3IN017 – TECHNOLOGIES DU WEB

MongoDB

18 mars 2025

Gilles Chagnon

Plan

SCIENCES SORBONNE UNIVERSITÉ

Cours précédents côté client, puis côté serveur, puis échanges entre client et serveur.

Mais où est la source des données ? Il nous faut une base de données pour les stocker...

Bases de données NoSQL

NoSQL – *Historique*

SCIENCES SORBONNE UNIVERSITÉ

Historique

- NoSQL : Not Only SQL
- 2004 : Big Table Google
- 2008 : Cassandra Facebook
- 2009 : lancement d'une communauté de développement logiciels open source NoSQL

Spécificités

- Manipulation de données peu (pas) structurées, schéma flexible, sans modèle pré-défini
- Manipulation de données volumineuses
- Haute disponibilité
- Capacités élevées de lectures/écritures

NoSQL

SCIENCES SORBONNE UNIVERSITÉ

NoSQL

NoSQL (Not Only SQL), apparu en 2009, et comme son nom l'indique une alternative au langage SQL et au modèle relationnel de base de données que vous avez l'habitude d'utiliser. L'idée est de proposer une architecture souple et puissante avec une forte disponibilité et de faibles contraintes.

Particulièrement à la mode, la majorité des grandes entreprises du web abandonnent leurs bases de données traditionnelles et portent leur propre projet NoSQL : Facebook et X (ex-Twitter) utilisent Cassandra (de la Fondation Apache), Amazon SimpleDB, LinkedIn Voldemort, *etc.*

Théorème CAP : Propriétés fondamentales pour les systèmes distribués

SCIENCES SORBONNE UNIVERSITÉ

- **C**oherence : tous les nœuds du système voient exactement les mêmes données au même moment
- **A**vailability (Disponibilité) : garantie que toutes les requêtes reçoivent une réponse ;
- **P**artition tolerance (Résistance au partitionnement) : sauf coupure totale du système, aucune panne ne peut empêcher le système de fonctionner normalement

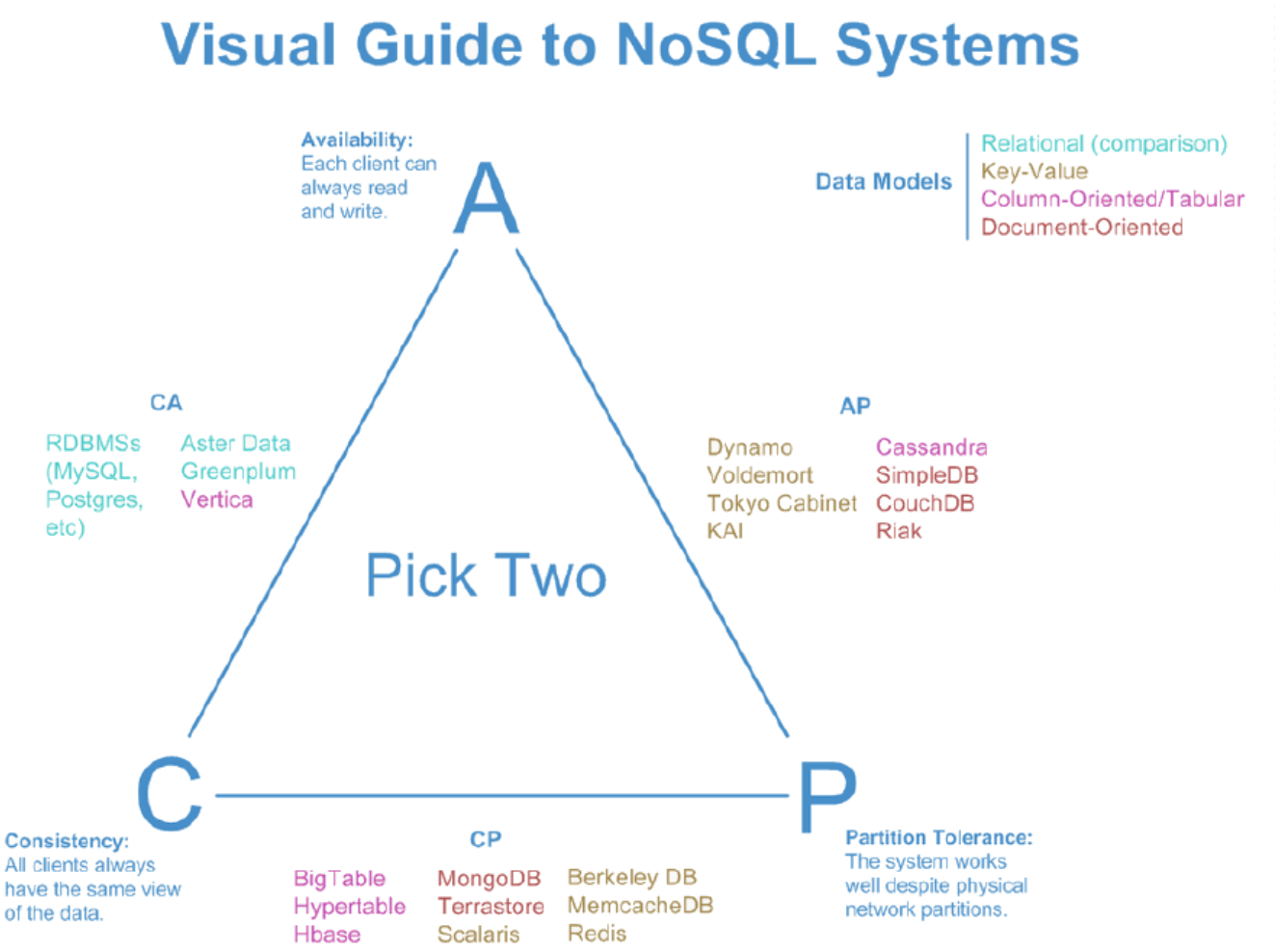
Dans un système distribué, seules deux propriétés sur trois peuvent être parfaitement assurées à un instant t.

Exemples

Soit A et B deux utilisateurs, soit N1 et N2 deux nœuds.

- Si A modifie une valeur sur N1, alors pour que B voie cette valeur sur N2 il faut attendre que N1 et N2 soient synchronisés (cohérence privilégiée).
- Si A fait une requête à N1, et B la même requête à N2, alors ils peuvent obtenir des résultats différents, mais cet inconvénient est moins grave que de ne pas avoir de résultat du tout (disponibilité privilégiée)
- Si on veut maintenir cohérence et résistance au partitionnement, on n'utilise qu'un seul serveur, mais c'est alors au détriment de la disponibilité

NoSQL


Visual Guide to NoSQL Systems
Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

Introduction à MongoDB

MongoDB

SCIENCES SORBONNE UNIVERSITÉ

MongoDB (Wikipédia)

MongoDB est un système de gestion de base de données :

- orientée documents,
- plus libre (changement de licences),
- montant bien en puissance (*scalable*),
- à performance raisonnable,
- ne nécessitant pas de schéma prédéfini des données,
- écrit avec le langage de programmation C++.

Il fait partie de la mouvance NoSQL et vise à fournir des fonctionnalités avancées. Utilisée par Foursquare, bit.ly, Doodle, Disqus, Walmart, Bosch pour leur IoT, Verizon...

Des *drivers* existent pour les langages de programmation les plus utilisés : C, C++ , Erlang, Haskell, Java, Javascript, .NET (C# F#, PowerShell, *etc.*), Perl, PHP, Python, Ruby, Scala...

MongoDB : Principes

SCIENCES SORBONNE UNIVERSITÉ

- Une instance de Mongodb peut contenir une ou plusieurs bases de données
- Une base de données peut avoir une ou plusieurs **collections** (à peu près l'équivalent des tables dans les bases classiques)
- Une collection peut avoir de zéro à plusieurs **documents**
  - Les documents d'une même collection n'ont pas obligatoirement le même « schéma » (les mêmes champs)
  - Les documents correspondent aux enregistrements (tuples)
  - Des documents peuvent contenir des documents (structure hiérarchique)
  - Chaque document possède une clef unique
- MongoDB stocke l'information au format BSON (Binary JSON)

MongoDB : Propriétés

SCIENCES SORBONNE UNIVERSITÉ

Avantages

- Requêtes « faciles »
- Fait sens avec la plupart des applications Web (utilisation naturelle avec JavaScript)
- Facilité d'intégration des données

Inconvénients

- Pas bien adapté pour les systèmes transactionnels complexes
- Représentation hiérarchique des données : difficile de faire figurer des relations entre objets

MongoDB – Document (1)

SCIENCES SORBONNE UNIVERSITÉ

Un document (= un JSON) :

```
user = {
        "name": "Dupont",
        "occupation": "détective",
        "location": "Moulinsart"
}
```

MongoDB – Document (2)

SCIENCES SORBONNE UNIVERSITÉ

Un autre document :

```
{
        "name": "Maude Zarella",
        "address": {
                "street": "Place Ticot",
                "number": 42,
                "city": "Saint-Nectaire",
                "zip": 63710
        },
        "hobbies": [ "parachutisme", "couture" ],
        "bearded": false
}
```

MongoDB – Collection

Une collection :

```
{       "_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
        "lastName": "Térieur",
        "firstName": "Alex",
        "dateOfBirth": "1980-5-24"
},
{       "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
        "lastName": "Térieur",
        "firstName": "Alain",
        "dateOfBirth": "1981-6-25",
        "Address": "7 rue des Preuves, 24130 la Force"
}
```

Les "_id" sont générés automatiquement par le SGBD.

Modélisation (1)

Imaginons que l'on veuille modéliser un blog :

Mauvais design

```
post = {
        id: 150,
        author: 100,
        text: "Recette de cuisine époustouflante",
        comments: [100, 105, 112]
}
author = {
        id: 100,
        name: "Jessica Sroll",
        posts: [150]
}
comment = {
        id: 105,
        text: "Intéressant point de vue"
}
```

L'utilisation d'id permet de se rapprocher d'une base relationnelle, mais les requêtes se compliquent quand on veut ajouter un nouveau commentaire.

Modélisation (2)

SCIENCES SORBONNE UNIVERSITÉ

Autre design

```
post = {
        author: "Jessica Sroll",
        text: "Recette de cuisine époustouflante",
        comments: [
                "Intéressant point de vue",
                "Tout à fait d'accord"
        ]
}
```

Pourquoi celui-ci est-il meilleur pour un blog ?

MongoDB – Installation, lancement et surveillance

Installation et lancement

Deux composants :

- le serveur lui-même. Il se lance :
  - en ligne de commande en tapant `mongod` qui lance le démon
  - avec l'utilitaire graphique Compass
  - comme un service (avec systemd, `sudo systemctl start mongodb` – c'est la situation dans les salles de la PPTI)

  Le serveur est accessible à l'adresse `mongodb://localhost:27017` (port par défaut).
- le client, qui utilise un *driver* pour accéder au serveur. Le *driver* s'installe pour un serveur Node.js avec `npm install mongodb`

Mongo shell

La commande `mongosh` ou `mongo` permet d'accéder à un environnement (en ligne de commande) pour réaliser un certain nombre d'opérations. Outre les commandes node.js que nous verrons plus loin, il y a aussi :

- `db` pour afficher la base de données courante (par défaut `test`)
- `show dbs` pour afficher la liste des bases de données du serveur
- `use db_name` pour utiliser la base `db_name`. Si cette base n'existe pas, elle est créée.
- une fois une base nommée `db_name` sélectionnée, on peut la supprimer avec `dropDatabase()`
- `show collections` pour afficher la liste des collections de la base courante
- `db.getCollectionInfos()` affiche des informations sur les collections de la base courante
- pour supprimer la collection `col1` de la base courante, il faut saisir `db.col1.drop()`

En résumé...

1. Installation du client avec `npm install mongodb`
2. Lancement du serveur avec `mongod` ou Compass s'il ne tourne pas déjà
3. Éventuellement monitoring du serveur avec `mongosh`

Utilisation avec Node.js

Connexion au serveur MongoDB local

SCIENCES SORBONNE UNIVERSITÉ

```
const {MongoClient} = require('mongodb');

async function main(){
        // URI de connexion. À modifier si on n'est pas en local
        const uri = "mongodb://localhost";
        const client = new MongoClient(uri);

        try {
                // Connexion au serveur
                await client.connect();

                // ici le code à exécuter...
                await faisCi(client);
                await faisCa(client);

        } catch (e) {
                // si une des promesses n'est pas réalisée
                console.error(e);
        } finally {
                // une fois que tout est terminé, on ferme la connexion
                await client.close();
        }
}
// main est une promesse, donc peut afficher un message en cas d'échec
main().catch(console.error);
```

Accès aux bases et aux collections

SCIENCES SORBONNE UNIVERSITÉ

- Accès à la base `base1` : après `await client.connect()`, donc une fois connecté, il suffit d'écrire `client.db("base1");`
- Accès à la collection collec1 de la base1 : `client.db("base1").collection("collec1");` ou directement

```
const col1 = await client.db("base1").collection("collec1");
```

Si cette collection n'existe pas, elle est créée.

Création de document(s)

SCIENCES SORBONNE UNIVERSITÉ

- Création d'un document (= ajout à une collection) : `insertOne()`

```
//newDoc est un objet JavaScript
const res1 = await
↪  client.db("base1").collection("collec1").insertOne(newDoc);
```

- Création de plusieurs documents : `insertMany()`

```
//newDocs est un tableau d'objets JavaScript
const resBeaucoup = await
↪  client.db("base1").collection("collec1").insertMany(newDocs);
```

Lecture d'un document

Avec `findOne()` appliquée à la collection choisie. Cette méthode prend un paramètre facultatif permettant de restreindre la recherche. `findOne()` renvoie le premier document répondant aux critères de recherche. Par exemple...

```
/* renvoie le premier document de la collection collec1 possédant une
↪  propriété valeurNum dont la valeur est supérieure ou égale à 50, et
↪  une propriété valeurBool de type booléen valant true */
const res1 = await client.db("base1").collection("collec1").findOne(
        {
                valeurNum: $gte: 50,
                valeurBool: $eq: true
        }
);
```

Voir la liste des filtres possibles :
`https://www.mongodb.com/docs/manual/reference/operator/query-comparison/`

Lecture de plusieurs documents

SCIENCES SORBONNE UNIVERSITÉ

Avec `find()` appliquée à la collection choisie. Cette méthode prend un paramètre facultatif permettant de restreindre la recherche. `find()` renvoie un « curseur » répondant aux critères de recherche. Par exemple...

```
/* renvoie un curseur pointant vers les documents de la collection
↪  collec1 possédant une propriété valeurNum dont la valeur est
↪  supérieure ou égale à 50, et une propriété valeurBool de type
↪  booléen valant true */
const cursor2 = await client.db("base1").collection("collec1").find(
        {
                valeurNum: $gte: 50,
                valeurBool: $eq: true
        }
);
// Il faut ensuite transformer cursor2 en Array pour l'exploiter:
const result2 = await cursor2.toArray();
```

Des méthodes de traitement sont disponibles pour le curseur :
`https://mongodb.github.io/node-mongodb-native/3.6/api/Cursor.html`

Mise à jour

Pour mettre à jour un document, on utilise `updateOne()`, qui prend un filtre (éventuellement vide) en premier paramètre, et la valeur de remplacement en deuxième. Par exemple...

```
const res1 = await client.db("base1").collection("collec1").updateOne(
        {
                valeurNum: $gte: 50,
        },
        {
                $set: doc2
        }
);
```

doc2 ne remplace pas le 1er document répondant au filtre ; mais ses propriétés viennent compléter ou remplacer celles de ce premier document portant le même nom ou n'étant pas renseignées.

On procède de manière similaire avec `updateMany()`.

Suppression

Sur le même principe de fonctionnement que `findOne` ou `updateOne` d'une part, `find` ou `updateMany` d'autre part avec le recours à des filtres, on peut supprimer un ou des documents avec respectivement `deleteOne()` et `deleteMany()`.

Pour le projet. . .

Pour le projet

- sur vos machines avec un client en ligne de commande, Compass ou en ligne avec Atlas (`https://www.mongodb.com/fr-fr/atlas/database`)
- sur les machines de la salle avec Compass

Si vous utilisez votre propre machine, vous pouvez installer MongoDB *mais* dans tous les cas il faudra fournir aux enseignants de quoi créer les bases de données avec des données suffisantes pour pouvoir vérifier le bon fonctionnement de votre code (par exemple *via* un service `populateDatabase` qui à la première connexion au serveur, vérifie si la base de données est vide et si c'est le cas la remplit, ou au lancement du serveur).

En vrai

- Vraie connexion à un vrai serveur
- Nécessite de gérer un serveur