Некоторые мысли по поводу системы в целом, постараюсь изложить все, что есть у меня в голове, причины, следствия, предположения. Понимаю, что выставляю себя Д’Артаньяном в такой то ситуации, за что извиняюсь, но все из лучших побуждений и стремлении к согласию и успешному завершению общих целей.

Надеюсь удастся достаточно структурировано, объективно и целостно описать, то что есть в голове.

Структура приложения:

* + Интерфейс для работы с базой данных
  + Интерфейс для работы с протоколом snmp
  + Резидентный процесс сбора, анализа и экспорта данных сети (демон)
  + Резидентный процесс, обеспечивающий поддержку протокола HTTP и обеспечивающий общение между клиентом и системой в целом
  + Клиентская часть (набор скриптов для отображения, анализа, структурирования данных и скрипты для обеспечения удобного пользования системой)

Примерно такая структура пришла ко мне в голову еще на первой недели, подумал, просто подумал, что так будет логично.

## Интерфейс для работы с базой данных

Интерфейс для абстракции работы с бд, обеспечивает удобное взаимодействие между системой и бд, так как в нем заранее определены все функции которые могут понадобиться системе в целом, структура его визуализирована на drive.google.com в файле InterfaceDB.

Структура создавалась исходя из требований заказчика, в первую очередь, хотя не без догадок, что примерно хотелось бы ему увидеть.

Так как данный модуль включает в себя класс пользователя системы который храниться в DB, то надобность в модуле авторизации отпадает, так как данный класс предполагает простое получение всех свойств из таблицы users, включая права на выполнение определенных задач, а делать прослойку которая на вход будет получать данную структуру и в одну строчку возвращать все что она получила я не вижу смысла. Плюс база данных будет обеспечивать кеширование данных записей, что позволит уменьшить, или возможно практически исключить обращения к базе данных по поводу пользователей.

Почему обращение к базе данных плохо? Не понимаю почему этот вопрос не очевиден, но все же. Во первых, все данные бд хранятся в файлах на жестком диске (да у жестких дисков есть кеш, у бд есть кеш и возможно все не так плохо), а это значит, что идет лишний цикл записи/чтения, что пагубно сказывается на оборудовании, хоть наша система и не настолько крута что бы за 0.1 секунду ей может понадобиться обрабатывать 16 000 записей и десятки мегабайт, и ей не придется обслуживать 50 000 одновременных подключений, но не хочется делать чего то плохо, что в последующей оптимизации системы потребует переписывание абсолютно всего, поэтому нужно сразу обеспечить минимальную нагрузку на бд (так же не стоит забывать, что база данных, какая бы крутая производительность у нее не была, должна обслуживать не только одного пользователя), поэтому и создается кеш, который хранится в оперативной памяти системы и обеспечивает максимально быстрый доступ к данным.

Почему вопрос производительности важен? Да у нас игра, да мы пишем на java и да система у нас не настолько сложная, но мне лично интересно получить опыт написания системы, которая не отвалится при большой нагрузке, вместо опыта создания беспонтовой фигни, так же это одно из основных требований заказчика: устойчивость к нагрузке и отображение графа при любых условиях.

Теперь касательно самого модуля, он состоит из нескольких классов, а именно классы которые принимаются/возвращаются основным интерфейсом, они по сути представляют из себя отображение записи из таблицы, и так же в модуль входит интерфейс для получения и отправки этих данных. Интерфейс обеспечивает такие действия как: работа с таблицей пользователей, работа с таблицей узлов, работа с таблицей нагрузок по каждому узлу, работа с таблицей конфигурации приложения.

Структура таблиц подробно изображена в документе, что написан выше. Хочется объяснить только касательно таблиц нагрузок. Исходя опять же из вопросов производительности, и уменьшении времени запроса, решил сделать по одной таблице на каждый узел, в имени таких таблиц есть id, который получается из общей таблицы узлов. При такой структуре, базы данных будет проще сделать выборку по определенному запросу, нежели искать имена узлов общей таблице а потом уже выбирать по дате, и я не думаю что выборкой должно заниматься наше приложение, во-первых потому что в бд все действия выполняются за log(n) (нам такие алгоритмы просто не осилить за то время что есть), во-вторых, потому что бд предоставляет удобных функционал для таких запросов, и в третьих таким образом мы сможем уменьшить время работы бд, так как ей придется обрабатывать и пересылать в конечном итоге меньший объем информации.

Структура модуля предполагает его простое изменение и доработку.

## Интерфейс для работы с протоколом snmp

Конечно нельзя назвать, что данный интерфейс будет работать только с snmp и больше ничего делать не будет. Нет, на нем лежит обязанность предоставить разработчику системы удобный интерфейс получения данных с сети, что по мне вполне логично. Естественно данный модуль будет разбит не несколько классов, таких как только протокольный класс, обработка и структурирование данных, кеширование данных и интерфейс для получения информации.

Что делает данный модуль? Кто-то модулю через интерфейс посылает запрос, типа какая там погода на серверном фронте. Интерфейс думает, обращусь-ка я к кешу, кеш смотрит, с момента последнего обращения к сети прошло слишком мало времени выдаду-ка я старые, или так же кеш может посмотреть и понять что данные протухли, а значит обращусь-ка я к протокольной части своего модуля и соберу все данные, потом отдам обработчику данных что бы он их привел в приличный вид (ну или немного не так обработка будет идти, но примерно так) сохраню в себя и выдаду уже после всего этого, не забыв обновить время последнего обновления данных. В первом случае данные получаются моментально, так как объект лежит в памяти и передается по сути только ссылка, во втором чуть по дольше так как идет опрос серверов и обработка полученных от них данных.

С кешем получится избежать нагрузку на сеть которую может создавать данный модуль если при каждом запросе данных будет собирать их заново. Если клиентов, которые используют данный модуль много, то в большинстве случаев будут выдаваться закешированные данные (ну например при частоте несколько обращений в секунду). А значит система в целом получится оптимальнее.

Почему именно данный модуль должен выполнять выдачу данных на прямую? Опять же потому, что при нагрузке мы получаем бОльшую производительность, чем если бы делать транзит через бд, записывая туда данные с каждого пука, так как в бд нужно записать (log(n)), а после считать (еще log(n)), плюс это затраты ресурсов ввода вывода жесткого диска и блокирование других процессов которым, придется ждать бд. А при кешировании данные моментально извлекаются из памяти. Да возможно при небольшом количестве клиентов это не критично, но если пользователей станет чуть больше нескольких, то затраты производительности работы через бд будут обретать линейную или даже экспоненциальную зависимость, нам конечно не важны проблемы оборудования, мы же пишем java, но все же мне не позволяет моя вера во что-то хорошее даже задумывать над использованием бд, как основного узла получения свежих данных, когда период свежести исчисляется в секундах.

Ах да, процесс обновления данных в кеше инициализируется именно обращением к кешу, если нету обращений то и сеть не опрашивается, это не отдельный процесс, а просто набор процедур, которые нужно вызывать.

## Резидентный процесс сбора, анализа и экспорта данных сети (демон)

Я бы отказался от использования слова “демон”, для обозначения данного модуля так как это всего лишь процесс, а назвал бы лучше сборщиком данных.

На этот модуль ложиться две важные задачи: создание истории и анализ.

Сборщик данных работает по расписанию, по истечению одного таймера он запрашивает данные от модуля для работы с snmp и аккумулирует их где-то у себя в буфере, причем если настройки стоят так, что время обновления кеша данных несколько больше чем время обращения данного модуля, то он получит одни и те же данные несколько раз, и это НОРМАЛЬНО. А по истечению другого таймера модуль высчитывает среднее значение из всех накопленных данных, пробует записать их в бд и сбрасывает свой аккумулятор.

Почему бы всем не иметь доступ к данным которые собирает данный модуль? Потому что данный модуль запускается и работает один и долго, он живет в своем уютном мире (отдельном потоке) и не хочет разбрасываться по мелочам, так как ему нужно работать точно, стабильно и так что бы никто не отвлекал. Добавление функционала расшаривания данных во первых усложняется тем, что это отдельный поток, шаблон одиночки решает проблему конечно, но в таком случае требуются установки критичных секций, а если клиентов будет много, которые захотят получить свеженькие, уже обработанные за них данные, укомплектованные и что бы все было круто, то вполне возможно что сборщик данных распыляясь на обслуживание клиентов, просто не сможет записать во время данные, не сможет отправить их в базу данных, один два раза конечно не критично, и система у нас не сильно нагружена скорей всего будет, но все же должно же быть стремление к чему то прекрасному, стабильному и непоколебимому, мы создаем титана, который не будет отвлекаться на всякие мелочи, а лишь четко, точно и дерзко будет выполнять свои задачи (ну по крайней мере мне так кажется, так как в слово “история” всегда заносится большой смысл и относиться к ней небрежно, немного не правильно что ли).

Проблема возникает только в том случает если нам нужно динамически изменить некоторые настройки, например параметры базы данных или время таймеров. Но и тут можно выкрутиться, например сделать метод который принуждает данный процесс запросить новый конфиг и сохранить его у себя. Класс конфига можно смело сделать одиночкой, там не нужно никаких критичных секций и чего то прочего.

Так же данный процесс анализирует полученные данные и у него есть некий список рассылок (который тоже можно обновлять динамически), если есть какие то отклонения в статистике, процесс наблюдает за ними, собирает информацию, формирует отчет и шлет уведомление всему списку, что то не ладно.

Очень важный модуль и очень не хочется что бы он делал еще какие то задачи.

## Резидентный процесс, обеспечивающий поддержку протокола HTTP и обеспечивающий общение между клиентом и системой в целом

Ну вот вроде в целом система и обрисовалась, есть модуль работы с базой данных, есть модуль опроса узлов и есть процесс собирающий и сохраняющий данные.

Но все это бессмысленно если человек для которого мы все это делаем просто не сможет в удобном формате получить эти данные. Обеспечим прослойку между нашими модулями и тем что увидит конечный пользователь.

Так как в требовании были веб интерфейс и java, то не вижу другого очевидно пути как кроме написания сервера работающего по HTTP, пользователь (в данном случае клиентская часть) шлет запрос серверу, сервер обрабатывает запрос, выделяет параметры, печеньки и все прочее, определят ту функцию что ему надо выполнить, ну например пользователь может запросить текущие данные со всех узлов, идем к модулю обработки snmp запрашиваем данные (которые могут взяться из кеши или не из кеша на усмотрение уже модуля, мы этого не узнаем) получаем, формируем ответ, отправляем, и серверу уже больше важно что там происходит дальше (может просто какой нибудь json в окно браузера выводится и все) он со своей работай справился. Так же у сервера можно запросить добавление нового пользователя, получаем параметры запроса, проверяем их, создаем пользователя через обращение к модулю работы с базой данных в бд, формируем ответ об успехе/неудаче, отправляем клиенту, все клево. Если например клиенту понадобилась аутентификация, то проверяем входные данные и отдаем ему печенки. Так же может понадобиться история (результат работы сборщика, всего лишь результат, который хранится в бд) мы обращаемся к бд и забираем ее. Все функции что есть в требовании к клиенту должен и поддерживать сервер, ну и для удобного функционирования сервер должен поддерживать такие базовые функции как выдача запрашиваемого файла, формирование отчетов об ошибках.

У каждого подключившегося клиента на сервере создается поток, в котором запускаются процесс формирования ответа на запрос клиента. Так что все параллельно и теоретически каждый поток может обращаться к любому из модулей, и вот тут возникает небольшая особенность веб-серверов, они должны работать даже больше чем при одном подключении, так есть ли смысл делать транзит через бд для запроса свежих данных? Учитывая, что данные из бд, берутся из файловой системы, на обращение к ней обычно устанавливаются мьютексы, то есть только у одного или нескольких активных клиентов будет возможность взять из базы данные (а это какое то время, и чем больше записей накопится тем больше времени), а остальные будут создавать очередь, чем больше клиентов тем больше очередь, производительности системы упирается в производительность жесткого диска, а клиенты данные захотят видеть свежие, хотя бы не старее 5-10 секунд, вот к памяти можно обратится моментально и кеш все выдаст, ну или на краняйк опросит и выдаст (причем опрос, надо синхронизировать, потому что сразу несколько клиентов его могут инициализировать, но здесь свои особенности (можно обойти конечно и сделать возможность инициализации опроса только у сборщика данных, ему то можно верить и мы знаем почему)), выдаст причем не обращаясь к фс, просто выдаст и все, практически моментально. По моему оперативная память предпочтительная, проще поставить новую планку как мне кажется нежели новый жесткий диск, планку воткнул и готово, диск же воткнул разметил, отформатировал, примонтировал, настроил пути, много с ним сложностей (Вот не понимаю серьезно откуда такая идея? Онлайн данные через бд? Все как то стараются минимизировать обращения к ней, ухищряются, придумывают велосипеды, а тут хоп и (предположительно) 20 \* 20 \* (4 + 4 + 4 + 4 + 4) / 5 + 100 = 1700 +/- 200 байт в секунду пишется в бд (самый минимум, думаю там в раз 20 больше), 1.7 кб в минуту 102, в час 6 мб, да немного, но количество записей по-моему будет колоссально, так как запросы просто непрерывны (а с каждой новой записью как известно база начинает работать чуть медленнее), ну и если даже все норм, ну не ложиться в голове такая модель, зачем нам загружать кого-то, зная что он справится медленно если сами можем решить все по быстрому?.

Короче сервак обслуживает множественные подключения и тоже должен кешировать какие то данные, потому что система у нас предполагается что должна работать при больших нагрузках на сеть, не обязательно на сервер, но при нагрузке на бд например, у нас не получится прочитать файл или еще что то вроде это и сервер не выдаст ответ так как нигде в кеше ничего нет. А если бы было то шанса ответа в разы увеличивается. А как известно работа при нагрузке на сеть основное требование тоже (это к вопросу зачем она нужна нам).

## Клиентская часть (набор скриптов для отображения, анализа, структурирования данных и скрипты для обеспечения удобного пользования системой)

Тут вроде все понятно, есть формочка аутентификации, управления конфигами, управления узлами и так далее, и основное это отображения графа с возможностью просмотра истории по связям, думаю тут все все примерно представляют, это просто набор скриптов и страничек.

**По поводу создания большого количества модулей и гибкой системы**

Задача не настолько объемная, и не предполагает дальнейшего развития проекта, а поэтому нужно идти по пути наименьшего сопротивления (мое мнение, это не плохо и это замечательно укладывается в рамки нашего проекта), то есть нужно избежать большого количества так называемых модулей и архитектура должна быть максимально прозрачной, что бы любой ее понимал, а делать обработчиков, отправщиков, кешироващиков, и другие классы, должны уже подгруппы, будет продуктивнее, если небольшая подгруппа объединившись спроектирует свой модуль а не будет его разносить его на несколько и выносить на обсуждение всей команде, а то как убедились сегодня за несколько часов обсуждения практически ничего полезного, а точнее ничего не было сделано. Поэтому если именно подгруппа будет обсуждать что-то свое то я думаю будет на много продуктивнее. Просто чем больше будет главных модулей, то тем больше нужно будет каждому разбираться во всем что бы суметь адекватно использовать интерфейс.

Например сегодня предлагали из модуля с snmp вынести обработчик данных, чем это плохо? Например я пишу серверный интерфейс, мне нужно получить данные он нагрузке, я хочу видеть у себя нагрузку в удобной таблице, что бы можно было сформировать отчет (еще лучше если форматов получения данных от туда будет несколько), но я должен создать объект для работы с сетевым протоколом, вызвать метод опрашивающий его, полученные данные переслать новому объекту который мне их сформирует, а после уже сформировать отчет, хотя я просто хочу получить табличку и меня не волнует через какой там протокол это все работает.

## Некоторые выводы

Буду очень рад любому другому видению системы или отдельного модуля, многие моменты опустил, так как просто не в состоянии продумать все, не начав писать код. Буду рад замечаниям и вообще любому мнению. И написал это все только в целях того что бы в следующий раз все знали что у меня в голове, почему именно так, почему это имеет какое то обоснование, а не просто слова, какие есть плюсы и минусы и в общем более менее такое общее описание, думаю многим кто еще плохо представляет, что это за модули и какие есть особенности технической реализации, будет полезно.

В субботу третей недели тендер прошел по-моему отвратительно, и что бы избежать полемики вокруг пустых вопрос прилагается все что выше.

*С уважением Максим*