Erlang et la programmation concurrente

Partie 1

programmation concurrente vs parallèle

Le terme programmation concurrente

ne doit pas être confondu

La programmation parallèle

- La programmation concurrente se réfère à un système décomposé en tâches pouvant être exécutées **dans un ordre quelconque**.
- Certaines tâches sont exécutées avant d'autres, et certaines le sont en parallèle

Types de la concurrence

Trois types de concurrence :

- **disjointe** : les entités concurrentes ne communiquent et n'interagissent pas,
- **compétitive** : un ensemble d'entités concurrentes en compétition pour l'accès à certaines ressources partagées (par exemple, une zone mémoire, un périphérique),
- **coopérative** : un ensemble d'entités concurrentes qui coopèrent pour atteindre un objectif commun. Des échanges ont lieu entre les processus. La coopération est un élément primordial de la programmation concurrente.

Avantages

- Programmation réactive: L'utilisateur peut interagir avec des applications pendant que des tâches sont en cours d'exécution, par exemple en arrêtant le transfert d'un gros fichier dans un navigateur Web
- Disponibilité des services: Les tâches longues ne doivent pas retarder les tâches courtes, par exemple, un serveur Web peut servir une page d'entrée tout en traitant une requête complexe
- Traitement parallèle: Les programmes complexes peuvent faire un meilleur usage des ressources multiples dans les architectures multi-cœur, LAN, WAN (exemple: les applications scientifique, les simulations, les jeux etc)
- Contrôle: Les tâches nécessitant certaines conditions préalables peuvent être suspendues et attendre que les conditions préalables soient respectées, puis reprendre l'exécution de manière transparente.

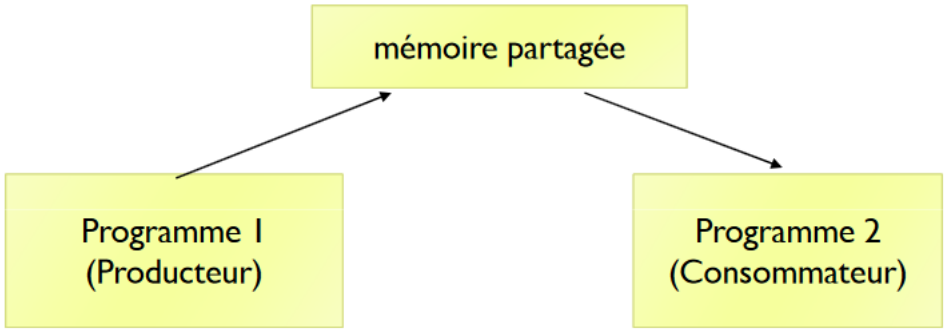
Inconvénients

- Sécurité – les tâches simultanées ne devraient pas corrompre l'état cohérent du programme
- temporisation – les tâches ne devraient pas suspendre et attendre indéfiniment l'une et l'autre (deadlock)
- Consommation des ressources – les Threads peuvent être coûteux. Trop de planification, la synchronisation, – Programmes simultanés peuvent fonctionner plus lentement par rapport au séquentiel même avec de multiples processeurs

Les modèles de communication

- par mémoire partagée (sharing memory): moyen de partager des données entre différents processus : une même zone de la mémoire vive est accédée par plusieurs processus. Ce modèle nécessite les mécanismes d’exclusion mutuelle (mutex, sémaphore, moniteur, ...)
- par envoi de message (message passing) : les composants concomitants communiquent en s’envoyant des messages asynchrones (ou synchrones: style de rendez-vous dans lequel l'expéditeur bloque jusqu'à ce que le message soit reçu). Le transfert de messages asynchrones peut être fiable ou peu fiable (parfois appelé «envoyer et prier»)

Besoin d’une synchronisation



- Soit le programme 1 produit des données et le programme 2 utilise ces données
- Besoin d’une synchronisation:
  - le programme 2 (consommateur) ne doit pas prendre des données si la mémoire partagée (mémoire tampon) est vide
  - Le programme 1 (unité du producteur) ne peut pas placer de nouvelles données dans la mémoire partagée si elle n'est pas vide

Compétition à une ressource

- Soit deux taches T1 , T2 et une variable partagée (SOMMATION)
  - La tâche T1 doit additionner 1 à SOMMATION
  - La tâche T2 doit multiplier SOMMATION par 2
  - Chaque tâche accomplit son opération en utilisant le processus suivant :
    - Chercher la valeur de SOMMATION
    - Effectuer l'opération arithmétique
    - Remettre la nouvelle valeur dans SOMMATION. Il est à noter que SOMMATION a une valeur originale de 3

Compétition à une ressource

- Sans synchronisation, 4 valeurs peuvent résulter de l'exécution des deux tâches :
  - Si T1 s'accomplit avant que T2 ne commence SOMMATION= 8
  - Si T1 et T2 cherchent la variable SOMMATION avant que l'un ou l'autre remette la nouvelle valeur, alors :
    - Si T2 remet la nouvelle valeur dans SOMMATION en premier SOMMATION = 6
    - Si T1 remet la nouvelle valeur dans SOMMATION en premier, SOMMATION = 4
  - Si T2 s'accomplit avant que T1 ne commence SOMMATION = 7

Cette situation est un état de course: plusieurs tâches font la course pour utiliser les ressources partagées et le résultat dépend de l'ordre de l'exécution des tâches

Erlang

- Le langage compile un byte-code pour une machine virtuelle dédiée;
- les entités concurrentes sont des processus ;
- les processus sont identifiés par un ID unique (un PID) ;
- les processus ne partagent pas de mémoire ou de données entre eux, autres que des messages ;
- les variables sont immutables ;
- les processus peuvent crasher (à cause d'un message inattendu par exemple ou du lancement d'une exception) ;

Pourquoi Erlang

- Bon choix pour votre application si:
  - L'application doit gérer un grand nombre d'activités simultanées.
  - L’application doit être facilement distribuables sur un réseau d'ordinateurs.
  - Rendre l'application tolérante aux pannes à la fois aux erreurs logicielles et matérielles.
  - La demande est évolutive. Cela signifie qu'il doit avoir la capacité d'étendre sur plusieurs serveurs avec peu ou pas de changement.
  - L’application sera facilement upgradable et reconfigurable sans avoir à arrêter et redémarrer l'application elle-même.
  - La demande est sensible aux utilisateurs (délais stricts).

Mise en place de l'Erlang

- Installation:

  http://www.erlang.org/downloads

- Ajouter le chemin …. \bin au PATH
- Lancer le shell Erlang par :

  erl

Quitter le shell:

halt().

Variables

- Syntaxe: toute instruction doit finir par .
- les noms des variables doivent débuter par une majuscule.

  X est une variable **x** ne l’est pas
- Une variable n'est liée qu'une fois (immuable)
- toutes les variables sont liées avec le symbole '=‘

Variables invariables

- Faire l'arithmétique est bien, mais vous n'irez pas loin sans être en mesure de stocker des résultats quelque part.
- One.
- One = 1.
- Un = Uno = One = 1.
- Two = One + One.
- Two = 2.
- Two = Two + 1. %attention
- two = 2. %attention

Variables invariables

- Techniquement, les variables peuvent commencer par un caractère de soulignement ('_'), mais par convention, leur utilisation est restreinte aux valeurs dont vous ne vous souciez pas. Exemple:

```
10> _ = 14+3.
17
11> _.
* 1: variable '_' is unbound
```

Quelques expressions

20+20.

A=10.

b=5. %Attention

A.

C=20.

A+B.

D=A+B.

A=10. % Donnez votre avis

A=20. % Attention

17.5.

-5.0.

6.022E23.

3.02E-2.

Quelques expressions

-10.

(42 + 77) * 66 / 3.

2 + 15.

49 * 100.

892 - 1472.

5 / 2.

5 div 2.

5 rem 2.

Quelques expressions

Vous pouvez utiliser plusieurs opérateurs dans une seule expression

(50 * 100) - 4999.

-(50 * 100 - 4999).

-50 * (100 - 4999).

-10*30-2+4/2 %attention à l’ordre de priorité

Quelques raccourcis en Shell

- b (). - Imprime les liaisons de variables actuelles
- f (). - Supprime toutes les liaisons de variables actuelles.
- f (x) - Supprime la liaison pour une variable, X, particulière.
- history (N) - Définit le nombre de commandes précédentes pour maintenir dans la liste d'historique (où N - est le nombre auquel la liste de l'historique des commandes doit être limitée).
- e (N) - Répète la commande N, si N est positif. Si elle est négative, la Nième commande précédente est répétée

Exercices-Partie 1

- Calculer (le rapidement possible) les nombres suivants :
- 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
- Le nombre de secondes de l’année 2016
- Le temps (à la minute près) mis par une voiture pour aller jusqu‘à la lune en ligne droite depuis la Terre à une vitesse de 300 km/h
- 22/7

Vers la base 10

- Pour convertir un nombre d’une base vers la base 10

Base#Valeur

- La base doit être de 2 à 36
- Faites attention aux nombres permis

2#101010.

8#0677.

16#AE.

Les atomes

- Les atomes représentent des valeurs constantes non-numériques.
- Un atome commence par une lettre minuscule, suivie d'une séquence composée de caractères alphanumériques, de tirets bas (‘_’), ou d'arobases (‘@’).

  Bonjour = bonjour.

  AutreNoeud = exemple@noeud.
- Les atomes de valeur autre qu'alphanumérique peuvent être délimités par des guillemets droits simples.
- AtomeAvecEspace = 'un atome contenant des espaces'.

Booléens

- Not: négation
- And: et logique classique
- Andalso: renvoie false des le premier opérande vaut false (sans évaluer le second)
- or: ou logique
- orelse: renvoie true des le premier opérande vaut true (sans évaluer le second)
- xor: or exclusif

Booléens

1> true and false.
false
2> false or true.
true
3> true xor false.
true
4> not false.
true
5> not (true and true).
false

Comparaison de termes

- == égale à
- /= différent de
- =:= exactement égale à
- =/= non exactement égale à
- =< inférieur ou égale à
- < strictement inférieur à
- >= supérieur ou égale à
- > strictement supérieur

Exercices

- Tester:

1 == 1.0.

1 /= 1.0.

1 =:= 1.0.

1 =/= 1.0.

Pour x = 5, a = (x≥12), b = (x ≤2), c = (x < 6).

- Quelle est la valeur des expressions suivantes :

- (a et b) ou c
- a et (b ou c)
- a ou exclusif b

- Quelle est la valeur des expressions pour x = 13 ?

Modules et fonctions

- Un langage de programmation n'est pas très utile si vous ne pouvez exécuter le code que depuis le shell.
- Code Erlang est divisé en modules.
- Un module se compose d'une séquence d'attributs et de déclarations de fonctions terminée par (.).

exemple

- Crérer un fichier nommé test1.erl

```
-module (test1).
-export ([double / 1]).
 Double (X) ->
    2 * X.
```

Appel dans le shell:

test1.erl peut être appelé dans le shell Erlang comme suit:

c (test1)

où c signifie compiler

Signification

- Le {ok, test1} signifie que la compilation est OK
- La première ligne de code dans le module est le nom du module (-module (test1).)
- L'appel d'un module:

module_name:function_name(arguments)

(exemple, test1:double(10).)

- La deuxième ligne indique également que cette fonction peut être appelée à partir de l' extérieur du module test1 (-**export** ([double / 1])).
- Utiliser ; si vous avez plusieurs instruction dans une même fonction

Exercices

- Changer l’Exemple précédent pour faire la somme de carré de deux nombre
- Écrivez un programme qui calcul la puissance d’un nombre (programme:power (3, 2) doit retourner 9)
- Écrivez un programme calcul le factoriel d’un nombre
- Écrivez un programme qui calcul la somme des carrés des premiers nombres

Exercices

- Écrivez un programme qui converti du degré Celsius vers Fahrenheit .
- Écrivez un programme qui converti du degré Fahrenheit vers Celsius.
- Dans un même programme, on veut faire les deux conversions: si l'utilisateur spécifie un nombre avec une unité c'est qu'il veut le convertir dans l'unité contraire

Références

- http://igm.univ-mlv.fr/~chilowi/teaching/subjects/java/net/thread/index.html
- http://erlangexamples.com/tag/datetime/
- http://learnyousomeerlang.com
- **Programmer en Erlang**