Simple Fair Call Queue Workaround

George Jahad

The Problem

The Ozone S3G uses a permanent connection to the OM to send s3 requests it receives. That connection masks the initial user so that all requests appear to come from the special S3G user.

This masking unfortunately eliminates the utility of the fair call q. (The q used by the connection between the S3G and the OM.) Since each request appears to come from the S3G user, the fcq has no way of prioritizing less frequent callers.

The fcq has a mechanism for overriding the identity info associated with the request. This mechanism is called the identityProvider. Christos has created a PR, https://github.com/apache/ozone/pull/4116 , to leverage identityProvider. Unfortunately, it doesn't work. The problem is that real user identity information required is not available until after the identityProvider is invoked, (in the ipc/Server Reader thread.)

Caller Context

We could workaround this problem, but it will require a minor change to the hadoopRPC implementation. The workaround takes advantage of the CallerContext field that already exists in HRPC to support audit logging. That field is unused by Ozone but is used by other parts of the ecosystem, like Yarn and Flink. In Ozone, it could be repurposed to carry the identity info in a manner accessible to the identity provider.

The identityProvider is invoked with a parameter of type Schedulable, like so:

```
public String makeIdentity(Schedulable schedulable)
```

The Schedulable class currently provides no access to the CallerContext, but could be extended to do so with a few lines of code like so:

```
public CallerContext getCallerContext() {
return this.callerContext;
}
```

With that simple change to hadoop, the Ozone S3G could set the real users identity. The identity provider on the OM would then read it and pass it to the fcq.

Implementation

The changes to the identityProvider in Christos PR would look like this:

```
+     CallerContext callerContext = schedulable.getCallerContext();
+     if (callerContext != null) {
+     if (!StringUtil.isNullOrEmpty( callerContext.getContext())) {
+     return callerContext.getContext();
+     }
```

Setting the caller context would look like this:

```
@@ -291,28 +292,18 @@ private OMResponse submitRequest(OMRequest omRequest)
+     if (threadLocalS3Auth.get() != null) {
+     if (!Strings.isNullOrEmpty(threadLocalS3Auth.get().getAccessID())) {
+     CallerContext callerContext =
+           new CallerContext.Builder(
+                 threadLocalS3Auth.get().getAccessID()).build();
+     CallerContext.setCurrent(callerContext);
      }
```

I've implemented this starting using the hadoop 3.3.4 branch and christos PR. It seems to work, but I haven't tested it extensively. (Here are my changes to hadoop: https://github.com/GeorgeJahad/hadoop/compare/branch-3.3.4..rel334-callerContext and to Christos PR: https://github.com/GeorgeJahad/ozone/compare/christosOriginalFairCallQ..testCC?w=1)

The advantages of this approach are it is simple, and likely won't affect performance at all.

The disadvantages are that it requires approval from the hadoop community and it only works on HRPC, not GRPC.

Next Steps

If we decide to move forward with this approach, the next thing I would do is run more extensive tests to confirm it really does work.

I would also update it to the latest version of hadoop-common, 3.3.5. Note that version doesn't currently seem to work with Ozone. So that would have to be investigated as well. (The current version on ozone is the one I modified, 3.3.4).

Also, I should note that this only adds support for the s3g connection, (and any other long running connection,) correctly identifies the user. It assumes that the fair call q itself outside of the s3g already works on ozone. (But I'm not sure if anyone has confirmed that. If not, then we need to confirm that as well.)

Finally, if that looks good, I would bounce it off the community to see if they think the hadoop team would allow it.