Item 31
git_comments:

1. 1 trillion
2. * * * An instance of this class is used to generate a stream of * pseudorandom numbers. The class uses a 64-bit seed, which is * modified using a linear congruential formula. * * see https://en.wikipedia.org/wiki/Linear_congruential_generator
3. * * Copy from {@link Random#seedUniquifier()}
4. * * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
5. * * Random64 is a pseudorandom algorithm(LCG). Therefore, we will get same sequence * if seeds are the same. This main will test how many calls nextLong() it will * get the same seed. * * We do not need to save all numbers (that is too large). We could save * once every 100000 calls nextLong(). If it get a same seed, we can * detect this by calling nextLong() 100000 times continuously. *
6. We regard seed as unsigned long, therefore used '>>>' instead of '>>'.
7. Use Random64 to avoid issue described in HBASE-21256.
8. * * Set this configuration if you want to scale up the size of test data quickly. * <p> * $ ./bin/hbase org.apache.hadoop.hbase.test.IntegrationTestBigLinkedList * -Dgenerator.big.family.value.size=1024 generator 1 10 output

git_commits:

1. **summary:** HBASE-21256 Improve IntegrationTestBigLinkedList for testing huge data
   message: HBASE-21256 Improve IntegrationTestBigLinkedList for testing huge data Backport to branch-1 Signed-off-by: Duo Zhang <zhangduo@apache.org>

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
2. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay

creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

3. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
   label: code-design
4. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
5. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
6. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
   label: code-design
7. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
8. **summary:** Improve IntegrationTestBigLinkedList for testing huge data

description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

9. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
   description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
   label: code-design
10. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
    label: test
11. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
12. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
13. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random.

!ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

14. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
15. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
16. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
17. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
18. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
label: code-design
19. **summary:** Improve IntegrationTestBigLinkedList for testing huge data

description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

20. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
21. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
22. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
23. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
label: test
24. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString

variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

25. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
26. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
27. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
28. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
label: code-design
29. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
30. **summary:** Improve IntegrationTestBigLinkedList for testing huge data

description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

31. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
    label: test
32. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
33. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
34. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
35. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
    description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString

variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

36. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
37. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
38. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
39. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
40. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
41. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root

cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master
label: code-design

42. **summary:** Improve IntegrationTestBigLinkedList for testing huge data
description: Recently, I use ITBLL to test some features in our company. I have encountered the following problems: 1. Generator is too slow at the generating stage, the root cause is SecureRandom. There is a global lock in SecureRandom(See the following picture). I use Random instead of SecureRandom, and it could speed up this stage(500% up with 20 mapper). SecureRandom was brought by HBASE-13382, but speaking of generating random bytes, in my opnion, it is the same with Random. !ITBLL-1.png! 2. VerifyReducer have a cpu cost of 14% on format method. This is cause by create keyString variable. However, keyString may never be used if test result is correct.(and that's in most cases). Just delay creating keyString can yield huge performance boost in verifing stage. !ITBLL-2.png! 3.Arguments check is needed, because there's constraint between arguments. If we broken this constraint, we can not get a correct circular list. 4.Let big family value size could be configured. 5.Avoid starting RS at backup master

jira_issues_comments:

1. Thanks for working on this. On SecureRandom, what about the original argument that Random is more likely to have a collision? A collision could cause the test to fail? #2 sounds like a nice saving. The other items sound great too. Thanks
2. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 13s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 7m 39s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 32s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 19s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 11s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 19s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 56s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 27s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 27s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 15s{color} | {color:red} hbase-it: The patch generated 1 new + 40 unchanged - 0 fixed = 41 total (was 40) {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} shadedjars {color} | {color:green} 4m 15s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 11m 31s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 11s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 51s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 13s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 36m 9s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12941820/HBASE-21256-v1.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 85e3734b95ca 3.13.0-139-generic #188-Ubuntu SMP Tue Jan 9 14:43:09 UTC 2018 x86_64 GNU/Linux |

| Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / fdbaa4c3f0 | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14541/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14541/testReport/ | | Max. process+thread count | 389 (vs. ulimit of 10000) | | modules | C: hbase-it U: hbase-it | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14541/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

3. **body:** In HBASE-13382 we want to avoid collision on a large data set so we use SecureRandom instead of Random...
   label: code-design
4. Zephyr: {code} 450 if (multipleUnevenColumnFamilies) { 451 int n = context.getConfiguration().getInt(BIG_FAMILY_VALUE_SIZE_KEY, 256); {code} Since you added precondition checks in your patch, can you add one for the above against the value of hbase.client.keyvalue.maxsize ? thanks
5. {quote}On SecureRandom, what about the original argument that Random is more likely to have a collision? {quote} Thanks for review. [~stack] [~Apache9] [~yuzhihong@gmail.com] I don't agree with that. This kind of collision ONLY depends on how long the key(128-bits) is. It is 48-bits seed algorithm (according to HBASE-13382 said) that Random utilizes and therefore it CAN generated 32-bits numbers in a completely random way. Consequently, a total of (2^32)^4 possibilities can be generated by calling nextInt() 4 times, which is equivalent to 2^128.
6. **body:** {quote} It is 48-bits seed algorithm (according to HBASE-13382 said) that Random utilizes and therefore it CAN generated 32-bits numbers in a completely random way. {quote} Not really. In HBASE-13161 it shows that SecureRandom can fix the problem. Anyway, the SecureRandom is not necessary as we do not need 'secure' here. See this page https://en.wikipedia.org/wiki/Linear_congruential_generator There is a table which shows different parameter for different LCGs, I think we could make use of the one introduced by Knuth. It uses a 64 bits seed, so we can use it to generate our 16 bytes key - just use two longs, and we can make sure that a 10B test will not have collisions, as we need to generate 2^64 times to make it repetitive. Thanks.
   label: code-design
7. Update patch v2: against hbase.client.keyvalue.maxsize
8. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 25s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 8m 34s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 36s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 20s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 18s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 20s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 6s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 28s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 28s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 16s{color} | {color:red} hbase-it: The patch generated 1 new + 40 unchanged - 0 fixed = 41 total (was 40) {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} shadedjars {color} | {color:green} 4m 11s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 10m 42s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 11s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 49s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 13s{color} | {color:green} The

patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 36m 51s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12941826/HBASE-21256-v2.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 008a50a2d285 3.13.0-153-generic #203-Ubuntu SMP Thu Jun 14 08:52:28 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / fdbaa4c3f0 | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14544/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14544/testReport/ | | Max. process+thread count | 388 (vs. ulimit of 10000) | | modules | C: hbase-it U: hbase-it | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14544/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

9. **body:** [~Apache9] After saw that page, I seem understand what you mean. If the first 32-bits is same, then all 128-bits will be same.Could we solve it in other ways? What about the following solution? {code:java} random bytes nanoTime map taskID iteration |----48bits----|--64bits--|----8bits---|---8bits---|{code} It seems that we don't need to care about collision.
   label: code-design
10. **body:** Just use the 64 bits LCG? It is very simple next = 6364136223846793005L * current + 1442695040888963407L
    label: test
11. Hi, [~Apache9] I attached the patch-v3, and use 64-bit LCG. Please review, thanks.
12. | (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:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 41s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 29s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 16s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 17s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 11s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 3s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 29s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 29s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 15s{color} | {color:red} hbase-it: The patch generated 2 new + 40 unchanged - 0 fixed = 42 total (was 40) {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} shadedjars {color} | {color:green} 4m 14s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 10m 54s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 10s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 43s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 8s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 33m 19s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12941853/HBASE-21256-v3.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 69345d48d882 3.13.0-139-generic #188-Ubuntu SMP Tue Jan 9 14:43:09 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-

Build/component/dev-support/hbase-personality.sh | | git revision | master / fdbaa4c3f0 | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14549/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14549/testReport/ | | Max. process+thread count | 393 (vs. ulimit of 10000) | | modules | C: hbase-it U: hbase-it | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14549/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

13. | (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:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 7m 58s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 34s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 20s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 19s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 19s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 3s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 27s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 27s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 14s{color} | {color:red} hbase-it: The patch generated 2 new + 40 unchanged - 0 fixed = 42 total (was 40) {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} shadedjars {color} | {color:green} 4m 14s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 11m 22s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 11s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 43s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 12s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 36m 24s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12941855/HBASE-21256-v3.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux cf327b19c862 3.13.0-139-generic #188-Ubuntu SMP Tue Jan 9 14:43:09 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / fdbaa4c3f0 | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14550/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14550/testReport/ | | Max. process+thread count | 393 (vs. ulimit of 10000) | | modules | C: hbase-it U: hbase-it | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14550/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.
14. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 13s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 8m

36s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 38s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 23s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 49s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 21s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 46s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 33s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 33s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 17s{color} | {color:red} hbase-it: The patch generated 2 new + 40 unchanged - 0 fixed = 42 total (was 40) {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} shadedjars {color} | {color:green} 4m 43s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 11m 55s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 11s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 46s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 13s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 39m 46s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12941856/HBASE-21256-v3.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux cf8bb24f6fd6 3.13.0-143-generic #192-Ubuntu SMP Tue Feb 27 10:45:36 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / fdbaa4c3f0 | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14551/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14551/testReport/ | | Max. process+thread count | 390 (vs. ulimit of 10000) | | modules | C: hbase-it U: hbase-it | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14551/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

15. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 9s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 6s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 28s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 15s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 12s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 10s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 53s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 27s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 27s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 15s{color} | {color:red} hbase-it: The patch generated 1 new + 40 unchanged - 0 fixed = 41 total (was 40) {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} shadedjars {color} | {color:green} 4m 12s{color} | {color:green} patch has no errors when building our shaded downstream artifacts.

{color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 10m 46s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 11s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 44s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 9s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 32m 16s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12941858/HBASE-21256-v4.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux d6294e3420e4 3.13.0-139-generic #188-Ubuntu SMP Tue Jan 9 14:43:09 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / fdbaa4c3f0 | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14552/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14552/testReport/ | | Max. process+thread count | 391 (vs. ulimit of 10000) | | modules | C: hbase-it U: hbase-it | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14552/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.
16. Mind uploading the patch to review board?
17. I uploaded the patch to review board (see issue links). [~Apache9] thanks.
18. **body:** Trying to follow along, I don't see listing of likelihood of collision on the wiki page? I left some notes on rb.
    label: code-design
19. Hey [~stack] I've relied on review board, that the likelihood of collision is related to the length of the period. The period for the java Random class is 2^48, which means that when you start from 48 bits integer, after 2^48 times you will get this integer again. And since we use 32 bits integer, which is the lowest or highest 32 bits for the 48 bits integer, it will be more likely to collision. And the period of the new Random64 is 2^64, and we will use the 64 bits long to generate keys, the likelihood of collision will be very very small.
20. [~Apache9] thanks. Stick this in an?
21. Overall LGTM. The only concerns is about the AtomicLong in Random64.
22. [~stack] I saw your notes on rb. I ran 50B times nextLong for Random64, and it looks good(I have already uploaded test code on rb). I will try to test 100B tonight.
23. **body:** [~Apache9] If Random64 is not thread-safe, I could be hard to write a test case with multi threads. Single thread to generate 100B numbers is impossible... Random is also thread-safe.
    label: test
24. [~gzh1992n] Please upload the newest patch here to try a pre commit? Thanks.
25. [~Apache9] thanks. I had uploaded the newest patch.
26. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 13s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 26s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 10s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 57s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 39s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 14s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 36s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 29s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 15s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:red}-1{color} | {color:red} mvninstall {color} | {color:red} 0m 59s{color} | {color:red} root in the patch failed. {color} | | {color:red}-1{color} | {color:red} compile {color} | {color:red} 0m 22s{color} | {color:red} hbase-it in the patch failed. {color} | | {color:red}-1{color} | {color:red}

javac {color} | {color:red} 0m 22s{color} | {color:red} hbase-it in the patch failed. {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 23s{color} | {color:red} hbase-common: The patch generated 4 new + 0 unchanged - 0 fixed = 4 total (was 0) {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 15s{color} | {color:red} hbase-it: The patch generated 1 new + 40 unchanged - 0 fixed = 41 total (was 40) {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s{color} | {color:green} The patch has no whitespace issues. {color} | | {color:red}-1{color} | {color:red} shadedjars {color} | {color:red} 1m 2s{color} | {color:red} patch has 11 errors when building our shaded downstream artifacts. {color} | | {color:red}-1{color} | {color:red} hadoopcheck {color} | {color:red} 0m 47s{color} | {color:red} The patch causes 11 errors with Hadoop v2.7.4. {color} | | {color:red}-1{color} | {color:red} hadoopcheck {color} | {color:red} 1m 39s{color} | {color:red} The patch causes 11 errors with Hadoop v3.0.0. {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 0m 44s{color} | {color:red} hbase-common generated 1 new + 0 unchanged - 0 fixed = 1 total (was 0) {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 27s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:red}-1{color} | {color:red} unit {color} | {color:red} 0m 18s{color} | {color:red} hbase-common in the patch failed. {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 0m 22s{color} | {color:red} hbase-it in the patch failed. {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} 20m 56s{color} | {color:black} {color} | \\ \\ || Reason || Tests || | FindBugs | module:hbase-common | | | Boxing/unboxing to parse a primitive org.apache.hadoop.hbase.util.Random64.main(String[]) At Random64.java:org.apache.hadoop.hbase.util.Random64.main(String[]) At Random64.java:[line 100] | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12942959/HBASE-21256-v10.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 1f0f8b5c428e 3.13.0-143-generic #192-Ubuntu SMP Tue Feb 27 10:45:36 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / c9213f752e | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | findbugs | v3.1.0-RC3 | | mvninstall | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-mvninstall-root.txt | | compile | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-compile-hbase-it.txt | | javac | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-compile-hbase-it.txt | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/diff-checkstyle-hbase-common.txt | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | shadedjars | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-shadedjars.txt | | hadoopcheck | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-javac-2.7.4.txt | | hadoopcheck | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-javac-3.0.0.txt | | findbugs | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/new-findbugs-hbase-common.html | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-unit-hbase-common.txt | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/artifact/patchprocess/patch-unit-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/testReport/ | | Max. process+thread count | 87 (vs. ulimit of 10000) | | modules | C: hbase-common hbase-it U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14602/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

27. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 29s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 28s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 46s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 41s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 34s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 37s{color} |

{color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 28s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 14s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 12s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 0s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 25s{color} | {color:red} hbase-common: The patch generated 4 new + 0 unchanged - 0 fixed = 4 total (was 0) {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 16s{color} | {color:red} hbase-it: The patch generated 1 new + 40 unchanged - 0 fixed = 41 total (was 40) {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} shadedjars {color} | {color:green} 4m 27s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 11m 49s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 0m 51s{color} | {color:red} hbase-common generated 1 new + 0 unchanged - 0 fixed = 1 total (was 0) {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 33s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 39s{color} | {color:green} hbase-common in the patch passed. {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 55s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 16s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 43m 21s{color} | {color:black} {color} | \\ \\ || Reason || Tests || | FindBugs | module:hbase-common | | | Boxing/unboxing to parse a primitive org.apache.hadoop.hbase.util.Random64.main(String[]) At Random64.java:org.apache.hadoop.hbase.util.Random64.main(String[]) At Random64.java:[line 102] | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12942979/HBASE-21256-v11.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux a50c1debc43a 3.13.0-153-generic #203-Ubuntu SMP Thu Jun 14 08:52:28 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / c9213f752e | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | findbugs | v3.1.0-RC3 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14606/artifact/patchprocess/diff-checkstyle-hbase-common.txt | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14606/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | findbugs | https://builds.apache.org/job/PreCommit-HBASE-Build/14606/artifact/patchprocess/new-findbugs-hbase-common.html | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14606/testReport/ | | Max. process+thread count | 391 (vs. ulimit of 10000) | | modules | C: hbase-common hbase-it U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14606/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

28. **body:** Please fix the checkstyle and findbugs issues?
    label: code-design
29. | (/) *{color:green}+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:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 27s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 16s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 8s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 43s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 41s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 44s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 32s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || |

{color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 14s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 16s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 2s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 2s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 42s{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} shadedjars {color} | {color:green} 4m 26s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 10m 54s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 0m 45s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 28s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 26s{color} | {color:green} hbase-common in the patch passed. {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 42s{color} | {color:green} hbase-it 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} 41m 32s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12943017/HBASE-21256-v12.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 336534619a1f 3.13.0-143-generic #192-Ubuntu SMP Tue Feb 27 10:45:36 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / c9213f752e | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | findbugs | v3.1.0-RC3 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14611/testReport/ | | Max. process+thread count | 391 (vs. ulimit of 10000) | | modules | C: hbase-common hbase-it U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14611/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

30. +1 on the latest patch. What do you think [~stack]?
31. **body:** I did a test for this Random64 that ran to a trillion, and no collision. It looks good.
    label: test
32. Will commit tomorrow if no objections.
33. Hello, [~Apache9] Could you resolve this issue today? Thanks.
34. Sorry, will commit soon. Thanks for your patient.
35. I was trying this out with a branch-1 backport and noticed this was necessary when running on a cluster. Needed on branch-2 and up too? In runRandomInputGenerator: {code} diff --git a/hbase-it/src/test/java/org/apache/hadoop/hbase/test/IntegrationTestBigLinkedList.java b/hbase-it/src/test/java/org/apache/hadoop/hbase/test/IntegrationTestBigLinkedLis t.java index 1b6ded1ed0..181dfcab7d 100644 --- a/hbase-it/src/test/java/org/apache/hadoop/hbase/test/IntegrationTestBigLinkedList.java +++ b/hbase-it/src/test/java/org/apache/hadoop/hbase/test/IntegrationTestBigLinkedList.java @@ -749,6 +786,7 @@ public class IntegrationTestBigLinkedList extends IntegrationTestBase { FileOutputFormat.setOutputPath(job, tmpOutput); job.setOutputFormatClass(SequenceFileOutputFormat.class); + TableMapReduceUtil.addDependencyJarsForClasses(job.getConfiguration(), Random64.class); boolean success = jobCompletion(job); {code} because now the generator needs hbase-common jar
36. Thanks [~apurtell], I've added the line of code when pushing to master and branch-2. And the patch does not apply cleanly to branch-1, could you please upload your patch here? Thanks.
37. Results for branch branch-2 [build #1378 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-2/1378/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (/) {color:green}+1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2/1378//General_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2/1378//JDK8_Nightly_Build_Report_(Hadoop2)/] (x) {color:red}-1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2/1378//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details. (/) {color:green}+1 client integration test{color}
38. Results for branch master [build #541 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/master/541/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (/) {color:green}+1 general checks{color} -- For more information [see general

report|https://builds.apache.org/job/HBase%20Nightly/job/master/541//General_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/master/541//JDK8_Nightly_Build_Report_(Hadoop2)/] (x) {color:red}-1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/master/541//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details. (/) {color:green}+1 client integration test{color}

39. Attached branch-1 port
40. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 19s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 2s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 2 new or modified test files. {color} | || || || || {color:brown} branch-1 Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 26s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 1m 42s{color} | {color:green} branch-1 passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 32s{color} | {color:green} branch-1 passed with JDK v1.8.0_181 {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 37s{color} | {color:green} branch-1 passed with JDK v1.7.0_191 {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 48s{color} | {color:green} branch-1 passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 2m 38s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 24s{color} | {color:green} branch-1 passed with JDK v1.8.0_181 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 29s{color} | {color:green} branch-1 passed with JDK v1.7.0_191 {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 14s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 1m 35s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 29s{color} | {color:green} the patch passed with JDK v1.8.0_181 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 29s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 37s{color} | {color:green} the patch passed with JDK v1.7.0_191 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 37s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 24s{color} | {color:red} hbase-common: The patch generated 1 new + 0 unchanged - 0 fixed = 1 total (was 0) {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 24s{color} | {color:red} hbase-it: The patch generated 1 new + 35 unchanged - 1 fixed = 36 total (was 36) {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} shadedjars {color} | {color:green} 2m 36s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 1m 36s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 23s{color} | {color:green} the patch passed with JDK v1.8.0_181 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 29s{color} | {color:green} the patch passed with JDK v1.7.0_191 {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 5s{color} | {color:green} hbase-common in the patch passed. {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 0m 23s{color} | {color:green} hbase-it in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 16s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black} {color} | {color:black} {color} | {color:black} 20m 6s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:61288f8 | | JIRA Issue | HBASE-21256 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12943686/HBASE-21256-branch-1.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux c3e01a757950 3.13.0-143-generic #192-Ubuntu SMP Tue Feb 27 10:45:36 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | branch-1 / b73aab8 | | maven | version: Apache Maven 3.0.5 | | Default Java | 1.7.0_191 | | Multi-JDK versions | /usr/lib/jvm/java-8-openjdk-amd64:1.8.0_181 /usr/lib/jvm/java-7-openjdk-amd64:1.7.0_191 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14672/artifact/patchprocess/diff-checkstyle-

hbase-common.txt | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/14672/artifact/patchprocess/diff-checkstyle-hbase-it.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/14672/testReport/ | | Max. process+thread count | 148 (vs. ulimit of 10000) | | modules | C: hbase-common hbase-it U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/14672/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

41. Pushed to branch-1 after fixing the checkstyle issues. Thanks [~apurtell] for the backporting. Thanks [~gzh1992n] for contributing.
42. Results for branch branch-1 [build #509 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/509/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (x) {color:red}-1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/509//General_Nightly_Build_Report/] (x) {color:red}-1 jdk7 checks{color} -- For more information [see jdk7 report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/509//JDK7_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/509//JDK8_Nightly_Build_Report_(Hadoop2)/] (x) {color:red}-1 source release artifact{color} -- See build output for details.
43. Results for branch branch-1.4 [build #509 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/509/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (x) {color:red}-1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/509//General_Nightly_Build_Report/] (x) {color:red}-1 jdk7 checks{color} -- For more information [see jdk7 report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/509//JDK7_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/509//JDK8_Nightly_Build_Report_(Hadoop2)/] (/) {color:green}+1 source release artifact{color} -- See build output for details.
44. This is a nice patch. Look forward to trying it out.