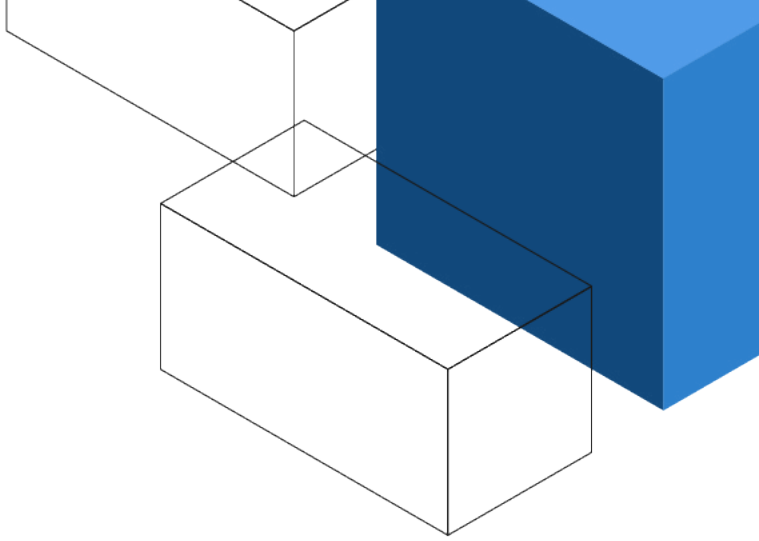
TiDB

WHITE PAPER

Maximizing Rewards Programs with TiDB

How to Provide Customers with a More Engaging Shopping Experience That Keeps Them Coming Back

Contents



Introduction

The e-commerce industry has undergone a seismic shift over the past decade, fundamentally changing how consumers shop and how businesses operate. This rapid expansion is fueled by the growing consumer preference for the convenience of online shopping and the proliferation of mobile devices.

Among the key strategies for success in this competitive landscape is the implementation of rewards programs. These programs are not just a tool for customer retention—they are a powerful driver of repeat purchases, higher customer spending, and frequent interactions. By maximizing rewards programs, businesses can differentiate themselves from the competition, foster deeper customer loyalty, and create a more engaging shopping experience that keeps customers coming back.

This white paper explores how TiDB, an open-source, distributed SQL database, can enhance the value of these rewards programs and other related use cases. TiDB offers a robust, scalable solution that supports growth and operational excellence in the digital marketplace.

Workload Details

A rewards points application is a critical component for any e-commerce platform, as it directly influences customer satisfaction and engagement. The workload for these applications is typically read-intensive, requiring systems that can efficiently handle high volumes of queries. As these businesses scale rapidly, the ability to support this growth with robust, scalable infrastructure becomes essential. Consistency is another crucial factor—any discrepancies in rewards points can lead to confusion, frustration, and a loss of trust among customers. Ensuring accurate and real-time updates to point balances is key to maintaining a seamless customer experience.

To illustrate the importance of this application, consider the following scenario: Imagine a digital wallet with two key tables—the Current Balance (CB) and Events, which logs debit and credit transactions. When a new wallet is created, and a user earns 100 points, the Events table records this as a credit, while the CB table updates the balance to reflect 100 points. This entire process is completed within a single database transaction to ensure consistency.

Later, when the user spends five points, the Events table logs a debit, and the CB table adjusts the balance to 95 points—again, all within a single transaction. The Events table can then be used to generate detailed account statements, while the CB table provides the current balance, crucial for determining whether a user has sufficient points to make a purchase. This seamless integration between transactions and balance management is important to foster trust and ensure the smooth operation of rewards programs.

Workload Types

This rewards points application functions as a classic online transaction processing (OLTP) application, managing tasks such as adding points, spending them, and retrieving balance information. It must handle large volumes of data while ensuring durability and maintaining high throughput. The system is also designed to operate with minimal latency, providing a seamless experience even as the demands on it grow. This combination of speed, reliability, and scalability is essential for supporting the real-time needs of these programs.

Workload Characteristics

- **Data Volume**: These applications typically manage large datasets, ranging between 15-20 TBs, to efficiently handle the extensive records generated by loyalty transactions.
- **Throughput**: The system is designed to support high activity levels, with an expected throughput of approximately 8,000-9,000 write queries per second (QPS) and 18,000-20,000 QPS.
- **Latency**: To ensure a seamless user experience, the application maintains low latency, with writes completed within 200ms and reads within 50ms.
- **Data Variety**: The application primarily processes structured data, focused on performing heavy CRUD operations to accurately credit and debit points within user wallets.

Technical Requirements

To effectively support rewards programs, the underlying system must meet several stringent technical requirements:

System Reliability

- **High Availability**: Rewards program applications must guarantee continuous data accessibility, with no single point of failure that could disrupt service. Since these applications directly interact with end users, any downtime can lead to customer frustration and dissatisfaction, particularly when users are unable to redeem or earn points during online activities.
- **Data Consistency**: Maintaining strict data consistency is critical, as these applications handle financial transactions where any discrepancies can lead to significant consequences, including financial loss and damage to the brand's reputation.
- **Automatic Fault Recovery**: The system should be equipped with automatic fault detection and recovery mechanisms to minimize downtime and ensure seamless business continuity, keeping the platform resilient and reliable under any circumstances.

System Scalability

- **Elastic Scaling**: The system must dynamically scale resources up or down to match fluctuations in user traffic and data processing demands. For example, during peak periods like festivals or sales promotions, rapid scaling is essential to handle the surge in requests. Once these high-demand periods end, the system should seamlessly scale down to optimize resource usage and cost efficiency.
- **Disaster Recovery (DR) Requirements**: High-usage applications must include robust DR setups to ensure that, in the event of a disaster, operations can swiftly shift to a secondary site. This capability is crucial for minimizing downtime and maintaining continuous service availability, even under unexpected circumstances.

Architecture

The initial architecture of a wallet application began with a single shard, utilizing a master MySQL node paired with a hot standby for high availability and a read replica dedicated to processing read queries. However, as the application experienced rapid growth, both data volume and throughput surged, necessitating further sharding, as shown in the below diagram.

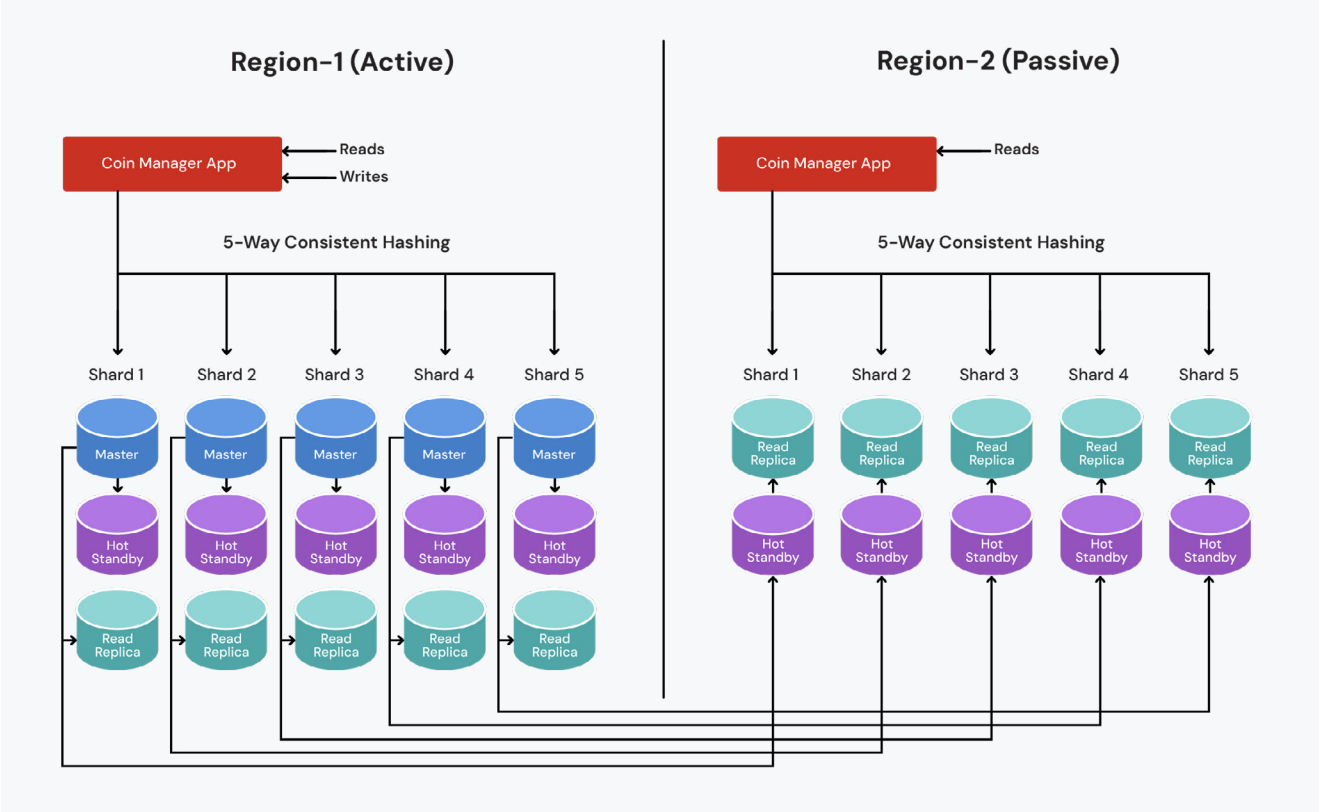


Figure 1: Original architecture of a wallet application with 25 MySQL database shards

To accommodate this growth, five additional shards were created, expanding the infrastructure to 15 MySQL databases. To meet internal compliance requirements, a disaster recovery (DR) setup was also implemented, mirroring these five shards in a passive DR cluster. This expansion introduced significant complexity into the architecture, requiring meticulous management of multiple databases across primary and DR environments.

Architecture Challenges

In the end, this complex architecture led to several significant challenges:

- **25 Independent MySQL Database Instances**: Managing 25 separate database instances resulted in a substantial increase in maintenance efforts, requiring extensive time and resources to ensure smooth operation.
- **20 Replication Channels**: The complexity of monitoring and troubleshooting network connections escalated, particularly during service interruptions, making it increasingly difficult to maintain system stability.
- **Day-to-Day Operational Complexity**: The need to oversee multiple replication channels and database instances demanded a larger administrative team and sophisticated monitoring tools, significantly complicating daily operations.
- **Failover and Resharding Complexities**: Resharding emerged as one of the most daunting tasks. Any reconfiguration of the sharding logic, or the addition or removal of nodes, required meticulous planning and often led to expected downtimes, disrupting business applications and causing potential service interruptions.

Proposed Changes

To eliminate this complexity, the 25 MySQL nodes were replaced with TiDB to simplify the architecture. After adopting TiDB, this architecture was simplified, reducing the overall complexity by 90%. Here's a diagram depicting what this new architecture looks like.

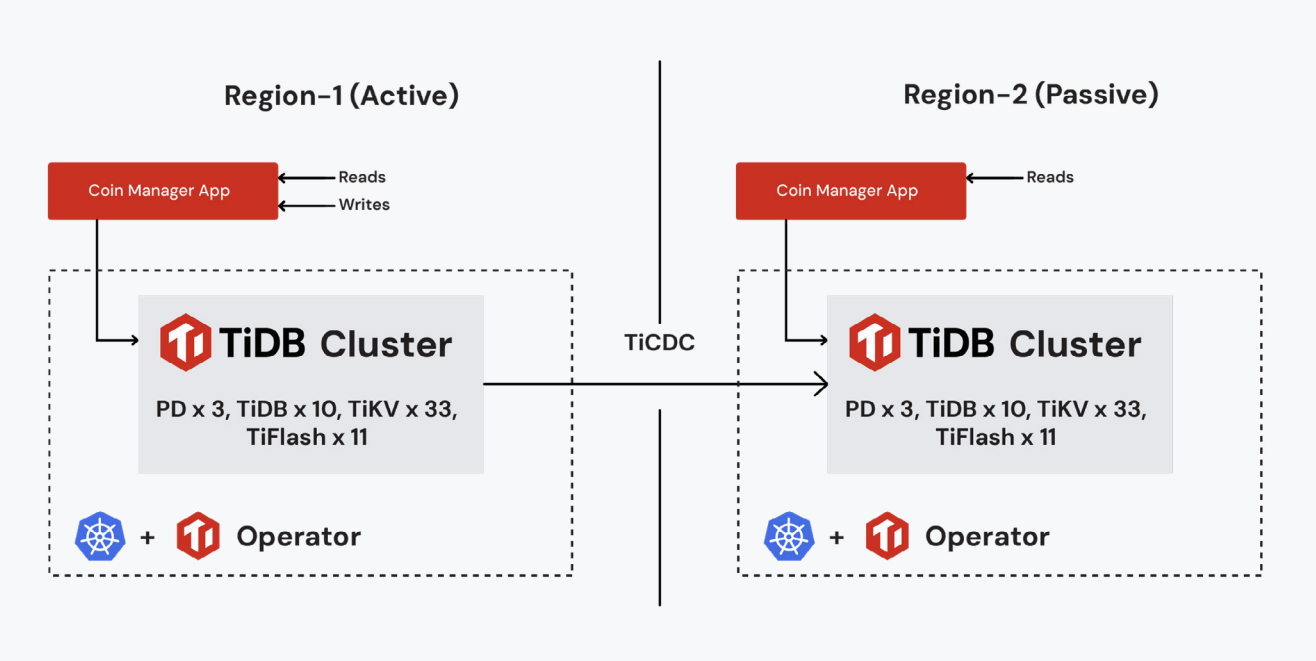


Figure 2: Evolved architecture of a wallet application with TiDB at its core

Instead of having multiple replication channels and maintaining multiple shards, there are only two TiDB clusters: One acting as a primary cluster, and the second as a passive cluster. The data between these two TiDB clusters get replicated and synchronized by using TiCDC, a tool used to replicate incremental data from TiDB.

The benefits of this transition to TiDB include:

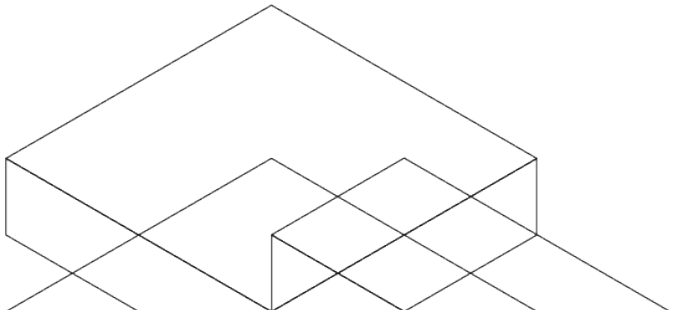
- **Simplified Architecture**: The overall system is much easier to manage, with fewer components and less complexity.
- **Low Maintenance**: With the elimination of manual shard maintenance, operational tasks are greatly reduced.
- **Automatic Data Sharding**: TiDB's automatic sharding removes the need for manual shard management, reducing operational overhead.
- **Auto Failover Mechanism**: TiDB provides a robust failover system that requires no manual intervention, ensuring high availability.
- **Flexible Scaling**: TiDB allows for easy scaling, both out and in, without the need for complex data resharding. Data rebalancing is handled automatically.
- **High Performance**: The system's performance is enhanced, delivering faster and more reliable results.
- **Data Consistency**: TiDB ensures strong data consistency through its Raft protocol, providing reliable transaction processing and data integrity.

This transformation highlights TiDB's ability to simplify complex architectures while maintaining high performance and reliability.

Integration Points

This use case highlights a seamless and efficient migration to TiDB, facilitated by the TiDB Data Migration (DM) tool. By automating the transfer of both data schemas and underlying data, the DM tool ensures a smooth migration process without manual intervention, making it hassle-free and highly efficient.

Additionally, integrating TiDB with downstream systems is equally straightforward. TiCDC enables real-time data transfer from TiDB to platforms like MySQL or Kafka, ensuring continuous and accurate data flow. TiCDC also supports integration with Pulsar, providing flexible and reliable options for downstream connections, enhancing the data pipeline's overall efficiency and reliability.



Suitability Assessment

TiDB offers several advantages as a distributed SQL database for e-commerce rewards program use cases:

- **High Availability**: TiDB automatically maintains multiple data replicas, ensuring that data is always protected. With its built-in automatic failure detection and recovery capabilities, TiDB guarantees continuous operation, even in the face of hardware failures or network issues.
- **Data Consistency**: TiDB provides ACID-compliant transactions and strong data consistency, akin to traditional relational databases. This is particularly crucial for wallet-like applications, where transaction accuracy and consistency are paramount.
- **Elastic Scaling**: TiDB excels in environments with dynamic workloads, thanks to its ability to scale storage and processing power linearly by simply adding nodes. Scaling out or in is as easy as executing a single command, and TiDB automatically triggers data rebalancing without disrupting ongoing operations.
- **Performance Stability**: With its robust distributed architecture, TiDB ensures that performance remains stable, regardless of changes in data volume or throughput. This guarantees consistent, reliable performance as your workloads grow.
- **Multi-Region Support**: TiDB enhances disaster recovery strategies by leveraging TiCDC for asynchronous data replication to passive clusters. This automated process simplifies DR maintenance and ensures high availability across multiple regions, providing resilience and peace of mind for global operations.

However, there are limitations to consider:

- **Infrastructure Requirements**: Implementing a distributed database can increase infrastructure demands in certain scenarios, requiring more advanced configurations that must be carefully managed.
- **Maintenance Complexity**: Unlike single-node databases, distributed systems come with added complexity in deployment, maintenance, and fault diagnosis. Managing these environments requires specialized personnel and tools to ensure smooth operations and quick problem resolution.
- **Cost Considerations**: As an enterprise-level solution, TiDB involves various costs, including licensing fees, hardware investments, and the need for personnel training. Organizations must carefully weigh these expenses against the benefits to determine the most cost-effective approach.
- **Data Archival Strategy**: Developing an effective data archival strategy remains a challenge. While TiDB excels in many areas, implementing a streamlined and efficient archival process is an area that still requires further refinement.

Competitive Analysis

TiDB presents a compelling solution, but to fully understand its value, it's essential to compare it against other leading database solutions. The below chart provides a detailed competitive analysis, highlighting how TiDB stands out in scalability, data consistency, ease of maintenance, and cost-effectiveness, and why it might be the ideal choice for powering modern e-commerce rewards program applications.

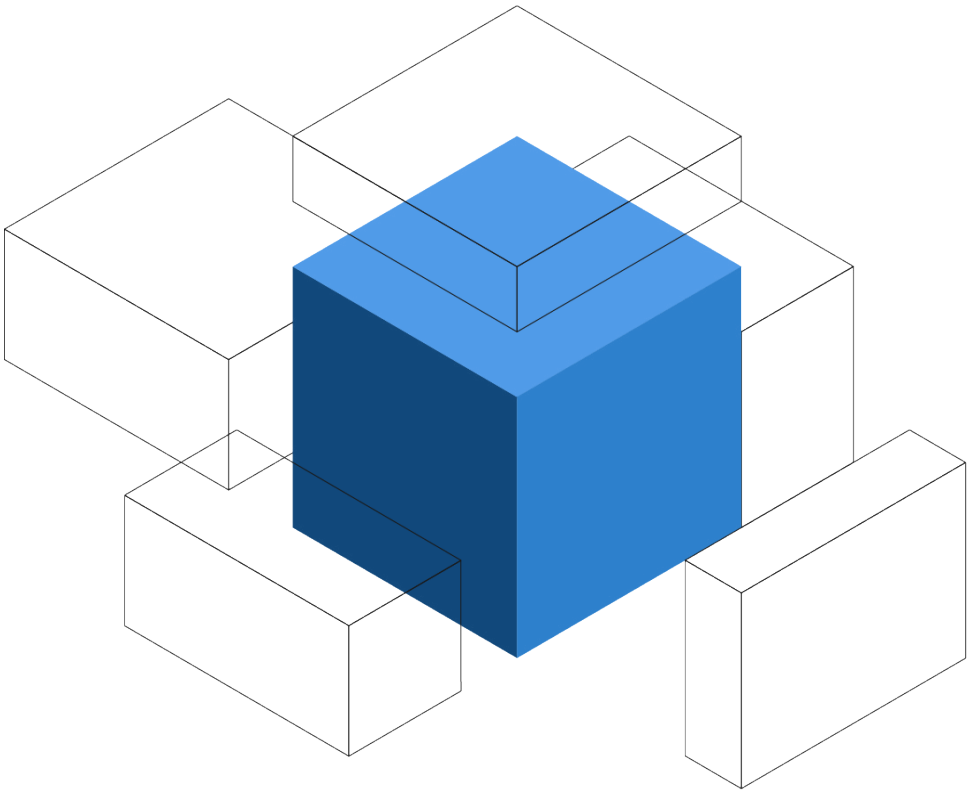
	TiDB	MySQL RDS	MySQL Sharding	Amazon Aurora
Average Latency	10-50ms	5ms	7ms	5ms
Write Throughput QPS	1 million +	Around 10,000	500,000 +	Around 20,000
Business Agility	High	In	Very low	In
Availability	• 99.99% • RTO<30s • Downtime affects 1/N traffic	• 99.99% • RTO<50s • Main downtime affects all traffic	• 99.99% • RTO<50s • Downtime affects 1/N traffic	• 99.99% • RTO<60s • Main downtime affects all traffic
Scalability	Compute/storage can be dynamically horizontally scaled, beyond 1PB	Writing and storage are limited by standalone capabilities, while reading has certain scalability, around 3TB	Scalable horizontally, but with a long cycle and high risk, 50TB	Writing and storage are limited by standalone capabilities, while reading has certain scalability, around 128TB
Management Efficiency	High	Low	Very low	In
Resource Utilization	High	Low	Low	In
Online DDL Efficiency	High	Low	Very low	Low
Data Recovery Efficiency	In	High (based on Cloud Block Storage Snapshot capability)	Based on Cloud Block Storage Snapshot capbility	High
Multi Cloud/Hybrid Cloud	Multi Cloud/Hybrid Cloud	Multi Cloud/Hybrid Cloud	Multi Cloud/Hybrid Cloud	Not supported

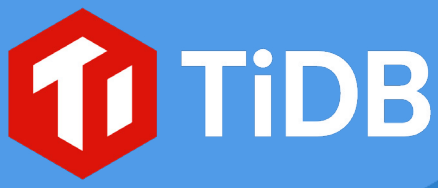
Conclusion

TiDB effectively addresses some of the most challenging issues faced by rapidly growing businesses, particularly those struggling to manage and support increasing data volumes with their existing data platforms. From day one, TiDB resolves these challenges without the need for manual configurations or workarounds, offering a seamless, scalable solution that supports growth and enhances operational efficiency.

To fully capitalize on TiDB's capabilities, it is recommended that organizations begin with a pilot project to evaluate its performance and integration within their specific environment. Following a successful pilot, plan for a phased migration of your most critical data workloads to TiDB, focusing on areas where data growth and performance demands are highest.

If you want to get started, schedule a free demo with one of our TiDB experts to learn how you and your team can become instantly productive with elastic scaling, zero downtime, and MySQL compatibility.





EVALUATE TiDB FOR YOURSELF

Start Your Free Trial

Contact us for a personalized demo at pingcap.com/demo/