THE DATABASE ARCHITECT'S GUIDE TO DISTRIBUTED SQL FOR FINANCIAL SERVICES

yugabyteDB

TABLE OF CONTENTS

THE CURRENT FINANCIAL SERVICES LANDSCAPE

Financial services companies have pursued digital transformation in different ways and at different rates. Some are moving their infrastructure to the public cloud, while others are going all-in and building microservices-based applications. Others are actively seeking to reduce their dependence on monolithic SQL databases like Oracle and Db2.

Everyone's methods may be different, but their goals are the same—to innovate and keep pace with more digitally-demanding customers in a world where competition is fiercer and regulation is ever increasing. All while trying to manage a huge range of legacy systems that may not be up to the task.

IT teams are under immense pressure to improve performance, reduce costs, and deliver a dependable, adaptable, and future-proofed database operation that can run at the speed of light. This Database Architect's Guide is intended to act as a guide for that journey and how distributed SQL (in general) and YugabyteDB (specifically) can help.

To pull ahead in 2023 (and 2032!) financial services organizations want SQL-compatible databases that are:

- MULTICLOUD/HYBRID CLOUD
- RELIABLE
- SCALABLE
- SECURE
- POSTGRESQL-COMPATIBLE

Top Database Trends and Predictions to Look For in 2023

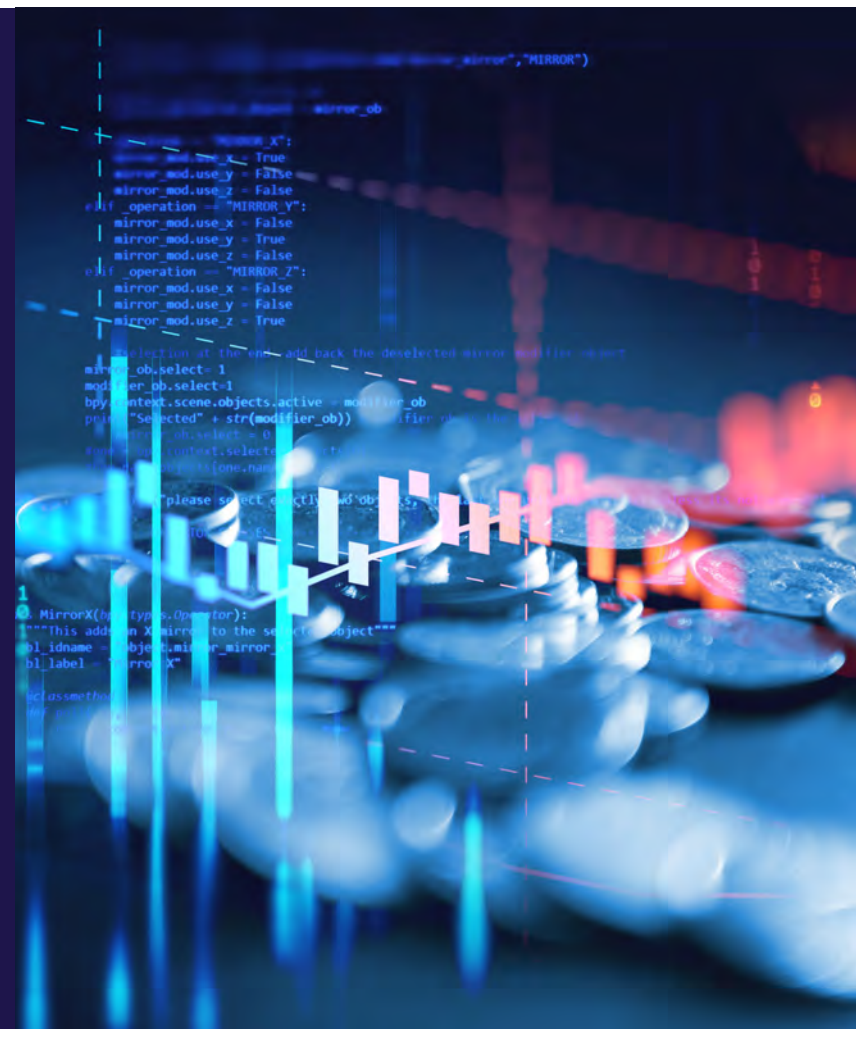
FINANCIAL SERVICES INDUSTRY INNOVATION

In recent years, there's been substantial investment in new technologies, initiatives, and projects aimed at meeting the expectations of mobile and digital-first customers, streamlining processes, and reducing costs within this highly competitive and increasingly regulated industry.

There is a compelling need to differentiate products and services and bring them to market quickly. Innovation is a 24/7/365 mindset.

Financial services ranked second to last across 12 sectors for perceived customer-centricity (only ahead of the US government).

Salesforce State of the Connected Consumer Report, 2020



It is also not one-dimensional. Products and services that are cutting edge one day are out of date the next. And customers expect a continual evolution of efficiency and digitization.

This raises the bar for financial services companies. They must not only find ways to stand out in a very crowded field of competitors, but also emerge from the increasingly crowded field of their own making.

Customers look for constant innovation in terms of their own digital products and services. But it's more than just apps and self-service access. Those are table stakes. Customer expectations are also driving changes to the very way many financial services firms are doing business. They are placing more importance on the firm's ability to intelligently personalize every step of their journey—not just the end. They expect decisions to be made in real time—not in batches once the data has been delivered and processed. They are embracing a more open ecosystem where they can access a variety of financial services through multiple channels and systems. They are not looking to stay within four discrete walls.

To offer these services and set themselves apart, financial services firms must innovate by unlocking their most valuable, proprietary asset—data. But many are hobbled by their back-end systems, which are proving to be inflexible and slow to change.

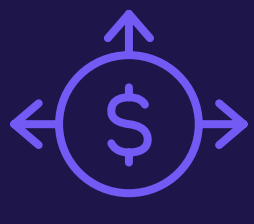
The Top Financial Services Challenges

Implementing new and innovative ways to serve and engage with customers

Streamlining operations and improving efficiency

Keeping pace with digital-first competitors

Standing out from rivals and improving outdated product offerings



Diversifying revenue streams

Navigating complex cross-border regulatory, compliance, and reporting requirements

AN INTRODUCTION TO DATA-DRIVEN INNOVATION AND THE MONOLITHS HOLDING IT BACK

Data holds value. It reduces risk, informs better decisions, and drives innovation. But innovating is difficult if you're trying to support a wide range of legacy, monolithic technologies that were not natively built to support the speed at which today's banks, fintechs, and financial services firms need to operate.

Many companies recognize this and have begun to extract this hidden value by modernizing their technology stack, including their infrastructure, applications, and databases.

However, while most financial services companies have embraced cloud platforms and microservices-based applications, many have not yet modernized their databases. They struggle to support cloud-native application code running at scale or to ingest data from various processing systems and bring it all together in one platform.

A modern data layer helps financial services organizations leverage their most valuable asset—data—so they can quickly develop new revenue streams, become more nimble, and future-proof their systems, processes, and workflows.

How Can Financial Services Firms Innovate?

01 **Modernize legacy systems**

02 **Monetize and maximize the value of their data**

03 **Improve access to real-time data**

04 **Future-proof systems, processes, and workflows**

05 **Improve their digital customer experience**

06 **Explore options to simplify the application development process**

07 **Embrace a multicloud strategy**

08 **Reduce operational burdens**

THE NEED FOR DATABASE MODERNIZATION IN FINANCIAL SERVICES

Digital transformation initially focused on the infrastructure layer, which included virtualization and cloud, before shifting to the application layer. Today's applications are being built specifically for the cloud (private and hybrid), and more financial services companies are moving away from monolithic applications to microservices and agile development processes.

This shift has given development teams the freedom to choose the infrastructure that best fits their apps, allowing them to deliver solutions with greater speed and flexibility. Infrastructure can now be provisioned in minutes, and applications can be built and scaled much faster.

However, the transactional database remains the last piece of the puzzle. If not modernized, it can strain resources and negate much of the progress made in terms of speed and agility.

Companies may have a flexible cloud infrastructure at the bottom and a proven approach to application development at the top, but for many, the transactional database remains a slow-moving part of the stack. If an app suddenly hits an inflection point and requires massive scale, the monolithic, legacy database will struggle and may even shut down.

It's this data layer that is holding back many financial services companies from:

- Meeting current and future customer expectations
- Finding new ways to monetize their data
- Handling compliance, security, and regulations with ease
- Truly differentiating themselves

HOW TO ACCELERATE MODERNIZATION AND INNOVATION WITH DISTRIBUTED SQL

Despite the rise of cloud infrastructures and cloud-native applications, many financial services firms still rely on legacy RDBMSs, like Oracle, SQL Server, and Db2, for their business-critical applications.

These monolithic databases were not designed for the world that financial services firms now inhabit—one of cloud computing and geo-distributed applications. In fact, these databases were built long before the first cloud architecture was doodled on a napkin.

Monoliths handle increasing demand for data volumes and processing performance by adding capacity. That is not as quick and easy as it sounds. If demand falls, it is just as difficult to scale them down. The only way to safeguard against the total collapse of the single host machine is to run a second, standby, machine and keep its data synchronized with the primary machine through replication. However, unless performance is unacceptably compromised, replication can only be asynchronous, resulting in additional complications.

This does not work when trying to run geo-distributed microservices with the millisecond responses that customers demand. The database needs to be agile, horizontally scalable, and continuously available. It needs to have been built for the cloud, not retrofitted for it. The database needs to be distributed SQL.

MONOLITHIC SQL VS. DISTRIBUTED SQL

What are the main feature differences between a monolithic, legacy SQL database and a modern, distributed SQL database? It comes down to scalability, replication, availability, and the type of infrastructure it resides on.

Feature	Monolithic SQL	Distributed SQL
Scalability	Vertical	Horizontal & Vertical
Strong Consistency	Local	Global
Replication	Manual	Automatic
Availability	Low	High
Caching	Independent	Built-in
Cloud Infrastructure	Cloud Hosted	Cloud Native

Distributed SQL is not only transformative in terms of innovation and customer satisfaction, but it can also have a positive impact on the bottom line.

- The ratio of operations personnel to developers can be very low, sometimes only requiring one operator for every 100 developers.
- Infrastructure costs are significantly lower due to distributed SQL's flexible use of commodity hardware vs the high-performance servers or specialized hardware usually required for databases like Oracle and IBM Db2. Additionally, data density can even further reduce the hardware footprint, especially compared to common NoSQL solutions.
- True open-source distributed SQL solutions have much lower software licensing costs compared to proprietary databases, and they are more capable thanks to the robust, developer community that surrounds them.
- Revenue loss due to downtime is eliminated with distributed SQL, helping to maintain high profit margins.
- Consolidating existing SQL and NoSQL workloads into a distributed SQL implementation can save money by minimizing database sprawl and reducing operational complexity.

NoSQL databases were developed as an alternative to SQL databases. Their goal was to provide horizontal scalability without compromising performance. However, NoSQL only looks cloud-ready from a distance. They present significant operational challenges when run at scale in production.

History of the Transactional Database

YugabyteDB—DISTRIBUTED SQL FOR FINANCIAL SERVICES

Enter YugabyteDB. YugabyteDB is a cloud native relational database for modern transactional applications. It is an industry leader in the powerful new category of databases called distributed SQL.

YugabyteDB is full-featured, offering continuous availability, and it can scale massively (up or down) on demand. It combines the best features of relational databases with the benefits of cloud native databases. It is the only distributed SQL database that is hybrid and multicloud-ready, and multi-API. YugabyteDB is also the most PostgreSQL-compatible, distributed SQL database on the market today.

YugabyteDB has an innovative architecture, featuring a distributed transactional storage layer and a pluggable query layer with a PostgreSQL-compatible API and a Cassandra-compatible semi-relational API. It can be deployed anywhere—on any public cloud infrastructure, private cloud, Kubernetes, bare metal, and more.

Many financial services companies have embraced YugabyteDB to support a range of innovation and modernization strategies. Some have top-down initiatives to modernize their database infrastructure, while others are looking to completely move off of proprietary database software. Most are building cloud native applications and require a scalable relational data layer. Many need powerful transactional support that spans regions. We are also seeing a lot more exploratory topologies as the need for distributed SQL grows.

> **So what really accelerated our interest in distributed SQL? It came down to two things—resiliency and scalability. It was about ensuring a resilient, active-active solution that could support a relational data model and ACID-compliant transactions, along with the ability to achieve near linear horizontal scalability.**
>
> Head of Database Strategy and Innovation
> Top 5 Multinational Financial Services Company

So now that we have discussed the "what" when it comes to distributed SQL and YugabyteDB, let's discuss the "why" around YugabyteDB. Why are banks, fintechs, and other financial services firms moving their most mission-critical, high-volume transactional workloads to YugabyteDB? Nine little reasons:

1. PostgreSQL Compatibility
2. Security
3. High Availability
4. Horizontal Scalability
5. Multicloud/Hybrid Cloud
6. Open Source
7. Geo-Distribution
8. Transactional Consistency
9. Operational Simplicity

1. PostgreSQL Compatibility

PostgreSQL is a widely used, open source database management system with decades of active development, a thriving community, and a rich ecosystem of extensions and tools. However, despite its popularity, PostgreSQL was not designed for modern, dynamic cloud platforms or the needs of cloud native, geo-distributed applications.

YugabyteDB solves this by providing the most complete set of PostgreSQL-compatible features in a distributed SQL database tailored-made for the cloud. By ensuring PostgreSQL compatibility, developers can work with familiar features, applications, drivers, and tools so they can immediately be productive.

PostgreSQL Compatibility Levels

1

Wire-Protocol Compatibility.
Allows PostgreSQL client drivers to communicate with the database so that developers experienced with PostgreSQL can easily build applications.

2

Syntax Compatibility.
Allows the database to parse the PostgreSQL syntax so developers can use some of PostgreSQL's tools and frameworks.

3

Feature Compatibility.
Supports the advanced features in PostgreSQL— triggers, partial indexes, and stored procedures—that help financial services firms get the most out of their data.

4

Runtime Compatibility.
Ensures that the database will match the PostgreSQL execution semantics at runtime. This is the highest level of PostgreSQL compatibility, automatically implying that the other three compatibility metrics are met.

PostgreSQL popularity has increased nearly 3x since 2014 thanks largely to its highly active open source community.

2. Security

YugabyteDB was designed and built from the ground up with security in mind. Financial services organizations can maintain a robust security posture with built-in controls like LDAP authentication, role-based access control (RBAC), data encryption at rest and in transit (TLS), audit logging, row-level security (RLS), and column-level permissions.

YugabyteDB Aeon, our self-managed database-as-a-service (DBaaS) offering, also simplifies security operations by providing automatic security key rotation, rolling software updates, and other capabilities.

3. High Availability

For many financial services applications, downtime is not an option. The impact can be disastrous, both financially and reputationally. However, even with the best design, infrastructure and network outages are inevitable in today's cloud-centric world. The question then becomes: how to survive an outage when application and data downtime is unacceptable?

Legacy databases rely on older replication models and manual processes to switch to standby configurations during failures. Further hands-on-keyboard intervention is required to reconcile all final transactions for complete data consistency.

Distributed SQL databases, like YugabyteDB, maintain continuous availability during infrastructure failures. Critical services remain available during node, zone, region, and data center failures. They heal themselves and automatically re-replicate data, enabling fast failover, so there's zero downtime during maintenance tasks like software upgrades, security patching, and distributed backups.

4. Horizontal Scalability

Scalability is crucial for financial organizations operating across multiple regions. Horizontal scalability is part of distributed SQL's DNA. It's a core feature, allowing for effortless scaling of writes and strongly consistent reads. You can add storage and connections merely by adding nodes to the cluster. This means no more designing and maintaining shards, read replicas, or other app-level scaling features. There is no need for specialized hardware since YugabyteDB automatically recognizes and rebalances the data load to utilize all available resources. It delivers high performance with low latency and minimal effort.

Distributed SQL is important to us because our SLAs are very low, so we have to be very strong in our ability to scale and in our ability to do so in under 200 milliseconds.

Jay Duraisamy, SVP of Technology for Data and Analytics, Fiserv

5. Multicloud and Hybrid Cloud

Want to avoid being locked-into a specific cloud provider? With YugabyteDB's cloud native architecture, you can deploy on the clouds of your choice. You can seamlessly manage your distributed databases across Google Cloud (GCP), Amazon Web Services (AWS), Microsoft Azure, VMware Tanzu, Red Hat, and even on-premises data centers.

81% OF PUBLIC CLOUD USERS CLAIM TO USE MULTIPLE CLOUDS TO:

- **Reduce operating and infrastructure costs by avoiding vendor lock-in.**
- **Improve application resiliency and redundancy through the use of geographically distributed data centers.**
- **Improve customer experience and performance optimization. By choosing a data center closest to end users they can serve the requested data with minimum latency.**
- **Achieve data compliance with laws such as the EU's GDPR which requires data to be held in particular geographical locations.**
- **Expand into new markets by taking advantage of regional data centers.**

Why Organizations Choose a Multicloud Strategy, Gartner

6. Open Source

Open-source software has been proven to be the most successful way to develop and distribute business-critical infrastructure software. It offers users absolute freedom Day 1, making exponential adoption growth possible.

As adoption grows, it powers the rapid feedback loop needed for fast, reliable, and high quality collaborative feature development. Security, ecosystem integrations, extensibility frameworks, etc. just get stronger.

Proprietary infrastructure software with a free-to-use tier can be successful, but at a much slower rate due to the reduced levels of collaborative development and a slower feedback loop. In turn, this increases the risk that the software will fail to gain market acceptance and fade away.

YugabyteDB is a fully open-source database. It eliminates adoption barriers and lowers the risk of failure, making it an extremely attractive option for developers creating mission-critical applications and operations engineers running them on cloud-native platforms.

The decision to make YugabyteDB open source was heavily influenced by the success and popularity of PostgreSQL, which is not only open source, but also transparent and widely adopted.

Karthik Rangatharan,
CTO and Co-Founder of Yugabyte

7. Geo-Distribution

Financial services firms are deploying globally distributed, microservices-based applications to deliver always-on, highly responsive services worldwide. But deploying application instances across multiple geographies and serving user data from a single location is just not enough to meet resiliency, performance, and compliance objectives.

A geo-distributed data layer is crucial to achieve faster response times for real-time interactions, ensure the data layer is resilient to zone and region failures in the cloud, and meet data protection and privacy laws.

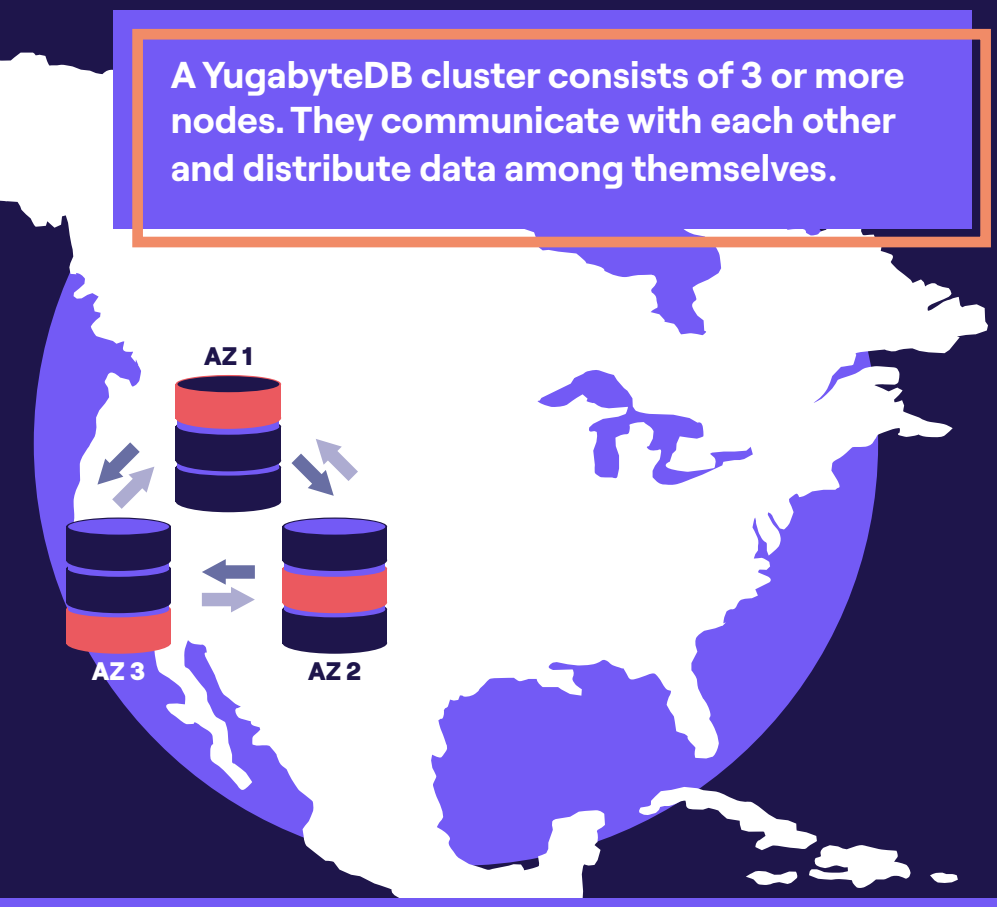
Luckily, YugabyteDB offers the most flexible deployment options for geo-distributed environments, including synchronous and asynchronous data replication and geo-partitioning. These options help financial services firms meet the high-performance demands of their customers and regulatory compliance thresholds.

We want to make sure our data is where it needs to be so that we can scale—not only across zones but also across regions—and handle different failure scenarios seamlessly. So, to make sure that the integrity of the transaction is maintained across zones and regions...that's really why we began looking at distributed SQL databases.

Managing Director of
Infrastructure, Operations,
and Cloud Platforms
Top 10 Multinational
Financial Services Company

6 GEO-DISTRIBUTED DEPLOYMENT OPTIONS YOU WANT TO CONSIDER

A YugabyteDB cluster consists of 3 or more nodes. They communicate with each other and distribute data among themselves.



MULTI-ZONE CLUSTER

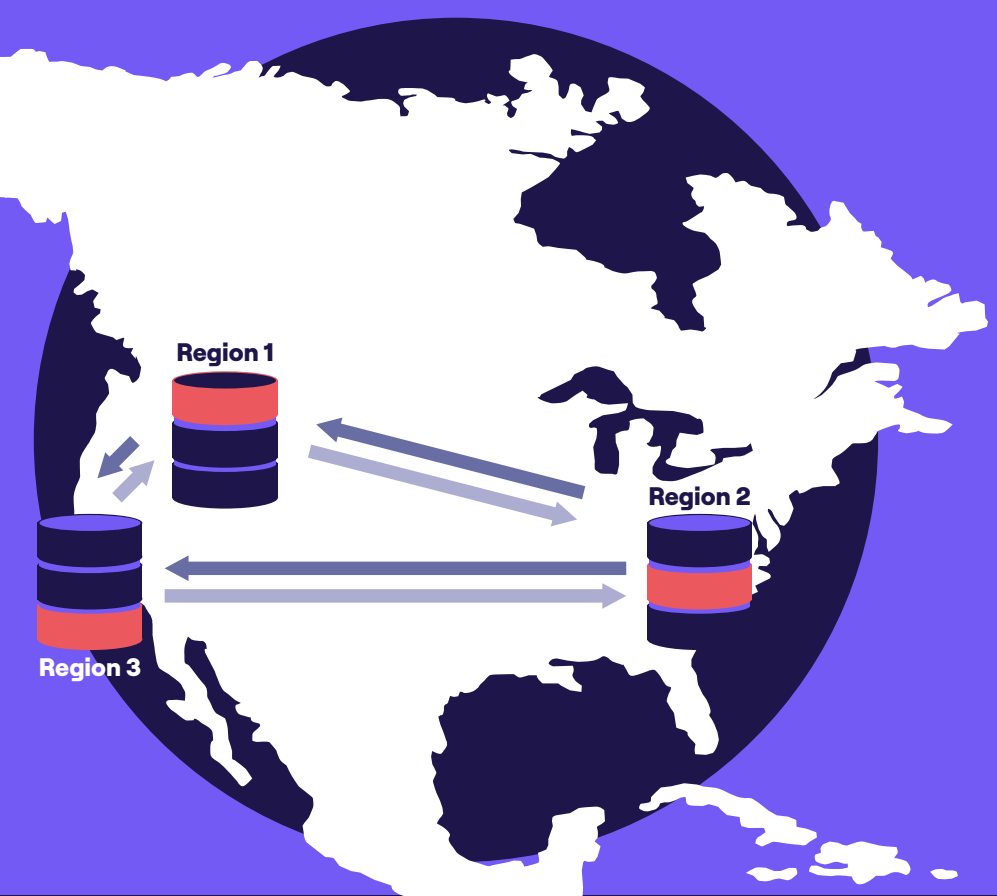
Deploy the nodes of a YugabyteDB cluster in different zones within the same region.

Benefits:

- Resilient to a zone failure
- High availability
- Strong consistency
- Low read and write latency within the region

Tradeoffs:

- Higher read/write latencies for remote regions
- Lack of resilience to region-level outages, such as natural disasters



MULTI-REGION "STRETCHED" CLUSTER

Deploy the nodes of a YugabyteDB cluster in different regions.

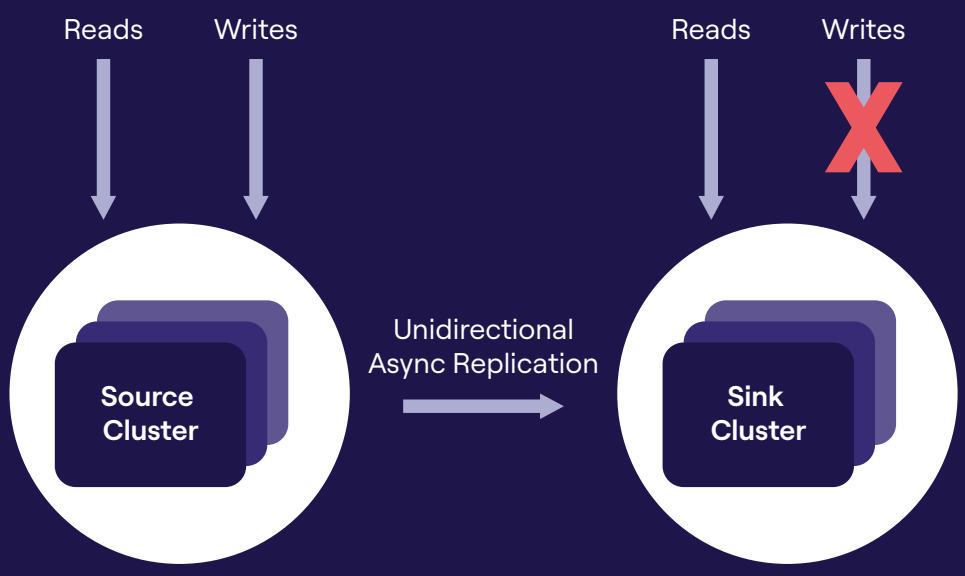
Benefits:

- Resilient to a region failure
- High availability
- Consistent writes; tunable reads
- Low read and write latency within the region

Tradeoffs:

- Write latency can be high depending on distance and/or network packet transfer times
- Follower reads trade consistency for latency

MULTI-REGION CLUSTERS WITH UNIDIRECTIONAL ASYNCHRONOUS REPLICATION



Deploy YugabyteDB cluster across two data centers or regions with asynchronous replication (active-passive configuration)

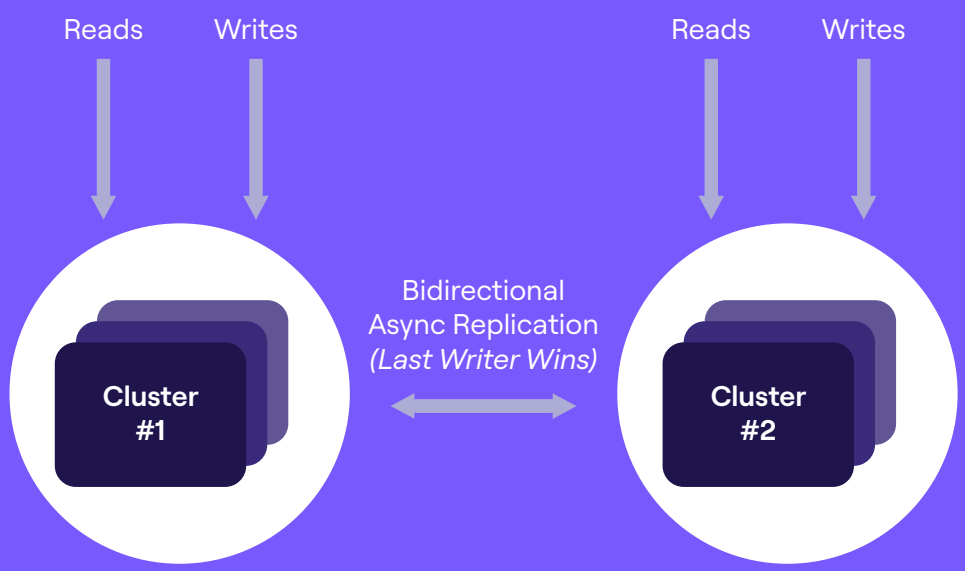
Benefits:

- Resilient to a zone failure, if nodes of each cluster are deployed across zones
- Strong consistency in source cluster; timeline consistent in sink cluster
- Low read and write latency within the source cluster region

Tradeoffs:

- High latency for clients outside source cluster region
- Database triggers won't get fired since xCluster bypasses the replicated records query layer

MULTI-REGION CLUSTERS WITH BI-DIRECTIONAL ASYNCHRONOUS REPLICATION



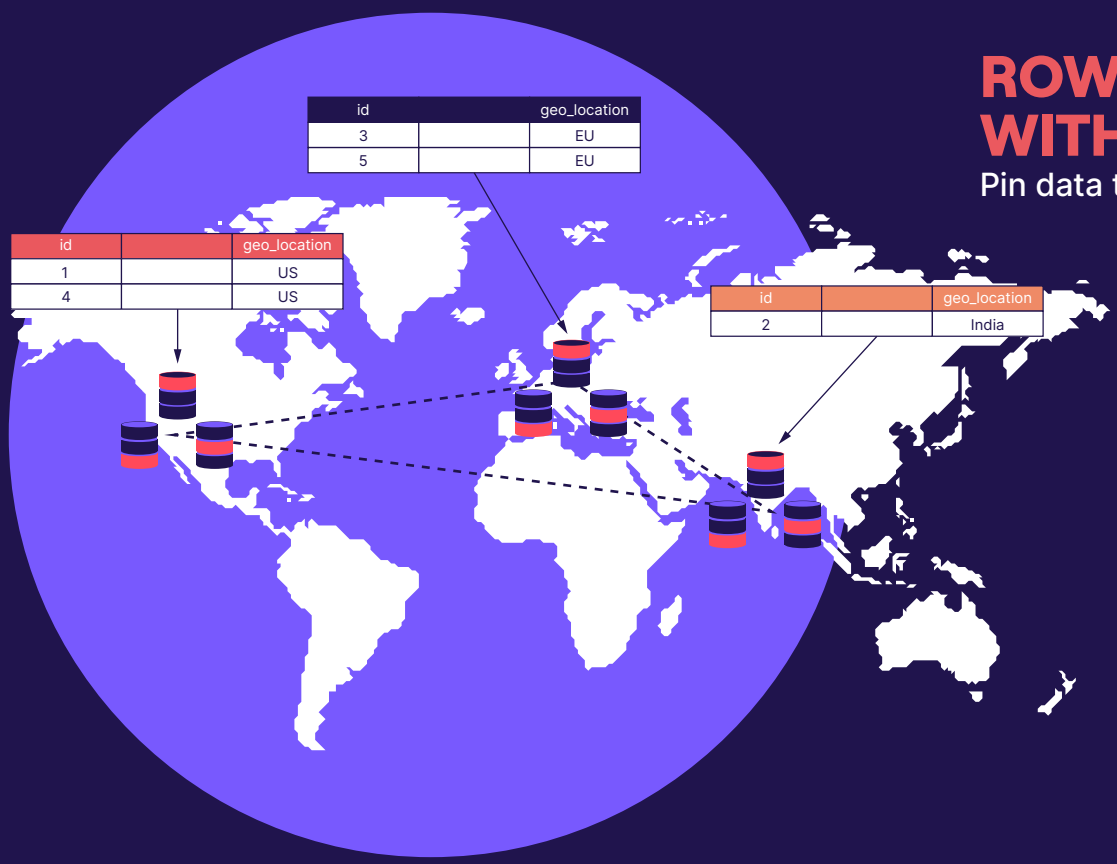
Deploy YugabyteDB cluster across two data centers or regions with asynchronous replication (active-active configuration)

Benefits:

- Resilient to a zone failure, if nodes of each cluster are deployed across zones
- Strong consistency in source cluster; timeline consistent in sink cluster
- Low read and write latency within either cluster

Tradeoffs:

- Database triggers won't fire since xCluster bypasses query layer, potentially leading to unexpected behavior
- Conflicting writes in separate universes can violate unique constraints and cause inconsistencies in the table and index
- The active-active mode does not support auto-increment IDs, so UUIDs are recommended instead



ROW-LEVEL GEO-PARTITIONING WITH DATA PINNING

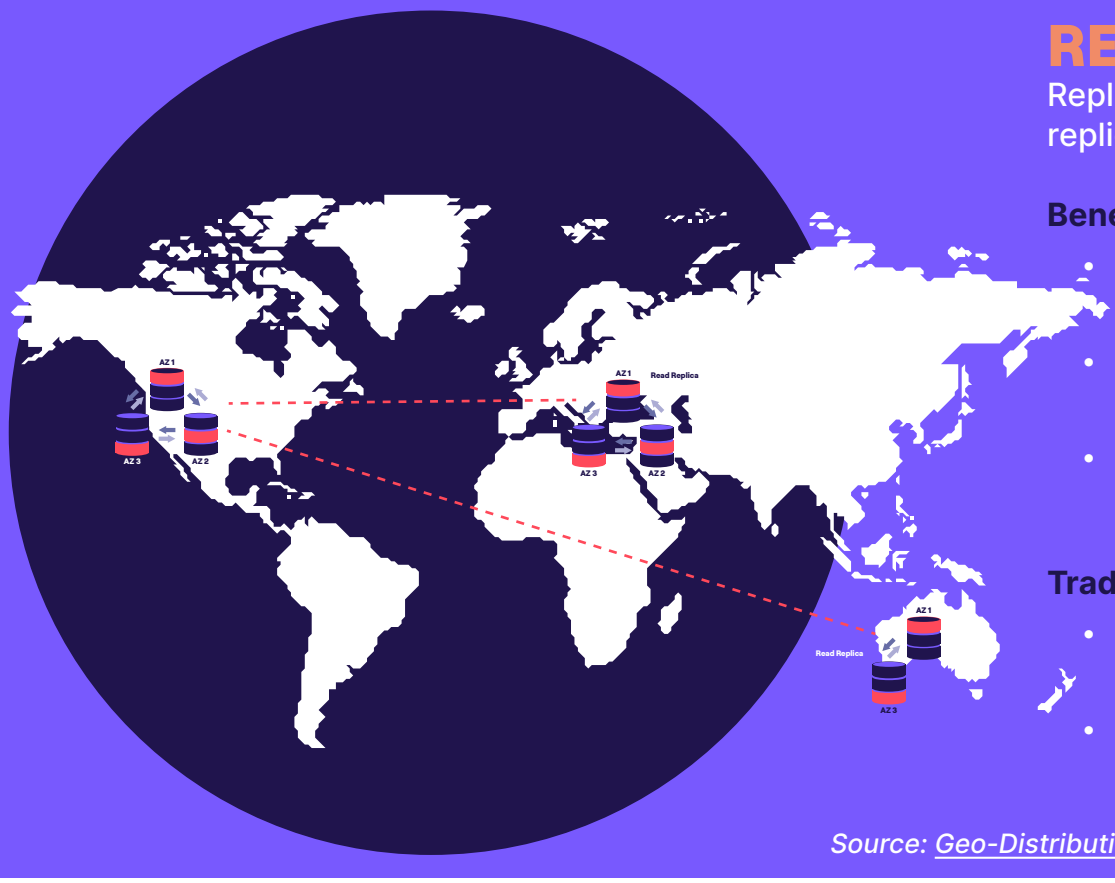
Pin data to regions for compliance and lower latencies

Benefits:

- Resilient to a zone failure, if nodes of each cluster are deployed across zones
- Strong consistency
- Low latency within region; high latency across regions

Tradeoffs:

- Best suited for datasets that can be logically partitioned
- Accessing pinned data from outside the region incurs cross-region latency



READ REPLICAS

Replicate data asynchronously to one (or more) read replica clusters

Benefits:

- Resilient to a zone failure, if nodes of primary cluster are deployed across zones
- Strong consistency in primary cluster; timeline consistent in replica clusters
- Low latency within region; write latency across regions dependent on distance

Tradeoffs:

- The primary cluster and read replicas are correlated, so adding read replicas does not improve resilience
- Read replicas cannot accept writes, leading to high write latency from remote regions even with a nearby read replica

Source: Geo-Distribution in YugabyteDB: Engineering Around the Physics of Latency

8. Transactional Consistency

By having transactional consistency in the data layer, YugabyteDB eliminates the burden of managing consistency across geo-distributed transactional applications and having to develop and maintain complex code.

YugabyteDB is fully ACID compliant across multiple rows, shards, and nodes without any scale, resiliency, or performance tradeoffs.

9. Operational Simplicity

With YugabyteDB Aeon, our fully managed database-as-a-service (DBaaS), you can focus on building business value while we take care of the database. Our goal is to simplify the entire process by providing data-layer services that streamline your operations.

For those that want more control and management over their data, YugabyteDB Anywhere allows financial services firms to effortlessly deploy YugabyteDB in their own data center or preferred cloud environment. Our built-in orchestration engine makes Day 2 operations a breeze, so you can rest easy knowing that your database is in good hands.

INTERVIEW WITH DIMITRI FARAFONOV, VP AND FELLOW ARCHITECT AT FISERV

Dimitri Farafonov, VP and Fellow Architect at Fiserv, recently sat down with Karthik Rangatharan, CTO and Co-Founder of Yugabyte, to discuss his role at this top global fintech and payments company.

During their conversation, they discussed the three priorities that guide Fiserv's database selection process along with the complexities involved in achieving low latency, data consistency, and massive bi-directional scalability, which are critical to Fiserv's operations.

Q: Can you provide more information about the role Fiserv plays in the financial services industry and how that impacts your approach to your role and responsibilities?

We help banks, big and small, to efficiently operate their credit and debit businesses. To achieve this, our solution is designed to be multitenant, allowing us to process transactions for multiple clients simultaneously. This adds another level of complexity. In addition, the banking industry is heavily regulated, which means we must comply with internal and external regulations, adding even more complexity to our work. As a result, we must follow numerous processes and protocols, which are constantly being reviewed to ensure we meet the highest standards.

The sheer volume of transactions we handle is enormous, with our online issuing business alone processing up to 15,000 financial transactions per second. In addition, there are over a billion account cards on file. It is just very large, with challenges that must be tackled on a massive scale. As a result, our focus is on delivering solutions that are scalable, highly available, and equipped with disaster recovery and data replication capabilities. These are the challenges we primarily deal with on a daily basis.

Q: What are some of Fiserv's priorities in terms of how you look at your market and how you work with vendors?

Our vendor selection process is guided by our priorities, with security being our top concern. Stability follows closely behind, with a focus on ensuring our software solutions are reliable and robust. Client satisfaction is also a key factor, which actually deals with how fast we can react to changes. So when we talk about vendor selection and how we work with Yugabyte, we always start with security.

For us, security and stability rank higher than innovation. We actually cut many "cutting edge" solutions from our selection process because they were not ready for primetime financial services applications. We only select the most secure and the most stable software to implement. For us, innovation usually entails taking very stable solutions and creating the right mix. We do a lot of integration work, and most of the movement we are making into real-time processing has everything to do with how we can react to changes quickly.

When it comes to databases, we distinguish between two categories: data capture and data aggregation. Data capture involves copying live transactions for audit purposes or to save for future processing. Data aggregation, on the other hand, requires transactionality, such as updating account balances after a purchase.

So when we make a database determination, again, we focus on our three priorities—security, production stability, and client satisfaction. Then we consider all the new demands of data capturing and data aggregation.

Q: You've mentioned that security is crucial, but would you take us through your evaluation process when considering different features of a database? Also, could you explain what brought you to YugabyteDB and which of the criteria it met during the selection process?

We work with multiple vendors. In fact, if you name a database, we probably have it. We ran open-source Cassandra for years and have vendor-managed Cassandra instances as well. But let me take a step back. As you know, in traditional, on-prem development, infrastructure capacity is allocated upfront based on predictions for how many clients will ultimately need to be onboarded for that product. And you usually go with the best-case scenario because you want everyone to love the product. So you allocate for maximum capacity, with enough physical services to manage it, up front. This often results in unused databases sitting idle for years until enough clients are onboarded to justify their existence.

So as we were moving to the cloud, our conversations with YugabyteDB and other vendors were focused on starting small and growing the database cluster as needed, without upfront infrastructure capacity allocation. We didn't want to buy 25 servers upfront. Our ask was to only pay for what we needed at that moment in time in terms of infrastructure, licensing, and compute allocations.

To meet both our internal high availability commitments and our customer SLAs, we mandate replication within the same region or data center. We also require cross-regional and cross-data center replication. So we run in two regions, rather than three, but we replicate in each region. So we triple replicate.

Our first YugabyteDB implementation was a reporting database. We used the CQL language with local quorum within a region and another copy in a different region, with eventual consistency for reporting. However, in our second implementation, we switched to SQL and the PostgreSQL client to ensure SQL transactionality and maintain records in a consistent state. We guarantee transactionality within one region with eventual consistency across the other region, and have set up preferred routes for disaster recovery events. This approach, which we call "active, active prefer local," allows for low latency transactionality in one region while maintaining a disaster recovery strategy in the other region. In this setup, one region can be constantly processing while the other is used for disaster recovery, with each region acting as primary and secondary for different clients. This enables us to achieve transactionality with low latency in one region while also having a disaster recovery strategy in the other.

Q: So you're using both the Cassandra interface (CQL) and PostgreSQL interface to the Yugabyte database? And this is based on the applications' needs?

Yes, that's correct. However, we maintain two separate database clusters to cater to the specific needs of each application. One application focuses on data capture and stores it for end-of-day reporting, while the other is a real-time API for a tracking system that involves lookups and insertions throughout the day.

Our approach entails utilizing SQL or CQL inserts for data capture and report generation while using transactional locking of SQL for standard production operations in the other application.

Q: What are some of your security requirements and how has that impacted your migration to the cloud and your relationships with the vendors you work with?

We demand a lot when it comes to security, making it difficult for many vendors to work with us. We demand a lot, and we don't compromise on security requirements.

It's not solely about data access. Yes, there does have to be access levels, but we have very specific auditing requirements for those access levels. For example, we categorize every interaction within the database from both an application and administrative perspective. Every SQL command run is audited. But the difficulty is not with the audit; the difficulty comes in terms of integrating it with an existing system, especially if that system is in a different region. Because the security integration is managed by different groups, we must submit our audit data to them for evaluation. So it's a process.

When selecting vendors, we have a checklist of security requirements, including access levels, specific auditing requirements, encryption at rest, and various other security protocols that must be implemented. We do not compromise on security protocols, and we mandate vendors implement specific ciphers and protocols. Encryption at rest is mandatory, even if we don't store PCI data in the cloud.

But once we select the vendor we are not done. We continually submit enhancements to vendors, such as token authentication and user management integration. Choosing a database-as-a-service provider is challenging due to our long checklist of security requirements. Because of that, we have a goal to achieve platform independence by building a private cloud based on Kubernetes and bringing YugabyteDB on-premises while applying the same protocols and requirements as the public cloud.

Q: So tell me a little bit more about that.

Our goal is to achieve true platform independence, and we're starting by constructing our own private cloud. We develop our applications to standards, utilizing a Java persistence API to interact with YugabyteDB without any reference to Yugabyte code. Let me explain why.

Previously, we encountered challenges when developing applications for specific databases. As we expanded, we had to rewrite the code for different regions that didn't support those databases. However, by coding to a standard, we can identify the most suitable PostgreSQL vendor for each region. That's why we write applications for Postgres and employ Yugabyte as the vendor that supports Postgres for us. This approach enables us to replace YugabyteDB with other Postgres vendors as required, thereby achieving the ultimate platform independence.

> **We intentionally make it challenging to work with us because if we fail to post your check to your account, you will definitely notice. We take our impact on people's lives seriously, and that's why all of our processes are designed with purpose. We prioritize security and stability over being on the cutting edge of technology.**
>
> –Dimitri Farafonov, VP and Fellow Architect at Fiserv

Q: So how has the relationship/partnership been between Fiserv and Yugabyte from a support standpoint? What is your confidence in deploying YugabyteDB into your mission-critical apps?

At Fiserv, we have a centralized procurement process for vendor onboarding. Once vendors meet security protocols and other requirements, we establish an enterprise relationship with them. We then implement the vendor's services, making them available for anyone within Fiserv to use. We have an enterprise relationship with YugabyteDB, and it is deployed across multiple lines of businesses here.

So now let's chat for a moment about what it is like to work with us operationally. We've talked about the vendor evaluation process, but once a vendor passes the evaluation, we enter the operational phase of our relationship. As we work with our vendors across the company, we need a way to manage multiple clusters consistently, rather than managing them individually. This is because scaling and efficiency rely on sharing actual solutions.

Creating consistent operational models and databases that fit those models is very important because our distributed application DBAs manage multiple databases, and they work with the vendors specifically for their operational requirements.

This is why I highlighted the importance of having an SQL client that can interact with multiple SQL databases. YugabyteDB should be no different from any other SQL database that our teams work with.

Day 2 operations can be difficult enough, so having consistent operational models and databases that fit within those models are essential for scalability and efficiency. This is why Fiserv pushes all vendors towards it, furthering our reputation as being hard to work with. But for Fiserv to scale, we need consistency.

Security mandates drive consistency as well. We coordinate patching, both at the operating system and software levels, with all vendors following the same schedule. This can be challenging to manage operationally, especially when implementing solutions in different regions and clouds internationally.

Fiserv is trying to push for consistency and uses cookie-cutter recipes to solve problems once instead of solving them multiple times. Coding and implementing to standards are essential; otherwise, you are faced with a nightmare to manage.

Interested in additional interviews with Fiserv?

YugabyteDB IN FINANCIAL SERVICES

YugabyteDB has become a critical database for some of the world's most demanding financial services companies, including Wells Fargo, Charles Schwab, Fiserv, and Temenos.

These industry leaders chose YugabyteDB because it delivers an open source, cloud native, distributed database that uniquely combines enterprise-grade relational database capabilities with the horizontal scalability and resilience of cloud native architectures.

> **...YugabyteDB delivered up to 98% lower latency, greatly enhancing our customers' experiences..." Fiserv.**

...With YugabyteDB, there was 50% cost savings compared to SQL Server implementation providing the flexibility to explore new cases and scale faster..." Xignite.

YugabyteDB provides several significant advantages to leading financial services firms, including:

- Improve digital customer experiences by accelerating developer productivity and focusing on new, innovative customer services
- Navigate regulations and compliance by storing data where it is needed or mandates and effectively controlling access.
- Developing new revenue streams, boosting growth, and reducing operating costs with new, personalized data-centric services
- Modernizing legacy systems to power new sets of cloud native applications and update existing apps at the pace that fits your business strategy.

ABOUT YUGABYTE

Yugabyte is the company behind YugabyteDB, the open source, high-performance distributed SQL database for building global, cloud-native applications. YugabyteDB serves business-critical applications with SQL query flexibility, high performance and cloud-native agility, thus allowing enterprises to focus on business growth instead of complex data infrastructure management. It is trusted by companies in cybersecurity, financial markets, IoT, retail, e-commerce, and other verticals. Founded in 2016 by former Facebook and Oracle engineers, Yugabyte is backed by Lightspeed Venture Partners, 8VC, Dell Technologies Capital, Sapphire Ventures, and others.

I think the world has moved from monolith to microservices and having microservices on top of a monolithic platform only allows you to scale to a certain extent. That's why many data-centric products and solutions move towards a NoSQL architecture. And as we know, NoSQL comes with certain trade-offs. I always look for a solution that can merge these two. YugabyteDB presents a solution with both PostgreSQL grammar and Cassandra grammar as well as being fully ACID compliant; it seems to be the next evolution of NoSQL architecture.

Jay Duraisamy, SVP of Technology for Data and Analytics Fiserv

We wanted to create a cloud-native version of Postgres, while addressing three key concerns. First, to ensure resilience and availability in the face of potential failures of commodity cloud compute. Second, we had to make sure we could scale in and out quickly by leveraging available compute through an API. Finally, we needed to tackle the challenge of data replication across multiple regions. These were the three critical issues we focused on when reimagining Postgres for YugabyteDB.

Karthik Rangatharan, CTO and Co-Founder of Yugabybte

yugabyteDB

Get in Touch

www.yugabyte.com | contact@yugabyte.com

Get started in minutes with a free, full-featured trial of YugabyteDB Aeon DBaaS with confidence.

Have a question?
Contact us. We cannot wait to hear from you.