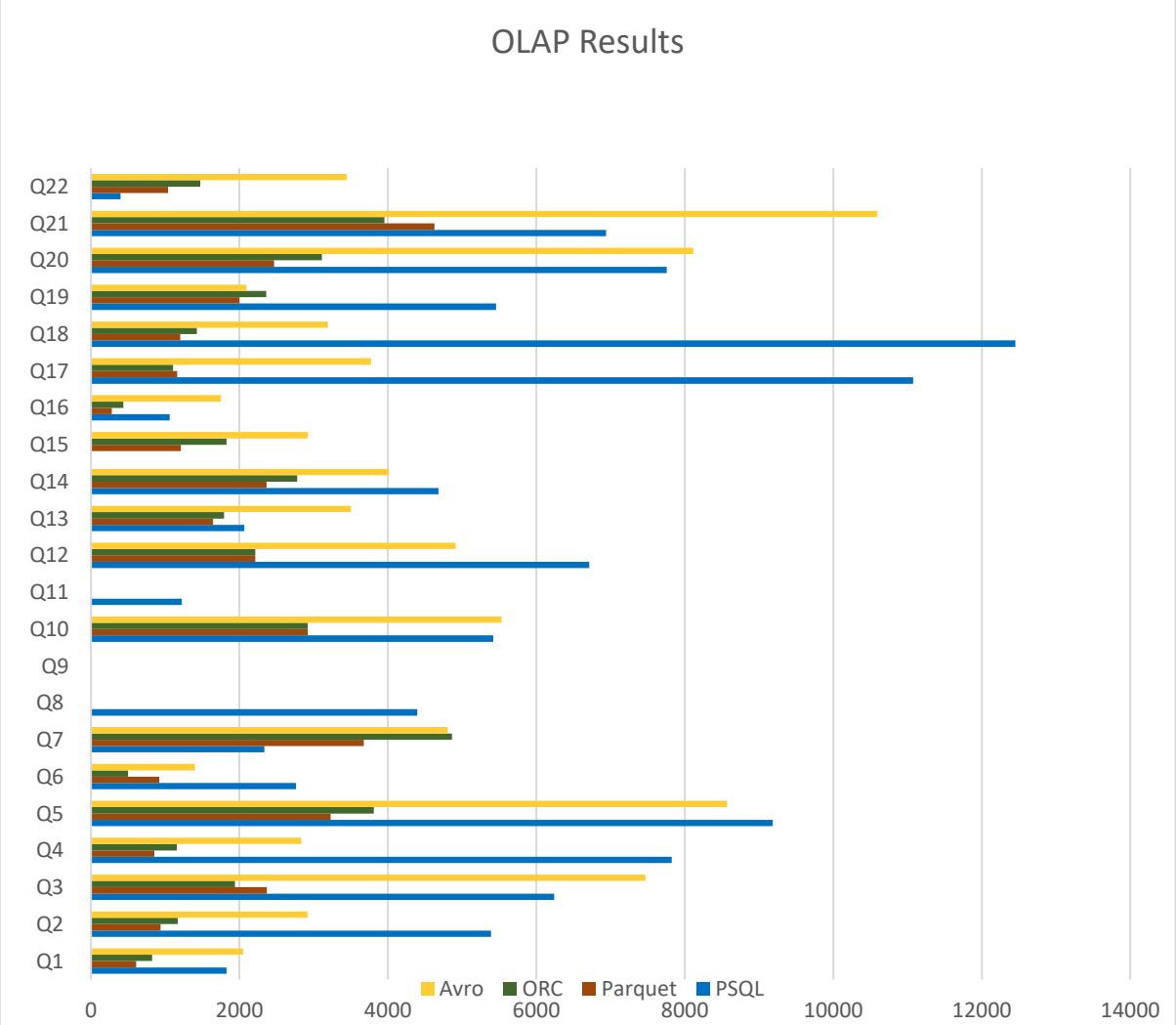
برای ورود به محیط pyspark ، کافی است پس از ورود به کانتینر spark-master در داکر، دستور pyspark را وارد کنید.

*توجه: نکته عجیبی که در اکثر کوئری‌های postgres با آن برخوردیم، تفاوت زمان چاپ شده توسط Explain Analyze و زمان واقعی اجرای کوئری بود. همچنین، در تمام مراحل این پروژه زمان‌های اجرایی هریک از دانشجویان با دیگری کاملا متفاوت است و گاهی اوقات تفاوت در حد چند ساعت است! این موضوع نشان دهنده تفاوت قدرت سرورها یا تفاوت منابع در دسترس هر کاربر در زمان اجرای آن کوئری دارد، احتمالاً تخصیص منابع در داکر به خوبی انجام نشده و فعالیت همزمان افراد رو سرور، اثر مستقیم روی نتایج کار یکدیگر دارد. با این حال، حتی زمانی که پروسه دیگری در حال اجرا نبود نیز روی بعضی کوئری‌ها تفاوت چند برابری زمان بین دو سرور مختلف مشاهده شده است. در نتیجه صحت بنچمارک‌های انجام شده قابل تضمین نبوده و به نظر می‌آید که نتایج حاصله برای مقایسه پایگاه داده‌های مختلف کاربردی نیستند.*

به دلیل کمبود وقت و درگیریهای شخصی متاسفانه موفق به انجام کامل قسمت Flink نشدم و نتایج آن در این گزارش نیامده است. همچنین اجرای کوئری های ۹ و ۱۵ در PSQL بسیار طولانی شده (بیش از ۴۸ ساعت) و حتی پس از ۲ بار تلاش و اضافه کردن Begin/Commit تغییری در نتیجه آن حاصل نشد. بعضی دیگر از کوئری ها نیز به دلیل پیچیدگی بیش از حد تبدیل آنها به dataframe انجام نشده اند.

نتایج مربوط به بخش OLAP در ادرس Results/OLAP ذخیره شده اند، همچنین نتایج Explain Analyses کوئری های Psql را می توانید به تفکیک در فولدر Postgresql Detailed پیدا کنید.

نمودار مقایسه خروجی OLAP نیز در فایل OLAP_Comparison.xlsx تولید و ذخیره شده است.



با مطالعه نمودار می‌توان نتیجه گرفت عملکرد روش های ORC و Parquet از نظر زمان بهتر از Avro و هر سه آنها بهتر از PSQL هستند. البته مجدداً ذکر این مطلب حائز اهمیت است که به دلیل عدم ثبات سرور ها و عدم اطلاع از نحوه کانفیگ نرم-افزارها در زمان نصب و همزمانی تست کاربران روی سرور، اعتبار این بنچمارک‌ها مورد سوال است.

## بخش دوم : OLTP

در این بخش، مقایسه زمان و نتایج اجرای پلتفرم OLTPBench روی دو پایگاه داده مختلف مورد بحث بود. پایگاه داده رابطه ای اول Postgresql و پایگاه داده دوم از نوع NewSQL بود که بین دو گزینه CockroachDB و VoltDB امکان انتخاب وجود داشت.

در قسمت psql مراحل بسیار ساده و طبق راهنمای موجود در کوئرای پروژه به آدرس https://github.com/oltpbenchmark/oltpbench/wiki کار انجام شد. پس از بیلد پروژه با ant و ساخت دیتابیس و تغییر فایل config.xml، دستور زیر برای شروع تست tpcc اجرا شد.

```
/.oltpbenchmark -b tpcc -c config/config.xml --create=true --load=true --execute=true -s 5 -o outputfile
```

خروجی اجرای این دستور هم در آدرس results/oltp/psql.csv موجود است.

برای بخش NewSQL ابتدا برای اجرای VoltDB تلاش شد. بین فایل‌های Git ریپازیتوری VoltDB در دو آدرس جداگانه نرم‌افزار تست tpcc قرارداده شده ولی پس از دانلود و کپی اطلاعات روی کانتینر، در build کتابخانه‌های جاوا و اصلاح آدرس و اجرای اسکریپت مشکل وجود داشت. نهایتاً علیرغم تلاش بسیار تصمیم به استفاده از CockroachDB شد. به کمک دوستان، یکی از branchهای ریپازیتوری oltpbench که مخصوصاً برای CockroachDB آماده شده بود را پیدا کرده و با استفاده از ant بیلد گرفته شد. سپس با دستور مشابه و اندکی تغییرات در فایل کانفیگ تست tpcc

اجرا شد. خروجی آن در آدرس results/oltp/cockroach.csv قرار گرفته است. البته لازم به ذکر است که برای رفع مشکلات بیلد و اجرای پلتفورم و ایجاد هماهنگی بین نسخه‌های JRE و JDK زمان زیادی صرف شد که خوشبختانه در نهایت نتیجه بخش بود.

نمودار زیر نتیجه مقایسه دو آماره Throughput (تعداد تراکنش در ثانیه) و average_lat (میانگین تاخیر به میلی ثانیه) است. نمودار دقیق تر را می‌توانید در فایل OLTP_Comparison.xlsx مشاهده کنید.

توجه: جهت تولید نمودارهای تمیزتر، روی چند رکورد از داده‌های outlier تغییراتی داده شده و مقداد میانگین بجای مقدار ۰ قرار داده شده است. اطلاعات دست‌نخورده و اصلی در فایل‌های csv موجودند.


بر خلاف انتظار، میانگین تعداد تراکنش در psql حدود ۷ تا ۹ برابر بیشتر از cockroachdb است ولی میانگین latency در cockroachdb حدوداً ۳ ثانیه کمتر است. احتمال می‌رود این تفاوت در خروجی به دلیل single node بودن سرور و برتری پایگاه‌داده‌های رابطه ای در این نوع سرورها باشد.