|  |  |  |
| --- | --- | --- |
| **Université Badji Mokhtar-Annaba**  Faculté des Sciences de l’Ingeniorat  Département d’Informatique |  | **جامعة باجي مختار – عنابـــــــــــــــة**  كلية علــــــــــــوم الهندســـــــــة  قسم الإعــــــــــــــلام الآلــــــــــــــي |

Année 2017-2018

Année 2008-2009

**Mémoire présenté en vue**

**de l’obtention du diplôme de Master**

|  |
| --- |
| **Étude comparative de construction d'une application distribuée avec RMI et JCSP net** |

**Domaine :** Mathématiques, Informatique et Applications aux Sciences

**Filière :** Informatique

**Spécialité :** Ingénierie des Logiciels Complexes

**Par : M. Rahim soufiane**

**Jury d’évaluation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualité** | **Nom et Prénom** | **Grade** | **Université** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Remerciements

J’ai l’honneur de remercier le dieu qui m’a aidé pour continuer mes études et qui m’a donné les aptitudes pour faire les efforts. Je ne remercierai pas Allah si je ne remercie pas mes enseignants et de plus près Mr Benouhiba mon encadrant, qui m’a beaucoup aidé au cours de ma formation et au cours de ma préparation du mémoire. Sans oublier aussi toute l’équipe ILC qui est une équipe très sérieuses, et c’est ce qui m’a encouragé a continuer après avoir eu la licence.

Je remercie le jury pour accepter à évaluer mon travail.

Et sans oublier ma famille et mes amis qui m’ont encouragé pour continuer mes études.

Et sans oublier mes collèges avec qui j’ai passé de bons moments.

Dédicaces

Les plus brèves possibles

Table des Matières

A générer automatiquement

Table des Illustrations

A générer automatiquement

Table des Programmes

A générer automatiquement (Cette table est facultative)

Introduction

Contexte de recherche

De nous jour les systèmes distribués ont une grande importance dans les solutions informatiques. Si la théorie est bien spécifiée, il reste l’aspect technique qui se traduit par l’utilisation des machine virtuelle, de middlewares et de plateformes sophistiquées afin de faciliter l’implémentation des solutions.

Les grandes firmes de l’informatique comme Microsoft et IBM et bien d’autres ont mis au point des techniques d’échange des données comme DCOM de Microsoft, RMI de Sun Microsystems (actuellement Oracle), CORBA de l’OMG, etc.

Certains peuvent se demander pourquoi ses firmes développent un système distribué, bien que nous nous trouvions plus en sécurité et plus à l'aise avec les systèmes centraux.

Enfaite, le recours à ces systèmes distribués présente des avantages.

Les systèmes distribués n'ont pas besoin d'un composant central pour survivre et si une partie du système est perturbée le système n'arrête pas l’exmple le plus conforme a cette regle est le reseaux Internet.

Le partitionnement facilite l'étude des éléments individuels

« le tout est plus grand que la somme de ses parties », il y a toujours une valeur ajoutée par l’assemblage des partie et le plus interissant c’est que cette valuer et en plus des fonctionnalités des neud assemblée.

Problématique

Par contre cette division de système en systèmes élémentaires faire apparaitre des problèmes de verification et validation, liés a la mise en interaction de ses éléments, se que l’en apelle le problème, dailleur C'est pourquoi les premier version des protocoles de communication ent des problèmes de sécurité dans, en fait cette deféculter de mise en euvre des système destrubue riside dans le faite que notre esprs n est pas abutuee a des systemme destrubuee mais a des sytemme centralisee.

En plus face à ces avancées technologique très diverses et très avantageuses les unes par à rapport aux autres, le concepteur et l‘architecte des systèmes peut rester perplexes devant la multitude des techniques et de plateformes existantes pour implémenter des systèmes distribués (solutions basées web, RMI, CORBA, JCSP.net, etc).

Parmi les problèmes de comparaisons qui sont apparus, on peut citre la comparaison entre RMI et JCSP.net qui sont des techniques de communication entre des applications distantes qui s’exécutent sur des machines distants

Motivations

Pour le premier problème j’usque a present il a un cout important dans la phase de develloppement il est le source des problème majors des boite de develloppement, malgris que la therrier de verification et validation dit que en a trois type de verification et validation a faire :

1. Les Tests ,qui conciste a faire le maximume des test au ssyteme pour essayer de summuler les plus grand nombre des cas que le système pout rencontre dans la mise en service, mais la pratique dit que les test ne montre pas le non existance de buge mais par contre il fait apparaitre les buge, sa c’est chause sure, a moi,s que en peut mettre en teste tout les cas posible, mais sa ne est pas pratique dans un systemme qui comporte u n nombre infinie d’etat dailleur si le cas majour des system reele.
2. Preuve de théorèmes en essaie dns ssetrte methode de throver demarche mathematiqque pour dire que le code ecrite ne conforme au exigance
3. Vérification de modèles consiste a mettre en plce un modelle equivalent au système ce medelle doit etre un modelle mathematique exacte qui en peut interoger pour teste le s propritee

En peut dir que en a des soulution partielle a ce jonre de probleme mais pas une methode definitife quelque soit la platformme utiliser alors en vas baser sur le deuxieme problème.

C’est de savoire quelle est la platforme ou le midelwer la moins contrainte par rapporte a ce problème de verification, de plus les autre fase de developpemment a savoire la conception et l’implimentation

C’est pour ca que les comparaisons entre ces systèmes trouve son impotence, parce que la comparaison rest un retour d’expérience qui raccourcit le chemin face au problème et donne une vision plus claire, rapide et économiquee des efforts et diminue le redéveloppement des applications.

Objectifs

Dans ce travail, on va comparer deux méthodes de communication a savoire RMI et JCSP.net, de point de vue conceptuelle et du point de vue implémentation, ainsi que de tout autre point technique. Nous cherchons à proposer une vision claire sur l’apport de chaque technique.

Contenu du mémoire

Le premier chapitre du mémoire est consacré à la présentation de RMI. Ensuite, le deuxième chapitre présente l’algèbre de processus CSP et les bibliothèques sous-jacentes JCSP et JCSP.net.

La deuxième partie du mémoire à faire la conception d’une application distribuée intitulée « MiniIsante » qui consiste à échanger les informations médicales des patients entre les médecines en utilisant les deux techniques RMI et JCSP.net.

On parlera ensuite de l’aspect implémentation de cette application avec les deux technique RMI et JCSP.net.

Le quatriéme chapitre consiste à faire une récapitulation .

A la fin , nous avons conclu par une conclusion et les perspectives.

Une comparaison sanctionnera alors ce travail dans laquelle les deux techniques seront comparées en nous basant sur le retour d’expérience du développement de l’application distribuée.

Chapitre 1   
Présentation de RMI

Introduction générale:

L’échange d’état entre machine dans un système distribué se fait par l’échange de messages, c’est pour cela que l’échange de messages devient le problème central dans le concept des applications distribuées. Mais cette distribution de système a un bénéfice majeur : le plus important est lorsqu’une machine tombe en panne, cette derniere n’affecte pas tout le système.

D’une autre part, cette distribution impose le resodre de ses problèmes :

1. L’hétérogénéité :

Une application distribuée est une couche applicative logique qui se situe au-dessus des autres couches, y compris la couche midellware,la couche système d’exploitation, et la couche matériel. Ces couches ont amenée a coopérer entre eux, mais chaque composant a ses spécifications, pour cela, le partage sera difficile. Par exemple pour la couche système d’exploitation en trouve beaucoup des système à savoir Windows ,Lunix ,Unix Mac-OS., en doit resodre le problème de non homogénité entre eux

De même pour la couche midellware on trouve : .net, RMI, RPC, CORBA et DCOM.

De même pour la couche application on trouve des langages de programmation différents qui ont des spécifications comme le langage JAVA et C#

Nous sommes alors amenés à résoudre ce problème pour que les échanger de message soit possible.

Problème de la concurrence :

Le partage de ressource pose un problème parce que ces ressources souvent ne sont pas partageables en même temps comme un ficher ou un table de base de donne ou une imprimante il faut gérer l’acces de ces resources pour ne pas avoir un conflit,. De plus, le partage de ressources dans un système réparti se fait par des messages et ces message pouvent être perdus, ce qui pose un autre problème dans la gestion de la concurrence par rapport aux systèmex centralisés.

1. Problème de sécurité :

Le problème de sécurité se pose dans tout système informatique. Cependant, dans un système distribué, ce problème se produit de la distribution elle même, par ce qu’un message échangé entre deux machines peut être intercepté, détruit ou même modifiée. En outre, il y a le problème d’authentification pour identifier un client auprès d’un serveur, il y a aussi l’attaque de service par l’essai de saturer les capacités de services offertes par les composantes de système, il y a aussi la sécurité de code mobile.

1. Problème de pannes

Une panne peut apparaître dans une couche inférieure, par exemple : la couche matérielle provoque un panne dans les couches supérieure.

1. L’absence d’information globale :

Un système distribué ne prend pas en considération une mémoire globale qui facilite la coordination entre ces composants, et le seul moyen de partager l’état d’un processus est l’envoi de message.

En effet, ce message peut faire du retard, entre temps, l’état du processus peut changer. Ainsi, le message ne reflète plus l’état correct du processus, c’est pour cela que la connaissance de l’état globale de système est un grand défi dans les systèmes distribués.

1. Le problème de test et de vérification:

Ce problème existe dans tous les systèmes, particulièrement dans les systèmes de multi-processus. Cependant, l’équipe de test va passer un temps considérable dans le test et validation d’un système distribué. Le coût de cette phase est de 60 % du coût global du système, mais dans la phase de mise en production il apparaît souvent des bugs qui ne sont pas mise en considération dans la phase de test, alors l’équipe de de développement pourrait passer des nuits blanches a essayer de détecter la cause de l’erreur pour la corriger. Ceci se rend le problème de test et de validation plus difficile, ce qui risque d’être fatal dans les systèmes distribués, comme par exemple le problème d’interblocage.

Pour résoudre ces problème, il faut adapter les phases de développement pour mieux pallier ce problème, soit dans la phase de développement, de test ou déploiement.

Et sa par l’utilisation des thechnique de devloppement ou des midelwer ou des API plus conforme au problème tretée et la méthode la plus conviviale est la méthode qui palier le mieux a ses problèmes

C’est pour cette rison que la comparaison ente les diffèrent midellwer par exemple jeux un rôle très important par ce que dans cette comparison qui est un retour d’experience en peut prevnir les chause pour ne pas faire un retour en arrier et dir que telle technique ne convien pas a telle problème faut miex essaiyer de redevlopper l’application avec telle technique

1.1 Introduction à RMI

La nature dynamique de la platforme Java a été étendue par la technique RMI qui permet d’appeller des méthodes d’un objet distant comme s’il était dans la même machine virtuelle locale. Elle est utilisée en standard dans la platforme JAVA depuis la version 1.1. Elle permet aux développeurs de se libérer des communications de la couche réseau (par exemple TCP/IP) et de concentrer sur la conception des services a faire par les objets comme si ils sont dans la même machine. De plus, la machine virtuelle JAVA aide beaucoup les développeurs pour ne pas s’occuper de l’hétérogénéité des plateformes, ce qui représente un point fort de la plateforme java.

RPC (Remote Procedure Call):

Pour comprendre RMI, il vaut mieux parler de RPC qui est un mécanisme de haut niveau pour échanger les messages entre les nœuds d’un système distribué.

RPC peut être présenté comme suit : une machine peut appeler une procédure dans une autre machine en lui transmettant des paramètres, la procédure peut renvoyer une résulta ou exécuter une action comme s’il s’agissait d’une procédure qui se situe dans la même machine. En effet, cela peut avoir des conséquences :

- une machine des deux peut tomber en pane entre le temps de l’appel et le retour de résultat,

- il y a un problème d’adressage des paramètres à résoudre parce que les paramètres sont dans des espaces d’adressage différents de la zone d’adressage de la machine qui exécute la procédure.

Pour résoudre ce problème le processus émetteur fait appel à une couche locale qui est un représentant de procédure distante et cette couche établit la connexion et passe les paramètres à un autre couche distante qui représente l’émetteur chez le processus distant. Alors, l’appel de procédure se fait simplement comme si elle est locale le la distribution soit transparent vis a vis de développeur, ce mécanisme est schématisé comme suivant :

Formalisme de message nom\_procedure (paramètres)

Extraire le résultat

Application client

Application serveur

Client

Couche client

Souche serveur

Serveur

Tempss

Formalisme de message de résultat

Extraire les paramètres de message

Communication entre client et serveur RPC

**Quelques remarques :**

* Le passage des paramètres par référence se fait par une technique de copie/restauration. Le client envoit une copie des paramètres au serveur et il reste bloquée jusque à se que le serveur traite le le résultat et il envoit une copier modifier au client qui fait la restauration de celle si sur l’ancien valeur.
* Le passage des structures complexes comme les arbres par références peut ce faire en va-et-viens par besoin.
* Les souches sont générer automatique et le programmeur coté client n’a besoin que d’une interface qui contient les entêtes des procédures à appeler. Dans la partie serveur, le programmeur doit implémenter cette interface.
* Dans le cas de procédure qui na pas de valeur de retour, le serveur envoit une notification au client pour le libérér.

1.3 Architecture RMI :

L’appel de méthode à distance n’est en réalité qu’un simple RPC,elle est assuré par ce que en appele stub de côté client, et Skelton de côté serveur, avec une capacité d’envoyer et de recevoir les objets autres que les tableaux et type primitive comme est le cas dans RPC.

Cette technologie est représentée par l’architecture suivante :

client

servuer

skilton

stub

Couche transport

Couche de referancement

Couche de referancement

Système RMI

Figure 1.1 Architecture RMI

1.2.1 Le stub : c’est l’objet représentant de l’objet distant dans la machine client ils reprisent l’objet distant a chaque appel de méthode. Il a les tâches suivantes:

1. Établer une connexion auprés du serveur
2. Envoyer l’appel de méthode avec les paramètres (sérialisation)
3. Attendre le résultat
4. Désérialiser le résultat

1.2.2 Le Skeleton : c’est le représentant de l’objet appellant de client dans la machine serveur. Il fait à chaque appel de méthode les tâches suivantes :

1. Ecouter les appelles des méthodes
2. Lecture des paramètres des méthodes si il ya des appels distant
3. Appeler la méthode de l’objet référencée
4. Le sérialisation des résultats
5. Envoyer le résultat au demandeur

Il est a noter que le Skelton et le stub sont généré automatiquement par le compilateur rmic le nom de la classe stub est XXX\_stub et le nom de la classe Skelton est xxx\_skel avec xxx est le nom de l’objet distant.

1.2.3 La couche de référencement : elle est responsable de référencer les objets distants pour donner les informations des objets à la couche transport,

Nb pour cette couche on peut utiliser la couche standard ou on peut par exemple utiliser le JNDI qui est un standard très interdisant pour laisser la configuration de serveur et client par un simple ficher de configuration sans soucier de gestion de cette couche

1.2.4 La couche transport : cette couche effectue les échanges des données entre les deux machines par un serveur socket côté serveur et un socket côté client.

On peut personnaliser cette couche par un protocole spécifique par une implémentation de SocketFactory.

1.2 Mécanisme de téléchargement des définitions des classes distantes :

Cette technique nous permet de localiser, communiquer et charger la définition des objets distant, si objet distant sont héberger dans un composent qui s’appel registre. Ce dernier envoit au client demandeur un stub qui est un proxy reprisent l’objet distant et à chaque appel de méthode, il envoit les paramètres et récupère le résultat. Il est à noter que tout cette transparence vis-à-vis de programmeur est garante par un simple serveur web.

Ce mécanisme de communication et représenté dans le schéma suivant :

Serveur Web

Client RMI

Serveur RMI

Registre RMI

Serveur Web

Mécanisme de communication RMI

Les lignes pointillées représentent le protocole de transfert les parametre et le réseltat qui est assuré par le serveur web de client et serveur.

Et les lignes continues représentent le protocole RMI proprement dit.

Techniquement ce mécanisme de transfert ulilise la sérialisation et déserialisation des objets pour permettre le transfert des paramètres, du résultat et les exceptions et même le stub. Avec la technique de chargement dynamique des definition des classes, le résultat c’est un platforme distribuée qui permet l’appel des méthodes distantes comme un appel local.

Le mécanisme d’activation des objets distants :

RMI utilise un technique d’activation dynamique des objets qui ne sont pas référencés par des clients mais sont juste exposées aux clients, parce que les implémentations des ces objet peuvent attendre un million d’objet et ne peuvent pas être tous activé à tout moment dans ce cas ils sont dans un état passif et le passage à l’état actif est retardé jusqu’au premier appel de méthode distant.

Gestionnaire de sécurité java

Le gestionnaire de sécurité Java est une API qui peut être utilisée dans le cas de tentative d’accès au poste par des machines distantes. On peut utiliser avec la technologie RMI pour empêcher ou autoriser quelque poster pour accéder au poste à distance. De plus, il nous permet de contrôler l’accès par un simple ficher de configuration que l’administrateur peut éditer afin de changer la stratégie.

Vous trouvez un exemple dans l’annexe de l’utilisation de Security Manager qu’on peut ne pas utiliser.

Garbage collector distribué

Le mécanisme Garbage collector distribué de Java utilise une technique de nombre de références pour chaque objet distant. Ce nombre est incrémenté chaque fois que il y a référencement par un client, ce client envoit un message indiquant que l’objet distant est référencé. Le nombre est décrémentée chaque fois que cette référence est libérée par le client. Dans ce cas, la machine virtuelle a le droit de récupérer cet objet s’il n ya pas une référence local de cet objet.

Multiplexage des communications

Cette technique est utilisée par RMI pour ne pas utiliser un socket serveur pour écouter les clients mais un simple socket. Par exemple, dans certains systèmes, l’administrateur empêche les applets à ouvrir une porte pour l’écoute des clients, et cela par un souci sécurité. Dans ce cas, le seul moyen et de multiplexer le canal de communication de client pour envoyer et recevoir les informations.

Annexe : RMI en Pratique

Pour comprendre ce mécanisme de communication, on va présenter en va utiliser un exemple de code basique qui reprend l’appel d’une méthode d’un objet distant par un client.

**Création de serveur RMI avec le registre :**

Cette étape est très simple à partir de version 6 de Java, parce que la définition de stub est transparente du point de vue de programmeur. Elle est gérée par la platforme java,

Cependant, le développeur n’a que faire la conception des objets à héberger. Cela se fait par la conception d’une simple interface Java qui représente les méthodes que les clients peuvent appeler à distance. Cette classe a deux règles simples :

1. Il faut qu’il étende la classe Remote
2. Il faut ajouter throw remoteException dans chaque méthode distante

Une interface qui représente une seule méthode a le squelette suivant :

public interface MonIterface extends Remote {

TypedeRetour NomMetode(TypeParametre t) throws RemoteException;

}

L’implémentation de cette classe par un objet a aussi des règles :

1. Bien sur il faut implémenter toutes les méthodes de cette interface et,
2. Une classe distante doit étendre la classe java.rmi.server.UnicastRemoteObject [[1]](#footnote-2)(d’autres possibilités´es existent) [[2]](#footnote-3)
3. Définir un constructeur

L’implémentation a la forme suivante :

public class nomObjetDestant implements NomInterface {

public nomObjetDestant () {

super();

}

public TypedeRetour executeTask(TypeParametre t) {

// Traitement a faire

}

}

Après, il faut créer un programme Java qui démarre le service de cet objet. Le programme a la forme suivant

public class Lanceur {

public static void main(String[] args) {

if (System.getSecurityManager() == null) {

System.setSecurityManager(new SecurityManager());

}

try {

nomInterface nomObjet = new nomObjetDestant();

nomInterface stub =(nomInterface) UnicastRemoteObject.exportObject(nomObjet, 0);

Registry registry = LocateRegistry.getRegistry();

registry.rebind("nomSercvice", stub);

} catch (Exception e) {

System.err.println("ComputeEngine exception:");

e.printStackTrace();

}

}

}

Explication :

La première instruction

if (System.getSecurityManager() == null) {

System.setSecurityManager(new SecurityManager());

}

Il permet d’installer un gestionnaire de sécurité Java parce que on a affaire avec un téléchargement de définition des classes et cela doit être contrôlé par des règles sur mesure. Ce point va être présenté après dans la compilation et lancement de l’application.

Après, on a l’instruction suivante:

nomInterface nomObjet = new nomObjetDestant();

Cette instruction est une instruction standard il représente la création d’une référence locale de l’objet serveur. Après on a :

nomInterface stub =(nomInterface) UnicastRemoteObject.exportObject(nomObjet, 0);

Cette instruction transforme cette référence locale en une référence qui peut être hébergée dans un registre RMI.

Il est à noter que ce 0 et un entier qui signifie que le port de l’écoute de ce service, est définie soit par le système d’exploitation ou il est par défaut. Si le programmeur veut spécifier le port, il doit changer le 0 par un nombre entier qui représente le port.

Registry registry = LocateRegistry.getRegistry();

Cette instruction lance un registre RMI pour héberger l’objet qui vient d’être créé. Après, on a l’instruction :

registry.rebind("nomSercvice", stub);

Cette instruction héberge l’objet distant dans le registre.

Tout cela doit être enveloppé par un try/catch, parce que ces instructions peuvent générer une exception (par exemple, échec d’ouvrir un registre, erreur d’hébergement de l’objet distant, etc).

Il faut préciser que bind, unbind, ou rebind peuvent être appelées pour un registre qui se trouve dans la même machine de l’application serveur par contre l’instruction lookup que l’on va utiliser du côté du client pour récupérer une référence de l’objet distant peut être appelée pour une application distante. La raison de cette restriction est la sécurité.

Après on vas créer un client qui va demander les services des objets distants. La création d’un client a la forme suivant :

public class ComputePi {

public static void main(String args[]) {

if (System.getSecurityManager() == null) {

System.setSecurityManager(new SecurityManager());

}

try {

Registry registry = LocateRegistry.getRegistry (args[0]);

nomInterface comp = (nomInterface) registry.lookup ("Compute");

TypeRetour valeur = comp.NomMethode();

System.out.println(valeur);

} catch (Exception e) {

System.err.println("ComputePi exception:");

e.printStackTrace();

}

}

}

Ce paragramme et très simple et basique il crée une référence au registre qui se trouve dans une machine qui est déterminée par l’argument 0 du programme. Il interroge ce registre pour récupérer une référence de l’objet distant qui est représenté par un stub transparent dans les versions antérieur de la version 5 de java. Après il peut invoquer une méthode de cet objet

Il est à noter qu’il faut créer une interface identique à l’interface du server pour récupérer la référence de l’objet distant.

Une autre remarque très importante concerne l’utilisation des objets comme paramètres d’un appel de la méthode distante ou lorsque le résultat est un objet. Dans ces cas, il faut que les types soient de type primitif, des classes distantes ou sérialisable.

Il est à noter que les constructeurs obligatoires dans l’implémentation de l’interface distant appel un contracture de stub qui est une tâche transparente. A partir de la version 6 de JDK, ce stub contient un référence de l’objet distant et une méthode invoke() qui a la tache d’appel de méthode distante.

Dans le site de oracle vous trouvez un exemple très puissant. Cet exemple est un serveur de calcul distribué, il est très utile dans scénario. Si on a une puissante machine qui peut exécuter un code arbitraire fourni par le cliente et transmettre le résultat au client, car il nous a permette de voire le cas de transmettre des informations de client ver le serveur (par les paramètres d’appel de méthode qui représente un classe contiens un méthode arbitraire a exécuter par le serveur) et de serveur au client (comme est le cas dans tout les application qui utilisent RMI c a d par le renvoi de définition de l’objet distant et le résultat après l’exécution).

Il est à noter que la gestion de concurrence est à la tâche du développeur et se fait par *synchronized* et *wait*, etc. Elle peut alors introduire inter-blocage si on ne fait pas attention.

Note sur la sécurité : dans le cas d’utilisation d’un gestionnaire de sécurité dans le code du client ou du serveur, il faut créer un fichier de configuration de sécurité. Voici un exemple de deux fichiers de sécurité qui donnent toutes les permissions à l’exécution dans un répertoire donnée.

Pour le serveur un ficher appeler server.policy

grant codeBase "file:/home/ann/src/" {

permission java.security.AllPermission;

};

Pour le client un fichier appeler client.polycie

grant codeBase "file:/home/jones/src/" {

permission java.security.AllPermission;

};

après il faut démarre le registre par cette commande :

start rmiregistry

Il est a note que la porte par défaut de registre et 1099 pour démarre le registre dans une autre porte vous devez utiliser la commande suivante par exemple:

start rmiregistry 2000

Exécution de serveur :

java -cp c:\home\ann\src;c:\home\ann\public\_html\classes\nomInteface.jar -Djava.rmi.server.codebase=file:/c:/home/ann/public\_html/classes/nominterface.jar - Djava.rmi.server.hostname=mycomputer.example.com

-Djava.security.policy=server.policy

PakageObjetDestant.nomObjetDestant

execution de client :

java -cp c:\home\jones\src;c:\home\jones\public\_html\classes\compute.jar -Djava.rmi.server.codebase=file:/c:/home/jones/public\_html/classes/

-Djava.security.policy=client.policy

PakageClient.nomClaseClient mycomputer.example.com 45

Le paramètre mycomputer.example.com désigne un nom de machine

Et le paramètre Djava.rmi.server.codebase=file:/c:/home/ann/public\_html/classes/nominterface.jar et Djava.rmi.server.codebase=file:/c:/home/ann/public\_html/classes/nominterface.jar reprisent le répertoire de client et le serveur

Pour plis d’explication sur le sujet de sécurité et de compilation et d’exécution consulter le site oracle dans la section RMI, sa ne défiance pas des base de compilation et exécution des applications java, il existe des exemples de code RMI dans le net qui n’utilisent pas le gestionnaire de sécurité dans ce cas le code va devenir plus facile avec la plate-forme eclipse.

1.4 Conclusion

Comme vous voyez il n y a ni sockette ni porte malgré que en peut paramétrer les portes mais par défaut tout marche bien et le développeur va concentrer sur la conception de la couche métier et il a l’avantage de développer une application distribuée comme si elle est une seule application dans une seule machine virtuelle,

***CHAPITRE II : JCSPnet***

Définition :

CSP est un langage formel de modélisation des processus concurrents introduits en premier par [C. A. R. Hoare](https://fr.wikipedia.org/wiki/C._A._R._Hoare) en 1978. Il a été depuis amélioré et implémenté par plusieurs outil de modélisation et des bibliothèques de programmation. Sa syntaxe a été enrichie par l’ajout des types de données et des formalismes comme le pi-calcul dans la dernière amélioration de occam-pi.

Basé sur les processus et les canaux de communication, ce langage permet de mettre les processus en concurrence, en séquentielle ou en alternative indéterministe. Ce qui permet aux développeurs de se concentrer sur le côté métier de chaque processus sans s’occuper de la partie technique de la communication. C’est comme si ces processus sont des objets du monde réel, ce qui n’est pas possible dans la programmation classique.

Occam permet aux programmes de bénéficier de deux ponts très important

1- il permet de rendre le programme très lisible par cacher l’aspect technique de code source

2- il permet la vérification de la validité de programme avec la logique temporelle ce qui n’est pas permet dans les autre langages comme C++, Fortron ou pascal.

Syntaxe CSP occam

Les processus dans occam son des objets qui exécutent une tache en permanente et n’acceptent la communication que par des canaux

Et ses processus peuvent être lancée séquentiellement ou parallèlement ou alternativement par un choix indéterminée

l’envoi des donnés sur un canal quelconque se fait par le mot clé ! sur le ce canaux

Et la réception se fait par le mot clé ? sur le canaux

Par exemple :

Mychannel? Ou Mychannel!

Et en a pas a soucier de l’implémentation de la communication

Et les canaux sont de type unidirectionnel ou bidirectionnelle de type un a plusieurs ou plusieurs a un

Il est a note que en peut mettre des processus encapsuler dans un autre processus par ce que la seul moyenne de communique un processus avec le monde extérieur et les canaux et alors le fête d ne pas exporter un canaux a l”Extérieur le rendre privée, avec se technique en peut exposer le canaux de processus conteneur et ne pas exposer le canaux des processus internes

JCSP

Introduction

L’utilisation de la programmation séquentielle pour créer un système concurrent est très difficile parce que le modèle implémenté ne reflète pas souvent la réalité. En plus, on a pas un moyen de vérifier les propriétés de système. Avec les méthodes de test traditionnelles, on peut être satisfait de fonctionnement de système mais il reste un ou plusieurs cas que l’on n’a pas testés, et qui sont générés chez le client. Ainsi, l’équipe de test vas passer des nuits blanches a provoquer le bug pour corriger, surtout dans une application multi-threads.

L’une des solutions de ce problème est une méthode mathématique qui base sur la vérification de modèle. modèle il faut qui il reflète la réalité ce qui n est pas facile a concevoir , s est pour sa que il est apparue des bibliothèque de programmation capable de mètre des threades en concurrence en suivant les concepts mathématique de langage CSP comme JCSP qui est un implémentation java de CSP aucam et JCSP.net qui est la version distribuée de JCSP qui suit ocame-PI.

Présentation de JCSP

##### Un processeur dans jcsp est un classe qui implémente l’interface CSProcess du package jcsp.lang;

 public interface CSProcess

{

    public void run();

}

De même, la base des canaux ce est deux type de canaux ChannelInput qui a un seule méthode Read() :

interface ChannelInput {

public Object read ();

}

Ce type de canaux est utilisé dans les autres types des canaux pour lire. Il est bloquant jusqu’à ce que l’autre bout du canal fasse la l’écriture. Il n a pas de tampon.

L’autre type de canaux est utilisé pour écrire à l’autre bout des canaux est ChannelOutput. Il a une seule méthode write(Object o) qui est bloquante et n’a pas de tampon.

interface ChannelOutput {

public void write (Object obj);

}

Les autres types de canaux héritent de ces deux types basiques. On peut citer :

* One2OneChannel : il est bidirectionnel
* One2AnyChannel : rédacteur et plusieurs lecteurs
* Any2OneChannel : plusieurs rédacteurs et un lecteur
* Any2AnyChannel : plusieurs rédacteurs et plusieurs lecteurs

Voici un exemple de processus en jcsp :

class SuccInt implements CSProcess {

private final ChannelInputInt in;

private final ChannelOutputInt out;

public SuccInt (ChannelInputInt in,ChannelOutputInt out) {

this.in = in;

this.out = out;

}

public void run () {

while (true) {

int n = in.read ();

out.write (n + 1);

}

}

}

Comme on l’a dit auparavant, par défaut les canaux n’ont pas de tampons. En effet, il y a une classe interne qui s’occupe de buffer, à savoir ChannelDataStore et bufer et InfiniteBuffer.

- ChannelDataStore : par défaut elle est de type ZeroBufer qui signifie pas de tampon.

- la classe Buffer qui a un tampon de type Fifo

- InfiniteBuffer qui a un tampon de type fifi infini qui bloque le lecteur s’il est vide et ne bloque pas les rédacteurs jusqu’à atteindre la limite physique de la mémoire.

Il est à noter que les canaux sont synchronisés, la synchronisation se fait à l’aide des moniteurs.

Le parallélisme se fait avec un classe parallèle qui prend en paramètre un tableau de processu. Cette classe exécute tous les processus en parallèle dans des threads différents, sauf le dernier qui est exécutée dans la même thread. Le thread se termine si tous le threads internes se terminent.

**Le choix non déterministe ALTernative :**

Jusqu’ici, on n’a parler que de l’exécution déterministe des choix ou commandes, c’est-à-dire à l’arrivée du message dans les canaux il doit être traité.Mais, JCSP a une commande ALT qui peut traiter des messages qui ont une garde vraie. Dans les gardes, on peut combiner des tests binaires avec l’arrivée des entrées, et le choix d’une commande se fait de manière non déterministe.

**CSTimer:**

C’est une classe qui donne la possibilité de créer un Timer pour qu’il soit utilisé comme garde dans la classe Alternative.

JCSP.net

L’extension de JCSP par l’aspect distribué a donné naissance à JCSP.net, c’est-à-dire que les canaux ne sont pas locaux mais sont des canaux distants qui ont des extrémités distantes. L’émetteur et le récepteur ne sont pas forcément dans la même machine.

Avec une différence syntaxique que en plus de CSP de Hoare, est ajoutée le Pi-calcul qui reprisent dans le calcul des processus comme le lambda calcul dans informatique traditionnelle.

Une autre différence sémantique est que les objets passent dans un canal local par référence mais dans un canau réseau, ils sont passés par copie.

Processus

Processus

Net Chanel output

RX

TX

NET Chanel input

TX

RX

Architecture de canaux virtuels de JCSP.net

Un canal en JCSO.net est divisé en client et serveur lien, ces deux liens sont déterminés par une adresse IP, un port et un numéro virtuel utilisé pour multiplexer et démultiplexer les messages d’un même processus. Les liens sont enregistrés dans un serveur (Chanel name server CNS)

Ce diagramme présente ce mécanisme:

CNS

“name” Q 42

Q

Processus sever

P

Processus Client

2

1

3

1) le serveur demande au CNS d ‘ouvrir un canal sur le numéro virtuel 42

2) et le client demande la connexion au canal 42

3) le client envoi au serveur les données sur le canal 42

Il est a noté qu’après la connexion du client et du serveur, on n’a pas de soucis sur le port et l’adresse IP et le numéro virtuel de l’autre bout de connexion, on n’a besoin que du nom de canal.

Les types de canaux réseaux en plus des canaux locaux vus précédemment sont :

* One2netchannel : une machine qui peut écrire sur le réseau
* Net2onechannel : une machine qui peut lire sur le réseau
* Net2Anychannel : plusieurs machine qui peuvent écrire sur le réseau
* Any2netCahnnel : plusieurs machin qui peuvent lire sur le réseau

**Remarque :** on peut ne pas utiliser un CNS mai envoyer un objet NetChannelLocation dans un canal existant. Cet objet représente les informations de location qui peut être utilisé comme paramètre dans le connecteur de canaux, on parle ici de canaux anonyme.

Cela peut être utile dans le cas où on peut communiquer entre processus sans que le canal partagé de CNS soit libre. C’est comme une connexion locale de type One2One, en effet, le CNS exige qu’une seule échangée soit possible et dans les connexions anonymes elle est privée et ce problème de synchronisation ne se pose pas.

On peut même envoyer un processus entier dans un canal. Voici un exemple qui peut être utile dans le cas où on veut exécuter un code dans une autre machine :

while (running) {

Bond james = (Bond) in.read ();

james.plugin (a, b, c);

james.run ();

NetChannelLocation escapeRoute =james.getNextLocation ();

One2NetChannel escape =new One2NetChannel (escapeRoute);

running = james.getNuke ();

escape.write (james);

escape.disconnect ();

}

Ce code représente la méthode run() d’un processus distant qui accepte un processus et l’exécute dans un autre thread ce qui est similaire au agent mobile dans les paradigme connu de code mobile.

On peut aussi déplacer les extrémités des canaux mais cela change la topologie de réseau qui peut mener à des situations inattendues soit disent interblocage

Tout ce qui est nécessaire est de communiquer le nom du canal CNS (ou NetChannelLocation) au nouveau processus d'écriture, c’est ce qu’on appellle migration de canal.

**Explication de mécanisme de migration de canaux :**

Q

P0

x

P1

R

y

Q

P0

x

P1

R

y

Avant

Après

Considérons un processus, x, sur le nœud Q, qui gère actuellement le canal "foo" enregistré par le CNS. Il veut transmettre cette responsabilité à un processus (volontaire), y, dans le nœud R, avec lequel il est en contact.

Les processus qui écrivent sur "foo" ne doivent pas être conscients de cette migration du canal. Revenons à l'état initial ("foo" desservi par x sur le noeud Q). D'abord, le processus x gèle le nom "foo" sur le CNS. Ce dernier renvoit une clé de dégel pour traiter X.

Le CNS ne résoud plus "foo" pour les nouveaux auteurs et interdit également les nouveaux enregistrements du nom. Ensuite, le canal réseau est supprimé du processeur Q. Tous les messages en attente et futurs pour ce canal (42) sur Q sont renvoyés (NetChannelIndexException)

La méthode write () en P1 gère ce rebond par appel au CNS pour le nouvel emplacement de "foo". Cela ne réussira pas avant le processus x (sur le nœud Q) passe sur le nom du canal ("foo") et la touche de déblocage du CNS . Le récepteur (processus y sur R) déverrouille le nom "foo" (en utilisant la touche) et le réenregistre. Le récepteur (processus y sur R) déverrouille le nom "foo" (en utilisant la touche) et le re-enregistre.

• La méthode write () en P1 entend maintenant depuis le CNS le nouvel emplacement de "foo" et renvoie le message qui a été rebondi. Le (s) processus d'écriture à P1 (et ailleurs) ignorent tout de la migration.

Résumé

JCSP.net permet la communication par canal virtuel entre des processus sur des machines séparées (JVM).

• Les canaux d'application / connexions entre les machines sont configurés (et supprimés) dynamiquement.

• Les canaux / connexions sont multiplexés sur les liens.

• Des liens peuvent être développés pour n'importe quel protocole de réseau et branché sur l'infrastructure JCSP.net.

• Aucune gestion centralisée - connexions per to per (amorçées à partir d'un serveur CNS de base).

• Les courtiers pour les services de correspondance définissables par l'utilisateur sont faciles à configurer en tant que serveurs d'applications ordinaires.

Les processus peuvent migrer entre processeurs (avec des classes chargées dynamiquement selon les besoins) - d'où les agents mobiles, les worker farmsl, le grid computation ...

• JCSP.net fournit exactement le même modèle de concurrence (CSP / occam) pour les systèmes en réseau que JCSP fournit dans chaque nœud physique de ce système.

• La logique du réseau est indépendante de la distribution physique (ou même si elle est distribuée).

• Accent mis sur la simplicité - à la fois dans la mise en place de réseaux applicatifs et dans le raisonnement à leur sujet.

Annexe A

Annexe B

***Résumé***

Le résumé ne doit pas excéder une douzaine de lignes. Il doit être écrit de sorte à permettre aux lecteurs de déterminer rapidement si le mémoire l’intéresse ou pas. Pour cela, il doit donner un bref aperçu sur le sujet, la problématique, la motivation qui pousse à la recherche de la solution ainsi que ce qui a été réalisé (aperçu de la solution proposée). Le résumé se termine par une note d’optimisme qui situe la solution par rapport à ce qui existe. Le résumé ne comporte pas de sous titre.

**Mots clés**. Plug-in, Eclipse, Composant logiciel,

**ملخص**

مع مرور الزمن

**كلمات مرشدة.** برمجيات، فصل الانشغالات،

1. « Java Remote Method Invocation ». [↑](#footnote-ref-2)
2. Voir l’exemple de site oracle le moteur de calcule qui n’etendre pas la classe java.rmi.server.UnicastRemoteObject [↑](#footnote-ref-3)