Architecture de principe

Fainsin Laurent - 2SN M2 \ Guillotin Damien - 2SN M2

Consignes

- On peut raisonnablement attendre que vous ayez au minimum une première version centralisée (sans eventRegister) qui fonctionne, avec un plan de test complet pour cette version et les suivantes
- Il serait plutôt normal que vous ayez une version centralisée complète (avec EventRegister) qui soit opérationnelle sur des scénarios/tests de base.
- Il est envisageable d'avoir une ébauche/une réflexion un peu avancée sur la version client-serveur
- Un rapport provisoire succinct (environ 2 pages) présentant l'architecture, les algorithmes des opérations essentielles, une explication claire des points délicats et de leur résolution envisagée (ou des blocages rencontrés)
- Le code complet de la partie réalisée.

Plan de travail inital

Version en mémoire partagée

Création de l'espace partagé de données typées Linda centralisé. Il faut dans un premier temps implémenter un jeu de primitives spécifiques (les méthodes de l'interface). L'ensemble des primitives `write`, `take`, `tryTake` et `takeAll`, `read`, `tryRead` et `readAll` et `eventRegister` ont été réalisées.

Version client / mono-serveur

Cette version ne diffère pas beaucoup de la précédente, la seule différence se situe dans l'implémentation de l'interface RMI. Ainsi nous avons créé l'interface `LindaRemote`, reprenant les méthodes de `Linda`. De même nous avons créé la classe `LindaServer` implémentant `LindaRemote` et dont le but est de publier dans un registre RMI une instance `CentralizedLinda`. `LindaClient` vient simplement chercher une instance `Linda` dans le registre RMI et passe l'ensemble de ses actions à cette instance. Nous n'avons écrits que quelques tests dans les `.java`, nous avons pour l'instant effectué la pluspart d'entre eux "en live". En effet, la version test du Client-Server permet d'écrit/lire avec une "invite de commande" dans le répertoire des tuples du serveur.

Application Eratosthène

Cette application est constituée d'un serveur responsable de distribuer les tuples à tester, et de clients responsables d'effectuer des calculs sur ces tuples. Ainsi `Server` instancie un `LindaServer` de la partie précédente et initialise le tuple-space avec des tuples formés d'un entier (l'entier dont ont veut tester la primalité) et de deux strings qui indiquent l'état de test de cet entier. Chaque `Client` se contente simplement de se connecter au serveur et de vérifier la primalité d'un tuple, une fois son calcul terminé, il place le résultat dans le tuple-space. Cette dynamique persiste tant qu'il existe encore un entier non testé dans le tuple-space.

Application Levenshtein

Cette application fonctionne directement grâce au code `Server` et `Client` écrit précédemment, mais aucune des améliorations demandés dans le sujets n'ont encore été rédigées.

Application Whiteboard

Nous n'avons pas encore avancé sur cette partie.