

Building a SIMD supported vectorized native engine for Spark SQL

Chendi Xue(chendi.xue@intel.com), Software Engineer Yuan Zhou(yuan.zhou@intel.com), Software Engineer Intel Corp

Agenda

- Native SQL Engine Introduction
- Native SQL Engine Design
 - Columnar Data Source
 - Columnar Shuffle
 - Columnar Compute
 - Memory Management
- Summary

#DataTeams #DataAlSummit

DATA+AI SUMMIT EUROPE

Motivations for Native SQL Engine

- Issue of current Spark SQL Engine:
 - Internal row based, difficult to use SIMD optimizations
 - High GC Overhead under low memory
 - JIT code quality relies on JVM, hard to tune
 - High overhead of integration with other native library

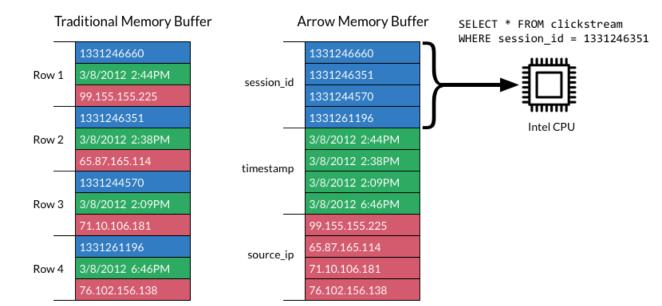
Proposed Solution

Issue of current Spark SQL Engine:

- Internal row based, not possible to use SIMD
- → Columnar-based Arrow Format
- High GC Overhead under low memory
- ightarrow native codes for core compute instead of java
- JIT code quality relies on JVM, hard to tune
- \rightarrow cpp / llvm / assembly code generation
- High overhead of integration with other native library
- \rightarrow Lightweighted JNI based call framework

DATA+AI SUMMIT EUROPE

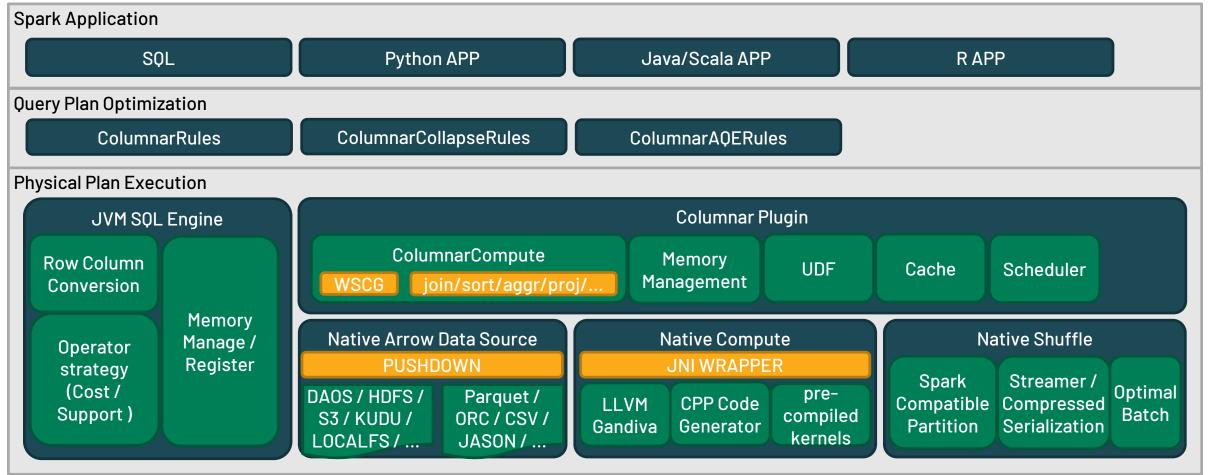
	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138



#DataTeams #DataAlSummit

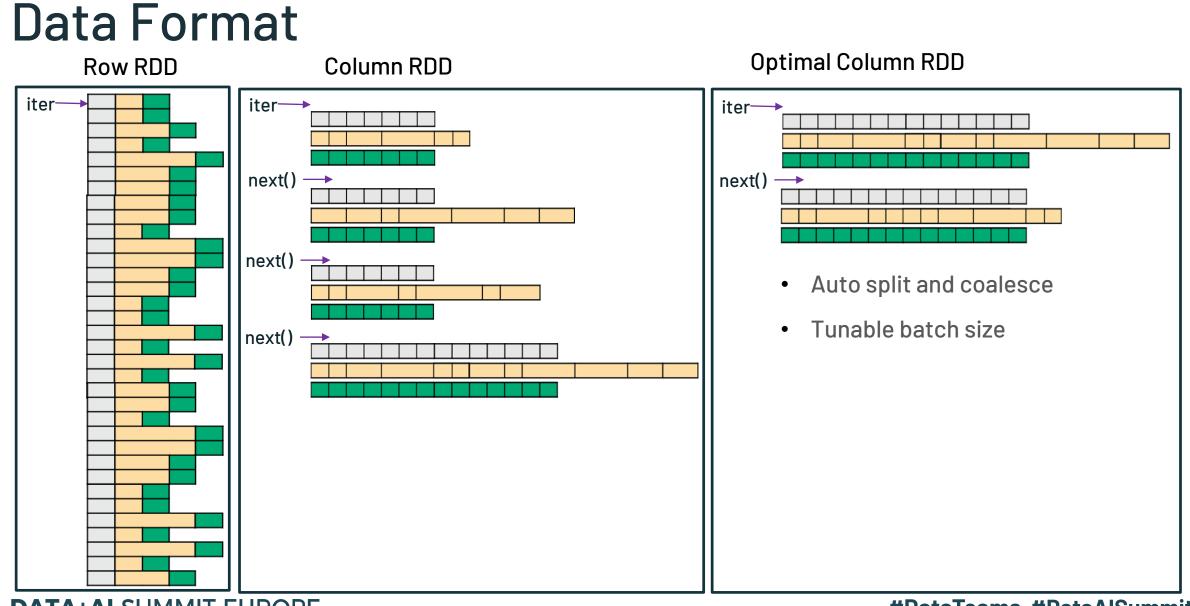
Source: https://www.slideshare.net/dremio/apache-arrow-an-overview

Native SQL Engine Layers



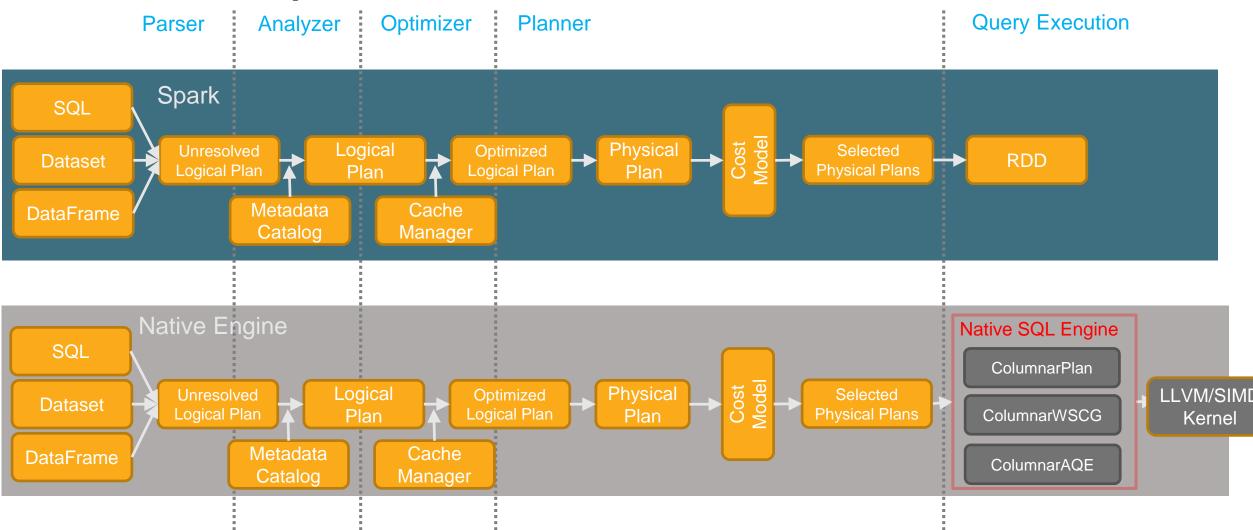
- A standard columnar data format as basic data format
- Data keeps on off-heap, data operations offload to highly optimized native library

DATA+AI SUMMIT EUROPE

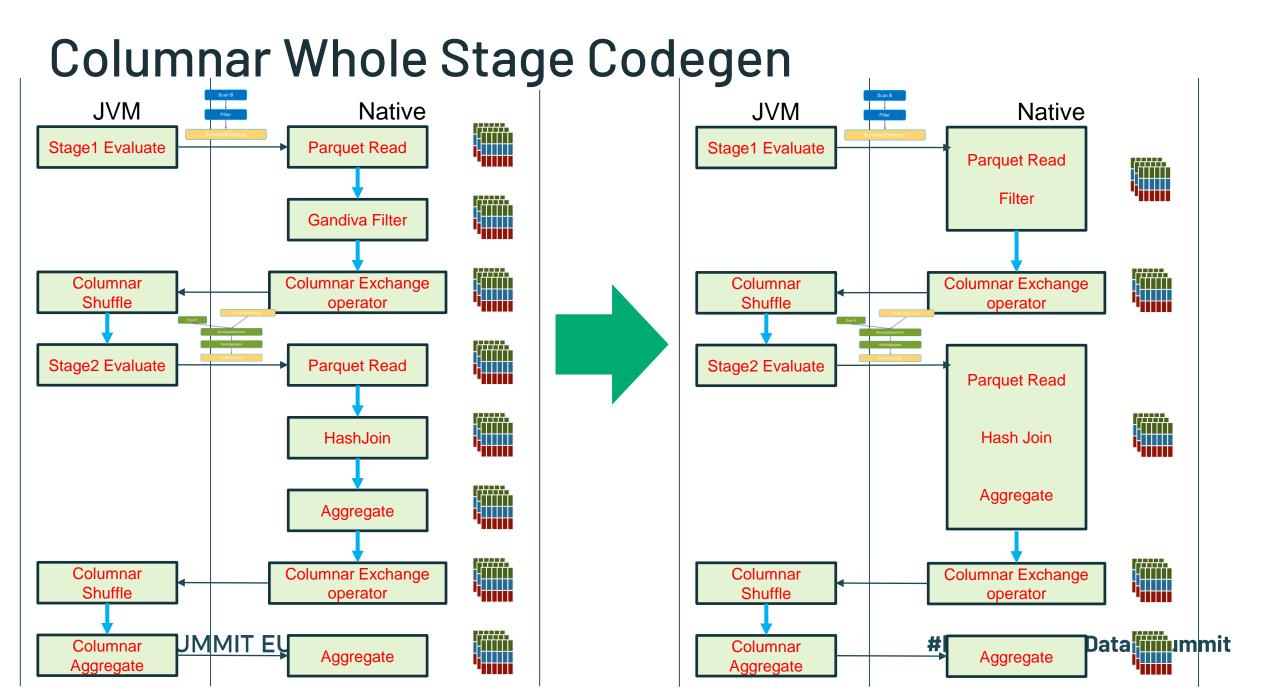


DATA+AI SUMMIT EUROPE

Columnar Spark Plan Rules



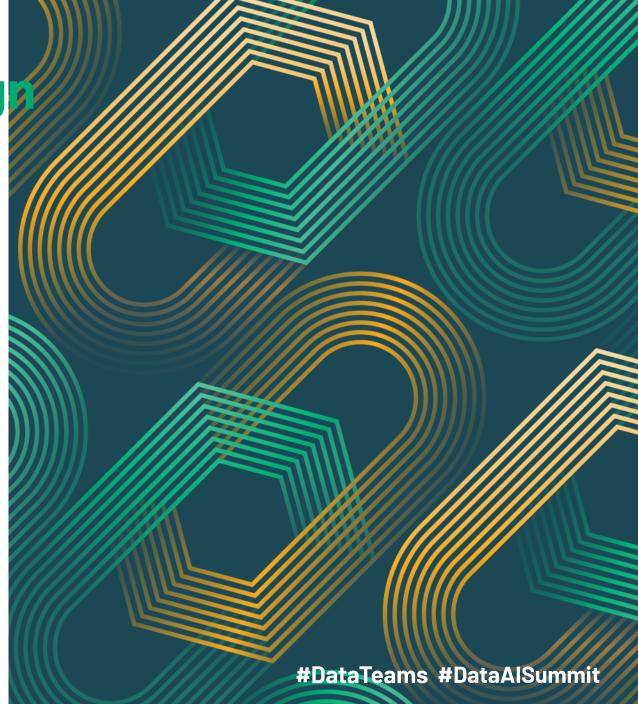
DATA+AI SUMMIT^{EUROPE}



Native SQL Engine Desig

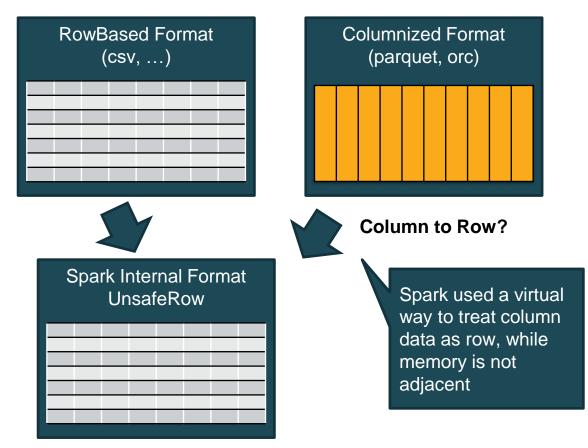
- Columnar Data Source
- Columnar Shuffle
- Columnar Compute
- Memory Management



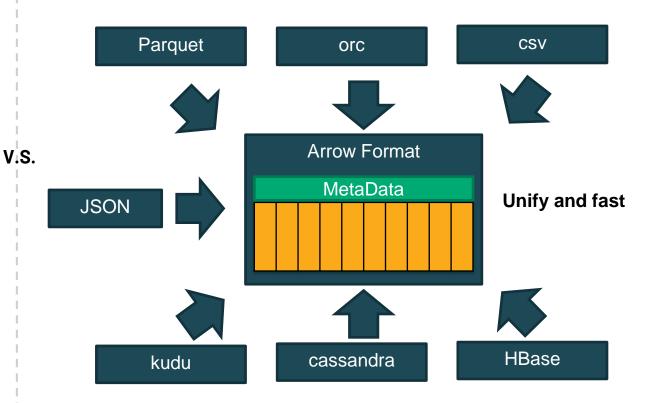


Columnar Data Source

Row-based Data Source



Arrow based Data Source



DATA+AI SUMMIT EUROPE

Columnar Data Source

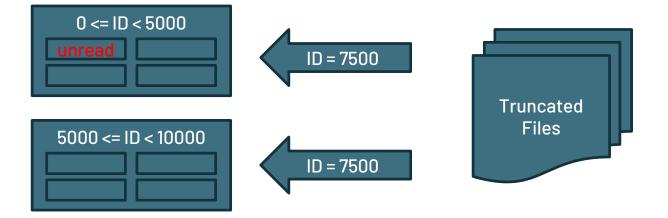
Spark Arrow DataSource (pyspark, thriftserver, sparksql,...)

Arrow Java Datasets API (Zero data copy, memory reference only) Arrow C++ Datasets API (HDFS, localFS, S3A) (Parquet, ORC, CSV, ..)

DATA+AI SUMMIT EUROPE

Columnar Data Source

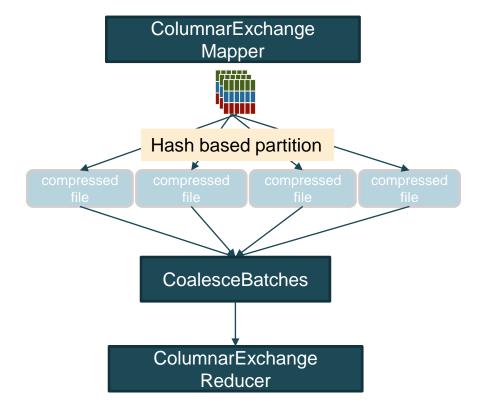
- Features
 - Fast / Parallel / Auth supported Native Libraries for HDFS / S3A / Local FS
 - PushDown supported Pre-executed statistics/metadata filters
 - Partitioned File and DPP enabled



DATA+AI SUMMIT EUROPE

Columnar Shuffle

- Hash-based partitioning(split) with LLVM optimized execution
- Ser/de-ser based on arrow record batch
- Efficient data compression for different data format
- Coalesce batches during shuffle read
- Supports Adaptive Query Execution(AQE)



DATA+AI SUMMIT EUROPE

Supported SQL Operators Overview

DATA+AI SUMMIT EUROPE

Operators	Expression	Expression
WindowExec	NormalizeNaNAndZero	IsNotNull
UnionExec	Subtract	GreaterThanOrEqual
ExpandExec	Substring	GreaterThan
SortExec	ShiftRight	EqualTo
ScalarSubquery	Round	ExtractYear
ProjectExec	PromotePrecision	Divide
ShuffledHashJoin	Multiply	Concat
BroadcastJoinExec	Literal	Coalesce
FilterExec	LessThanOrEqual	CheckOverflow
ShuffleExchangeExec	LessThan	Cast
BroadcastExchangeExec	KnownFloatingPointNormalized	CaseWhen
datasources.v2.BatchScanExec	IsNull	BitwiseAnd
datasources.v1.FileScanExec	And	AttributeReference
HashAggregateExec	Add	Alias

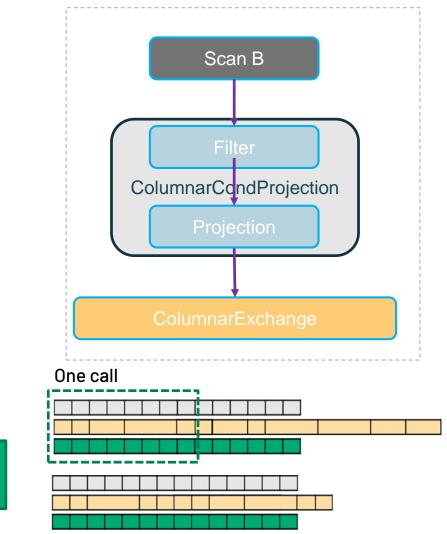
Automatically fallback to row-based execution if there are unsupported operators/expressions

Columnar Projector & Filter

- LLVM IR based execution w/ AVX optimized code
- Based on Arrow Gandiva, extended with more functions
- Combined filter & projection into CondProjector
- ColumnarBatch based execution

Example: If (Field_A + Field_B + Field_C) as Field_new > Field_D output [Field_new, Field_A, Field_B, Field_C, Field_D]



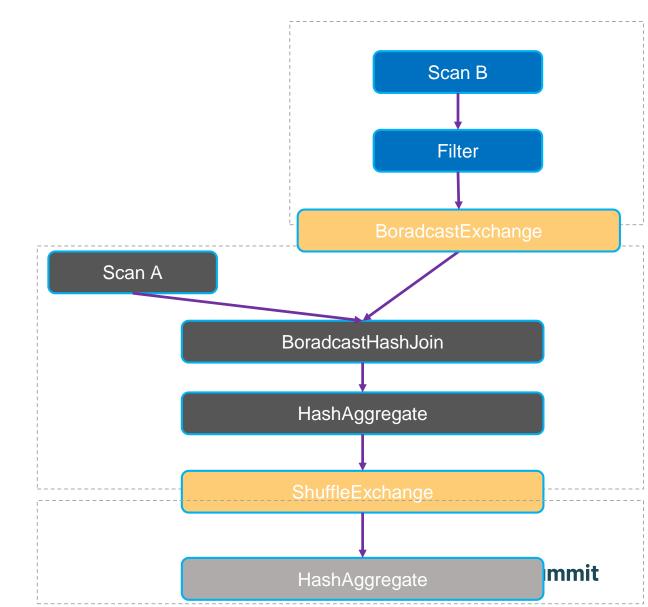


IIVM

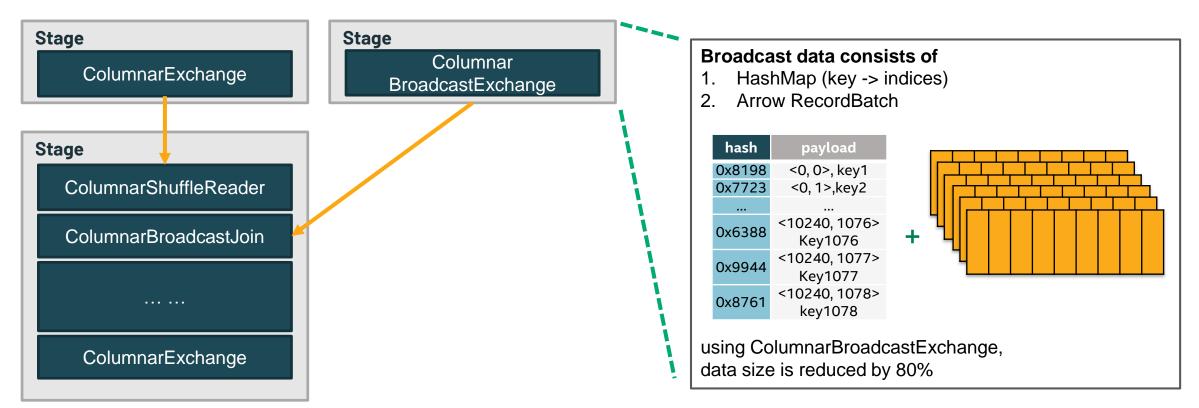
IR

Native Hashmap

- Faster Hashmap building and lookup w/ AVX optimizations
- Compatible with Spark's
 murmurhash if configured
- Performance benefits for HashAggregation and HashJoins



ColumnarBroadcastHashJoin



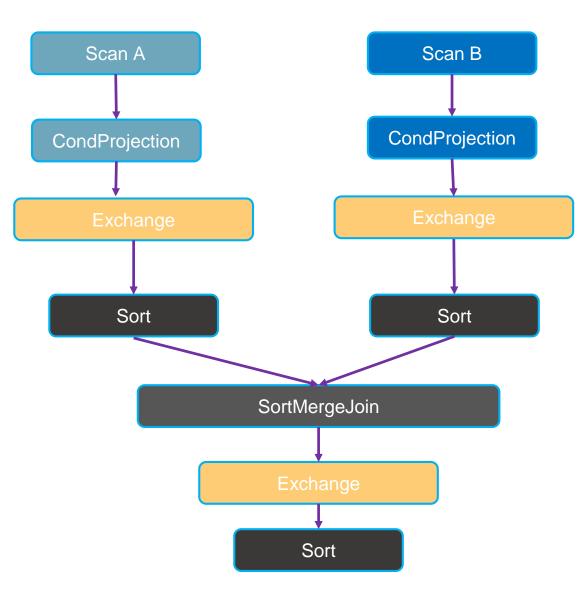
Stage

ColumnarShuffleReader

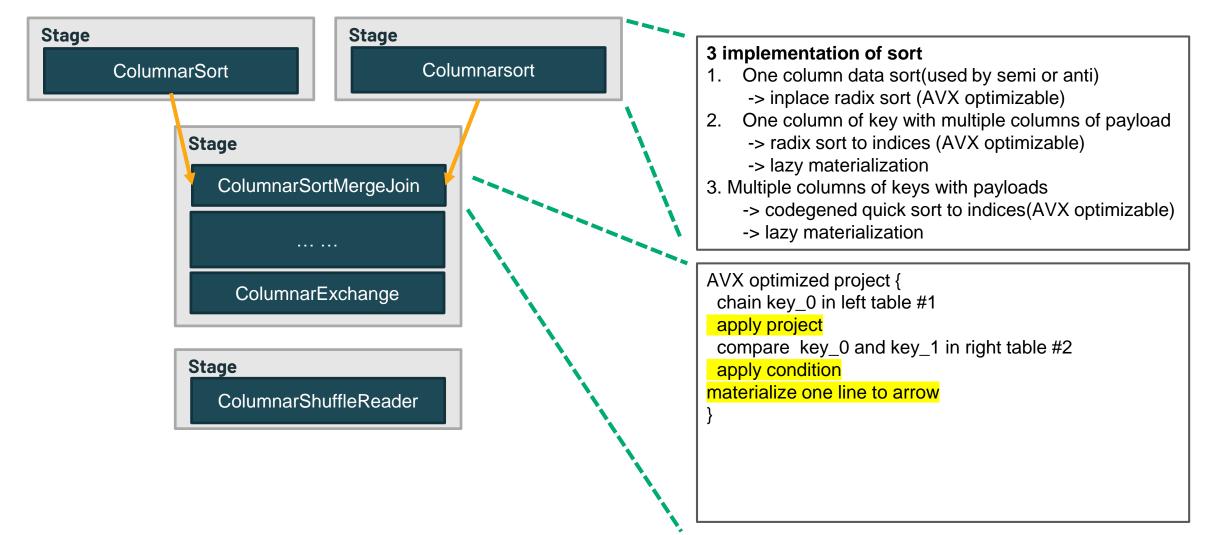
DATA+AI SUMMIT EUROPE

Native Sort

- Faster sort implementation w/ AVX optimizations
- Most powerful algorithms used for different data structures
- Performance benefits for sort and sort merge joins

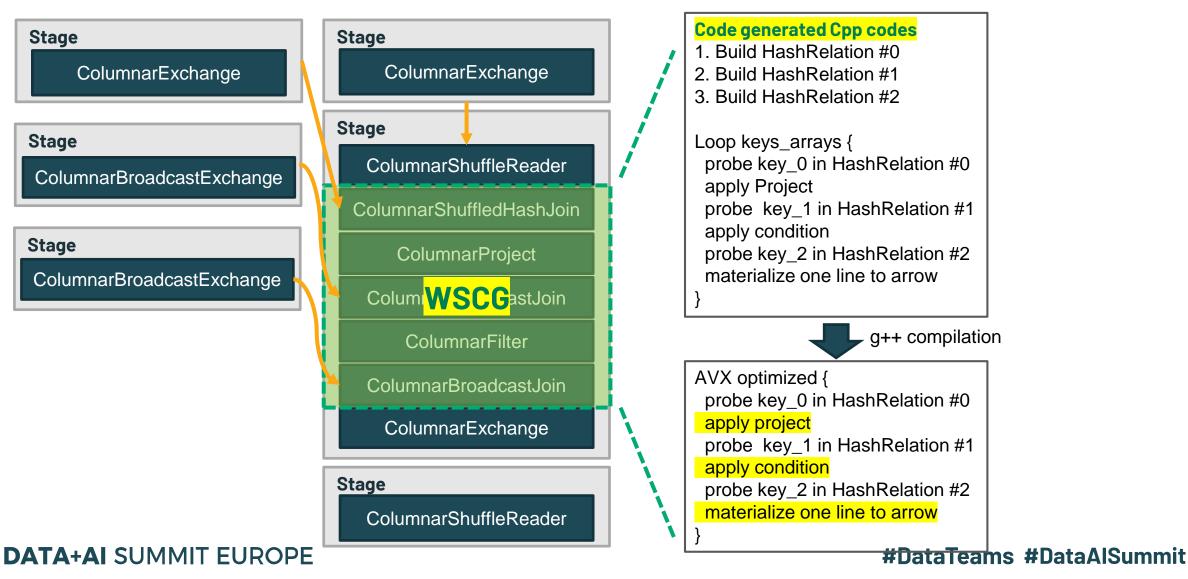


ColumnarSort and ColumnarSortMergeJoin

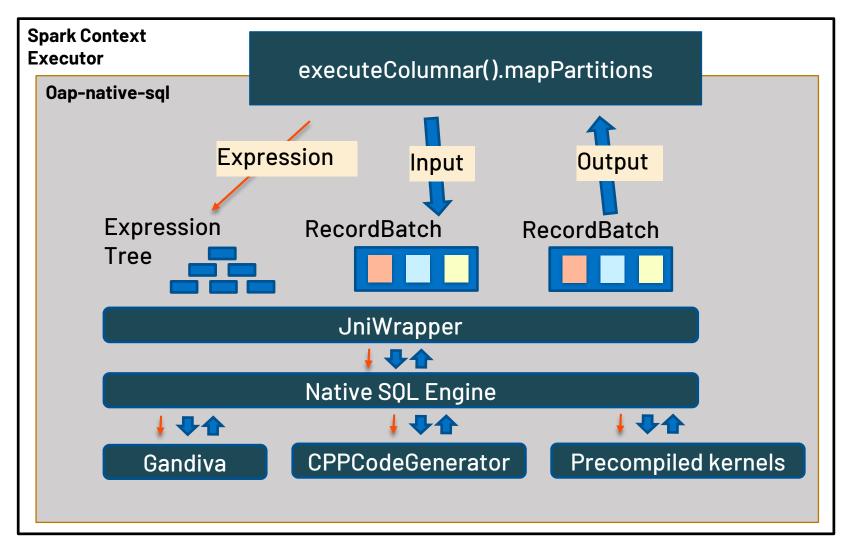


DATA+AI SUMMIT EUROPE

Columnar WholeStageCodeGen

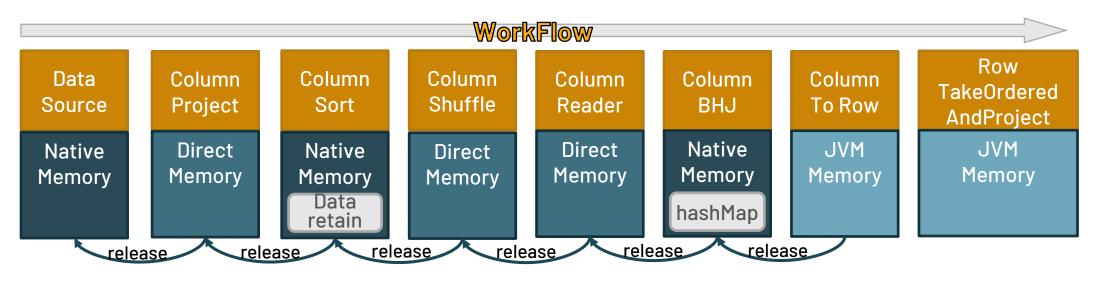


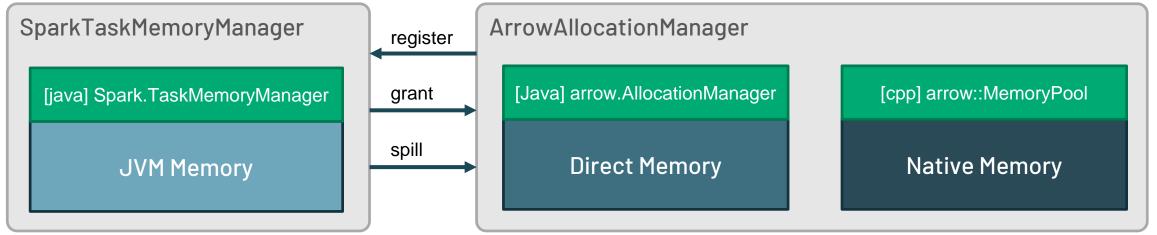
Native SQL engine call flow



DATA+AI SUMMIT EUROPE

Memory Management





DATA+AI SUMMIT EUROPE

Example run of TPCH-Q4

Scan arrow default.orders

number of files read: 576 scan time total (min, med, max (stageld: taskld)) 3.8 m (266 ms, 378 ms, 1.1 s (stage 59.0: task 5600)) metadata time: 0 ms size of files read: 99.4 GIB number of output rows: 2,304,000,000

ColumnarConditionProject

output_batches: 113,472 input_batches: 113,472 number of output rows: 88,098,609 totaltime_condproject total (min, med, max (stageld: taskld)) 2.2 s (0 ms, 0 ms, 545 ms (stage 59.0: task 5628))

ColumnarExchange

DATA+AI SUMMIT EUR

shuffle records written: 113,472 shuffle write time total (min, med, max (stageld: taskId)) 22.0 s (17 ms, 34 ms, 99 ms (stage 59.0: task 5681)) totaltime_split total (min, med, max (stageld: taskId)) 2.2 m (110 ms, 210 ms, 869 ms (stage 59.0: task 5935)) number of input rows: 88,098,609 shuffle spill time total (min, med, max (stageld: taskId)) 0 ms (0 ms, 0 ms, 0 ms (stage 59.0: task 5710)) records read: 165.888 totaltime_computepid total (min, med, max (stageId: taskId)) 2.8 s (2 ms, 4 ms, 25 ms (stage 59.0: task 5913)) local bytes read total (min, med, max (stageld; taskId)) 33.1 MiB (64.7 KiB, 117.6 KiB, 149.4 KiB (stage 63.0: task 6817)) fetch wait time total (min, med, max (stageld: taskId)) 1 ms (0 ms, 0 ms, 1 ms (stage 63.0: task 6875)) remote bytes read total (min, med, max (stageld: taskId)) 1156.8 MiB (4.0 MiB, 4.0 MiB, 4.1 MiB (stage 63.0: task 6914)) avg read batch num rows (min, med, max (stageld: taskId)): (458, 529, 587 (stage 63.0: task 6753)) shuffle bytes spilled total (min, med, max (stageld: taskId)) 0.0 B (0.0 B, 0.0 B, 0.0 B (stage 59.0: task 5710)) number of output rows: 88,098,609 local blocks read: 4,608 remote blocks read: 161.280 data size total (min, med, max (stageld: taskId)) 1735.5 MiB (3.0 MiB, 3.0 MiB, 3.0 MiB (stage 59.0: task 5814)) shuffle bytes written total (min, med, max (stageld; taskId)) 1189.9 MiB (2.1 MiB, 2.1 MiB, 2.1 MiB (stage 59.0: task 5814))

Scan arrow default.lineitem

number of files read: 576 scan time total (min, med, max (stageld: taskld)) 7.9 m (427 ms, 817 ms, 1.5 s (stage 60.0: task 6528)) metadata time: 0 ms size of files read: 383.9 GiB number of output rows: 9,216,034,399

ColumnarConditionProject

output_batches: 453,209 input_batches: 453,209 number of output rows: 5,826,623,976 totaltime_condproject total (min, med, max (stageld: taskld)) 2.7 s (0 ms, 2 ms, 176 ms (stage 60.0: task 6114))

ColumnarExchange

shuffle records written: 453,209 shuffle write time total (min, med, max (stageld: taskId)) 5.4 m (323 ms, 524 ms, 1.1 s (stage 60.0: task 6359)) totaltime_split total (min, med, max (stageld: taskId)) 15.8 m (1.3 s, 1.6 s, 2.5 s (stage 60.0: task 6218)) number of input rows: 5.826.623.976 shuffle spill time total (min, med, max (stageld: taskId)) 2.4 m (173 ms, 240 ms, 409 ms (stage 60.0: task 6555)) records read: 331.776 totaltime_computepid total (min, med, max (stageId: taskId)) 56.0 s (79 ms, 97 ms, 123 ms (stage 60.0: task 6533)) local bytes read total (min, med, max (stageld: taskId)) 916.8 MiB (3.2 MiB, 3.2 MiB, 3.2 MiB (stage 63.0: task 6930)) fetch wait time total (min, med, max (stageld: taskId)) 3.2 m (185 ms, 585 ms, 1.4 s (stage 63.0: task 6879)) remote bytes read total (min, med, max (stageld: taskId)) 31.3 GiB (111.3 MiB, 111.4 MiB, 111.6 MiB (stage 63.0: task 6793)) avg read batch num rows (min, med, max (stageld: taskId)): (17,226, 17,553, 17,812.5 (stage 63.0: task 6744)) shuffle bytes spilled total (min, med, max (stageld: taskId)) 18.8 GiB (33.4 MiB, 33.4 MiB, 33.4 MiB (stage 60.0; task 6417)) number of output rows: 5.826.623.976 local blocks read: 4,608 remote blocks read: 161,280 data size total (min, med, max (stageld: taskId)) 44.1 GiB (78.3 MiB, 78.4 MiB, 78.4 MiB (stage 60.0: task 6542)) shuffle bytes written total (min, med, max (stageld; taskId)) 32.2 GiB (57.3 MiB, 57.3 MiB, 57.3 MiB (stage 60.0: task 6542))

#DataTeams #DataAlSummit

ColumnarCustomShuffleReader

ColumnarCustomShuffleReader

Summary

- AVX instructions can greatly improve performance on SQL workloads
- Native SQL is open sourced. For more details please visit: <u>https://github.com/Intel-bigdata/OAP</u>
- Native SQL engine is under heavy development, works for TPC-H/TPC-DS now



DATA+AI SUMMIT EUROPE

Feedback

Your feedback is important to us. Don't forget to rate and review the sessions.

DATA+AI SUMMIT EUROPE

Legal Information: Benchmark and Performance Disclaimers

- Performance results are based on testing as of Feb. 2019 & Aug 2020 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information, see Performance Benchmark Test Disclosure.
- Configurations: see performance benchmark test configurations.

DATA+AI SUMMIT EUROPE

Notices and Disclaimers

- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- This document contains information on products, services and/or processes in development. All
 information provided here is subject to change without notice. Contact your Intel representative
 to obtain the latest forecast, schedule, specifications and roadmaps.
- The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.
- Intel, the Intel logo, Xeon, Optane, Optane Persistent Memory are trademarks of Intel Corporation in the U.S. and/or other countries.
- *Other names and brands may be claimed as the property of others
- © Intel Corporation.