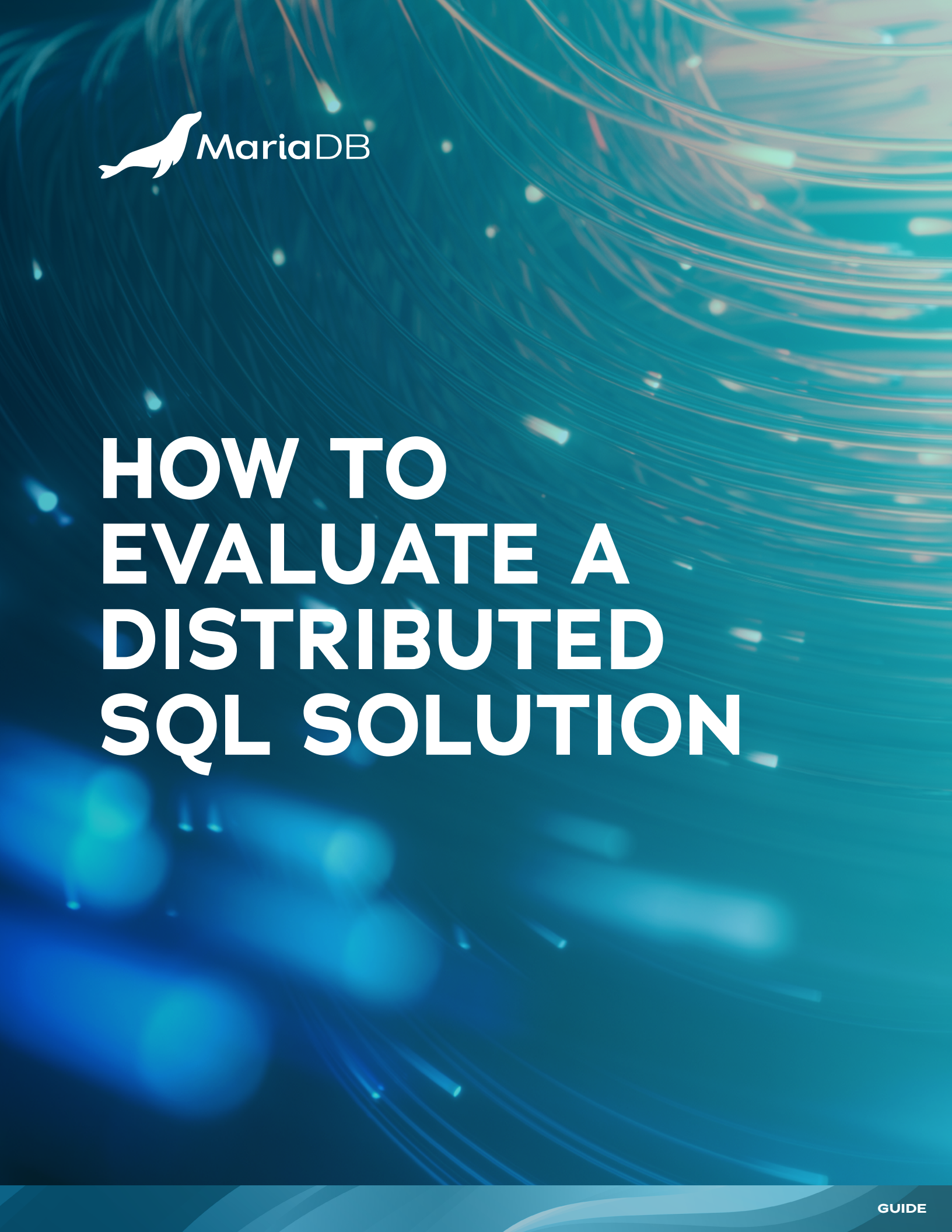


HOW TO EVALUATE A DISTRIBUTED SQL SOLUTION

GUIDE

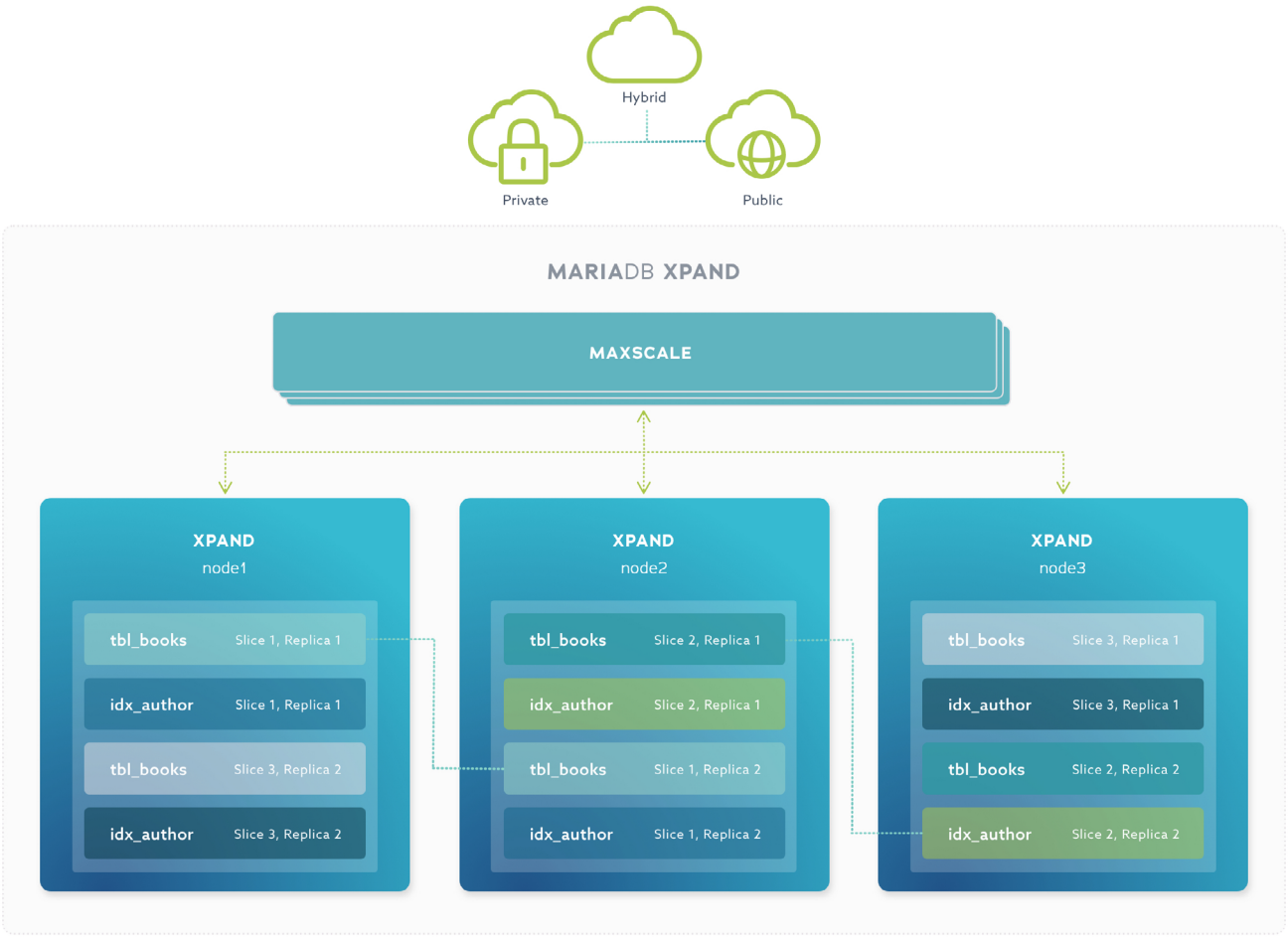
TABLE OF CONTENTS

INTRODUCTION

Distributed SQL databases are among the fastest-growing, most exciting new data and cloud computing technologies. For years IT organizations have relied on expensive, proprietary client-server relational databases based on 40-year-old technology. While client-server databases have been dependable, they were not designed for cloud computing or modern scale and availability requirements.

Distributed SQL databases maintain the same relational model and query language that have served millions of applications and billions of users in everything from e-commerce to banking and back-office applications like inventory management and financial systems. However, unlike traditional relational databases , they distribute reads and writes, query processing, and even indexing throughout a cluster of database nodes. Unlike NoSQL databases, they use a standard SQL dialect and provide real transactional integrity. Some distributed SQL databases like MariaDB Xpand, with its MaxScale database proxy, scale linearly.

Ultimately, distributed SQL databases are the new home for systems of record in both public and private clouds. Some distributed SQL databases can even replicate between public and private clouds.



USE CASES

Distributed SQL databases allow transaction processing at scale. They are a good option for systems that have a high volume of data, a large number of users, or strong high availability and disaster recovery (DR) requirements and systems of record migrating to the cloud. Distributed SQL databases such as MariaDB's Xpand are being used by large enterprises like Samsung to manage hundreds of millions of users accessing Samsung Cloud, and smaller companies like ShortStack to handle their core customer data.

There is a trade-off to distributing any kind of workload, including database reads, writes and queries. Compared with a distributed SQL database, a non-distributed database like MariaDB Server using the InnoDB storage engine will be faster and lower latency for a single write or a query that returns a relatively small number of rows. However, once the workload exceeds the abilities of a single server to return in a reasonable amount of time, distributed SQL databases scale linearly. Non-distributed databases suffer from diminishing returns because each new server adds less capacity than the last. While client-server databases can use replication, multi-master and various segmented read and write strategies, when a fault happens, operations are usually affected dramatically in terms of scale or actual unavailability.

Not every system is a good candidate for a distributed SQL database. Many systems are perfectly fast running on a local RDBMS like MariaDB Server and work great with a primary-secondary replication method. Availability in this setup is sufficient, cost-effective and provides extremely low latency.

In deciding to use a distributed SQL database, consider if high availability, increases in total capacity and performance at scale are worth the trade-off of lower single-query write and read performance.

There are some key pain points that distributed SQL databases are particularly good at addressing:

- Maxing out transactional workloads
- Locking issues
- Online schema changes
- Slow backup times
- Cost of operations

After scaling a client-server relational database to its natural cost barrier, organizations that encounter these pain points should strongly consider a distributed SQL database such as MariaDB Xpand.

KEY FEATURES

DISTRIBUTION OF WORKLOAD

While a single instance of MariaDB/InnoDB will outperform three instances of Xpand for a query like "select * from orders_line_items where order_id=42," the workload will go to only one node and almost directly to an index and the disk. When there are thousands of those queries per second, the advantage reverses. The fact that Xpand distributes to all three instances allows it to outperform the single-instance database. This performance advantage is especially true when the query uses more indexes and joins, which Xpand also distributes. As the workload increases, administrators can add more nodes to Xpand, and if the number of nodes doubles, the capacity doubles. For a client-server database, although RAC can distribute some of the processing, adding double the number of nodes more than doubles the price but does not double the capacity.

Distributed SQL databases have some generally standard features. These include:

- Distribution of workload
- High transactional throughput
- Scalability and elasticity
- Consistency
- SQL support
- High availability
- Disaster recovery

HIGH TRANSACTIONAL THROUGHPUT

Distributed SQL databases like Xpand have been proven in production at rates as high as 120K transactions per second (tps).

SCALABILITY AND ELASTICITY

Distributed SQL databases should scale almost linearly. Adding three nodes to a three-node cluster should double the database's overall maximum capacity (double the number of transactions at the same latency). Distributed SQL databases differ in their ability to "scale back" or remove nodes when traffic slackens. Xpand can remove nodes at runtime and automatically rebalance the load.

CONSISTENCY

While ACID compliance is not unique to distributed SQL databases, they use consensus protocols like Paxos or Raft to execute distributed transactions and ensure state is consistent across the cluster. This level of consistency is on par with client-server databases like MySQL and MariaDB Server, PostgreSQL, Oracle and SQL Server. This capability is beyond what is practical in NoSQL databases.

KEY FEATURES

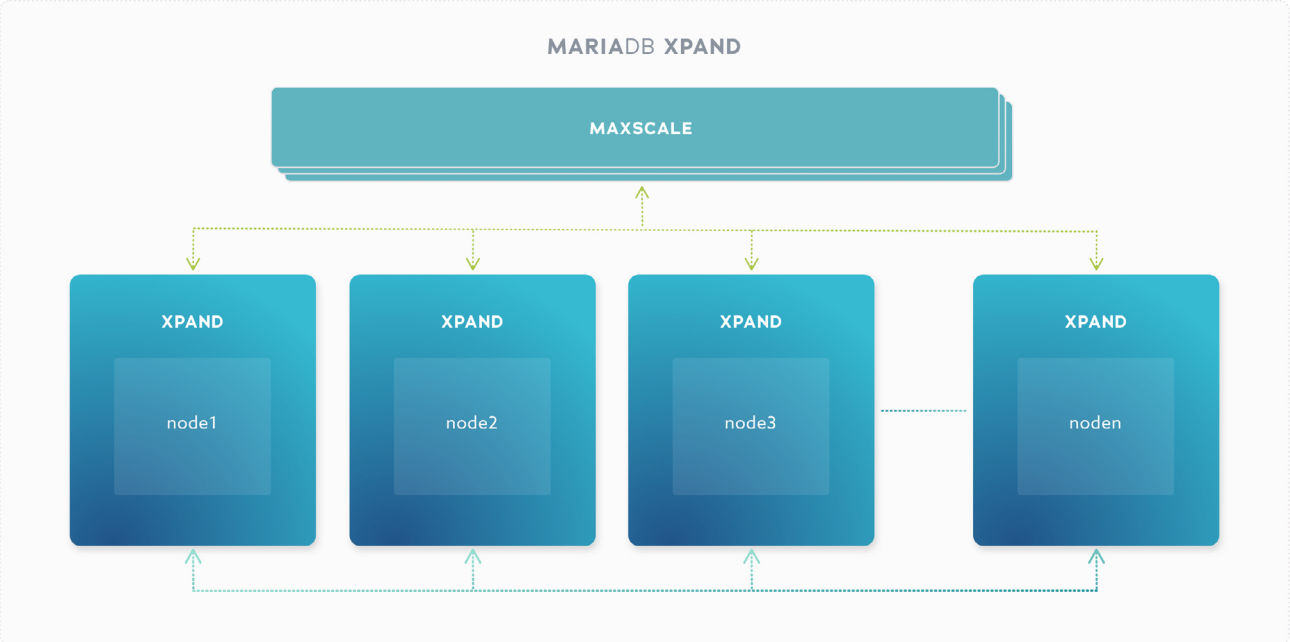
(Continued)

SQL SUPPORT

Distributed SQL databases are relational databases and support the SQL query language. They differ in terms of SQL dialect and overall compatibility. Xpand is compatible with ANSI SQL and the MariaDB/MySQL dialect of SQL. It also supports various Oracle extensions, including PL/SQL capabilities. Other distributed SQL implementations support subsets of PostgreSQL's dialect or their own proprietary subset.

HIGH AVAILABILITY

Distributed SQL databases create a duplicate copy of data on more than one node in the cluster. These replicas mean that no data is lost and service is not interrupted even if a node goes down. Distributed SQL databases differ in what happens next. MariaDB Xpand automatically recovers and rebalances data in a live cluster as nodes are removed or added.



DISASTER RECOVERY

All distributed SQL databases presently on the market can maintain a replica of the database in a second data center or cloud geographic region. Cross-data center replication is distinct from in-cluster replication in that it takes place over a distance and may have lower transactional guarantees than in-cluster replication.

Distributed SQL databases have different backup capabilities. MariaDB Xpand supports a fast backup and restore capability that operates at the row level in parallel to writes to the cluster. Backups can also be compressed to reduce bandwidth requirements for remote locations and the amount of storage utilized.

EVALUATION CRITERIA

DBAAS OFFERING

All current distributed SQL vendors claim to have a database-as-a-service (DBaaS) offering. However, not all of them are fully functional beyond the trial stage or have public reference accounts in production.

CLOUD PROVIDER AVAILABILITY

Distributed SQL databases can generally be self-hosted on whichever cloud a customer may choose. However, when it comes to supporting a cloud vendor's Kubernetes implementation or supporting that cloud provider using a database-as-a-service, there are different capabilities. MariaDB Xpand presently supports both AWS and GCP in SkySQL. It supports all three major cloud platforms for self-hosted installs.

There are important things to consider and measure while comparing distributed SQL solutions against traditional databases and each other. These include:

- DBaaS offering
- Cloud provider availability
- Private and public capability
- Throughput
- Scalability
- Latency
- Availability
- Compatibility
- Cost
- Credibility / longevity

PRIVATE AND PUBLIC CAPABILITY

Aside from public clouds, many IT organizations also run private clouds for security or network latency reasons (i.e., a manufacturing shop floor system tends to be local, with a public cloud DR site). Some distributed SQL offerings are available only in public cloud format. MariaDB Xpand is available in public, private and hybrid clouds. It can be easily tested and deployed on the SkySQL database-as-a-service.

THROUGHPUT

Ultimately, a significant reason to choose a distributed SQL database is higher throughput at a lower cost. Different architectural choices and limitations exist in any platform. Some of the biggest companies in the world run MariaDB Xpand in production at rates as high as a sustained 120K tps. These are numbers not easily achieved with traditional databases and unrealized by most other distributed SQL vendors.

EVALUATION CRITERIA

(Continued)

SCALABILITY

Nearly all databases claim to be scalable. Distributed SQL databases naturally follow a more scalable architecture than traditional databases or many NoSQL databases. Ultimately, this is proved by benchmarks. MariaDB Xpand has been shown to scale linearly. It can also downsize a cluster automatically, with no data or service loss, by removing nodes.

LATENCY

A distributed database will have a higher latency than a local disk or client-server database. Still, it should perform normal operations adequately in the tens of milliseconds, even with thousands or hundreds of thousands of requests per second.

Consider latency as a holistic budget. Establish a goal from when a user clicks to when the request returns (including HTML, graphics, JSON, etc.). For instance, for an everyday application, the response time expectation might be three seconds. The database should consume only a fractional share of that overall budget. For real-time trading applications, the budget could be a few dozen milliseconds. Establish your latency budget based on your application's real requirements and evaluate database latency as part of that budget.

AVAILABILITY

Distributed SQL databases provide resilience that is not possible with client-server databases. Resilience is especially poignant in the cloud, where an entire availability zone can be lost without a loss of service.

COMPATIBILITY

A key question in selecting a distributed SQL database is how much work it will be to migrate existing applications. Different distributed SQL databases support different dialects of SQL, data types and other semantics. For existing applications written to MariaDB Server or MySQL, Xpand and SkySQL are the most natural distributed SQL migration targets on the market. In addition, Xpand and SkySQL supply compatibility features for Oracle applications, including support for PL/SQL. For SQL applications written to other databases, there will be about as much effort as swapping between two different relational databases, including some query changes and changing the type names used in SQL DDL statements.

COST

When comparing cost consider the cost of licenses, cloud resources, risk of outages, staff training and overall maintenance. For many applications, a DBaaS offering like SkySQL will be more cost-effective and "pay as you go." For some organizations with considerable existing sunk costs, deploying a database like MariaDB Enterprise Server on premises will be optimal.

DESIGNING A POC

The most important aspect of designing a proof of concept (PoC) is to focus on data and queries that closely match your actual application. There is a temptation to test the platform's limits with unrealistic queries (i.e., 15 joins with 6 tables that pulls back 1M rows or a single-row point query) and measure the performance between different systems. Database technologies make trade-offs and optimize for particular usage patterns. In the case of distributed SQL, the database optimizes for throughput of transactional volume.

In designing a PoC, actual production data and application traffic is optimal. Second best is a simulation that closely matches the general pattern in terms of table structure, query complexity and proportion of reads and writes. It is important to set goals beyond a single factor such as pure database latency and focus on overall application performance at nominal and peak usage. This means that if at nominal use a traditional database offers 1ms latency but 1,000ms at peak usage, and the application performs at 4s but has a performance goal of 3s, it is not meeting the objective. If a distributed SQL database performs at 15ms under nominal usage but performs at 20ms at peak usage and the application meets its 3s goal, it has met the requirement.

It is essential to ensure that the infrastructure can generate sufficient load to test the database system capacity at the intended performance goal. For instance, if observed latency increases significantly at 1,000 transactions per second, but overall resource utilization of disk, CPU and network do not appear to be bottlenecked, it may be the load generation infrastructure is maxed rather than the system under test. It is equally essential to ensure the client network and other infrastructure between the load generator and system under test have sufficient capacity.

COST CONSIDERATIONS

Costs are more complex than licensing, cost per hour or any other vendor-advertised measure. Consider the entire cost of the system, including factors such as:

- **Staff training**
- **Ongoing maintenance**
- **Risk of loss of service during a failure**
- **Downtime during upgrades**
- **Support and support quality**
- **IOPS for cloud services**

CONCLUSION

Distributed SQL databases are an excellent choice for applications that require elastic scale, performance at high capacity, very high availability and disaster recovery capabilities. Any distributed database will have a higher latency than a non-distributed database when running at low capacity. Designing a PoC requires carefully considering the way the application uses data and the overall requirements of the organization.

If you are considering a distributed SQL database, MariaDB has several options available. SkySQL allows you to use MariaDB Xpand in the cloud without having to install any infrastructure. MariaDB professionals are available to provide guidance on the best way to architect and run a proof of concept and evaluate your options.

ADDITIONAL RESOURCES

Sign up for SkySQL

MariaDB Xpand fact sheet