Интеллектуальные информационные системы

СУБД HBase

Кафедра управления и интеллектуальных технологий НИУ «МЭИ»

2023 г.

СУБД HBase

Распределенная NoSQL СУБД.

- Column-oriented хранилище данных
- Нет фиксированной структуры колонок – произвольное количество колонок для одного ключа.
- Поддержка больших таблиц: 10^9 строк и 10^6 колонок

- Не предоставляет SQL-доступ
- Не поддерживает реляционную модель
- Предоставляет возможность «произвольного» чтения и записи

https://hbase.apache.org/

СУБД HBase

- Основана на идеях Google BigTable

- Работает на кластере серверов поверх Hadoop

- Абстрагирует распределенность.
- Легко горизонтально масштабируется за счет шардинга

- Работает автоматическая обработка отказов оборудования
- Интеграция с MapReduce

https://hbase.apache.org/

Когда использовать HBase

- Большие объемы данных. Иначе – лучше РСУБД
- Выборка значений по заданному ключу или по диапазону ключей
- Свободная схема данных:
  - Строки могут существенно отличаться по своей структуре.
  - Динамическое изменение структуры
  - Большое количество столбцов, возможно – разреженных (NULL)

Когда НЕ использовать HBase

- Традиционный доступ к данным в стиле РСУБД
- Приложения с транзакциями
- Реляционная аналитика (joins, group by, where column like,…)
- Фильтрация данных (where) есть, но плохо поддерживается строковый поиск ‘LIKE %text%’

Модель данных

Key	CF1:C1	CF1:C2	CF2:C1	CF2:C2
1	Some text	7	True	
2		10	False	
3	For example	1234		
4	Not empty		False	
5		11	True	
6		52		

Модель данных

- Ключ представляются массивом байт, сравниваются лексикографически.

- Каждая «строка» таблицы делится на column family (CF) – семейство колонок.
- Семейство колонок состоит из непосредственно колонок, в которых хранятся значения.

- Каждая ячейка хранит несколько последних версий (задается). Нужно для смягчения проблемы согласованности данных

Модель данных

- Column Family описывает общие свойства колонок:
  - Сжатие
  - Количество версий
  - Хранение In-memory
- Column Family хранится в отдельном файле (HFile).

Пример:

- CF1 – непосредственно данные о просмотре сайта (user, site)
- CF2 – метаданные – cookie, проведенное время

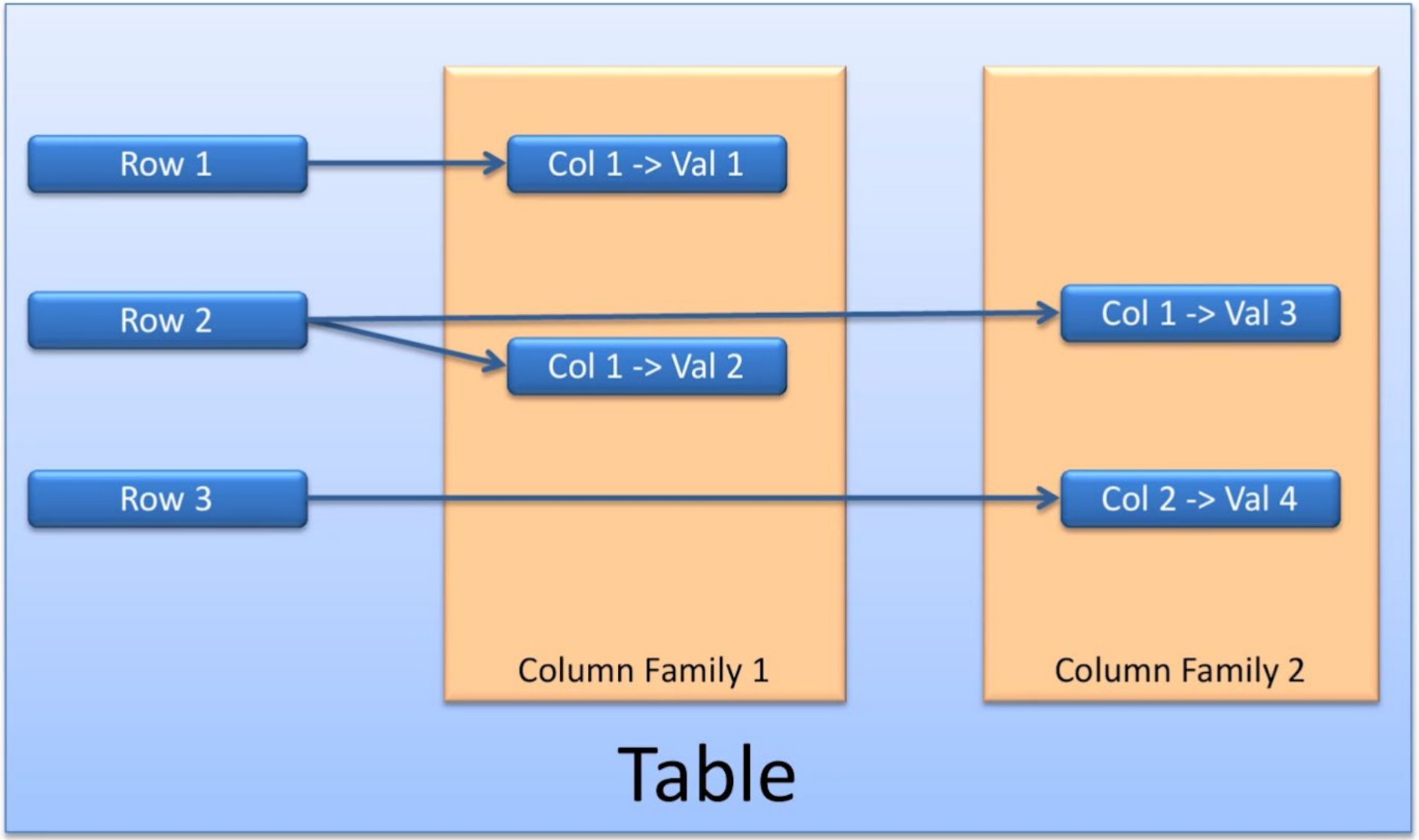
- CF1 – стоимость товара, скидка, стоимость доставки
- CF2 – характеристики товара (цвет, объем, частота CPU, срок годности,…)
- CF3 – наличие у каждого из продавцов

Column Family

- Конфигурация CF – статична, задается при создании таблицы.
- Может быть изменено при отключении БД.
- Количество CF ограничено небольшим числом (до 10)

- Колонки внутри CF – динамичны, создаются в runtime.
- В одной CF может быть сотни тысяч колонок

Column Family



HBase Timestamps

Каждая ячейка хранит несколько последних версий значений (по умолчанию 3, настраиваются в конфигурации CF)

Данные имеют timestamp – задаются неявно при записи, но могут указываться вручную.

Версии хранятся в порядке убывания timestamp.

Value = Table + RowKey + CF + Column + Timestamp

Value = Table + RowKey + Column - в РСУБД

HBase Пример

Row Key	Timestamp	CF: “Data”		CF: “Meta”		
		Url	Html	Size	Date	Log
row1	t1	Mail.Ru				Log text 1
	t2				123456	Log text 2
	t3		<html>…	1234		Log text 3
	t4		<HTML>…	2345		Log text 4
row2	t1	OK.Ru			123765	Log text 1
	t2					Log text 2

HBase Пример

09.01.2023 22:19: Big Data
10.01.2023 10:15: mpei.ru
11.01.2023 17:23: Hadoop
11.01.2023 21:40: MapReduce
01.02.2023 12:05: HBase

При разборе логов, оказалось, что запрос за 10.01 не сохранился в таблице (произошла ошибка).
Ошибку исправили и записали 13.01.

Что будет выведено 02.02 в качестве 3 последних версий ячейки?

Архитектура HBase

Key	Column1	Column3	TimeStamp
Row1	value1		timestamp1
		value2	timestamp2
Row2	value4		timestamp1
Row3	value1	value6	timestamp1

HFile

Row1:ColumnFamily:column1:timestamp1:value1
Row1:ColumnFamily:column3:timestamp2:value3
Row2:ColumnFamily:column1:timestamp1:value4
Row3:ColumnFamily:column1:timestamp1:value1
Row3:ColumnFamily:column3:timestamp1:value6

KeyValues

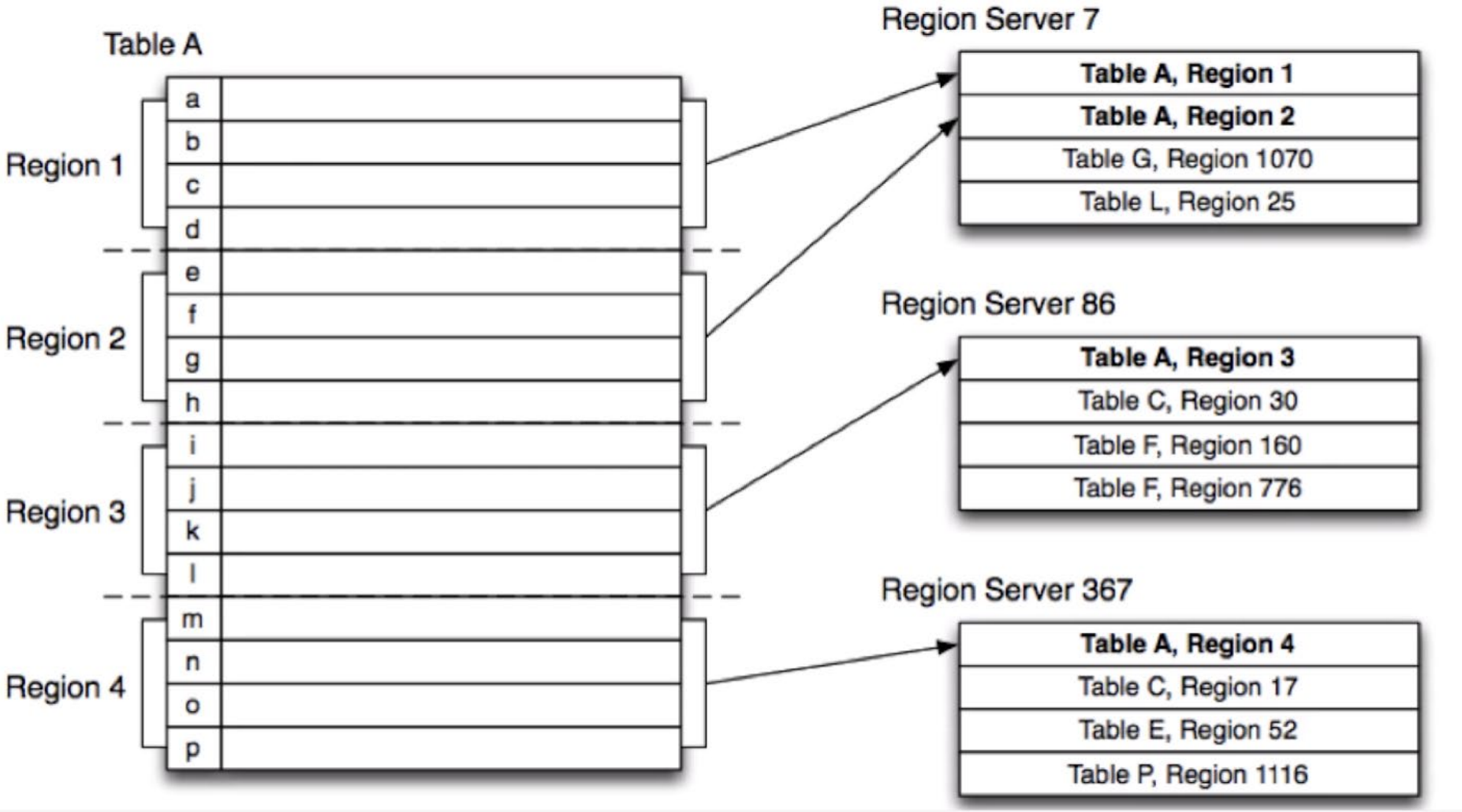
Масштабируемость

Таблица делится на регионы
Регион – группа строк, которые хранятся вместе.

- Единица шардинга
- Может делиться пополам

RegionServer – демон, который управляет регионами (одним или несколько, причем, возможно, из разных таблиц) на узле.
При этом, регион принадлежит только одному RegionServer
Hmaster (Master Server) – демон, который управляет всеми RegionServer

Масштабируемость

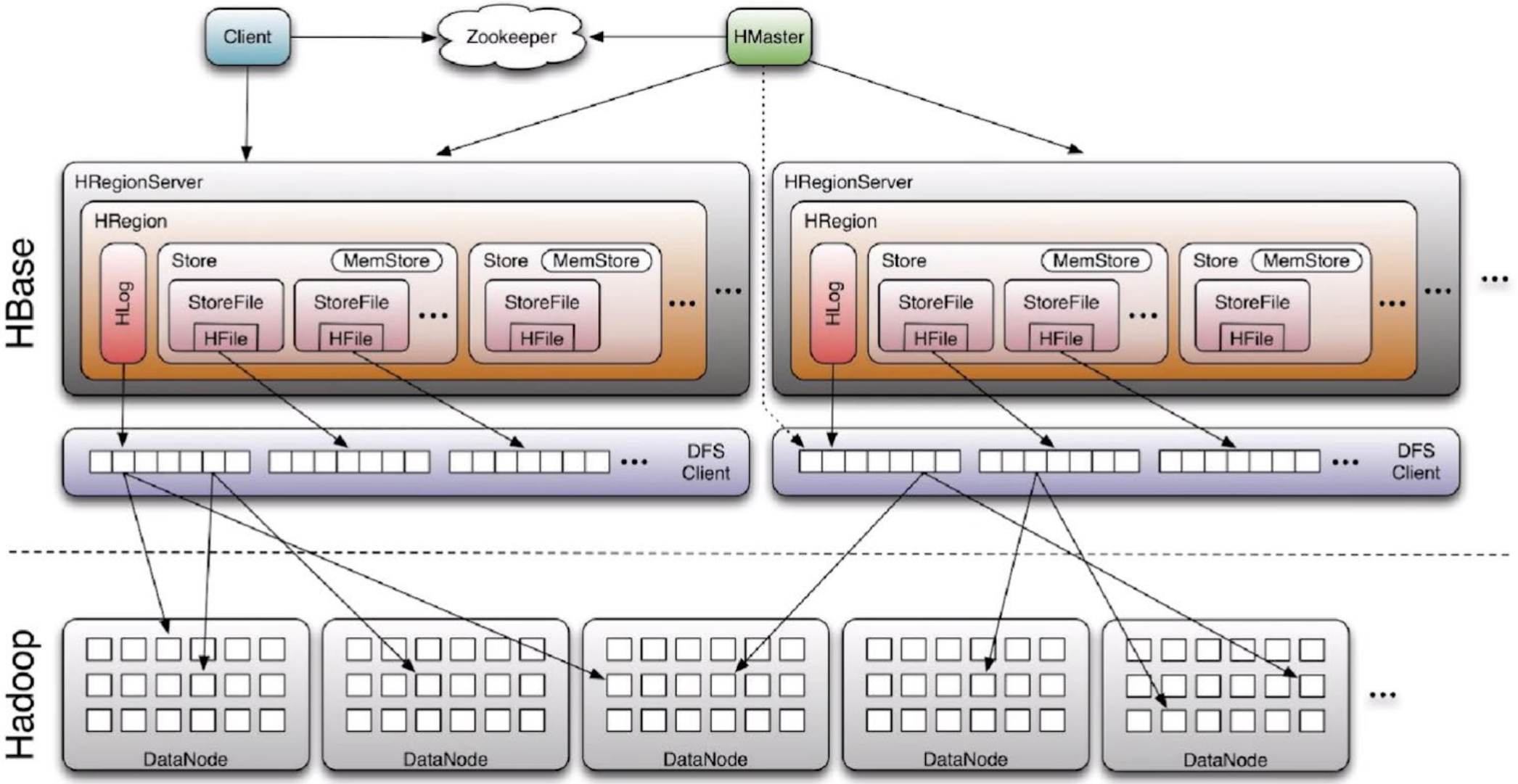


Регионы

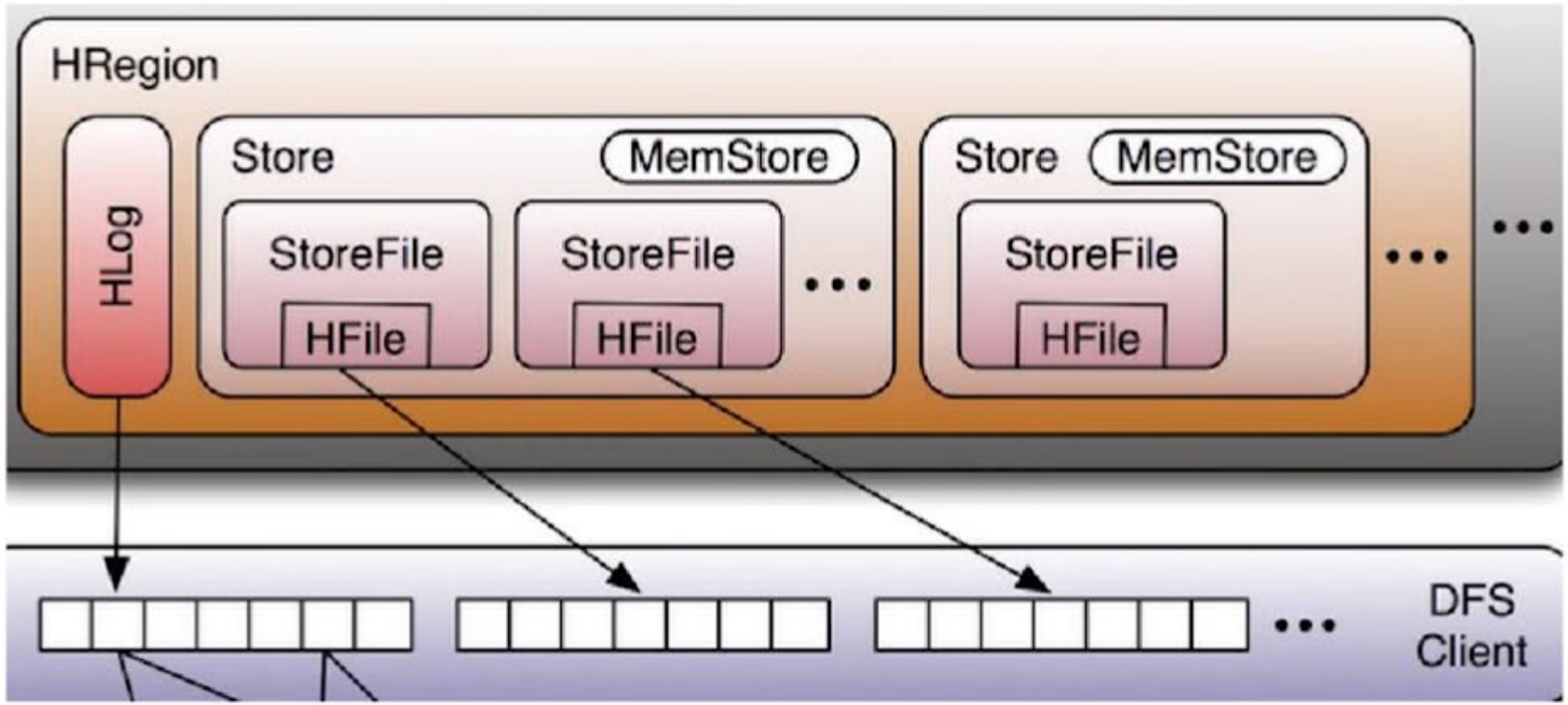
- Регион определяется диапазоном ключей: [Start key; Stop key).
- Исходно, по умолчанию, имеется только один регион.
- При превышении лимита, регион разбивается на 2 части. Split – быстрая операция. Часто, лучше производить вручную в случае активной работы с данными.

- Регионы сбалансированы по размеру внутри таблицы.
- HBase старается балансировать нагрузку на RegionServer.

Архитектура HBase



Хранение данных



Хранение данных

- **HLog** – хранение истории изменений.
- Каждое хранилище **Store** отвечает за свою CF. На слайде – 2 CF, которые физически хранятся в Hfile.
- **MemStore** – хранилище в памяти – хранятся данные, которые еще не записаны в Hfile. При заполнении MemStore проводится сортировка данных по ключу и выгрузка в новый HFile.

- При удалении – ключу присваивается специальный флаг. Если при дальнейшем чтении натыкаемся на флаг, то по ключу не возвращаем значения. Физически данные остаются на диске и занимают место.

Compaction

Minor Compaction

- Маленькие файлы объединяются в большие
- Быстрая операция
- Маркеры удаления не применяются

Major Compaction

- Для каждого региона все файлы в рамках одной CF объединяются в один файл
- «Удаленные» данные удаляются физически.
- Долго.

Запрос данных из HBase

- По RowId или набору - если не заданы доп.критерии, то отдаются все ячейки из всех CF заданной таблицы – т.е. слияние множества Hfiles

- По ColumnFamily / ColumnName – только нужные файлы

- По TimeStamp – только файлы, в которых содержатся заданные timestamp

- По Value – может быть медленно, т.к. проверка каждой ячейки
- Множественные фильтры для каждого критерия (CF, Column, Value,…

Примеры запросов

— **Put**: добавить новую запись в hbase. Timestamp этой записи может быть задан вручную, в противном случае он будет установлен автоматически как текущее время.

— **Get**: получить данные по определенному RowKey. Можно указать Column Family, из которой будем брать данные и количество версий которые хотим прочитать.

— **Scan**: читать записи по очереди. Можно указать запись с которой начинаем читать, запись до которой читать, количество записей которые необходимо считать, Column Family из которой будет производиться чтение и максимальное количество версий для каждой записи.

— **Delete**: пометить определенную версию к удалению. Физического удаления при этом не произойдет, оно будет отложено до следующего Major Compaction.

Примеры запросов

- **list** – перечислить имеющиеся таблицы
- **describe** – описание таблицы
- **create** – создать таблицу

  > create ‘TABLENAME’, ‘COLUMNFAMILY’

- **put** – записать строку
- **scan** – просмотр таблицы
- **disable\enable** – отключить\включить таблицу для проведения работ
- **alter** – изменение структуры таблицы и ее параметров
- **drop** – удалить таблицу (сначала нужно отключить)

Примеры запросов

$ > hbase shell --- Заходим в интерактивную облочку

…

hbase(main):010:0> create 'newtbl', 'CF1' ---- Создаем таблицу newtbl с ColumnFamily CF1

0 row(s) in 1.2250 seconds

=> Hbase::Table - newtbl

hbase(main):002:0> put 'newtbl', 'vk.com', 'CF1:timevisit', ' 1234' -- Добавляем строку

hbase(main):003:0> scan 'newtbl' **----** Выводим таблицу

vk.com column= 'CF1:timevisit, timestamp=1701751451048, value= 1234

2 row(s) in 0.0100 seconds