**Client Side Buffering In Hbase**

**HBase:**

HBase is an [open source](http://en.wikipedia.org/wiki/Open_source), non-[relational](http://en.wikipedia.org/wiki/Relational_model), [distributed database](http://en.wikipedia.org/wiki/Distributed_database) modeled after [Google's](http://en.wikipedia.org/wiki/Google). It runs on top of [Hadoop](http://en.wikipedia.org/wiki/Hadoop_Distributed_Filesystem) distributed file system, providing BigTable-like capabilities for Hadoop.Hbase provides a way to store large amounts of data in a fault tolerant way.Hbase can be also termed as the hadoop database.

We can perform operations like insertion into HBase using Put method. Put operation is like an RPC that transfers data from client to server and back. When the number of insertions are less, it is feasible but when we need to have like 1000 insertions per second, it’s better that we go for buffering.

Client Side Buffering

HBase API has a built in client side buffer that can buffer all the puts and sent it in a single RPC call. The following methods are used for this purpose.

void setAutoFlush(boolean autoFlush)

boolean isAutoFlush()

We need to set the ‘autoFlush’ to ‘false’ to enable the client side buffer as by default it’s not enabled.

table.setAutoFlush(false)

Once this has been done, we can check the state of the ‘autoFlush’ flag using the isAutoFlush() method. Now, all the put instances we created are stored in the memory in the client process. To force the data out of the buffer ,we need to call another function provided by API.

void flushCommits() throws IOException

This method flushes the data from buffer and ships it to the servers. Client batches the puts and sends them to the appropriate region servers in a single RPC call. You also have the option of setting the buffersize(client-side write buffer).In this case , API will track the size of data buffered and the required heap size for each instance. So it tracks the total overhead of the data. When it goes beyond a certain limit , it will implicitly flush the data. We can configure the max client side buffer size using the following:

void setWriteBufferSize(long writeBufferSize) throws IOException

You can retrieve the client side buffer size using getWriteBufferSize() method in the API. The default size is 2MB.

The data is hence flushed in 2 cases:

**a**. **Implicit Flush :**

It’s triggered whenever put() or setWriteBufferSize() is called. This will cause the API to compare the current buffer size with the configured maximum buffer size. If it exceeds the limit, then it triggers the flush. If the client buffer is disabled then each invocation of the put() will cause a buffer flush.close() also triggers buffer flush.

b. Explicit Flush

The flushCommits() is called explicitly to clear the buffer and sent the data to the server.

Here’s an example of client side buffering,

HTable table = new HTable(config, "example\_table");

table.setAutoFlush(false); //setting autoFlush to false to enable buffering.

Put put1 = new Put(Bytes.toBytes("row\_1"));

put1.add(Bytes.toBytes("column\_fam1"), Bytes.toBytes("column\_qual1"),

Bytes.toBytes("value\_1"));

table.put(put1);

Put put2 = new Put(Bytes.toBytes("row\_2 put1.add(Bytes.toBytes("column\_fam2"), Bytes.toBytes("column\_qual2"),

Bytes.toBytes("value\_2"));

table.put(put2);

Put put3 = new Put(Bytes.toBytes("row\_3"));

put1.add(Bytes.toBytes("column\_fam3"), Bytes.toBytes("column\_qual3"),

Bytes.toBytes("value\_3"));

table.put(put3);

Get get = new Get(Bytes.toBytes("row\_1"));

Result res1 = table.get(get);

table.flushCommits(); //an explicit flush is performed here.

In this example , we are creating puts one by one and inserting into the table.

When you need to access the write buffer content ,you can use

ArrayList<Put> getWriteBuffer()

This will return the internal list of the put instances that have been buffered so far by invoking table.

In the previous example, the put instances were being added one by one. We can also create a list of put instances and insert in one go using,

void put(List<Put> puts) throws IOException