Lecture #22: System Analysis (Amazon Redshift)

15-721 Advanced Database Systems (Spring 2024)
`https://15721.courses.cs.cmu.edu/spring2024/`
Carnegie Mellon University
Prof. Andy Pavlo

1 Historical Context

During the early to mid-2000s, numerous companies attempted to create distributed versions of PostgreSQL, achieving varying levels of success. Many of these OLTP and OLAP offerings were eventually acquired by larger companies for substantial amounts, only to be discontinued shortly after.

As 2010 approached, AWS was gaining momentum, and they sought to add an OLAP DBMS to their catalog. An internal debate ensued regarding whether to develop an in-house solution or purchase an existing product. One of the few remaining systems from the PostgreSQL distributed database boom was ParAccel, a share-nothing distributed OLAP database. Amazon ultimately decided to invest around $20 million in their Series E funding round, securing a license to the source code.

Rebranded as Redshift, Amazon began offering the OLAP DBMS on AWS in 2011. Since then, extensive efforts have been made to transform Redshift into a state-of-the-art, enterprise-grade DBaaS. Today, Redshift generates billions of dollars in revenue for Amazon annually.

2 Overview

Redshift [1, 2] is Amazon's flagship OLAP DBaaS.

- Began as ParAccel
- Switched to S3 disaggregated storage in 2017
- Added serverless in 2022
- More of traditional data warehouse compared to BigQuery and Spark
- By default, data is ingested into Redshift Managed Storage (RMS)
- Goal of removing as much Admin/Config choices from users

Flavors

Since its inception, Redshift has been offered in several flavors.

Today, Spectrum is a feature of 'Redshift' that can be enabled and Athena remains a separate service.

3 Architecture

Topics in bold will be covered with more depth.

- Shared-disk storage
- Push-based Vectorized Query Processing

OG Redshift (2012)	Share-nothing DBMS with managed storage, can spill to S3.
Athena (2016)	Rebrand of Presto, lakehouse query engine for S3 data files.
Redshift Spectrum (2017)	Extension to OG that allows reading directly from S3 bypassing RMS.

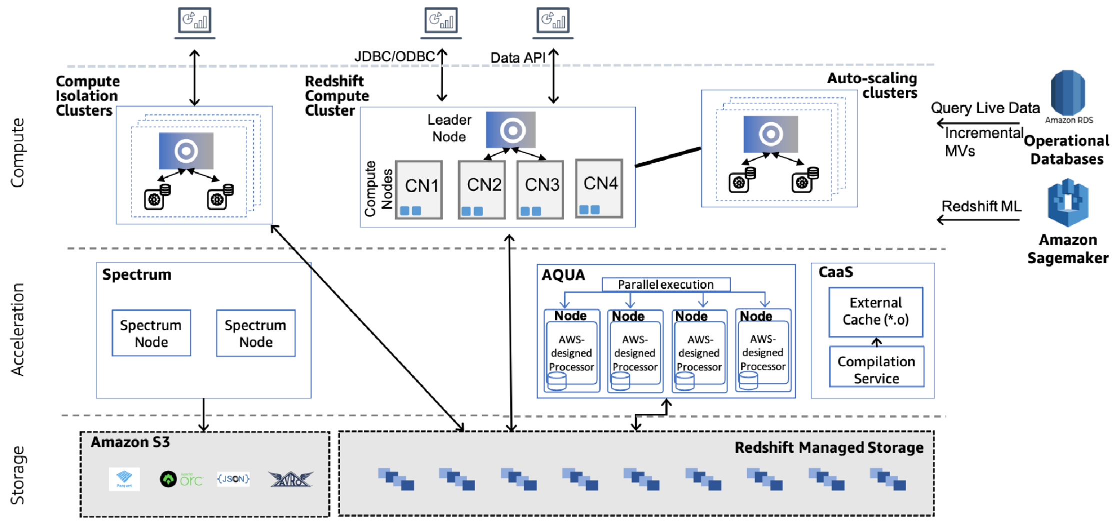

Figure 1: Amazon Redshift Architecture

- **Transpilation Query Codegen**
- **Pre-Compiled Primitives**
- **Compute-Side Caching**
- Proprietary PAX Columnar Storage
- Supports Sort-Merge, Nested-Loop, and Hash Joins
- **Hardware accelerator AQUA**
- Stratified Query Optimizer

3.1 Query Execution

Amazon Redshift employs a **push-based vectorized query execution engine**, with HIQUE [3] style transpilation code generation and late materialization. Although it may appear to be pull-based in the paper, the original design presented performance issues, leading to the adoption of a push-based approach. Unlike some systems, Redshift does not rely on automatic vectorization and instead utilizes fully hand-written SIMD with AVX2 intrinsics.

To optimize cache performance, Redshift incorporates software prefetching to load memory into the L3 caches, preventing scans from stalling while filling vectors. Prefetching instructions are strategically inserted into the generated code at calculated intervals. If too much work is performed between fetches, data may be evicted; conversely, if insufficient work is done, the data may not be ready.

Redshift offers some adaptivity, albeit less aggressive compared to BigQuery, Snowflake, and others. For instance, it attempts to use vectorized versions of string functions for ASCII data and falls back if necessary. Another area where adaptivity is leveraged is during the build phase of a hash-join when constructing bloom filters. If the hash table becomes so large that it risks spilling to disk, Redshift can adaptively increase the bloom filter size to prevent disk access.

Lastly, to minimize compilation time, Redshift employs **pre-compiled primitives**. These are hand-tuned, optimized sections of code that are injected into compiled queries, improving overall performance.

Compilation Service

Redshift employs separate computing nodes dedicated to compiling query plans, with each node utilizing GCC for compilation. The service implements an aggressive caching strategy to enhance performance.

- Each Redshift customer possesses a local cache wherein compiled versions of templated query fragments are stored.
- A global cache is available to the entire fleet of Redshift customers.

If a compiled query fragment is not found in the local cache, the system checks the global cache. If the fragment remains elusive, it proceeds to compile the query plan using the dedicated compilation nodes.

Background workers continuously monitor for new releases or updates to the Redshift DBMS. Upon detecting a new version, these workers proactively recompile query plans to ensure compatibility and optimization with the latest DBMS version.

Hardware Acceleration

In 2021, Amazon Web Services (AWS) introduced the **Advanced Query Accelerator (AQUA)** for Amazon Redshift to enhance the performance of query processing.

AQUA was initially introduced as a specialized hardware accelerator for Redshift, employing separate compute and cache nodes with FPGAs for enhanced query processing performance. However, it was subsequently replaced by Nitro cards integrated into the compute nodes, potentially offering comparable or superior performance with a more streamlined architecture.

Query Optimization

Redshift employs a **stratified query optimizer** that follows a multi-step process to optimize query execution. Initially, the optimizer executes rewrite rules on incoming queries to potentially enhance their performance. After the rewrite phase, the optimizer employs a cost-based query optimization strategy to evaluate different execution plans for the query and selects the one with the lowest estimated cost.

For queries involving Redshift **Spectrum**, Redshift employs optimization techniques to improve performance. Specifically, Redshift extracts Parquet/ORC zone maps from the data stored in S3 to enable predicate pushdown.

Additionally, AWS developed the **Query Rewriting Framework (QRF)** as a mechanism for introducing rules to manipulate query plans. These rules can be used to modify query plans at runtime, either to optimize performance, enforce access controls, or apply other custom logic.

4 Storage

Amazon Redshift relies on a sophisticated storage architecture to efficiently manage and query vast datasets. Central to this architecture is the **Redshift Managed Store (RMS)**, a dedicated storage infrastructure optimized for performance and scalability.

Redshift Managed Store

- RMS comprises separate storage nodes utilizing SSDs, dedicated to storing data in Amazon's proprietary file format.
- Compute nodes within a Redshift cluster may also maintain a local SSD cache.

If the local SSD cache on compute nodes reaches its capacity limit, Redshift will spill excess data to S3.

Column Encoding Schemes

Users can specify encoding schemes per column when creating table schemas, optimizing storage and query performance within Redshift. Supported schemes include: Dictionary, Delta, RLE, and AZ64.

References

[1] N. Armenatzoglou, S. Basu, N. Bhanoori, M. Cai, N. Chainani, K. Chinta, V. Govindaraju, T. Green, M. Gupta, S. Hillig, E. Hotinger, Y. Leshinksy, J. Liang, M. McCreedy, F. Nagel, I. Pandis, P. Parchas, R. Pathak, O. Polychroniou, F. Rahman, G. Saxena, G. Soundararajan, S. Subramanian, and D. Terry. Amazon redshift re-invented. In *SIGMOD/PODS 2022*, 2022. URL `https://www.amazon.science/publications/amazon-redshift-re-invented`.

[2] A. Gupta, D. Agarwal, D. Tan, J. Kulesza, R. Pathak, S. Stefani, and V. Srinivasan. Amazon redshift and the case for simpler data warehouses. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 1917–1923, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450327589. doi: 10.1145/2723372.2742795. URL `https://doi.org/10.1145/2723372.2742795`.

[3] K. Krikellas, S. Viglas, and M. Cintra. Generating code for holistic query evaluation. In *IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 613–624, United States, Mar. 2010. Institute of Electrical and Electronics Engineers (IEEE). ISBN 978-1-4244-5445-7. doi: 10.1109/ICDE.2010.5447892.