Moving your data to the Cloud

A Snowflake vs. Synapse comparison

Evaluation score total

Snowflake - **Synapse**

50.0 **33.5**

Summary

This white paper is directed at decision makers and data architects who are thinking about implementing a cloud data warehouse solution. It presents the result of a detailed comparison between two large products in this space, Snowflake and Microsoft Synapse (formerly known as Azure SQL Data Warehouse).

This comparison has been done both using on-paper evaluations and hands-on-keyboards physical testing of functions, providing a strong argument for why Snowflake is currently a better option in most cases. Below follows a grading matrix where each evaluation point is summarized for an overall score for each solution. **A higher score is a better result.**

Evaluation area	Snowflake	Synapse
Loading Efficiency – JSON, XML, CSV	4.5	2
ETL/ELT	3	4
Visualization	4.5	2
Governance & Maintenance	5	2
Scaling for concurrency	4	3
Scaling for performance	4	3
Cloning and snapshot restore	5	2
Cost for processing & storage	4	3
Support for semi-structured data	4	3
Active Directory integration	4	4.5
Support & services	4	3
Training required for SQL/DW team	4	2
Summary	**50.0**	**33.5**

Author

Therese Ahlstam

therese.ahlstam@topofminds.se

Therese is a senior specialist at Top of Minds AB. Therese is specialized in data warehousing with specific focus on Information Modeling and consolidating diverse business data. She has extensive experience in Manufacturing, Bank & Finance, Retail and Gaming. Therese is a driving force in the data community with a strong belief in real data solutions with use of Data Vault modeling and data integration and data to information value creation.

Top of Minds is a family of specialized companies that offers services mainly in the data and business intelligence area. We are premier partner in the Nordic countries on data warehouse, data analytics, cloud data development using an agile approach where we use Data Vault to increase the benefits of the projects, we participate in. We are a company with great focus on competence, our expertise and our clients' expertise and we are constantly working to spread our knowledge. We work as competence specialist at our customers site and we arrange on- and off-site training in Data Vault and Cloud Data development.

Contact us at:
www.topofminds.se
info@topofminds.se

Evaluation areas

Before starting the testing on either solution, a set of evaluation areas was established as listed below.

- Ingestion efficiency both in regards of technical performance and ease-of-use.
- Support for both structured and semi-structured data. The solution needs to be able to both ingest data and support parsing semi-structured data like JSON, XML and Avro without having to unpack the structures into relational tables first, as well as be able to join semi-structured and structured sources together.
- Built-in ELT support; without having to use an external ETL/ELT tool, evaluate the options of transforming data via stored procedures and similar included in the solution.
- Visualization efficiency; how easy and performant is the solution with regards to visualization of data to users
- Governance & Maintenance options; what is the effort required to maintain existing data structures and create new ones, and how well does the solution support auditing and billing control.
- Scaling for both performance and concurrency. Evaluate the effort involved and the performance increase when scaling is done to support both heavier queries and multiple concurrent users.
- Cloning and environment spin-up; evaluate how easy it is to spin up development and testing environments from existing data and what options exist for snapshot or cloning as a backup solution
- Overall cost of processing and storage
- Integration with Azure Active Directory. This specific AD was chosen as an evaluation point as it was already the preferred solution for cloud architecture where the evaluation was made.
- Support & Services; evaluate the quality of general support and the services available for a customer.
- Training; evaluate the time and effort needed to train developers in the new solution, given that these developers already are very proficient in traditional on-premise SQL databases.
- Security features included in the tool, specifically with regards to role and attribute-level access to business users and options for restricting sensitive data.

Score per area

Each evaluation area was scored with 0 to 5 points dependent on its rating. Five is the highest and best score and zero is lowest and worst result in the area.

Test Cases

Based on the evaluation areas established in the previous section a set of test cases were defined, designed to give sufficient answers to these questions. Due to the somewhat differing nature of the solutions to begin with and due to discoveries made during the testing phase the test cases were modified as work progressed but for the most part, we were able to execute them unchanged. To ensure a comparative evaluation was made we used the exact same set of test data for both solutions. Data sets where stored as CSV, JSON and XML files in an external data lake, and both solutions ingested data from this data lake directly.

1. Data Loading
   Load data from Azure data lake Gen 2 in CSV, JSON and XML format. Measure performance and note effort involved. Use the smallest possible compute configuration to get an idea of what performance is with minimal cost.
2. ELT processing
   Write two separate stored procedures; one doing standard insert/update of new data into a persistent table, and one creating a UUID function for sql externalization for more complex functionality evaluation. Evaluate the effort involved in creating these procedures. Execute them using an external CLI such as PowerShell or SnowSQL to verify that ELT processing can be scheduled and monitored with an external tool.
3. Scaling for performance
   Run a purposefully heavy query that takes a long time (>15 min) to execute on the minimum processing configuration and note execution time. Scale the computing resources x4, i.e. two 'steps' and run the query again. Note the performance gained vs the cost increase.
4. Data Visualization and concurrency testing
   Run a set of standard reporting queries including groups, sorts and aggregates on a large data set using Power BI, OBIEE and Python. Ensure all tools use Direct Query method and thus pushing execution down to the database rather than doing calculations in reporting cache. Using Power BI, also run several different queries at once to simulate multiple concurrent users to verify concurrent scaling options.
5. Separation of ingestion and consumption / support for multiple workloads
   Execute business user queries from reporting tools at the same time as ingestion of new data is taking place to verify if there is any impact on performance on either side.
6. Semi-structured support
   Execute queries that join structured data in relational tables with semi-structured, unpacked data. The queries should select specific columns (no select *) from both sets, join data on specific columns and use sum() or similar group aggregates from both sets.
7. Cost-effective maintenance
   Evaluate the efforts involved with the following operations: Adding partitions to a table, adding indexes, creating new tables and schemas and removing tables and schemas.
8. Azure Active Directory synchronization
   Set up SAML against AAD, and if possible also SCIM to enable users and roles to be defined in AAD and propagated to the solution. Evaluate the effort involved and the ongoing work needed to maintain this over time.
9. Support and Services
   Perform an evaluation of the standard support available for this solution. Include the average response time, need to escalate support queries and overall helpfulness and knowledge of the support services. Also verify if there are any additional costs to receive support for the solution.

Conclusion

The overall execution of the test cases in Snowflake have been successful, and in some cases exceeded our expectation. No single test case has failed. In Synapse we were also successful in most cases, but we required some additional aid from Microsoft to understand the workings of Synapse table structures and also had to exclude some tests as some features like XML support is not yet available in Synapse. In fact, a big part of the Synapse solution is still in preview; the Synapse Studio. JSON ingestion also requires Azure Data Factory for load of files larger than 4K, but both solutions have comparable native support once loaded. Compressed formats like Avro and Parquet are also supported by both.

Synapse has better support for column-level security with ability to grant only certain columns to users. Snowflake instead offers the secure views feature to hide table sources & logic from users.

Performance is similar in both solutions but only after applying proper tuning and index definitions on Synapse. Index rebuilds causes table to be offline while running and requires duplication of tables to achieve full availability for users. Snowflake requires no tuning or indexing to achieve same performance.

Cloning a table or database is instantaneous in Snowflake and causes no downtime and no extra storage costs until the clone is changed. In Synapse we have the option of using established backup/restore procedures to clone databases with expected run times and additional storage costs – however in both solutions storage costs make up a very small percentage of the total.

Capability of automatic scaling for concurrency is only possible in Snowflake but comes at increased run cost during surges (automatically spinning up more parallel compute resources and shutting them down when surge is over). In Synapse we allocate a certain % of compute resources to different user groups. This requires manual work when setting up and maintaining / adjusting (can be “automated” via scripts) over time but does not come at any additional processing cost during surges – but at the added cost of manual operation and monitoring instead.

The processing costs of the tests performed have been calculated as well as storage costs and in both cases the cost of Snowflake is around 40-50% of the cost of Synapse.

If you were to start using Snowflake now, you would have a cost-effective, scalable and flexible solution that can function with existing environments and be controlled through existing channels. Using Snowflake will require you to develop with a different approach than today to make full use of the effectiveness of the platform. The development process could be made very efficient and agile utilizing virtualization in a larger extent due to improved performance and necessary training can take place using a ‘Train the Trainers’ method, where the existing knowledge about Snowflake that a small number of people have is shared with the rest of the team. We do not immediately see a need for extensive courses and certifications for more than a minimum number of people.

Resulting evaluation score

Overall, Snowflake is both a more inexpensive, more efficient and more mature solution than Synapse today, which additionally also require less training before getting started and less risk of building something ‘wrong’ compared to Synapse.

Total score: Snowflake 50.0 - Synapse 33.5

Evaluation area	Snowflake	Synapse
Loading Efficiency – JSON, XML, CSV	4.5	2
ETL/ELT	3	4
Visualization	4.5	2
Governance & Maintenance	5	2
Scaling for concurrency	4	3
Scaling for performance	4	3
Cloning and snapshot restore	5	2
Cost for processing & storage	4	3
Support for semi-structured data	4	3
Active Directory integration	4	4.5
Support & services	4	3
Training required for SQL/DW team	4	2
Summary	**50.0**	**33.5**

Execution of Test Cases

Below is a brief description and analysis of the execution of each test scenario starting with all Snowflake cases and then all Synapse cases.

Execution on Synapse

Cost effective processing
Calculating processing cost is a little harder on Synapse compared to Snowflake. The compute warehouse in Synapse does not shut down or resume automatically when not in use, so unless it is manually shut down the spend is constant as per the scaling settings. We have not been shutting down / starting up warehouses in this comparison, as in the real world it would be difficult for a single user to determine if the warehouse can be shut down or if there are still others using it.
It is worth noting that that Synapse bills current sizing in hourly increments, so 1 hour + 1 minute of use equals 2 hrs of billing even if the warehouse was shut down after.

Data Loading
Immediately when testing started it became apparent that Synapse is not yet a finished product, and test cases needed to be modified or excluded since there are features that are still in beta preview or only part of the future road map. For instance, support for XML files was not available when this comparison was conducted. It was also not possible to ingest JSON files larger than 4kb without resorting to an external tool in the Azure tool set, Data Factory, which incurs quite significant extra costs.
Structured data in CSV format was ingested from the data lake into relational table structures without problems. It is however worth noting that the relational tables containing large amounts of data needed to be re-created with different settings to be performant, something we required Microsoft help to do.

ETL/ELT
Since Synapse is at its core a SQL Database it has full support for T-SQL procedural language, so all ELT procedures were easy to create and executed well with no extra need for training. However, we were advised by Microsoft to not attempt any virtualization i.e. creating views for facts and dimensions, something that can significantly lower development time. Instead we created persistent fact and dimension tables, requiring proper unpacking of any semi-structured data and loading to those tables before data was made available for visualization.

Scaling
Scaling for both concurrent users and heavier queries is not an automated process in Synapse. When it comes to concurrency scaling is made using resource groups, which are always allocated a certain percentage of the total compute capacity. AD users and groups are allocated into these resource groups and are then allowed only to use what is assigned to that group, regardless of the total available compute capacity at each given time. Additional Azure programmed processes can be used to monitor and scale compute/costs dynamically via T-SQL calls or other means. The resource allocations can be dynamically varied via scripts/sql calls, but these need to be created and maintained regularly to allow for optimized use of Synapse with regards to performance / cost balance. This means that while Synapse has a more cumbersome and manual resource allocation model, the cost for computing is a flatter rate – in Snowflake the concurrency scaling can be set to automatic which is easier but would incur extra costs during peak use times.

Separation of ingestion and consumption / support for multiple workloads
This resource allocation method is also the method used to separate ingestion and consumption loads – since each process is only ever allowed to use the percentage allocated, no ELT/ETL process will ever hinder an ongoing report query. However, you are billed for the full amount the compute warehouse is set to regardless of how many percent you use, and there is no way to increase the percentage allocated ad-hoc.
There are scenarios where ETL/ELT may have severe impact on visualization; if indexes or tables need to be rebuilt for performance the data set is unavailable until the operation is complete.

Visualization and reporting
Our initial visualization tests were unsuccessful, with queries taking between 5 and 9 minutes to complete from either Power BI or OBIEE (and in some cases never completing at all, like when Power BI wants to cache all data) – this when selecting data from a wide table containing 4.4 billion rows. A tuning session with Microsoft showed us that the table was initially created with the wrong parameters, and we were forced to recreate it using different key words and settings, a process that took about 10 hours to complete. After recreation and repopulation of the table performance was greatly improved and comparable to Snowflake.
We can conclude that with Synapse and proper knowledge and training on when to use the different types of table structures and keywords we could provide access to “raw” staged data with no performance problems even on the smallest processing configuration, compared to an on-premise database.

Support for semi-structured data
We have been able to create views and ad-hoc queries that parse our JSON data and makes it available for joining with structured relational data.
We have not been able to do the same on XML data since it is currently not supported by Synapse (but is in road map).
In the JSON case we can conclude that the built-in functionality that supports semi-structured data works well and is easily understandable; the OPENJSON, JSON_QUERY and JSON_VALUE functionality let us for instance take the metadata element in our JSON files and flatten it out to be included into every row of the data element in the same file – allowing us to easily create the required unique key for each rows (one field from the metadata and one field from the data) directly in a view without having to do any ETL or ELT, or having to unpack the data into relational structures first.

Cost-effective maintenance
When it comes to maintenance and what is traditionally known as ‘DBA’-work, Synapse does not differ much from a traditional on-premise database. The same work required to create indexes, partitioning strategies and rebuilding indexes and tables in a SQL Database is required in Synapse – but there is not much to gain from having a previous knowledge about SQL DB since Synapse introduces a whole new legacy of storage and indexing keywords to control how tables and indexes operate. These need to be learned and understood in full since we have proven it is quite easy to make mistakes and create subpar tables if you only go by online documentation and similar sources. As such, there is a fair amount of training and re-training required for Synapse to function properly.
Since Synapse does operate in the cloud (specifically the Azure provider only) there is no longer any need to allocate storage to specific tables or schemas.
In conclusion, the management and maintenance that a DBA would traditionally do is somewhat reduced when using Synapse, but most of the work surrounding performance, indexing etc. will remain. Tables and schemas still must be created and modified as necessary, but no storage

allocation or monitoring is needed. Performance tuning is still something we need to do in the traditional manner (with some retraining for new features) as well as having the option of temporarily or permanently scaling up a processing warehouse to handle the heavier query. This applies also in terms of snapshot and clone creation – it is the same process in Synapse as in an on-premise SQL database, essentially doing a restore of a previously made backup into a different table. The concept of zero-copy cloning as it exists in Snowflake does not exist in Synapse. 'Cloning' a large table in this manner took almost 12 hours.

Azure Active Directory integration
We were able to not only get SSO running but also SCIM provisioning, allowing the AAD to create roles and users in Synapse to match those created in the AD. Initially we had some trouble using our AAD users to sign on using any other tool than Sql Server Management Studio, but after some reinstalling of tools and clearing of saved settings we were able to log in using Azure AD from both Power BI and OBIEE.
With SCIM provisioning, the only administration required directly in Synapse would be to do one-time grants on the relevant database objects and processing warehouse to the roles created by the AD as they are imported.
We have tested that login using the AAD works and that SCIM is functioning as it should. Active directory integration works very well including provisioning.

Support Services
While we had excellent support from Microsoft architecture team while comparison testing was done, actual support tickets in Microsoft tend to have to be escalated once or twice, causing additional wait times, before you get to the level of knowledge required to solve the issues. There is an expectation this would be even more prevalent using Synapse since it is a very new solution and even the expert engineers, we had direct access to sometimes had trouble answering questions and had to do some extra investigation.

Execution on Snowflake

Cost effective processing
The compute warehouse in Snowflake can be set to shut down and resume automatically when not in use which makes for a very cost-efficient environment. Billing occurs per second after the first minute of use, making Snowflake very well suited for having several separate warehouses set up per (for instance) department within a company, and having usage costs allocated based on actual use and limiting costly idle time to virtually nothing.

Data Loading
JSON, XML and CSV files were ingested into Snowflake with no initial learning curve and no problems at any time. JSON and XML files were ingested into VARIANT-type tables meaning no unpacking or parsing of the structures were needed during ingestion. CSV files were ingested into standard relational table structures. The ingestion of XML files was not performant, but investigation showed that this is not a Snowflake-specific problem, rather the XML file organisation in the data lake was sub-par, with many millions of very small XML-files. It is not the ingestion of the files that is the problem, but rather the parsing of all the file names in the data lake that takes a long time. Snowflake recommends having a pre-step where several files are combined into larger files before ingesting which would solve the parsing problem. This is something that all xml ingestion engines would have problems with.

In addition to batch ingestion of files we also ingested files using Snowpipe, simulating a streaming data environment. Again, no issues were encountered, and it only took an hour to understand the process, set up the data stream and start loading data from our data lake.

ETL/ELT
Writing the stored procedures was easy for someone already familiar with ANSI SQL and general stored procedure language in relational databases and they functioned flawlessly and with good performance even when operating on large and growing data sets.

We ran these procedures both during batch ingestion and during Snowpipe-ingested streaming data, and there was no performance difference between the two sets of data.

We did not create a persisted star schema mart for reporting tests; rather we created a virtualized mart consisting of views on top of a large (4.4 billion rows) flat fact table and a JSON-file using flatten() functionality in Snowflake. This mart was created in an hour and performed just as well if not better than many existing on-premise data marts – the development time can be compared to the time it takes today to build ETL workflows and persistent tables for mart needs.
Having virtualized marts also mean maintenance and further development can be done much faster and in a more agile fashion.
Virtualized marts also provide zero downtime for business users during changes – a very useful feature for many companies.

Scaling
During Power BI testing we purposefully ran a query selecting all fields from a broad table containing 4,4 billion rows. With an XS warehouse, the smallest configuration, this query took 50 minutes to parse all rows and return data to Power BI. We then scaled the warehouse to a Medium, two steps up, effectively doubling the capacity twice. The same query then took 11 minutes to run.

The scaling operation itself is very simple to perform and takes but milliseconds. The upscaling then takes effect on all querys issued after the upscaling; i.e. queries already running are not affected. The same is true for down-scaling. We can conclude that vertical scaling for managing heavy operations is truly linear and performs exactly as promised, and that the scaling process itself is both fast and easy.
It is worth noting that the cost to run a Medium warehouse for 11 minutes is almost equal to running a XS warehouse for 50 minutes.
Scaling for concurrency can be set to automatic in Snowflake, which effectively mean that whenever the number of queries exceed the capacity of the warehouse, a separate identical warehouse is spun up and handles the extra load. It is then shut down when the queries are processed. This means that concurrency scaling is an entirely hands-off process, but that extra costs will occur during peak use. It is entirely configurable if and how many extra warehouses are allowed to be spun up.

Separation of ingestion and consumption / support for multiple workloads
The use of several simultaneous compute warehouses in Snowflake allow for complete separation of ingestion and consumption, ensuring no process can disturb another. Additionally, in Snowflake there is no need for any classic performance maintenance like index creation and rebuilds or partition management – making data wholly available to users at all times and eliminating the need for 'maintenance windows.
We had at most three parallel warehouses running; one for streaming data into Snowflake using Snowpipe, one for ingesting data via batch process, and one for running reporting and visualization queries. All were operating against the same database and schema, and in some instances, they were operating on the same tables. There was no impact on any of these processed caused by other warehouses running queries – reporting queries were unhindered when there was batch and pipe ingestion running against the same database and schema, and vice versa.

Visualization and reporting
We have run tests using OBIEE, Power BI and Python. In all cases performance was as good as when executing stored procedures in Snowflake, and in many cases outperformed both our expectations and existing environments. A variety of these tests were also performed on the previously mentioned virtualized mart, with no discernible performance changes.
It is also worth noting that once a query against a specific table / set of tables has been run on a warehouse, the data in those tables will reside in the warehouse cache until the warehouse is suspended due to inactivity. This means all subsequent queries using those tables can use the in-memory warehouse cache with even better performance. One such example is from our Python tests, where the first query parsing 4.4 billion rows and returning a subset of them took 8 minutes in a XS warehouse. The second query, against the same table but using sum() functions and filtering on different columns, took only 30 seconds as it was able to fully utilize the warehouse cache and never had to access the storage part of Snowflake at all. This means it is not only the query result that is cached but the underlying table data, something that should prove very useful for self-service reporting and dashboards alike.
We can conclude that with Snowflake and a virtualized data mart pattern you could drastically reduce development time for marts and can also much easier and faster provide access to "raw" staged data with no performance problems even on the smallest processing configuration.

Support for semi-structured data
We have done several ad-hoc queries against both the XML and JSON data, and our virtualized star schema mart combines both relational/structured data as well as dimensions on top of unpacked JSON data in views.
In all cases we can conclude that the built-in functionality that supports semi-structured data works well and is easily understandable; the lateral flatten functionality let us for instance take the metadata element in our JSON files and flatten it out to be included into every row of the data element in the same file – allowing us to easily create the required unique key for each rows (one field from the metadata and one field from the data) directly in a view without having to do any ETL or ELT, or having to unpack the data into relational structures first.
We also see that using the semi-structured support it would be easy and fast to create views that make semi-structured data usable by the business as if the data was in relational tables – without having to actually unpack the information.

Cost-effective maintenance
The management and maintenance that a DBA would traditionally do is drastically reduced when using Snowflake. Tables and schemas still have to be created and modified as necessary, but no storage allocation or monitoring is needed, no statistics need to be gathered, no partition strategies maintained, and no performance tuning of individual queries or tables need to be done. There is no special syntax to learn and nothing to keep in mind when creating tables, no special keywords are needed. Performance tuning is simply a matter of temporarily or permanently scaling up a processing warehouse to handle the heavier query.
In addition, Snowflake supports zero-copy cloning, allowing us to clone individual tables, schemas and entire databases. In all cases the operation took less than two minutes, and we have also verified that changes to the cloned object does not cause any changes to the original and vice versa.
We have also cloned a database to check potential storage increase. Before clone, our total Storage Used for our account was 476,424 GB.
After the clone was made, the total Storage Used for the account was still 476,424 GB – proving that creating a clone does not duplicate the actual stored data, only creating new metadata pointers to it.
If we were to start ingesting new data into the clone and altering existing data, storage would naturally increase as new versions of the data is created, but we can conclude that creating dev/test environments as well as creating manual backups does not increase our storage costs and is both very simple and very fast to do, with no downtime or performance hindrance on the original source.

Azure Active Directory integration
With a little help from Snowflake support we were able to not only get SSO running but also SCIM provisioning, allowing the AAD to create roles and users in Snowflake to match those created in the AD. With SCIM provisioning, the only administration required directly in Snowflake would be to do one-time grants on the relevant database objects and processing warehouse to the roles created by the AD as they are imported.

We have tested that login using the AAD works and that SCIM is functioning as it should.
The one thing we have noticed is that while both the AAD role and user is created and assigned correctly in Snowflake via SCIM provisioning, the role is not set as default on the user. For Power BI to function properly, the correct role needs to be default as role switch is not possible.
It is however a very small and simple task to set the AD role as default on the AD user, but it requires that the AD user performs a log in to Snowflake in order to be created first. In talking to

Snowflake about this we were also informed that they are currently working together with Microsoft to create functionality whereby default roles may be assigned from AAD.
Aside from default role assignment, the Active directory integration works very well including provisioning.

Support Services
During our setup of SSO and SCIM provisioning with Azure Active Directory we had some trouble getting the SCIM provisioning (which allows users and roles to be set up in Azure and imported automatically into Snowflake, requiring no AD management within Snowflake) so we opened a support ticket with Snowflake.
The initial response took less than two hours and came directly from a Snowflake engineer, and within 48 hours we had a conference call set up where he was able to guide us through the process live and get it up and running. This level of support would be included if you were to use Snowflake, at no extra cost.
In addition to Snowflake Support we would also have access to the Snowflake Lodge community, where there is a plethora of use cases, example codes and previously asked and answered questions from Snowflake users around the world.