GRIDKIT: Pluggable Overlay Networks for Grid Computing

Paul Grace, Geoff Coulson, Gordon Blair, Laurent Mathy, Wai Kit Yeung,
Wei Cai, David Duce, Chris Cooper

Computing Department, Lascaster University

Department of Computing, Oxford Brookes University

Sérgio RODRIGUES DE MORAIS

Problematique

• Les applications plus avancées de la Grille ont des demandes importantes que ne sont plus adressées par les platforms de services web actuellement;

• Par exemple, les platforms courantes ne supportent pas la riche diversité de types d'interaction pour la communication qui sont demandés par les applications avancées (publish-subscribe, media streaming, peer-to-peer interaction);

• En cet article sera décrit le « Gridkit middleware » qui augmente la architecture *service-oriented* basique pour adresser cette déficience particulière.

• Le principal cible est le support d'une infrastructure de communication requis pour supporter les multiples types d'interaction dans une façon unifié et extensible. Ce qui sera présenté comme un nouveau concept de *pluggable overlay networks.*

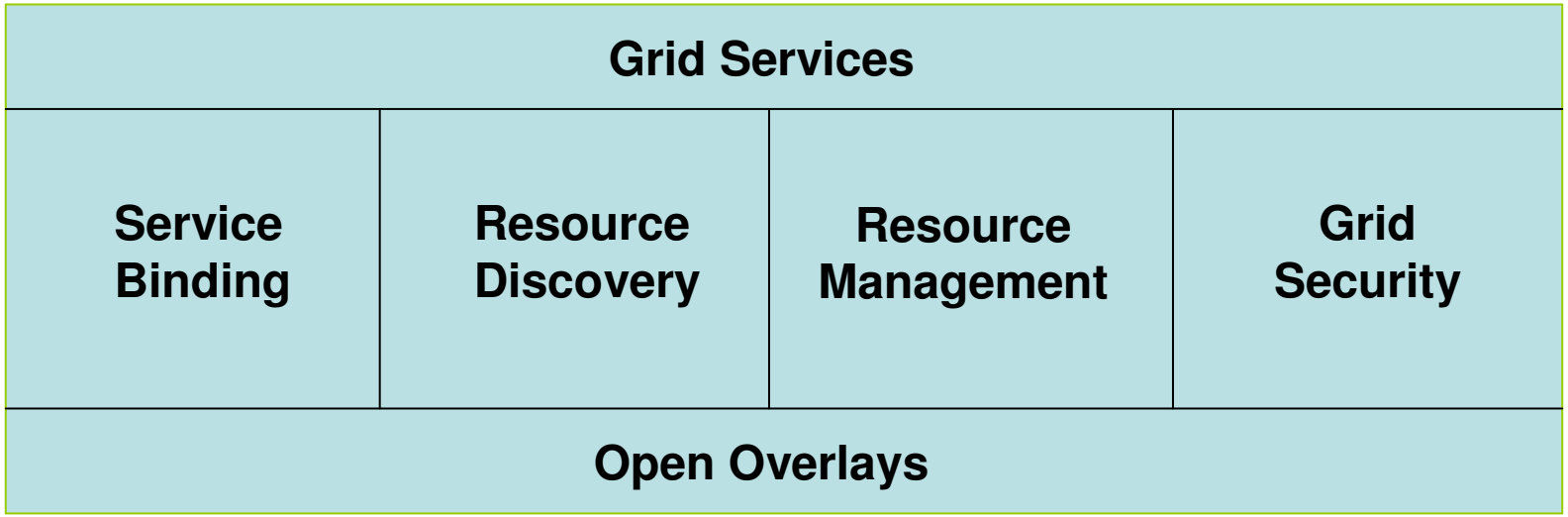
Plan

Introduction

L'état d'art de la Grille-middleware est encore inadéquat pour supporter les applications avancées qui demandent des caractéristiques comme: end-systems et infrastructures de network hétérogènes, large échelle, haute complexité, collaboration interactive en temps réel, multiples media-types, sensibilité à la QoS, et le besoin de reconfiguration dynamique en réponse aux changements de l'environement.

Le but dans cet article est de fournir extensibilité en ajoutant des nouvelles fonctionnalités (types d'interaction et éléments d'infrastructure de communication) comme *plug-in components* qui sont définis dans de *component frameworks*. Le GridKit est entièrement composé de tels *component frameworks*. Chaque couche d'infrastructure de communication est structurée comme un *component framework* où les *overlay networks* sont les unités de *pluggable functionality.*

Une vision général sur GRIDKIT



La vision de GRIDKIT est de fournir un middleware de support en chacun de ces quatre branches, identifiés comme clés pour le support des applications de la Grille.

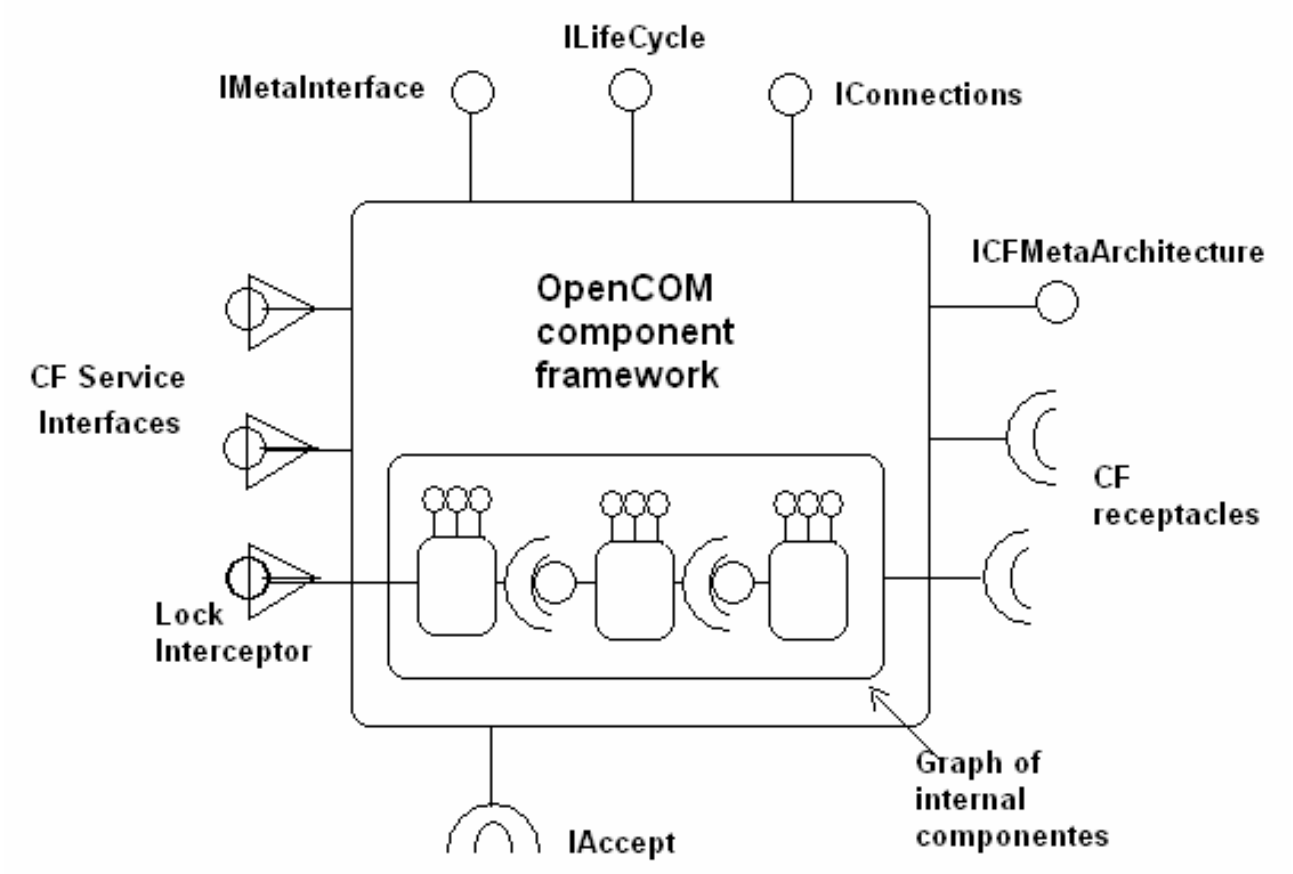
Service binding. Contient les types d'interaction *pluggable* et aussi fourni des APIs génériques qui permettent au programmateur des applications de créer et utiliser exemples du type d'interaction choisi.

Resource discovery. Fourni services de découverte et sur tout des services de découverte de ressources. Exemples des technologies alternatives sont, SLP ou UPnP pour les plus traditionnels services de découvertes, GRAM pour les découvertes de CPU dans uns contexte de la Grille et les protocoles P2P pour les découvertes de ressource plus généreux.

Resource Management. Comprit tous les deux, « coarse-grained distributed resource management » comme est fourni par services comme GRAM, et « fine-grained local resource management », qui est exigé pour construire « end-to-end QoS.

Grid Security. Contient des « pluggable services » qui supportent la communication sécurisée entre les participants.

Ces quatre genres de fonctionnalités middlewares sont implémentés in Gridkit comme « component frameworks » indépendants.



Component Framework Model

- Gridkit est une évolution de la ancienne « middleware platform » OpenORB
- Alors il suit la philosophie de construire les systèmes en terme de:
  1. *Composants* (en utilisant le modèle de composant OpenCOM)
  2. *CFs*
  3. *Réflexion*

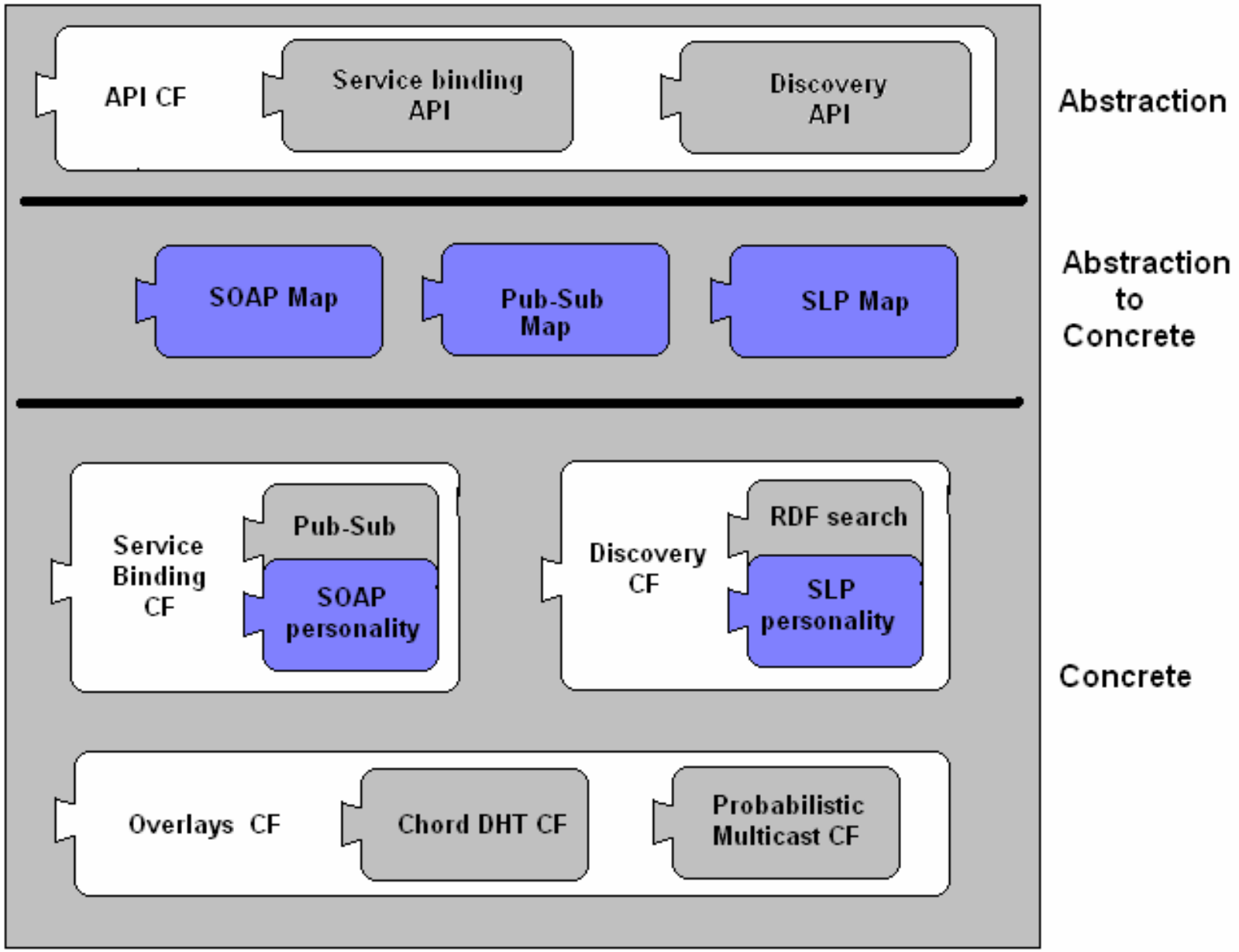
Sur l'architecture générique du Gridkit/OpenCOM CF, les CFs regardent ce qui vient de l'extérieur exactement comme un component OpenCOM normal. Néanmoins chaque « Gridkit CF » crée une interface « ICFMetaArchitecture » qui fait des opérations avaliser et reconfigurer la structure interne des CFs (appelée 'graph' des souscomponents internes.

En plus, pour assurer que les changements dynamique du framework (comme ce qui arrive quand un nouveau « plug-in component » est inclus) seront valider, chaque CF export un réceptacle appelé « IAccept »

De différentes stratégies de validation peuvent être « plugged » sur ce modèle de façon que une fois que un changement est fait sur la structure des CFs, la « plug-in checking strategy » est exécutée.

La topologie des souscomponent interne est vérifiée une deuxième fois par en ensemble de descriptions architecturaux « XML-based » des configurations valides des components.

Alternativement des stratégies plus complexes peuvent être « plugged in ». Les interfaces « IMetaInterface », « ILifeCycle » et « IConnections » sont associées aux standard de meta-modèles réflectives et ne sont pas discutées dans l'article.



The GRIDKIT Architecture

L’architecture de Gridkit est divisée en trois couches:

1. Une couche de « abstract middleware »;
2. Une couche de conversion de abstract-concret;
3. Et une couche de middleware concret.

Chaque une des couche est formée par plusieurs CF. Ca rende l’architecture configurable et extensible. Alors des components qui implémentent de fonctions spécifiques peuvent être « plugged in » où et quand il faut.

La couche d’abstraction consiste d’un « Grid-oriented API component framework » qui est construit en termes d’abstractions de services web. Dedans, le « Service Binding API » est utilisé pour créer et utiliser sortes de « user-to-service bindings » qui peuvent employer plusieurs types d’interactions.

Les interactions des services abstracts sont décrites en utilisant la Langage de Définition de Services Web (WSDL), qui permet aux interfaces de services d'être uniformément spécifiées indépendamment du type de l'interaction.

« Discovery », au niveau de services et ressources, forme partie du API CF. Encore une fois, WSDL est utilisée par abstraire sur le différents modes d'interaction avec service et mécanismes de découverte de ressources.

La couche « abstract to concrete » prend les informations abstracts les fait passer par la couche de middleware abstract et les amène aux interfaces avec la couche plus basse de middleware concret.

Finalement, la couche 'concret' est composée de trois CFs organisés en deux couches. La couche supérieure est formée par CFs qui supportent « concrete service binding » et « resource discovery ». Puis, le « Discovery CF » permet technologies de découvertes multiples (SLP, UDDI, Jini, P2P-based, etc) pour être simultanément « plugged » dans le framework.

Conclusion

• Cet article a décrit le « GRIDKIT », une façon de supporter des applications avancées au niveau de la Grille-Middleware.

• Le GRIDKIT a utilisé une architecture multi niveaux qui a montré avoir considérable puissance et généralité en supportant telles applications. Sa caractéristique d'extensibilité permette que de nouveaux types d'interaction et aussi de nouveaux overlays puissent être développer et « plugged » au middleware, même en « runtime ».

• Pour les travails futures les auteurs veulent encore évaluer leur middleware par la construction de scénarios basés sur les travaux de l'informatique environnemental et aussi explorer le « self-management » des services et applications en le GRIDKIT.


QUESTIONS
?