GICOM – M2GI SRR
RICOM – M2RICM

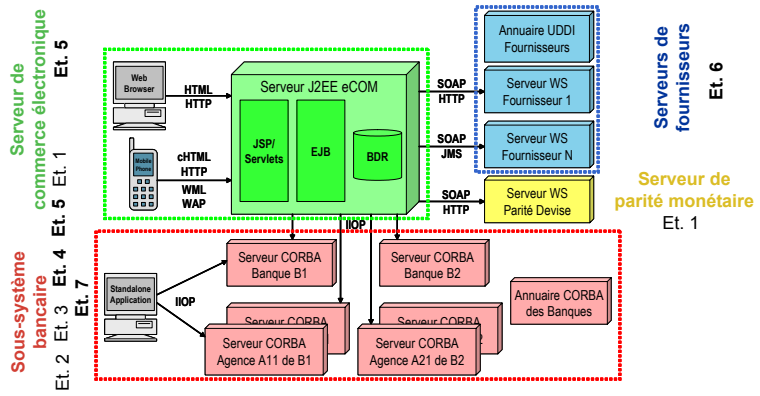
Etape 4 : Sous-système bancaire persistant et fiable

Sara Bouchenak, Johann Bourcier, Didier Donsez
Pierre-Yves Gibello

{Sara.Bouchenak, Johann.Bourcier, Didier.Donsez}@imag.fr
{Pierre-Yves.Gibello}@imag.fr

http://www-adele.imag.fr/~donsez/ujf/GICOM/GICOM_ENS
http://sardes.inrialpes.fr/~bouchena/teaching/

Architecture du projet GICOM



Etapes de GICOM 1

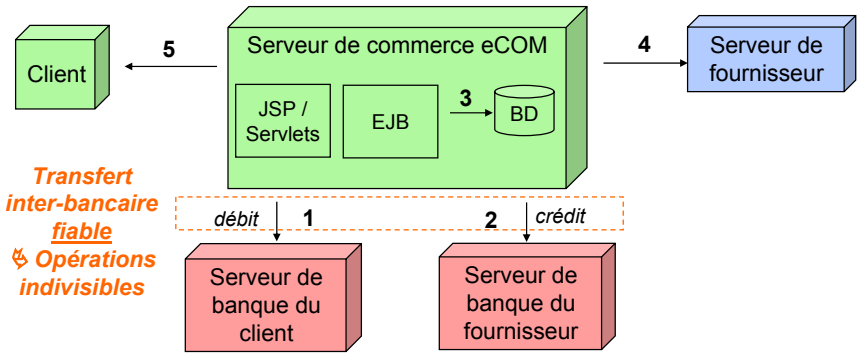
- *Etape 2 : Sous-système bancaire*
- *Etape 3 : Sous-système bancaire persistant*
- Etape 4 : Sous-système bancaire persistant et fiable

Exemple : Ordre d'achat d'un client

- Un client envoie un ordre d'achat au serveur de commerce :
  1. Débit du compte du client
  2. Crédit du compte du fournisseur
  3. Stockage de la commande
  4. Transmission de la commande au fournisseur
  5. Envoi d'un e mail au client avec une clé de preuve d'achat (à utiliser en cas de litige)

Transfert inter-bancaire

Ordre d'achat dans un contexte réparti



Opérations indivisibles (atomiques)

Débit client	Crédit fournisseur	Transfert inter-bancaire
OK	OK	Effectué
OK	Pb	Non effectué
Pb	OK	Non effectué
Pb	Pb	Non effectué

Opérations effectuées dans le contexte d'une transaction

Plan

1. Système transactionnel
2. Contrôle de l'atomicité
3. Contrôle de la concurrence
4. Synthèse

Transaction

- Une transaction regroupe une suite d'opérations
- Une transaction se termine par :
  - Un succès : validation (*commit*) des opérations
  - Un échec : abandon (*abort*/*rollback*) des opérations, pour cause de
    - Défaillance du système
    - Violation de contraintes d'intégrité
    - Programme se rétracte avant la fin

Propriétés ACID

- Atomicité
  - Séquence d'opérations constituant une transaction est indivisible : soit toutes les opérations sont exécutées, soit aucune ne l'est
- Cohérence
  - Contraintes d'intégrité du système vérifiées à la fin d'une transaction
- Isolation
  - Avant la fin d'une transaction, les modifications apportées par celle-ci ne sont pas visibles par les autres transactions concurrentes
- Durabilité
  - Les mises à jour validées par une transaction sont durables malgré les pannes potentielles du système

Atomicité – CID

- Séquence d'opérations constituant une transaction est indivisible :
  - soit toutes les opérations sont exécutées
  - soit aucune opération n'est exécutée

Débit client	Crédit fournisseur	Transfert inter-bancaire
OK	OK	Effectué
OK	Pb	Non effectué
Pb	OK	Non effectué
Pb	Pb	Non effectué

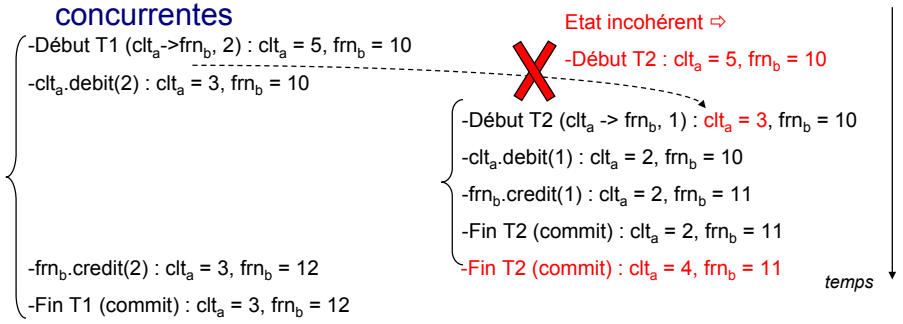
A – **C**ohérence – ID

- Contraintes d'intégrité du système vérifiées à la fin d'une transaction :
  - Pas d'argent "en plus " créé
  - Pas d'argent qui disparaît
  - Exemple : Transfert inter-bancaire d'une somme d'argent *a*
    - compte_client.debit(a) : nouv_solde_clt = anc_solde_clt - a
    - compte_fournisseur.credit(a) : nouv_solde_frn = anc_solde_frn + a

    ↳ (nouv_solde_clt + nouv_solde_frn) =
    (anc_solde_clt + anc_solde_frn)

AC – **I**solation – D

- Avant la fin d'une transaction, les modifications apportées par celle ci ne sont pas visibles par les autres transactions concurrentes



ACI – **D**urabilité

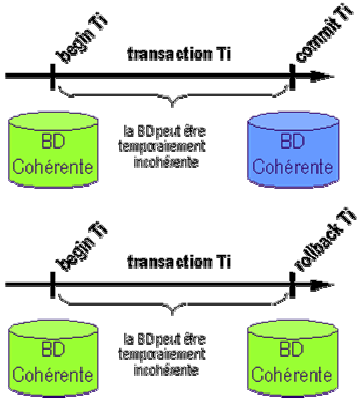
- Les mises à jour validées par une transaction sont durables malgré les pannes potentielles du système

-Début T1 (clt_a->frn_b, 2) : clt_a = 5, frn_b = 10
…
-Fin T1 (commit) : $\underline{clt_a = 3, frn_b = 12}$
Panne

-Début T2 (clt_a->frn_b, 1) : $\underline{clt_a = 3, frn_b = 12}$
…

temps

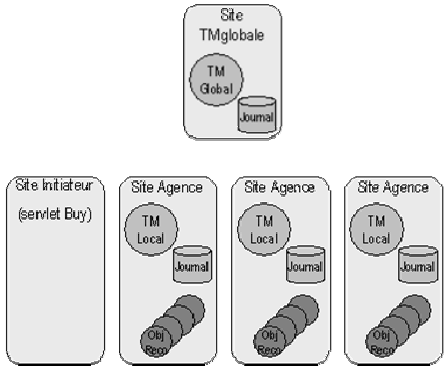
Transaction : *commit* ou *rollback*



GICOM : Hiérarchie transactionnelle

- Transaction répartie sur plusieurs sites
  - **Initiateur**
    - Démarre une transaction auprès du coordinateur global
    - C'est le serveur de commerce
  - **Coordinateur global**
    - Coordonne la transaction entre les différents participants jusqu'à sa terminaison (*Transaction Manager*)
    - Générique : indépendant de tout type d'application (banque, réservation de billets d'avion, …)
    - Un serveur CORBA
  - **Participants**
    - Sur chaque site participant, un gestionnaire local de transactions gère des ressources locales "recouvrables "
    - Coopère avec le coordinateur global
    - C'est une agence bancaire

GICOM : Transaction répartie



GICOM : Ressource *recouvrable*

- Objet dont il est possible de défaire les modifications tant qu'une transaction n'est pas validée
- Objet recouvrable
  - Objet persistant : revenir à un état stable précédemment sauvegardé
  - Objet dont l'action n'a pas encore été effective (envoi différé de mail vers un fournisseur)

Plan

1. Système transactionnel
   - Définition d'une transaction
   - Propriétés ACID
   - Hiérarchie transactionnelle
   - Objets recouvrables
2. *Contrôle de l'atomicité*
3. Contrôle de la concurrence
4. Synthèse

Coordonner les validations

- Atomicité des opérations réparties
  - Toutes les mises à jour exécutées sur tous les participants
  - Ou aucune mise à jour n'est exécutée sur aucun des participant
- Chaque participant est responsable de la validation (exécution ou non) des mises à jour locales
  - ↳ Coordonner l'ensemble des validations pour éviter les incohérences

Validation à deux phases (2PC)

- Coordonner les validations effectuées par les participants d'une transaction
- Une transaction : divisée en deux phases
  - Phase 1
    - Préparation des mises à jour sur les participants
  - Phase 2
    - Si réussite de phase 1 : Intégration effective des mises à jour préparées en phase 1
    - Si échec de phase 1 : Annulation des mises à jour préparées en phase 1

Validation à deux phases : pannes

- Panne du coordinateur
  - Phase 1 : Au redémarrage du coordinateur, envoi d'un *Rollback* à tous les participants
  - Phase 2 : Au redémarrage du coordinateur, envoi d'un *Commit* ou *Rollback* à tous les participants (selon décision prise en phase 1)
- Panne d'un participant
  - Phase 1 : Décision prise par coordinateur est *Rollback*. Au redémarrage du participant, contact coordinateur, devenir de la transaction (*Rollback*), abandon
  - Phase 2 : Au redémarrage du participant, contact coordinateur, devenir de la transaction (*Commit* ou *Rollback*), validation ou abandon

Validation à deux phases : pannes

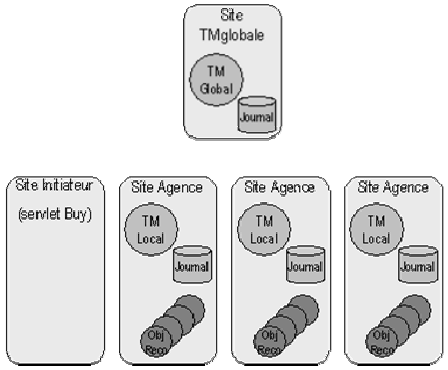
Etat du coordinateur lors de la panne	*Reprise après panne*
INITIAL	Transaction oubliée
PREPARE	Passer à l'état ROLLBACKED (annuler les opérations)
COMMITTED	Poursuivre la validation
ROLLBACKED	Poursuivre l'abandon

Etat du participant lors de la panne	*Reprise après panne*
INITIAL	Transaction oubliée
PREPARE	Passer à l'état ROLLBACKED (annuler les opérations)
READY	Passer à l'état READY
COMMITTED	Passer à l'état READY
ROLLBACKED	Passer à l'état READY

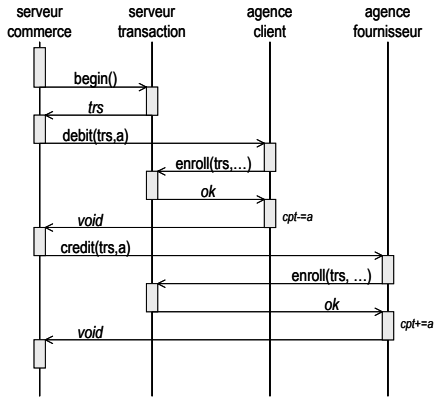
Validation à deux phases : Traitement des pannes

- Panne d'un intervenant (coordinateur ou participant)
  - Action effectuée lors de la reprise dépend de l'état de l'intervenant lors de la panne
  - Pouvoir défaire des opérations
- Utilisation de journaux (*logs*)
  - Ecrire (log, état)
  - Ecrire (log, infos sur les opérations à effectuer)
  - Exécuter (opérations)

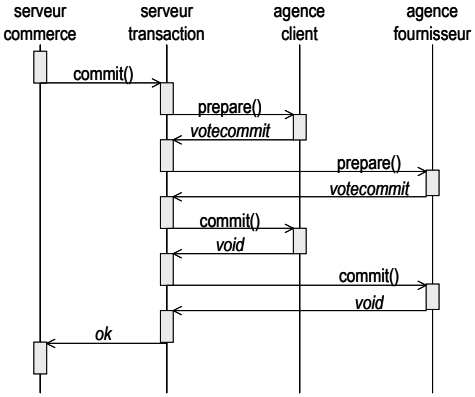
Utilisation de journaux (*logs*)



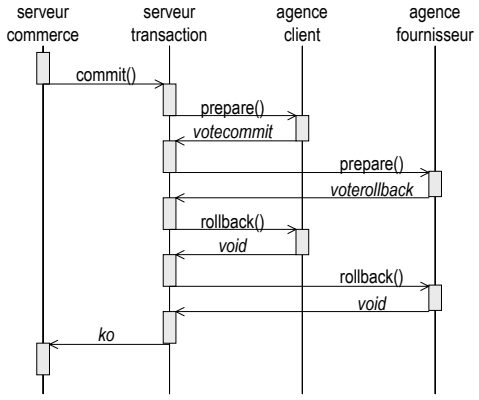
Réalisation du 2PC : Démarrage de la transaction



Réalisation du 2PC : Transaction validée



Réalisation du 2PC : Transaction abandonnée



Réalisation du 2PC

- Interactions
  - entre le serveur de commerce et le coordinateur global (*begin*, *commit*, *rollback*)
  - entre le coordinateur global et les gestionnaires locaux aux participants (*enroll*, *prepare*, *commit*, *rollback*)

  réalisées par des RPC (requêtes CORBA)
- Gestionnaire local : méthode *prepare* fournie
  - Retour normal de la méthode : accord validation transaction
  - Exception : pas d'accord de validation ou panne du participant
- Identifiant transaction
  - Apparaît explicitement dans les appels de méthodes

Réalisation du 2PC : Objets persistants recouvrables

- Objet persistant recouvrable
  - Objet persistant
    - Hérite de la méthode *save* : sauvegarde atomique de l'état
  - Objet recouvrable
    - Nouvelle méthode *prepare* : sauvegarde temporaire de l'état en attendant le *save* effectué en phase 2 de la validation
    - Nouvelle méthode *undo* : abandon de l'état temporairement sauvegardé, si *save* non effectué

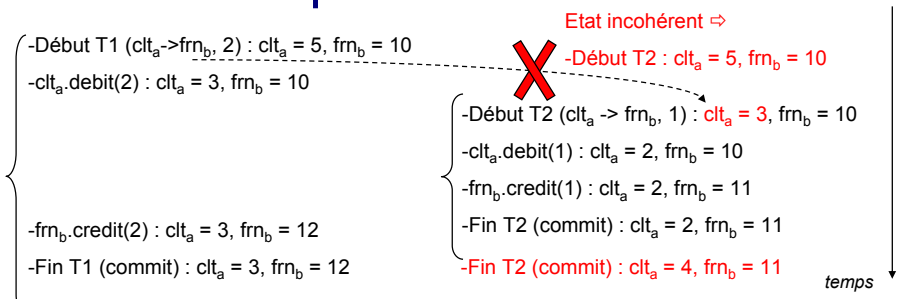
Réalisation du 2PC : Serveur de commerce

- Servlet *Buy*
  - begin : connexion avec le serveur de transaction, récupération d'un identifiant de transaction
  - débit du compte client
  - crédits des comptes fournisseurs
  - enregistrement de la commande dans la base du serveur de commerce électronique
  - envois de mail aux fournisseurs
  - envoi d'un mail au client de la preuve d'achat (avec la référence de la commande)
  - commit : validation de la transaction d'achat
  - ou rollback : annulation de la transaction d'achat

Plan

1. Système transactionnel
2. Contrôle de l'atomicité
   - Protocole de validation à 2 phases (2PC)
   - Gestion des pannes dans le 2PC
   - Réalisation du 2PC
3. *Contrôle de la concurrence*
4. *Synthèse*

Contrôle de la concurrence : Problématique



- Transactions concurrentes
  - Accès aux mêmes objets recouvrables
  - Problème d'incohérence
- Ordonnancer les transactions concurrentes
  - T1 avant T2 ou T2 avant T1

Ordonnancement total des transactions par estampillage

- A chaque transaction est associée une estampille unique
- Ordre d'accès à un objet recouvrable suivant estampilles croissantes des transactions
- Si transaction "en retard " (estampille inférieure), transaction abandonnée

Algorithme d'ordonnancement total

- Read (Ti, O)

```
int objStamp = O.getLastStamp();
int trsStamp  = Ti.xid;

if (objStamp < trsStamp) {
   // Read O
   …
   O.setLastStamp(trsStamp);
} else {
   Ti.rollback();
}
```

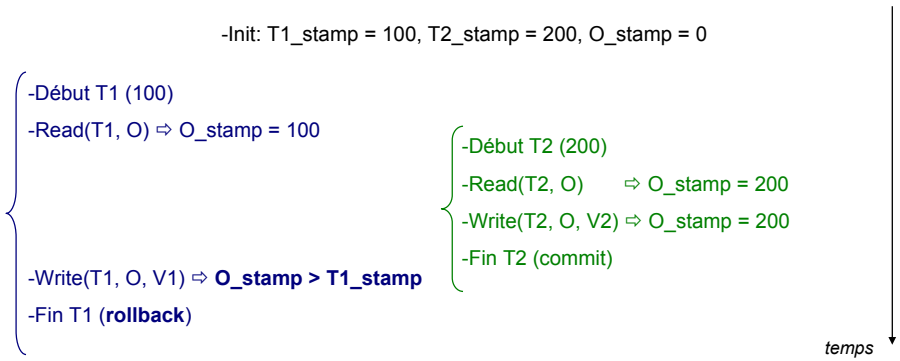

Algorithme d'ordonnancement total

- Write (Ti, O, Val)

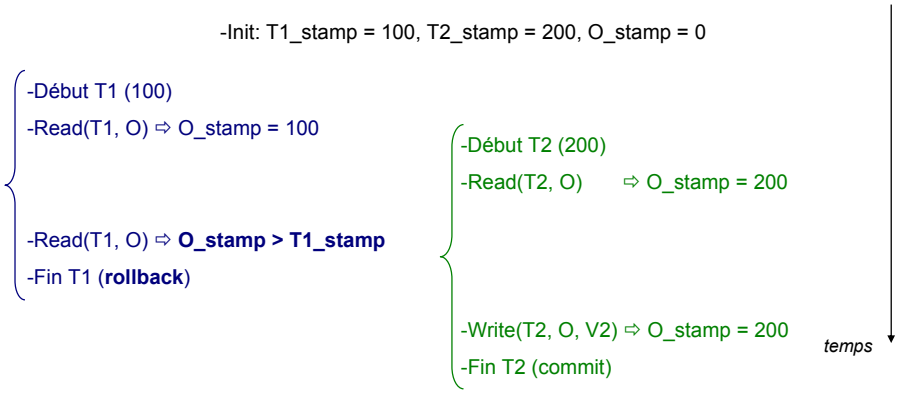
```
int objStamp = O.getLastStamp();
int trsStamp  = Ti.xid;

if (objStamp < trsStamp) {
   // Write Val in O
   …
   O.setLastStamp(trsStamp);
} else {
   Ti.rollback();
}
```


Algorithme d'ordonnancement total Exemple 1



Algorithme d'ordonnancement total Exemple 2



Algorithme d'ordonnancement total Conclusion

- Pas nécessaire d'abandonner T1 après T2 si
  - action effectuée par T1 est une lecture
  - action effectuée par T2 est une lecture
- Il faut différencier entre
  - les action permutables (lecture-lecture)
  - et les action non permutables (lecture-écriture, écriture-lecture, écriture-écriture)
- Possibilité d'optimiser l'ordonnancement total

Algorithme d'ordonnancement partiel

- Ordonnancement partiel = ordonnancement total
  - pas d'abandon si lectures entrecroisées
- Ordonnancer les actions non permutables
- Deux estampilles par objet recouvrable
  - estampille de lecture
  - estampille d'écriture

Algorithme d'ordonnancement partiel

- Read (Ti, O)

```
int objReadStamp = O.getLastReadStamp();
int objWriteStamp = O.getLastWriteStamp();
int trsStamp      = Ti.xid;

if (objWriteStamp > trsStamp) {
   Ti.rollback();
} else {
   // Read O
   …
   O.setLastReadStamp(Max(trsStamp, objReadStamp ));
}
```

Algorithme d'ordonnancement partiel

- Write (Ti, O, V)

```
int objReadStamp = O.getLastReadStamp();
int objWriteStamp = O.getLastWriteStamp();
int trsStamp      = Ti.xid;

if ((objReadStamp > trsStamp) || (objWriteStamp > trsStamp)) {
   Ti.rollback();
} else {
   // Write V in O
   …
   O.setLastWriteStamp(trsStamp);
}
```


Implantation de l'ordonnancement partiel

- Chaque objet recouvrable
  - 2 nouveaux attributs : estampille de lecture et estampille d'écriture
  - Chaque méthode : teste si la transaction courante peut lire/écrire l'objet avant d'exécuter la méthode
  - Chaque appel de méthode : peut lever une exception *ConcurrencyControlException*
  - Peut provoquer l'abandon de la transaction *rollback()*

Plan

Implantation d'un sous-système bancaire persistant et fiable : Synthèse

- Hiérarchie transactionnelle
  - Transaction répartie sur plusieurs sites
    - **Initiateur**. Démarrage transaction
    - **Coordinateur global**. Gestionnaire de transactions (un serveur CORBA)
    - **Participants**. Sur chaque site participant, un gestionnaire local de transactions

Synthèse : 2PC

- Transaction divisée en deux phases
  - Phase 1
    - Préparation des mises à jour sur les participants
  - Phase 2
    - Si réussite de phase 1 : Intégration effective des mises à jour préparées en phase 1
    - Si échec de phase 1 : Annulation des mises à jour préparées en phase 1
- Pouvoir défaire des opérations : Utilisation de *logs*
  - Ecrire (log, état)
  - Ecrire (log, infos sur les opérations à effectuer)
  - Exécuter (opérations)

Synthèse : 2PC

- Interactions
  - serveur de commerce – coordinateur global (*begin*, *commit*, *rollback*)
  - coordinateur global – gestionnaires locaux (*enroll*, *prepare*, *commit*, *rollback*)

  réalisées par des RPC (requêtes CORBA)
  - Gestionnaire local : méthode *prepare* fournie (accord validation)
- Identifiant transaction
  - Apparaît explicitement dans appels de méthodes
- Objet persistant recouvrable
  - Persistant : hérite de la méthode *save*
  - Recouvrable : nouvelles méthodes *prepare* et *undo*

Synthèse : Ordonnancement partiel

- Chaque objet recouvrable
  - 2 nouveaux attributs : estampille de lecture et estampille d'écriture
  - Chaque méthode : teste si la transaction courante peut lire/écrire l'objet avant d'exécuter la méthode
  - Chaque appel de méthode : peut lever une exception *ConcurrencyControlException*
  - Peut provoquer l'abandon de la transaction *rollback()*

Synthèse : Servlet Buy

- begin : connexion avec le serveur de transaction, récupération d'un identifiant de transaction
- débit du compte client
- crédits des comptes fournisseurs
- enregistrement de la commande dans la base du serveur de commerce électronique
- envois de mail aux fournisseurs
- envoi d'un mail au client de la preuve d'achat (avec la référence de la commande)
- commit : validation de la transaction d'achat
- ou rollback : annulation de la transaction d'achat