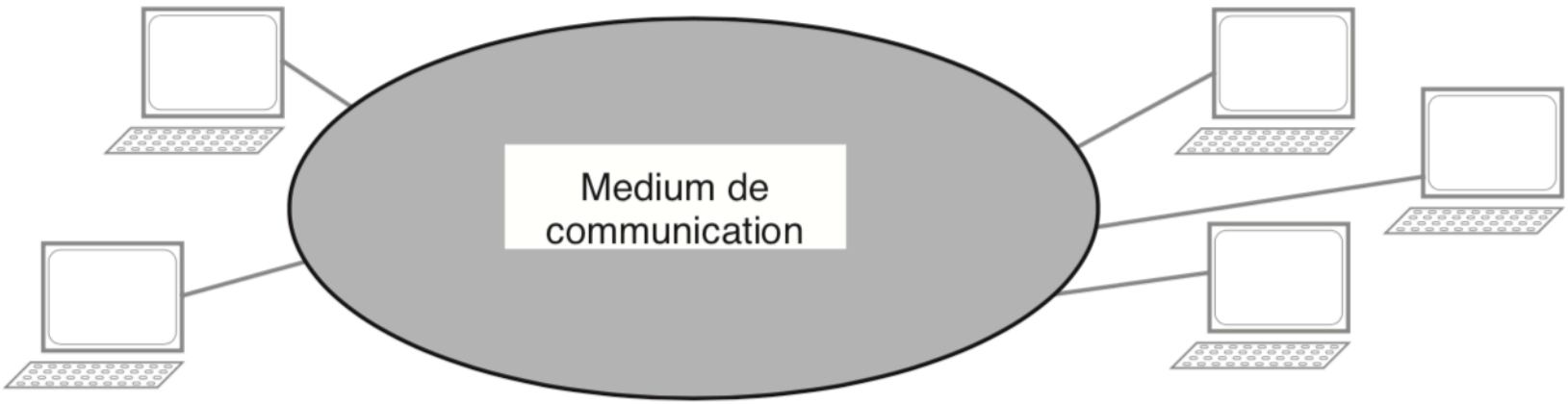
Objets Dupliqués Robustes

Projet Données Réparties (S8)

11 mars 2023

Plan

Système réparti



Ensemble de sites (processeurs) tel que

- les sites réalisent un *traitement concurrent* : ils sont *indépendants* du point de vue de l'activité, mais *liés* par la réalisation ou l'utilisation d'un *service* commun, *global*
- les sites sont géographiquement *séparés*
- les sites *communiquent* via un *réseau asynchrone* (délais de communication non bornés),
- les sites et le réseau sont *non fiables*
- le système est *ouvert* : des sites peuvent à tout moment rejoindre ou quitter le système

Contexte du projet : JVMs sur réseau local (voire même machine)

Cohérence : la contrepartie de la duplication

Partager une donnée en environnement réparti
→ dupliquer cette donnée (efficacité/disponibilité)

Contrainte : la *duplication* devrait rester *transparente* :
les copies d'une même donnée devraient se comporter comme une copie unique, être *cohérentes*

→ la mise à jour d'*une* copie doit affecter *l'ensemble* des copies

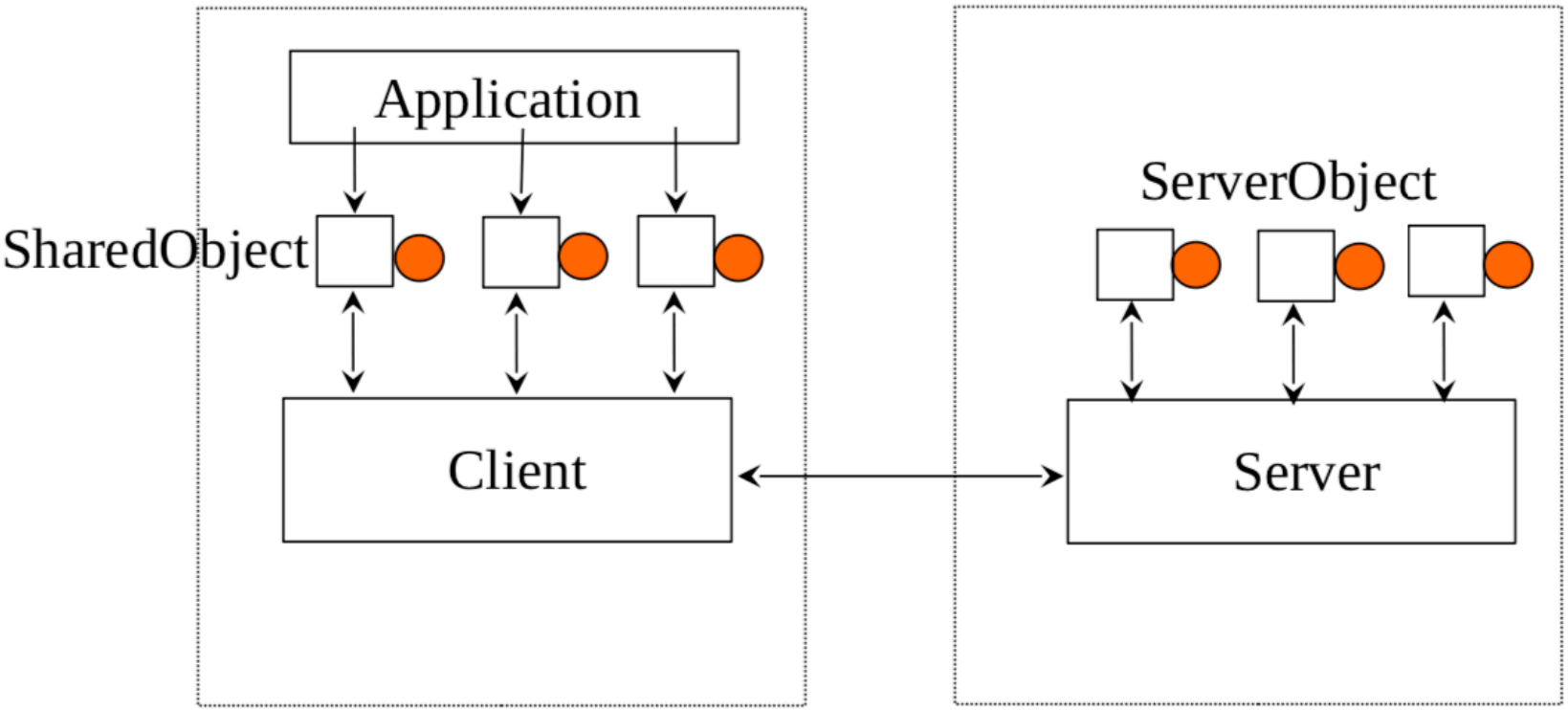
Problème : coût d'un maintien « strict » de l'identité entre copies

- *en temps* : coordination des mises à jour des différentes copies d'une même donnée
- *en volume* : propagation/diffusion des mises à jour vers les différentes copies d'une même donnée

→ *arbitrage* nécessaire entre le *coût* et la *qualité* de la cohérence.

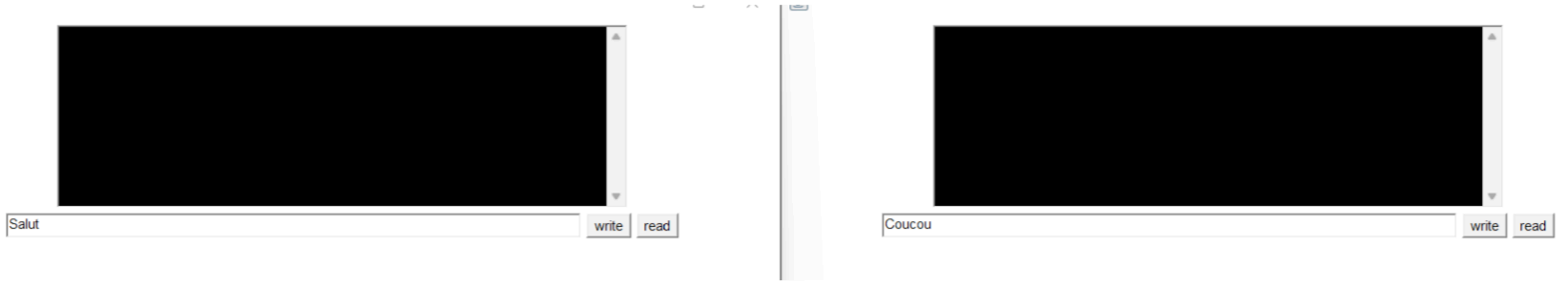
Plan

Réponse sobre : protocole paresseux (PODP)



- schéma lecteurs rédacteurs réparti (verrou avec mode partagé)
- mise à jour des objets uniquement au moment où l'accès est demandé
- utilisation d'un cache de verrous pour limiter les interactions

Inconvénient de la sobriété : IRC



→ permettre de synchroniser les mises à jour

- compléter le PODP en proposant un suivi des mise-à jour par un schéma publier/s'abonner
- généralisation : compléter le PODP en proposant un mode de mise à jour synchrone automatique

Durée estimée : 3 semaines au maximum

Plan

Pannes...

Panne franche (ou *panne d'arrêt (fail stop)*)

- soit le (sous-)système a un comportement correct,
  (il n'omet pas de message etc. . .)
- soit il est en panne (défaillant), et ne fait rien

Remarques

- la panne franche est le cas le plus simple à détecter ($\rightarrow$ à traiter)
  - $\rightarrow$ technique *fail-fast* classique :
    forcer l'arrêt dès qu'une erreur interne est détectée
- et pourtant, PODP n'y résiste pas, en particulier dans le mode synchrone) (risque de blocage en cas de panne d'un abonné)

Défi

$\rightarrow$ concevoir un nouveau protocole (PODR), robuste et non bloquant, en conservant l'architecture générale.

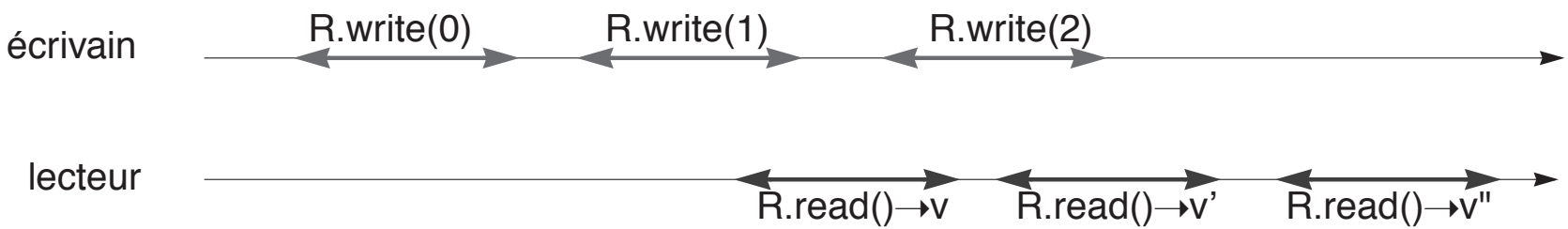
Non bloquant →] permettre des accès concurrents

- le modèle proposé n'est plus celui des verrous
- lock_read/write + unlock → read/write
- lecture et écritures concurrentes
- les mises à jour passent toujours par le serveur écritures → séquentiellles (ordonnées)

Problème : maintenir une cohérence acceptable pour l'utilisateur entre écritures et résultat des lectures

Registres réguliers

- Hypothèses
  - un écrivain, plusieurs lecteurs
  - pas de conflit en écriture
  - écritures séquentielles
- Résultat d'une lecture
  - non concurrente avec une écriture
    → valeur courante du registre (dernière valeur écrite)
  - concurrente avec une/des écritures
    → valeur courante ou valeur d'une des écritures concurrentes



$v \in \{0,1,2\}$ / $v' \in \{1,2\}$ / $v'' = 2$

Remarque

inversion possible de valeurs (v=2, v'=1)
→ un *registre régulier* n'a *pas de spécification séquentielle* :
il existe des exécutions possibles non séquentielles

Registres atomiques

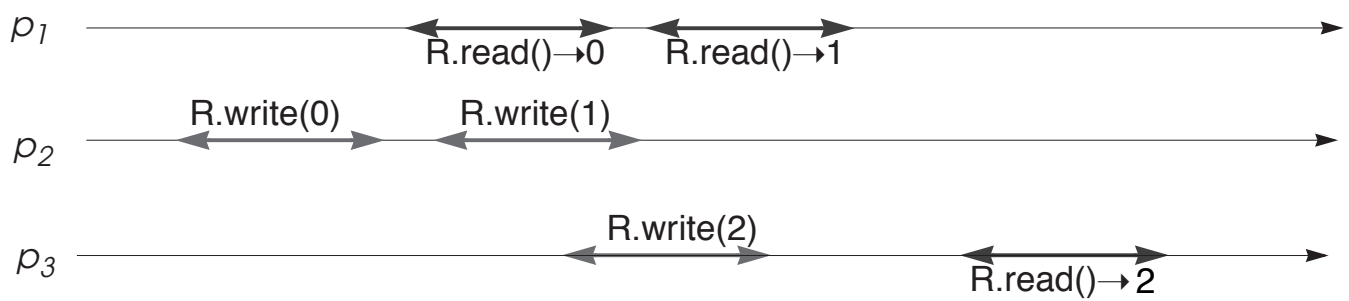
Hypothèse : plusieurs écrivains, plusieurs lecteurs

Propriété : accès *linéarisables*

Pour tout ensemble d'accès concurrents au registre,
il existe une exécution séquentielle S

- donnant les mêmes résultats
- respectant la chronologie des opérations non concurrentes
- *légale* :
  toute lecture fournit la valeur de l'écriture immédiatement précédente

Exemple



$S = R.write(0)\,;R.read()\rightarrow 0\,;\ R.write(2)\,;\ R.write(1)\,;R.read_1()\rightarrow 1\,;\ R.read_3()\rightarrow 2$

Plan

Architecture générale

PODR sera conçu pour la même architecture que PODP :

- le serveur gère et conserve la version « à jour » de l'objet dupliqué (copie maître) ;
- les clients gèrent une copie locale, possiblement en retard par rapport à la copie maître ;
- les écritures se font toujours et uniquement sur (via) le serveur
  - → une écriture demandée (et obtenue) par un client peut ne pas être répercutée instantanément sur la copie de ce client ;
- les lectures se font en collectant une ou plusieurs copies locales et doivent au final fournir la dernière valeur écrite par le serveur, ou la valeur en cours d'écriture par le serveur.

Indications

- numéroter (version) chaque valeur prise par l'objet partagé.
- identifier de manière unique les différentes requêtes émises par les lecteurs (par exemple : référence, ou encore couple <id client, compteur géré par chaque client>)
- le sujet propose une ébauche d'interface, à adapter selon la direction de vos réflexions
- Pour construire la solution, il peut être utile de séparer les problèmes :
  - implémenter un service de registre régulier
  - étendre ce service pour implanter un registre atomique pour des écritures séquentielles et des lectures concurrentes, ce qui consiste essentiellement à gérer le problème de l'inversion de valeurs.
  - (*Bonus*) étendre ce service pour implanter un registre atomique « général », ce qui consiste essentiellement à ordonner globalement les écritures

Travail demandé et organisation

- Projet orienté vers la conception
  - rapports (documentation des idées)
  - expérimentation (validation, démonstration) expérimentales
- Projet réalisé *en trinôme, ou binôme à défaut*.
- **Déroulement**
  - Des séances de suivi jalonnent le déroulement du projet.
  - Les livrables correspondant aux différents protocoles seront à déposer sur Moodle : début mars pour les 2 premiers protocoles, et début mai (a priori) pour le protocole robuste.
  - Une restitution finale aura lieu mi-mai.
- **Evaluation**
  L'évaluation tiendra compte à parts sensiblement égales
  - des livrables portant sur les 2 premiers protocoles (extensions de PODP)
  - du rapport de conception relatif au protocole robuste
  - de l'implémentation et de la démonstration du protocole robuste.
- Calendrier et les modalités précises et actualisées → Moodle