Analysis.md 4/11/2023

Theoretical Speed Improvement if we were to reconfigure tables in database

Current config Summary

Currently, our database consists of 7 tables, loosely divided based on product category. This is how the data was configured in the files we downloaded to populate the database. All tables are 15 columns wide. The average file size of each column (in bytes) is shown in the table below.

**Attribute size (bytes)**

market\_place 2

customer\_id 8

review\_id 13

product\_id 10

product\_parent 8

product\_title 69

product\_category 10

star\_rating 4

helpful\_votes 4

total\_votes 4

vine 4

verified\_purchase 4

review\_headline 34

review\_body 332

review\_date 10

average record size 516

By running the command

select bpname,pagesize from syscat.bufferpools

The default pagesize is 4096 bytes so that is what we assume the page size to be. Another simplifying assumption that we will make is that the data for review\_body is stored along with the record. However, the db2 documentation states the CLOB data is not stored along with the records, only the description of the data is stored along with the record. We will ignore this fact to simplify calculations and to exagerate the speed improvement when we reconfigure the database as the reduction in page size due to moving review\_body to a separate table will be dramatic. https://www.ibm.com/docs/en/db2/11.5?topic=tables-space-requirements, https://www.ibm.com/docs/en/db2/11.5? topic=tables-table-page-sizes, https://www.ibm.com/docs/en/db2/11.5?topic=pools-designing-buffer

1 / 8

Analysis.md 4/11/2023

For the calculations, B = 1000, will be used https://www.columbia.edu/sec/acis/db2/db2help/sqld0001.htm. This is the default number created for UNIX-based platforms by version 5 of DB2. We were unable to determine the number of buffer tables for Version 11 that we are using.

For our default database configuration:

records 36961803

record size 516

page size 4096

records/page 7

pages 5280258

New database configuration. In order to reduce the width of each table and combine all the tables, the data would be separated into 3 large tables. a product table, review\_details table, and review table as shown below:

product(product\_id, product\_parent, product\_title, product\_category) sorted by product\_id

reveiw\_details(review\_id, market\_place, star\_rating, helpful\_votes, total\_votes,vine, verified\_purchase, review\_headline, review\_body, review\_date) sorted bt review\_id

reviews(review\_id,customer\_id,product\_id) sorted by review\_id.

Below is information about the above tables. We hold are assumptions made for original attribute sizes.

**table records record size records/page pages**

products 5257914 97 42 125189

review\_details 36961803 411 9 4106867

reviews 36961803 31 132 280014

To compare the two table configurations, We will show a hypothetical I/O cost for several of the queries. We will assumed each query has been optimized by creating appropriate indexes for index-nested loop joins but no index-only scans are allowed.

Query

WITH review\_count as (

SELECT product\_category, sum(r) as reviews FROM (

SELECT product\_category, count(review\_id) as r FROM MCDONELC.apparel

GROUP BY product\_category

UNION ALL

…

UNION ALL

SELECT product\_category, count(review\_id) as r FROM MCDONELC.wireless

GROUP BY product\_category

)

GROUP BY product\_category

),

fstar\_count as (

SELECT product\_category, sum(five) as five\_stars FROM (

SELECT product\_category, count(STAR\_RATING) as five FROM MCDONELC.apparel

2 / 8

Analysis.md 4/11/2023

WHERE STAR\_RATING = 5

GROUP BY product\_category

UNION ALL

…

UNION ALL

SELECT product\_category, count(STAR\_RATING) as five FROM MCDONELC.wireless WHERE STAR\_RATING = 5

GROUP BY product\_category

)

GROUP BY product\_category

)

SELECT review\_count.product\_category, reviews, five\_stars,

DEC(DEC(five\_stars,9,2)/DEC(reviews,9,2)\*100,3,1) as five\_stars\_per\_review FROM review\_count JOIN fstar\_count ON review\_count.product\_category LIKE

fstar\_count.product\_category

ORDER BY five\_stars\_per\_review DESC

LIMIT 10;

The I/O cost of review count

In the original db setup, the steps to perform the review\_count CTE are

1. Load each table

2. Do sorting-based projection for each table

3. Aggregate each table

4. Join each table

5. Sort combined table

6. Aggregate combined table

The I/O cost to load the apparel table and perform the sorting-based projection is calculated by

$$5874693 \space records \times { 1 \space page \over 7 \space records} +5874693 \space records \times {1 \space page \over 240 \space records} = 863720$$

We then must sort the projection and write sorted result. The I/O cost of sorting + writing results is:

$$863720 + 2 \times 24478 \times (\lceil log\_{999} \lceil {24478 \over 1000} \rceil \rceil - 1) + 2 \times 24478= 863720 + 2 \times 24478 = 912676 $$

The cost of aggregating on product\_category is the cost of a scan plus the cost of writing the results. There is 1 distinct product categories in the APPAREL table. The I/O cost is $24478 + 1 = 24479$.

The table below show the number of distinct categories per subtable. Only the multilingual table has multiple product categories.

**Table Distinct Categories**

APPAREL 1

APPLIANCES 1

AUTOMOTIVE 1

MULTILINGUAL 38

MUSIC 1

3 / 8

Analysis.md 4/11/2023

**Table Distinct Categories**

PC 1

WIRELESS 1

The following calculation can be performed on all the original subtables to obtain the results below.

**Tables Records PagesPages after Projection**

**Sorting-Based Projection Cost**

**Aggregation Cost**

APPAREL 5874693 839242 24478 912676 24479 APPLIANCES 96792 13828 404 14232 405 AUTOMOTIVE 3497760 499680 14574 543402 14575 MULTILINGUAL 6882248 983179 28677 1069210 28678 MUSIC 4722502 674644 19678 733678 19679 PC 6902673 986097 28762 1072383 28763 WIRELESS 8985135 1283591 37439 1395908 37440 Total: 5280261 154012 5741489 154019

When we union all the tables, we can fit all data in buffer pages so the cost is reading each subtable, sorting them by product category, computing aggregations, and then writing result. There are 39 distinct product categories in the dataset so each once aggregated we have 39 records which fit on one page. The I/O cost is

$$7+1 = 8$$

The total cost of the review\_count CTE is

$$5741489+154019+8 = 5,895,516$$

The I/O cost of fstar count

In the original db setup, the steps to perform the five\_star\_count CTE are

1. Load each table

2. Use sorting-based projection on each table

3. Apply WHERE clause

4. Aggregate each table

5. Join each table

The cost of generating the second CTE five\_star\_count is as follows.

The I/O cost to load the apparel table and perform the sorting based projection of product\_category and star\_rating is

$$5874693 \space records \times { 1 \space page \over 7 \space records} +5874693 \space records \times {1 \space page \over 292 \space records} = 859361$$

We then must sort the projection and write sorted result. The I/O cost of sorting + writing results is:

$$859361 + 2 \times 20119 \times (\lceil log\_{999} \lceil {20119 \over 1000} \rceil \rceil - 1) + 2 \times 20119 = 859361 + 2 \times 20119 = 899599 $$

4 / 8

Analysis.md 4/11/2023

Applying where STAR\_RATING = 5 is the cost of scanning the projection and writing the result. An index here slows down this operation because there are more applicable records than pages. I/O cost is

$$ 20119 + 5874693 \space records \times {1 \over 5} \times {1 \space page \over 292 \space records} = 20119 + 4024 = 24143$$

Since the table was sorted in the sorting-based projection phase, we do not need to sort again.

The cost of aggregating on product\_category is the cost of a scan plus the cost of writing the results. There is 1 distinct product category in the APPAREL table. The I/O cost is $4024 + 1 = 4025$.

The following calculation can be performed on all the original subtables to obtain the results below.

**Tables Records**

**Records**

**where**

**star\_rating = 5**

**Pages**

**Pages**

**after**

**Projection**

**Sorting Based**

**Projection Cost**

**Where cost**

**Pages after**

**Where**

**Aggregation Cost**

APPAREL 5874693 1174939 839242 20119 899599 24143 4024 4025 APPLIANCES 96792 19359 13828 332 14160 399 67 68 AUTOMOTIVE 3497760 699552 499680 11979 535617 14375 2396 2397 MULTILINGUAL 6882248 1376450 983179 23570 1053889 28284 4714 4715 MUSIC 4722502 944501 674644 16173 723163 19408 3235 3236 PC 6902673 1380535 986097 23640 1057017 28368 4728 4729 WIRELESS 8985135 1797027 1283591 30772 1375907 36927 6155 6156 Total: 126585 5659352 151904 25319 25326

As before, when we union all the tables, we can fit all data in buffer pages so the cost is reading each subtable, we can sort them by product category and compute aggregations while holding them in memory. There are 39 distinct product categories in the dataset so once each is aggregated we have 39 records which fit on one page. The I/O cost is

$$7+1 = 8$$

The total cost of the fstar\_count CTE is

$$5659362+151904+25326+8 = 5,836,600$$

To join the two CTE's, the I/O cost is reading each CTE, joining them, aggregating, and outputting the results. Luckily, both tables fit in memory so the only I/O cost is reading each of them. The I/O cost is

$$1+1 = 2$$

The total cost is therefore

$$5,895,516+5,836,600+2 = 11,732,118$$

Cost of the query with new database configuration

With the new database configuration, the query to get the same results would be written as:

5 / 8

Analysis.md 4/11/2023

WITH review\_count AS (

SELECT product\_category, count(review\_id) as reviews FROM

products p JOIN reviews r ON p.product\_id = r.product\_id

GROUP BY product\_category

),

fstar\_count AS (

SELECT products\_category, count(star\_rating) as five\_stars FROM

products p JOIN reviews r ON p.product\_id = r.product\_id

JOIN review\_details d ON r.review\_id = d.review\_id

WHERE star\_rating = 5

GROUP BY product\_category

)

SELECT review\_count.product\_category, reviews, five\_stars,

DEC(DEC(five\_stars,9,2)/DEC(reviews,9,2)\*100,3,1) as five\_stars\_per\_review FROM review\_count JOIN fstar\_count ON review\_count.product\_category LIKE

fstar\_count.product\_category

ORDER BY five\_stars\_per\_review DESC

LIMIT 10;

The I/O cost of review count

In the improved database setup, the steps to perform the review\_count CTE are

1. Sort reveiws table by product\_id

2. Join products to reveiws table on product\_id

3. Perform sorting-based projection on product\_category, review\_id

4. Aggregate the table (table is already sorted from the projection)

5. Write results to memory

Recall that the database setup data

**table records record size records/page pages**

products 5257914 97 42 125189

reviews 36961803 31 132 280014

Since both the reviews and review details table are sorted by review\_id, the cost of joining them is reading the pages in each and then writing the result. I/O cost is

The cost to sort the reveiws table by product\_id is

$$2 \times 280014 \times (1 + \lceil log\_{999} \lceil 280014/1000 \rceil \rceil) = 1,120,056$$ The I/O cost of the join by sort-merge join is

$$1,120,056 + 125189 + 280014 = 1,525,259$$

The number of records in the joined table is $280014$ since product\_id is unique in the products table. The pages in the joined table is

**table records record size records/page pages**

products JOIN reveiws 36961803 31+97-10=118 34 1087111

6 / 8

Analysis.md 4/11/2023

The number of records in the projected table is the same as the reviews table. The number of pages is 1087111. These pages need to be written to memory between steps. The I/O cost is

I/O cost = $1087111$

The I/O cost of getting the projection of product\_category and review\_id is calculated as below. The number of pages of results is:

$$1087111 \times {23 \over 118} = 211895$$

The I/O cost of sorting-based projection is

$$1087111+2 \times 211895 + 2 \times 211895 \times (\lceil log\_{999} \lceil {211895 \over 1000} \rceil \rceil - 1) + 211895 = 1722796$$

Since the table is already sorted by product\_category, the cost of the aggregation is reading all pages and writing results. There are 39 distinct product categories so we will only have one page of results. The I/O cost is

$$211895 + 1 = 211896$$

Total I/O cost of review\_count CTE is

$$ 1,525,259 + 1087111 + 1722796 + 211896 = 4547062$$

The I/O cost of fstar\_count

In the improved database setup, the steps to perform the fstar\_count CTE are

1. Perform a projection of review\_details table on reveiw\_id and star\_rating

2. Scan table and only keep records where star\_rating = 5

3. Join with reviews on review\_id using sorting-based join since both are sorted by reveiw\_id. 4. Join result with products table on product\_id

5. Perform sorting-based projection on product\_category and star\_rating

6. Aggregate on product\_category

7. Write results to memory

Recall the database setup

**table records record size records/page pages**

products 5257914 97 42 125189

review\_details 36961803 411 9 4106867

reviews 36961803 31 132 280014

I/O cost of performing projection of review\_details on review\_id and star\_rating. Since review\_id is unique we do not need to sort to eliminate duplicates.

$$4106867+36961803 \space records \times {1 page \over 241 \space records} = 4106867+153369 = 4260236$$ Now we will scan the table and only keep the records where star\_rating = 5. The I/O cost is $$153369 + 153369 \times {1 \over 6} = 153369 + 25562 = 178931$$

The cost of joining reveiws on review\_id is the cost of scanning both tables and writing the result since they are already sorted by review\_id. The result will have the same number of results as in the projected review table. The number of

7 / 8

Analysis.md 4/11/2023

records is $36961803/6 = 6160301$

$$280014+25562 + 6160301 \space records \times {1 \space page \over 117}=305576 + 52653 = 358229$$ Next, we must sort this table on product\_id so it can be joined with products table. The I/O cost is $$2 \times 52653 \times (1 + \lceil log\_{999} \lceil 52653/1000 \rceil \rceil) = 210612$$

The I/O cost to join this table with the products table is the cost of scanning both of them. We also need to write the result to memory. There will be the same number of records (since product\_id is unique in products table).

The number of resulting pages is

$$6160301 \times {1 \over {4096 \over 114}} = 176009 \space pages$$

I/O cost of join is

$$52653+125189+176009 = 353861$$

Now, we need to perform a projection on this table to get product\_category and star\_rating. The pages of results will be $$6160301 \times {1 \over {4096 \over 14}} = 21025 \space pages$$ The I/O cost is

$$176009+2 \times 21025 + 2 \times 21025 \times (\lceil log\_{999} \lceil {21025 \over 1000} \rceil \rceil - 1) + 21025 = 239084$$

Aggregating on product\_category is the cost of scanning the table. As before the results will fit on one page. $$21025+1 = 21026$$

Total I/O cost of fstar\_count CTE is

$$4260236+178931+358229+210612+353861+239084+21026 = 5,621,979$$

To join the two CTE's, the I/O cost is reading each CTE, joining them, aggregating, and outputting the results. Luckily, both tables fit in memory so the only I/O cost is reading each of them. The I/O cost is

$$1+1 = 2$$

The total cost is therefore

$$4,547,062+5,621,979+2 = 10,169,043$$

Conclusion

The difference in I/O cost was between the two database configurations for this query was $$11,732,118-10,169,043=1,563,075$$

The new configuration results in a 13% improvement in I/O cost. The margin off improvement would depend on the query, the new database would likely require less I/O's. For instance, any queries that only need one of the table information would be substantially faster in the improved database configuration since no join operations would be needed.

8 / 8