HADOOP-17377

brandonvin

2020-11-13

Contents

Chapter 1

Root issue HADOOP-17377

1.1 Summary

ABFS: Frequent HTTP429 exceptions with MSI token provider

1.2 Description

Summary
The MSI token provider fetches auth tokens from the local instance metadata service.
The instance metadata service documentation states a limit of 5 requests per second: [https://docs.microsoft.com/en-us/azure/virtual-machines/windows/instance-metadata-service#error-and-debugging] which is fairly low.

Using ABFS and the MSI token provider, especially when there are multiple JVMs running on the same host, ABFS frequently throws HTTP429 throttled exception. The implementation for fetching a token from MSI uses ExponentialRetryPolicy, however ExponentialRetryPolicy does not retry on status code 429, from my read of the code.

Perhaps the ExponentialRetryPolicy could retry HTTP429 errors? I'm not sure what other ramifications that would have.

Environment
This is in the context of Spark clusters running on Azure Virtual Machine Scale Sets. The Virtual Machine Scale Set is configured with a user-assigned identity. The Spark cluster is configured to download application JARs from an 'abfs://' path, and auth to the storage account with the MSI token provider. The Spark version is 2.4.4. Hadoop libraries are version 3.2.1. More details on the Spark configuration: each VM runs 6 executor processes, and each executor process uses 5 cores. The FileSystem objects are singletons within each JVM due to the internal cache, so on each VM, I expect my setup is making 6 rapid requests to the instance metadata service when the executor is starting up and fetching the JAR.

Impact
In my particular use case, the download operation itself is wrapped with 3 additional retries. I have never seen the download cause all the tries to be exhausted and fail. In the end, it seems to contribute mostly noise and slowness from the retries. However, having the HTTP429 handled robustly in the ABFS implementation would help application developers succeed and write cleaner code without wrapping individual ABFS operations with retries.

Example

Here's an example error message and stack trace. It's always the same stack trace. This appears in my logs a few hundred to low thousands of times a day.

```
AADToken: HTTP connection failed for getting token from AzureAD. Http response:
429 null
Content-Type: application/json; charset=utf-8 Content-Length: 90 Request
ID:  Proxies: none
First 1K of Body: {"error":"invalid_request","error_description":"Temporarily
throttled, too many requests"}
at org.apache.hadoop.fs.azurebfs.services.AbfsRestOperation.executeHttpOperation(AbfsRes
at org.apache.hadoop.fs.azurebfs.services.AbfsRestOperation.execute(AbfsRestOperation.ja
at org.apache.hadoop.fs.azurebfs.services.AbfsClient.getAclStatus(AbfsClient.java:506)
at org.apache.hadoop.fs.azurebfs.services.AbfsClient.getAclStatus(AbfsClient.java:489)
at org.apache.hadoop.fs.azurebfs.AzureBlobFileSystemStore.getIsNamespaceEnabled(AzureBlo
at org.apache.hadoop.fs.azurebfs.AzureBlobFileSystemStore.getFileStatus(AzureBlobFileSys
at org.apache.hadoop.fs.azurebfs.AzureBlobFileSystem.getFileStatus(AzureBlobFileSystem.j
at org.apache.hadoop.fs.FileSystem.isFile(FileSystem.java:1717)
at org.apache.spark.util.Utils$.fetchHcfsFile(Utils.scala:747)
at org.apache.spark.util.Utils$.doFetchFile(Utils.scala:724)
at org.apache.spark.util.Utils$.fetchFile(Utils.scala:496)
at org.apache.spark.executor.Executor.$anonfun$updateDependencies$7(Executor.scala:812)
at org.apache.spark.executor.Executor.$anonfun$updateDependencies$7$adapted(Executor.sca
at scala.collection.TraversableLike$WithFilter.$anonfun$foreach$1(TraversableLike.scala
at scala.collection.mutable.HashMap.$anonfun$foreach$1(HashMap.scala:149)
at scala.collection.mutable.HashTable.foreachEntry(HashTable.scala:237)
at scala.collection.mutable.HashTable.foreachEntry$(HashTable.scala:230)
at scala.collection.mutable.HashMap.foreachEntry(HashMap.scala:44)
at scala.collection.mutable.HashMap.foreach(HashMap.scala:149)
at scala.collection.TraversableLike$WithFilter.foreach(TraversableLike.scala:791)
at org.apache.spark.executor.Executor.org$apache$spark$executor$Executor$$updateDepender
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:375)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)
```

CC [~mackrorysd], [~stevel@apache.org]

1.3 Attachments

No attachments

1.4 Comments

1. **stevel@apache.org:** [~snvijaya]