

Configuration de middleware dirigée par les applications

Vivien QUEMA, Luc BELLISSARD

Projet SARDES INRIA Rhône-Alpes – LSR-IMAG
ScalAgent Distributed Technologies

17 Octobre 2002



Application distribuée à composants

- Entités fonctionnelles ou *composants* répartis sur les différents sites (aux ressources variées)
- Une plate-forme locale sur chaque site (adaptée aux ressources locales)
- Un medium de communication entre les différentes entités fonctionnelles

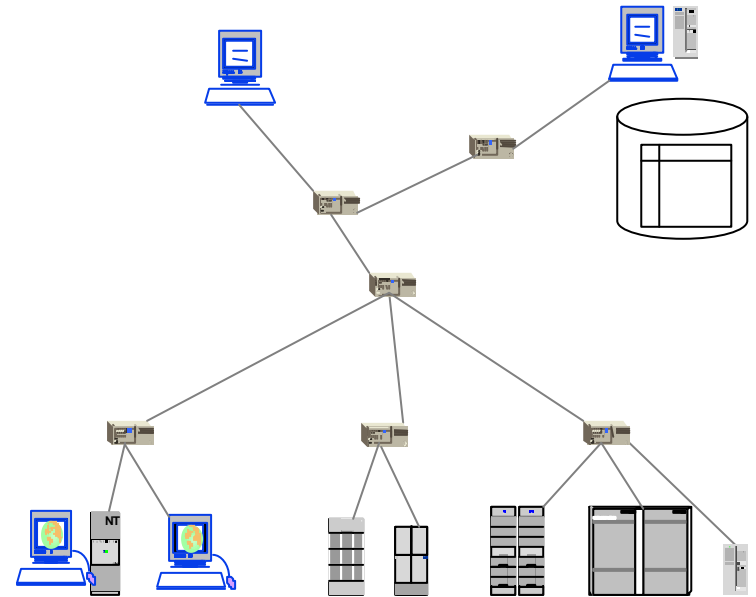
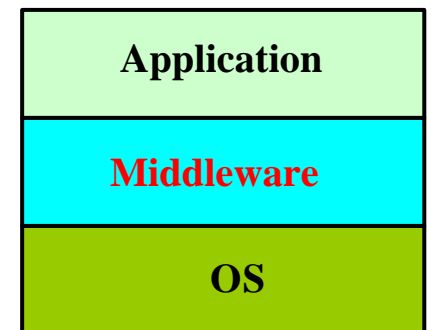
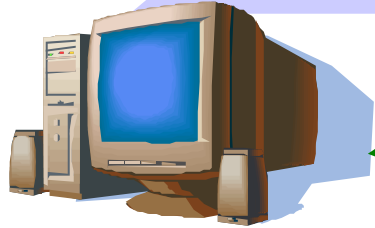


Plate-forme locale + medium = *middleware*



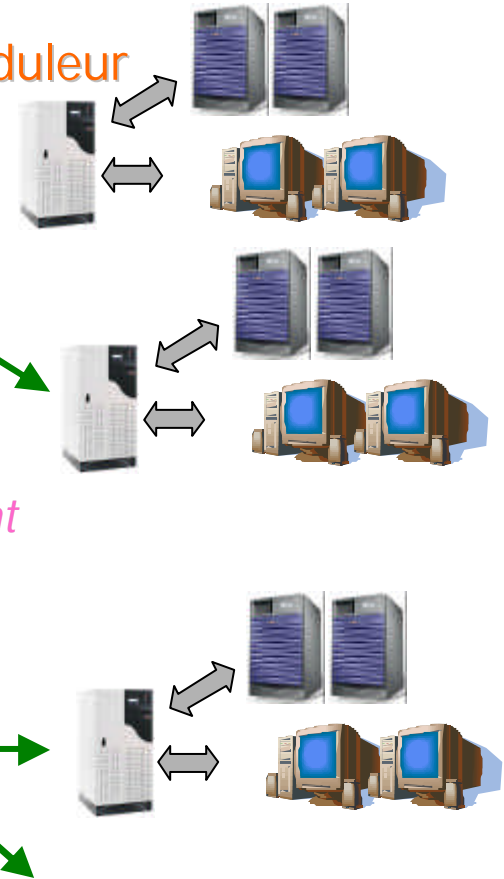
Exemple

Site de maintenance



Technicien

Onduleur



Centre local

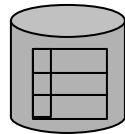
Centre régional

ordonnancement

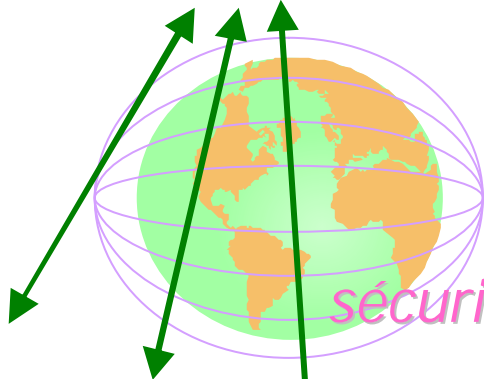


persistance

BD



sécurité



autres sites

Centre national

Les problématiques du développement d'une application répartie (1)

- Nombreuses tâches à effectuer
 - Développement des différentes entités fonctionnelles
 - Gestion des communications
 - Prise en charge des propriétés non fonctionnelles : persistance, sécurité, transaction, ...

- Nécessité de réduire les coût de développement: développer des composants applicatifs génériques indépendamment de la plate-forme d'exécution
 - Nécessité d'une couche middleware (généricité) adaptée aux ressources locales et qui prennent en charge les propriétés non fonctionnelles

Problématiques (2)

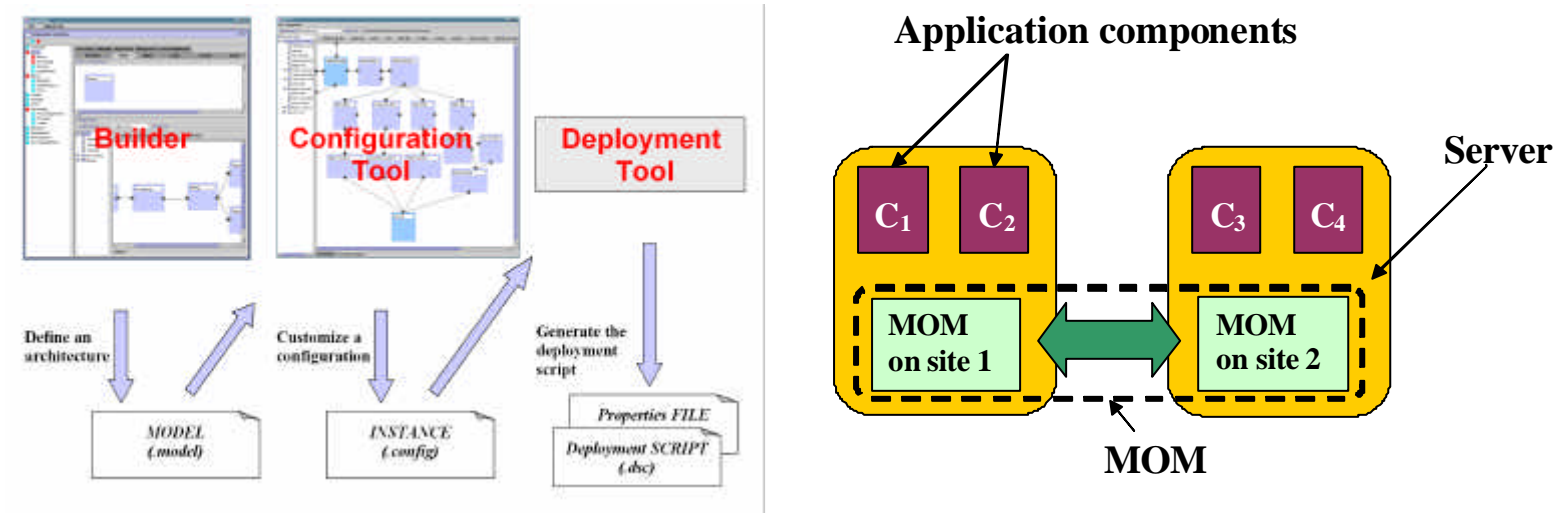
- Les propriétés non fonctionnelles sont assurées par le middleware
- Mais ...
 - Il n'est pas **possible** ni **désirable** que le middleware fournissent systématiquement toutes les propriétés non fonctionnelles à tous les composants
 - ➔ Nécessité d'avoir un middleware **configurable** ... et de le **configurer** (automatiquement) !
 - Pour qu'il soit adapté à la plate-forme d'exécution
 - Pour qu'il prenne en charge les propriétés non fonctionnelles nécessaires: ordonnancement des messages, sécurité, ...

Travaux connexes (1)

➤ Configuration des middlewares

- Nombreuses techniques pour la prise en charge de propriétés non fonctionnelles
 - Utilisation de la réflexivité (openORB, dynamicTAO, ...)
 - Composition de composants middlewares (Aster, Cactus, ...)
 - Utilisation de la programmation par aspects (Lasagne, ...)
- Le processus de configuration est souvent ad hoc ou complexe (Aster)

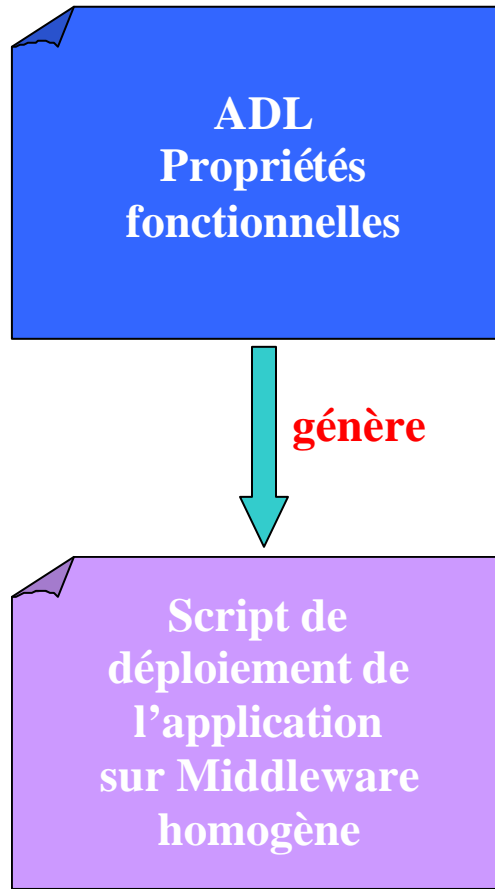
Contexte de travail



➤ La plate-forme ScalAgent

- Outils de description (ADL), configuration et déploiement d'applications à base de composants
- Plate-forme d'exécution : Middleware orienté message (MOM) constitué par un ensemble de serveurs hébergeant les composants applicatifs

Fonctionnement de la plate-forme

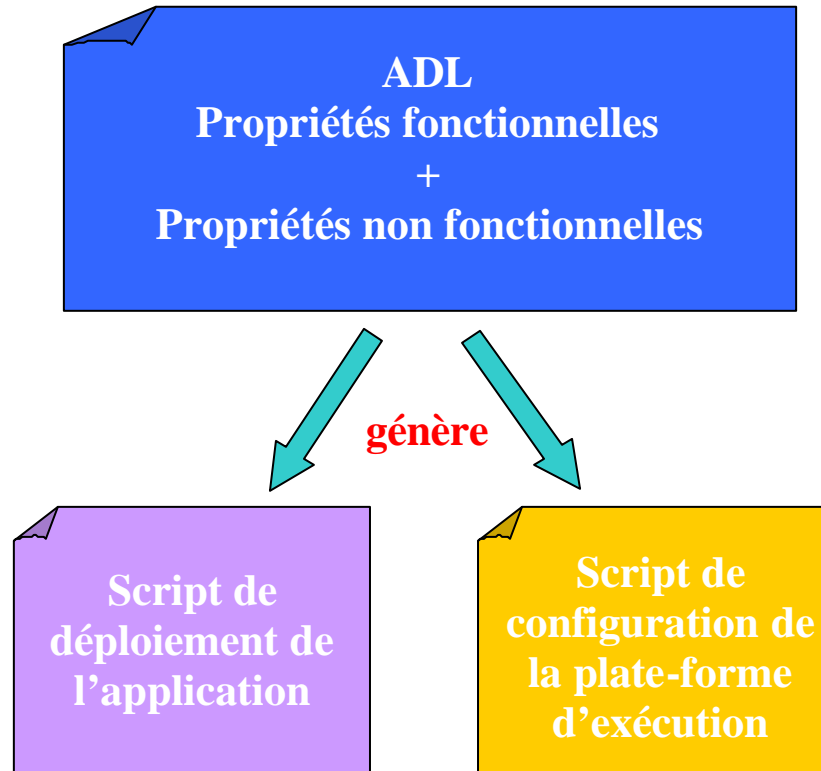


Travaux connexes (2)

➤ Langages de description d'architectures

- 2 catégories d'ADL
 - Ceux qui permettent de générer un exécutable à partir de la description → prennent peu ou pas en charge les propriétés non fonctionnelles (Olan, Unicon, ...)
 - Ceux qui ont un pouvoir de spécification plus puissant et qui effectuent des analyses sur le comportement dynamique de l'exécution du système → ne permettent pas de générer un exécutable (Rapide, Wright, ...)

Proposition



Objectifs

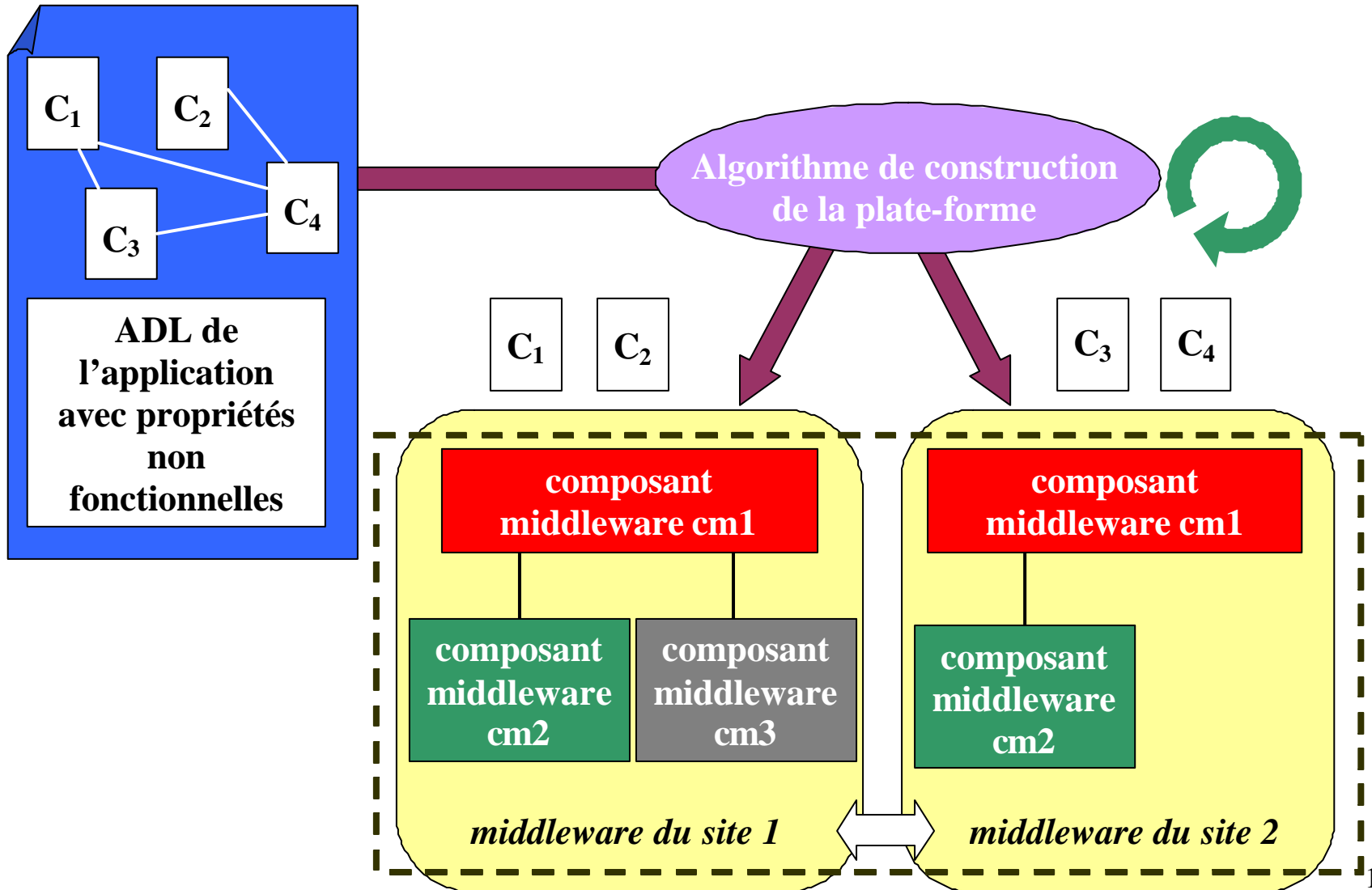
➤ Plate-forme ScalAgent

- Middleware homogène : configuration quasi-identique des serveurs hébergeant les composants applicatifs

➤ Contribution

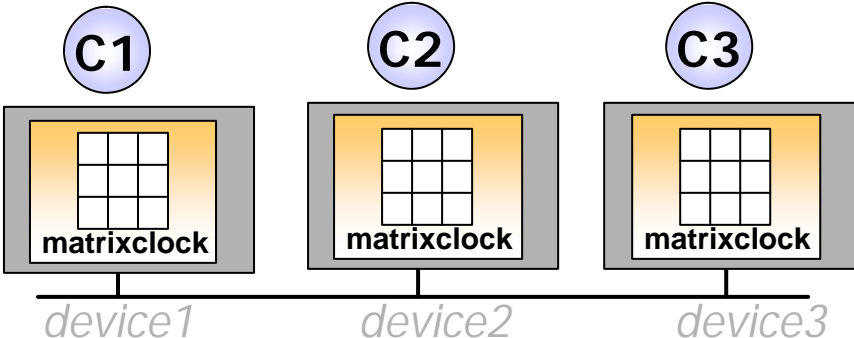
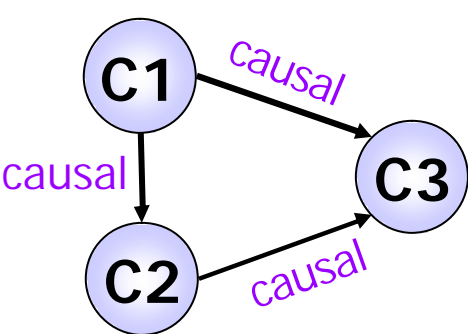
- Permettre de spécifier les propriétés non fonctionnelles requises par une application dans un ADL
- Savoir configurer la plate-forme d'exécution en fonction de la spécification des besoins applicatifs
 - Chaque propriété non fonctionnelle est prise en charge par un ou plusieurs composant(s) middleware
 - On détermine les *composants middlewares* nécessaires sur chaque serveur de la plate-forme d'exécution
 - On détermine la politique de gestion de la propriété pour chaque composant middleware de façon à améliorer les performances
- ➔ **Génération automatique d'un middleware hétérogène**
- Buts
 - Aider le développeur de l'application
 - Employer un middleware optimisé pour l'application → améliorer les performances de l'application

Middleware hétérogène

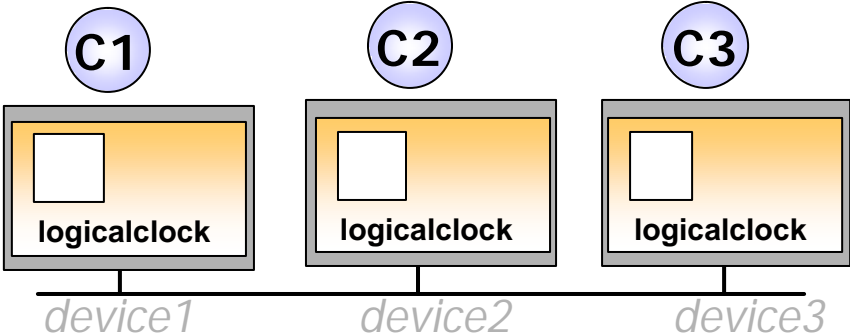


Nécessité de la configuration dirigée par les applications

- Exemple d'une propriété : ordonnancement causal
 - Méthode classique : horloge matricielle
 - Augmentation quadratique de la taille de l'horloge

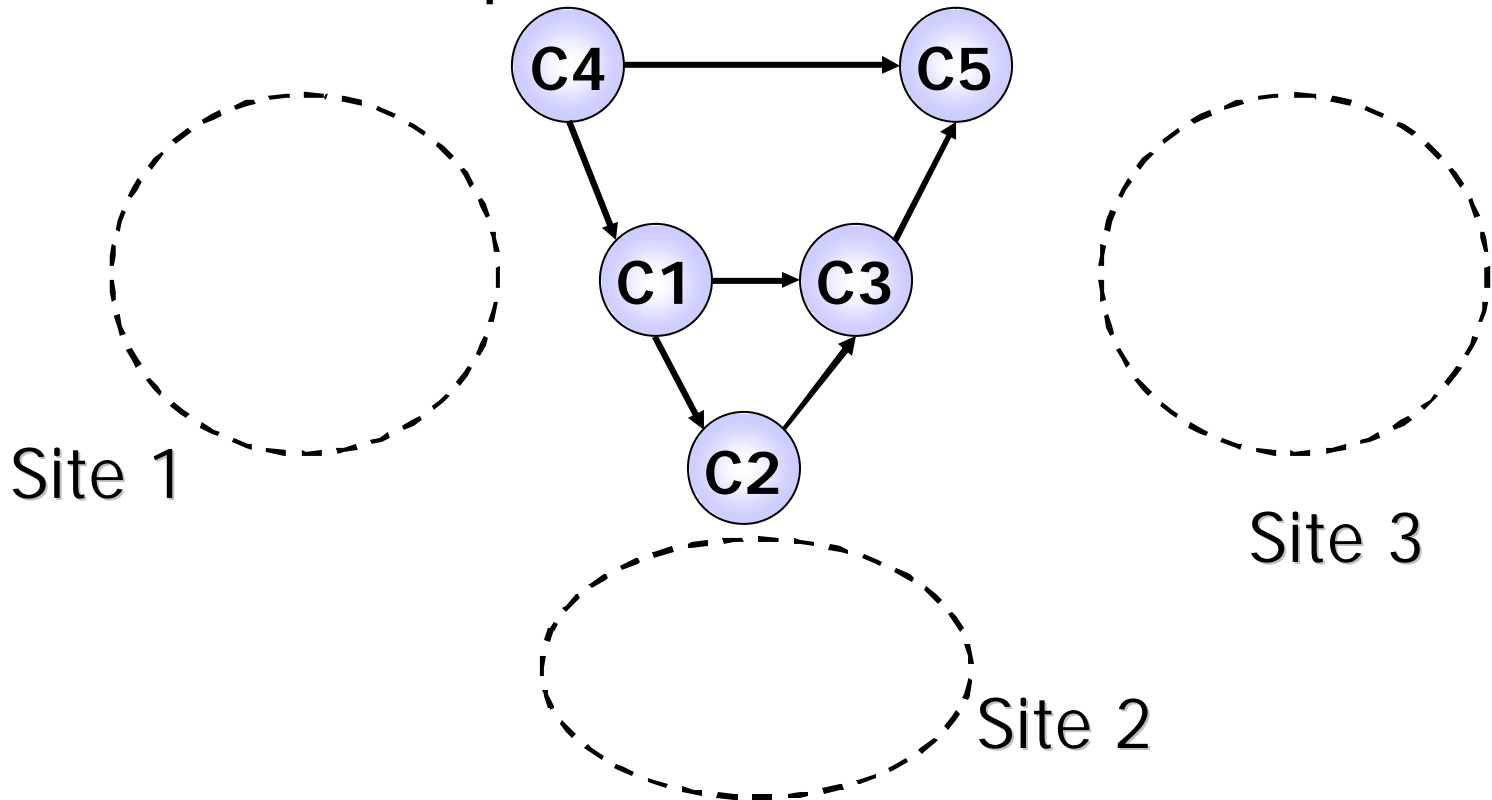


- Configuration dirigée par l'application



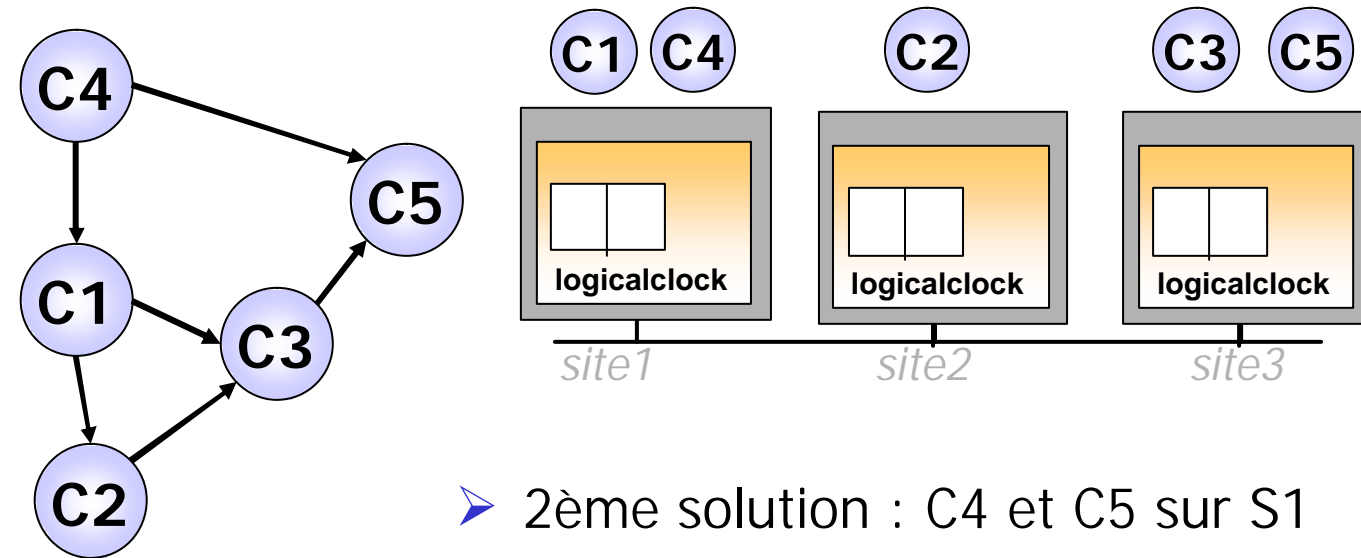
Exemple de configuration dirigée par les applications

- Exemple d'une propriété : ordonnancement causal
 - 3 sites (S1 S2 et S3), 5 composants (C1, C2, ... C5)
 - C1 sur S1, C2 sur S2 et C3 sur S3
 - C4, C5 : **placement libre... à déterminer !**



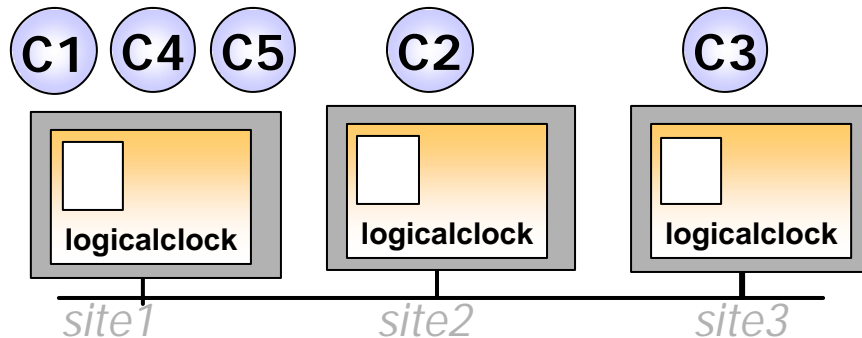
Exemple de configuration dirigée par les applications

- 1ère solution : C4 sur S1 et C5 sur S3



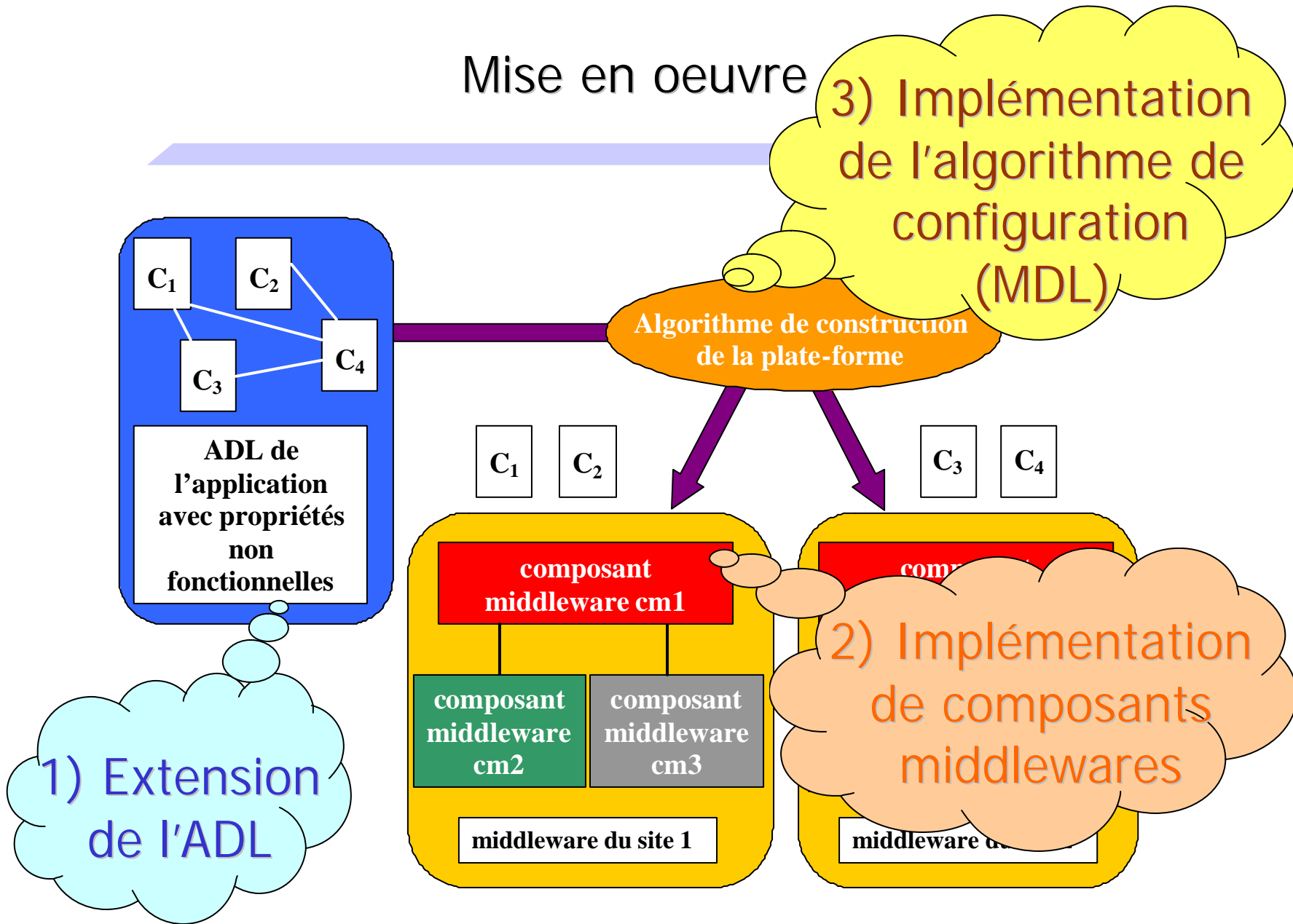
Taille par
Site = 2

- 2ème solution : C4 et C5 sur S1



Taille par
Site = 1

Mise en oeuvre



Mise en œuvre (1) : Extension de l'ADL

➤ Modification de la DTD du langage XOlan

➤ Spécification de propriétés de composants

```
<component name=« C2 »>
```

```
  <set property=« persistency » />
```

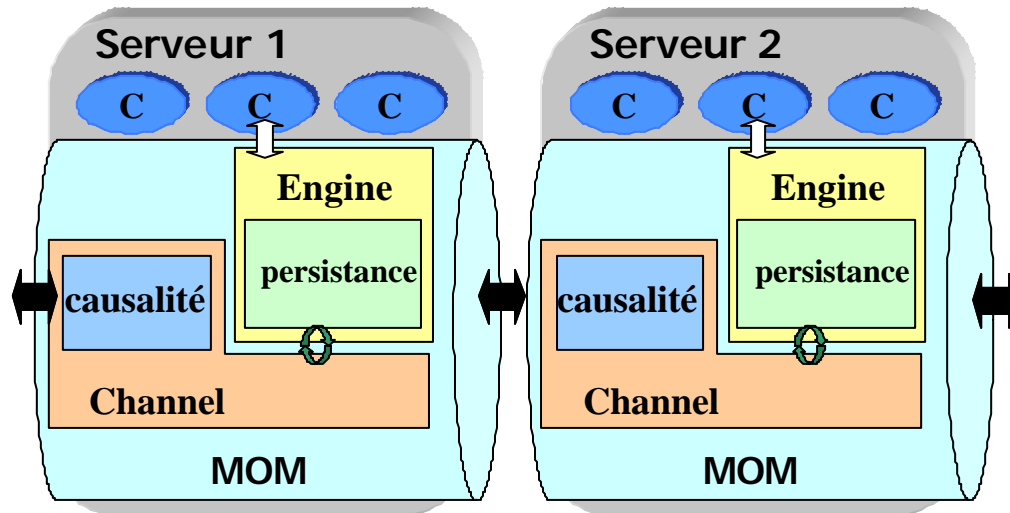
```
</component>
```

➤ Spécification de propriétés de connecteurs

```
<connect compOut=« C1 » compIn=« C2 » causality=« true »>
```

Mise en œuvre (2) : Implémentation de composants middleware

➤ Implémentation de composants middlewares



Mise en œuvre (3) : Implémentation de l'algorithme de configuration

- L'algorithme détermine la configuration du middleware à partir de l'ADL
- Cette configuration est décrite dans un fichier XML: langage de description du middleware (MDL)
- ```
<serverConfiguration sid=« 1 » >
 <persistencyConfiguration management=« common » />
 <securityConfiguration management=« independent »>
 <componentLink from=« C1 » to=« C2 » />
 <componentLink from=« C3 » to=« C5 » />
 </securityConfiguration>
 ...
</serverConfiguration>
```

# Tests de performances (1)

---

- Le but de la configuration du middleware est double
  - Aider le développeur
  - Générer un middleware optimal : performances
- Tests de performance sur grappe de PC pour plusieurs types d'applications



# Conclusions

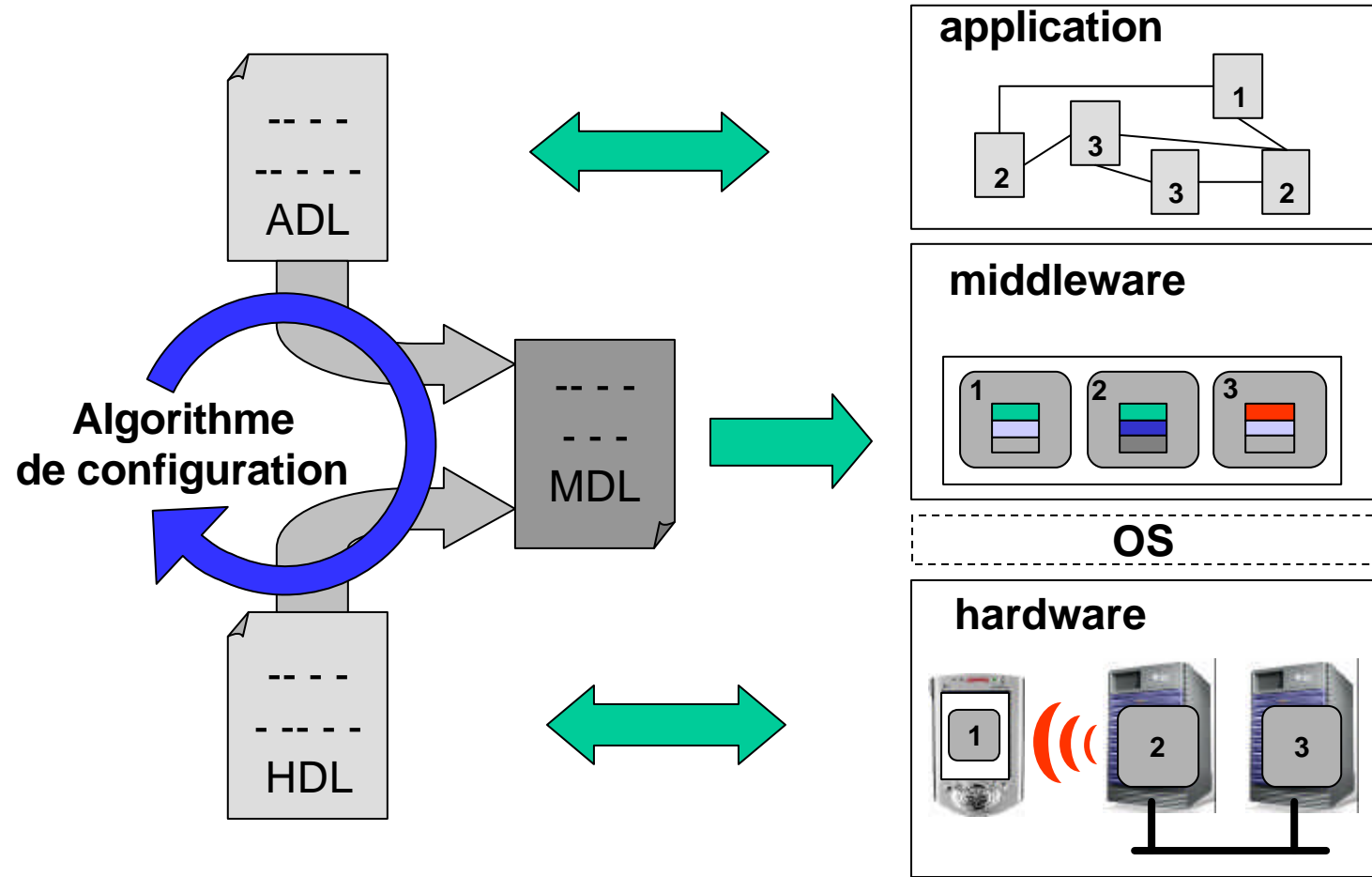
---

- Méthode de configuration des middlewares dirigée par les applications
  - Facilite la conception d'applications réparties
  - Améliore les performances sensiblement pour certaines propriétés
    - Permet d'envisager la construction d'applications à plus grande échelle
  - Démarche générique : création d'un fichier de description de middleware (MDL)

# Perspectives

- Approfondissement de l'étude sur les paramètres de calcul de coût
  - Empirique ou formel ?
  - Quels paramètres évaluer ?
- Faisabilité d'une configuration
  - Etudier les critères de compatibilité entre deux propriétés non fonctionnelles (intervention du paramètre temps)
- Etude de la reconfiguration
  - Porter le MOM dans une technologie à composants réflexive (FCF) pour le rendre configurable dynamiquement
- Extension des possibilités de spécification de l'ADL
  - Utiliser des formalismes (Wright) pour spécifier le comportement interne d'un composant
  - Spécifier les caractéristiques physiques du système

# Spécifications des caractéristiques physiques du système



# Questions ?

---

Quelques liens...

- <http://www.inrialpes.fr>
- <http://www.scalagent.com>
- <http://sardes.inrialpes.fr/~quema/>