Actor Bug: Concurrency - Cooperation

Actors should be capable of running simultaneously with each other without error and these issues are based on problems occurring from actors performing or existing simultaneously.

There was a classical co-operation bug in Ensime-Server where a SearchService used semaphores for it's sub cases. This reinforces the paper by Tasharofi from 2013 which goes into length about how programmers mix other forms of concurrency with the actor model. In this case, it led to a semaphore leak where an Actor ask, or future, would try to acquire a semaphore which then would be acquired but one of the return cases after trying to complete the task wouldn't release that semaphore. This was fixed by adding the release to that case and tightening the request constraints.

The code below shows how they sought to plug their semaphore leak. They created a retry operation and added a clause to acquire the semaphore in the Future directly. They also added a case to release a semaphore in the result of a failure on the first attempt. Each attempt that fails calls the retry method to try again.

Link: https://github.com/ensime/ensime-server/pull/1531

Code:

ensime/indexer/SearchService.scala

```
+      def retry(): Unit = {
+        batch.foreach(self !)
+      }
+
       Future.sequence(batch.map {
-        case (_, f) =>
-          if (!f.exists()) Future.successful(f -> Nil)
-          else searchService.extractSymbolsFromClassOrJar(f).map(f -> )
+        case (url, f) =>
+          val filename = f.getName.getPath
+          // I don't trust VFS's f.exists()
+          if (!File(filename).exists()) {
+            Future {
+              searchService.semaphore.acquire() // nasty, but otherwise we leak
+              f -> Nil
+            }
+          } else searchService.extractSymbolsFromClassOrJar(f).map(f -> )
       }).onComplete {
         case Failure(t) =>
-          log.error(t, s"failed to index batch of ${batch.size} files")
+          searchService.semaphore.release()
+          log.error(t, s"failed to index batch of ${batch.size} files. $advice")
```

```diff
+           retry()
          case Success(indexed) =>
            searchService.delete(indexed.map(_._1)(collection.breakOut)).onComplete {
              case Failure(t) =>
                searchService.semaphore.release()
-               log.error(t, s"failed to remove stale entries in ${batch.size} files")
+               log.error(t, s"failed to remove stale entries in ${batch.size} files. $advice")
+               retry()
              case Success(_) => indexed.foreach {
                case (file, syms) =>
                  val boost = searchService.isUserFile(file.getName)
                  val persisting = searchService.persist(FileCheck(file), syms, commitIndex = true, boost = boost)

                  persisting.onComplete {
                    case _ => searchService.semaphore.release()
                  }

                  persisting.onComplete {
-                   case Failure(t) => log.error(t, s"failed to persist entries in $file")
+                   case Failure(t) =>
+                     log.error(t, s"failed to persist entries in $file. $advice")
+                     retry()
                    case Success(_) =>
                  }
              }
```