Large DeleteObject requests are their own Thundering Herd

Details

Type:	Sub-task	Status:	RESOLVED
Priority:	Major	Resolution:	Fixed
Affects Version/s:	3.2.1	Fix Version/s:	3.3.0
Component/s:	fs/s3		
Labels:	None		
Target Version/s:	3.3.0		

Release Note: The page size for bulk delete operations has been reduced from 1000 to 250 to reduce the likelihood of overloading an S3 partition, especially because the retry policy on throttling is simply to try again.

The page size can be set in "fs.s3a.bulk.delete.page.size"

There is also an option to control whether or not the AWS client retries requests, or whether it is handled exclusively in the S3A code. This option "fs.s3a.experimental.aws.s3.throttling" is true by default. If set to false: everything is handled in the S3A client. While this means that metrics may be more accurate, it may mean that throttling failures in helper threads of the AWS SDK (especially those used in copy/rename) may not be handled properly. This is experimental, and should be left at "true" except when seeking more detail about throttling rates.

Description

Currently AWS S3 throttling is initially handled in the AWS SDK, only reaching the S3 client code after it has given up.

This means we don't always directly observe when throttling is taking place.

Proposed:

- disable throttling retries in the AWS client library
- add a quantile for the S3 throttle events, as DDB has
- isolate counters of s3 and DDB throttle events to classify issues better

Because we are taking over the AWS retries, we will need to expand the initial delay en retries and the number of retries we should support before giving up.

Also: should we log throttling events? It could be useful but there is a risk of logs overloading especially if many threads in the same process were triggering the problem.

Proposed: log at debug.

Note: if S3 bucket logging is enabled then throttling events will be recorded as 503 responses in the logs. If the hadoop version contains the audit logging of ~~HADOOP-17511~~, this can be used to identify operations/jobs/users which are triggering problems.

Issue Links

is related to

HADOOP-13811	s3a: getFileStatus fails with com.amazonaws.AmazonClientException: Failed to sanitize X...	RESOLVED
HADOOP-17935	Spark job stuck in S3A StagingCommitter::setupJob	RESOLVED

links to

- GitHub Pull Request #1814
- GitHub Pull Request #1826

Activity

Steve Loughran added a comment - 29/Jan/20 18:48

between prepaid IO and load; ITestDynamoDBMetadataStoreScale is the example of this.

special callout for test setup

```
[ERROR] test_070_putDirMarker(org.apache.hadoop.fs.s3a.s3guard.ITestDynamoDBMetadataStoreScale)  Time elapsed: 314.323
s  <<< ERROR!
org.apache.hadoop.fs.s3a.AWSServiceThrottledException: getVersionMarkerItem on ../VERSION:
com.amazonaws.services.dynamodbv2.model.ProvisionedThroughputExceededException: The level of configured provisioned
throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API. (Service:
AmazonDynamoDBv2; Status Code: 400; Error Code: ProvisionedThroughputExceededException; Request ID:
LUKART1RQBVKV0T7BPURUN95QVVV4KQNSO5AEMVJF66Q9ASUAAJG): The level of configured provisioned throughput for the table was
exceeded. Consider increasing your provisioning level with the UpdateTable API. (Service: AmazonDynamoDBv2; Status
Code: 400; Error Code: ProvisionedThroughputExceededException; Request ID:
LUKART1RQBVKV0T7BPURUN95QVVV4KQNSO5AEMVJF66Q9ASUAAJG)
        at
org.apache.hadoop.fs.s3a.s3guard.ITestDynamoDBMetadataStoreScale.createMetadataStore(ITestDynamoDBMetadataStoreScale.jav
        at
org.apache.hadoop.fs.s3a.s3guard.ITestDynamoDBMetadataStoreScale.setup(ITestDynamoDBMetadataStoreScale.java:162)
Caused by: com.amazonaws.services.dynamodbv2.model.ProvisionedThroughputExceededException: The level of configured
provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable
API. (Service: AmazonDynamoDBv2; Status Code: 400; Error Code: ProvisionedThroughputExceededException; Request ID:
LUKART1RQBVKV0T7BPURUN95QVVV4KQNSO5AEMVJF66Q9ASUAAJG)
        at
org.apache.hadoop.fs.s3a.s3guard.ITestDynamoDBMetadataStoreScale.createMetadataStore(ITestDynamoDBMetadataStoreScale.jav
        at
org.apache.hadoop.fs.s3a.s3guard.ITestDynamoDBMetadataStoreScale.setup(ITestDynamoDBMetadataStoreScale.java:162)
```

Note how long the retry count was there. We were backing off big but it still failed on us.

Looking at the AWS metrics, part of the fun is that the way bursty traffic is handled, you may get your capacity at the time of the initial load, but get blocked after. That is: the throttling may not happen under load, but during the next time a low-load API call is made.

Also, S3GuardTableAccess isn't retrying, and some code in tests and the purge/dump table entry points go on to fail when throttling happens when iterating through scans. Fix: you can ask a DDBMetastore to wrap your scan with one bonded to its retry and metrics...plus use of this where appropriate.

ITestDynamoDBMetadataStoreScale is really slow; either the changes make it worse, or its always been really slow and we haven't noticed as it was happening during the (slow) parallel test runs. Proposed: we review it, look at what we want to show and then see if we can make things fail faster

Steve Loughran added a comment - 30/Jan/20 13:55

FWIW, in a bulk DELETE operation, it is the #of objects deleted which sets the limit, not the number of calls.

Steve Loughran added a comment - 02/Feb/20 14:43

I've realised and that if we handle all failures in the S3A code, throttling on multi part copies may not be recovered from.

We delegate that work to the transfer manager into the SDK which partitions the file, copies each part in a separate thread/HTTP request and then coalesces. The only retry all logic our client puts around that is the full copy operation -farriers on individual part copy made immediately escalate into the the failure of the full copy. So imagine if you are copying a 2 GB file and a one 128 MB copy failed - the whole operation would fail and need to be retried.

We don't want that.

Ultimately we are going to have to move away from transfer manager -we have discussed that in the past and given it is not in the 2.0 SDK something we will be forced to do next year anyway. It's just been clear this would be an extra piece of work we didn't really want to do until forced just because of the effort it would take -at least 2-3 weeks and more scale tests to write.

I don't want to have to do it right now on what should be a small throttling fix.

Plan:

- add a switch to throttle in the AWS client SDK or not -but leave on by default
- but shrink page size of a bulk delete to 200 entries, so it is less likely to trigger overloads and the retries less traumatic.
- issue: hard code that vs configurable? fixed: simplifies test, avoids people recreating the throttle problem and intentionally. Configurable: lets you alter the settings during debugging without having to push out a whole new release

- Add a special throttling log4j log; logs normal throttling at info, DeleteObjects at warn + size of entry and first & last paths. That way, blame assignment even without full request context propagation,

---

Hudson added a comment - 13/Feb/20 19:26

SUCCESS: Integrated in Jenkins build Hadoop-trunk-Commit #17953 (See https://builds.apache.org/job/Hadoop-trunk-Commit/17953/)
~~HADOOP-16823~~. Large DeleteObject requests are their own Thundering Herd. (stevel: rev 56dee667707926f3796c7757be1a133a362f05c9)

- (edit) hadoop-tools/hadoop-aws/src/site/markdown/tools/hadoop-aws/testing.md
- (edit) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/s3a/s3guard/ITestDynamoDBMetadataStore.java
- (edit) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/s3a/s3guard/ThrottleTracker.java
- (edit) hadoop-common-project/hadoop-common/src/main/resources/core-default.xml
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/s3guard/DumpS3GuardDynamoTable.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/s3guard/S3GuardTableAccess.java
- (edit) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/s3a/impl/ITestPartialRenamesDeletes.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/DefaultS3ClientFactory.java
- (add) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/s3guard/RetryingCollection.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/S3AInstrumentation.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/s3guard/DynamoDBMetadataStore.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/S3AFileSystem.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/impl/DeleteOperation.java
- (edit) hadoop-common-project/hadoop-common/src/test/java/org/apache/hadoop/fs/contract/AbstractContractRenameTest.java
- (add) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/impl/BulkDeleteRetryHandler.java
- (edit) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/s3a/scale/ITestS3ADeleteManyFiles.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/impl/InternalConstants.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/impl/RenameOperation.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/s3guard/PurgeS3GuardDynamoTable.java
- (edit) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/s3a/commit/ITestCommitOperations.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/Statistic.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/impl/CallableSupplier.java
- (edit) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/contract/s3a/ITestS3AContractRename.java
- (add) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/s3a/scale/ILoadTestS3ABulkDeleteThrottling.java
- (edit) hadoop-common-project/hadoop-common/src/test/java/org/apache/hadoop/fs/contract/ContractTestUtils.java
- (edit) hadoop-tools/hadoop-aws/src/main/java/org/apache/hadoop/fs/s3a/Constants.java
- (edit) hadoop-tools/hadoop-aws/src/test/java/org/apache/hadoop/fs/s3a/s3guard/ITestDynamoDBMetadataStoreScale.java

People

Assignee:

Steve Loughran

Reporter:

Steve Loughran

Votes:

0 Vote for this issue

Watchers:

7 Start watching this issue

Dates

Created:

21/Jan/20 18:41

Updated:

21/Apr/22 13:33

Resolved:

13/Feb/20 20:07