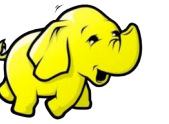


Comprender HBase y BigTable

Este documento es una libre traducción del artículo en inglés "Understanding HBase and BigTable": http://jimb qjw.com/wiki/index.php?title=Understanding_Hbase_and_BigTable

La parte más difícil de aprender HBase (la implementación de código abierto de BigTable de Google), es entender los conceptos reales que se manejan.

Es un poco triste ver que estos dos grandes sistemas contienen las palabras "table" y "base" en sus nombres, lo cual tiende a causar confusión entre los individuos acostumbrados a los Sistemas de Gestión de Bases de Datos Relacionales (Relational Database Management Systems en inglés, RDBMS de aquí en adelante).

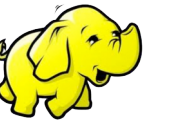
Este documento pretende describir estos sistemas de almacenamiento de datos distribuidos de un punto de vista conceptual. Después de leerlo, se estará en mejores condiciones para tomar una decisión respecto a cuándo es preferible utilizar HBase o, por otro lado, cuándo seria mejor una base de datos "tradicional".

Tabla de Contenidos

COMPRENDER HBASE Y BIGTABLE.....1

TABLA DE CONTENIDOS..... 1

TODO ESTÁ EN LA TERMINOLOGÍA 2
HASH-MAP 2
PERSISTENTE 3
DISTRIBUIDO 3
ORDENADOS 3
MULTIDIMENSIONAL 4
DENSIDAD BAJA 7



Todo esta en la terminología

Afortunadamente, el paper BigTable de Google explica claramente lo que BigTable realmente es. Esta es la primera frase del "Modelo de datos":

Una Bigtable es un hash-map de datos persistente, multidimensional, ordenado, poco denso y distribuido.

El documento BigTable continua explicando que:

El hash-map esta indexado por una Have de fila, una Have de columna y un timestamp, cada valor en el hash-map es una matriz de bytes no interpretados.

En este sentido, la pagina HBaseArchitecture del wiki de Hadoop plantea que:

HBase utiliza un modelo de datos muy similar al de Bigtable. Los usuarios almacenan filas de datos en las tablas de la etiqueta. Una fila de datos tiene una Have ordenada y un número arbitrario de columnas. La densidad de la tabla es baja, por lo que las filas en la misma tabla pueden tener un número muy variable de columnas.

A pesar de que todo esto puede parecer un poco críptico, tiene sentido una vez que se analizan los siguientes conceptos (mencionados en los párrafos anteriores) uno por uno: hash-map, persistente, distribuido, ordenados, multidimensional, y densidad baja. Este análisis se realizara en los apartados siguientes.

Hash-map

En su esencia, HBase / BigTable es un hash-map.

Desde el artículo de wikipedia, un hash-map es "un tipo de datos abstracto compuesto por una colección de Haves y una colección de valores, donde se asocia a cada Have con un valor".

Usando JavaScript Object Notation (JSON), se muestra aquí un ejemplo de un hash-map sencillo en el que todos los valores son cadenas de texto:

```
{
" z z z z z " : "woot",
"xyz": "hola",
" a a a a b " : "el mundo'
" 1 " : "x",
" a a a a a " : "y"
}
```

Persistente

Persistencia simplemente significa que los datos almacenados en el hash-map especial siguen existiendo después de que el programa que lo creó o accedió a él termina, de la misma forma que un archivo en un sistema de archivos puede ser persistente.

Distribuido

HBase y BigTable se basan en sistemas de archivos distribuidos de manera que el almacenamiento de archivos subyacente puede ser repartido entre un conjunto de máquinas independientes.

HBase puede utilizar cualquier sistema de archivos distribuido de Hadoop (HDFS) o Amazon Simple Storage Service (S3), mientras que BigTable hace uso del sistema de archivos de Google (GFS). En el caso de los clusters Hadoop que monta Pragsis, se configura HBase para que use HDFS, su almacenamiento más "natural".

Los datos se replican a través de una serie de nodos que participan de una manera análoga a cómo los datos están almacenados en los discos en un RAID del sistema. Esto proporciona una capa de protección contra fallos del sistema, como, por ejemplo, un nodo del cluster que deja de funcionar.

Ordenados

A diferencia de la mayoria de las implementaciones de hash-map, en HBase / BigTable los pares Have / valor se mantienen en estricto orden alfabetico. Es decir que la fila de la Have "aaaaa" debe estar al lado de la fila con la Have "aaaab" y muy lejos de la fila con la Nave "zzzzz".

Continuando con el ejemplo JSON, la version ordenada seria:

```
{
"1" :  "x" ,
"aaaaa" :  "y",
"aaaab" :  "el mundo",
"xyz" :   " hola",
"zzzzz" :  "woot"
}
```

Debido a que estos sistemas tienden a ser tan grandes y distribuidos, esta característica de ordenación es realmente muy importante. La proximidad espacial de filas con Haves similares asegura que, cuando hay que recorrer la tabla, los elementos de mayor interés para la búsqueda están cerca unos de otros.

Esto es importante al elegir una convencion de Have de fila. Por ejemplo, considérese una tabla cuyas Haves son los nombres de dominio. En este caso, tiene más sentido almacenar los dominios en notación inversa (es decir, "com.jimbojw.www" en lugar de "www.jimbojw.com"), de modo que las filas de un subdominio estén cerca de la fila de dominio principal.

Continuando con el ejemplo de dominio, la fila para el dominio "mail.jimbojw.com" estaria almacenada justo al lado de la fila para "www.jimbojw.com" (en lugar de estar almacenada justo al lado de la fila "mail.xyz.com" tal como seria si se usase una notación tradicional).

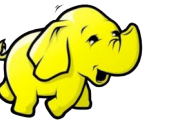
Es importante senalar que el término "ordenados" cuando se aplica a HBase / BigTable no quiere decir que los "valores" estén ordenados sino que son las Haves las que están ordenadas.

Multidimensional

Hasta este momento, no se ha mencionado ningún concepto de "columnas", sustituyendo el concepto de tabla por el concepto de un hash-map tradicional. Esto es totalmente intencional. La palabra "columna" es otro término, como "tabla" y "base", que tiene asociadas ciertas connotaciones debido a anos de experiencia con RDBMSs.

En su lugar, resulta más fácil pensar en una base de datos HBase como un mapa multidimensional - un mapa de mapas, si se quiere. Si se ahade una dimensión al ejemplo JSON, el hash-map seria asi:

```
{
"1" : {
  "A" : "x",
  "B" : "z"
};
"aaaaa" : {
  "A" : "y",
  "B" : "w"
};
"aaaab" : {
  "A" : "el mundo",
  "B" : "oceano"
};
"xyz" :  {
 "A" : " hola",
 "B" : "no"
};
"zzzzz" : {
"A" : "woot",
"B" : "1337"
 }
}
```



En el ejemplo anterior, se aprecia ahora que cada una de las Haves apunta a un hash-map con exactamente dos Haves: "A" y "B". De aqui en adelante, se denominará a la asociacion llave/hash-map de nivel superior como una fila. Además, en la nomenclatura de BigTable/HBase, las Haves "A" y "B" se llaman "las familias de columna".

Las familias de las columnas se especifican cuando se crea la tabla, y es dificil o imposible modificarlas después. También puede ser costoso ahadir nuevas familias de columna, por lo que es una buena idea especificar todas las que se necesitarán durante el ciclo de vida de la tabla.

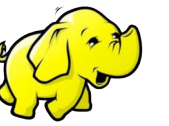
Afortunadamente, una familia de columna puede tener cualquier número de columnas, que se denota por una columna de "clasificación" o "etiqueta". Aqui está una parte del ejemplo anterior, esta vez con la dimensión de una columna de clasificación:

```
//...
"aaaaa" : {
  "A" :   {
    "Foo" :   "y"
    "Bar" :   "d"
   },
  "B" : {
    "" :    "w",
   }
},
"aaaab" : {
  "A" :{
    "Foo" :   "el mundo",
    "Bar" :   "dominación",
  };
  "B" :{
    "" :   "Oceano",
   }
 },
//...
}
```

Es importante tener en cuenta que en las dos filas, la familia de columna "A" tiene dos columnas: "Foo" y "Bar" y el "B" tiene una sola columna, cuya clasificación es la cadena vacia ("").

Cuando se hace una consulta en HBase / BigTable, hay que proporcionar el nombre de la columna completa en la forma "<FAMILIA>:<qualifier>". Asi, por ejemplo, las dos filas en el ejemplo anterior tienen tres columnas: "A: Foo", "A: Bar" y "B".

Aunque las familias columna son estáticas, las mismas columnas no lo son. Existe la posibilidad de ampliar esta fila:



```
{
 //...
 "zzzzz" : {
  "A" :  {
    "Catch_Phrase" : "woot",
  }
 }

}
```

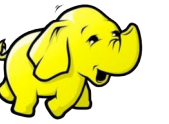
En este caso, la fila "zzzzz" tiene exactamente una columna, "A: Catch_phrase".

La última dimensión representada en HBase / BigTable es el tiempo. Todos los datos están versionados, ya sea usando un timestamp (segundos desde la época), o cualquier otro entero de su elección. El cliente puede especificar la fecha y hora en el momento de la inserción de los datos.

Considérese este ejemplo actualizado con la utilización de timestamps arbitrarios:

```
{
 //...
 "Aaaaa" : {
   "A" :  {
     "Foo" :  {
        15 :  "y",
        4 :  "m"
      }
     "Bar" : {
        15 :   "d",
      }
   "B" :  {
     "" :  {
       6 :  "w",
       3 :  "o"
       1 :  "w"
      }
    }
  },
 //...
}
```

Cada familia de columna puede tener sus propias reglas con respecto a la cantidad de versiones que se quiere mantener para una determinada celda (una celda se identifica por la pareja fila/columna). En la mayoría de los cases, las aplicaciones sólo piden datos de una celda determinada, sin especificar una feTMcha y hora. En ese caso, HBase / BigTable devolverá la versión más reciente (el que tiene la mayor marca de
tiempo), ya que se almacenan en orden cronólogico inverso.



Si una aplicación solicita una fila con una fecha determinada y una hora determinada, HBase devolverá los datos de la celda que tiene el mayor timestamp de entre todas las que tienen un timestamp menor o igual.

Usando la base de datos HBase del ejemplo, la consulta de la fila / columna de "Aaaaa'V'A: Foo" devolverá "y". Si se consulta por la fila/ columna/timestamp "Aaaaa'V'A: Foo"/10 se devolverá "m". Y la consulta de una "Aaaaa'V'A: Foo"/2 devolverá un resultado nulo.

Densidad baja

El último concepto clave es densidad baja. Como ya se mencionó, una fila determinada puede tener cualquier número de columnas en cada familia de la columna, o ninguno en absoluto.

Esto, por supuesto, tiene mucho sentido si se ha estado pensando en HBase / BigTable en los términos basados en el concepto de hash-map analizados en este capitulo, en lugar del significado de conceptos similares en los RDBMSs.