How Safe is Asynchronous Active-Active Setup in MySQL

PERCONA
TECH DAYS

September 10, 2020

Sveta Smirnova

Sveta Smirnova

- MySQL Support engineer
- Author of
  - MySQL Troubleshooting
  - JSON UDF functions
  - FILTER clause for MySQL
- Speaker
  - Percona Live, OOW, Fosdem, DevConf, HighLoad...

Table of Contents

Asynchronous Built-In Replication

Source

Replica

← Initiates

Asynchronous Built-In Replication

Source

Replica

← Initiates

← Requests a packet

Asynchronous Built-In Replication

Source	Replica
	← Initiates
	← Requests a packet
Sends the packet →	

Asynchronous Built-In Replication

Source	Replica
	← Initiates
	← Requests a packet
Sends the packet →	
	... ?

Asynchronous Two Active Nodes

Node1
Initiates

Node2
Initiates

Asynchronous Two Active Nodes

Node1		Node2
Initiates	→ ←	Initiates
Requests a packet	→ ←	Requests a packet

Asynchronous Two Active Nodes

Node1			Node2
Initiates	→	←	Initiates
Requests a packet	→	←	Requests a packet
Sends the packet	→	←	Sends the packet

Asynchronous Two Active Nodes

Node1			Node2
Initiates	→	←	Initiates
Requests a packet	→	←	Requests a packet
Sends the packet	→	←	Sends the packet
	... ?		... ?

What Happens with 2 Active Nodes?

INSERT

Node1

```
INSERT SET id=42
```

Node2

```
INSERT SET id=42
```

INSERT

Node1

`INSERT SET id=42`

Sends the **update** →

Node2

`INSERT SET id=42`

← Sends the **update**

INSERT

Node1		Node2
`INSERT SET id=42`		`INSERT SET id=42`
Sends the **update**	→	
	←	Sends the **update**
Receives **update**	→	
	←	Receives the **update**

INSERT

Node1		Node2
`INSERT SET id=42`		`INSERT SET id=42`
Sends the **update**	→ ←	Sends the **update**
Receives **update**	→ ←	Receives the **update**
Duplicate key error!		**Duplicate key error!**

DELETE

Node1

```
DELETE WHERE id=42
```

Node2

```
DELETE WHERE id=42
```

DELETE

Node1

`DELETE WHERE id=42`

Sends the **update** →

Node2

`DELETE WHERE id=42`

← Sends the **update**

DELETE

Node1		Node2
`DELETE WHERE id=42`		`DELETE WHERE id=42`
Sends the **update**	→ ←	Sends the **update**
Receives **update**	→ ←	Receives the **update**

DELETE

Node1		Node2
`DELETE WHERE id=42`		`DELETE WHERE id=42`
Sends the **update**	→ ←	Sends the **update**
Receives **update**	→ ←	Receives the **update**
Key not found error!		**Key not found error!**

UPDATE

Node1

```
UPDATE SET i=42
```

Node2

```
UPDATE SET i=25
```

UPDATE

Node1

`UPDATE SET i=42`

Sends the **update** →

Node2

`UPDATE SET i=25`

← Sends the **update**

UPDATE

Node1		Node2
`UPDATE SET i=42`		`UPDATE SET i=25`
Sends the **update** →	←	Sends the **update**
Receives **update** →	←	Receives the **update**

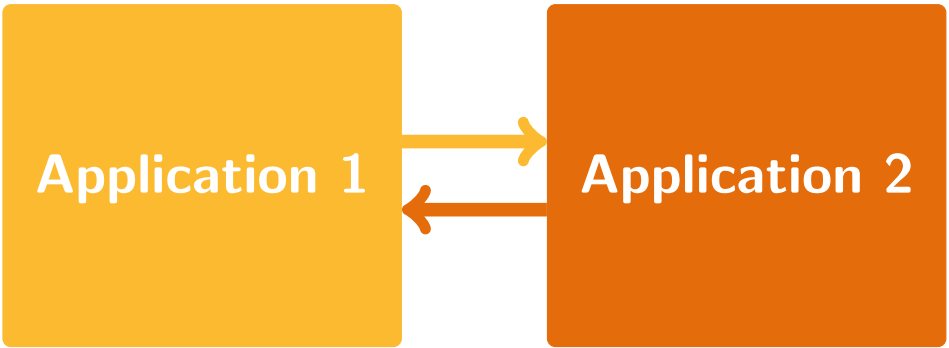
UPDATE

Node1		Node2
`UPDATE SET i=42`		`UPDATE SET i=25`
Sends the **update** →	←	Sends the **update**
Receives **update** →	←	Receives the **update**
`i=25`		`i=42`

Why to Write to Multiple Nodes?

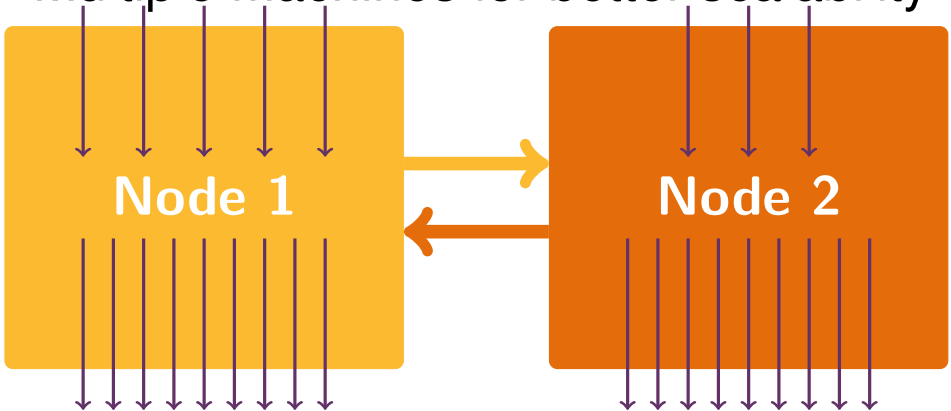
Data Distribution

- Each node for its own purpose



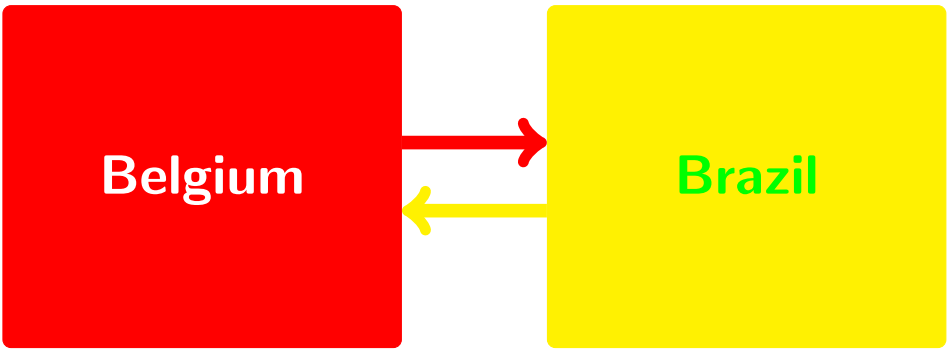
[Read] Scale

- Multiple machines for better scalability



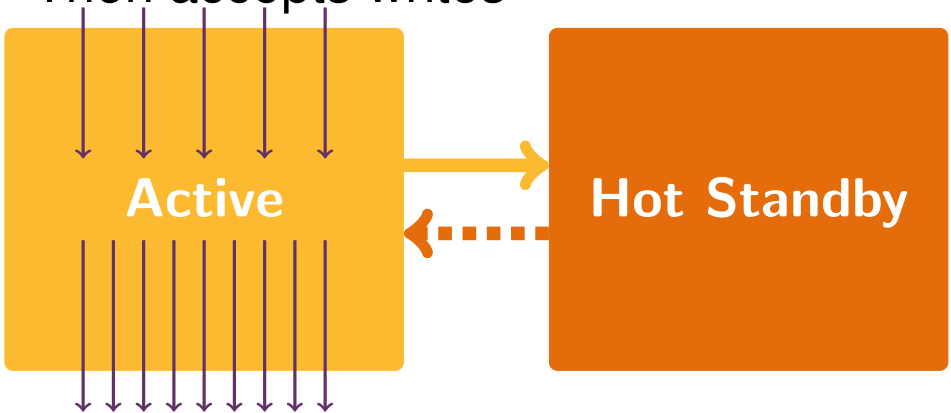
GEO Distribution

- Nodes in different geographical regions



Hot Standby

- Waits when the active node dies
- Then accepts writes



Galera, PXC, InnoDB Cluster

Fully Synchronous Cluster

Node1

```
UPDATE ...
```

Node2,3,...

Fully Synchronous Cluster

Node1

`UPDATE ...`

Sends the packet $\rightarrow$

Node2,3,...

Fully Synchronous Cluster

Node1

`UPDATE ...`

Sends the packet $\rightarrow$

Waits

Node2,3,...

$\leftarrow$ Receives the packet

Fully Synchronous Cluster

Node1		Node2,3,...
`UPDATE ...`		
Sends the packet	$\rightarrow$	
Waits		
Waits	$\leftarrow$	Receives the packet
		Applies changes

Fully Synchronous Cluster

Node1	Node2,3,...
`UPDATE ...`	
Sends the packet →	
Waits	← Receives the packet
Waits	Applies changes
Waits	← Confirms

Fully Synchronous Cluster

Node1	Node2,3,...
`UPDATE ...`	
Sends the packet →	
Waits	← Receives the packet
Waits	Applies changes
Waits	← Confirms
Receives answer →	

Fully Synchronous Cluster

Node1	Node2,3,...
`UPDATE ...`	
Sends the packet →	
Waits	← Receives the packet
Waits	Applies changes
Waits	← Confirms
Receives answer →	
Ready for new update	

Synchronization Penalty

- Speed of the slowest member

Synchronization Penalty

- Speed of the slowest member
- Writes or reads should wait

Synchronization Penalty

- Speed of the slowest member
- Writes or reads should wait
- Tunable

Synchronization Penalty

- Speed of the slowest member
- Writes or reads should wait
- Tunable
- Galera
  - `wsrep_sync_wait`
- InnoDB Cluster
  - `group_replication_consistency`

GEO Replication is Painful

- Synchronization takes too much time
- Nodes often disconnect

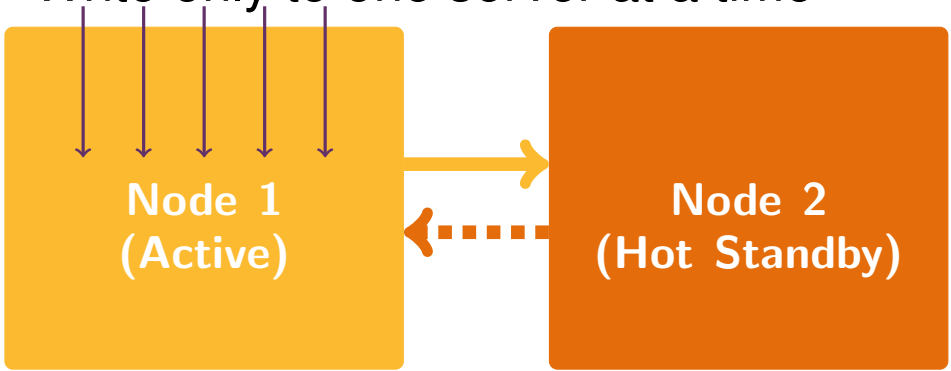
How to Setup Safe Active-Active

Divide and Rule

- Business logic
- Servers
- Databases
- Tables
- Rows

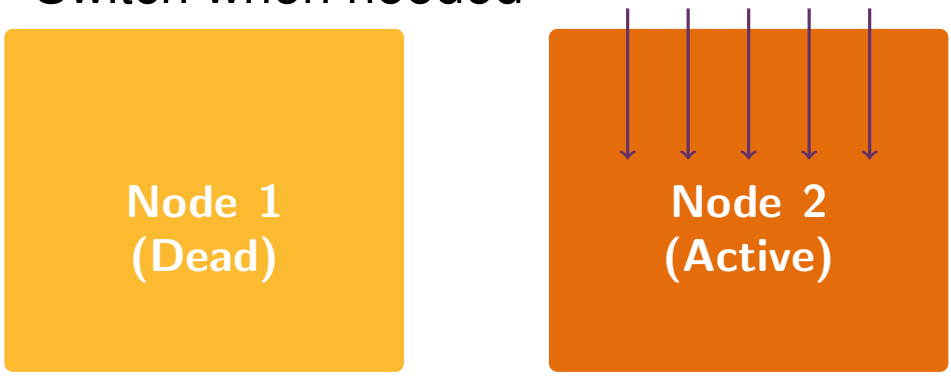
Servers

- Write only to one server at a time



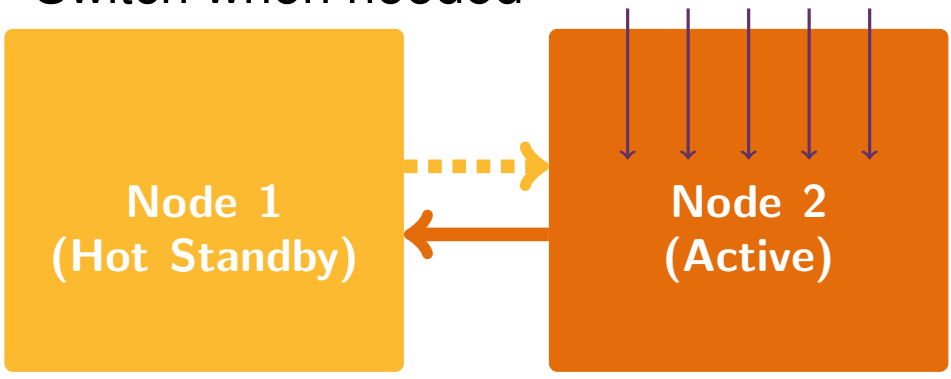
Servers

- Write only to one server at a time
- Switch when needed



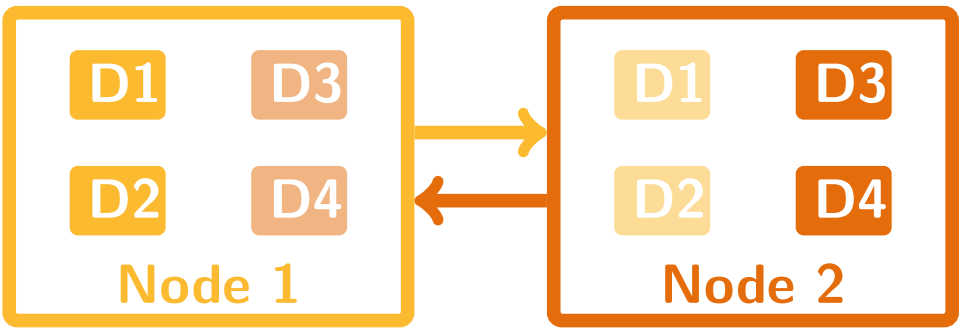
Servers

- Write only to one server at a time
- Switch when needed



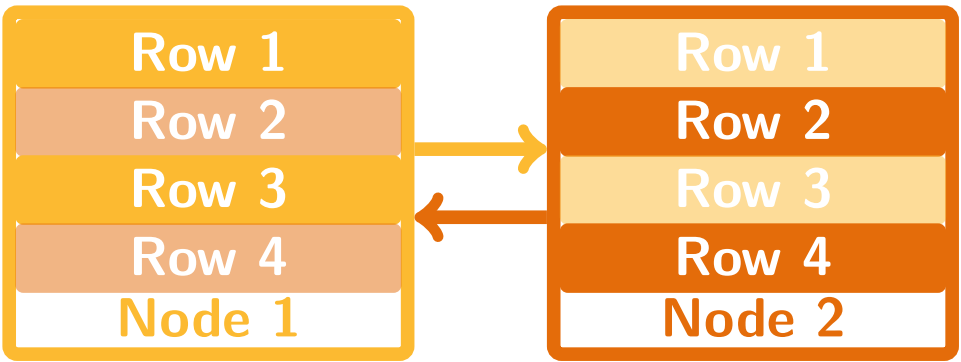
Databases and Tables

- Each server writes to its own set of tables



Rows

- Agreement on which rows



Rows

- Agreement on which rows
- E.g. `AUTO_INCREMENT`
  - `auto_increment_increment=NUMBER_OF_SERVERS`
  - `auto_increment_offset = server_id`
- Custom index
  - Create your own unique pattern for each node

Conclusion

- Hot Standby
  - Easy to setup and maintain
- Separating writes by database/table
  - Comparatively easy to setup and maintain
- Separating writes by row
  - Should be used very carefully

More information


MySQL User Reference Manual


Why MySQL Replication Fails, and How...


MySQL High Availability

Thank you!


www.slideshare.net/SvetaSmirnova


twitter.com/svetsmirnova


github.com/svetasmirnova