شرح پروژه:

قسمت OLTP:

در این قسمت از بنچ مارک استاندارد TPC-H استفاده کرده ایم که شامل هشت تیبل است و بیست و دو کوییری را در پلت فرم های POSTGRES , SPARK اجرا کرده ایم نمودار زمان اجرای آن ها به طور جداگانه و مقایسه آن ها در فایل پیوست قرار داده شده است.

در قسمت POSTGRES ابتدا تیبل های مورد نظر را با دستور DROP TABLE IF EXISTS REGION CASCADE;

CREATE TABLE REGION (

R\_REGIONKEY SERIAL PRIMARY KEY,

R\_NAME CHAR(25),

R\_COMMENT VARCHAR(152)

);

که در رفرنس پنج قرارداده شده است ساختیم. پس از آن داده در فایل OLAP\_Benchmark\_data قرار داده شده است .صد گیگ دیتا را در تیبل های مورد نظر با دستور\copy nation from /data/OLAP\_Benchmark\_data/nation.tbl with delimiter ‘|’ کپی کرده ایم که سه تیبلlineitem, order,partsupp به نسبت دیگر تیبل ها زمان زیادی را برای لود کردن سپری کرده است و حتی به حالت idel رفته و بااجرا کردن مجدد آن سریع تر انجام شده است.سپس جداول مورد نظر را با داده آن ساختیم. سپس از پارامتر های شخصی سازی شده که برای هر کوییری به طور جداگانه در منبع یک که قرار داده شده است استفاده کرده ایم و کوییری مورد نظر را با استفاده از QUERY ANALYZE اجرا کرده ایم .همچنین خروجی آن ها در فایل پیوست قرار داده شده است. و نمودار آن را رسم کرده ایم و در فایل پیوست قرار داده ایم.

پارامتر های شخصی سازی شده برای هر کوییری به شرح زیر است:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Q7 | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Query |
| ETHIOPIA  JAPAN | 1996-01-01  0.03  24 | AFRICA  1996-01-01 | 1996-11-01 | MACHINERY  1995-03-25 | 45, EUROPE | 110 | parametrs |
| 241.505 | 1.515 | 6.972 | 104.662 | 18.096 | 9.066 | 0.243 | Planning time(ms) |
| 1037500.303 | 240029.743 | 4710824.964 | 564248.344 | 2509990.864 | 533008.896 | 500599.083 | Executing time(ms) |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Q14 | Q13 | Q12 | Q11 | Q10 | Q9 | Q8 | QUERY |
| 1996-03-01 | %special%packages% | ('MAIL', 'RAIL')  1995-01-01 | ALGERIA  0.0001000000 | 1994-11-01 | orange | AMERICA  ECONOMY ANODIZED STEEL  BRAZIL | parametrs |
| 26.334 | 169.506 | 78.357 | 222.165 |  | 222.165 | 37.459 | Planning time |
| 730548.214 | 775562.670 | 1506730.834 | 11723406.596 |  | 11723406.596 | 829614.850 | Executing Time |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q22 | Q21 | Q20 | Q19 | Q18 | Q17 | Q16 | Q15 | Query |
| ('28', '43', '22', '39', '31', '30', '41') | INDONESIA | forest%  INDIA  1996-01-01 | 7  12  25  Brand#54  Brand#22 Brand#51 | 314 | Brand#13  WRAP PKG | Brand#21  LARGE PLATED%  %Customer%Complaints%  (47, 30, 18, 20, 48, 10, 39, 20) | 1994-05-01 | Parametrs |
| 52.640 | 3.477 | 56.854 | 7.329 | 25.526 | 0.434 | 110.231 | 1.047 | Planning Time |
| 626712.611 | 1849430.305 | 5028110.012 | 584955.038 | 2955627.088 | 2460479.630 | 229349.270 | 1218798.089 | Executing Time |

زمانی که کوییری های نهم و یازدهم را با استفاده از پارامتر های شخصی سازی شده اجرا کرده بودیم مدت زمان طولانی تری نسبت به بقیه کوییری ها برای اجرا سپری کرده اند همانطور که در نمودار در فایل پیوست مشاهده نموده اید.سپس این دو کوییری را که با استفاده از دستور SELECT datname,pid,state,query FROM pg\_stat\_activity مشاهده نموده ایم و متوجه شدیم که به حالت idel رفته اند و آن ها را مجبور شدیم مجددا اجرا کنیم که به نسبت قبلی سریع تر اجرا شده اند اما در کل مدت زمان بیشتری را سپری کرده اند.

در قسمت اسپارک از سه فرمت PARQUET , AVRO,ORC استفاده کرده ایم و کوییری های مورد نظر را اجرا کرده ایم که کوییری های که در قسمت postgres اجرا کرده ایم را باید در سه فرمت spark نیز این بار ران کنیم که برای اینکار ابتدا باید آن هارا به کد پایتون تبدیل کرده و سپس کویری های مورد نظر را اجرا کرده ایم.برای ساختن تیبل ها در فرمت های مختلف ابتدا باید تیبل های مورد نظر را تبدیل به RDDکنیم و سپس RDD را تبدیل بهDataframe وسپس Dataframe را به فرمت های parquet , orc ,avro در آوریم که دستورات ساختن تیبل ها در فرمت های مورد نظر در فایل command پیوست شده است. همچنین نمودارهای آن ها در فایل پیوست قرار داده شده است.

دستوراتی که برای هر فرمت در داکرزده شده است در فایل پیوست آورده شده است.

حال باید به مقایسه آن ها در قسمت OLAP بپردازیم.

**Running the Benchmark**:

این قسمت مربوط به اجرا کردن کوییری ها در پلت فرم های مختلف و پارامتر های شخصی سازی شده که برای هر کوییری به طور جداگانه مشخص شده است.که ما کوییری های یکسان را روی یک دیتاست یکسان در پلت فرم های POSTGRES,SPARK اجرا کرده ایم و برای آن نمودار نیز کشیده ایم.در منبع شش که قرار داده شده است کوییری ها را از نظر runtime وقتی که سایز دیتاست را افزایش میدهد نیز بررسی کرده است اما در پروژه ما سایز دیتاست ثابت است .برای مقایسه از نظرruntime کوییری های اسپارک خیلی کند عمل میکدد و زمان runtime زیادی دارند همانطور که در نمودار های فایل پیوست مشاهده میکنید بنابراین از این نظر postgres خوب عمل نمیکند و runtime زیادی دارد به طوری که برای مقایسه با اسپارک نمودار های spark کاملا زیر نمودارهای postgres قرار داشند و اصلا قابل قیاس با postgres نبودند که در خود postgres کوییری ها runtime های متفاوت داشتند اما دو کوییری نهم و یازدهم به شدت runtime زیادی داشتند و خیلی کند عمل میکردند اما در مورد spark کوییری ها به نسبت و قیاس با postgres بهتر عمل کردند و سریع تر اجرا شده و runtime کمتری نسبت به postgres داشتند که ما سه فرمت parquet , orc,avro را با هم مقایسه کرده ایم و نمودار های آن ها را هم به طور جداگانه و با مقایسه با یکدیگر قرار داده ایم که در خود spark اگر بخواهیم مقایسه انجام دهیم به این صورت که فرمت orc نسبت به دو کوییری دیگر خیلی سریع تر اجرا شده است و نمودار آن کاملا زیر دو نمودار دیگر است همانطور که کوییری ها را اجرا میکردیم متوجه این موضوع شده و نمودار نیز این را ثابت میکند .برای دو فرمت دیگر avro زمان طولانی تری را نسبت به orc,parquet داشت و از نظر runtime کند عمل میکند همانطور که در نمودار مشاهده میکنید نمودار آن کاملا بالای دو فرمت دیگر قرار دارد و فرمت parquet نیز از orc کند تر عمل میکند و runtime بیشتری دارد و از avro کمی بتر عمل میکند و runtime کمتری دارد که به نمودار های فایل پیوست مراجعه شود.

مشخصات سیستم:

Linux**: Linux Kernel Name**

**Linux Kernel Release**: 4.15.0-42-generic

**Linux Kernel Version** : #45-Ubuntu SMP Thu Nov 15 19:32:57 UTC 2018

node5:**namenode**

**Machine Hardware Architecture**: x86\_64

**Operating System information:** GNU/Linux

node5

description: Computer

width: 64 bits

capabilities: smp vsyscall32

\*-core

description: Motherboard

physical id: 0

\*-memory

description: System memory

physical id: 0

size: 39GiB

\*-cpu:0

product: Intel(R) Xeon(R) CPU E5530 @ 2.40GHz

vendor: Intel Corp.

physical id: 1

bus info: cpu@0

size: 2465MHz

width: 64 bits

\*-cpu:1

product: Intel(R) Xeon(R) CPU E5530 @ 2.40GHz

vendor: Intel Corp.

physical id: 2

bus info: cpu@1

size: 2467MHz

width: 64 bits

**cpu:**

Architecture: x86\_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

CPU(s): 16

On-line CPU(s) list: 0-15

Thread(s) per core: 2

Core(s) per socket: 4

Socket(s): 2

NUMA node(s): 2

Vendor ID: GenuineIntel

CPU family: 6

Model: 26

Model name: Intel(R) Xeon(R) CPU E5530 @ 2.40GHz

Stepping: 5

CPU MHz: 1609.904

CPU max MHz: 2401.0000

CPU min MHz: 1600.0000

BogoMIPS: 4799.82

Virtualization: VT-x

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 8192K

NUMA node0 CPU(s): 0-3,8-11

NUMA node1 CPU(s): 4-7,12-15

**-memory**

description: System memory

physical id: 0

size: 39GiB

**Uptime**: 19:42:06 up 42 days, 1:26, 3 users, load average: 4.10, 4.19, 4.12

**Disk usage**:

Filesystem 1K-blocks Used Available Use% Mounted on

udev 20562308 0 20562308 0% /dev

tmpfs 4118584 2180 4116404 1% /run

/dev/sda2 479614640 329861016 125320832 73% /

tmpfs 20592908 0 20592908 0% /dev/shm

tmpfs 5120 0 5120 0% /run/lock

tmpfs 20592908 0 20592908 0% /sys/fs/cgroup

/dev/sdb1 3844607992 1161672120 2487570344 32% /db

**Memory usage:**

total used free shared buff/cache available

Mem: 40220 4972 290 249 34957 34491

Swap: 8191 2835 5356

**Network:**

\*-network:0

description: Ethernet interface

product: 82576 Gigabit Network Connection

vendor: Intel Corporation

physical id: 0

bus info: pci@0000:06:00.0

logical name: enp6s0f0

version: 01

serial: 18:a9:05:3d:51:2a

size: 100Mbit/s

capacity: 1Gbit/s

width: 32 bits

clock: 33MHz

capabilities: bus\_master cap\_list rom ethernet physical tp 10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation

configuration: autonegotiation=on broadcast=yes driver=igb driverversion=5.4.0-k duplex=full firmware=1.7.2 ip=192.168.53.14 latency=0 link=yes multicast=yes port=twisted pair speed=100Mbit/s

resources: irq:42 memory:c0400000-c041ffff memory:c0420000-c043ffff ioport:e880(size=32) memory:c04c0000-c04c3fff memory:c0440000-c045ffff memory:c04c4000-c04e3fff memory:c04e4000-c0503fff

\*-network:1 DISABLED

description: Ethernet interface

product: 82576 Gigabit Network Connection

vendor: Intel Corporation

physical id: 0.1

bus info: pci@0000:06:00.1

logical name: enp6s0f1

version: 01

serial: 18:a9:05:3d:51:2b

capacity: 1Gbit/s

width: 32 bits

clock: 33MHz

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 465.7G 0 disk

├─sda1 8:1 0 1M 0 part

└─sda2 8:2 0 465.7G 0 part /

sdb 8:16 0 3.7T 0 disk

└─sdb1 8:17 0 3.7T 0 part /db

sr0 11:0 1 1024M 0 rom

processor Intel(R) Xeon(R) CPU E5530 @ 2.40GHz

processor Intel(R) Xeon(R) CPU E5530 @ 2.40GHz

memory 39GiB System memory

network 82576 Gigabit Network Connection

network 82576 Gigabit Network Connection

\*-storage

description: RAID bus controller

product: Smart Array G6 controllers

vendor: Hewlett-Packard Company

physical id: 0

bus info: pci@0000:05:00.0

version: 01

width: 64 bits

clock: 33MHz

capabilities: storage bus\_master cap\_list rom

configuration: driver=hpsa latency=0

\*-storage

description: SATA controller

product: 82801JI (ICH10 Family) SATA AHCI Controller

vendor: Intel Corporation

physical id: 1f.2

bus info: pci@0000:00:1f.2

version: 00

width: 32 bits

clock: 66MHz

capabilities: storage ahci\_1.0 bus\_master cap\_list

configuration: driver=ahci latency=0

OLTPBench:

در این قسمت از بنچ مارک استاندارد TPC-C استفاده کرده ایم که شامل نه تیبل است.همه تراکنش ها در TPC-C-یکwarehouse id دارد که یک کلید خارجی به بقیه تیبل ها است.ما در این قسمت از پلت فرم oltpbench برای POSTGRES استفاده کرده ایم. قبل از آن دیتابیس tpcc را در postgres می سازیم سپس ابتدا فایل git را کلون کرده ایم با دستور git clone https://github.com/oltpbenchmark/oltpbench.git بعد آن را با ant کامپایل کرده ایم سپس سراغ فایل کانفیگ رفته ایم و یک سری پارامتر های آن را تغییر داده که شامل dbtype ,dburl,drive است و سپس بنچ مارک را با دو دستور زیر ران میکنیم:

./oltpbenchmark -b tpcc -c config/sample\_tpcc\_config.xml --create=true --load=true

./oltpbenchmark -b tpcc -c config/sample\_tpcc\_config.xml --execute=true -s 5 -o outputfile

با اجرای دستور آخر خروجی در فایل output به دوصورت res که خلاصه شده و csv است قرار داده شده است.

در این قسمت خروجی ما وابسته به پلت فرم oltpbench است که پلت فرم oltpbench را با پارامترهای مختلف در فایل config تغییر داده ایم و به ازای آن بنچ مارک را ران کرده ایم .در پارامتر هایی که قرار داده شده است ابتدا با scalafactor یک و time برابر 7200 و rate برابر 10000 قرار داده ایم و وقتی اجرا کرده ایم به بن بست یا همان deadlock در تاپل(5649,10) در رابطه stock برخورده ایم و سپس سعی کردیم که پارامتر ها را طوری تغییر دهیم که مشکل بن بست را برطرف کنیم که سپس آن را با پارامتر های terminal=10,scalafactor=2,time=600, rate=unlimited قرار داده که باز هم در زمان اجرای بنچ مارک به مشکل بن بست خورده و سعی کرده که پارامتر ها را طوری تغغیر داره که این مشکل بن بست به خاطر یک سری x-lock هست برطرف شود و سپس آن را با پارامتر های scalafactor=2,terminal=10,time=500, rate=unlimited تنظیم کرده که باز هم این مشکل مجددا در اجرای بنچ مارک به وجود آمده است سپس آن را با پارامتر های scalafactor=1,time=300,rate=1000,terminal=10 ست کرده ایم که همچنان این شکل بن بست برقرار بود و مجددا آن را با پارامترهای scalafactor=3 ,terminal=10,time=300,rate=unlimited آن را اجرا کرده ایم که باز هم به مشکل بن بست برخورده ایم در نهایت با آزمون و خطا با پارامتر هایی که برابر با scalafactor=2,terminal=4,time=60,rate=1000 که فایل کانفیگ نیز در فایل پیوست قرار داده شده است تنظیم کرده و توانستیم مشکل بن بست را برطرف کنیم و فایل خروجی را که در فولدر results است را نیز داشته باشیم .حال به ما نیز دو خروجی میدهد یک خروجی res که خلاصه شده اسن و یک خروجی csvکه کامل است با توجه به خروجی csvکه هر پنچ ثانیه یک بار یک رکورد در خروجی به چاپ میرساند ما دو نمودار برای گذردهی و latency رسم کرده ایم که محور x ها برای خروجی پنچ ثانیه یکبار به صورت 5,10,15,20,25,30,35,40,45,…,495 است و محور y ها نیز گذردهی و تاخیر را نمایش میدهد.و نمودار های آن در فایل پیوست قرار داده شده است.

در قسمت cockroachdb که از oltpbench برای تست tpc-c استفاده کرده ایم تنها تغییر فایل کانفیگ و اجرای دستورات بالا برای create tpc-c کافی نبود بلکه باید دستور (COCKROACHDB("org.postgresql.Driver", false, true, true را به مسیر scr/com/oltpbench/types/DatabaseType.java اضافه میکردیم و سپس دستورات را اجرا میکردیم.

برای مقایسه روی tpc-c برای postgres و cockroachdb با توجه به دستوراتی که برای ساخت دیتابیس tpc-cانجام میدادیم همچنین در زمان ران کردن بنچ مارک با فایل کانفیگ xml داریم که:

با توجه به نمودار های throughput وlatency برای postgres و cockroachdb که در فایل پیوست قرار داده شده است و مقایسه مقدار ها در فایل خروجی که مقدار هایtime, throughput,latency و رسم نمودار های آن همچنین درزمانی که دستور لود دیتا را اجرا کرده بودیم و هنگام لود شدن دیتا در تیبل های tpc-c به این نتیجه رسیدیم که postgres در oltpbenchmark کارایی و performance بهتری دارد ومقادیر throughput آن نسبت به cockroachdb بیشتر است و cockroach گذردهی کمتری نسبت به postgres دارد و کند تر عمل میکند به این معنا که نیز latency بیشتری دارد.