Data Warehouse Internals

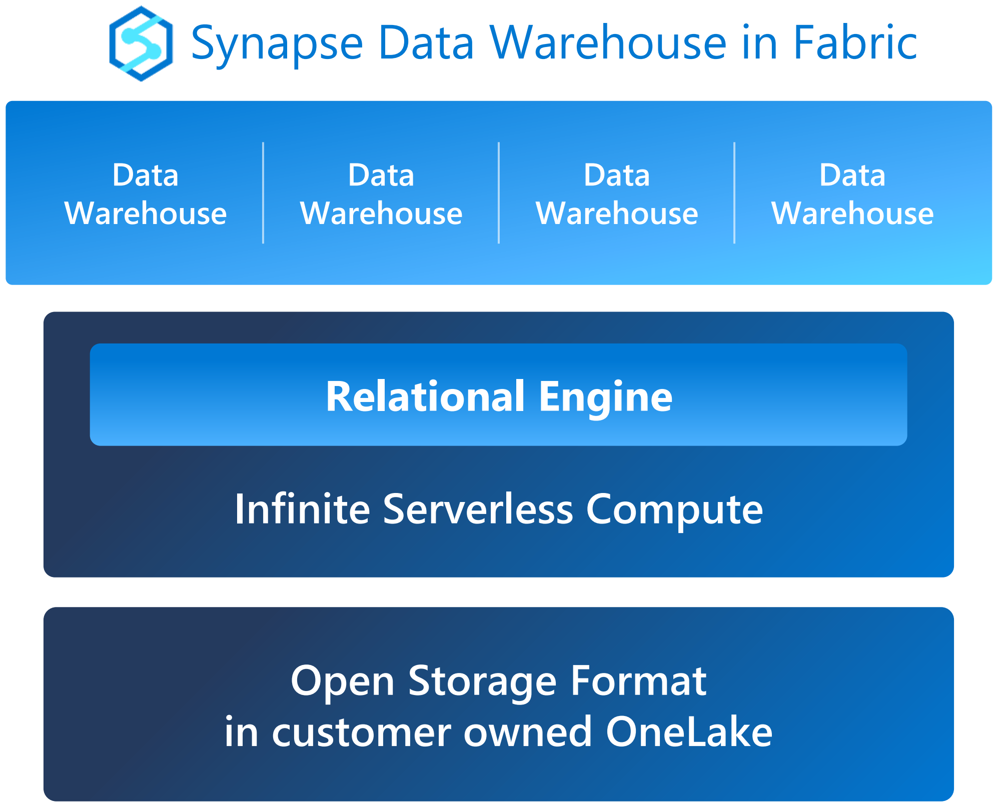
Andrea Benedetti
Sr Cloud Architect, Microsoft

/in/abenedetti @anBenedetti https://github.com/anbened

#FabricGarage | 008 | 2024.07.04

Infinitely scalable and open

- Compute engine infrastructure is serverless and provisioned on-demand
  - Physical compute resources assigned within milliseconds to jobs
  - Infinite scaling with dynamic resource allocation tailored to data volume and query complexity
  - Instant scaling up/down with no physical provisioning involved
  - Resource pooling providing significant efficiencies and pricing
- Data is stored in open Delta/Parquet format natively in OneLake


Synapse Data Warehouse in Fabric
Data Warehouse
Data Warehouse
Data Warehouse
Data Warehouse
Relational Engine
Infinite Serverless Compute
Open Storage Format in customer owned OneLake

Intelligent Data Distribution

- All Warehouse data is ingested using Round Robin distribution
  - Round-robin distribution is useful for improving loading speed
    - The assignment of rows to distributions is random
  - No need to specify distribution column(s)
  - No need to concern with data skew
    - Data skew means the data is not distributed evenly across the distributions
  - Source data is split into smaller data cells
  - Each new parquet file is assigned with a cell ID between 0 and 1023
    - As a result, tables have a fixed number of cells: 1024
    - As tables grow, we allocate cell IDs to new parquet files ensuring an evenly-distributed number of parquet files per cell ID

- Results:
  - Uniform work distribution during query processing
  - Scale-out ingestion operations with multiple nodes processing source data

Intelligent Data Distribution

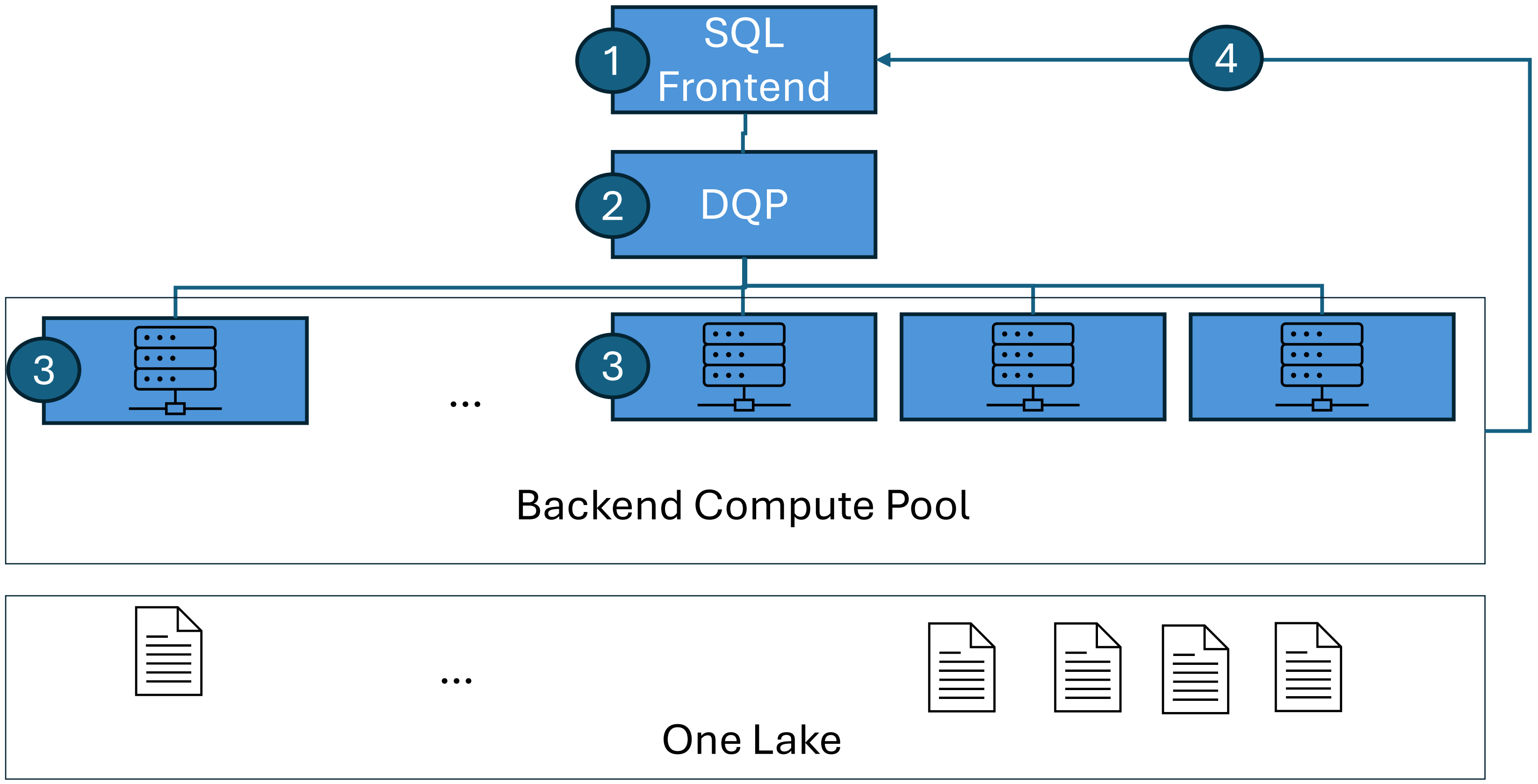
- At its core is a SQL MPP Engine with Delta Tables and TDS endpoints
  - MPP = massively parallel processing

- The first transactional data warehouse to natively support an open data format
  - Enable data engineers and business users to collaborate seamlessly
  - Data is stored in OneLake
  - Data is clearly separated from the compute used by the SQL engine → Compute and storage are decoupled
  - No need to copy the data from a Data Warehouse for other compute engines

- Built on:
  - SQL Server Query Optimizer
  - Distributed Query Processing engine

Data Processing Architecture

1. SQL frontend (FE) optimizes the query to find the best plan based on data size and complexity & passes it to the Distributed Query Processing (DQP) engine
2. DQP manages the distributed processing of the query by dividing it into smaller queries that run on backend computing nodes.
3. A task is a small query that runs in a distributed way. It reads file(s) from OneLake, combines data from other tasks, and groups or sorts the data from other tasks. For ingestion jobs, it also stores data in the right destination tables.
4. The SQL frontend gets the results from data processing and sends them back to the user or application.



Key Differences

- Elasticity and resiliency
  - The backend compute power benefits from a fast provisioning design. Normally, new nodes are acquired in a matter of seconds. As the need for resources increases, new workloads use the enhanced capacity. Scaling happens smoothly without interfering with query processing, ensuring continuous operations.
- Scheduling and resourcing
  - The distributed query processing scheduler works at a task level. Queries are shown to the scheduler as a directed acyclic graph (DAG) of tasks. A DAG enables parallelism and concurrency, since tasks that are independent of each other can run at the same time or in any order.
  - As queries come in, their tasks are scheduled based on first-come-first-served (FCFS) principles. If there is spare capacity, the scheduler might use a “best fit” method to maximize concurrency.
  - When the scheduler detects resourcing stress, it triggers a scale operation. Scaling is handled automatically, and backend topology expands as concurrency grows. When stress decreases, backend topology shrinks and frees up resource for the region.
- Ingestion isolation
  - Warehouses in Microsoft Fabric use a high-performance distributed query engine that lets customers run workloads with natural isolation.
  - Loading activities have **separate** resources from analytical workloads. This makes them faster and more reliable, as ingestion jobs can run on their own nodes that are ETL-optimized and do not interfere with other queries or applications. The system automatically adjusts and releases resources to offer the best performance with built-in scale and concurrency.

Warehouse Collation

- Collations provide the locale, code page, sort order, and character sensitivity rules for character-based data types
- Warehouse collation is set to **Latin1_General_100_BIN2_UTF8**
  - **Case-sensitive (CS) on data and metadata**: Queries must match data case as stored. Warehouse object names must match case in T-SQL statements.
  - **Accent-sensitive (AS)**: Binary collations (BIN2) sort and compare data based on Unicode code points. E.g.: It differentiates "A" from "Á".
  - **Support for multilingual data**: UTF-8 encoded data can be stored normally in char/varchar columns. No need for nchar/nvarchar.

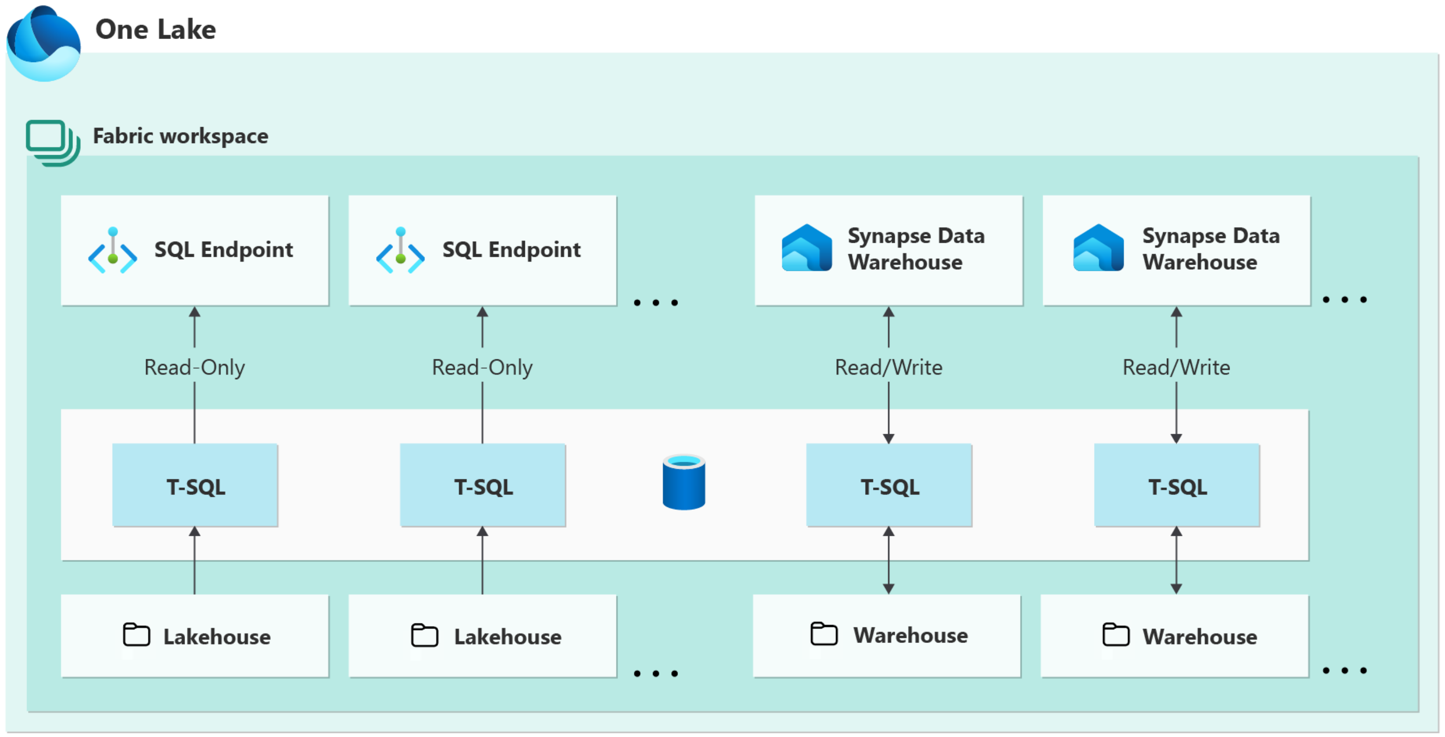
- These characteristics enable a consistent querying experience across Fabric workloads including Power BI, Lakehouses and Warehouses

V-Order Optimization

- V-Order is a write-time optimization specifically designed for the Parquet file format within the Microsoft Fabric ecosystem
  - Primary goal: enhance read performance under various compute engines, including Power BI, SQL, and Spark.
  - Key Features: Sorting, Row Group Distribution, Dictionary Encoding, Compression
  - Benefits: Lightning-Fast Reads, Performance Boost, Cost Efficiency
- Warehouse queries benefit from faster read times with v-order, still ensuring files are 100% compliant to Parquet's open-source specification
- All warehouse data is written with v-order optimization at ingestion time
  - This is **not** configurable
- V-Order sorting has a 15% impact on average write times but provides up to 50% more compression

Endpoints

- Difference between the SQL Analytics Endpoint and the Data Warehouse Endpoint



Andrea Benedetti
Sr Cloud Architect, Microsoft

/in/abenedetti @anBenedetti https://github.com/anbened

#FabricGarage | 008 | 2024.07.04