Manticore Search adventures

This is my page of notes after playing with Manticore Search to adopt it.

Manticore Search is an open-source **search engine designed specifically for search, including full-text search**, with focus on low latency and high throughput. It was born in 2017 as a continuation of the famous *Sphinx* Search engine.

Things I like:

- SQL-first, you can connect to the server using just a MySQL/MariaDB client or your mysql connection library of choice, from any language.
  - Default port: 9306 instead of 3306 (default for mysql)
- Official PHP interface (complete HTTP API integration via cURL) for index maintenance, etc. (Searches could be done from mysql or API)
- Real Time indexes that allow instant updates
  - Attaching a plain index to a real-time index: A plain -static- index can be converted into a real-time index or added to an existing real-time index.
- Supports Main+Delta schema: There's a frequent situation when the total dataset is too big to be reindexed from scratch often, but the amount of new records is rather small. Example: a forum with a 1,000,000 archived posts, but only 1,000 new posts per day. In this case, “live” (almost real time) index updates could be implemented using so called “main+delta” scheme.
- Fast geospatial search
- You could go for distributed architecture for faster indexing and searching over petabytes of data

Other notes

- A confusing concept to understand is how *searchd* is run in “RT mode” OR “Plain mode” VS. the index types (RT index and Plain index also). **RT mode is <u>required</u> if you want to enable *replication*** and does <u>NOT</u> allow to create Plain indexes from config (RT mode is set by setting `data_dir` in config, which is to say “RT mode”).

  Basically:
  - REAL-TIME MODE **requires** no index definition in the configuration file and having a *data_dir* directive in searchd section. Index files are stored inside this `data_dir`. <u>Replication is available only in this mode.</u>
  - PLAIN MODE allows to specify index schema in config which will be read on Manticore start and created if missing. This mode is especially useful for plain indexes that need to be built from an external storage. Dropping indexes is only possible by removing them from the configuration file or by removing the path setting and sending a HUP signal to the server or restarting it.
    You can still use REAL-TIME INDEX (RT indexes) in this Plain Mode since it supports ALL index types.

Windows

Extract the zip, then edit the conf. file:

manticore.conf.in

```


# holy crap

common {
    plugin_dir = /usr/local/manticore/lib
}

searchd {
    listen = 127.0.0.1:9312
    listen = 127.0.0.1:9306:mysql
#   listen = 127.0.0.1:9308:http # http(s) port can be the same of binary
protocol port (9312)
    log = E:/manticore36/log/searchd.log
    query_log = E:/manticore36/log/query.log
    pid_file = E:/manticore36/log/searchd.pid
#   PLAIN MODE is enabled by omitting "data_dir" (permits ALL index types,
RT and Plain)
#   RT-MODE is required only if you need to enable REPLICATION
#   data_dir = E:/manticore36/data-rtmode # warning, data_dir *enables RT
MODE* and does NOT allow index definitions at this config. (plain indexes)
    query_log_format = sphinxql
}
```

Let's install as a service:

```
E:\Manticore\bin\searchd --install --config E:\Manticore\manticore.conf.in -
-servicename Manticore
```

Manticore can be started and stopped from the Services Control Panel or manually from the command line:

```
sc.exe start Manticore
```

```
sc.exe stop Manticore
```

If you don't install Manticore as Windows service, you can start it from the command line:

```
.\bin\searchd -c manticore.conf.in
```

To ensure a fast connection, use `127.0.0.1` and **not** `localhost` which can be poorly resolved:

```
mysql -P9306 -h127.0.0.1
```

Scripted configuration

Manticore configuration supports shebang syntax, meaning that the configuration can be written in a programming language and interpreted at loading, allowing dynamic settings.

For example, indexes can be generated by querying a database table, various settings can be modified depending on external factors or external files can be included (which contain indexes and/sources).

The configuration file is parsed by declared declared interpreter and the output is used as the actual configuration. This is happening each time the configuration is read (not only at searchd startup).

This facility is not available on Windows platform.

In the following example, we are using PHP to create multiple indexes with different name and we also scan a specific folder for file containing extra declarations of indexes.

`manticore.conf.in`

```
#!/usr/bin/php
...
<?php for ($i=1; $i<=6; $i++) { ?>
index test_<?=$i?> {
  type = rt
  path = /var/lib/manticore/data/test_<?=$i?>
  rt_field = subject
  ...
 }
 <?php } ?>
 ...

 <?php
 $confd_folder='/etc/manticore.conf.d/';
 $files = scandir($confd_folder);
 foreach($files as $file)
 {
         if(($file == '.') || ($file =='..'))
         {} else {
                 $fp = new SplFileInfo($confd_folder.$file);
                 if('conf' == $fp->getExtension()){
                         include ($confd_folder.$file);
                 }
         }
 }
```

Comments

The configuration file supports comments, with # character used as start comment section. The comment character can be present at the start of the line or inline.

Extra care should be considered when using # in character tokenization settings as everything after it will not be taken into consideration. To avoid this, use # UTF-8 which is U+23.

# can also be escaped using \. Escaping is required if # is present in database credential in source declarations.