Item 143
git_comments:

1. rpcTimeout overwrites pingInterval
2. * Construct a client-side proxy object that implements the named protocol, * talking to a server at the named address.
3. shutdown a data node
4. create a file
5. get block info
6. * * The following test first creates a file. * It verifies the block information from a datanode. * Then, it stops the DN and observes timeout on connection attempt.
7. shutdown a data node
8. create a file
9. get block info
10. * * The following test first creates a file. * It verifies the block information from a datanode. * Then, it stops the DN and observes timeout on connection attempt.
11. start server
12. start client
13. set timeout to be less than MIN_SLEEP_TIME
14. set timeout to be bigger than 3*ping interval

git_commits:

1. **summary:** HADOOP-6889. Make RPC to have an option to timeout - backport to 0.20-security. Contributed by John George and Ravi Prakash.
   message: HADOOP-6889. Make RPC to have an option to timeout - backport to 0.20-security. Contributed by John George and Ravi Prakash. git-svn-id: https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20-security@1153219 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Make RPC to have an option to timeout
   description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
   label: code-design
2. **summary:** Make RPC to have an option to timeout
   description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to

retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

3. **summary:** Make RPC to have an option to timeout
   description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
4. **summary:** Make RPC to have an option to timeout
   description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
5. **summary:** Make RPC to have an option to timeout
   description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
6. **summary:** Make RPC to have an option to timeout
   description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
7. **summary:** Make RPC to have an option to timeout
   description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is

still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
label: code-design

8. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
9. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
10. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
label: test
11. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

12. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
13. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
14. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
15. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
16. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a

SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

17. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
18. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
19. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
20. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
    label: code-design
21. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout

exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

22. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
    label: code-design
23. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
24. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
25. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
26. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example,

for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

27. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
28. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
29. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
30. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
    label: code-design
31. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to

retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

32. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
33. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
34. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
35. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
36. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is

still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
label: test

37. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
38. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
39. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
label: code-design
40. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

41. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
42. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
43. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
44. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
45. **summary:** Make RPC to have an option to timeout
description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a

SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

46. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
47. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
48. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
49. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
50. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings

that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

51. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
52. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
53. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
54. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
55. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout

exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

56. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
57. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
58. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.
    label: code-design
59. **summary:** Make RPC to have an option to timeout
    description: Currently Hadoop RPC does not timeout when the RPC server is alive. What it currently does is that a RPC client sends a ping to the server whenever a socket timeout happens. If the server is still alive, it continues to wait instead of throwing a SocketTimeoutException. This is to avoid a client to retry when a server is busy and thus making the server even busier. This works great if the RPC server is NameNode. But Hadoop RPC is also used for some of client to DataNode communications, for example, for getting a replica's length. When a client comes across a problematic DataNode, it gets stuck and can not switch to a different DataNode. In this case, it would be better that the client receives a timeout exception. I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

jira_issues_comments:

1. **body:** On a second thought, it would be nice if RPC proxies could configure differently. For example, one may want to have timeout on RPCs to DataNodes but no timeout on RPCs to NameNode. So I propose to add an additional parameter, maxRpcWaitingTime, to RPC#getProxy, meaning for every RPC call to this

proxy, SocketTimeout is thrown if a client has not received a response in maxRpcWaitingTime. If maxRpcWaitingTime is zero, a RPC call will not timeout. In this way, a client could configure maxRpcWaitingTime differently for its communicatations to NameNode and DataNodes.
label: code-design

2. Hi Hairong. I agree that it should be per-proxy, and also that this is useful. We have a customer who has occasionally run into this with problematic "stuck" datanodes blocking pipeline recovery indefinitely (at least until ops notices the hung machine and powers it down, causing TCP connection to drop).
3. As I recall, RPC calls used to have a timeout that, combined with the retry proxy, would spiral load on servers. But I think timeouts when retries go to a different server should not have the same problem.
4. This patch passes a rpcTimout parameter to RPC#getProxy method. A a non-positive rpcTimeout means that RPC does not timeout as the default behavior. If rpcTimeout is positive, a RPC client throws SocketTimeoutException if the client has not received a response in rpcTimeout period.
5. ipcTimeout1.patch additionally makes waitForProxy has rpcTimeout as a parameter.
6. +1 this patch looks reasonable to me, pending Hudson's scrutiny.
7. **body:** -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12451180/ipcTimeout1.patch against trunk revision 981714. +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. -1 javadoc. The javadoc tool appears to have generated 1 warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: http://hudson.zones.apache.org/hudson/job/Hadoop-Patch-h4.grid.sp2.yahoo.net/662/testReport/ Findbugs warnings: http://hudson.zones.apache.org/hudson/job/Hadoop-Patch-h4.grid.sp2.yahoo.net/662/artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Checkstyle results: http://hudson.zones.apache.org/hudson/job/Hadoop-Patch-h4.grid.sp2.yahoo.net/662/artifact/trunk/build/test/checkstyle-errors.html Console output: http://hudson.zones.apache.org/hudson/job/Hadoop-Patch-h4.grid.sp2.yahoo.net/662/console This message is automatically generated.
   label: code-design
8. This fixed the failed tests.
9. Resubmit since the previous one seemed to stuck.
10. **body:** Hudson is down. I ran ant clean test on my linux box twice. They all passed: BUILD SUCCESSFUL Total time: 11 minutes 8 seconds BUILD SUCCESSFUL Total time: 9 minutes 55 seconds
    label: test
11. I've just committed this!
12. I know I'm a little late to the party. But I'm just wondering if it's better to pass rcpTimeout using the existing conf param rather than adding a new parameter to getProxy()?
13. Kan, thanks for taking a look at the patch! I thought about introducing a configuration parameter. But clients or DataNodes want to have timeout for RPCs to DataNodes but no timeout for RPCs to NameNodes. Adding a rpcTimeout parameter makes this easy.
14. I see.
15. {code} - @Override + @Override // simply use the default Object#hashcode() ? public int hashCode() { - return (address.hashCode() + PRIME * System.identityHashCode(protocol)) ^ - (ticket == null ? 0 : ticket.hashCode()); + return (address.hashCode() + PRIME * (+ PRIME * (+ PRIME * System.identityHashCode(protocol) ^ + System.identityHashCode(ticket) +) ^ System.identityHashCode(rpcTimeout) +)); {code} I wonder if System.identityHashCode(rpcTimeout) is correct in the above code. I think Java does an autoboxing to get an Integer object and what identityHashCode() returns is the address of this object (it doesn't call the object's hashcode() method). So even if two ConnectionId objects have the same rpcTimeout you will get two different hashcodes. I think you can simply use rpcTimeout itself as the hashcode.
16. > I think you can simply use rpcTimeout itself as the hashcode. If that's what you intended. I haven't considered its randomness.
17. PS. Maybe Java optimizes on reusing the same object for the same integer. I don't know if this is something we can depend on.
18. Kan, thanks for your insight. It seems that you are right, identifyHashcode is based on references. Do you mind if we simply use rpcTime as the hashcode?

19. If we can confirm Java always uses the same object for the same integer, your code is correct. Otherwise, I'm fine with using rpcTimeout as hashcode. I've seen recommendations on computing hashcode iteratively as follows. {code} hash = PRIME * hash + component_value {code} Seems we are doing ^ instead of +. Does it matter?
20. **body:** It looks that Java returns the same value for every call of System.indentifyHashcode(x) as long as int x remains the same. But I feel more comfortable using x as hashcode directly. Either ^ or + should be fine. But I think we should consistently use either ^ or +.
    label: code-design
21. This patch's hashcode() implementation also changed the UGI semantics. It used to be that two UGI's having the same subject object are equal. Now they are not equal. I hope to fix both issues in HADOOP-6907.
22. **body:** Hay Hairong, I have seen waitForProxy is passing 0 as rpcTimeOut. It is hardcoded value. {code} return waitForProtocolProxy(protocol, clientVersion, addr, conf, 0, connTimeout); {code} If user wants to control this value then , how can he configure? Here we have a situation, where clients are waiting for long time.HDFS-1880. I thought, this issue can solve that problem. But how this can be controlled by the user in Hadoop. {quote} I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever. {quote} We have choosen this implementation for our cluster. I am just checking , whether i can use rpcTimeOut itself to control. (since this change already committed). Can you please clarify more?
    label: code-design
23. Can you please clarify more?
24. I would like to port this to branch-20-security.
25. Attaching a patch for .20-branch-security. Could someone please review this?
26. Attaching the patch with new name so that it is obvious it is for .20
27. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12487319/HADOOP-6889-for20.patch against trunk revision 1148933. +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 9 new or modified tests. -1 patch. The patch command could not apply the patch. Console output: https://builds.apache.org/job/PreCommit-HADOOP-Build/756//console This message is automatically generated.
28. It was not my intention to remove "0.20-append" and "0.22" form the Fix Versions. So adding it back...
29. Output of test-patch on my desktop: [exec] [findbugs] Classes needed for analysis were missing [exec] [findbugs] Output saved to /home/johngeo/hadoop/branch-0.20-security/build/test/findbugs/hadoop-findbugs-report.xml [exec] [xslt] Processing /home/johngeo/hadoop/branch-0.20-security/build/test/findbugs/hadoop-findbugs-report.xml to /home/johngeo/hadoop/branch-0.20-security/build/test/findbugs/hadoop-findbugs-report.html [exec] [xslt] Loading stylesheet /hadoop/config/findbugs-1.3.9/src/xsl/default.xsl [exec] [exec] BUILD SUCCESSFUL [exec] Total time: 6 minutes 33 seconds [exec] [exec] [exec] [exec] [exec] +1 overall. [exec] [exec] +1 @author. The patch does not contain any @author tags. [exec] [exec] +1 tests included. The patch appears to include 9 new or modified tests. [exec] [exec] +1 javadoc. The javadoc tool did not generate any warning messages. [exec] [exec] +1 javac. The applied patch does not increase the total number of javac compiler warnings. [exec] [exec] +1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings. [exec] [exec] [exec] [exec] [exec]
    ========== [exec]
    ========== [exec]
    Finished build. [exec]
    ========== [exec]
    ========== [exec]
    [exec]
30. **body:** Hi John, I have seen waitForProxy is passing 0 as rpcTimeOut. It is hardcoded value. {code} return waitForProtocolProxy(protocol, clientVersion, addr, conf, 0, connTimeout); {code} If user wants to control this value then , how can he configure? Here we have a situation, where clients are waiting for long time.HDFS-1880. I thought, HADOOP-6889 can solve that problem. But how this can be controlled by the user in Hadoop (looks no configuration parameters available). {quote} I plan to add a new configuration ipc.client.max.pings that specifies the max number of pings that a client could try. If a response can not be received after the specified max number of pings, a SocketTimeoutException is thrown. If this configuration property is not set, a client maintains the current semantics, waiting forever.

{quote} We have choosen this implementation for our cluster. I am just checking , whether i can use rpcTimeOut itself to control. (since this change already committed). Can you please clarify more? Can you just check HDFS-1880. @Hairong {quote} I thought about introducing a configuration parameter. But clients or DataNodes want to have timeout for RPCs to DataNodes but no timeout for RPCs to NameNodes. Adding a rpcTimeout parameter makes this easy. {quote} I think considering HA, clients and NameNode also requires some timeout. If Active goes down, then clients should not wait in timeouts right?
label: code-design

31. hey Uma, I have responded to your questions in HDFS-1880
32. John, there was a related change - HADOOP-6907, are you planning to port that to 0.20.s as well?
33. Thanks for the pointer Suresh. Seems like HADOOP-6907 is already in branch-0.20-security.
34. I just checked and confirmed that HADOOP-6907 was submitted to 0.20-security sustaining "trunk" just before the 20.203 branch was created. Updated the "Fix Versions" of HADOOP-6907 to reflect that correctly.
35. @Uma, I agree that the new param you suggest should be dealt with in a separate ticket, not in this long-closed ticket, which has been re-opened just for a back-port to the sustaining branch. Thanks.
36. **body:** Unfortunately this patch diverges a lot from the trunk patch (presumably because of 0.20/0.23 code tree divergence, of course), so I could not usefully diff the patches and had to review this like a new patch. In terms of code review, I found no problems. But it's a large enough patch that we are dependent on thorough unit testing to be confident in the patch. So I have two questions: 1. I see a single new test case, TestIPC.testIpcTimeout(), that tests the lowest-level timeout functionality, between a client and a TestServer server. However, I do not see any test cases that check whether the integration of that timeout functionality with, eg, the InterDatanodeProtocol works as expected. (The mod to TestInterDatanodeProtocol merely adapts to the change, it does not test the change.) Similarly, no test of timeout in the context of DFSClient with a MiniDFSCluster. Granted the original patch to trunk doesn't test these either. But do you feel confident in the patch without such additional tests, and why? 2. Are the variances between the trunk and v20 patches due only to code tree divergence, or are there changes added to the v20 patch that are not in v23 and perhaps should be? Thanks.
label: test
37. Thanks for your review and comments Matt. {quote}1. I see a single new test case, TestIPC.testIpcTimeout(), that tests the lowest-level timeout functionality, between a client and a TestServer server. However, I do not see any test cases that check whether the integration of that timeout functionality with, eg, the InterDatanodeProtocol works as expected. (The mod to TestInterDatanodeProtocol merely adapts to the change, it does not test the change.) Similarly, no test of timeout in the context of DFSClient with a MiniDFSCluster. Granted the original patch to trunk doesn't test these either. But do you feel confident in the patch without such additional tests, and why?{quote} I'm uploading a new patch with the added tests on behalf of John George. {quote}2. Are the variances between the trunk and v20 patches due only to code tree divergence, or are there changes added to the v20 patch that are not in v23 and perhaps should be? Thanks.{quote} John told me the variances are indeed only because of the tree divergence.
38. Results from test-patch {noformat} [exec] +1 overall. [exec] [exec] +1 @author. The patch does not contain any @author tags. [exec] [exec] +1 tests included. The patch appears to include 12 new or modified tests. [exec] [exec] +1 javadoc. The javadoc tool did not generate any warning messages. [exec] [exec] +1 javac. The applied patch does not increase the total number of javac compiler warnings. [exec] [exec] +1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings. [exec] [exec] [exec] [exec] [exec]
   ========== [exec]
   ========== [exec]
   Finished build. [exec]
   ========== [exec]
   ==========
   {noformat}
39. **body:** Thanks Ravi. The following things needed cleanup: * Comments for the two new methods needed fixing after copy from earlier source. * Wrong LOG stream in TestDFSClientRetries.testDFSClientTimeout(). * Spurious imports (probably snuck in due to Eclipse). Please review attached, and re-run the test. If okay with you we can commit.
label: code-design
40. Hi Matt, Thanks for the changes! Yes I ran the tests again and they all passed. Please commit the patch. +1

41. Patch is for 0.20-security branch.
42. +1 for code review. Committed to 0.20-security. Ravi / John, thanks for adding those two new test cases, they are just what the doctor ordered. Can you please port them up to v23 trunk, before we close this jira? Thanks.
43. The tests included with the previous patch was not testing the exact corresponding change. Hence, a new patch for tests are included (both for trunk and branch-20-security): ---------- branch-20-security patch-test results as follows: [exec] +1 overall. [exec] [exec] +1 @author. The patch does not contain any @author tags. [exec] [exec] +1 tests included. The patch appears to include 7 new or modified tests. [exec] [exec] +1 javadoc. The javadoc tool did not generate any warning messages. [exec] [exec] +1 javac. The applied patch does not increase the total number of javac compiler warnings. [exec] [exec] +1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings. [exec] ---------- trunk patch-test results: It was failing on empty patch as well for release audit and system test framework. [exec] BUILD FAILED [exec] /home/johngeo/hadoop/trunk/hdfs/src/test/aop/build/aop.xml:222: The following error occurred while executing this nclude 8 new or modified tests. [exec] [exec] +1 javadoc. The javadoc tool did not generate any warning messages. [exec] [exec] +1 javac. The apline: [exec] /home/johngeo/hadoop/trunk/hdfs/src/test/aop/build/aop.xml:203: The following error occurred while executing this line: [exec] /hplied patch does not increase the total number of javac compiler warnings. [exec] [exec] +1 findbugs. The patch does not introduce any nome/johngeo/hadoop/trunk/hdfs/src/test/aop/build/aop.xml:90: compile errors: 18 [exec] [exec] Total time: 35 seconds [exec] ew Findbugs (version 1.3.9) warnings. [exec] [exec] -1 release audit. The applied patch generated 2 release audit warnings (more than the trunk's current 0 warnings). [exec] [exec] -1 system test framework. The patch failed system test framework compile.
44. Regarding patchfile names: Please note that all these attachments have been targeted for branch-0.20-security, despite their unfortunate names (I also made a naming error with "20.3"): * HADOOP-6889.patch * HADOOP-6889-for20.patch * HADOOP-6889-for20.2.patch * HADOOP-6889-for20.3.patch * HADOOP-6889-for-20security.patch. The fact that all these are branch-0.20-security patches is clear from the comments accompanying each submission.
45. John, please re-submit the trunk patch to trigger CI build. Also, for consistency, could you please set the timeout in TestDFSClientRetries to 500 rather than 1000, to match the other test cases, unless there's a reason it needs to be different? Thanks.
46. Unit test update committed to branch-0.20-security. Thanks, John!
47. That was a +1 on the patch for branch-0.20-security.
48. Changing timeout in TestDFSClientRetries to 500 rather than 1000
49. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12491848/HADOOP-6889-fortrunk-2.patch against trunk revision . +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 8 new or modified tests. -1 patch. The patch command could not apply the patch. Console output: https://builds.apache.org/job/PreCommit-HADOOP-Build/91//console This message is automatically generated.
50. Integrated in Hadoop-Common-trunk-Commit #811 (See [https://builds.apache.org/job/Hadoop-Common-trunk-Commit/811/]) HDFS-1330 and HADOOP-6889. Added additional unit tests. Contributed by John George. mattf : http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1163463 Files : * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/CHANGES.txt * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/TestDFSClientRetries.java * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/server/datanode/TestInterDatanodeProtocol.java
51. Integrated in Hadoop-Hdfs-trunk-Commit #888 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk-Commit/888/]) HDFS-1330 and HADOOP-6889. Added additional unit tests. Contributed by John George. mattf : http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1163463 Files : * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/CHANGES.txt * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/TestDFSClientRetries.java * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/server/datanode/TestInterDatanodeProtocol.java
52. Integrated in Hadoop-Mapreduce-trunk-Commit #821 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk-Commit/821/]) HDFS-1330 and HADOOP-6889. Added additional unit tests. Contributed by John George. mattf : http://svn.apache.org/viewcvs.cgi/?root=Apache-

SVN&view=rev&rev=1163463 Files : * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/CHANGES.txt * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/TestDFSClientRetries.java * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/server/datanode/TestInterDatanodeProtocol.java

53. Submitted patch to HDFS-1330 to get correct run of test-patch. Passed; please see HDFS-1330 for details.
54. +1 for code review. Thanks, John! Committed to trunk. Also asked Arun if he wanted this in branch-0.23, he said yes. Committed to v23.
55. Hairong, can we close this, or do you want it held open for a corresponding patch to 0.20-append? Thanks.
56. Integrated in Hadoop-Hdfs-trunk #777 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk/777/]) HDFS-1330 and HADOOP-6889. Added additional unit tests. Contributed by John George. mattf : http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1163463 Files : * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/CHANGES.txt * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/TestDFSClientRetries.java * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/server/datanode/TestInterDatanodeProtocol.java
57. Integrated in Hadoop-Mapreduce-trunk #802 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk/802/]) HDFS-1330 and HADOOP-6889. Added additional unit tests. Contributed by John George. mattf : http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1163463 Files : * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/CHANGES.txt * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/TestDFSClientRetries.java * /hadoop/common/trunk/hadoop-hdfs-project/hadoop-hdfs/src/test/java/org/apache/hadoop/hdfs/server/datanode/TestInterDatanodeProtocol.java
58. I think we should port this to 0.20-append. Otherwise the branch is unusable.
59. If no one is going to port this to 0.20-append, we should remove that version from the "Target Versions" list and close this jira.
60. Removing 20append from Target Versions since we have not heard back from Hairong. If needed in 20-append, please feel free to re-open this JIRA.
61. Resolving as fixed since this was fixed in all the expected target versions.
62. Closed upon release of 0.20.205.0