Ecole Polytechnique

Cours : "Parallélisme"
Travaux dirigés
E. Goubault, S. Putot & O. Bouissou

TD 5

8 mars 2006

Dans les exercices suivants, on va distribuer des calculs. Afin de ne pas trop "polluer" le réseau et les machines de toutes les salles, vous ne démarrerez des programmes et des serveurs que sur votre machine, la machine immédiatement à gauche de la votre (ou à droite selon), et immédiatement derrière vous (ou devant, selon). Chacun lancera `rmiregistry` en indiquant un numéro de port, selon *sa place* dans la salle TD, comme indiqué à la figure 1.

Vous ferez bien attention à démarrer `rmiregistry` une seule fois, si possible dans le répertoire contenant vos classes JAVA (sinon faites attention à votre `CLASSPATH`!), et de le quitter en fin de TD (en faisant `fg` et Ctrl-C par exemple). Autre solution: vous ne demarrez le registre de services que dans votre code serveur.

Pour connaitre votre `CLASSPATH`:

```
echo $CLASSPATH
```

Faites attention à ce qu'il contienne `.` ainsi que le répertoire dans lequel vous avez vos `.class`. Si vous désirez le modifier, faites par exemples, en instanciant correctement `USER` etc.:

```
setenv CLASSPATH .:/home/USER/TD5/:${CLASSPATH}
```

(en `csh`, `tcsh` etc. - faites `echo $SHELL` pour savoir)

```
export CLASSPATH=.:/home/USER/TD5/:${CLASSPATH}
```

(en `bash`)

Enfin, rajoutez à votre `PATH` le chemin `/usr/java/jdk1.5.0_06/bin/`.

Pour tester tout cela, récupérez les codes vus en cours;

`HelloInterface.java`, `Hello.java`, `HelloServer.java` et `HelloClient.java`

sur la page web du cours:

`http://www.enseignement.polytechnique.fr/profs/informatique/Eric.Goubault/ParaI06.html`

1 Calcul de π

Supposons que nous voulions calculer π en parallèle par la formule suivante,

$$\begin{aligned} \pi &= \int_0^1 \frac{4}{1+x^2}dx \\ &\cong \sum_{i=1}^{n} \frac{1}{n} \frac{4}{1+\left((i-\frac{1}{2})\frac{1}{n}\right)^2} \end{aligned}$$

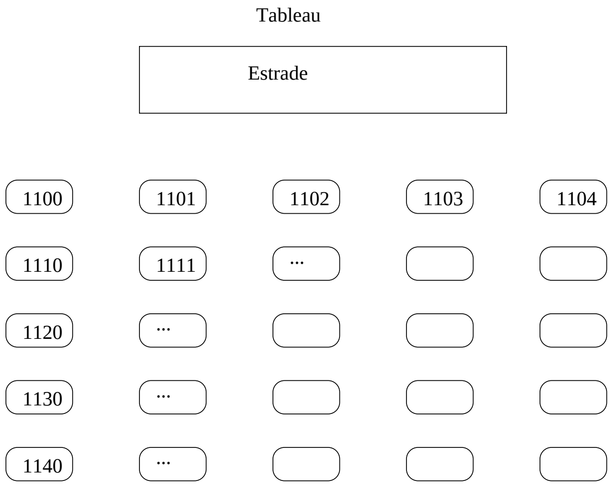


Figure 1: Répartition des numéros de port en salle TD.

Une solution est le paradigme **Maître/Esclave**: Un maître va lancer `N` esclaves chargés de calculer les sommes partielles,

$$P_k \quad = \quad \sum_{i=k*n/N+1}^{(k+1)*n/N} \frac{1}{n} \frac{4}{1+\left((i-\frac{1}{2})\frac{1}{n}\right)^2}$$

pour $k = 0, \cdots, \texttt{N-1}$.

Implémenter le calcul de π en le distribuant réellement (par RMI):

- On pourra se contenter dans un premier temps d'imiter ce qui est fait dans `Hello` et faire des appels bloquants
- Ensuite on pourra soit implémenter un client/serveur non bloquant, soit utiliser les threads JAVA pour rendre moins bloquants les appels aux esclaves (les serveurs).

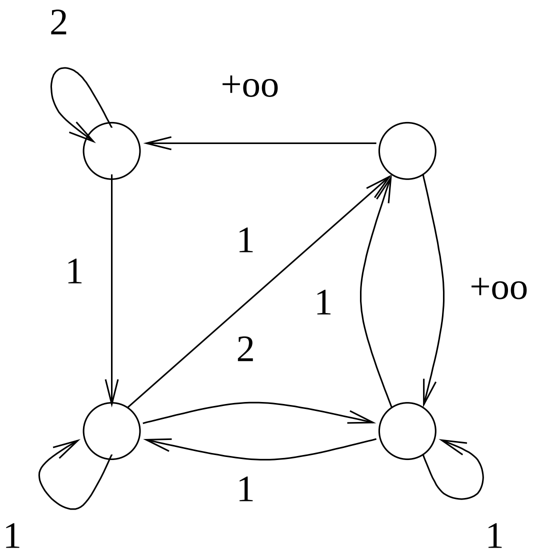
2 Points morts

(1) On considère un système constitué d'un 4-sémaphore x auquel accèdent 3 processus PV en parallèle, chacun d'entre eux n'ayant jamais besoin de plus de deux verrous sur x à tout instant. Montrer qu'il ne peut pas y avoir d'interblocage.

(2) On généralise la question 1: on considère maintenant un m-sémaphore x, auxquels accèdent n processus. On supposera que chacun des processus a besoin à tout instant de 1 à m verrous sur x et que la somme des besoins de chacun des processus est inférieur strictement à $m+n$. Montrer que le système est sans interblocage.

3 Dépendances de données

Donner un nid de boucle parfait où toutes les dépendances sont uniformes, et dont le GDRN est exactement le graphe suivant:



Exécuter l'algorithme d'Allen et Kennedy sur ce nid de boucles.