Item 74
git_comments:

1. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
2. ========================== non-expression reference queries ========================== 0: non-expression timeseries reference, 1 columns
3. 15: string longer
4. the show must go on
5. 17: time floor, non-expr col + expr agg
6. 1: non-expression timeseries reference, 2 columns
7. 16: time floor, non-expr col + reg agg
8. 9: mixed math - 2 longs, 1 double, 1 float
9. * * Benchmark that tests various SQL queries.
10. 4: non-expression timeseries reference, 5 columns
11. 5: group by non-expr with 1 agg
12. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
13. 22: time floor offset by 1 day + non-expr agg (group by)
14. 23: time floor offset by 1 day + expr agg (group by)
15. 21: time floor + expr agg (group by)
16. ========================== expressions ========================== 7: math op - 2 longs
17. 13: unary negate
18. 20: time floor + non-expr agg (group by)
19. expressions
20. 8: mixed math - 2 longs, 1 double
21. non-expression reference
22. 12: cos
23. 25: group by non-expr with expr agg
24. 19: time floor + expr agg (timeseries)
25. 3: non-expression timeseries reference, 4 columns
26. 18: time floor + non-expr agg (timeseries) (non-expression reference)
27. 10: mixed math - 3 longs, 1 double, 1 float
28. 11: all same math op - 3 longs, 1 double, 1 float
29. 14: string long
30. 24: group by long expr with non-expr agg
31. 2: non-expression timeseries reference, 3 columns
32. 6: group by non-expr with 2 agg
33. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
34. * * common machinery for processing two input operators and functions, which should always treat null inputs as null * output, and are backed by a primitive values instead of an object values (and need to use the null vectors instead of * checking the vector themselves for nulls)

35. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
36. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
37. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
38. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
39. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
40. * * specialized {@link UnivariateFunctionVectorProcessor} for processing (double[]) -> double[]
41. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
42. * * specialized {@link BivariateFunctionVectorProcessor} for processing (double[], long[]) -> double[]
43. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
44. * * specialized {@link BivariateFunctionVectorProcessor} for processing (double[], double[]) -> double[]
45. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file

except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.

46. * * specialized {@link BivariateFunctionVectorProcessor} for processing (long[], double[]) -> double[]
47. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
48. * * specialized {@link UnivariateFunctionVectorProcessor} for processing (long[]) -> double[]
49. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
50. * * specialized {@link BivariateFunctionVectorProcessor} for processing (long[], long[]) -> double[]
51. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
52. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
53. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
54. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
55. * * Result of {@link ExprVectorProcessor#evalVector} which wraps the actual evaluated results of the operation over the * input vector(s). Methods to get actual results mirror vectorized value and object selectors. * * The generic parameter T should be the native java array type of the vector result (long[], String[], etc.)
56. non-primitives should implement this
57. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The

ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.

58. * * Interface describing vectorized expression processors, which can be specialized using input type information to * produce optimized expression evaluators, which can operate on batches of primitive data with minimal object overhead
59. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
60. * * specialized {@link UnivariateFunctionVectorProcessor} for processing (long[]) -> long[]
61. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
62. * * specialized {@link BivariateFunctionVectorProcessor} for processing (long[], long[]) -> long[]
63. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
64. * * specialized {@link UnivariateFunctionVectorObjectProcessor} for processing (String[]) -> long[]
65. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
66. * * common machinery for processing single input operators and functions, which are backed by an object value instead of * a primitive value (so do not need to use the null vector, and instead can check the value vector itself for nulls)
67. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
68. * * common machinery for processing single input operators and functions, which should always treat null input as null * output, and are backed by a primitive value instead of an object value (and need to use the null vector instead of * checking the vector itself for nulls)
69. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * *

70. No instantiation
71.
72. No instantiation
73. * * Make a 1 argument math processor with the following type rules * long -> long * double -> double
74. * * Make a 2 argument, math processor with the following type rules * long, long -> long * long, double -> double * double, long -> double * double, double -> double
75. * * Make a 1 argument math processor with the following type rules * long -> double * double -> double
76.
77. Strip leading 0x from hex strings.
78. No instantiation
79.
80. * * randomize inputs to various vector expressions and make sure the results match nonvectorized expressions * * this is not a replacement for correctness tests, but will ensure that vectorized and non-vectorized expression * evaluation is at least self consistent...
81. * * expression has no inputs and can be optimized into a constant selector
82. * * expression must be implicitly mapped across the multiple values per row of known multi-value inputs
83.
84. * * expression has a single input, which may produce single or multi-valued output, but if so, it must be implicitly * mappable (i.e. the expression is not treating its input as an array and not wanting to output an array)
85. * * expression produces explict multi-valued output, or implicit multi-valued output via mapping
86. * * expression has inputs whose type was unresolveable, or was incomplete, such as unknown multi-valuedness
87. * * expression has a single, single valued input, and is dictionary encoded if the value is a string
88. * * expression explicitly using multi-valued inputs as array inputs
89. * * expression is vectorizable
90. vectorized expressions do not currently support unknown inputs, multi-valued inputs or outputs, implicit mapping
91. if analysis, inferred output type, or implicit mapping is in play, output will be multi-valued
92. * * Druid tries to be chill to expressions to make up for not having a well defined table schema across segments. This * method performs some analysis to determine what sort of selectors can be constructed on top of an expression, * whether or not the expression will need implicitly mapped across multi-valued inputs, if the

expression produces * multi-valued outputs, is vectorizable, and everything else interesting when making a selector. * * Results are stored in a {@link ExpressionPlan}, which can be examined to do whatever is necessary to make things * function properly.

93. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
94. These flags allow for selectors that wrap a single underlying column to be optimized, through caching results and via allowing deferred execution in the case of building dimension selectors. SINGLE_INPUT_SCALAR is set if an input is single valued, and the output is definitely single valued, with an additional requirement for strings that the column is dictionary encoded. SINGLE_INPUT_MAPPABLE is set when a single input string column, which can be multi-valued, but if so, it must be implicitly mappable (i.e. the expression is not treating its input as an array and not wanting to output an array)
95. check and set traits which allow optimized selectors to be created
96. if any multi-value inputs, set flag for non-scalar inputs
97. if expression needs transformed, lets do it
98. if we didn't eliminate this expression as a single input scalar or mappable expression, it might need automatic transformation to map across multi-valued inputs (or row by row detection in the worst case)
99. No instantiation.
100. if satisfied, set single input output type and flags
101. find any inputs which will need implicitly mapped across multi-valued rows
102. only set output type
103. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
104. * * expressions + extractions
105. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
106. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
107. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
108. null capabilities should be backed by a nil vector selector since it means the column effectively doesnt exist
109. No instantiation.

110. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
111. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
112. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License.
113. * * expressions + extractions
114. * * Check if an apply function can be 'vectorized', for a given {@link LambdaExpr} and set of {@link Expr} inputs. * If this method returns true, {@link #asVectorProcessor} is expected to produce a {@link ExprVectorProcessor} which * can evaluate values in batches to use with vectorized query engines. * * @see Expr#canVectorize(Expr.InputBindingTypes) * @see Function#canVectorize(Expr.InputBindingTypes, List)
115. * * Builds a 'vectorized' function expression processor, that can build vectorized processors for its input values * using {@link Expr#buildVectorized}, for use in vectorized query engines. * * @see Expr#buildVectorized(Expr.VectorInputBindingTypes) * @see Function#asVectorProcessor(Expr.VectorInputBindingTypes, List)
116. * * Mechanism to supply batches of input values to a {@link ExprVectorProcessor} for optimized processing. Mirrors * the vectorized column selector interfaces, and includes {@link ExprType} information about all input bindings * which exist
117. * * {@link InputBindingTypes} + vectorizations stuff for {@link #buildVectorized}
118. * * Builds a 'vectorized' expression processor, that can operate on batches of input values for use in vectorized * query engines.
119. * * Check if an expression can be 'vectorized', for a given set of inputs. If this method returns true, * {@link #buildVectorized} is expected to produce a {@link ExprVectorProcessor} which can evaluate values in batches * to use with vectorized query engines.
120. * * Check if all provided {@link Expr} can infer the output type as {@link ExprType#isNumeric} with a value of true. * * There must be at least one expression with a computable numeric output type for this method to return true.
121. * * Check if every provided {@link Expr} computes {@link Expr#canVectorize(InputBindingTypes)} to a value of true
122. * * The expression system does not distinguish between {@link ValueType#FLOAT} and {@link ValueType#DOUBLE}, and * cannot currently handle {@link ValueType#COMPLEX} inputs. This method will convert {@link ValueType#FLOAT} to * {@link #DOUBLE}, or null if a null {@link ValueType#COMPLEX} is encountered.
123. * * Check if a function can be 'vectorized', for a given set of {@link Expr} inputs. If this method returns true, * {@link #asVectorProcessor} is expected to produce a {@link ExprVectorProcessor} which can evaluate values in * batches to use with vectorized query engines. * * @see Expr#canVectorize(Expr.InputBindingTypes) * @see ApplyFunction#canVectorize(Expr.InputBindingTypes, Expr, List)
124. only single argument and 2 argument where the radix is constant is currently implemented the canVectorize check should prevent this from happening, but explode just in case
125. * * Builds a 'vectorized' function expression processor, that can build vectorized processors for its input values * using {@link Expr#buildVectorized}, for use in vectorized query engines. * * @see Expr#buildVectorized(Expr.VectorInputBindingTypes) * @see ApplyFunction#asVectorProcessor(Expr.VectorInputBindingTypes, Expr, List)
126. * * Make a {@link VectorValueSelector} for primitive numeric or expression virtual column inputs.

127. * * Return the {@link ColumnCapabilities} which best describe the optimal selector to read from this virtual column. * * The {@link ColumnInspector} (most likely corresponding to an underlying {@link ColumnSelectorFactory} of a query) * allows the virtual column to consider this information if necessary to compute its output type details. * * Examples of this include the {@link ExpressionVirtualColumn}, which takes input from other columns and uses the * {@link ColumnInspector} to infer the output type of expressions based on the types of the inputs. * * @param inspector column inspector to provide additional information of other available columns * @param columnName the name this virtual column was referenced with * @return capabilities, must not be null
128. schema for benchmarking expressions
129. numeric dims
130. string dims
131. * * expressions + extractions
132. Optimization for constant expressions
133. if we got here, lets call it single value string output
134. fallback to
135. if float was explicitly specified preserve it, because it will currently never be the computed output type
136. null constants can sometimes trip up the type inference to report STRING, so check if explicitly supplied output type is numeric and stick with that if so
137. we don't have to check for unknown input here because output type is unable to be inferred if we don't know the complete set of input types
138. always a multi-value string since wider engine does not yet support array types
139. If possible, this should only be used as a fallback method for when capabilities are truly 'unknown', because we are unable to compute the output type of the expression, either due to incomplete type information of the inputs or because of unimplemented methods on expression implementations themselves, or, because a ColumnInspector is not available
140. VectorProcessors used by the "allProcessors" tasks.

git_commits:

1. **summary:** vectorized expressions and expression virtual columns (#10401)
   message: vectorized expressions and expression virtual columns (#10401) * vectorized expression virtual columns * cleanup * fixes * preserve float if explicitly specified * oops * null handling fixes, more tests * what is an expression planner? * better names * remove unused method, add pi * move vector processor builders into static methods * reduce boilerplate * oops * more naming adjustments * changes * nullable * missing hex * more

github_issues:

github_issues_comments:

github_pulls:

1. **title:** vectorized expressions and expression virtual columns
   body: ### Description This PR adds vectorization support to Druid expressions, building on top of #10370, #10388, and a handful of other recent PRs, bringing an often-times _dramatic_ performance improvement for queries which utilize expression virtual columns, in many cases (especially with math functions) 3-5x or more. ``` SELECT SUM(long1 * long2) FROM foo Benchmark (query) (rowsPerSegment) (vectorize) Mode Cnt Score Error Units SqlExpressionBenchmark.querySql 7 5000000 false avgt 5 295.973 ± 20.824 ms/op SqlExpressionBenchmark.querySql 7 5000000 force avgt 5 62.102 ± 7.447 ms/op SELECT SUM(float3 + ((long1 * long4)/double1)) FROM foo Benchmark (query) (rowsPerSegment) (vectorize) Mode Cnt Score Error Units SqlExpressionBenchmark.querySql 9 5000000 false avgt 5 705.915 ± 11.720 ms/op SqlExpressionBenchmark.querySql 9 5000000 force avgt 5 105.604 ± 2.613 ms/op SELECT SUM(long5 - (float3 + ((long1 * long4)/double1))) FROM foo Benchmark (query) (rowsPerSegment) (vectorize) Mode Cnt Score Error Units SqlExpressionBenchmark.querySql 10 5000000 false avgt 5 1002.108 ± 16.323 ms/op SqlExpressionBenchmark.querySql 10 5000000 force avgt 5 189.578 ± 19.048 ms/op SELECT TIME_FLOOR(TIMESTAMPADD(DAY, -1, __time), 'PT1H'), SUM(long1 * long4) FROM foo GROUP BY 1 ORDER BY 1 Benchmark (query) (rowsPerSegment) (vectorize) Mode Cnt Score Error Units SqlExpressionBenchmark.querySql 23 5000000 false avgt 5 956.000 ± 22.778 ms/op SqlExpressionBenchmark.querySql 23 5000000 force avgt 5 300.598 ± 11.318 ms/op SELECT (long1 * long2), SUM(double1) FROM foo GROUP BY 1 ORDER BY 2 Benchmark (query) (rowsPerSegment) (vectorize) Mode Cnt Score Error Units SqlExpressionBenchmark.querySql 24 5000000 false avgt 5 1436.968 ± 15.226 ms/op SqlExpressionBenchmark.querySql 24 5000000 force avgt 5 646.508 ± 104.810 ms/op SELECT string2, SUM(long1 * long4) FROM foo GROUP BY 1 ORDER BY 2 Benchmark (query) (rowsPerSegment)

(vectorize) Mode Cnt Score Error Units SqlExpressionBenchmark.querySql 25 5000000 false avgt 5 493.000 ± 6.847 ms/op SqlExpressionBenchmark.querySql 25 5000000 force avgt 5 89.494 ± 3.105 ms/op ``` This PR focuses on numeric expressions, but does lay out some scaffolding for processing string inputs as well. Implemented so far: * constants, identifiers * numeric unary operators: `-` * numeric binary operators: `+`,`-`, `*`, `/`, `^`, `%`, `>`, `>=`, `<`, `<=`, `==`, `!=` * numeric unary math functions: `atan`, `cos`, `cosh`, `cot`, `sin`, `sinh`, `tan`, `tanh` * numeric binary math functions: `max`, `min`, `pow`, `div` * time functions: `timestamp_floor` (for constant granularity inputs) * other functions (for both numeric and string inputs): `cast`, `parse_long` ### Design On the expression side, the primary new change is the addition of two new methods to `Expr` ```java default boolean canVectorize(InputBindingTypes inputTypes) { return false; } default <T> ExprVectorProcessor<T> buildVectorized(VectorInputBindingTypes inputTypes) { throw Exprs.cannotVectorize(this); } ``` (and similar additions to `Function` and `ApplyFunction`) and where `ExprVectorProcessor` describes an interface for evaluating input vectors: ```java public interface ExprVectorProcessor<TOutput> { ExprEvalVector<TOutput> evalVector(Expr.VectorInputBinding bindings); ExprType getOutputType(); } ``` and `ExprEvalVector` is the vectorized analog of `ExprEval`, but with methods that more closely mimic vectorized value and object selectors (`getLongVector`, `getNullVector`, `getObjectVector`, etc). **additional design description TBD** ``` Benchmark (expression) (rowsPerSegment) (vectorize) Mode Cnt Score Error Units ExpressionVectorSelectorBenchmark.scan long1 * long2 1000000 false avgt 5 73.194 ± 24.618 ms/op ExpressionVectorSelectorBenchmark.scan long1 * long2 1000000 true avgt 5 9.943 ± 4.011 ms/op ExpressionVectorSelectorBenchmark.scan double1 * double3 1000000 false avgt 5 87.064 ± 13.787 ms/op ExpressionVectorSelectorBenchmark.scan double1 * double3 1000000 true avgt 5 6.648 ± 0.914 ms/op ExpressionVectorSelectorBenchmark.scan float1 + float3 1000000 false avgt 5 69.627 ± 20.911 ms/op ExpressionVectorSelectorBenchmark.scan float1 + float3 1000000 true avgt 5 4.961 ± 0.665 ms/op ExpressionVectorSelectorBenchmark.scan (long1 - long4) / double3 1000000 false avgt 5 138.942 ± 39.220 ms/op ExpressionVectorSelectorBenchmark.scan (long1 - long4) / double3 1000000 true avgt 5 24.474 ± 7.421 ms/op ExpressionVectorSelectorBenchmark.scan max(double3, double5) 1000000 false avgt 5 97.063 ± 6.140 ms/op ExpressionVectorSelectorBenchmark.scan max(double3, double5) 1000000 true avgt 5 23.053 ± 4.847 ms/op ExpressionVectorSelectorBenchmark.scan min(double4, double1) 1000000 false avgt 5 75.836 ± 5.153 ms/op ExpressionVectorSelectorBenchmark.scan min(double4, double1) 1000000 true avgt 5 18.574 ± 2.189 ms/op ``` <hr> This PR has: - [ ] been self-reviewed. - [ ] using the [concurrency checklist](https://github.com/apache/druid/blob/master/dev/code-review/concurrency.md) (Remove this item if the PR doesn't have any relation to concurrency.) - [ ] added documentation for new or modified features or behaviors. - [ ] added Javadocs for most classes and all non-trivial methods. Linked related entities via Javadoc links. - [ ] added or updated version, license, or notice information in [licenses.yaml](https://github.com/apache/druid/blob/master/licenses.yaml) - [ ] added comments explaining the "why" and the intent of the code wherever would not be obvious for an unfamiliar reader. - [ ] added unit tests or modified existing tests to cover new code paths, ensuring the threshold for [code coverage](https://github.com/apache/druid/blob/master/dev/code-review/code-coverage.md) is met. - [ ] added integration tests. - [ ] been tested in a test Druid cluster. <!-- Check the items by putting "x" in the brackets for the done things. Not all of these items apply to every PR. Remove the items which are not done or not relevant to the PR. None of the items from the checklist above are strictly necessary, but it would be very helpful if you at least self-review the PR. --> <hr> ##### Key changed/added classes in this PR * `Expr` * `ExpressionVirtualColumn` * `ExpressionVectorSelectors` * `ExpressionSelectors` * `ExpressionPlanner` * `ExpressionPlan` * `ExprVectorProcessor` * `VectorProcessors` * `VectorComparisonProcessors` * `VectorMathProcessors`

github_pulls_comments:

1. This pull request **introduces 1 alert** when merging 88f898fbf4276878b341b063fa525225c589b6bb into 94226f1b3d0aadb3a77cabfd5c4c81c1c3388326 - [view on LGTM.com](https://lgtm.com/projects/g/apache/druid/rev/pr-6e8c7e59a9f5e36f8436cfacbbc2ef6df6547f37) **new alerts:** * 1 for Dereferenced variable may be null
2. Finished looking through all the files... I'm super stoked! 🤘 None of my comments need to be addressed in this PR - some are just for my curiosity, others are nice to haves and can be done in another, smaller 🙃 , pr once this is merged. My last question on this change * Do you think we need a new feature flag to disable vectorized expressions by default, maybe something static like a system property? Since the vectorization flag is enabled by default since 0.19, any expressions running in production will hit this new code path on an upgrade, so cluster operators will have to disable vectorization in case they run into issues we haven't tested.
3. >Do you think we need a new feature flag to disable vectorized expressions by default, maybe something static like a system property? Since the vectorization flag is enabled by default since 0.19, any expressions running in production will hit this new code path on an upgrade, so cluster operators will have to disable vectorization in case they run into issues we haven't tested. I think it measures nice enough that it should be enabled by default, but was planning to do a follow-up PR to add an option to specifically disable vectorized expressions as an

escape hatch for cluster operators to selectively disable this part of vectorization without having to disable all of it.
4. > I think it measures nice enough that it should be enabled by default, but was planning to do a follow-up PR to add an option to specifically disable vectorized expressions as an escape hatch for cluster operators to selectively disable this part of vectorization without having to disable all of it. I see the advantage of having a separate escape hatch for vectorized expressions (hopefully - never to be used 😃) We should probably have a separate discussion on whether it should be enabled or disabled by default in the next PR. I think I'm on the fence because the perf gain is huge, so it would be awesome for everyone upgrading. Maybe I'm just a little more cautious because I've been bitten by upgrades (of other software) in the past. I'll take another look through the changes soonish
5. thanks for review @jihoonson and @suneet-s 🎉
6. Leaving a comment before I forget.. I tagged this PR with "Release Notes" as the vectorized operation can lead to a different query result when it computes floats and doubles. For example, in `DoubleSumVectorAggregator`, it first computes the `sum` of the given vector and then adds it to the accumulated sum in the buffer, while `DoubleSumBufferAggregator` keeps updating the accumulated sum directly. This change is not strictly caused in this PR (actually the issue was introduced in https://github.com/apache/druid/pull/6794), but I guess people will likely see this issue more often, since now SQL expressions can be vectorized. Perhaps we should update all release notes to call out this issue since 0.16.

github_pulls_reviews:

1. javadocs for both of these functions please
2. Can be simplified to `areNumeric(Arrays.asList(args))`.
3. Same here. Can be `canVectorize(Arrays.asList(args))`.
4. Hmm, what is the reason for splitting `VectorInputBindingTypes` and `VectorInputBinding`? The latter extends the former and there is only one implementation of the latter in this PR. Can `VectorInputBinding` extend `InputBindingTypes` and `VectorSizeInspector` instead? `VectorSizeInspector` needs to be moved in that case though as it is currently in `processing`.
5. nit: maybe `Numbers` is a better home.
6. Heh, we have a couple of similar methods such as `Numbers.parseLongObject()`, `GuavaUtils.tryParseLong()`, etc. We should perhaps clean up them by merging similar methods later.
7. This code seems duplicate in binary operators, but I guess it would be nice to keep both `canVectorize()` and `buildVectorized()` together in the same class.
8. Is this something that should be resolved in this PR?
9. Maybe better to explicitly mention that this class is not for string vectors?
10. Nice tests.
11. Should this logic match to its [non-vectorized version](https://github.com/apache/druid/blob/master/core/src/main/java/org/apache/druid/math/expr/Function.java#L507-L512)?
12. Please annotate these with `@Nullable`.
13. Hmm, is this correct? Should this be `capabilities.hasMultipleValues().isMaybeTrue()` instead?
14. Maybe good to mention that `float` is default for a historical reason?
15. Did you intend to check if `rowCount` and `RowCountCursor` match?
16. Looking at what this method does, it seems pretty useful. What do you think about making this test a unit test, so that CI can run? Or, if we already have enough unit tests which cover the same logic, I guess we don't need this to make the benchmark faster.
17. Are warmups and measurements too small?
18. Maybe we should add this testing for the benchmark queries in `CalciteQueryTest` so that CI can run?
19. nit: I guess we will want to keep this method until we merge `ExprType` and `ValueType`. Myabe `getExprType()` better to be more clear?
20. nit: duplicate `ExprType.toValueType(inferredOutputType)`.
21. Should `hasMultipleValues` be set when the plan has the `NON_SCALAR_OUTPUT` trait?
22. Please add javadoc for this method and update the javadoc of the other `capabilities(String columnName)`. Also, should we deprecate the other one if we want to eventually use it only as a fallback?
23. done
24. done
25. I think `VectorInputBindingTypes` and `VectorInputBinding` could be consolidated. They exist in split form from an earlier prototype before I had expression output type inference using `InputBindingTypes` existed. Conceptually it sort of makes sense to me to have them split because you don't really need a full binding backed by selectors in order to build a vectorized expression processor, just the input types and the max vector size. But, there isn't currently a need for this, so I will consider/look into consolidating these interfaces. It probably does

make sense to consider moving `VectorSizeInspector` as well, but I would rather not move into `core` from `processing` in this PR.

26. I agree that we should look into consolidating these methods, not in this PR though.
27. Maybe, but I think I would rather save this until a follow-up PR when we look into consolidating these redundant methods re: the other comment
28. Yeah, I was going to clean this up once all operators are implemented.
29. I don't think it needs to be resolved in this PR. This comment just refers to that the variable input radix (second argument is not a constant) case is not implemented, but `canVectorize` will return false, so this line should not be reached.
30. Eh, is it necessary since strings vectors are object vectors? (and so are the array types, which I think will be handled by similar processors)
31. Oops yes, I stuck the constant 2 argument version in last minute and forgot about this :+1:
32. added
33. No, this is what it was doing in the previous check, looking for either explicitly true or false, which are both ok, but I can't quite remember why unknown isn't ok... But, this does match the previous logic in `ExpressionSelectors.makeDimensionSelector` for checking single column string inputs to determine if it can use `SingleStringInputDimensionSelector`, so I would rather not change it as part of this PR
34. oops yes, added, but moved this entire check into a newly added `ExpressionVectorSelectorsTest`.
35. added new test `ExpressionVectorSelectorsTest`
36. Ah probably for super accurate results if I were making plots, but this was close enough for ballpark measurements while testing changes since the timing usually seemed to settle down after 2 warmup iterations.
37. It seemed like too much work to add this to `CalciteQueryTest`, but I did add a new `SqlVectorizedExpressionSanityTest` which does some of these query tests as part of CI, and migrated this logic there so that this benchmark can call into the test method.
38. fixed
39. Ah, unknown is effectively true in most cases (except for `SINGLE_INPUT_MAPPABLE`), but will change since both of these do imply multiple values
40. I was initially not planning to deprecate for virtual columns that don't care about the other columns in the segment, but I went ahead and did it, since they can just ignore the `ColumnInspector`.
41. javadocs for these functions please.
42. Why is an empty list non numeric? Maybe add javadocs to clarify this behavior.
43. 🎉 so much more vectorization!
44. Does this mean a virtual column can't reference another virtual column? since the ColumnInspector is built with just he `baseRowSignature`?
45. should outNulls be null if we're not in sql compatible mode?
46. hmm interesting that this used to pass 🤔
47. Similar question as the `UnivariateFunctionProcessor` I think with it written like this, `ExprEvalDoubleVector` and `ExprEvalLongVector` can't take advantage of the fact that there are no nulls in default mode. I don't know if there are any functions that produce nulls though... maybe that's why we need to do it this way?
48. Should we add a comment that this vector isn't thread safe? - because of how `computeNumbers` is written Are any of them thread safe?
49. If I'm reading this correctly, this change isn't really needed in this PR, but it's some consolidation that was done across the various aggregator factories
50. Hmm, should `RowSignature` simply implement `ColumnInspector`?
51. Could you add these details in the comment?
52. Ah yeah, I was just consolidating. Since these were the same across all 3 still, I've moved this into a new static `AggregatorUtil.canVectorize` method.
53. These (and the selectors on top of them) don't need to be thread safe since only a single thread should be doing this per segment
54. ah it still parses into the expression just complains about syntax on stderr
55. it was easier to just always set it, but yeah this is an area that could probably be optimized a bit since we could ignore it totally for default mode I guess. I sort of dream of a world where default mode value coercion madness only lives at the borders and doesn't exist in expressions though...
56. Like i said in the other comment i just did this way because was slightly easier, it is worth revisiting this in follow-up work
57. yeah, but this was true before. It would be possible to examine the expressions and resolve an execution order probably to allow it, but we haven't built it yet. This code would need to change to match I think
58. there was nothing to compute an output type from, im not sure if nothing counts as numeric. added javadocs to explain this and other methods
59. added
60. added
61. oof, yes, changed

62. It would be nice to explain the relationship between `<T>` and `getOutputType` here and in similar docs. Specifically I'm thinking something like "As an implementor of the interface, do I need to ensure that the types are always the same? What happens if the types don't match? Is there something in the system that will throw an exception? or will be slow, because of some implicit casting somewhere else?"

jira_issues:

jira_issues_comments: