Projet Données Réparties

Architecture de principe

Paul Anaclet

Guohao Dai

Théo Desprats

2A SN

2021

1 Plan de travail

Nous avons mis au point le plan de réalisation suivant :

Tâche	Membre(s) assigné(s)	Commentaires
Version centralisée		
- Implémentation Serveur	Theo, Paul	Primitives Linda
- Synchronisation des accès	Paul, Guohao	
Version client/serveur		
- Implémentation Serveur	Theo, Paul	Interface, implémentation, enregistrement du serveur ...
- Implémentation Client	Theo, Guohao	
Mise au point de nouveaux tests	Guohao	
Applications		
- Nombres premiers < K	Theo, Guohao	
- Recherche dans un fichier	Paul	

2 Architecture

2.1 Classes et structures de données envisagées

Stockage des callbacks:

- `ReadCallbacks = Map<Tuple, List<Callback>>` : map associant la liste des callbacks des clients en mode READ au motif qui les intéresse.

- `TakeCallbacks = Map<Tuple, List<Callback>>` : map associant la liste des callbacks des clients en mode TAKE au motif qui les intéresse.

Version Client/Serveur :

- `interface LindaServer extends Remote` : interface du serveur

- `class LindaServerImpl extends UnicastRemoteObject implements LindaServer`: objet distant utilisable par le client pour appeler les procédures du serveur Linda

2.2 Difficultés identifées

Certains points de conceptions à propos de l'implémentation de Linda semblent plus complexes :

- gestion des abonnements : correctement informer les 'abonnés' à chaque primitive 'write'

- gestion de la synchronisation : utilisation d'une méthode adaptée pour rendre les primitives qui en ont besoin bloquantes (moniteur ? Sémaphores ?...)

- choix arbitraires : faire des choix à la fois optimaux et peu complexes à mettre en place pour les blocages / déblocages (priorités, …)

Pour les applications nous nous sommes posés plusieurs questions dont les réponses ne sont pas encore claires :

- utilisation de Linda pour le crible d'Erathostène : comment utiliser et manipuler les tuples pour les calculs ? Quel est l'avantage ?

- critère de parallélisation du crible d'Erathostène : quelle partie de l'algorithme pouvons nous paralléliser de manière efficace ?

2.3 Organisation des tests

Il faudra dans un premier temps réaliser des tests permettant de confirmer nos spécifications arbitraires.

Les tests devront couvrir une grande variété d'enchaînements de primitives.