T.P. Parallélisme

Craquage de mots de passe fictifs par force brute

Réalisé par

HABASSI Seif Eddine

EL AFRIT Mohamed Amine

TP PVM
ENSEIRB Département Informatique Option PRCD
2008/2009

I. Craquage de mots de passe fictifs par force brute

On veut écrire un programme PVM qui craque un mot de passe par force brute, selon une organisation en ferme de calcul.

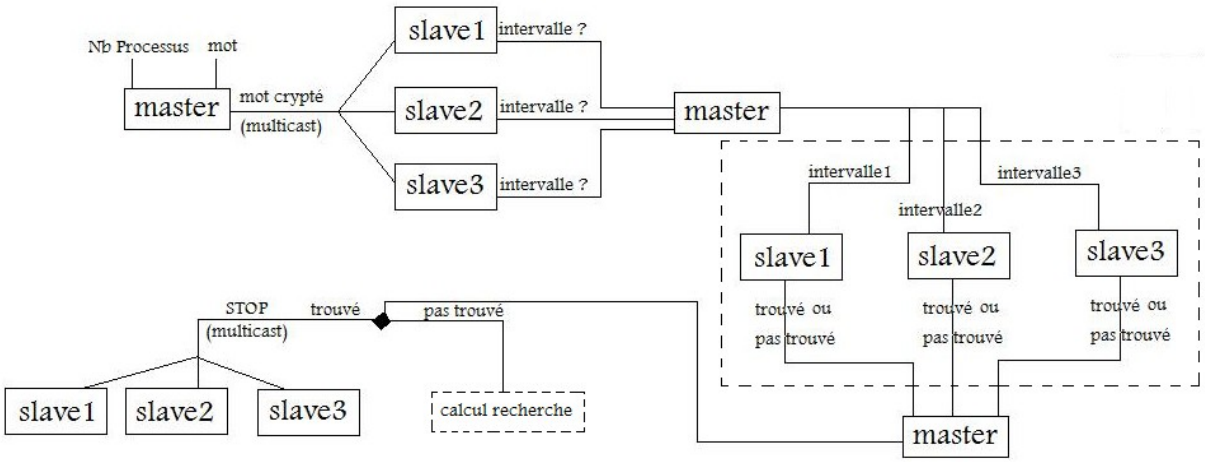
Un processus maître se charge de distribuer des intervalles de mots à tester, de tailles comparables, à chacun des processus esclaves, qui répondent au processus maître soit en indiquant le mot correspondant au mot de passe crypté s'il a été trouvé dans l'intervalle, soit en demandant un nouvel intervalle à tester.

L'utilisateur spécifiera le nombre de processus à lancer, puis le programme s'exécutera jusqu'à ce que le mot de passe ait été trouvé.

II. Travail réalisé

1.Algorithme principal

L'algorithme est expliqué par la figure suivante :



1- Le master envoi le mot de passe crypté en multicast à tout les slaves

2- Chaque slave demande au master un intervalle de recherche

3- Le master envoi un intervalle

4- Chaque slave reçoit un intervalle et pour chaque mot possible de l'intervalle il le crypte et compare avec le mot de passe crypté envoyé par le master à l'étape 1

5- Le slave envoie le résultat (« trouvé » ou « pas trouvé ») au master

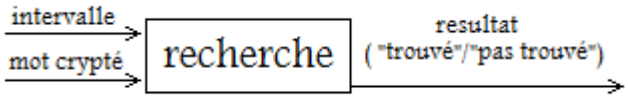
6- Selon le résultat reçu :

   a. Si le master reçoit « trouvé » alors il ordonne l'arrêt de calcul des autres slave

   b. Sinon le master passe à l'tape 3

2.Algorithme de recherche

La recherche s'effectue selon l'algorithme suivant :

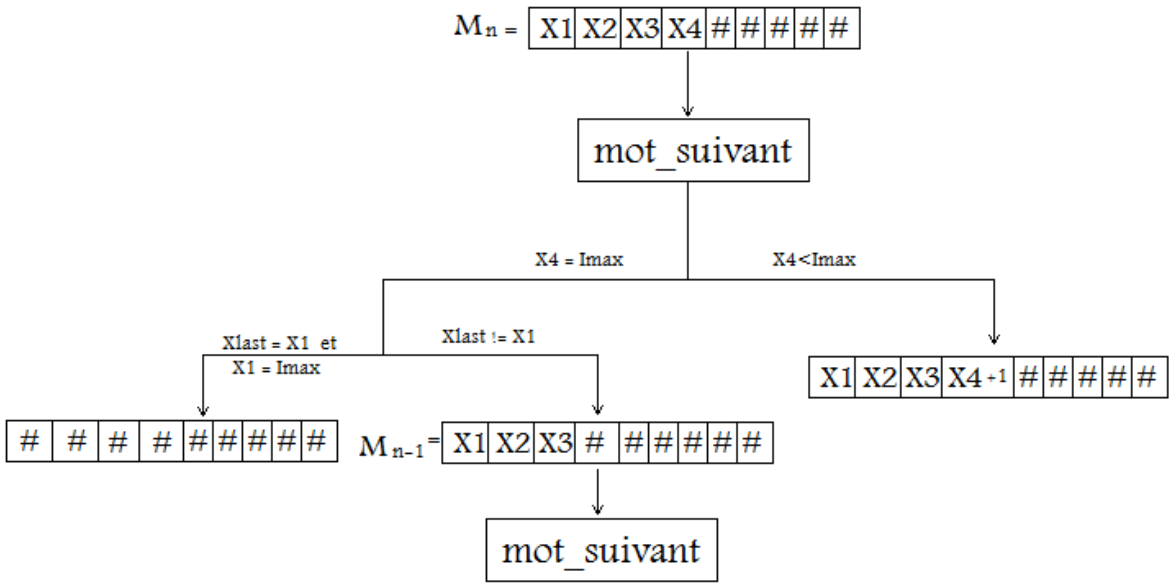


Intervalle = [minValue , maxValue]

```
Début  Recherche
        Recevoir(intervalle, mot crypté)
         Current ← minValue
         resultat = « pas trouvé »
         Tant que ( (current != maxValue) et (resultat == « pas trouvé » ) ) faire
                 crypted_current ← Crypt(current, mot crypté)
                 resultat  ← compare (crypted_current, mot_crypté)
                 Si  resultat == « trouvé » alors
                         Envoyer (resultat)
                         Exit
                 Sinon
                         Current ← mot_suivant (current)
                         envoyer (« pas trouvé »)
         Fin tant que
Fin Recherche
```

3.Algorithme de la fonction mot_suivant

L'algorithme de la fonction mot_suivant est décrit par la figure suivante :



Exemple de mots successifs avec une longueur max de 3 pour l'intervalle des mots entre 'a' et 'b'

a
aa
aaa
aab
aac
.
.
.
aaz
ab
aba
abb
.
.
.
abz
ac
.
.
.
az
b

III. État du code:

On ne pouvait pas suivre les slaves en mode débogage. On n'a pas réussi à lancer la recherche sur les slaves depuis le master.

La fonction qui va itérer les différents mot d'un intervalle a été mise en place et il reste à la tester sur les différent cas d'utilisation.