Привет. Меня зовут Алексей Плетнёв и вот уже 18 лет я работаю программистом в компании Базис-Центр, где мы делаем САПР «БАЗИС», занимающий лидирующие позиции на рынке отечественного ПО для автоматизации мебельных производств. Современный САПР — это не только конструкторские модули, но также модули управления производством, решения для продажи мебели в салонах, модули управления складским хранением и другие. Все их объединяет то, что им где-то нужно хранить свои данные. Когда мы только выходили в онлайн, то нам хватало одной БД на виртуальном выделенном сервере (VPS). Постепенно количество пользователей росло, а вместе с ним росли количество хранящихся данных и риски оставить клиентов без всех этих данных, либо без их части, которая накапливается от бэкапа до бэкапа. И тогда мы задумались о том, что неплохо бы начать хранить свои данные более надёжно и доступно. Под доступно я имею ввиду с максимальной доступностью и минимальным даунтаймом соответственно. Мы поняли, что пришло время нам распробовать облака, а конкретно – managed database. На тот момент облачные сервисы в России были не столь распространены, и мы пошли изучать, что предлагают Amazon, Microsoft и Google. Запросили триалы у всех трёх, почитали документацию, поигрались с настройками и поняли, что идеология и ценовая политика Google Cloud SQL нам подходит лучше всего.

В своём докладе я хочу поделиться накопленным за эти годы опытом использования различных Managed Database, попутно рассказывая, как они устроены с точки зрения пользователя. Расскажу с какими проблемами мы столкнулись и почему нам приходилось мигрировать из одного облако в другое и не один раз. Принципы реализации и работы с сервисами управляемых баз данных в облаке не сильно зависят от движка, но конкретно мой опыт базируется на MySQL, потому во всех случаях (для модных ребят кейсах) речь идти будет именно о нём.

Для начала разберёмся что же такое база данных в облаке, или Managed Database, и как она поможет нашему сервису стать быстрее и отказоустойчивее. В простейшем случае, это просто виртуальная машина с предустановленной на ней СУБД, расположенная на вычислительных мощностях облачного провайдера. Он же, провайдер, берёт на себя все хлопоты связанные с обслуживанием этой СУБД, такие как обновление, резервное копирование, обеспечение доступности и т.п. Нам, как клиентам этого сервиса остаётся только залить свои базы, настроить свои сервисы на их использование и спать спокойно, находясь в полной уверенности, что с нашими данными и данными наших пользователей ничего не случится. Но это не совсем так, ведь даже у крупных облачных провайдеров случаются не просто аварии, а целые катастрофы с потерей не только данных, но и самого дата-центра. Так, например, совсем недавно, в 2021 году в Страсбурге сгорел дата-центр OVH SBG2, а вместе с ним оборудование и данные. И сами облачные провайдеры предупреждают о подобных рисках, поэтому делают несколько так называемых зон доступности, или, попросту говоря, масштабируют своё облако на несколько физических дата-центров, сгруппированных по регионам, и призывают для рабочих нагрузок использовать отказоустойчивую конфигурацию из, как минимум, двух инстансов, находящихся в разных географических локациях.

Теперь, пожалуй, самая интересная часть доклада – рассказ про то, как же устроена такая катастрофоустойчивая система. Начинается всё с создания двух инстансов интересующей нужной СУБД в двух разных зонах доступности. Один инстанс назначается мастером, второй слэйвом и включается классическая Master-Slave репликация. Естественно, что делается это всё в удобном веб-интерфейсе в пару кликов. В случае, если мастер по каким-либо причинам выбывает, слэйв становится новым мастером, а в другой зоне доступности создаётся новый инстанс, который становится слэйвом. Всё понятно и красиво. Но давайте теперь посмотрим, как это выглядит со стороны пользователя и какие проблемы могут при этом возникать.

Первым будет Google Cloud SQL, так как именно с него мы начали использование облачной СУБД. Итак, у нас есть мастер, с которого идёт репликация на слэйв и они находятся в разных зонах доступности. У мастера свой ip, у слэйва свой. Нашему приложению мы указываем в настройках ip мастера и, собственно, всё – начинается штатная работа. Что произойдёт при сбое мастера? Всё просто и красиво:

1. Слэйв дождётся, пока применятся все существующие изменения.
2. IP мастера назначится слэйву.
3. В другой зоне доступности создастся новый инстанс, которому будет назначен ip слэйва и всё продолжит работать как ни в чём не бывало.

Как видим, никаких действий со стороны приложения не требуется. За долгие годы пребывания наших баз в Google Cloud SQL мастер ни разу не сбоил, но пару раз мы, в экспериментальных целях пользовались ручной функцией переключения с мастера на слэйв – всё происходило быстро и практически бесшовно. Всё благодаря искусным манипуляциям с протоколом bgp в инфраструктуре Google Cloud.

Но, настало время импортозамещения, и мы поспешно начали искать, какому облачному провайдеру нашей страны можно доверить свои базы данных. Первым в списке было облако от компании Яндекс. Мы так же создали два инстанса, немного их потестили и решили, что готовы к переезду. За пару дней мигрировали данные и… начали ловить проблемы. А именно, периодически приложения не могли получить доступ к базам данных. Что же такого случилось? Ребята в Яндексе не стали заморачиваться манипуляциями с ip адресами и катастрофоустойчивость реализовали на уровне DNS. Каким образом? Аналогично смене ip-адресов, только вместо них dns имена. Т.е. есть имя, которое указывает на мастер, если имя, которое указывает на слэйв. Если мастер отъехал, то его имя начинает указывать на бывший слэйв. А имя бывшего слэйва на новый слэйв. Никаких сложных манипуляций с bgp – просто смена значений записей в dns, который при этом имеют минимальный ttl, равный 1 минуте.

Так, а в чём же заключалась проблема, которая мешала нашим сервисам нормально жить? В том, что на некоторое время запись, указывающая на мастер, переставала резолвиться. Т.е. nslookup в течение некоторого времени говорил, что нет такого сервера. И такие провалы случались буквально каждый час на 1 минуту. Сервисы начинали сходить с ума, так как не просто не могли достучаться до СУБД – они просто в этом время не знали куда им стучаться. Естественно, мы обратились в поддержку, где нам сказали, что, вероятно, проблема кроется в используемых нами DNS серверах. Тогда мы сделали следующее:

1. В качестве DNS для своих сервисов указали публичные DNS серверы самого Яндекса.
2. В Google Cloud настроили uptime check на СУБД в облаке Яндекса.

Результат? Проблема никуда не ушла. Мастер периодически не резолвился что на DNS Яндекса, что в метриках Google Cloud.

Через пару дней служба поддержки рапортовала, что проблема найдена и исправлена. Но, увы, ничего не изменилось.

Тогда нас попросили попробовать вместо fqdn использовать текущий ip мастера. И всё заработало как надо – за сутки ни одного не достучавшегося сервиса, жаждущего получить данные. Но в этом случае теряется не просто катастрофоустойчивость, а вообще какая-либо устойчивость, так как даже после перезапуска инстанса у него будет другой публичный адрес.

Ещё несколько дней переписки привели к тому, что нам посоветовали интегрировать наш внутренний DNS с DNS Яндекса. Т.е. поддержка Яндекса снова грешила на промежуточные DNS. Но уже тогда я понял, что вряд ли это поможет и пошёл простым путём. Я написал небольшую функцию, используя Yandex Cloud Functions, которая каждую минуту просто пыталась подключиться к базе в том же облаке. И эту функцию не обошла стороной данная проблема. Больше никаких вариантов решений предложено не было, поэтому мы снова начали планировать переезд.

Кстати, на прошедшей пару лет назад в Москве конференции HighLoad я задавал вопрос ребятам из команды облачного MySQL Яндекса о причинах такого странного поведения их сервиса. Они тактично сослались на команду, которая занимается DNS, поэтому что стало источником проблем осталось неясным до сих пор. Видимо, не зря команда Google Cloud SQL заморочилась именно с ip адресами, а не с доменными именами.

Следующим в нашем списке облаком стало VK Cloud. Мы запросили тестовые ресурсы, создали инстанс и начали его грузить, в результате чего на второй день он… завис. Ну ок, бывает, сказали мы и нажали ребут. Но не тут-то было – инстанс не захотел уходить ни в какой ребут. Очередные пара попыток перезапуска так же не привели к успеху. Мы, естественно, обратили в поддержку и описали ситуацию, однако ответа в этот день не последовало. И в следующий тоже. И через день. На 4 день мы решили, что такой сервис нам не подходит и убили проблемный инстанс. Ещё через пару дней пришёл ответ из поддержки, в котором нам сообщили, что проблему диагностировать не удастся, так как инстанс мы собственноручно грохнули. На этом наши эксперименты с VK Cloud закончились.

Куда же можно ещё мигрировать подумали мы и наткнулись на облако от компании Сбер, которое нынче называется просто cloud.ru. Созвонились с менеджером, который рассказал, что, по сути, данное облако представляет из себя решение Huawei Cloud, развёрнутое на мощностях Сбера и полностью обслуживаемое специалистами того же Сбера. Изучили документацию и поняли, что принцип смены мастер в случае его отказа абсолютно такой же, как у Google Cloud SQL. Потестили какое-то время, нам всё понравилось, и мы благополучно мигрировали. Всё вполне себе хорошо работало, пока не настало 10 января 2024 года. В этот день ~~Земля~~ наш MySQL мастер в облаке Сбера остановился. Точнее перешёл в статус abnormal и перестал принимать запросы. При это никакого автоматического переключения на слэйв в другом дата-центре не произошло. Ок, мы нажали на кнопку «переключить мастер в другой ДЦ», но чуда не случилось – случилась ошибка «Another operation is being performed on the DB instance or the DB instance is faulty.». И мы, конечно, побежали за помощью в поддержку, где нам сказали, что проблема критическая, затронула многих клиентов (лежал, например, мегамаркет) и скоро ей поправят. И действительно, через примерно 1.5 часа её поправили и телефонные трубки уже нашей техподдержки начали потихоньку остывать. Мы, естественно, потребовали объясниться, что это было, на что получили вот такой ответ:

сформировали запрос по поводу вашего обращения в вендора, производим диагностику тикет до 15.01 приостонавливаем до выяснения причин такого поведения сервиса

А через пару дней такой:

сформировали запрос по поводу вашего обращения в вендора, производим диагностику тикет до 17.01 приостонавливаем до выяснения причин такого поведения сервиса

А потом такой:

сформировали запрос по поводу вашего обращения в вендора, производим диагностику тикет до 19.01 приостонавливаем до выяснения причин такого поведения сервиса

И так 1.5 месяца. Это следствие того, что, в отличие от других облачных провайдеров, которые эксплуатируют собственные решения и могут в них поковыряться, облачным провайдерам вроде Сбера, а это Билайн, Мегафон, МТС и многие другие, приходится только разводить руками и ждать ответа от вендора.

Но ответа они, по всей видимости, не дождались, потому что нам ответили следующее:

Алексей, добрый день!

Получили анализ инцидента, вследствие того, что в дата-центре в серверном зале во время проведения работ пропало резервное питание, в моменте ушли в оффлайн сетевые ресурсы, что не позволило переключению корректно случиться.

Уточните, пожалуйста, у Вас остались вопросы в рамках данной заявки?

Тикет можно закрывать?

С уважением,

Команда Cloud.ru

Т.е. сбой питания на резервной линии остановил мастер и не дал переключиться на слэйв в другом дата-центре вручную. О какой катастрофоустойчвости тут может идти речь?

На тот момент у нас уже довольно сильно подросли в размерах наши базы и мы задумались о том, чтобы попробовать развернуть отказоустойчивый кластер на своих серверах, благо они располагаются у нас в 4 ДЦ в разных регионах России. Мы разворачивали разные типы кластеров и проводили тесты с ними на тот момент в спокойном режиме, так как инцидент остался позади. Пока не настало 26 января 2024 года. Тогда Cloud.ru атаковали, вследствие чего начал пропадать доступ ко всем сервисам, в том числе нашим базам. Они, похоже не были готовы к такой атаке, потому как подавляли её больше недели. Всё это время наши сервисы пребывали в состоянии 504. Ни сроков, ни вариантов решения не было. Тут уже наше терпение лопнуло, и мы решили, что наш собственный кластер достаточно оттестирован и вполне можно мигрировать на него. Как только атаки ослабли мы смогли забрать свои данные и благополучно съехать. Но это тема уже отдельного доклада.

Самое обидное, что когда мы потребовали компенсацию за простой согласно SLA, то, во-первых, нам сказали, что атака была всего два дня, а не 10, а потом сервис «удалось стабилизировать». И компенсации никакой так мы и не получили.

Подытоживая свой опыт работы с облачными БД, я сформировал следующие выводы.

Из позитивного:

* Нет необходимости самому настраивать инстансы – всё делается в пару кликов
* Катастрофоустойчивость из коробки – ваши данные сразу распределены географически
* Надёжное хранение и доступность – за всем этим 24/7 следит команда профессионалов, вследствие чего вы можете сэкономить на своей.

Из негативного:

* Не всё так гладко как это описано в разделе «из позитивного»
* Если что-то поломается, будет атака, то можно ждать разрешения ситуации без временных ориентиров и никак повлиять на это не получится
* Если для организации облака используется решение стороннего вендора, то проблемы будут решаться ещё дольше, причины могут быть не найдены, а запросы на добавление/изменение функционала не реализованы.
* При большом объёме данных цена сильно выше, чем если хранить у себя.
* Ещё один не затронутый аспект – сетевые задержки.

Я не знал куда его впихнуть в середине доклада, потому оставил на конец. В России ни у одного иностранного облачного провайдера дата-центров нет. Когда мы пользовались Google Cloud SQL, то самые ближайшие серверы находились в Финляндии. У российских провайдеров подобные карты отсутствуют, но у всех них физически облака располагаются в Московском, либо Северо-Западном регионе. Т.е. Московская и Ленинградская области. У нас же пользователи есть по всей стране, потому мы размещаем свои сервера в Московской, Самарской, Новосибирской областях, а также в Республике Татарстан. Пинги от Новосибирска до Московских облаков могут достигать 50 мс., которые прибавляются к каждому запросу соответственно.

* Последний аспект с собственным слайдом – масштабируемость.

Казалось бы – облака просто созданы для масштабируемости и их ресурсы практически безграничны. Однако с сервисом Managed Database не всё так просто. Как я говорил ранее, все они построены на основе Master-Slave репликации. Т.е. есть одна нода, которая может выполнять операции чтения и записи – остальные только чтение с небольшим временным лагом. Это означает, что, когда ваш проект упрётся в производительность одной ноды, вам нужно будет заняться шардированием. Самостоятельно.

Исключение – cloud native базы данных, изначально созданные для работы в облаках, такие как YDB или BigQuery. Но тут тоже есть свой нюанс – начав их использовать можно привязать себя к конкретному вендору навсегда.