Systèmes à agents mobiles

Daniel Hagimont
INRIA Rhône-Alpes

http://sirac.inrialpes.fr/~hagimont

Plan de la présentation

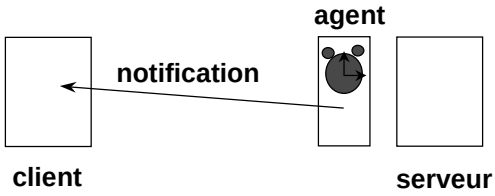
- Principes
- Motivations des agents mobiles
- Définitions
- Aglets
- Expérimentations

Principes

- Agent Mobile : processus, incluant du code et des données, pouvant se déplacer entre des machines pour réaliser une tâche
- Agent
  - objet actif
  - communique potentiellement avec d'autres agent
- mobile
  - se déplace en fonction de ses besoins
  - suit parfois un itinéraire

Motivations

- Agent notificateurs/réactifs
  - attend une information ou un événement
  - prévient un usager ou déclenche une action
  - exemples
    - recherche d’emploi dans des journaux
    - gestion d’un portefeuille d’actions

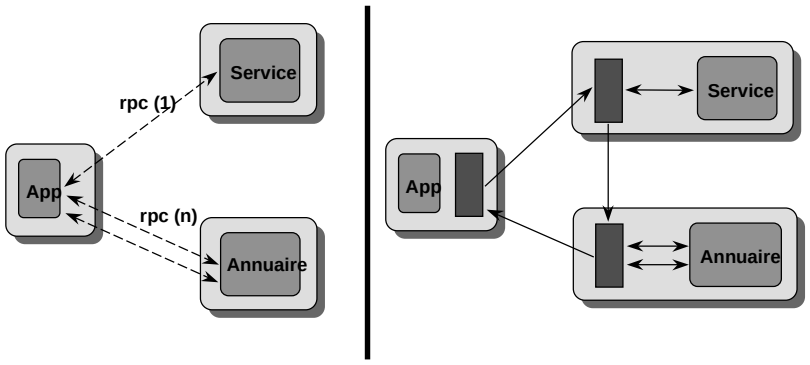


Motivations

- Agents itinérants
  - réalisant une suite d'interaction avec des serveurs
  - privilégie les accès locaux
  - exemple : jointure entre deux sources de données
    - un serveur retourne une liste de noms (Hotels)
    - un serveur gère un annuaire téléphonique
    - les serveur sont gérés par des institutions différentes

Motivations

- Agents itinérants
  - exemple : jointure entre deux sources de données

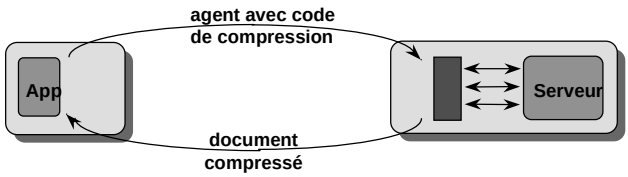


Motivations

- Agent d’adaptation
  - extension des fonctions du service
  - adaptation aux besoins spécifiques des clients
  - exemple : compression/chiffrement de documents
    - un client demande un document à un serveur
    - le client passe un algorithme de compression ou de chiffrement des données
    - le code de cet algorithme est propre au client

Motivations

- Agent d’adaptation
  - exemple : compression/chiffrement de documents



Définitions

- Agents
  - unité de structuration des applications
  - agents mobiles ou stationnaires
  - ressources allouées (contrôlées) aux agents
- Places
  - endroit que peut visiter un agent
  - plus fin que le site (différents services)
  - certaines places prédéfinies pour un agent (*home/proxy*)

Définitions

- Déplacement
  - entre des places
  - parfois notion d’itinéraire
  - permet de co-localiser des agents pour des interactions locales
- Interactions
  - entre des agents co-localisés (*meetings*)
    - appel de procédure ou d’objet
  - entre des agents distants : messages

Définitions

- Contrôle d’accès
  - autorité
    - associé à un agent ou une place
    - permet une authentification d’un agent ou d’une place
  - permis
    - des capacités associées à des agents ou des places
    - permet de limiter des droits d’accès

Plates-formes disponibles

- Sur Java
  - Aglets (IBM)
    - http://www.trl.ibm.co.jp/aglets
  - Odyssey (General Magic Inc.)
    - http://genmagic.com/technology/mobile_agent.html
  - Concordia(Mitsubishi)
    - http://www.meitca.com/HSL/Projects/Concordia
  - Voyager (Object Space)
    - http://www.objectspace.com/voyager
  - MOA (OSF/OpenGroup)
    - http://www.osf.org/RI/java/moa

Plates-formes disponibles

- Autres environnements (Tcl, Python ...)
  - AgentTcl (Dartmouth College)
    - http://www.cs.dartmouth.edu/~agent
  - Ara (Université de Kaiserslautern)
    - http://www.uni-kl.de/AG-Nehmer/Projekte/Ara
  - Tacoma (Université de Tromsø et Cornell)
    - http:/www.cs.uit.no/DOS/Tacoma

Aglets

- En général
  - un projet d’IBM Japon
  - conçu sur Java
- Les abstractions
  - Contexte : une place
  - Aglet : un agent mobile
    - *AgletIdentifier* : un identifiant unique
    - *AgletProxy* : un objet d’accès
  - Message
    - message asynchrone
    - appel de procédure synchrone ou différée

Un Aglet

- Une classe qui étend la classe *Aglet*
- Un graphe d’objets sérialisable
- Des méthodes prédéfinies de gestion des aglets
  - Aglet.dispatch (URL url)
  - Aglet.deactivate (long time)
  - Aglet.clone ()
  - Aglet.getAgletContext ()
- Des méthodes définies par le programmeur
  - Aglet.onCreation (Object init)
  - Aglet.run ()
  - Aglet.HandleMessage (Message msg)
  - Aglet.onDisposing ()

Un Aglet

```
public class Hello extends Aglet {
    public void onCreation (Object init) {
        System.out.println (“created!”);
    }
    public void run () {
        System.out.println (“hello!”);
    }
    public boolean handleMessage (Message msg) {
        if (msg.sameKind(“sayHello”)) {
            System.out.println (“hello!”);
            return true;
        }
        return false;
    }
    public void onDisposing () {
        System.out.println (“bye!”);
    }
}
```

Un proxy

- Un objet intermédiaire pour la manipulation des aglets
  - moyen d’appeler une méthode sur un aglet (pour déplacer, envoyer un message ...)
  - masque la localisation de l’aglet (non cohérent)
  - limite les méthodes appelables sur l’aglet
- Un aglet peut obtenir un proxy désignant un aglet
  - AgletContext.getAgletProxies()
  - AgletContext.getAgletProxy (AgletID)
  - ou en le recevant dans un message

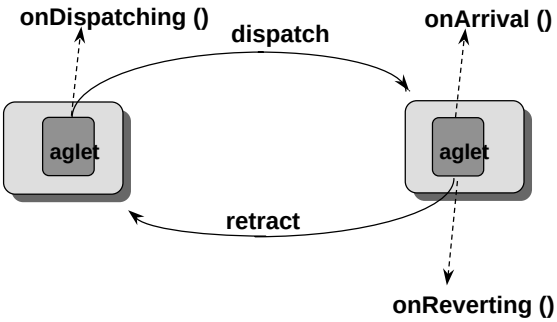
Un message

- Composé de deux champs
  - un type (*String*)
  - un objet
- Peut être passé de trois façons
  - appel synchrone
    - AgletProxy.sendMessage (Message msg)
  - appel à résultat différé
    - AgletProxy.sendFutureMessage (Message msg)
  - message asynchrone
    - AgletProxy.sendOnewayMessage (Message msg)

Programmation de la mobilité

- Pas de mobilité “forte” du code
- Programmation basée sur des événements
  - une gestion d’événement de mobilité associée à un Aglet (*MobilityListener*)
  - définies par le programmeur
    - onArrival (MobilityEvent l)
    - onDispatching (MobilityEvent l)
    - onReverting (MobilityEvent l)
- Même chose pour le clonage (*CloneListener*) et la persistance (*PersistencyListener*)

Programmation de la mobilité



Programmation de la mobilité

```
class myListener implements MobilityListener {
    public void onDispatching (MobilityEvent l) {
        System.out.println (“I am leaving!”);
    }
    public void onReverting (MobilityEvent l) {
        System.out.println (“I am going home!”);
    }
    public void onArrival (MobilityEvent l) {
        System.out.println (“I have arrived!”);
    }
}

public class MyAglet extends Aglet {
    public void onCreation (Object init) {
        MobilityListener listener = new myListener();
        addMobilityListener(listener);
    }
}
```


Programmation d’un contexte

- Programmation basée sur des événements (*ContextListener*)
  - événements concernant le contexte
    - contextStarted (ContextEvent ev)
    - contextShutdown (ContextEvent ev)
  - événements concernant les aglets
    - agletCreated (ContextEvent ev)
    - agletCloned (ContextEvent ev)
    - agletArrived (ContextEvent ev)
    - agletDispatched (ContextEvent ev)
    - agletReverted (ContextEvent ev)
    - agletStateChanged (ContextEvent ev)

Limitation des systèmes

- La mobilité faible du code impose une programmation fastidieuse
- La localisation des agents reste un problème difficile sur un réseau à grande échelle.
  - doit-elle être prise en charge entièrement par le système ?
- Peu de retour sur l’intérêt de la technologie
  - peu de grandes applications
  - pas de mesures

Expérimentation sur Java

- Avantages de Java
  - Diffusion sur toutes les machines
  - Mobilité du code et des objets
    - chargement dynamique de classes (ClassLoader)
    - sérialisation des objets Java
  - Sécurité assurée par l'environnement
    - confinement : pas d’adresses virtuelles
    - contrôle d'accès aux ressources (SecurityManager)
- Expérimentation
  - évaluation des mécanismes de base de Java
  - intérêt par rapport au RPC

Conception sur Java

- Sérialisation de l'état de l'agent
- Envoi du code et de l'état de l'agent
- Destruction de l'agent sur le site origine
- Création d'un *thread* sur le site destination
- Chargement du code de l'agent (chargeur privé)
- Dé-sérialisation de l'agent
- Exécution

Expérimentation : coûts de base

- Objectif : mesurer les coûts des principaux mécanismes de Java rentrant dans la conception
- Développement d'une plate-forme minimale
- Environnement :

Latence sur le réseau local	0.6 ms
Latence sur Internet	20 ms
Débit sur le réseau local	842 Ko/s
Débit sur Internet	129 Ko/s
Appel local de méthode Java	1 µs
Appel à distance de méthode Java sur le réseau local	3,1 ms
Appel à distance de méthode Java sur Internet	44 ms

Expérimentation : coûts de base

- Coûts des principales phases
  - coût de Java
    - prohibitif par rapport au réseau local
    - acceptable par rapport à l'Internet
  - grande partie provenant du chargeur privé

Sérialisation de l'agent	3,2 ms
Transfert de l'agent sur le réseau local	8 ms
Transfert de l'agent sur Internet	121 ms
Installation de l'agent avec le chargeur de Java	4.3 ms
Installation de l'agent avec un chargeur privé à l'agent	23,8 ms

Expérimentation : coûts de base

- Coût de la migration (plate-forme minimale)

Coût de la migration d'un agent minimal sur réseau local	35 ms
Coût de la migration d'un agent minimal sur Internet	148 ms

- Coût de la migration (Aglets)

Coût de la migration d'un Aglet minimal sur réseau local	240 ms
Coût de la migration d'un Aglet minimal sur Internet	369 ms

Expérimentation : intérêt/RMI

- Objectif : étudier l’intérêt de la technologie agents mobiles en utilisant
  - le modèle client serveur (Java-RMI)
  - une plate-forme minimal que nous avons réalisée
  - une plate- forme très répandue (Aglets)
- Résultats visés
  - comparaison entre le modèle client-serveur et le modèle à agents mobiles
  - comparaison entre une plate-forme minimale et une vrai plate-forme à agents mobiles

Expérimentation : intérêt/RMI

- Trois machines réparties sur Internet
  - en France (INRIA)
  - en Grande-Bretagne (QMW)
  - en Suisse (Université de Genève)
- Aglets
- Java-RMI
- Notre plate-forme minimale

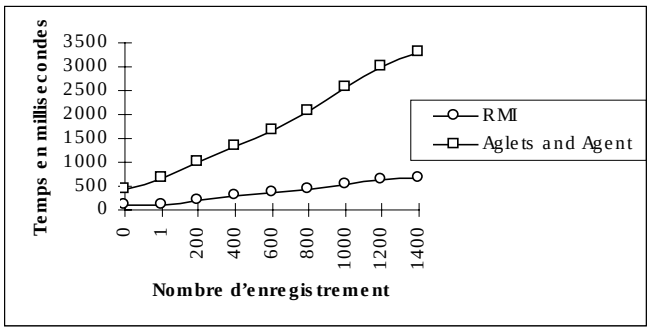
Expérimentation : intérêt/RMI

- Capacités du réseau

Connexion	Latence (ms)	Débit (Ko/sec)
France - GB	20	129
GB - Suisse	16	280
Suisse - France	26	123

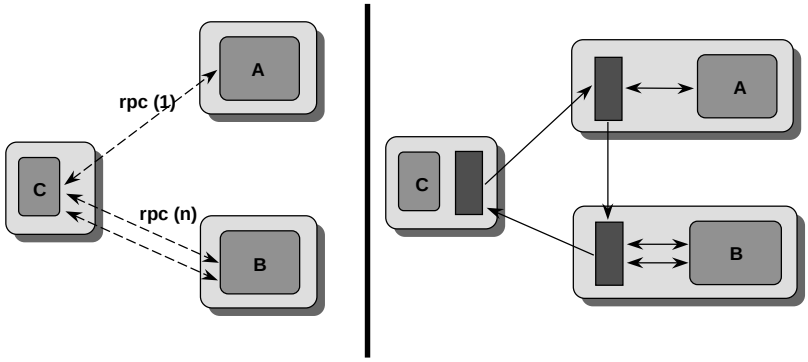
Expérimentation : intérêt/RMI

- Comparaison aller-retour avec agent mobile et RMI
  - Taille enregistrement : 80 octets
  - entre France et GB



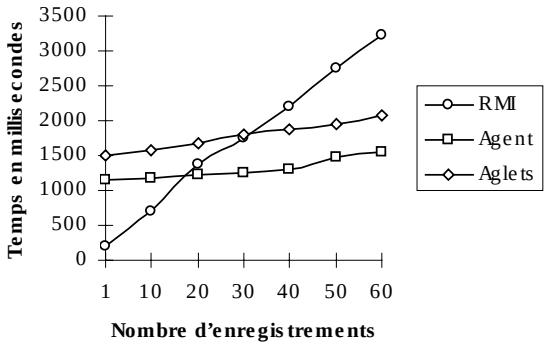
Expérimentation : intérêt/RMI

- Application de jointure
  - $rpc_{C\text{-}A}(n) + n*rpc_{C\text{-}B}(1) > Ag_{C\text{-}A}(0) + Ag_{A\text{-}B}(n) + Ag_{B\text{-}C}(2*n)$
  - Ag(n) augmente plus vite que rpc(n) ...



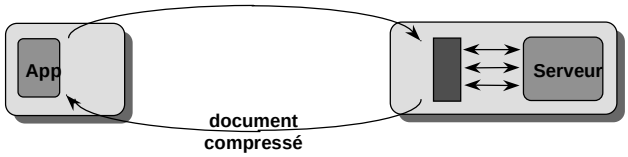
Expérimentation : intérêt/RMI

- Application de jointure : résultats



Expérimentation : intérêt/RMI

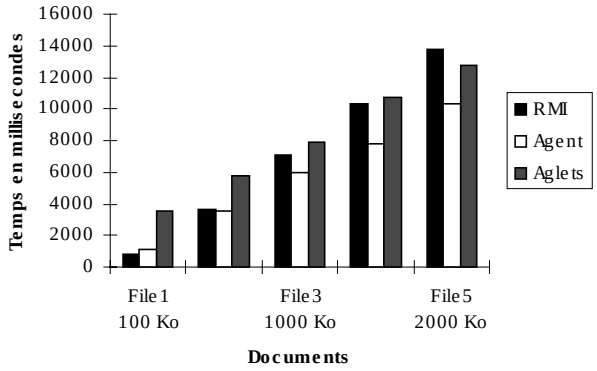
- Application de compression



  - La compression est potentiellement rentable si
    - tc + fc * S/dr + td < S/dr
    - ce qui donne :
      - dr < 74 Ko/s pour un fichier de 100 Ko
      - dr < 164 Ko/s pour un fichier de 500 Ko
  - Mais le coût d’un transfert par agent augmente plus vite ...

Expérimentation : intérêt/RMI

- Application de compression : résultats



Conclusion

- Conclusion
  - agents mobile : nouveau modèle basé sur la mobilité
  - Java fournit les outils pour le support d'agents mobiles
  - intéressant pour des réseaux lents ou lors de déconnexions
- Limites et perspectives
  - intégration avec les autres modèles
  - transparence de la programmation
    - migration faible/forte
    - utilisation de ressources partagées
    - localisation des objets

Bibliographie

- J.E. White, Telescript Technology: The Foundation for the Electronic Market-place, *General Magic Inc.*, Mountain View, CA, 1994.
- W. T. Cockayne et M. Zyda, Mobile Agents, Edition Manning, 1998.
- D. B. Lange et M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Edition Addison-Wesley, 1998.
- L. Ismail et D. Hagimont, A performance evaluation of the mobile agent paradigm, *ACM Conference on Object-Oriented Programming, Systems, Languages and Application (OOPSLA)*, Novembre 1999.