CS422 - Project 1

Arnaud Garin - 228275

March 2019

1 Introduction

This paper will answer the problematic raised in question 3.

2 Queries

2.1 Query 1

Query 1 is just a quick equality check in line item

```
SELECT L_ORDERKEY, L_QUANTITY
FROM LINEITEM
WHERE L_QUANTITY == 6
```

2.2 Query 2

Query 2 is a lot like query 1 but this time we look for an inequality

```
SELECT L_ORDERKEY, L_QUANTITY
FROM LINEITEM
WHERE L_QUANTITY > 6
```

2.3 Query 3

Query 3 will start to check the aggregates we start by the classic COUNT

```
SELECT COUNT(*)
FROM LINEITEM
```

2.4 Query 4

Query 4 will check the aggregate AVG

```
SELECT AVG(L_EXTENDEDPRICE)
FROM LINEITEM
WHERE L_QUANTITY >= 20
```

2.5 Query 5

For the last query we will check the merge

```
SELECT L_SHIPMODE
FROM LINEITEM l, ORDERS o
WHERE l.l_orderkey=o.o_orderkey
```

3 Results

3.1 Material

The code was run on a computer with the following characteristics.

- OS: Arch Linux x86 64
- Kernel: 5.0.3-arch1-ARCH
- CPU: Intel i7-2600 (8) @ 3.800GHz
- RAM: 8 GiB
- JDK: java-8-openjdk

3.2 Volcano layout results

The following table shows the results for the Volcano-Operator for row store and PAX-stores with different number of tuples per pages

3.2.1 Results table 1

Layouts			
Queries	NSM-Tuples	PAX-Tuples, 200	PAX-Tuples, 2000
Query1	0.66 ms	0.18 ms	0.19 ms
Query2	0.04 ms	0.11 ms	0.13 ms
Query3	24.37 ms	192 ms	83 ms
Query4	45.47 ms	160 ms	241 ms
Query5	37 min	45 min	42 min

3.3 Columnar and vectorial

The following table presents the results for columnar and vectorial layout. For the columnar early and late materialisation is shown and for vectorial two different vector size are used.

3.3.1 Results table 2

Layouts		
Queries	Columnar early	Vector 5000
Query1	84 min	82 min
Query2	19 min	81 min
Query3	251 ms	>10h
Query4	7175 ms	>10h
Query5	>10h	>10h

4 Insights

4.1 NSM and PAX layouts

Overall the row store and the PAX store perform relatively well. It is important to note that the load() function is way faster for the PAX layout than any other layouts.

The query execution is a bit slow on the join which might be due to a not so optimal implementation of the hashjoin.

It was also found that the size of the PAX page does not seem to yield a faster execution in a consistent fashion.

4.2 Columnar layout

Theoretically speaking the columnar layout works in the sense of giving back the results it is supposed to. Objectively speaking it is not not usable in any way its performances are just too awful.

It apparently comes from the moment when there is an attempt to copy part of the columns composing the column store. The most probable option is that the data is fully copied instead of just passed via reference (which is quite difficult to identify in java).

Oddly enough it yields acceptable performances for aggregates (query 3 and 4) which is probably due to the getStats() function directly implemented in the column, but everything else is just too slow.

The vectored option makes things even worse which confirm that repeated calls to columns content is where the implementation lacks performances.

5 Conclusion

Overall the code works and gives back what it is suppose to. The performances for row store and PAX store are good but everything that has to deal with columnar layout is way too slow. It is hard to pin point where everything goes south but it might come from a confusion with arguments passed by value instead of reference.