Item 88
git_comments:

1. keep increasing length until we find first row of next partition or we reach the very last batch
2. we are only interested in the first batch's records
3. true when at least one window function needs to process all batches of a partition before passing any batch downstream
4. this is the last batch of current partition if partition ends before the end of the batch it's the last available batch next batch contains a different partition
5. we didn't compute the whole partition length in the previous partition, we need to update the length now
6. true if we don't know yet the full length of this partition size of this partition (if partial is true, then this is a partial length of the partition) remaining non-processed rows in this partition remaining non-processed peers in current frame
7. we need at least 2 batches even when window functions only need one batch, so we can detect the end of the current partition
8. isPeer also checks if it's the same partition
9. true if window definition contains an order-by clause
10. * * @return true when all window functions are ready to process the current batch (it's the first batch currently * held in memory)
11. Clear the memory for the incoming batch
12. * * compares two rows from different batches (can be the same), if they have the same value for the partition by * expression * @param b1Index index of first row * @param b1 batch for first row * @param b2Index index of second row * @param b2 batch for second row * @return true if the rows are in the same partition
13. * * compares two rows from different batches (can be the same), if they have the same value for the order by * expression * @param b1Index index of first row * @param b1 batch for first row * @param b2Index index of second row * @param b2 batch for second row * @return true if the rows are in the same partition
14. CUME_DIST, PERCENT_RANK and NTILE require the length of current partition before processing it's first batch
15. * * @param numBatchesAvailable number of batches available for current partition * @param hasOrderBy window definition contains an ORDER BY clause * @param frameEndReached we found the last row of the first batch's frame * @param partitionEndReached all batches of current partition are available * * @return true if this window function can process the first batch immediately
16. for CUME_DIST, PERCENT_RANK and NTILE we need the full partition otherwise we can process the first batch immediately
17. * * @param hasOrderBy window definition contains an ORDER BY clause * @return true if this window function requires all batches of current partition to be available before processing * the first batch

git_commits:

1. **summary:** DRILL-3952: Improve Window Functions performance when not all batches are required to process the current batch
   message: DRILL-3952: Improve Window Functions performance when not all batches are required to process the current batch this closes #222
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
   label: code-design
2. **title:** Drill 3952

body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?

3. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
4. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
5. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
6. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
7. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
   label: code-design
8. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
   label: code-design
9. **title:** Drill 3952
   body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
10. **title:** Drill 3952
    body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
11. **title:** Drill 3952
    body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
    label: code-design
12. **title:** Drill 3952
    body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?
13. **title:** Drill 3952
    body: Improve Window Functions performance when not all batches are required to process the current batch This patch also improves how the window functions operator handles limits by avoiding processing anymore batches after the kill signal has been received @amansinha100 can you please review ?

github_pulls_comments:

1. **body:** Overall, there are 2 categories of the window functions: one that require all batches of the partition before they can start processing and second that can start work with partial partition as long as we have all peer rows of the current row. It seems to me that some refactoring is needed to have derived classes dedicated to the 2 categories, otherwise the logic in the common code is getting complicated. What do you think ? This need not necessarily be done as part of this patch...but we should discuss.
   label: code-design

2. Window operator has become increasingly complex. I already opened [DRILL-3662] (https://issues.apache.org/jira/browse/DRILL-3662) to refactor the code.
3. +1. Per previous comments, there is room for improvement in terms of code organization and reducing the number of passes which I understand will be covered by DRILL-3662.
4. thanks @amansinha100 , I updated DRILL-3662 description with the new refactoring tasks

github_pulls_reviews:

1. This should be a long since the number of peers could easily be greater than Integer.MAX_VALUE.
2. **body:** I think isPeer() could also check whether two rows are in the same partition...because the notion of peer is only valid within a partition. This will simplify the generated code a bit.
   label: code-design
3. **body:** The performance could be pretty bad due to the sequential scan through the remaining rows. The data is sorted right ?
   label: code-design
4. why not. We still need to have a separate `isSamePartition` though
5. will fix
6. **body:** This may be a relic from early implementations. I should be able to remove it now. I may also be able to get rid of `updatePartitionSize()` but this may require too much work for this patch and may do it in a separate patch
   label: code-design
7. change to 'rows in the same partition and are peers'

jira_issues:

1. **summary:** Improve Window Functions performance when not all batches are required to process the current batch
   description: Currently, the window operator blocks until all batches of current partition to be available. For some queries it's necessary (e.g. aggregate with no order-by in the window definition), but for other cases the window operator can process and pass the current batch downstream sooner. Implementing this should help the window operator use less memory and run faster, especially in the presence of a limit operator. The purpose of this JIRA is to improve the window operator in the following cases: - aggregate, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - lead can process current batch as soon as it receives 1 more batch - lag can process current batch immediately - first_value can process current batch immediately - last_value, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - row_number, rank and dense_rank can process current batch immediately
2. **summary:** Improve Window Functions performance when not all batches are required to process the current batch
   description: Currently, the window operator blocks until all batches of current partition to be available. For some queries it's necessary (e.g. aggregate with no order-by in the window definition), but for other cases the window operator can process and pass the current batch downstream sooner. Implementing this should help the window operator use less memory and run faster, especially in the presence of a limit operator. The purpose of this JIRA is to improve the window operator in the following cases: - aggregate, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - lead can process current batch as soon as it receives 1 more batch - lag can process current batch immediately - first_value can process current batch immediately - last_value, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - row_number, rank and dense_rank can process current batch immediately
3. **summary:** Improve Window Functions performance when not all batches are required to process the current batch
   description: Currently, the window operator blocks until all batches of current partition to be available. For some queries it's necessary (e.g. aggregate with no order-by in the window definition), but for other cases the window operator can process and pass the current batch downstream sooner. Implementing this should help the window operator use less memory and run faster, especially in the presence of a limit operator. The purpose of this JIRA is to improve the window operator in the following cases: - aggregate, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - lead can process current batch as soon as it receives 1 more batch - lag can process current batch immediately - first_value can process current batch immediately - last_value, when order-by

clause is available in window definition, can process current batch as soon as it receives the last peer row - row_number, rank and dense_rank can process current batch immediately

4. **summary:** Improve Window Functions performance when not all batches are required to process the current batch
   description: Currently, the window operator blocks until all batches of current partition to be available. For some queries it's necessary (e.g. aggregate with no order-by in the window definition), but for other cases the window operator can process and pass the current batch downstream sooner. Implementing this should help the window operator use less memory and run faster, especially in the presence of a limit operator. The purpose of this JIRA is to improve the window operator in the following cases: - aggregate, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - lead can process current batch as soon as it receives 1 more batch - lag can process current batch immediately - first_value can process current batch immediately - last_value, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - row_number, rank and dense_rank can process current batch immediately
5. **summary:** Improve Window Functions performance when not all batches are required to process the current batch
   description: Currently, the window operator blocks until all batches of current partition to be available. For some queries it's necessary (e.g. aggregate with no order-by in the window definition), but for other cases the window operator can process and pass the current batch downstream sooner. Implementing this should help the window operator use less memory and run faster, especially in the presence of a limit operator. The purpose of this JIRA is to improve the window operator in the following cases: - aggregate, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - lead can process current batch as soon as it receives 1 more batch - lag can process current batch immediately - first_value can process current batch immediately - last_value, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - row_number, rank and dense_rank can process current batch immediately
6. **summary:** Improve Window Functions performance when not all batches are required to process the current batch
   description: Currently, the window operator blocks until all batches of current partition to be available. For some queries it's necessary (e.g. aggregate with no order-by in the window definition), but for other cases the window operator can process and pass the current batch downstream sooner. Implementing this should help the window operator use less memory and run faster, especially in the presence of a limit operator. The purpose of this JIRA is to improve the window operator in the following cases: - aggregate, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - lead can process current batch as soon as it receives 1 more batch - lag can process current batch immediately - first_value can process current batch immediately - last_value, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - row_number, rank and dense_rank can process current batch immediately
   label: code-design
7. **summary:** Improve Window Functions performance when not all batches are required to process the current batch
   description: Currently, the window operator blocks until all batches of current partition to be available. For some queries it's necessary (e.g. aggregate with no order-by in the window definition), but for other cases the window operator can process and pass the current batch downstream sooner. Implementing this should help the window operator use less memory and run faster, especially in the presence of a limit operator. The purpose of this JIRA is to improve the window operator in the following cases: - aggregate, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - lead can process current batch as soon as it receives 1 more batch - lag can process current batch immediately - first_value can process current batch immediately - last_value, when order-by clause is available in window definition, can process current batch as soon as it receives the last peer row - row_number, rank and dense_rank can process current batch immediately

jira_issues_comments:

1. pull request created: https://github.com/apache/drill/pull/222
2. [~amansinha100] can you please review ? thx
3. pull request updated. thx
4. Merged in ef1cb72
5. Verified with 1.4.0 (gitId: 32b871b). LGTM.