HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY

Виртуальная файловая система для ClickHouse

Ершов Олег Владиславович, студент БПМИ 165
Научный руководитель: доцент, Миловидов Алексей Николаевич

Актуальность задачи

- Глобальный тренд на хранение и обработку огромных объемов данных
- Hadoop — свободно распространяемый набор утилит и библиотек для разработки и выполнения распределенных программ
- ClickHouse — высокопроизводительная аналитическая СУБД с открытым исходным кодом

Цели и задачи дипломной работы

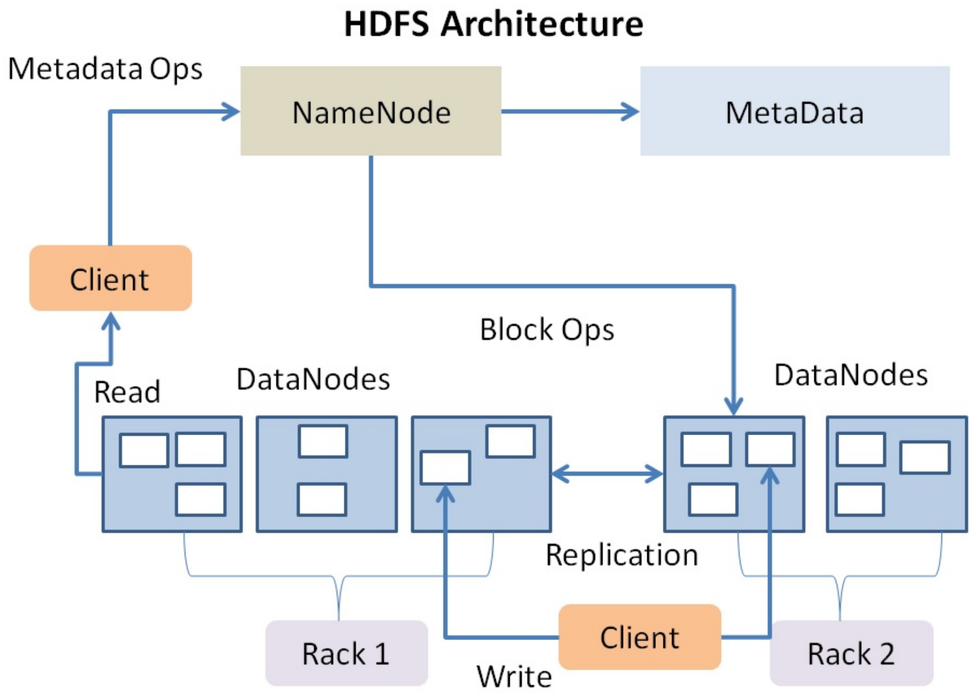
- Изучить существующие решения по использованию распределенных систем в качестве хранения данных
- Изучить архитектуру Hadoop Distributed Filesystem, ее особенности и ограничения
- Разобрать внутреннее устройство ClickHouse, его интерфейсы по работе с файловой системой
- Спроектировать и реализовать программное решение для работы ClickHouse над HDFS

Обзор существующих методов

- Hadoop FUSE-DFS — стандартный модуль, который позволяет монтировать HDFS как стандартную файловую систему
- HBase — распределенная СУБД класса NoSQL с открытым исходным кодом, работает поверх HDFS
- NoSQL база данных Amazon DynamoDB
- Amazon FSx for Lustre - сервис для работы с высокопроизводительно файловой системой Lustre

Обзор HDFS

- NameNode — центральный узел имен, хранит метаданные файловой системы и расположение блоков
- DateNode — основные узлы, хранят блоки и отвечают на запросы пользователей или NameNode

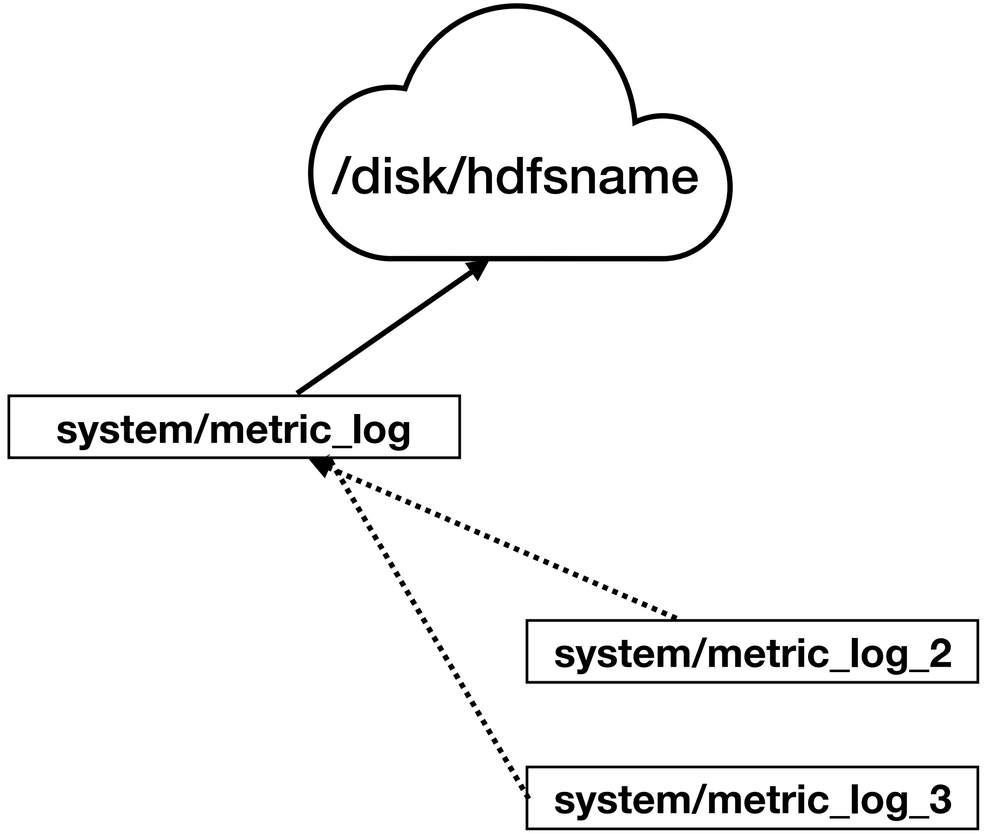


Предложенный метод

- Используется интерфейс для взаимодействия с файловой системой IDisk
- Локально для каждого объекта хранится лишь файл с метаинформацией
- Взаимодействие с HDFS осуществляется через библиотеку libhdfs3

Структура метаинформации

- *String* **disk_path** - путь до каталога диска в локальной файловой системе
- *String* **metadata_file_path** - относительный путь до метаинформации в локальной файловой системе
- *size_t* **total_size** - размер объекта в HDFS
- *String* **hdfs_object** - путь объекта в HDFS
- *UInt32* **ref_count** - счетчик жестких ссылок на наш путь



Методы интерфейса IDisk

- создание, удаление и перемещение файлов
- создание, удаление и перемещение директорий
- установка и возвращение информации о временной метке последних изменений
- создание жёстких ссылок на файлы и директории
- создание интерфейсов для чтение или записи в файл ReadBufferFromFile и WriteBufferFromFile

Интерфейсы ввода-вывода

- Для побайтового ввода/выводы существуют специальные абстрактные классы ReadBuffer и WriteBuffer, которые являются улучшенной заменой std::iostream

- ReadBuffer и WriteBuffer - это буфер ограниченного размера и курсор, указывающий на текущую позицию в нем. Для всех наследников достаточно переопределить виртуальный метод **nextImpl**

- Для корректной работы MergeTree должен быть поддержан **SEEK_CUR** и **SEEK_SET**

BufferBase

Position — **char ***, Buffer — структура из *Position* **begin**, *Position* **end,**
Memory — замена std::vector<char> в буферах

- *Position* **pos** — позиция чтения и записи
- size_t **bytes** — счетчик сколько байт уже было прочитано
- *Buffer* **working_buffer** — ссылка на участок памяти, с которым можно работать
- *Buffer* **internal_buffer** — ссылка на участок памяти, где хранятся данные
- *Memory* **memory** — участок памяти, которым владеет буфер

ReadIndirectBufferFromHDFS

- Дополнительно храним **absolute_position** — текущая позиция относительно начала файла
- При операции **seek** сначала проверяем, что результат находится внутри текущего буфера, иначе перечитываем данные с новой позиции
- За счет этого поддерживаем как **SEEK_CUR**, так и **SEEK_SET**

Результаты работы

- Произведен обзор текущих решений по работе с распределенными хранилищами
- Изучена архитектура основных используемых систем, а именно HDFS и ClickHouse
- Спроектировано и реализовано программное решение для работы ClickHouse над HDFS

https://github.com/ClickHouse/ClickHouse/pull/11058

Спасибо

Олег Ершов, *overshov@edu.hse.ru*