Mix ‘n’ Match Async and Group Replication for Advanced Replication Setups

Pedro Gomes (pedro.gomes@oracle.com)
Software Engineer

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

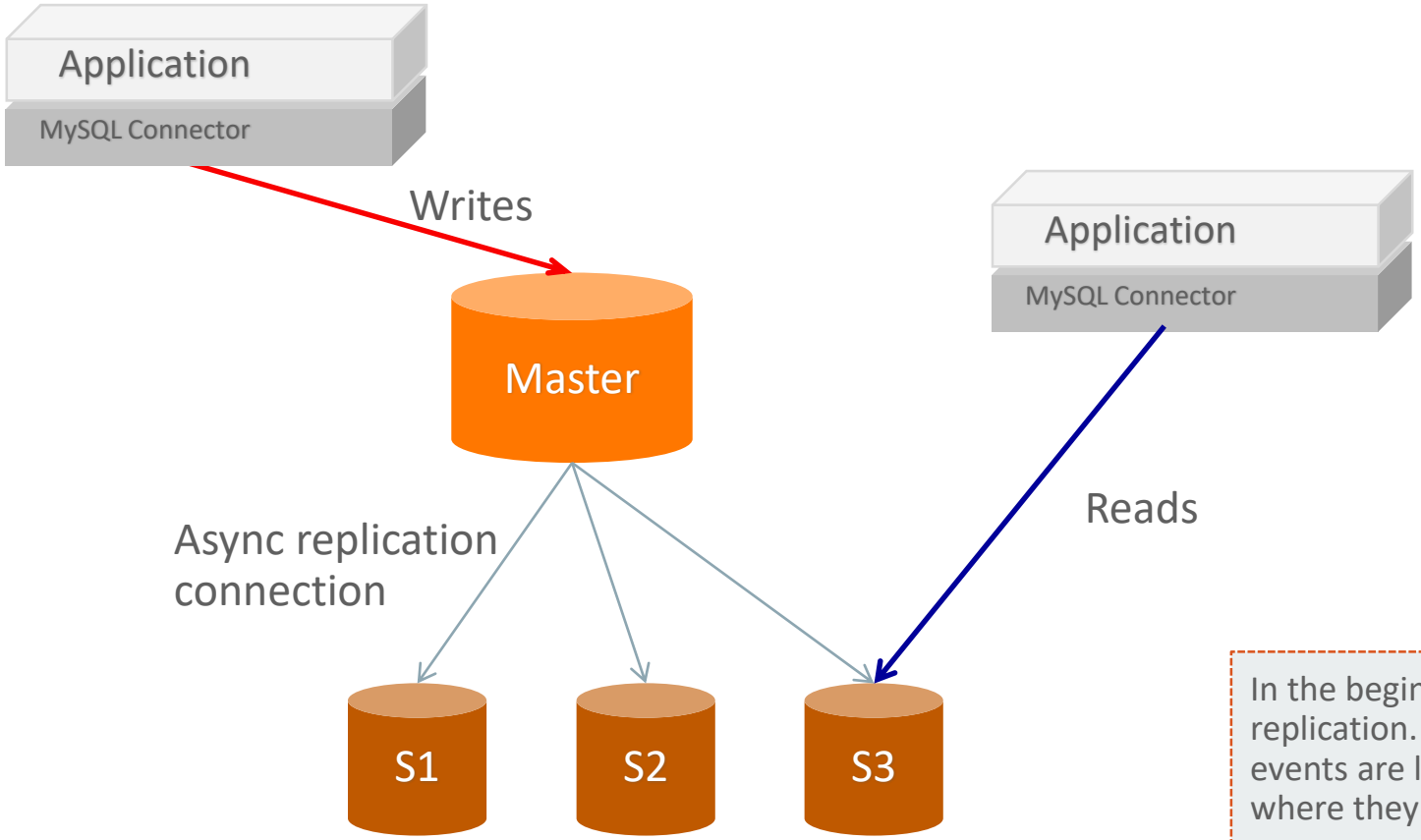
Program Agenda

Program Agenda

1. Background
2. Mix it!
3. Migrations
4. Conclusion

1 Background

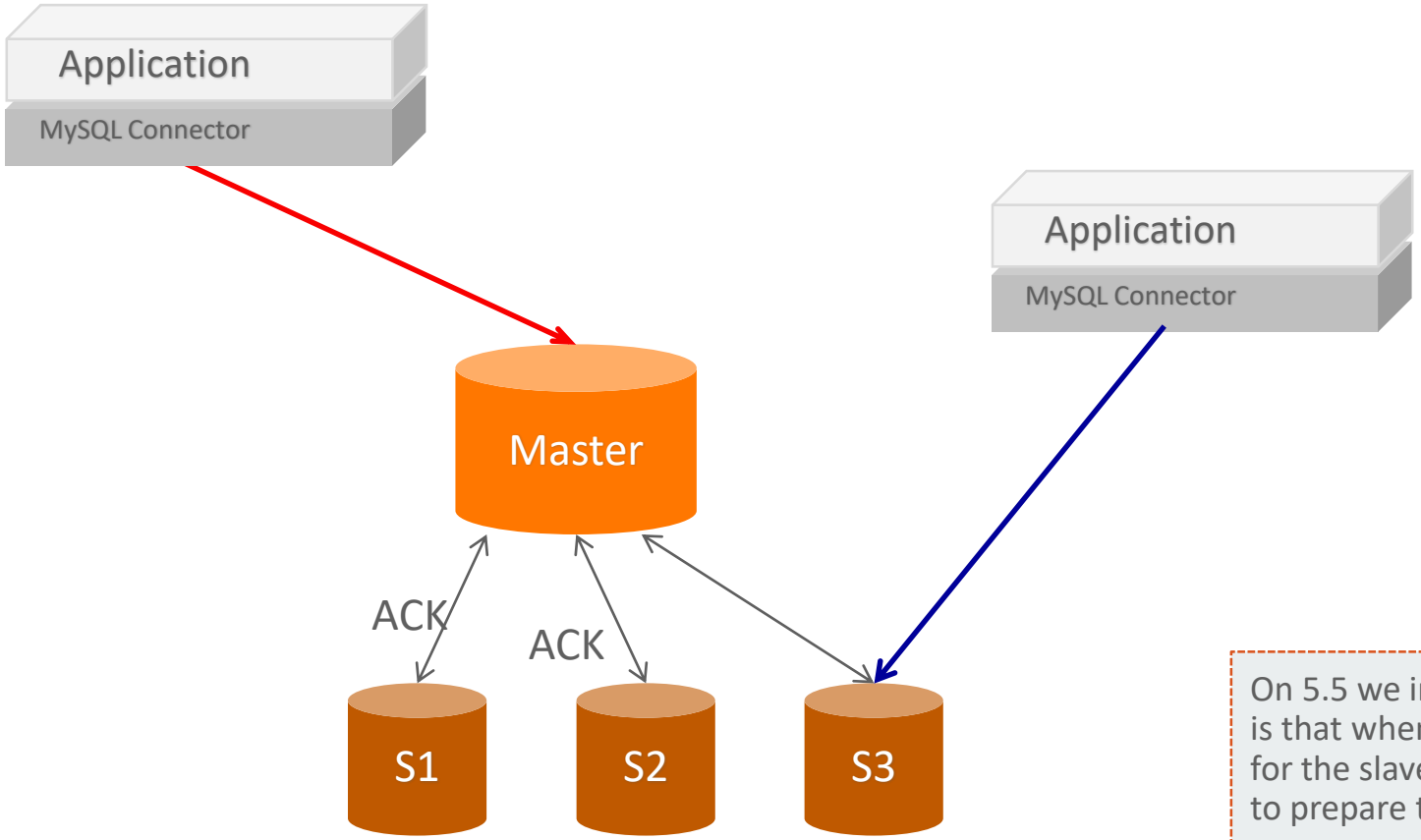
In the beggining there was Async Replication



In the beginning there was asynchronous replication. You write in the master where events are logged and sent to the slaves where they are queued and applied.

This can be used for data safety on master crashes or read scalability

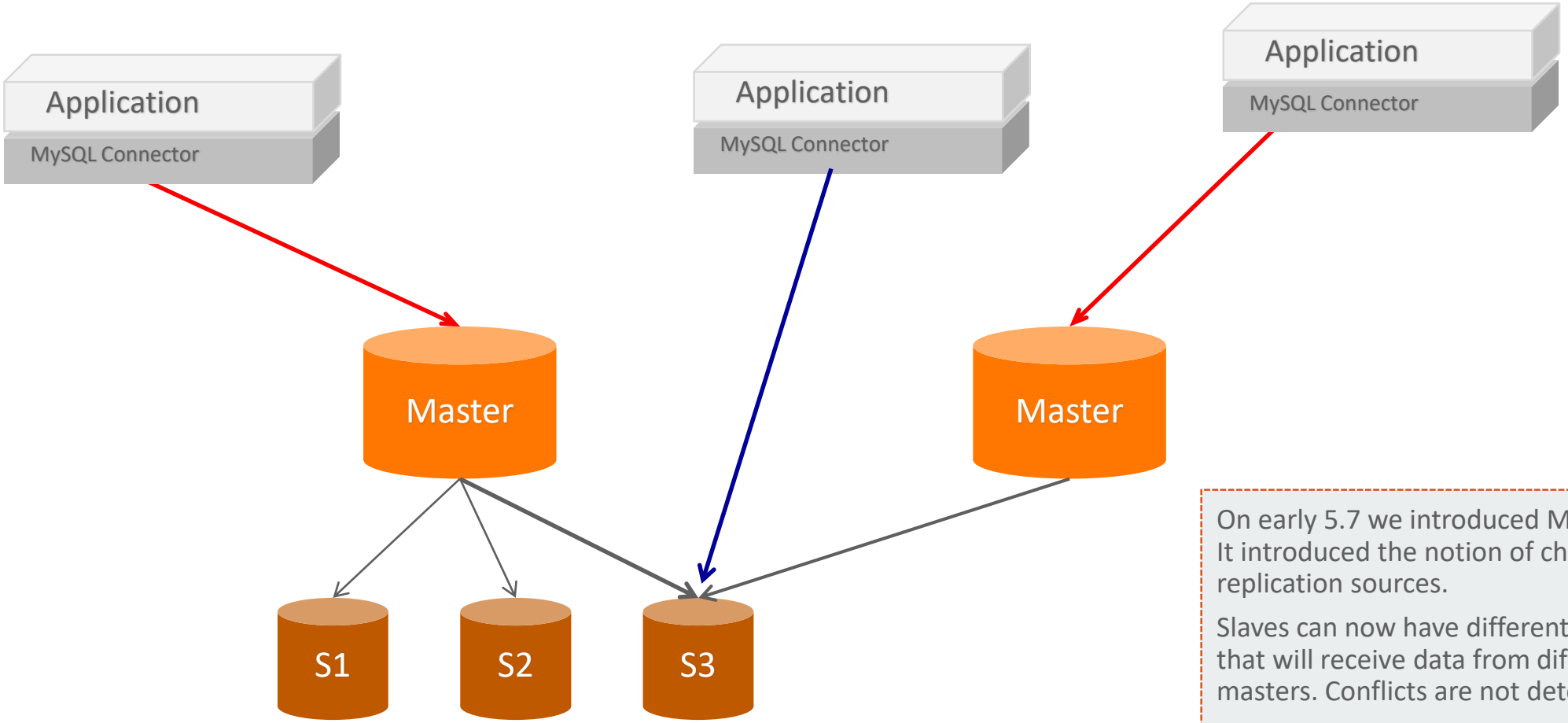
Semi sync Replication



On 5.5 we introduced semi sync. The idea is that when the master commits it waits for the slaves to queue the transaction or to prepare the commit.

You can wait on all or on a set number of slaves.

Multi Source Replication

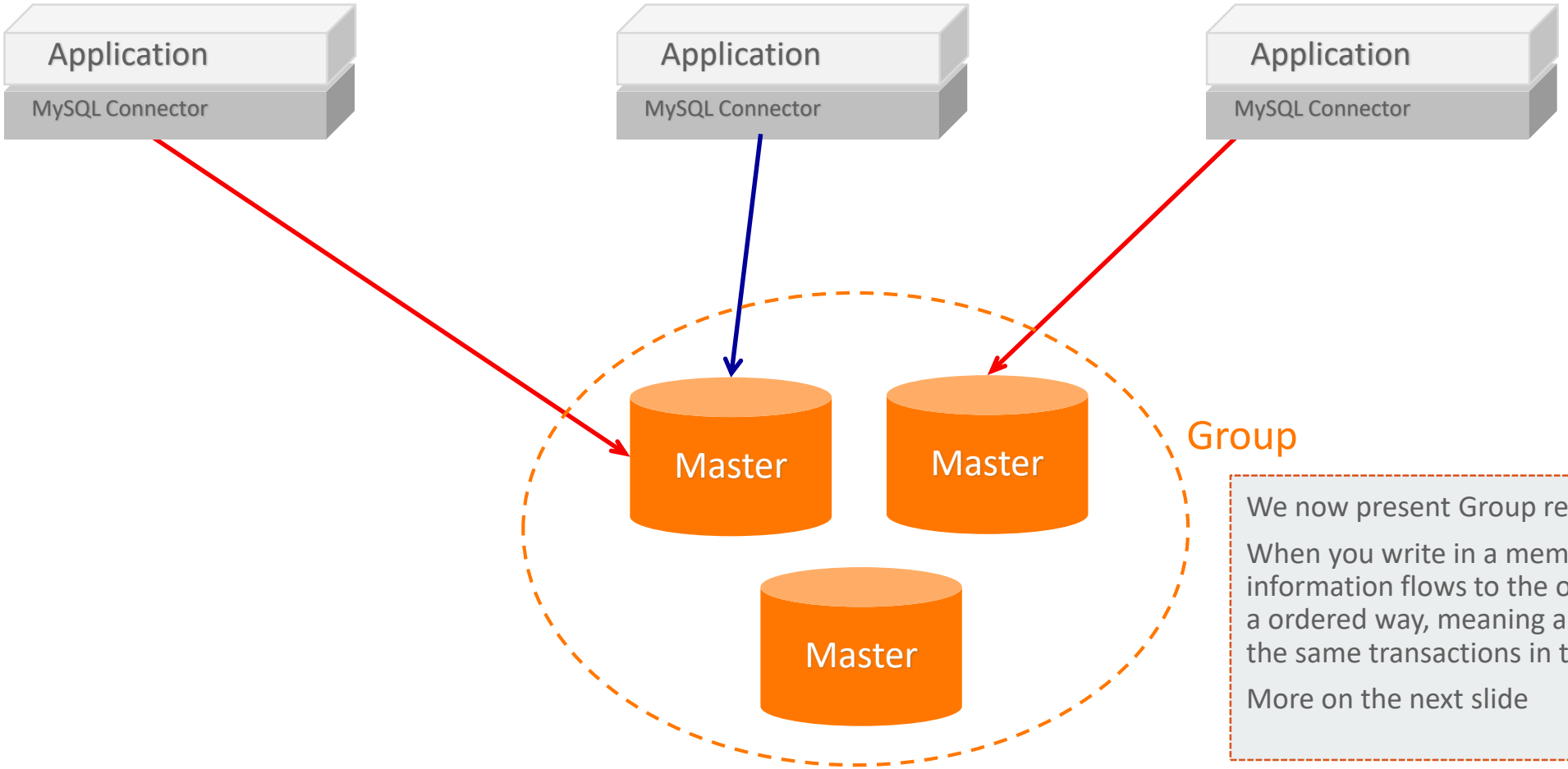


On early 5.7 we introduced Multi source. It introduced the notion of channel, i.e., a replication sources.

Slaves can now have different channels that will receive data from different masters. Conflicts are not detected.

This can be used for data aggregation or full backups of sharded data

Group Replication



We now present Group replication.

When you write in a member, the information flows to the other members in a ordered way, meaning all members get the same transactions in the same order.

More on the next slide

MySQL Group Replication

- **What is MySQL Group Replication?**

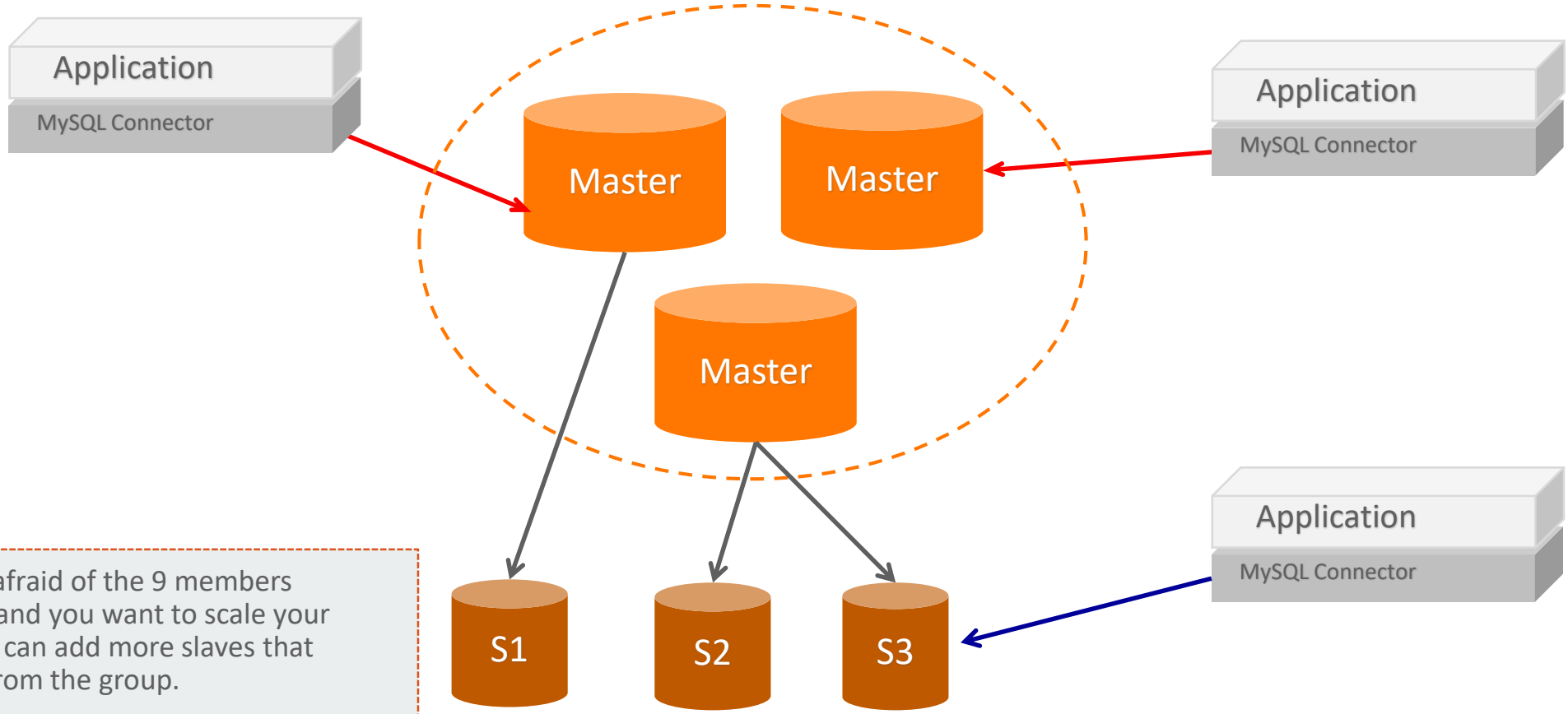
  "Multi-master **update everywhere** replication plugin for MySQL with built-in **automatic distributed recovery**, **conflict detection** and **group membership**."

- **What does the MySQL Group Replication plugin do for the user?**
  - Removes the need for handling server fail-over.
  - Provides fault tolerance.
  - Enables update everywhere setups.
  - Automates group reconfiguration (handling of crashes, failures, re-connects).
  - Provides a highly available replicated database.

2 Mix it!

2.1 Basic scenarios

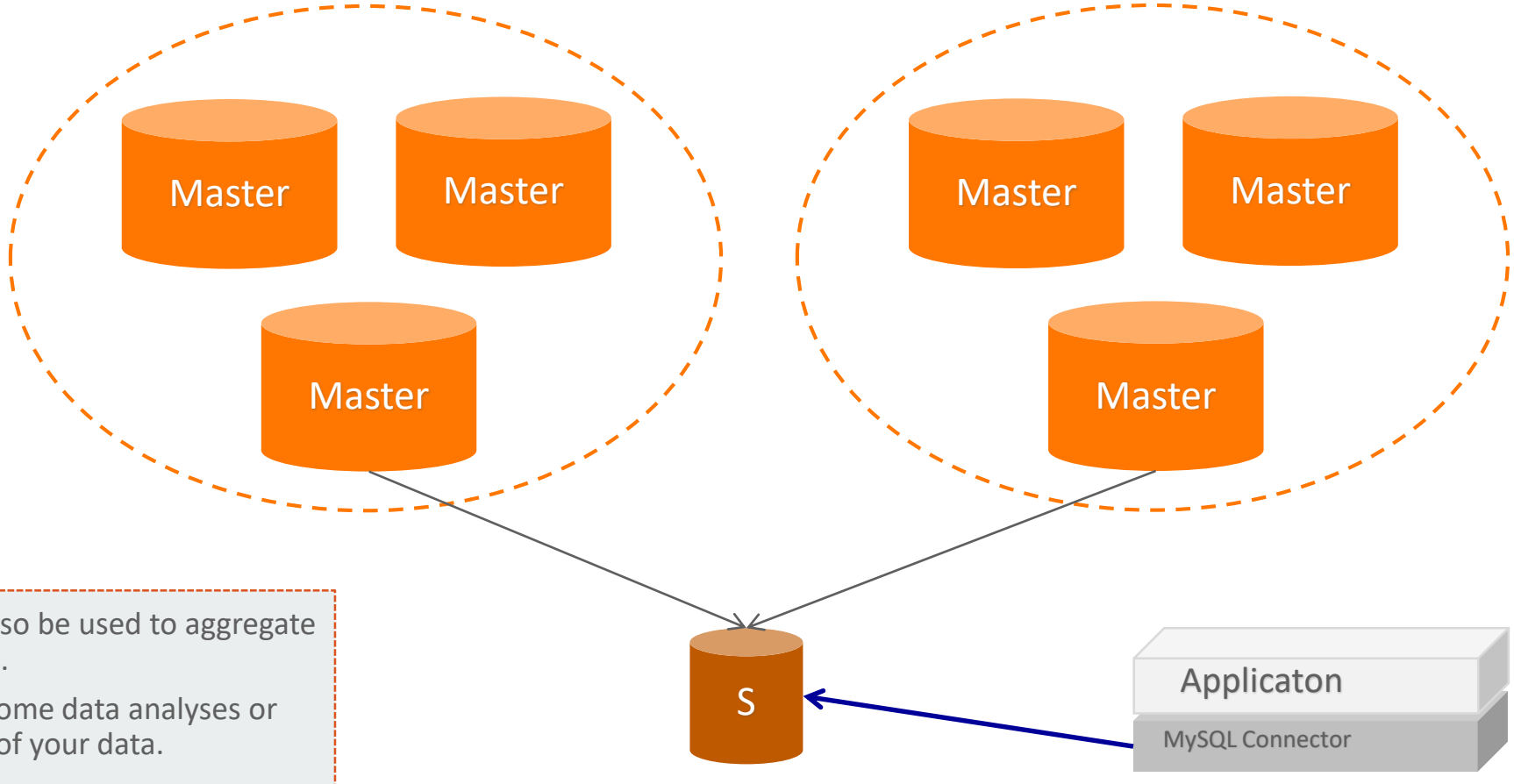
Read out Scalability



If you are afraid of the 9 members limitation and you want to scale your reads, you can add more slaves that replicate from the group.

Remember that the group has automatic failover, but not your slaves.

Aggregate some data



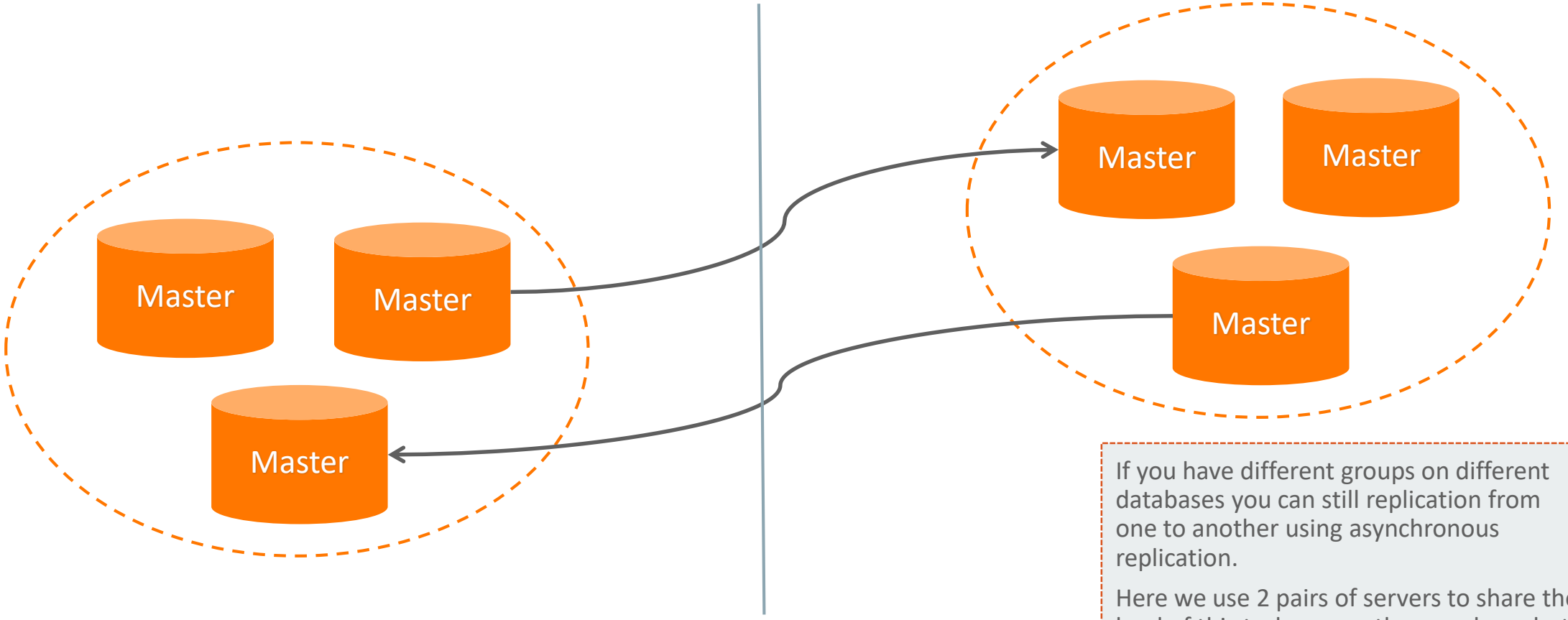
Multi source can also be used to aggregate data from 2 groups.

Again you can do some data analyses or have a full backup of your data.

2 Mix it!

2.2 Advanced scenarios

Inter data center replication



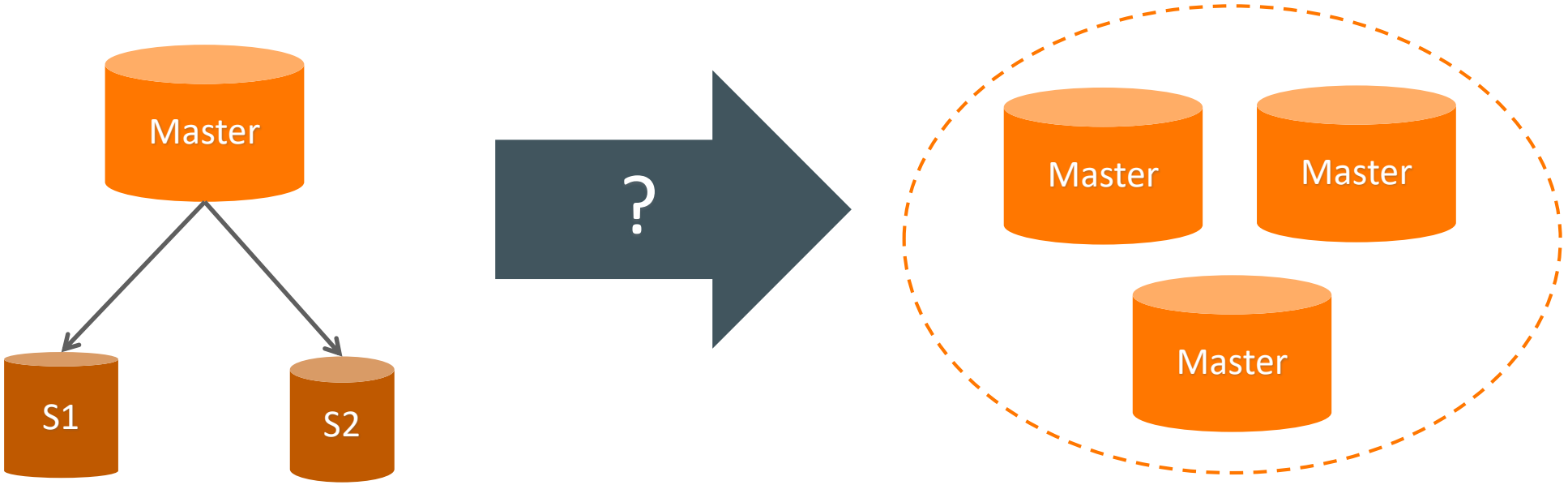
If you have different groups on different databases you can still replication from one to another using asynchronous replication.

Here we use 2 pairs of servers to share the load of this task among the members, but you can use only one circular pair or have even more connections if desired.

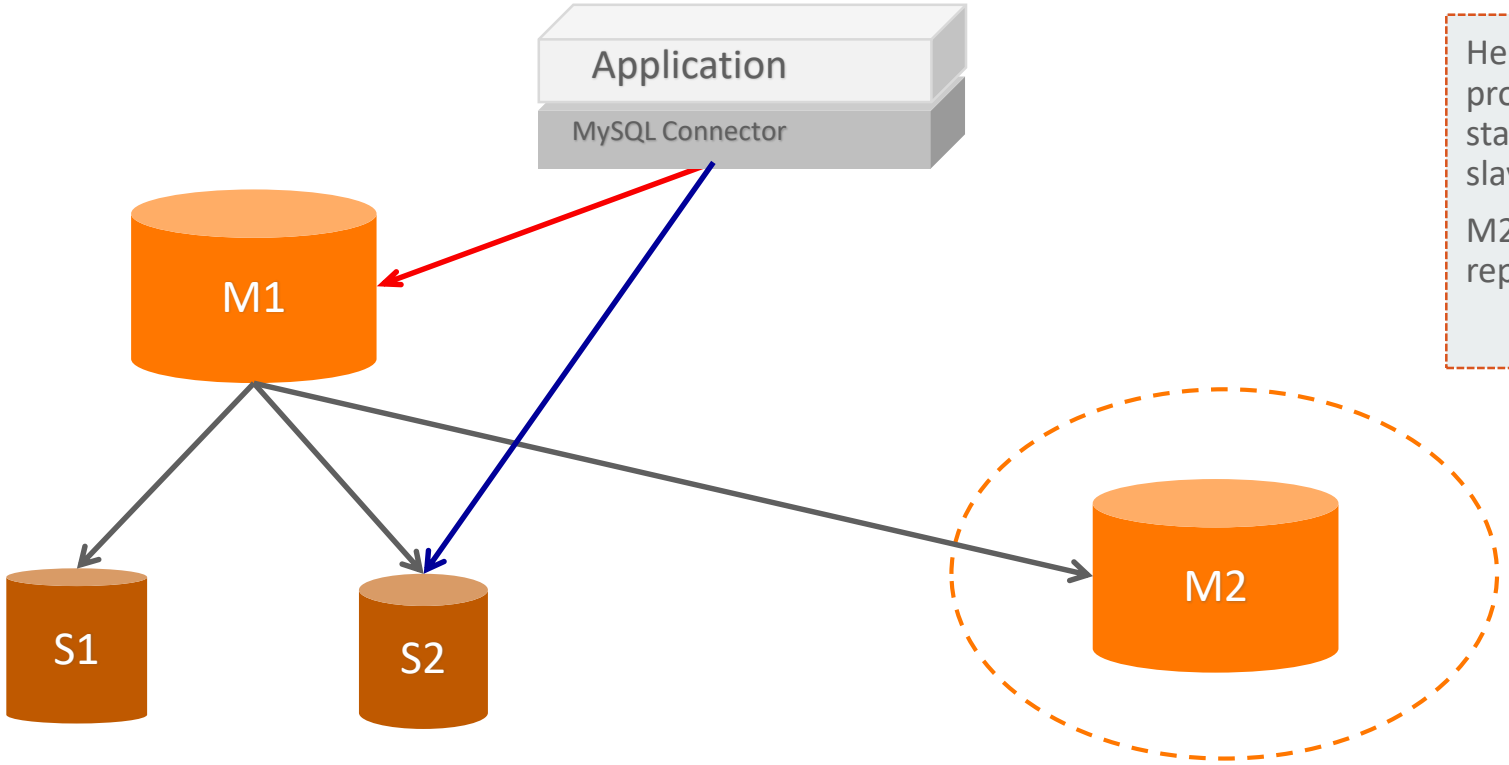
3 Migrations

3.1 How to migrate from asynchronous replication

How to migrate from asynchronous scenarios



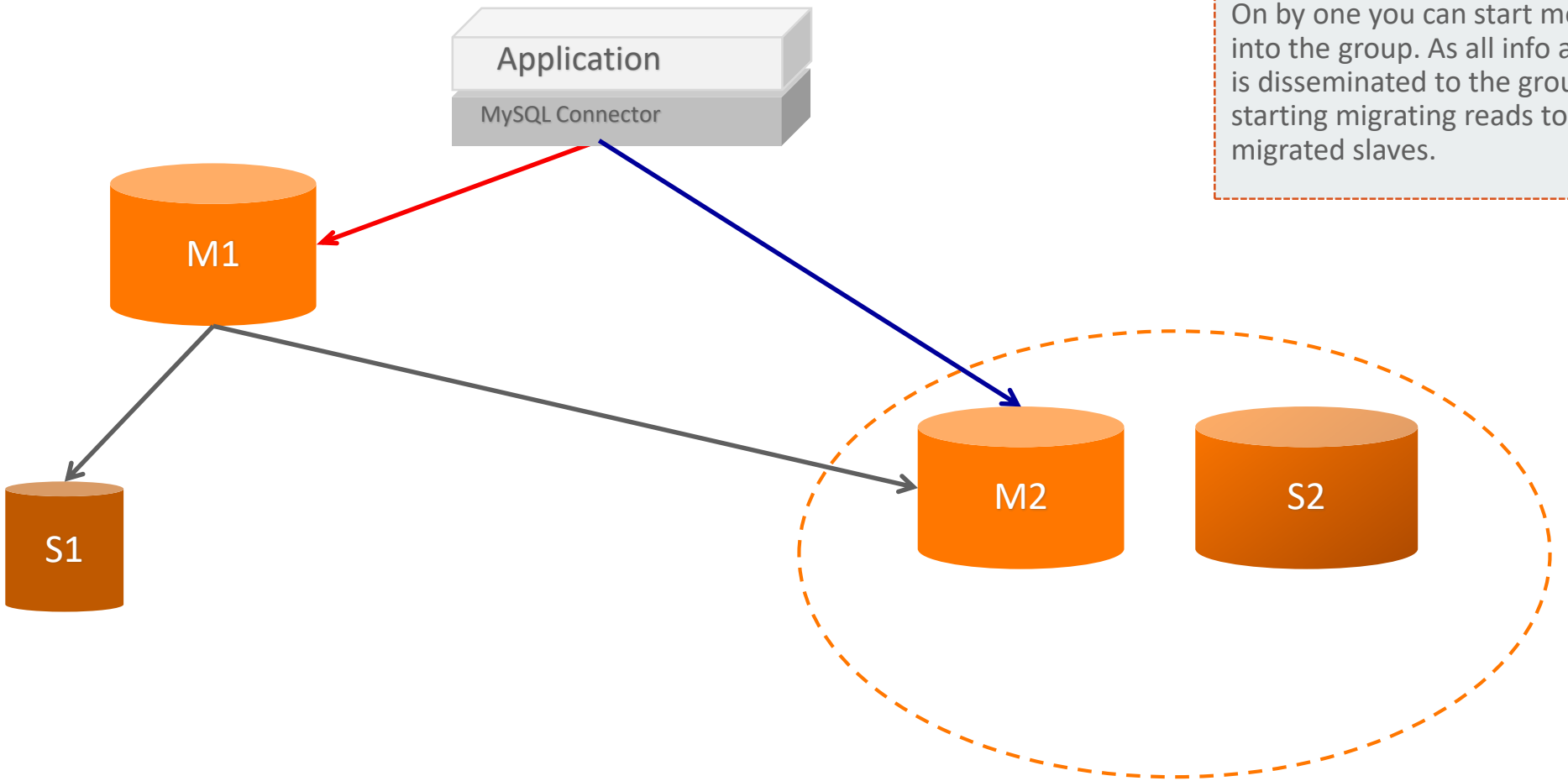
How to migrate from asynchronous scenarios



Here I chose to start the group in a new provisioned member but you can also start by doing this in on of the existing slaves .

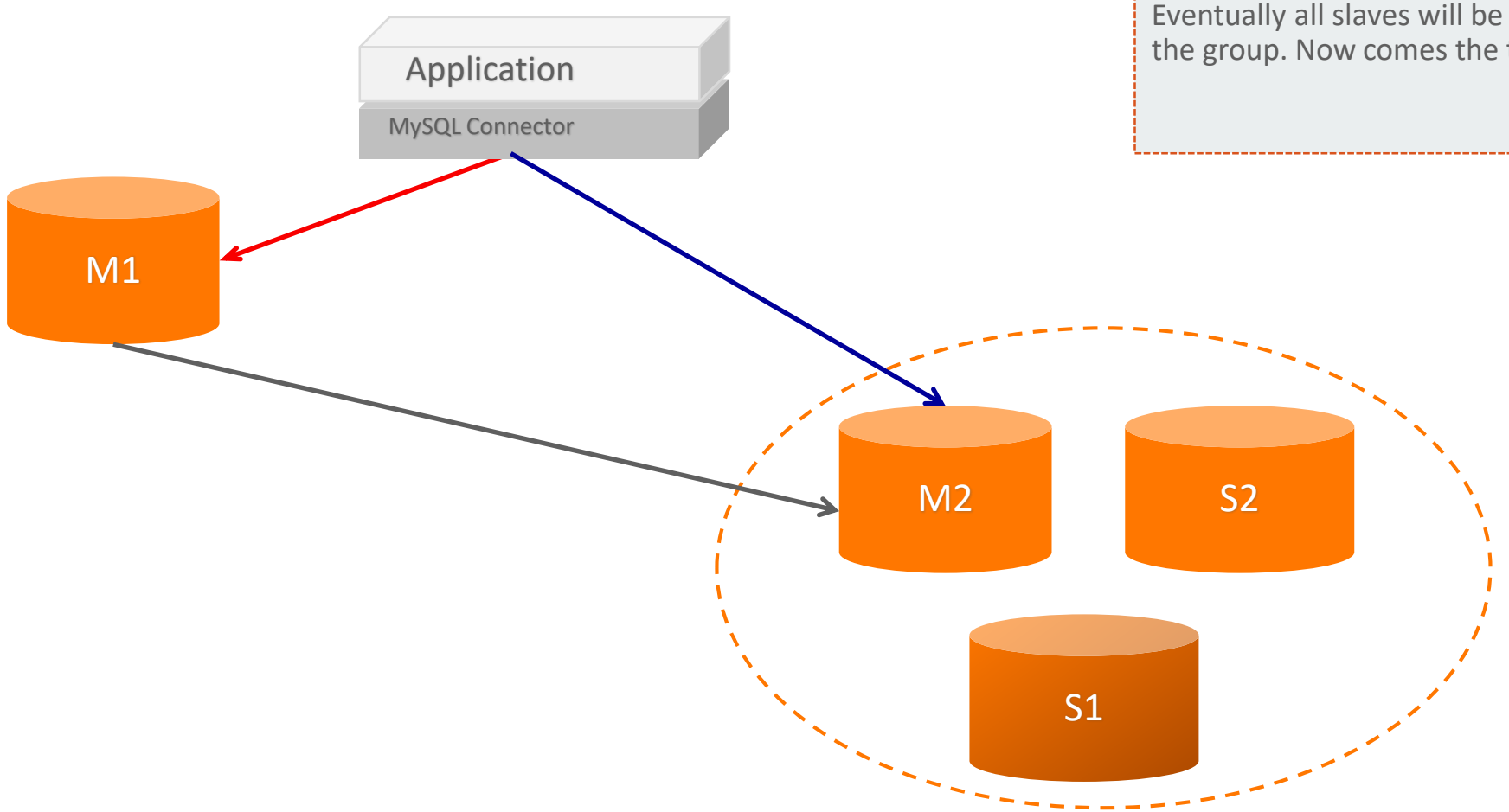
M2 will replicate from M1 using async replication.

How to migrate from asynchronous scenarios



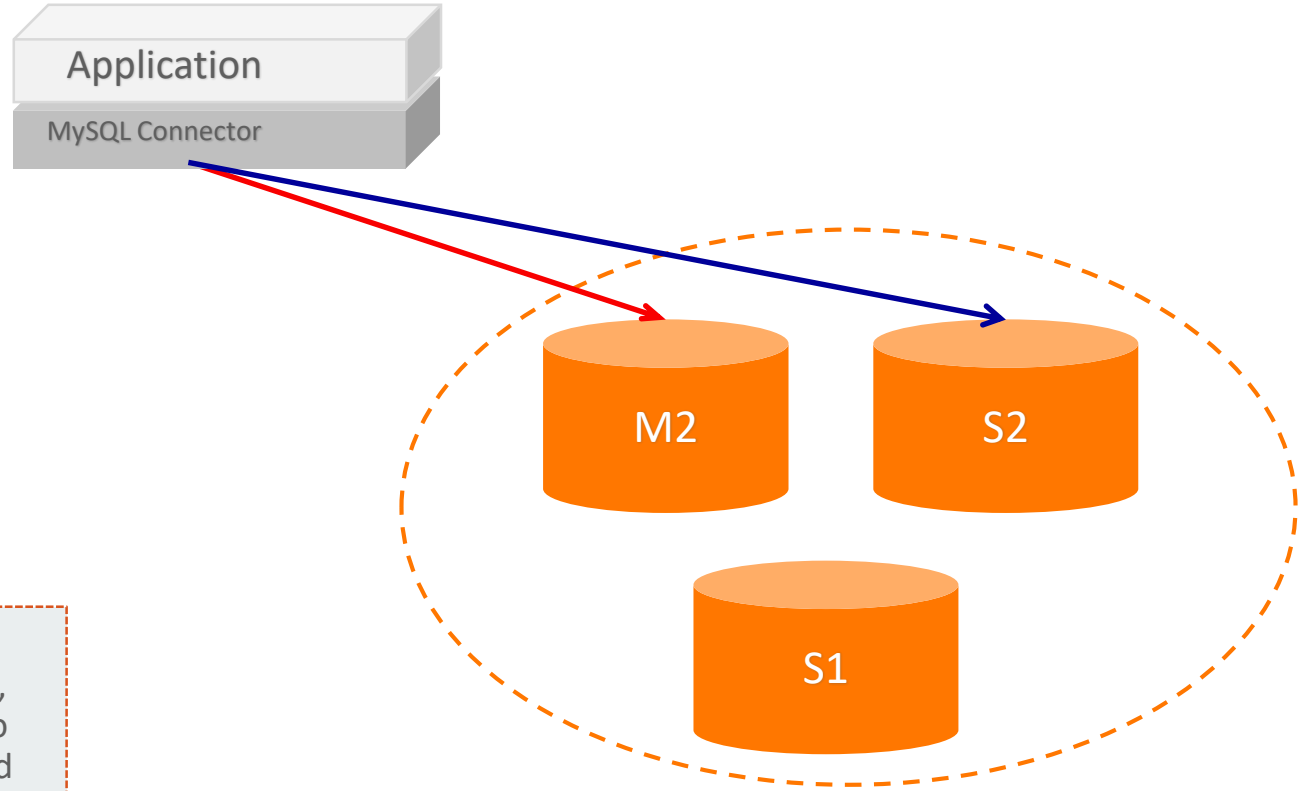
On by one you can start moving the slaves into the group. As all info applied into M1 is disseminated to the group, you can starting migrating reads to it and the migrated slaves.

How to migrate from asynchronous scenarios



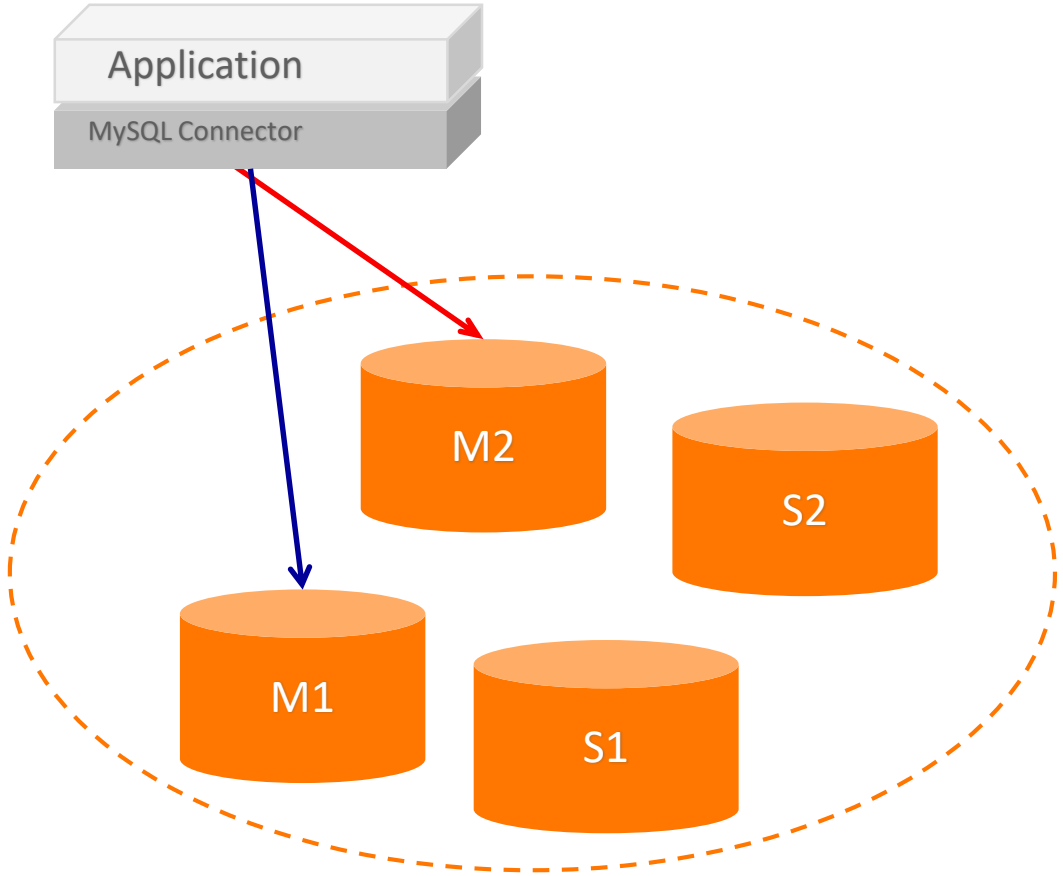
Eventually all slaves will be migrated to the group. Now comes the trick part.

How to migrate from asynchronous scenarios



If your workload allows it, start migrating some writes to the group gradually. If not, you need to wait for the delay from M1 to M2 to be nearly 0, block writes on M1 and then switch to M2 when that last N transactions from M1 are replicated.

How to migrate from asynchronous scenarios



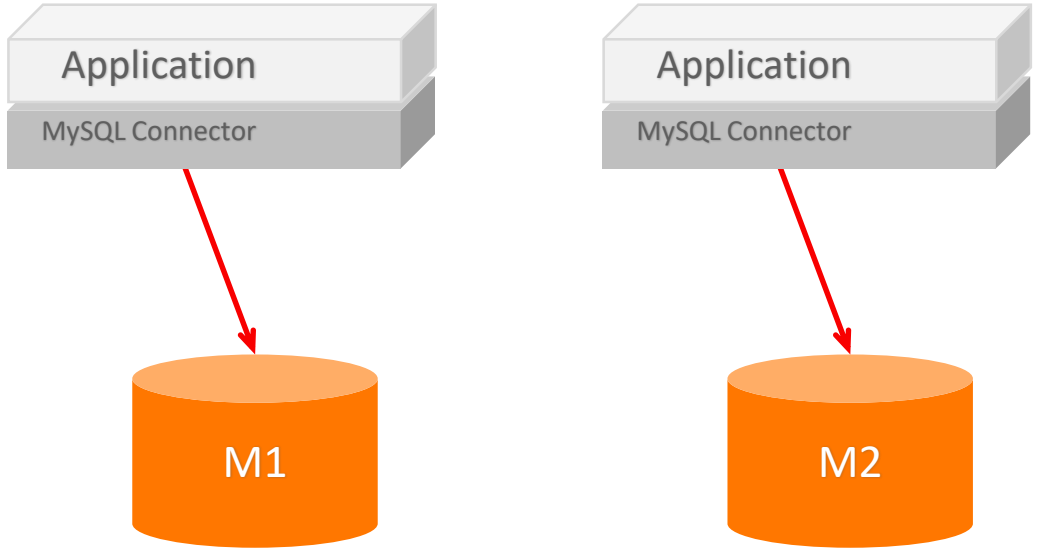
Now you just need to add M1 to the group.

Distributed recovery will make the member catch up to the writes it missed in the transition to the group.

3 Migrations

3.2 Different servers

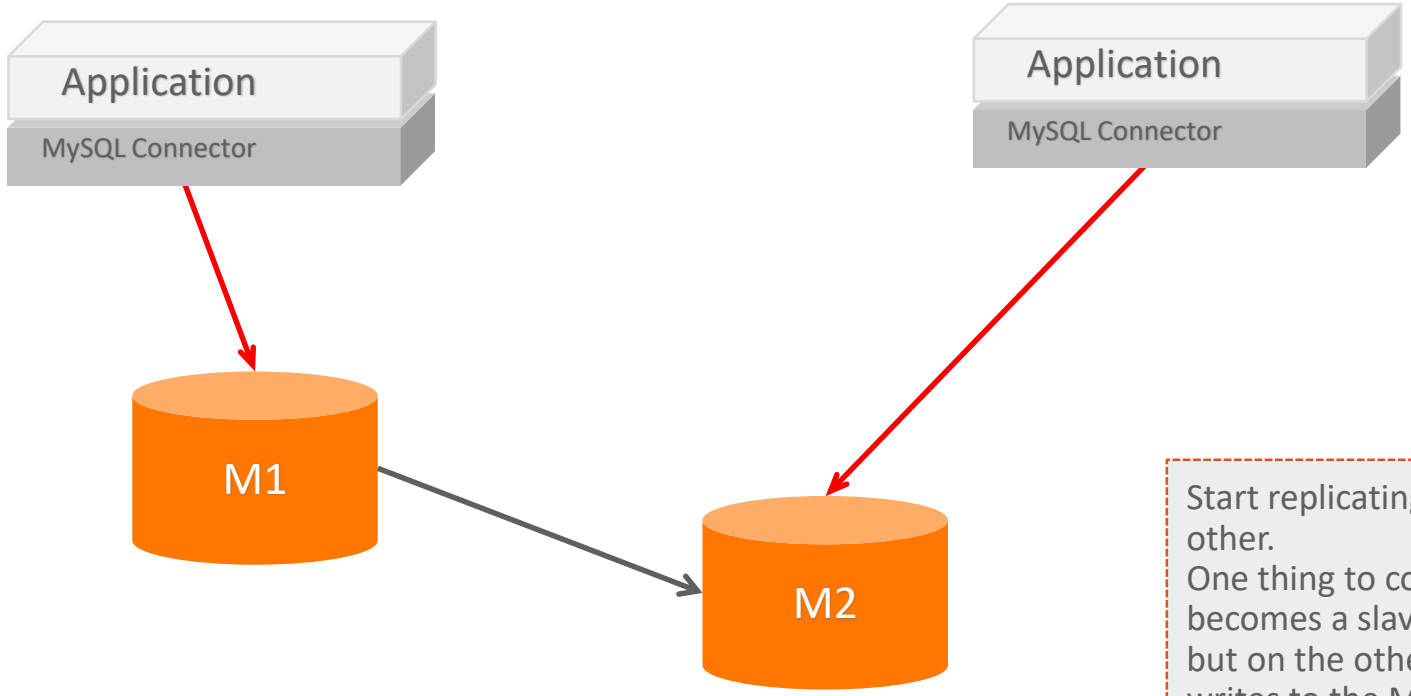
Migrate disjoint servers into Group replication



This one is a less common case, but still interesting as a migration example.

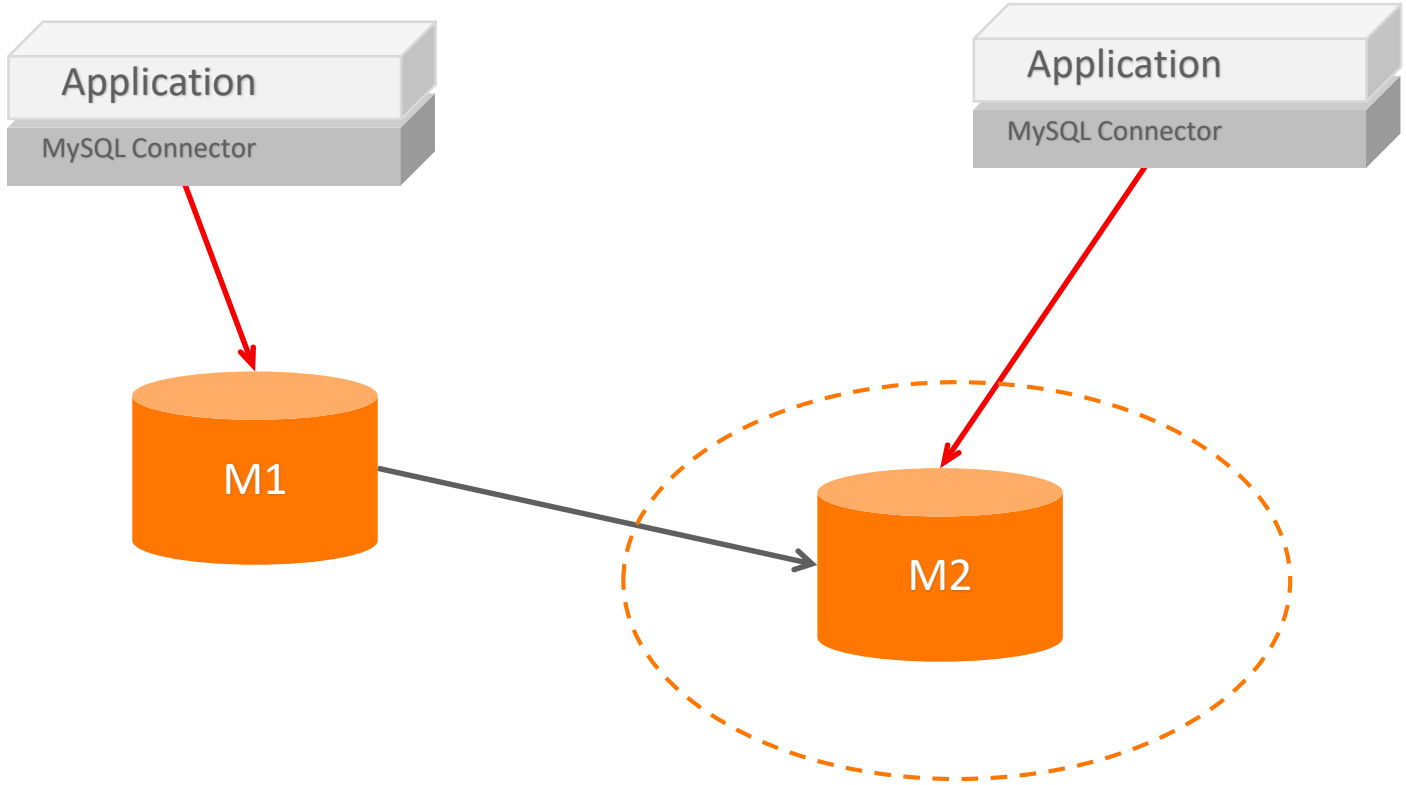
What if you want to merge shards or two applications into a group to make it HA?

Migrate disjoint servers into Group replication



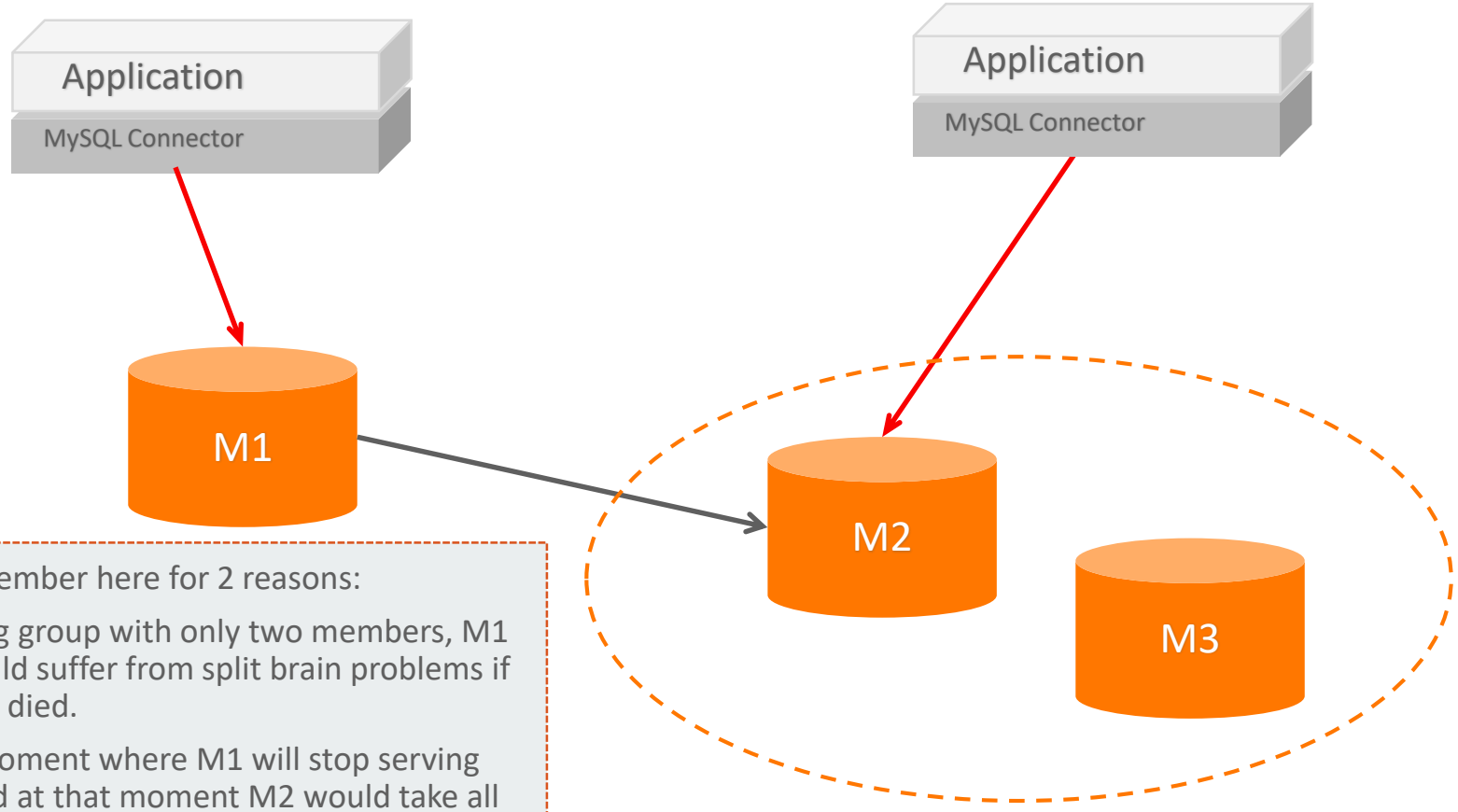
Start replicating from one of the servers into the other.
One thing to consider here: the member that becomes a slave has to deal with this increased load, but on the other hand, we must eventually block writes to the M1 for a brief moment.

Migrate disjoint servers into Group replication



Start a group with M2

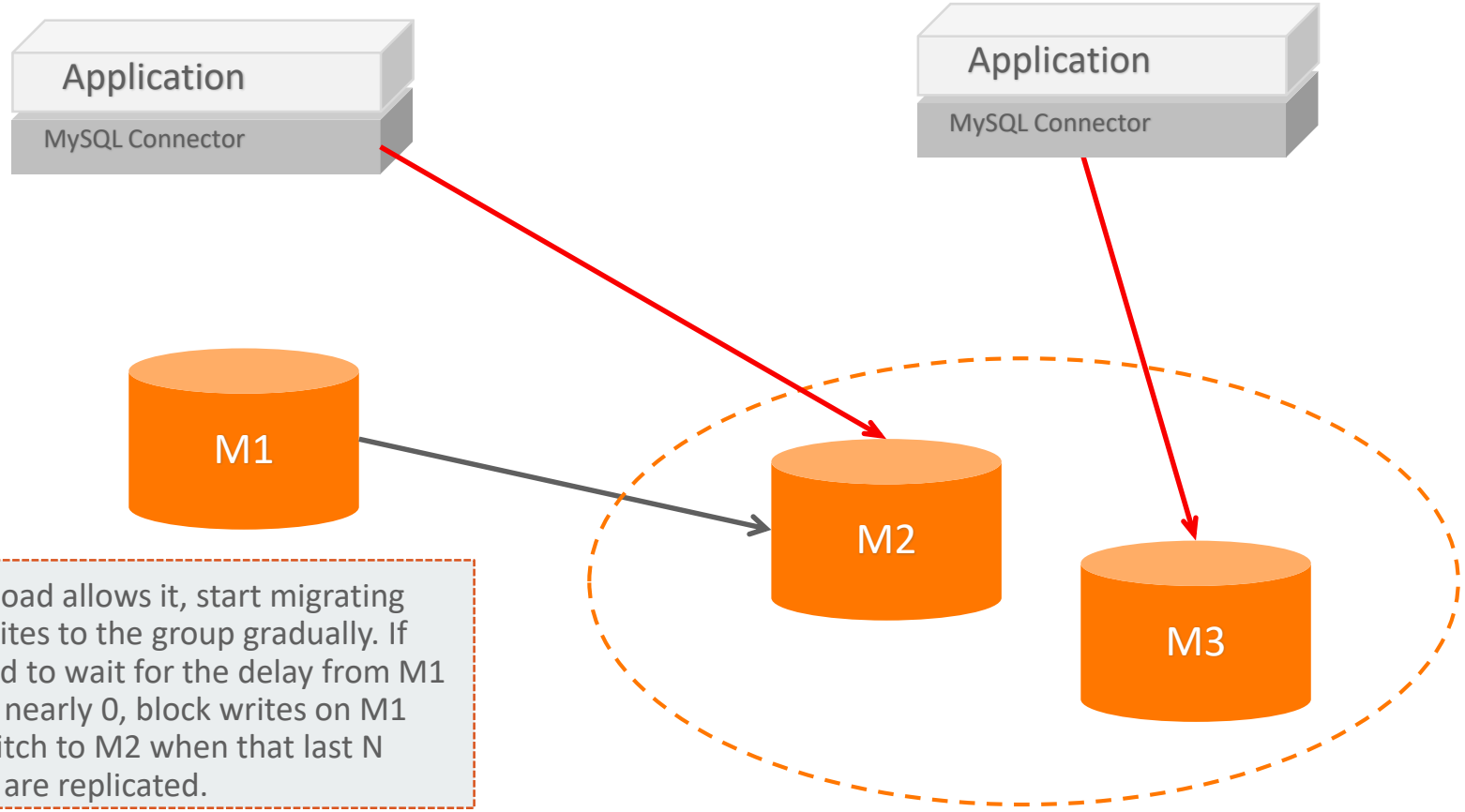
Migrate disjoint servers into Group replication



We add a new member here for 2 reasons:

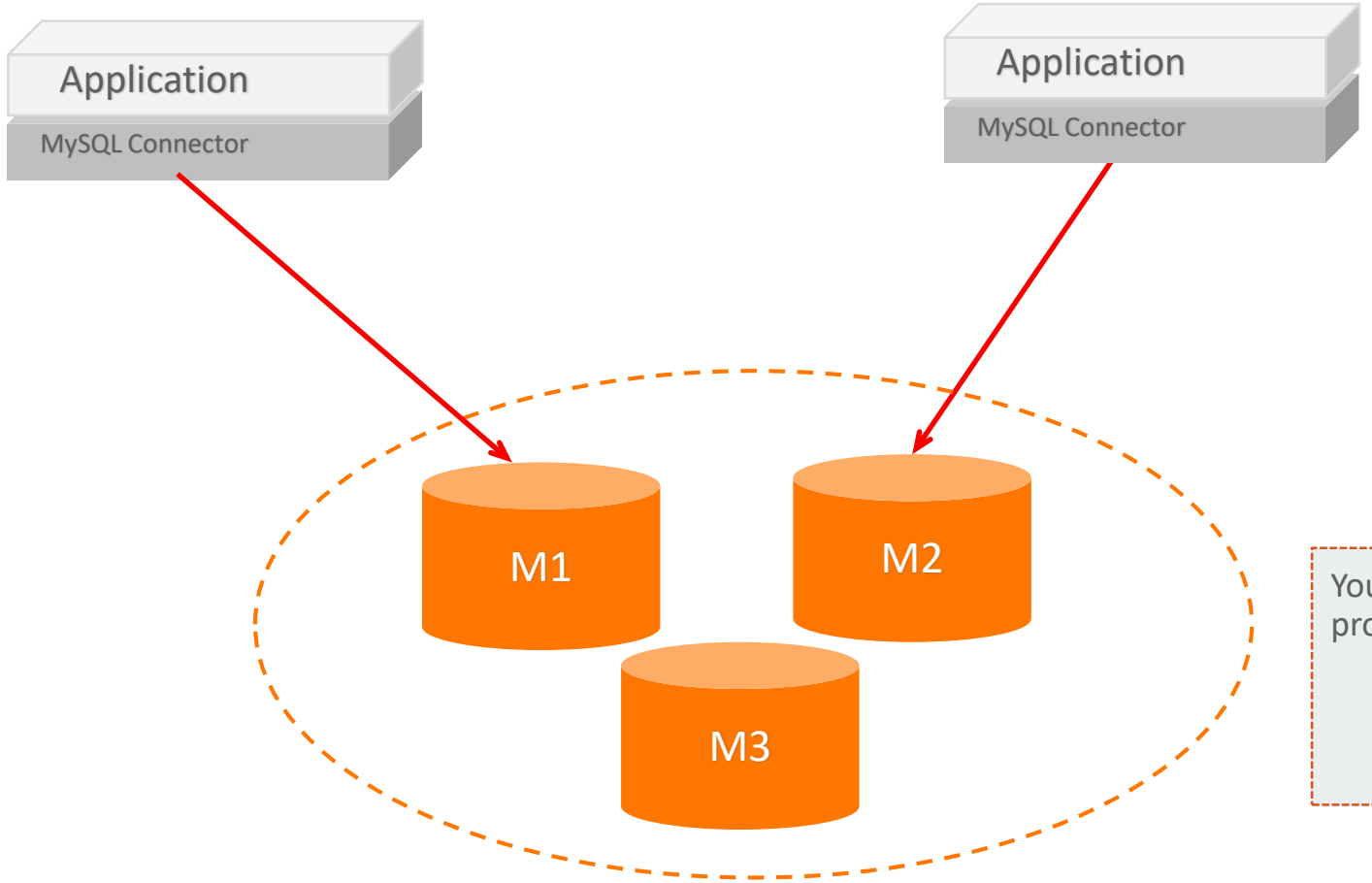
1. The resulting group with only two members, M1 and M2, could suffer from split brain problems if one of them died.
2. There is a moment where M1 will stop serving requests and at that moment M2 would take all the load alone.

Migrate disjoint servers into Group replication



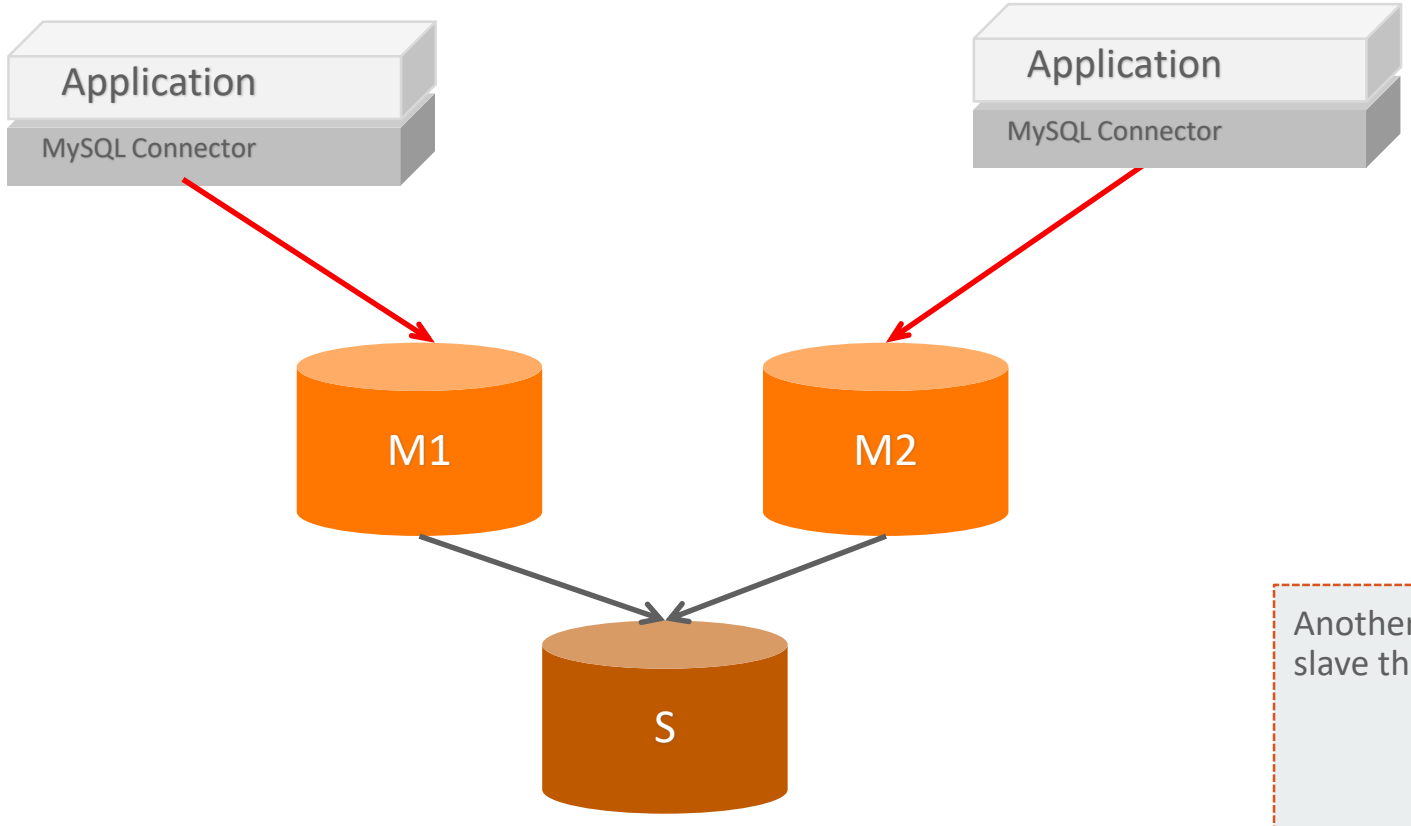
If your workload allows it, start migrating some M1 writes to the group gradually. If not, you need to wait for the delay from M1 to M2 to be nearly 0, block writes on M1 and then switch to M2 when that last N transactions are replicated.

Migrate disjoint servers into Group replication



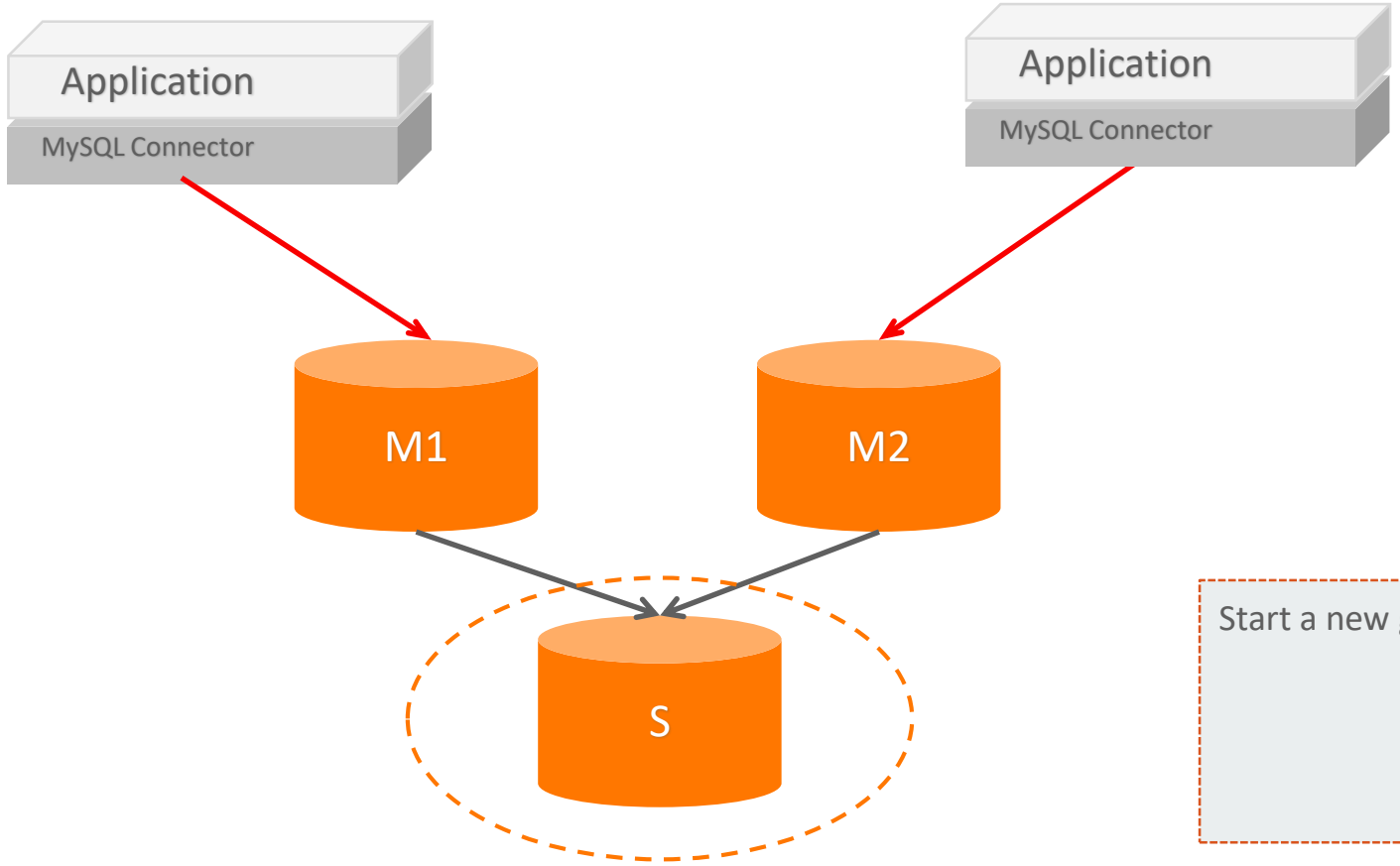
You can now join M1 to the group, being probably best to do some provisioning here

Migrate disjoint servers into Group replication (alternative)



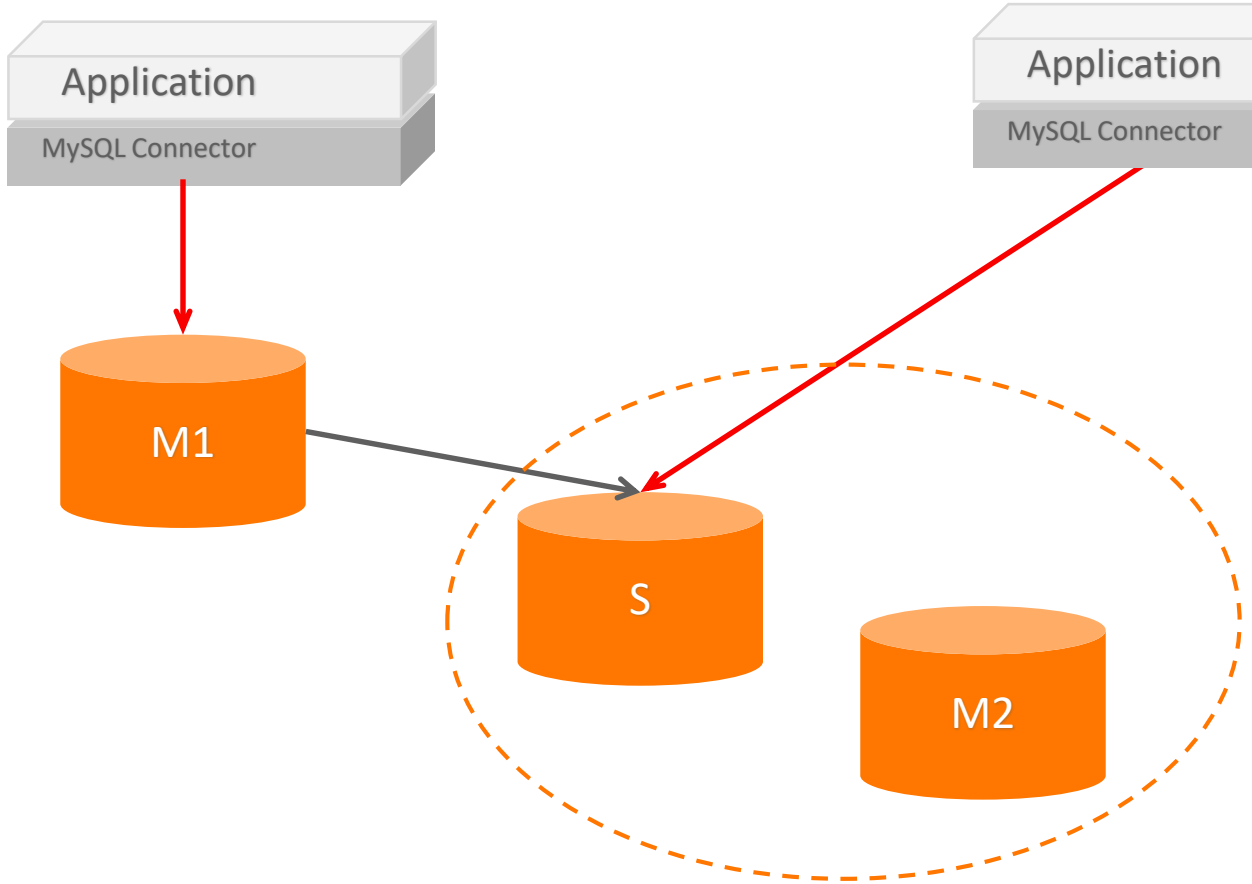
Another alternative is to add a multi source slave that will replicate from both members.

Migrate disjoint servers into Group replication (alternative)



Start a new group on this slave.

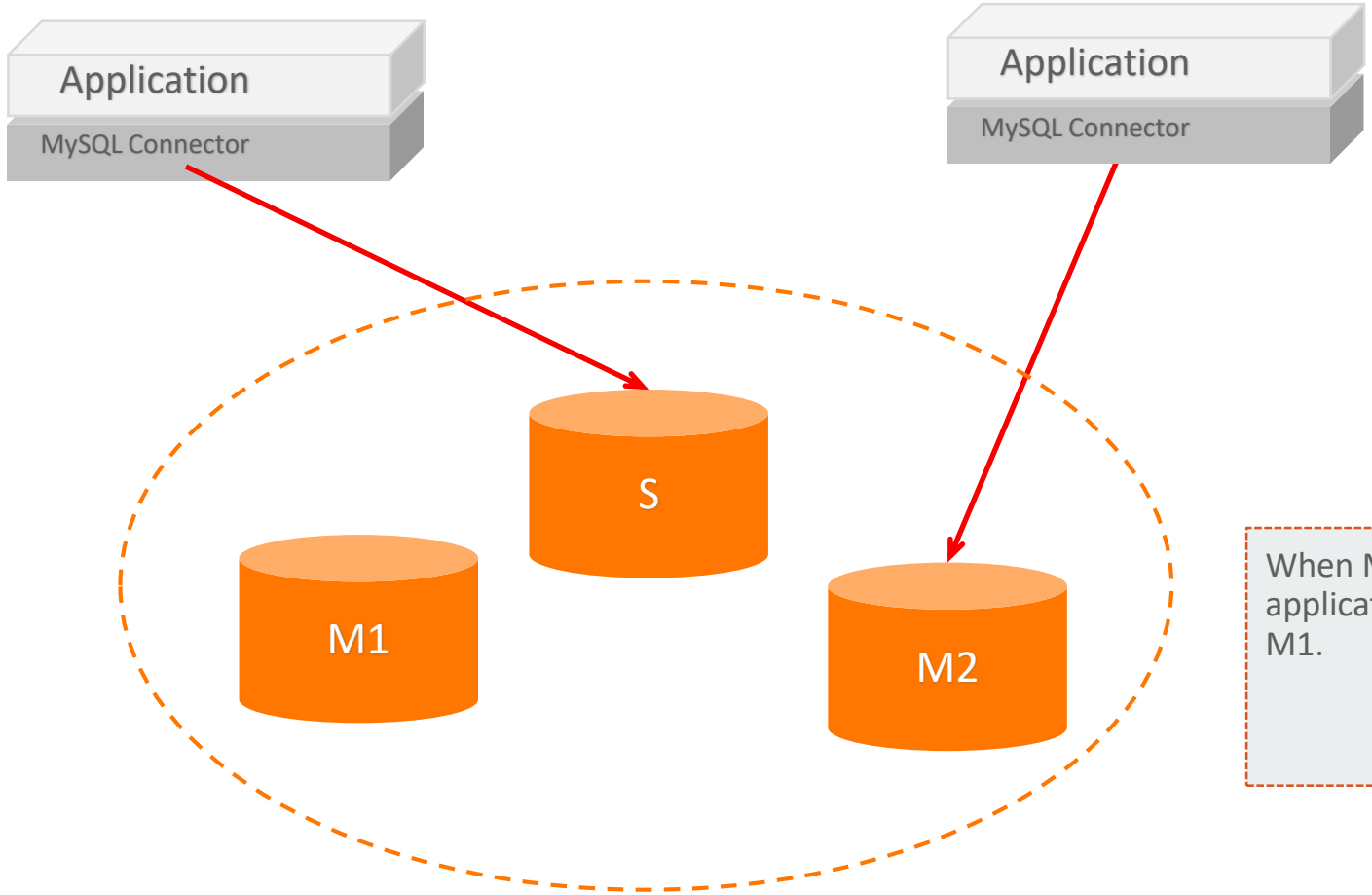
Migrate disjoint servers into Group replication (alternative)



If your workload allows it, start migrating some writes to S gradually. If not, you need to wait for the delay from M2 to S to be nearly 0, block writes on M2 and then switch to S.

Add M2 to the group.

Migrate disjoint servers into Group replication (alternative)

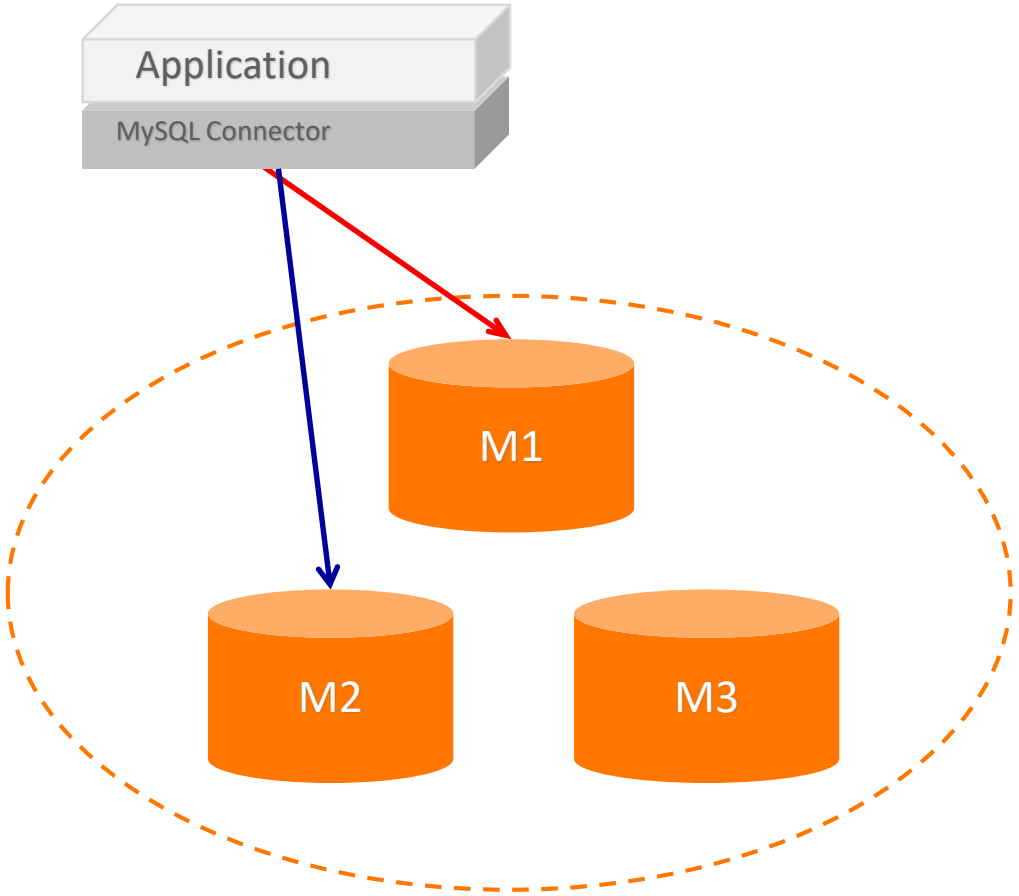


When M2 can take writes from its original application repeat the previous step with M1.

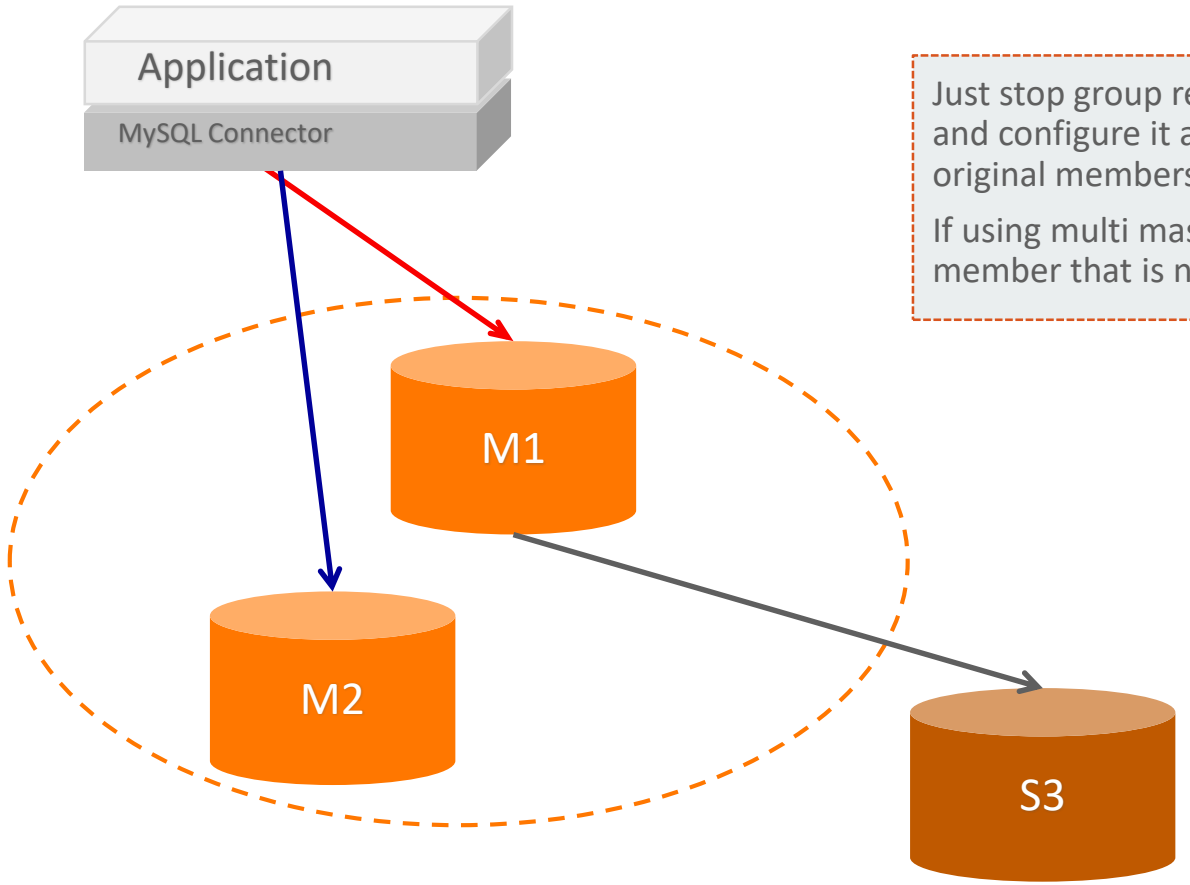
3 Migrations

3.3 Going Back

From Group Replication to Asynchronous topologies



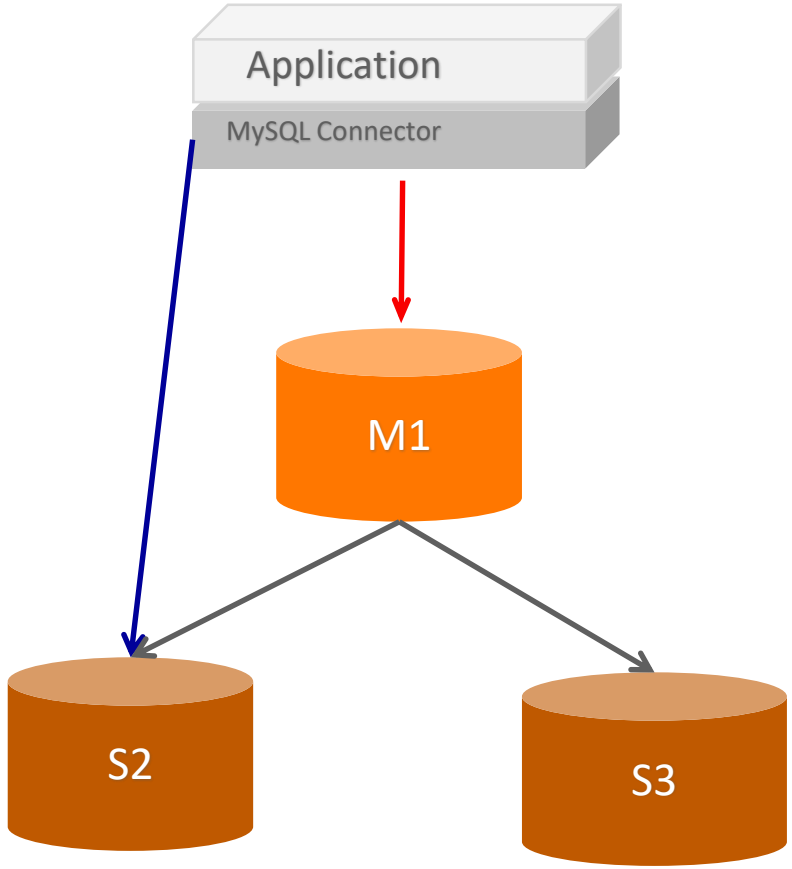
From Group Replication to Asynchronous topologies



Just stop group replication in one member and configure it as a slave of the one the original members.

If using multi master direct all writes to this member that is now a master.

From Group Replication to Asynchronous topologies



Repeat the same process for the other members until only the master remains in the group. At that point stop group replication on the master as well.

4 Conclusion

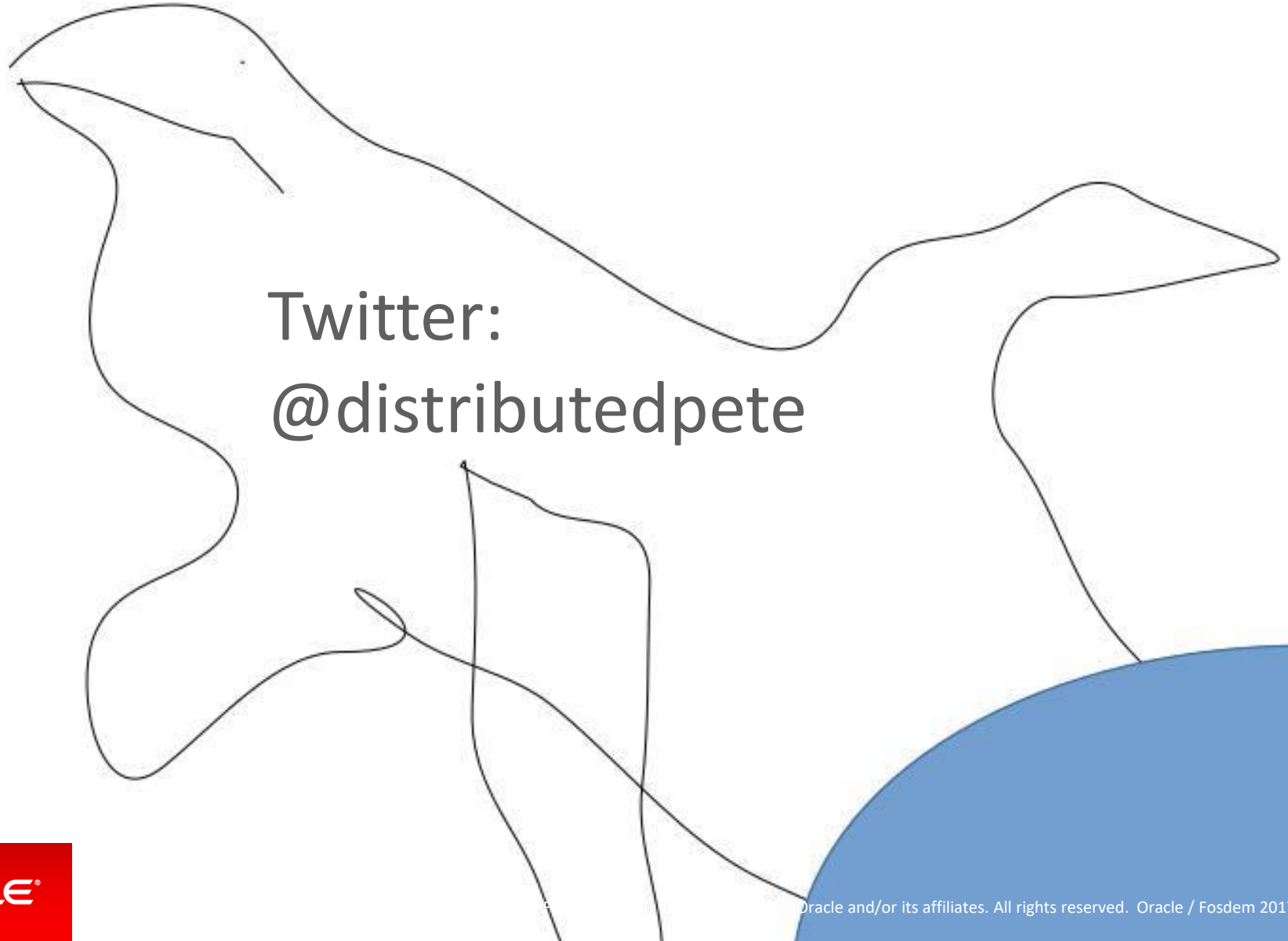
This presentation only contains examples of possible setups. Other possibilities may exist, just try it

Where to go from here?

- GA Packages
  - http://www.mysql.com/downloads/
- Documentation
  - http://dev.mysql.com/doc/refman/5.7/en/group-replication.html
- Blogs from the Engineers (news, technical information, and much more)
  - http://mysqlhighavailability.com


ORACLE®

Thanks!



Twitter:
@distributedpete