MariaDB

OPEN WORKS

BE UNSTOPPABLE

MariaDB

OPENWORKS

MariaDB Xpand Under the Hood: From Architecture to Reality

Anders Karlsson, Principal Sales Engineer, MariaDB
Aman Singh, Customer Engineer, MariaDB

Agenda

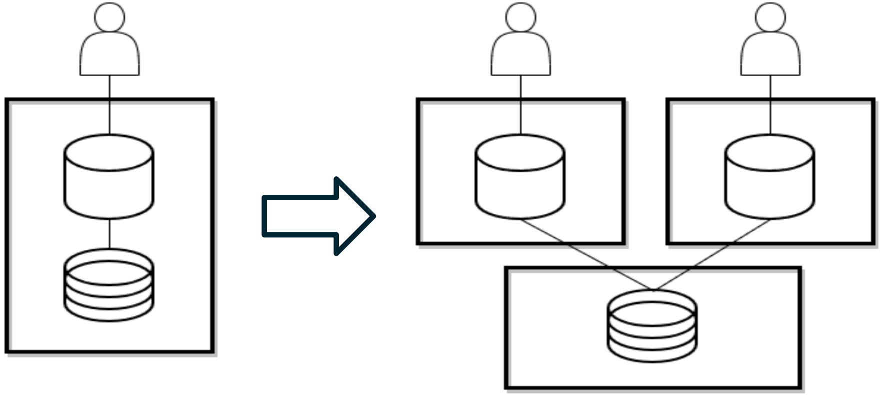
- Why Distributed SQL makes so much sense
- What is MariaDB Xpand
- A slice of Xpand – A unique MariaDB Xpand feature
    - MariaDB Xpand High Availability
    - MariaDB Xpand Elasticity
    - MariaDB Xpand Resource usage and cost
- The Xpand Storage Hierarchy
- The Xpand Rebalancer
- Conclusions

Why Distributed SQL Makes Sense

MariaDB

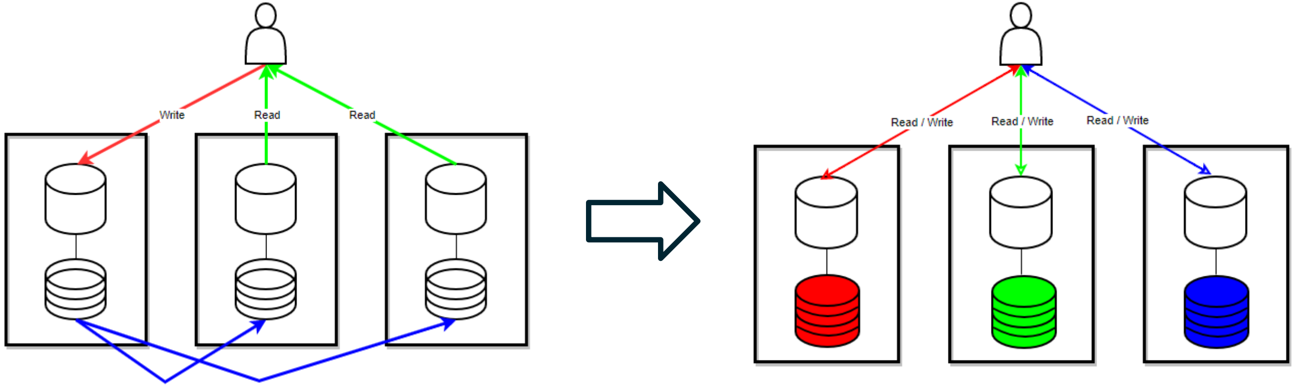
In the beginning, there was the database

- One server running **the database** was what we had
- For cost reason, for management reasons and for **licensing** reasons
- For high performance requirements, we needed **more CPU** power and **RAM**
- **Shared everything** databases became the norm for HA and performance



But servers grew, as did the load on them

- But shared everything was **expensive**, **complicated** and disk performance a bottleneck
- With much **more reads** than writes, **read scale-out** became the next big thing
- Read scale-out was easy and inexpensive, but **writes didn't scale**
- **Sharding** became the solution and suddenly **everyone was interested**, but…

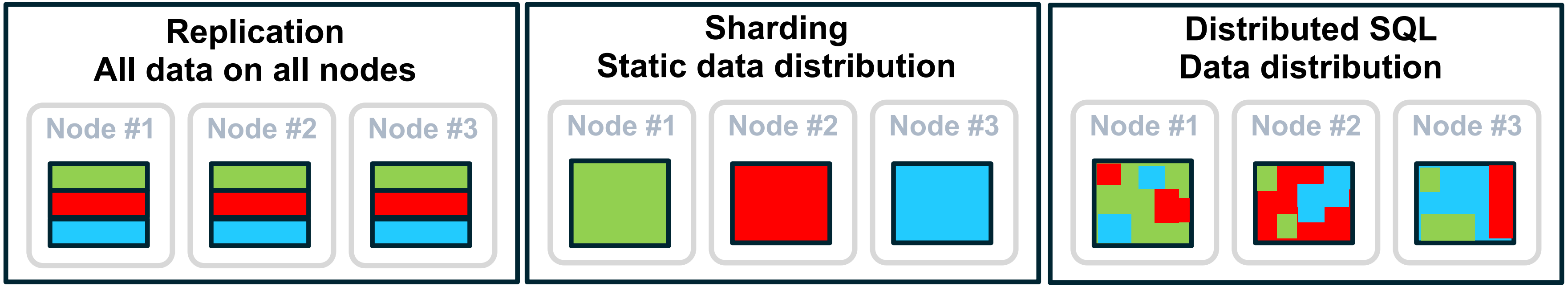


But sharding didn't cut it either

- Sharding meant that data was distributed, but **distribution is static**
- When new data was added, data had to be **redistributed**
- When data or processing became **unbalanced**, data has to be redistributed
- How do you **scale writes** when to data to be written isn't **evenly distributed**
- And applications need to **sharding aware**
- And by the way, how do we deal with **High Availability**

Distributed SQL solves the puzzle

- Data is **distributed dynamically** in the cluster across nodes
- **Processing** is distributed across the cluster
- All nodes are created equal
- Data is distributed without the need for **shard keys**



Replication
All data on all nodes

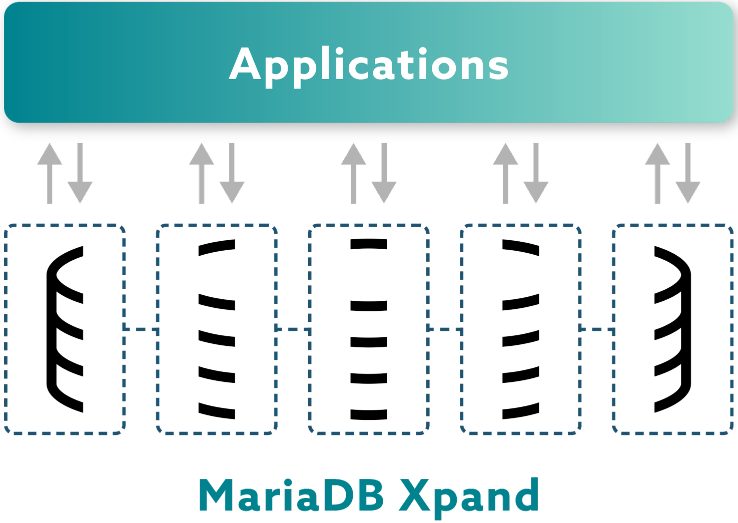
Sharding
Static data distribution

Distributed SQL
Data distribution

What is MariaDB Xpand?

MariaDB Xpand – Quick facts

- MariaDB Xpand is the **Distributed SQL** solution from MariaDB
- MariaDB Xpand is, from the application point of view, a **SQL based RDBMS**
- MariaDB Xpand is **SQL and protocol compatible** with MariaDB
- MariaDB Xpand is typically load balanced by **MariaDB MaxScale**



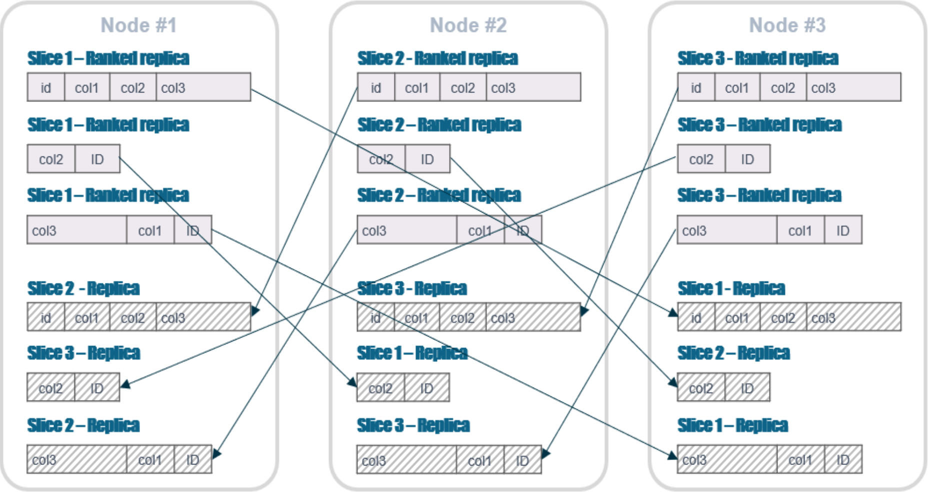
MariaDB Xpand – Quick facts

- MariaDB Xpand is set up in a cluster with typically **3 nodes or more**
- **All nodes are created equal** and can take any load
- **High Availability is built in** from the ground and up
- Key features are **scalability, availability, elasticity and cost effectiveness**
- MariaDB Xpand combines ultra fast performance with analytics capabilities using **Columnar Indexes**
- MariaDB Xpand is available in **MariaDB SkySQL** DBaaS as well as on-prem

A Slice of Xpand

MariaDB Xpand - Slices

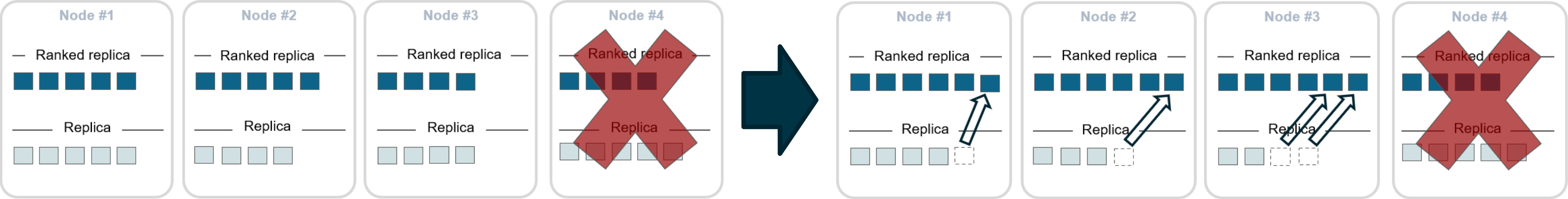
- Tables and Indexes in MariaDB Xpand are **divided into *Slices***
- A table / index is distributed by **distributing the slices**
- There are **multiple copies** of each slice in **different nodes**
- The slices are **synchronously replicated**
- One copy is considered a **ranked replic**



MariaDB

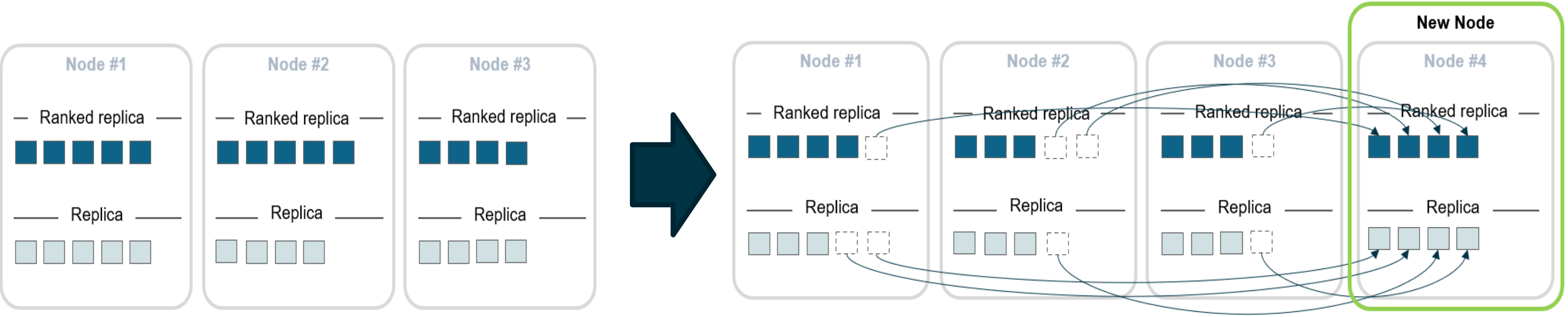
MariaDB Xpand – High Availability

- If a node fails, there are always **copies of slices in** the surviving nodes
- When this happens, first non-ranked replicas are **promoted to ranked replicas**
- Secondly, **new non-ranked replicas** are created in the surviving nodes
- With **No down-time**!



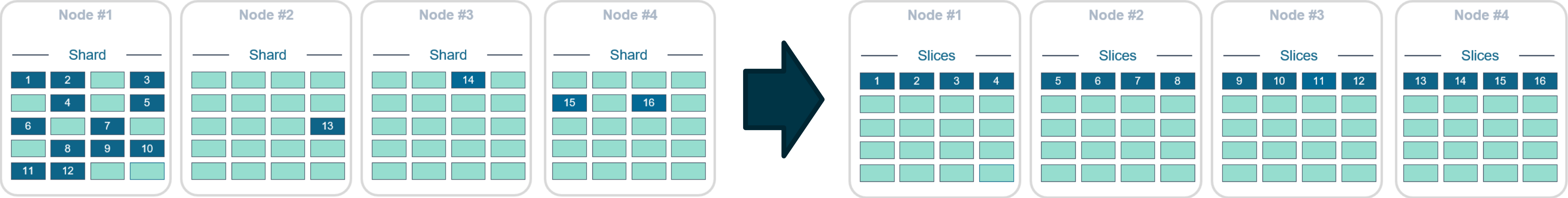
MariaDB Xpand – Elasticity

- Nodes can be **added to an existing cluster**
- Slices will be **automatically re-distributed** to the new nodes
- With **No downtime**!



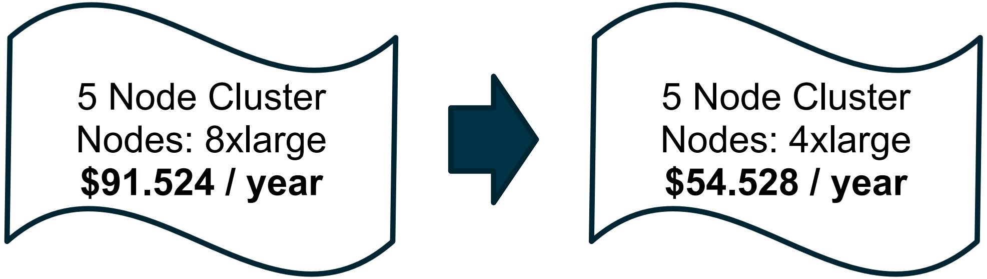
MariaDB Xpand – Balancing

- Slices are **balanced** for best processing and workload distribution
- Balancing is largely done **without moving** or **copying** data
- Due to this workloads move in a running system and **with minimal overhead**
- With **No downtime**!



MariaDB Xpand – Resource usage and cost

- Data and processing is balanced with **no movement of data**
- Hardware can be **utilized optimally**
- Even with a **varying workload** and when the system is live
- Allowing for use of **lower cost servers**
- With **No Downtime**!



5 Node Cluster
Nodes: 8xlarge
$91.524 / year

5 Node Cluster
Nodes: 4xlarge
$54.528 / year

Xpand Storage Heirarchy

Distribution terminology

Distribution Key

Each table and index has a distribution key. Xpand hashes the distribution key to determine which slice owns the row or index entry.

The Primary Key functions as the distribution key for each table, and the indexed columns function as the distribution key for each index.

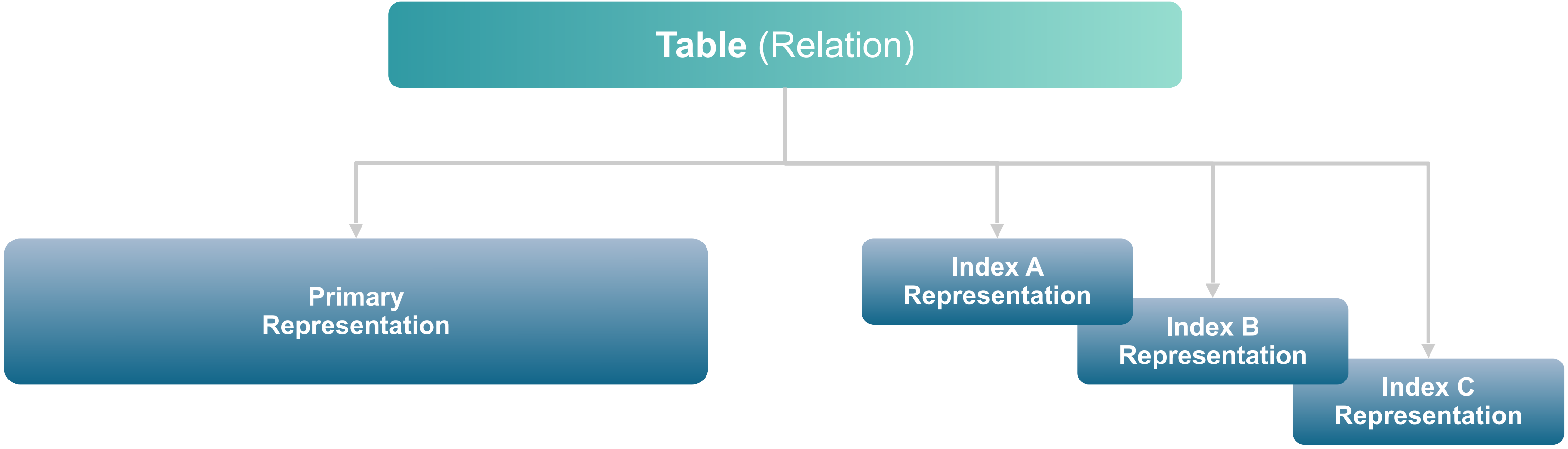
Slices

Each table and index are distributed independently among the nodes, in chunks called slices.

Replicas

Xpand maintains multiple copies of each slice for fault tolerance. The copies are called replicas. When a slice does not have a sufficient number of replicas for fault tolerance, the Xpand Rebalancer automatically creates new replicas of that slice.

Xpand Storage Hierarchy



Primary representation contains all row data

Index representation(s) contain key column(s) and primary key

Xpand Distribution key

MariaDB Xpand uses a hash to determine where a given row of data or a table's index (`representation`) should reside in the cluster

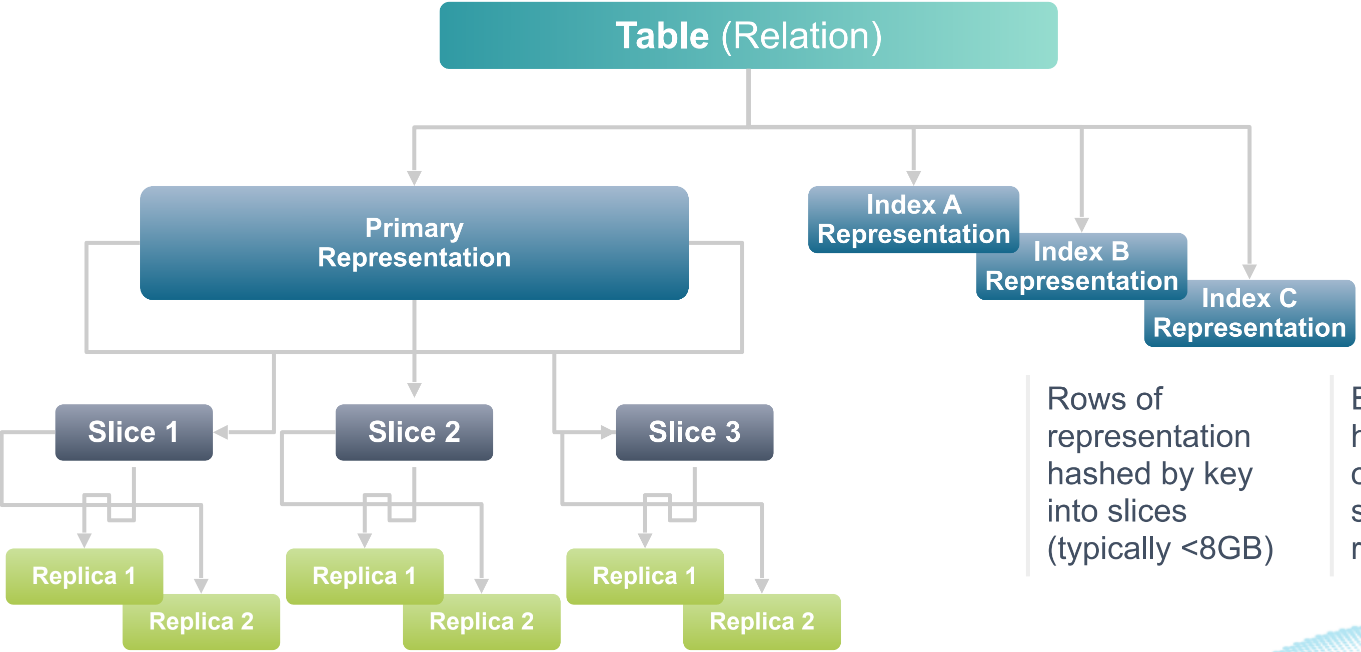
The columns selected for hashing are referred to as the `distribution key` for that representation.

By default, the `distribution key` uses the first column of an index, regardless of how many columns comprise the index. This is true for all indices including the primary key.

MariaDB Xpand uses independent index distribution rather than a single-key for tables and indexes

- This allows for a much broader range of distributed query plans that scale with cluster node count
- This requires strict support within the system to guarantee that indexes stay consistent with each other and the main table

Xpand Storage Hierarchy



Rows of representation hashed by key into slices (typically <8GB)

Each slice has two or more synchronous replicas

Distribution of Slices and Replicas

Each node has slices of each representation

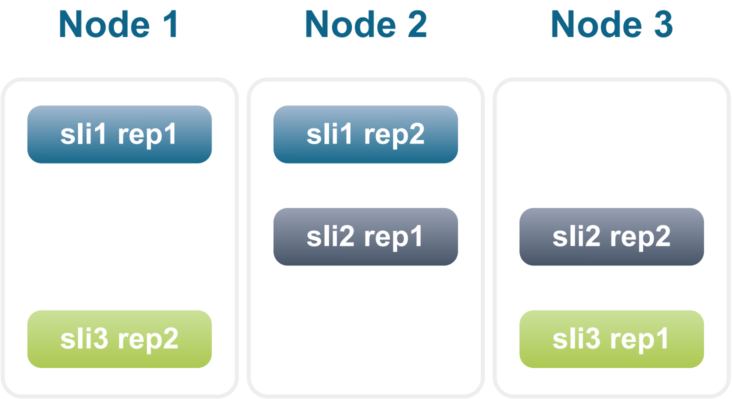
- This distributes load across all nodes

Every slice has at least two replicas (by default)

- On different nodes
- On different Zones (If zone configured)
- This enables fault tolerance

Result

- All nodes (zones) have an equal amount of data
- No slice is lost when 1 node (zone) fails



The Xpand Rebalancer

MariaDB

The **Perfect Distribution**

The Rebalancer is the key to maintaining the perfect distribution

Perfect distribution at the start is easy

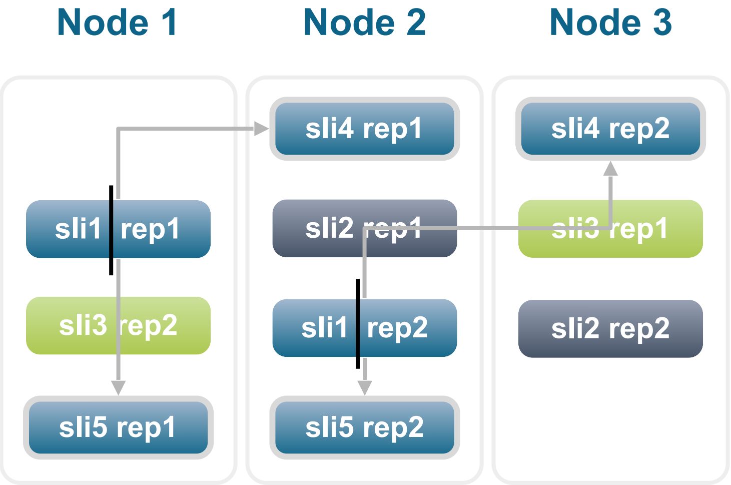
But what happens when…

1	**The table grows much larger?**	**Split** the slices
2	**A node is added?**	**Move** replicas to new node
3	**We lose a node or disk?**	**Copy** to make new replicas
4	**Read imbalance?**	**Rerank** replicas

Splitting slices to accommodate growth

When slice grows past 8GB limit

1. Split slice into two new slices
2. Dispose of old slices
3. Populate two replicas of each new slice



Slice Size

The default max slice size is 8GB

You may want to have larger slice sizes if you have tables greater than 100GB

- To reduce database's overhead of slice management
- For example 1000 slices in a single table is probably to many

A **100GB table** would have

1GB slices = 100 slices (old default)

2GB slices = 50 slices (common)

8GB slices = 13 slices (new default)

16GB slices = 7 slices (extreme cases)

More relevant with larger tables, but no impact to smaller tables

Because slice size is controlling the max size, not the min size

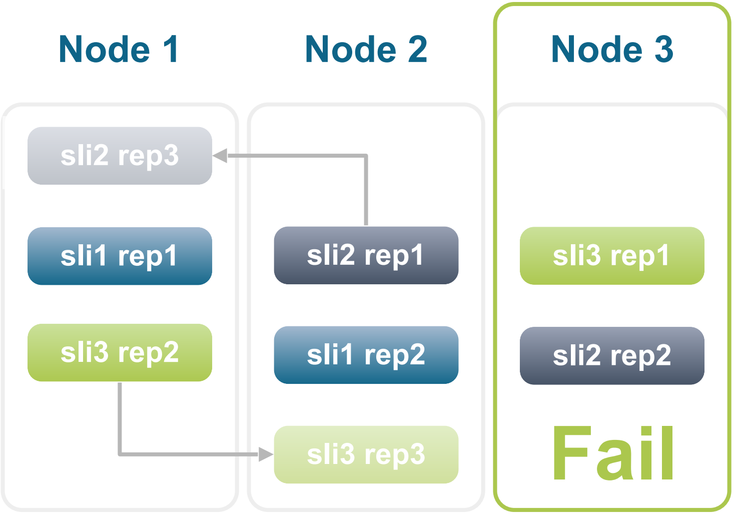
Global variable

`rebalancer_split_threshold_kb`

This is a cluster-wide setting (not per-table)

Reprotect by Re-creating missing replicas

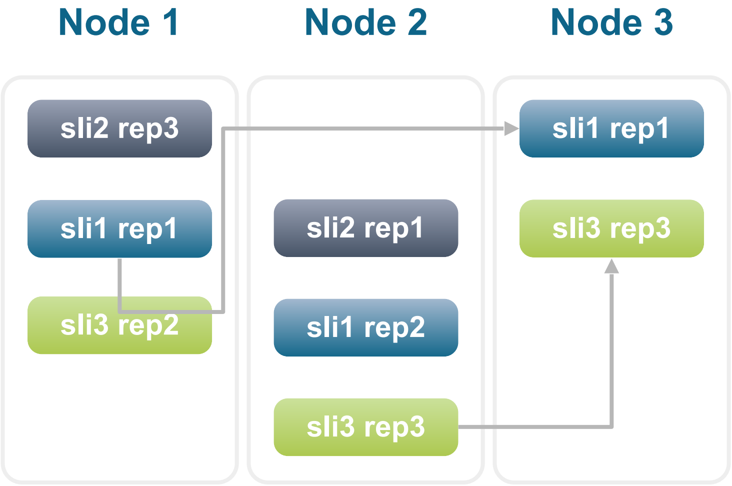
In the event of node failure, surviving replicas are used to `reprotect` thus restoring full fault tolerance



Rebalance after adding a node

A new node is added to the cluster

And replicas are moved (a.k.a. **`rebalanced`**) to evenly distribute data

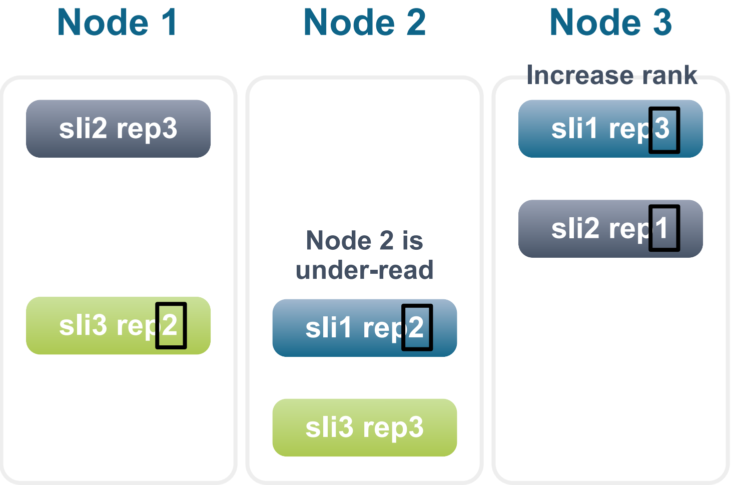


Rerank to Distribute Read Load

Xpand always reads from the lowest-rank replica

This maximizes cache efficiency

`Rerank` evens out read imbalance by changing read preference to other replica



Conclusion

MariaDB Xpand – Conclusion

- Distributed SQL means data and processing is **distributed** and **scalable**
- MariaDB Xpand is a Distributed SQL solution with **all nodes in the cluster created equal** to process any data item
- MariaDB Xpand allows data to be **distributed without it being physically moved or copied**
- MariaDB Xpand provides **built-in high availability**
- MariaDB Xpand is truly elastic, **allowing nodes to be added as needed** to a running cluster
- MariaDB Xpand combines scalability benefits of a Distributed SQL database solution with **Analytical indexes**

MariaDB
OPENWORKS

THANK YOU

MariaDB

OPEN WORKS

BE UNSTOPPABLE