Examining the Enormous Oracle MySQL HeatWave TCO and TCO/Performance Advantage

by, Marc Staimer

2022

Introduction

The very rapid customer adoption of Oracle's unique MySQL HeatWave has been nothing less than extraordinary. The highly integrated cloud database service that unifies online transaction processing (OLTP), online analytical processing (OLAP), extensive use of machine learning (ML) automation, real-time elasticity, and ML modeling with best-in-class performance—based on standard TPC-H benchmarks published on GitHub—at a price much lower than its competition, has proven to be compelling. According to Oracle's publicly released information, customer uptake has grown exponentially. This has taken many Oracle competitors by surprise. It shouldn't have.

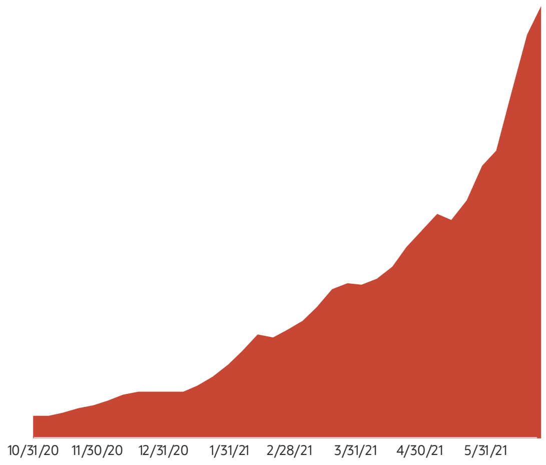


Chart 1: Oracle MySQL HeatWave Rapid Adoption Growth

This unique Oracle MySQL HeatWave multi-model, multi-workload converged cloud database service has translated into real-world exceptional cost/performance advantages in customers' production environments, making it very appealing indeed. Real-world customer engagement has revealed that to be a major understatement. MySQL HeatWave customers have reported 10 to 20x performance improvements over Amazon Web Services (AWS), Snowflake, Google Cloud Platform (GCP), and Microsoft Azure while their reported costs were reduced by as much as 83%.

The results for these customers are in line with the published TPC-H benchmarks Oracle made available on GitHub. Those benchmarks demonstrated MySQL HeatWave OLAP performance as much as an amazing 1392x superior to other MySQL cloud database competitors at a much lower cost. That's unheard of in modern cloud database services. And yet, Oracle has made it easy to replicate these results without bias. Oracle also published TPC-DS benchmarks versus competitive OLAP cloud database services including Amazon Redshift with AQUA (Advanced Query Accelerator), Snowflake, Google Cloud Platform (GCP) BigQuery, and Azure Synapse. Once again MySQL HeatWave's performance came out on top—ranging from 58% to 347% faster, and at a much lower cost.

But what about the total cost of ownership (TCO)? These benchmarks were comparing Oracle's multi-model, multi-workload, MySQL converged database to a single model database. None of the overall costs and inter-service performance latencies of other cloud database services were included. How does MySQL HeatWave compare to those competitors when those are added? And not just one-year static costs but over a three-year dynamic growth lifecycle. That's what this Wikibon research set out to answer.

Research Highlights

Oracle MySQL HeatWave running OLAP has the following TCO advantages.

- AWS is **642%** (**6.4x**) more costly.
- Snowflake is **723%** (**7.2x**) more costly.
- GCP is **512%** (**5.1x**) more costly.
- Azure is **327%** (**3.3x**) more costly.

Put another way, Oracle MySQL HeatWave is:

- **84%** less costly than AWS.
- **86%** less costly than Snowflake.
- **80%** less costly than GCP.
- **69%** less costly than Azure.

The MySQL HeatWave's OLAP TCO/Performance advantage is more undeniable.

- Versus Amazon Redshift—**MySQL HeatWave** is **3.73x** better in TCO/performance.
- Versus Snowflake—**MySQL HeatWave** is **11.27x** better in TCO/performance.
- Versus GCP BigQuery—**MySQL HeatWave** is **15x** better in TCO/performance.
- Versus Azure Synapse—**MySQL HeatWave** is **11.36x** better in TCO/performance.

MySQL HeatWave ML has the following TCO advantages.

- AWS is **1,049%** (**10.5x**) more costly.
- GCP is **515%** (**5.2x**) more costly.
- Azure is **389%** (**3.9x**) more costly.

Put another way, MySQL HeatWave is:

- **90%** less costly than AWS.
- **81%** less costly than GCP.
- **74%** less costly than Azure.

MySQL HeatWave ML's TCO/Performance advantage increases radically.

- Versus Amazon Redshift—**MySQL HeatWave** is **266.82x** better in TCO/performance.
- Versus GCP BigQuery—**MySQL HeatWave** is **130.88x** better in TCO/performance.
- Versus Azure Synapse—**MySQL HeatWave** is **98.85x** better in TCO/performance.

Research Framework

Calculating a three-year TCO for on-premises non-cloud databases is generally a relatively simple process. That is unfortunately not the case for most cloud database services. Far from it.

AWS, Snowflake, GCP, and Azure cloud database services have highly variable and unpredictable costs. The introduction of ML exacerbates the variability. Customers' invoices tend to fluctuate appreciably among several of these cloud providers, playing havoc with fixed budgets. According to interviews with dozens of AWS, Snowflake, GCP, and Azure users, predicting the actual true total costs at the end of each month—let alone a budget year is nearly impossible. Monthly invoices have been known to cause extensive sticker shock. Interviews with these customers and an examination of their bills illustrates the reasons behind those fluctuations. Amazon is the poster child for this unpredictability. They specifically charge for I/O costs. But I/O rates are particularly random. Just forecasting the I/O cost is an exercise in frustration. Amazon ML training is another volatile process that makes cost estimates nigh impossible without performing the ML process.

The primary reason behind that unpredictability is the requirement by many database cloud service providers to use several of their cloud services to meet the application needs. Each service ratchets up the bill. Compounding the bill shock is the distinct lack of clarity to the customer of all the underlying services being invoked such as the I/O charges, SageMaker for ML fees, etc. This problem would be largely eliminated or at least mitigated once the cloud provider offered a database with all or many of these capabilities built in.

It is obviously much simpler and lower cost from the customer point-of-view to utilize a single converged cloud database service. But for the cloud database service provider, a multi-model, multi-workload, converged database cloud service demands orders of magnitude more engineering on their part. It's much easier for them to link together multiple single-model database cloud services and other ancillary services. It shifts the work and risk to their customers, significantly reduces performance, and increases customer costs.

AWS, Snowflake, GCP, and Azure mostly offer single-model database cloud services. Only recently have they begun to introduce some nascent converged database cloud services due to competitive pressure. AWS, Azure, and GCP primarily offer separate relational databases, data warehouses, ML, and more cloud database services. AWS, Azure, and GCP have integrated various degrees of ML functionality into their data warehouses. However, the convergence being offered is still very limited. Snowflake is principally a data warehouse cloud service. They have added semi-structured, and unstructured data to their virtual data warehouse and only recently—June 2022—added a proprietary OLTP capability called Unistore[1], but is still only available in private preview. They do not currently offer integrated ML.

Oracle on the other hand, has been providing multi-model, multi-workload, converged databases for several decades and in their cloud since 2016. MySQL HeatWave is their converged open source cloud database service.

This TCO analysis is for MySQL users, the most popular transactional database on the planet per Enlyft. It's also very popular as a database cloud service being available from most public cloud service providers. The intent is to provide MySQL users a reasonable TCO comparison to allow them to make a better informed decision when moving to a MySQL cloud database service.

Solving the MySQL Cloud Database Service Shortcomings

It's a rare situation where a MySQL database service user doesn't need a data warehouse. It is well known the MySQL database is not optimized for reporting and analytics and as a result such queries can take a very long time to execute. MySQL users need data warehouse analytic capabilities to get the necessary detailed reporting and data analytics. The demand for ML has quickly made it table stakes for cloud database providers as it provides predictive modeling, faster, more accurate, and automated decision making, delivering faster more meaningful results. All three of these types of services are no longer "nice-to-haves," but rather "must" haves. There are two ways to solve this.

The first is to deeply integrate an online analytical processing—OLAP—engine and ML into the MySQL database. It's called a multi-model, multi-workload, converged cloud database service. That's what Oracle delivers with their MySQL HeatWave cloud database service.

[1] In preview (a.k.a. alpha/beta) in only a couple of AWS regions as of June 2022. Unistore is a Snowflake proprietary OLTP and not MySQL compatible or compatible with any of the current market standard OLTP databases.

The alternative is to provide connections between the MySQL cloud database service and separate OLAP database and ML cloud services. This is called single-model cloud database services. This is what AWS, Snowflake, GCP, and Azure elected to provide.

An essential problem with that single-model data warehouse cloud service is the necessity of copying the data, duplicating the data storage, manipulating, transforming the data, and moving the data. This is more commonly known as ETL or extract, transform, and load. Almost all the data in data warehouse cloud services does not originate within that data warehouse. Data warehouses are designed for OLAP. The data they analyze generally originates somewhere else. When that data originates in an OLTP database such as with MySQL, it must be moved and go through an ETL process—i.e., a time-consuming, usually manual, thankless process requiring ongoing adjustments while consuming cloud services, tools, and human resources. ETLs also increase data vulnerability because of the expressively expanded attack surfaces.

ETL cloud services and tools are usually not free and add noticeable cost to the TCO. Even when customers write their own code to perform the ETLs, they must still use compute, storage, and networking from the cloud service provider. In other words, more cost. But rather than code, document, patch, maintain, QA, and troubleshoot on an ongoing basis, it's much simpler, faster, easier, and generally cheaper to use automated ETL cloud services, which is why automated ETL services have become very popular. It's also important to note that ETLs inevitably add significant time to data analysis, so much so that OLAP and ML cannot analyze data in real-time. Often the results are stale, leading to reduced prediction accuracy. This affects time-to-market, time-to-actionable-insights, and time-to-revenue. These hard-to-measure, but potentially much higher costs are **not** captured in this TCO analysis. This lag problem is something that Snowflake has tacitly admitted with its introduction of Unistore, a proprietary and unproven integrated OLTP capability within its virtual data warehouse.

As previously stated, customers increasingly want to run machine learning on their data to train models and then run inference. ML has become competitive table stakes because of its importance in providing predictive analysis and more accurate automated decisions. ML needs to gather the data from data warehouses to train, correlate, and provide that predictive analysis. It does not necessarily require another ETL since several data warehouses do some of that ML data shaping and modeling. But it frequently requires copying, moving, and storing the data ML needs for its modeling. Amazon Redshift, Azure Synapse, and GCP BigQuery have limited ML modeling built into their data warehouses. However, each requires quite significant knowledge, expertise, and the skills of a data scientist to build successful ML models. They also require a lot of trial and error in using those models effectively—all of which adds cost. This Wikibon TCO research does not include the data scientist or trial and error costs.

Make no mistake, these costs are there. Please note that the extensive trial and error required for a given model and workload is outside the scope of this research. Also note that there are data warehouse ML cloud services that invoke other ML services to provide the ML processing, such as Amazon Redshift ML. It invokes Amazon SageMaker service underneath for ML processing and then those costs are included in the customer's monthly bill. SageMaker is not native to Redshift ML.

MySQL HeatWave's multi-model, multi-workload converged cloud database service, avoids all of the single-model issues. As previously discussed, MySQL HeatWave is deeply integrated with a high-performance analytics engine, extensive ML-based automation, real-time elasticity, and ML modeling. That ML modeling is also unique in that it incorporates AutoML which recommends the best model based on a very small sample—less than 0.1% of the data—for any given project, while providing crystal clear explanations of how it made its decisions. Oracle provides all of this at an affordable cost that appears to be much lower than its competitors, and targets MySQL database users. There is no additional cost to customers of MySQL HeatWave for running machine learning since it is all being executed inside the same service. With MySQL HeatWave, there are no ETLs, duplicate storage, multiple cloud database services, or even multiple interfaces, training, and support. It's a unique high-performance and lower-cost cloud database service. The question becomes how does it compare with other offerings on a TCO basis over three years?

Determining the TCO of MySQL HeatWave is fairly straight forward. Determining the TCO with most of the same capabilities for its competitors is not. The complexity of the single-model databases makes TCO comparisons complicated.

Research Scope

This Wikibon research tackles this complicated TCO comparison. As previously stated, estimating the TCO for multiple interconnected single-model database cloud services is likely to have inaccuracies. There are various reasons why estimating their TCO is so difficult. A key reason is that those cloud services providers strongly recommend that customers run a cloud hardware infrastructure through a trial-and-error process. This enables the customer to arrive over time, at an optimal cloud infrastructure for a given workload to achieve the required performance. In other words, customers are encouraged to tune their cloud environment to achieve acceptable performance. They are essentially telling their customers that it's all guesstimation. Of course, workloads change over time, requiring ongoing trial-and-error guesstimation and more cost.

Predicting performance for a given cloud infrastructure configuration can be extremely difficult. This is why customers are urged to adjust their configurations to meet their performance requirements based on trial and error. If a configuration comes up short, adjust the infrastructure with more resources until it meets performance requirements. If the configuration is overkill, adjust it downward. This will vary by workload and cloud database service. What makes it worse is when cloud providers abstract their cloud infrastructure from the underlying hardware. This abstraction decouples the performance and pricing from the actual hardware infrastructure. They instead use a proprietary measurement unit such as "SLOT" (GCP), "DWU" (Azure), or shirt sizes (Snowflake)—S, M, L, XL, 2XL, 3XL, 4XL, 5XL, 6XL—with no hardware referenced. Customers have historically estimated performance based on their experience with the hardware infrastructure they know. Until customers gain experience with these abstractions, there will be a lot of trial and error— meaning yet more undocumented additional cost. It's also why many customers go with "On-Demand" versus "Reserved" pricing. Unfortunately, On-Demand pricing often ends up costing more, although it can occasionally cost less.[2]

In addition, there are a lot of moving parts such as reserved versus on-demand infrastructure, broadly different hardware shapes with vastly different performance characteristics. There are also abstractions that relate to the underlying hardware shapes but lack detailed information about what they are. And, of course, there are diverse storage types that differ in both performance and cost.

The Wikibon TCO calculations attempt to level the hardware playing field by using equivalent or close to equivalent hardware infrastructure configurations. Configurations are based on vCPUs, memory, and bandwidth. The underlying assumptions of equivalent hardware should be relatively close in performance characteristics and cost. That turns out to be a bit naïve based on TPCH benchmark testing. Furthermore, when the underlying hardware infrastructure is not documented by the service provider as a direct result of their abstraction, it requires more unverifiable assumptions and additional guesswork. The TCO calculations in those cases where hardware abstractions are poorly correlated to predicted performance are likely to be underestimated.

Without substantial and time consuming trial and error, trying to level the infrastructure playing field is the best of bad choices.

It would be somewhat easier comparing the TCO of each different single-model cloud database service against a comparable service. Even so, the trial and error component makes it suspect. But, comparing the multiple interconnected single-model cloud database services to MySQL HeatWave's multi-model, multi-workload, converged cloud database service is much more complicated and difficult. It's a key reason many cloud database service providers recommend trial and error to achieve the correct underlying cloud hardware infrastructure for a given workload[2].

This Wikibon research strongly attempts to minimize these variables or call them out. Real-world workloads will vary. So will TCO. The many assumptions in this TCO analysis are detailed in Appendix A.

Premises

As previously stated, estimating the TCO for cloud database services is likely to have inaccuracies because so many services recommend a trial-and-error process for a given workload to achieve the acceptable performance.

[2] Oracle MySQL HeatWave Autopilot uniquely eliminates time consuming and costly trial and error by automatically determining the optimal size required for running a given workload simply by sampling less than .1% of the data.

The Enormous Oracle MySQL HeatWave TCO and TCO/Performance Advantage

Take the example of an Amazon Aurora MySQL OLTP customer who needs analytics or just simple reporting. To do so necessitates a separate Amazon Redshift data warehouse cloud service or another third party data warehouse cloud service such as Snowflake. The result is that the customer is now paying for two cloud database services. Azure MySQL customers need Azure Synapse or a third-party data warehouse cloud service. GCP CloudSQL–MySQL customers need BigQuery or a third-party data warehouse cloud service. That's not all the services they need to complete their reporting or analytics task. As previously discussed, MySQL OLTP data must go through an ETL before a data warehouse can analyze it. Those ETLs can be provided via custom developed code—time consuming, poorly documented, inadequately automated, difficult to QA, and troubleshoot—or through an additional cloud service, preferably an automated ETL cloud service.

Going back to that hypothetical Amazon customer, the most popular route is to use AWS' automated ETL service called "Glue". Glue provides the ETLs between Aurora MySQL and Redshift. Activating the Glue ETL process requires either another service to trigger Glue, called Lambda, or minimally a custom-coded time-based trigger. Time-based triggers are somewhat common, and unlike Lambda, do not add cost. For Snowflake, Azure, and GCP there are a several of third-party ETL services such as Hevo and Stitch, all of which have healthy fee structures based on row count. And there are yet more required fee-based services such as duplicated data on object storage.

In the case of AWS Glue, the ETL data must be copied into its own S3 object storage bucket, which obviously adds yet more cost. For Snowflake, the ETL MySQL data is copied and stored in the Snowflake data warehouse cloud, which also means more cost. Azure and GCP require similar additional fee-based object storage services.

The undeniable wild card in calculating TCO is ML. Whereas ML adds another pay per use cloud service, the cost is wildly ML model and run-time dependent. The AWS ML service is SageMaker. But, when utilizing Redshift ML, SageMaker is an underlying service that it is not specifically contracted as a separate service but is still used and billed. Amazon Redshift ML is a feature of Redshift that does not cost any more than Redshift without ML. However, Redshift ML only provides a ML model framework. In fact, choosing the ML model and run time is completely up to the customer. Redshift ML provides no insight or recommendations. Redshift ML calls SageMaker to perform the actual ML training, runtime, and conclusions. SageMaker fees show up in the customer's bill, often as a surprise. The actual SageMaker hardware infrastructure used in any given run is selected by SageMaker algorithms. And it will vary based on the ML model. Therefore, predicting SageMaker costs is nearly impossible in this scenario—even when using Amazon pricing calculators.

Azure requires Synapse ML and Apache Spark pools for ML, whereas GCP appears to have integrated ML into its BigQuery data warehouse. Unlike MySQL HeatWave, neither of these ML cloud services suggest the best or optimal model to use for any given set of data at this time. Azure and GCP data warehouse integrated ML (Synapse and Big Query) are as convoluted as AWS, making TCO generally unpredictable and at best a guesstimation.

Snowflake does not currently provide an integrated ML cloud service. Snowflake only positions its data warehouse cloud as pre-conditioning the data for ML. There are no ML modeling tools, modeling suggestions, or pre-built models. Snowflake has several third party ML partners that can be contracted with separately. Note that all of them demand data scientist skills as well as trial and error. Snowflake recommends that the customer run with the ML tools they're most comfortable. Estimating a Snowflake ML cost is a non-starter because there are simply way too many unknowns in third-party tools, services, and fees.

Estimating the MySQL HeatWave costs is considerably easier and predictable. That's true for analytics with and without ML. The Oracle Cloud Infrastructure (OCI) calculator is straightforward. First and foremost, it's a single tightly integrated cloud database service that includes the data warehousing and ML. There are also no ETLs, third-party services, or tools to determine the cost. That makes calculating its TCO straight forward and orders of magnitude simpler.

Comparing the TCO of these very different cloud database services is a considerable challenge requiring many assumptions. Keep in mind that costs are fluid depending on locations and many other highly dynamic variables. The intent of this Wikibon research is to provide TCO comparisons that are as accurate and normalized as possible. However, the results are likely to be imprecise especially when looking at AWS, Snowflake, GCP, and Azure resulting from the issues raised above.

An excellent example that contrasts the simplicity of MySQL HeatWave with the complexity of their competitors is their cost estimator calculators. The MySQL HeatWave cost estimator calculator is brain dead simple. All it asks for is the storage capacity used by MySQL HeatWave and backup storage if that's desired. The total monthly costs are right there. AWS, Snowflake, GCP, and Azure are a completely different story. This is because the cost of each cloud service must be estimated separately to get a complete picture. It's neither simple nor self-explanatory.

Complicating the matter is TCO/performance. Establishing performance comparisons goes back to public TPC-DS benchmarks. Those TPC-H benchmarks have not been run at the assumed parameters. Since AWS, Snowflake, GCP, and Azure are all relying on ETLs and duplicated copied data, those benchmarks do not include the ongoing considerable lag or latency that are part of the ETLs. In other words, they're demonstrably slower than the benchmarks results. They were already much slower than MySQL HeatWave without the additional ETL latencies. The following TPC-DS benchmark charts—test and results available on GitHub—show the performance and price comparison of MySQL HeatWave, Amazon Redshift with no-cost AQUA (Advanced Query Accelerator), Azure Synapse, GCP BigQuery, and Snowflake.

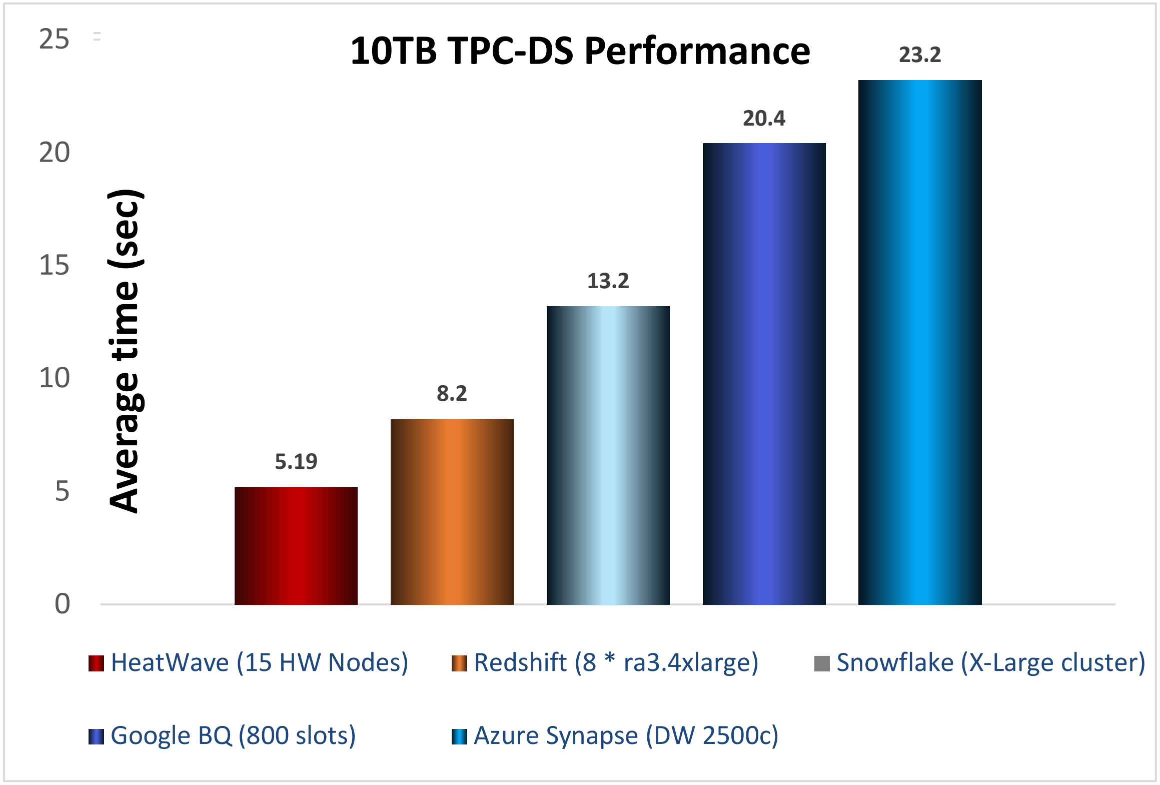


Chart 2: 10TB TPCDS Performance Comparison

The Enormous Oracle MySQL HeatWave TCO and TCO/Performance Advantage

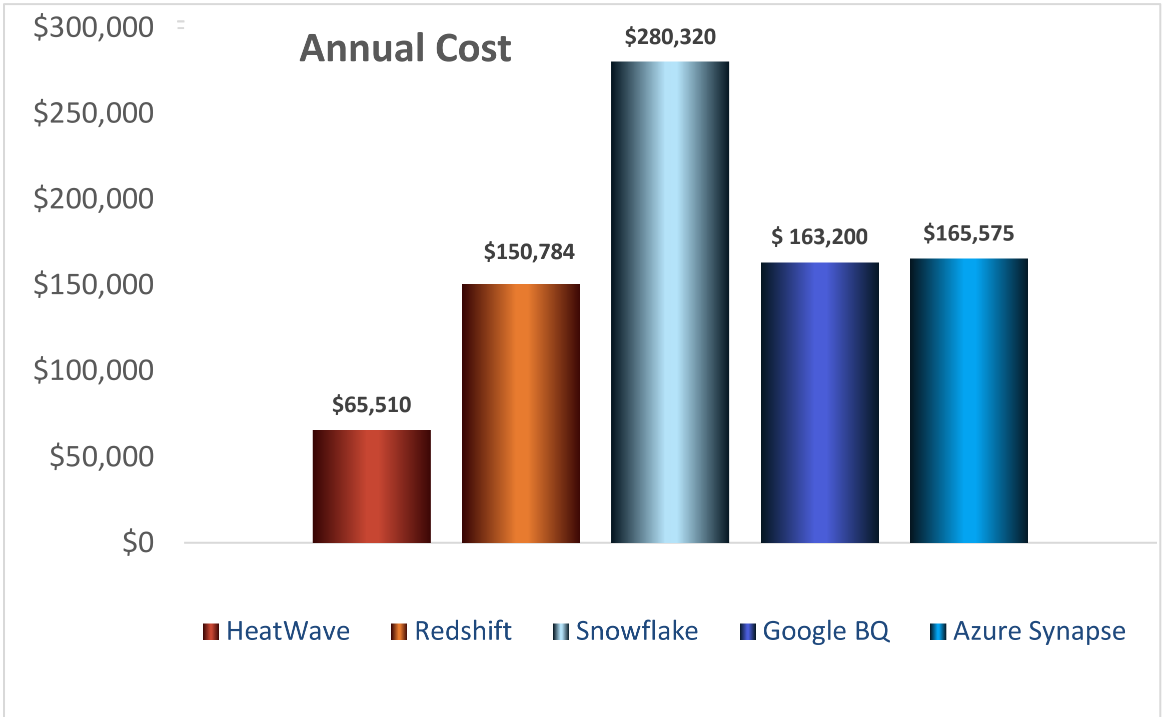


Chart 3: 10TB TPCDS Annual Cost Comparison

The TPC-DS is a data warehouse query benchmark and not a ML benchmark. The Amazon Redshift, Snowflake, GCP BigQuery, and Azure Synapse comparisons do not include the other cloud services required or ETL lag and thus are not reflected in their performance or annual cost. Even so, these are difficult to ignore.

The performance of MySQL HeatWave in the TPC-DS benchmark is overwhelmingly higher than other cloud databases:

- **58%** faster than Amazon Redshift.
- **154%** faster than Snowflake.
- **293%** faster than GCP BigQuery.
- **347%** faster than Azure Synapse.

On annual costs running the same TPC-DS benchmark, MySQL HeatWave is significantly lower than other cloud databases:

- **57%** lower cost than Amazon Redshift.
- **77%** lower cost than Snowflake.
- **60%** lower cost than GCP BigQuery.
- **60%** lower cost than Azure Synapse.

This equates into a huge cost/performance disparity:

- MySQL HeatWave came in at **$.26/s.**
- Amazon Redshift came in at **$.94—MySQL HeatWave** is **3.64x** better price/performance.
- Snowflake came in at **$2.82—MySQL HeatWave** is **10.88x** better price/performance.
- GCP BigQuery came in at **$2.53—MySQL HeatWave** is **9.79x** better price/performance.
- Azure Synapse came in at **$2.92—MySQL HeatWave** is **11.30x** better price/performance.

Expanding this to total cost of ownership starts with the assumptions listed in Appendix A.

The TCO Component Parts

The scope of this Wikibon TCO research will cover two areas. The first will be TCO for MySQL with OLAP. The second will also include ML. The TCO for ML comparison will not include Snowflake since they do not have a built-in or integrated ML service.

Oracle Cloud Infrastructure (OCI): MySQL HeatWave with Autopilot, and ML, Oracle Object Storage

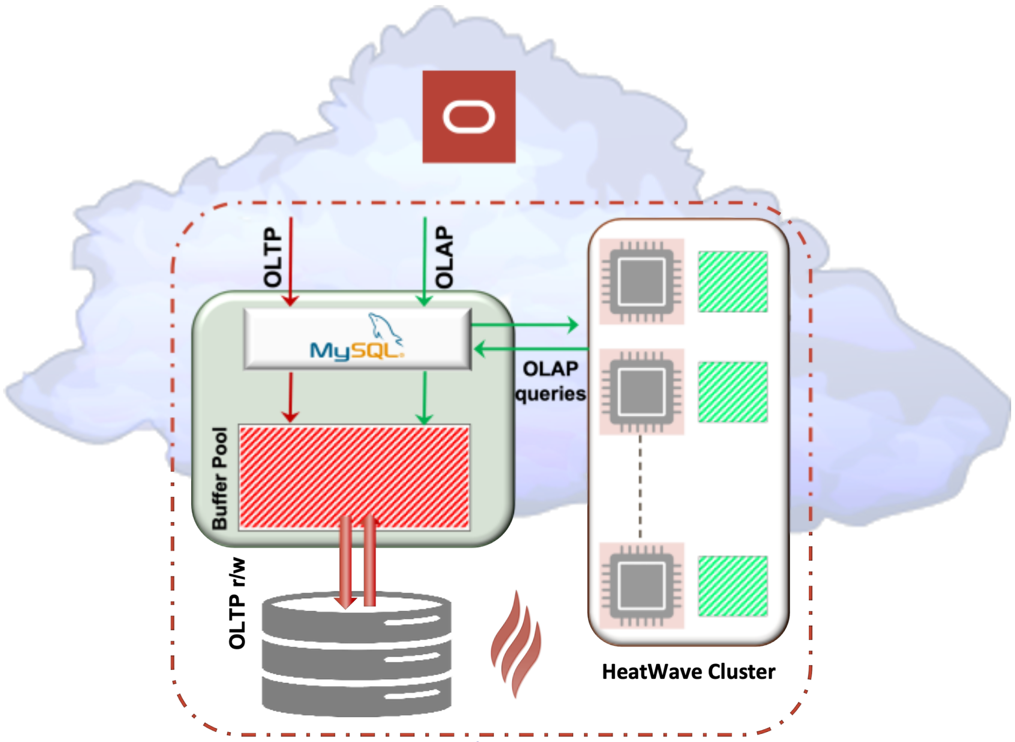


Figure 1: Oracle MySQL HeatWave

MySQL HeatWave is a multi-model, multi-workload, converged cloud database service that includes the MySQL transactional database, optimized for performance, and deeply integrated with a built-in analytics engine, ML powered automation, ML modeling, and real elastic scaling. This easy to use service comes with an easy-to-understand fee structure. Costing out the MySQL HeatWave cloud database service with the OCI pricing calculator was very simple and straightforward, especially when compared to the pricing calculators of the other cloud database service providers.

No other currently available MySQL cloud database service has an equivalent tightly integrated service or can match the performance of MySQL HeatWave.

One more noteworthy aspect of MySQL HeatWave is the built in scale-out storage. It offers a distinctive parallel data restore from the scale-out storage that's lightning fast at no additional cost. It provides this capability for all operations being done on the MySQL HeatWave cluster.

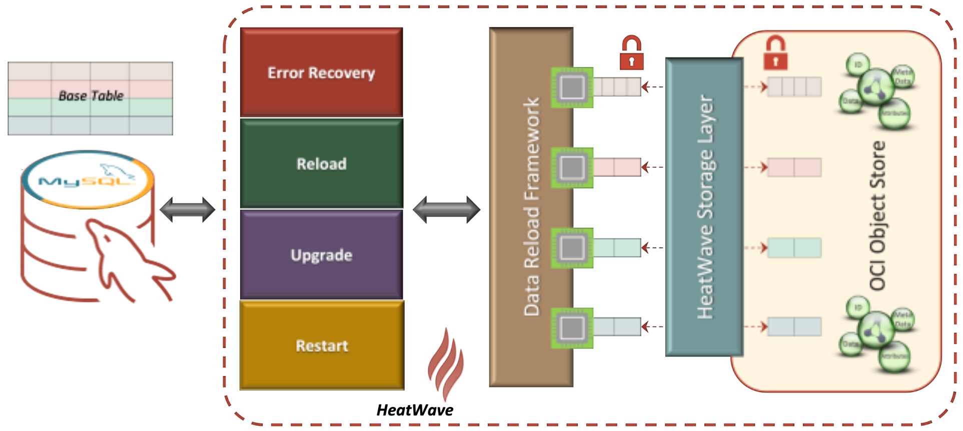


Figure 2: MySQL HeatWave Ultra-Fast Recovery From Scale-out Storage

While backup storage is included in the TCO calculations for MySQL HeatWave, it is not for any of the other cloud database service providers. The logic is that their OLTP data has already been copied, gone through an ETL process at least once, and stored on additional storage. If there is a failure on either the OLTP or OLAP side, the data is recoverable from the other. Albeit there will be an ETL process. And the recovery process will be quite slow. Although their recovery processes today are primarily sequential versus parallel—meaning not fast. Therefore, making an additional copy of the data is potentially redundant.

AWS: Aurora MySQL, EBS, Redshift, AQUA, Glue, S3 Object storage, Redshift ML, and SageMaker

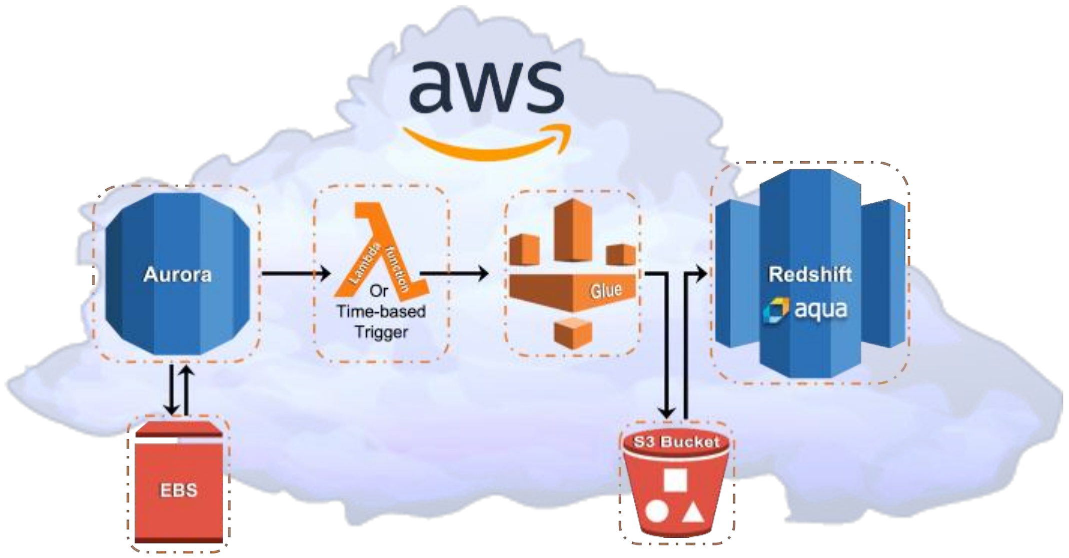


Figure 3: Amazon Aurora MySQL with Glue and Redshift AQUA

The Amazon business philosophy is to offer specialized services for every type of database and connectors to link databases. AWS delivers on this by employing individual database services for each different database model. Each service operates as a separate paid for service. This is known as a best-of-breed model. Even though AWS cloud database services are solid services, Wikibon, as well as other analysts and customers, does not consider them as proven best-of-breed. More importantly, the AWS model goes counter to the needs of modern IT organizations. They need higher levels of analytical integration with different models to address the value in their data. The way AWS addresses this problem is with additional services that ETL or pipeline data between the different analytical models. AWS layers these additional fee services—generally on a per use basis only—on top of their cloud database services, and sometimes without the customer's knowledge, as in SageMaker as an adjunct to Redshift ML.

Therefore, in order to compare a relatively equivalent AWS TCO to MySQL HeatWave requires the combination of Aurora MySQL (transactional database), EBS (elastic block storage for Aurora MySQL), Redshift (data warehouse), AQUA (data warehouse query accelerator), Lambda (trigger), Glue (automated ETL), S3 (landing object store bucket for ETL MySQL massaged data to be read by Redshift.

For ML there's an additional separate S3 object store bucket for SageMaker, Redshift ML (Redshift modeling and interface to SageMaker) and SageMaker (ML analytics).

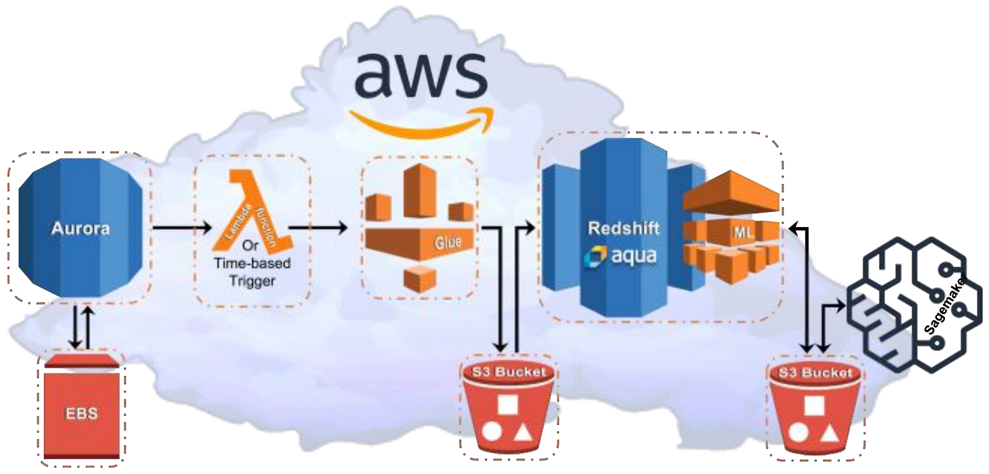


Figure 4: AWS MySQL with Glue, Redshift AQUA, and ML

This is a considerable number of fee-based moving parts. The combination of these required services can be, and often is confusing, with fee structures/bills that are frequently surprisingly high and unpredictable.

Snowflake: Aurora MySQL, S3 Object Storage, Stitch, Snowflake Virtual Data Warehouses

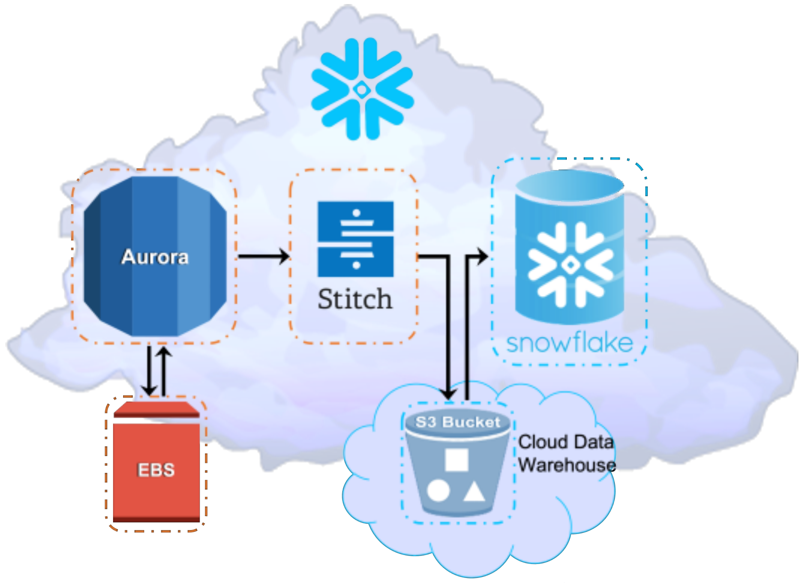


Figure 5: Amazon Aurora MySQL with Stitch ETL services and Snowflake Virtual Data Warehouses

Whereas Amazon is a full-service public cloud, Snowflake is primarily a data warehouse cloud service that runs in the AWS, Azure, or Google public clouds. It has limited transactional capability with the still in-private preview Unistore. But it has no MySQL cloud database service, ML, or ETL services. Those other cloud services must be accounted for to provide somewhat equivalent functionality to MySQL HeatWave.

An essential TCO assumption is determining which MySQL cloud database service to use since Snowflake's Unistore is proprietary and not a MySQL fork or clone. The excessive effort and cost to consolidate multiple MySQL databases to Unistore is outside the scope of this research. For the purposes of this TCO analysis, Amazon Aurora MySQL costs are used. It could just as easily be Azure MySQL or GCP Cloud SQL–MySQL. However, the cost differences are nominal. Stitch was used for the automated ETL service costs because it works and is relatively inexpensive.

One key problem in determining TCO for Snowflake is estimating which virtual hardware size to use. Snowflake abstracts their virtual data warehouse sizes from the underlying cloud hardware infrastructure.

There are no references as to how those sizes correlate to actual hardware configurations. Once again, customers follow the trial and error process to optimize hardware infrastructure.

Snowflake HW		Standard		Enterprise		Business Critical	
Shapes	Nodes	Per Hr.	Per Sec.	Per Hr.	Per Sec.	Per Hr.	Per Sec.
XS	1	$2	$0.00056	$3	$0.00083	$4	$0.00111
S	2	$4	$0.00111	$6	$0.00167	$8	$0.00222
M	4	$8	$0.00222	$12	$0.00333	$16	$0.00444
L	8	$16	$0.00444	$24	$0.00667	$32	$0.00889
XL	16	$32	$0.00889	$48	$0.01333	$64	$0.01778
2XL	32	$64	$0.01778	$96	$0.02667	$128	$0.03556
3XL	64	$128	$0.03556	$192	$0.05333	$256	$0.07111
4XL	128	$256	$0.07111	$384	$0.10667	$512	$0.14222
5XL*	256	$512	$0.14222	$768	$0.21333	$1,024	$0.28444
6XL*	512	$1,024	$0.28444	$1,536	$0.42667	$2,048	$0.56889
*Preview in 2 AWS regions only. Pricing is subject to change without notice.							

Table 1: Snowflake Compute/Hr. in USD $ - Billed/Hr. and per Sec.

In addition, Snowflake 'Time Travel'—querying, cloning, and restoring historical data in tables, schemas, and databases—is free for up to one day of data in with the standard edition. It does add storage costs. If more than one day is required, the extended edition provides it up to 90 days. However, the software costs are then increased by 50%. For the purposes of this TCO research, the standard edition is used.

The lack of integrated ML within Snowflake by Snowflake makes it a non-starter in estimating TCO. Regrettably, there's no simple, convenient, or fast process to use ML with Snowflake without considerable custom code or trial and error with a third party partner. There is nothing from Snowflake even as rudimentary as Redshift ML. Therefore, TCO modeling for ML in Snowflake is not included.

One final note on Snowflake TCO. Snowflake runs in Amazon, Azure, and GCP. Costs will fluctuate depending on which cloud Snowflake is running. Moving data between clouds will add substantial egress fees. This Wikibon TCO research used the Amazon Aurora MySQL costing because it is the most prevalent. But it could have just as easily used Azure or GCP MySQL cloud database service.

GCP: Cloud SQL – MySQL, Block Storage, Object Storage, Stitch, BigQuery, and BigQueryML

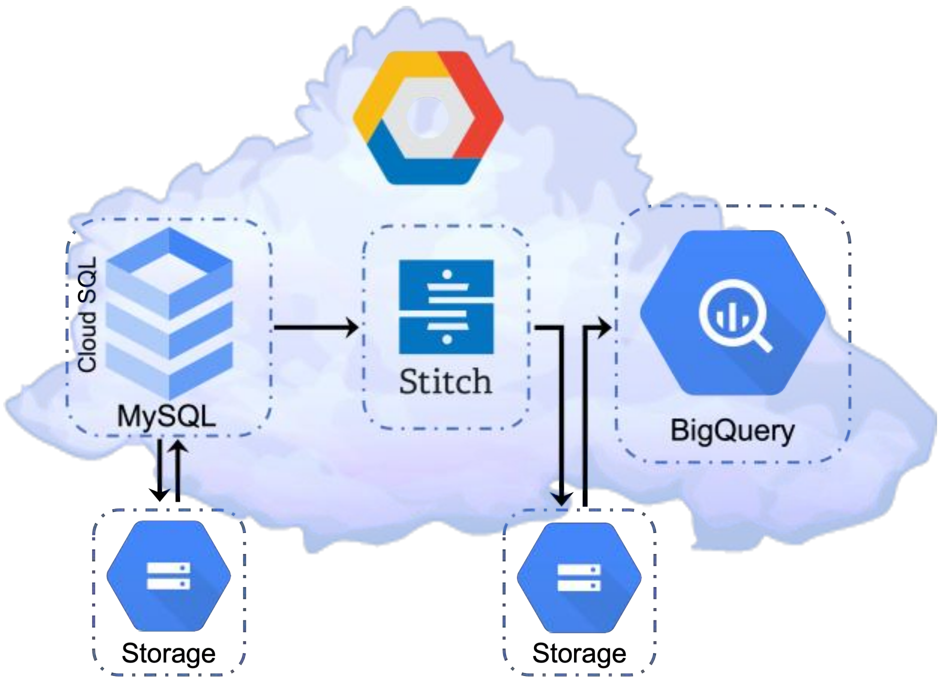


Figure 6: GCP Cloud SQL–MySQL, Stitch, BigQueryML, Block, Object Storage

GCP has a similar philosophy to AWS in that they offer multiple single-model cloud database services. They do not offer quite as many as AWS, but that has no impact on this TCO research. Google's nomenclature is somewhat different from AWS. The MySQL OLTP is called Cloud SQL. GCP's data warehouse cloud service is BigQuery. Stitch automated ETLs are again used due to their simplicity, ease of use, relatively low cost, and users say they're satisfied with how it works.

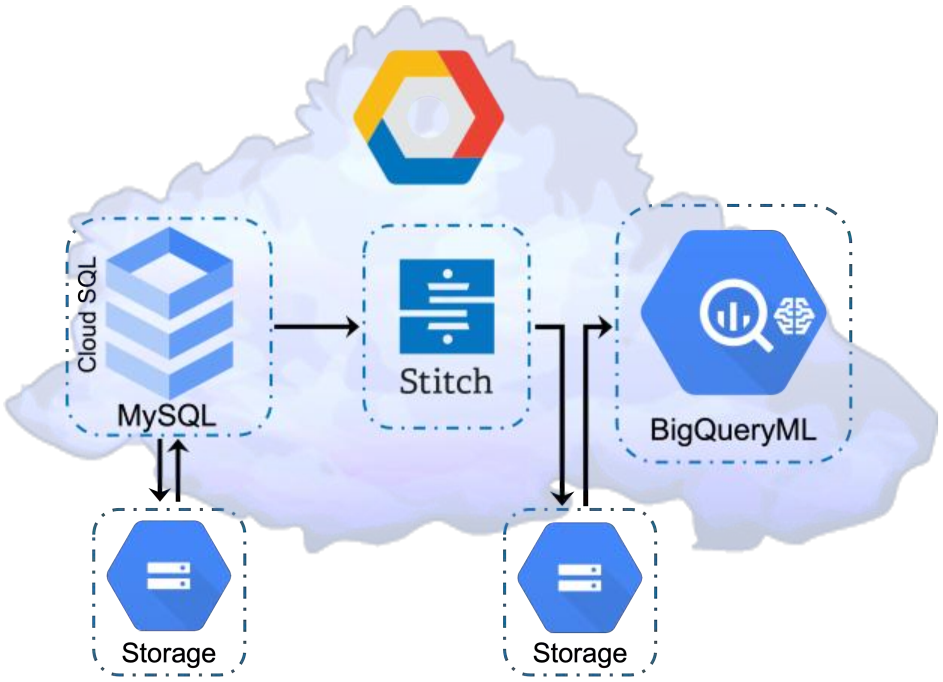


Figure 7: GCP Cloud SQL–MySQL, Stitch, BigQueryML, Block, Object Storage

BigQuery also has some built-in ML modeling. The modeling is somewhat rudimentary and requires a data scientist's understanding of which model to use and configure. And, like AWS, demands trial-and-error guesstimation. BigQueryML may or may not require more 'SLOTS'. SLOTS are GCP's abstraction for underlying cloud hardware infrastructure.

Azure: MySQL, Block Blob Storage, Object Blob Storage, Stitch, Synapse, SynapseML, and Spark Pools

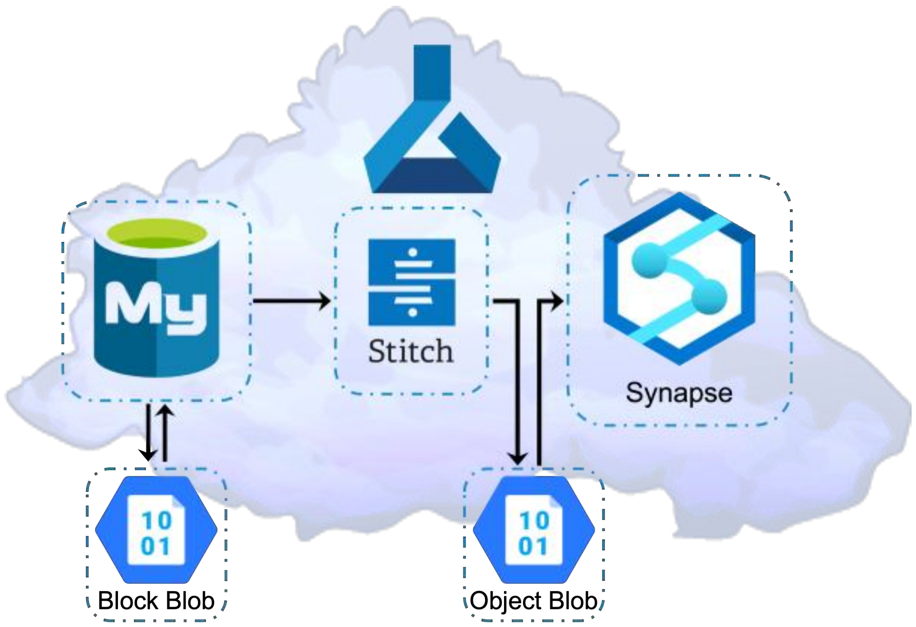


Figure 8: Azure MySQL, Stitch, Synapse, Block, Object Storage

Azure's philosophy is similar to AWS and GCP in that they offer multiple single-model cloud database services. However, they do not offer quite as many as AWS. Azure's nomenclature is different from both. Azure's cloud data warehouse service is Synapse. Stitch automated ETLs are also used for this TCO research.

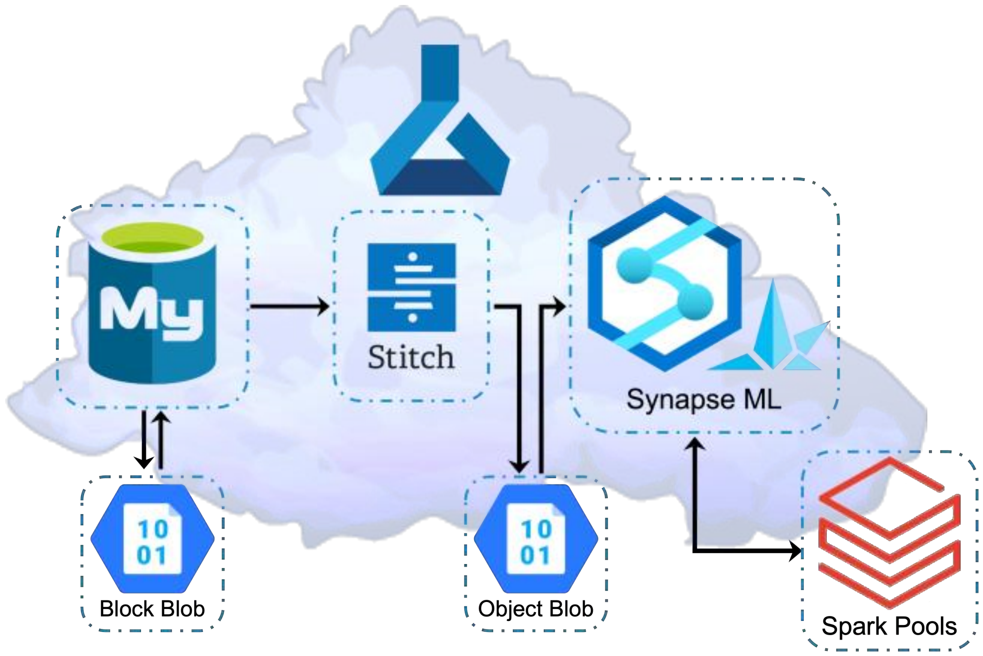


Figure 9: Azure MySQL, Stitch, SynapseML, Block, Object Storage, Apache Spark Pools

Synapse similarly has some built-in ML modeling. The modeling is relatively basic and requires a data scientist's understanding of which model to use and configure. And like AWS and GCP, hardware configuration demands substantial trial and error. Azure makes it pretty clear that 'Spark Pools' are required for ML processing. What is unclear is the amount of DWU blocks (cloud hardware infrastructure abstraction) required for performance.

TCO Analysis

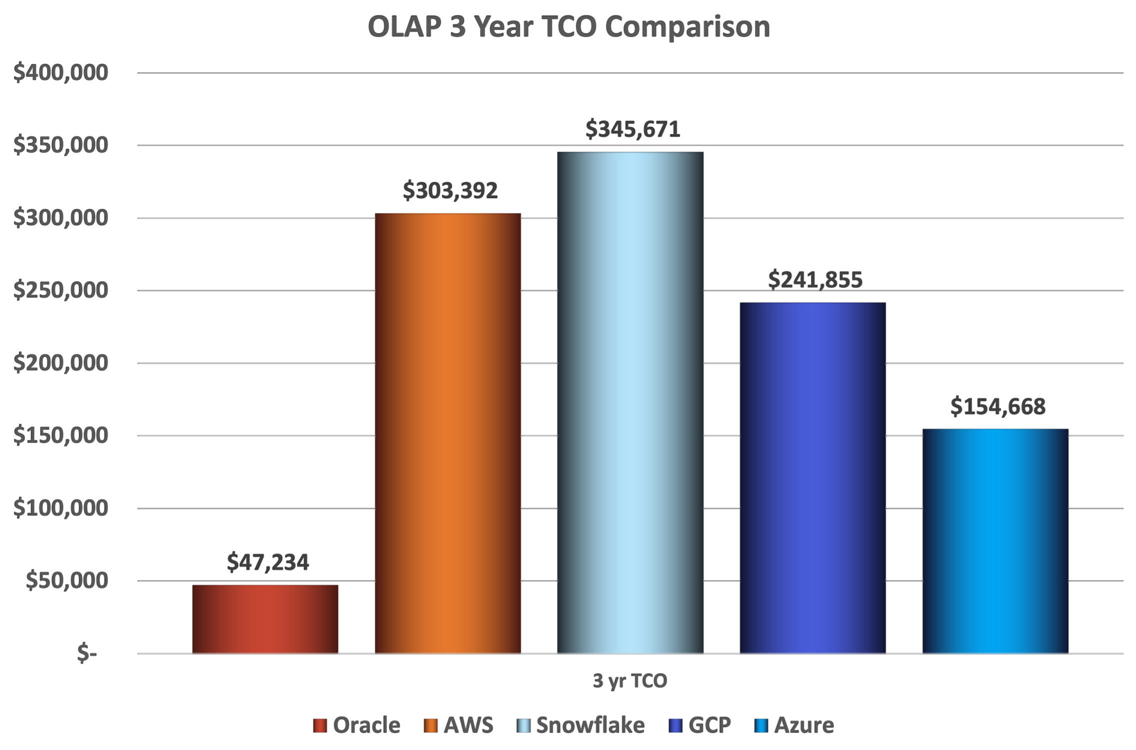


Chart 4: OLAP 3 year TCO Comparison

OLAP 3 Year TCO Comparison					
	Oracle	AWS	Snowflake	GCP	Azure
Year 1	$ 10,984	$ 85,353	$ 98,296	$ 52,803	$ 38,207
Year 2	$ 15,156	$ 98,672	$ 115,067	$ 78,761	$ 54,164
Year 3	$ 21,094	$ 119,367	$ 132,308	$ 110,291	$ 62,296
3 yr. TCO	**$ 47,233**	**$ 303,392**	**$ 345,671**	**$ 241,855**	**$ 154,668**

Table 3: OLAP 3 year TCO Comparison

Based on these results, compared to MySQL HeatWave:

- AWS is **642%** (**6.4x**) more costly.
- Snowflake is **723%** (**7.2x**) more costly.
- GCP is **512%** (**5.1x**) more costly.
- Azure is **327%** (**3.3x**) more costly.

Put another way, MySQL HeatWave is:

- **84%** less costly than AWS.
- **86%** less costly than Snowflake.
- **80%** less costly than GCP.
- **69%** less costly than Azure.

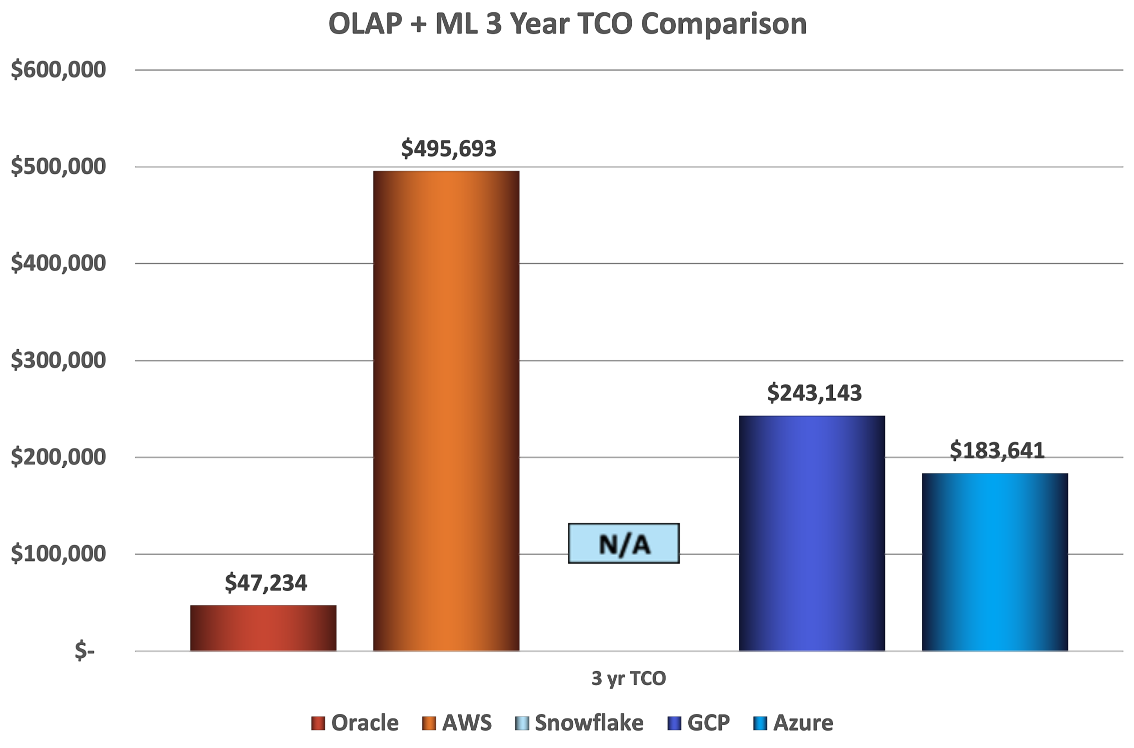


Chart 5: OLAP + ML 3 year TCO Comparison

OLAP + ML 3 Year TCO Comparison					
	Oracle	AWS	Snowflake	GCP	Azure
Year 1	$ 10,984	$ 248,710	N/A	$ 53,311	$ 47,865
Year 2	$ 15,156	$ 109,799		$ 79,071	$ 63,822
Year 3	$ 21,094	$ 137,183		$ 110,761	$ 71,954
3 yr. TCO	$ 47,233	$ 495,693		$ 243,143	$ 183,641

Table 4: OLAP + ML 3 year TCO Comparison

Based on these results, compared to MySQL HeatWave:

- AWS is **1,049%** (**10.5x**) more costly.
- GCP is **515%** (**5.2x**) more costly.
- Azure is **389%** (**3.9x**) more costly.

Put another way, MySQL HeatWave is:

- **90%** less costly than AWS.
- **81%** less costly than GCP.
- **74%** less costly than Azure.

A key reason for the increasing disparity comes from the fact that Oracle includes comprehensive ML within the MySQL HeatWave cloud database service.

This TCO analysis for AWS, GCP, and Azure is highly conservative. In the Oracle ML dataset benchmark testing, MySQL HeatWave ML actually came in at **99%** lower cost than Amazon Redshift ML because there is no charge for ML in MySQL HeatWave.

TCO Analysis Caveats

Putting together an accurate configuration for a given workload on AWS, Snowflake, GCP, and Azure requires a considerable amount of trial and error. As important as it is to get the configuration right for OLAP, it's absolutely essential for ML. As previously discussed, extensive trial and error was outside the scope of this Wikibon research. It likely means that the configurations used in determining TCO for these services is under-configured and under-costed. It will not have a material impact on the vast disparity in costs and performance compared to MySQL HeatWave.

Additionally, this TCO analysis does not consider several major factors that further increase the disparities. The first is how performance affects time-to-market, time-to-actionable-insights, and time-to-revenue. Faster performance means faster time-to-market/actionable-insights/decisions/revenue. That faster revenue is revenue that would not be attainable otherwise and is likely to dwarf the cloud database service costs. MySQL HeatWave's performance is appreciably faster than their competition as demonstrated in the published benchmarks. However, as previously mentioned, those benchmarks do not account for the additional lag and cost of the ETLs. Even if the competing cloud service throws more hardware infrastructure at their cloud database services, they cannot close the gap.

The second major impact not accounted for is automation or lack thereof. MySQL HeatWave exclusively includes substantial machine learning-based automation in Autopilot, eliminating much of the manual labor-intensive tasks required by the DBA in the other cloud database service providers.

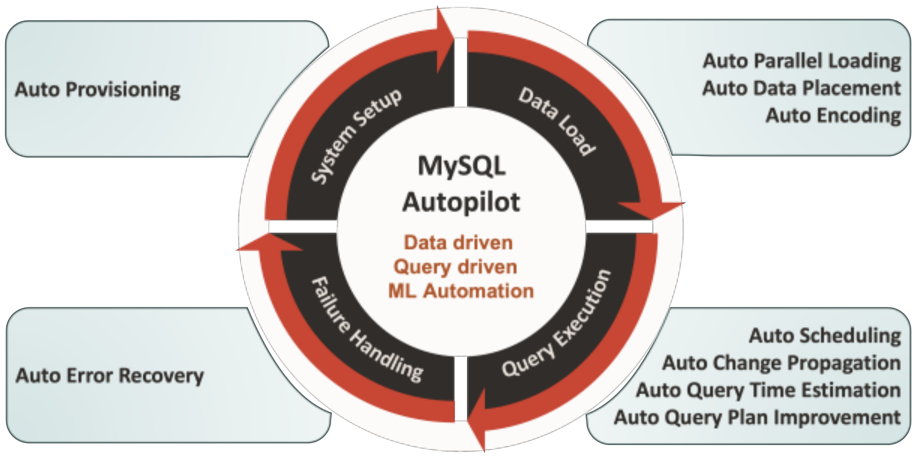


Figure 10: Oracle MySQL HeatWave Autopilot Automation

One example is data placement. Autopilot automatically uses AutoML and workload-aware placement keys to place data in the best location to limit data movement. Limiting data movement means better performance.

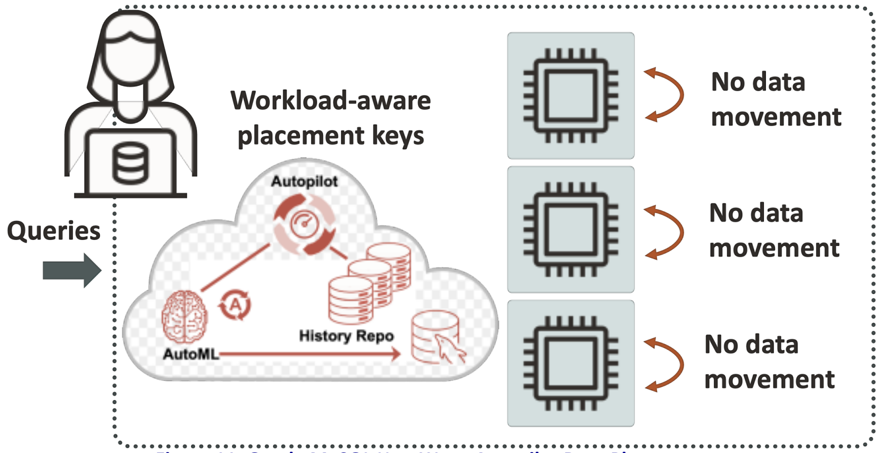
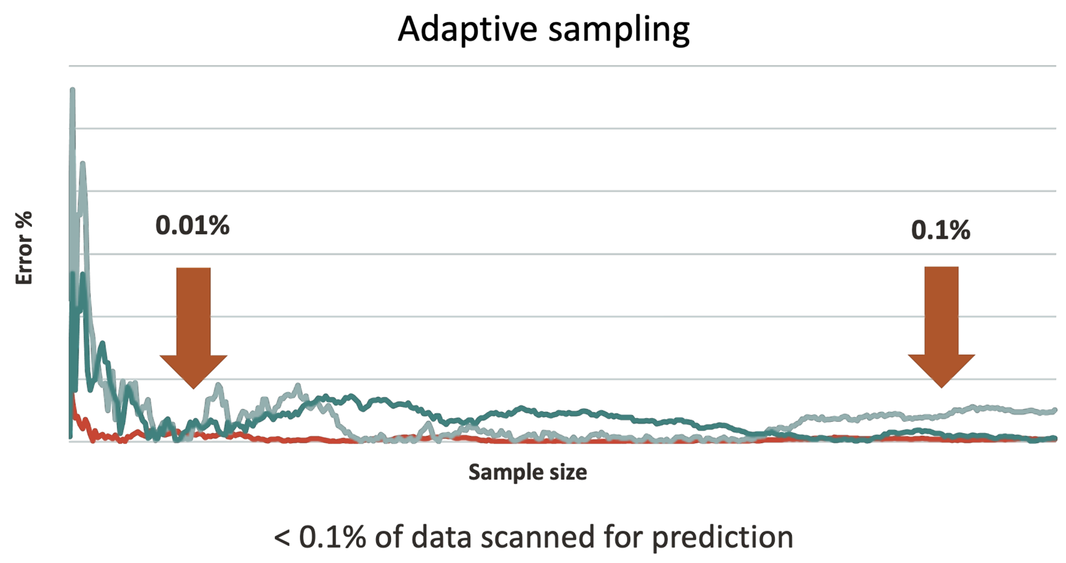
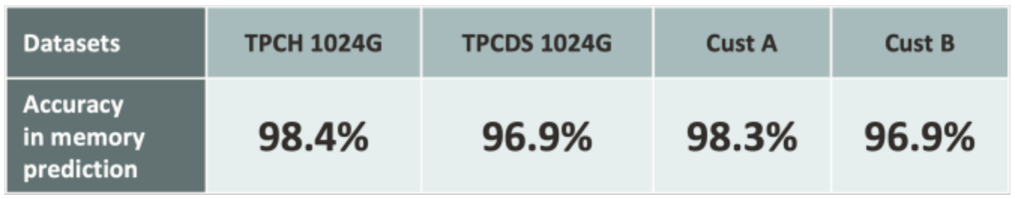


Figure 11: Oracle MySQL HeatWave Autopilot Data Placement

Another major task eliminated with Autopilot is the extensive trial and error. Autopilot samples as little as 0.1% of the data to automatically configure the most optimal and cost effective HeatWave cluster.





Datasets	TPCH 1024G	TPCDS 1024G	Cust A	Cust B
Accuracy in memory prediction	98.4%	96.9%	98.3%	96.9%

Figure 12: Oracle MySQL HeatWave Autopilot Adaptive Sampling

Nor does this TCO consider the non-disruptive real-time scaling (elasticity) of MySQL HeatWave. AWS, Snowflake, GCP, and Azure all require a disruption when scaling. The disruption can range from minutes to significantly longer.

More detailed information on MySQL HeatWave Autopilot can be found here.

TCO/Performance Analysis

Based on TPC-DS and TPC-H benchmarks for OLAP, MySQL HeatWave has demonstrated a performance advantage of 58% over AWS, 154% over Snowflake, 293% over GCP, and 347% over Azure. And again, that is without the ongoing ETL process latencies and added cost.

Just using these performance metrics, the MySQL HeatWave TCO/performance advantage comes in somewhat greater than the previous benchmark price/performance comparisons.

- Versus Amazon Redshift—**MySQL HeatWave** is **3.73x** better in TCO/performance.
- Versus Snowflake—**MySQL HeatWave** is **11.27x** better in TCO/performance.
- Versus GCP BigQuery—**MySQL HeatWave** is **15x** better in TCO/performance.
- Versus Azure Synapse—**MySQL HeatWave** is **11.36x** better in TCO/performance.

The MySQL HeatWave OLAP TCO/Performance advantage is compelling and in line with what actual customers are experiencing. But what about ML?

That performance advantage grows even more with ML. The MySQL HeatWave ML performance advantage based on all 18 of the ML benchmarks, averages more than 25x better performance than Amazon Redshift ML (see Table 5 below) without AWS' additional ongoing required ETL process latencies and added cost for the Amazon SageMaker service. In addition, MySQL HeatWave ML was more accurate than Redshift ML, doesn't require any manual step and is offered at no additional charge to MySQL HeatWave customers.

The Enormous Oracle MySQL HeatWave TCO and TCO/Performance Advantage

Dataset	Accuracy		Training Time (min)		Performance Advantage
	Oracle (2 nodes) HeatWave ML	Amazon Redshift ML	Oracle (2 nodes) HeatWave ML	Amazon Redshift ML	
Classification					
Airlines	**0.6524**	0.5	**2.71**	90	**33.21x**
Bank	0.7115	**0.8378**	**3.72**	90	**24.19x**
CNAE-9	**0.9167**	**FAILED**	**5.91**	**FAILED**	**∞**
Connect-4	**0.697**	0.6752	**7.13**	90	**12.62x**
Fashion MNIST	**0.9073**	**FAILED**	**181.85**	**FAILED**	**∞**
Nomao	**0.9602**	0.9512	**3.3**	90	**27.27x**
Numerai	**0.5184**	0.5	**0.34**	90	**264.71x**
Higgs	**0.758**	0.5	**68.58**	90	**1.31x**
Census	0.7946	**0.7985**	**1.22**	90	**73.77x**
Titanic	0.766	**0.9571**	**0.47**	90	**191.49x**
CC Fraud	**0.9256**	0.9154	**29.06**	90	**3.1x**
KDD Cup	**0.5**	**FAILED**	**3.55**	**FAILED**	**∞**
GEOMETRIC MEAN	**0.75**	**0.71**	**3.56**	**90**	**25.27x**
Regression					
Black Friday	**0.53**	0.54	**1.14**	90	**78.8x**
Diamonds	**0.98**	0.98	**2.4**	90	**37.42x**
Mercedes	**0.61**	**FAILED**	**1.16**	**FAILED**	**∞**
News Popularity	0.01	0.02	**0.6**	90	**149.13x**
NYC taxi	**0.25**	0.19	**7.34**	90	**12.26x**
Twitter	0.93	0.88	**44.24**	90	**2.03x**
GEOMETRIC MEAN	**0.26**	**0.27**	**3.52**	**90**	**25.58x**

Table 5: Oracle MySQL HeatWave ML versus Amazon Redshift ML Standard Dataset Benchmarks

There are no published ML dataset benchmarks for GCP and Azure at this time. However, based on the previous OLAP benchmarks, it is assumed for this research that similar MySQL HeatWave performance advantages are realized versus GCP and Azure. Based on these results and assumptions the TCO/performance advantages for MySQL HeatWave ML are even more compelling.

- Versus Amazon Redshift ML—**MySQL HeatWave** is **266.82x** better in TCO/performance.
- Versus GCP BigQuery – **MySQL HeatWave** is **130.88x** better in TCO/performance.
- Versus Azure Synapse – **MySQL HeatWave** is **98.85x** better in TCO/performance.

This is again in line with what actual customers are experiencing.

The question becomes, how is MySQL HeatWave ML training so much faster? It's because HeatWave ML is able to train the model in a single pass by using meta learned proxy models. It uses a highly parallel hyperparameter tuning technique in addition to intelligent sampling that does not compromise accuracy. In addition, the training automatically converges and scales with cluster size.

Conclusion

Oracle MySQL HeatWave is an unprecedented breakthrough in query processing and machine learning for MySQL users. Its performance is in a class by itself. And all these capabilities are available inside a single cloud database service. That alone should be enough to persuade most MySQL users the ideal cloud database service to use.

However, when combined with its incredibly low total cost of ownership, extraordinarily high degree of automation, easy to use and incredibly fast and accurate ML, with unique explainability, it's one of the easiest decisions database purchasers can make. Clearly, as demonstrated in this research, single-model cloud database services from AWS, Snowflake, GCP, or Azure cannot compete MySQL HeatWave.

What about on-premises MySQL users? Although not included in this research, it's obvious to the lay observer that MySQL HeatWave delivers much higher performance, greater scalability, and much lower cost than any attempt to use OLAP and ML on-premises with MySQL.

The good news for those users is that they can replicate their on-premises MySQL databases to Oracle MySQL HeatWave and take advantage of this incredibly easy to use low cost service.

The proof is in the pudding. Oracle has a free tier to try MySQL HeatWave. Prove it to yourself.

Appendix A: TCO Calculation Assumptions.

	Oracle	AWS	Snowflake	GCP	Azure
Hardware (HW) infrastructure	Same HW for all functions	Separate HW for MySQL & Data Warehouse & ML; HW same as OLAP for GCP. Snowflake has no ML.			
Pricing	PAYG	1 yr. reserved			
OLTP DB Cloud Service	MySQL HeatWave	Aurora MySQL	AWS Aurora MySQL	Cloud SQL (MySQL)	MySQL
HW infrastructure (nodes)	VMs	db.m5.xlarge	db.m5.xlarge	Not stated	E2 v4, 2 vCore
HW Quantity	3 7	2	2	2	2
Avg. OLTP transaction size	500 bytes (or less) based on interviews with MySQL customers				
OLTP data growth %/mo.	4% based on interviews with several dozen Backup-as-a-Service (BaaS) managed service providers (MSP) that backup MySQL				
OLAP Cloud Service	MySQL HeatWave	Redshift	vDW Standard Edition	BigQuery	Synapse
OLAP HW Infrastructure	Same as OLTP	dc2.8xlarge	XL	SLOTS	DWU Blocks
HW Quantity	3 7	2	Unspecified	200/300/400	500
ML Cloud Service	MySQL HeatWave ML	Redshift ML &SageMaker	N/A	BigQuery ML	Synapse ML & Spark pools
ML HW Infrastructure	Same as OLAP & OLTP	ml.m5.4xlarge	N/A	None	Large 16 vCPUs 128GB
HW Quantity	3 7	2	N/A	Unspecified	4 controllers
ETL Auto-Cloud Service	No ETL needed	Glue	Stitch		
Parameters	None needed	Transactional	Up to 100 million rows per month		
Backup Storage	Included	Not included in the TCO			
Faster Rev Acquisition	Not included in the TCO				
FTE Reduction	Not included in the TCO				