Couponsmule Audit Report

## 1.Server and software

OS Debian 10 Buster

CPU 2x CPU E5-2660 (Sandy 2.2Ghz - boost to 3 Ghz) - 16 C / 32 T in total

RAM: 256GB, 160GB free

PHP 7.3

Since there is plenty of RAM and cpu cores, I don’t see anything to address in short term.

Manticore 3.4.2

* Engine runs manually - should be managed by a service manager (systemd)
* Should upgrade to latest version

## 2. Manticore Configuration

Indexer mem\_limit to 64m - this value is a bit low, can be increased to speed up indexing

xmlpipe\_fixup\_utf8 = 1 - in sources, not needed, this is an option for XML pipe sources, not MySQL

Searchd:

* Listeners on all interfaces (potential security risk if machine is not firewalled to external)
* Binlog\_data - not need if no update attributes is made or RT indexes are used
* Max\_children, workers=threads , mva\_updates\_pool - deprecated in latest releases

## 

## 3. Manticore statistics

* Not much to say here, the Manticore instance doesn’t have many queries executed and average query wall time is 48ms.

## 4.Indexes

### Products\_index

* morphology = stem\_en
* expand\_keywords = 1
* min\_infix\_len = 3, html\_strip = 1

Notes:

* stem can be replace with lemmatizer, it’s a morphology processing that uses a dictionary, stemmer only use some algorithms. Stem will extract a root from the words. In some cases the root is the same for words with different meanings (this also means it can provide more results).
* Expand keywords can become a burden in future. While it’s “nice” because will auto-expand words to their possible expansions, as the data will grow there will be more effort (CPU mostly) for getting the hitlists of the expansions.

mysql> describe products\_index;

+---------------+--------+------------+

| Field | Type | Properties |

+---------------+--------+------------+

| id | bigint | |

| gtin | field | indexed |

| mpn | field | indexed |

| item\_group\_id | field | indexed |

| imageurl | field | indexed |

| name | field | indexed |

| category\_name | field | indexed |

| brand\_name | field | indexed |

| other\_offers | string | |

| category\_id | uint | |

| brand | uint | |

| gtin | string | |

| mpn | string | |

| name | string | |

| price | float | |

| category\_name | string | |

| brand\_name | string | |

+---------------+--------+------------+

17 rows in set (0.00 sec)

indexed\_documents | 17194064

Typical returned row:

id: 247

other\_offers: 247,1166365,17936393,24394301

category\_id: 4099

brand: 5

gtin: 010343876446

mpn: 125,T125220,T125220S

name: Epson ,Ink Cartridge

price: 14.090000

category\_name: Toner & Inkjet Cartridges

brand\_name: Epson

20 rows in set (1.66 sec)

Other\_offers seems to be an array of numbers -> could be MVA, need to know how this is used

MPN,GTIN could be JSON For searching operations they could be used as fulltext fields (as they are now defined). If you want to perform grouping, they will need to be put in a structure - JSON field. This could be used also for sorting, but I suspect it won’t be the case for them. There is the question how these two will be used in searches.

Size on disk 78.1G

As Sergey said, the current sql\_query needs a redesign, from the issue you had, it creates a temporary table that uses a lot of disk (most likely because of the GROUP BY). In manticore there is mechanism to get the data in batches.

This will look like

sql\_query\_range= SELECT MIN(id), MAX(id) from cte\_image

sql\_range\_step = 10000

Sql\_query WITH ….. FROM cte\_image WHERE cte\_image.id >= $start and cte\_image.id < $end ….

The $start and $end will be replaced by indexer with steps (1 and 10000, 10001 and 20000, 20001 and 30000 etc.). There are no optimal values, needs to be tested.

### Categories\_index

* Use min\_infix\_len =2 , expand\_keywords, blending html stripping, stopwords (with unstemmed=1)

This table is small in size and not sure how exactly is used

mysql> describe categories\_index;

+--------------------+--------+------------+

| Field | Type | Properties |

+--------------------+--------+------------+

| id | bigint | |

| name | field | indexed |

| google\_category\_id | field | indexed |

| search\_by | field | indexed |

| description | field | indexed |

| name | string | |

| search\_by | string | |

+--------------------+--------+------------+

7 rows in set (0.00 sec)

Typical returned row:

id: 5582

name: Paintball Hoppers

search\_by:

Size on disk 768k

# 5.Issues initial reported

**1. 2.**

On 2. Example I see the following queries are executed:

8:59:34.533 2021 conn 1543 real 0.000 wall 0.000 found 1 \*/ SELECT MAX(price) as max\_price,MIN(price) as min\_price FROM products\_index WHERE MATCH('1v32048') LIMIT 0,1;

/\* Tue Feb 16 18:59:34.535 2021 conn 1543 real 0.000 wall 0.000 found 1 \*/ SELECT category\_id,category\_name,COUNT(\*) as cnt FROM products\_index WHERE MATCH('"1v32048"~10') GROUP BY category\_id ORDER BY cnt DESC LIMIT 0,300;

/\* Tue Feb 16 18:59:34.535 2021 conn 1543 real 0.000 wall 0.000 found 1 \*/ SELECT brand,brand\_name,COUNT(\*) as cnt FROM products\_index WHERE MATCH('"1v32048"~10') GROUP BY brand ORDER BY cnt DESC LIMIT 0,300;

/\* Tue Feb 16 18:59:34.535 2021 conn 1543 real 0.000 wall 0.000 found 1 \*/ SELECT COUNT(\*) as cnt,INTERVAL(price,9,18,27,36,45) as price\_segment FROM products\_index WHERE MATCH('"1v32048"~10') GROUP BY price\_segment ORDER BY price\_segment ASC LIMIT 0,10;

/\* Tue Feb 16 18:59:34.535 2021 conn 1543 real 0.000 wall 0.000 found 6 \*/ SELECT \*,WEIGHT() as w FROM products\_index WHERE MATCH('"1v32048"~10') ORDER BY w DESC LIMIT 0,30;

Last query is the one showing the results. **There is no grouping.**

This is more a problem of the data source rather than a searching one. The engine can group but it needs to know on what. And for that it needs to be established which taxonomy or taxonomies are the measure of **common**. For the current data as it is I need a grouping by name and brand (brand because there might be products with same name from different brands). Better than name would be an identificator that is common to all those products (barcode?). From performance perspective the best would be to do these groupings on integer attributes (right now won’t be much of a problem, but at some level of traffic it will be noticeable).

Grouping on MPN or GTIN has no use on these cases since they are different and we need something common. It might make sense to have MPN and GTIN in some different structures (like JSON) if we want to perform searches on them - maybe even grouping if there is the case of having product duplicated among multiple documents (rows).

Another thing here: the left bar facets I see they run separate from main query, they could be set as facets:

SELECT \*,WEIGHT() as w FROM products\_index WHERE MATCH('mens belt') ORDER BY w DESC LIMIT 0,30 FACET category\_id,category\_name FACET brand,brand\_name FACET INTERVAL(price,9,18,27,36,45) as price\_segment;

The select with facet execute the match finding only once and all groups (facets are groups also) are applied on the result set of the match. Running separate queries means network trips, multiple running of matching etc.

If you want the segments dynamic, you’d still need to run the query with min/max price as right now dynamic facet segmentation is not implemented (it is possible).

One thing to notice is that the counts are counts of the rows - this might look weird as the facet count will show ‘6’ but the search result will show one product.

**3.** This is a source problem. In the index we have:

mysql> SELECT \*,WEIGHT() as w FROM products\_index WHERE MATCH('stetson alamo') ORDER BY w DESC LIMIT 0,30;

+----------+--------------+-------------+-------+------+---------------+---------------------------------------+------------+---------------+------------+------+

| id | other\_offers | category\_id | brand | gtin | mpn | name | price | category\_name | brand\_name | w |

+----------+--------------+-------------+-------+------+---------------+---------------------------------------+------------+---------------+------------+------+

| 27022562 | 27022562 | 0 | 19906 | 0 | SSALMO-304081 | Stetson Alamo - (8X) Straw Cowboy Hat | 109.980003 | Undefined | Stetson | 3610 |

+----------+--------------+-------------+-------+------+---------------+---------------------------------------+------------+---------------+------------+------+

1 row in set (0.01 sec)

Nothing to do from the engine since we get Undefined for the category name. The engine shows what it has, it has no knowledge about any external data/source.

**4.** CALL SUGGEST can be used. This function returns closest word. However :

mysql> call suggest('carrhart','products\_index');

+----------+----------+------+

| suggest | distance | docs |

+----------+----------+------+

| carhart | 1 | 18 |

| carhartt | 2 | 7138 |

| carrara | 2 | 1228 |

| earhart | 2 | 213 |

| barnhart | 2 | 42 |

+----------+----------+------+

5 rows in set (0.00 sec)

The closest word is not the one wanted because ‘carhart’ exists (it’s an author name).

It’s not what we want. Actually in this particular case, it will work because you have expand\_keywords and ‘carhart’ will be expanded to ‘carhartt’, but in many cases the difference between the suggestions will be inside the word, not at the edges.

What is needed to do:

* Run the original query
* If provides zero results, check the keywords info from META. This will tell ‘carrhart’ has zero hits
* Run CALL SUGGEST, remake the query.
* If new query still gets zero results, use 2nd suggested term, run the query again, and so on

**5.** All product names could be dumped in a single fulltext only field (not name). Searching on this field should get the matches. There is a potential issue for some searches to cross words between names (if those are different) and give some false positives matches - something to watch, but I’d expect to not have names that different to get into this situation.