Ecole Polytechnique

Cours : "Parallélisme"
Travaux dirigés
E. Goubault & M. Martel

E. Goubault & M. Martel

TD 5

5 mars 2003

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).

1 Calcul de π

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

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

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

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

pour $k = 0, \cdots,$ `N-1`.

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

2 Méthode aux différences finies

Implémenter le code de l'équation de la chaleur en RMI. En séquentiel, cela revient à calculer pour un certain nombre d'itérations la formule suivante, sur une matrice `a` représentant la valeur d'une température sur un plan (discrétisé):

```
for (i=0;i<=N-1;i++)
  for (j=0;j<=N-1;j++)
    a[i][j]=1/4*(a[i-1][j]+a[i+1][j]+a[i][j-1]+a[i][j+1]);
```

On pourra commencer par en faire une implémentation utilisant les threads JAVA, ensuite on essaiera de la transformer avec RMI. Afin de simplifier l'écriture du code, on supposera que chaque processus s'occupera d'une portion égale de `a` de la forme `a[0,...,r-1][0,...,N-1]` et que donc,

- Le processus 0 aura un seul voisin ("à droite") avec lequel communiquer
- Le processus $N-1$ aura un seul voisin ("à gauche") avec lequel communiquer
- Les processus `k` avec `0<k<N-1` ont deux voisins avec lesquels communiquer

On supposera que les "conditions aux limites" c'est à dire les éléments de la forme `a[0][0,...,N-1]`, `a[N-1][0,...,N-1]`, `a[0,...,N-1][0]` et `a[0,...,N-1][N-1]`, ont une température imposée, ici `100`, alors qu'à l'instant initial, la valeur de la température à l'intérieur vaut `0`.

On supposera pour simplifier que le nombre total de processus divise `N`, et que `N` est divisible par `2`. On pourra par exemple afficher la température maximale *à l'intérieur* de chaque domaine (pour chaque processus "esclave").

Qu'est-ce que chaque processus doit avoir comme information à sa création? Quelles seront les synchronisations entre processus nécéssaires à la bonne résolution de l'équation de la chaleur?

En RMI, on pourra considérer chaque processus comme étant un serveur d'une itération en temps pour l'un des sous-domaines, étant donné les valeurs aux bords de ce sous-domaine. Le client sera chargé de demander à chacun des processus une itération, et ainsi de suite. En threads JAVA, on pourra soit programmer une version "mémoire partagée", soit une version (plus facile à transformer en RMI) qui fasse explicitement des "envois" et des "réceptions" de tranches de tableaux.

3 Simulation de processeurs en anneau

Implémenter en RMI un objet permettant de passer un message d'une machine à une autre, de telle façon à simuler les opérations de passage de message sur un réseau en anneau vues en cours. On appelera un serveur sur chaque noeud du réseau en anneau que l'on voudra simuler. A la ligne de commande, on donnera au serveur le nom de la machine successeur, et le nom de la machine prédécesseur. Chaque objet créé par ce serveur pourra exécuter les méthodes suivantes:

- `void send(int x);`
- `int recv();` comme dans le cours sur les réseaux en anneau
- `void store(int x);`: stocke la valeur `x` dans l'entier local (privé) à l'objet distant
- `String print();`, qui imprime la valeur de l'entier local à l'objet distant

Une façon de programmer cela est de considérer chaque objet distant s'exécutant sur un noeud comme étant un client pour le serveur du noeud suivant, et un serveur pour l'objet représentant son noeud prédécesseur.