Module BDR
Master d'Informatique (SAR)

Cours 8- Transactions réparties

Stephane Gançarski
Stephane.Gancarski@lip6.fr

Transactions réparties

Gestion de transactions

Transactions dans un système réparti

Protocoles de verrouillage

Concepts des moniteurs transactionnels

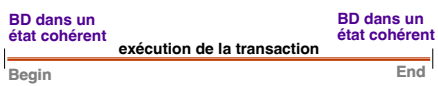
Standardisation des systèmes transactionnels

Exemples de moniteurs

Concept de transaction

Une transaction est une collection d'actions qui transforment la BD (ou des fichiers) depuis un état cohérent en un autre état cohérent

- BD cohérente (garantie par le système)
- transaction cohérente (garantie par le programmeur)



Exemple de transaction

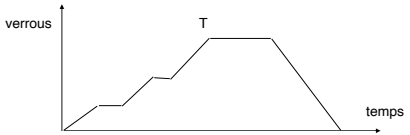
Réduire la cde no 10 de 5 unités et les reporter à la cde 12

```
Transaction   Report-qté
begin
   exec sql   UPDATE Cde
              SET    qte = qté - 5
              WHERE  ncde = 10;
   exec sql   UPDATE Cde
              SET    qté = qté + 5
              WHERE  ncde = 12;
   exec sql   COMMIT WORK;
end.
```

Problèmes liés à la répartition

- Les mises à jour sont effectuées sur TOUS les sites, ou sur AUCUN => nécessité de coordonner.
- Items locaux (physique) et items globaux (logique).
- Transaction globale et transaction locale : le système doit garantir la sérialisabilité globale.
  - Verrous globaux
  - Graphe de précédence réparti
- Verrouillage
  - centralisé vs décentralisé
  - verrous logiques et verrous physiques
  - interblocage intersites (difficile à détecter, prévention)

Isolation par verrouillage 2 Phases



- Les verrous en lecture sont partageables; ceux en écriture sont exclusifs
- Règle 1
  - avant d'accèder à un granule x, une transaction doit acquérir un verrou sur x. Si x est déjà verrouillé de façon exclusive, la transaction attend
- Règle 2
  - dès qu'une transaction relâche un verrou, elle ne peut plus acquérir de nouveau verrou

Performances

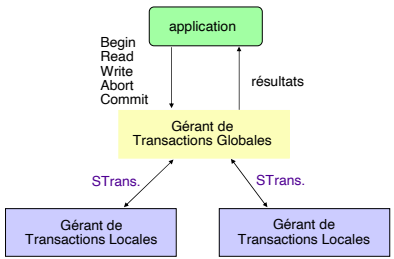
L'objectif est de réduire:

- les blocages
  - une transaction attend qu'une autre transaction relâche ses verrous
- les inter-blocages
  - un ensemble de transactions attend que l'une d'entre elles relâche ses verrous : « circuit d'attente »

Degrés d'isolation

- Degré 0
  - verrou en écriture sur x relâché après écriture de x (verrous courts)
  - pas d'écritures "sales"
- Degré 1
  - verrouillage en écriture à deux phases (verrous longs en écriture)
  - pas de pertes de mise à jour
- Degré 2
  - verrou en lecture sur x relâché après lecture de x (verrou court en lecture)
  - pas de lectures "sales"
- Degré 3
  - verrouillage en lecture à deux phases (verrous longs en lecture)
  - les lectures sont répétables, sérialisabilité

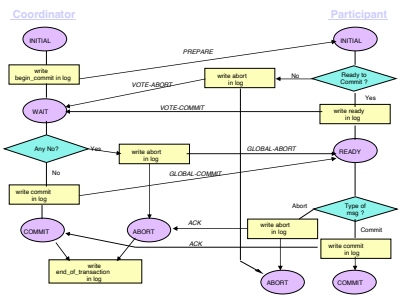
Gestion de transactions réparties



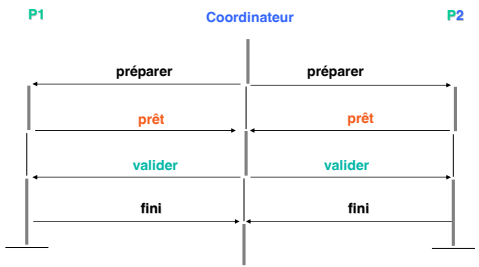
Protocole de validation en deux étapes (2PhaseCommit)

- Objectif : Exécuter COMMIT pour une transaction répartie
- Phase 1
  - Préparer à écrire les résultats des mises à jour dans la BD
- Phase 2
  - Ecrire ces résultats dans la BD
- Coordinateur
  - composant système d'un site qui applique le protocole (lance la transaction, la découpe en sous-transactions à exécuter sur les différents sites, coordonne la terminaison de la transaction)
- Participant
  - composant système d'un autre site qui participe dans l'exécution de la transaction

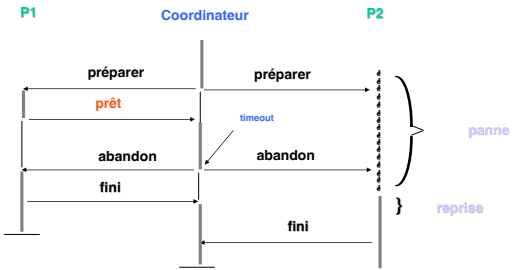
Actions du protocole



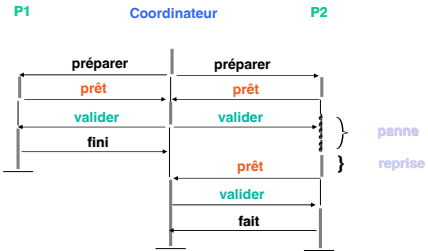
Validation normale



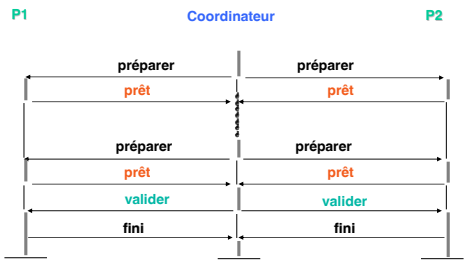
Panne d'un participant avant d'être prêt



Panne d'un participant après s'être déclaré prêt



Panne du coordinateur



Protocoles de verrouillages répartis

Traduire le verrouillage logique en verrouillages physiques.

1. Pas de réplication :
   - gestionnaire de verrous local sur chaque site
     - \+ facile à implémenter
     - \+ peu de messages (2 pour verrouiller, 1 pour libérer)
     - \- gestion des interblocages complexe
2. Avec réplication :
   - plusieurs méthodes, dont le coût (nb de messages) dépend du nombre de copies, du rapport lectures/écritures, de la concurrence (verrous refusés)

Méthode du nœud central

Un seul gestionnaire de verrous, sur un seul site.

Table de verrous logiques.

Si un verrou est accordé, la transaction peut lire la donnée sur n'importe quel site comportant une copie. Pour l'écriture, tous les sites sont impliqués.

+ peu de messages (3)
+ Implémentation simple
+ gestion des interblocages simple (algos classiques)
- goulot d'étranglement
- vulnérable : blocage si le site du gestionnaire de verrous est en panne.

Copie primaire

Amélioration de la méthode du nœud central :

Le gestionnaire est réparti sur plusieurs sites.

Le verrouillage d'un item logique est sous la responsabilité d'un seul site, le site primaire.

+ suppression du goulot d'étranglement
+ peu de messages (3 en général)
+ simple à implémenter
- complique la gestion des interblocages

Majorité

Un gestionnaire de verrous sur chaque site, qui gère les verrouillages des objets se trouvant sur le site.

Pour verrouiller un item X dupliqué sur n sites, la transaction envoie une demande de verrouillage sur ~~(n+1)/2 sites où se trouvent des copies de X. Chaque gestionnaire détermine s'il accorde ou refuse le verrou~~. Demande à n sites

- Na et Nl : nb de verrous à obtenir
- Na > (n+1)/2 → Si on a (n+1/2) → on est sûr que personne
- Nl + Ne > n → conflit écriture/écriture n'en a + que nous
  → conflit lecture/écriture.

+ pas de contrôle central
- beaucoup de messages (3 (n+1)/2)
- gestion de interblocages complexe, les interblocages peuvent même arriver sur une seule donnée!

Ne = n
Nl = 1
} ROWA (A) → "Available" → On ne travaille que sur les données disponibles.

Détection des interblocages

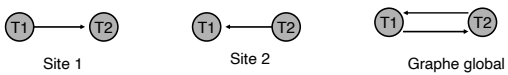
1. Timeout

   Pb : bien déterminer le délai
   + pas de messages
   - risque d'annulation de plusieurs transactions au lieu d'une seule

2. Graphe d'attente

Les nœuds sont les transactions. On a un arc de Ti ->Tj si Ti attend un verrou tenu par Tj.

On construit des graphes locaux, mais les cycles doivent être détectés sur le graphe global (union des graphes locaux).



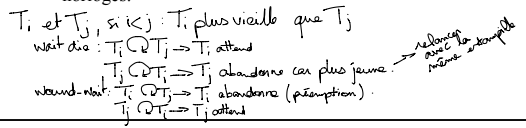
Détection des interblocages

Estampilles

même principe qu'en centralisé. Les transactions s'exécutent sur n'importe quel site, et laissent une estampille pour la copie en question. En cas d'écriture, la transaction écrit sur tous les sites où se trouvent des copies.

Comparaison des estampilles wait-die et wound-wait

Pb. Étendre la notion d'estampille au réparti, harmoniser les horloges.



Moniteur transactionnel

middleware

Extension du concept de transaction à d'autres contextes que les BD

- Support des transactions ACID (à un ens. d'accès à des objets)
- Accès continu aux données
- Reprise rapide du système en cas de panne
- Sécurité d'accès
- Performances optimisées
  - partage des connexions
  - réutilisation de transactions
- Partage de charge
  - distribution de transactions sur les serveurs
- Support de fichiers et bases de données hétérogènes
- API standardisées

Fonctions transactionnelles

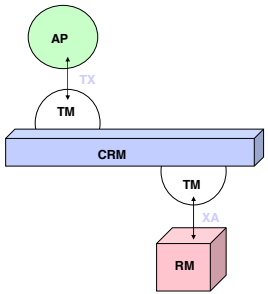
- Etapes du traitement transactionnel
  - traduire la requête utilisateur en format interne au moniteur
  - déterminer le type de transaction et le programme nécessaire à son exécution
  - débuter la transaction et lancer son exécution
  - valider, ou abandonner, la transaction (validation 2Phases)
  - retourner le résultat à l'émetteur de la requête

Fonctions systèmes

- Un moniteur assure aussi :
  - routage de la requête → accès à un catalogue réparti
  - équilibrage de charge entre requêtes et serveurs
  - résistance aux pannes (client, serveur)
  - gestion de configuration (processus, sessions, ...)
  - contrôle de performances et réglage
  - sécurité par certification des requêtes
  - gestion des communications et des services

Le modèle DTP de l'OpenGroup

- Composants
  - Programme d'application AP
  - Gérant de transactions TM
  - Gérant de communications CRM
  - Gérant de ressources RM
- Interfaces standards
  - TX = interface AP-TM
  - XA = interface TM- RM
  - intégration de TP
- Types de RM
  - gestionnaire de fichiers
  - SGBD
  - périphérique



Interface applicative TX

- **tx_open**
  - ordonne au TM d'initialiser la communication avec tous les RM dont les librairies d'accès ont été liées à l'application
- **tx_begin**
  - ordonne au TM de demander aux RM de débuter une transaction
- **tx_commit** ou **tx_rollback**
  - ordonne au TM de coordonner soit la validation soit l'abandon de la transaction sur tous les RM impliqués
- **tx_set_transaction_timeout**
  - positionne un " timeout " sur les transactions
- **tx_info**
  - permet d'obtenir des informations sur une transaction

Interface ressource XA

- **xa_open**
  - ouvre un contexte pour l'application
- **xa_start**
  - débute une transaction
- **xa_end**
  - indique au RM qu'il n'y aura plus de requêtes pour le compte de la transaction courante
- **xa_prepare**
  - lance l'étape de préparation du commit à deux phases
- **xa_commit**
  - valide la transaction
- **xa_rollback**
  - abandonne la transaction

Moniteurs transactionnels

- Encina de Transarc
  - issu de CMU (1992), racheté par IBM
  - construit sur DCE (OSF) pour la portabilité et la sécurité
  - transactions imbriquées
  - conformité DTP : Xa, CPI-C, TxRPC
- Open CICS de IBM
  - construit sur Encina (et DCE)
  - reprise de l'existant CICS (API et outils)
  - conformité DTP : Xa, CPI-C

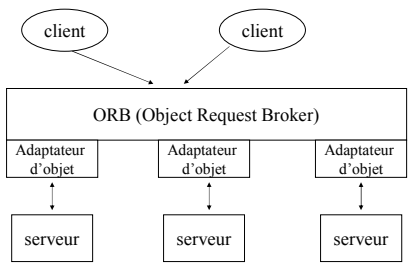
Moniteurs transactionnels (suite)

- Tuxedo de BEA
  - éprouvé (depuis 1984), à la base de DTP
  - supporte l'asynchronisme, les priorités et le routage dépendant des données
  - conformité DTP: Xa, Tx, XaTMI, CPI-C, TxRPC
- Top End de NCR
  - produit stratégique d'AT&T
  - respecte le modèle des composants DTP (AP, RM, TM, CRM)
  - haute disponibilité
  - conformité DTP: Xa, Xa+, Xap-Tp, Tx
- Autres : UTM de Siemens, Unikix

Object Transaction Service de l'OMG

- Service transactionnel de CORBA offrant
  - transactions imbriquées
  - interopérabilité avec des transactions non-CORBA conformes au standard DTP
  - transactions multi-ORB

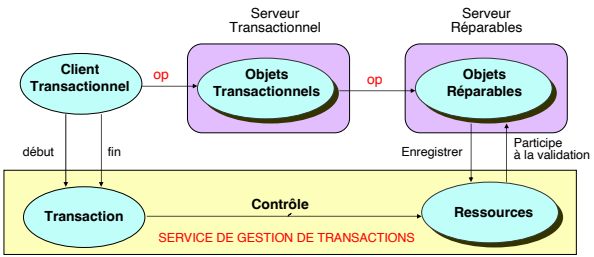
Architecture CORBA



Objets OTS

- Client transactionnel
  - {Begin-trans, Op1, ..., OpN, End-trans} ⇒ ORB
    - Begin-trans, End-trans ⇒ Service transactionnel
    - Op1, ..., OpN ⇒ objets transactionnels
- Objet transactionnel
  - objet impliqué dans la transaction mais sans ressource à protéger (fichier, bd, etc). Le comportement est affecté par la décision de terminaison, mais pas les données.
    - Op1, ..., OpM ⇒ objets réparables
- Objet réparable (recoverable object)
  - objet transactionnel avec des ressources à protéger. Les données sont affectées par la décision de terminaison.
    - enregistre les ressources à protéger
    - participe à la validation

Objets OTS dans une transaction



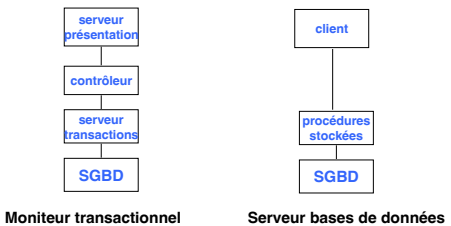
Principaux OTM

- OrbixOTM d 'Iona
  - basé sur Encina (Transarc-IBM)
- M3 de BEA
  - basé sur Tuxedo
- Integrated Transaction Service d'Inprise
  - implém. de JTS (Java Transaction Service = spéc. Java de OTS)
- Open CICS et TXSeries d'IBM
  - basé sur Encina

Microsoft Transaction Server

- OTM intégré à DCOM et Windows
  - moniteur et serveur d'application
- Partage de grappes de Windows NT (cluster)
- Les disques sont supposés partagés
- Allocation des ressources en pool aux requêtes :
  - pool de connexion aux ressources (SQL Server)
  - pool de transactions (support)
  - pool de machines

SGBD vs Moniteur : configurations



SGBD vs Moniteur : propriétés

- Avantages moniteur :
  - routage dans des applications de très grande taille
  - langages et environnement plus riches
  - serveurs bases de données répartis hétérogènes
  - communications client-serveur
  - environnement de gestion système
- Avantages SGBD :
  - simplicité et prix
  - performances pour petites et moyennes applications