Item 66
git_comments:

git_commits:

1. **summary:** SOLR-9859: Don't log error on NoSuchFileException (Cao Manh Dat)
   message: SOLR-9859: Don't log error on NoSuchFileException (Cao Manh Dat)

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
   description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code} java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}
2. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
   description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code} java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

3. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}
4. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}
5. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at

org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

6. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

7. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

8. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at

org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

9. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
   description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
   java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
   sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
   sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
   sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
   sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
   java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown
   Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
   org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
   org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
   org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
   org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
   org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
   org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
   org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
   org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
   org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
   java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
   Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
   java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
   java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}
   label: code-design
10. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
    description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
    java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
    sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
    sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
    sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
    sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
    java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown
    Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
    org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
    org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
    org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
    org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
    org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
    org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
    org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
    org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
    org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
    java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
    Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
    java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
    java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}
    label: code-design
11. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
    description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}

java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

12. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}
label: code-design

13. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at

java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

14. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}
label: test

15. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

16. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at

org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

17. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown
Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

18. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown
Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

19. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at

java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

20. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

21. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

22. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create

{{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown
Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

23. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown
Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

24. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown
Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown
Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at

java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

25. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

26. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at
org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at
org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at
org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at
org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at
java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

27. **summary:** replication.properties cannot be updated after being written and neither replication.properties or index.properties are durable in the face of a crash
description: If a shard recovers via replication (vs PeerSync) a file named {{replication.properties}} gets created. If the same shard recovers once more via replication, IndexFetcher fails to write latest replication information as it tries to create {{replication.properties}} but as file already exists. Here is the stack trace I saw {code}
java.nio.file.FileAlreadyExistsException: <solr_home>\shard-3-001\cores\collection1\data\replication.properties at
sun.nio.fs.WindowsException.translateToIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsException.rethrowAsIOException(Unknown Source) at
sun.nio.fs.WindowsFileSystemProvider.newByteChannel(Unknown Source) at
java.nio.file.spi.FileSystemProvider.newOutputStream(Unknown Source) at java.nio.file.Files.newOutputStream(Unknown Source) at org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:413) at
org.apache.lucene.store.FSDirectory$FSIndexOutput.<init>(FSDirectory.java:409) at
org.apache.lucene.store.FSDirectory.createOutput(FSDirectory.java:253) at
org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) at
org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) at

org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) at org.apache.solr.cloud.RecoveryStrategy.replicate(RecoveryStrategy.java:157) at org.apache.solr.cloud.RecoveryStrategy.doRecovery(RecoveryStrategy.java:409) at org.apache.solr.cloud.RecoveryStrategy.run(RecoveryStrategy.java:222) at java.util.concurrent.Executors$RunnableAdapter.call(Unknown Source) at java.util.concurrent.FutureTask.run(Unknown Source) at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$0(ExecutorUtil.java:229) at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source) at java.lang.Thread.run(Unknown Source) {code}

jira_issues_comments:

1. Proposed solution 'Delete existing file and create a new one' ?
2. I may be seeing this in the tests as well, but we just WARN about it. Should review this code: {noformat} [junit4] 2> 254570 WARN (indexFetcher-1159-thread-1) [x:collection1] o.a.s.h.IndexFetcher Exception while updating statistics [junit4] 2> java.io.IOException: file "replication.properties" was already written to [junit4] 2> at org.apache.lucene.store.MockDirectoryWrapper.createOutput(MockDirectoryWrapper.java:654) [junit4] 2> at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:689) [junit4] 2> at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:501) [junit4] 2> at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:265) [junit4] 2> at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:397) [junit4] 2> at org.apache.solr.handler.ReplicationHandler.lambda$setupPolling$2(ReplicationHandler.java:1154) [junit4] 2> at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) [junit4] 2> at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:308) [junit4] 2> at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$301(ScheduledThreadPoolExecutor.java:180) [junit4] 2> at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:294) [junit4] 2> at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) [junit4] 2> at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) [junit4] 2> at java.lang.Thread.run(Thread.java:745) {noformat}
3. So this is an easy fix, but if you crash at the wrong time this isn't great. I wonder if we can try to load the index directory with the newest timestamp on startup if no index.properties file exists.
4. Is there a way we can write a temp file and do a mv to rename/overwrite replication.properties Alternate solution would be to keep appending to existing file and read the latest stats from the file.
5. Just by looking at the code at {{IndexFetcher#logReplicationTimeAndConfFiles}}, I think we have a problem here : - We open old {{replication.properties}} at {{ReplicationHandler#loadReplicationProperties()}} - We update the props in memory - We create a new file {{replication.properties}} and write down newest replication props to that file. So we always encounter this error if {{replication.properties}} exist, plus we can't update {{replication.properties}} so {{ReplicationHandler#getStatistics()}} always return the replication props of first run. The error happen in master-slave mode as well ({{TestReplicationHandler}}) Solution: I think we can simply delete the exist {{replication.properties}} before write a new one.
6. Here is the patch for this issue. In this patch we simply delete the {{replication.properties}} if the file already exists.
7. bq. Is there a way we can write a temp file and do a mv to rename/overwrite replication.properties Nothing great. Java 7 gives us an atomic move that can overwrite an existing file, but it's impl dependent on if that is supported and it wouldn't work for the arbitrary FileSystem's we support. We would still need some start up logic that could address a crashed state. bq. Alternate solution would be to keep appending to existing file and read the latest stats from the file. The problem is we use this same strategy for index.properties which is not so straightforward to do this way. bq. I think we can simply delete the exist replication.properties before write a new one. That is the easy fix I mention above, but it's fragile, and like index.properties, not robust in a crash.
8. From what I've read, this is one possible solution. Would still want to test a little on Windows I think.
9. **body:** bq. but it's impl dependent on if that is supported and it wouldn't work for the arbitrary FileSystem's we support I've read this should actually work across the major operating systems (I'd expect it on Unix systems, but seems Windows should be fine too). We can support it on HDFS as well it seems. So perhaps something like this is the easiest solution.
   label: code-design
10. **body:** [~markrmiller@gmail.com] looks like in the `atomicRename` file you are deleting existing file and then renaming the temp file. How is this better than just deleting the file a writing a new file, if we crash at a wrong time (as you have mentioned above). Would we need to manually rename the temp file in such a scenario?
    label: code-design
11. That is the fallback behavior. See the overrides.
12. **body:** Patch polished up a bit.
    label: code-design
13. If we have consensus I'll commit this soon.
14. **body:** Looks good to me Can we write a test to validate the patch?
    label: test
15. Seems to be the same issue * SOLR-9580 ?
16. Yeah. Looks like these are the same issue
17. Added a test for not being able to write replication statistics after the first replication.
18. Commit 96ed221fb6924dd167591004a5eaf70d53f92e4f in lucene-solr's branch refs/heads/master from markrmiller [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=96ed221] SOLR-9859: replication.properties cannot be updated after being written and neither eplication.properties or ndex.properties are durable in the face of a crash.

19. Commit c6ea1718675beebdddd988b08b2b213155d8b20f in lucene-solr's branch refs/heads/branch_6x from markrmiller [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=c6ea171] SOLR-9859: replication.properties cannot be updated after being written and neither eplication.properties or ndex.properties are durable in the face of a crash.
20. [~markrmiller@gmail.com] It seems we still have an exception being logged. It belong to the case when "replication.properties" do not exist. {code} java.nio.file.NoSuchFileException: /tmp/solr.cloud.OnlyLeaderIndexesTest_77D87333D3A12E8B-001/tempDir-001/node2/collection1_shard1_replica3/data/replication.properties at sun.nio.fs.UnixException.translateToIOException(UnixException.java:86) at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:102) at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:107) at sun.nio.fs.UnixFileSystemProvider.implDelete(UnixFileSystemProvider.java:244) at sun.nio.fs.AbstractFileSystemProvider.delete(AbstractFileSystemProvider.java:103) at java.nio.file.Files.delete(Files.java:1126) at org.apache.lucene.store.FSDirectory.privateDeleteFile(FSDirectory.java:373) at org.apache.lucene.store.FSDirectory.deleteFile(FSDirectory.java:335) at org.apache.lucene.store.FilterDirectory.deleteFile(FilterDirectory.java:62) at org.apache.solr.core.DirectoryFactory.renameWithOverwrite(DirectoryFactory.java:193) at org.apache.solr.core.MetricsDirectoryFactory.renameWithOverwrite(MetricsDirectoryFactory.java:201) at org.apache.solr.handler.IndexFetcher.logReplicationTimeAndConfFiles(IndexFetcher.java:726) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:519) at org.apache.solr.handler.IndexFetcher.fetchLatestIndex(IndexFetcher.java:274) at org.apache.solr.handler.ReplicationHandler.doFetch(ReplicationHandler.java:406) at org.apache.solr.handler.ReplicationHandler.lambda$setupPolling$2(ReplicationHandler.java:1163) at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:308) at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$301(ScheduledThreadPoolExecutor.java:180) at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:294) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) at java.lang.Thread.run(Thread.java:745) {code}
21. Thanks Cao - looks like we are only ignoring FileNotFoundException and this is throwing NoSuchFileException.
22. Commit 25290ab5d6af25c05cbbb4738f49329273a7d693 in lucene-solr's branch refs/heads/master from markrmiller [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=25290ab] SOLR-9859: Don't log error on NoSuchFileException (Cao Manh Dat)
23. Commit 3919519a22491f01c993b82bf1470f0d3967771c in lucene-solr's branch refs/heads/branch_6x from markrmiller [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=3919519] SOLR-9859: Don't log error on NoSuchFileException (Cao Manh Dat)
24. Thanks all!
25. Is there a workaround for fixing this on a 6.3.0 host? Does it work to delete replication.properties then start Solr?