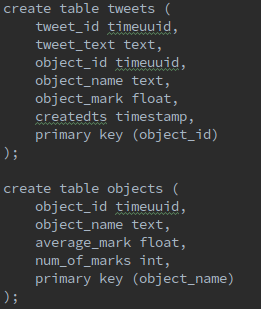
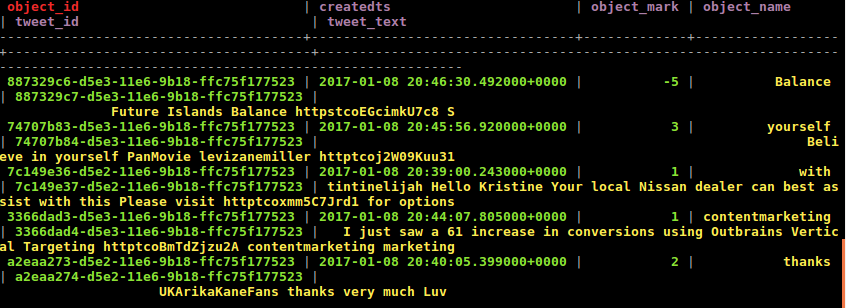
В CAP пытался доказать, что мне не так важна консистентность, и моя система AP, заявлял, что буду реализовывать на MongoDB, так как думал, что драйвер Cassandra на C++ может не стать. Но установить, и настроить получилось, поэтому реализовал систему с использованием Cassandra(подходил, спрашивал у Вас, 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:

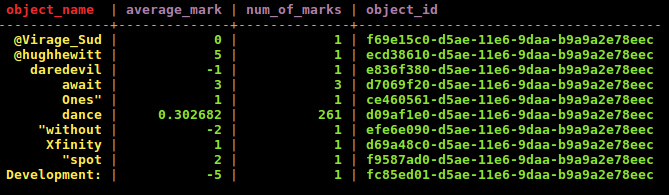


Пример данных в каждой из таблиц:

* tweets:



* objects:



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**](http://192.168.56.20/tweets)получаем response:

{

"id" : 100,

"mark" : -3,

"objects" : [

{

"id" : 15,

"mark" : -4.0,

"name" : "hello"

}

],

"text" : "hello world"

}

C 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. Insertы 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 c 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](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.