Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Ben M'hidi, Oum El Bouaghi
Département de mathématiques et d'informatique

Correction Examen : Architectures distribuées

Master : AD Le 16 /01/ 2023

QCM (10 Pts)

1.	Une architecture logicielle est un document qui décrit les composants logiciels et leurs dépendances mutuelles		
OUI	**X**	NON	
2.	Dans la démarche d'architecture, un connecteur entre deux composants permet de définir à la fois, un lien local de communication entre deux composants et un lien distant de communication entre deux composants situés dans des machines différentes.		
OUI	**X**	NON	
3.	Le principe de persistance en base de données est une propriété intrinsèque aux architectures à base de composants dans lesquelles tous les composants enregistrent en temps réel ses données dans un support de persistance qui est la base de données.		
OUI		NON	**X**
4.	Quand cela est possible, dans une démarche d'architecture, on essaye de privilégier un couplage faible entre les composants		
OUI	**X**	NON	
5.	Un Web Service est un service informatique qui repose sur les standards de RMI.		
OUI		NON	**X**
6.	Une couche de persistance des données d'un Système d'Information est :		
X	a. Une couche logicielle développée soi-même permettant de garder en base de données les attributs des objets (données)		
	b. Une couche logicielle qui permet de garder en mémoire d'un composant dynamique les attributs des objets (données)		
7.	Le principe d'une Architecture à Base de Composants est particulièrement adapté pour réaliser les architectures :		
X	a. Distribuées		
X	b. N-tiers		
	c. Orientées évènements		
8.	Le protocole SOAP des architectures WEB Services est un protocole basé sur :		
X	a. Les standards d'Internet (HTTP)		
	b. Les standards d'Intranet (RMI ou CORBA)		
9.	Dans une architecture logicielle, un composant est :		
X	a. Une unité de composition logicielle, exposant des interfaces bien spécifiées		
X	b. Une unité de composition logicielle, susceptible d'être déployé de manière indépendante		
	c. Une unité de composition logicielle spécifique qui ne peut plus se décomposer en d'autres unités de composition logicielle		
10.	Une architecture 4-tiers est un style d'architecture qui est composée des tiers suivants :		
	a. Client + Présentation + Base de Données + Gestion des erreurs		
X	b. Client + Présentation + Composants métier + Base de Données X		
X	c. Client + Composants métier + Composant DAO + Base de Données X		

EXERCICE 1 (4 Pts)

Vous travaillez dans une entreprise qui propose de réaliser pour ses clients des applications réparties avec 3 technologies différentes : les sockets TCP/IP, Java RMI ou CORBA.
Donner et expliquer les critères qui vous conduiront à choisir les sockets, Java RMI ou CORBA pour la réalisation d'une application donnée.

- Differences between RMI and Socket :

RMI	Socket
L'invocation de méthode à distance est essentiellement une API qui permet à un objet d'invoquer une méthode sur un objet s'exécutant dans la JVM d'une autre machine.	Les sockets ne sont rien d'autre que des liens de communication bilatéraux entre deux programmes (client et serveur) dans un réseau.
RMI est une invocation de méthode à distance, ce qui signifie que les méthodes sont appelées à distance ou accèdent à des sites distants dans la communication client-serveur.	Les sockets sont comme des passerelles qui fournissent des points d'accès aux programmes via certains numéros de port spécifiques.
RMI est construit sur des sockets. Sans prises, RMI n'existerait pas.	En cela, nous devons gérer les sockets et les protocoles que l'application utilisera. Même si nous pouvons formater les messages voyageant entre le client et le serveur.
RMI est orienté objet	Alors que non.
RMI gère le formatage des messages entre le client et le serveur.	Ici, nous spécifions le type TCP ou UDP, nous devons gérer tout le formatage des messages circulant entre le client et le serveur.
RMI est une technologie spécifique à Java.	La communication par socket est indépendante des langages de programmation.
RMI est destiné à l'informatique distribuée Java vers Java de haut niveau.	Les sockets sont destinés à la communication réseau de bas niveau.

RMI	CORBA
RMI est une technologie spécifique à Java.	CORBA a une implémentation pour de nombreuses langues.
Il utilise l'interface Java pour la mise en œuvre.	Il utilise le langage de définition d'interface (IDL) pour séparer l'interface de l'implémentation.
Les objets RMI sont récupérés automatiquement.	Les objets CORBA ne sont pas ramassés car ils sont indépendants du langage et certains langages comme C++ ne prennent pas en charge le ramasse-miettes.
Les programmes RMI peuvent télécharger de nouvelles classes à partir de JVM distants.	CORBA ne prend pas en charge ce mécanisme de partage de code.
RMI transmet les objets par référence distante ou par valeur.	CORBA passe les objets par référence.
Java RMI est un modèle centré sur le serveur.	CORBA est un système peer-to-peer.
RMI utilise le Java Remote Method Protocol comme protocole de communication à distance sous-jacent.	CORBA utilise le protocole Internet Inter-ORB comme protocole de communication à distance sous-jacent.
La responsabilité de localiser une implémentation d'objet incombe à JVM.	La responsabilité de localiser une implémentation d'objet incombe à l'adaptateur d'objet, soit à l'adaptateur d'objet de base, soit à l'adaptateur d'objet portable.

On dit généralement que RMI est une solution "tout Java", contrairement à la norme Corba de l'OMG (Object Management Group) permettant de manipuler des objets à distance avec n'importe quel langage. Corba est toutefois beaucoup plus compliqué à mettre en oeuvre, c'est la raison pour laquelle de nombreux développeurs se tournent généralement vers RMI.

EXERCICE (6 Pts)

Envisagez la conception de systèmes distribués mobiles. Comparez les principaux paradigmes de la mobilité du code dans les systèmes distribués et comment ils affectent les principaux objectifs de conception.

ELEMENTS DE REPONSE

- La notion mobilité permet à un objet logiciel (OL) de se déplacer d'un site à un autre en cours d'exécution pour accéder à des données ou à des ressources. Il se déplace avec son code et ses données propres, mais aussi avec son état d'exécution. Il décide lui-même de manière autonome de ses mouvements. Ainsi, la mobilité est contrôlée par l'application elle-même, et non par le système d'exécution comme dans le cas de la migration de processus dans les systèmes opératoires.

- Ici le savoir-faire appartient au client. Les Objets mobiles peuvent être vus comme une généralisation de l'évaluation distante, où en plus du code, l'unité d'exécution est mobile. Les Objets mobiles peuvent transporter des données paramétrées. Afin d'accéder aux ressources, les processus migrent de manière autonome vers les composants qui les proposent.
- Au niveau de la mise en oeuvre, les migrations d'objets mobiles peuvent s'effectuer selon deux modes :

A. **Migration forte** : La migration forte, où la totalité de l'objet (c'est-à-dire code, données et unité d'exécution) migre vers le nouveau site. Pour cette migration réelle, l'objet est suspendu ou capturé avant d'être transféré. Une fois arrivé sur le site distant, il redémarre son exécution au point de contrôle précédent, en conservant l'état du processus.

Une autre possibilité proposée consiste à stopper l'exécution de l'objet avant la migration puis d'en créer une copie distante identique sur le site distant (migration par réplication).

B. **Migration faible :** La migration faible où seul le code et les données migrent.

MigrationFaible ~ code + données-paramètre

Une fois arrivé sur le site distant, l'objet est réexécuté. Le programmeur doit donc préserver, dans les données, les informations d'état permettant la poursuite logique de l'exécution.

Comment ils affectent les principaux objectifs de conception.

En pratique, la mobilité permet de rapprocher le client et le serveur et en conséquence de réduire le nombre et le volume des interactions distantes (en les remplaçants par des interactions locales), de spécialiser des serveurs distants ou de déporter la charge de calcul d'un site à un autre. Une application construite à base du code mobiles peut se redéployer dynamiquement suivant un plan préétablit ou en réaction à une situation particulière, afin par exemple d'améliorer la performance ou de satisfaire la tolérance aux pannes, de réduire le trafic sur le réseau, ou de suivre un composant matériel mobile. La mobilité du code offre un premier niveau de flexibilité aux applications. La décentralisation de la connaissance et du contrôle à travers les objets, et la proximité physique entre les objets et les ressources du système renforce la réactivité et les capacités d'adaptation.

La mobilité ne se substitue pas aux capacités de communication des objets (la communication distante reste possible) mais les complète ; afin de satisfaire aux contraintes des réseaux de grande taille ou sans fil (latence, non permanence des liens de communication), les objets communiquent par messages asynchrones le plus souvent.

Les objets mobiles, bien qu'ils connaissent quelques difficultés, restent une architecture appropriée pour la manipulation de grands volumes de données notamment des données multimédia.

Conclusion

La migration du code vers les données offertes par l'utilisation d'un code mobile :

- Réduit la latence de certaines étapes liée à l'utilisation réseau,
- Évite les transferts de données intermédiaires à travers le réseau lors d'un calcul,
- Permet de continuer l'exécution du calcul malgré la présence de coupures du réseau.

Si ces différents avantages permettent le plus souvent à un code mobile de réaliser des calculs plus rapidement qu'avec une solution traditionnelle de type client/serveur, il arrive que l'utilisation d'un code mobile puisse ralentir l'exécution d'un calcul. En effet, le code peut être plus important (en nombre de bytes) que les données avec lesquelles il travaille. Dans ce cas le transfert de code est plus long que le transfert des données. De la même façon, si le réseau offre des temps de transfert rapides alors l'exécution d'un code mobile peut être plus lente que le transfert des données.