Département d'informatique

Master Informatique - UE MIF32 Parallélisme

TP Projet 2015/2016

Laurent Lefèvre (*Laurent.Lefevre@ens-lyon.fr*)
Daniel Balouek-Thomert (*Daniel.Balouek@ens-lyon.fr*)
Hélène Coullon (*Helene.Coullon@ens-lyon.fr*)

Introduction

Le protocole **CAN (Content Addressable Network)** décrit une architecture pair à pair distribuée et décentralisée, qui fournit des mécanismes de hachage à une échelle comparable à celle de l'Internet.
CAN est une des propositions originales de tables de hachage distribuées, en parallèle des protocoles Chord, Pastry et Tapestry.

CAN se base sur une table de hachage distribuée (DHT). Une DHT permet de stocker chaque donnée en un point unique du réseau, défini à partir de l'identifiant de la donnée. Un de ses buts principaux est d'assurer qu'une information est retrouvée, indépendamment de l'infrastructure dynamique du réseau (ex: Ajout/Suppression de noeuds).

CAN est caractérisé par une division de l'espace en coordonnées cartésiennes. Dans ce contexte, chaque noeud se voit attribuer un point unique dans l'espace, ce qui détermine ensuite sa position par rapport aux autres noeuds. Chaque donnée est elle aussi associée à un point unique grâce à une fonction de hachage ; on peut ainsi créer de manière très simple un lien entre une donnée et le noeud sur lequel elle doit être stockée.

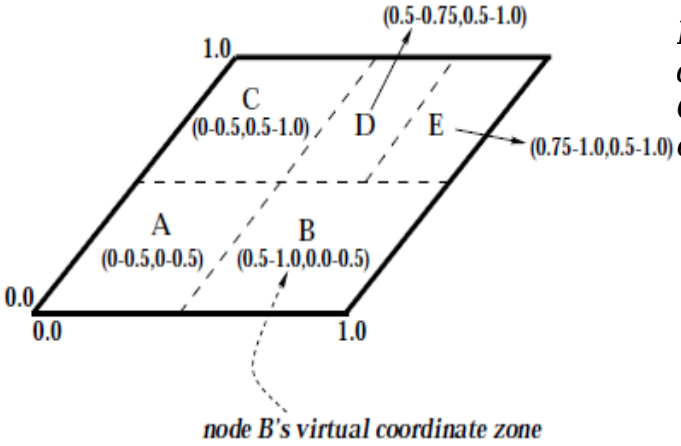

Figure 1: Un exemple de division 2D de l'espace cartésien entre 5 noeuds. Chaque noeud est associé à une zone et détient les données de cette zone.

Objectif

Le but de ce projet est de simuler une implémentation bi-dimensionnelle de CAN en MPI. L'espace des coordonnées est le sous-ensemble des entiers naturels [0, 1000] x [0, 1000].
Votre implémentation devra permettre les opérations suivantes :

- Insertion d'un noeud
- Insertion d'une donnée
- Localisation d'une donnée
- Supression d'un noeud

Travail à réaliser

Dans l'ensemble de ce projet, chaque processus représentera un noeud et le rang du processus représentera l'adresse du noeud. Le processus de rang 0 fait exception : il ne participe pas à l'overlay mais sert de coordinateur. Son rôle sera dans un premier temps de synchroniser les opérations d'insertion des noeuds, puis d'agir en tant que client pour insérer/rechercher des données.
Le réseau overlay (noeuds interconnectés, chacun d'entre eux étant associés à une zone de l'espace cartésien) contiendra N-1 noeuds en tout. Le protocole de routage et les structures de données conservées localement sur chaque noeud sont laissées à l'appréciation des étudiants concepteurs.

Chacune des étapes doit donner lieu à la création d'un fichier texte de log contenant 3 colonnes :

- Identifiants des processus
- Coordonnées de la zone dont il a la charge
- Données présentes dans la zone

Etape 1 : Création du réseau et insertion de nœud

Le réseau overlay devra être crée à partir d'un ensemble de processus MPI, crées lors de l'exécution du programme.
Chaque processus tire aléatoirement un identifiant : un couple de valeurs dans l'espace des coordonnées. Le coordinateur contacte un premier processus (celui de rang 1) pour l'inviter à s'insérer dans l'overlay. Chaque noeud une fois inséré notifie le coordinateur, qui invite alors le noeud suivant dans l'overlay.
Le premier processus à s'insérer (rang 1) hérite de l'ensemble de l'espace et devient également le noeud de bootstrap par lequel tous les autres noeuds passeront pour s'insérer.
Chaque noeud invité d'identifiant I émet une requête d'insertion de noeud qui est routée jusqu'au noeud R responsable de l'espace contenant les coordonnées de I. Le partage de l'espace des coordonnées entre I et R se fait alors de la manière suivante.

1. On divise l'espace de R en deux. La coupe se fait en largeur si la largeur est plus grande que la hauteur, et inversement. En cas d'égalité, la coupe se fait en largeur. On constitue deux sous-espaces issus de cette division.
2. R conserve la responsabilité du sous-espace qui contient son identifiant.
3. I acquiert la responsabilité de l'autre sous-espace. Si l'identifiant I n'appartient pas à ce sous-espace, I tire aléatoirement un nouvel identifiant dans ce sous-espace.

Tips : L'utilisation des primitives de broadcast est recommandée. Les rangs MPI des voisins doivent être stockés.

Etape 2 : Insertion de données

Le coordinateur tire aléatoirement N*10 coordonnées (a, b), et la valeur de la donnée à insérer est calculée par addition de a et b.
Le couple clé-valeur inséré dans la DHT est donc : ((a, b), (a+b)).
Le coordinateur insère ces données une par une, en émettant une requête d'insertion de donnée auprès d'un processus de rang aléatoire et en attendant un acquittement du noeud qui stocke la donnée avant de demander une nouvelle insertion.
Le coordinateur conserve en mémoire les coordonnées des 5 premières insertions et celles des 5 dernières insertions.

Tips : Une diffusion de proche en proche (voisin) est recommandée.

Etape 3 : Recherche et lecture de données

a) Lorsque toutes les données ont été insérées, le coordinateur va émettre une par une des requêtes de recherche de données associées aux 10 coordonnées qu'il a conservées en mémoire.

Comparer les performances de recherche en utilisant :
- une recherche via les primitives de broadcast
- une recherche aléatoire puis de proche en proche

Attention : Une donnée (a+b) peut avoir deux clés différentes (a,b).

b) Le coordinateur va également initier 4 recherches pour des clés distinctes tirées au hasard. Une recherche via un nœud aléatoire puis de proche en proche est recommandée.

Etape 4 : Suppression de noeud

Le coordinateur tire aléatoirement une clé. Le noeud responsable de l'espace contenant cette clé doit alors être retiré de l'overlay. Lorsque cette suppression est effective, le coordinateur lance une recherche de données associée à la même clé pour vérifier qu'un des noeuds restants a bien endossé la responsabilité de l'espace.

Rendu

Pour ce projet, vous êtes invités à travailler en binôme. Votre travail devra être fonctionnel et rendu, sous la forme d'une archive contenant votre code et un mini-rapport, par mail à l'ensemble des membres de l'équipe enseignante en respectant l'objet de mail suivant.

[MIF32] Rendu de projet CAN Nom1-Nom2

La date limite de rendu est fixée au 21 mai 2016 à 23h59.
Tout projet reçu après cette date sera pénalisé.

L'archive devra contenir:

- L'ensemble des fichiers de code
- Un makefile permettant l'exécution de votre application
- Un mini-rapport (max. 2 pages) au format PDF expliquant les choix de conception, les algorithmes et les structures de données mises en oeuvre.

Les 4 étapes décrites ci-dessus sont à implémenter dans l'ordre.
Notez qu’un travail incomplet mais fonctionnel aura une bien meilleure
note qu’un travail plus “complet” mais qui ne compile pas ou ne fonctionne que rarement.
La propreté du code et la présence de commentaires sont également nécessaires.

Références

Ratnasamy, Sylvia, et al. "A Scalable Content-Addressable Network." (2001).

http://en.wikipedia.org/wiki/Content_addressable_network