Hadoop Common / HADOOP-1411

AlreadyBeingCreatedException from task retries

Details

Type:	Bug	Status:	CLOSED
Priority:	Blocker	Resolution:	Fixed
Affects Version/s:	0.13.0	Fix Version/s:	0.13.0
Component/s:	None		
Labels:	None		

Description

~~HADOOP-1407~~ indicates 2 bugs: a mapred bug which will be fixed as part of 1407, and a DFSClient bug that will be fixed here.

Note that the test run in 1407 was without speculation.

Attachments

createRetry.patch	6 kB	23/May/07 20:55
createRetry1.patch	5 kB	24/May/07 21:23
createRetry-tw.patch	6 kB	25/May/07 08:41
createRetry-tw2.patch	10 kB	25/May/07 12:45

Issue Links

is related to

HADOOP-1396 FileNotFound exception on DFS block CLOSED

Activity

Nigel Daley created issue - 22/May/07 04:57

Devaraj Das made changes - 22/May/07 07:06

Field	Original Value	New Value
Link		This issue duplicates ~~HADOOP-1396~~ [~~HADOOP-1396~~]

Devaraj Das made changes - 22/May/07 07:09

Link		This issue is related to ~~HADOOP-1396~~ [~~HADOOP-1396~~]

Devaraj Das made changes - 22/May/07 07:09

Link	This issue duplicates ~~HADOOP-1396~~ [~~HADOOP-1396~~]	

Nigel Daley made changes - 22/May/07 17:00

Assignee	dhruba borthakur [dhruba]	Hairong Kuang [hairong]

Hairong Kuang added a comment - 23/May/07 01:28

I did put some logic in the patch to ~~HADOOP-1263~~ to handle retries for AlreadyBeingCreatedException. But it turned out it did not work. The problem is that any IPC call returns only RemoteException. When a server operation throws

AlreadyBeingCreatedException, the client only gets RemoteException and it has to examine the content of RemoteException to find out the real exception. So the DFSClient retry framework implemented in ~~HADOOP-1263~~ never catches an AlreadyBeingCreatedException and therefore it never gets retried.

I'd like to propose the following changes to the general retry framework so it is able to handle RemoteException well:
Method shouldRetry of RetryPolicies.exceptionDependentRetry checks if the exception is a RemoteException. If yes, find out the retry policy for the real exception from the exceptionToPolicyMap. Because RemoteException contains only the class name of the real exception, I would also propose to change the exceptionToPolicyMap to map an exception class name to a retry policy. Currently it is a map from a exception class to a retry policy.

Hairong Kuang added a comment - 23/May/07 20:55

Tom, could you please review the patch to see if you are happy with my changes to the io retry framework?

Hairong Kuang made changes - 23/May/07 20:55

Attachment	createRetry.patch [12358030]

Thomas White added a comment - 24/May/07 14:49

Hairong,

+0

While the patch will solve the problem, I think if possible it would be better not to have a special case for ipc.RemoteException embedded in RetryPolicies.retryByException. Instead we could have another static factory, RetryPolicies.retryByRemoteException say, that should be used for remote interfaces. Also, note that the signature for this doesn't need to have a Map keyed by String: keying by Exception provides more type safety (at the slight cost of creating a new Map keyed by String for internal use) and consistency with RetryPolicies.retryByException.

What do you think?

Hairong Kuang added a comment - 24/May/07 21:23

The new patch reflects Tom's suggestions.

> I think if possible it would be better not to have a special case for ipc.RemoteException embedded in RetryPolicies.retryByException. Instead we could have another static factory, RetryPolicies.retryByRemoteException say, that should be used for remote interfaces.

I agree this interface is cleaner. But one disadvantage is that it is harder for a client to use the framework to support RemoteException. A client needs to build two exception2Policy maps and shouldRetry needs to be recusively called one more time to search one more map.

> Also, note that the signature for this doesn't need to have a Map keyed by String: keying by Exception provides more type safety (at the slight cost of creating a new Map keyed by String for internal use) and consistency with RetryPolicies.retryByException.

I agree that keying by Exception provides more type safety. But again calling Class.forName is more costly than calling Class.getName. That's why I perfer usign class name as the map's key. Alternatively we could enforce type safety by providing an "add" handler to the map with Exception as a parameter and internally implements the map using the class name as the key. But we could do it in a different jira.

Hairong Kuang made changes - 24/May/07 21:23

Attachment	createRetry1.patch [12358161]

Thomas White added a comment - 25/May/07 08:41

>I agree this interface is cleaner. But one disadvantage is that it is harder for a client to use the framework to support RemoteException. A client needs to build two exception2Policy maps and shouldRetry needs to be recusively called one more time to search one more map.

It is slightly harder for clients to set up, but I would argue it is clearer. I wouldn't worry about the cost of an extra call to shouldRetry since it will be dwarfed by the time between retries.

>I agree that keying by Exception provides more type safety. But again calling Class.forName is more costly than calling Class.getName. That's why I perfer usign class name as the map's key. Alternatively we could enforce type safety by providing an "add" handler to the map with Exception as a parameter and internally implements the map using the class name as the key. But we could do it in a different jira.

Agreed. I was imagining turning the Map<Exception, RetryPolicy> to a Map<String, RetryPolicy> on construction of RemoteExceptionDependentRetry rather then using Class.getName. I've attached a patch which does this. Does this work OK for you?

Finally, it would be nice to have a test for this new behaviour.

Thomas White made changes - 25/May/07 08:41

Attachment		createRetry-tw.patch [12358201]

Thomas White added a comment - 25/May/07 12:45

New patch with unit test.

Thomas White made changes - 25/May/07 12:45

Attachment		createRetry-tw2.patch [12358232]

Owen O'Malley added a comment - 25/May/07 20:25

I think the underlying source of the problem is that the Map parameter is too big. I think it would be better if the handlers were done more explicitly as:

```
addHandler(Class<? extends Exception>, RetryHandler);
```

But I think this patch should go in since it fixes the problem and we can address any further concerns later.

Thomas White made changes - 25/May/07 20:40

Status	Open [1]	Patch Available [10002]

Hadoop QA added a comment - 25/May/07 21:13

+1

http://issues.apache.org/jira/secure/attachment/12358232/createRetry-tw2.patch applied and successfully tested against trunk revision r541754.

Test results: http://lucene.zones.apache.org:8080/hudson/job/Hadoop-Patch/205/testReport/
Console output: http://lucene.zones.apache.org:8080/hudson/job/Hadoop-Patch/205/console

Thomas White added a comment - 25/May/07 21:27

I've just committed this. Thanks Hairong!

Thomas White made changes - 25/May/07 21:27

Resolution		Fixed [1]
Status	Patch Available [10002]	Resolved [5]

Hadoop QA added a comment - 26/May/07 11:22

Integrated in Hadoop-Nightly #101 (See http://lucene.zones.apache.org:8080/hudson/job/Hadoop-Nightly/101/)

Doug Cutting made changes - 08/Jun/07 20:40

Status	Resolved [5]	Closed [6]

Owen O'Malley made changes - 08/Jul/09 16:42

Component/s	dfs [12310710]	

Transition	Time In Source Status	Execution Times
Thomas White made transition - 25/May/07 20:40 OPEN → PATCH AVAILABLE	3d 15h 43m	1
Thomas White made transition - 25/May/07 21:27		

PATCH AVAILABLE → RESOLVED	46m 48s	1
Doug Cutting made transition - 08/Jun/07 20:40 RESOLVED → CLOSED	13d 23h 13m	1

People

Assignee:

Hairong Kuang

Reporter:

Nigel Daley

Votes:

0 Vote for this issue

Watchers:

1 Start watching this issue

Dates

Created:

22/May/07 04:57

Updated:

08/Jul/09 16:42

Resolved:

25/May/07 21:27