Entreport réparti en mémoire

CLIENTS
SENS Pierre - Pierre.Sens@lip6.fr
KORDON Fabrice - Kordon.Fabrice@lip6.fr

CONCEPTEURS
HE Chenhui - chenhui.he@outlook.fr
MARECAR Farzana- farzana.marecar@outlook.fr
.
1e Année en Master Informatique spécialité SAR
Faculté des Sciences de la Sorbonne Université
Année universitaire 2017/2018

SCIENCES SORBONNE UNIVERSITÉ

Contents

1 Introduction **2**

2 Les Problèmes **3**

2.1 Les Besoins 3

2.2 Les Conditions d'Utilisation 4

3 Les Solutions **5**

3.1 L'Architecture Logicielle choisie 5

3.2 La(les) Base(s) technologique(s) employée(s) 8

3.3 Pour aller encore plus loin 9

3.4 État de l'art 10

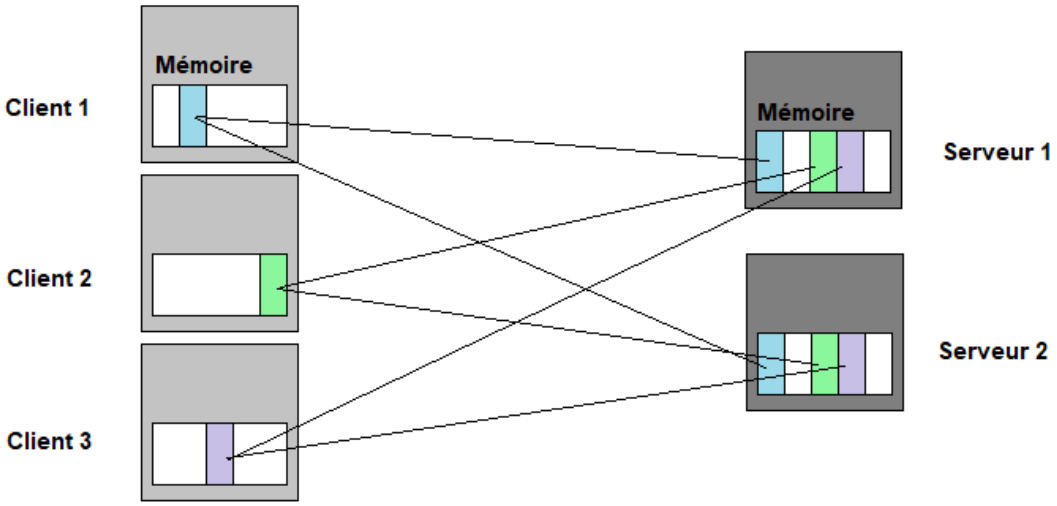

Figure 1: Schéma modélisant le système répartie

SCIENCES SORBONNE UNIVERSITÉ

1 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 [GCK94]
>
> *G. Coulouris, J. Dollimore, and T. Kindberg*

> 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 manipulées par nos clients. Créer surtout un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ses ressources.

SCIENCES SORBONNE UNIVERSITÉ

2 Les Problèmes

2.1 Les Besoins

Le système se composera de deux types d'hôtes:

- Les clients qui doivent pouvoir:
  - se connecter aux serveurs, n'importe lequel des N
  - y créer des données
  - récupérer une copie des données en local depuis ces serveurs
  - accéder à une donnée en lecture et/ou écriture
  - modifier une donnée
  - libérer les données

- Les serveurs qui doivent:
  - allouer une page mémoire pour une nouvelle donnée, chez les N serveurs
  - gérer les requêtes clients
  - mettre à jour une donnée modifiée sur n'importe lequel des N serveurs
  - assurer une cohérence des données
  - assurer une tolérence aux pannes
  - se répartir la charge

SCIENCES SORBONNE UNIVERSITÉ

2.2 Les Conditions d'Utilisation

Il faut distinguer deux conditions d'exécution:

- Les conditions optimales d'exécution:
  - Les clients peuvent se connecter à n'importe lequel des N serveurs à tout moment
  - Chaque serveur peut gérer plusieurs clients simultanément
  - A la création d'une donnée ou à sa modification, les N serveurs mettrons à jour simultanément la donnée dans leur segment mémoire
  - Les clients pourront récupérer une donnée de n'importe lequel des N serveurs
  - La lecture et/ou l'écriture d'une donnée doit pouvoir se faire en local pour chaque client
  - La donnée manipulée par le client est bien la dernière version à jour
  - Plusieurs clients peuvent récupérer le même fichier en mode lecture
  - A la fin de leur utilisation les clients relâchent la donnée

- Et les cas d'erreurs à prévenir:
  - Même si N-1 serveurs ne fonctionnent plus, le dernier doit avoir les données à jour et être disponible
  - Si un client accède à une donnée en lecture/écriture, les autres doivent attendre que la donnée soit mise à jour avant de pouvoir la manipuler
  - Si une donnée est accédée en lecture/écriture sur un serveur S1 par un client C1, elle ne sera pas accessible sur les N-1 autres serveurs pour tout autre client
  - Les clients ne peuvent pas accéder à une donnée inexistante, si le nom de la donnée indiquée est erronée et n'existe pas dans la table d'informations sur les pages que contiennent tous les serveurs, une message d'erreur sera renvoyé
  - Si le client récupère une donnée après l'avoir demande en lecture simple, il ne lui sera pas possible de la modifier

SCIENCES SORBONNE UNIVERSITÉ

3 Les Solutions

3.1 L'Architecture Logicielle choisie

1. Persistance des données

   (a) Cohérence entre Serveurs

   A l'initialisation du système, les N Serveurs découperont leur segment mémoire en nombre égale de pages mémoires. Chaque page permettra d'y stocker une donnée, la donnée aura donc pour taille maximale la taille d'une page, soit 4096 octets. Chaque donnée sera associée à un nom, il faudra donc sur chaque serveur stocker les informations de chaque page: une table qui contiendra l'adresse de début d'une page, le nom associée à la donnée contenue dans cette page, et un ou plusieurs flags indiquants si la donnée a été demandée en lecture et/ou en écriture.
   Notre système se doit d'avoir une cohérence entre les N serveurs; pour cela, chaque serveur qui réserve une zone libre dans le segment persistant pour pouvoir stocker une nouvelle données préviendra les autres serveurs en précisant le nom de la donnée avant de confirmer au client la création réussite de la nouvelle donnée. On suivra le même procédée lors des accès aux données, ou lors de leur libération.

   (b) Entre Clients et Serveurs Les Clients auront N serveurs répliqués à

   leur disposition, N étant un nombre assez grand pour l'importance des données de notre clientèle. A chaque modification de la donnée par le Client, l'information sera retransmise vers le Serveur qui traite la requête et ce dernier mettra à jour automatiquement la donnée avec les modifications apportées. Non seulement ce serveur sera à jour mais il préviendra également l'ensembles des serveurs pour qu'ils mettent leur table à jour. Ainsi les données seront mis à jour automatiquement.

2. Gestion des accès

   (a) Les droits d'accès

   Les serveurs ont tous les droits sur les pages de leur segment mémoire, les clients n'ont aucun droit sur ces zones. Lorsque les clients veulent manipuler une donnée, il doivent émettre une requête au serveur, ce dernier enverra une copie de la page mémoire. Le Serveur attribuera si nécessaire des droits en lecture et éventuellement en écriture à la donnée pour le client, pour qu'il ne puisse pas par exemple écriture lorsqu'il a demandé la donnée en lecture simple. Ainsi le serveur gère à distance la manupilation de cette donnée.
   Une fois la manipulation de la donnée terminée, le client indiquera au serveur qu'il libère la donnée, cette information tout comme l'accès auront été communiqué à tous les serveurs en marche. A ce moment, ce client perd de nouveau tous ces droits.

   (b) Gestion des accès conccurents critiques

   Comme mentionné à multiples reprises, lorsque un client1 accède en lecture/écriture, tout autre client voulant accéder à cette même donnée ne pourra pas, car dans l'immédiat la donnée qu'il voudra récupérer ne sera celle à jour tant que client1 n'a pas soumis sa modification. Ainsi, tout client voulant accéder à une donnée en lecture écriture sera mis en attente jusqu'à ce que le client1 libère la donné. Parmi les clients qui souhaitent accéder à la donnée, le premier dans la queue si possible obtiendra les droits d'accès. C'est par ce mécanisme que les serveurs gère l'accès aux données. On fait le choix de laisser plusieurs clients lire de manière conccurente la même donnée car cela ne compromet en rien l'intégrité de la donnée, mais si une personne veux modifier cette donnée elle devra aussi patienter que tous relâchent la donnée pour l'obtenir.

3. Gestion des pannes

   (a) Détection par Clients et Tolérence Les Clients auront une table des hôtes Serveurs associés aux ports sur lesquels les contacter (non modifiable et lisible que par le code de traitement) pour ne pas être bloqué sur un serveur occupé, ou qui a crashé, mais pour pouvoir se connecter à un autre serveur disponible immédiatement. Etant donnée que tous les serveurs en marchent sont supposés avoir la dernière version à jour de la donnée, le client pourra retrouver sa donnée sur n'importe lequel de ces N serveurs tant qu'au moins un serveur est en mache. Si un serveur est détecté non fonctionnel par le client, il mettre à jour sa table d'hôtes en ajoutant un indicateur pour ne plus tenter de se connecter à ce dernier.

   (b) Détection entre Serveurs et Tolérence

   Les Serveurs auront eux aussi une table des hôtes serveurs, et le port sur lequel les contacter pour la gestion de la cohérence systèmes différente du port pour la gestion des requêtes clients.
   Si un serveur tombe en panne, cela sera détecté par une non réception d'acquittement après l'une des requêtes de gestion de cohérence et la table des hôtes serveurs sera mise à jour, une flag informera le dysfonctionnement du serveur en question ici aussi.
   Et si tous les N serveurs tombent en panne ? Qu'advient-il des données désormais inaccessibles ? Cela est encore une autre question, une ouverture vers un projet plus avancé. Le projet courant proposera une nombre important de serveur justement pour prévenir ces pannes serveurs, la propabilité que les N tombent en panne sous 20 ans par expemple est très faible, surtout avec un bon entretien. Si on veut tout de même prévenir ce cas, on pourrait ajouter un signalement par les serveurs lorsqu'ils auront dans leur table d'hôtes serveurs au plus 5 serveurs en marche pour laisser le temps aux administrateurs de déployer X autres serveurs qui se synchroniseront au système et solidifier le système persistant et le faire perdurer dans le temps.

3.2 La(les) Base(s) technologique(s) employée(s)

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 ou encore memcpy nous permettant de gérer les droits d'accès aux différentes pages par exemple, leur "division" et leur copie pour l'envoie aux clients. 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.

3.3 Pour aller encore plus loin

L'idée de tables d'hôtes serveurs pour la répartition des charges et la tolérence aux pannes et bien mais cela nécessite d'apporter une attention particulière à ces tables afin de prévenir des usages malveillants et autres risques de détruire notre système.
Une autre façon de faire serait de mettre entre les clients et les serveurs répliqués un serveur Kerberos plus résistant vers qui se dirigeraient toutes les connexions clients, ils n'auront plus qu'à connaître un seul port de cette façon. Ce Kerberos ressencera une table des serveurs, marquera ceux disponibles et fonctionnels, ceux qui ne répondent plus par échanges de ping par exemple. Il redirigera à la connection les clients vers un de ces serveurs et se chargera de la répartition des charges.
Il est probable que ce Kerberos s'avère être un goulot d'étranglement, un point de faille unique, que à cause de lui tout le système risque de ne plus marcher. A nous de prendre les précautions pour prévenir tous les échecs.

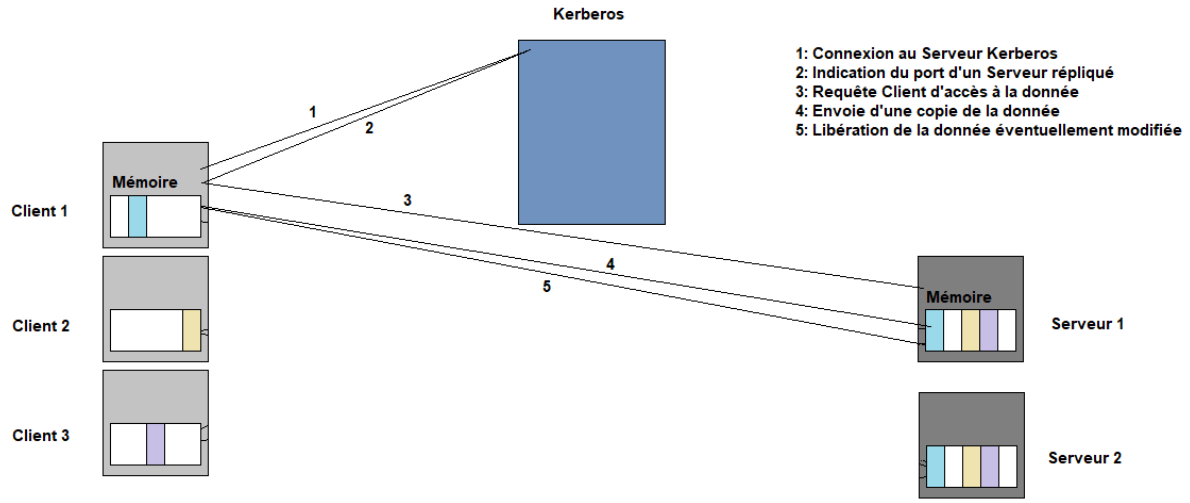

Figure 2: Schéma modélisant le système avec un serveur central Kerberos

3.4 État de l'art

Parmi les projets qui nous on servi d'inspiration, le système Ivy de Kai Li [Han], pionnier en matière de la mémoire partagée répartie, conçu en 1986, se rapproche de l'idée que nous souhaitons reproduire .
Il repose sur l'unité de partage et de distribution par défaut, la page. A chaque page mémoire est associée un unique hôtes propriétaire, seul à avoir les droits de lecture et d'écriture sur le fichier, tout autre hôte n'a que des droits de lectures sur le fichier. Pour lire cette page, les autres hôtes doivent demander au propriétaire et charger depuis celui-ci.
Si entre temps une modification est apporté par le propritaire sur la donnée, les autres copies conservés chez les autres hôtes sont invalidés, ils pourronts demander à la recharger en local.

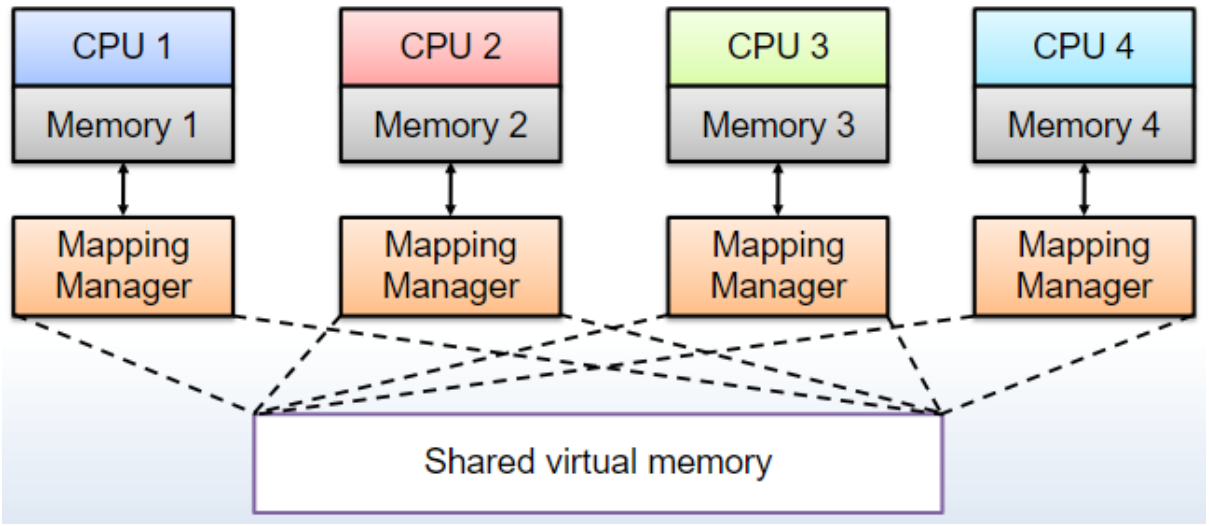

Figure 3: Schéma modélisant le système Ivy [KL09]

SCIENCES SORBONNE UNIVERSITÉ

References

[GCK94] J. Dollimore G. Coulouris and T. Kindberg. *Distributed Systems - Concepts and Design*. Addison-Wesley Publishers Ltd., 2nd ed. edition, 1994.

[Han] Jay Han. Conception et réalisation d'une mémoire partagée répartie. Technical report, https://tel.archives-ouvertes.fr/tel-00004993/document.

[KL09] Paul Hudak Kai Li. Memory coherence in shared virtual memory system. http://slideplayer.com/slide/9278086/, 2009.