

# Un protocole de cohérence customizable pour des DataObjects répliqués

Corina Ferdean, Mesaac Makpangou

INSTITUT NATIONAL  
DE RECHERCHE  
EN INFORMATIQUE  
ET EN AUTOMATIQUE



**INRIA**  
ROCQUENCOURT

CDUR 2005



# Plan

---

- Motivation
- Etat de l'art
- Objectif et résumé de la solution
- Protocole de cohérence générique
- Framework de cohérence
- Conclusion et travaux en cours



# Motivation

---

- **Réplication de Data Objects:**
  - **Réplication:** technique classique pour améliorer la qualité de partage des données en réparti (performance, disponibilité ...)
  - **Modèle objet:** modularité, ré-utilisabilité, adaptabilité
- **Gestion de la cohérence des copies est un problème difficile**
  - Qualité (fraicheur) de l'état observable par une opération?
  - La concurrence des invocations non-commutatives?
  - Le délai de propagation des m.a.j sur les copies distantes?
  - Défis:
    - Caractéristiques des environnements distribués
    - Besoins de performance: compromis
    - Hétérogénéité des opérations (besoins de cohérence)



# Etat de l'art

---

- **Modèles de cohérence faible:**
  - **Scheduling (concurrency):** Bayou, IceCube
  - **Qualité de l'état observable:** TACT, lazy réplication
  - **Limitation:** les deux types de contraintes peuvent être requis
- **Globe**
  - Protocole de cohérence générique (parametres: *when, how, by whom* sont les updates échangés)
  - **Limitation:** pas de garanties sur l'état observable
- **Khazana**
  - **Modèle de cohérence flexible:** visibilité des updates distants, visibilité des updates locaux, degré de staleness toléré et le type d'accès
  - **Limitation:** comment elles sont supportées?

# Objectif et résumé de la solution (1/2)

---

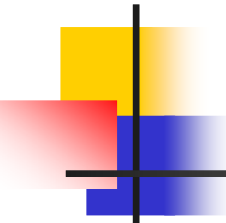
- **Objectif:** supporter les besoins de cohérence spécifiques des opérations
  
- Solution en 3 étapes:
  1. Meta-modèle de cohérence
  2. **Protocole générique qui le réalise**
  3. **Implementation du protocole**
  
- **Concepts de base:** *replica*, *access*
- Modèle de la réplication active

# Objectif et resume de la solution (2/2)



A Generic and Flexible Model for Replica Consistency Management, Corina Ferdean, Mesaac Makpangou , 1st International Conference on Distributed Computing & Internet Technology (ICDCIT 2004), Bhubaneswar, India.

- **Meta-modèle**: effort de classification et formalisation des conditions de cohérence existantes
- 3 concerns: *la qualité de l'état observé, le scheduling et la visibilité*
- Différents types des contraintes de cohérence, définis au niveau de l'opération
- La combinaison appropriée des conditions: *modèle de l'objet*
- Propriétés: *flexibilité et extensibilité*



# Protocole de cohérence générique

---

- Protocole de cohérence supporte le meta-modèle
- Définit en 3 étapes:
  1. Concept de **ConstraintResolver**
  2. **Cycle de vie** d'une invocation
  3. Définition des **actions composites**, utilisant des resolvers

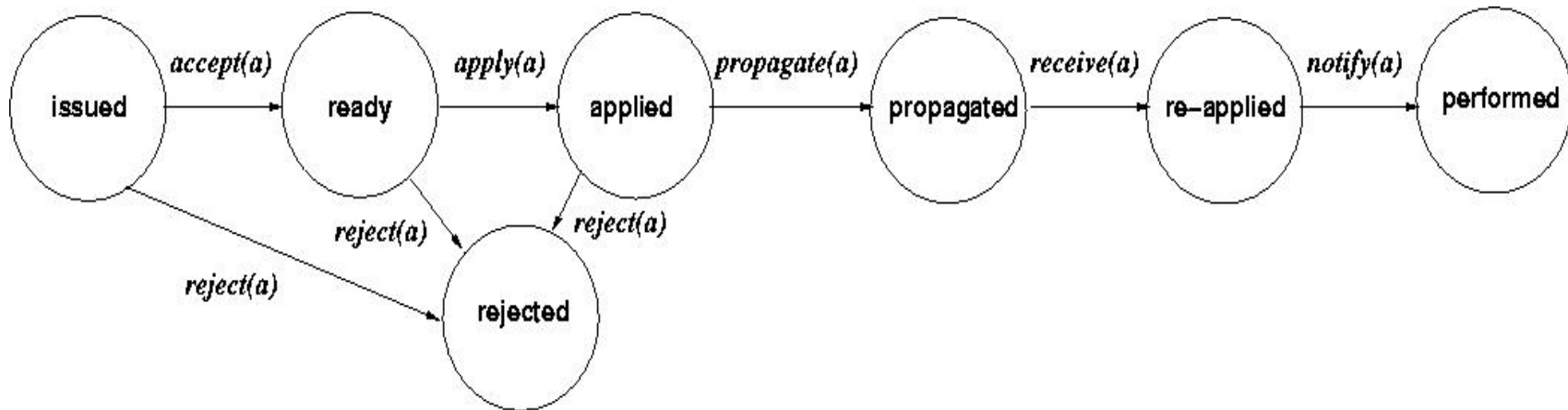


# Le concept de StateResolver

---

- **ConstraintResolver** associé a une contrainte particulière, que le resolver vérifie/satisfait
- 2 méthodes:
  - **check(Access\* a)**
  - **resolve(Access\* a)**
- 3 types de ConstraintResolver: **StateResolver**, **Scheduler**, **Propagator**
- On propose un StateResolver spécifique pour:
  - Chaque métrique de TACT
  - Les user session garanties de Bayou
  - Pour la dépendance sémantique (Lazy réplication)
- On utilise des algorithmes existants

# Le cycle de vie d'une invocation



# La définition des actions composites: *accept*

```
■ accept(a) {  
    conds = get _state _conditions(a)  
    if (conds) {  
        StateResolver = get _resolver(conds)  
        code = StateResolver- > resolve(a)  
        if (code != OK) {  
            reject(a); return;  
        }  
    }  
    mode = get _exec _mode(acc)  
    if (mode == pessimistic) {  
        code = Serializer- > resolve(a)  
        if (code != OK) {  
            reject(a); return  
        }  
    }  
    apply(a)  
}
```

# La définition des actions composites: *propagate*

```
■ propagate(a) {  
    period = get_transfer_instant(a)  
    Propagator->resolve(a, period)  
    mode = get_exec_mode(a)  
    if (mode == optimistic) {  
        code = Reconciliator->resolve(a)  
        if (code != OK)  
            reject(a)  
    }  
}}
```

# La définition des actions composites: *receive*

---

```
■ receive(a) {  
    code = Acceptor->check(a)  
  
    if (code == OK) {  
        apply(a)  
    }  
}
```



# Framework de coherence (1/3)

---

- Un modèle de cohérence est spécifié en XML
- Implémentation en C++
- **ConsistencyManager**: implémente les actions composites, en invoquant les méthodes des ConstraintResolvers
- **ReplicaWrapper**: offre la propriété de transparence de la gestion de cohérence

# Framework de coherence (2/3)





# Exemple du code généré

---

```
short f1(int i);
```

```
short AReplicaWrapper :: f1(int i) {  
    Access* acc = create(_f1, i);  
    short code = m_consMng->accept(acc);  
    if (code == SUCCESS) {  
        m_consMng->propagate(acc);  
        ParamWrapper<short>* wrap =  
        dynamic_cast<ParamWrapper<short>*>(acc->getWResult());  
        short result = wrap->get();  
        delete acc;  
        return result;  
    }  
    delete acc;  
    abort();  
}
```



# Framework de coherence (3/3)

---

- Propriétés du framework:
  - **Customisation** à partir d'un modèle de cohérence
  - **Séparation des concerns**: le code fonctionnel de l'application et la gestion de la cohérence
  - **Semi-transparence** de la gestion de la cohérence
  - **Granularité** fine de la cohérence: l'opération
  - **Extensibilité**: un nouveau StateResolver



# Conclusion et travaux en cours

---

- Protocole de cohérence générique
  - Support des réquis de cohérence qui varient d'une opération à l'autre
  - Transparence de la gestion de cohérence
- Evaluation de l'overhead des actions composites

