Chapitre1
Introduction aux systèmes répartis

Introduction

Les systèmes distribués ont vu le jour avec l'arrivée des réseaux informatiques qui permettent la connexion des ordinateurs entre eux.

Dans un système centralisé:
Tout est localisé sur la même machine,

- 1 processeur : une horloge commune
- 1 mémoire centrale : un espace d'adressage commun
- 1 système d'exploitation
- Gestion centralisée des ressources
- Etat global du système facilement reconstituable
- Accès local aux ressources

Introduction

Les progrès technologiques :

- Des réseaux,
- Des ordinateurs et
- Des support de communication

Permettent aux différentes applications sur des ordinateurs distincts et distants de coopérer pour réaliser des tâches coordonnées

Définition

❖ Un ensemble de machines connectées par un réseau, et équipées d'un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ses ressources. "

« Coulouris et al. »

❖Un système réparti est un système qui s'exécute sur un ensemble de machines sans mémoire partagée, mais que pourtant l'utilisateur voit comme une seule et unique machine.

« Tanenbaum »

Définition

❖ Un système réparti est un système informatique dans lequel les ressources ne sont pas centralisées.

- ✓ les moyens de stockage (données, fichiers)
- ✓ la charge CPU
- ✓ les utilisateurs
- ✓ les traitements …

❖ Systèmes répartis : le fruit du mariage entre l'informatique et les moyens de télécommunication (réseau) système réparti système distribué

❖Des systèmes distribués aux systèmes à agents mobiles (à large échelle dans un contexte réseau dynamique)

❖Les calculs répartis : une autre dimension des systèmes répartis

❖La mondialisation de l'informatique + Internet = répartir l'architecture d'un système d'information

Définition

❖ La problématique de la répartition est née avec l'idée de faire communiquer des ordinateurs via un réseau de communication

❖ Modélisation : les ressources (ordinateurs) sont les "nœuds" du modèle et la communication se fait par échange de message

Nœud -> serveur -> ressources -> protocole de répartition -> Middleware

❖ Les middlewares = moyens logiciels de mise en œuvre des applications distribués (RMI, CORBA, …)

Domaines d'application

- Mathématique, algorithme, "puissance" de calcul Innovations (agents, collaboration, IA, …)

- Systèmes d'information d'entreprise

- Les particuliers: partager ses données, jeux, accès et collaboration de services communs, …

- Sites Internet: un site est un lieu de partage de l'information et de distribution de service et bien

- Le monde des systèmes répartis est en constantes évolutions : les réseaux évolues, les techniques logicielles sont en constantes révolutions, les performances sont en constante croissance.

Concepts des systèmes répartis

- L'évolution et les performances actuelles des réseaux informatiques permettent de mettre en pratique les concepts de la répartition dans des applications informatiques de tous les jours

- En quoi la programmation d'application répartis se distingue d'une application centralisée

  - PAS D'ETAT GLOBAL
  - PAS D'HORLOGE GLOBALE
  - FIABILITE RELATIVE
  - SECURITE RELATIVE

Pas d'état global :

✓ Un nœud ne possède pas une connaissance immédiate exacte de l'état d'un autre nœud

✓ Cette connaissance passe par l'échange d'un message qui introduit obligatoirement un délai

=>

Solution: utilisation des algorithmes qui permettent de construire des clichés globaux qui représentent un état passé possible de l'application

Pas d'horloge global :

✓ Horloge propre à chaque nœud (ordinateur) pas de synchronisation des horloges

✓ L'ordre des évènements répartis dans l'application n'est pas déductible à partir des datations locales

=>

définir des datations logiques

Fiabilité relative :

✓ Malgré les solutions de tolérance à la panne utilisées:

- Réplication
- Délocalisation

✓ La défaillance d'un nœud augmente avec leur nombre.

Sécurité relative :

✓ Difficulté de protéger une architecture répartie contre les intrusions.

✓ Les nœuds peuvent être dynamique => architecture réseau variable donc difficile à protéger

Avantages des systèmes répartis

- **Partage et Mise à disposition:**
  Service de gestion de fichiers partagés et répartis

- **Répartition géographique:**
  Système de réservation d'hôtel répartis en différents pays, compagnies ou agences.
  Système bancaire avec ses agences régionales et son siège social

- **Puissance de calcul:** le calcule parallèle

- **Disponibilité:** la réplication d'un même service

- **Flexibilité:** l'ajout ou la suppression d'un site est une opération simple, et n'influe pas sur le fonctionnement total du système

Principes de conception

Un bon système réparti est un système qui masque le plus possible les propriétés délicates de la répartition.

Mais, il est impossible d'être transparent sur tous les aspects Les propriétés de transparence :

- ✓transparence d'accès
- ✓transparence de localisation
- ✓transparence du partage
- ✓transparence de la réplication
- ✓transparence des pannes
- ✓transparence de la migration
- ✓transparence de charge
- ✓transparence d'échelle

Principes de conception

Transparence d'accès

✓ L'interface d'accès à une ressource ne doit pas dépendre à l'emplacement de la ressource (locale ou distante)

✓ La syntaxe utilisée ne doit pas être différente

✓ Très souvent difficile à mettre en place en fonction de la nature de la ressource

✓ Exemple:

- L'accès un fichier (local ou distant) grâce à la transparence obtenue par l'utilisation de NFS (network file system)
- L'accès à l'interface d'un objet distribué (méthode distante)

Transparence de localisation

✓ La désignation de la ressource est indépendante de la localisation de cette ressource

✓ Nécessite un service de nommage global (annuaire) exemple : utilisation des "Naming services" de la norme CORBA pour les objets distribués

✓ Nécessite une connaissance de l'architecture du réseau

Principes de conception

Transparence du partage

✓ Les accès concurrents à la ressource sont contrôlés de telle façon que l'intégrité de la ressource est assurée

✓ Synchronisation des accès en lecture et écriture de la ressource

✓ Nécessite de décrire les traitements informatiques sous la forme de "transaction" pour garantitr l'intégrité de la données

- exemple des transactions dans des bases de données réparties
- exemple de l'enchaînement des méthodes synchronisées d'un objet distribué

Principes de conception

Transparence de la réplication

✓ Consiste à s'assurer que l'accès à une ressource soit identique quel que soit la forme d'implantation d'une ressource répliquée

✓ Utilisé pour augmenter la tolérance aux pannes

✓ L'accès à une ressource répliquée est indépendante de l'indisponibilité d'une de ses occurrences

✓ La mise en œuvre de la réplication est un exercice difficile. Mais facilité par l'usage de mécanismes de réplication intégrés comme ceux des SGBD

✓ La réplication permet un routage des transactions (lecture / écriture)

Principes de conception

Transparence des pannes

✓ Détection de la défaillance des nœuds du système réparti

✓ Pallier à cette défaillance d'une manière transparente

✓ Tolérance aux pannes de l'ensemble des services d'un système réparti ne pas bloquer le fonctionnement global de l'application

✓ La réplication permet une transparence de l'indisponibilité d'une ressource et/ou d'un service

Principes de conception

Transparence de la migration

✓ La migration consiste à assurer qu'une ressource peut migrer d'un nœud à un autre sans que les usagers s'en aperçoive.

✓ Déplacer un service dynamiquement vers un serveur moins chargé

✓ Mettre en œuvre des stratégies de régulation de charge sur une architecture répartie

Principes de conception

Transparence de charge

✓ L'équilibrage de la charge de chaque nœud peut permettre une meilleure exploitation du système global et une meilleure satisfaction des utilisateurs.

✓ La mise en œuvre est délicate car une partage de la charge globale implique une bonne connaissance de l'état global du système (difficile à obtenir)

✓ Nécessite la mise en place d'algorithmes répartis spécifiques

✓ Nécessite une bonne connaissance des contraintes du réseau utilisé

Principes de conception

Transparence d'échelle

✓ Un changement d'échelle d'une application réparti consiste à augmenter le nombre de nœud

✓ceci est facilité par la propriété modulaire et dynamique d'une architecture répartie

✓ L'ajout de nœuds ne doit pas nécessiter l'arrêt du système

✓ce changement d'échelle n'est pas forcement transparent pour les usagers, il faut donc le contrôler et le rendre le plus transparent possible.

Principes de conception

Transparence d'échelle

✓ Un changement d'échelle d'une application réparti consiste à augmenter le nombre de nœud

✓ceci est facilité par la propriété modulaire et dynamique d'une architecture répartie

✓ L'ajout de nœuds ne doit pas nécessiter l'arrêt du système

✓ce changement d'échelle n'est pas forcement transparent pour les usagers, il faut donc le contrôler et le rendre le plus transparent possible.

Exemple: Propagation des états

CONTEXTE :

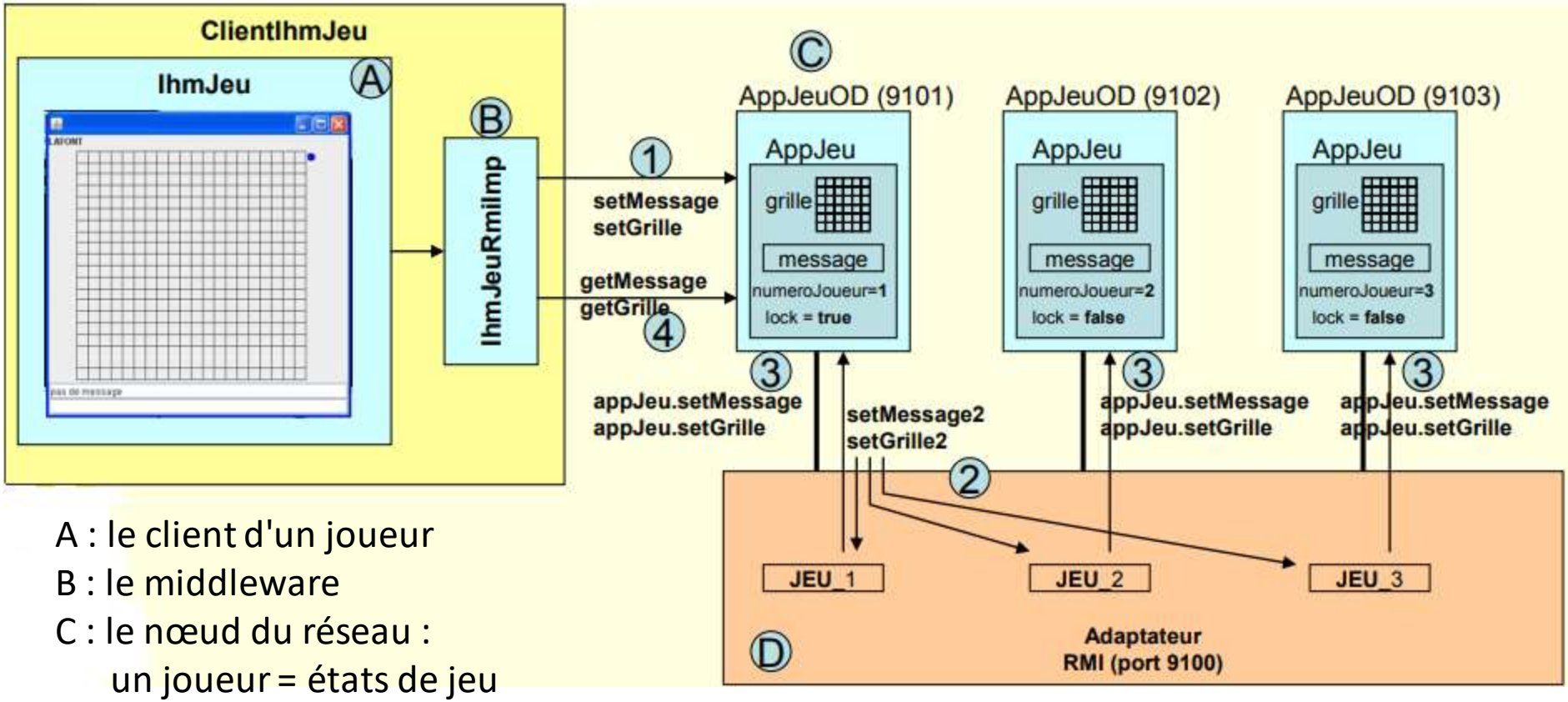
Les nœuds du réseau contiennent de l'information communes mais répartis, exemple :

✓ Un joueur gère une grille de jeu et un message de communication…

✓Chaque joueur modifie sa grille de jeu et/ou son message

✓Tous les joueurs connectés sont mises à jour par propagation des setteurs de l'objet (nœud du réseau) : la grille et le message

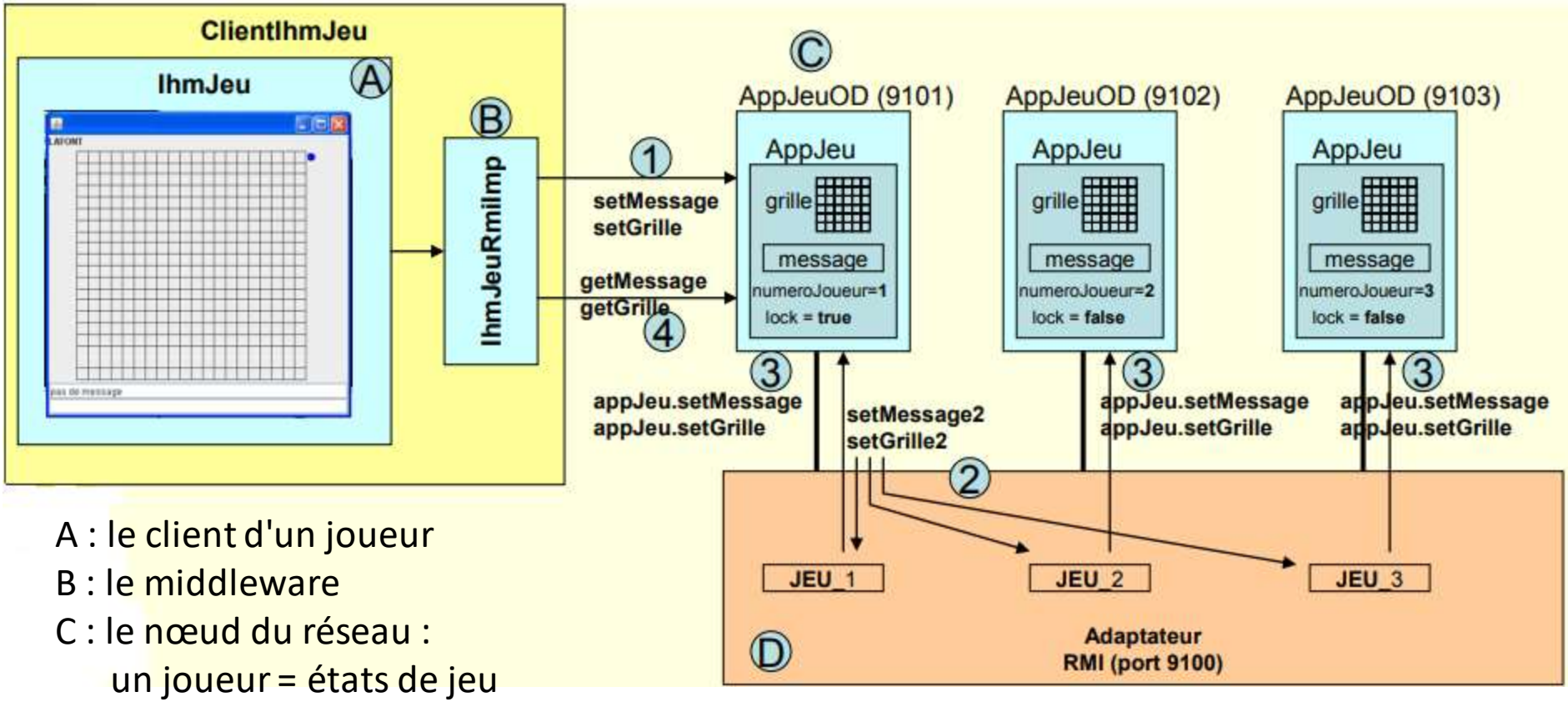
Exemple: Propagation des états



A : le client d'un joueur
B : le middleware
C : le nœud du réseau :
un joueur = états de jeu
D : le service de nommage

1/ via le client l'état de jeu d'un joueur évolue : valeurs de la grille et valeur du message
2/ via le service de nommage : propagation des setteurs
3/ changement des états de jeu de tous les joueurs : appel aux setteurs
4/ le client récupère les états de jeu : appel aux getteurs

Exemple: Propagation des états



A : le client d'un joueur
B : le middleware
C : le nœud du réseau :
un joueur = états de jeu
D : le service de nommage

1/ via le client l'état de jeu d'un joueur évolue : valeurs de la grille et valeur du message
2/ via le service de nommage : propagation des setteurs
3/ changement des états de jeu de tous les joueurs : appel aux setteurs
4/ le client récupère les états de jeu : appel aux getteurs