Item 160
git_comments:

git_commits:

1. **summary:** HDFS-8674. Improve performance of postponed block scans. Contributed by Daryn Sharp.
   message: HDFS-8674. Improve performance of postponed block scans. Contributed by Daryn Sharp. (cherry picked from commit 0d8a35bd6de5d2a5a9b816ca98f31975e94bd7c6) Conflicts: hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/server/blockmanagement/BlockManager.java

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
   label: test
2. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
3. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
   label: code-design
4. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
5. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes

with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.

6. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
7. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
   label: code-design
8. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
9. **summary:** Improve performance of postponed block scans
   description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
10. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
11. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
12. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses

multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.

13. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
14. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
15. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
16. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
17. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
18. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
19. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.

20. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
21. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
22. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
23. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
24. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
25. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
26. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
27. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes

with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.

28. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
29. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
30. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
31. **summary:** Improve performance of postponed block scans
    description: When a standby goes active, it marks all nodes as "stale" which will cause block invalidations for over-replicated blocks to be queued until full block reports are received from the nodes with the block. The replication monitor scans the queue with O(N) runtime. It picks a random offset and iterates through the set to randomize blocks scanned. The result is devastating when a cluster loses multiple nodes during a rolling upgrade. Re-replication occurs, the nodes come back, the excess block invalidations are postponed. Rescanning just 2k blocks out of millions of postponed blocks may take multiple seconds. During the scan, the write lock is held which stalls all other processing.
    label: code-design

jira_issues_comments:

1. **body:** No additional tests due to simple performance optimization that existing tests cover.
   label: test
2. \\ \\ | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:red}-1{color} | pre-patch | 15m 16s | Findbugs (version) appears to be broken on trunk. | | {color:green}+1{color} | @author | 0m 0s | The patch does not contain any @author tags. | | {color:red}-1{color} | tests included | 0m 0s | The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. | | {color:green}+1{color} | javac | 7m 30s | There were no new javac warning messages. | | {color:green}+1{color} | javadoc | 9m 38s | There were no new javadoc warning messages. | | {color:green}+1{color} | release audit | 0m 22s | The applied patch does not increase the total number of release audit warnings. | | {color:green}+1{color} | checkstyle | 0m 50s | There were no new checkstyle issues. | | {color:green}+1{color} | whitespace | 0m 0s | The patch has no lines that end in whitespace. | | {color:green}+1{color} | install | 1m 35s | mvn install still works. | | {color:green}+1{color} | eclipse:eclipse | 0m 34s | The patch built with eclipse:eclipse. | | {color:red}-1{color} | findbugs | 3m 24s | The patch appears to introduce 1 new Findbugs (version 3.0.0) warnings. | | {color:green}+1{color} | native | 3m 14s | Pre-build of native portion | | {color:green}+1{color} | hdfs tests | 162m 31s | Tests passed in hadoop-hdfs. | | | | 204m 57s | | \\ \\ || Reason || Tests || | FindBugs | module:hadoop-hdfs | \\ \\ || Subsystem || Report/Notes || | Patch URL |

http://issues.apache.org/jira/secure/attachment/12742164/HDFS-8674.patch | | Optional Tests | javadoc javac unit findbugs checkstyle | | git revision | trunk / 60b858b | | Findbugs warnings | https://builds.apache.org/job/PreCommit-HDFS-Build/11502/artifact/patchprocess/newPatchFindbugsWarningshadoop-hdfs.html | | hadoop-hdfs test log | https://builds.apache.org/job/PreCommit-HDFS-Build/11502/artifact/patchprocess/testrun_hadoop-hdfs.txt | | Test Results | https://builds.apache.org/job/PreCommit-HDFS-Build/11502/testReport/ | | Java | 1.7.0_55 | | uname | Linux asf902.gq1.ygridcore.net 3.13.0-36-lowlatency #63-Ubuntu SMP PREEMPT Wed Sep 3 21:56:12 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux | | Console output | https://builds.apache.org/job/PreCommit-HDFS-Build/11502/console | This message was automatically generated.

3. **body:** Thanks [~daryn]! * The motivation of randomization is to make sure we don't end up scanning the same blocks if there isn't much update to postponedMisreplicatedBlocks. But maybe that doesn't matter much as it will eventually be drained; also HashSet doesn't make order guarantee. * Does the latency of couple seconds come from HashSet iteration? Some quick test indicates iterating through 5M entries of a HashSet take around 50ms. * What is the reason to change from HashSet to LinkedHashSet, to have some order guarantee? HashSet is faster than LinkedHashSet. * During the scan, if a block is marked as POSTPONE, it will be removed from postponedMisreplicatedBlocks first and add it back later via rescannedMisreplicatedBlocks. Can it just remove those blocks from postponedMisreplicatedBlocks that aren't marked as POSTPONE without using rescannedMisreplicatedBlocks?
   label: code-design
4. [~daryn] / [~mingma], any update on this? Considering this for a 2.7.2 RC this weekend. Thanks.
5. \\ \\ | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:red}-1{color} | patch | 0m 0s | The patch command could not apply the patch during dryrun. | \\ \\ || Subsystem || Report/Notes || | Patch URL | http://issues.apache.org/jira/secure/attachment/12742164/HDFS-8674.patch | | Optional Tests | javadoc javac unit findbugs checkstyle | | git revision | trunk / 96677be | | Console output | https://builds.apache.org/job/PreCommit-HDFS-Build/13211/console | This message was automatically generated.
6. Moving this out of 2.7.2 as there's been no update in a while.
7. **body:** [~mingma] It's a LinkedHashSet to cover the cases you state. bq. The motivation of randomization is to make sure we don't end up scanning the same blocks if there isn't much update to postponedMisreplicatedBlocks. <...> What is the reason to change from HashSet to LinkedHashSet, to have some order guarantee? HashSet is faster than LinkedHashSet. <...> During the scan, if a block is marked as POSTPONE, it will be removed from postponedMisreplicatedBlocks first and add it back later via rescannedMisreplicatedBlocks. Can it just remove those blocks from postponedMisreplicatedBlocks that aren't marked as POSTPONE without using rescannedMisreplicatedBlocks? The LHS iterator is effectively functioning as a cheap circular array with acceptable insert/remove performance. Yes, the order guarantee ensures all blocks are visited w/o "skipping" to a random location. The block always has to be removed and re-inserted for the ordering to work. Otherwise the same blocks will be scanned over and over again. bq. Does the latency of couple seconds come from HashSet iteration? Some quick test indicates iterating through 5M entries of a HashSet take around 50ms. Yes, the cycles wasted to skip the through the set were the killer. A micro-benchmark doesn't capture the same chaotic runtime environment of over a 100 threads competing for resources. Kihwal says (ironically) the performance "improved" under 5M. In multiple production incidents, the postponed queue backed up with 10-20M+ blocks. Scans determined the blocks are all still postponed. Pre-patch, the cycles took up to a few seconds and averaged in the many hundreds of ms. Performance was obviously atrocious. Post-patch, same scans took no more than a few ms.
   label: code-design
8. [~mingma], comments?
9. Thanks [~daryn]. Your explanation makes sense. It is good to know the perf difference between "the # of items < 5M" and "larger number of items". +1 on the patch.
10. Updated patch since a data structure change prevented it from applying cleanly.
11. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 0s {color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s {color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s {color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} |

{color:green} mvninstall {color} | {color:green} 7m 30s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 40s {color} | {color:green} trunk passed with JDK v1.8.0_66 {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 41s {color} | {color:green} trunk passed with JDK v1.7.0_91 {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 15s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 51s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 14s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 52s {color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 1m 5s {color} | {color:green} trunk passed with JDK v1.8.0_66 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 1m 52s {color} | {color:green} trunk passed with JDK v1.7.0_91 {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 48s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 38s {color} | {color:green} the patch passed with JDK v1.8.0_66 {color} | | {color:red}-1{color} | {color:red} javac {color} | {color:red} 6m 16s {color} | {color:red} hadoop-hdfs-project_hadoop-hdfs-jdk1.8.0_66 with JDK v1.8.0_66 generated 1 new issues (was 32, now 32). {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 38s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 40s {color} | {color:green} the patch passed with JDK v1.7.0_91 {color} | | {color:red}-1{color} | {color:red} javac {color} | {color:red} 6m 57s {color} | {color:red} hadoop-hdfs-project_hadoop-hdfs-jdk1.7.0_91 with JDK v1.7.0_91 generated 1 new issues (was 34, now 34). {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 40s {color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 14s {color} | {color:red} Patch generated 1 new checkstyle issues in hadoop-hdfs-project/hadoop-hdfs (total was 161, now 160). {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 50s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 14s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s {color} | {color:green} Patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 2s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 1m 4s {color} | {color:green} the patch passed with JDK v1.8.0_66 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 1m 46s {color} | {color:green} the patch passed with JDK v1.7.0_91 {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 52m 16s {color} | {color:red} hadoop-hdfs in the patch failed with JDK v1.8.0_66. {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 50m 8s {color} | {color:red} hadoop-hdfs in the patch failed with JDK v1.7.0_91. {color} | | {color:red}-1{color} | {color:red} asflicense {color} | {color:red} 0m 25s {color} | {color:red} Patch generated 58 ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 128m 44s {color} | {color:black} {color} | \\ \\ || Reason || Tests || | JDK v1.8.0_66 Failed junit tests | hadoop.hdfs.TestDFSClientRetries | | | hadoop.hdfs.server.namenode.ha.TestDNFencingWithReplication | | | hadoop.hdfs.server.namenode.TestDecommissioningStatus | | | hadoop.hdfs.server.balancer.TestBalancerWithHANameNodes | | | hadoop.hdfs.server.namenode.ha.TestDNFencing | | | hadoop.hdfs.TestDFSStripedOutputStreamWithFailure050 | | | hadoop.hdfs.TestDFSUpgradeFromImage | | | hadoop.hdfs.TestSetrepDecreasing | | | hadoop.hdfs.server.blockmanagement.TestNodeCount | | | hadoop.hdfs.TestDFSStripedOutputStreamWithFailure | | | hadoop.hdfs.server.blockmanagement.TestBlocksWithNotEnoughRacks | | JDK v1.7.0_91 Failed junit tests | hadoop.hdfs.server.namenode.ha.TestDNFencingWithReplication | | | hadoop.hdfs.TestLeaseRecovery2 | | | hadoop.hdfs.server.balancer.TestBalancerWithHANameNodes | | | hadoop.hdfs.server.namenode.ha.TestDNFencing | | | hadoop.hdfs.TestSetrepDecreasing | | | hadoop.hdfs.server.blockmanagement.TestNodeCount | | | hadoop.hdfs.server.blockmanagement.TestBlocksWithNotEnoughRacks | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:0ca8df7 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12778098/HDFS-8674.patch | | JIRA Issue | HDFS-8674 | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux 4f4d2df9b67c 3.13.0-36-lowlatency #63-Ubuntu SMP PREEMPT Wed Sep 3 21:56:12 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/hadoop/patchprocess/precommit/personality/provided.sh | | git revision | trunk / 0f708d4 | |

findbugs | v3.0.0 | | javac | hadoop-hdfs-project_hadoop-hdfs-jdk1.8.0_66: https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/diff-compile-javac-hadoop-hdfs-project_hadoop-hdfs-jdk1.8.0_66.txt | | javac | hadoop-hdfs-project_hadoop-hdfs-jdk1.7.0_91: https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/diff-compile-javac-hadoop-hdfs-project_hadoop-hdfs-jdk1.7.0_91.txt | | checkstyle | https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/diff-checkstyle-hadoop-hdfs-project_hadoop-hdfs.txt | | unit | https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/patch-unit-hadoop-hdfs-project_hadoop-hdfs-jdk1.8.0_66.txt | | unit | https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/patch-unit-hadoop-hdfs-project_hadoop-hdfs-jdk1.7.0_91.txt | | unit test logs | https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/patch-unit-hadoop-hdfs-project_hadoop-hdfs-jdk1.8.0_66.txt https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/patch-unit-hadoop-hdfs-project_hadoop-hdfs-jdk1.7.0_91.txt | | JDK v1.7.0_91 Test Results | https://builds.apache.org/job/PreCommit-HDFS-Build/13903/testReport/ | | asflicense | https://builds.apache.org/job/PreCommit-HDFS-Build/13903/artifact/patchprocess/patch-asflicense-problems.txt | | modules | C: hadoop-hdfs-project/hadoop-hdfs U: hadoop-hdfs-project/hadoop-hdfs | | Max memory used | 75MB | | Powered by | Apache Yetus 0.2.0-SNAPSHOT http://yetus.apache.org | | Console output | https://builds.apache.org/job/PreCommit-HDFS-Build/13903/console | This message was automatically generated.

12. Daryn, from failed tests's logs, the latest patch uses LightWeightLinkedSet whose iterator doesn't support remove.
13. I'll try to rebase again.
14. [~daryn] / [~mingma], is it possible to get this in 2.7.3 in a week? Tx?
15. Haven't gotten any update, dropping this off into 2.7.4..
16. Added some more target versions for tracking. Seems like this would still be good to get in.
17. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 0s{color} | {color:blue} Docker mode activated. {color} | | {color:red}-1{color} | {color:red} patch {color} | {color:red} 0m 6s{color} | {color:red} HDFS-8674 does not apply to trunk. Rebase required? Wrong Branch? See https://wiki.apache.org/hadoop/HowToContribute for help. {color} | \\ \\ || Subsystem || Report/Notes || | JIRA Issue | HDFS-8674 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12778098/HDFS-8674.patch | | Console output | https://builds.apache.org/job/PreCommit-HDFS-Build/17194/console | | Powered by | Apache Yetus 0.4.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.
18. I'll update this patch.
19. Resolved minor conflicts.
20. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 0s{color} | {color:blue} Docker mode activated. {color} | | {color:red}-1{color} | {color:red} patch {color} | {color:red} 0m 9s{color} | {color:red} HDFS-8674 does not apply to trunk. Rebase required? Wrong Branch? See https://wiki.apache.org/hadoop/HowToContribute for help. {color} | \\ \\ || Subsystem || Report/Notes || | JIRA Issue | HDFS-8674 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12840925/HDFS-8674.2.patch | | Console output | https://builds.apache.org/job/PreCommit-HDFS-Build/17702/console | | Powered by | Apache Yetus 0.4.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.
21. Looks like {{HDFS-8674.2.patch}} is for branch-2. Can you post a trunk version?
22. fix conflicts on trunk stemming from under replicated now being low redundancy.
23. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 12s{color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s{color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s{color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 15m 46s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 2m 5s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 23s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 1m 21s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} |

{color:green} mvneclipse {color} | {color:green} 0m 13s{color} | {color:green} trunk passed {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 1m 49s{color} | {color:red} hadoop-hdfs-project/hadoop-hdfs in trunk has 1 extant Findbugs warnings. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 41s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 49s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 49s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 49s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 30s{color} | {color:green} hadoop-hdfs-project/hadoop-hdfs: The patch generated 0 new + 119 unchanged - 2 fixed = 119 total (was 121) {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 54s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 10s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s{color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 56s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 39s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 74m 34s{color} | {color:red} hadoop-hdfs in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 21s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}105m 37s{color} | {color:black} {color} | \\ \\ || Reason || Tests || | Failed junit tests | hadoop.hdfs.server.blockmanagement.TestNodeCount | | | hadoop.hdfs.server.blockmanagement.TestBlocksWithNotEnoughRacks | | | hadoop.hdfs.server.datanode.TestDataNodeUUID | | | hadoop.hdfs.server.datanode.TestDataNodeMultipleRegistrations | | | hadoop.hdfs.server.blockmanagement.TestReconstructStripedBlocksWithRackAwareness | | | hadoop.hdfs.server.blockmanagement.TestRBWBlockInvalidation | | | hadoop.hdfs.server.datanode.checker.TestThrottledAsyncChecker | | | hadoop.hdfs.server.balancer.TestBalancerWithHANameNodes | | | hadoop.hdfs.server.datanode.TestDataNodeMXBean | | | hadoop.hdfs.server.namenode.TestAddStripedBlockInFBR | | | hadoop.hdfs.TestLeaseRecoveryStriped | | | hadoop.hdfs.server.datanode.TestDirectoryScanner | | | hadoop.hdfs.server.namenode.snapshot.TestSnapshot | | | hadoop.hdfs.server.namenode.ha.TestDNFencingWithReplication | | | hadoop.hdfs.server.namenode.ha.TestDNFencing | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:a9ad5d6 | | JIRA Issue | HDFS-8674 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12841132/HDFS-8674.trunk.patch | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux 9ceb14463035 3.13.0-95-generic #142-Ubuntu SMP Fri Aug 12 17:00:09 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/hadoop/patchprocess/precommit/personality/provided.sh | | git revision | trunk / 3fd844b | | Default Java | 1.8.0_111 | | findbugs | v3.0.0 | | findbugs | https://builds.apache.org/job/PreCommit-HDFS-Build/17715/artifact/patchprocess/branch-findbugs-hadoop-hdfs-project_hadoop-hdfs-warnings.html | | unit | https://builds.apache.org/job/PreCommit-HDFS-Build/17715/artifact/patchprocess/patch-unit-hadoop-hdfs-project_hadoop-hdfs.txt | | Test Results | https://builds.apache.org/job/PreCommit-HDFS-Build/17715/testReport/ | | modules | C: hadoop-hdfs-project/hadoop-hdfs U: hadoop-hdfs-project/hadoop-hdfs | | Console output | https://builds.apache.org/job/PreCommit-HDFS-Build/17715/console | | Powered by | Apache Yetus 0.4.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.

24. {{LightWeightLinkedSet}} does not support {{remove()}}.
25. Posted wrong patches. Changed the set from the lightweight back to standard one due to iterator issues.
26. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 10s{color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s{color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s{color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 6m 45s{color} | {color:green} trunk passed {color} | |

{color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 44s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 26s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 51s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 12s{color} | {color:green} trunk passed {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 1m 41s{color} | {color:red} hadoop-hdfs-project/hadoop-hdfs in trunk has 1 extant Findbugs warnings. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 39s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 45s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 43s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 43s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 24s{color} | {color:green} hadoop-hdfs-project/hadoop-hdfs: The patch generated 0 new + 119 unchanged - 2 fixed = 119 total (was 121) {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 48s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 10s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s{color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 45s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 36s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 61m 9s{color} | {color:green} hadoop-hdfs in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 18s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 79m 16s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:a9ad5d6 | | JIRA Issue | HDFS-8674 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12841300/HDFS-8674.trunk.2.patch | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux 06df29fbcd5a 3.13.0-93-generic #140-Ubuntu SMP Mon Jul 18 21:21:05 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/hadoop/patchprocess/precommit/personality/provided.sh | | git revision | trunk / e0fa492 | | Default Java | 1.8.0_111 | | findbugs | v3.0.0 | | findbugs | https://builds.apache.org/job/PreCommit-HDFS-Build/17731/artifact/patchprocess/branch-findbugs-hadoop-hdfs-project_hadoop-hdfs-warnings.html | | Test Results | https://builds.apache.org/job/PreCommit-HDFS-Build/17731/testReport/ | | modules | C: hadoop-hdfs-project/hadoop-hdfs U: hadoop-hdfs-project/hadoop-hdfs | | Console output | https://builds.apache.org/job/PreCommit-HDFS-Build/17731/console | | Powered by | Apache Yetus 0.4.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.

27. +1 The patch looks good.
28. Committed to trunk, branch-2 and branch-2.8. Thanks for working on this, Daryn.
29. SUCCESS: Integrated in Jenkins build Hadoop-trunk-Commit #10924 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/10924/]) HDFS-8674. Improve performance of postponed block scans. Contributed by (kihwal: rev 96c574927a600d15fab919df1fdc9e07887af6c5) * (edit) hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/server/blockmanagement/BlockManager.java
30. Attaching backport patch from HDFS-11868. Committed this to branch-2.7. Thank you [~elgoiri].