Synchronisation Distribuée
Contexte et modèles

Laurent PHILIPPE

Master 2 Informatique ISL
Ingénierie des Systèmes et Logiciels

2021/2022

UNIVERSITÉ DE FRANCHE-COMTÉ

femto-st SCIENCES & TECHNOLOGIES

Sommaire

Sommaire

Sommaire

Système Centralisé

Caractérisation

Ensemble de composants (cœurs, processeurs, périphériques) partageant une mémoire et une horloge

Propriétés

- Possibilité de dater de manière non ambiguë les évènements
- Possibilité d'accéder à une donnée commune
- Partage de ressources : processeur, mémoire, disque, périphérique, ...
- Programmation concurrente synchronisée : sémaphore, mutex
- Exemple : problème de synchronisation en Java multi-thread

Système Distribué

Caractérisation

Ensemble de composants (ordinateurs, téléphones, IoT, ...) autonomes reliés par un réseau de communication, sans partage.

Programmation distribuée

- Pas d'état global : un processus ne peut pas connaître instantanément l'état des autres processus
- Pas d'horloge globale : chaque processus a sa propre horloge
- Échanges : communication et synchronisation, uniquement par messages
- Non déterminisme : deux exécutions donnent des résultats différents
- Risque de perte de messages et fautes

Systèmes Distribués : avantages / inconvénients

Avantages

- Accès et partage de ressources distantes : distribution permet la mise à disposition de services
  - ressources physiques : imprimantes, espace disque, processeurs, etc
  - ressources logiques : fichiers, données textuelles, audio, images, vidéo
- Répartition géographique : système de réservation avec des centres dans différents lieux
- Puissance de calcul/stockage : connexion de machines en réseau permet d'obtenir une plus grande puissance de calcul/stockage
- Disponibilité : relative indépendance des défaillances des différents processuss
- Flexibilité : architecture distribuée plus modulaire

Systèmes Distribués : avantages / inconvénients

Inconvénients

- Pas d'horloge globale
- Pas d'état global immédiat accessible à un processus
- Non déterminisme : debug difficile
- Fiabilité relative : distribution permet d'introduire de la tolérance aux fautes
- Sécurité relative : architecture distribuée plus difficile à protéger (plusieurs points d'accès aux ressources, évolution dynamique de l'architecture)

Sommaire

Les systèmes de calculs distribués

Les clusters de calcul

- Ensemble de nœuds identiques ayant le même OS connectés par un réseau local haut débit et faible latence (ex : InfiniBand)
- Architecture caractérisée par une certaine homogénéité
- Utilisés pour la programmation parallèle : un même programme est exécuté en parallèle sur plusieurs machines. (SPMD)
- Application : calcul intensif

Les systèmes de calculs distribués

Les clouds / data centres

- Système composé d'un ensemble de machines qui sont ou ne sont pas dédiées
- Architectures hétérogènes : au niveau hardware, système ....
- Calcul, données, applications, etc.
- Allocation à la demande de VM
- Réseau haut débit mais latence plus élevée

Les systèmes d'information distribués

Systèmes orientés données

- Les systèmes à base de transactions : systèmes de gestion de bases de données distribués.
- Les EAI (Enterprise Application Integration) : coordination de différents systèmes
- Systèmes coopératifs
- Midleware : gestion coordonnées de systèmes
- WEB Services : accès à des serveurs distants

Stabilité des systèmes précédents

Propriétés

- Nœuds fixes
- Nœuds fiables : peu de fautes
- Connexion permanente et de haute qualité

Les systèmes mobiles et "pervasifs"

Exemples

- Ordinateurs portables, téléphones, etc.
- Robots, drônes, etc
- Réseaux de capteurs : collecte dans un environnement, surveillance médicale, ...
- Réseaux diffus : domotique, réseaux had-hoc, ...

Instabilité

- Nœuds peuvent apparaître/disparaître
- Pas toujours de connexion réseau
- Mobilité des nœuds
- Systèmes peu traités dans ce cours

Sommaire

Centralisation et systèmes distribués

Centralisation

- Mise en place d'un maître : processus spécifique ou parmi les participants
- Le maître prend les décisions
- Retour à un système centralisé : prise de décision unique
- Cas des systèmes client-serveurs

Attention confusion

- Système centralisé, une seule machine
- Algorithme centralisé dans un système distribué

Distribué ou centralisé ?

Centralisé souvent plus facile mais...

- Concentration en un point conduit à la surcharge
- Fragilité face à la tolérance aux pannes
- Pas toujours adapté aux besoins (ex : traitements locaux)

Distribué souvent plus résistant mais...

- Difficulté de mise en œuvre
- N'évite pas tous les risques de surcharge (ex : diffusion systématique de messages)
- Preuve complexe

Distribué ou centralisé ?

Les deux concepts ne sont cependant pas opposés

- Modèles mixtes :
  - Hiérarchique (ex : un maître délègue aux serveurs secondaires)
  - Redondance (ex : serveur de backup)
  - Super-nodes
- Choix dépend des difficultés de réalisation : pragmatisme

Sommaire

Modèle conceptuel de système distribué

Objectifs

- Modèle
- Description statique et comportementale
- Abstraction pour faciliter l'analyse
- Validation et preuve de propriétés

Les éléments de modélisation

- Processus, sites, nœuds : notion de processus logique
- Communication : liens, topologie, protocoles (point à point, diffusion), ...
- Connaissance partielle de chaque processus logique

Sommaire

Processus logique

Définitions

- Élément logiciel effectuant une tâche donnée
- Tâche : éxécution d'un ensemble d'instructions
- L'éxécution de la ième instruction au sein du processus p génère un **évènement**, noté e_p^i
- Les **évènements** considérés sont :
  - changement d'état du processus
  - émission de message
  - réception de message

Exemples

Robots, ordinateurs, téléphones, ...

Processus logique

Possède

- Un état local, noté c_p^i
- Une mémoire locale
- Une horloge locale
- Des procédures de communication

Connaît par hypothèse

- Un identifiant unique, p
- Les processus avec qui il peut communiquer et leur nombre : voisins, successeur, etc.

Sommaire

Modèle de système distribué

Graphe de processus

- Graphe : $G(\mathbb{P}, \mathbb{C})$
- Sommets $= \mathbb{P}$ ensemble de processus
- Arcs = liaisons de communication entre les processus, $\mathbb{C}$ ensemble de liens
- Arc entre P_i et P_j signifie que P_i et P_j peuvent communiquer sans intermédiaire

La communication

Topologie

Elle a une incidence sur la façon dont les processus peuvent communiquer entre eux, donc sur l'algorithme et le nombre de messages échangés :

- Diffusion : un seul message pour tous les autres processus,
- Sans fil/Graphes : ne peut pas contacter tous les processus, prévoir routage
- Anneau : limitation du nombre de messages, intéressant pour limiter la consommation de batterie
- Arbre/Topologies régulières : routage dépendant de la topologie

Modèles de communication

Protocoles de communication

- Point à point : envoi d'un message à un destinataire explicite
- Diffusion : envoi d'un même message à tous les destinataires ou seulement à un groupe de destinataires

Fonctions de communication

- Point à point : envoi d'un message à un destinataire explicite
  - Différents types de messages (ACK, REQ, etc.)
  - Envoyer : P_i **envoie** *TYPE*(*parametres*) *à destinataire*
  - Recevoir : P_i **reçoit** *TYPE*(*parametres*) *de émetteur*
- Diffusion : envoi d'un même message à tous les voisins

Sommaire

Modèle d'exécution distribuée

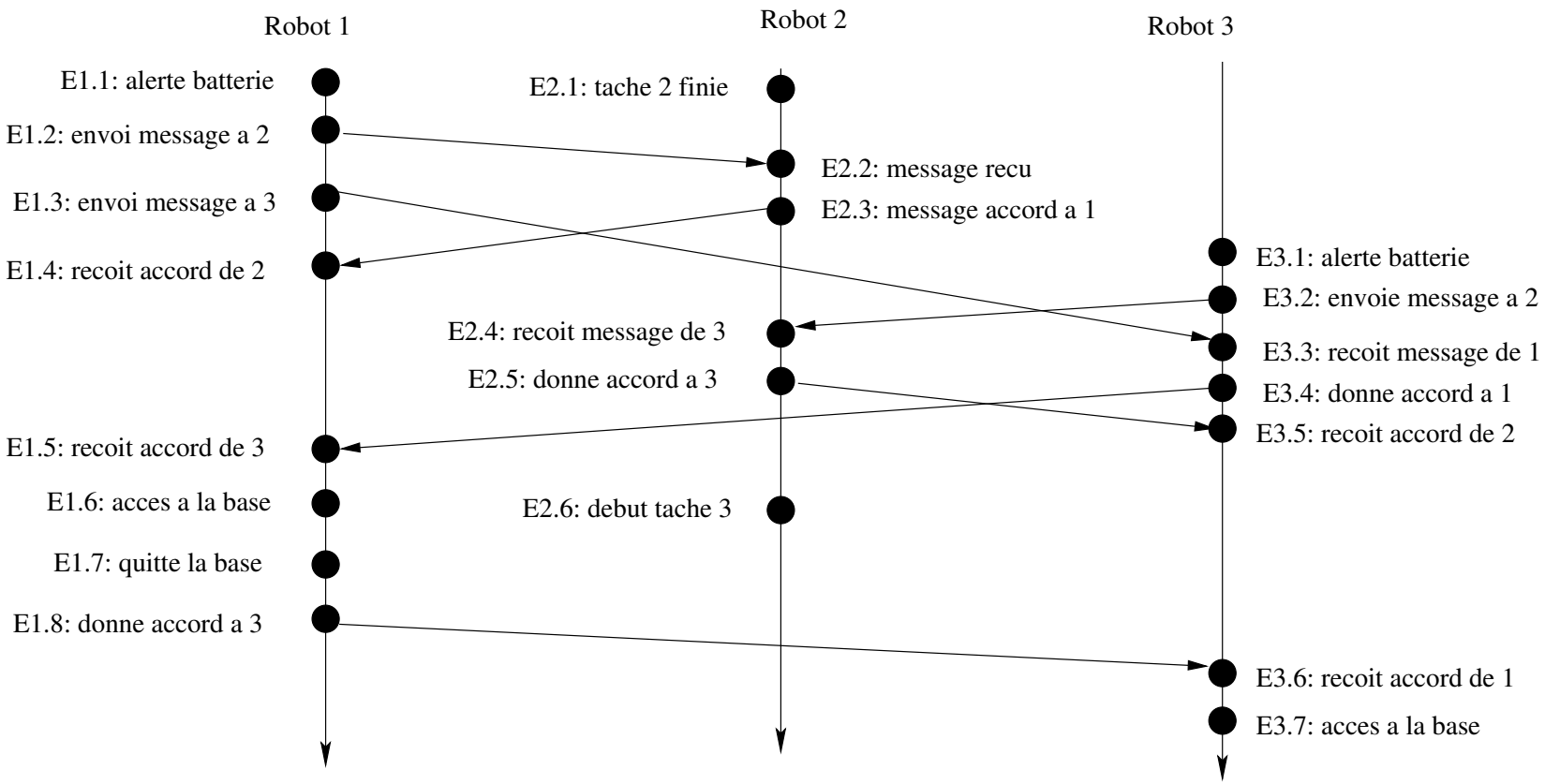
Chronogramme

- Représentation graphique
- Ordre des évènements
- Déroulement chronologique sur chaque processus
- Représentation simple mais limitée

Évènements

- Émission de message
- Réception de message
- Évènement interne

Exemple de chronogramme



Modèle d'exécution distribuée

Modèle des évènements

- Pour une exécution $\mathcal{C}$ (calcul) distribuée d'un ensemble $\mathbb{P}$ de processus
- L'ensemble des évènements est noté E_v,
- L'exécution locale d'un processus p est composée d'une séquence $c_p^0, c_p^1, c_p^2, ...$ d'états du processus, où c_p^0 est l'état initial.
- La transition de l'état c_p^{i-1} à l'état c_p^i du processus p est marquée par l'occurrence de l'évènement e_p^i
- On définit $E_v = \bigcup_{p \in \mathbb{P}} \{e_p^0, e_p^1, e_p^2, ...\}$

Modèle d'exécution distribuée

Modèle des communications

- Un évènement peut-être interne, une émission ou une réception
- L'état c_p^i contient donc l'ensemble des variables du processus dans l'état i et l'historique complète des communications :
  - La liste $sent_{pq}^i$ des messages envoyés de p à q parmi les évènements e_p^0 à e_p^i,
  - la liste $rcvd_{pq}^i$ des messages que q a reçu de p parmi les évènements e_p^0 à e_p^i,

Modèle d'exécution distribuée

P_0	0								
jeton	F								
SC	F								

P_1	0								
jeton	F								
SC	F								

P_2	0								
jeton	F								
SC	F								

Num	Action

Modèle d'exécution distribuée

P_0	0								
jeton	F								
SC	F								

P_1	0								
jeton	F								
SC	F								

P_2	0								
jeton	F								
SC	F								

Num	Action
1	P_0 envoie *JETON* à P_1
2	P_1 envoie *JETON* à P_2
3	P_2 envoie *JETON* à P_1

Modèle d'exécution distribuée

P_0	0								
jeton	F								
SC	F								

P_1	0	4							
jeton	F								
SC	F	T							

P_2	0								
jeton	F								
SC	F								

Num	Action
1	P_0 envoie *JETON* à P_1
2	P_1 envoie *JETON* à P_2
3	P_2 envoie *JETON* à P_1
4	P_1 demande la Section Critique

Modèle d'exécution distribuée

P_0	0								
jeton	F								
SC	F								

P_1	0	4	6						
jeton	F		T						
SC	F	T							

P_2	0								
jeton	F								
SC	F								

Num	Action
1	P_0 envoie *JETON* à P_1
2	P_1 envoie *JETON* à P_2
3	P_2 envoie *JETON* à P_1
4	P_1 demande la Section Critique
5	P_0 envoie *JETON* à P_1
6	P_1 accède à la Section Critique

Modèle d'exécution distribuée

P_0	0								
jeton	F								
SC	F								

P_1	0	4	6	8					
jeton	F		T	F					
SC	F			F					

P_2	0								
jeton	F								
SC	F								

Num	Action
1	P_0 envoie *JETON* à P_1
2	P_1 envoie *JETON* à P_2
3	P_2 envoie *JETON* à P_1
4	P_1 demande la Section Critique
5	P_0 envoie *JETON* à P_1
6	P_1 accède à la Section Critique
7	P_1 quite à la Section Critique
8	P_1 envoie *JETON* à P_2

Modèle de système distribué

Restrictions

Le modèle utilisé peut inclure des hypothèses. Par exemple :

- Nombre de processus est constant
- Topologie de communication est fixe
- Aucun processus n'est isolé
- Pas de perte de message
- Connaissance d'un ID

Ces hypothèses définissent la portée de l'algorithme

Sommaire

L'algorithmique distribuée

Qu'est ce que c'est ?

- Algorithmique
- Point de vue modélisé
- Spécifique aux problèmes soulevés par les systèmes distribués

Caractéristiques

- Programmes + Communication par messages
- Pas de donnée partagée : chaque programme défini ses propres données
- Exécution : déroulement en parallèle des programmes, synchronisation/échanges sur les messages
- Preuve

Modèle algorithmique

Algorithme des processus

- Définir quel processus exécute quel algorithme
- Algorithme du processus décomposé en :
  - Données, les données locales aux processus, valeur d'initialisation
  - Messages, les messages utilisés (envoyés/reçus) par le processus, peuvent contenir des données si besoin
  - Règles, les fonctions du processus. Composées en général de :
    - Règle d'initialisation, processus initiateur
    - Règles internes, liées aux évènements interne (ex : batterie vide)
    - Règles de réception des messages, sur le mode message

  En général on fait l'hypothèse du traitement atomique, au moins sur les règles de réception des messages

Modèle algorithmique

Modèle de communication

- Définition de règles associées aux réceptions

Propriétés

- Synchronisation dépend des propriétés des fonctions de communication
- Peuvent être définies spécifiquement (précisé)
- Mais généralement asynchrone : envoi du message sans attendre sa réception

Modèle algorithmique

Exemple : diffusion dans un anneau

Variables pour chaque processus P_i :
$Succ_i$: Successeur de P_i dans l'anneau

Messages utilisés :
$BROADC(P, Msg)$: Message de diffusion

Règle 1 : *P_i demande de diffuser Msg sur l'anneau*
début
  P_i **envoie** $BROADC(i, Msg)$ à $Succ_i$

Règle 2 : *P_i reçoit $BROADC(init, Msg)$ de P_j*
début
  if $init != i$ **then**
    P_i **envoie** $BROADC(init, Msg)$ à $Succ_i$

Sommaire

Performance d'un algorithme distribué

Deux points de vue sur la performance

Comme en centralisé on cherche à évaluer l'efficacité

1. Locale au processus
   - Complexité classique de l'algorithme
2. Complexité distribuée
   - Nombre de messages, a une incidence sur la charge du réseau ou la consommation énergétique
   - Nombre de processus impliqués, a une incidence sur la charge des processus
   - Temps d'exécution de l'algorithme, dépend par exemple de l'interrogation séquentielle ou parallèle des autres processus

Choix de complexité dépend des propriétés du système : performances réseau, topologie, etc.

Propriétés d'un algorithme distribué

Propriété de sûreté

- Un état catastrophique ne sera jamais atteint
- Exemple : accès en exclusion mutuelle où une section critique est accessible par au plus un processus

Propriété de vivacité

- Un état de satisfaction sera fatalement atteint
- Exemple : toute demande d'accès à la section critique sera satisfaite ou encore tout message émis sera reçu

Propriétés d'un algorithme distribué

Propriété d'équité

- Garantir que les processus ont tous les mêmes chances d'accès à une ressource
- Exemple : il n'y a pas un processus qui accède plus à une section critique

Propriété de terminaison

- L'algorithme distribué se termine ou conduit à une décision
- Exemple : cas de l'élection d'un maître

Propriété de ponctualité

- Des contraintes temporelles peuvent être fixées
- Exemple : une échéance de remise de messages au plus tard

Suite du cours

Objectifs et contenu

- Première approche : comprendre les problèmes
- Connaître les algorithmes : cours / déroulements
- Savoir les appliquer à différents problèmes
- Savoir concevoir un algorithme : exercices
- Cas de l'exclusion mutuelle