

# Simulation d'algorithmes d'équilibrage de charge dans un environnement distribué

Identifications des besoins

Kevin Barreau

Guillaume Marques

Corentin Salingue

1<sup>er</sup> février 2015

## **Résumé**

Ce document dégage une première identification des besoins. Il s'agit d'un document support pour l'élaboration du cahier des charges

# Sommaire

<b>1</b>	<b>Définition du projet</b>	<b>3</b>
1.1	Contexte . . . . .	3
1.2	Finalité . . . . .	3
<b>2</b>	<b>Hierarchisation des besoins</b>	<b>5</b>
2.1	Priorité . . . . .	5
2.2	Criticité . . . . .	5
<b>3</b>	<b>Besoins fonctionnels</b>	<b>6</b>
3.1	Environnement distribué . . . . .	6
3.2	Gestion d'un réseau . . . . .	6
3.2.1	Gestion des noeuds . . . . .	6
3.2.2	Gestion des objets . . . . .	6
3.2.3	Popularité d'un objet . . . . .	6
3.2.4	Topologie du réseau . . . . .	6
3.2.5	Sauvegarde d'un réseau . . . . .	7
3.2.6	Importation d'un réseau . . . . .	7
3.3	Algorithmes . . . . .	7
3.3.1	Algorithmes à implémenter . . . . .	7
3.3.2	Conformité des implémentations . . . . .	7
3.3.3	Autres algorithmes . . . . .	8
3.4	Simulation de requêtes . . . . .	8
3.4.1	Gestion des requêtes . . . . .	8
3.4.2	Envoi d'une requête . . . . .	8
3.4.3	Création d'un jeu de requête . . . . .	8
3.4.4	Sauvegarde d'un jeu de requête . . . . .	8
3.4.5	Importation d'un jeu de requête . . . . .	8
3.5	Visualisation des données . . . . .	8
<b>4</b>	<b>Besoins non fonctionnels</b>	<b>9</b>
4.1	Cassandra . . . . .	9
4.2	Gestion d'un réseau . . . . .	9
4.2.1	Communication entre noeuds . . . . .	9
4.2.2	Méthode de calcul de la popularité . . . . .	9
4.3	Algorithmes . . . . .	9
4.4	Visualisation des données . . . . .	9
4.4.1	Vues . . . . .	9
4.4.2	Actualisation des vues . . . . .	9
<b>5</b>	<b>Livrables</b>	<b>10</b>

# 1 Définition du projet

## 1.1 Contexte

L'expansion, au cours des deux dernières décennies, des réseaux et notamment d'Internet a engendré une importante création de données, massives par leur nombre et leur taille. Stocker ces informations sur un seul point de stockage (ordinateur par exemple) n'est bien sûr plus envisageable, que ce soit pour des raisons techniques ou pour des raisons de sûreté (pannes potentielles par exemple). Pour cela des systèmes de stockages dits distribués sont utilisés en pratique afin des les répartir sur différentes unités de stockages.

**Définition** Un environnement distribué est constitué de plusieurs machines (ordinateurs), appelées *noeuds*, sur lesquelles sont stockées des données.

**Définition** Une donnée est une suite binaire de 0 et de 1 dont le contenu n'est pas important pour l'application.

Le client souhaite répartir toutes ces données de manière équitable entre les noeuds. De plus, il souhaite que ces données soient accessibles afin de pouvoir les requêter et récupérer de l'information.

**Définition** Une requête est un message envoyé à une machine (ou plusieurs machines) afin de récupérer ou de modifier de l'information sur des données. Nous noterons que la nature de l'information est inutile pour le bon fonctionnement de l'application.

Pour répartir toutes ces données, notre client a développé de nouveaux algorithmes d'équilibrage de charge et de réplication qu'il souhaite tester dans un environnement distribué.

**Définition** Une *charge* est associée à un noeud et désigne le nombre de requêtes que le noeud doit traiter.

**Définition** La réplication d'une donnée consiste à faire des copies de cette donnée sur d'autres noeuds.

## 1.2 Finalité

Nous devons développer une solution logicielle permettant de tester ces nouveaux algorithmes d'équilibrage de charge et de réplication dans un environnement distribué.

**Définition** Un réseau est un ensemble de noeuds qui sont reliés entre eux (en général par Internet) et qui communiquent ensemble afin de traiter toutes les requêtes reçues.

**Définition** La topologie d'un réseau représente l'architecture physique ou logicielle des liens entre les noeuds. Elle comporte aussi des informations sur la hiérarchie des noeuds, le placement spatial et les divers équipements reliant les noeuds.

**Définition** L'état d'un réseau est l'ensemble des informations caractérisant un réseau (topologie par exemple) ainsi que des informations sur les noeuds (comme leur charge actuelle).

**Définition** Un jeu de données est un ensemble de données dont on connaît la position sur les noeuds (la donnée numéro X est sur le noeud numéro Y), éventuellement, leur contenu et qu'on est capable d'exporter et de reproduire. L'export pourra se faire sous la forme d'un jeu de requêtes pour placer les données choisies.

**Définition** Un jeu de requêtes est un ensemble de requêtes qu'on est capable d'exporter sous forme d'un programme ou d'un fichier qui sera exécuté par la solution et qui peut être reproduit.

Cette solution doit permettre le paramétrage d'un réseau, c'est-à-dire le nombre de noeuds souhaité et la topologie du réseau. Le client pourra simuler différents jeux de données et jeux de requêtes sur ce réseau. Cela permettra de comparer l'efficacité de ces algorithmes avec le même environnement (même jeux de données et de requêtes). Il pourra tester ses algorithmes implémentés. Enfin, il pourra visualiser la topologie et l'état du réseau *à tout moment*.

## 2 Hiérarchisation des besoins

Nous avons dégagé des précédentes réunions, une liste de besoins fonctionnels et non-fonctionnels. Pour mieux les comparer, nous les avons hiérarchisés en fonction de leur priorité et de leur criticité.

### 2.1 Priorité

La priorité est un indicateur de l'ordre dans lequel nous devons implémenter les besoins afin de satisfaire au mieux les exigences du client.

Valeur	Signification	Description
1	Priorité haute	A implémenter dans les premiers temps
2	Priorité moyenne	A implémenter
3	Priorité faible	A implémenter (en fonction du temps restant)

### 2.2 Criticité

Le niveau de criticité d'un besoin est un indicateur de l'impact qu'aura la non-implémentation de ce besoin sur le bon fonctionnement de l'application.

Valeur	Signification	Description
1	Criticité extrême	L'application ne fonctionnera pas
2	Criticité haute	Certaines fonctionnalités de l'application ne fonctionneront pas
3	Criticité moyenne	Certaines fonctionnalités seront perturbées
4	Criticité faible	L'application fonctionnera correctement

## 3 Besoins fonctionnels

L'environnement de simulation voulu est un système distribué constitué de  $n$  noeuds de stockage dans lequel on souhaite stocker  $m$  objets.

### 3.1 Environnement distribué

Nous évoluerons dans un environnement distribué (*Priorité* : 1, *criticité* : 1) . En effet, le client souhaite tester des algorithmes d'équilibrage de charges et de gestion de copies qui ne peuvent fonctionner uniquement dans un environnement distribué.

### 3.2 Gestion d'un réseau

Comme défini précédemment, un réseau est un ensemble de noeuds qui sont reliés entre eux (en général par Internet) et qui communiquent ensemble afin de traiter toutes les requêtes reçues.

#### 3.2.1 Gestion des noeuds

Un noeud est une machine (ordinateur généralement) pouvant stocker des données et traiter des requêtes. Un noeud possède des données internes. Ces données sont propres au fonctionnement du noeud (comme la charge par exemple).

**Création d'un noeud** La création d'un noeud (*Priorité* : 1, *criticité* : 1) se fait au moment où le réseau n'existe pas encore. Le projet ne prend en compte ni la création dynamique de noeud (création et ajout d'un noeud après que le réseau soit créé), ni la suppression de noeud. Il est possible de séparer ce besoin en plusieurs sous-besoins :

- Créer un noeud dans l'environnement de simulation
- Initialiser les données internes d'un noeud

**Données d'un noeud** De plus, l'application doit permettre :

- Récupérer les données internes d'un noeud (*Priorité* : 1, *criticité* : 1)
- Mettre à jour les données internes d'un noeud (*Priorité* : 1, *criticité* : 1)

#### 3.2.2 Gestion des objets

Un objet est...

Le client souhaite pouvoir :

- Créer un objet
- Supprimer un objet

#### 3.2.3 Popularité d'un objet

Les algorithmes du client nécessitent de connaître la popularité d'un objet dans le réseau. La popularité d'un objet est fonction du nombre de requêtes sur cet objet. Plus ce nombre de requêtes est grand, plus l'objet est populaire.

#### 3.2.4 Topologie du réseau

Un noeud maître est un noeud connu de tous les autres noeuds dans le réseau. Un noeud communique à un intervalle de temps régulier avec les autres noeuds du réseau. Tous les noeuds du réseau possèdent des informations sur les autres noeuds du réseau.

L'application doit permettre :

- Communiquer les données internes à un noeud maître (*Priorité* : 1, *criticité* : 1)
- Paramétrer l'intervalle de temps de communication (*Priorité* : 2, *criticité* : 3)
- Stocker les informations des autres noeuds (*Priorité* : 1, *criticité* : 2)

### 3.2.5 Sauvegarde d'un réseau

Objet de la prochaine réunion

### 3.2.6 Importation d'un réseau

Objet de la prochaine réunion

## 3.3 Algorithmes

### 3.3.1 Algorithmes à implémenter

**Définitions** La charge minimum est ...

La charge moyenne est ...

Les algorithmes d'équilibrage de charge à implémenter sont (extrait du modèle du client) :

- **SLVO** (*Priorité* : 1, *criticité* : 1)
  - Déterminer la charge minimum courante.
  - Si la charge du noeud est inférieure ou égale à la charge minimum, il s'affecte toutes les requêtes en attente et en avertit les autres noeuds.
- **AverageDegree** (*Priorité* : 1, *criticité* : 1)
  - Déterminer la charge moyenne courante
  - Si la charge du noeud est inférieure ou égale à la charge moyenne, il s'affecte toutes les requêtes en attente et en avertit les autres noeuds.

**Gestion des copies** Entité à part entière ? (prochaine réunion)

### 3.3.2 Conformité des implémentations

La conformité des algorithmes implémentés est assurée par des jeux de tests suivant la démarche :

- Créer un jeu de test à la main, comprenant les paramètres de création et l'ensemble des requêtes à traiter
- Faire fonctionner l'algorithme à la main
- Stocker l'état final du fonctionnement de l'algorithme
- Faire valider le processus à la main par le client
- Lancer la simulation de l'algorithme avec le jeu de test créé à la main
- Vérifier les résultats constatés avec les résultats attendus

Si il y a une différence entre le résultat de l'exécution de l'algorithme à la main et le résultat de l'exécution par simulation, une vérification par le client peut être envisagée dans le cas de résultats *presque* similaires. La notion de similitude est laissée à l'appréciation de l'équipe en charge du projet, lors de la vérification.

### 3.3.3 Autres algorithmes

Le client souhaite aussi avoir la possibilité de tester d'autres algorithmes. Notre solution doit donc proposer un système permettant d'implémenter ces algorithmes et de les tester. (*Priorité : 3, criticité : 3*)

## 3.4 Simulation de requêtes

Une requête est un message envoyé à une machine (ou plusieurs machines) afin de récupérer ou de modifier de l'information sur des données.

### 3.4.1 Gestion des requêtes

Le client souhaite pouvoir :

- Créer une requête
- Supprimer une requête

### 3.4.2 Envoi d'une requête

L'envoi d'une requête peut se faire

### 3.4.3 Création d'un jeu de requête

Objet de la prochaine réunion

### 3.4.4 Sauvegarde d'un jeu de requête

Objet de la prochaine réunion

### 3.4.5 Importation d'un jeu de requête

Objet de la prochaine réunion

## 3.5 Visualisation des données

- Temps de réponse moyen sur les requêtes passées.
  - Charge d'un noeud
  - Popularité des objets
- Note** Bien définir ces items.



## **4 Besoins non fonctionnels**

### **4.1 Cassandra**

Cassandra est une base de données distribuée.

### **4.2 Gestion d'un réseau**

#### **4.2.1 Communication entre noeuds**

Lorem ipsum

#### **4.2.2 Méthode de calcul de la popularité**

Lorem ipsum

### **4.3 Algorithmes**

### **4.4 Visualisation des données**

Une vue correspond à une fenêtre de l'application, c'est à dire ce que voit l'utilisateur.

#### **4.4.1 Vues**

Lorem ipsum

#### **4.4.2 Actualisation des vues**

Lorem ipsum

## 5 Livrables

