Привет. Меня зовут Алексей Плетнёв и я ведущий программист компании Базис-Центр. Наша компания делает САПР для автоматизации мебельных производств, и мы имеем дело с внушительным количеством трёхмерных моделей, которые нужно где-то хранить, когда речь заходит о предоставляемых нами интернет-сервисах. Для этих целей мы используем геораспределённый кластер на базе Минио и год назад на хайлоаде я рассказывал почему мы выбрали именно Минио и на какие грабли мы наступали пока его разворачивали. Доклад нашёл отклик и в течение года мне пришло много писем от слушателей с уточняющими вопросами. Раз тема вызвала интерес, то в этом году я решил эти вопросы и ответы на них систематизировать и представить в виде отдельного доклада. Кроме того, в прошедшем году нам пришлось масштабировать свой кластер как по объёму хранения, так и по точкам его присутствия – добавился один дата-центр и один (кстати питерский) мы сменили на другой (не питерский). Названий я, естественно, называть не буду, чтобы не делать никому рекламы и антирекламы – важен сам факт такой масштабной миграции и её нюансы.

Начнём с секции ответов на часто задаваемые вопросы.

И первый интересный вопрос звучит так.

Как всё-таки лучше поднимать кластер из нескольких нод, когда суммарное количество дисков превышает максимальный размер Erasure set, т.е. 16?

Для начала напомню, что такое Erasure set. Это группа дисков от 4 до 16, обеспечивающая надёжное хранение данных и допускающая, в зависимости от настроек, потерю до половины дисков без потери данных. Размер группы выбирается автоматически и равен максимальному делителю из диапазона от 4 до 16, при котором общее количество дисков, подключенное ко всем нодам, делится на него нацело. В данном случае мы будем иметь 200 ЕС групп по 16 дисков в каждой.

Год назад я показывал вот такую строку с примером запуска Минио на 16 машинах, к каждой из которых подключено по 200 дисков. И из её содержимого, как и из документации, абсолютно непонятно как именно будут распределяться диски из групп между нодами. Логично предположить, что каждый диск должен присоединяться к группе с новой ноды. Это логично, потому что для объяснения подобран красивый пример, где количество нод соответствует максимальному размеру группы. Взять для примера 6 нод по 50 дисков и всё становится уже не так очевидно. В этом случае мы получаем 6 \* 50 = 300 дисков. При таком раскладе мы получим 20 групп по 15 дисков в каждой и как они будут распределены между нодами непонятно. Как непонятно и то, могут ли все 15 дисков группы оказаться на одном сервере, который ляжет в неподходящий момент? Нигде это не описано, а разработчики не готовы давать ответы на такие вопросы бесплатно и просят оплатить подписку, стоимость которой, напоминаю, начинается от 1000 долларов в месяц.

За прошедший год конкретного ответа на этот вопрос найти не удалось, поэтому мы поступаем просто – делаем так, чтобы общее количество дисков в кластере не превысило максимальный размер группы, т.е. 16. Чтобы этого добиться, на каждой ноде мы объединяем диски в пулы с помощью ZFS. Об этом я тоже рассказывал год назад и поэтому среди вопросов был следующий.

Вместо ZFS рассматривали LVM? Если да, то почему выбрали именно ZFS?

Ответ – да, рассматривали. Первый кластер подняли на LVM + XFS. Причём XFS – рекомендация из документации самого Минио (<https://min.io/docs/minio/container/operations/install-deploy-manage/expand-minio-deployment.html>). Разработчики для достижения максимальной производительности крайне не рекомендуют использовать различные прослойки между диском и Минио, вроде raid, пулов и т.п. Однако наш опыт разнится с этими рекомендациями. Например, из-за предыдущего вопроса. Итак, когда мы дошли до того, что диски нужно объединять в пулы, естественно, самым логичным было бы сделать это с помощью LVM, но мы сделали это на ZFS. Почему? Во-первых, по рекомендации куратора моего прошлогоднего доклада Ивана Агаркова. В их компании уже был подобный успешный опыт, только локальным кластером. Во-вторых, что интересно, на тот момент я вообще не был знаком с ZFS. После рекомендации Ивана начал её изучать, быстро проникся идеологией и понял, что это именно то, что надо. Изначально она разрабатывалась как коммерческая файловая система, предназначенная для хранения Зеттабайта данных (собственно ZFS расшифровывается как Zettabyte file system). Она крайне проста в управлении и все манипуляции с ней сводятся к использованию одной утилиты zpool.

Мы, конечно, используем LVM на наших серверах и до сих пор банальное расширение диска я делаю со шпаргалкой, сохранённой в отдельном файле. Шпаргалка эта состоит из десятка шагов. Люди целые статьи в интернете публикуют на тему «как расширить том LVM». Для ZFS шпаргалку даже писать не пришлось – всё сразу понятно. А я по жизни придерживаюсь принципа – чем система прозрачнее и понятнее, тем лучше. Если 100 ручек можно заменить одной кнопкой и при этом будет абсолютно понятно, что произойдёт после её нажатия, то это просто отлично.

Так же отмечу, что, несмотря на то, что LVM умеет расширять тома на лету, иногда это становится причиной некорректного поведения сервисов, работающих с диском. Мы пока глубоко не разбирались в причинах, но написали для себя памятку с рекомендацией, по возможности, перезагрузить сервер после расширения диска.

Ещё одна интересная плюшка ZFS – мониторинг IOPS. Когда речь заходит об объектных хранилищах, важным показателем загруженности и здоровья кластера является контроль этого параметра для дисков кластера. Наверняка для LVM + XFS существуют утилиты, которые позволяют это сделать, но простые atop, iotop и т.п. такого не показывают, а zpool iostat выдаёт именно то, что надо.

Чтобы не создалось впечатления, что я вышел на эту сцену только для того, чтобы восхвалять ZFS, будем считать, что на вопрос я ответил достаточно обоснованно и перейдём к следующему.

Как лучше проводить обновление кластера? Ты говорил, что нельзя обновлять узлы один за другим, но надолго останавливать кластер не хочется?

Ответ на этот вопрос будет короче, чем сам вопрос:

mc admin update <кластер>

У Минио есть родная утилита для его администрирования, которая позволяет скачать и установить обновления на все ноды кластера и разом их перезапустить. Нужно только учесть пару нюансов:

1. Минио имеет давнюю проблему со скоростью скачивания дистрибутива. Они меняли провайдера своего облака с AWS на OVH, но это не помогло и в трекере до сих пор висит задача сделать что-то со скоростью скачивания.

https://github.com/minio/minio/issues/9847

1. Обычно первое обновление кластера заканчивается ошибкой. Причиной служит то, что согласно инструкции минио ставится в папку /usr/local/bin, прав на запись в которую у него нет. А для обновления он использует атомарное переименование, для которого доступ на запись нужен. Поэтому после установки лучше сразу давать права на запись в эту папку. Это выглядит как дырка в безопасности – возможность писать в папку с бинарниками и об этом разработчиков спрашивали на их гитхабе. Суть обсуждения сводилась примерно к тому, что навряд ли на этой машине будет установлено что-то помимо самого минио, поэтому ничего страшного нет.

https://github.com/minio/minio/issues/8226

Как видите, ответ короткий, но мелочей, которые надо учесть, хватает. Как и всегда с Минио. Идём дальше.

Мы собрали кластер из X нод по Y дисков на каждой, установили EC:M согласно калькулятору, но при падении одной ноды падает весь кластер. Почему?

«Видимо что-то не так собрали» ответил я, а после подумал, что ответ достоен первой линии поддержки, а не ведущего программиста и попросил конфиги и описание кластера. В ответе я был уверен, так как перед началом эксплуатации неоднократно отключал-включал ноды для проверки того, на сколько упадёт производительность и что на самом деле может пережить Минио и всё было более-менее предсказуемо, а тут такое. Вдохновлённый рассказами ребят из Яндекса о том, как они проводят учения и отключают один из дата-центров, чтобы убедиться, что их сервисы готовы к катастрофам такого рода, я решил, что и нам бы не мешало... и выключил одну из нод. Минуты через 3 Минио перестал отвечать на запросы. Пришлось стартовать ноду обратно и быстрее лезть на гитхаб описывать ситуацию.

<https://github.com/minio/minio/issues/16620>

Оказалось, что проблема появилась в одном из обновлений и буквально пару недель назад была исправлена. Получается, что собрали всё правильно, просто не из той версии, а я на тот момент не имел привычки после обновления кластера проверять осталась ли отказоустойчивость на том же уровне. А надо бы. В-общем, проблема была решена обновлением как у меня, так и у вопрошающего, но сама ситуация заставила задуматься. В очередной раз.

Развернули кластер, включили компрессию, льём данные. Как посмотреть объем именно сжатых данных в кластере? Может можно как-то посмотреть в mcli?

Я тоже задался этим вопросом, когда включил компрессию. Компрессия на лету вообще была жирным плюсом в нашей таблице сравнения различных систем в процессе принятия решения какую из них использовать. Однако ответ на это был и остаётся простой – нет. Кому интересны подробности, вот мой вопрос на гитхабе:

<https://github.com/minio/minio/discussions/13943>

Почему при выпадении нод начинает сильно тупить веб-консоль?

Я тоже задавался этим вопросом, когда тестировал отказоустойчивость и отключал ноды. Очень раздражает, когда при выпадении нод вход в Минио растягивается на минуты. Проблема в том, что при входе Минио на уровне фронта опрашивает все сервера кластера, чтобы отобразить статистику их доступности. Соответственно, время входа в веб консоль увеличивается на время таймаута ожидания ответа от упавшей ноды. Логичнее было бы сделать при переходе на вкладку мониторинга, но пока оно остаётся так как есть. В такие моменты иногда удобнее использовать cli.

Следующий вопрос подводит нас ко второй части доклада, в которой я расскажу, о нашем опыте развёртывания кластера в 4 дата-центрах и переносе данных в него из старого кластера. Звучит он так:

Всё-таки способен минио работать как геораспределённая система хранения, или ему место только в одной стойке? Как влияют сетевые задержки на общую скорость работы?

Здесь ответ будет более развёрнутым. Для начала я напомню, про какой кластер я рассказывал в прошлый раз. Это три сервера с установленными в них обычными HDD ёмкостью 5 ТБ по 4 шт. в каждом под холодное хранилище и по 2 шт. NVME SSD ёмкостью по 500 ГБ под горячее хранилище. Один сервер находится в нашей серверной в г. Коломна Московской области, один в Питере и один в Казани. Между собой они связаны линками по 500 Mbps. Задержка между любыми двумя дата-центрами составляет порядка 20 мс. Получается такой треугольник.

Что сподвигло нас расшириться до 4 дата-центров? Во-первых, дополнительная отказоустойчивость, которая повлекла за собой увеличение полезного пространства для хранения. Поясню. Но сначала наша новая схема, чтобы лучше понимать - что было и что стало. Выглядит она так.

В каждом дата-центре теперь стоит сервер с подключенной к нему дисковой полкой. В дисковой полке 25 SAS дисков по 600 ГБ со скоростью вращения шпинделя 15 000 RPM. Дата-центра 4 – один в Коломне, один в Казани, один в Самаре и один... в Новосибирске. Все дата-центры соединены каналами в 1 Gbps. Пинги между Коломной, Казанью и Самарой также как и раньше не превышают 20 мс., но чтобы долететь до Новосибирска и вернуться обратно пакету требуется уже порядка 50 мс. – физику не обманешь.

Большая вводная, сейчас расскажу детальнее.

Почему теперь дисковые полки и почему нет больше деления на горячее и холодное хранилище?

Ответ прост – Минио действительно умеет по максимуму использовать предоставленные ему диски. Он пишет и читает одновременно с большого их количества и чем дисков больше – тем выше производительность. Как я рассказал раньше, мы используем ZFS. Диски в каждой из полок разбиты на 4 пула по 6 дисков в каждом. Один диск остаётся в резерве, чтобы быстро можно было подменить вышедший из строя, не дожидаясь пока замена доедет до дата-центра, а ломаются они, согласно закону подлости, не в Коломне. Таким образом мы получаем 4 ноды с 4 дисками на каждой или всего 16 дисков, что как раз укладывается в максимальный размер EC группы Минио. Тесты показали, что такой кластер запросто забирает порядка 80 мегабайт на запись и столько же отдаёт чтение, чего нам вполне достаточно. Стоимость же обычных SAS дисков вместе с полками гораздо ниже NVME SSD, поэтому мы отказались от разбиения на горячее и холодное хранилище и храним всё вместе. Плюс один кластер проще администрировать и легко масштабировать – достаточно притащить в ДЦ новую полку, воткнуть её каскадом в старую и расширить пул. Но нужно быть аккуратными и помнить две вещи:

* Не стоить делать слишком большие пулы с помощью ZFS – в случае выпадения дисков пул нужно будет пересобрать и наполнить данными. Процесс восстановления большого пула может занять длительное время. Лично я бы не стал делать его размер больше 6.4 ТБ. Сейчас у нас пулы на 3.2 ТБ.
* В Минио нельзя просто так добавить новые диски – нужно собрать новый кластер и присоединить его к старому. Все операции записи пойдут на новый кластер и будут идти на него до тех пор, пока объёмы занятного пространства на старом и новом кластерах не сравняются. Затем запись будет распределяться между ними. Каждый кластер будет иметь свою откзоустойчивость, но при этом они будут равны для обоих, так как кластеры фактически крутятся на одном железе.

Вернусь к отказоустойчивости и полезному пространству для хранения. Понятно, что 4 ДЦ лучше, чем 3. Но почему при этом пространство получилось использовать более эффективно и с большей надёжностью? Раньше холодное хранилище состояло из 3 нод по 4 диска на каждом, т.е. итого 12 дисков. В формуле EC:N, определяющей отказоустойчивость кластера, N равнялось 4, что позволяло потерять полностью 1 ноду с 4 дисками, сохранив при этом полную работоспособность. Полезный объём в данном случае равен объёму 8 дисков. Теперь же мы имеем 4 ноды с 4 дисками, итого 16 дисков. Значние N выбрано равным 6. Полезный объём равен уже объёму 10 дисков. Можно было бы сделать и 12, выбрав N равным 4 и точно так же пережить выпадение одного дата-центра. Но, как выяснилось, дата-центр может выпасть не только вследствие аварии, но и потому, что так было задумано. В нашем случае мы задумали переезд и Питера в Самару. Нужно было погасить ноду вместе с дисками на одном месте, перевезти в другое и там запустить. На это потребовалась примерно неделя. Если бы N был равен 4, то всю эту неделю кластер находился бы в опасном состоянии – вылет всего 1 диска привёл бы к его остановке, что недопустимо. Напомню, что когда я говорю о 4 дисках на каждой ноде, то имею ввиду 4 дисковых пула ZFS. По факту это 96 физических дисков и вероятность выхода из строя одного из них весьма велика. А так мы уменьшаем свою привязанность к конкретному дата-центру и, если, например, один из них вдруг неадекватно взвинчивает цены на свои услуги, как это было с нами, мы можем с ним легко распрощаться и переехать, минимизируя при этом риск остановки кластера и не тратя на это лишнее дисковое пространство.

Теперь немного о железе. Каждая нода представляет из себя виртуалку со следующими параметрами: 8 vCPU Intel Xeon v3, 16 GB RAM DDR3. Кластер из 4 виртуалок справляется с обработкой 348 запросов в секунду. При этом Минио наиболее требователен к памяти и к производительности сети и гораздо менее требователен к процессору и сетевым задержкам. Это позволило нам разместить одну из нод в Новосибирске, поближе к дорогим пользователям восточной части нашей необъятной страны, сильно не потеряв в производительности (мы потерь даже не заметили если честно, но теоретически они должны быть – физику же не обманешь).

Дальше расскажу про процесс переезда данных со старого кластера на новый. Основная проблема такого переезда состоит в том, что текущий кластер работает и данные на нём постоянно изменяются, а копирование процесс небыстрый. Всего нужно было перенести 12 ТБ данных. Чтобы не мешать нормальному функционированию других сервисов, скорость миграции была ограничена до 20 МБ/с. С учётом того, что оба кластера использовали одни и те же каналы для обмена данными между нодами, фактическая нагрузка на сеть составила порядка 40 МБ/с в каждую сторону. Процесс копирования занял порядка двух недель. Человек с калькулятором в руках воскликнет в этом месте «почему так долго». Дальше расскажу почему, а пока скажу, что процесс должен представлять из себя не просто копирование, а синхронизацию старого кластера с новым. Поэтому нужно было использовать что-то вроде всем знакомого rsync, но применимого к S3 хранилищам. И первая утилита, которой я решил воспользоваться – это стандартный cli клиент minio под название mc. У него есть команда mirror, которая позволяет сделать то, что нам нужно – отзеркалировать бакет со старого кластер в бакет на новом при первом проходе и догнать новый кластер до состояния старого при втором, ведь за время первой синхронизации, которая потребует определённого времени, данный в источнике изменятся, так как кластер живёт и обслуживает клиентов.

Собственно, я запустил зеркалирование и ушёл спать. С утра проснулся, решил посмотреть сколько же данных перелилось и увидел, что очень мало. Причиной послужил повреждённый файл в старом хранилище, который прервал синхронизацию. Когда мы только запускали минио, случилось несколько ситуаций, которые привели к появлению пары тысяч повреждённых файлов. Об этом я рассказывал в прошлый раз и с потерей этих файлов мы смирились. Поэтому первая мысль была найти такой параметр, который бы игнорировал ошибки копирования и просто отмечал бы их в логе чтобы решить, что с ними сделать, но потом. Увы, такого параметра не оказалось. Пришлось написать небольшой скрипт, который, в случае появления подобной ошибки удалял бы проблемный файл и запускал процесс заново. Копирование продолжилось, но возникла новая проблема – мы ведь не просто копируем файлы, а синхронизируем источник с приёмником. Поэтому при каждом перезапуске зеркалирования, утилита mc сначала строит список изменений, а уже затем начинает копирования. И чем больше данных переехало, тем больше времени нужно на построение списка. Меня это абсолютно не устроило, и я решил поискать альтернативу.

И нашёл. Называется она rclone. Очень удобная и мощная утилита, которая позволяет оперировать с различными S3 совместимыми хранилищами и в том числе делать синхронизацию. Помимо этого, ей так же можно задать ограничение по ширине используемого канала и по количеству одновременно копируемых объектов, подобрав таким образом оптимальные настройки под своё окружение. Я запустил rclone и снова пошёл спать. Утром я, уже привычно, взглянул на процесс копирования – он шёл и довольно успешно. Rclone пропускал битые файлы как мне и было нужно. Прошёл день. Я снова заглянул в консоль, чтобы проверить как идёт копирование. А никак. Rclone показывал, что копирование идёт, но со скоростью 5-10 байт/с и закончится оно такими темпами ой как не скоро. Я остановил процесс. Проверил, что с обоими кластерами всё хорошо и запустил процесс снова. Rclone так же, как и mc, посчитал разницу между бакетами и продолжил копирование. Прошёл ещё день. Заглядываю в консоль – та же ситуация – скорость копирования не превышает скорости черепахи.

Тут на поверхность всплывает ещё одна проблема. Идеология объектных хранилищ такова, что в них нет структуры и вложенности. Т.е. все объекты хранятся плоско в одном месте без деления на папки, как в привычной файловой системе. Папки, в понимании объектного хранилища, это такие же объекты, в которых хранятся метаданные о принадлежности к ним других объектов, которые мы воспринимаем как файлы. Различные S3 браузеры для нашего удобства показывают нам объектные хранилища в виде древовидной структуры, но под капотом это не так. И сами хранилища, такие как минио, без особых проблем справляются с таким плоским хранением, используя различные методики и алгоритмы. Но речь сейчас не об этом. Речь об утилитах, с хранилищами работающих. Как оказалось, им сложно переварить бакет, в котором лежит миллион объектов общим весом 10 ТБ, от чего они начинают захлёбываться. На наше счастье, мы используем тот же подход к хранению объектов в минио, что и при хранении файлов на диске. А именно - создаём структуру папок нужной вложенности в зависимости от имени файла. Например, если файл называется 1234567.txt, то он будет лежать по пути s3:bucket/12/34/1234567.txt. Таким образом можно копировать не весь бакет целиком, а по частям. Для этих целей я написал скрипт, который синхронизировал в цикле папки с именами от 10 до 99. В этот раз всё прошло успешно и никаких лагов при копировании не возникало.

Собственно, на данный момент мы имеем геораспределённый по 4 локациям кластер, в котором хранится 1.2 М объектов общим объёмом 11 ТБ. Суммарный полезный объём составляет 32 ТБ и может быть довольно оперативно увеличен на столько, на сколько потребуется без необходимости копирования данных. Весь кластер состоит из 96 дисков, мы можем потерять 36 из них и кластер продолжит работу на чтение и запись.

Вопросы.