Item 11
git_comments:

1. * * Computes the number of bytes that a <code>KeyValue</code> instance with the provided * characteristics would take up in its underlying data structure for the key. * * @param rlength row length * @param flength family length * @param qlength qualifier length * * @return the key data structure length
2. * * Write KeyValue format into a byte array. * * @param row row key * @param roffset row offset * @param rlength row length * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * @param timestamp version timestamp * @param type key type * @param value column value * @param voffset value offset * @param vlength value length * @return The newly created byte array.
3. * * Checks the parameters passed to a constructor. * * @param row row key * @param rlength row length * @param family family name * @param flength family length * @param qualifier column qualifier * @param qlength qualifier length * @param value column value * @param vlength value length * * @throws IllegalArgumentException an illegal value was passed
4. Write key, value and key row length.
5. * * Checks if column matches. * * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * * @return True if column matches
6. * * Computes the number of bytes that a <code>KeyValue</code> instance with the provided * characteristics would take up for its underlying data structure. * * @param rlength row length * @param flength family length * @param qlength qualifier length * @param vlength value length * * @return the <code>KeyValue</code> data structure length
7. * * Constructs KeyValue structure filled with specified values. Uses the provided buffer as the * data buffer. * <p> * Column is split into two fields, family and qualifier. * * @param buffer the bytes buffer to use * @param boffset buffer offset * @param row row key * @param roffset row offset * @param rlength row length * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * @param timestamp version timestamp * @param type key type * @param value column value * @param voffset value offset * @param vlength value length * @throws IllegalArgumentException an illegal value was passed or there is insufficient space * remaining in the buffer
8. * * Returns the value wrapped in a new <code>ByteBuffer</code>. * * @return the value
9. * * Create a KeyValue for the specified row, family and qualifier that would be * smaller than all other possible KeyValues that have the same row, * family, qualifier. * Used for seeking. * * @param buffer the buffer to use for the new <code>KeyValue</code> object * @param boffset buffer offset * @param row the value key * @param roffset row offset * @param rlength row length * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * * @return First possible key on passed Row, Family, Qualifier. * * @throws IllegalArgumentException The resulting <code>KeyValue</code> object would be larger * than the provided buffer or than <code>Integer.MAX_VALUE</code>
10. * * Loads this object's value into the provided <code>ByteBuffer</code>. * <p> * Does not clear or flip the buffer. * * @param dst the buffer where to write the value * * @throws BufferOverflowException if there is insufficient space remaining in the buffer
11. * * Create a KeyValue for the specified row, family and qualifier that would be * smaller than all other possible KeyValues that have the same row, * family, qualifier. * Used for seeking. * * @param buffer the buffer to use for the new <code>KeyValue</code> object * @param row the value key * @param family family name * @param qualifier column qualifier * * @return First possible key on passed Row, Family, Qualifier. * * @throws IllegalArgumentException The resulting <code>KeyValue</code> object would be larger * than the provided buffer or than <code>Integer.MAX_VALUE</code>
12. * * Write KeyValue format into the provided byte array. * * @param buffer the bytes buffer to use * @param boffset buffer offset * @param row row key * @param roffset row offset * @param rlength row length * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * @param timestamp version timestamp * @param type key type * @param value column value * @param

voffset value offset * @param vlength value length * * @return The number of useful bytes in the buffer. * * @throws IllegalArgumentException an illegal value was passed or there is insufficient space * remaining in the buffer
13. * * Returns the value wrapped in a new <code>ByteBuffer</code>. * * @param foffset family offset * @param flength family length * @param qoffset qualifier offset * @param qlength qualifier length * * @return the latest version of the column, or <code>null</code> if none found
14. * * Checks if the specified column contains a non-empty value (not a zero-length byte array). * * @param family family name * @param qualifier column qualifier * * @return whether or not a latest value exists and is not empty
15. * * Checks if the specified column contains an empty value (a zero-length byte array). * * @param family family name * @param qualifier column qualifier * * @return whether or not a latest value exists and is empty
16. pad to the smallest multiple of the pad width
17. side effect possibly.
18. never will exact match
19. doesn't exist
20. * * Loads the latest version of the specified column into the provided <code>ByteBuffer</code>. * <p> * Does not clear or flip the buffer. * * @param family family name * @param qualifier column qualifier * @param dst the buffer where to write the value * * @return <code>true</code> if a value was found, <code>false</code> otherwise * * @throws BufferOverflowException there is insufficient space remaining in the buffer
21. * * Checks for existence of a value for the specified column (empty or not). * * @param family family name * @param qualifier column qualifier * /** * Checks for existence of a value for the specified column (empty or not). * * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * * @return true if at least one value exists in the result, false if not
22. pos === (-(insertion point) - 1)
23. * * The KeyValue for the most recent timestamp for a given column. * * @return the KeyValue for the column, or null if no value exists in the row or none have been * selected in the query (Get/Scan) /** * The KeyValue for the most recent timestamp for a given column. * * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * * @return the KeyValue for the column, or null if no value exists in the row or none have been * selected in the query (Get/Scan)
24. * * Checks if the specified column contains an empty value (a zero-length byte array). * * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * * @return whether or not a latest value exists and is empty
25. * * Loads the latest version of the specified column into the provided <code>ByteBuffer</code>. * <p> * Does not clear or flip the buffer. * * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * @param dst the buffer where to write the value * * @return <code>true</code> if a value was found, <code>false</code> otherwise * * @throws BufferOverflowException there is insufficient space remaining in the buffer
26. * * Checks if the specified column contains a non-empty value (not a zero-length byte array). * * @param family family name * @param foffset family offset * @param flength family length * @param qualifier column qualifier * @param qoffset qualifier offset * @param qlength qualifier length * * @return whether or not a latest value exists and is not empty
27. never use directly
28. pos is now insertion point
29. * * Calls non-functional test methods. * * @param args
30. warm up
31. * * Microbenchmark that compares {@link Result#getValue} and {@link Result#loadValue} performance. * * @throws Exception

git_commits:

1. **summary:** HBASE-5625 Avoid byte buffer allocations when reading a value from a Result object (Tudor Scurtu)

message: HBASE-5625 Avoid byte buffer allocations when reading a value from a Result object (Tudor Scurtu) git-svn-id: https://svn.apache.org/repos/asf/hbase/trunk@1333159 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
   label: code-design
2. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
3. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
4. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
5. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
6. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.

label: code-design

7. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
   label: code-design
8. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
   label: code-design
9. **summary:** Avoid byte buffer allocations when reading a value from a Result object
   description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
10. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
    label: code-design
11. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
12. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
13. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
14. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a

separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
label: code-design

15. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
16. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
17. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
18. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
19. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
20. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
label: code-design
21. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
label: code-design
22. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a

separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.

23. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
24. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
25. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
26. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
27. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
28. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
    label: code-design
29. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
30. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.

31. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
32. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
33. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
34. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
35. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
36. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
    label: code-design
37. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
38. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
    label: code-design
39. **summary:** Avoid byte buffer allocations when reading a value from a Result object

description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.

40. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
41. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
42. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
43. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
44. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
45. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
46. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
47. **summary:** Avoid byte buffer allocations when reading a value from a Result object
description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value

destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.

48. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
49. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
50. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
51. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.
    label: code-design
52. **summary:** Avoid byte buffer allocations when reading a value from a Result object
    description: When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. The current functionality is maintained, and we have added a separate method call stack that employs the described changes. I will provide more details with the patch. Running tests with a profiler, the reduction of read time seems to be of up to 40%.

jira_issues_comments:

1. **body:** The current patch introduces a method (Result.loadValue()) that uses pre-allocated buffers. Momentarily the new methods duplicate code from the original ones so as not to impact the current architecture.
   label: code-design
2. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12519598/5625.txt against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. -1 patch. The patch command could not apply the patch. Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1274//console This message is automatically generated.
3. @Tudor: Can you submit a patch for trunk? Thanks
4. Correct trunk diff.
5. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12519634/5625v2.txt against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what

manual steps were performed to verify this patch. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 3 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: org.apache.hadoop.hbase.mapreduce.TestImportTsv org.apache.hadoop.hbase.mapred.TestTableMapReduce org.apache.hadoop.hbase.mapreduce.TestHFileOutputFormat Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/1275//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/1275//artifact/trunk/patchprocess/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1275//console This message is automatically generated.

6. **body:** I don't see where loadValue() is called. Minor comments: {code} + long longkeylength = KEY_INFRASTRUCTURE_SIZE + rlength + flength + qlength; {code} we normally use variable names such as longKeyLength where the second and subsequent words have leading capital letter. {code} + if(flength != 0) { ... + if(qlength != 0) { {code} Please insert space between if and (. {code} + KeyValue searchTerm = getSearchTerm(kvs[0].getRow(), family, foffset, flength, qualifier, qoffset, qlength); {code} Recent vote is to limit ling length within 100 characters. {code} + if (lLength > Integer.MAX_VALUE) + throw new IllegalArgumentException("KeyValue length " + lLength + " > " + Integer.MAX_VALUE); {code} Please enclose the throw statement in curly braces. {code} + * The KeyValue for the most recent for a given column. If the column does + * not exist in the result set - if it wasn't selected in the query (Get/Scan) + * or just does not exist in the row the return value is null. {code} I think the above can be phrased as: {code} + * The most recent KeyValue for a given column. If the column does + * not exist in the result set - if it wasn't selected in the query (Get/Scan) + * or just does not exist in the row - the return value is null. {code}
   label: code-design
7. **body:** How do I know the provided buffer big enough? {code} + public KeyValue(byte [] buffer, {code} What can we do to ensure that the checks in createByteArray and writeByteArray, your new method, are shared rather than duplicated? Put all of the below on one line or add curlies: {code} + if (!Bytes.equals(family, foffset, flength, this.bytes, o, fl)) + return false; {code} Ditto for other similar formattings Do we need both containsNonEmptyColumn and containsEmptyColumn? Should loadValue be in KeyValue? I like createFirstOnRow as method name instead of getSearchTerm. The former describes what is happening? We should use your new binarySearch instead of what we have now? Thanks for looking into this stuff. What made you start digging here? You have big Result objects with lots of kvs?
   label: code-design
8. **body:** @Zhihong: I initially submitted a minimalistic version so I wouldn't have to make extensive modifications after reviews. Now the method is called from a unit test class. The variable names were copied from existing methods and I wanted them to have a uniform naming scheme. I renamed the variables, but not the method parameters. The same for the method comment. Implemented code style comments. @stack: Implemented code style comments. Added check for buffer size. Created 'KeyValue.checkParameters()' method. Should 'createEmptyByteArray()' call it as well? 'containsNonEmptyColumn()' checks if the value exists & is not empty; 'containsEmptyColumn()' checks if the value exists & is empty. If you would have only one, for the other case you would have to actually read the value and check it. Moved most 'loadValue()' functionality to 'KeyValue'. This raises a problem: how do we elegantly treat the case when the buffer (provided from 'Result') isn't big enough? Refactored 'Result.getSearchTerm()' as another 'KeyValue.createFirstOnRow()'. The new 'binarySearch()' method avoids allocating a byte array. We run incremental jobs that update values; we also have to read different values form the same row in different places.
   label: code-design
9. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12520125/5625v3.txt against trunk revision . +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 2 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: org.apache.hadoop.hbase.client.TestFromClientSide org.apache.hadoop.hbase.mapreduce.TestImportTsv org.apache.hadoop.hbase.mapred.TestTableMapReduce

org.apache.hadoop.hbase.mapreduce.TestHFileOutputFormat Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/1322//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/1322//artifact/trunk/patchprocess/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1322//console This message is automatically generated.

10. **body:** @Tudor: Would it make sense to put your changes in testBasic() and testMultiVersion() into their own loop so that you can show us the performance improvement ? The call to flip() should be documented in non-test code.
    label: code-design
11. @Zhihong: I added a new method, 'TestResult.testPerformance()'. This must be of course removed if the patch is accepted. It prints the results, so it doesn't require an integrated profiler. It takes a few minutes to run... Mentioned that the buffer isn't cleared or flipped in the 'KeyValue.loadValue()' method comment as well. Is that what you meant?
12. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12520387/5625v4.txt against trunk revision . +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 1 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: org.apache.hadoop.hbase.mapreduce.TestImportTsv org.apache.hadoop.hbase.mapred.TestTableMapReduce org.apache.hadoop.hbase.mapreduce.TestHFileOutputFormat Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/1338//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/1338//artifact/trunk/patchprocess/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1338//console This message is automatically generated.
13. Microbenchmarking is a bit trickier than this. Perhaps we should mention some guideline it in the HBase developer documentation. There's a lot of stuff going on in the HotSpot (e.g. dynamic compilation) that needs to be taken into account. Here are some resources: http://www.slideshare.net/drorbr/so-you-want-to-write-your-own-benchmark-presentation https://wikis.oracle.com/display/HotSpotInternals/MicroBenchmarks We might be able to use something like junit-benchmarks. http://labs.carrotsearch.com/junit-benchmarks.html Cosmin
14. **body:** Here is output from TestResult#testPerformance: {code} loadValue(): 122281 getValue(): 141426 {code} Improvement was about 13.5%
    label: code-design
15. I have put some comments here: https://reviews.apache.org/r/4559/ @Tudor: Feel free to create your own review request for the new patch.
16. --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/ --- Review request for hbase. Summary ------- When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. This addresses bug HBASE-5625. https://issues.apache.org/jira/browse/HBASE-5625 Diffs ----- src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 Diff: https://reviews.apache.org/r/4607/diff Testing ------- Added value check to TestResult#testBasic and TestResult.testMultiVersion. Thanks, Tudor
17. @Zhihong: Thanks for the review request. I actually had to make my own in order to upload the diff: https://reviews.apache.org/r/4607/ The performance actually depends on the system capabilities. It's hard to write a microbenchmark test for an issue that manifests itself on large I/O intensive jobs that put a lot of gc pressure. I implemented a few of Cosmin's suggestions.
18. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12520978/5625v5.txt against trunk revision . +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 3 new or

modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 1 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: org.apache.hadoop.hbase.mapreduce.TestMultithreadedTableMapper org.apache.hadoop.hbase.mapreduce.TestImportTsv org.apache.hadoop.hbase.mapred.TestTableMapReduce org.apache.hadoop.hbase.mapreduce.TestHFileOutputFormat org.apache.hadoop.hbase.mapreduce.TestTableMapReduce Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/1369//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/1369//artifact/trunk/patchprocess/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1369//console This message is automatically generated.

19. --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6622 --- I ran TestTableMapReduce and TestMultithreadedTableMapper with patch v5. They passed. Some minor comments below. src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14303> Please include vlength in the exception message src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14304> Should read 'BufferOverflowException if there' src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14305> Add a space between comma and fl. src/main/java/org/apache/hadoop/hbase/client/Result.java <https://reviews.apache.org/r/4607/#comment14306> Is this comment needed ? src/main/java/org/apache/hadoop/hbase/client/Result.java <https://reviews.apache.org/r/4607/#comment14307> This line can be removed. src/test/java/org/apache/hadoop/hbase/client/TestResult.java <https://reviews.apache.org/r/4607/#comment14308> white space. src/test/java/org/apache/hadoop/hbase/client/TestResult.java <https://reviews.apache.org/r/4607/#comment14309> Since benchmarking is hard to do, this test case can be dropped. - Ted On 2012-04-02 14:22:48, Tudor Scurtu wrote: bq. bq. --- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. --- bq. bq. (Updated 2012-04-02 14:22:48) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.
20. **body:** --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6623 --- Looks good. Some comments below. src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14311> How do we know this buffer is big enough? Maybe should add an override that takes an offset into the buffer? src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14312> I like this refactoring. src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14313> How do I know the buffer is big enough? src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14315> Why not create a ByteBuffer? or called ByteBuffer wrap? Why not call it toByteBuffer? src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment14318> Is this an override? src/main/java/org/apache/hadoop/hbase/KeyValue.java

<https://reviews.apache.org/r/4607/#comment14317> Why pull out all these values here? Maybe we'll fail the first test (q1 is not used by the first test and these extractions cost.. they create byte arrays...) src/main/java/org/apache/hadoop/hbase/KeyValue.java
<https://reviews.apache.org/r/4607/#comment14319> How do I know the buffer is big enough? Should there be an override that takes an offset? src/main/java/org/apache/hadoop/hbase/KeyValue.java
<https://reviews.apache.org/r/4607/#comment14320> Please follow the formatting that is present in the rest of the file. Notice placement of exceptions and params. Do not add your own style. Thanks. src/main/java/org/apache/hadoop/hbase/KeyValue.java
<https://reviews.apache.org/r/4607/#comment14321> This is a critical base class. I'm nervous when it gets refactored. You have tests for each of your changes? And all the old KV tests pass? src/main/java/org/apache/hadoop/hbase/client/Result.java
<https://reviews.apache.org/r/4607/#comment14322> This should be lazily instantiated src/main/java/org/apache/hadoop/hbase/client/Result.java
<https://reviews.apache.org/r/4607/#comment14323> Please follow convention that the rest of the file has. src/main/java/org/apache/hadoop/hbase/client/Result.java
<https://reviews.apache.org/r/4607/#comment14324> ditto src/main/java/org/apache/hadoop/hbase/client/Result.java
<https://reviews.apache.org/r/4607/#comment14325> This comment should be on the @return javadoc rather than here. src/main/java/org/apache/hadoop/hbase/client/Result.java
<https://reviews.apache.org/r/4607/#comment14326> Rename hasContent or isNotEmpty or isNotEmptyColumn src/main/java/org/apache/hadoop/hbase/client/Result.java
<https://reviews.apache.org/r/4607/#comment14327> Why we have both isNonEmptyColumn and isEmptyColumn? Why not just one and then check return with a !? src/main/java/org/apache/hadoop/hbase/client/Result.java
<https://reviews.apache.org/r/4607/#comment14328> Rename hasColumn or isColumn. src/test/java/org/apache/hadoop/hbase/client/TestResult.java
<https://reviews.apache.org/r/4607/#comment14310> Or leave it and change the name of the test so it doesn't have the test prefix. Add a main and have it call this. I think the method useful. Might as well keep it. src/test/java/org/apache/hadoop/hbase/client/TestResult.java
<https://reviews.apache.org/r/4607/#comment14329> I think there needs to be tests for the new KV functionality because KV is a fundamental type and we can't mess it up. - Michael On 2012-04-02 14:22:48, Tudor Scurtu wrote: bq. bq. --- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. --- bq. bq. (Updated 2012-04-02 14:22:48) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.
label: code-design

21. **body:** Implemented coding style comments. Added separate unit tests for all modified or added methods.
label: code-design
22. bq. On 2012-04-02 16:50:48, Ted Yu wrote: bq. > I ran TestTableMapReduce and TestMultithreadedTableMapper with patch v5. bq. > They passed. bq. > bq. > Some minor comments below. Thanks! Fixed all except Result#259 and TestResult#117. bq. On 2012-04-02 16:50:48, Ted Yu wrote: bq. > src/test/java/org/apache/hadoop/hbase/client/TestResult.java, line 117 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97956#file97956line117> bq. > bq. > Since benchmarking is hard to do, this test case can be dropped. Refactored as 'doReadBenchmark()'; called from 'main()'. bq. On 2012-04-02 16:50:48, Ted Yu wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 259 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line259> bq. > bq. > Is this comment needed ? This was copied from original. Should I remove both occurences? - Tudor --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6622 --- On 2012-04-02 14:22:48, Tudor Scurtu wrote: bq. bq. --- bq.

This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. --- bq. bq. (Updated 2012-04-02 14:22:48) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.

23. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > Looks good. Some comments below. Thanks! Please read the changes below. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 531 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line531> bq. > bq. > How do we know this buffer is big enough? Maybe should add an override that takes an offset into the buffer? Added overload. Added exception comment for when there is insufficient space remaining in the buffer. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 1443 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line1443> bq. > bq. > Is this an override? Yes. Modified the original method to call the new one with default parameters. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 1448 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line1448> bq. > bq. > Why pull out all these values here? Maybe we'll fail the first test (q1 is not used by the first test and these extractions cost.. they create byte arrays...) Moved. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 2001 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line2001> bq. > bq. > How do I know the buffer is big enough? Should there be an override that takes an offset? Added buffer offset parameter. We know exactly how many bytes we have to write and how much free space we have in the buffer. An 'IllegalArgumentException' is thrown when there isn't enough free space. Is that what you were asking? bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 2002 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line2002> bq. > bq. > Please follow the formatting that is present in the rest of the file. Notice placement of exceptions and params. Do not add your own style. Thanks. Hope I fixed this. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 251 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line251> bq. > bq. > Please follow convention that the rest of the file has. Hope I fixed this. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 255 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line255> bq. > bq. > ditto Hope I fixed this. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 478 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line478> bq. > bq. > Rename hasColumn or isColumn. This overloads an original method. A refactoring here would be a major non-backwards compatible change. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 398 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line398> bq. > bq. > Rename hasContent or isNotEmpty or isNotEmptyColumn This method was named to mirror 'containsColumn()'. See below. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 431 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line431> bq. > bq. > Why we have both isNonEmptyColumn and isEmptyColumn? Why not just one and then check return with a !? They are not complementary. containsColumn = value exists containsEmptyColumn = value exists & is empty byte array containsNonEmptyColumn = value exists & is not empty byte array The value could be missing, in which case all methods would return false. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/test/java/org/apache/hadoop/hbase/client/TestResult.java, line 117 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97956#file97956line117> bq. > bq. > Or leave it and change the name of the test so it doesn't have the test prefix. Add a main and have it call this. I think the method useful. Might as well keep it. Refactored as 'doReadBenchmark()'; called from 'main()'; bq. On

2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 2056 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line2056> bq. > bq. > This is a critical base class. I'm nervous when it gets refactored. You have tests for each of your changes? And all the old KV tests pass? Added checks in 'TestKeyValue.testFirstLastOnRow()' for the new 'KeyValue.createFirstOnRow()' methods. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 616 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line616> bq. > bq. > How do I know the buffer is big enough? Added exception comment for when there is insufficient space remaining in the buffer. Is that what you meant? bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 297 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line297> bq. > bq. > This comment should be on the @return javadoc rather than here. This was copied from original. Should I remove both occurences? bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/test/java/org/apache/hadoop/hbase/client/TestResult.java, line 178 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97956#file97956line178> bq. > bq. > I think there needs to be tests for the new KV functionality because KV is a fundamental type and we can't mess it up. Moved 'loadValue()' checks to their own test methods. Also took the liberty to move 'getColumn()' checks in the same manner in order to keep consistency. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 1138 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line1138> bq. > bq. > Why not create a ByteBuffer? or called ByteBuffer wrap? Why not call it toByteBuffer? You need to be able to pass your own buffer when you have to compose multiple values. I added a new method that uses 'ByteBuffer.wrap()'. I feel that this method should contain the word 'value' in its name so as not to create the impression that it is writing the entire underlying 'KeyValue' structure to the buffer, so I called it 'getValueAsByteBuffer()'. Please comment if it is inadequate. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 87 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line87> bq. > bq. > This should be lazily instantiated Fixed. - Tudor --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6623 --- On 2012-04-02 14:22:48, Tudor Scurtu wrote: bq. bq. --- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. --- bq. bq. (Updated 2012-04-02 14:22:48) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.
24. --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/ --- (Updated 2012-04-04 17:08:03.581526) Review request for hbase. Changes ------- Implemented coding style comments. Added separate unit tests for all modified or added methods. Summary ------- When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. This addresses bug HBASE-5625. https://issues.apache.org/jira/browse/HBASE-5625 Diffs (updated) ----- src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef src/test/java/org/apache/hadoop/hbase/TestKeyValue.java fae6902 src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 Diff: https://reviews.apache.org/r/4607/diff Testing ------- Added value check to TestResult#testBasic and TestResult.testMultiVersion. Thanks, Tudor
25. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12521342/5625v6.txt against trunk revision . +1 @author.

The patch does not contain any @author tags. +1 tests included. The patch appears to include 6 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 3 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed unit tests in . Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/1386//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/1386//artifact/trunk/patchprocess/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1386//console This message is automatically generated.

26. @Zhihing, Stack: What do you think about the new changes? Tudor
27. @Stack: Can you give Tudor some advice ?
28. **body:** bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 531 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line531> bq. > bq. > How do we know this buffer is big enough? Maybe should add an override that takes an offset into the buffer? bq. bq. Tudor Scurtu wrote: bq. Added overload. Added exception comment for when there is insufficient space remaining in the buffer. So, the way this works, we just allocate N and hope that stuff fits inside N? If it doesn't we throw an exception? There is no correlation between data that comes across and the N allocation? bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 616 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line616> bq. > bq. > How do I know the buffer is big enough? bq. bq. Tudor Scurtu wrote: bq. Added exception comment for when there is insufficient space remaining in the buffer. Is that what you meant? I am not understanding how the allocation works. It seems arbitrary unrelated to the actual result size that comes over from the server. Is that so? If so, it seems unfriendly throwing an exception when allocated size and what is returned from the server do not match. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 1138 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line1138> bq. > bq. > Why not create a ByteBuffer? or called ByteBuffer wrap? Why not call it toByteBuffer? bq. bq. Tudor Scurtu wrote: bq. You need to be able to pass your own buffer when you have to compose multiple values. bq. I added a new method that uses 'ByteBuffer.wrap()'. I feel that this method should contain the word 'value' in its name so as not to create the impression that it is writing the entire underlying 'KeyValue' structure to the buffer, so I called it 'getValueAsByteBuffer()'. Please comment if it is inadequate. Sounds good Tudor. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 297 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line297> bq. > bq. > This comment should be on the @return javadoc rather than here. bq. bq. Tudor Scurtu wrote: bq. This was copied from original. Should I remove both occurences? Sorry. I did not notice it was problem on original. If you can fix it, that'd be sweet. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 431 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line431> bq. > bq. > Why we have both isNonEmptyColumn and isEmptyColumn? Why not just one and then check return with a !? bq. bq. Tudor Scurtu wrote: bq. They are not complementary. bq. containsColumn = value exists bq. containsEmptyColumn = value exists & is empty byte array bq. containsNonEmptyColumn = value exists & is not empty byte array bq. The value could be missing, in which case all methods would return false. OK. If not clear from comments, please add your notes above. Will help those that come after. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 478 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line478> bq. > bq. > Rename hasColumn or isColumn. bq. bq. Tudor Scurtu wrote: bq. This overloads an original method. A refactoring here would be a major non-backwards compatible change. Ok. Thanks for pointing this out. - Michael ----------- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6623 ----------- On 2012-04-04 17:08:03, Tudor Scurtu wrote: bq. bq. ----------- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. ----------- bq. bq. (Updated 2012-04-04 17:08:03) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that

will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/TestKeyValue.java fae6902 bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.
label: code-design

29. Added check with reallocation in 'Result.binarySearch()'.
30. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 531 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line531> bq. > bq. > How do we know this buffer is big enough? Maybe should add an override that takes an offset into the buffer? bq. bq. Tudor Scurtu wrote: bq. Added overload. Added exception comment for when there is insufficient space remaining in the buffer. bq. bq. Michael Stack wrote: bq. So, the way this works, we just allocate N and hope that stuff fits inside N? If it doesn't we throw an exception? There is no correlation between data that comes across and the N allocation? Please see below. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 616 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line616> bq. > bq. > How do I know the buffer is big enough? bq. bq. Tudor Scurtu wrote: bq. Added exception comment for when there is insufficient space remaining in the buffer. Is that what you meant? bq. bq. Michael Stack wrote: bq. I am not understanding how the allocation works. It seems arbitrary unrelated to the actual result size that comes over from the server. Is that so? If so, it seems unfriendly throwing an exception when allocated size and what is returned from the server do not match. Added check with reallocation in 'Result.binarySearch()'. For this I had to add two methods in 'KeyValue' that calculate the number of bytes that are taken up in a 'KeyValue' object's underlying buffer ('getKeyValueDataStructureSize()' and 'getKeyDataStructureSize()'). Is this ok, and if so, how about replacing all manual calculations of these values in the project with calls to the new methods? bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 297 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line297> bq. > bq. > This comment should be on the @return javadoc rather than here. bq. bq. Tudor Scurtu wrote: bq. This was copied from original. Should I remove both occurences? bq. bq. Michael Stack wrote: bq. Sorry. I did not notice it was problem on original. If you can fix it, that'd be sweet. Fixed. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 431 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97955#file97955line431> bq. > bq. > Why we have both isNonEmptyColumn and isEmptyColumn? Why not just one and then check return with a !? bq. bq. Tudor Scurtu wrote: bq. They are not complementary. bq. containsColumn = value exists bq. containsEmptyColumn = value exists & is empty byte array bq. containsNonEmptyColumn = value exists & is not empty byte array bq. The value could be missing, in which case all methods would return false. bq. bq. Michael Stack wrote: bq. OK. If not clear from comments, please add your notes above. Will help those that come after. Added. - Tudor --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6623 --- On 2012-04-04 17:08:03, Tudor Scurtu wrote: bq. bq. --- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. --- bq. bq. (Updated 2012-04-04 17:08:03) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 243d76f bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/TestKeyValue.java fae6902 bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.

31. --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/ --- (Updated 2012-04-25 16:01:29.035293) Review request for hbase. Changes ------- Added check with reallocation in 'Result.binarySearch()'. Summary ------- When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. This addresses bug HBASE-5625. https://issues.apache.org/jira/browse/HBASE-5625 Diffs (updated) ----- src/main/java/org/apache/hadoop/hbase/KeyValue.java 9ae9e02 src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef src/test/java/org/apache/hadoop/hbase/TestKeyValue.java 786d2df src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 Diff: https://reviews.apache.org/r/4607/diff Testing ------- Added value check to TestResult#testBasic and TestResult.testMultiVersion. Thanks, Tudor
32. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12524295/5625v7.txt against trunk revision . +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 6 new or modified tests. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 7 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: org.apache.hadoop.hbase.util.TestHBaseFsck Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/1646//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/1646//artifact/trunk/patchprocess/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1646//console This message is automatically generated.
33. @Ted, the 13% performance gain is from 100000000 iterations. The big uncertainty is the unknown size to pre-allocate. Is it possible not to copy of value? For example, return an immutable wrap to the original value?
34. @Jimmy I tried this approach, but the performance is comparable to that of the original API. Nevertheless, I could include this functionality if requested.
35. @Tudor, thanks for trying it out. There is no need if the performance is comparable.
36. **body:** bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 616 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line616> bq. > bq. > How do I know the buffer is big enough? bq. bq. Tudor Scurtu wrote: bq. Added exception comment for when there is insufficient space remaining in the buffer. Is that what you meant? bq. bq. Michael Stack wrote: bq. I am not understanding how the allocation works. It seems arbitrary unrelated to the actual result size that comes over from the server. Is that so? If so, it seems unfriendly throwing an exception when allocated size and what is returned from the server do not match. bq. bq. Tudor Scurtu wrote: bq. Added check with reallocation in 'Result.binarySearch()'. For this I had to add two methods in 'KeyValue' that calculate the number of bytes that are taken up in a 'KeyValue' object's underlying buffer ('getKeyValueDataStructureSize()' and 'getKeyDataStructureSize()'). Is this ok, and if so, how about replacing all manual calculations of these values in the project with calls to the new methods? I think I am beginning to understand what you are at (pardon me, I am a little slow). You want to speed up finding KVs in big Results and part of the way in which you do this is reuse of a buffer you keep private in Result. The buffer will not match a specific KV.... usually it'll be too big and if it is too small, you'll allocate a buffer big enough, a new one. What locations would you put getKeyValueDataStructureSize into place? For example? - Michael --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6623 --- On 2012-04-25 16:01:29, Tudor Scurtu wrote: bq. bq. --- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. --- bq. bq. (Updated 2012-04-25 16:01:29) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq.

src/main/java/org/apache/hadoop/hbase/KeyValue.java 9ae9e02 bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/TestKeyValue.java 786d2df bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.
label: code-design

37. --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review7365 --- Ship it! +1 on patch. I have a few comments below. Small potatoes. src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment16250> So you have plans to use these elsewhere in the codebase? If so, its ok that they are public. src/main/java/org/apache/hadoop/hbase/KeyValue.java <https://reviews.apache.org/r/4607/#comment16251> This could be 'as its backing data buffer'? If so, I can address that on commit src/main/java/org/apache/hadoop/hbase/client/Result.java <https://reviews.apache.org/r/4607/#comment16252> Here you are trying to make a smart guess on a buffer size that will serve for multiple invocations on binarySearch? Your hope is that you'll not have to reallocate the buffer the next time you come through here because the buffer should have enough space in it to hold the next random KV that comes through here? - Michael On 2012-04-25 16:01:29, Tudor Scurtu wrote: bq. bq. --- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. --- bq. bq. (Updated 2012-04-25 16:01:29) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 9ae9e02 bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/TestKeyValue.java 786d2df bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.
38. **body:** This is a pretty big change to what appears only a small overall performance gain and only on the client. I am a bit skeptical. +-0.
label: code-design
39. w.r.t. https://reviews.apache.org/r/4607/#comment16252 : For size of 129 (2^n + 1), the actual space allocated this way would be 256. I think it is bigger than what is needed. Tudor made the suggestion of padding to the closest multiple of 128 bytes (configurable).
40. @Lars: The patch leaves the original implementation relatively intact, as for the performance gain YMMV.
41. Modified 'Result' private buffer reallocation to pad to a size equal to the smallest multiple of a configurable constant.
42. bq. On 2012-04-28 23:39:41, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 181 bq. > <https://reviews.apache.org/r/4607/diff/3/?file=104237#file104237line181> bq. > bq. > So you have plans to use these elsewhere in the codebase? If so, its ok that they are public. Replaced manual calculations of infrastructure sizes with calls to the new methods in 'KeyValue'. bq. On 2012-04-28 23:39:41, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 542 bq. > <https://reviews.apache.org/r/4607/diff/3/?file=104237#file104237line542> bq. > bq. > This could be 'as its backing data buffer'? If so, I can address that on commit Fixed. bq. On 2012-04-28 23:39:41, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/client/Result.java, line 256 bq. > <https://reviews.apache.org/r/4607/diff/3/?file=104238#file104238line256> bq. > bq. > Here you are trying to make a smart guess on a buffer size that will serve for multiple invocations on binarySearch? Your hope is that you'll not have to reallocate the buffer the next time you come through here because the buffer should have enough space in it to hold the next random KV that comes through here? Exactly. The power of 2 implementation could have grown too rapidly, so I modified it to pad to a size equal to the smallest multiple of a configurable constant. - Tudor --- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review7365 -

---------- On 2012-04-25 16:01:29, Tudor Scurtu wrote: bq. bq. ---------- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. ---------- bq. bq. (Updated 2012-04-25 16:01:29) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 9ae9e02 bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/TestKeyValue.java 786d2df bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.

43. bq. On 2012-04-02 17:34:38, Michael Stack wrote: bq. > src/main/java/org/apache/hadoop/hbase/KeyValue.java, line 616 bq. > <https://reviews.apache.org/r/4607/diff/1/?file=97954#file97954line616> bq. > bq. > How do I know the buffer is big enough? bq. bq. Tudor Scurtu wrote: bq. Added exception comment for when there is insufficient space remaining in the buffer. Is that what you meant? bq. bq. Michael Stack wrote: bq. I am not understanding how the allocation works. It seems arbitrary unrelated to the actual result size that comes over from the server. Is that so? If so, it seems unfriendly throwing an exception when allocated size and what is returned from the server do not match. bq. bq. Tudor Scurtu wrote: bq. Added check with reallocation in 'Result.binarySearch()'. For this I had to add two methods in 'KeyValue' that calculate the number of bytes that are taken up in a 'KeyValue' object's underlying buffer ('getKeyValueDataStructureSize()' and 'getKeyDataStructureSize()'). Is this ok, and if so, how about replacing all manual calculations of these values in the project with calls to the new methods? bq. bq. Michael Stack wrote: bq. I think I am beginning to understand what you are at (pardon me, I am a little slow). You want to speed up finding KVs in big Results and part of the way in which you do this is reuse of a buffer you keep private in Result. The buffer will not match a specific KV.... usually it'll be too big and if it is too small, you'll allocate a buffer big enough, a new one. bq. bq. What locations would you put getKeyValueDataStructureSize into place? For example? Replaced manual calculations of infrastructure sizes with calls to the new methods in 'KeyValue'. - Tudor ---------- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/#review6623 ---------- On 2012-04-25 16:01:29, Tudor Scurtu wrote: bq. bq. ---------- bq. This is an automatically generated e-mail. To reply, visit: bq. https://reviews.apache.org/r/4607/ bq. ---------- bq. bq. (Updated 2012-04-25 16:01:29) bq. bq. bq. Review request for hbase. bq. bq. bq. Summary bq. ------- bq. bq. When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. bq. bq. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. bq. bq. bq. This addresses bug HBASE-5625. bq. https://issues.apache.org/jira/browse/HBASE-5625 bq. bq. bq. Diffs bq. ----- bq. bq. src/main/java/org/apache/hadoop/hbase/KeyValue.java 9ae9e02 bq. src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef bq. src/test/java/org/apache/hadoop/hbase/TestKeyValue.java 786d2df bq. src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 bq. bq. Diff: https://reviews.apache.org/r/4607/diff bq. bq. bq. Testing bq. ------- bq. bq. Added value check to TestResult#testBasic and TestResult.testMultiVersion. bq. bq. bq. Thanks, bq. bq. Tudor bq. bq.

44. ---------- This is an automatically generated e-mail. To reply, visit: https://reviews.apache.org/r/4607/ ---------- (Updated 2012-05-02 08:36:45.758830) Review request for hbase. Changes ------- Modified 'Result' private buffer reallocation to pad to a size equal to the smallest multiple of a configurable constant. Summary ------- When calling Result.getValue(), an extra dummy KeyValue and its associated underlying byte array are allocated, as well as a persistent buffer that will contain the returned value. These can be avoided by reusing a static array for the dummy object and by passing a ByteBuffer object as a value destination buffer to the read method. This addresses bug HBASE-5625. https://issues.apache.org/jira/browse/HBASE-5625 Diffs (updated) ----- src/main/java/org/apache/hadoop/hbase/KeyValue.java 9ae9e02

src/main/java/org/apache/hadoop/hbase/client/Result.java df0b3ef src/test/java/org/apache/hadoop/hbase/TestKeyValue.java 786d2df src/test/java/org/apache/hadoop/hbase/client/TestResult.java f9e29c2 Diff: https://reviews.apache.org/r/4607/diff Testing ------- Added value check to TestResult#testBasic and TestResult.testMultiVersion. Thanks, Tudor

45. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12525265/5625v8.txt against trunk revision . +1 @author. The patch does not contain any @author tags. +1 tests included. The patch appears to include 6 new or modified tests. +1 hadoop23. The patch compiles against the hadoop 0.23.x profile. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 3 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed unit tests in . Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/1718//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/1718//artifact/trunk/patchprocess/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/1718//console This message is automatically generated.
46. +1 on patch v8.
47. Will integrate patch v8 in 3 hours if there is no objection.
48. Patch v8 integrated to trunk. Thanks for the patch, Tudor. Thanks for the review, Stack.
49. Thank you too. Tudor
50. Integrated in HBase-TRUNK-security #190 (See [https://builds.apache.org/job/HBase-TRUNK-security/190/]) HBASE-5625 Avoid byte buffer allocations when reading a value from a Result object (Tudor Scurtu) (Revision 1333159) Result = SUCCESS tedyu : Files : * /hbase/trunk/src/main/java/org/apache/hadoop/hbase/KeyValue.java * /hbase/trunk/src/main/java/org/apache/hadoop/hbase/client/Result.java * /hbase/trunk/src/test/java/org/apache/hadoop/hbase/TestKeyValue.java * /hbase/trunk/src/test/java/org/apache/hadoop/hbase/client/TestResult.java
51. Marking closed.