mongoDB
FOR GIANT IDEAS

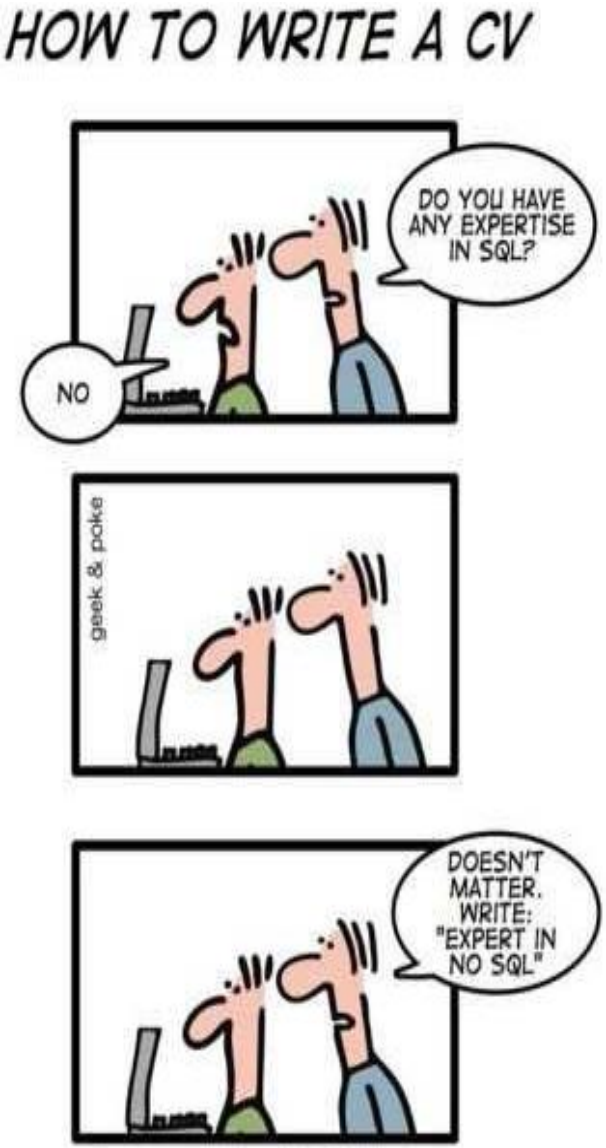
mongoDB
{ name: mongo, type: DB }

Cours de Bases de données NOSQL (MongoDB)

madaniabdellah@gmail.com

Plan du cours

- Rappels :
  - JSON
  - Bases de Données Relationnelles
- Introduction : Mouvement NoSQL
- MongoDB
  - Caractéristiques de MongoDB
  - Requêtes avec MongoDB
  - MongoDB et Java
  - MongoDB et PHP
  - **Réplication et reprise sur panne**
  - Partitionnement dans MongoDB
  - TextSearch



Réplication et reprise sur panne

madaniabdellah@gmail.com

Répliquer, à quoi ça sert ?

La réplication est un outil indispensable, universel pour la robustesse des systèmes distribués.

- Tolérance aux pannes : Vous cherchez un document sur S1, qui est en panne, On le trouvera sur S2
- Distribution des lectures : Répartissons les lectures sur S_1, S_2, … S_n pour satisfaire les millions de requêtes de nos clients.
- Distribution des écritures : Oui, mais attention, il faut réconcilier les données ensuite.

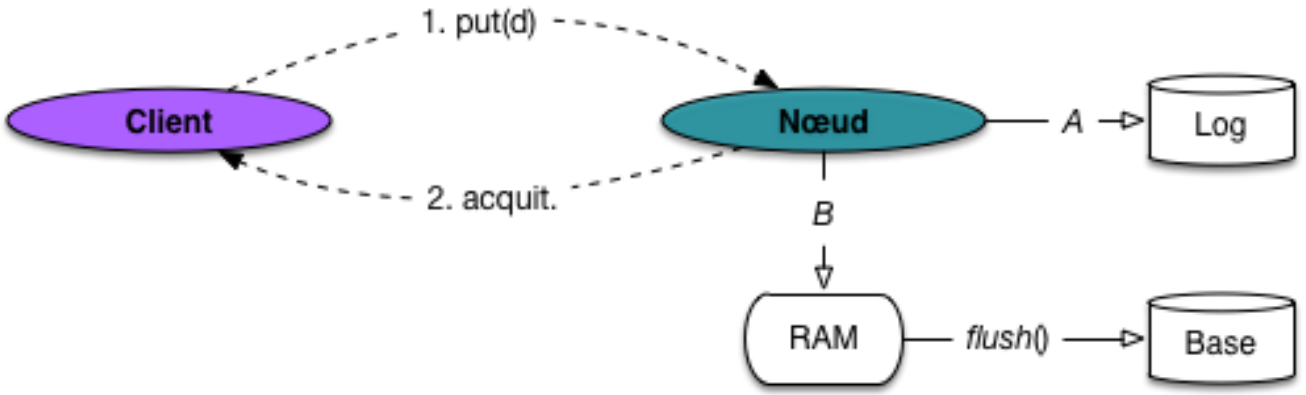
Trois copies pour une sécurité totale ; deux au minimum

Méthode générale de réplication

- Une application (le client) demande au système (le serveur) S1 l'écriture d'un document (unité d'information).
  - le serveur S1 écrit le document sur le disque ;
  - S1 transmet la demande d'écriture à un ou plusieurs autres serveurs S2, …., Sn, créant des replicas ou copies ;
  - S1 "rend la main" au client.
- Deux types de réplication synchrone/asynchrone ?
  - Synchrone : S1 attend confirmation de S2, …, Sn avant de rendre la main au client ; (Réplication bloquante)
  - Asynchrone : S1 rend la main sans attendre la confirmation des autres serveurs. (Réplication non bloquante)
- Conséquences
  - Une réplication asynchrone est beaucoup plus rapide, mais elle favorise les incohérences, au moins temporaires.

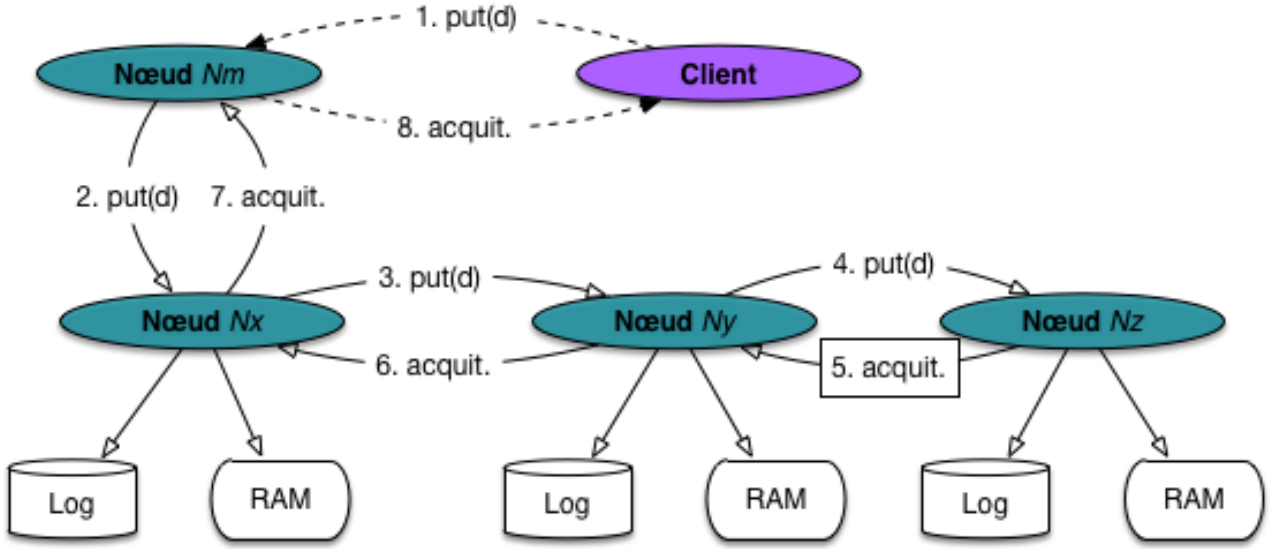
Utilisation du fichier journal dans la réplication

- Pour éviter la latence disque : le fichier journal
- Technique universellement utilisée en centralisé



Réplication avec écritures synchrones

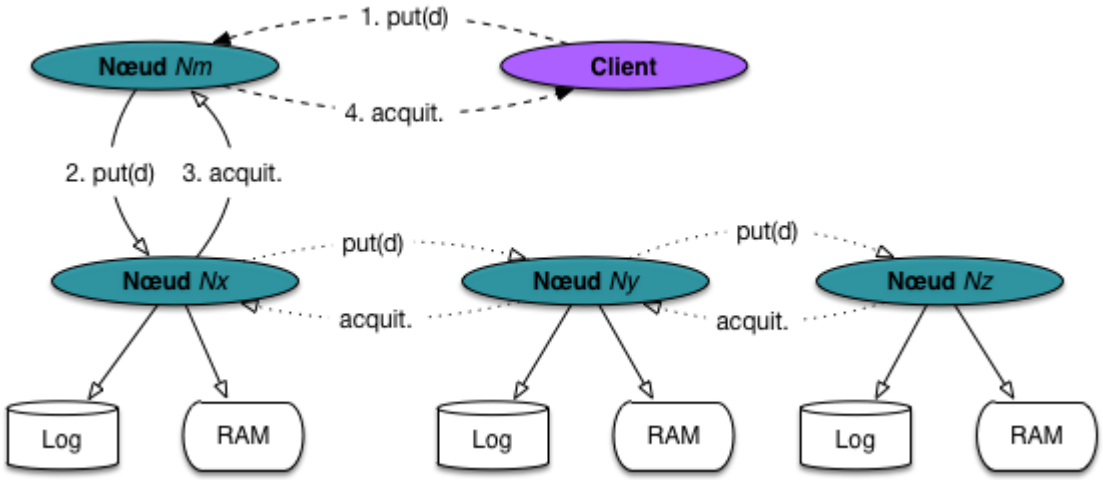
- Le client est acquitté quand tous les serveurs ont effectué l'écriture.



- Sécurité totale ; cohérence forte ; **très lent**.

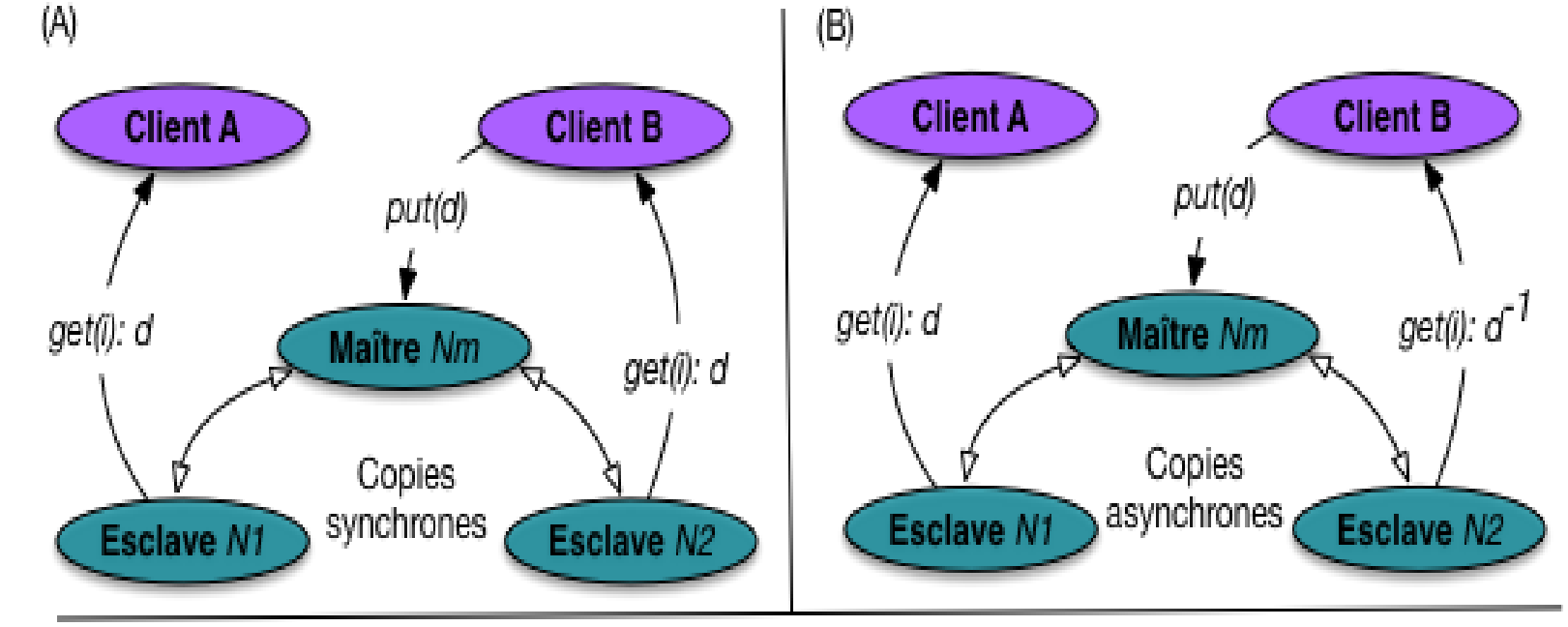
Réplication avec écritures asynchrones

- Le client est acquitté quand un des serveurs a effectué l'écriture. Les autres écritures se font indépendamment.



- Sécurité partielle ; cohérence faible ; Efficace.

Réplication synchrone vs asynchrone



Réplication synchrone vs asynchrone

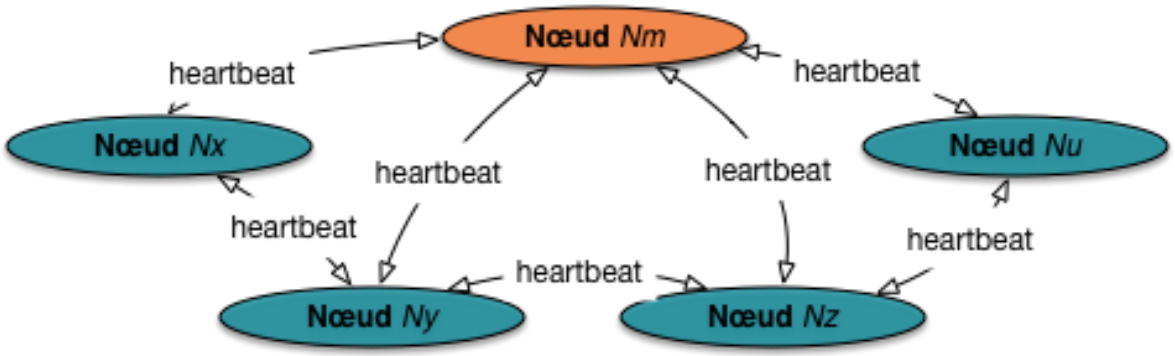
- Cas (A) : la topologie est de type maître-esclave, et les écritures sont synchrones. Toutes les écritures se font par une requête adressée au nœud-maître qui se charge de les distribuer aux nœuds-esclaves. L'acquittement n'est envoyé au client que quand *toutes* les copies sont en phase.
- Ce cas assure la cohérence *forte*, car toute lecture du document, quel que soit le nœud sur lequel elle est effectuée, renvoie la même version, celle qui vient d'être mise à jour. Cette cohérence se fait au prix de l'attente que la synchronisation soit complète, et ce à chaque écriture.

Réplication synchrone vs asynchrone

- Cas (B), la topologie est toujours de type maître-esclave, mais les écritures sont asynchrones.
- La cohérence n'est plus forte: il est possible d'écrire en s'adressant au nœud-maître, et de lire sur un nœud-esclave.
- Si la lecture s'effectue *avant* la synchronisation, il est possible que la version du document retournée soit non pas d mais d^{-1}, celle qui précède la mise à jour.

Réplication et reprise sur panne

- Principe général : tout le monde est interconnecté et échange des messages (heartbeat).



- Si un esclave tombe en panne, le système continue à fonctionner, tant qu'il est en contact avec une majorité d'esclaves
- Si le maître tombe en panne, les esclaves doivent élire un nouveau maître

Distribution et réplication dans MongoDB

MongoDB, système distribué maître-esclave

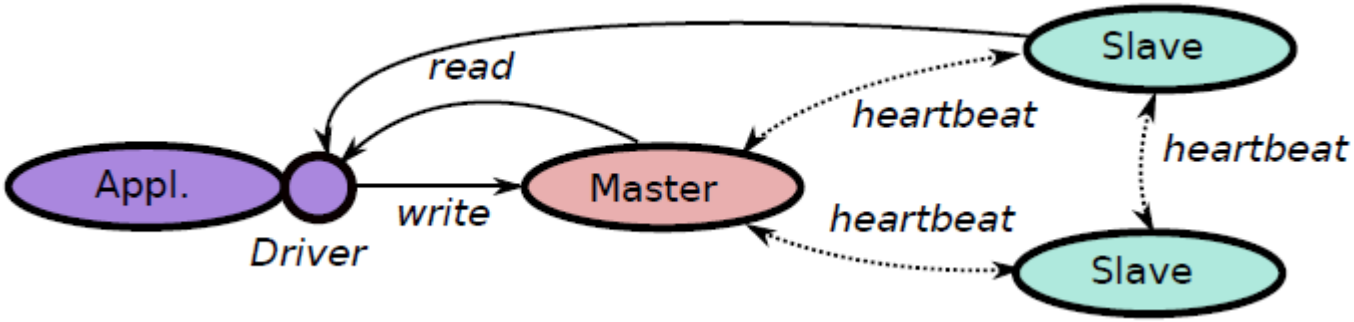
- MongoDB réalise la réplication en utilisant un réplica set (RS).
- Un RS est un groupe de nœud qui hébergent le même jeu de données. (Dataset)
- Dans un RS, un nœud est principal (primary, master) et les nœuds restants sont secondaires (Secondaries, Slaves)
- Le RS ne peut avoir qu'un seul nœud principal.
- Dans une réplication, le nœud principal reçoit toutes les opérations d'écriture.
- Tous les autres nœuds, appliquent des opérations à partir du nœud principal afin qu'elles aient le même ensemble de données.
- Toutes les données sont répliquées du nœud principal aux nœuds secondaires.

MongoDB, système distribué maître-esclave

- Le RS est un groupe de deux nœuds ou plus (généralement 3 nœuds minimum sont requis).
- Au moment du basculement automatique ou de la maintenance, l'élection est établie pour le nœud principal et un nouveau nœud principal est élu.
- Après la récupération du nœud défaillant, ce dernier rejoint à nouveau le RS et fonctionne en tant que nœud secondaire.

MongoDB, système distribué maître-esclave

- MongoDB gère la replication asynchrone au sein d'un replica set (RS)
- Le RS comprend un maître (primary) et des esclaves (secondary)
- Si nombre pair : ajout d'un processus supplémentaire (arbiter)
- Les écritures ont toujours lieu sur le maître
- Le driver associé à l'application peut lire sur le maître (cohérence forte) ou sur un secondary (cohérence faible, mais répartition)



Election d'un maître dans MongoDB

Une élection est déclenchée dans les cas suivants :

- Initialisation d'un nouveau RS
- Un esclave perd le contact avec le maître
- Le maître perd son statut car il ne voit plus qu'une minorité de participants.

Remarque

- Tous les clients (drivers) connectés au RS sont réinitialisés pour dialoguer avec le nouveau maître.
- Beaucoup d'options pour paramétrer le vote (p.e., une priorité donnée à chaque serveur).

Cohérence et répartition des lectures

- Par défaut, les clients lisent sur le maître : cohérence forte.
- Une option de connexion permet à l'application d'indiquer qu'elle accepte de lire sur un esclave : cohérence faible

Remarque

- La réplication n'est pas, dans MongoDB, le moyen privilégié de passer à l'échelle. Le partitionnement est un mécanisme plus puissant (à suivre !)

Test de la réplication

- On crée trois instances du serveur sur la même machine, avec une base chacun.

cd /data; md noeud1; md noeud2; md noeud3

- Lançons maintenant les 3 serveurs, chacun sur un port. Ils forment le RS test.

mongod --port 27017 --replSet test --dbpath /data/noeud1

mongod --port 27018 --replSet test --dbpath /data/noeud2

mongod --port 27019 --replSet test --dbpath /data/noeud3

- Connectons un client au premier serveur.

mongo --port 27017

- Et initialisons le RS

rs.initiate()

- On peut alors ajouter les autres noeuds (remplacer localhost par le nom de la machine).

rs.add("localhost:27018")

rs.add("localhost:27019")

Test de la réplication (suite)

- La fonction suivante indique le statut du nœud auquel on est connecté.

db.isMaster()

- Information plus complète sur le RS avec :

rs.status()

- Tout va bien ? Alors insérons.

use maBase;

db.test.insert ({"produit": "Grulband", prix: 230, enStock: true})

- On devrait trouver le document sur le esclave ?

mongo --port 27018

use maBase;

db.test.find ()

- Que se passe-t-il… ? Et après rs.slaveOk() ? Pourquoi donc ?

Test de la reprise sur panne

- Tuons notre maître.

db.shutdownServer()

- Question : le second nœud va-t-il s'élire maître ?
- Relancer le premier serveur. Que se passe-t-il ?
- Redémarrer le premier serveur. Que se passe-t-il ?

Remarque

- Un bon driver devrait enregistrer la liste des serveurs du RS, et se reconnecter automatiquement au maître quand celui-ci change.