Зависимости:
boost
rabbitmq-c
redox (https://github.com/hmartiro/redox.git)
hiredis
libev
protobuf

Сервер хранения данных: redis 4.0.1
Сервер приема/отправки сообщений: rebbitmq 3.6.9
Формат сообщений: protobuf

Описание работы:
Часть на C++ состоит из трех взаимосвязанных объектов и вспомогательных лямбд. Два подключения: одно к серверу сообщений(rebbitmq), другое к серверу хранения данных (redis). Логика по сортировке и обработке данных реализована в redis'е штатными средствами и хранимыми процедурами на lua. Также на C++ реализован генератор запросов информации о пользователе периодически или при подключении пользователя. Генератор запросов подписан на сообщения подключения/отключения пользователей сервера сообщений(rabbitmq).
Подключение к серверу сообщений через лямбду подписано на сигнал о формировании информации о пользователе подключения к серверу хранения данных.
Схема взаимодействий приведена ниже.

Описания способа хранения данных:
Для каждого пользователя хранится текущий оборот прибыльных сделок (общий за 7 дней), текущий день года, история оборотов прибыльных сделок за последние 7 дней (по каждому дню отдельно), имя пользователя.
При наступлении нового дня года оборот за самый старый день истории (7й) вычитается из общего оборота за последние 7 дней, счетчики оборотов по дням сдвигаются на 1. Данный алгоритм реализован в хранимой процедуре.
Сортировка по оборотам и выборка соседних позиций и первых 10 позиций в рейтинге осуществляется штатными средствами redis, т.к. для этого необходимо выполнить несколько запросов реализация также в виде хранимой процедуры.

Что можно сделать лучше:
Запрос на top10 можно вынести из запроса информации о пользователе для его ускорения, сейчас он выполняется каждый раз.
В нескольких местах оставлены TODO, эти места узкие или теоретически могут быть узкими.
Не реализован ежесуточный проход по БД для обновления данных пользователей не совершавших сделок в течение суток, есть реализация в виде хранимой процедуры но не сделан вызов в коде.
Не проводился анализ при кол-ве пользователей более 60млн, т.к. на ноутбуке не достаточно памяти.
Нет нормального вывода, используется cout без отметок времени и уровней логирования.

Перспективы:
Т.к. в качестве хранилища выбран redis обновление данных и формирование информации о пользователях можно разнести, также возможна репликация redis’а для увеличения надежности и уменьшения времени отклика.

Примерные показатели:
При 1млн записей redis занимает в памяти около 2-3Гб, на диске около 300-500Мб.
При 60млн записей redis занмает в памяти около 5-7Гб, на диске около 2Гб.
Запрос информации о пользователя занимает около 1мсек.

Схема взаимодействия объектов кода