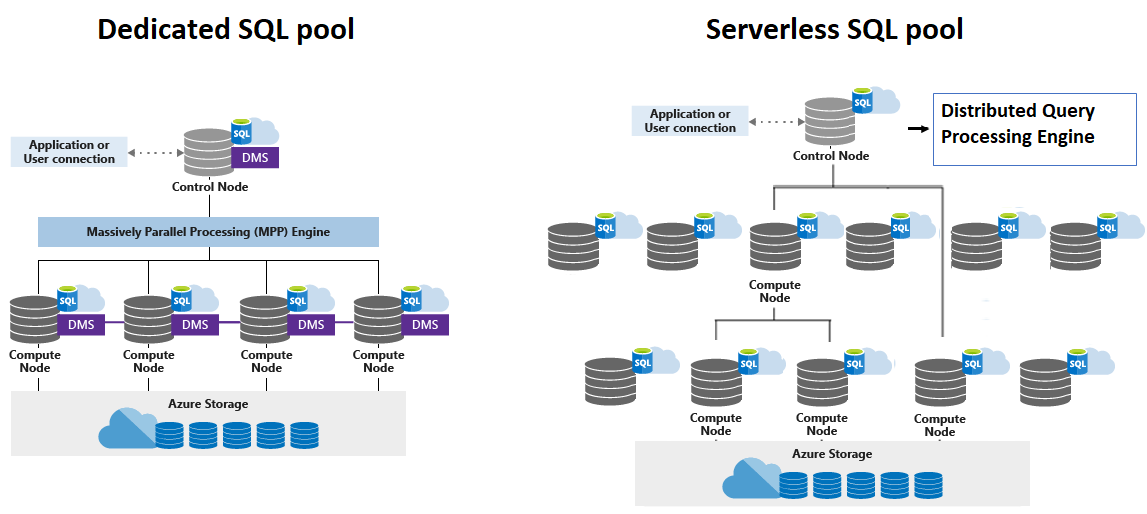
**Synapse SQL architecture components**

Synapse SQL uses a node-based architecture. Applications connect and issue T-SQL commands to a Control node, which is the single point of entry for Synapse SQL.

****

**1)Architecture components:**

* **Control node:** The Control node is the brain of the architecture. It is the front end that interacts with all applications and connections.
* **Compute nodes:** The Compute nodes provide the computational power.
* **Data Movement Service:** DMS is the data transport technology in dedicated SQL pool that coordinates data movement between the Compute nodes.
* **Distributions:** A distribution is the basic unit of storage and processing for parallel queries that run on distributed data in dedicated SQL pool.

**2) Processing:**

**2.1) Dedicated SQL pool:**

Use of Control Node: The distributed query engine runs on the Control node to optimize and coordinate parallel queries. When you submit a T-SQL query to dedicated SQL pool, the Control node transforms it into queries that run against each distribution in parallel.

Data Movement Service: DMS is the data transport technology in dedicated SQL pool that coordinates data movement between the Compute nodes. When data movement is required, DMS ensures the right data gets to the right location.

Use of Compute Node: In dedicated SQL pool, distributions map to Compute nodes for processing. The number of compute nodes ranges from 1 to 60, and is determined by the service level for the dedicated SQL pool. Each Compute node has a node ID that is visible in system views.

In Short: SQL queries enters via Control Node which is distributed in form of small distributions over Compute Nodes(The movement takes place by the help of DMS). DMS finally collects & combines processed data from all the Compute Nodes and stores it as Output in Azure Storage.

**2.2) Serverless SQL pool:**

Use of Control Node: In serverless SQL pool, the DQP engine runs on Control node for distributed execution of user query by splitting it into smaller queries that will be executed on Compute nodes. It also assigns sets of files to be processed by each node. Each small query is called task and represents distributed execution unit.

Use of Compute Node: In serverless SQL pool, each Compute node is assigned task and set of files to execute task on. Task is distributed query execution unit, which is actually part of query user submitted. Automatic scaling is in effect to make sure enough Compute nodes are utilized to execute user query.

In Short: SQL queries enters via Control Node which is distributed in form of smaller queries/task over Compute Nodes(The movement takes place by the help of DQP). These small queries utilize Azure Storage to process the data.

**3)Azure Storage:**

The output data from dedicated & serverless SQL pool is stored in Azure Storage.

* Serverless SQL pool allows you to query your data lake files.
* Dedicated SQL pool allows you to query and ingest data from your data lake files.

For **serverless** SQL pool scaling is done automatically.

When data is ingested into **dedicated** SQL pool, the data is sharded into distributions to optimize the performance of the system. We can choose following sharding pattern to use to distribute the data when we define the table.

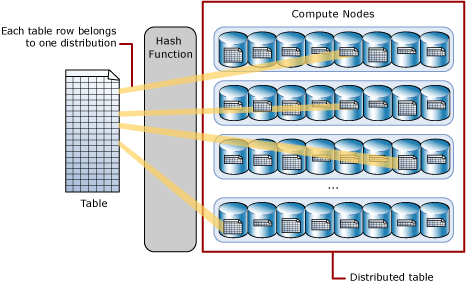
* Hash
* Round Robin
* Replicate

**Hash-distributed tables**

Benefit: A hash distributed table can deliver the highest query performance for joins and aggregations on large tables.

Processing:

* SQL pool uses a hash function to assign each row to one distribution.
* In the table definition, one of the columns is designated as the distribution column.
* The hash function uses the values in the distribution column to assign each row to a distribution.



**Round-robin distributed tables:**

Benefit: A round-robin table is the simplest table to create and delivers fast performance when used as a staging table for loads.

Process:

* A round-robin distributed table distributes data evenly across the table but without any further optimization.
* A distribution is first chosen at random and then buffers of rows are assigned to distributions sequentially.

Comparison to Hash-distributed tables:

* It is quick to load data into a round-robin table, but query performance can often be better with hash distributed tables.
* Joins on round-robin tables require reshuffling data, which takes additional time.

**Replicated tables**

Benefit: A replicated table provides the fastest query performance for small tables.

Concept:

* A table that is replicated caches a full copy of the table on each compute node.
* So, replicating a table removes the need to transfer data among compute nodes before a join or aggregation.

Con: Extra storage is required and there is additional overhead that is incurred when writing data, which make large tables impractical.

