TiDB

WHITE PAPER

Azure Database for MySQL vs. TiDB

A Guide to Scaling Cloud-Native Applications

Contents

Introduction

Relational Database Management Systems (RDBMSs) are the cornerstone of modern applications, particularly those with demanding requirements for data integrity, security, and transactional consistency.

For complex applications serving industries such as SaaS, fintech, gaming, and most digital-native business, RDBMSs provide robust transaction support, enforcing the ACID properties of Atomicity, Consistency, Isolation, and Durability. Their sophisticated query capabilities, typically through SQL, allow for intricate data retrieval and reporting, which is essential for decision-making processes.

MySQL and TiDB have emerged as popular database options for developers and organizations looking for reliable and scalable database solutions. Many teams choose cloud-hosted MySQL services such as Azure Database for MySQL for their simplicity, built-in management, and MySQL's familiar ecosystem.

But as data volume, concurrency, and uptime requirements grow, these managed MySQL offerings start to reveal their limits. Because they're still single-node, monolithic systems at their core, scaling beyond the boundaries of a single instance often means:

- Manual sharding and complex replication setups
- Higher operational costs to maintain HA across multiple instances
- Performance degradation as workloads compete for limited resources
- Separate infrastructure for transactional and analytical workloads

TiDB changes that equation. It combines full MySQL compatibility with a cloud-native, distributed SQL architecture that delivers seamless horizontal scaling, built-in high availability, and mixed workload processing, all without sacrificing ACID guarantees. For teams already invested in MySQL syntax and tooling, TiDB is the natural next step when Azure MySQL's vertical-scaling ceiling becomes a business bottleneck.

In this white paper, we compare MySQL and TiDB to provide deeper insights into the unique paradigms they represent within the database industry. As the volume of data and concurrent user access increases, which database management system you choose has profound implications on the scalability, maintainability, and performance of modern applications. Knowing the architectural differences, strengths, and limitations of each system can guide stakeholders in making informed decisions that align with their specific use cases.

Understanding TiDB and MySQL

First released in 1995, MySQL is a popular open-source RDBMS that has played a pivotal role in the development of modern web applications. It was developed to handle large databases much faster than existing solutions at the time. It was designed to be open source from the start, which contributed to its widespread adoption.

In 2003, MySQL made a significant leap in enterprise adoption with the release of MySQL 4.0, which introduced InnoDB, a storage engine providing ACID-compliant transaction features. Throughout its history, MySQL has continued to evolve, adding features like stored procedures, triggers, and views, while maintaining a strong following and substantial market share. MySQL remains a cornerstone of web development, widely used by individuals, organizations, and large-scale enterprises.

What is TiDB?

TiDB, powered by PingCAP, is an open-source, distributed SQL database that features horizontal scalability, strong consistency, and high availability. It's designed to support both traditional RDBMS workloads and the more demanding real-time data workloads of the future. TiDB's design is inspired by Google's Spanner and F1 databases. However, it aims to provide similar capabilities without the need for a globally-distributed file system.

Since its inception, TiDB has quickly gained traction in the global tech community for its scalability and MySQL compatibility, which allows for easy migration from MySQL. New versions of TiDB continually enhance the database's capabilities in performance, ease of use, and stability. TiDB has been adopted by numerous companies around the globe for mission-critical applications due to its ability to handle large-scale data with the consistency and durability required by high-volume transactions.

Monolithic vs. Distributed Architecture

MySQL and TiDB represent two different classes of database management systems (DBMS). Each has distinct approaches to storing historical data and handling database operations. Their core architectural differences stem from the traditional monolithic structure of MySQL versus the distributed design of TiDB.

MySQL's monolithic architecture was designed to run on a single server with a storage engine to handle database operations. The most commonly associated storage engine is InnoDB, which provides ACID-compliant transaction features and utilizes a transaction log, a double-write buffer, and undo logs for handling transactions and maintaining

maintaining historical data. The transaction log records all changes made to the data, which is essential for point-in-time recovery and rollbacks. MySQL's monolithic architecture means that all components of the DBMS—including query processing, transaction management, and storage—are tightly integrated within a single node.

This architecture excels in simplicity and is generally easier to manage and deploy. However, it also means that scaling out (i.e., adding more nodes to distribute the load) is more challenging, as MySQL was not designed to natively distribute data across multiple nodes. Scaling is typically achieved through replication and sharding.

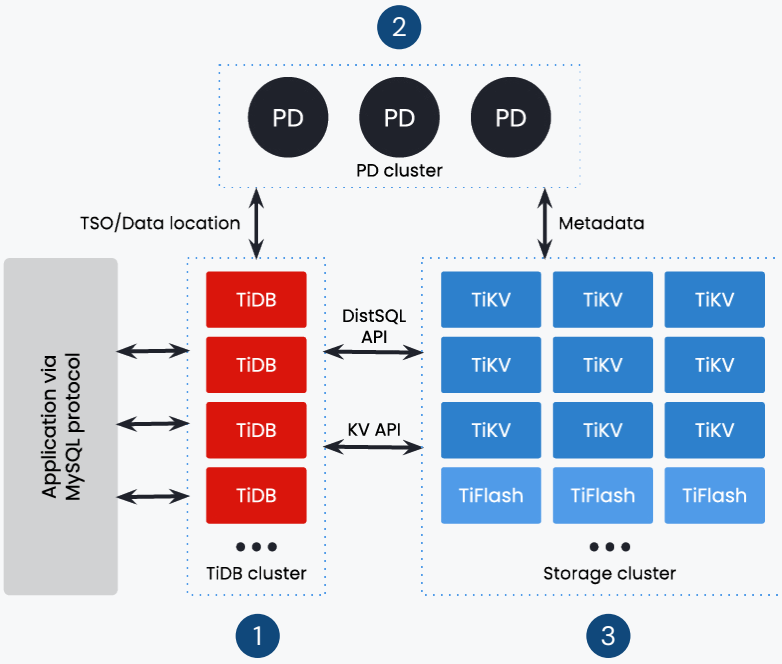
However, these methods can add complexity and come with performance, consistency, and maintainability trade-offs.

In contrast, TiDB is a distributed SQL database designed to support horizontal scalability, strong consistency, and high availability. TiDB's architecture decouples the computation from storage, allowing each component to scale out independently. The TiDB cluster consists of three main components, as shown in the below diagram:

1. **TiDB server**: A stateless SQL layer that's MySQL compatible and decouples compute from storage to simplify scaling. Each node can take both reads and writes.

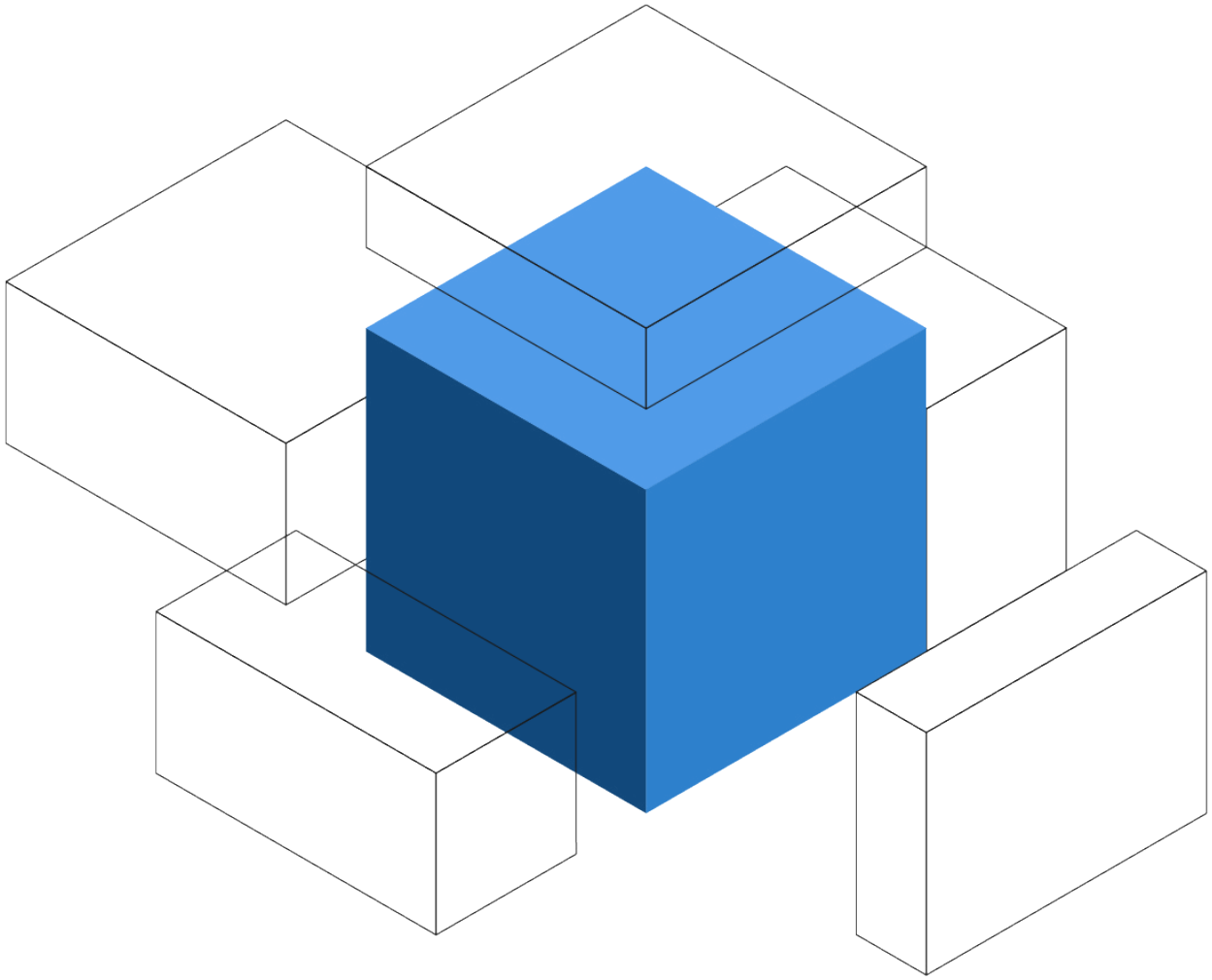
2. **Placement Driver (PD) server**: A cluster manager that dynamically balances the data load in real time while mitigating hotspots and providing for the implementation of customized scheduling policies.

3. **Storage server**: A distributed storage layer that offers built-in high availability and strong ACID consistency that can auto-scale to hundreds of nodes and petabytes of data.



The distributed nature of TiDB allows it to easily scale out by adding more nodes. The data is automatically sharded into small chunks and distributed across storage nodes. This sharding is transparent to the user and can handle massive amounts of data far beyond what a single node can manage. This design enables TiDB to better support cloud-native applications that require global data distribution and high availability.

MySQL's monolithic structure makes it simpler but less scalable than TiDB's distributed architecture. TiDB can handle horizontal scaling natively, managing historical data across nodes in a way that supports consistent, distributed transactions with increased flexibility in deployment models.



Comparing TiDB and MySQL: Key Features and Capabilities

When evaluating TiDB and MySQL, it's crucial to understand the distinct features and capabilities each brings to the table. These characteristics directly impact their suitability for various operational needs and objectives. Below is a feature-by-feature comparison that delineates how each database system addresses key functionalities.

Feature/Capability	TiDB	MySQL
Horizontal Scalability	Native support, scales seamlessly across multiple nodes	Not native, relies on sharding or other strategies
Architecture	Native distributed SQL, compute-storage decoupling	Monolithic, traditional RDBMS architecture
MySQL Compatibility	Strong, supports MySQL protocol and similar syntax	Originator, fully compatible by default
ACID Compliance	Yes, across distributed environments	Yes, within traditional single-node setups
Robust Transaction Support	Distributed transactions with strong consistency	Comprehensive transaction support with immediate consistency
Established User Base	Growing, with increasing global adoption	Vast, with long-standing industry presence
Performance and Scalability	Designed to perform well as it scales out	Needs replication/sharding to scale, performance returns typically diminish with scale
Operational Complexity	Higher by default. Lower at scale	Lower by default. Higher at scale
Ecosystem and Tooling	Shares the MySQL ecosystem, also has tooling focused on distributed SQL	Mature, with extensive tools and community support
Cloud-Native Support	Strong alignment, offered as DBaaS or integrates with Kubernetes for self-managed	Can be adapted, requires additional configurations

TiDB Features and Capabilities

- **Horizontal scalability**: TiDB is designed for horizontal scalability, enabling it to distribute data and workload across multiple nodes seamlessly. This capability is particularly beneficial for applications that anticipate growth in user base and data volume, as it allows for scaling without significant architectural changes.

- **Native distributed SQL architecture**: The architecture of TiDB is inherently distributed, decoupling compute from storage, which allows each to scale independently. This design is advantageous for cloud-native applications that require elasticity and high availability.

- **MySQL compatibility**: TiDB offers strong compatibility with MySQL, supporting the same protocol and a similar syntax. This compatibility ensures a smooth transition for applications migrating from MySQL and allows for the continued use of existing SQL expertise and tools.

- **ACID compliance**: TiDB is ACID compliant. This ensures reliable transaction processing and maintaining data integrity and consistency across its distributed environment.

- **Robust transaction support**: TiDB provides distributed transaction support with strong consistency. It can handle transactions with reliability comparable to traditional RDBMS systems.

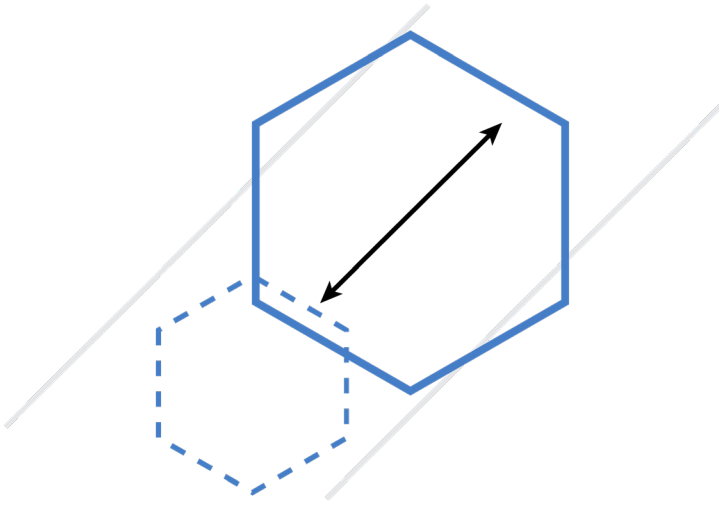
MySQL Features and Capabilities

- **ACID compliance**: MySQL is known for its ACID compliance, ensuring transactional integrity and data consistency, a feature that is essential for applications where data accuracy is critical.

- **Established user base**: With a long history in the industry, MySQL has a vast and established user base. This has led to a rich ecosystem of tools, extensive documentation, and a large community for support.

- **Robust transaction support**: MySQL's transaction support is comprehensive, with features like consistent nonlocking reads, full ACID compliance, and immediate consistency, making it a reliable choice for a wide range of applications.

- **MySQL protocol compatibility**: As the originator of the protocol, MySQL's widespread adoption has made its protocol a standard in the industry. TiDB's wire protocol compatibility allows users to take advantage of a broad array of existing tools and integrations.



Detailed Feature-by-Feature Comparison

- **Performance and scalability**: TiDB is natively designed for horizontal scalability, allowing it to handle increasing loads by distributing them across multiple servers. MySQL, while not horizontally scalable by design, often relies on increasing server capacity (i.e., vertical scaling) or implementing complex replication and sharding strategies to handle growth.

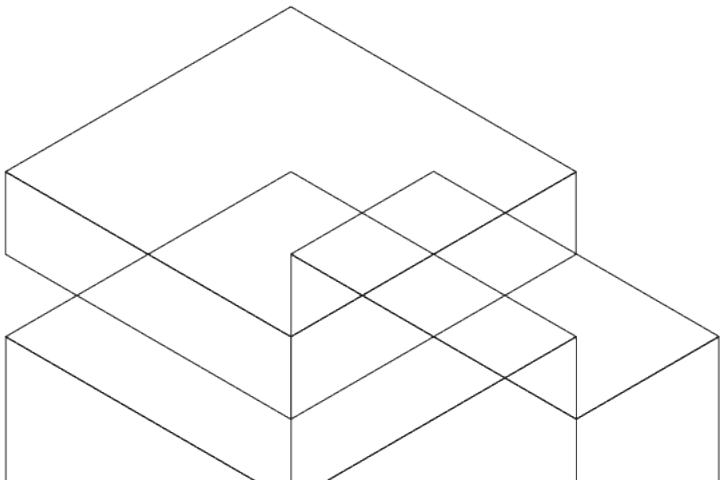
- **Data consistency and ACID compliance**: Both TiDB and MySQL are committed to data consistency and ACID compliance. TiDB extends these principles to a distributed architecture for transactional integrity across multiple nodes and data centers.

- **Operational complexity**: The distributed nature of TiDB introduces complexities in deployment and management but offers scalability and resilience. MySQL's operational complexity is initially lower due to its monolithic architecture but may require additional considerations for achieving high availability. Scaling MySQL horizontally introduces significant complexity in the form of replication and sharding. Performance returns from scaling MySQL typically diminish with each incremental expansion.

- **Ecosystem and tooling**: MySQL's mature ecosystem is a product of its longevity and widespread use. While TiDB shares much of the MySQL ecosystem, due to its native MySQL compatibility, TiDB is also growing its own ecosystem, with tools and integrations increasingly supporting distributed systems and cloud-native applications.

- **Cloud-native support**: TiDB's design aligns well with cloud-native principles, offering TiDB as a cloud DBaaS option as well as out-of-the-box integration with Kubernetes and other cloud technologies. MySQL is offered by multiple Cloud vendors as a DBaaS option and can also be adapted to private cloud environments with additional tooling and configurations.

While TiDB and MySQL offer robust transaction support and ACID compliance, they differ significantly in their scalability approach and architecture. These differences are pivotal when selecting a database system, as they will influence the operational capabilities, scalability potential, and alignment with future industry trends and business growth.



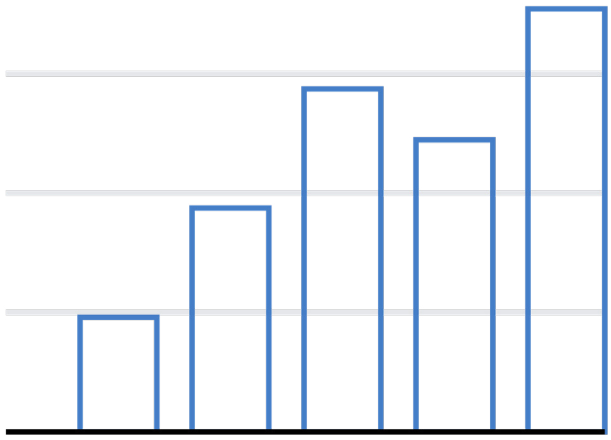
Comparing Database Scalability

Scalability is a critical consideration in database management, particularly for businesses that anticipate rapid growth or experience unpredictable workloads.

TiDB has been architected with these challenges in mind, offering significant advantages when it comes to handling high-concurrency workloads and managing massive data sets. Its horizontal scalability allows for the distribution of data across a cluster of machines. This facilitates the processing of a high volume of transactions and queries in parallel.

This design is particularly beneficial for applications that require both the transactional consistency of traditional RDBMS and the scalability typically associated with NoSQL systems. TiDB's ability to add nodes to a cluster without downtime is a testament to its suitability for dynamic, data-intensive environments.

On the other hand, MySQL's performance characteristics are well-understood in the context of its monolithic architecture, which is optimized for a single-node setup. While MySQL delivers excellent performance on a single server, it faces limitations as demand grows. Scaling MySQL typically involves vertical scaling, which has its limits, or implementing sharding and read replicas, which can introduce complexity and potential consistency issues. Although MySQL can handle high-concurrency to a certain extent, it may require additional engineering efforts to achieve the level of scalability inherent to TiDB's distributed architecture.



Comparing Database Reliability

Reliability in database systems encompasses several key aspects, including data consistency, high availability, fault tolerance, and disaster recovery. These factors are crucial for businesses that rely on their database to provide uninterrupted service and safeguard their data against failures and data loss.

Data Consistency Models in TiDB and MySQL

TiDB ensures transactional consistency across its distributed database system by employing the Percolator transaction model with a two-phase commit protocol. Each transaction receives a globally unique timestamp from a Timestamp Oracle (TSO) for serializability. When a transaction begins, it locks the affected data and writes intents, which are essentially promises to perform certain actions. These locks prevent conflicts by ensuring that no other transactions can alter the data until the transaction is complete.

In the first phase of the commit, write intents are placed; in the second, after all locks are verified, the transaction is committed and all changes are applied atomically. If a transaction fails or is aborted, TiDB rolls back any changes and cleans up the locks to maintain a consistent state. For reads, TiDB leverages multi-version concurrency control (MVCC), allowing transactions to read a consistent snapshot of the database without interference from concurrent write operations. This meticulous protocol ensures that TiDB maintains ACID compliance.

MySQL, with its default InnoDB storage engine, also provides a strong data consistency model with ACID compliance. It also uses a combination of locking and MVCC to maintain consistency, which works well within the confines of a single node or across tightly coupled replication setups.

Disaster Recovery vs. High Availability

Disaster recovery (DR) and high availability (HA) are two distinct strategies for ensuring business continuity in IT environments.

High availability is designed to minimize downtime by providing a failover solution that allows for near-instantaneous switching to a redundant system or network in the event of a failure. This approach often involves clusters of servers where if one fails, another can take over immediately, thus maintaining the service's availability.

Disaster recovery, on the other hand, is a broader strategy that focuses on the ability to restore systems, applications, and data to full functionality after a catastrophic event, such as natural disasters, system crashes, or data corruption. Disaster recovery solutions typically involve keeping backup data and system images in geographically separate locations, allowing an organization to rebuild its IT infrastructure from scratch if necessary. While high availability deals with immediate failover to prevent downtime, disaster recovery is concerned with the restoration of IT operations after significant disruptive incidents.

Disaster Recovery Comparison

Both TiDB and MySQL place a strong emphasis on disaster recovery.

MySQL's source-replica replication is a mechanism where changes made on one database server (the source) are duplicated to one or more other servers (the replicas). This is achieved by recording changes in the source's binary log, which are then read by the replica. The replica connects to the source, pulling the changes from the binary log and applying them to itself via its own relay log to maintain a consistent state. This process supports various configurations and types of replication, including filtering specific data, asynchronous and semi-synchronous replication, and even multi-source replication setups. It is a key feature for load balancing, ensuring high availability, and facilitating backup and disaster recovery strategies.

TiDB's distributed SQL environment relies on TiCDC for its DR implementation. TiCDC, short for TiDB Change Data Capture, is a data synchronization tool in the TiDB ecosystem. It captures and replicates incremental data changes from a TiDB cluster in real-time. By pulling change logs from the underlying storage engine, TiKV, TiCDC can provide a consistent view of the data at any given point in time. It supports multiple destinations for data replication, including MySQL-compatible databases, Kafka, and S3-compatible storage. TiCDC enables seamless integration and synchronization of data between TiDB and other systems. This makes it a vital component for building data pipelines and enabling real-time analytics.

Both TiDB and MySQL offer robust features to provide DR capabilities. However, they approach these challenges from different architectural perspectives. TiDB's ecosystem employs a sophisticated Change Data Capture architecture which can be used for replication both inside the ecosystem as well as to third-party data stores. On the other hand, the MySQL DR subsystem is much less complex in its implementation. It is also very mature and has been battle tested around the globe for many years. When selecting a database, it is essential to consider the specific reliability requirements of the application and operational complexity that the business is prepared to manage.

High Availability, Fault Tolerance, Data Integrity, and Self-Healing

MySQL achieves high availability through the use of InnoDB, a default storage engine that provides ACID-compliant transaction support, along with Group Replication (GR), a plugin that enables the creation of fault-tolerant systems with multiple equal peers automatically synchronized. GR complements this by adding a consensus algorithm. This algorithm manages a group of

MySQL servers that work together, replicating data within a group to multiple servers effectively and in real-time.

This setup allows for transactions committed on one server to be replicated to all others, ensuring that each server has an up-to-date copy of the database. If one server fails, others can take over, providing high availability. The consensus capabilities of GR are provided by a technology called Group Communication System (GCS) protocol, which is an implementation of the Paxos algorithm. The combination of InnoDB's robust data handling and GR's synchronized data distribution provides a resilient infrastructure where MySQL can automatically handle failover and ensure continuous service with minimal downtime.

The limits of scalability for GR become visible at the point where scaling the GR cluster, by adding nodes, begins to negatively impact write latency. The fundamental cause for the increase in response time is the implementation of the GR consensus algorithm (GCS) at the database server level. GCS requires a majority of the cluster members to certify that they have received the transaction. Consequently, as the number of cluster members increases, so does the number of members needed to make up a majority. This can have the effect of causing commit latency to increase.

This is not a problem for workloads that are static in nature, mostly read only, or that implement sophisticated data lifecycle management (DLM) strategies to keep the data volume consistent. However, DLM strategies can be complex and costly to maintain. As a result, most organizations will find the scalability limitations of GR too restrictive and overly cumbersome for workloads with moderate to high growth patterns. It's important to note that the scalability and performance implications of GR will vary greatly depending on the specific use case and operational environment.

TiDB achieves high availability through an architecture that separates compute from storage and a distributed storage tier that employs the Raft consensus algorithm. By decoupling compute (SQL processing) from storage (data persistence), TiDB ensures that the compute layer can scale out independently from the storage layer, allowing for more flexible resource management and improved fault tolerance.

TiDB's storage layer is inherently distributed, meaning that data is spread, and replicated, across multiple nodes. This reduces the risk of a single point of failure and enables the system to maintain operations even if some nodes fail. The Raft consensus algorithm is central to this architecture; it is used to replicate data across multiple nodes in the storage layer, ensuring consistency and data reliability. This design allows TiDB to automatically recover from hardware failures, network issues, and other unexpected incidents.

MySQL and TiDB present robust solutions for achieving high availability in database management. Each database uses a distinct approach that leverages the strengths of their respective architectures. MySQL's combination of InnoDB and Group Replication provides a traditional yet powerful model, which scales

within limits. TiDB's innovative architecture separates compute from storage, enabling scalability and resilience, while its use of the Raft consensus algorithm across a distributed storage tier ensures consistent data replication and system robustness. Both systems exemplify the advances in database technology that cater to the demands of modern applications.

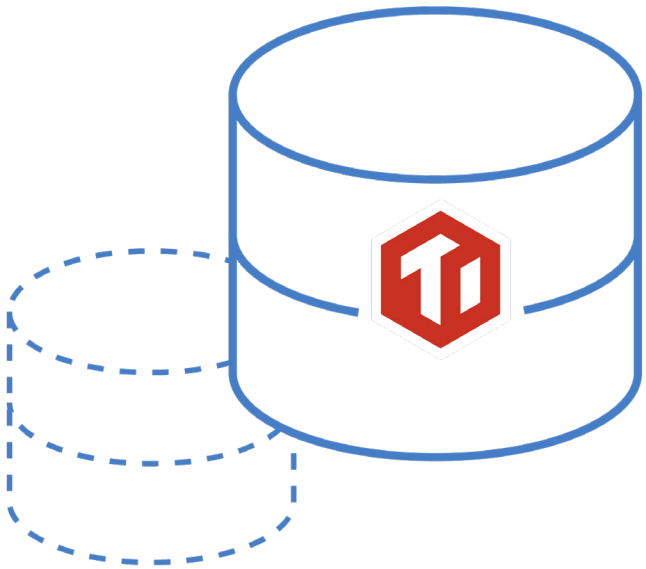
Comparing Database Versatility

The versatility of a database system is a measure of its ability to adapt to various workloads, environments, and operational requirements. This adaptability is crucial for businesses that need to pivot quickly or that have workloads which do not fit neatly into traditional transactional or analytical categories.

Cloud-Native, Distributed Architecture vs. Single-Node, Monolithic Architecture

TiDB's cloud-native, distributed architecture is designed to provide scalability and resilience in cloud environments. It is engineered to work seamlessly with cloud services and container orchestration systems like Kubernetes, which makes it well-suited for modern, cloud-based applications. This distributed nature allows TiDB to spread across multiple cloud instances, as it can handle large-scale workloads and provide the flexibility to scale up or down as needed.

MySQL's architecture, in contrast, is monolithic and traditionally operates on a single node. While this has the advantage of simplicity and is well-suited for a wide range of applications, it may not be as naturally adaptable to cloud-native environments that require distributed computing resources. However, with the introduction of cloud service providers and orchestration tools, MySQL is also deployed in the cloud through multiple DBaaS services. However, this involves more overhead and complexity to ensure high availability and scalability. The primary drawback to MySQL in cloud-native deployments is its inability to scale horizontally.



Streamlined Tech Stack vs. Increasing Technical Complexity

TiDB offers a streamlined tech stack that simplifies the management of large and complex data sets. Its integrated solution for both transactional and analytical processing reduces the need for additional middleware or complex configurations. This can result in a reduction of the operational burden and a more straightforward scaling process.

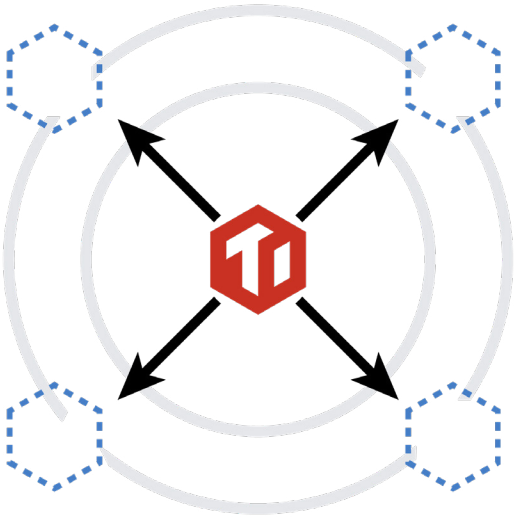
MySQL, while starting out with a relatively simple stack, can become more complex as the need for scaling and high availability increases. Implementing manual sharding methods, clustering, or sophisticated replication to achieve these goals can introduce additional layers of complexity into the tech stack. This can increase the maintenance overhead and the need for specialized expertise.

Mixed Workload Processing

One of the key features of TiDB is its ability to efficiently process mixed workloads, a concept also known as Hybrid Transactional/Analytical Processing (HTAP). TiDB's architecture allows it to handle both online transaction processing (OLTP) and online analytical processing (OLAP) within the same system. This eliminates the need for separate systems for transactional and analytical workloads, streamlining operations and reducing latency when accessing analytical data.

MySQL is traditionally used for OLTP workloads. And while it can perform analytical queries, it may not be as efficient as dedicated OLAP systems or HTAP-capable systems like TiDB. For complex analytical queries, especially those involving large data sets, MySQL may require additional tools or integrations with external data warehouses.

TiDB's cloud-native, distributed architecture and HTAP capabilities position it as an ideal choice for businesses. It's a database for a wide range of scenarios that can also scale for future growth. MySQL, with its single-node, monolithic architecture, offers simplicity and reliability, but may require additional solutions to meet the demands of highly scalable, cloud-native, or mixed workload environments.



Comparing Database Deployment and Management

The deployment and ongoing management of a database are critical factors that can significantly influence operational efficiency and resource allocation. Both TiDB and MySQL offer a range of deployment options, but they differ in their setup complexity and management requirements.

Deployment Options and Ease of Setup

TiDB is designed to be cloud native. This means it can be deployed across various cloud platforms, private data centers, and hybrid environments. Its architecture is particularly suited for containerized environments, and it integrates smoothly with Kubernetes, which simplifies deployment, scaling, and management operations. TiDB also provides the TiDB Operator, a Kubernetes extension that automates mundane operational tasks, further easing the deployment process.

MySQL, being one of the most popular databases in the world, offers a wide range of deployment options, including cloud services, on-premises installations, and a variety of packaged distributions. The ease of setting up a MySQL database is well-known; it can be as simple as installing a package on a server and performing basic configuration. However, setting up a highly-available, scalable MySQL environment can be more complex. This often involves additional components such as replication managers, load balancers, and sharding tools.

Management, Maintenance, and Operational Considerations

Management and maintenance of TiDB involves monitoring the performance and health of various components, including TiDB servers, TiKV stores, and the Placement Driver (PD). While TiDB's distributed nature provides high availability and resilience, it also introduces cluster management and tuning complexity. However, TiDB aims to reduce this complexity with automation tools and dashboards that provide a central view of the cluster's status and performance metrics.

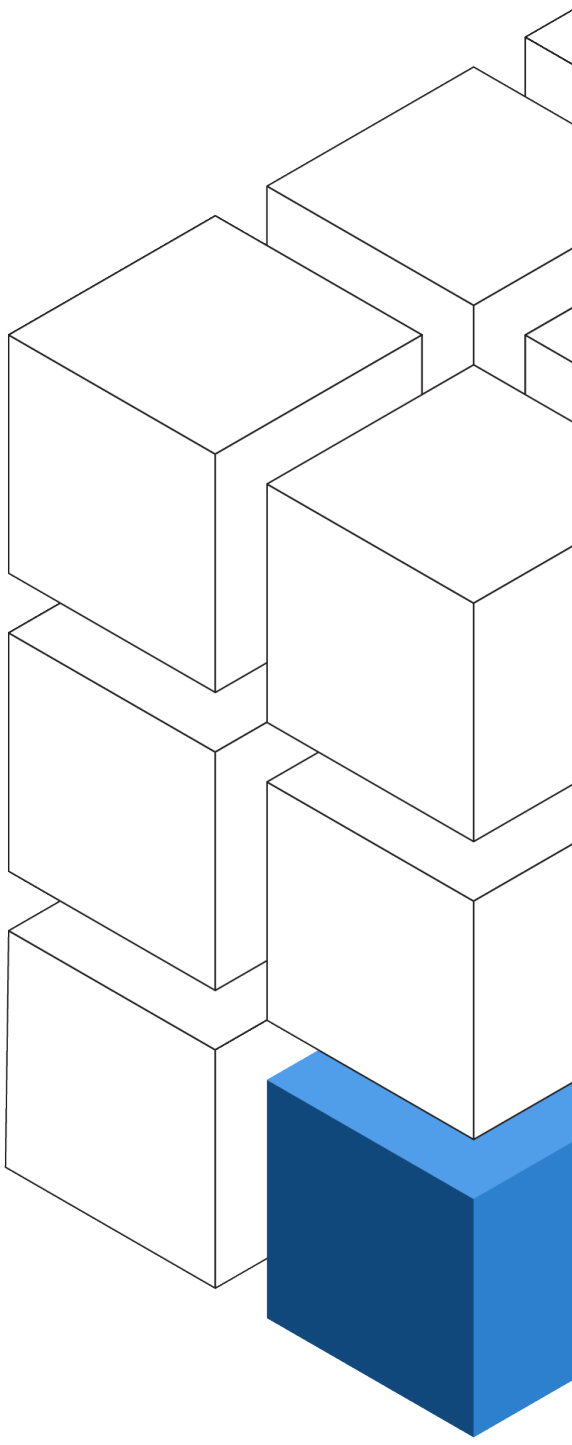
MySQL management is generally well-understood due to its maturity and extensive community knowledge base. Routine maintenance tasks for MySQL include backups, index optimization, and query performance tuning. While individual MySQL server maintenance is straightforward, managing a scaled-out MySQL environment with replication or sharding can be more challenging. This requires careful planning and execution of maintenance operations to avoid downtime.

Commercial Support Options and Resources

For organizations that require enterprise-level support, both TiDB and MySQL offer commercial support options.

TiDB, powered by PingCAP, offers enterprise support that includes technical assistance, performance tuning, upgrades, and troubleshooting. PingCAP also provides training and consulting services to help organizations optimize their TiDB deployment and operations. MySQL, now owned by Oracle, has a range of commercial support options through Oracle's MySQL Support. This includes 24/7 technical assistance, access to MySQL engineers, and proactive support. Additionally, there is a wealth of third-party support services and consultants specializing in MySQL due to its widespread adoption.

TiDB and MySQL offer flexible deployment options and comprehensive support resources. TiDB's cloud-native design and tooling cater to modern, distributed environments and aim to simplify operational complexity. MySQL is renowned for its ease of setup and has a vast array of support options due to its long-standing presence in the industry. The choice between the two systems will depend on the specific deployment scenarios, the scale of operations, and the level of expertise available within the organization.



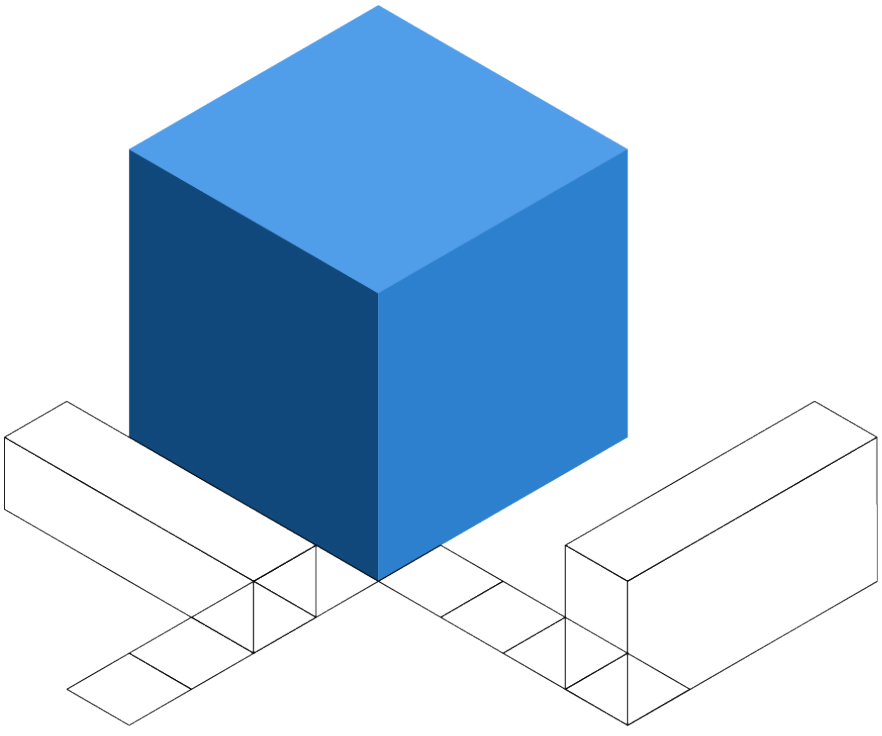
Conclusion

The choice between TiDB and cloud-hosted MySQL services like Azure MySQL reflects a broader shift in database strategy from traditional, single-node architectures to modern, distributed systems built for cloud scale.

Cloud-hosted MySQL services offer a low-friction entry point for MySQL workloads, but its scaling model comes with operational and cost trade-offs: sharding complexity, multiple instance management, and infrastructure sprawl as you grow. TiDB removes these barriers by delivering unlimited horizontal scale, native high availability, and mixed workload support in a single platform. It does all this while speaking the same MySQL protocol your applications already use.

If your team is already comfortable with MySQL but your workloads are pushing the limits of cloud-hosted MySQL services, TiDB is the logical next step. It lets you preserve the MySQL compatibility you rely on while gaining the cloud-native scale, resilience, and efficiency your future demands.

Want to explore how leading companies are adopting TiDB for horizontal scalability and high availability? Check out our Customer Story page for detailed case studies.



About TiDB

TiDB, powered by PingCAP, unlocks limitless scale for data-intensive businesses. Our advanced distributed SQL database enables leading enterprises, SaaS, and digital native companies to build petabyte-grade clusters while managing millions of tables, concurrent connections, frequent schema changes, and zero-downtime scaling. Large organizations who have adopted TiDB, such as Databricks, Pinterest, and Plaid, are finally free to focus on their future growth instead of complex data infrastructure management. With AI-driven innovations and multi-cloud flexibility, TiDB offers unmatched agility, resilience, and security. For more information, please visit TiDB.io.