Loading Coprocessors

Introduction

There are four ways to load table coprocessors:

1. Shell
2. Configuration
3. HTableDescriptor.setValue()
4. HTableDescriptor.addCoprocessor()

#1 uses #3 internally. Only #2 and #4 though check if the specified coprocessor class is already loaded, resulting in each class loaded only once

Question: What is the overall assumption? Load classes twice with ascending (or assigned) priorities? Or, load the class only once?

Example Code

Notes:

- The example contains an Observer, as well as a main() function to load the same
- There is a static counter that is incremented by one for each preGetOp() call
- The postGetOp() hook adds the current counter value as an artificial field into the result
- We expect to see increments by one, or else we have incurred multiple invocations!

```
package coprocessor;

import java.io.IOException;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellScanner;
import org.apache.hadoop.hbase.Coprocessor;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.coprocessor.BaseRegionObserver;
import org.apache.hadoop.hbase.coprocessor.ObserverContext;
import org.apache.hadoop.hbase.coprocessor.RegionCoprocessorEnvironment;
import org.apache.hadoop.hbase.regionserver.HRegion;
import org.apache.hadoop.hbase.util.Bytes;
import util.HBaseHelper;

// cc DuplicateRegionObserverExample Example of attempting to load the same observer multiple times
public class DuplicateRegionObserverExample extends BaseRegionObserver {
  public static final Log LOG = LogFactory.getLog(HRegion.class);

  public static final byte[] FIXED_COLUMN = Bytes.toBytes("@@@GET_COUNTER@@@");
  private static AtomicInteger counter = new AtomicInteger(0);

  @Override
  public void preGetOp(ObserverContext<RegionCoprocessorEnvironment> e,
    Get get, List<Cell> results) throws IOException {
    int count = counter.incrementAndGet();
    LOG.info("Current preGet count: " + count + " [" + this + "]");
  }

  @Override
  public void postGetOp(ObserverContext<RegionCoprocessorEnvironment> e,
    Get get, List<Cell> results) throws IOException {
    Put put = new Put(get.getRow());
    put.addColumn(get.getRow(), FIXED_COLUMN, Bytes.toBytes(counter.get()));
    CellScanner scanner = put.cellScanner();
    scanner.advance();
```

```
    Cell cell = scanner.current();
    LOG.debug("Adding fake cell: " + cell);
    results.add(cell);
  }

  public static void main(String[] args) throws IOException {
    Configuration conf = HBaseConfiguration.create();
    conf.set("hbase.rootdir</name>", "hdfs://master-1.internal.larsgeorge.com:9000/hbase");
    conf.set("hbase.cluster.distributed", "true");
    conf.set("hbase.zookeeper.quorum", "master-1.internal.larsgeorge.com,master-2.internal.larsgeorge.com,m
    HBaseHelper helper = HBaseHelper.getHelper(conf);
    helper.dropTable("testtable");

    Connection connection = ConnectionFactory.createConnection(conf);
    TableName tableName = TableName.valueOf("testtable");

    HTableDescriptor htd = new HTableDescriptor(tableName)
      .addFamily(new HColumnDescriptor("colfam1"))
      .addCoprocessor(DuplicateRegionObserverExample.class.getCanonicalName(),
        null, Coprocessor.PRIORITY_USER, null)
      .addCoprocessor(DuplicateRegionObserverExample.class.getCanonicalName(),
        null, Coprocessor.PRIORITY_USER, null);

    Admin admin = connection.getAdmin();
    admin.createTable(htd);
    System.out.println(admin.getTableDescriptor(tableName));

    System.out.println("Adding rows to table...");
    helper.fillTable("testtable", 1, 10, 10, "colfam1");

    Table table = connection.getTable(tableName);
    Get get = new Get(Bytes.toBytes("row-1"));
    Result result = table.get(get);

    helper.dumpResult(result);

    table.close();
    admin.close();
    connection.close();
  }}
```

Tests

I tried a few test scenarios since I have managed to create duplicate entries before.

Test #1 - Use main() of example code

This fails before even creating the table, while setting up the HTableDescriptor instance, throwing the following exception at runtime:

```
Exception in thread "main" java.io.IOException: Coprocessor \
  coprocessor.DuplicateRegionObserverExample already exists.
  at org.apache.hadoop.hbase.HTableDescriptor.addCoprocessor(HTableDescriptor.java:1232)
  at coprocessor.DuplicateRegionObserverExample.main(DuplicateRegionObserverExample.java:69)
  ...
  at com.intellij.rt.execution.application.AppMain.main(AppMain.java:140)
```

Result: Allows to load same coprocessor class to be added only **once**.

Test #2 - Using the shell's ALTER command

Adding coprocessor using the HBase shell uses its own table attribute insertion (see approach #3 above) with no check during the ALTER comm

```
$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.1, rd0a115a7267f54e01c72c603ec53e91ec418292f, Tue Jun 23 14:44:07 PDT 2015

hbase(main):001:0> create 'testtable', 'colfam1'
0 row(s) in 2.1750 seconds

=> Hbase::Table - testtable
hbase(main):002:0> alter 'testtable', 'coprocessor' => 'file:///opt/hbase-book/hbase-book-ch04-2.0.jar|copr
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.9970 seconds
```

```
hbase(main):003:0> describe 'testtable'
Table testtable is ENABLED
testtable, {TABLE_ATTRIBUTES => {coprocessor$1 => 'file:///opt/hbase-book/hbase-book-ch04-2.0.jar|coprocess
COLUMN FAMILIES DESCRIPTION
{NAME => 'colfam1', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS
KSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0680 seconds

hbase(main):004:0> alter 'testtable', 'coprocessor' => 'file:///opt/hbase-book/hbase-book-ch04-2.0.jar|copr
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.1740 seconds

hbase(main):005:0> describe 'testtable'
Table testtable is ENABLED
testtable, {TABLE_ATTRIBUTES => {coprocessor$1 => 'file:///opt/hbase-book/hbase-book-ch04-2.0.jar|coprocess
ocessor.DuplicateRegionObserverExample|'}
COLUMN FAMILIES DESCRIPTION
{NAME => 'colfam1', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS
KSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0200 seconds

hbase(main):006:0> put 'testtable', 'row-1', 'colfam1:col1', 'val1'
0 row(s) in 0.1090 seconds

hbase(main):007:0> get 'testtable', 'row-1'
COLUMN                                          CELL
 colfam1:col1                                   timestamp=1439549519940, value=val1
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
3 row(s) in 0.0380 seconds

hbase(main):008:0> get 'testtable', 'row-1'
COLUMN                                          CELL
 colfam1:col1                                   timestamp=1439549519940, value=val1
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
3 row(s) in 0.0200 seconds

hbase(main):009:0>
```

We are seeing increments in two, which is expected because the same class is loaded twice and incrementing the static, shared counter variable

Result: Allows to load same class **multiple** times.

Test #3 - Using the Configuration in hbase-site.xml

In `hbase-site.xml` I added:

```
  <property>
    <name>hbase.coprocessor.user.region.classes</name>
    <value>coprocessor.DuplicateRegionObserverExample,coprocessor.DuplicateRegionObserverExample</value>
  </property>
```

and in `hbase-env.sh`:

```
export HBASE_CLASSPATH=/opt/hbase-book/hbase-book-ch04-2.0.jar
```

Notes:

- Needs to be done on all servers (especially the slaves here)
- The HBase processes need to be restarted to make the changes available
- It was *not* necessary before, since the ALTER command can load explicit JAR files as shown
- Here we do *not* use the main() of the example class, but just the Observer instance
    - Instead, we use a system-wide user table observer hook, which is applied to all user tables

Then in the HBase shell do this:

```
hbase(main):009:0> disable 'testtable'
0 row(s) in 4.5290 seconds

hbase(main):010:0> drop 'testtable'
0 row(s) in 1.2800 seconds

hbase(main):011:0> create 'testtable', 'colfam1'
```

```
0 row(s) in 1.2950 seconds

=> Hbase::Table - testtable
hbase(main):012:0> describe 'testtable'
Table testtable is ENABLED
testtable
COLUMN FAMILIES DESCRIPTION
{NAME => 'colfam1', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS
KSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0280 seconds

hbase(main):013:0> put 'testtable', 'row-1', 'colfam1:col1', 'val1'
0 row(s) in 0.0730 seconds

hbase(main):016:0> get 'testtable', 'row-1'
COLUMN                                          CELL
 colfam1:col1                                   timestamp=1439552857290, value=val1
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
3 row(s) in 0.4570 seconds

hbase(main):017:0> get 'testtable', 'row-1'
COLUMN                                          CELL
 colfam1:col1                                   timestamp=1439552857290, value=val1
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
3 row(s) in 0.0380 seconds

hbase(main):018:0> describe 'testtable'
Table testtable is ENABLED
testtable
COLUMN FAMILIES DESCRIPTION
{NAME => 'colfam1', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS
KSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0470 seconds
```

First, we can see that the artificial field was added, so the coprocessor was loaded as expected. But we still see double posts. Using the logs we

```
2015-08-14 05:11:29,124 INFO  [RS_OPEN_REGION-slave-1:16020-1] coprocessor.CoprocessorHost: System coproces
2015-08-14 05:11:29,124 INFO  [RS_OPEN_REGION-slave-1:16020-1] coprocessor.CoprocessorHost: System coproces
...
2015-08-14 05:13:13,889 INFO  [B.defaultRpcServer.handler=1,queue=1,port=16020] regionserver.HRegion: Curre
2015-08-14 05:13:13,889 INFO  [B.defaultRpcServer.handler=1,queue=1,port=16020] regionserver.HRegion: Curre
2015-08-14 05:13:17,599 INFO  [B.defaultRpcServer.handler=2,queue=2,port=16020] regionserver.HRegion: Curre
2015-08-14 05:13:17,599 INFO  [B.defaultRpcServer.handler=2,queue=2,port=16020] regionserver.HRegion: Curre
```

If you remove one of the coprocessor definitions in the site file (then copying it to all servers, and restart of processes etc.) like so

```
  <property>
    <name>hbase.coprocessor.user.region.classes</name>
    <value>coprocessor.DuplicateRegionObserverExample</value>
  </property>
```

Then you get the expected result on the shell:

```
hbase(main):014:0> get 'testtable', 'row-1'
COLUMN                                          CELL
 colfam1:col1                                   timestamp=1439552857290, value=val1
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
2 row(s) in 0.0540 seconds

hbase(main):015:0> get 'testtable', 'row-1'
COLUMN                                          CELL
 colfam1:col1                                   timestamp=1439552857290, value=val1
 row-1:@@@GET_COUNTER@@@                        timestamp=9223372036854775807, value=\x00\x00\x00\x0
```

The server side code actually checks for duplicates:

```
 /**
   * Load system coprocessors. Read the class names from configuration.
   * Called by constructor.
   */
  protected void loadSystemCoprocessors(Configuration conf, String confKey) {
    ...
    for (String className : defaultCPClasses) {
      className = className.trim();
      if (findCoprocessor(className) != null) {
```

```
        continue;
      }
      ClassLoader cl = this.getClass().getClassLoader();
      Thread.currentThread().setContextClassLoader(cl);
      try {
        implClass = cl.loadClass(className);
        configured.add(loadInstance(implClass, Coprocessor.PRIORITY_SYSTEM, conf));
        LOG.info("System coprocessor " + className + " was loaded " +
            "successfully with priority (" + priority++ + ").");
      } catch (Throwable t) {
        // We always abort if system coprocessors cannot be loaded
        abortServer(className, t);
      }
    }
    // add entire set to the collection for COW efficiency
    coprocessors.addAll(configured);
  }

  /**
   * Find a coprocessor implementation by class name
   * @param className the class name
   * @return the coprocessor, or null if not found
   */
  public Coprocessor findCoprocessor(String className) {
    for (E env: coprocessors) {
      if (env.getInstance().getClass().getName().equals(className) ||
          env.getInstance().getClass().getSimpleName().equals(className)) {
        return env.getInstance();
      }
    }
    return null;
  }
```

The problem is that at region open time, when iterating over the list of existing coprocessor environment instance, there is none yet. These are ac

The RegionServer UI lists the duplicated only once, since it provides the lists of loaded coprocessors as a Set instance, which eliminates all the d

Coprocessors	[DuplicateRegionObserverExample]	Cop regi

Result: Allows to load same class **multiple** times.

Summary

Here are the findings:

Test #	Name	Result
1	API: HTableDescriptor.addCoprocessor()	Only the addCoprocessor() call checks for duplica
2	Shell: ALTER command	The shell uses setValue(), bypassing the check.
3	Configuration (hbase-site.xml)	Loads duplicate classes, gets inconsistent display

I believe, looking at the code, that it should NOT be possible to load the same class more than once, or else it would assign each a unique key, a

1. Fix duplicate check in loadSystemCoprocessors()
2. Switch shell command to use addCoprocessor() (could also add extra check, but why the extra work?)
3. Make sure other loadCoprocessors() calls are in line

Input?