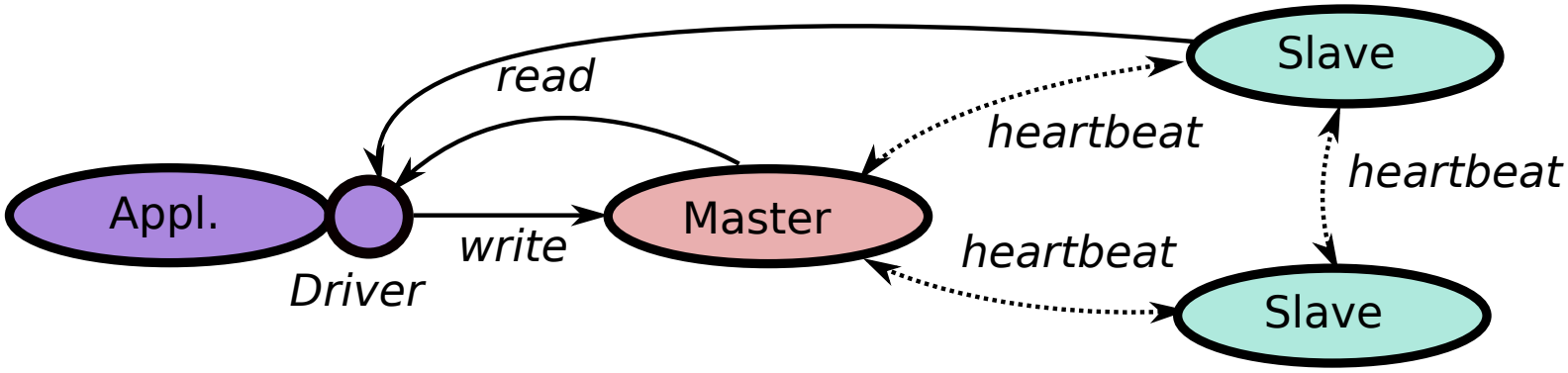
Bases de données documentaires et distribuées
Cours NFE04
Distribution et réplication dans MongoDB

Auteurs : Raphaël Fournier-S'niehotta, Philippe Rigaux, Nicolas Travers
prénom.nom@cnam.fr

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

MongoDB, système distribué *maître-esclave*



Un replica set, MongoDB

- 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 connection permet à l'application d'indiquer qu'elle accepte de lire sur un *esclave* : **cohérence faible**

Autre option : l'application peut attendre que *n* écritures soient faites avant de reprendre la main.

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 !)

A l'action ! Test de la réplication

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

```
cd /data; mkdir noeud1; mkdir noeud2
```

Lançons maintenant les 2 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&
```

Connectons un client au premier serveur.

```
mongo --port 27017
```

Et initialisons le RS

```
rs.initiate()
```

On peut alors ajouter le second nœud (remplacer `localhost` par le nom de la machine).

```
rs.add("localhost:27018")
```

Continuons

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

```
db.ismaître()
```

Information plus complète sur le RS avec :

```
rs.status()
```

Tout va bien ? Alors insérons.

```
use nfe204;
db.test.insert ({"produit": "Grulband", prix: 230, enStock: true})
```

On devrait trouver le document sur le *esclave* ?

```
mongo --port 27018
use nfe204;
db.test.find ()
```

Hmmm que se passe-t-il... ? Et après `rs.slaveOk()` ? Pourquoi donc ?

Et la reprise sur panne ?

Tuons (gentiment) notre *maître*.

```
db.shutdownServer()
```

Question : le second va-t-il s'élire *maître* ?

Relancer le premier serveur. Que se passe-t-il ?

Répéter l'expérience en ajoutant un troisième nœud.

```
mkdir /db/noeud3;
mongod --port 27019 --replSet test --dbpath /data/noeud3\&
```

Remarque

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