Easy Scaling: The Most Critical Consideration for Choosing Analytical Database Management Systems in the Cloud Era

Jie Liu[1] and Genyuan Du[2,3]
[1]*Computer Science Division, Western Oregon University, Monmouth, Oregon, U.S.A.*
[2]*School of Information Engineering, Xuchang University, Xuchang, Henan, China*
[3]*Henan International Joint Laboratory of Polarization Sensing and Intelligent Signal Processing, Xuchang, Henan, China*

Keywords: Analytical Database Management Systems (ADBMSs), Performance, Cost, Scaling.

Abstract: Analytical database management systems offer significant advantages for organizations practicing data-driven decision-making. ADBMSs rely on massively parallel processing for performance improvement, increased availability and other computation related resources, and improved scalability and stability. In this position paper, we argue that (1) Gustafson-Barsis' Law aligns well with use cases suitable for Cloud-based ADBMS, still, neither Amdahl's law nor Gustafson's law is sufficient in guiding us on answer the question "how many processors should we use to gain better performance economically", and (2) ADBMS's capability of utilizing parallel processing does not translate directly into easy scaling, specially scaling horizontally by adding more instances or nodes to distribute the workload at will, so when costs are somewhat controllable, allowing easy scaling should be by far the most critical consideration for choosing an ADBMS.

1 INTRODUCTION

The proliferation of big data and the increasing demand for real-time analysis has put a spotlight on the need for efficient analytical database management systems (ADBMSs). In the cloud era, these systems are expected to be able to handle large amounts of data, provide fast query performance, and be highly available and scalable to discover useful insights and distribute the finding to stakeholders.

Naturally, all ADBMSs rely on massively parallel processing for performance improvement, increased availability and other computation related resources, and improved scalability and stability (Chaiken 2008). Historically, we were using both Amdahl's law and Gustafson's law to guide us in decisions on whether we should utilize more processors to gain better performance for our parallel algorithms. Clearly, ADBMSs execute our database queries using some form of parallel algorithms; therefore, these two laws still apply.

Based on our studies on parallel processing, data engineering, and our experiences using several different ADBMSs, we would like to argue, in this position paper, (1) despite the fact that Gustafson-Barsis' Law aligns well with use cases suitable for Cloud-based ADBMS, neither Amdahl's law nor Gustafson's law is sufficient in guiding us on answer the question "how many processors should we use to gain better performance **cost-effectively**", and (2) ADBMS's capability of utilizing parallel processing does not translate directly into **easy** scaling, specially scaling horizontally by adding more instances or nodes to distribute the workload at will, so when costs are somewhat controllable, allowing easy scaling should be by far the most critical consideration for choosing an ADBMS. The second argument is our main argument.

In the next section, we will briefly discuss Amdahl's Law and Gustafson-Barsis' Law. We will then present our opinion that Gustafson-Barsis' Law is more suitable in guiding us with using more processors to realize performance gain for queries running on Cloud-based ADBMS However, neither Amdahl's law nor Gustafson's law is sufficient in guiding us on cost efficiency. In section 3, we will discuss scaling in a few well known ADBMSs, such as Snowflake, Redshift, Databricks, and BigQuery. We will also present our opinion that allowing easy scaling should be by far the most critical consideration for choosing an ADBMS because that really is the main reason organizations are using ADBMSs. Understanding cost control is important,

Liu, J. and Du, G.
Easy Scaling: The Most Critical Consideration for Choosing Analytical Database Management Systems in the Cloud Era.
DOI: 10.5220/0012207600003598
In *Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2023) - Volume 1: KDIR*, pages 373-378
ISBN: 978-989-758-671-2; ISSN: 2184-3228

we present our best practices in balancing the trade-offs between costs and performance. We conclude our paper in section 5.

2 AMDAHL'S LAW AND GUSTAFSON-BARSIS' LAW

Amdahl's Law (Amdahl, 1967), provides a theoretical framework for analysing the potential speedup of a parallel algorithm when a portion of the computation remains serial. It helps in understanding the fundamental limitations of parallelization and emphasizes the importance of optimizing the sequential portion of a program. Following Amdahl's law, let s be the fraction of operations in a computation that must be performed sequentially, where $0 \leq s \leq 1$. The maximum speedup ψ achievable by a parallel computer with p processors performing the computation is

$$\psi \leq \frac{1}{s+(1-s)/p} < \frac{1}{s} \quad (1)$$

However, Amdahl's law assumes a fixed problem size and workload. This is not really realistic in today's analytical related use cases, especially in the context of big data and cloud computing. For example, one of our clients has seen an ETL job of theirs grows by forty times as big, in terms of record count, in 3 years.

On the other hand, Gustafson-Barsis' law Gustafson, (Gustafson, 1988), proposed by John Gustafson and R. Eric Barsis in 1988, presents a contrasting perspective. It suggests that the scalability and performance of parallel algorithms can be improved by increasing the problem size as the number of processors increases. Following Gustafson-Barsis' law, given a parallel program solving a problem using p processors, let s' denote the fraction of the total execution time performed sequentially. The maximum speedup ψ achievable by this program is

$$\psi \leq p + (1-p) * s' \quad (2)$$

Gustafson-Barsis' law emphasizes that larger problem sizes can benefit greatly from increased parallelism and can effectively harness the additional computing resources to solve bigger problems within a reasonable amount of time.

2.1 Gustafson-Barsis' Law Aligns Well with Use Cases Suitable for Cloud-Based ADBMS

Considering the evolving landscape of big data and cloud computing, our recommendation would be to prioritize Gustafson-Barsis' Law over Amdahl's Law when making decisions regarding the allocation of computing resources. Here are a few reasons for this preference:

Scalability: Gustafson-Barsis' Law acknowledges the potential benefits of scaling up the problem size and workload, aligning well with the demands of processing large volumes of data in modern applications. By increasing the problem size, more processing units can be effectively utilized to keep the overall query execution time to be reasonable.

Real-World Scenarios: In practical scenarios involving big data processing and analytics, the size and complexity of the problems tend to grow significantly over time. Gustafson-Barsis' Law provides a more realistic and applicable approach by considering the benefits of scaling up the workload to fully utilize parallel resources.

Cloud Computing: Cloud-based ADBMSs often allow their users to scale resources based on workload demands or allocate more processors to their users tasks. Clearly, Gustafson-Barsis' Law aligns well with this approach of leveraging the elasticity of cloud platforms.

While Gustafson-Barsis' Law offers a more relevant framework for modern computing environments, it does not invalidate Amdahl's Law. Amdahl's Law still holds value in understanding the impact of sequential portions of an algorithm and optimizing performance within those constraints. Just that, in the context of requesting computing resources to manage overall query time, Gustafson-Barsis' Law provides a more suitable guiding principle.

2.2 Both Laws Ignore Costs

Neither Amdahl's law nor Gustafson's law is sufficient in guiding us to answer the question "how many processors should we use to gain better performance **economically,**" the key word here is this **economically.** Generally speaking, adding processors reduces overall execution time of the same exact query (let's disregard result caching for a moment).

However, some ADBMSs have a minimum charge policy. For example, Snowflake has a minimum charge policy of 1 minute, even if your query only takes 10 seconds. Other systems require their users to pay upfront, such as Redshift, regardless whether users are running queries or not. So, the pricing model requires their users to consider the economic implications of resource allocation and query execution time, which may not support using more processors to keep on reducing the overall execution time. That is, such a minimum cost structure introduces an economic constraint that needs to be considered when determining the optimal allocation of computing resources.

Gustafson-Barsis' Law focuses on maximizing the benefits of scaling up to improve performance. However, in the presence of a minimum cost, it becomes crucial to strike a balance between the desired performance gains and the associated costs. When deciding on the allocation of computing resources, it is essential to consider the cost implications at different scales. Scaling up resources may lead to better performance, but it also increases costs proportionally.

For some systems. underutilization of resources leads to unnecessary waste and drives up costs, while other system overutilization may result in diminishing returns in terms of the costs. Gustafson-Barsis' Law only tells us adding more processors to reduce overall execution time. Just, we have to keep in mind that adding processors results in additional costs. However, when problem size increases, not adding more processors may also incur unnecessary additional costs due to the impact of storage limitations on query execution time. This is primarily because smaller warehouses or smaller processor count have less storage capacity, which can lead to data spillage into secondary storage, which comprises lower-cost, higher-capacity, higher latency and slower read/write speeds storage options. Such a spillage can significantly degrade query performance, even crashing the entire ADBMS server.

When deciding on the allocation of computing resources, it is essential to consider the cost implications at different scales. We need to evaluate the cost-effectiveness of different resource allocation strategies. This evaluation should be performed on a case-by-case basis, making informed choices to maximize performance while minimizing costs. The bottom line is that neither Amdahl's law nor Gustafson's law is sufficient in guiding us to answer the question "how many processors should we use to gain better performance economically."

3 POPULAR ADBMSs

The authors have been exposed to ADBMSs for more than 10 years, started with Vertica, an ADBMS runs either on-premises or in an enterprise's own Virtual Private Cloud, that is, it does not deliver database functionalities as a service. ADBMSs are not designed to support OLTP operations. Instead, they provide super-fast response times for aggregated queries known as OLAP queries (Heavy.AI, 2023). ADBMSs are generally more scalable, distributed, columnar in data stores, and heavily compressing their data. In addition, because the data is distributed, they naturally utilize concurrency for performance improvement (Heavy.AI, 2023).

In this section, we briefly discuss three extremely popular cloud based ADBMSs: Google's BigQuery, Amazon's Redshift, and Snowflake, all are fully managed cloud service, which means their vendors handle the underlying infrastructure, such as hardware provisioning, software patching, and backup management. We select these three because we have either used them in the past or are still using them. Performance and cost related discussions and analysis are available (Fraser, 2022), so we will not repeat the same experiment. Instead, we will provide our position on selecting an ADBMS at the end of the section.

All the three ADBMSs discussed here, and most of the other systems on the market, leverage columnar storage and Massively Parallel Processing (MPP) to deliver high-performance analytics capabilities. This columnar storage approach offers advantages for analytical workloads as it enables efficient compression, improves query performance, and reduces I/O requirements. By storing columns together, ADBMSs can read and process only the necessary columns and utilize special algorithms that take advantage of compression type, which leads to faster query execution. ADBMSs utilize MPP architecture to distribute query processing across multiple computing nodes to be processed. This parallel processing capability allows ADBMSs to handle large datasets and complex analytical queries efficiently. Combining columnar storage and MPP, ADBMSs can achieve faster query performance, efficient data compression, and scalability to handle large volumes of data. These features make ADBMSs well-suited for analytical workloads that require complex queries, data aggregations, and ad-hoc analysis.

3.1 Snowflake

Snowflake is a cloud-native ADBMS known for its flexibility, scalability, and ease of use. It was one of the first decoupled storage and computing architectures, making it the first to have nearly unlimited compute scale and workload isolation, and horizontal user scalability. It separates storage and compute, allowing users to scale resources independently based on their needs. This architecture enables elastic scalability and efficient resource allocation, resulting in high performance for analytical workloads. Snowflake's strengths also include its automatic query optimization, support for structured and semi-structured data, and the ability to seamlessly integrate with various data processing and analytics tools. It provides a true SaaS and is designed to address the challenges of handling large-scale data analytics and offers a range of features to support data warehousing, querying, ETL, and analysis tasks. Snowflake operates on a shared-nothing but data and MPP architecture, where data is stored and processed across multiple compute clusters, enabling high-performance query execution for many users simultaneously and independently without interfering with each other's performance. Users access Snowflake databases using a browser based online UI.

When creating a virtual warehouse in Snowflake, users can specify the initial size of the warehouse, which determines the amount of compute resources allocated to the warehouse. The size options range from X-Small (1 credit per hour) to 6X-Large (512 credits per hour), a total of 10 levels, indicating the relative compute capacity of the warehouse. Our rough understanding is that each credit is equivalent to a computing node, which is defined by a CPU, plus a predefined amount of RAM and SSD. The compute capacity remains constant unless manually adjusted by the user. If the workload exceeds the capacity of the virtual warehouse, it can result in performance degradation or increased query times. However, users can select and start using a larger warehouse with just a few clicks and experience faster query execution. Depending on the type of services and security requirements, each credit is mapped into a fixed dollar amount. Data storage has separate charges. The best part is that users can easily select to increase the warehouse size to reduce the overall query execution time.

It is important to note that using smaller warehouses does not translate to saving cost because the query cost is processor count times execution time. A smaller warehouse generally takes longer time to execute the same query because computing nodes come with RAM and SSD. In the few extreme cases, if the data required to complete the query is too large to completely fit into the primary storage, users may notice a drastic increase in overall execution time.

3.2 Google's BigQuery

BigQuery is also a fully managed ADBMS provided by Google Cloud Platform. It is not a typical data warehouse in part because it started as an on-demand serverless query engine, which determines how many virtual CPUs a query requires. BigQuery is not really relational. Instead, it allows nested tables as an attribute. BigQuery's integration with other Google Cloud services, such as Cloud Storage and Cloud Machine Learning Engine, allows users to build end-to-end data pipelines and incorporate advanced analytics capabilities seamlessly. Like Snowflake, users access BigQuery databases using a browser based online UI.

BigQuery works either in a "flat-rate pricing model" where virtual CPUs are reserved in advance or in an "on-demand pricing model", where virtual CPU assignment is completely in the hands of BigQuery and the state of the shared resource pool. The reserved slots model allows users more control over compute resources and costs. BigQuery's "on-demand pricing model," which has been our experience using BigQuery, scales relatively well to process large data volumes. It automatically assigns more compute resources when needed behind the scenes. In this case, users are charged on a $/TB scanned basis. However, in this model, users are not able to scale up or down at all. We often find this being undesirable.

3.3 Amazon Redshift

Amazon Redshift, based on PostgreSQL, is "…a fully managed, petabyte-scale data warehouse service in the cloud" (Amazon, 2023). It is widely used by organizations for their data warehousing and analytics needs. Its combination of high performance, scalability, and seamless integration with the AWS ecosystem makes it a popular choice for handling large-scale data analytics workloads. Amazon Redshift Spectrum is an optional service that enables you to query all types of data stored in Amazon S3 buckets. The data in S3 does not need to be loaded into the Redshift data warehouse first to be able to be queried by Redshift if you have spectrum enabled.

We access Redshift using SQL Workbench or Dbeaver.

Pricing model of Redshift can be complicated. Amazon provides several pricing choices, including a serverless model similar to that of BigQuery. Still, our management selected Compute Node Pricing where our organization's cost is primarily based on the type and number of compute nodes provisioned for our Redshift cluster. In our situation, we have many different users with very different expectations. It is almost impossible to accommodate all requests. For example, with Redshift, users can select computing nodes that are emphasizing on compute or accentuating storage. However, we have users who need to run complex queries that do not process large volumes of data; but, we also have many users who run mostly straight-forward queries that process very large amounts of data. Also, scaling up and down can be done in minutes if we are doubling or halving the number of computing nodes. One time, it took the system several hours to scale down from 16 to 10. Suffers similar to this negatively affected our experience using Redshift. Another limitation on Redshift is introduced by its bundling of compute and storage services. As the data volume increases, users have to reserve more nodes to have a bigger storage capacity to accommodate the increase in data. All that said, we know plenty of teams that are happy with Redshift.

3.4 Keep It Simple and Straightforward

Security, Governance, ease of use etc. aside, most organizations pay attention to performance and cost. Fortunately, there are many reputable companies that have published their experimental results for us to refer and to study. The CEO of Fivetran has recently published their comprehensive report "Warehouse Benchmark: Redshift vs. Snowflake vs. BigQuery" (Fraser, 2022). The article compares the performance and cost of Amazon Redshift, Snowflake, and Google BigQuery in some benchmarking studies. An unofficial summary of the article is:

1. All three ADBMSs improved their systems in terms of performance. Still Snowflake enjoys the best performance, even the gap among the three is very insignificant.
2. The cost of a system can be difficult to compare, but the differences under "normal" condition and usage are generally insignificant. It is worth noting that for a small amount of data, Google's BigQuery can be free.
3. The chart that plots the cost and execution time for different systems with different computing capability (aligned mostly with overall costs) is very informative.

The key take-away based on our experiences and research is that most ADBMSs on the cloud are similar in performance, costs, support, security. The main difference is their business model. It is our opinion and our position that BigQuery's "we will take care of everything" is too simple and takes too much control away from the users; Redshift's "you have many options to choose from" and pay upfront is too complicated because it is just hard to know what will come to you; Snowflake's "keep the important thing, using computing node count as a way to balance between cost and performance, simple and straightforward" is the best approach.

That is, with all other characteristics roughly equal, allowing easy scaling should be by far the most critical consideration for choosing an ADBMS because that really is the main reason organizations are using ADBMSs.

4 TOO SMALL A COMPUTING NODE COUNT COSTS MORE

One of the counter intuitive views is to use a smaller number of computing nodes, which translate into smaller warehouse size in Snowflake, to save cost because it costs less. Actually, often, selecting too small a computing node count may cost more.

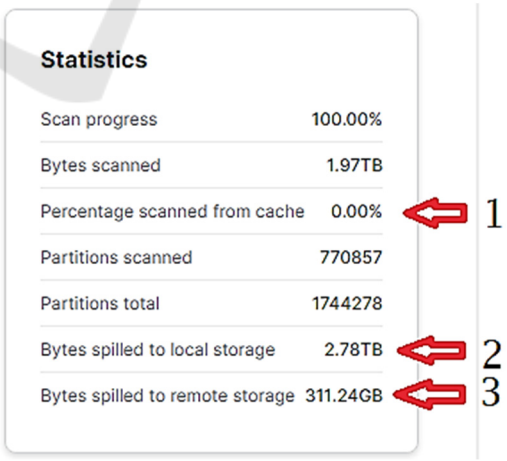

Figure 1: Statistics of executing an SQL statement on an extra small Snowflake warehouse.

Figure 1 shows statistics of in the middle of executing a costly SQL statement on an extra small

Snowflake warehouse the first time. Snowflake's extra small warehouse has only one computing node. The node's actual configuration is not public information. The details of the figure show that (1) since this is the first time the SQL statement is executed, none of the tables referred in the statement is in the cache, (2) a large amount of data is spilled into local storage because the warehouse's main storage is too small to hold the data needed to run the SQL statement, and (3) some data even get stilled into remote storage, which will drastically increase the overall execution time of the SQL statement.

The execution of the SQL statement, with its snapshot of execution shown in Figure 1 was eventually cancelled after more than five hours of execution. When using a 6xLarge warehouse, which is 512 times more powerful, the spilling of data disappeared and the query was done within three minutes.

Using less CPUs does not save money because the query cost is processor count times execution time. Less CPUs may translate into much longer overall query execution time because "CPUs" come with storage. However, finding the most cost optimal size of warehouse is not easy because there are a lot of factors that can affect that.

5 CONCLUSIONS

By studying three extremely popular ADBMSs (Google's BigQuery, Amazon's Redshift, and Snowflake), we show that (1) users may not always have the options of easily and cheaply adding more processing units to improve query performance, (2) when they do have the option, Gustafson-Barsis' Law provides a better guidance than Amdahl's Law does, and; however, we argue that neither Amdahl's law nor Gustafson's law is sufficient in guiding us on answer the question "how many processors should we use to gain better performance economically", and (3) with all other characteristics roughly equal, allowing easy scaling should be by far the most critical consideration for choosing an ADBMS because that really is the main reason organizations are using ADBMSs . Finally, using less CPUs does not reduce costs because query cost is calculated by processor count times execution time. Less CPU may take longer time to execute because "CPUs" come with storage.

ACKNOWLEDGEMENTS

We thank Western Oregon University's Faculty Development Committee and China's Henan provincial government projects 192102210275 and GH201944 for their financial support for this work.

REFERENCES

Amdahl, Gene (1967). Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, In AFIPS Conference Proceedings, Vol. 30 (Atlantic City, N.J., Apr. 18–20), AFIPS Press, Reston, Va., pp. 483–485.

Amazon. (2023). What is Amazon Redshift. https://docs.aws.amazon.com/redshift/latest/mgmt/welcome.html. Amazon

Chaiken, R., Jenkins, B., Larson, P., Ramsey, B., Shakib, D., Weaver, S., & Zhou, J. (2008). SCOPE: easy and efficient parallel processing of massive data sets. Proc. VLDB Endow., 1, 1265-1276.

Cook, H., Adrian, M., Greenwald, R., and Gu, X. (2022). Magic Quadrant for Cloud Database Management Systems. https://www.gartner.com/doc/reprints?id=1-2AIUY4M7&ct=220707&st=sb. Gartner

Du, Genyuan and Liu L (2021). On Cloud Analytical Database Management Systems Suitable for Data Intensive Biomedical Related Research. Am J Biomed Sci & Res. 2021 - 11(4).

Fraser, George (2022). Cloud Data Warehouse Benchmark https://www.fivetran.com/blog/warehouse-benchmark. Fivetran.

Al-hayanni, Mohammed A. Noaman; Xia, Fei; Rafiev, Ashur; Romanovsky, Alexander; Shafik, Rishad; Yakovlev, Alex (2020). Amdahl's law in the context of heterogeneous many‐core systems‐a survey. IET Computers & Digital Techniques. 14 (4): 133‐148.

Hill, Mark D.; Marty, Michael R. (2008). "Amdahl's Law in the Multicore Era". Computer. 41 (7): 33–38.

G2. (2023). https://www.g2.com/compare/amazon-redshift-vs-snowflake-vs-google-cloud-bigquery. G2

Google (2023). What is BigQuery. https://cloud.google.com/bigquery/docs/introduction. Google.

Liu, Jie (2017). Gustafson's law vs Amdahl's law. https://stackoverflow.com/questions/34910585/gustafsons-law-vs-amdahls-law

Gustafson, John L. (May 1988). "Reevaluating Amdahl's Law". Communications of the ACM. 31 (5): 532–3.

Heavy.AI (2023). Analytical Database. https://www.heavy.ai/technical-glossary/analytical-database. Heavy.AI