

SORBONNE UNIVERSITÉ

Réponse au question du projet Moteur d'exécution de workflows

Tarik Atlaoui

05 Mai 2020

1. Pourquoi ne peut-on pas utiliser le nom de la méthode en tant qu'identifiant de tâche ?

Si on utilise le nom d'une méthode en tant qu'identifiant de tâche l'utilisateur ne pourrait pas surcharger les méthodes. Car deux méthodes avec le même nom mais des paramètres différents représenteraient que un seul nœud

Exercice 3

2. Donner les grandes lignes de votre algorithme de la méthode exécutée. Il est attendu un algorithme décrit de manière synthétique et non un copier/coller de votre code.

Algorithm 1 Algorithm Exercice 3

```
1: procedure EXECUTE( $a, b$ )
2:    $G \leftarrow \text{Graph} < \text{String} >$ 
3:    $C \leftarrow \text{Context} < \text{String}, \text{Object} >$ 
4:    $Mtab \leftarrow \text{Method}[]$ 
5:    $result \leftarrow \text{Map} < \text{String}, \text{Object} >$ 
6:   for  $\langle \text{node in Graph} \rangle$  do                                     ▷ ont exécuté les méthodes racine
7:      $\langle \text{Chercher la méthode qui correspond au node} \rangle$ 
8:     if  $\text{nodeParam}$  is in  $C$  or  $\text{nodeNbParam} == 0$  then
9:        $result \leftarrow resultat$ 
10:    else if si il y a au moins une dépendance qui n'est pas dans  $C$  then
11:       $Mtab \leftarrow \text{methode}$                                      ▷ la méthode de node
12:    else
13:       $\langle \text{invoke méthode de node en récupérant les paramètres dans } C \rangle$ 
14:       $\langle \text{ou sans paramètres} \rangle$ 
15:    end if
16:  end for
17:
18:  for  $\langle m \text{ in } Mtab \rangle$  do                                       ▷ Ont exécuté les méthodes qui présentent des dépendances
19:     $\langle \text{invoke } m \text{ en récupérant les paramètres dans } G \text{ et } C \rangle$ 
20:  end for
21:  return  $result$ 
22: end procedure
```

Pour résumer l'algorithme utiliser, au début en récupérer tous les résultats des tâches racine tout en gardant les méthodes des autres tâches qui peuvent pas encore être exécutées, puis on parcourt les méthodes restantes et on les exécute.

Exercice 4

1. Décrire comment vous avez assuré le parallélisme des tâches indépendantes.

Nous avons considéré chaque nœud du graphe et lui avons dédié un thread, il commence par chercher la méthode qu'il doit appeler, puis pour chacun de c'est argument essaye de le récupérer, si ce dernier n'existe pas encore il attend une fois que il a fini de récupérer la valeur de c'est argument il invoque la fonction et mets le résultat dans la map

Exercice 5

1. Décrire votre protocole de communication

Le JobExecutorRemote demande le calcul d'un job à une jvm distante puis il peut choisir celle-ci lui renvoie un tuple contenant le nombre de tâche à exécuter ainsi que la Map String, Object calculer

2. Justifier votre choix de l'API de communication que vous avez utilisée.

Nous avons choisie Rmi, car elle paraissait être la plus adaptée au problème sans besoin de gérer une base de données juste l'exécution de tâche sur une ou plusieurs Jvm, aussi elle est incluse avec Java donc pas besoin d'installation préalable ce qui est un plus.

3. Quelle hypothèse doit-on faire sur le type des objets du contexte ? Justifiez.

On doit faire l'hypothèse que les types utilisés dans "Object" sont sérialisables car sinon on ne peut pas les transmettre d'une machine à une autre.

4. Quel mécanisme avez-vous utilisé pour la notification d'avancement au client ?

Dans le host qui fait le calcul il met une variable à false avant le calcul et à vrai après dès que la tâche est finie il la place à true (l'idéal c'est de garder une trace car si deux jobs se demandent la notification peut être perturbée)

Exercice 6

1. Quel est le protocole de communications entre maître-esclaves et éventuellement entre esclave-esclave ?

- 1- le Maître reçoit la demande à qui il affecte un id
- 2- le client demande son résultat et se met en attente si il n'existe pas encore
- 3- il lance un thread qui va chercher des esclaves à qui il va demander le travail, il demande à chacun s'il a des slots disponibles
- 4- une fois trouvé il lui passe l'id de la tâche ainsi que la tâche
- 5- une fois terminée il met la valeur de retour sur Taskmaster et le client qui attend son résultat peut le récupérer

2. Comment le maître affecte ses tâches équitablement sur les esclaves ?

À chaque demande il déplace l'index du prochain esclave qui va recevoir le job (si il a atteint la borne max il avance encore)

3. Comment les esclaves sont assurés de respecter leur borne de tâche courante ?

Les esclaves ont un pool de thread de taille fixe si le nombre de tâches en cours d'exécution atteint la borne de cette dernière il refuse les jobs

4. Comment gérez-vous la communication d'un résultat entre deux tâches dépendantes ? Donner les avantages et les inconvénients de votre solution.

Elle communique par le biais d'une map sur le Taskmaster qui est remplie par les esclaves et vidée par le consommateur

5. (niveau 2) Comment gérez-vous le fait d'avoir plusieurs jobs en cours d'exécution sur le cluster ?

6. (niveau 3) Décrire le mécanisme qui permet de détecter la panne d'un esclave.

Comment gérez-vous la réaffectation des tâches perdues ?