

# Mémoire atomique auto-reconfigurable pour systèmes de pairs

Emmanuelle Anceaume   Maria Gradinariu   Vincent Gramoli  
Antonino Virgilitto

13 octobre 2005



# Objectifs

- Partage *Peer-to-Peer* (P2P) des ressources.
- Read-Only → Read-Write Model.
- Équilibrage de charge.
- Auto-adaptation face au dynamisme.

# Plan

- 1 Introduction
  - Motivations et Applications
  - Modèle
- 2 Procédure de démarrage
- 3 Problématique
  - Surcharge
  - Cohérence atomique
- 4 Atomicité
- 5 Auto-Reconfiguration
- 6 Conclusion

# Motivations et Applications



- Groupware (Collecticiel)
- Web Services
- Intelligence collective

...en P2P, sans coût de maintenance, de stockage, etc.

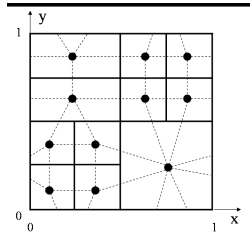
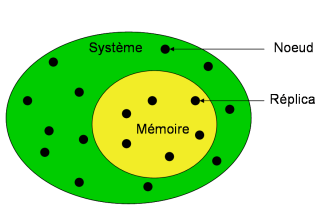
- Approches existantes [routage proactif].
  - Opérations atomiques/linéarisables rapides : environ 4 hops.
  - Pour cela : Reconfiguration avec échange de messages all-to-all.
  - Donc, Dynamisme + "Mise à l'échelle" = Impossible :  
**Ex.**  $10^3$  nœuds  $\Rightarrow$   $10^6$  messages (À chaque connexions/déconnexions).
- Ici, on utilise des connaissances restreintes localement pour les opérations [routage hybride].

# Modèle

- Distribué
  - Réseau de  $n$  nœuds connectés
- Dynamique
  - Arrivées et départs de nœuds
  - Pannes crash
  - Détection de fautes ultimes
- Asynchrone
  - Communication par voisinage
  - Délai de message arbitrairement long
  - Perte éventuelle de messages

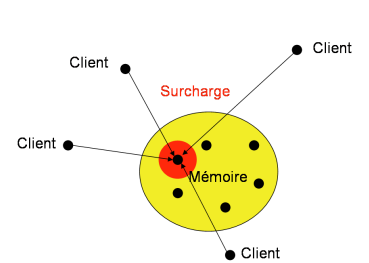
## Procédure de démarrage

- Responsabilité de l'objet  $X = \text{Plan fictif } [0, 1) \times [0, 1)$ .
- Initialement un seul nœud  $R$  responsable maintient une copie de l'objet.
- $R$  copie  $X$  sur  $m$  nœuds : les *réplicas*.
- Plan partagé au fur et à mesure des ajouts de réplicas.
- Réplicas forment un overlay en tore, la *mémoire* :
  - Les responsables des zones adjacentes sont *voisins*.



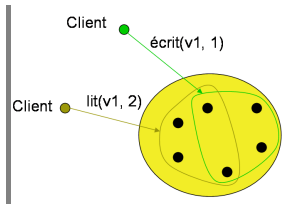
# Surcharge

- Tout nœud est client potentiel.
- Un nœud envoie une requête sur un réplica de la mémoire (qu'il connaît) pour effectuer une lecture/écriture.
- Il est possible qu'un réplica ne puisse traiter toutes les requêtes reçues. Il est alors surchargé.



## Cohérence atomique

- Plusieurs lectures simultanées doivent renvoyer la même valeur.
- Donc : les écritures doivent être ordonnées totalement, et
- les lectures doivent être ordonnées par rapport aux écritures.
- Pour cela, au moins un réplica contacté durant une opération doit pouvoir témoigner des opérations antérieures (on utilise des ensembles intersectés, *quorums*).



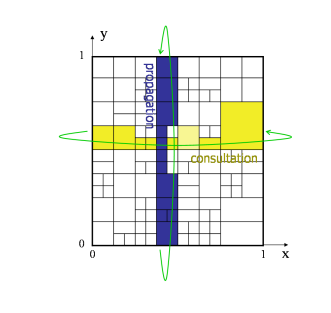
# Mémoire Atomique

## Définition ( $\langle tag, val \rangle$ )

*Tout réplica de l'objet considéré conserve sa valeur,  $val$ , et un compteur d'écriture,  $tag$ , indiquant la version de cette valeur.*

- Consultation
  - Les paires  $\langle tag, val \rangle$  des responsables de tout une **ligne** du plan sont consultées :
  - La paire possédant le plus grand  $tag$  est retournée.
- Propagation
  - La paire  $\langle tag, val \rangle$  est propagée à l'ensemble des responsables d'une **colonne**.

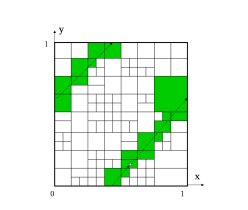
- $Lecture(v?) \Leftrightarrow Consultation(\langle t, v \rangle?) + Propagation(\langle t, v \rangle!)$
- $Écriture(v!) \Leftrightarrow Consultation(\langle t', v' \rangle?) + Propagation(\langle t' ++, v \rangle!)$



- Atomicité : Le *tag* utilisé dans une opération définit son numéro d'ordre.

# Mémoire Auto-Reconfigurable

- Surcharge :
  - En cas de réceptions quasi-simultanées, un réplica exécute plusieurs requêtes en une seule.
- Équilibrage de Charge :
  - Un réplica surchargé transfère la requête au voisin de son voisin (diagonal).
  - S'il récupère le même message, il approxime une surcharge du système.



- Extension
  - En cas de surcharge, une réplication est effectuée sur un nœud actif.
  - Le plan de responsabilité est partagé.
- Réduction
  - Un réplica sans sous-charge prévient et sort du système.

# Conclusion

- Résumé
  - Réplication  $\Rightarrow$  Tolérance aux fautes.
  - Quorums  $\Rightarrow$  Atomicité.
  - Auto-adaptation  $\Rightarrow$  Compromis entre charge et complexité.
  - Connaissance locale  $\Rightarrow$  Scalabilité.
- Discussion
  - Opérations et Reconfiguration scalables.
  - Reconfiguration peu couteuse (temps & #messages).
  - Opérations couteuses en temps ( $O(\sqrt{m})$  hops).
- Travaux en cours et futures :
  - Temps des opérations :  $O(\log(\sqrt{m}))$ . hops.
  - Modèle byzantin.
  - Opérations transactionnelles.