Metacat - Bug #6136

files left open causes too many file descriptors on OS

10/09/2013 01:03 PM - Matt Jones

Status:	Closed	**Start date:**	10/09/2013
Priority:	Immediate	**Due date:**	
Assignee:	Matt Jones	**% Done:**	0%
Category:	metacat	**Estimated time:**	0.00 hour
Target version:	2.2.1	**Spent time:**	0.00 hour
Bugzilla-Id:			

Description

Metacat writes temp files to disk, and in the process has been failing to close file handles. Over time, especially with operations that touch many files, the number of file handles in use by Metacat increases and eventually exceeds the operating systems hard limit, causing exceptions when Metacat tries to open any additional files. Need to be sure to close all file handles properly after usage.

History

#1 - 10/09/2013 01:05 PM - Matt Jones

Closed file handles in DocumentImpl and ReplicationService in commit r8297.

#2 - 10/09/2013 01:28 PM - Matt Jones

Also added a close() to the finally block in DocumentImpl.readFromFileSystem() in r8298. This is probably not causing the issue but won't hurt either.

#3 - 10/09/2013 01:53 PM - Matt Jones

Switched to using IOUtils.closeQuietly() which handles nulls and already closed file handles in r8299.

#4 - 10/09/2013 03:05 PM - Matt Jones

After looking into this, its not just file handles that are left open, but many InputStream and OutputStream instances of various sorts. The code needs a full audit to find all of these issues and fix them.

#5 - 10/09/2013 11:57 PM - Matt Jones

Checked all instances of FileInputReader and ensured that close() was being called properly. See commit r8304.

#6 - 10/10/2013 12:43 PM - Matt Jones

A brief overview of how I was inspecting memory, threads, file descriptors, etc. while stress testing Metacat. This should probably be added to the Metacat Admin guide in the developers section.

Checking system limits for file descriptors
On Ubuntu, user limits are seen as:

```
      ulimit -a
```

On Ubuntu, systemwide open file descriptors and max allowed can be seen using:

```
      sysctl fs.file-nr
```

Monitoring open file descriptors for the tomcat process:
On Mac OS X:

```
       while (true); do lsof -p 94375 |grep -v txt |grep -v cwd |wc -l; sleep 1; done
```

On Ubuntu:

```
       while (true); do lsof -a -p `pidof java` |wc -l; sleep 1; done
```

To generate a lot of get() requests, set up one or more terminals (on the same machine if it has enough CPUs, or on another machine), and use curl to download a list of pids (assumes that the file 'ids' exists with one id per line):

```
    while (true); do cnt=1; for id in `cat ids`; do echo "Downloading $id : "; cnt=$((cnt + 1)); curl http://l
ocalhost:8080/metacat/d1/mn/v1/object/$id -o mdf-${cnt}; done; done
```

If that causes a crash, I found that sleeping for 1 or more seconds between curl calls will slow things down enough to let Metacat keep up.

To monitor memory, threads, and object creation on the JVM, I used VisualVM that ships with Java. Locally this provides graphs and lists of threads and memory used by tomcat, and allows a capture of the memory usage of individual classes. I found it useful to start a capture of memory usage, and then filter to show just 'metacat' or just 'dataone' classes to see what happens in these respective collections of classes as the system is loaded at various levels. To run VisualVM on a remote host, one can set up jstatsd to see the threads and memory usage, and then connect to the remote host with VisualVM. To also get the object instance counts and memory usage via the sampling interface, you would also have to set up JMX on the VM.

#7 - 10/15/2013 10:34 AM - ben leinfelder

- Status changed from In Progress to Closed