FULL OUTER MapJoin Code Changes

Source files listed in the order they appear in the Review Board.

* HiveConf
  + New variables to control FULL OUTER MapJoin:
    - hive.mapjoin.full.outer
      * Do FULL OUTER MapJoin planning?
    - hive.test.mapjoin.full.outer.override
      * Ignore the hive.mapjoin.full.outer flag during a test run? In other words, even though a Q file might set that variable to true, just ignore it and don’t do FULL OUTER MapJoin query planning. Used to get base line query results.
    - hive.mapjoin.full.outer.hybridgrace
      * Do FULL OUTER MapJoin planning when Hybrid Grace is enabled?
      * I concerned that might be Hybrid Grace related FULL OUTER MapJoin bugs.
* data/files/fullouter\_\*
  + Randomly generated data for Big Table and Small Table
    - Test table data covers the 3 variations for Native Vector MapJoin:
      * Single Long, Single String, and Multi-Key (for all other key combinations).
    - Ensured there are at least 1 NULL key pattern on each side.
    - Designed for testing FULL OUTER MapJoin.
* itests/hive-jmh
  + MapJoin benchmark code needed updates because of changes made to code shared with TestMapJoinOperator.
* CommonJoinOperator
  + I hit NPEs dealing with NULLs.
* ExplainTask.java
  + I started to add a new feature and cleaned up code duplication.
  + Dropped the feature but kept the clean up.
  + Then, instead of the new feature, I added a new DEBUG option to EXPLAIN VECTORIZATION statement. It allows adding new debug display methods (e.g. to MapJoinDesc) for things that only developer geeks like us want to see.
* MapJoinOperator.java
  + Big changes for FULL OUTER MapJoin.
    - Different code paths for the 2 plan modes:
      1. Dynamic Partition Hash Join (DPHJ)
      2. Shared-Memory
  + Inner loop in process method now has 3 modes:
    1. Original for non-FULL OUTER mode.
    2. FULL OUTER hash table lookup with new MatchTracker object.
       - The MatchTracker object lets us determine if the key has been matched for the first time. If so, we forward that row to the FULL OUTER INTERSECT Reducer.
    3. FULL OUTER INTERSECT hash table key lookup only with new MatchTracker object.
       - Here we only want to remember if the key was used as our intersect word. So, we don’t want to fetch the Small Table result yet.
       - Later at closeOp time, we will find all non-matched keys from our intersect work and output the keys and Small Table results at that time. Those results will have NULLs for the Big Table columns.
  + The closeOp method has a “key” role so to speak to output results for any Small Table key that was not matched.
  + Grace Hybrid
    - In theory, each partition can be handled by a different MatchTracker and the non matched Small Keys for it can generate the Small Table results.
    - The MatchTracker object supports partitioning.
      * I.e. An internal array of MatchTracker objects.
      * The partitions need to be managed.
      * Each partition MatchTracker is configured with the proper logical hash table size for its hash map.
* Operator.java
  + We need the ability to forward only some rows to the new FULL OUTER INTERSECT Reducer.
    - Introduced the new concept of an auxiliary child.
    - The forward methods to not go to the auxiliary child. Only explicit calls to forwardAuxiliary do, etc.
  + Separated out vectorization from the forward method.
    - Now we have vectorForward with a properly typed VectorizedRowBatch parameter.
* TableScanOperator.java
  + Change to new vectorForward method.
* BytesBytesMultiHashMap.java
  + Plumb the new MatchTracker object in.
    - On lookup, tell it about key matches.
    - Provide a new method findNextNonMatched that can iterate over the hash map and indicate which keys were not matched.
* HashMapWrapper.java
  + Add dummy methods for new MatchTracker method variations.
* HybridHashTableContainer.java
  + Uses the MatchTracker partitioning feature.
  + The non matched Small Key iterator must walk over the partitions.
* MapJoinBytesTableContainer.java
  + Plumb the new MatchTracker object through to the hash table.
    - New lookup methods using MatchTracker.
    - Create new iterator NonMatchedSmallTableIterator that returns the key and Small Table values for keys that were not matched.
  + Miscellaneous FilterTag NULL issues.
* MapJoinKey.java
  + To produce results for FULL OUTER when there are Small Table keys that are not matched we need to be able to deserialize the Small Table values.
* MapJoinObjectSerDeContext.java
  + Add debug display method stringify.
* MapJoinTableContainer.java
  + Add new MatchTracker methods to interface.
  + And, new NonMatchedSmallTableIterator interface.
* MatchTracker.java
  + Our new object that is basically a glorified bit map for the hash table slot indices.
  + Includes Hybrid Grace support as partitioned mode with an internal array of MatchTracker objects.
* ReusableGetAdaptorDirectAccess.java
  + New MatchTracker parameter.
* UnwrapRowContainer.java
  + Miscellaneous FilterTag NULL issue.
* ReduceRecordProcessor.java
  + Better error message.
* VectorAppMasterEventOperator.java
  + New vectorForward method.
* VectorAssignRow.java
  + Assign from ArrayList<Object>, too.
* VectorCopyRow.java
  + Misc changes.
* VectorDeserializeRow.java
  + Misc changes.
* VectorFilterOperator.java
* VectorGroupByOperator.java
* VectorLimitOperator.java
  + New vectorForward method.
* VectorMapJoinBaseOperator.java
  + Add a new helper object MapJoinBigTableInfo and get method in an attempt to simplify logic in new FullOuterMapJoinOptimization class.
  + Make internalForward handle ArrayList<Object>, too.
  + New vectorForward method.
* VectorMapJoinOperator.java
  + Don’t null out the FIlterMap – it breaks things.
  + New MatchTracker and intersect method overrides.
  + Flush the auxiliary batch on closeOp.
* VectorSMBMapJoinOperator.java
* VectorSelectOperator.java
  + New vectorForward method.
* VectorizationContext.java
* VectorExpression.java
  + Clean up.
* VectorMapJoinCommonOperator.java
  + Needed to rework column information for FULL OUTER MapJoin since we now have non matched Small Table keys and their results to deal with.
  + Added new way to setup the operators to support FULL OUTER operator variations to extend the existing OUTER operators.
    - This included pulling up the process method into the common superclass and adding calls to standard override methods commonSetup, firstBatchSetup, hashTableSetup, and processBatch.
* VectorMapJoinFullOuterIntersect{Long|MultiKey|String}Operator.java
  + Override the VectorMapJoinOuter{Long|MultiKey|String}Operator superclass processBatch method to just lookup keys for the intersect and to process no Small Table values.
  + The superclass closeOp will spill out non matched Small Table keys and their Small Table values.
* VectorMapJoinFullOuter{Long|MultiKey|String}Operator.java
  + Override the hash table setup for the VectorMapJoinOuter{Long|MultiKey|String}Operator superclass with FULL OUTER specific code.
* VectorMapJoinGenerateResultOperator.java
  + Common operator setup changes.
  + More of rework column information for FULL OUTER MapJoin since we now have non matched Small Table keys and their results to deal with.
  + New vectorForward method.
* VectorMapJoinInnerBigOnlyGenerateResultOperator.java
* VectorMapJoinInnerBigOnly{Long|MultiKey|String}Operator.java
* VectorMapJoinInnerGenerateResultOperator.java
* VectorMapJoinInner{Long|MultiKey|String}Operator.java
* VectorMapJoinLeftSemiGenerateResultOperator.java
* VectorMapJoinLeftSemi{Long|MultiKey|String}Operator.java
  + Common operator setup changes.
* VectorMapJoinOuterGenerateResultOperator.java
  + Common operator setup changes.
  + Add new matchTracker member.
    - Null for regular OUTER joins.
    - Set by new VectorMapJoinFullOuter[Intersect]{Long|MultiKey|String}Operator classes.
  + New FULL OUTER support methods including no matching Small Table key result generation.
* VectorMapJoinOuter{Long|MultiKey|String}Operator.java
  + Common operator setup changes.
  + Add matchTracker member as parameter to hash table lookup calls.
  + Cleanup.
  + Forward first-match keys to FULL OUTER INTERSECT Reducer.
* VectorMapJoinFastBytesHashMap.java
  + Add new NonMatchedBytesHashMapIterator class for iterating over non match Small Table keys and returning their Small Table values.
  + Support new methods signatures with MatchTracker parameter.
  + Cleanup.
* VectorMapJoinFastBytesHashMultiSet.java
* VectorMapJoinFastBytesHashSet.java
  + Some rework to support MatchTracker method changes in superclass.
* VectorMapJoinFastBytesHashTable.java
  + Change findReadSlot to return slot index instead of value ref to support MatchTracker work.
* VectorMapJoinFastHashTable.java
  + Methods to create MatchTracker and iterator objects.
* VectorMapJoinFastKeyStore.java
  + New method for getting a key when found by iterating (above) over the slot table with the MatchTracker object to find non matching Small Table keys.
* VectorMapJoinFastLongHashMap.java
  + Add new NonMatchedLongHashMapIteratorclass for iterating over non match Small Table keys and returning their Small Table values.
  + Support new methods signatures with MatchTracker parameter.
* VectorMapJoinFastLongHashMultiSet.java
* VectorMapJoinFastLongHashSet.java
  + Some rework to support MatchTracker method changes in superclass.
* VectorMapJoinFastLongHashTable.java
  + Change findReadSlot to return slot index instead of value ref to support MatchTracker work.
  + Cleanup.
* VectorMapJoinFastMultiKeyHash{Map|MultiSet|Set}.java
  + Misc.
* VectorMapJoinFastNonMatchedIterator.java
  + New non matched Small Key iterator interface for Fast hash table.
* VectorMapJoinFastStringCommon.java
* VectorMapJoinFastStringHashMap.java
* VectorMapJoinFastStringHashMultiSet.java
* VectorMapJoinFastStringHashSet.java
* VectorMapJoinFastTableContainer.java
  + Misc
* VectorMapJoinBytesHashMap.java
* VectorMapJoinHashMap.java
* VectorMapJoinHashTable.java
* VectorMapJoinLongHashMap.java
* VectorMapJoinLongHashTable.java
* VectorMapJoinNonMatchedIterator.java
  + Interface additions/changes.
* VectorMapJoinOptimizedCreateHashTable.java
  + Cleanup.
* VectorMapJoinOptimizedHashMap.java
  + Add new NonMatchedBytesHashMapIterator class for iterating over non match Small Table keys and returning their Small Table values.
  + Support new methods signatures with MatchTracker parameter.
* VectorMapJoinOptimizedHashMultiSet.jav
* VectorMapJoinOptimizedHashSet.java
  + Misc
* VectorMapJoinOptimizedHashTable.java
  + Cleanup
  + Some rework to support MatchTracker.
* VectorMapJoinOptimizedLongCommon.java
  + Cleanup
* VectorMapJoinOptimizedLongHashMap.java
  + Add new NonMatchedLongHashMapIteratorfor iterating over non match Small Table keys and returning their Small Table values.
  + Support new methods signatures with MatchTracker parameter.
* VectorMapJoinOptimizedMultiKeyHashMap.jav
  + Cleanup
* VectorMapJoinOptimizedNonMatchedIterator.java
  + New non matched Small Key iterator interface for Optimized hash table.
* VectorMapJoinOptimizedStringCommon.java
  + Cleanup.
* VectorMapJoinOptimizedStringHashMap.java
  + Cleanup.
  + Write String adapted NonMatchedStringHashMapIterator
  + Support new methods signatures with MatchTracker parameter.
* VectorMapJoinOptimizedStringHashMultiSet.java
* VectorMapJoinOptimizedStringHashSet.java
  + Cleanup
* ConvertJoinMapJoin.java
  + Important class for FULL OUTER MapJoin.
  + If a FULL OUTER MapJoin operator gets created, additional planning is done for FULL OUTER INTERSECT Reducer by calling the FullOuterMapJoinOptimization class.
  + The new MapJoinProcessor.determineEnableFullOuterMapJoin method is called to determine if FULL OUTER MapJoin is enabled/supported.
  + And, we remove vestigial Filter Maps.
* FullOuterMapJoinOptimization.java
  + New FULL OUTER MapJoin planning class.
  + Most of it is for planning the FULL OUTER INTERSECT Reducer.
* MapJoinProcessor.java
  + New method determineEnableFullOuterMapJoin to see if we are willing to plan for FULL OUTER MapJoin.
  + Change getBigTableCandidates to support FULL OUTER MapJoin.
* Vectorizer.java
  + Add debug display method debugDisplayJoinOperatorTree
  + Add MergeJoinWork EXPLAIN VECTORIZATION support to say we don’t support vectorization of Merge Join rather than being silent about it.
  + Add the new Native Vector MapJoin operator classes for FULL OUTER.
  + Support rework column information for FULL OUTER MapJoin since we now have non matched Small Table keys and their results to deal with.
* ExplainConfiguration.java
* ExplainSemanticAnalyzer.java
* HiveLexer.g
* HiveParser.g
* Explain.java
* ExplainWork.java
  + Add new DEBUG option to EXPLAIN
* JoinCondDesc.java
  + Rename the FULL OUTER display name.
* JoinDesc.java
  + NPEs in EXPLAIN description methods get caught – by why throw if we avoid that?
* MapJoinDesc.java
  + Add new isFullOuterIntersect flag
  + Add a few EXPLAIN DEBUG display methods
  + Add DynamicPartitionHashJoin to EXPLAIN display.
  + Tinker with EXPLAIN VECTORIZATION display.
* MergeJoinWork.java
  + More add MergeJoinWork EXPLAIN VECTORIZATION support to say we don’t support vectorization of Merge Join rather than being silent about it.
* OperatorExplainVectorization.java
  + EXPLAIN VECTORIZATION display tinkering.
* ReduceSinkDesc.java
  + Add a few EXPLAIN DEBUG display methods
* VectorMapJoinDesc.java
* VectorMapJoinInfo.java
  + Add FULL OUTER enum.
  + Support rework column information for FULL OUTER MapJoin since we now have non matched Small Table keys and their results to deal with.
* TestBytesBytesMultiHashMap.java
  + Add null matchTracker parameter.
* CollectorTestOperator.java
* RowCollectorTestOperator.java
* RowVectorCollectorTestOperator.java
* RowTestObjectsMultiSet.java
  + Enhance test classes.
* TestTimestampWritableAndColumnVector.jav
  + Cleanup
* TestVectorSelectOperator.java
  + New vectorForward method
* MapJoinTestConfig.java
* MapJoinTestData.java
* MapJoinTestDescription.java
  + Support testing FULL OUTER MapJoin and rework column information for FULL OUTER MapJoin since we now have non matched Small Table keys and their results to deal with.
* NoOpExpression.java
  + Add test class.
* TestMapJoinOperator.java
  + Support testing FULL OUTER MapJoin.
  + Includes setting up a 2nd MapJoin operator for FULL OUTER INTERSECT.
* CheckFastHashTable.java
* TestVectorMapJoinFastBytesHashMapNonMatched.java
  + Add testing for non matching Small Table key iterators.
* TestVectorMapJoinFastBytesHashSet.java
  + Parameter changes.
* TestVectorMapJoinFastLongHashMapNonMatched.java
  + Add testing for non matching Small Table key iterators.
* TestVectorMapJoinFastLongHashSet.java
  + Parameter changes.
* VerifyFastRow.java
  + Cleanup
* VectorBatchGenerator.java
* VectorColumnGroupGenerator.java
  + Add more test data types.
* Q file changes
  + New Q files created that are same except for setting for vectorization:
    - fullouter\_mapjoin\_1\_optimized.q
    - vector\_fullouter\_mapjoin\_1\_fast.q
    - vector\_fullouter\_mapjoin\_1\_optimized.q
    - vector\_fullouter\_mapjoin\_1\_optimized\_passthru.q
  + Added vector\_full\_outer\_join.q as a variation of vector\_left\_outer\_join.q
  + To existing Q files that have FULL OUTER JOIN queries:
    - Set hive.mapjoin.full.outer to true and false
    - And, copied the query
    - Sometimes added EXPLAIN [VECTORIZATION] statements.
* VerifyFast.java
  + Cleanup
* BytesColumnVector.java
  + For some reason I was tripping on a null in the vector and length 0.
* TestTimestampColumnVector.java
  + Cleanup