MySQL HeatWave

One MySQL Database service for transactions, real-time analytics, and Machine Learning—without ETL

March 2024
Copyright © 2024, Oracle and/or its affiliates
Public
Purpose statement

This document provides an overview of features and enhancements included in HeatWave. It is intended solely to help you assess the benefits of HeatWave and to plan your I.T. projects.

Disclaimer

This document in any form, software, or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Benchmark queries are derived from the TPC-H and TPC-DS benchmark, but results are not comparable to published TPC-H and TPC-DS benchmark results since they do not comply with the TPC-H TPC-DS specification.
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose statement</td>
<td>2</td>
</tr>
<tr>
<td>Disclaimer</td>
<td>2</td>
</tr>
<tr>
<td>Executive Summary</td>
<td>4</td>
</tr>
<tr>
<td>Challenges of Existing Data &amp; Analytics Solutions</td>
<td>4</td>
</tr>
<tr>
<td>Performance: Real Time Analytics</td>
<td>4</td>
</tr>
<tr>
<td>Mixed workload</td>
<td>5</td>
</tr>
<tr>
<td>Deployment Scenarios</td>
<td>6</td>
</tr>
<tr>
<td>HeatWave Architecture</td>
<td>8</td>
</tr>
<tr>
<td>Native MySQL Analytics</td>
<td>10</td>
</tr>
<tr>
<td>Scale-out Data Management</td>
<td>11</td>
</tr>
<tr>
<td>Real-time Elasticity</td>
<td>12</td>
</tr>
<tr>
<td>MySQL HeatWave Lakehouse</td>
<td>13</td>
</tr>
<tr>
<td><strong>MySQL Autopilot</strong></td>
<td>15</td>
</tr>
<tr>
<td>Auto Provisioning</td>
<td>17</td>
</tr>
<tr>
<td>Auto Shape Prediction</td>
<td>17</td>
</tr>
<tr>
<td>Auto Schema Inference</td>
<td>18</td>
</tr>
<tr>
<td>Adaptive data sampling</td>
<td>19</td>
</tr>
<tr>
<td>Autopilot Indexing</td>
<td>19</td>
</tr>
<tr>
<td>Auto Parallel Load</td>
<td>20</td>
</tr>
<tr>
<td>Auto Encoding</td>
<td>20</td>
</tr>
<tr>
<td>Auto Data Placement</td>
<td>21</td>
</tr>
<tr>
<td>AutoUnload</td>
<td>22</td>
</tr>
<tr>
<td>Auto Compression</td>
<td>22</td>
</tr>
<tr>
<td>Adaptive Data Flow</td>
<td>22</td>
</tr>
<tr>
<td>Auto Query Plan Improvement</td>
<td>22</td>
</tr>
<tr>
<td>Auto Query Time Estimation</td>
<td>23</td>
</tr>
<tr>
<td>Auto Change Propagation</td>
<td>23</td>
</tr>
<tr>
<td>Auto Scheduling</td>
<td>24</td>
</tr>
<tr>
<td>Adaptive Query Optimization</td>
<td>24</td>
</tr>
<tr>
<td>Auto Thread Pooling</td>
<td>25</td>
</tr>
<tr>
<td>Auto Error Recovery</td>
<td>25</td>
</tr>
<tr>
<td>MySQL HeatWave AutoML</td>
<td>25</td>
</tr>
<tr>
<td>Current challenges of Machine Learning in databases</td>
<td>25</td>
</tr>
<tr>
<td>HeatWave AutoML Approach</td>
<td>26</td>
</tr>
<tr>
<td><strong>JavaScript Support</strong></td>
<td>27</td>
</tr>
<tr>
<td>Integration with Oracle Cloud Services</td>
<td>27</td>
</tr>
<tr>
<td>Conclusion</td>
<td>28</td>
</tr>
</tbody>
</table>
Executive Summary

MySQL is the world’s most popular open source database because of its reliability, high-performance, and ease of use. It powers the world's most trafficked web sites including Facebook, Twitter, LinkedIn and Booking.com.

The companies that will thrive in the evolving digital landscape, will be those that make data, analytics, and machine learning a core part of their strategy. According to McKinsey, 92% of company leaders surveyed believed that their business model would not remain viable at the current rate of digitization. This fear of disruption is the leading driver behind the investment in modern data analytics and machine learning platforms.

MySQL HeatWave is a fully managed database service that lets developers quickly develop and deploy secure, cloud native applications using the world’s most popular open source database. MySQL HeatWave is the only MySQL cloud service with a built-in massively-scalable real-time query accelerator, and a fully automated in-database machine learning engine. This service overcomes the limitations of traditional data warehouse, analytics, and machine learning environments that use periodic long-running ETL batch jobs to refresh the data. MySQL HeatWave provides a unified MySQL cloud database service for transactions, real-time analytics across data warehouses and data lakes, and machine learning—without the complexity, latency, risks, and cost of ETL duplication.

Challenges of Existing Data & Analytics Solutions

MySQL is optimized for OLTP, but it is not designed for analytic processing (OLAP). As a result, organizations that need to efficiently run analytics need to move their data to a separate analytics database.

This approach of moving data to another database introduces complexity and additional costs to customers in multiple ways:

1. **Applications need to define complex logic** for extracting relevant data from MySQL.
2. **The extracted data needs to be transported to another database** across networks securely, consuming network bandwidth and incurring latency.
3. **Data in the other database needs to be manually kept in sync** with the MySQL database and as a result, the data on which analytics is performed can become stale.
4. **Additional cost and overhead of managing multiple databases** for running OLTP and analytics applications.

Performance: Real Time Analytics

HeatWave is designed to enable customers to run analytics on data which is stored in MySQL databases, or in object storage, without the need for ETL.

“We successfully migrated our 6TB database and in-house digital marketing and media management applications from AWS Aurora to MySQL HeatWave on OCI that reduced our costs by 60 percent and improved performance for complex queries by more than 1000x and overall workloads improved 85 percent.”

Amit Palshikar
Co-Founder, CTO
Red3i
This service is built on an innovative, in-memory analytics engine that is architected for scalability and performance. This results in a very performant solution for SQL analytics at a fraction of the cost of other solutions.

Compared to other MySQL solutions, with a 4TB TPC-H Benchmark workload, HeatWave is much faster and less expensive:

Compared to Amazon Aurora, MySQL HeatWave offers dramatic improvements in performance for complex and analytic queries. MySQL HeatWave is 1,400x faster than Amazon Aurora, delivering 2,200X better price-performance. Furthermore, with HeatWave there is no need to create indexes on the base table, which can take days with Amazon Aurora. As a result, the data is available to query much sooner with HeatWave than with Aurora.

**Mixed workload**

Most real-world applications have a mix of OLTP and complex OLAP queries. For such workloads, MySQL HeatWave is much faster and costs a fraction of Amazon Aurora. Using the industry standard CH-benchMark on a 100 GB dataset for OLAP queries, Amazon Aurora is 18X slower, provides 110X less throughput, and is 35% more expensive than MySQL HeatWave. For OLTP queries, Amazon Aurora has the same performance as MySQL HeatWave and is 35% more expensive.

Amazon Redshift, which is designed for analytics, is offered in multiple shapes. With their latest instance shape (ra3.4xlarge) and latest advanced query accelerator (AQUA) turned on, HeatWave is 4X faster with 10X better price-performance as demonstrated by a 10TB TPC-H benchmark.

“We found MySQL HeatWave improved performance by 10 times and significantly dropped our costs after migrating from AWS Aurora. We also did not have to modify our application for a great experience.”

Kanami Suzuki
Developer
Fan Communications
Also, unlike Amazon Redshift, MySQL HeatWave is capable of both OLTP and OLAP, without the need for ETL.

Customers who use MySQL HeatWave will benefit from significantly better performance and eliminating the need for ETL. They get support for real-time analytics, a reduced monthly cost, and a single database for OLTP, OLAP, and machine learning.

Similarly, as demonstrated by a 10TB TPC-H benchmark, MySQL is 4X faster than Snowflake with 15X better price-performance.

“MySQL HeatWave dramatically reduced our AWS Aurora and Redshift cost by more than 50 percent, we are no longer moving data around so now we have blazing fast, real-real-time insights with no effort. More importantly, scalability has made our expansion plan possible, allowing us to onboard more data and new clients without impact to costs. It’s a dream come true.”

Pablo Lemos
Co-Founder, CTO
Tetris.co

Deployment Scenarios

MySQL HeatWave is a fully managed service. The MySQL database has been enhanced to natively integrate HeatWave. As a result, customers who store data in MySQL can seamlessly run analytics and machine learning by enabling this service.

A MySQL HeatWave instance is a cluster composed of a MySQL instance and multiple HeatWave nodes. When HeatWave is enabled, the HeatWave server is installed on the MySQL node. It is responsible for cluster management, loading data into the memory of the HeatWave nodes, query scheduling, and query execution. MySQL applications written in Java, PHP, Ruby, etc. work seamlessly with HeatWave using standard MySQL ODBC/JDBC connectors. HeatWave is an in-memory accelerator and supports all MySQL syntax. Hence, all existing tools and applications built using standard SQL will work without requiring any modification to queries.
Existing tools and applications work seamlessly with HeatWave.

Data that is needed for analytic processing is stored in the memory of the HeatWave nodes, in a hybrid columnar compressed format. The number of nodes needed to run a workload depends on the amount of data present for analytic processing, the compression factor that is achieved on the dataset, and the query characteristics. The number of nodes needed can be automatically derived by using the Auto Provisioning advisor which is available with HeatWave.

HeatWave supports 2 shapes and up to 64 nodes per cluster for workload up to 64TB of data (for data originating from the MySQL database):

1. HeatWave.32GB – This shape has 32GB of memory and can process up to 50GB of data per node
2. HeatWave.512GB – This shape has 512GB memory and can process up to 1TB of data per node

64TB is the approximate maximum amount of data which can be populated in the memory of the HeatWave nodes at a given moment. There is no limit to the amount of data which can be stored in the MySQL database and customers can choose which tables or columns from the MySQL database schema to load into the memory of HeatWave nodes. If the tables are no longer needed by queries, they can be removed from memory to make room for other data.

HeatWave provides a great solution for customers who need to run transactional and analytical workloads. While transactional queries are run in the MySQL node, data updated in MySQL InnoDB is transparently propagated to the HeatWave cluster in real-time for accelerated analytical processing. This enables customers to run OLTP and real-time analytics workloads simultaneously within a single database platform.

“We recently migrated our production workload from another cloud solution to MySQL HeatWave. Doing so reduced our cost by 3x and it also significantly accelerated many of our queries which were taking a long time before.”

Chien Hoang
Director of Engineering
Tamara.co
Single MySQL database for OLTP and OLAP applications, no ETL required.

For customers with a multi-cloud strategy, MySQL HeatWave is available on OCI, AWS, and Azure, as well as in customers’ data centers with OCI Dedicated Region. Customers can take advantage of MySQL HeatWave on the cloud platform that they prefer.

MySQL HeatWave is optimized to deliver best price performance on OCI, AWS ans Azure respectively

On-premises customers who cannot move their MySQL deployment to a cloud due to compliance or regulatory requirements, can still leverage HeatWave by using the hybrid deployment model. In such a hybrid deployment, customers can leverage MySQL replication to replicate on-premises MySQL data to HeatWave without the need for ETL.

Hybrid deployment for enabling analytics on data stored on premise

**HeatWave Architecture**

There are seven key architectural choices that lead to a compelling performance and cost advantage with HeatWave:

1. **Innovative in-memory hybrid columnar analytics engine** designed for scalability and performance which implements state of the art algorithms.

2. **Optimized for the cloud** to provide the best price performance database service based on commodity hardware.

3. **HeatWave scale out data management layer** enabling the HeatWave representation of data to be stored in OCI object storage (or Amazon S3),
which allows fast data reload for operations like error recovery, maintenance, and system restart to increase service uptime.

4. **MySQL Autopilot** provides machine learning-powered automation that improves the performance, scalability, and ease of use of HeatWave. It automates the database lifecycle operations including provisioning, data loading, query processing, and error handling. It also provides capabilities for OLTP workloads.

5. **HeatWave AutoML** provides familiar MySQL interfaces enabling MySQL users to build and train machine learning models, as well as to generate inferences and explanations without the need to have extensive ML expertise.

6. **HeatWave Lakehouse** is designed to process and query hundreds of terabytes of data in the object store in a variety of file formats (CSV, Parquet, and exports from Aurora/Redshift/MySQL) without loading the data into MySQL.

7. **MySQL-Javascript** supports JavaScript stored programs in MySQL Heatwave (limited availability). This enables developers to develop rich procedural logic inside the database and access their MySQL datasets seamlessly.

The Heatwave engine uses a columnar in-memory representation that facilitates vectorized processing, leading to very good query performance. The data is encoded and compressed prior to being loaded in memory. This compressed and optimized in-memory representation is used for both numeric and string data. This results in significant performance speed-up and reduced memory footprint, which translates to reduced cost for customers.

![Vectorized in-memory columnar representation for analytic processing](image)

One of the key design points of the HeatWave engine is to massively partition data across a cluster of HeatWave nodes, which can be operated in parallel in each node. This enables high cache hits for analytic operations and provides very good inter-node scalability. Each HeatWave node within a cluster and
each core within a node can process partitioned data in parallel, including parallel scans, joins, group-by, aggregation and top-k processing.

HeatWave has implemented high performance algorithms for distributed in-memory analytic processing. Joins within a partition are processed fast by using vectorized build and probe join kernels. The highly-optimized network communication between analytics nodes is achieved by using asynchronous batch I/Os. The algorithms are designed to overlap compute time with the communication of data across nodes, which helps achieve good scalability.

HeatWave implements state of art algorithms for distributed in-memory analytic processing.

**Native MySQL Analytics**

The integration of HeatWave with MySQL provides a single data management platform for OLTP, OLAP, and ML. HeatWave is designed as a MySQL pluggable storage engine, which completely shields all the low-level implementation details at the storage level from the end users. As a result, users can manage both HeatWave and MySQL with the same management tools, including the OCI console, REST APIs, and the command line interface.

Since HeatWave is an in-memory processing engine, data is persisted in the MySQL InnoDB storage engine. This allows users to manage analytics data the same way they manage transactional data in MySQL.

Users and applications interact with HeatWave through the MySQL database node in the cluster. Users connects to HeatWave through standard tools and standard-based ODBC/JDBC connectors. HeatWave supports the same ANSI SQL standard and ACID properties as MySQL and supports diverse data types. This enables existing applications to take advantage of HeatWave without any changes to their application, allowing easy and quick integration.

Once users submit a query to the MySQL database, the MySQL query optimizer transparently decides if the query should be offloaded to the HeatWave cluster for accelerated execution. This is based on whether all operators and functions referenced in the query are supported by HeatWave and if the estimated time to process the query with the HeatWave engine is less than with MySQL. If both conditions are met, the query is pushed to
HeatWave nodes for processing. Once processed, the results are sent back to the MySQL database node and returned to users.

MySQL Integration for Query Processing

Any updates to the tables in InnoDB that are also loaded in HeatWave are automatically propagated to the memory of the HeatWave nodes in real-time. This allows subsequent queries to always have access to the latest data. This is done behind the scenes by a light-weight change propagation algorithm that can keep up with MySQL data update rates.

Scale-out Data Management

The MySQL InnoDB storage engine stores data in a row-based format, while HeatWave stores data in memory in a hybrid columnar format. Loading data from MySQL to HeatWave involves transforming data into the HeatWave columnar format, which can take time depending on data size. Since data is stored in memory in HeatWave, operations like error recovery, maintenance and system restart can take a long time as data needs to be re-transformed and reloaded after the cluster is ready.

To improve service uptime, HeatWave introduced a new storage layer that is built on OCI Object Storage, or Amazon S3 when using MySQL HeatWave on AWS. This new architecture enables storing HeatWave formatted data in a persistent storage, allowing reload of data in constant time regardless of data size.
The above diagram shows the new HeatWave architecture with the scale-out storage layer. In the HeatWave storage layer, persisted data is organized in the same way as that of in-memory data. Each HeatWave node can restore data independently and in parallel, allowing a very fast and near constant time data reload.

**Real-time Elasticity**

With the HeatWave storage layer, each HeatWave node can load data independently and in parallel based on different data sizes and different cluster sizes. This enables HeatWave to provide a flexible, predictable, and performant online cluster scaling capability that allows users to resize their HeatWave cluster when their workload and data demand change.

The HeatWave cluster can be scaled up or down to any number of nodes. During the scaling operation, the cluster continues to work, there is no interruption in client connections for read or write and there is minimal effect on query performance in HeatWave. For sizing up, data is loaded from the HeatWave storage layer to the newly created nodes. Once the data is loaded, HeatWave updates its metadata and queries. They will then be processed in all the nodes in the new cluster size. For sizing down, additional data is loaded from the HeatWave storage layer to the HeatWave nodes that will be kept for the new reduced cluster size. Once the additional data is loaded, HeatWave updates its metadata and queries, and they will be processed using the reduced cluster size. For both operations, data is balanced across all nodes automatically without user intervention to ensure optimal query performance.

The below chart shows how fast and predictable the data loading time is based on different cluster sizes, and the neglectable time (in microseconds) that the HeatWave cluster needs to update its metadata.
HeatWave real-time elasticity performance

**MySQL HeatWave Lakehouse**

We are facing a huge growth in data stored outside of databases (social media files, data from IoT sensors…etc.) that businesses want to use to rapidly generate new insights.

MySQL HeatWave includes MySQL HeatWave Lakehouse, letting users process and query half a petabyte of data in the object store—in a variety of file formats, such as CSV, Parquet, Avro, Aurora/Redshift export files, and on-premises MySQL export files. Customers can leverage the benefits of HeatWave even when their data is not stored inside a MySQL database. With MySQL HeatWave Lakehouse, MySQL HeatWave provides one service for transaction processing and analytics across data warehouses and data lakes, and machine learning—without ETL across cloud services. The HeatWave scales to 512 nodes to process data in object storage.
Faster than Snowflake, Amazon Redshift, Databricks, and Google BigQuery

As demonstrated by a 500 TB TPC-H benchmark, the query performance of MySQL HeatWave Lakehouse is 15X faster than Amazon Redshift, 18X faster than Snowflake, 18X faster than Databricks, and 35X faster than Google BigQuery. The load performance of MySQL HeatWave Lakehouse is 9X faster than Amazon Redshift, 8X faster than Google BigQuery, 6X faster than Databricks, and 2X faster than Snowflake, as demonstrated by the 500 TB TPC-H benchmark.

Fast analytics on all data

Customers can query transactional data in MySQL databases, data in various formats in object storage, or a combination of both using standard MySQL commands. Querying the data in the database is as fast as querying data in the object store, as demonstrated by 10 TB and 30 TB TPC-H benchmarks.
Machine learning on all data

Customers can train, predict, and explain their machine learning models on data loaded from object storage. HeatWave AutoML uses a common set of APIs to train, predict, and explain a model, irrespective of whether the data is in the lakehouse or in the database. This simplifies tasks for the user as they have a single, unified API to perform machine learning.

To learn more: read the MySQL HeatWave Lakehouse technical brief

MySQL Autopilot

MySQL Autopilot automates many of the most important and often challenging aspects of achieving high query performance at scale - including provisioning, data loading, query execution and failure handling. It uses advanced techniques to sample data, collect statistics on data and queries, and build machine learning models to model memory usage, network load and execution time. These machine learning models are then used by MySQL Autopilot to execute its core capabilities. MySQL Autopilot makes the HeatWave query optimizer increasingly intelligent as more queries are executed, resulting in continually improving system performance over time. MySQL Autopilot also provides capabilities to improve the performance and price-performance of OLTP workloads.

MySQL Autopilot focuses on four aspects of the service lifecycle: system setup, data load, query execution, and failure handling.

MySQL Autopilot automates different aspects of the service to improve performance, scalability and usability of the system

System Setup

1. **Auto provisioning** predicts the number of HeatWave nodes required for running a workload by adaptive sampling of table data on which analytics is required. This means that customers no longer need to manually estimate the optimal size of their cluster.

2. **Auto shape prediction** continuously monitors the OLTP workload, including throughput and buffer pool hit rate, to recommend the right
compute shape at any given time—allowing customers to always get the best price-performance.

3. **Auto schema inference** automatically infers the mapping of file data to data types in the database. As a result, customers don’t need to manually specify the mapping for each new file to be queried by MySQL HeatWave Lakehouse, saving time and effort.

4. **Adaptive data sampling** intelligently samples files to derive information needed for automation and the nature of the data in question. Using these novel techniques, MySQL Autopilot can scan and propose schema predictions on a set of data files totalling 500 TBs in under one minute.

**Data Load**

1. **Autopilot indexing (currently in limited availability)** recommends the right set of indexes for columns in order to improve OLTP query performance. It balances cost, storage space, and performance by adding or removing indexes.

2. **Auto parallel load** monitors the MySQL workload, identifies queries that can be accelerated by HeatWave, and automatically loads relevant tables into HeatWave to maximize query acceleration. It also optimizes load time and memory usage by predicting the optimal degree of parallelism for each table being loaded into HeatWave.

3. **Auto encoding** determines the optimal representation of columns being loaded into HeatWave taking queries into consideration. This optimal representation provides the best query performance and minimizes the size of the cluster to minimize the cost.

4. **Auto data placement** predicts the column on which tables should be partitioned in-memory to achieve the best performance for queries. It also predicts the expected gain in query performance with the new column recommendation.

5. **Auto unload** automatically unloads unused or rarely used tables in HeatWave, and predicts the memory saved from unloading these tables.

6. **Auto Compression** dynamically determines which compression algorithms to use for each column based on its data characteristics. This enables it to provide the optimal memory usage and query performance.

7. **Adaptive data flow** learns and coordinates network bandwidth utilization to the object store across a large cluster of nodes, dynamically adapting to the performance of the underlying object store, resulting in optimal performance and availability.

**Query Execution**

8. **Auto query plan improvement** learns various statistics from the execution of queries and improves the execution plan of future queries. This improves the performance of the system as more queries are run.

9. **Auto query time estimation** estimates the execution time of a query prior to executing the query, allowing quick tryout and testing on different queries.

10. **Auto change propagation** intelligently determines the optimal time when changes in MySQL Database should be propagated to the
HeatWave storage layer. This ensures that changes are being propagated at the right optimal cadence.

11. **Auto scheduling** determines which queries in the queue are short-running and prioritizes them over long-running queries in an intelligent way to reduce overall wait time.

12. **Adaptive Query Execution** dynamically adjusts the query execution plan based on runtime statistics to improve query execution time.

13. **Auto thread pooling** lets the database service process more transactions for a given hardware configuration, delivering higher throughput for OLTP workloads and preventing it from dropping at high levels of transactions and concurrency.

**Failure Handling**

14. **Auto error recovery**: Provisions new nodes and reloads necessary data from the HeatWave storage layer if one or more HeatWave nodes are unresponsive due to software or hardware failure.

**Auto Provisioning**

Auto Provisioning provides a recommendation on how many HeatWave nodes are needed to run a workload. When the service is started, database tables on which analytics queries are run need to be loaded to the HeatWave cluster memory. The size of the cluster needed depends on the tables and columns required to load, and the compression achieved in memory for this data. The diagram below compares the traditional (i.e., manual) approach to estimating the cluster size with Auto Provisioning. In traditional provisioning, users need to guess a cluster size. Underestimation results in data load or query execution failure due to space limitations. Overestimation results in additional costs for unneeded resources. As a result, users iterate until they determine the right cluster size, and this size estimate becomes inaccurate when tables are updated.

The right side of the above diagram shows how auto provisioning, a ML-based cluster size estimation advisor, solves this problem. By leveraging well-trained and accurate ML models, the user consults the auto provisioning advisor to obtain the right cluster size for their dataset. As a result, users do not need to guess the cluster size. Later, if the customer data grows or additional tables are added, the users can again take advantage of the auto provisioning advisor.

**Auto Shape Prediction**
To alleviate the burden of experimenting with different MySQL shapes to determine the most performant shape for a given workload, Auto Shape Prediction provides suggestions for the right MySQL server shape, based on highly accurate predictions from machine-learning models inside the MySQL server and the most recent query execution metrics and traces. Since Auto Shape Prediction continuously collects workload execution statistics, it can adapt to the evolving workload patterns and hence provide suggestions based on the most recent workload.

**Auto Schema Inference**

Auto Schema Inference automatically predicts the table definitions for the data stored in object storage. This is valuable, because the schema for the data stored in object storage needs to be defined in order for HeatWave to perform data loading and query processing. It accurately infers the data type and its precision, for structured or semi-structured data, even when metadata is not present in the files (for example, CSV). Auto Schema Inference generates the DDL for the table definition, which can be easily reviewed and modified. It makes it easy to load data into HeatWave from various data sources where the schema is not known in advance. The below example shows a sample DDL output from Auto Schema Inference for TPC-H linetime table.
Adaptive data sampling
To minimize the time it takes to collect statistics for hundreds of terabytes of data stored in object storage, Adaptive Data Sampling intelligently samples a fraction of the data and uses advanced machine learning models to accurately predict the statistics for the whole data set. This enables other MySQL Autopilot features such as Auto Schema Inference to use the statistics to predict the table definition for the data in a fraction of the time. As shown in the table below, Auto Schema Inference can infer table definitions for 400TB of data in under 50 seconds.

Autopilot Indexing
MySQL Autopilot Indexing (currently in limited availability) is an ML-based technology designed to optimize database systems for better cost and performance.
With Autopilot Indexing, database administrators no longer need to manually identify which indexes are most beneficial for their workload. Autopilot Indexing automatically generates secondary index recommendations to create or drop indexes based on the current workload. Autopilot Indexing considers both the query performance and the cost of maintaining the indexes when generating recommendations. It provides performance and storage estimations, as well as explanations for the recommendations it generates. The Autopilot Indexing interface consists of a simple and intuitive console that customers can use to view and analyze the projected performance and storage impact of recommended index suggestions. This makes it easy to foresee the impact of changes to the database systems before applying the suggestions.

For more information, read the Autopilot Indexing Technical Brief

**Auto Parallel Load**

Loading data into HeatWave involves several manual steps. The time required to perform these steps depends on the number of schemas, tables, and columns, and statistics. Auto parallel load automates these steps by using machine learning models to predict the degree of parallelism per table needed to achieve optimal load speed and memory usage.

Auto parallel load continuously monitors workloads run in MySQL, and uses machine learning models to predict the queries that can be offloaded to HeatWave for query acceleration. Based on the list of queries, auto parallel load dynamically loads or unloads tables to the HeatWave cluster. This occurs without any user intervention, allowing the amount of data loaded into HeatWave to dynamically adapt to the ever-changing workload. Now, to improve performance, queries are automatically offloaded to HeatWave without any manual setup or maintenance. This reduces work for the DBA and provides developers and data analysts using MySQL HeatWave a transparent and completely hands-off experience.

**Auto Encoding**

HeatWave supports two string column encoding types: variable-length and dictionary. The type of encoding affects the query performance as well as the supported query operations. It also affects the amount of memory required for HeatWave nodes. By default, HeatWave applies variable-length encoding to string columns when data is loaded, which may not be the optimal encoding
choice for query performance and cluster memory usage for certain workloads.

Auto encoding provides recommendations for string columns that reduce memory usage and help improve query performance. The above diagram shows the difference between default encoding and auto encoding. In the default case, the variable-length encoding ensures the best query offload capability. However, this can impact ideal query performance due to data movement between HeatWave nodes and MySQL nodes. Auto encoding uses machine learning to analyze column data, HeatWave query history, and available MySQL node memory to identify which string columns can be coded in dictionary encoding. When the suggestion is applied, the overall query performance is improved due to reduced data movement in the system, and HeatWave memory usage is reduced due to efficient (i.e., smaller) dictionary codes and the corresponding dictionaries that maintain the mapping between the strings and the codes reside in the memory of the MDS node.

**Auto Data Placement**

Data placement keys are used to partition table data when loading tables into HeatWave. Partitioning table data by JOIN and GROUP BY key columns can improve query performance by avoiding costs associated with redistributing data among HeatWave nodes at query execution time.

Determining the best data placement key is a tedious task requiring understanding query access patterns and system behavior. Moreover, picking wrong partitioning keys can lead to sub-optimal performance due to increased data distribution costs during query execution time.

The diagram above depicts a comparison between default query execution and execution with auto data placement. Based on machine learning models,
auto data placement recommends appropriate data placement keys by analyzing table statistics and HeatWave query history and provides an estimation of query performance improvement. Once the suggestion is applied, query performance is improved by minimizing the data movement between nodes during execution.

**Auto Unload**

Users can save costs by unloading tables that are never or rarely queried in HeatWave to lower the total memory required and decrease the HeatWave cluster size. However, this requires understanding all queries and access patterns of the tables. Auto unload tracks table usage and automatically unloads unused tables, while providing an explanation to support each action taken.

**Auto Compression**

Data stored in HeatWave is compressed using different compression algorithms. To minimize memory usage while providing the best query performance, auto compression dynamically determines the compression algorithm to use for each column based on its data characteristics.

Auto compression employs an adaptive sampling technique during the data loading process, and automatically selects the optimal compression algorithm without user intervention. The algorithm selected is based on the compression ratio and the compression and decompression rates, which balance the memory needed to store the data in HeatWave with query execution time.

**Adaptive Data Flow**

Adaptive data flow learns and coordinates network bandwidth utilization to the object store across a large cluster of nodes, dynamically adapting to the performance of the underlying object store, resulting in optimal performance and availability.

**Auto Query Plan Improvement**

Auto query plan improvement enhances query performance by improving query plan statistics based on previous query executions. By maintaining more accurate query statistics, HeatWave creates better query plans and makes better decisions on the underlying physical operators; consequently improving the overall query performance.

Query 2 benefits from statistics of a previous similar query (query 1) with auto query plan improvement.
The above shows how auto query plan improvement works without user intervention. After a query (Q1) executes on HeatWave, auto query plan improvement collects and stores the cardinalities of all operations in the query execution plan (e.g., scan, join, group by). When a similar (or identical) query arrives (Q2), the system checks whether it can take advantage of the previously collected statistics information for Q2. If the system determines a similarity between the two query plans, a better query plan is generated based on statistics information from Q1. In doing so, it improves query performance and cluster memory usage significantly.

**Auto Query Time Estimation**
Users are often interested in accurate query time estimates before running the query. Such functionality allows users to estimate their application performance better and to understand the resources needed. Auto query time estimation not only provides user-visible estimations for query run times, but it also uses the same building blocks internally to improve query performance by optimizing query (sub-)plans.

Instead of using static, analytical models, auto query time estimation integrates a data-driven query time estimation module, which improves as queries run. To do so, HeatWave leverages load- and run-time statistics and dynamically tunes query cost models during execution. As a result, auto query time estimation improves with time as more queries are executed on the system.

**Auto Change Propagation**
Data updated in MySQL is propagated and persisted to the HeatWave data layer as change logs. During data reload, HeatWave first restores data from the base data, then applies the data from the change logs. Over time, the persisted change log volume increases, which can result in an increased reload time as all the change logs need to be applied to the base data. So, the change logs are consolidated from time to time to alleviate increased reload latency. However, determining when to consolidate is not an easy task, which depends on several factors such as transaction rate, system load, and failure probability.

To minimize consolidation time during reloading from the storage layer, auto change propagation uses data-driven mechanism to determine the best
change propagation interval and choice. Auto change propagation analyzes the rate of changes, incoming DMLs, object storage resources, and previously seen change activity. As a result, the changes are propagated at the best time interval, which results in optimized consolidation time for critical system operations.

**Auto Scheduling**

Traditional database systems process queries based on their arrival time, which can result in long-running queries starving short-running queries.

![Traditional database system vs HeatWave auto scheduling](image)

On the left, is a sub-optimal case where three queries (Q1, Q2, Q3) from three user sessions arrive one after the other and are scheduled in the FIFO order. After the execution completes, one can identify that waiting time for Q3 could be reduced significantly with minimal impact on Q2 latency.

On the right, it shows how auto scheduling improves the user experience for short running queries in a multi-session application. Auto scheduling identifies and prioritizes short-running queries by automatically classifying queries into short or long queries using HeatWave data driven algorithms. Therefore, Q3 is prioritized before Q2 as Q3 is identified as a short-running query.

Auto Scheduling reduces the elapsed time for short-running queries significantly when the multi-session applications consist of a mix of short and long running queries. It also ensures long-running queries are not penalized and are not postponed indefinitely.

**Adaptive Query Optimization**

Adaptive query optimization automatically improves query performance and memory consumption, and mitigates skew-related performance issues as well as “out of memory” issues. It uses various statistics to adjust data structures and system resources after query execution has started—independently optimizing query execution for each node based on actual data distribution at runtime. This helps improve the performance of ad hoc queries by up to 25%.

The HeatWave optimizer generates a physical query plan based on statistics collected by Autopilot. During query execution, each HeatWave node executes the same physical plan. With adaptive query execution, each individual HeatWave node adjusts the local physical query plan based on statistics such as cardinality and distinct value counts of intermediate relations collected locally in real-time. This allows each HeatWave node to tailor the data
structures it needs, resulting in better query execution time, lower memory usage, and improved data skew-related performance.

**Auto Thread Pooling**
With Auto Thread Pooling, MySQL HeatWave prioritizes not only peak single-thread performance but also high throughput in the presence of concurrent clients running concurrent queries on a MySQL server. With this feature, the MySQL server now can perform workload-aware admission control of the incoming transactions. It eliminates the resource contention created by too many awaiting transactions, automatically queuing them to maximize performance while sustaining the throughput in the face of high concurrency.

**Auto Error Recovery**
HeatWave automatically provisions new HeatWave node(s) when a hardware or software failure is detected on a node. When the cluster is restored, auto error recovery automatically reloads the data only to the re-provisioned node(s), allowing a very fast recovery.

*Note: Auto thread pooling and auto shape prediction in MySQL Autopilot are available in MySQL HeatWave on AWS, and will be available soon on OCI.*

**MySQL HeatWave AutoML**

**Current challenges of Machine Learning in databases**
Developing and using machine-learning models requires skill sets in topics such as:

- Candidate algorithms/models to select from
- Hyperparameters that need to be tuned per algorithm
- Features to engineer and select from
- Data preprocessing approach per data type
- Drift detection and retraining
- Knowledge of Python, as most ML algorithm frameworks are available only in Python
Even with the above expertise, users still need to extract data out of the database to train and test the model, which leads to trust and security issues.

**HeatWave AutoML Approach**

HeatWave AutoML is an in-database set of machine learning capabilities that enable users to train ML models, and generate inference and explanations across data stored in MySQL databases and data lakes. All machine learning activities are performed inside the database, meaning there’s no need to extract data out of the database to perform ML. It brings several advantages:

- **Fully Automated**: HeatWave AutoML fully automates the creation of tuned models, generating inferences and explanations, thus eliminating the need for the user to be an expert ML developer
- **SQL interface**: Provides the familiar MySQL interface for invoking machine learning capabilities
- **Security and Efficiency**: Data and models never leave the MySQL Database. Clients or any other services never see the data or models stored in the DB service
- **Explanations**: All models created by HeatWave AutoML can be explained. Enterprises have a growing need to explain the predictions of machine learning models to build trust, demonstrate fairness, and comply with regulatory requirements.
- **Performance and Scalability**: The performance of HeatWave AutoML is much better at a lower cost than competing services such as Redshift ML. Furthermore, HeatWave AutoML scales with the size of the cluster.
- **Easy Upgrades**: HeatWave AutoML leverages state-of-the-art open-source Python ML packages that enable continual and swift uptake of newer (and improved) versions.
- **Supported Models**: HeatWave AutoML supports multiple model types such as Classification, Regression, Time Series Forecasting, Anomaly Detection, Recommender System, etc. This enables organizations to use HeatWave AutoML for many different types of business use cases.

All these capabilities are available to MySQL HeatWave customers without any additional charge.

HeatWave AutoML is a native, in-database solution, eliminating the cost, complexity and risk of ETL

To learn more, read the MySQL HeatWave AutoML Technical Brief
**JavaScript Support**

JavaScript support (limited availability) enables rich procedural programming capability directly inside the database, further enabling the user to cut down on data movement costs in favor of server-side solutions. The feature allows users to write JavaScript stored functions and procedures in the server that are executed via GraalVM. The JavaScript functions and procedures can manipulate existing MySQL data irrespective of the underlying storage engine (i.e. InnoDB, HeatWave, or Lakehouse, all work transparently).

Users can now reorganize applications and move data-intensive complex operations close to their data. This reduces the cloud egress cost and the effort required to maintain data pipelines. In addition, it improves end-to-end application performance and security by eliminating the need for client-server data movement.

**Integration with Oracle Cloud Services**

OCI offers a wide range of services for data analytics, machine learning, and data lake. Native integration with these services makes it easier for existing applications to use HeatWave.

Oracle Analytics Cloud (OAC) provides the industry’s most comprehensive cloud analytics in a single unified platform, including self-service visualization and inline data preparation to enterprise reporting, advanced analytics, and self-learning analytics that deliver proactive insights. Integration with OAC provides a BI visualization platform for users to analyze their MySQL data.

OCI Data Integration Service provides extract, transform and load (ETL) capabilities to target data warehousing scenarios on the OCI platform. It supports various data sources, starting with relational, cloud, and Hadoop. Integration with OCI Data Integration allows users to easily transform and import data from data sources other than MySQL to HeatWave, expanding the scope of data that can take advantage of HeatWave.
Conclusion

MySQL HeatWave is the only cloud service that combines transactions, real-time analytics across data warehouses and data lakes, and machine learning in one MySQL database—without the complexity, latency, risks, and cost of ETL duplication. It delivers unmatched performance and price-performance. HeatWave AutoML enables native, in-database machine learning, allowing users to build, train, deploy, and explain machine learning models inside MySQL, without machine learning expertise. MySQL Autopilot provides machine learning-powered automation that improves the performance, scalability, and ease of use of HeatWave, saving developers and DBAs significant time. JavaScript support (limited availability) enables rich procedural programming capabilities directly inside the database, and enables developers to write JavaScript stored functions and procedures in MySQL HeatWave that are executed via GraalVM. MySQL HeatWave can be deployed in OCI, AWS, Azure, in a hybrid environment, and in customers’ data centers with OCI Dedicated Region.