В CAP пытался доказать, что мне не так важна консистентность, и моя система AP, заявлял, что буду реализовывать на MongoDB, так как думал, что драйвер Cassandra на C++ может не стать. Но установить, и настроить получилось, поэтому реализовал систему с использованием Cassandra(подходил, спрашивал у Вас, можно ли поменять свой выбор), так как лучше подходит под наши нужды, и не придется реализовывать вспомогательных запросов для поддержки консистентности при разрывах связи между нодами.

Схему таблиц, можно увидеть в create.sql, для удобства приведу здесь:

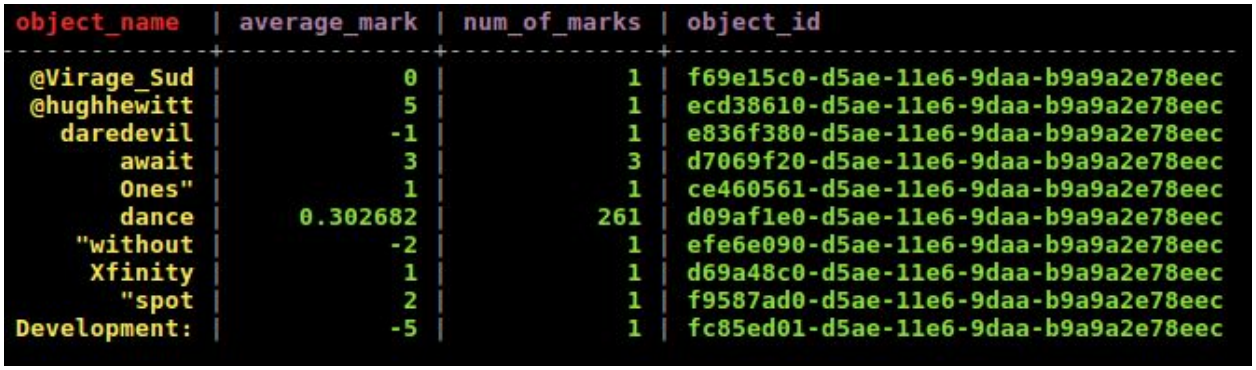
```
create table tweets (
    tweet_id timeuuid,
    tweet_text text,
    object_id timeuuid,
    object_name text,
    object_mark float,
    createdts timestamp,
    primary key (object_id)
);

create table objects (
    object_id timeuuid,
    object_name text,
    average_mark float,
    num_of_marks int,
    primary key (object_name)
);
```

Пример данных в каждой из таблиц:

- tweets:

```
 object_id                            | createdts                       | object_mark | object_name
 | tweet_id                            | tweet_text
--------------------------------------+---------------------------------+-------------+------------------
+--------------------------------------+-------------------------------------------------------------------
-------------------------------------------------------
 887329c6-d5e3-11e6-9b18-ffc75f177523 | 2017-01-08 20:46:30.492000+0000 |          -5 |          Balance
 | 887329c7-d5e3-11e6-9b18-ffc75f177523 |
             Future Islands Balance httpstcoEGcimkU7c8 S
 74707b83-d5e3-11e6-9b18-ffc75f177523 | 2017-01-08 20:45:56.920000+0000 |           3 |         yourself
 | 74707b84-d5e3-11e6-9b18-ffc75f177523 |                                                          Beli
eve in yourself PanMovie levizanemiller httptcoj2W09Kuu31
 7c149e36-d5e2-11e6-9b18-ffc75f177523 | 2017-01-08 20:39:00.243000+0000 |           1 |             with
 | 7c149e37-d5e2-11e6-9b18-ffc75f177523 | tintinelijah Hello Kristine Your local Nissan dealer can best as
sist with this Please visit httptcoxmm5C7Jrd1 for options
 3366dad3-d5e3-11e6-9b18-ffc75f177523 | 2017-01-08 20:44:07.805000+0000 |           1 | contentmarketing
 | 3366dad4-d5e3-11e6-9b18-ffc75f177523 |   I just saw a 61 increase in conversions using Outbrains Vertic
al Targeting httptcoBmTdZjzu2A contentmarketing marketing
 a2eaa273-d5e2-11e6-9b18-ffc75f177523 | 2017-01-08 20:40:05.399000+0000 |           2 |            thanks
 | a2eaa274-d5e2-11e6-9b18-ffc75f177523 |
                   UKArikaKaneFans thanks very much Luv
```

- objects:



object_name	average_mark	num_of_marks	object_id
@Virage_Sud	0	1	f69e15c0-d5ae-11e6-9daa-b9a9a2e78eec
@hughhewitt	5	1	ecd38610-d5ae-11e6-9daa-b9a9a2e78eec
daredevil	-1	1	e836f380-d5ae-11e6-9daa-b9a9a2e78eec
await	3	3	d7069f20-d5ae-11e6-9daa-b9a9a2e78eec
Ones"	1	1	ce460561-d5ae-11e6-9daa-b9a9a2e78eec
dance	0.302682	261	d09af1e0-d5ae-11e6-9daa-b9a9a2e78eec
"without	-2	1	efe6e090-d5ae-11e6-9daa-b9a9a2e78eec
Xfinity	1	1	d69a48c0-d5ae-11e6-9daa-b9a9a2e78eec
"spot	2	1	f9587ad0-d5ae-11e6-9daa-b9a9a2e78eec
Development:	-5	1	fc85ed01-d5ae-11e6-9daa-b9a9a2e78eec

Итак, что реализовано:

- GET запрос типа http://192.168.56.20/objects?object_name=iphone получаем response:


```
{
  "averageMark" : 0.8000000119209290,
  "id" : 100,
  "name" : "iphone"
}
```


- POST запрос типа **curl -X POST -d '{"text" : "hello world"}' http://192.168.56.20/tweets** получаем response:


```
{
  "id" : 100,
  "mark" : -3,
  "objects" : [
    {
      "id" : 15,
      "mark" : -4.0,
      "name" : "hello"
    }
  ],
  "text" : "hello world"
}
```


С этими двумя запросами(добавление нового твита, получение средней оценки тональности для какого либо объекта и работал). Таблица objects заполнена на около 40 тыс. объектов(учитывая, что в англ. языке всего около 300-400 тыс. слов, посчитал, что это вполне удовлетворительно). Соответственно при новом insert для конкретного объекта пробегаем по objects и смотрим, если такой уже есть, то обновляем с учетом новой оценки, если нет - добавляем новый. Inserts здесь блокирующие(хоть потом заметил, что драйвер в принципе поддерживал асинхронную вставку, но тогда бы пришлось значительно усложнять код).

Тестировал используя поочередно GET и POST запросы в соотношении 50/50(предполагается, что будет поступать предположительно одинаковое количество, как новых твитов, так и запросов на тональность какого-либо объекта). Их подготовил в файле ammo.txt(сгенерировал с использованием python скрипта).

К сожалению, мониторинг так и не стал(пробовал несколько раз с нуля переустаналивать как сам yandex-tank, так и telegraf, который требуется для мониторинга).

Путем экспериментов выбрал threads=200, queue=5000, в sentiment.conf.

Добился rps 450(при больших вываливаются сетевые ошибки), ссылка на тест: https://overload.yandex.net/online/6323

При данном уровне видим, что 99% запросов покрывается 0,7 секунды, да и пытался мониторить нагрузку CPU - все ядра загружены в среднем на половину.

Вся основная логика в папке cpp-driver в Sentiment.cpp и CassandraManager.cpp.