Apache DataFusion Comet 0.5.0 Release

This document is read only now and the blog post PR is at:

https://github.com/apache/datafusion-site/pull/52

The Apache DataFusion PMC is pleased to announce version 0.5.0 of the Comet subproject.

Comet is an accelerator for Apache Spark that translates Spark physical plans to DataFusion physical plans for improved performance and efficiency without requiring any code changes.

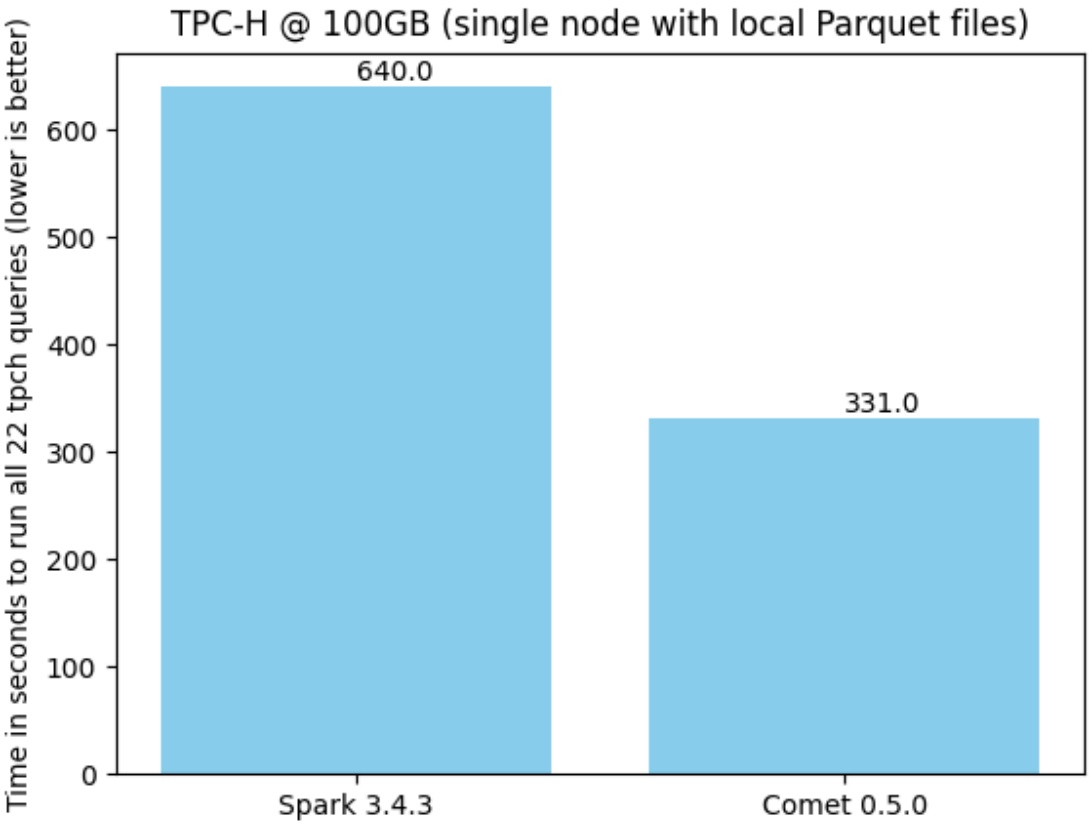
Comet runs on commodity hardware and aims to provide 100% compatibility with Apache Spark. Any operators or expressions that are not fully compatible will fall back to Spark unless explicitly enabled by the user. Refer to the compatibility guide for more information.

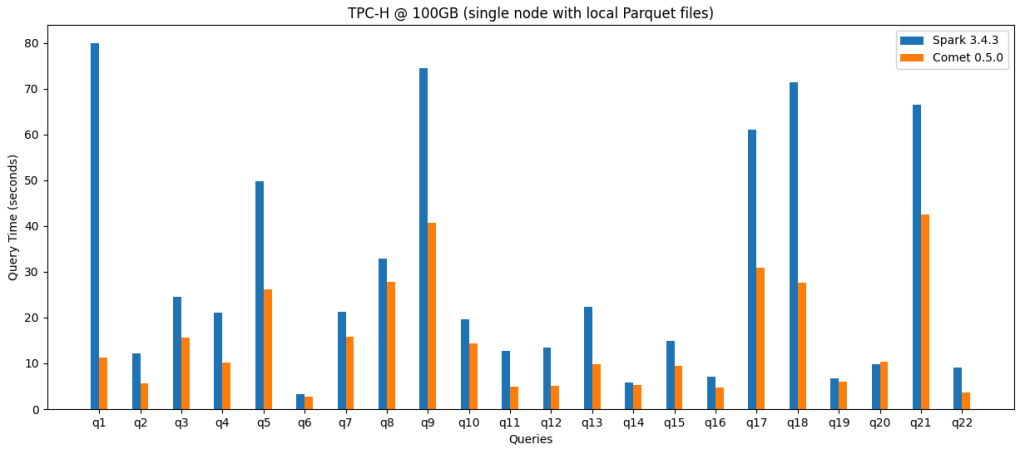
This release covers approximately eight weeks of development work and is the result of merging 69 PRs from 15 contributors. The complete change log is available here (TBD).

Release Highlights

Performance

Comet 0.5.0 achieves a 1.9x speedup for single-node TPC-H @ 100 GB, an improvement from 1.7x in the previous release.





Shuffle Improvements

Comet now supports multiple compression algorithms for compressing shuffle files. Previously, only ZSTD was supported but Comet now also supports LZ4 and Snappy. The default is now

LZ4, which is faster than ZSTD in these benchmarks but has a lower compression ratio. ZSTD may be a better choice when the compression ratio is more important than CPU overhead.

Previously, Comet used Arrow IPC to encode record batches into shuffle files. Although Arrow IPC is a good general-purpose framework for serializing Arrow record batches, we found that we could get better performance using a custom serialization approach optimized for Comet. One optimization is that the schema is encoded once per shuffle operation rather than once per batch.

Comet provides two shuffle implementations. Comet native shuffle is the fastest and performs repartitioning in native code. Comet columnar shuffle delegates to Spark to perform repartitioning and is used in cases where native shuffle is not supported, such as with RangePartitioning. Comet generally tries to use native shuffle first, then columnar shuffle, and finally falls back to Spark if neither is supported. There was a bug in previous releases where Comet would sometimes fall back to Spark shuffle if native shuffle was not supported and missed opportunities to use columnar shuffle. This bug was fixed in this release but currently requires the configuration setting “spark.comet.exec.shuffle.fallbackToColumnar=true”. This will be enabled by default in the next release.

On-Heap Memory

Comet 0.4.0 required Spark to be configured to use off-heap memory. In this release it is no longer required and there are multiple options for configuring Comet to use on-heap memory instead. More details are available in the Comet Tuning Guide.

Metrics

- Improved Spark SQL metrics for native Parquet scan and native shuffle

Crate reorganization

One of the goals of the Comet project is to make Spark-compatible functionality available to other projects that are based on DataFusion. In this release, many implementations of Spark-compatible expressions were moved from the unpublished “datafusion-comet” crate, which provides the native part of the Spark plugin, into the “datafusion-comet-spark-expr” crate. There is also ongoing work to reorganize this crate to move expressions into subfolders named after the “group” name that Spark uses to organize expressions. For example, there are now subfolders named “agg_funcs”, “datetime_funcs”, “hash_funcs”, and so on.

Update on Complex Type support

Good progress has been made with proof-of-concept work using DataFusion's ParquetExec, which has the advantage of supporting complex types. This work is available on the comet-parquet-exec branch, and the current focus is on fixing test regressions, particularly regarding timestamp conversion issues.

Getting Involved

The Comet project welcomes new contributors. We use the same Slack and Discord channels as the main DataFusion project and have a weekly DataFusion video call.

The easiest way to get involved is to test Comet with your current Spark jobs and file issues for any bugs or performance regressions that you find.

There are also many good first issues waiting for contributions. See the contributors guide to get started with contributing to the project.