Пользователи часто регистрируются и логинятся/выходят из приложения (нагрузка на систему аутентификации и работы с паролями)

Функционал реализован, но не протестирован.

Много пользователей одновременно приходят на одну из функций приложения (отдельно на "быструю", "вычислительную" и “медленную”)

Все графики были построены автоматически через плагин Overload, метрики хоста собирались через Telegraf. Запуски и все графики хранятся в системе, поэтому привожу только ссылки на запуски.
(не зря же я провозился с этим Overload, документация не соответствует продукту, чтобы запустить этот плагин, пришлось читать код танка, чтобы понять, что плагин был объединен с другим и вырезан — opensource as is)
Во вкладке Responses метрики клиента, во вкладке Monitoring метрики сервера.

На быструю

https://overload.yandex.net/18474
Использовалась база с 48314 записями о городах в мире, у каждого города есть id региона где он расположен, запрос заключается в выводе всех городов по id региона. База была проиндексирована по id городов, а sql-запрос использовал id региона, поэтому такой запрос получился совсем не “быстрым”. На графиках видно, как сервер не смог уже обрабатывать больше 300 RPS и стал копить backlog, хотя и было запущено 8 воркеров, на машине с 4 ядрами (8 с гипертредингом).

На вычислительную

https://overload.yandex.net/18412
https://overload.yandex.net/18423
Поиск множителя происходил самым примитивным перебором до корня, запросы отправлялись на факторизацию чисел 4349953 (мин множитель 1553), 287437699 (16007), 1219387079 (34781). На первом запуске нагрузка повышалась до 500 RPS, с чем сервер справился, на втором запуске заведомо был сделан для перегрузки с 2000 RPS, судя по графику, сервер справился ещё с 1000 RPS. Видимо, вычислительная функция будет показывать быструю функцию, на фоне неоптимального запроса к БД.

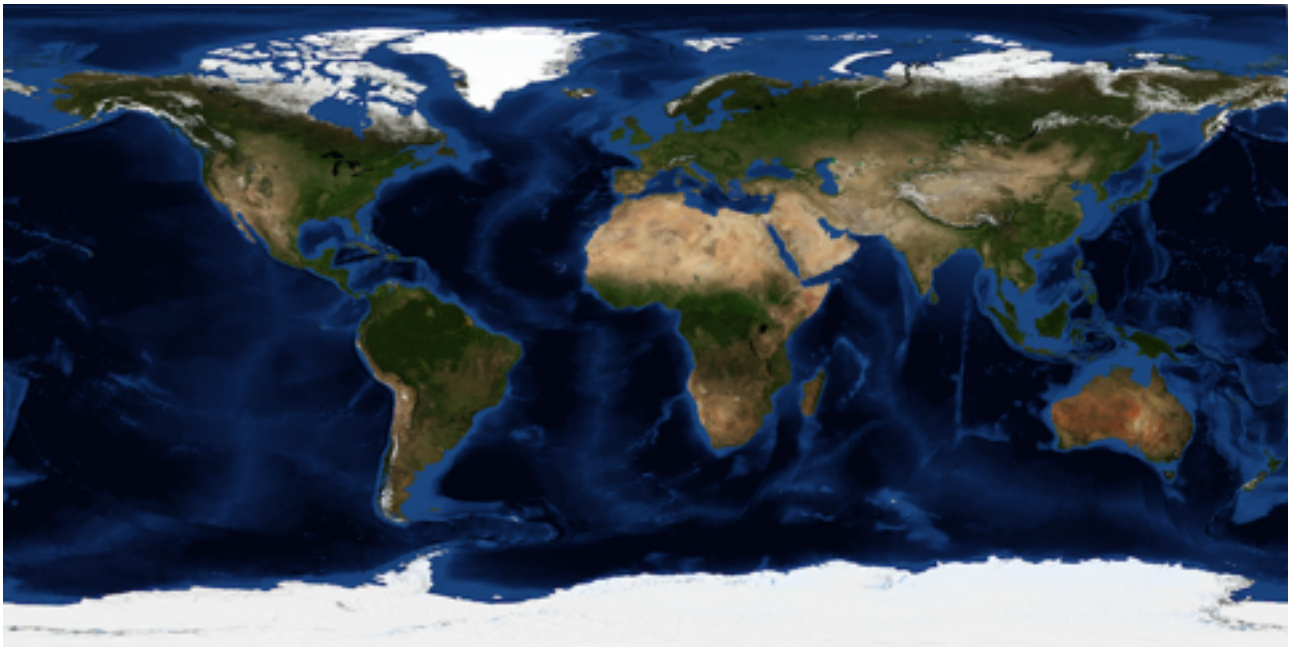
На медленную

Функционал реализован, сервер постепенно отдает лог ping-а, с помощью ответов на LongPoll запросы. Сессия логирования привязана к каждому пользователю, и запускается в единичном экземпляре на пользователя, либо перезапускается. Как полностью протестировать такой функционал yandex-tank-ом, я не придумал.

Много пользователей запрашивают статические ресурсы

Nginx

Тестировать Nginx на маленьких файлах не интересно, известно, что он прекрасно раздает статику, а поломаться может только когда файловые дескрипторы кончатся. Поэтому поскачиваем относительно большой файл (14MB) со снимком из космоса.



Понятно, что все проблемы будут в том, что файл долго отдается. А когда одновременно отдается несколько файлов, то всё ещё хуже. Стабильно держится 20 RPS, дальше уже отдачи не хватает, так как всё тестируется внутри одной системы, то поправок на скорость передачи нет, все ограничения связаны лишь с производительностью машины. Внезапно, во время работы тестов на статику, мониторинг агент сервера отвалился во время теста, не отдав никаких данных. 32 RPS, которые были взяты из примерной оценки возможностей сервера, с некоторыми проблемами под конец, но всё же проходят. Второй тест, опять же, заведомо, чтобы завалить систему. На графике виды скачки, это связано с тем, что куча запросов получает почти одновременно таймаут, соответственно это легкие сервисные ответы ответы. Если отфильтровать чисто 200 OK, то производительность не выше 30 RPS, а в среднем 20.
https://overload.yandex.net/18455
https://overload.yandex.net/18456

Flask

Flask из коробки не сильно хуже раздает статику, относительно Nginx. Сравнивалась работа в многопоточном режиме, если запускать один процесс, очевидно, что производительность будет несравнима. Файл здесь не кэшировался, как и в предыдущем случае, поэтому ограничения связаны ещё и с постоянным чтением файла.

https://overload.yandex.net/18445