UE INF4003 : TP 8
Solution au problème des philosophes avec JavaSpaces

Frédéric Raimbault

Rappel : ce TP doit être réalisé avant la scéance ; au début de la scéance, un travail complémentaire vous sera remis et vous serez évalué sur la réalisation de l'ensemble à la fin de la séance.

Attention : la mémoire JavaSpaces disponible sur le *cluster-pédago* est partagée entre tous les utilisateurs du cluster et elle est permanente. Les entrées que vous y déposerez seront accessibles d'une exécution à l'autre. Pour éviter de polluer la mémoire JavaSpaces vous veillerez donc à :

- créer une hiérarchie de packages Java qui porte votre nom de login (variable bash `LOGNAME`) pour éviter de mélanger vos classes avec celles des autres utilisateurs ; n'oubliez pas de modifier les sources fournis !
- utiliser des entrées qui sont une sous-classe de `MyEntry` pour éviter de mélanger vos objets avec ceux des autres utilisateurs,
- préciser un temps fini de conservation des entrées lors de leur dépot : paramètre `lease` mis à la valeur `Lease.ANY` ou un nombre de millisecondes raisonnable,
- vérifier régulièrement vos entrées présentes dans la JavaSpaces et les supprimer le cas échéant avec les scripts `jslist` et `jsclean`.

Le problème du dîner des philosophes

Soient N philosophes qui passent leur temps soit à réfléchir soit à manger. Chaque philosophe i est assis à la place i autour d'une table circulaire. Dans la figure 1, chaque place est représentée par une assiette. Il y a un grand bol de riz au centre de la table. Un philosophe doit avoir deux baguettes pour manger. Cependant, on dispose seulement de N fourchettes, réparties entre chaque assiette. Un philosophe a besoin de prendre la baguette située immédiatement à sa droite et celle située immédiatement à sa gauche pour manger. Les philosophes ont exactement le même comportement cyclique : ils pensent, ils attendent de pouvoir saisir leur deux baguettes, puis ils mangent, et de nouveau puis ils pensent, ... Le problème consiste à proposer un ordonnancement qui assure que tous les philosophes peuvent manger, plusieurs à la fois, sans être bloqués.

La solution de Dijsktra

La solution proposée par Dijsktra est d'utiliser :

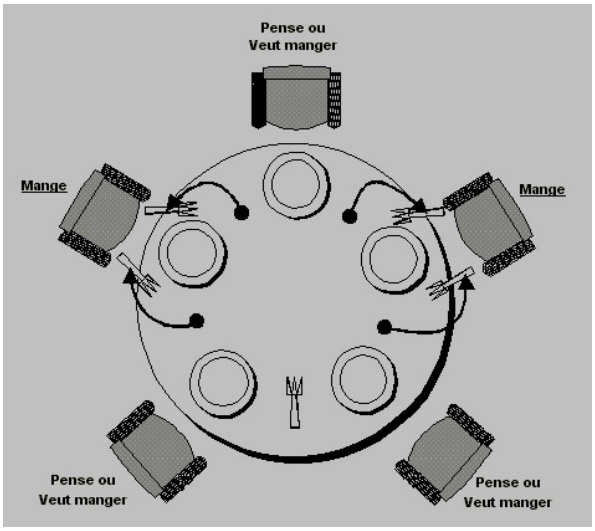

Fig. 1: illustration du dîner des philosophes

— un sémaphore sur chaque baguette pour s'assurer qu'un seul philosophe peut s'en emparer.
— un sémaphore sur l'accès aux assiettes permettant seulement à $N - 1$ philosophes de manger simultanément : on évite ainsi une situation d'interblocage où tous les philosophes ont une baguette et attendent l'autre.

Avant de tenter de prendre une baguette le philosophe doit obtenir l'accès à une assiette. Une fois qu'il a fini de manger il libère ses baguettes et son accès à l'assiette.

Travail à réaliser (première partie)

1. Familiarisez-vous avec JavaSpaces et son utilisation sur le cluster en testant le programme d'exemple (Producer/Consummer) vu en CM (sources et lanceur fournis sur l'ENT).
2. Familiarisez-vous avec les classes fournies (classes `SpaceAccessor`, `MyEntry` et `Philosopher` du package `philosophers_V0`) pour la mise en oeuvre avec JavaSpaces de la solution de Dijsktra au problème des philosophes.

   Les questions suivantes ont pour but d'écrire les autres classes ; vous trouverez leur documentation *javadoc* sur l'ENT.
3. Écrivez une mise en œuvre de compteur distribué (classe `Counter`).
4. Écrivez une mise en œuvre de sémaphore distribué (classes `SemaphoreEntry` et `SemaphoreAccessor`).
5. Testez le programme complet `Philosophers` en étudiant le temps cumulé par les philosophes dans chacune de leur action.

La solution développée jusqu'à présent souffre de l'absence de garantie d'équité pour l'accès à une assiette.

Accès par ticket numéroté à la table

Pour garantir que chaque philosophe pourra accéder à une assiette de manière équitable on met en place un système de ticket numéroté qui évolue de manière cyclique :

— chaque philosophe possède un numéro d'accès unique à l'assiette,
— un philosophe ne peut accéder à une assiette que si c'est à son tour de le faire,
— une fois qu'il a accès à une assiette il incrémente et remet le ticket d'accès pour le philosophe suivant,
— le tour revient au premier philosophe après le tour du dernier philosophe.

Travail à réaliser (seconde partie)

1. Faîtes une copie de votre package `philosophers_V0` sous le nom `philosophers`.
2. Ecrivez une mise en oeuvre de compteur distribué qui évolue de manière cyclique (classe `CyclicCounter`) en étendant la classe `Counter`.
3. Ecrivez une classe Java (classe `CyclicCounterAccessor`) qui serve d'interface aux philosophes pour accéder à un compteur distribué.
4. Reprenez la solution au problème du dîner des philosophes (classe `Philosophers`) en intégrant le compteur distribué cyclique : avant d'accéder à une assiette il attend son tour. Une fois qu'il a obtenu l'accès à une assiette (et avant qu'il ne cherche à accéder aux baguettes) le tour passe au philosophe suivant.
5. Testez votre nouvelle solution au problème du dîner des philosophes sur le cluster et vérifiez les temps cumulés des actions des philosophes.