Base de Données Avancées - Livrable 3

Phase 3 : Distribution et Replica Set

Exertier Hugo

Janvier 2026

Table des matières

1 Introduction

L'objectif de cette phase est de migrer d'une instance MongoDB unique vers une architecture distribuée (Replica Set) pour assurer la haute disponibilité des données. Nous avons également connecté l'application Django à ce cluster.

2 T3.1 : Configuration du Replica Set

2.1 Architecture

Le cluster `rs0` est composé de 3 nœuds hébergés en local sur des ports distincts :
— **Nœud 1 (Primary initial) :** Port 27017 (`data/mongo/db-1`)
— **Nœud 2 (Secondary) :** Port 27018 (`data/mongo/db-2`)
— **Nœud 3 (Secondary) :** Port 27019 (`data/mongo/db-3`)

2.2 Initialisation

Les instances ont été démarrées avec l'option `-replSet rs0`. La configuration a été appliquée via la commande suivante :

```
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "localhost:27017" },
    { _id: 1, host: "localhost:27018" },
    { _id: 2, host: "localhost:27019" }
  ]
})
```

3 T3.2 : Tests de Tolérance aux Pannes

Nous avons suivi le protocole de test en 7 étapes défini dans le sujet. Un script Python (`test_failover.py`) insère un document par seconde pour monitorer la disponibilité.

3.1 État nominal

1. État initial : La commande `rs.status()` confirme que le cluster est sain avec 1 Primary et 2 Secondaries. **2. Écriture :** Le script insère les documents séquentiellement. Le Primary accepte les écritures et les réplique.

3.2 Cycle de Panne Simple (Failover)

3. Panne Primary : Nous avons arrêté le processus du nœud Primary (27017) brutalement (SIGTERM).

4. Nouveau Primary (Élection) : Les nœuds restants ont détecté l'absence de heartbeat. Une élection a eu lieu immédiatement. Le nœud 27019 a été élu nouveau Primary. *Mesure :* Le temps de basculement a été inférieur à 1 seconde. Le client n'a subi aucun timeout visible.

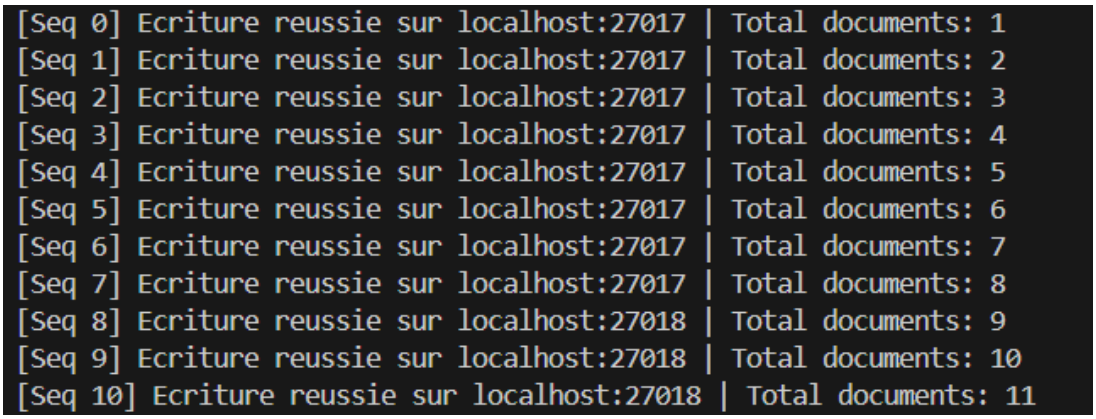

FIGURE 1 – Basculement transparent : Le client passe du port 27019 au 27017 sans interruption.

5. Lecture : Les données sont restées accessibles en lecture/écriture sur le nouveau Primary.
6. Reconnexion : Le nœud 27017 a été relancé. Il a rejoint le cluster en tant que `SECONDARY` et s'est synchronisé.

3.3 Scénario Catastrophique (Double Panne)

7. Double panne : Alors qu'un nœud était déjà éteint, nous en avons arrêté un deuxième. Il ne restait qu'un seul nœud actif sur les 3.

Observation Client : Le script de test n'arrive plus à écrire. Il retourne des erreurs de connexion (Timeout) car aucun serveur ne peut accepter les écritures (pas de Primary).

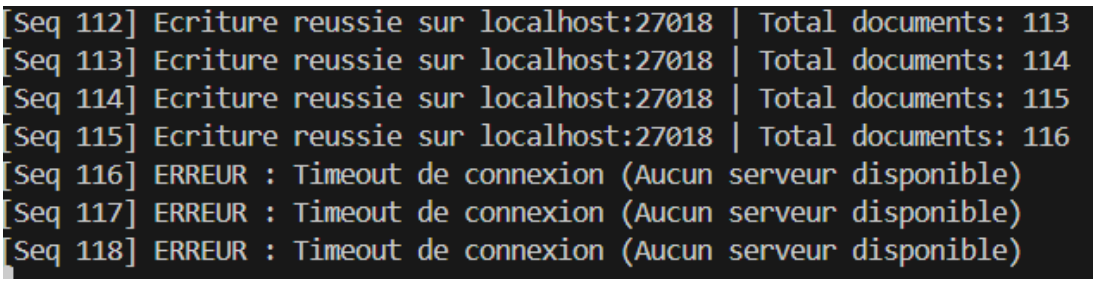

FIGURE 2 – Erreurs client (Timeout) lors de la perte de majorité

Observation Serveur : Le dernier nœud passe en état `SECONDARY`. Il refuse de devenir Primary. Les logs confirment la raison : *"Not standing for election because I cannot see a majority"*.

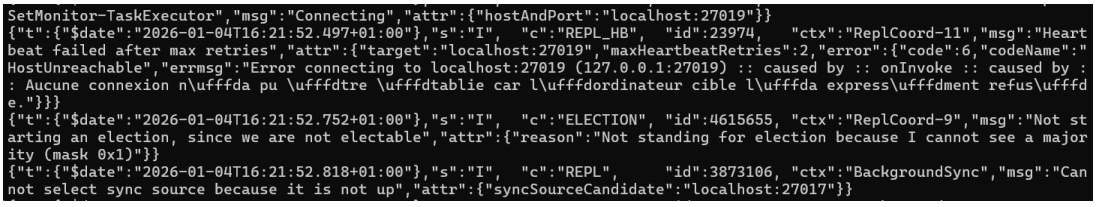

FIGURE 3 – Log serveur : Refus d'élection par manque de quorum (1/3)

Conclusion : Ce test valide le principe de **Consistance** (CP). MongoDB exige une majorité stricte ($N/2+1$) pour élire un leader. Avec 1 nœud sur 3, le quorum n'est pas atteint, le système se verrouille pour éviter le "Split-Brain".

4 T3.3 : Intégration Django

L'application Django a été configurée pour utiliser le Replica Set via `pymongo`.

4.1 Configuration

Dans `settings.py`, l'URI de connexion déclare les trois hôtes :

```
MONGO_URI = "mongodb://localhost:27017,localhost:27018,localhost:27019/?
    replicaSet=rs0"
```

4.2 Validation de la connexion

Une vue de test a été implémentée pour vérifier la communication. Django parvient à se connecter au cluster, identifier le Primary et lire la collection `movies_complete` (36 859 documents). On peut y accéder au chemin suivant : http://127.0.0.1:8000/test/

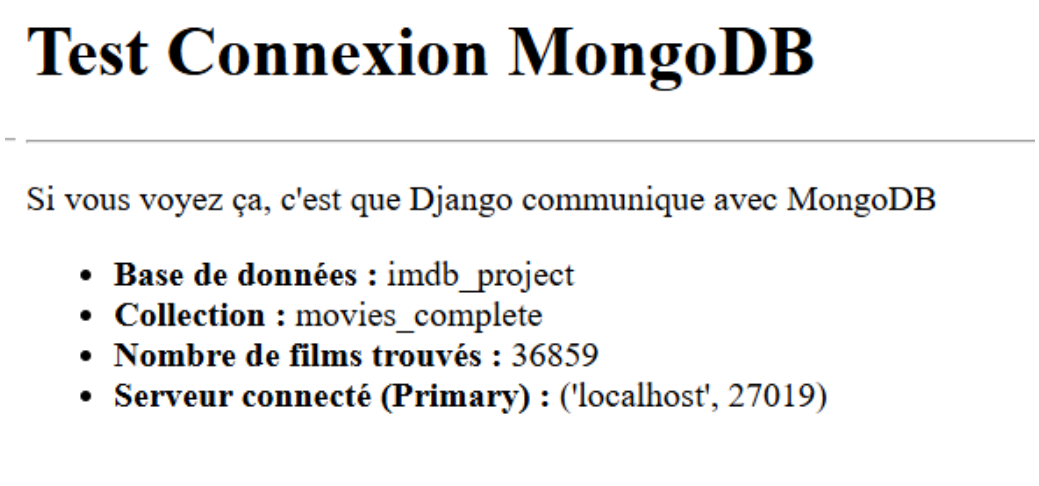

FIGURE 4 – Page de test Django confirmant l'accès aux données du cluster

5 Conclusion

La phase 3 est validée. Le cluster MongoDB assure la haute disponibilité requise et l'application Django est correctement connectée pour la suite du développement.