[](mailto:chenhe.he@outlook.fr)

Entrepôt réparti en mémoire

**Cahier des charges**

**Clients :** SENS Pierre, KORDON Fabrice

[Pierre.Sens@lip6.fr](mailto:Pierre.Sens@lip6.fr), [Fabrice.Kordon@lip6.fr](mailto:Fabrice.Kordon@lip6.fr)

**Concepteurs :** HE Chenhui - MARECAR Farzana

[chenhui.he@outlook.fr](mailto:chenhe.he@outlook.fr), [farzana.marecar@outlook.fr](mailto:farzana.marecar@outlook.fr)

01/03/2018

# 

[**Introduction**](#_c3yklfr3tmfs) **1**

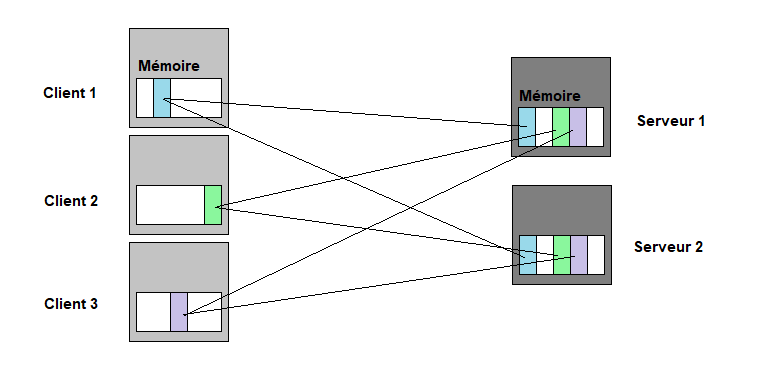
[**Les Besoins**](#_s4wqr413xuuk) **2**

[**Les Conditions d’utilisation**](#_obe38p2a41os) **4**

[**L’Architecture logicielle choisie**](#_po49r4owah1b) **5**

[**La(les) Base(s) technologique(s) employés**](#_5hceptqr3pxa) **6**

[**État de l’art**](#_gye633uill9y) **6**

****

# 

# Introduction

*Un système réparti est un ensemble de machines autonomes connectées par un réseau, et équipées d’un logiciel dédié à la coordination des activités du système ainsi qu’au partage de ses ressources -* **G. Coulouris, J. Dollimore, and T. Kindberg.**

*Un système réparti est un ensemble d’ordinateurs indépendants qui apparaît à un utilisateur comme un système unique et cohérent -* **A. Tanenbaum**

*A distributed system is one on which I can’t do my work some computer has failed that I never heard of -* **L. Lamport**

La sauvegarde des données est indispensable pour sécuriser un système d’informations : les entreprises ne peuvent pas négliger ce maillon dans la chaîne.

Un ensemble d’outils est disponible aujourd’hui pour sauvegarder les donnée de toute structure. Il y a des systèmes traditionnels comme les librairies de bandes ou bien librairies virtuelles avec le plu souvent un moteur de déduplication (Falconstor VTL, Exagrid, Quantum DXi). Nous retrouvons ensuite les systèmes plus évolués faisant appel aux clichés instantanés (snapshots) et à la réplication de données.

C’est l’objet de notre demande pour ce projet : créer un système persistant avec une architecture répartie qui duplique sur un ensemble de N serveurs des segments mémoire de données allouées et manipulés par nos clients. Créer surtout un logiciel dédié à la coordination des activités du système ainsi qu’au partage de ses ressources.

Les Problèmes

## Les Besoins

Le systèmes se composent de 2 types d’hôtes:

* Les clients:
  + Ils auront à disposition ce système persistant et doivent pouvoir créer des données qui seront stockées dans les N serveurs
  + Ils doivent pouvoir se connecter à n’importe lequel des N serveurs et pouvoir récupérer les données chez n’importe lequel
  + Ils doivent pouvoir récupérer en local une copie d’une donnée existante afin de la consulter (seulement la possibilité de lire)
  + Ils doivent aussi pouvoir accéder à une donnée en lecture/écriture, càd récupérer locale d’une donnée existante tant que personne d’autre manipule la donnée, et la manipuler directement en mémoire
  + A la fin de son utilisation, le client doit pouvoir libérer la donnée pour libérer aux autres l’accès, s’il l’a modifié, le serveur doit en être informé
* Les serveurs:
  + A la création d’une donnée ils doivent réserver une zone libre dans le segment persistant pour la stocker
  + Doivent alerter les autres serveurs pour mettre à jour la donnée modifiée chez tous les serveurs afin d’être coordonnés
  + Si un client est en cours d’utilisation sur une donnée en mode «read-write», tout appel pour accéder à la donnée, que ce soit pour une lecture simple ou en mode read-write, doit être bloquant jusqu’à celle-ci soit libérée
  + Tant qu’au moins un serveur est en marche les clients ont accès à leurs données ce qui assure la tolérance aux pannes
  + Ils doivent gérer l’accès concurrente en écriture à une donnée, afin qu’elles soient exclusives et synchronisées
  + Lorsque les clients libère une donnée, modifié ou non, le serveur doit débloquer les clients en attente d’accès à la donnée

## 

## Les Conditions d’utilisation

* En condition normale
  + Le client peut lancer une demande de connexion vers le serveur à chaque action , cette demande sera traiter par un processus fils du serveur.
  + Plusieurs clients peuvent récupérer le même fichier en mode lecture.
  + La synchronisation des données entre serveurs doit être faite après tout modification.
* En cas d’échec
  + Il est impossible que plusieurs clients obtiennent le droit d’écriture sur le même fichier.

[Ici schema]

Chaque donnée créée doit avoir un espace alloué dans les N serveurs de manière quasi-instantanée.

Les Solutions

## L’Architecture logicielle choisie

* Le langage de programmation : C

Nous choisissons de coder notre système dans le langage de programmation C, le langage demandé par le client, mais aussi un langage de bas niveau, proche de plusieurs Kernel dont le noyau Linux que nous visons tout particulièrement. En effet, dans notre cas le code vise à être utilisé sur un système Unix. L’application est de type utilitaire, de ce fait il s’est avéré que ce langage est le plus adéquat.

* Outils du langage C

Le langage C nous fournit des primitives de gestion mémoire telles que mmap et mmprotect nous permettant de gérer les droits d'accès aux différentes pages par exemple. Il y a également des bibliothèques pour la manipulation de sockets qui sera le moyen d’échange entre Clients et Serveur(s), et entre Serveurs.

* Les sockets TCP

S’agissant d’échanges de données entre clients et serveurs mais aussi entre les serveurs eux mêmes, leurs bon acheminement, une connexion sécurisée pour la préservation et la protection des serveurs, ainsi que pour l’utilisation du protocole NFS nous semble un choix judicieux et dans la lignée du système persistant.

* Network File System protocol

## La(les) Base(s) technologique(s) employés

A la création d’une donnée les serveurs réservent une zone libre dans le segment persistant, et informent les autres afin que les N serveurs se synchronise sur cette création. Clients, et serveurs communiqueront par sockets interposées. Les clients tout comme les serveurs auront une listes des noms d’hôte des serveurs (non modifiable et lisible que par le code de traitement) pour ne pas être bloqué si le 1e serveur est occupé, ou a crashé, mais pour pouvoir se connecter à un autre serveur disponible immédiatement.

## État de l’art

Le système Ivy de Kai Li 1986

[ Notre solution consiste à développer le dernier schéma cité, une architecture multi-sites. Il s’agit de mettre en place un système de stockage persistant à toutes pannes, pertes des données, corruption ou tout autre soucis par duplication sur N serveurs d’une zone mémoire qui sera mise à jour à chaque modification. Chaque client aura une zone de sa mémoire définie pour être synchronisée avec ces serveurs et chaque serveur auront une fraction de leur mémoire allouée pour répliquer et se synchroniser avec la zone mémoire du client. Les clients conservent cette zone persistante tant qu’au moins un serveur est fonctionnel et en marche. ]

[Le langage C est exigé pas le client pour la mise en place de ce système.

Le délais maximum accordé pour la réalisation de ce projet est de quatre mois. En effet avant le 11 mai 2018 ce dernier devra vous être rendu.

Aucune contrainte n’est formulée quant à la nécessité d’une interface graphique, ou sur la nature de la base de donnée choisie.]