**Snowflake**

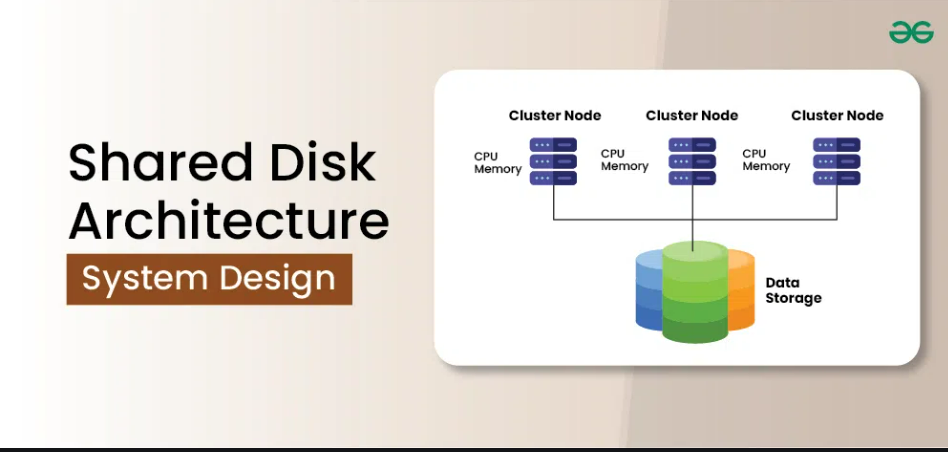
**Pricing in snowflake.**

1. Storage cost: Storage cost is measured as the average amount of data stored in Snowflake monthly.
2. Compute costs are incurred based on the size and duration of virtual warehouses you use for executing queries, data transformations, or running workloads.
3. Cloud services manage end-to-end solution of the user's task. It automatically assigns resources based on requirements of a task. Snowflake provides free usage of cloud service up to 10% of daily compute credits

**Architectures:**

1. **Shared disk architecture:**

distributed computing architecture in which the nodes share same disk devices, but each node has its own private memory and CPU.



**Advantages**:

1. **Ease of Management:**Centralization of storage
2. **Consistency**

**Disadvantages:**

1. **Performance Bottlenecks:** Nodes must wait, because of synchronization
2. **Scalability is limited**

1. **Shared nothing architecture:**

In this each node has its own mass storage as well as main memory. The [processor](https://www.tutorialspoint.com/embedded_systems/es_processors.htm) at one node may communicate with another processor at another node by a high speed interconnection network.

A diagram of a computer server

Description automatically generated

**Advantages:**

1. Scalability: Nodes operate independently, so you can add more nodes to handle increased load without disrupting the system.
2. Performance: Nodes can process their own data without interference, which leads to faster processing times.
3. Fault tolerance: If one node fails, it doesn't affect the functionality of others

**Disadvantages:**

1. Data is distributed across the cluster requires shuffling data between nodes
2. Performance is heavily dependent on how data is distributed across the nodes in the system
3. Compute can’t be sized independently of storage.
4. Snowflake’s architecture is a hybrid of traditional shared-disk and shared-nothing database architectures. Like shared-disk architectures, Snowflake uses a central data repository for persisted data that is accessible from all compute nodes in the platform. But like shared-nothing architectures, Snowflake processes queries using MPP (massively parallel processing) compute clusters where each node in the cluster stores a portion of the entire data set locally.

A blue and white diagram with words

Description automatically generated with medium confidence

1. **Service Layer:**  Which accepts SQL requests from users, coordinates queries, managing transactions and results.  Logically, this can be assumed to hold the *result cache*– a cached copy of the results of every query executed.
2. **Compute Layer:**  Which does the heavy lifting.  This is where the actual SQL is executed across the nodes of a Virtual Data Warehouse.  This layer holds a cache of data queried and is often referred to as Local Disk I/O although this is implemented using SSD storage.  All data in the compute layer is temporary, and only held if the virtual warehouse is active.
3. **Storage Layer:**  Which provides long term storage of results.  This is often referred to as *Remote Disk* and is currently implemented on either Amazon S3 or Microsoft Blob storage.

**Snowflake Cache Layers**

1. **Result Cache:**Which holds the results of every query executed in the past 24 hours. Query Cache is ideal for quick results of repeated queries. [Service layer]
2. **Local Disk Cache:**Queried data is stored onto the virtual warehouse’s SSD.When the query is not identical, result cache cannot retrieve it. Data in SSD cache is stored as long as warehouse is running(until suspended). [Computer layer]
3. **Remote disk:** Actual data on disk [Storage layer]

**Snowflake Performance Summary**

The sequence of tests was designed purely to illustrate the effect of data caching on Snowflake. The tests included:-

* Raw Data:  Including over 1.5 billion rows of TPC generated data, a total of over 60Gb of raw data
* Initial Query:  Took 20 seconds to complete and ran entirely from the remote disk.  Quite impressive.
* Second Query:  Was 16 times faster at 1.2 seconds and used the *Local Disk*(SSD) cache.
* Result Set Query:  Returned results in 130 milliseconds from the result cache (intentionally disabled on the prior query).

Important points to remember:

1. You always need a virtual warehouse to execute queries.
2. Always use limit clause with select \* from queries.
3. During dev activity, always keep auto suspend high
4. Share virtual warehouse when group of users are working on the common tables.
5. Never disable your cloud service layer result cache.
6. Reusing query result in snowflake is free.

**Snowflake Clustering**

**What happens when we submit the query to snowflake?**

1. **Query parsing:**
2. **Query Submission**: When you submit an SQL query to Snowflake, it is first sent to the Query Parser. This is where Snowflake checks the syntax and structure of the query to ensure its valid SQL
3. **Semantic Analysis**: The system also checks for semantic errors—whether the objects referenced (tables, columns, views, etc.) actually exist, whether the user has permission to access them, and if the query is logically valid
4. **Optimization:**
5. **Query optimization:** After parsing, the query is sent to the Query Optimizer. The optimizer looks at the parsed query and determines the most efficient way to execute it
6. **Query Plan Generation**: The optimizer generates an execution plan that outlines how the query will be run. This includes how data will be accessed, how joins will be performed, and which operations will be done in parallel
7. **Execution:**
8. **Virtual Warehouse Allocation:** Snowflake operates on a "multi-cluster" architecture, so at this point, a virtual warehouse (a compute resource cluster) is allocated to process the query
9. **Distributed Execution:** The query execution is distributed across one or more nodes (virtual machines) in the virtual warehouse. This allows for parallelized query processing, especially for large datasets, and ensures high performance for complex queries.
10. **Data Access:**
11. **Storage Layer Interaction:** Snowflake’s storage is separate from its compute resources, so during query execution, the data needs to be read from the centralized storage layer
12. **Micro-partitions**: Snowflake stores data in **micro-partitions**, which are immutable, compressed data blocks. The system reads only the necessary micro-partitions required for the query, optimizing storage access
13. **Result Calculation**: Data is fetched from storage, processed, and aggregated according to the query logic. This can involve filtering, joining, grouping, and applying any transformations requested in the query.
14. **Caching:**
15. **Result Caching**: Snowflake has a **result cache**. If the same query (with identical parameters) has been executed recently and the underlying data hasn't changed, Snowflake can return the cached results instead of re-running the entire query. This improves performance and reduces cost.
16. **Query Caching**: Snowflake also caches intermediate results of queries. For instance, if a query requires multiple steps (e.g., filtering and aggregation), results of intermediate operations might be cached.
17. **Returning Results**:
18. Once the query has been processed, the **virtual warehouse** sends the results back to the user or application that submitted the query.

**What are Micro-partitions?**

All data in Snowflake tables is automatically divided into micro-partitions, which are contiguous units of storage. Each micro-partition contains between 50 MB and 500 MB of uncompressed data.

Groups of rows in tables are mapped into individual micro-partitions, organized in a columnar fashion.

Snowflake stores metadata about all rows stored in a micro-partition, including:

* The range of values for each of the columns in the micro-partition.
* The number of distinct values.
* Additional properties used for both optimization and efficient query processing

**Virtual warehouse**

A virtual warehouse in Snowflake is a set of **compute resources** (CPU, memory, and disk) that allows you to run SQL queries and other operations. It is completely separate from the storage layer, which holds your data.

**Loading data**

**What is Staging in snowflake?**

In **Snowflake**, the **staging area** refers to a temporary storage space where data is stored before it is loaded into the final, production-ready tables of your data warehouse.

1. Internal staging [snowflake]
2. External staging [ **Amazon S3**, **Azure Blob Storage**, or **Google Cloud Storage]**

**Internal staging:**

1. **Creating internal stage**

Create or replace stage mydb.schema\_name.stage\_name

**External staging:**

1. **Creating external stage**

Create or replace stage mydb.schema\_name.stage\_name url = ‘s3://…’

credentials=(aws\_key\_id=’’ aws secret\_key=’’);

**Desc command:**

–To know the object properties

Syntax: Desc stage\_name | Desc file format

**File format:**

1. **Creating file format:**

**Create or replace file format db.schema.file-format-name**

**Type = csv**

**Field\_delimiter = ‘,’**

**Skip\_header = 1**

**Null\_if = (‘NULL’,’null’)**

**Empty\_field\_as\_null = true**

**Compression = gzip;**

**Copy command: Used to copy data from staging to actual storage**

**File format:**

**COPY INTO <table\_name>**

**FROM <stage\_or\_path>**

**FILE\_FORMAT = (TYPE = ‘<file\_format>’ [additional options])**

**[ON\_ERROR = CONTINUE | SKIP\_FILE | SKIP\_FILE\_<n> | ABORT\_STATEMENT]**

**[OTHER\_OPTIONS];**

**How to upload from local to snowflake tables?**

Method 1: Upload directly from snowflake UI

Method 2: Install Snow SQL on local machine

* 1. Connect to snowflake from CLI

snowsql -a ngntzmb-vq11634

* 1. PUT command, to upload from local to stage [In this example table stage]

PUT file://test.csv @test.college.%demo;

* 1. Move from stage to snowflake table on web UI SQL editor

copy into test.college.demo

from @test.college.%demo

file\_format=(type=csv field\_optionally\_enclosed\_by='"' field\_delimiter=',' skip\_header = 1)

pattern = '.\*test.csv.gz'

on\_error = 'skip\_file';

**How to download data from snowflake to local?**

Method 1: Upload download from snowflake UI (Max 100 mb limit)

Method 2: Copy data to a stage

1. Copy from snowflake table to table stage

Copy into mydb.schema.%tbale

From mydb.schema.table

File\_format=(type=csv)

1. Download from stage using get command

Get \_\_stage\_area\_path\_\_ file:///destination\_path

**How to upload data from local to snowflake using named stage?**

1. Create a stage
2. Upload files to stage using put command

PUT <file:///source_path> @db.schema.stage\_name

External staging:

1. Upload data to s3
2. Setup policy and role

Policy:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:GetObject",

"s3:GetObjectVersion",

"s3:DeleteObject",

"s3:DeleteObjectVersion"

],

"Resource": "arn:aws:s3:::koushik-snow/\*"

},

{

"Effect": "Allow",

"Action": "s3:ListBucket",

"Resource": "arn:aws:s3:::koushik-snow",

"Condition": {

"StringLike": {

"s3:prefix": [

"\*"

]

}

}

}

]

}

Role:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::850995569012:user/1ssu0000-s"

},

"Action": "sts:AssumeRole",

"Condition": {

"StringEquals": {

"sts:ExternalId": "QI16362\_SFCRole=2\_r/o2R8dLuFaf7etpJYR939rcWm0="

}

}

}

]

}

1. Create a storage integration object on snowflake

create or replace storage integration s3\_int

type = external\_stage

storage\_provider = s3

enabled = true

storage\_aws\_role\_arn = 'arn:aws:iam::557690593255:role/snowflake-role'

storage\_allowed\_locations = ('s3://koushik-snow/emp/');

1. Create a stage:

create or replace stage test.college.s3\_data

storage\_integration = s3\_int

url='s3://koushik-snow/emp/'

file\_format = test.college.s3\_incoming;

1. Use copy into, select etc

**How to fetch failed records while copying from stage to table?**

**Select \* from table(validate(table\_name, job\_id=>’’));**

**Snowflake features:**

1. **Time travel:**
2. Access Historical Data: Query previous versions of tables or objects.
3. Restore Data:

* Recover tables dropped or truncated by mistake.
* Undo recent data modifications.

1. Auditing and Debugging: Review how data looked at a specific point in time to troubleshoot issues.

SELECT \* FROM my\_table BEFORE (OFFSET => 5 \* 60);

1. Retention period:
2. Standard Retention: Up to 1 day (24 hours) for all Snowflake accounts.
3. Extended Data Retention: Up to 90 days for Enterprise Edition and above.
4. The retention period begins from the time a data change occurs.
5. Clone

CLONE command creates a duplicate or snapshot of an existing table, schema, or database at a specific point in time. This cloned object is independent of the original object, meaning any changes made to the clone will not affect the original, and vice versa.

1. **Zero-Copy Cloning**:

Snowflake performs zero-copy cloning, which means the clone doesn’t initially take up additional storage space. The storage is only allocated when the clone or the original object is modified.

1. **Point-in-Time Snapshot**:

Cloning captures the data as it is now the clone is created. The data in the clone is frozen, and any changes to the source after cloning are not reflected in the clone.

1. **Use of Metadata**:

Cloning works by duplicating the metadata and not the actual data, ensuring a fast creation of the clone.

1. Example:

CREATE TABLE cloned\_table\_name CLONE original\_table\_name;

1. SWAP:

SWAP refers to the SWAP operation between two tables. The SWAP command is a simple and efficient way to exchange the data or schema of two tables without requiring physical copies or data movements, ensuring high performance.

**Key Characteristics:**

1. **Zero-Copy**:

Snowflake’s SWAP operation is performed without any actual data movement. This makes the swap operation highly efficient, regardless of the size of the tables.

1. **Meta Data Exchange**:

It changes only the references to the tables involved. It doesn't actually copy or move data—only the metadata references are swapped.

1. **Changes to One Table Will Reflect in Another**:

After the swap, both tables will reflect the structure (and data) of the other table. Any changes made to old\_table after the swap will now affect new\_table and vice versa.

Example: Alter table table1 swap with table2;

1. Sample
2. FileSafe

**Permanent vs transient vs temporary tables.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Type** | **Retention period** | **File safe** |
| **Create table emp** | **Permanent** | **YES** | **YES** |
| **Create transient table** | **Transient** | **YES** | **NO** |
| **Create temporary table** | **Temporary** | **NO** | **NO** |

**4000 + 4000 = 8000**

**3500 + 3000 = 6500**

1. **Sampling:**

Sampling in Snowflake is used to extract a subset of rows from a table, either randomly or based on specific rules, to facilitate testing, debugging, or data exploration.