



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ

Технологии хранения в системах кибербезопасности

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень

бакалавриат

(бакалавриат, магистратура, специалитет)

Форма обучения

очная

(очная, очно-заочная, заочная)

Направление(-я)
подготовки

10.05.04 Информационно-аналитические системы безопасности

(код(-ы) и наименование(-я))

Институт

Кибербезопасности и цифровых технологий (ИКБ)

(полное и краткое наименование)

Кафедра

КБ-2 «Прикладные информационные технологии»

(полное и краткое наименование кафедры, реализующей дисциплину (модуль))

Лектор

к.т.н., Селин Андрей Александрович

(сокращенно – ученая степень, ученое звание; полностью – ФИО)

Используются в данной редакции с учебного года

2024/2025

(учебный год цифрами)

Проверено и согласовано «___» _____ 2024 г.

А.А. Бакаев

*(подпись директора Института/Филиала
с расшифровкой)*

Москва 2024 г.



Технологии хранения в системах кибербезопасности

2024 год



Лекция 11.

Объектно-ориентированные, столбцовые БД.

Учебные вопросы лекции:

1. **Объектно-ориентированная БД MinIO**
2. **MinIO. Распределенный режим**
3. **Столбцовые (колоночные) БД**

MinIO

MinIO – это высокопроизводительное объектное хранилище, выпущенное под лицензией GNU Affero General Public License v 3.0. API-интерфейс совместим с облачным хранилищем Amazon S3, способное работать с неструктурированными данными, такими как фотографии, видео, файлы журналов, резервные копии и образы контейнеров, с максимальным поддерживаемым размером объекта 50 ТБ. Minio – это легкий сервис, который можно объединить с другими приложениями, аналогично Nodejs, Redis или MySQL.

У него нет мощного графического пользовательского интерфейса с дашбордами, графиками и многочисленными меню. MinIO запускает свой сервер одной командой, на котором можно хранить данные, используя всю мощь S3 API. Однако простота эта может быть обманчива, когда речь заходит об используемых ресурсах (RAM и CPU).

MinIO

Стек хранилища MinIO состоит из трех основных компонентов:

- **MinIO Server,**
- **MinIO Client,**
- **MinIO Client SDK.**

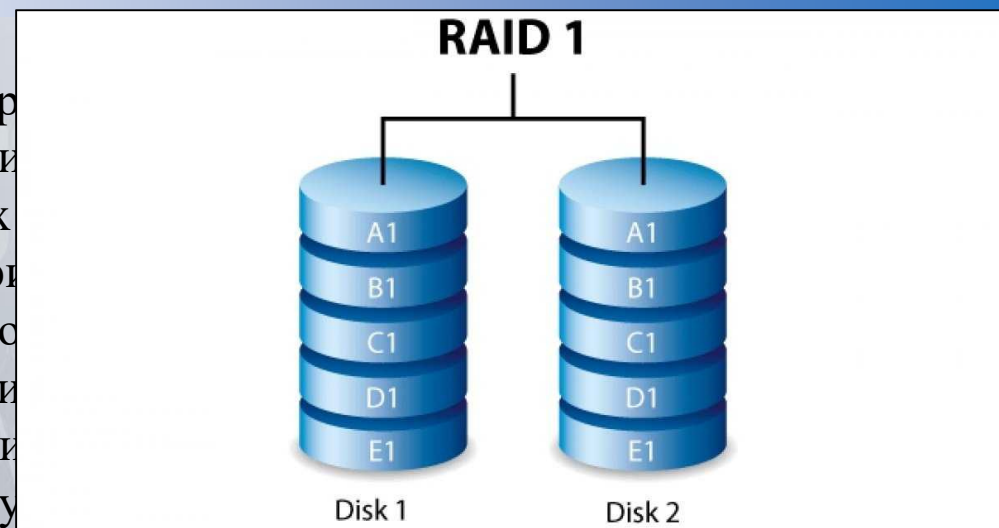
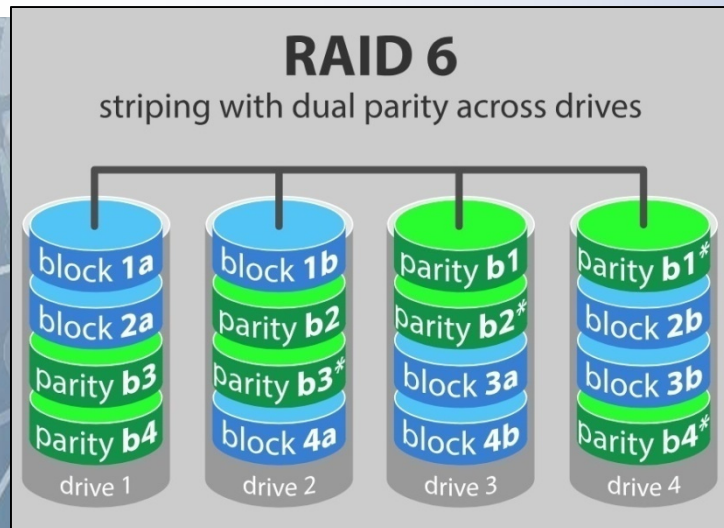
MinIO Server спроектирован так, чтобы быть минимальным и масштабируемым. Он достаточно легкий, чтобы его можно было включить в состав стека приложений, аналогично NodeJS и Redis.

MinIO Client SDK предоставляет API для доступа к любому, совместимому с Amazon S3, серверу хранения объектов. Языковые привязки доступны для Go, Java, Python, JavaScript, Haskell и языков, размещенных поверх .NET Framework.

не зависит от аппаратного обеспечения и поэтому может быть установлен как на физических, так и на виртуальных машинах или запущен как контейнеры Docker и развернут на платформах оркестрации контейнеров, таких как Kubernetes.

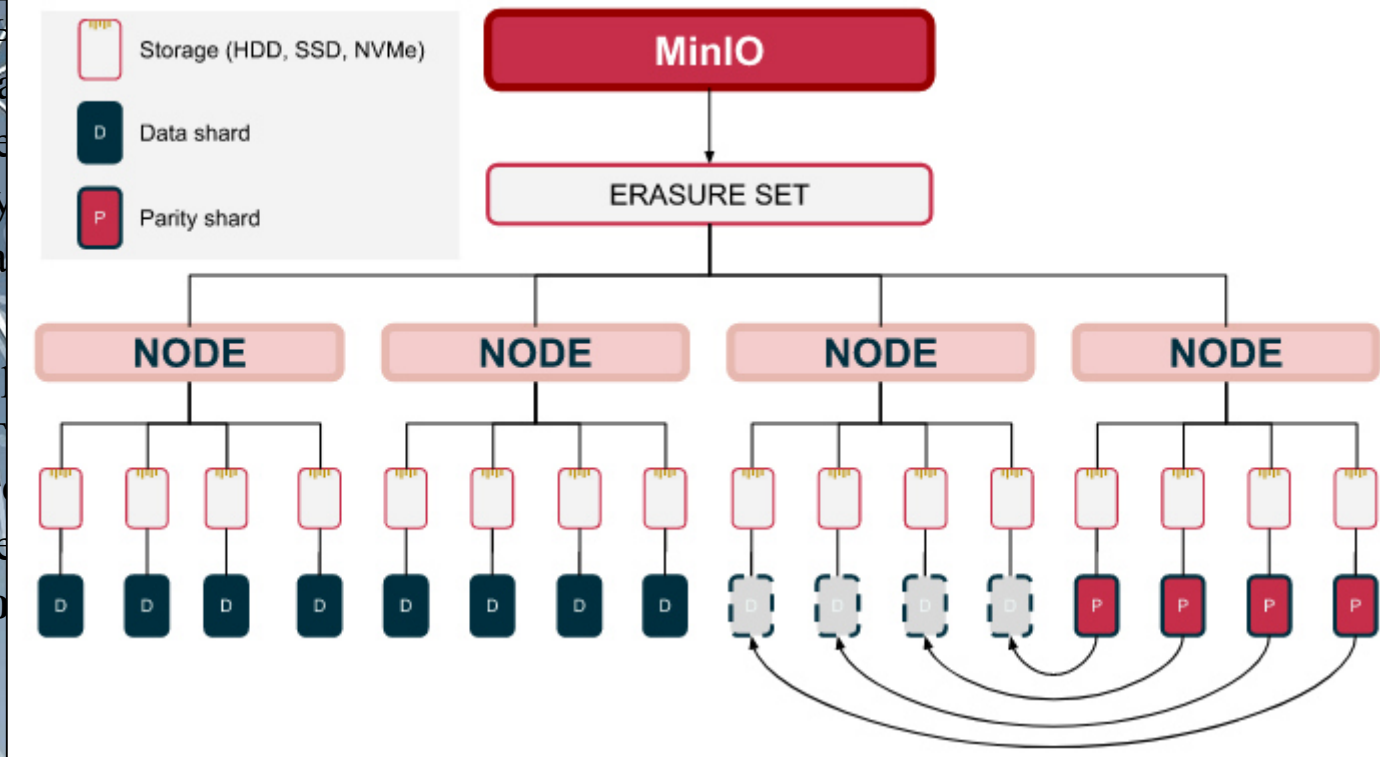
MinIO Client предоставляет альтернативу стандартным командам UNIX (например, `ls`, `cat`, `cp`, `mirror`, `diff`, и т.д.) и добавляет поддержку облачных сервисов хранения данных, совместимых с Amazon S3 и т.д. Он работает на платформах Linux, Mac и Windows

Erasure Code



Реальность
восстановления
разделяется
Поэтому
Минимум
4.

Набор
как 2 г
накопитель
Размер
размер



уровне и
0. MinIO
5 дисков.
чисел.
О, равно

строены
по 12
е.
еньшим

Высокая доступность

Распределенная установка MinIO с m -серверами и n -дисками защитит данные, пока $m/2$ серверы или $m*n/2$ или более дисков находятся в сети. При этом данные будут доступны для чтения, но для записи понадобится $m / 2 + 1$ серверов.

Например, 16-серверная распределенная установка с 200 дисками на узел будет продолжать обслуживать файлы, даже если до 8 серверов отключены в конфигурации по умолчанию, т.е. около 1600 дисков могут выйти из строя и MinIO продолжит обслуживание файлов. Но для создания новых объектов понадобится как минимум 9 серверов.

MinIO даёт возможность использовать классы хранения, чтобы установить пользовательское распределение четности для каждого объекта

Bit Rot защита

Защита от порчи данных. MinIO использует высокоскоростные контрольные суммы HighwayHash для защиты от Bit Rot.

Bit rot - это бесшумное повреждение данных в результате случайных изменений на уровне носителя. Для накопителей данных это, как правило, является результатом ослабления электрического заряда или магнитной ориентации, которые представляют данные. Причиной возникновения могут варьироваться от небольшого скачка тока во время отключения электроэнергии до случайного космического луча, приводящего к переворачиванию битов. Возникающий в результате “битовый сбой” может привести к незначительным ошибкам или повреждению носителя данных без запуска инструментов мониторинга или аппаратного обеспечения. Оптимизированная реализация алгоритма HighwayHash в MinIO гарантирует, что он перехватывает и лечит поврежденные объекты на лету.

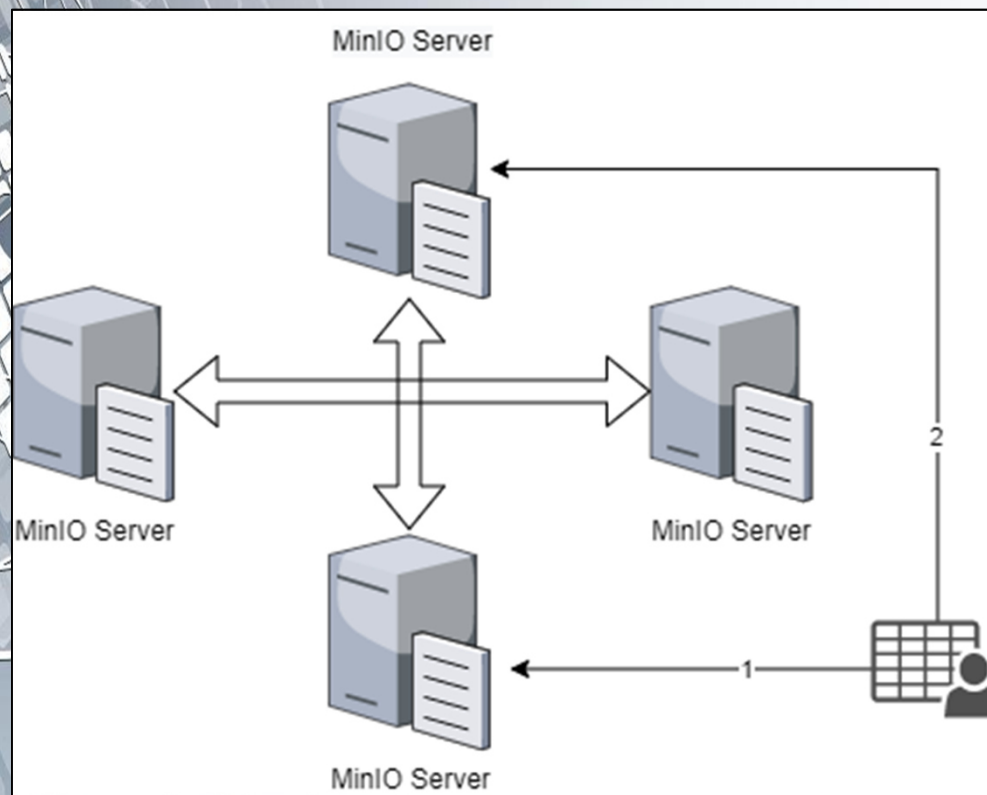
Целостность обеспечивается от начала до конца путем вычисления хэша при чтении и проверке его при записи из приложения, по сети и в память/на диск. Реализация рассчитана на скорость и может достигать скорости хэширования более 10 ГБ/сек на одном ядре процессоров Intel.

Преимущества MinIO:

- Высокая производительность! MinIO способен читать/записывать со скоростью ~ 170 ГБ /с.
- Масштабируемость – используйте кластеризацию и масштабируйте по мере необходимости
- Cloud-native
- Защита данных с использованием метода Erasure code
- Поддерживается множественное шифрование, включая AES-CBC, AES-256-GCM, ChaCha20
- Совместим с обычным KMS
- Уведомление о событии
- Совместим с использованием etcd и CoreDNS

MinIO - распределенный режим (Distributed Mode)

В распределенном режиме подразумевает использование 4 или более дисков (каталогов), которые могут быть отданы одной ноде либо нескольким. При этом запускать всё на одной ноде не рекомендуется. Каждая нода **MinIO** в распределенном режиме имеет полную информацию о других нодах, поэтому приложение может подключаться к любой ноде.



MinIO - Распределенный режим (Distributed Mode)

В распределенном режиме автоматически включается опция *erasure code*, которая дублирует ваши данные, обеспечивая возможность не потерять их при выходе из строя нод либо дисков.

Данную функцию можно сравнить с RAID, но *Erasure coding* обеспечивает восстановление на уровне объекта с меньшими затратами.

В зависимости от того, какая четность *erasure code* настроена, можно продолжать операции чтения и записи только с $m/2$ серверами или $m*n/2$ дисками, доступными и подключенными к сети.

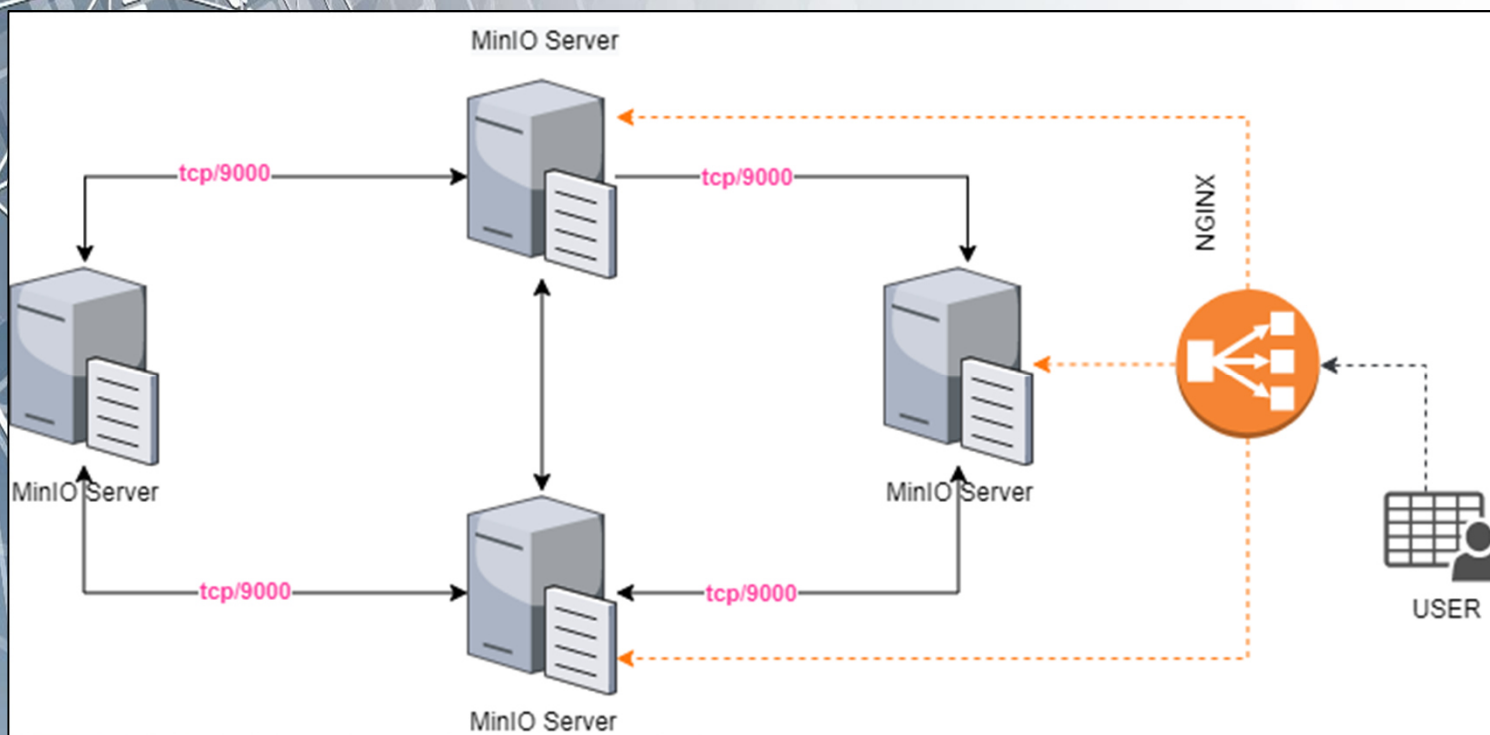
Распределенный режим также поддерживает следующие функции:

- репликация объектов;
- блокировка однократной записи и многократного чтения;
- версионность объекта.

Совместно с **MinIO** рекомендуют использование балансировщика нагрузки для управления подключением к кластеру (NGINX).

MinIO - Распределенный режим (Distributed Mode)

Каждая нода должна иметь доступ к другим нодам по *API*-порту. Также все серверы MinIO должны быть запущены с использованием одинаковых портов на всех нодах.

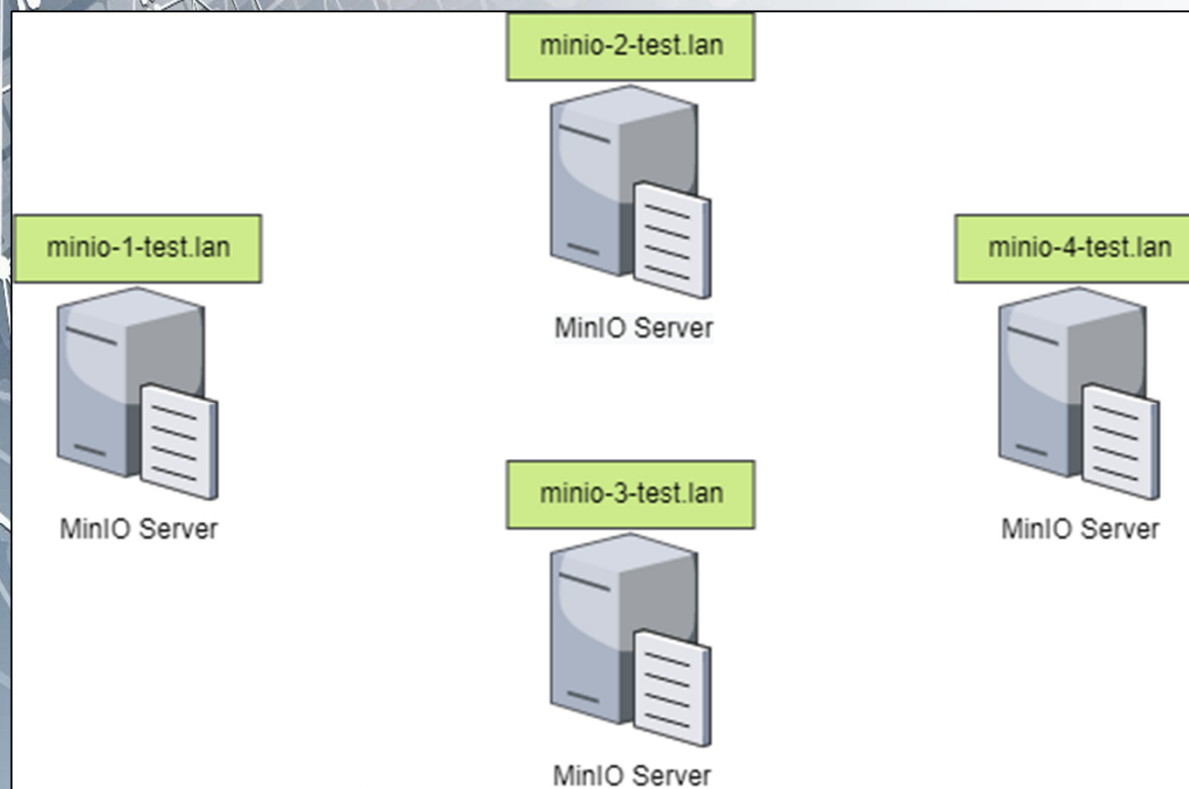


MinIO - Распределенный режим (Distributed Mode)

Для того чтобы развернуть MinIO в распределенном режиме, имена серверов должны иметь такие названия, чтобы можно было использовать регулярные выражения для передачи этих имён одной строкой, создавая таким образом пул серверов (server pool).

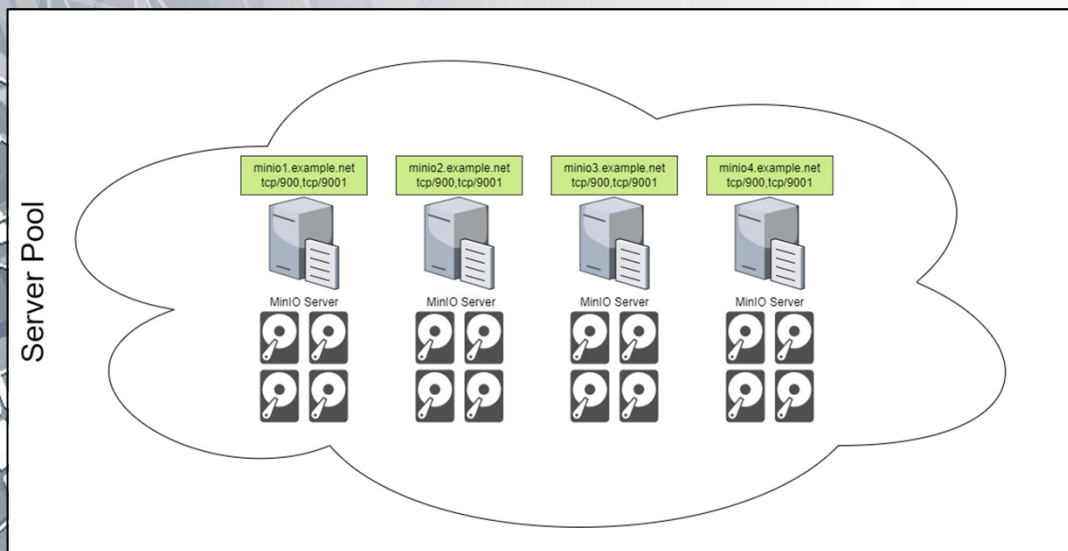
Например, для имен серверов: minio-1-test.lan, minio-2-test.lan, minio-3-test.lan, minio-4-test.lan можно легко перечислить их одной строкой типа:

minio-{1...4}-test.lan.



MinIO - Распределенный режим (Distributed Mode)

При использовании *MinIO* рекомендуется использовать голые диски (*JBOD*) с типом файловой системы *XFS*.



По информации с официального сайта, использование файловых систем, отличных от *XFS* как правило, приводит к более низкой производительности.

Ещё одна рекомендация – это использование одинаковых дисков во всех нодах пула серверов (тип жесткого диска и его размер), так как *MinIO* ограничивает размер используемого диска до самого маленького используемого диска. Например, если в ноде есть 15 дисков по 10 ТБ и 1 диск по 1 ТБ, *MinIO* ограничивает емкость каждого диска до 1 ТБ, по сути, как и СХД.

На сайте *min.io* сказано, что строго рекомендуется использовать сервера с одинаковым аппаратным обеспечением.

Столбцовые (колоночные) БД

В столбцовой БД данные каждого столбца хранятся отдельно (независимо) от других столбцов. Это означает, что с точки зрения SQL-клиента данные представлены как обычно в виде таблиц, но физически эти таблицы являются совокупностью колонок, каждая из которых представляет собой таблицу из одного поля. При этом физически на диске значения одного поля хранятся последовательно друг за другом: [X1, X2, X3], [Y1, Y2, Y3], [Z1, Z2, Z3] и т.д.

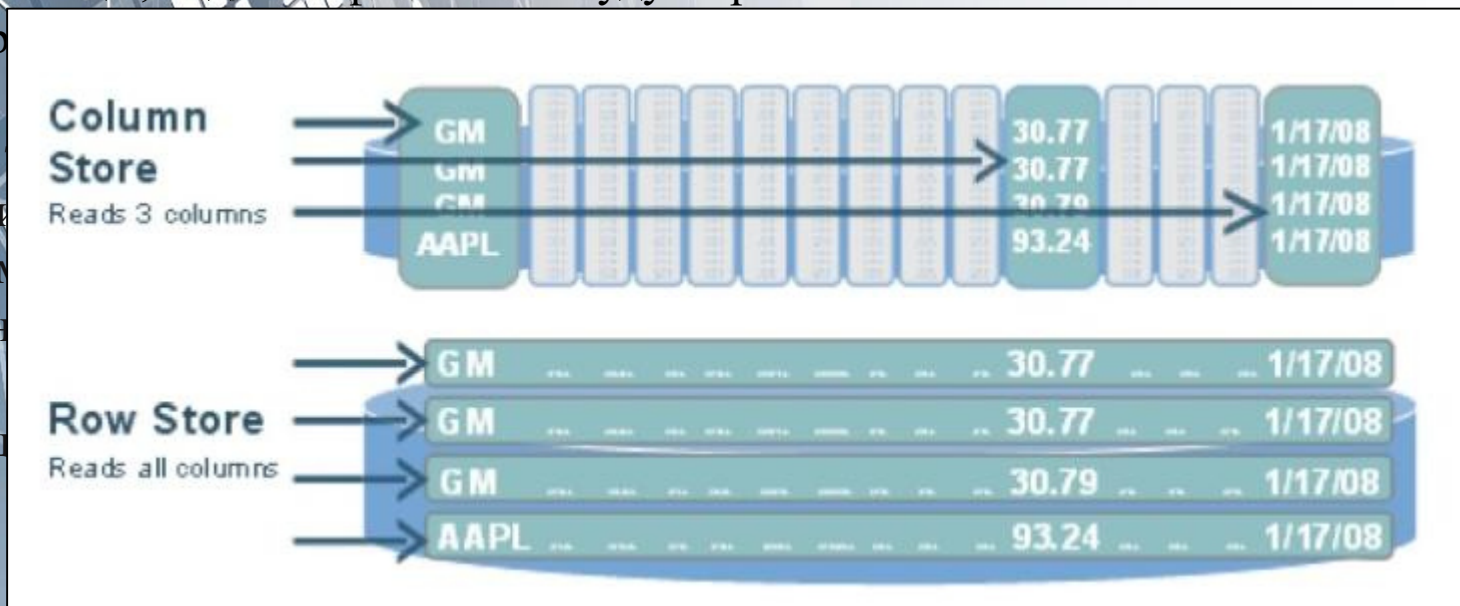
Такой принцип хранения позволяет при выполнении запроса считывать с диска данные только тех столбцов, которые непосредственно участвуют в этом запросе.

Например, при выполнении «select», в котором фигурируют только 5 поля из 100 полей таблицы, с диска физически будут прочитаны только 5 колонок. Это означает

что нагрузка меньше
сторона
строками

Преимущества:

- выполнение запросов сразу;
- агрегация;
- сжатие



0) раз
атная
й над
лицей

Столбцовые (колоночные) БД

Пример извлечения данных для отчетов при использовании обычной строковой СУБД и столбцовой СУБД:

Строковая СУБД

The diagram illustrates a timeline with three labeled points: Time, Location, and Mobile Phone. A red horizontal line spans the width of the timeline grid.

Столбцовая СУБД

Столбцовые (колоночные) БД

Колоночные СУБД призваны решить проблему неэффективной работы традиционных СУБД в аналитических системах и системах в подавляющем большинстве операций типа «чтение». Они позволяют на более дешевом и маломощном оборудовании получить прирост скорости выполнения запросов в 5, 10 и иногда даже в 100 раз, при этом, благодаря компрессии, данные будут занимать на диске в 5-10 раз меньше, чем в случае с традиционными СУБД.

У колоночных СУБД есть и недостатки – они медленно работают на запись, не подходят для транзакционных систем и как правило, ввиду «молодости» имеют ряд ограничений для разработчика, привыкшего к развитым традиционным СУБД.

Колоночные СУБД применяются как правило в аналитических системах класса business intelligence (ROLAP) и аналитических хранилищах данных (data warehouses). Причем объемы данных могут быть достаточно большими – есть примеры по 300-500ТБ и даже случаи с ≥ 1 ПБ данных.

ClickHouse – типичный пример столбцовой СУБД.

ClickHouse

ClickHouse – СУБД с открытым исходным кодом, построенная на основе колонок. Она стала широко популярной среди ИТ-организаций благодаря своим способностям по быстрой обработке данных и масштабируемости.

Высокопроизводительная обработка запросов в ClickHouse делает ее идеальным выбором для работы с большими объемами данных и оперативной аналитики.

Архитектура ClickHouse состоит из механизма обработки данных по столбцам, репликации на основе дерева объединения (Merge Tree), распределенного выполнения запросов и других архитектурных решений, обеспечивающих надежную систему.

В механизме обработки данных информация хранится в виде небольших частей, называемые «чанками», и каждый чанк содержит информацию о нескольких колонках. Они обрабатываются с помощью векторных вычислений. Это означает выполнение операций над значениями данных в виде массивов или векторов, вместо их индивидуальной обработки. Данный метод позволяет эффективно и параллельно обрабатывать большие объемы данных, что улучшает производительность и скорость выполнения запросов. Общая стоимость обработки данных уменьшается, что позволяет успешно разворачивать ClickHouse на серверах среднего уровня производительности.

Кроме того, благодаря встроенным возможностям по репликации повышается отказоустойчивость системы, улучшается балансировка нагрузки и обеспечивается распределенное выполнение запросов для ускорения их обработки.

Особенности ClickHouse



Ключевым преимуществом ClickHouse считается **высокая скорость выполнения SQL-запросов** на чтение (OLAP-сценарий), которая обеспечивается благодаря следующим особенностям:

- **столбцовое хранение данных** позволяет считывать данные только из нужных колонок и эффективно сжимать однотипную информацию;
- **физическая сортировка данных по первичному ключу** позволяет быстро получать конкретные значения или диапазонов;
- **векторные вычисления** по кусочкам столбцов снижают издержки на диспетчеризацию и позволяют более эффективно использовать CPU;
- **распараллеливание операций** как в пределах одного сервера на несколько процессорных ядер, так и в рамках распределенных вычислений на кластере за счет механизма шардирования;
- **поддержка приближенных вычислений** на части выборки, что снижает число обращений к жесткому диску и еще больше повышает скорость обработки данных.

Clickhouse. Примеры использования

Возможности быстрой обработки данных и аналитический механизм ClickHouse позволяют использовать его для обработки потоковых данных и создания дашбордов в реальном времени. К числу наиболее распространенных вариантов использования ClickHouse относятся:

Аналитика и бизнес-аналитика: Построение дашбордов и пользовательских отчетов для реализации корпоративных стратегий и принятия бизнес-решений.

Обработка журналов и событий: Обработка журналов и событий для быстро меняющихся систем, таких как социальные медиа и игровая индустрия.


Хранение и анализ данных IoT: Хранение и обработка “сырых” данных от устройств IoT в реальном времени. Обработанные данные могут использоваться для анализа, машинного обучения или инициирования дальнейших действий.

Анализ данных потока кликов (clickstream): Сбор и хранение информации о поведении посетителей на веб-сайтах. К ним относятся нажатые кнопки и просмотренные страницы. Эта информация важна для целевого маркетинга и бизнес-анализа.

Обработка данных временных рядов: Обработка потока данных для временного анализа. Это помогает создавать финансовые отчеты для бизнес-прогнозирования или приложений машинного обучения, таких как прогнозирование цен на акции.

Недостатки ClickHouse

ClickHouse не лишена недостатков. Для начинающих пользователей эта система может быть сложной для понимания. Колоночный формат хранения данных делает ClickHouse полезной только в ситуациях, связанных с OLAP. К другим недостаткам относится ограниченная поддержка SQL, например, невозможность использования JOINS (для объединения данных из разных таблиц на основе определенных условий) и неэффективные операции UPDATE и DELETE (ClickHouse реализует подход "вставка и забывание", поощряя добавление новых данных, но не всегда обеспечивая эффективное изменение или удаление существующих записей).



ClickHouse vs. ElasticSearch

ClickHouse и ElasticSearch – это две разные системы для хранения и анализа данных.

ClickHouse специализируется на быстрой агрегации и анализе больших объемов данных, в то время как ElasticSearch ориентирован на поиск и анализ текстовых данных.

ElasticSearch – это распределенная система индексации и поиска текстовых данных. Она используется для поиска, агрегации и анализа данных с использованием мощных функций поиска и фильтрации.

Кроме того, в ElasticSearch есть механизмы для обработки и анализа неструктурированных данных, таких как логи, что позволяет делать многое, что не может быть сделано в ClickHouse.

Таким образом, ClickHouse и ElasticSearch оба используются для обработки данных, но они имеют различные процессы и использование в разных настройках может дать более эффективные результаты.

Заключение

Высокая скорость обработки и распределенные вычисления дают возможность обрабатывать большие объемы данных. Распределенные вычисления позволяют использовать несколько серверов и ресурсов для обработки данных одновременно, что обеспечивает оперативное получение результатов и анализ данных в реальном времени, без больших задержек.

Однако эта система баз данных сложна в освоении и имеет ряд ограничений, связанных с ее использованием в качестве БД общего назначения. Хотя ClickHouse отлично подходит для чтения больших объемов данных, ее производительность значительно снижается при частых операциях обновления и запросах, использующих целые строки.

СПАСИБО ЗА ВНИМАНИЕ!

