Zhechen Su 294180

CS-422

Project1: Task3 Report

Before:

1. My environment is MacOS, 2.9 GHz Intel Core i5, 8 GB memory.
2. Use IntelliJ as IDE run all test several time and get average.
3. Vectorsize = 10 and TuplePerPage = 100

Experiment1 :

SELECT L_SUPPKEY, L_RETURNFLAG, L_LINESTATUS

FROM LINEITEM

WHERE L_SUPPKEY>4508

	ColumnarBlock	ColumnarVector	PaxVolcano	RowVolcano
Time	21s640ms	19s263ms	15s521ms	22s218ms

PaxVolcano runs best. This test is a combination of select and projection. For select, columnar is more fast than row way which can access value directly, and it is hard to get row and select. However, row storage is easier to get column value. So we can see the combination method could run faster than simple way since it has advantage of both storage. And in this case, PaxVolcano is more friendly to memory.

Experiment2 :

SELECT L.L_COMMENT, O.O_SHIPPRIORITY

FROM order O, lineitem L JOIN lineitem

ON (o_orderkey = orderkey)

	ColumnarBlock	ColumnarVector	PaxVolcano	RowVolcano
Time	24s938ms	19s391ms	22s694ms	25s756ms

ColumnarVector runs best. In this test, the system should perform a join and a select. Quite like query before, combination approaches act better than pure ways. But join operation values more and need more time to do, so the performance is mainly depended on who can do join operation faster. And ColumnarVector wins.

Experiment3 :

SELECT MAX(L_DISCOUNT)

FROM lineitem

	ColumnarBlock	ColumnarVector	PaxVolcano	RowVolcano
Time	19s719ms	21s729ms	15s69ms	18s669ms

PaxVolcano runs best. The result is quite confusing, because for aggregate operation system needs to access by column most of time. And that means column storage method should perform better than row. In this case, the query does not use row information, so in my expectation ColumnarBlock should be the best and RowVolcano is the worst. Maybe is an implement issue, but sadly I check for long time and cannot figure out.