Cluster SQL Avec MariaDB Galera

Capitoul, 11/04/2019

Louis Chanouha

Université Fédérale Toulouse Midi-Pyrénées – Service Numérique - Pôle Infrastructures

CGP*

* conditions générales de présentation

Plan

- I - Présentation du produit
- II - Chronologie de la définition d’une architecture
- III - Optimisation
- IV - Bilan de 3,5 années d’exploitation

I – Présentation du produit (1) : MariaDB

- MariaDB est un fork (2010) d'Oracle MySQL, sous GPLv2
  - Fait suite à un changement de licence d'Oracle
  - 100% des fonctionnalités de MySQL 5.1
  - MariaDB et MySQL reste fortement liés (backports MySQL vers MariaDB)
  - Présent dans les distributions en remplacement de MySQL (Debian, Fedora, OpenSuse)

- Les deux commencent à diverger
  - Nouvelles fonctionnalités / optimisations exclusives à l'un ou l'autre

Oracle PL/SQL

- PL/SQL compatibility parser added for easier migration from Oracle to MariaDB
- sql_mode='oracle'
- Data types (have synonyms in MariaDB): VARCHAR2 (VARCHAR), NUMBER (DECIMAL), DATE (DATETIME), RAW (VARBINARY), BLOB (LONGBLOB), CLOB (LONGTEXT)
- CURRVAL, NEXTVAL
- EXECUTE IMMEDIATE
- Existing stored procedures, triggers
- ROW datatype for stored routines
- Cursors with parameters
- Packages
- https://mariadb.com/kb/en/library/sql_modeoracle-from-mariadb-103/

I – Présentation du produit (2) : Galera

Université Fédérale
Toulouse Midi-Pyrénées

- MariaDB Galera Cluster
  - Intégration de MariaDB et Galera (produit ~ indépendant) – présent aussi sur MySQL (pas testé)
  - Permet un cluster multi-master actif-actif : tous les nœuds acceptent des lectures <u>et</u> écritures
  - Réplication synchrone
  - Haute-disponibilité et répartition de charge
  - Ajout / Suppression dynamique des membres (téléchargement de l'incrément au démarrage)

- Limitations
  - InnoDB uniquement (pas de MyISAM)
  - LOCK TABLES
  - Tables avec clés primaires
  - AUTO-INCREMENT est modifié (incrément de max N=NBNOEUDS pour éviter conflits)
  - Recommande un quorum (3 nœuds) en cas de split brain

Plan

- I - Présentation du produit
- II - Chronologie de la définition d’une architecture
- III - Optimisation
- IV - Bilan de 3,5 années d’exploitation

Définition de l'architecture (1)

Université Fédérale
Toulouse Midi-Pyrénées

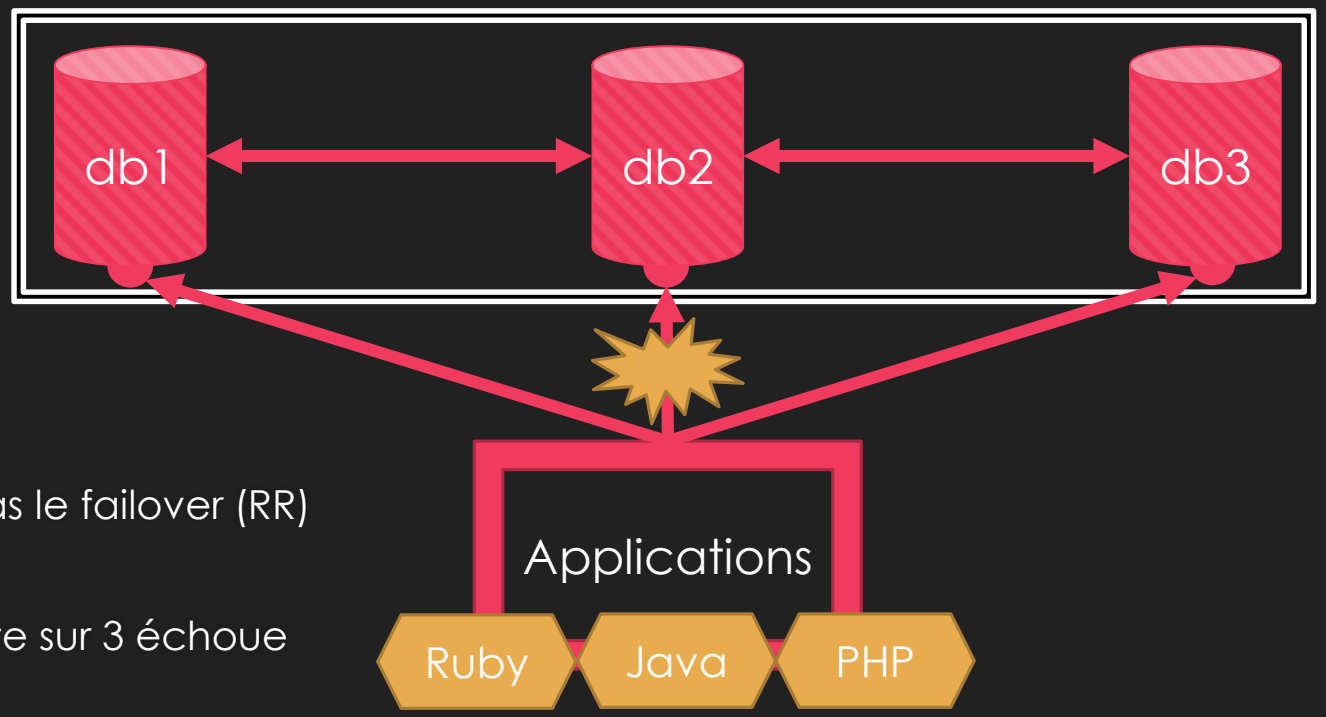
- Besoin technique
  - *« On veut faire du SQL comme avec MySQL »*
  - *« On veut que ce soit redondé »*
  - *« On a pas trop de temps »*

- *Exclusion de PostgreSQL*
  - *Compatibilité*
  - *Utilisation depuis le client (mutli-master R/W)*
  - *Expertise requise / Complexité*

Définition de l'architecture (2)

Université Fédérale
Toulouse Midi-Pyrénées

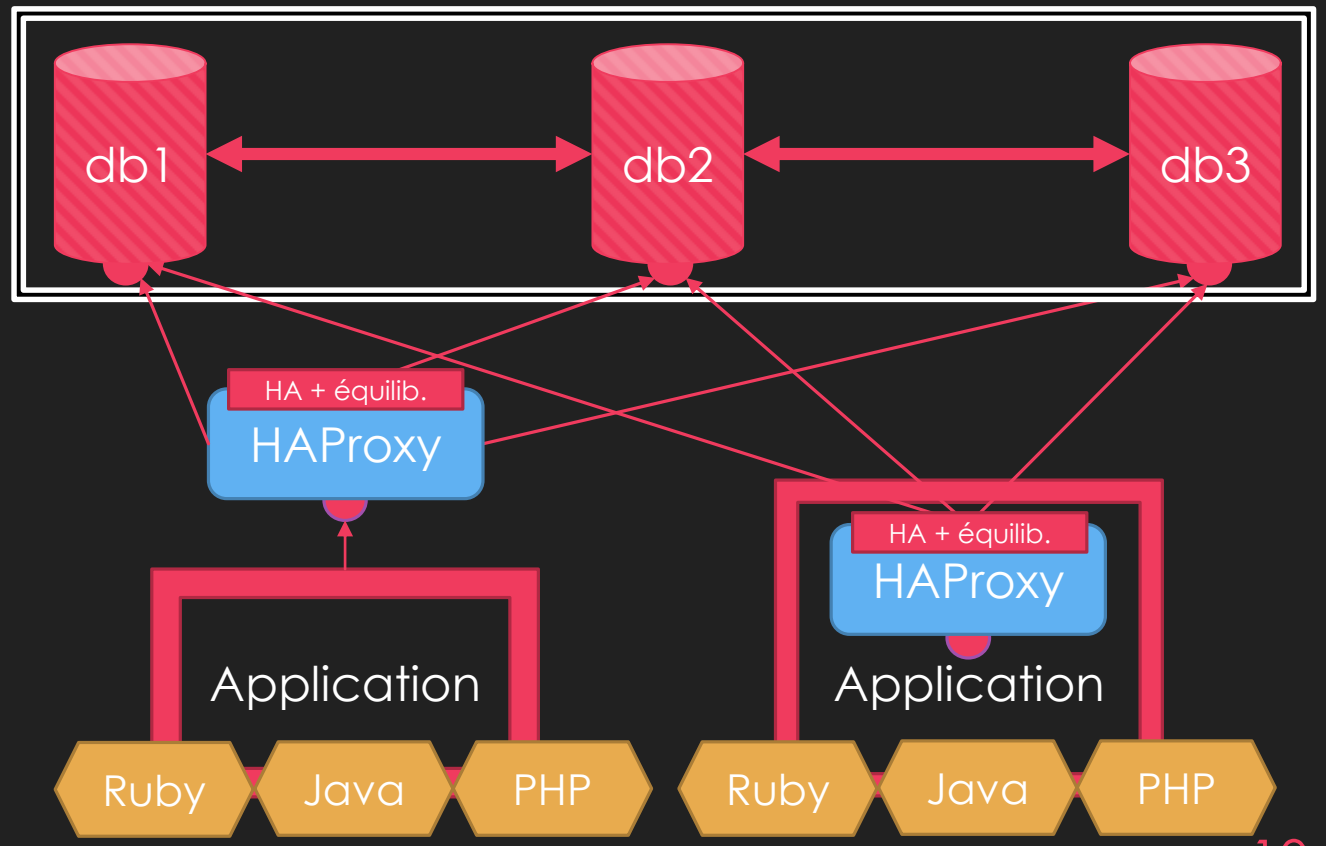
- Solution initiale
  - Mise en place d'un cluster Galera
    - avec 3 nœuds
  - Configuration des 3 nœuds sur l'applicatif
    - Round-Robin (RR) DNS
    - Liste d'hôtes



- Problèmes rencontrés
  - Les connecteurs d'applications ne gèrent pas le failover (RR)
    - Ou pas correctement (timeout applicatifs)
  - En cas d'interruption d'un nœud, une requête sur 3 échoue

Définition de l'architecture (3)

- Solution 2
  - Utilisation d'un reverse proxy L4
    - HAProxy sur machine spécifique
    - Gestion du statut des backend / pool
    - Checks L7
  - Problème rencontré: SPOF inutile / performances ?
- Solution 2bis
  - Intégration du reverse proxy sur la machine applicative
- Problèmes rencontrés
  - Exploitation: Aucun * (< 10 secondes de bascule)
  - Composant à installer sur chaque machine
    - Mais très léger (et on a Puppet)



Définition de l'architecture (4)

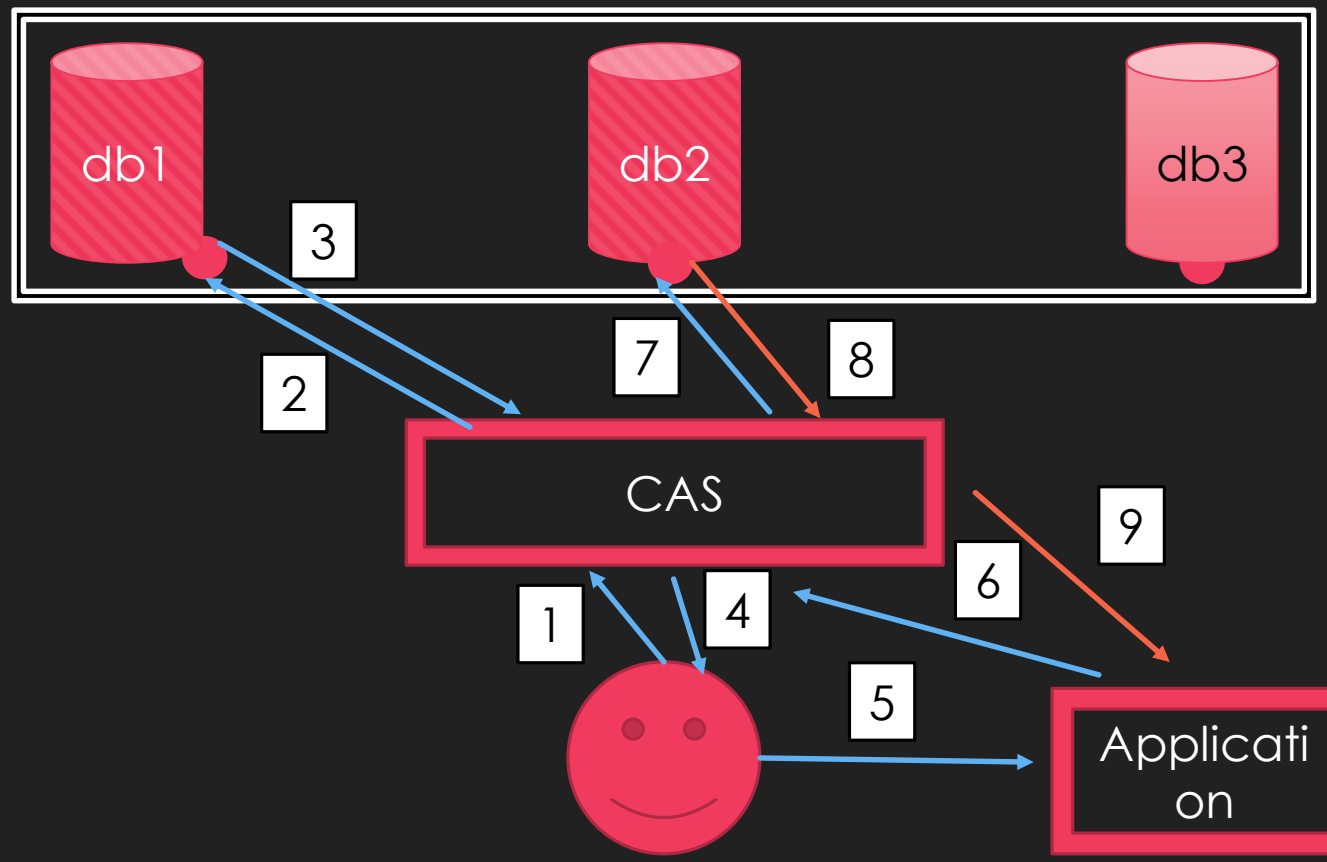
- Et puis un jour, sur le CAS…

"Ticket does not exist"

Définition de l'architecture (5)

- Que s'est-il passé ?

1. Login
2. Création du ticket de service
3. Retour de la requête SQL
4. Retour du ticket à l'utilisateur
5. Connexion utilisateur au service
6. serviceValidate
7. Validation du ticket
8. Retour de la requête SQL
9. Retour serviceValidate



Définition de l'architecture (6)

Université Fédérale
Toulouse Midi-Pyrénées

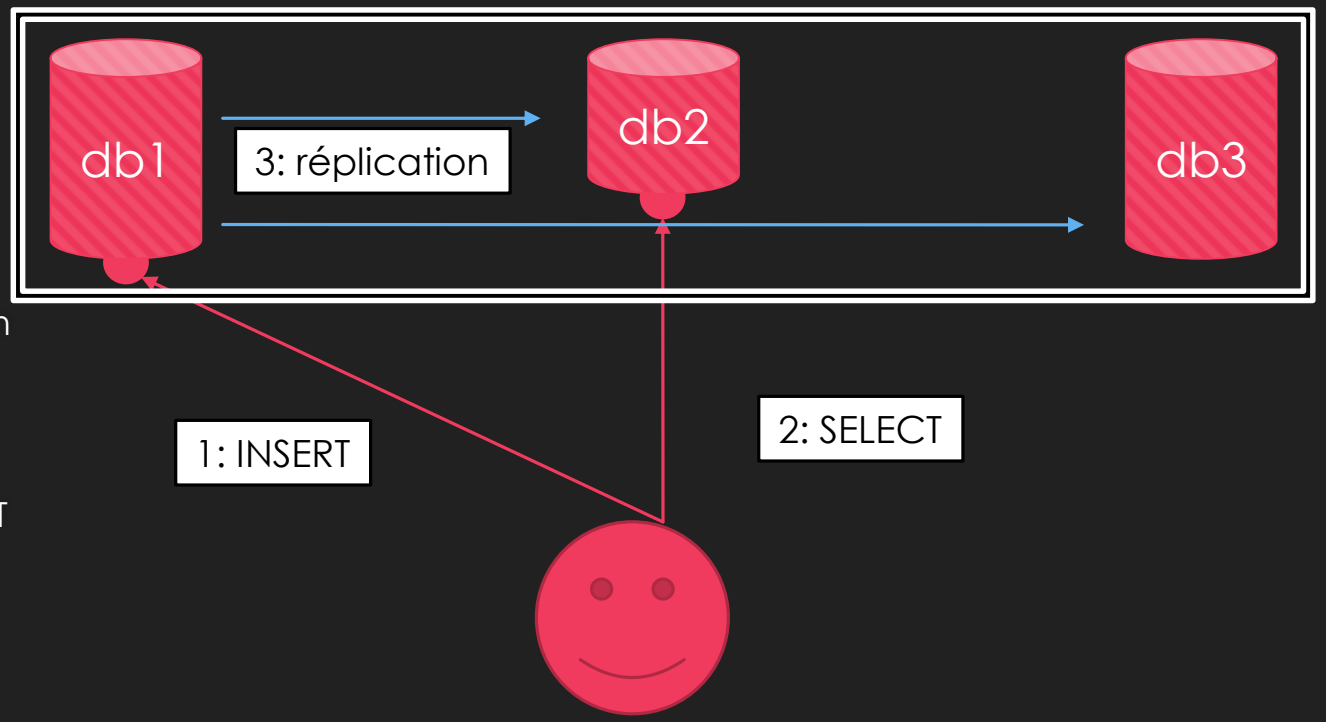
```php
<?php
// 3 membres dans le cluster

// Insertion d'une donnée sur le membre 1
    $db1 = new Mysqli("p:db1", "xxx", "uuu", "cas_dev");
    $rand = uniqid();
    $db1->query("INSERT INTO casino_service_tickets (ticket, service, ticket_granting_ticket_id, casino_service_rules_id)
                VALUES('$rand', 'https://scout.univ-toulouse.fr',1861, 61);");
    $db1->commit();

// Insertion d'une donnée sur le membre 2
    $db2 = new Mysqli("p:db2", "xxx", "uuu", "cas_dev");
    $result = $db2->query("SELECT COUNT(*) FROM casino_service_tickets WHERE ticket = '$rand';");
    print $result->fetch_row()[0];
```

CAS 1	CAS 2
$ php test.php 1	$ php test.php 0

DB2 en iowait

Définition de l'architecture (7)

- Que s'est-il passé ?
  - Mais ? On avait pas dit « synchrone » ?
  - Réplication virtuellement synchrone *
    - * sur le nœud qui a reçu le « write set »
    - * commit vaut acceptation de la transaction
  - * Réplication asynchrone
    - Dans des conditions normales, Tsynchro = RTT
    - Mais si un node est plus lent ?



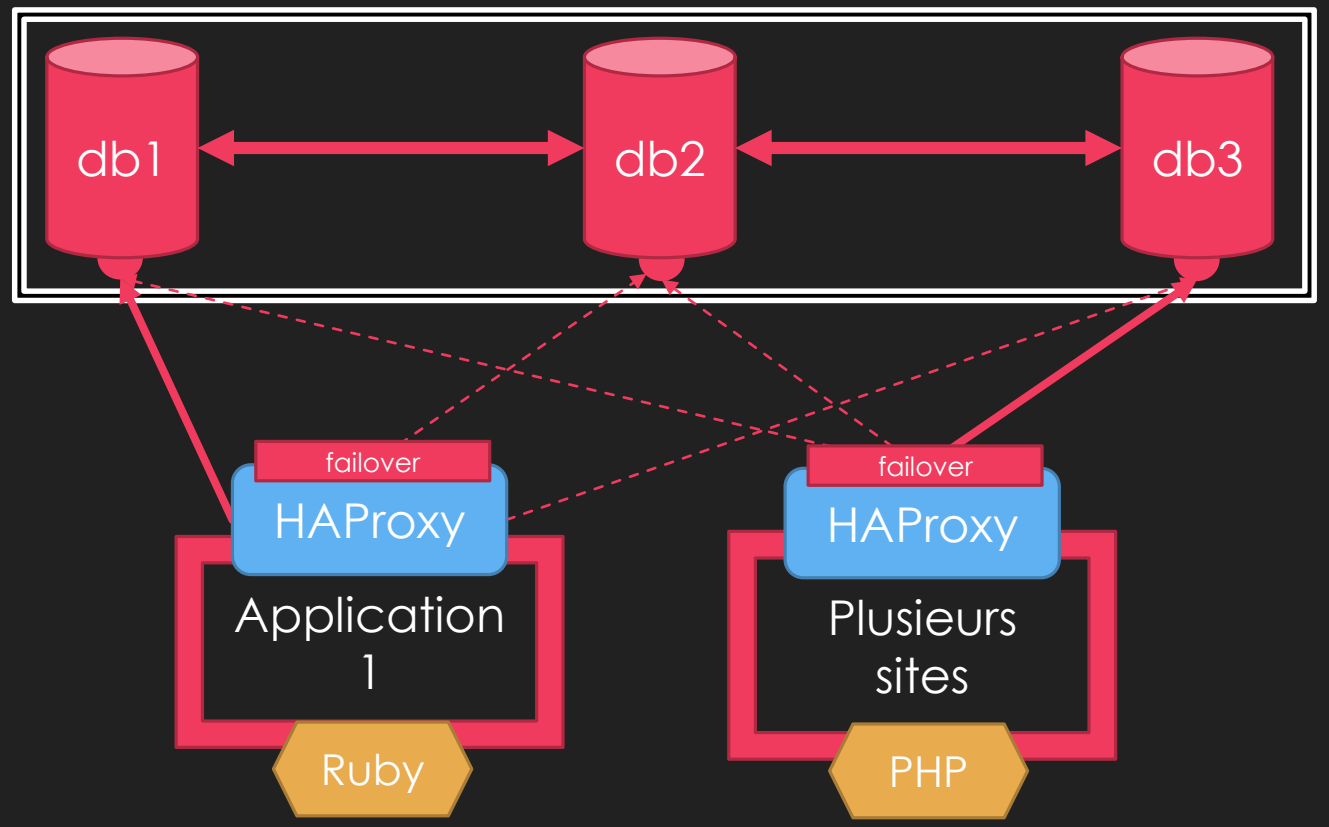
Définition de l'architecture (8)

- De ce constat, on a identifié 2 solutions :
  - Configurer le cluster pour qu'ils soit totalement synchrone:
    - wsrep_sync_wait=1 (attente de la synchro et blocage des requêtes lors d'un SELECT)
      - Non recommandé, a un impact fort sur les performances
  - Se positionner dans le contexte synchrone de Galera

Définition de l'architecture (9)

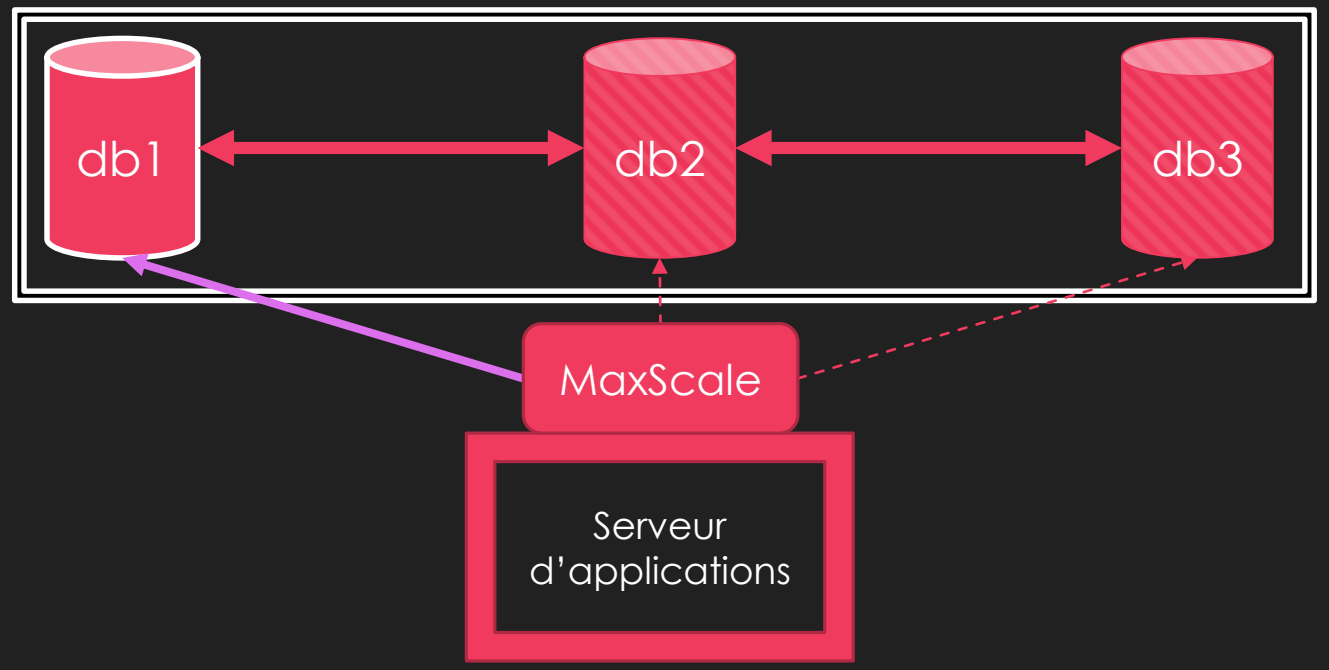
Université Fédérale
Toulouse Midi-Pyrénées

- Solution 3
  - Stick des applications sur un nœud
  - Toujours avec reverse-proxy (HAProxy)
- Avantage
  - Résoud le problème de synchronisation



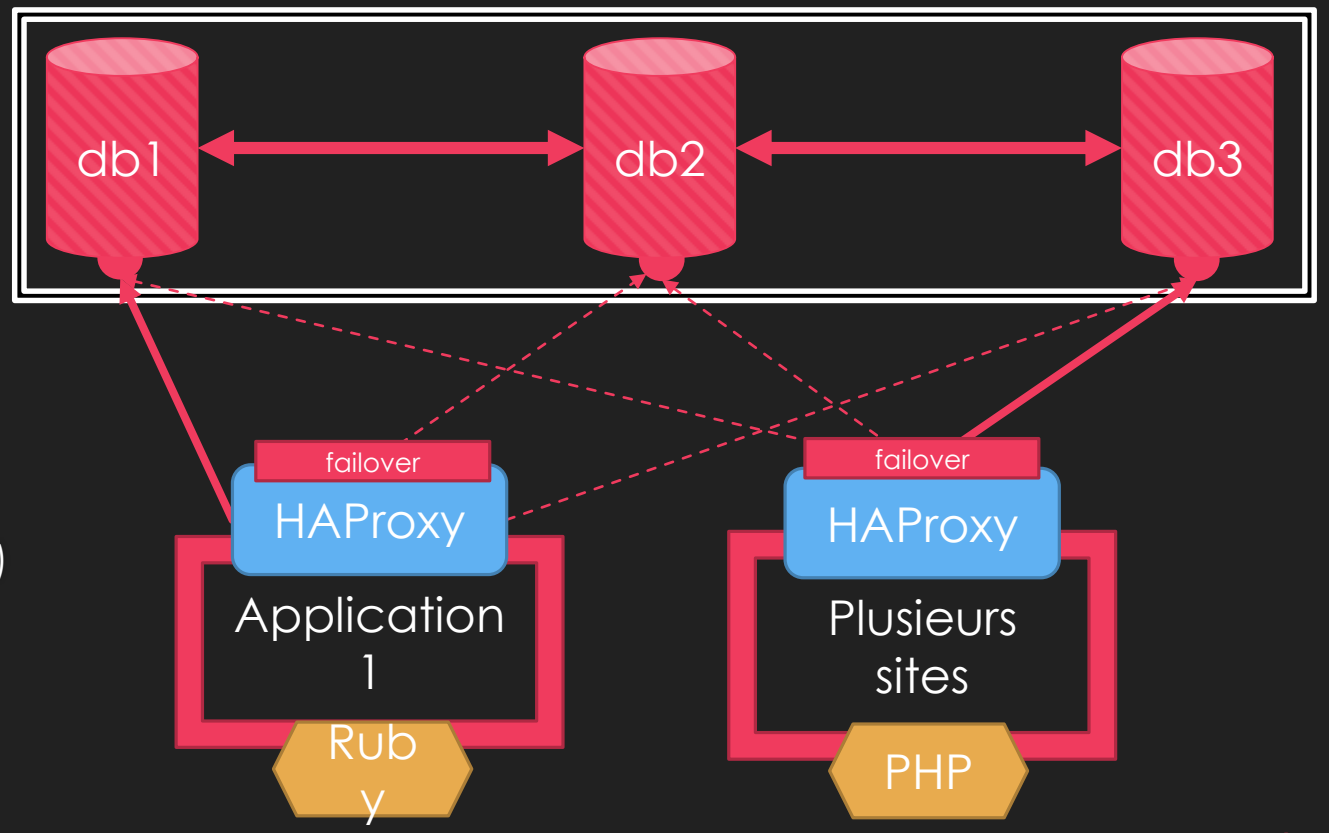
Définition de l'architecture (10)

- Solution 3bis
  - Stick des applications sur un nœud
  - Utilisation d'un reverse-proxy (MariaDB MaxScale)
- MariaDB MaxScale
  - R-proxy niveau 7 (SQL)
- Problèmes
  - Pas réussi à faire mieux qu'HAProxy
    - Répartition (serveur assigné par base)
    - FailOver (délai de bascule)
  - => Complexité inutile dans ce cas



Définition de l'architecture: sol. retenue

- Réponse au besoin technique
  - Compatibilité avec applications
    - OK (rien à faire)
  - Haute-disponibilité
    - OK
  - Équilibrage de charge
    - KO (répartition par application uniquement)



Plan

- I - Présentation du produit
- II - Chronologie de la définition d’une architecture
- III - Optimisation
- IV - Bilan de 3,5 années d’exploitation

Optimisations / Outils (1)

- De base (essentiel)
- Spécifiques à Galera

- De base
  - Innodb_db_buffer_pool_size, innodb_buffer_pool_instance, table_open_cache,join_cache_level, optimiser_use_condition_selectivity, innodb_log_file_size, join_buffer_size, max_heap_table_size, innodb_flush_log_at_trx_commit=0
  - Noyau Linux (tcp, scheduler noop, max_open_files, swapinness, iptables)

Optimisations / outils (2)

- Spécifiques à MariaDB / Galera
  - Mode de restauration des esclaves (wsrep_sst_method) => indispensable
    - Rsync (freeze du cluster pendant la copie) => mariabackup indispensable
  - wsrep_retry_autocommit=4
  - Mod-Proxy (10.3, proxy-protocol-networks)
    - Équivalent du X-Forwarded-For en HTTP
  - Sondes Nagios efficaces (*wsrep_flow_control_paused*)
  - ClusterControl ?

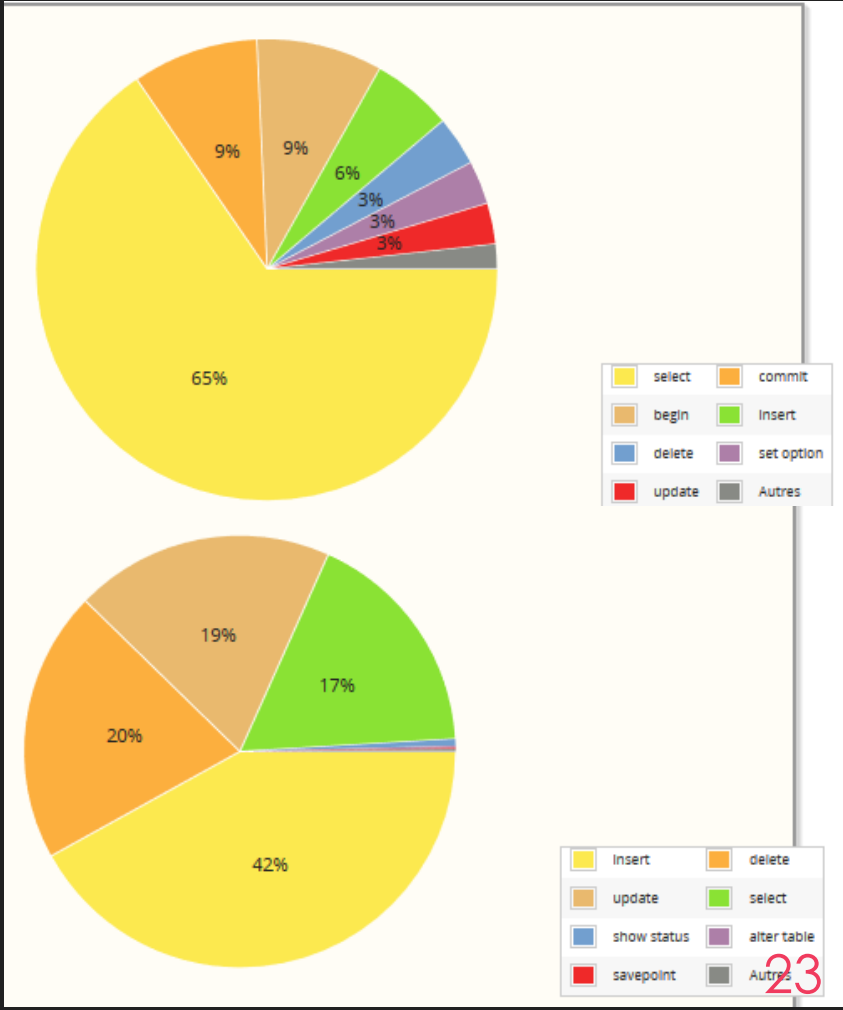
Plan

- I - Présentation du produit
- II - Chronologie de la définition d’une architecture
- III - Optimisation
- IV - Bilan de 3,5 années d’exploitation

BILAN d'EXPLOITATION (1)

Université Fédérale
Toulouse Midi-Pyrénées

- Chiffres, depuis 3 ans
  - 60 applications (prod et test)
    - 40 web (Drupal, Symfony majoritairement)
    - CAS
    - Shibboleth
  - 600 k Requêtes / heure
  - 17go de base de données / 8go de RAM par serveur virtuel
  - Une dizaine d'incidents notables, la moitié à résolution complexe
    - Problèmes architecturaux mais surtout côté MariaDB



BILAN d'EXPLOITATION (2)

Université Fédérale
Toulouse Midi-Pyrénées

- Sur 3 critères:
  - Stabilité du service
  - Performances
  - MCO et mises à jour

- Stabilité du service relative
  - Pas d'auto-exclusion d'un node défectueux
    - Cas d'un node avec disque plein (/data ou /log) qui bloque tout le cluster
    - Cas d'une transaction bloquée (lock wait) qui bloque tout le cluster
  - Disaster recovery (perte du cluster suite à une coupure électrique.. ou d'un cron :|)
    - Point fort: jamais de perte de données
    - Pas de redémarrage automatique
    - État incohérent du cluster: impossibilité aux nœuds de rejoindre l'applicatif
      - Plusieurs situations distinctes qui ont imposé d'investiguer en profondeur et intervenir sur le cluster de production
      - Souvent nécessité d'une resynchronisation complète des données (incrémental ne fonctionne pas)

BILAN d'EXPLOITATION (3)

- Performances
  - Nivelé par le bas en écriture (plus mauvais nœud), car les opérations sont répliquées sur tous les nœuds qui connaissent toutes les bases
  - Pas de priorisation
    - Un import/export (grosse transaction) d'une grosse base web peut « freezer » un autre service (iowait)
  - Toutes les bases sont sur toutes les nœuds. Impossibilité de passer à une échelle dépassant la capacité d'écriture d'un nœud
- Qualité des évolutions
  - 10.2.12: freeze du cluster complet
  - 10.3: conflits de transaction (remplit les binlogs)
  - Des régressions…
    - Impose un protocole sérieux de test (quand on peut)

BILAN d'EXPLOITATION (4)

- Bilan d'exploitation
  - Ça fonctionne !
    - Centralisation bénéfique en terme de coût d'exploitation (gestion des base, droits, sauvegarde, phpMyAdmin)
    - Des contraintes sur les données (InnoDB, Lock Tables, clé primaires) raisonnables et acceptables
      - Pas besoin d'adapter l'applicatif (en sticky node)
      - L'architecture est simple (sur Debian, un paquet maintenu par l'éditeur)
    - Incidents avec perte de production souvent liés à des mises à jour ou opérations (renforcer les procédures de m-à-j ?)
  - Mais
    - Plus gros facteur d'interruption du CAS (est-ce anormal pour un SGBDR) ?
    - Prudence avec les m-à-j : soyez prêt à rollbacker (race conditions difficiles à reproduire sur du test)
    - Attention au suivi des indicateurs machines : Galera ne le fera pas pour vous
      - Espace disque
      - Nombre max de connexions simultanées
    - Quelques services non éligibles (base de donnée spécifique)
      - PostgreSQL ou dans les restrictions de Galera
    - On améliore pas le taux de disponibilité

BILAN

Pas idéal mais

Le coût de maintenance / niveau de service est acceptable

Peut-on faire mieux avec les mêmes ressources ?

Peut-on faire mieux avec de telles contraintes applicatives ?

Fin

Questions ?

Installation de MariaDB Galera

Université Fédérale
Toulouse Midi-Pyrénées

```
apt::key { 'F1656F24C74CD1D8': server  => 'keyserver.ubuntu.com', id => '0x177F4010FE56CA3336300305F1656F24C74CD1D8
# Ajout de la source dans le fichier /etc/apt/sources.list
apt::source { 'mariadb':
            location => 'http://mariadb.mirrors.ovh.net/MariaDB/repo/10.3/debian',
            release  => $::lsbdistcodename,
            repos    => 'main',
            notify   => Exec['apt_update'],
}
# ------------ Fin gestionnaire de paquet (apt)
# Une fois le packge téléchargé, on fait l'installation
package { "mariadb-server": ensure => "held" }
service{'mysql':}
file {"/etc/mysql/conf.d/cluster.cnf":
            notify   => Service['mysql'],
            content  => template("ut_db/etc/mysql/conf.d/cluster.cnf")
}
ensure_packages(['mariadb-backup'])
file_line { 'limits pour mysql':
            path => '/etc/security/limits.conf',
            line => 'root - nofile 16000',
}
# Correction systemd
file {"/etc/systemd/system/mariadb.service.d/":
            ensure => directory,
}
file {"/etc/systemd/system/mariadb.service.d/override.conf":
            content => "[Service]\nTimeoutStartSec = 600\n",
            notify  => [Exec['systemd-reload'], Service['mysql']],
}
```

```
# Ce fichier est géré par Puppet
# Toute modification manuelle sera écrasée
[mysqld]
query_cache_size=0
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
query_cache_type=0
bind-address=0.0.0.0
#innodb_large_prefix=1
max_connect_errors=10000
innodb_buffer_pool_size=5G
innodb_log_file_size=1G
innodb_stats_on_metadata=0
enforce_storage_engine=InnoDB

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_provider_options="gcache.size=500M"

# Galera Cluster Configuration
wsrep_cluster_name="comue_db"
wsrep_cluster_address="gcomm://10.[redacted]0,10.[redacted]1,10[redacted]2"

# Log des transactions
wsrep_log_conflicts=ON
wsrep_provider_options="cert.log_conflicts=YES;gcs.fc_limit=200"
```