CoRDAGe

Co-déploiement et redéploiement d'applications sur grille

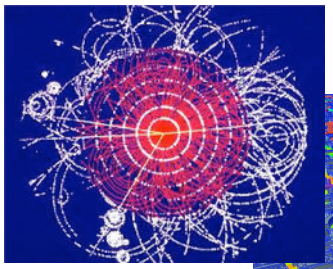
Loïc Cudennec
Projet PARIS, IRISA, INRIA Rennes
Université de Rennes 1
Avec le soutien de la Région Bretagne et de Sun Microsystems

15 janvier 2009

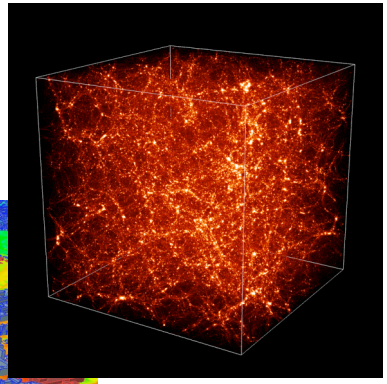
Directeurs de thèse : Luc Bougé et Gabriel Antoniu

Région Bretagne · Sun microsystems · INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE · INRIA · centre de recherche RENNES - BRETAGNE ATLANTIQUE · IRISA · Institut de Recherche en Informatique et Systèmes Aléatoires · UNIVERSITÉ DE RENNES 1 · ENS CACHAN BRETAGNE

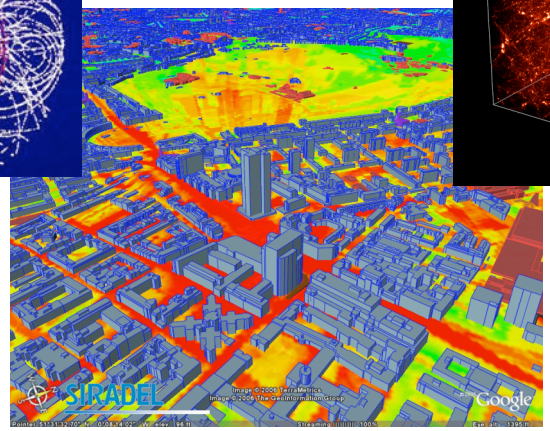
Contexte : applications



Détection du Boson de Higgs par le LHC (CERN)



Formation des structures de l'Univers (CEA)



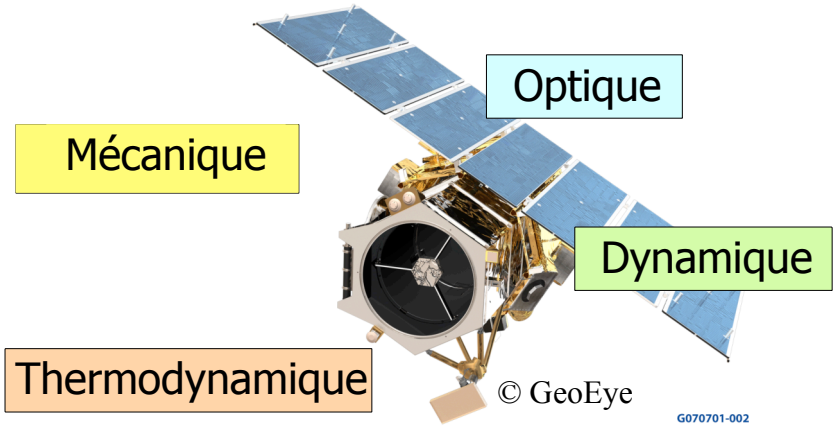
Propagation des ondes radio UMTS (SIRADEL)

Exemples d'applications

- Simulation numérique
- Conception collaborative
- Analyse de grandes masses de données

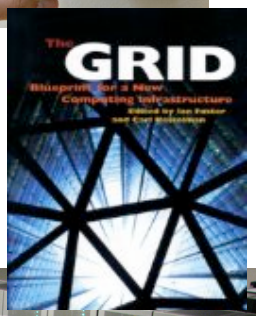
Nouveaux besoins en infrastructure

- Puissance de calcul
- Espace de stockage
- Bande passante



Contexte : grilles de calculateurs



Analogie avec le réseau de distribution d'électricité

- Simplicité d'utilisation : branchement sur une prise
- Transparence : la source de production et l'acheminement sont masqués

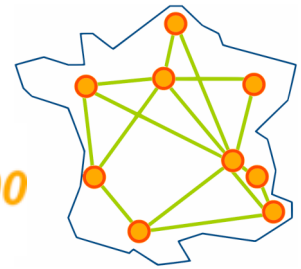
Définition de la grille (Foster 1998)

- Fédération de ressources physiques (calcul, transport et stockage)
- Interconnexion des ressources par un réseau hiérarchique

Propriétés de la grille

- Echelle : 1000 à 10000 ressources
- Ressources hétérogènes
- Dynamicité de l'infrastructure

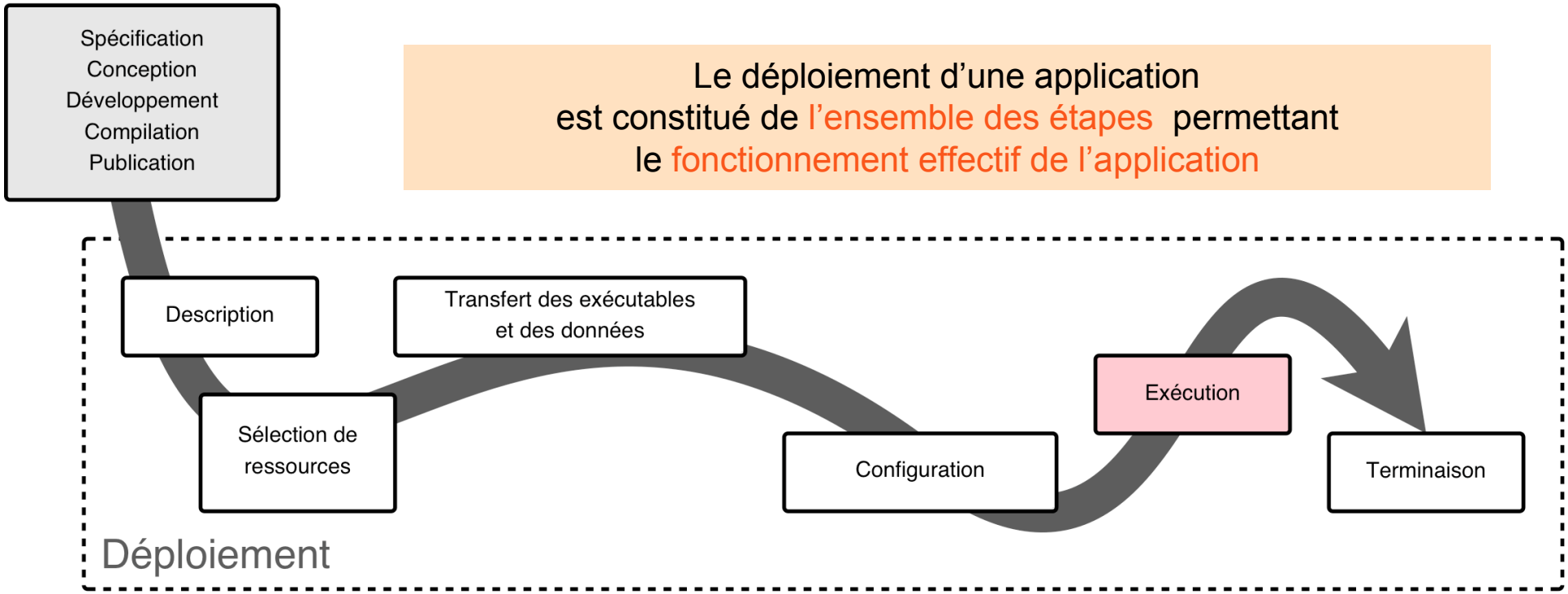
Grid'5000



Problématique : le déploiement d'applications

Le cycle de vie d'une application

- Spécification, conception, développement, compilation, exécution
- Déploiement



Le déploiement d'une application
est constitué de l'ensemble des étapes permettant
le fonctionnement effectif de l'application

Comment simplifier le déploiement sur les grilles de calculateurs
en prenant en compte la **dynamicité** de l'application et de l'environnement ?

Problématique : le déploiement dynamique

Dynamicité :

- Infrastructure
  - Apparition des ressources
  - Disparition des ressources
- Application
  - Expansion
  - Rétraction

Traduction au niveau de l'application :

- Migrations de tâches entre ressources (arrêt, redémarrage)
- Ajout et retrait de tâches

CoRDAGe

Etat de l'art

Déploiement : état de l'art

Objectif : prendre en compte la dynamicité

Transparence

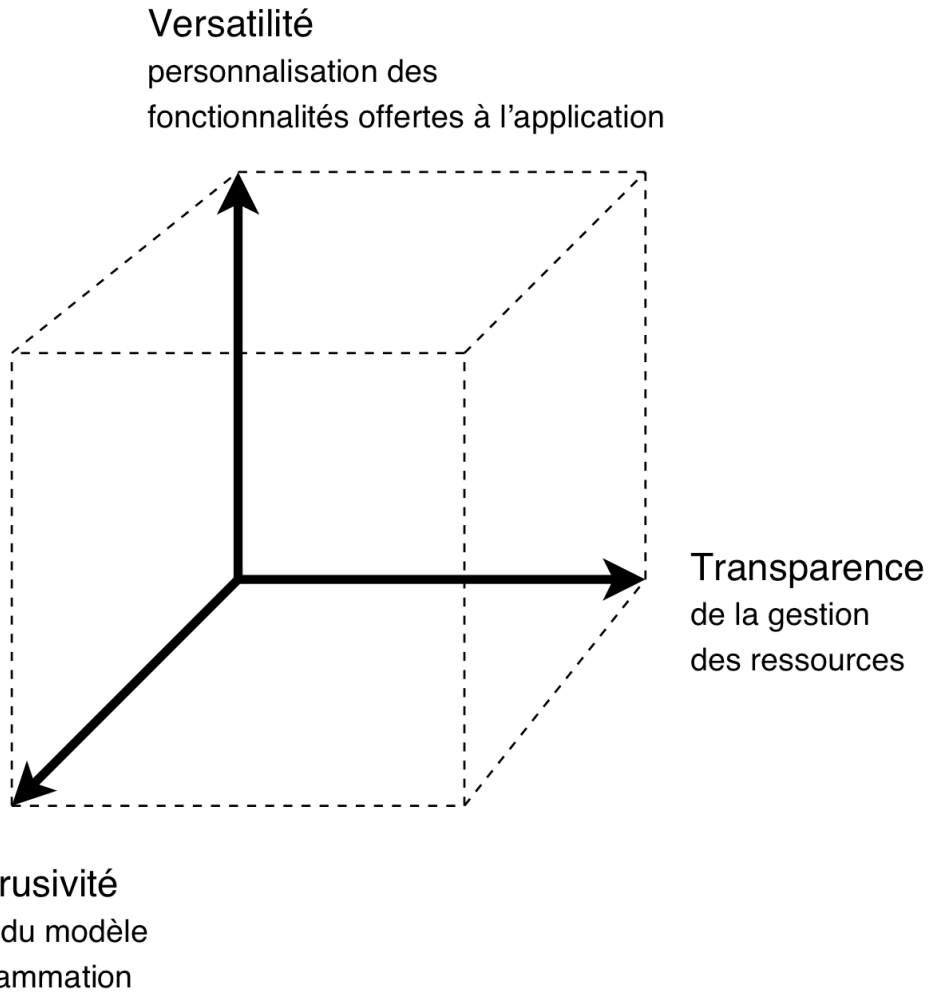
- Masquer la réservation et le déploiement des ressources physiques

Versatilité

- Permettre à l'application de fournir des actions spécifiques

Non-intrusivité

- Ne pas imposer un modèle de programmation



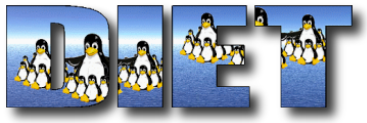
Déploiement intégré avec un environnement d'exécution externalisé

Principe

- L'utilisateur soumet son application à un environnement d'exécution
- L'environnement prend en charge tout le déploiement

Exemples

- Au niveau des services :
  GridSolve (ICL), DIET (ENS Lyon), Zorilla (Vrije U. Amsterdam)
- Au niveau du système d'exploitation :
  XtreemOS (Europe), Vigne (IRISA)



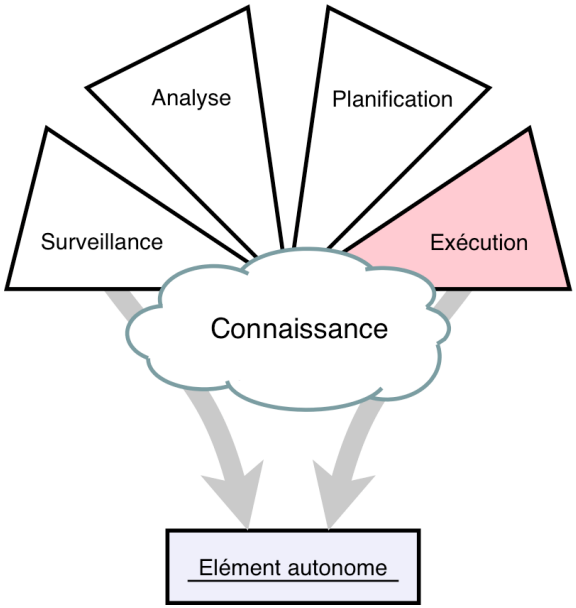
Discussion

- Dynamicité : principalement adapté pour l'équilibrage de charge et la migration de tâches
- Permet difficilement de personnaliser l'interface avec l'application

Déploiement dans le contexte de l'informatique autonome

Principe

- Autogestion des systèmes
- Proposition d'architecture en 4 phases (IBM 2001)



Exemples

- Canevas de développement :
  Jade (INRIA Grenoble), Dynaco (IRISA)
- Environnement d'exécution :
  Entropy (INRIA, EMN Nantes), GrADS (RICE)

Discussion

- Dynamicité : contrôle de tous les aspects liés à l'autonomie
- Propriétés de transparence et versatilité
- Nécessite souvent l'utilisation d'un modèle de programmation à base de composants

GrADS
Grid Application Development Software Project

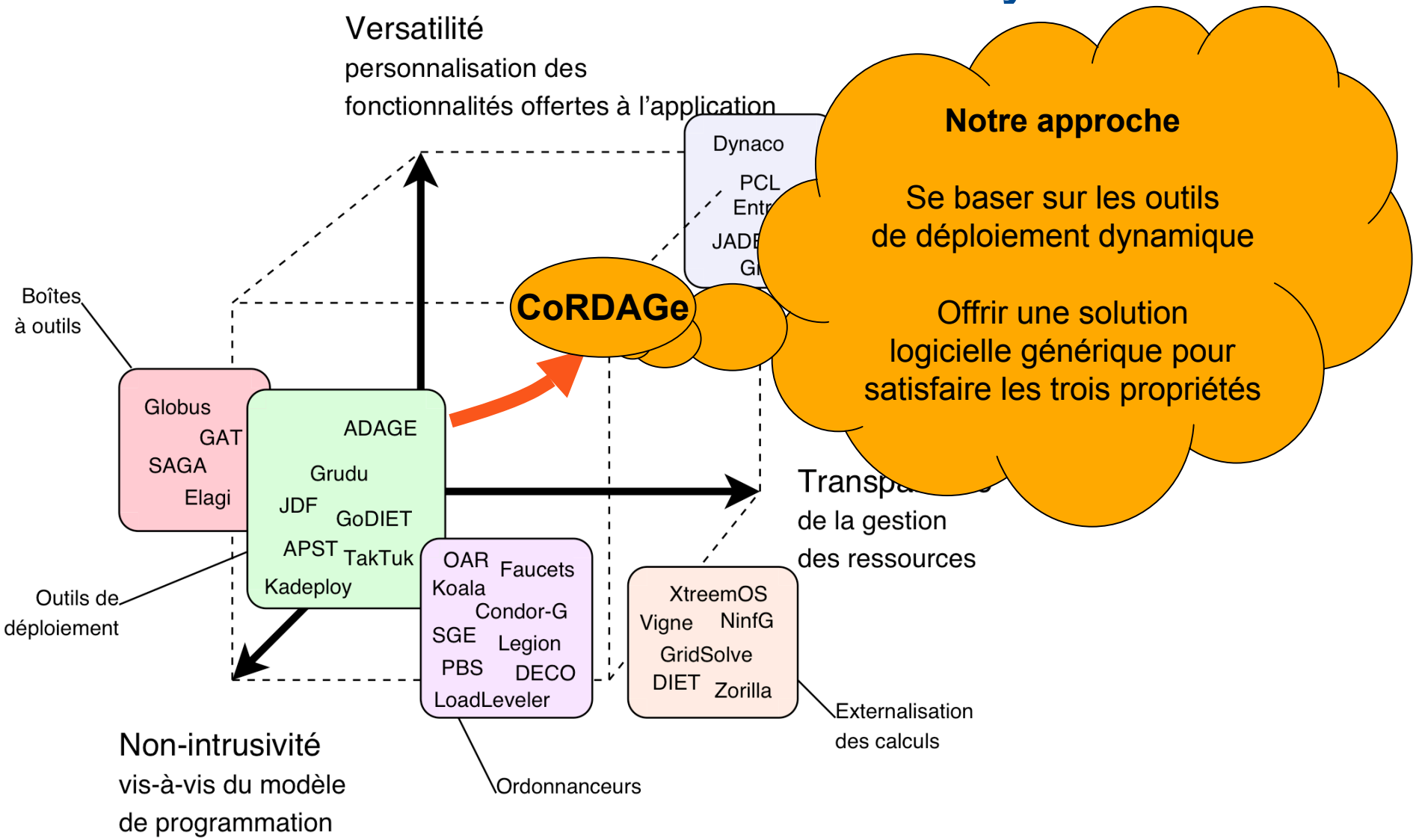
Déploiement explicite : outils et services pour la gestion de la grille

- Systèmes d'information
  - Construire une vision globale ou locale des ressources
  - NWS (UCSB), Ganglia (Berkeley)
- Ordonnanceurs
  - Organiser, planifier l'exécution des tâches
  - Condor-G (WISC), Faucets (UIUC), Koala (TU Delft), SGE (Sun), OAR (INRIA Grenoble)
- Boîtes à outils
  - Offrir un ensemble d'utilitaires bas-niveau pour faciliter l'utilisation de l'infrastructure
  - Globus (ANL), GAT (GridLab), Saga (LSU/VU)
- Outils de déploiement
  - Configurer les ressources et démarrer les tâches
  - JDF (Sun/IRISA), GoDIET, Grudu (ENS Lyon), Kadeploy (INRIA Grenoble), ADAGE (IRISA), DeployWare (LIFL)

Discussion

- Approche non-intrusive pour l'application
- Dynamicité : peu supportée
- Aucun système n'offre les trois propriétés (transparence, versatilité et non-intrusivité) en même temps

Positionnement des différents systèmes



Contributions et plan de l'exposé

Etude de cas
- Prise en charge de différents types d'applications (JXTA, JuxMem, Gfarm)

Etude des mécanismes mis en jeu pour
- Le déploiement coordonné de plusieurs applications (co-déploiement)
- Le redéploiement d'applications

Proposition d'un modèle pour le déploiement dynamique
- Générique
- Satisfaisant les trois propriétés : transparence, versatilité et non-intrusivité

Proposition d'une architecture et d'un prototype
- CoRDAGe (IRISA, équipe PARIS)
- Basé sur OAR et ADAGE

Validation
- Fonctionnelle dans le cadre des projets LEGO et Respire de l'ANR
- Expérimentale dans le contexte de Grid'5000

CoRDAGe

Etude de cas

Point de départ : déploiement dynamique avec ADAGE

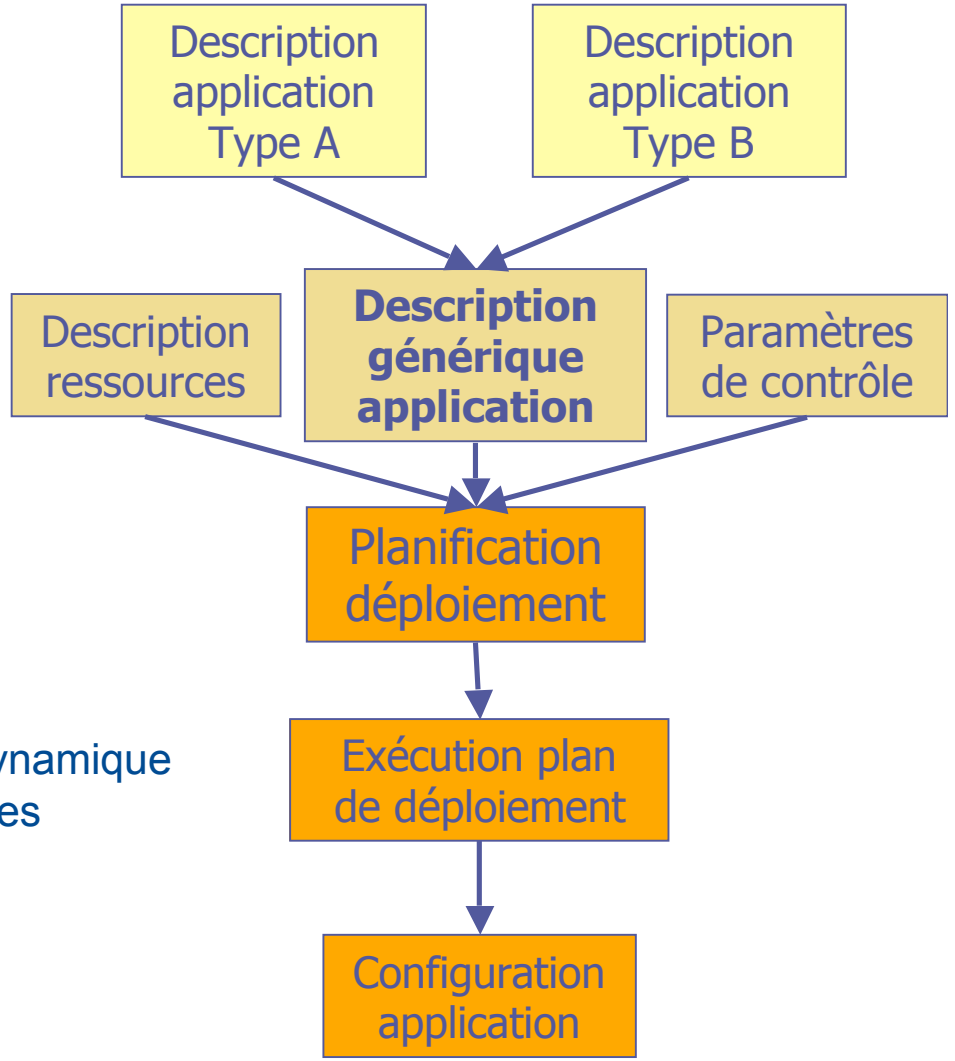
ADAGE (IRISA, équipe PARIS)

Automatisation du déploiement d'applications

Support pour les déploiements complexes

Limites

- L'utilisateur est au centre du déploiement
- Utilisation difficile, encore plus en mode dynamique
- Pas de transparence pour la réservation des ressources physiques



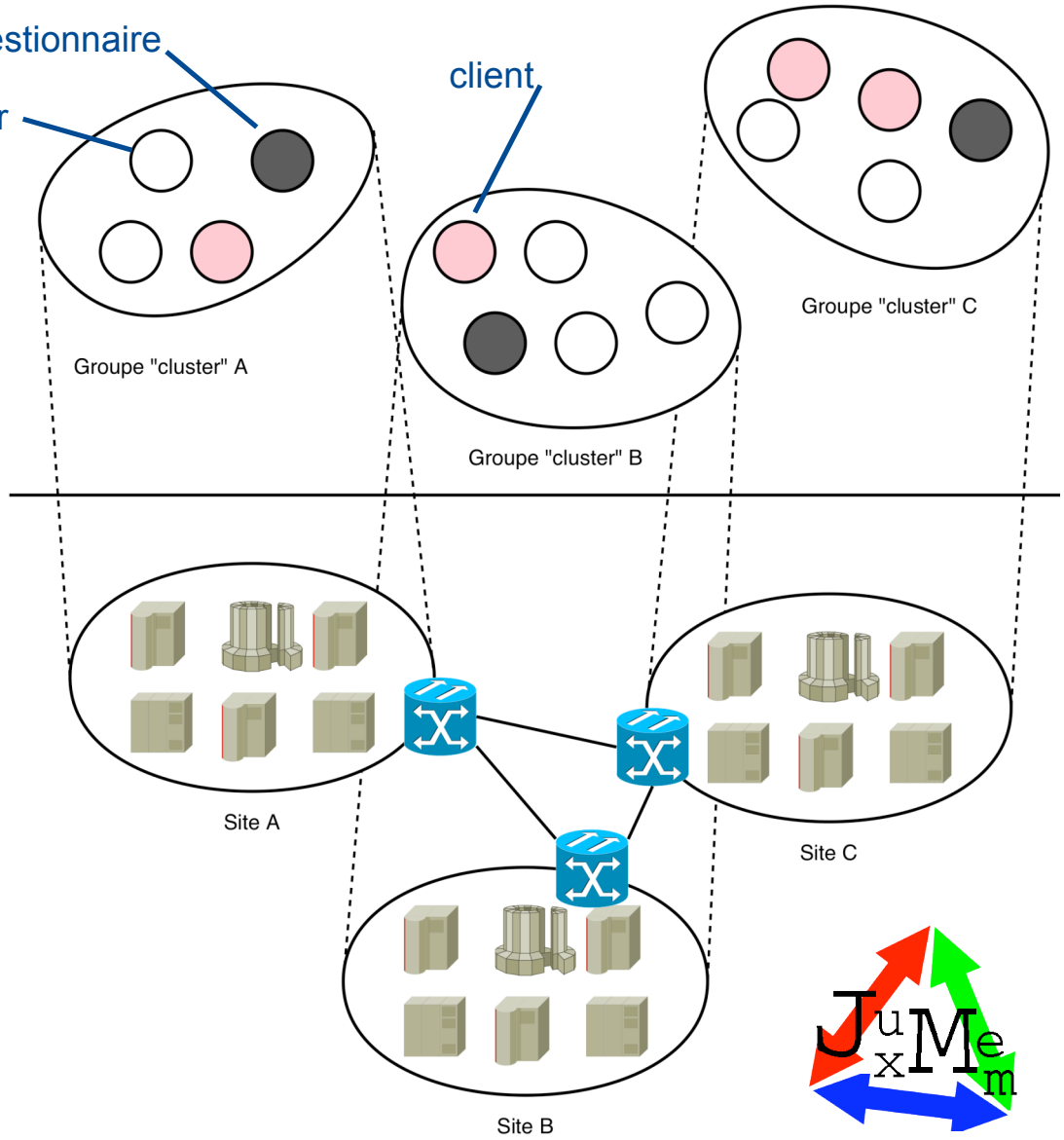
Etude de cas : un service de partage de données

JuxMem (IRISA, équipe PARIS)

- Transparence
- Cohérence des données
- Tolérance aux défaillances

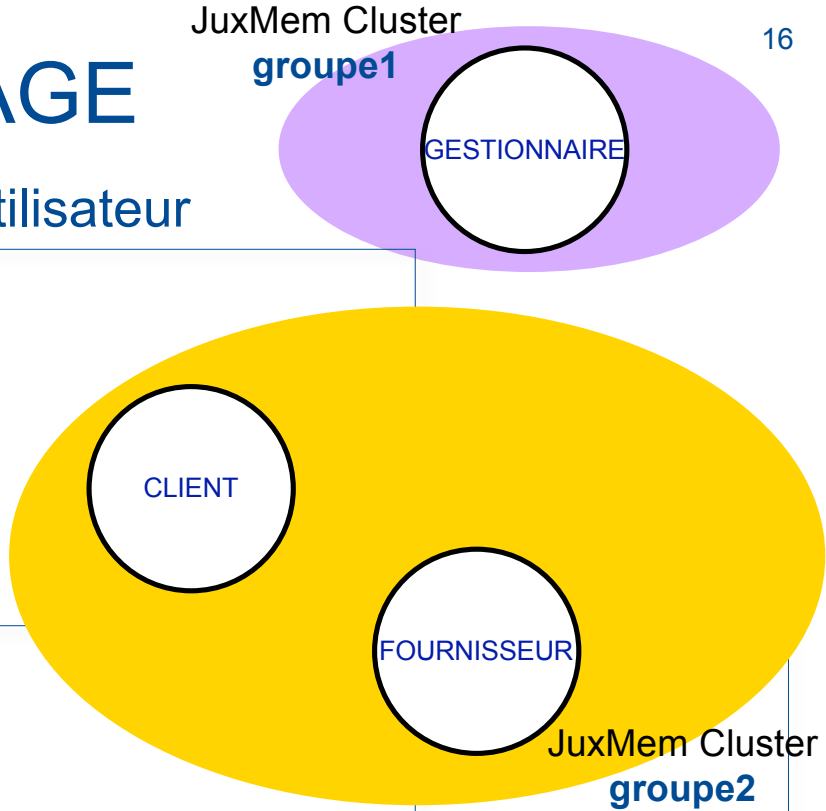
Dynamicité

- Taille espace de stockage liée aux nombre de fournisseurs
- Evolution dans le temps



Déployer JuxMem avec ADAGE

1. Description de l'application fournie par l'utilisateur
2. Réservations de ressources effectuées par l'utilisateur avec OAR
3. Contraintes de placement fournies par l'utilisateur

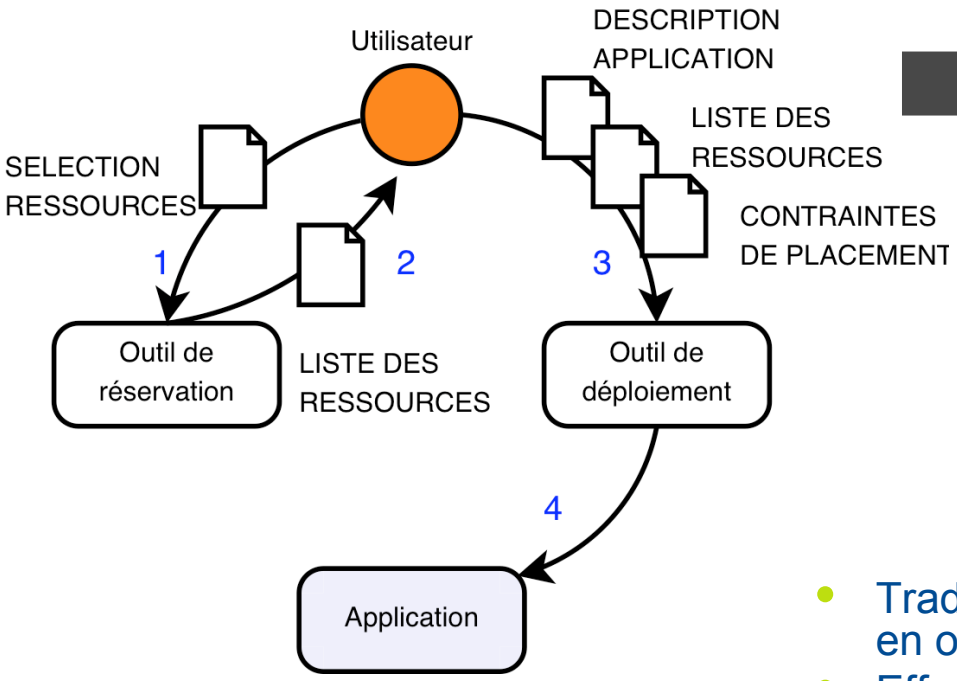


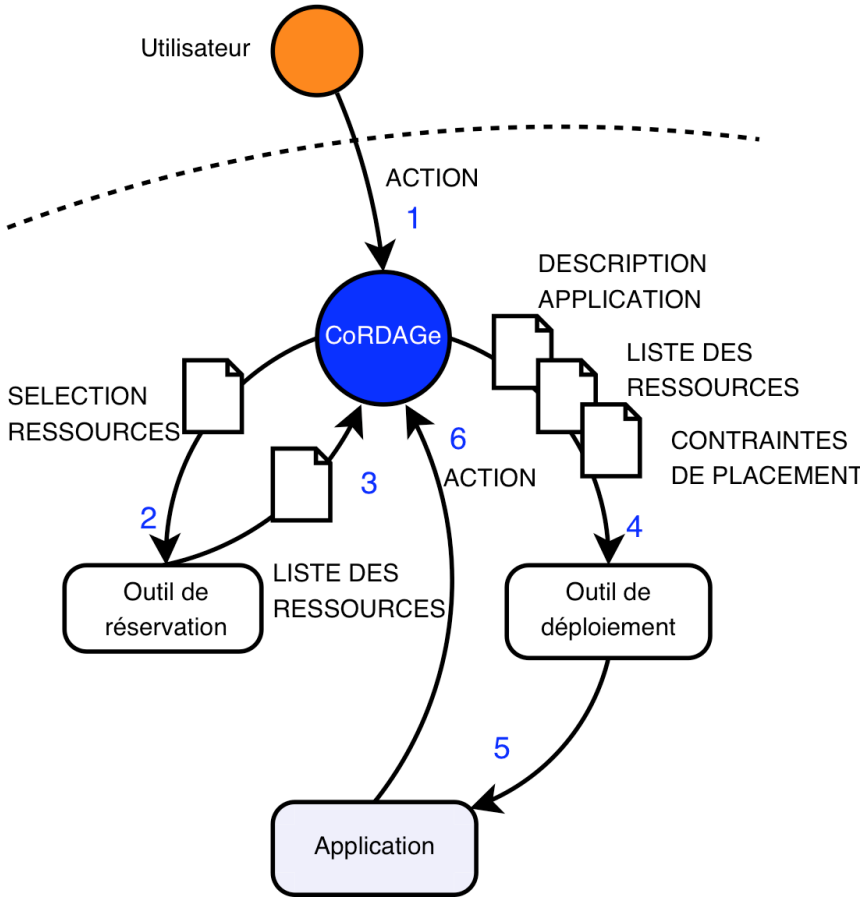

```
<JUXMEM_application>

  <juxmem_cluster id="groupe1">
    <managers>
      <manager id="gestionnaire" port="9876" uptime="60" />
    </managers>
    <providers />
    <clients />
  </juxmem_cluster>
<ctrl_params>
  <submission_method name="oarsh"/>
  <juxmem_cluster id="groupe2" rdv="groupe1">
  <planner mode="RoundRobin"/>
    <managers/>
  <transfer_files method="oarcp" type="cfg|data"/>
    <providers>
  <placement_constraints>
      <provider id="fournisseur" port="9876" uptime="60" mem="10000" card="1" />
    </providers>
      <associate id="gestionnaire" id_type="pattern" match_type="resgroup" match="oargrid-17283-group"/>
      <associate id="fournisseur" id_type="pattern" match_type="resgroup" match="oargrid-17291-group"/>
    <clients>
      <associate id="client" id_type="pattern" match_type="resgroup" match="oargrid-17292-group"/>
      <client id="client" port="9871" bin="juxmem-writer"
  </placement_constraints>
                args="@managerhostname 4" card="1" />
</ctrl_params>
    </clients>
  </juxmem_cluster>

</JUXMEM_application>
```

Contribution : la vision CoRDAGe

Co-déploiement et redéploiement d'applications sur grilles

- Interfacer les applications avec les outils de réservation et de déploiement
- Assister l'application pendant le temps d'exécution





- Traduire des actions de haut niveau en opérations de bas niveau
- Effectuer une pré-planification du déploiement

CoRDAGe

Modèle

CoRDAGe : un modèle pour la représentation des applications

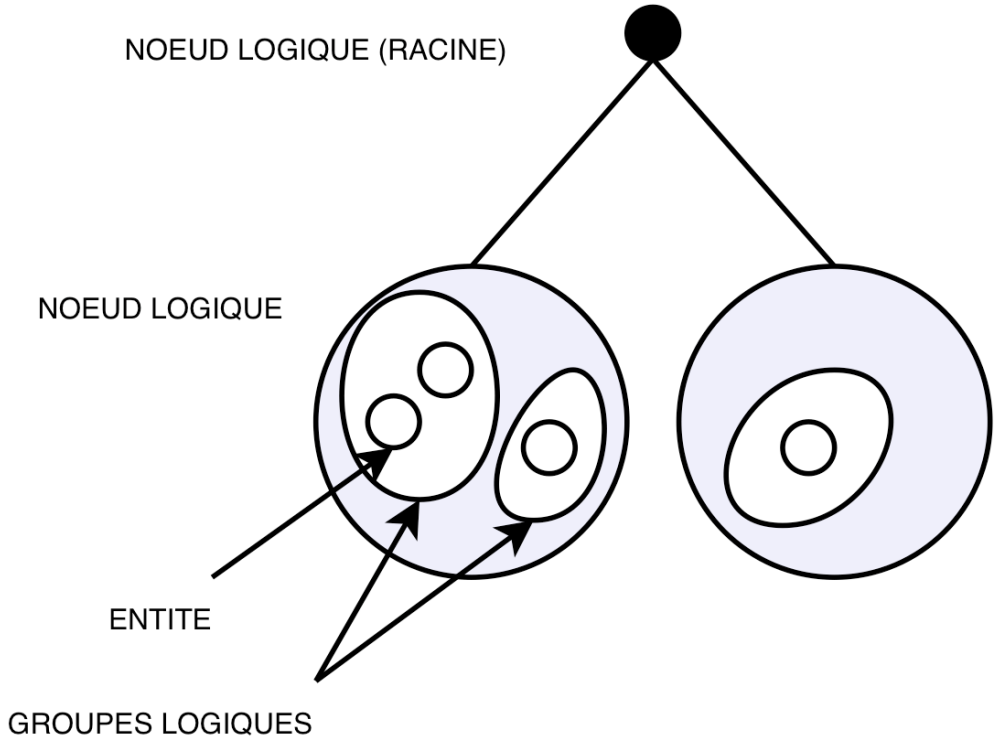
CoRDAGe

Représentation logique

- Entité
- Groupe logique
- Nœud logique

Organisation hiérarchique

- Arbre logique



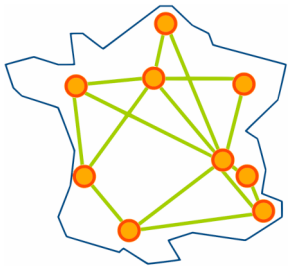
CoRDAGe : un modèle pour la représentation des ressources

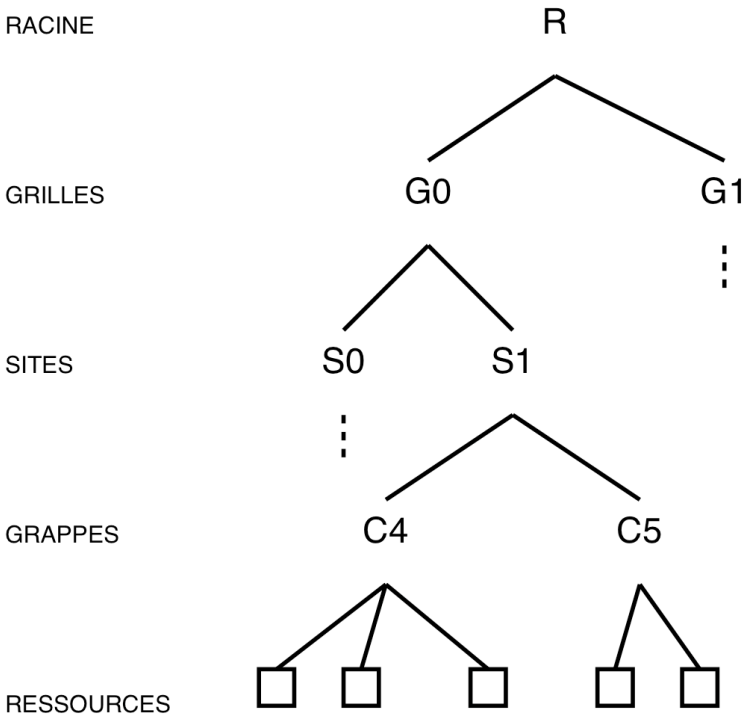
CoRDAGe

Représentation logique

- Ressource
- Groupe physique

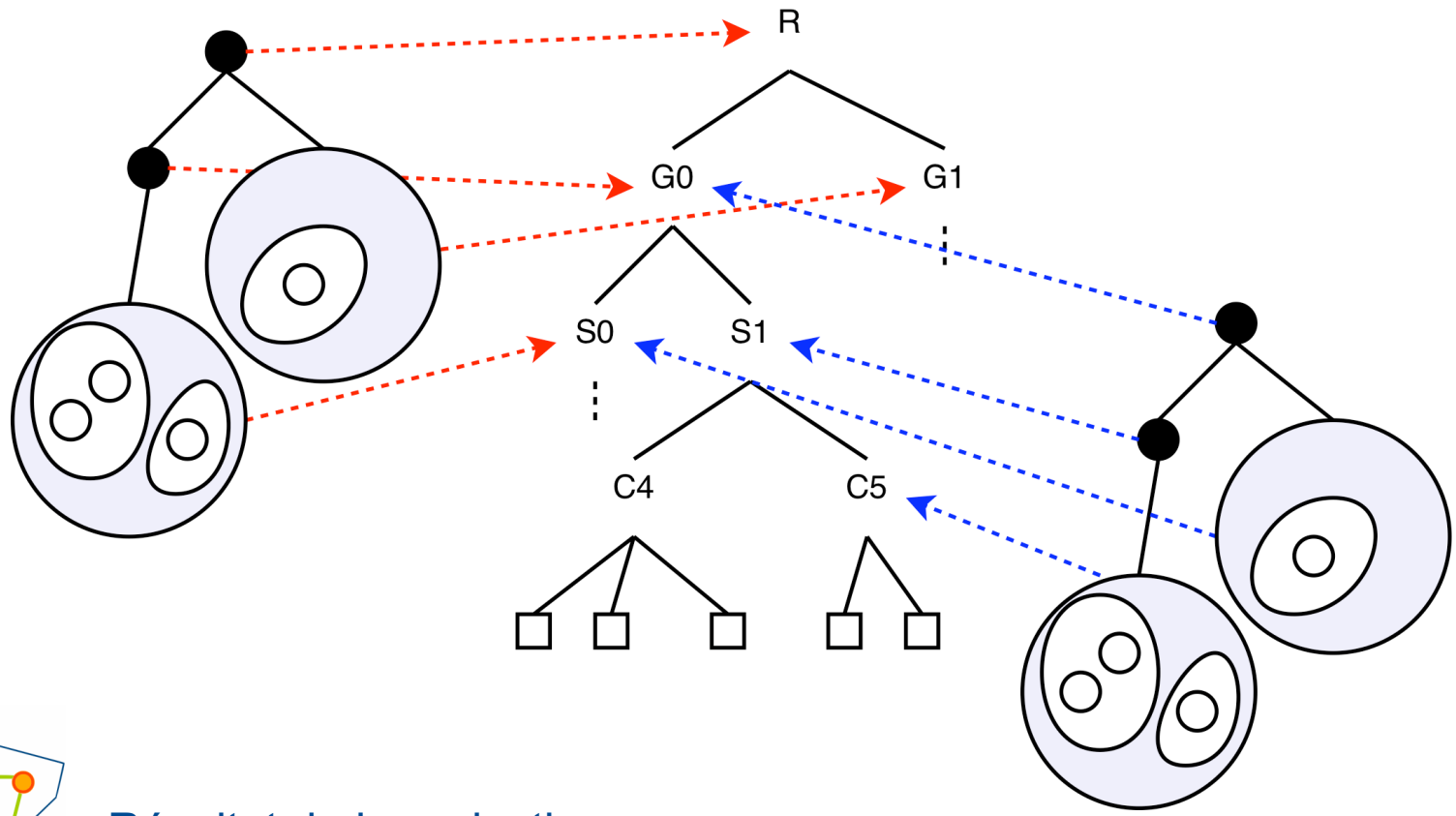
Représentation hiérarchiques

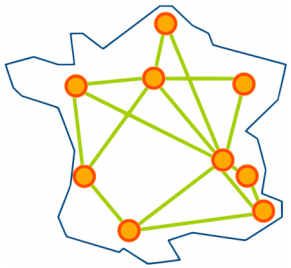




Déploiement simple : projection des représentations

CoRDAGe



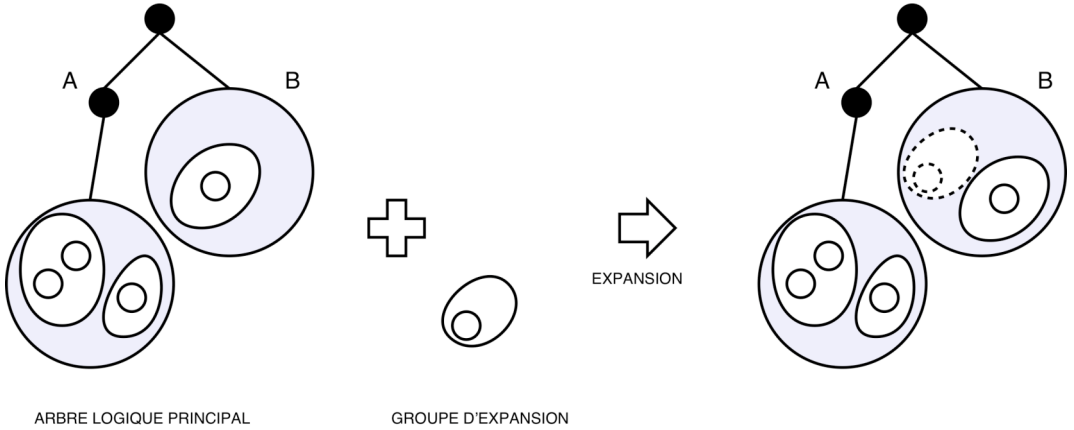


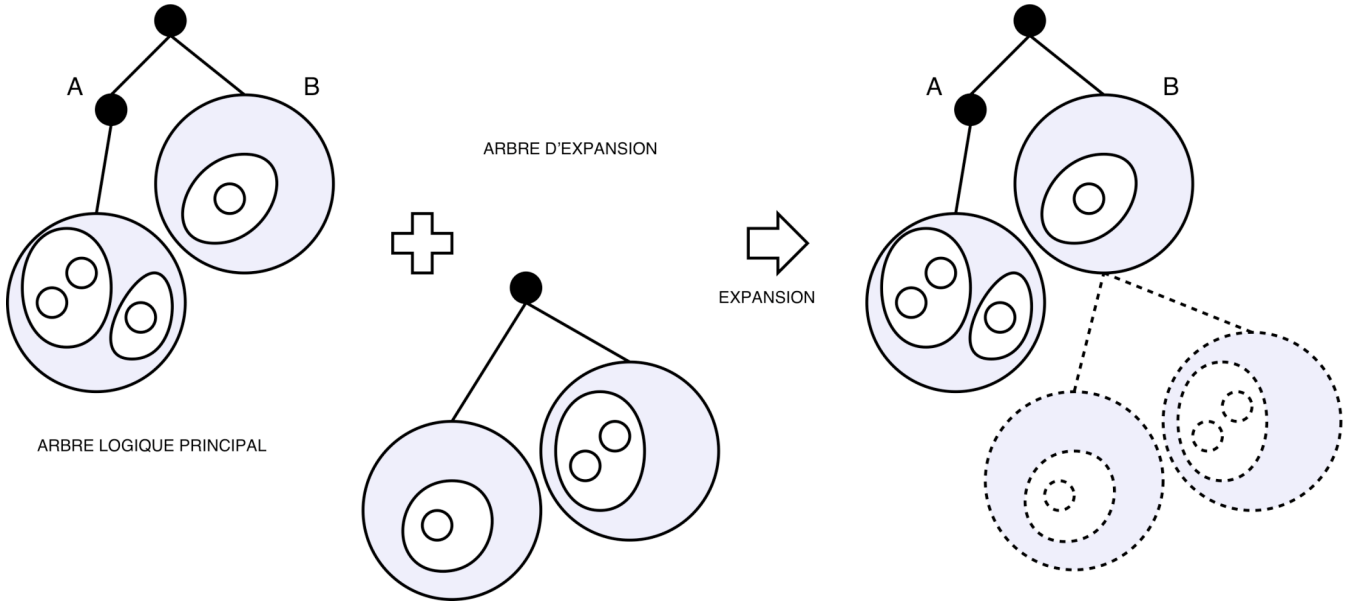
Résultat de la projection

- Ensemble d'identifiants pour la réservation des ressources
- Ensemble de contraintes de placement entre entités et ressources

Expansion de la représentation

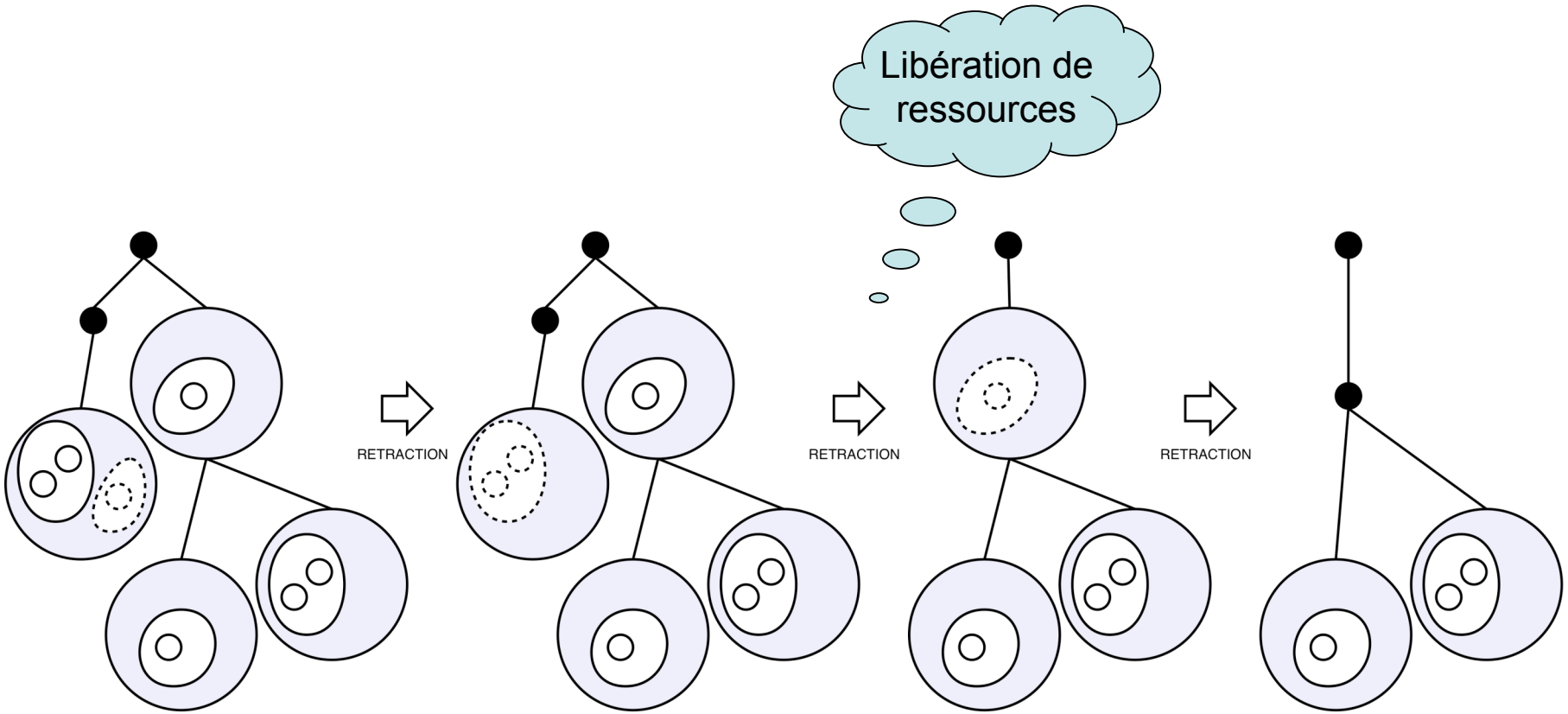
CoRDAGe





Rétraction de la représentation

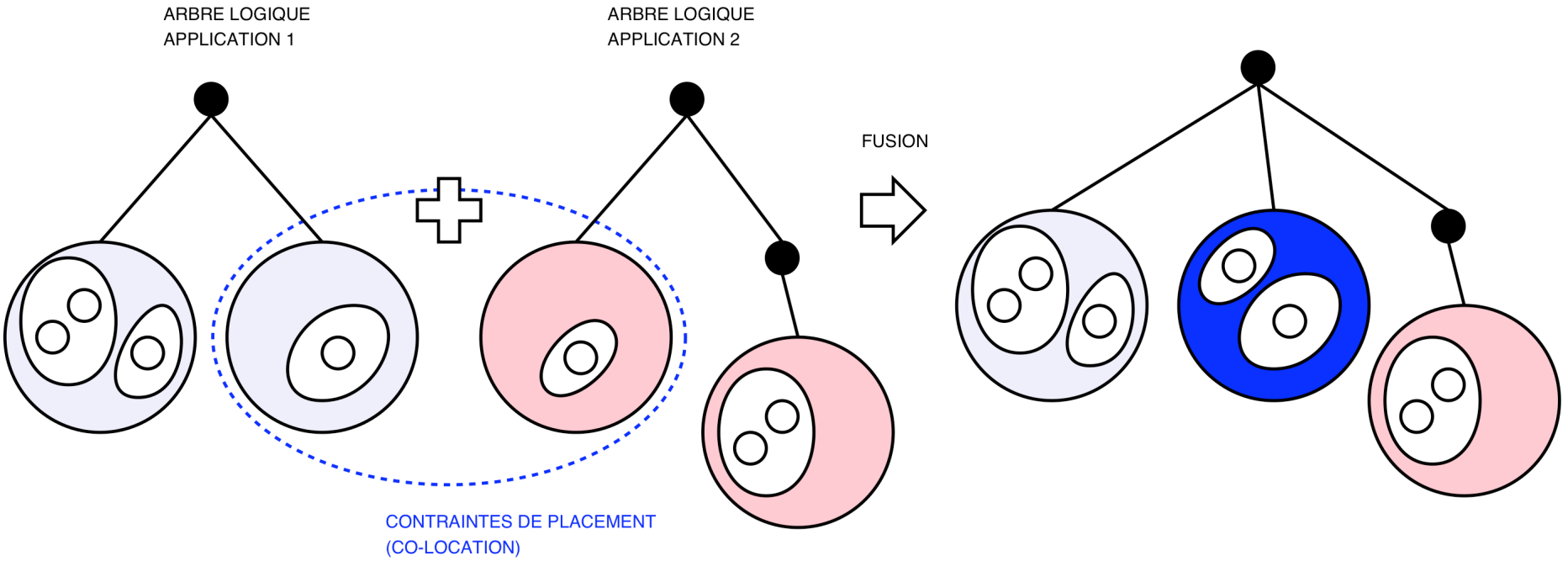
CoRDAGe



Gérer le co-déploiement

CoRDAGe

Objectif : déployer deux applications couplées, de manière coordonnée en prenant en compte des contraintes temporelles et de placement



Discussion sur le modèle

CoRDAGe

Un modèle dédié au déploiement dynamique

- redéploiement
- co-déploiement

Transparence

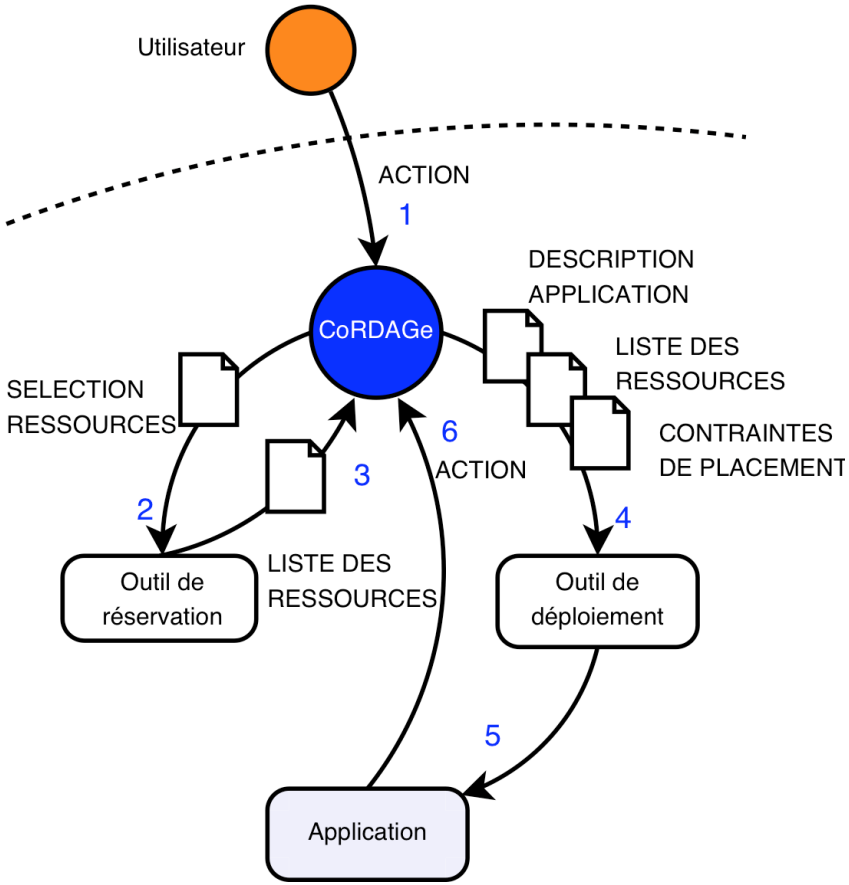
- Interfaçage entre l'application et les outils et services de la grille
- Gestion transparente des réservations et des déploiements

Versatilité

- Gérer des actions personnalisées à chaque type d'application

Non-intrusivité

- Au niveau architectural

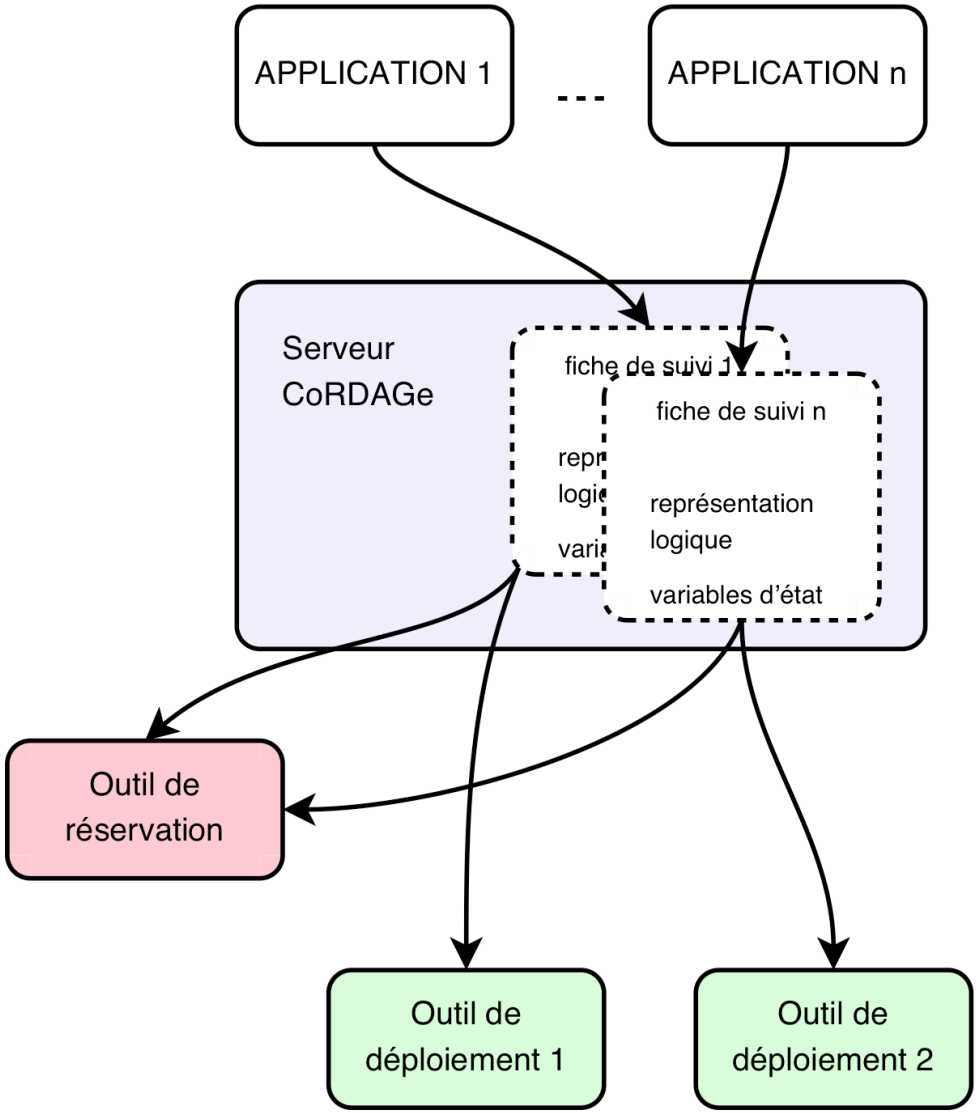


CoRDAGe

Architecture

Architecture : vue d'ensemble

CoRDAGe



Interface client : intégration

CoRDAGe

- Interface basée sur l'appel de procédure à distance (RPC)
- Deux classes d'actions :
  - **Générique** : *déclarer*, *configurer*, *déployer*, *surveiller*
  - **Spécifique** : actions personnalisées pour un type d'application

```
#include cordage.hh
cdgclient* app1 = new cdgclient(conf_app1);
cdgclient* app2 = new cdgclient(conf_app2);

app1->add_sub_app(app2);
app1->deploy(walltime);
app2->perform_action(ADD_PROVIDER, 2, "mon_groupe");

delete app2;
delete app1;
```

Actions génériques

Action spécifique

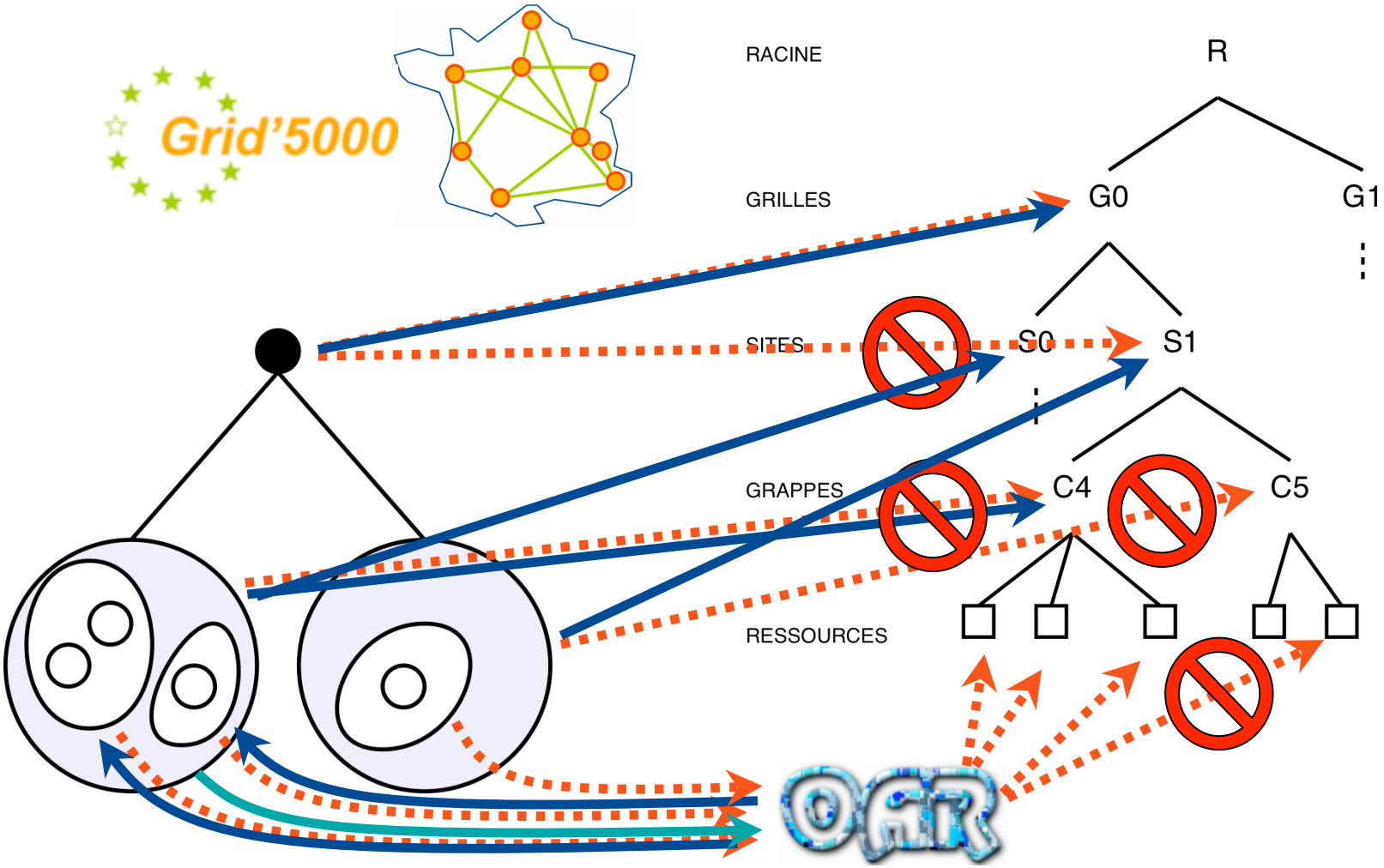
Exemple d'action générique : le déploiement

CoRDAGe

```
fonction deploiement ()
{
                                        // génération de la représentation logique de notre application
  genere_representation_logique(description_application);

                                        // pour toutes les fiches de suivi correspondant aux sous-applications
  pourtout(SA appartenant aux sous-applications) {
                                        // demande récursive de construction de la représentation logique
    SA->deploiement();
                                        // fusion avec notre propre représentation logique
    arbre_logique->fusion(SA->arbre_logique);
  }

                                        // si nous sommes la racine de l'arbre des sous-applications
  si (racine()) alors {

    projection(arbre_logique, arbre_physique);

    outil_deploiement->ajout_contraintes_placement(arbre_logique);

    outil_deploiement->ordre_deploiement();
  }
}
```

Un point difficile : la projection

CoRDAGe



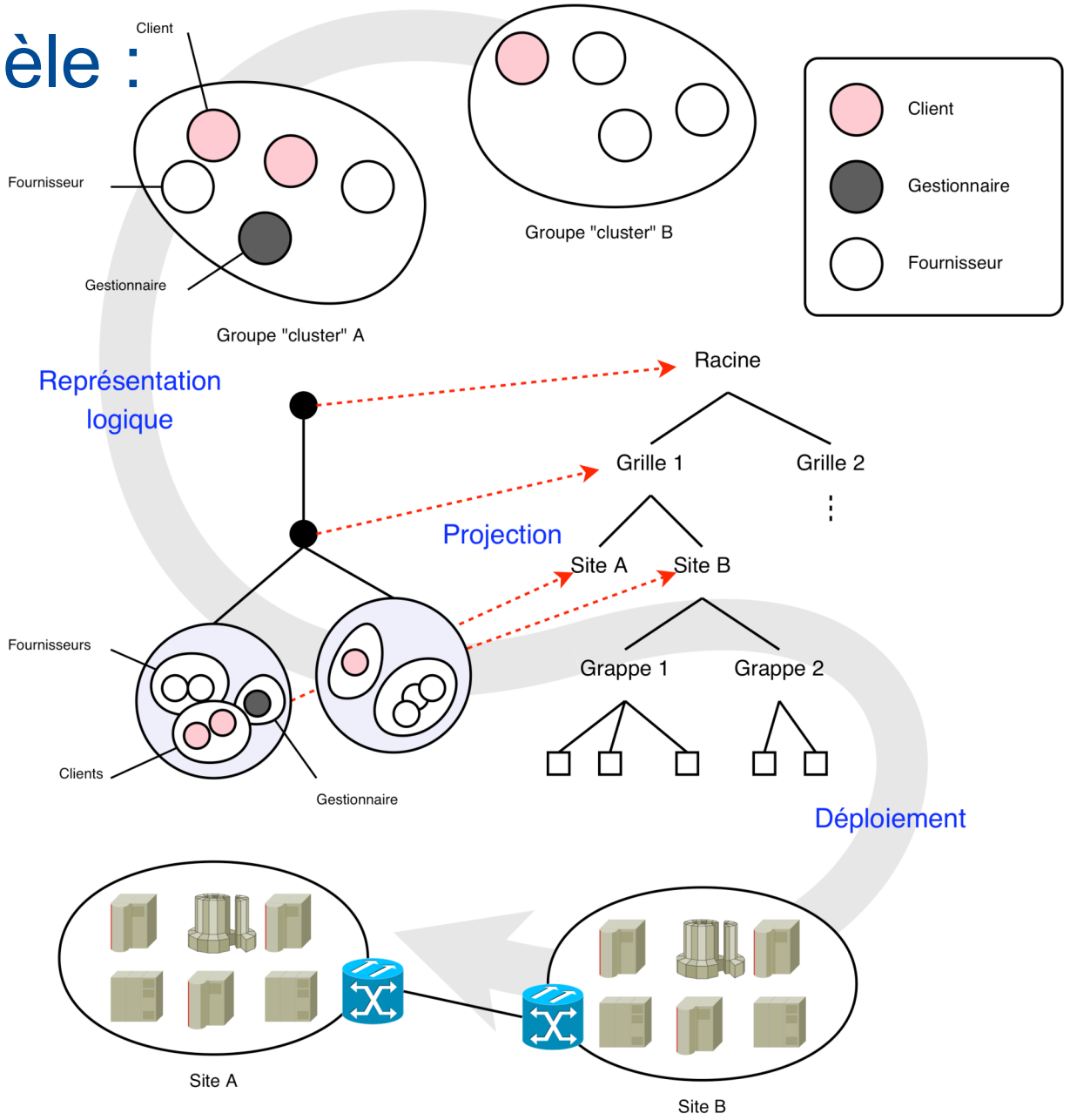
Libération des ressources

Outil de réservation

CoRDAGe

Validation fonctionnelle

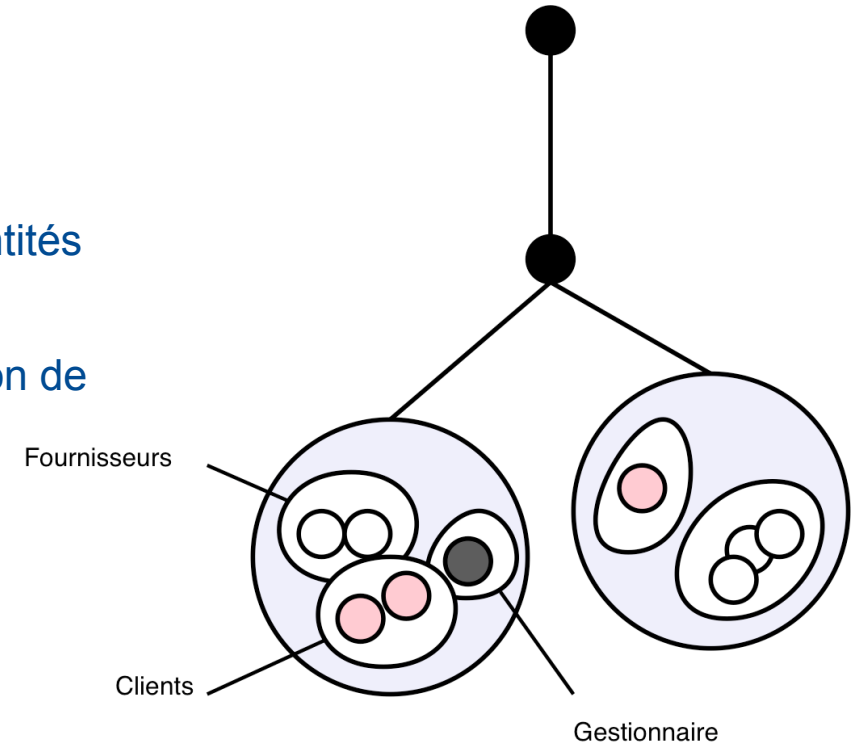
Validation du modèle : déploiement dynamique dans JuxMem



Actions spécifiques à JuxMem

Exemple : Ajout de nouveaux *fournisseurs*

- *Action* : ADD_PROVIDER
- *Paramètres* : **n** = nombre de fournisseurs
  c = identifiant du cluster
- *Effet* :
  - Ajoute un groupe logique contenant **n** entités fournisseurs dans le nœud logique correspondant à **c**
  - Commande le déploiement de l'expansion de l'arbre modifié



Intégration dans le code de JuxMem

Démarrage d'une entité (gestionnaire, fournisseur, client)

```
#include cordage.hh

cdgclient* cordage = new cdgclient(conf.juxmem);
```

Terminaison d'une entité (gestionnaire, fournisseur, client)

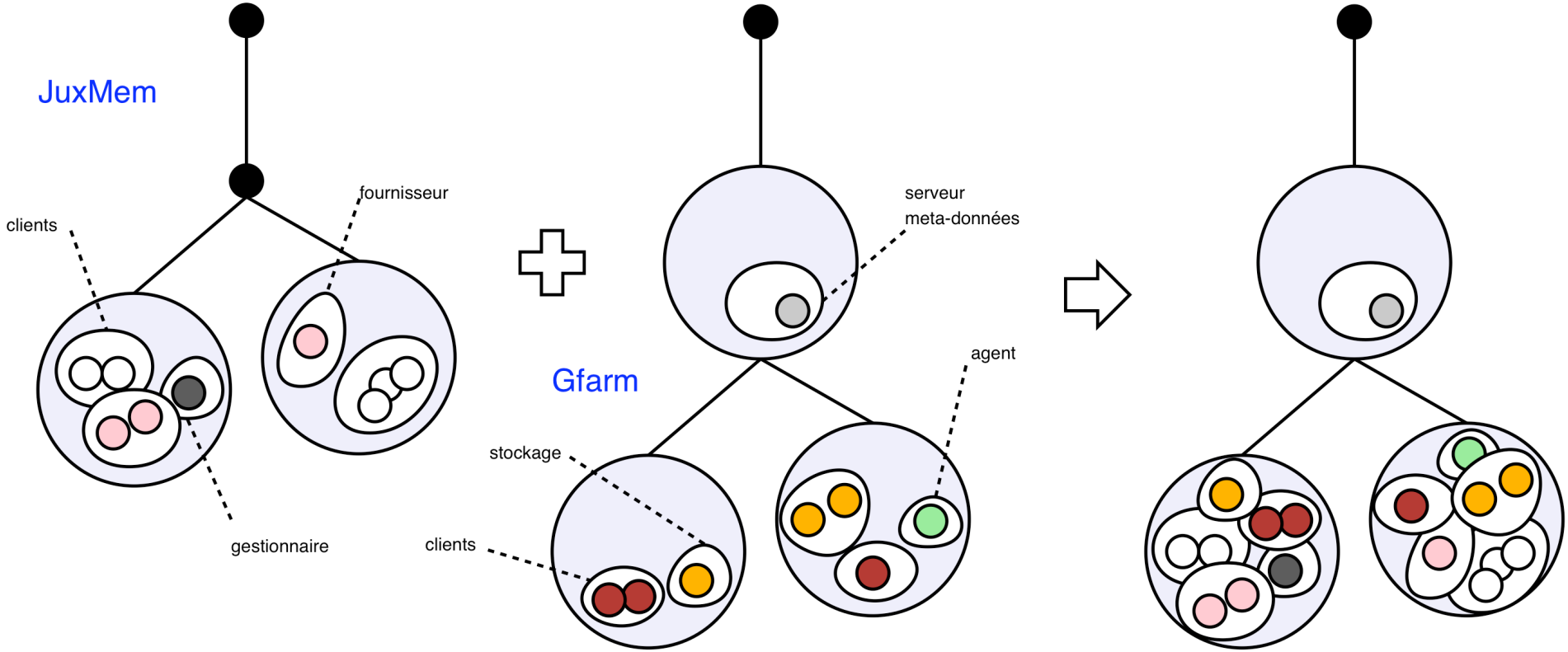
```
cordage->discard();
delete cordage;
```

Echec du traitement d'une requête d'allocation mémoire (gestionnaire)

```
cordage->perform_action( ADD_PROVIDER,
                         nb_fournisseurs);
```

=> Une dizaine de lignes de code

Validation (2) : co-déploiement JuxMem + Gfarm (AIST/Univ. Tsukuba)




```
cdgclient* cordage_juxmem = new cdgclient(conf_juxmem);
cdgclient* cordage_gfarm = new cdgclient(conf_gfarm);
cordage_juxmem->add_sub_app(cordage_gfarm);
cordage_juxmem->deploy(walltime);
```

CoRDAGe

Validation expérimentale

Validation expérimentale

CoRDAGe

Objectifs

- Déploiement statique : impact sur le coût du déploiement du
  - Nombre de groupes logiques par noeud
- Déploiement dynamique : impact sur le coût du déploiement du
  - Nombre de clients
  - Nombre d'applications
- Déploiement large échelle : impact du multi-site

Méthodologie expérimentale

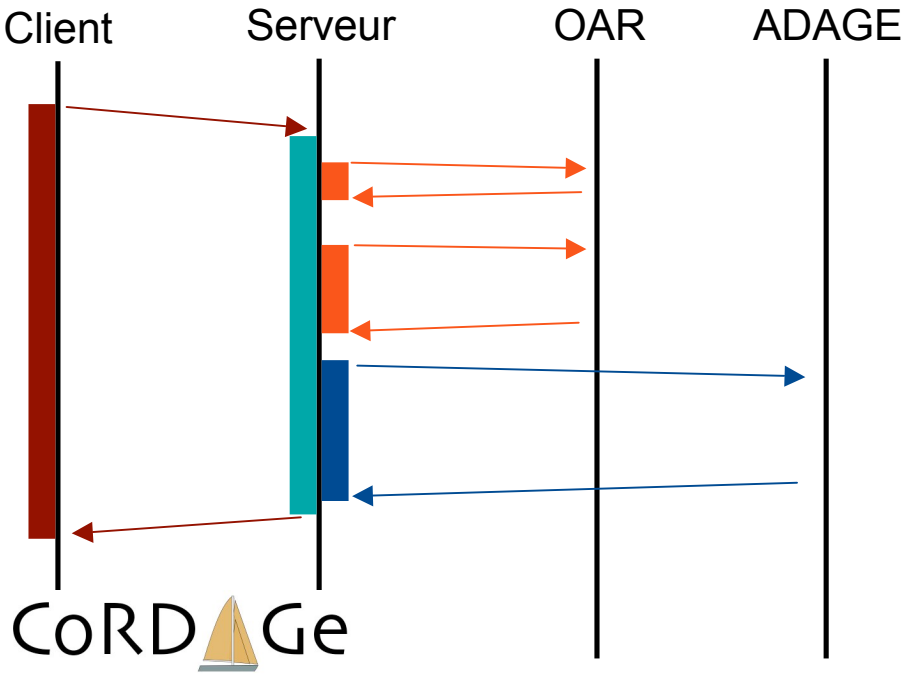
Grid'5000

Protocole

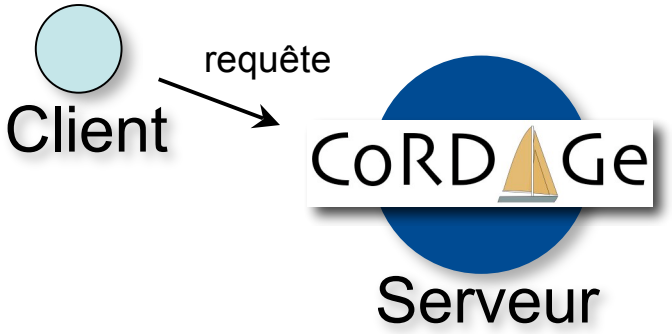
- Lancement du **serveur** CoRDAGe sur la frontale de Rennes
- Interactions avec l'ordonnanceur **OAR** sur chacun des sites de la grille
- Utilisation d'une instance de l'outil de déploiement **ADAGE** par application

Configuration

- Jusqu'à 386 ressources réparties dans 6 sites de Grid'5000
- Configurations statiques puis dynamiques

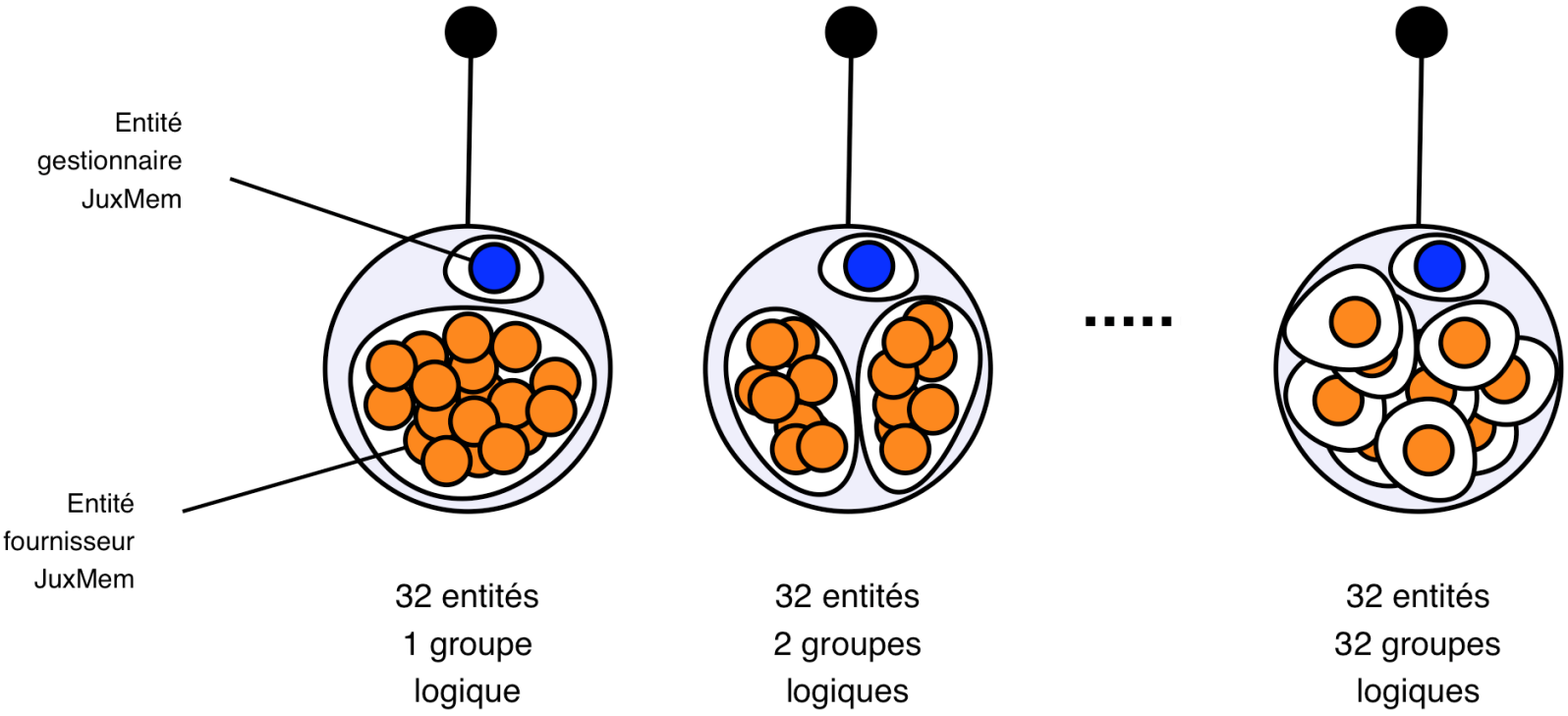


Impact du nombre de groupes par nœud sur le coût du déploiement



Expérimentation statique

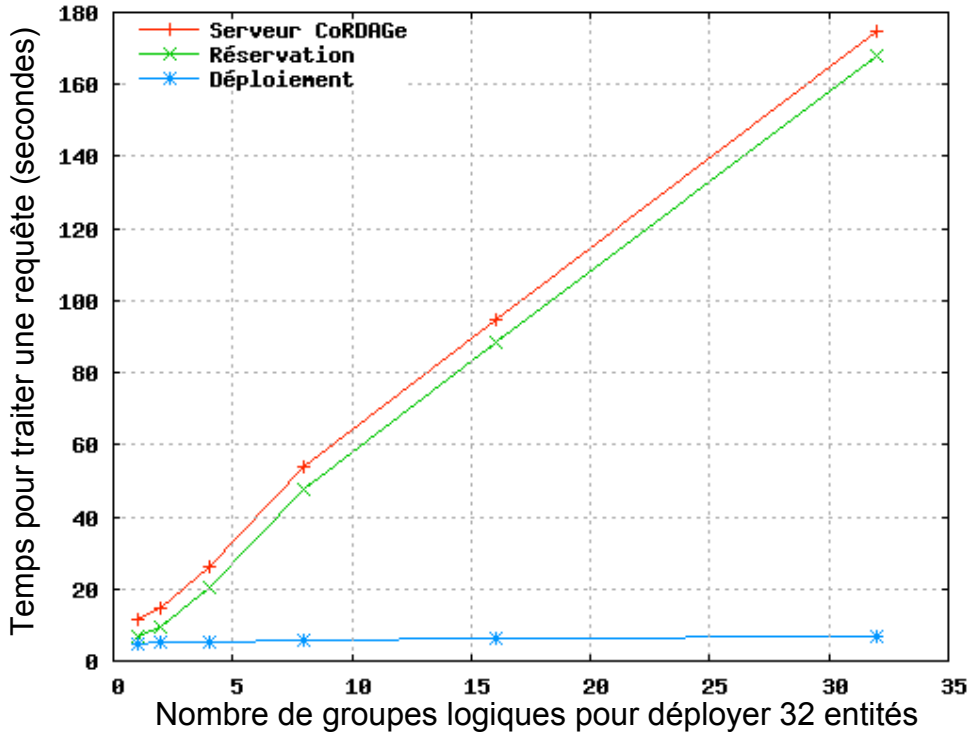
- 1 déploiement par expérience
- 32 entités réparties dans
  1, 2, 4, 8, 16 puis 32 groupes logiques

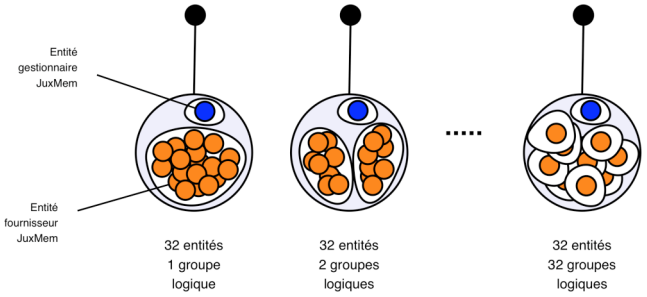


Impact du nombre de groupes par nœud sur le coût du déploiement

Résultats

- Augmentation du temps de traitement liée à OAR
- Faible surcoût lié à CoRDAGe : moins de 1,5% du temps total





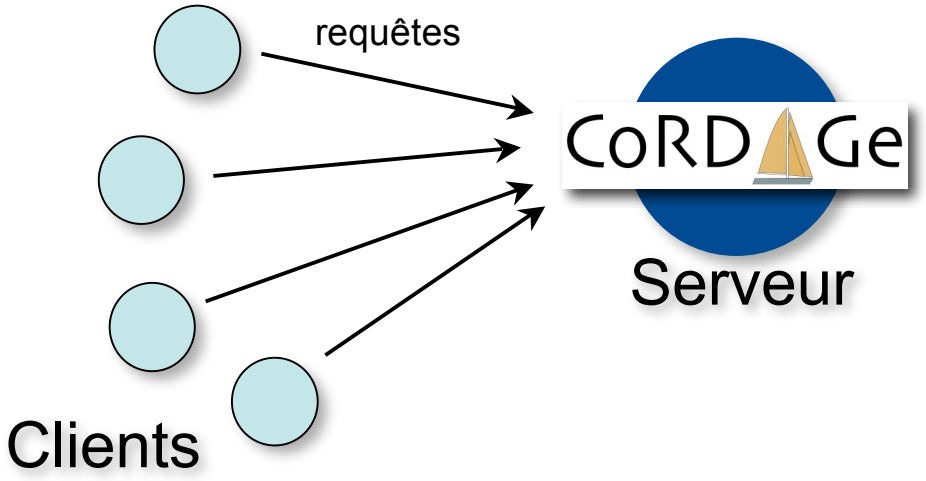
Efficacité du déploiement

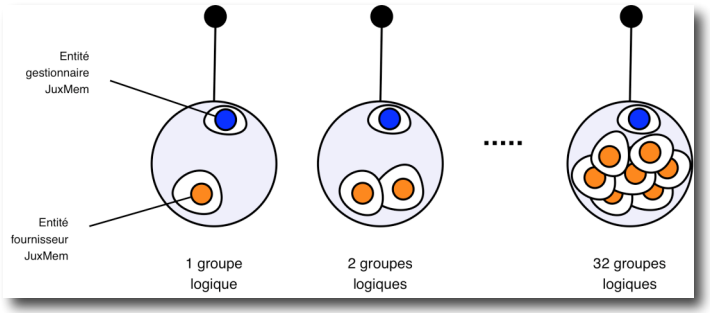
Gestion fine des réservations

Impact du nombre de clients sur le coût du déploiement

Expérimentation dynamique

- 1 serveur CoRDAGe
- 1 application de type JuxMem
- Plusieurs clients : 1, 2 puis 4 clients
- Requêtes de déploiement d'un nouveau fournisseur JuxMem
- Requêtes séquentialisées par le serveur





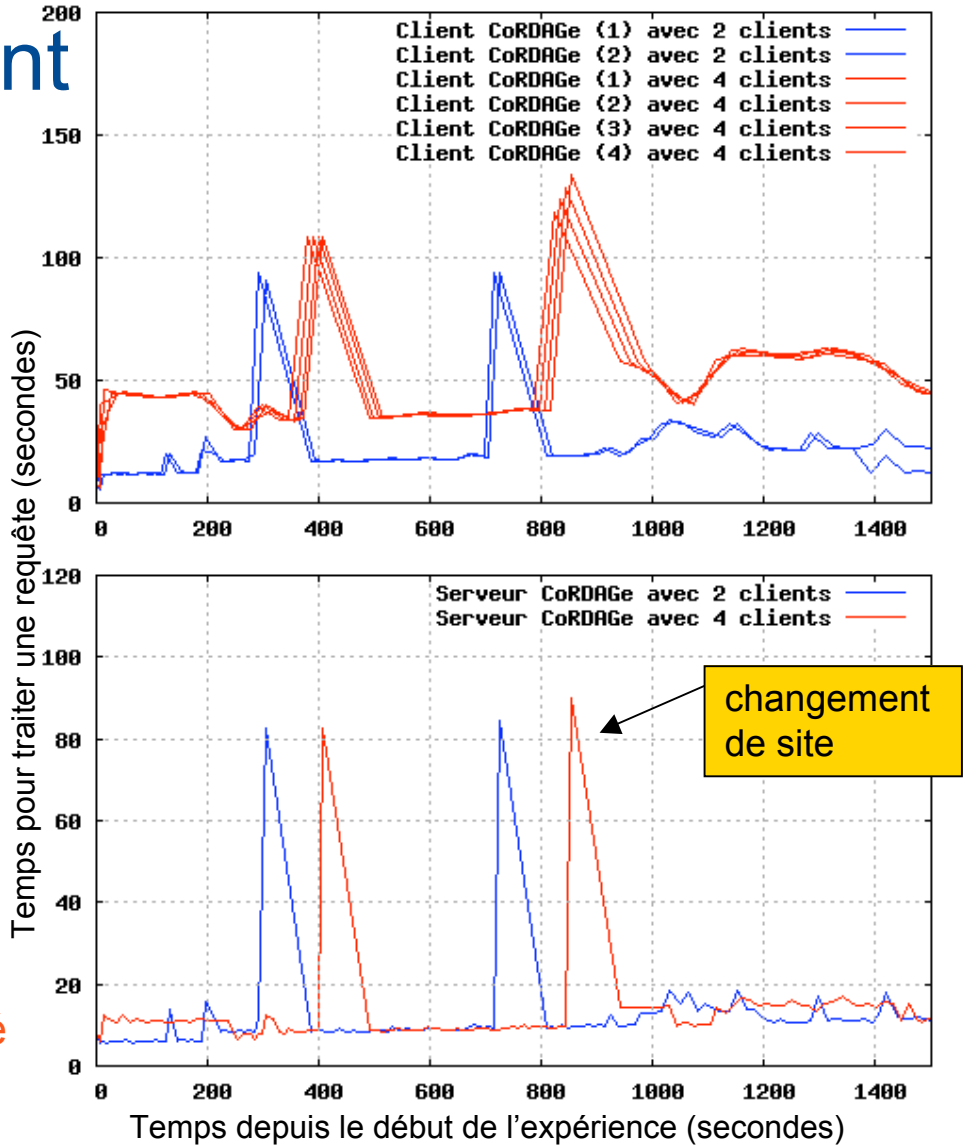
Impact du nombre de clients sur le coût du déploiement

Résultats

- 2 clients
  - 150 requêtes au total en 28 minutes
  - Temps moyen sur le serveur : 11 secondes
  - Temps moyen sur les clients : 21 secondes
- 4 clients
  - 143 requêtes au total en 29 minutes
  - Temps moyen sur le serveur : 12 secondes
  - Temps moyen sur les clients : 46 secondes

Conclusions

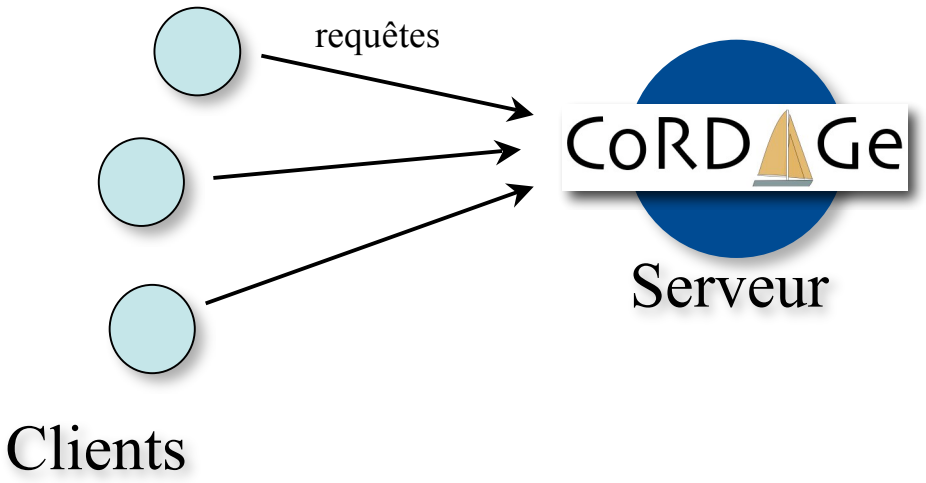
- Pas de surcoût sur le serveur
- Surcoût lié à la synchronisation répercuté sur le client

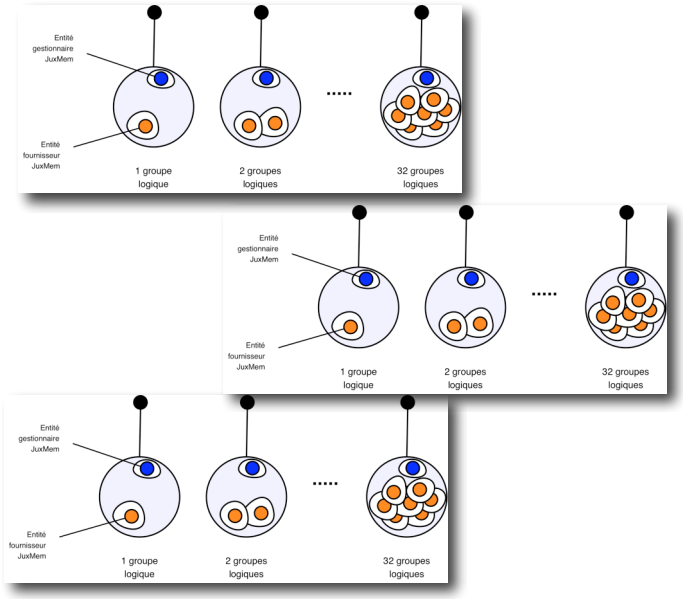


Impact du nombre d'applications sur le coût du déploiement

Expérimentation dynamique

- 1 serveur CoRDAGe
- Plusieurs applications de type JuxMem : 1, 2 puis 3 applications
- 1 client par application
- Requêtes de déploiement d'un nouveau fournisseur JuxMem
- OAR partagé entre toutes les applications, une instance d'ADAGE par application





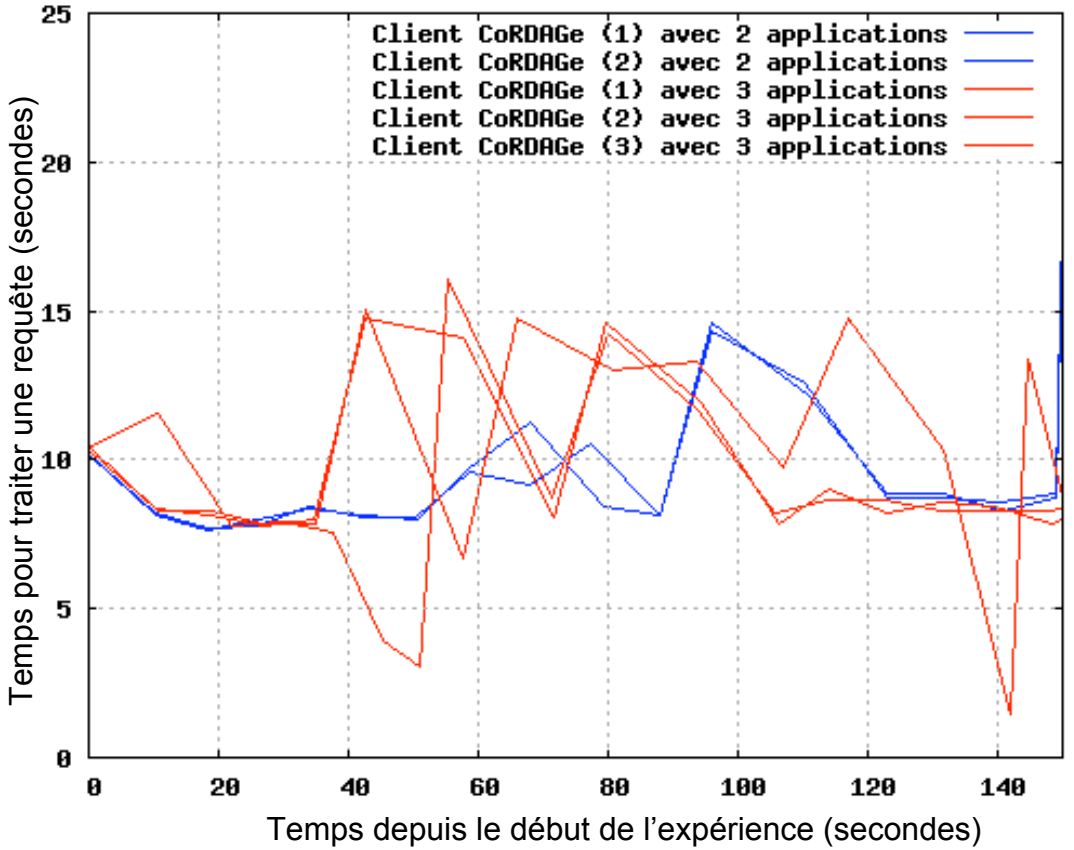
Impact du nombre d'applications sur le coût du déploiement

Résultats

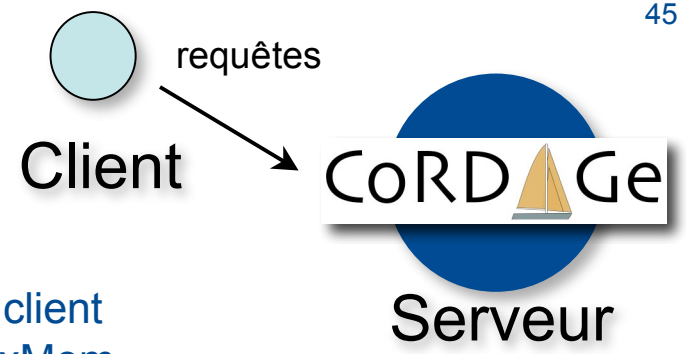
- 3 applications
  - Temps moyen sur le client : 10 secondes

Conclusions

- Pas de surcoût significatif lors de la gestion de plusieurs applications
- Requêtes traitées en parallèle par le serveur CoRDAGe

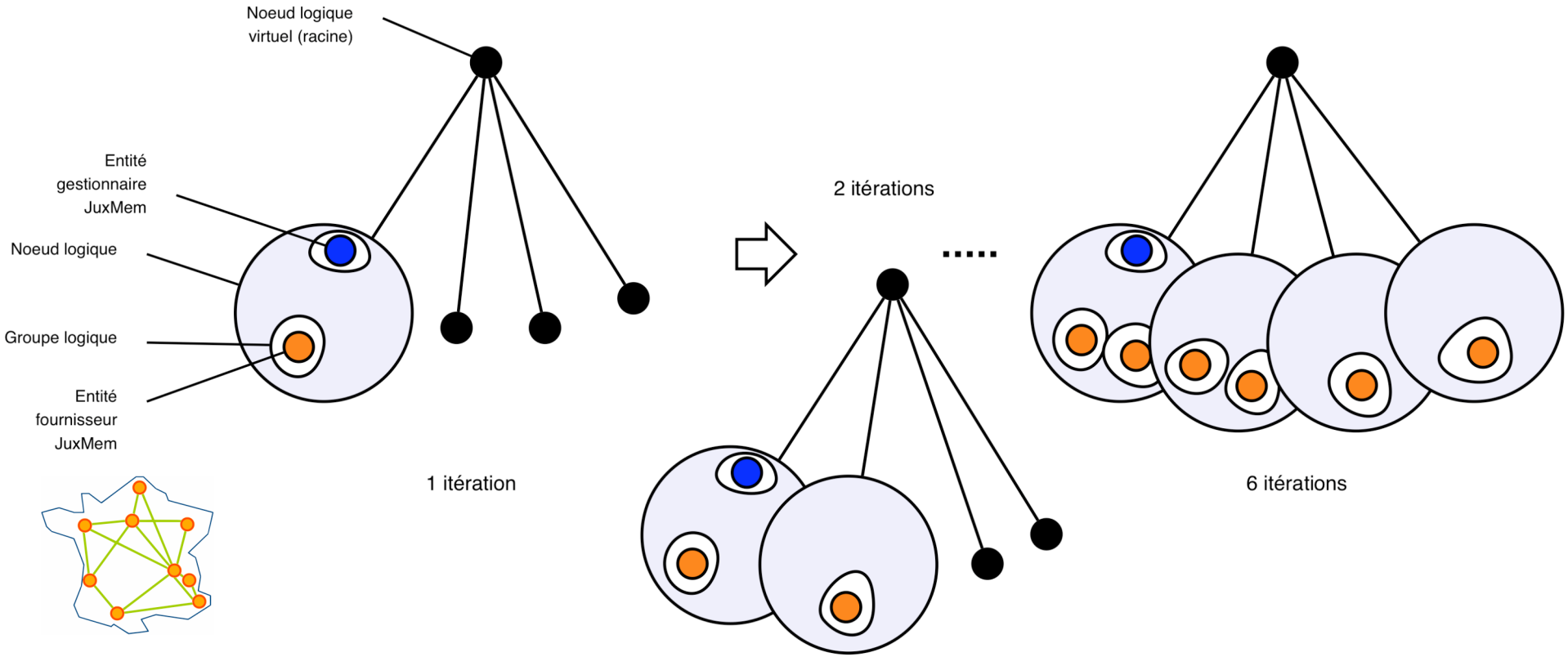


Passage à l'échelle : expérimentations multi-site

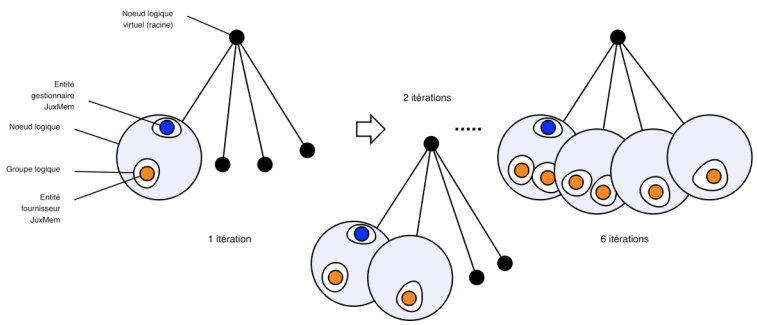


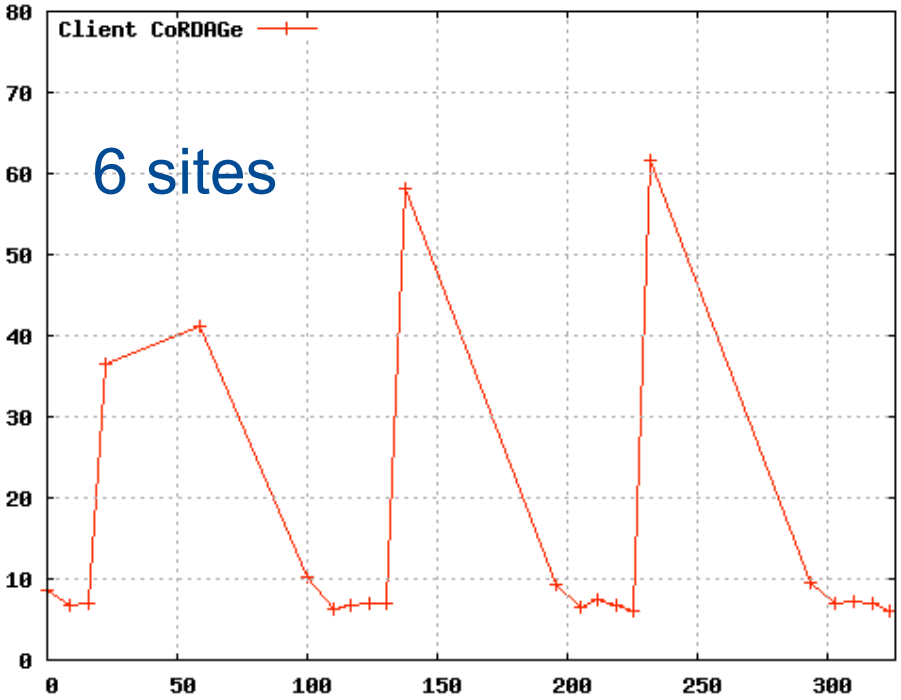
Expérimentation dynamique

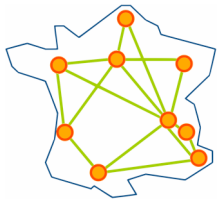
- 1 serveur CoRDAGe, 1 application de type JuxMem, 1 client
- Requêtes de déploiement d'un nouveau fournisseur JuxMem
  déployé à tour de rôle dans 4 puis 6 sites différents choisis par CoRDAGe

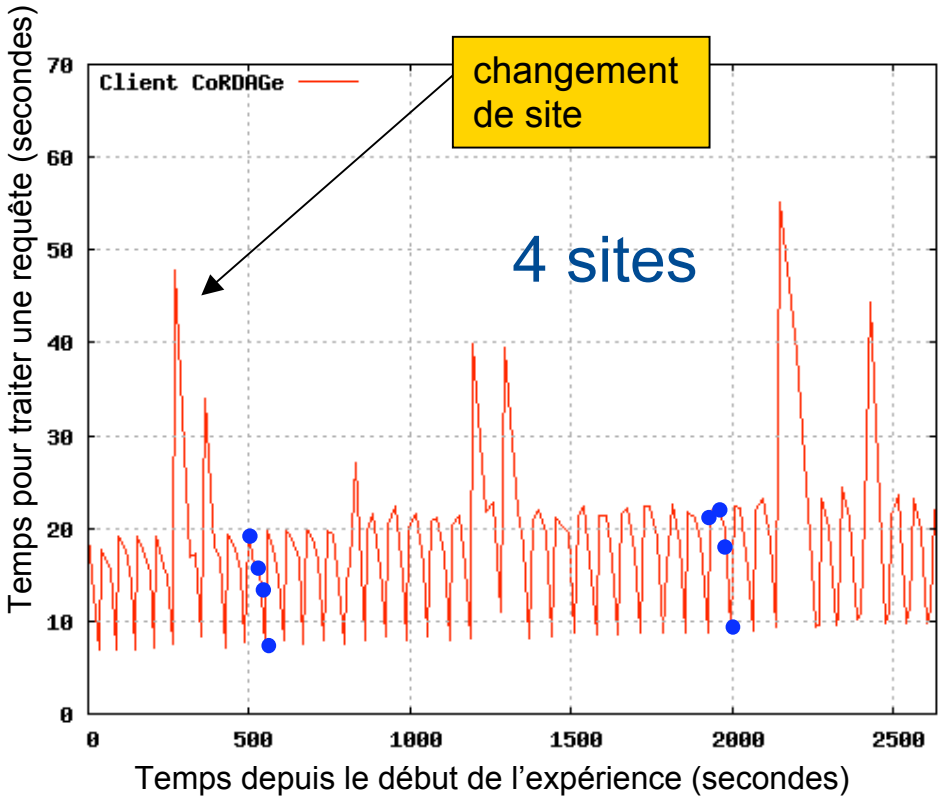


Passage à l'échelle : expérimentations multi-site









CoRDAGe

Conclusion et perspectives

Contributions : le déploiement dynamique

Proposition d'un modèle pour le déploiement dynamique

- Générique
- Satisfait les trois propriétés : transparence, versatilité et non-intrusivité
- Traduction d'actions de haut niveau en opération de déploiement bas niveau
- Pré-planification du déploiement

Proposition d'une architecture (CoRDAGe) pour illustrer ce modèle

- Architecture client-serveur, appel de procédure à distance
- Gestion de plusieurs applications en parallèle

Exploration intensive des outils OAR et ADAGE

- Mise en évidence des limites
- Suggestions d'amélioration
  - OAR : traitement des réservations par lot
  - ADAGE : interface de plus haut niveau

CoRDAGe

Contributions : mise en oeuvre

- Prototype fonctionnel
  - Plus de 7000 lignes de code (C, C++, Perl, Shell)
  - Dépôt APP et licence L-GPL 3
  - Projet GForge : http://cordage.gforge.inria.fr

- Projets LEGO et Respire de l'ANR
  - Proposition de brique pour le déploiement dynamique
  - Prise en charge de JuxMem et JXTA (Sun Microsystems)
  - Expérimentations sur Grid'5000

- Projet DISCUSS (AIST/Univ. Tsukuba)
  - Prise en charge de Gfarm
  - Co-déploiement JuxMem + Gfarm
  - Evaluation préliminaire sur la plateforme OmniRPC (Japon)

CoRDAGe

Perspectives

Vers un CoRDAGe distribué

- Faire communiquer les différents serveurs entre eux
- Tolérance aux défaillances
- Echange d'informations sur les déploiements des différents utilisateurs

Intégration dans un canevas pour applications autonomes

- 4 phases : surveillance, analyse, planification et exécution
- CoRDAGe traite des actions de haut niveau, spécifiques à l'application
- Exemple de canevas : Dynaco (IRISA, équipe PARIS)

Approche par programmation par contraintes

- Concerne principalement les opérations sur les arbres : projection et fusion
- Rechercher ou tendre vers une solution optimale
- Etablir un compromis avec la performance de traitement des requêtes

CoRDAGe

CoRDAGe

Co-déploiement et redéploiement d'applications sur grille

Loïc Cudennec
Projet PARIS, IRISA, INRIA Rennes
Université de Rennes 1
Avec le soutien de la Région Bretagne et de Sun Microsystems

15 janvier 2009

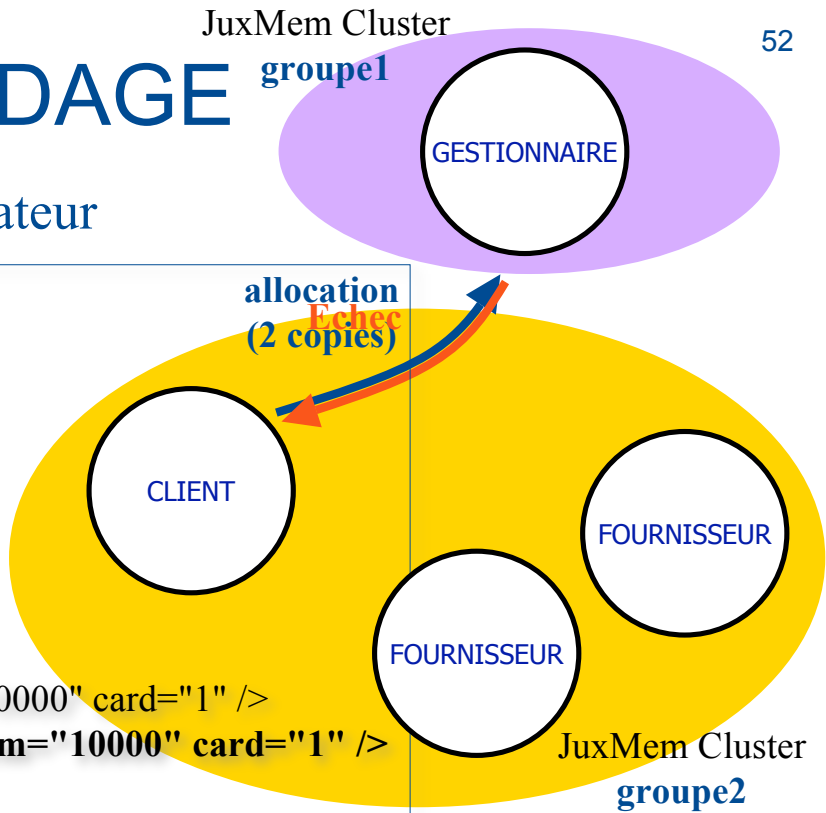
Directeurs de thèse : Luc Bougé et Gabriel Antoniu

Région Bretagne · Sun microsystems · INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE · INRIA · centre de recherche RENNES - BRETAGNE ATLANTIQUE · IRISA · Institut de Recherche en Informatique et Systèmes Aléatoires · UNIVERSITÉ DE RENNES 1 · ENS CACHAN BRETAGNE

Redéployer JuxMem avec ADAGE

Description de l'application modifiée par l'utilisateur

```
<JUXMEM_application>

 <juxmem_cluster id="groupe1">
  (...)
 </juxmem_cluster>

 <juxmem_cluster id="groupe2" rdv="groupe1">
  <managers/>
  <providers>
   <provider id="fournisseur" port="9876" uptime="60" mem="10000" card="1" />
   <provider id="fournisseur2" port="9876" uptime="60" mem="10000" card="1" />
  </providers>
  <clients>
   <client id="client" port="9871" bin="juxmem-writer"
           args="@managerHostname 4" card="1" />
  </clients>
 </juxmem_cluster>
</JUXMEM_application>
```



Approche compliquée pour un utilisateur

- Beaucoup de potions propres à ADAGE
- De nombreuses étapes pour déployer, complexifiées par :
  - La dynamicité
  - L'échelle du déploiement

Déploiement dynamique

Modèle : utilisation

CoRDAGe

- Déploiement simple (déploiement initial)
- Déploiement dynamique
  - Expansion
  - Rétraction
- Co-déploiement : déploiement coordonné

Interface client

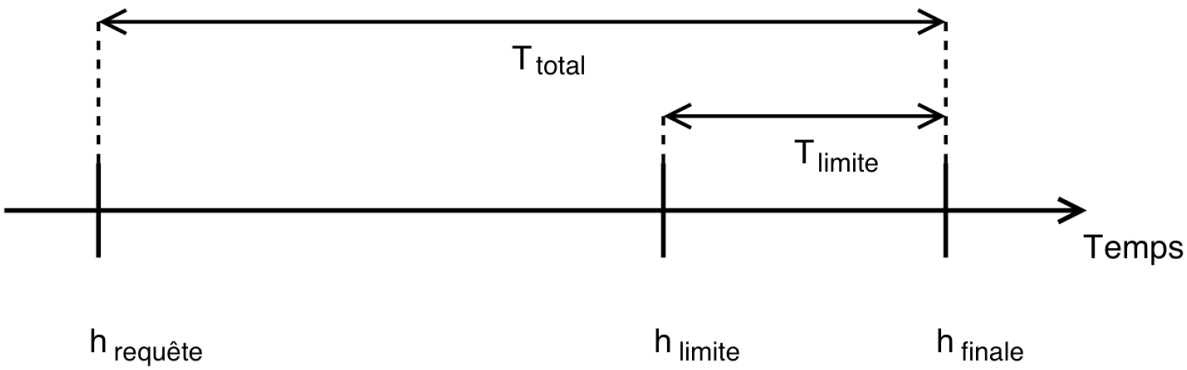
CoRDAGe

Format des requêtes entre les applications et le serveur

- Basées sur le principe de l'appel de procédure à distance (RPC)
- Contiennent :
  - Un identifiant de fiche de suivi
  - Un numéro d'action
  - Une liste de paramètres
- Deux classes d'action :
  - Générique : actions commune à toutes les applications
    par exemple : *déclarer*, *configurer*, *déployer*, *surveiller*
  - Spécifique : actions utilisées dans le contexte d'un type d'application

Commande du déploiement

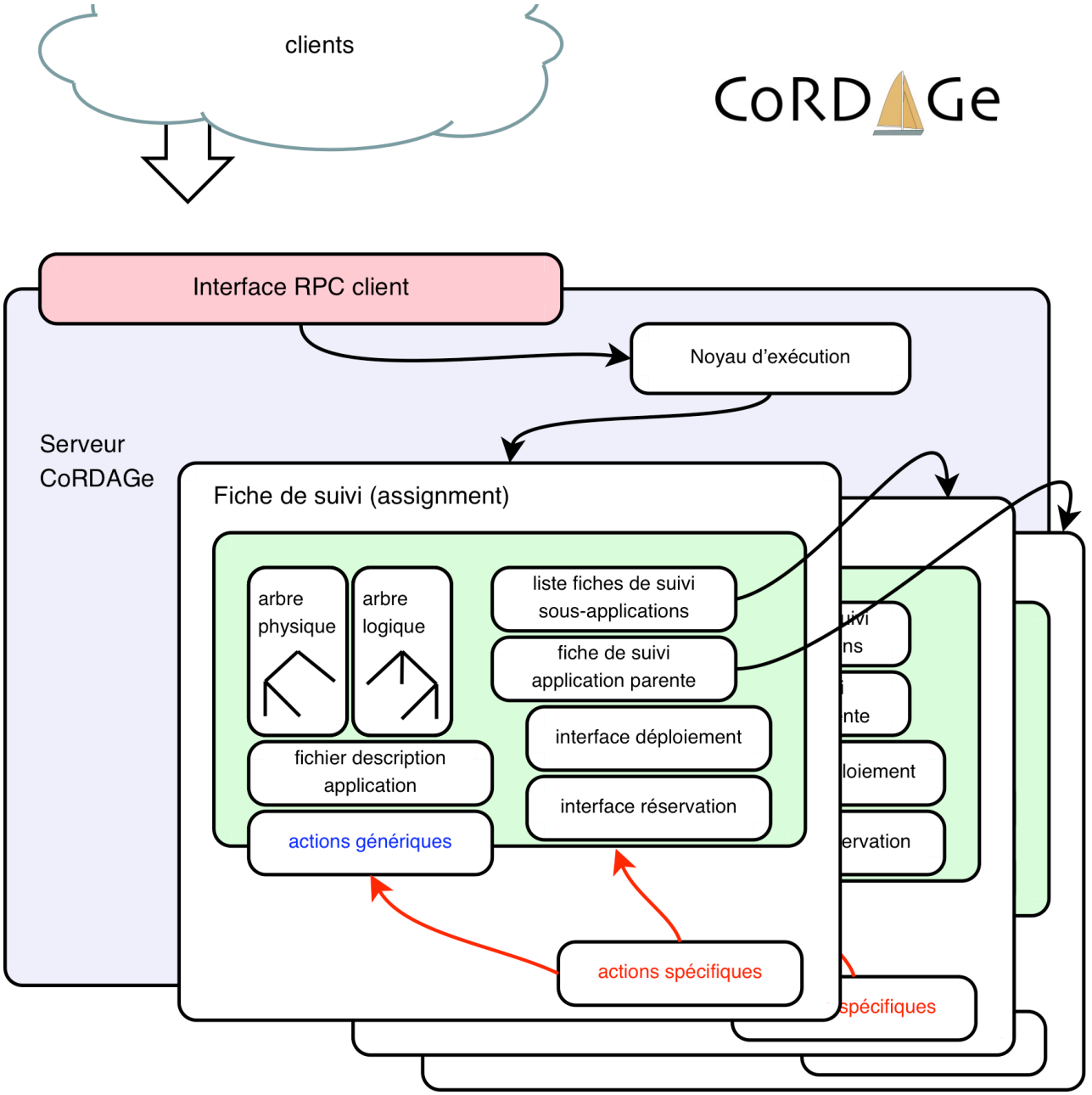
- Notion de temps total (walltime)
- Toutes les réservations se terminent à l'heure finale



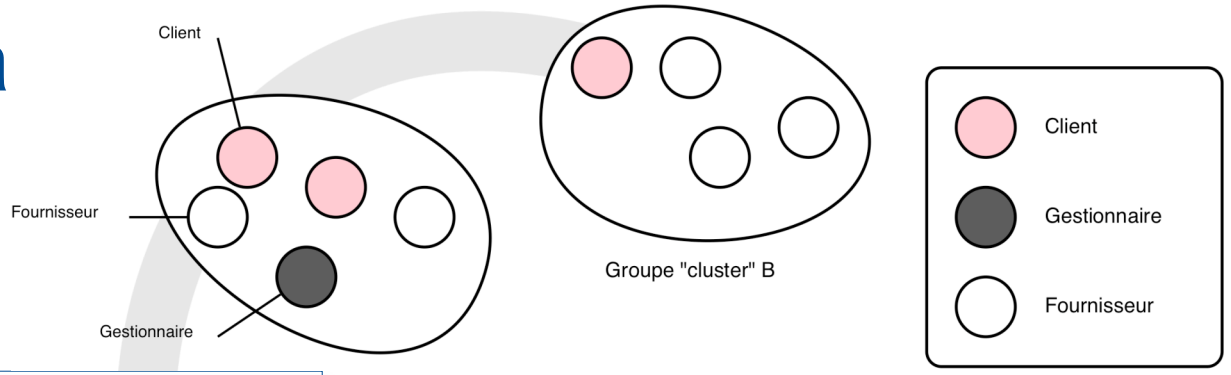
Architecture du serveur

Fiche de suivi de base contient les actions génériques

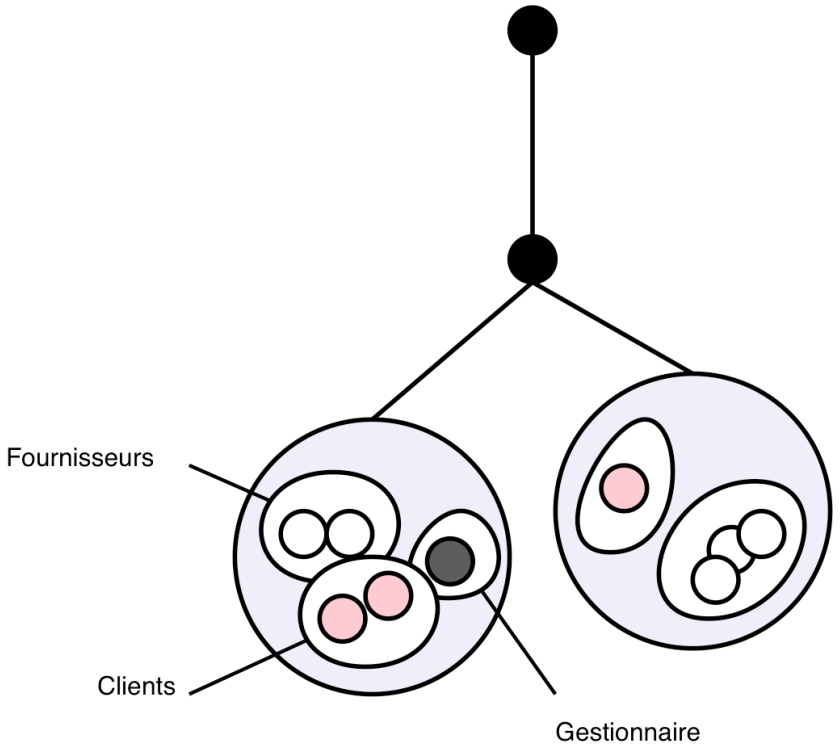
Par application, une fiche de suivi spécialisée est dérivée de la fiche de base



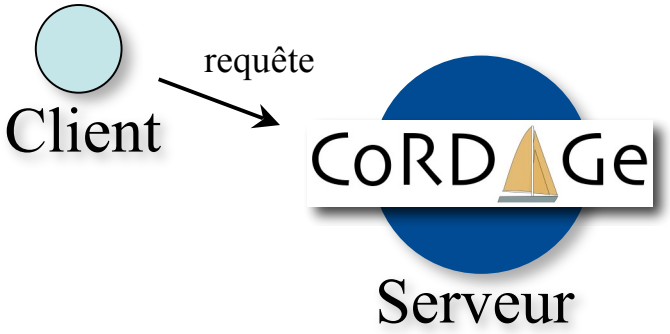
Construction de la représentation logique




```
logtree->add_node(virtuel);

Pourchaque(GroupeCluster C)
{
 logtree->add_node(C);

 Pourchaque(Rôle R présent dans C) {
  // R ∈{gestionnaire, fournisseur, client}
  logtree->add_group(R);

  Pourchaque(Entité E ∈ R) {
   logtree->add_entity(E);
  }
 }

 logtree->move_up();
}
```

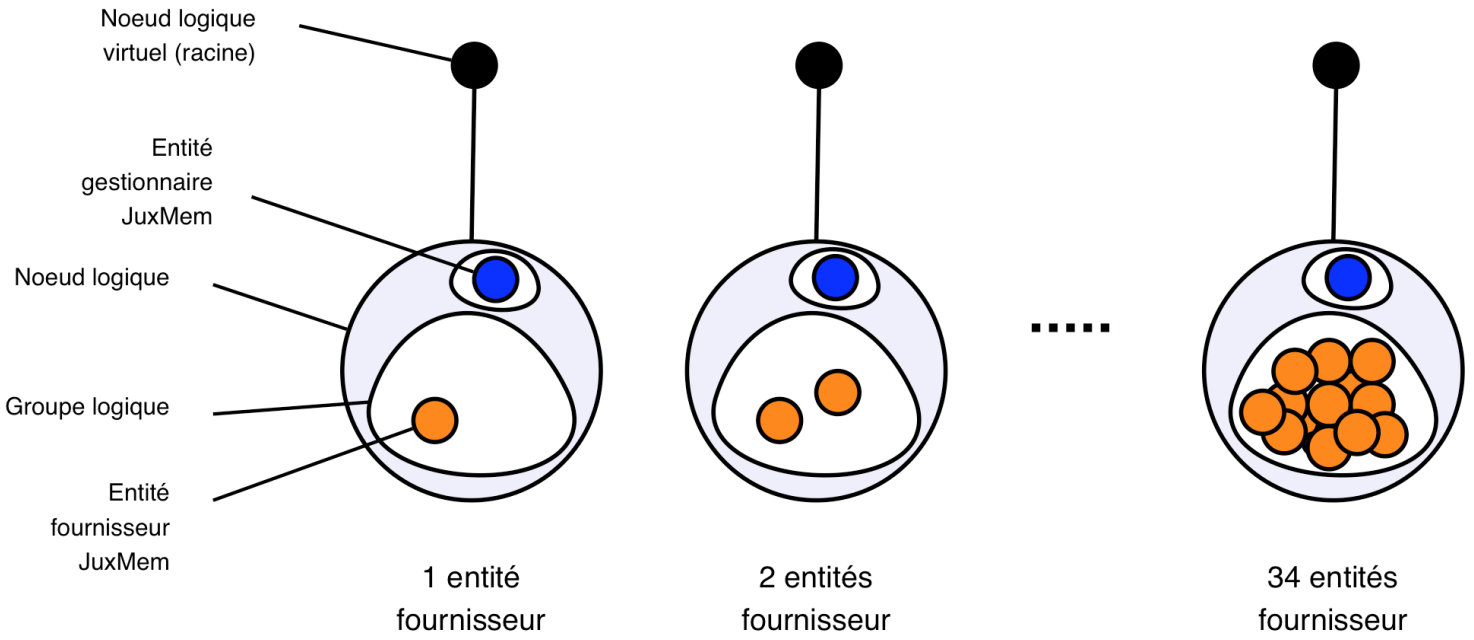


Impact de la taille du groupe logique sur le coût du déploiement



Expérimentation statique

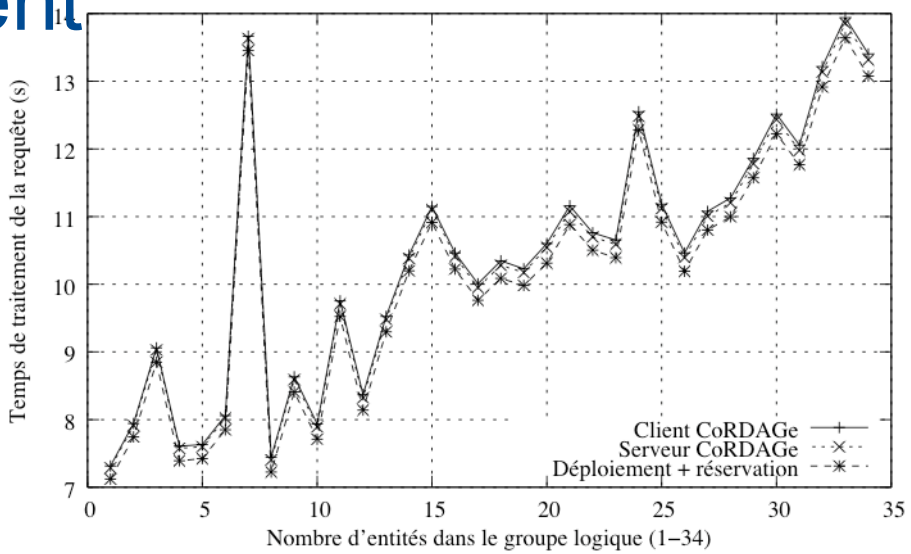
- 1 déploiement par expérience
- Variation du nombre d'entités dans un groupe logique de 1 à 34

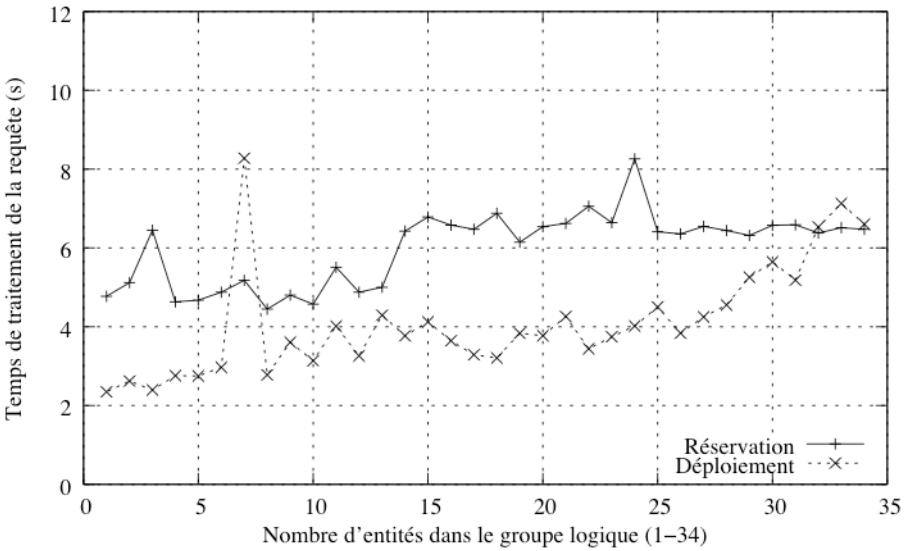


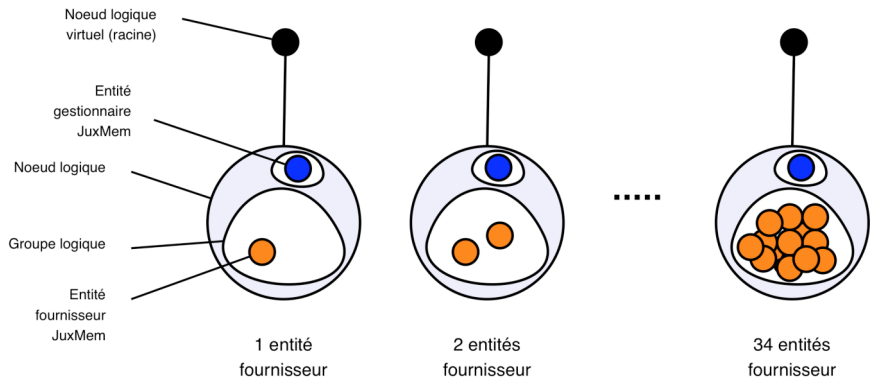
Impact de la taille du groupe logique sur le coût du déploiement

Résultats

- Temps total : augmentation de 7 à 14 secondes
- L'augmentation du temps de traitement est liée à l'outil de déploiement
- Faible surcoût lié à CoRDAGe : moins de 1,5% du temps total







Contributions annexes

Extension d'un protocole de cohérence des données
- Adaptation pour la visualisation de données

Caractérisation du comportement d'un système pair-à-pair
- Etude de JXTA à large échelle sur Grid'5000 (Sun Microsystems)

Une base de données distribuée sur un service de partage de données
- Offrir une interface orientée BD sur JuxMem (Respire)

Un service de partage de données hiérarchique
- Couplage de JuxMem et Gfarm (DISCUSS, AIST/Univ Tsukuba)