بردیا حامدمحسنی

گزارش پروژه

درس پایگاه­­داده پیشرفته

دانشگاه صنعتی امیرکبیر

دی ماه 97

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 نرم­افزار 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:

1. Docker exec –it containerName bash که با آن به کانتینر مورد نظرمان وارد می­شویم.
2. Docker container ls که با آن لیست کانتینر های موجود و وضعیت آن­ها را می­توان بررسی کرد.
3. 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 9 و 15 در PSQL بسیار طولانی شده (بیش از 48 ساعت) و حتی پس از 2 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 نرم­افزار­ها 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 0 قرار داده شده است. اطلاعات دست­نخورده و اصلی در فایل­های csv موجودند.



بر خلاف انتظار، میانگین تعداد تراکنش در psql حدود 7 تا 9 برابر بیشتر از cockroachdb است ولی میانگین latency در cockroachdb حدوداً 3 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.