Слайд 1

Слайд 2

5 лет назад я посетил конференцию highload++ 2012 на которой мне пришла идея создать комет сервер. У меня не было объективных причин это делать. Мне просто понравилась эта затея. Тем более у меня были наработки из прошлых проектов которые я бы мог применить.

С тех пор я потратил на этот проект почти 2000 часов. За которые я написал SaaS комет сервис и его опенсорс аналог таким образом что АПИ у них совместимо что позволяет переходить без изменений кода между SaaS версией и опенсорс версией.

Ещё за это время я успел написать подробную документацию на Русском и Английском языках.

Не смотря на то что я использую **CppComet** в продакшене уже почти 4 года я его всё равно активно развиваю постепенно добавляя новые функции. Например на прошлой неделе добавил функционал кластеризации.

На протяжении 4 лет я занимался техподдержкой пользователей проекта. И на основании каждого вопроса улучшал апи и документацию так чтоб следующим пользователям было проще.

Слайд 3

Мой проект не делает что то революционное что нельзя реализовать с помощью других инструментов. Поэтому на этом слайде я собрал список конкурирующих проектов.

Слайд 4

Наверное перед каждым разработчиком рано или поздно возникает задача создания своего чата или реализации каких либо риалтайм уведомлений.

И хорошо если проект новый и вы можете ещё на этапе проектирования учесть где и какие данные должны обновляться в реалтайме.

Но так бывает не всегда. Иногда надо добавить к существующему проекту реалтайм обновление данных.

Предположим захотели мы чтоб комментарии к статьям на сайте обновлялись у пользователей сразу, а не после перезагрузки страницы.

Первое и самое простое решение это запрашивать с сервера аякс запросом данные о новых комментариях раз в секунду.

Решение простое и вполне рабочее, некоторые так и делают. На пример в админке, где максимум 1 - 2 человека сидит.

Слайд 5

Но если посетителей больше одного то этот код отлично подходит если вы хотите устроить ддос своего сайта даже при сравнительно небольшой посещаемости.

Слайд 6

Вместо того чтоб отправлять множество запросов от клиента на сервер, рациональнее отправлять на клиент новые данные в момент когда они появляются на сервере.

В данной схеме клиенты из JavaScript подключаются к комет серверу по вебсокетам и ждут от него уведомлений от бэкенда.

А веб сервер отправляет уведомления о событиях в комет сервер, в надежде что эти уведомления будут доставлены на фронтенд.

По своему опыту скажу что удобно отправлять данные в комет сервер сразу после того как эти данные были записаны в бд сайта.

Слайд 7

У комет сервера 2 апи интерфейса, одно для подключения по websockets от браузеров. И второй интерфейс для отправки запросов от вебсерверов (от бэкенда)

Для JavaScript апи особого выбора в протоколах нет. Работаем по websockets.
А вот апи для обращений с бэкенда можно реализовать по разному.

Слайд 8

Сначала я не заморочился и написал простенькое апи и php клиента к нему. Но получившийся велосипед был прост в разработки и неудобен в поддержке.
В итоге поддерживать обратную совместимость всех версий апи под несколько платформ оказалось трудоемкой задачей.

На фоне своего протокола работа по всем известному REST API выглядит очень привлекательно. Но на практике при работе по http протоколу на каждый запрос надо

установить сетевое соединение, выполнить запрос и закрыть сетевое соединение. Это дольше чем вариант работы по tcp где все запросы выполняются в рамках одного сетевого соединения.

И в итоге я решил пойти по пути который был реализован в SphinxSearch, я реализовал серверную часть протокола mysql и работать с комет сервером стало возможно используя mysql клиенты которые есть под каждый более менее популярый язык.

Слайд 9

Комет сервер стал притворяться что он mysql сервер. А апи стало похожим на обращения к базе данных.

- **insert** для отправки данных клиенту, **select** для получения сведений из комет сервера.
- Имя таблицы говорит о объекте над которым выполняется операция.
- Имя базы данных символизирует версию апи по которой мы работаем. Пока версия 1. Так как уже 2 года не ломается обратная совместимость.

Слайд 10

JavaScript API скрывает в себе много кода для обработки ошибок, функций для кластеризации и различных оптимизаций. На пример сколько бы вкладок одного сайта мы не открыли будет установлено только одно сетевое соединени е с комет сервером а остальные вкладки будут общаться с комет сервером через общее для всех соединение.

Слайд 11

Чтоб получить информацию о каком либо событии в JavaScript надо подписаться на это событие.

На пример если мы хотим получать новые комментарии на той странице которую мы сейчас открыли в браузере, то подпишемся на канал через который нас сервер уведомит о чём либо.

После того как мы подписались можно из сервера отправить данные в тот канал на который мы подписались.
Имя канала может быть любым, главное чтоб сервер отправлял сообщения в тот канал на который мы подписались из JS

Слайд 12

В прошлом примере получить сообщение мог любой кто знал на какой канал надо

подписаться. Но не всегда это удобно.
На пример если мы пишем чат и хотим доставить сообщение кому то конкретному так чтоб другие пользователи не могли получить это сообщение то вместо каналов удобнее использовать механизм личных сообщений.

Для этого подпишемся на особый канал с именем “msg”. В него будут приходить только те сообщения которые адресованы лично нам.
Эта возможность доступна после авторизации пользователя на комет сервере.

Слайд 13

Замеры производительности и потребления памяти я провожу с помощью tsung в среднем расход памяти менее 5 гб RAM на 64000 пользователей. И при этом за счёт того что комет сервер работает в многопоточном режиме ему удаётся почти равномерно задействовать все доступные ядра.

Тест конечно синтетический, реальной нагрузки у меня в продакшене случается гораздо меньше. В среднем по 600 человек онлайн. И в итоге сервера заняты всего на несколько процентов от своих возможностей.

Слайд 14

В итоге мы вернулись к почти тому же объёму кода который я приводил на первом слайде, но сделали всё грамотно. И теперь способны выдержать нагрузку из расчета **примерно 5 Гб RAM на 64 000 онлайн**. Плюс есть возможности для кластеризации, масштабирования и отказоустойчивости.

Слайд 15

Всегда полезно заранее предусмотреть возможности масштабирования архитектуры.

Я в комет сервере предусмотрел возможность кластеризации в которой каждый сервер кластера может принимать запросы и пересылать их тем серверам кластера которые надо уведомить о событии.

Операции вставки данных (insert и set) выполняются асинхронно, это значит что вы не будете ждать пока запрос будет разослан по всем серверам кластера.

Операции выборки данных (select и show) работают синхронно.

Слайд 16

Если что то сломается и комет сервер станет недоступен то в лучшем случае сообщения перестанут приходить пользователям.
Если у нас не кластер то очевидно всё может сломаться разом при проблемах с сервером.
Если кластер из комет серверов то выход из строя одной ноды тоже может оказаться болезненным если именно через эту ноду мы пытаемся отправить запрос.

Слайд 17

В реальном мире ошибки иногда случаются. И писать такой код для обработки ошибок не самое приятное занятие. А главное если часть серверов не доступна то мы будем подключаться не с первой попытки.
Поэтому можно включить в схему кластера haproxy для балансировки запросов между живыми нодами кластера.

Слайд 18

В haproxy есть механизм опроса mysql серверов и исключения из списка тех которые не работают. А ещё есть возможность задать пропорцию для распределения нагрузки между нодами кластера.
В javascript api тоже встроена возможность подключиться к другой ноде кластера если одна из нод недоступна. В итоге если выключать ноды кластера комет серверов то работоспособность сохранится до тех пор пока в строю есть хоть одна рабочая нода.

Более того клиентам из web браузеров не потребуется даже обновлять страницу. Всё будет скрыто внутри javascript api.

Слайд 19

Первое это улучшения работы в кластере. И второе это добавление функционала видеочатов и видео конференций.

Уже есть демка работы видео чата. Но АПИ ещё не устоялось до конца. И хотя видео чаты уже можно использовать. Вероятно следующие релизы не будут иметь обратной совместимости для видеочатов.

Слайд 20

На самом деле мой комет сервер используется на нескольких десятках сайтов. Здесь только часть проектов о которых я знаю. В первых трёх встроен простой общий для всех участников чат.

Во второй строке чат для социальной сети и чат для сайта знакомств.