HBase: BigTable en Hadoop

Bases de Datos No Relacionales

Instituto de Computación, FING, UdelaR – 2021

CC-BY Lorena Etcheverry lorenae@fing.edu.uy

Agenda

- ¿Qué es HBase?
- ¿Cómo se usa HBase?
- ¿Cuándo usar HBase?

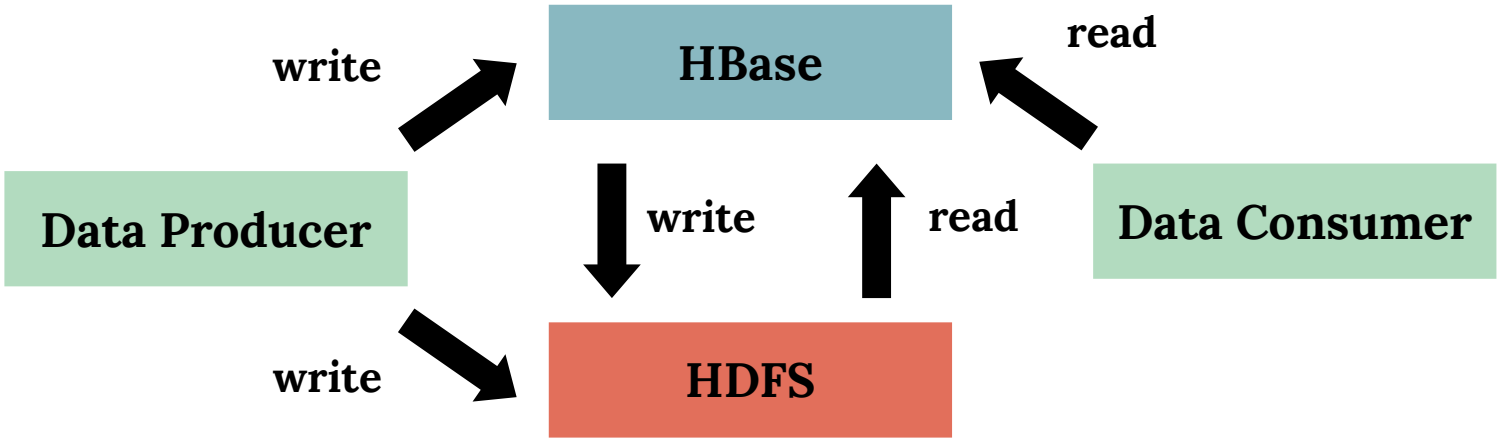
¿Qué es Hbase?

"The canonical HBase use case is the **webtable**, a table of crawled web pages and their attributes (such as language and MIME type) **keyed by** the web page URL.

The webtable is **large**, with row counts that run into the billions.

Batch analytic and parsing MapReduce jobs are continuously run against the webtable, deriving statistics and adding new columns of verified MIME-type and parsed-text content for later indexing by a search engine.

Concurrently, the table is **randomly accessed** by crawlers updating **random rows** while random web pages are served in real time as users click on a website's cached-page feature." [2]



[2] Hadoop: The Definitive Guide, Tom Gray. (2015, 4th edition)

HBase es ...

*"Una **base de datos** open source, distribuída y no-relacional modelada a semejanza de Google BigTable[1]"*
(Wikipedia)

- Surgió como parte del proyecto Apache Hadoop y usa HDFS.
- Puede pensarse como un ***sorted map*** disperso (poco denso), distribuido, consistente y multidimensional.
  - conjuntos de **filas**
  - cada fila contiene **celdas** (parejas **key-value**)

Más detalles sobre el modelo de datos en un rato!

[1] http://research.google.com/archive/bigtable.html

HBase no es ...

- Una base de datos relacional.
- Una base de datos que soporte consultas SQL.
- No tiene ***joins.***
- No tiene un lenguaje de consultas en alto nivel.
- No soporta transacciones en forma nativa.
- No soporta índices secundarios en forma nativa.

NO es un reemplazo directo al RDBMS!

Algunas características

- Lecturas y escrituras **parciales** (*random*) sobre HDFS.
  - En contraposición a hacer *append* en archivos
- **Escalabilidad**.
- **Particionamiento** (*sharding*) automático de las tablas.
- Tolerancia a fallas.
- Lecturas y escrituras **consistentes a nivel de fila**.
- ...

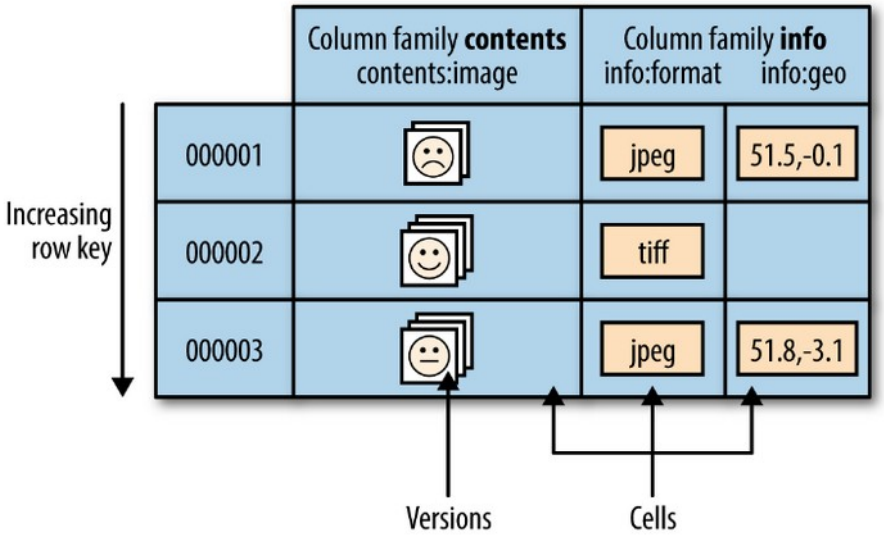
El modelo de datos

Fila: pareja **key/value**, con una clave por fila.

Celdas: valores para cada columna. Cada valor tiene un *timestamp* (versiones)

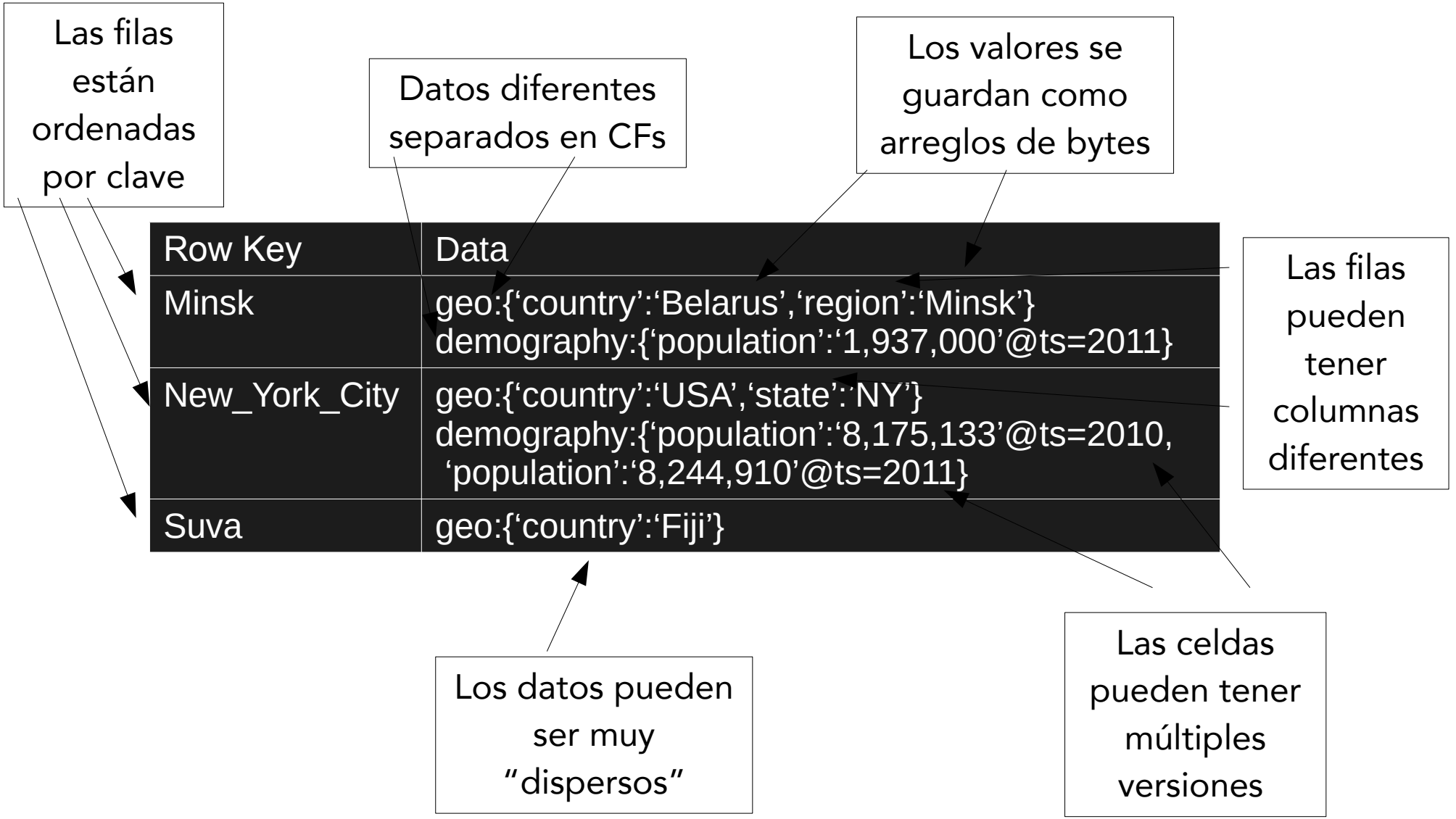
Las columnas se agrupan en **column families** (CF)

Las CFs se definen cuando se crea el esquema, pero se pueden agregar columnas luego.



(row key,column family,column, **timestamp**)-> value

Fuente: Hadoop: The Definitive Guide, Tom Gray. (2015, 4th edition)


Las filas están ordenadas por clave
Datos diferentes separados en CFs
Los valores se guardan como arreglos de bytes
Las filas pueden tener columnas diferentes
Los datos pueden ser muy "dispersos"
Las celdas pueden tener múltiples versiones
Row Key
Data
Minsk
geo:{'country':'Belarus','region':'Minsk'}
demography:{'population':'1,937,000'@ts=2011}
New_York_City
geo:{'country':'USA','state':'NY'}
demography:{'population':'8,175,133'@ts=2010,
'population':'8,244,910'@ts=2011}
Suva
geo:{'country':'Fiji'}

Lecturas y escrituras

- Escrituras atómicas a nivel de **fila:**
  - Más de una fila NO es atómico (no hay transacciones nativas)
- Lecturas de dos tipos:
  - Una fila: **get**
  - Múltiples filas: **scan.**
    - Típicamente se indica clave de inicio y clave de fin
    - Muy rápido (filas ordenadas por clave)

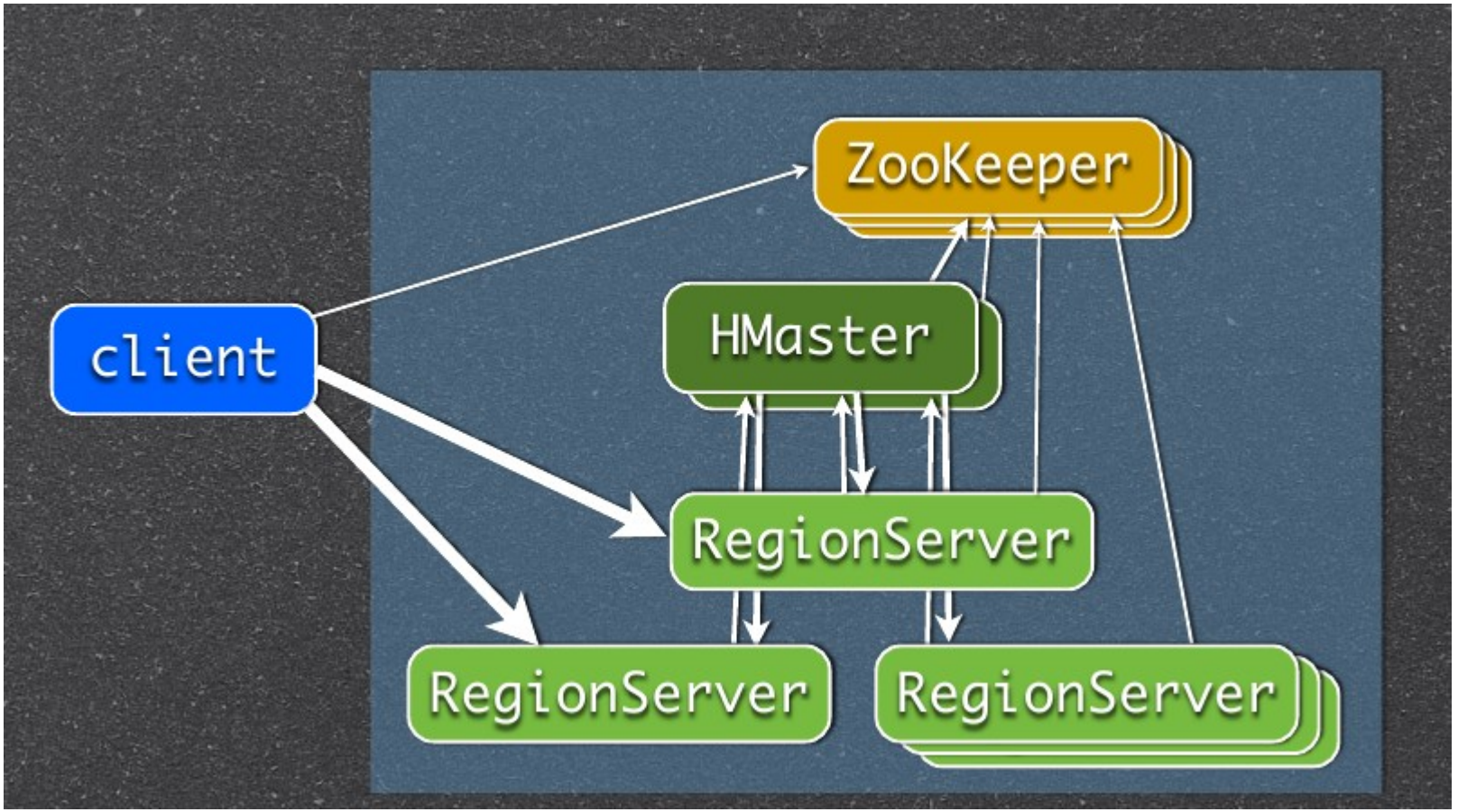
Particionamiento horizontal

- Las tablas se particionan automáticamente:
  - al llegar a cierto tamaño (fijo o variable *IncreasingToUpperBoundRegionSplitPolicy*)
  - En función de los valores de un prefijo de la clave (*KeyPrefixRegionSplitPolicy*)
- Estas particiones se denominan **regiones**
- Cada región se asigna a un *RegionServer*
  - un *RegionServer* puede gestionar más de una region

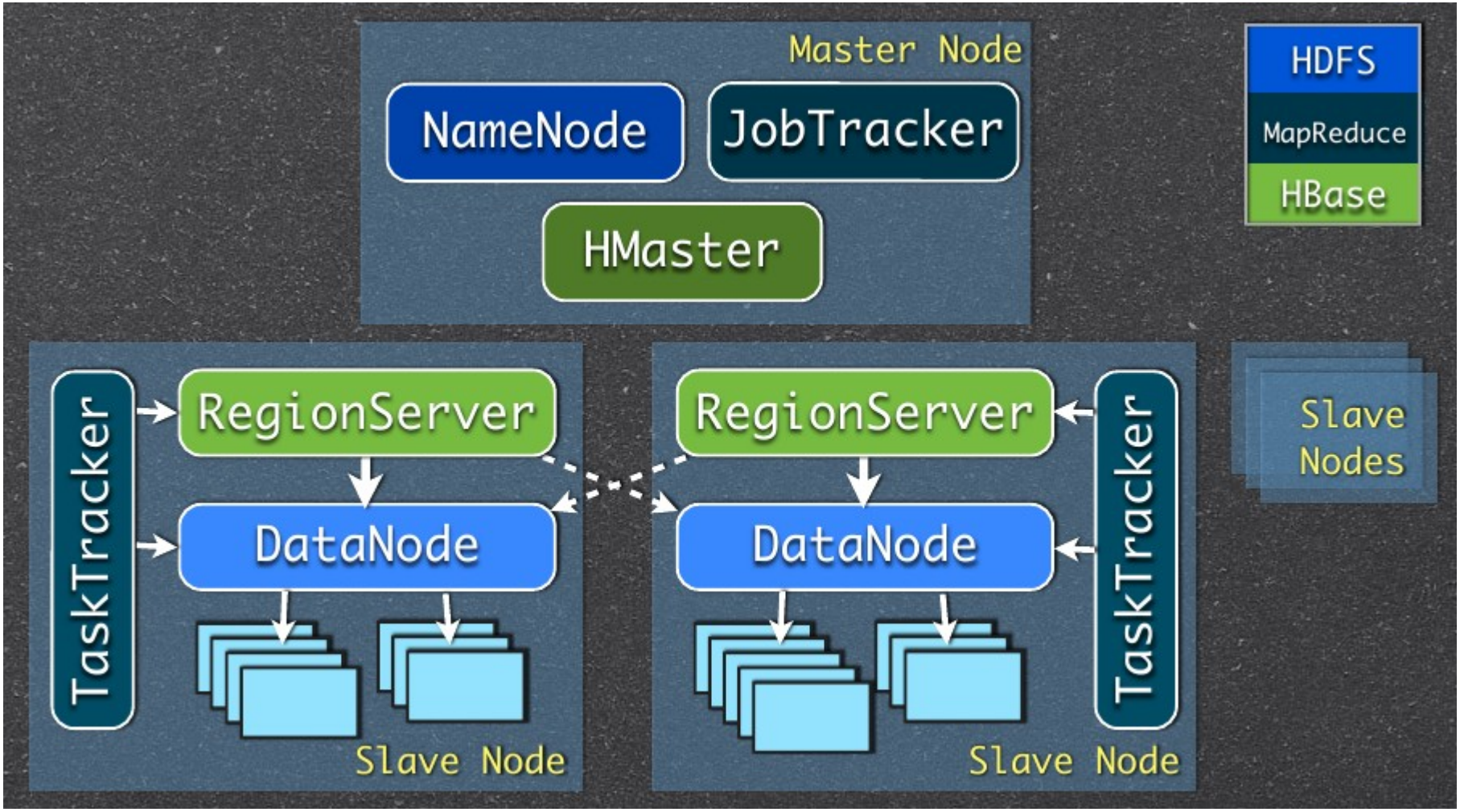
Particionamiento vertical

- Por cada *column family (CF)*
  - Cada CF se almacena junta en el HDFS

Componentes



HBase + HDFS


Master Node
NameNode
JobTracker
HMaster
HDFS
MapReduce
HBase
TaskTracker
RegionServer
DataNode
Slave Node
RegionServer
DataNode
TaskTracker
Slave Node
Slave Nodes

Tolerancia a fallas

- Las fallas en los *DataNode* las maneja HDFS (por defecto replica x 3)
- Las fallas en los *RegionServer* (RS) las maneja HBase
  - HMaster reasigna regiones a los RS disponibles
  - Fallo en el HMaster? Se recupera automáticamente si hay múltiples HMasters

¿Cuándo usar HBase?

¿Cuándo usar HBase?

- Grandes volúmenes de datos (billones de filas)
- Aplicaciones con muchas escrituras
- Escrituras del tipo ***append*** (insertar o sobreescribir) en lugar de leer-modificar-escribir

Un ejemplo de caso de uso

- Auditoria de logs
  - Registro de acciones de usuarios
  - Responder preguntas del estilo:
    - ¿cuáles fueron las últimas N acciones hechas por un usuario?
      - Clave de fila: *userId_timestamp*
    - ¿qué usuarios se *loggearon* ayer?
      - Clave de fila: *action_timestamp_userId*

Algunas consideraciones al diseñar

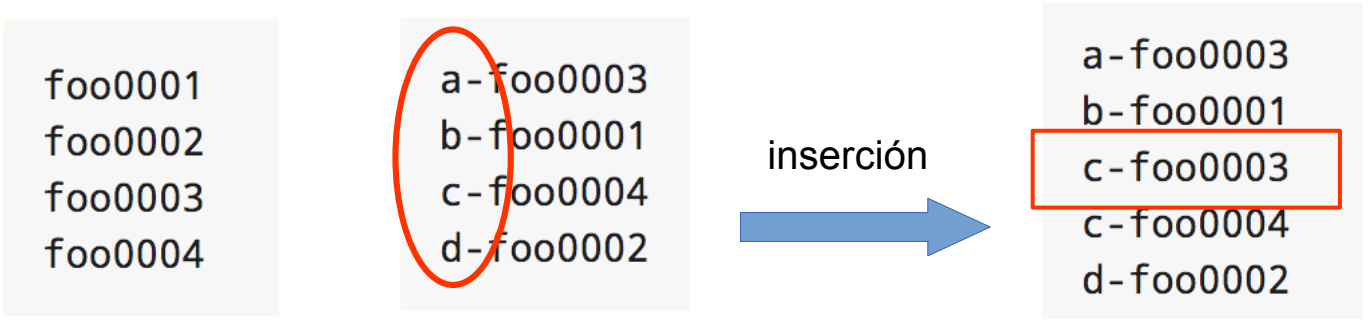
- La elección de las claves es **crucial**
  - Para el particionamiento
  - Para la búsqueda
  - Para el balance de:
    - Cantidad de filas por región
    - Accesos por región
- Debo tener en cuenta las preguntas/consultas que quiero resolver al momento de diseñar las claves
- No hay joins!! desnormalizar y anidar entidades para representar relaciones

Hotspotting

- Las filas se ordenan lexicográficamente por clave
- Si la clave está "mal diseñada" la mayoría del tráfico se dirige a uno o algunos nodos (*hotspotting*)
- Puede llevar a que algunas regiones no estén disponibles:
  - Las que son muy accedidas
  - Las que están en el mismo RegionServer

Algunas recomendaciones (I)

- Número de familias de columnas menor a 10
- Estrategias de diseño de claves para mitigar *hotspotting*:
  - **Salting:** agregar prefijos aleatorios a cada clave para alterar el orden



Mejora tiempos de escritura, empeora la lectura

Algunas recomendaciones (II)

- Aplicar *hashing* para asignar prefijo (en lugar de ser aleatorio)
- Invertir la clave:
  - Para claves numéricas
  - Poner al comienzo la parte que cambia más a menudo
  - This effectively randomizes row keys, but sacrifices row ordering properties.

Un caso de estudio

- Clientes y órdenes de compra

Cliente: Customer number, Customer name, Address (e.g., city, state, zip), Phone numbers, etc.

Compra: Customer number, Order number, Sales date, A series of nested objects for shipping locations and line-items

- ¿Una tabla o dos tablas?

- Diseño de la clave

- ¿cómo modelo el objeto orden?

http://hbase.apache.org/book.html#schema.casestudies

Referencias y material adicional

- HBase Reference Guide
  - http://hbase.apache.org/book.html
- HBase architecture
  - https://data-flair.training/blogs/hbase-architecture/
- HBase schema design, Apache HBase 2017.
  - http://hbase.apache.org/book.html#schema
- HBase schema design, Ian Varley, 2012
  http://www.slideshare.net/cloudera/5-h-base-schemahbasecon2012