Meta-grid
Un premier pas vers un framework grid basé sur les agents

Badr Jabari[*,**], Vincent Englebert[*], El Mostafa Oualim[**], Jean-Pol Vigneron[***]

[*] Precise: Pôle Ingénierie des Systèmes d'Information
Faculté d'Informatique, Université de Namur
Rue Grandgagnage 21, B-5000 Namur, Belgique
{bja, ven}@info.fundp.ac.be
[**] Laboratoire d'Optique Appliquée & Transfert d'Energie
Faculté des Sciences et Techniques, Université Hassan 1er
B.P. 461, Settat, Maroc
oualim.elmostafa@menara.ma
[***] Laboratoire de physique du solide
Faculté des Sciences, Université de Namur
Rue de Bruxelles 61, B-5000 Namur, Belgique
jean-pol.vigneron@fundp.ac.be

Résumé. Vu l'importance et l'ampleur accordée au développement des grilles de calcul (grid), il est nécessaire d'avoir une abstraction adéquate afin de les modéliser. Dans ce but, cet article proposera une plate-forme propice au bon fonctionnement d'un futur framework orienté agents développé au dessus de grids existants. Ce meta-grid, qui est une sorte de grid abstrait représentant et incarnant des grids "idéaux" existants, prendra la forme d'une couche qui sera développée pour faire face aux problèmes dus aux différences existantes entre les grids, et aux détails techniques spécifiques à chacun d'entre eux. L'idée consiste à cacher ces détails derrière un modèle uniforme que nous allons spécifier et modéliser. La création d'une telle plate-forme exemptera le programmeur d'applications grid de se soucier des problèmes spécifiques aux environnements.

1 Introduction

L'intelligence, le raisonnement et la simulation du comportement humain sont parmi les notions qui ont fait beaucoup parler d'elles et qui ont suscité énormément de recherches. Elles ont apporté une grande amélioration et un développement remarquable du monde des technologies de l'information et de la communication (TIC). Les agents ainsi que les systèmes multi-agents (MAS) représentent une des méthodes les plus efficaces et adéquates pour mettre en oeuvre et implémenter ces notions. L'idée en soi est profitable et très intéressante, d'autant plus si nous l'appliquons et l'assemblons avec d'autres techniques. Une telle intégration a déjà donné ses fruits dans le domaine des grilles de calcul. Mais avant d'aller plus loin, et pour bien comprendre comment les grids sont arrivés à ce stade d'évolution, nous allons définir ce concept et tenter de survoler son histoire.

Les *grilles de calcul* (grid)[1] (Foster et al., 2001a) sont définies comme un environnement informatique large et distribué (à l'échelle d'Internet). Elles doivent être distinguées des systèmes distribués conventionnels. En effet, elles se concentrent sur le partage des ressources à grande échelle, les applications innovantes, et sur tout ce qui est orienté haute performance. Autrement dit, c'est "une infrastructure émergente qui relie des grids régionaux et nationaux multiples pour créer une source universelle de puissance de calcul". Le mot grid a été choisi par analogie à la grille d'énergie électrique, qui fournit un accès dominant à la puissance (Foster et Kesselman, 1999). On pense que l'origine du terme est le projet CASA qui a établi une liaison entre des sites contenant des superordinateurs fournissant ce qu'ils ont appelé, autrement, du méta-calcul (Catlett et Smarr, 1992).

L'histoire récente de cette technologie comporte trois étapes principales. Les *grids ad hoc* de la première génération. Ces grids nécessitaient des solutions pour partager des ressources informatiques de haute performance. Le projet I-WAY (*Information Wide Area Year*) était sans doute le plus célèbre. Il concernait dix-sept sites différents aux États-Unis et était relié par dix réseaux sur lesquels résidaient les environnements virtuels, les ensembles de données, et les ordinateurs.

Avec l'arrivée des Web Services (WS), la seconde génération a migré vers des solutions *middlewares* qui supportent le traitement de l'information à grande échelle. Cette nouvelle technologie fournit un moyen d'interopérabilité qui est essentiel pour réaliser un calcul à grande échelle. Les middlewares représentent pour les grids ce que représentent les systèmes d'exploitation pour le matériel. Ils sont développés pour gérer la couche de bas niveau contenant les ressources, tout en cachant leur nature hétérogène et en fournissant des utilisateurs et des applications avec un environnement parfaitement homogène. Pour assurer ces fonctionnalités, les middlewares fournissent un ensemble d'interfaces normalisées pour accéder aux différents services offerts. Parmi les plus connus et les plus utilisés nous trouvons Globus (Foster et Kesselman, 1997) et Legion (Grimshaw et al., 1997).

Une nouvelle génération des grids basés sur la théorie des *agents* (Jennings, 2001) est maintenant en train d'émerger. Elle se présente comme une solution pour fournir des composants logiciels flexibles, autonomes, et intelligents. Les *MAS*[2] (Chao et al., 2004) sont des collections d'agents logiciels qui interagissent à travers le coopération, la coordination, ou/et la négociation afin d'atteindre des buts individuels ou/et communs. Dans ce contexte, l'utilisation de telles technologies nécessite une interaction entre les agents pour contrôler les composants principaux du grid. Pour effectuer ces opérations, les agents doivent avoir une connaissance parfaite de l'environnement où ils vont évoluer. En plus, d'une part, pour être universelle et indépendante d'une plateforme particulière, une couche abstraite est nécessaire entre les agents et le grid. D'autre part, et en raison de la nature distribuée de l'environnement, ceci exige la décomposition de cette couche en plusieurs composants qui satisferont mieux les contraintes du déploiement. En dernier lieu, des ponts ou des connecteurs sont nécessaires entre les éléments de cette nouvelle couche et les vrais fragments du grid physique.

Cet article présente les analyses de cette situation afin de définir un cadre théorique pour notre future recherche. Le but de ce travail est de définir une plateforme qui représente un grid "idéal", appelé *meta-grid*, qui peut abstraire tous les grids existants.

[1] En réalité, une grille de calcul est une sorte de grille informatique, dont la spécificité est la puissance de calcul, au même titre que les grilles de données, les grilles d'information, etc.

[2] un agent est un composant d'un MAS, mais dans cet article, nous utiliserons les deux d'une manière interchangeable pour référencer la même signification, les systèmes intelligents qui utilisent des agents.

Cet article commence par présenter quelques recherches qui ont abordé le même problème (Section 2), et expose dans les grandes lignes des questions encore en suspens qui motivent notre travail. Après, nous présentons le coeur de notre travail (Section 3) : le meta-grid. Nous donnons une brève vue d'ensemble sur la méthode que nous avons suivie. Ensuite, nous définissons la couche abstraite comme un diagramme de classes qui modélise l'ensemble des grids en tant qu'un seul modèle unique. Quelques scénarios sont donnés pour montrer comment l'architecture peut être animée. La section 4 présente la validation de notre modèle. À la fin, nous concluons en donnant quelques futures perspectives de ce travail (Section 5).

2 Travaux liés et motivations

La conception et le développement des systèmes logiciels pour le grid sont loin d'être des tâches faciles. En effet, ils doivent satisfaire une variété de conditions et de besoins potentiellement contradictoires : simplicité, passage à l'échelle, évolution, sécurité, interopérabilité, gestion facile, distribution optimale de charge, ordonnancement efficace, etc. Des nouvelles architectures logiciel seront nécessaires pour répondre à ces besoins (Shi et al., 2006), et la technologie agent apparaît comme une approche très prometteuse.

L'intégration grid-MAS est un sujet de recherches très actif. Cette nouvelle génération de systèmes grid est focalisée sur l'automatisation. Dans (Foster et al., 2004), les auteurs expliquent que les grids et les agents font partie des systèmes ouverts distribués, quoique leurs perspectives sont différentes. Ils décrivent le grid comme un "muscle" parce qu'il est concentré sur les infrastructures interopérables et les outils pour un partage de ressources sécurisé et fiable au sein des organisations virtuelles (VO) dynamiques et géographiquement distribuées. En revanche, dans cette métaphore, les agents sont des "cerveaux", parce qu'ils sont liés au développement des concepts, des méthodologies, et des algorithmes pour des solveurs de problèmes autonomes qui peuvent fonctionner avec souplesse dans des environnements incertains et dynamiques afin d'atteindre leurs buts et objectifs.

De ce point de vue, nous pouvons constater, encore une fois, que les agents sont l'une des solutions possibles qui peuvent aider à surmonter quelques faiblesses du grid dans l'état actuel. Surtout la rigidité et l'inflexibilité en terme de leurs interactions.

Nous pouvons voir que le grid résultant devrait avoir les capacités suivantes :
- contenir les connaissances détaillées de ses composants et de son statut ;
- construire les systèmes dynamiquement ;
- chercher à optimiser son comportement pour réaliser ses buts ;
- être conscient de son environnement ;
- et être facile à utiliser.

Guidé par ces objectifs, beaucoup de recherches ont été développées dans cette direction. Les travaux les plus intéressants pourraient être le concept de "Agent Grid" proposé par le programme "DARPA ISO's Control of Agent-Based System" (CoABS) (Manola et Thompson, 1999) et ARPMS (Cao et al., 2001).

Il y a d'autres travaux qui sont importants concernant l'utilisation des agents pour améliorer quelques fonctionnalités et caractéristiques spéciales des grids. À titre d'exemple, nous prenons l'ordonnancement (Gradwell, 2003), la gestion de la charge (Junwei et al., 2005), la gestion des ressources (Forestiero et al., 2006; Cao et al., 2002), ou la formation et la gestion des VO

(Shao et al., 2004), etc. D'autres chercheurs s'investissent dans des travaux plus théoriques visant la construction d'une nouvelle ontologie (Duvert et al., 2006). En revanche, d'autres essayent d'aboutir à une telle intégration d'un point de vue implémentation (Shi et al., 2006).

Pour illustrer l'importance des agents dans la résolution de problèmes, nous prendrons un exemple de la littérature. L'utilisation d'agents dans la formation de VO est sans doute une réelle opportunité. Dans un environnement grid, les fournisseurs de services sont autonomes. Toutefois, pour répondre à certains problèmes, leur alliance à un moment donné, est souvent requises afin de former une VO (*Virtual Organisation*) en répondant à une condition particulière du marché. La capacité de créer des VO fiables et *scalables* sur demande, dans un environnement ouvert et compétitif peut être une contribution des agents dans le monde de grid (Shao et al., 2004).

Tous ces travaux sont intéressants, et chacun d'eux résout un volet du problème d'une manière ou d'une autre. Mais, nous constatons qu'il y a beaucoup de travail redondant et par conséquent une perte significative d'efforts. À titre d'exemple, il y a un manque important de coordination entre les gens bien qu'ils soient guidés par les mêmes objectifs. Ceci est probablement dû aux spécificités techniques de leurs environnements (plateformes grid physiques).

D'autres problèmes sont encore en suspens. Par exemple, des grids installés avec des middlewares distincts ont souvent des problèmes pour communiquer entre eux. Cette difficulté mène à un mauvais partage de ressources et à une répartition de charge déséquilibrée : un site peut être surchargé par des tâches tandis que d'autres ressources sur d'autres sites sont en chômage. Le système pourrait alors bénéficier des capacités offertes par des grids physiques en acceptant les tâches venant des sites surchargés.

La modélisation des grids est très intéressante pour les MAS. Les agents doivent partager des connaissances communes concernant l'environnement où ils évoluent afin de négocier, coordonner leurs actions, et agir et réagir sur leur environnement. Les agents peuvent même être des entités mobiles. Leur indépendance des grids physiques est donc importante. Les technologies basées sur Java fournissent déjà des outils intéressants, mais les applications sont isolées juste par rapport aux systèmes d'exploitation. Les agents ont besoin d'un isolement au niveau des API (*Application programming interface*) et au niveau ontologique. Le premier principe leur permettra de discuter avec un grid de "référence" unique par l'intermédiaire d'une API commune. Le second permettra aux agents de parler une "lingua franca" au sujet des jobs, des décisions d'ordonnancement, des ressources, etc.

La dernière raison résulte de la complexité rencontrée par les développeurs à chaque fois qu'ils doivent développer des applications dédiées au grid : ils doivent comprendre les particularités de chaque grid avant de commencer leur travail proprement dit. En plus, ces applications demeureront spécifiques au grid. Les avantages d'un grid abstrait sont évidents pour parvenir à résoudre ce point.

Récapitulons les raisons essentielles :

- redondance des efforts ;
- manque de communication entre les différents grids ;
- absence de conscience des agents de leurs environnements ;
- concentration des développeurs sur les choses principales plutôt que sur des particularités des grids.

3 Meta-Grid

3.1 Principes

Nous proposons une architecture, le *meta-grid*, qui abstraira la notion du grid. Cette architecture agira en tant que couche indépendante entre les applications (typiquement les systèmes MAS) au niveau supérieur et les grids physiques comme nous le montrons dans la FIG. 1.

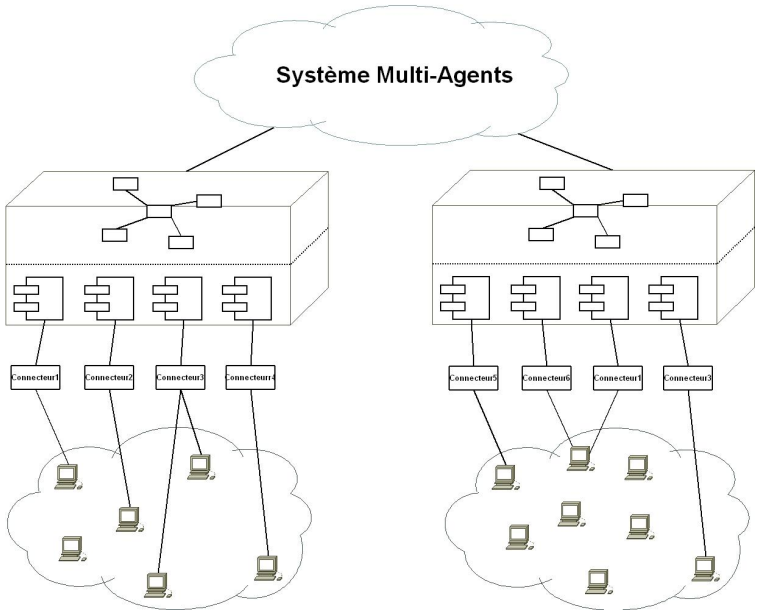

FIG. 1 – *Architecture du meta-grid*

Cette couche permet aux agents d'interagir avec le grid physique sans s'inquiéter de leurs détails techniques ni de leurs spécificités. D'autant plus, les agents deviennent indépendants des détails du déploiement. Les composants de cette couche représentent un environnement idéal pour notre plateforme grid basé sur les agents.

En pratique, chaque grid sera caché derrière une architecture de composants. Ces composants exposeront une interface bien spécifiée. Cette architecture peut elle-même être distribuée pour correspondre aux contraintes de déploiement du grid physique. Cette architecture sera reproduite pour chaque grid avec lequel un système MAS doit interagir. Chaque composant est relié au grid physique à l'aide des connecteurs qui lui sont spécifiques.

3.2 Méthode de recherche

Notre méthodologie pour obtenir cette architecture a commencé par l'analyse des études précédentes (Foster et al., 2001b; Nemeth et Sunderam, 2003), des grids représentatifs (Asadzadeh et al., 2004) et des normes[3] afin d'obtenir leurs modèles spécifiques. Ces modèles ont

[3]Cfr. Le forum "Open Grid Forum" : `http ://www.ogf.org`

été intégrés dans un modèle commun rassemblant tous les concepts présents. Ce modèle est présenté dans FIG. 2 comme un diagramme de classes UML. Cette proposition a été validée par des experts, des administrateurs et des utilisateurs des grids appropriés. Finalement, le diagramme de classes a été transformé en un diagramme de composants en agrégeant certaines classes. De cette façon, nous avons obtenu des composants avec une interface orientée métier.

Pendant notre étude des grids existants, nous nous sommes concentrés sur quatre technologies grid, à savoir : UNICORE[4], Globus[5], Legion[6], et Gridbus[7]. Ce choix a été motivé par leur vaste utilisation ainsi que la diversité des domaines couverts (scientifiques, industriels et commerciaux).

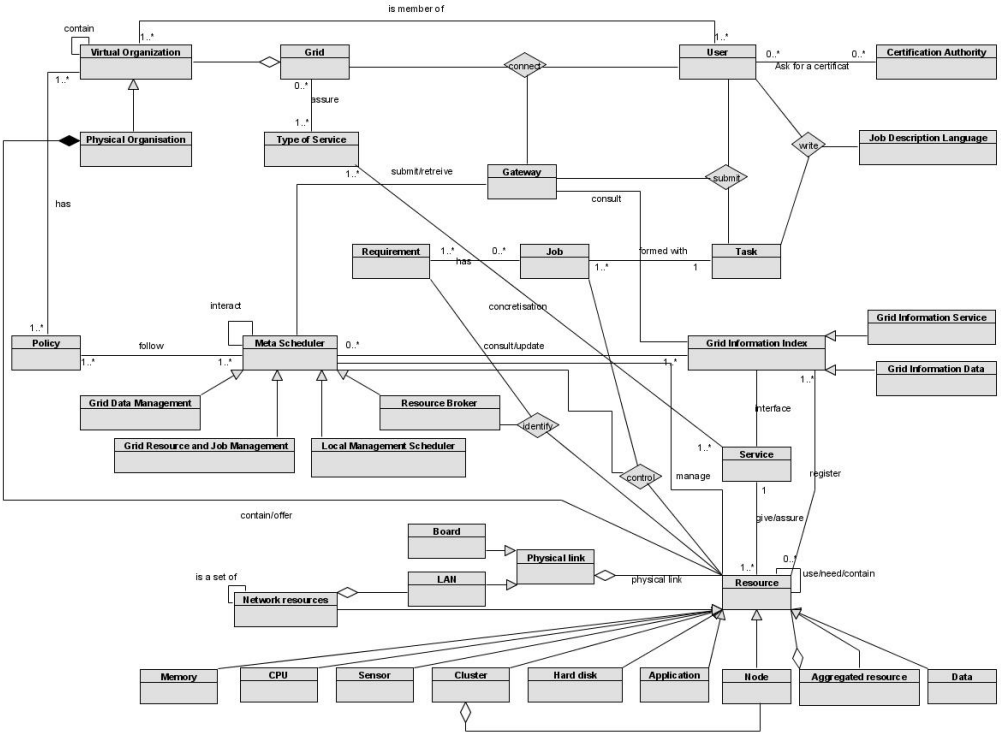

FIG. 2 – *Diagramme générique de classes du meta-grid*

Par la suite, nous avons intégré ces quatre modèles dans un modèle unifié (cf. FIG. 2). Les concepts semblables qui ont des noms différents ont été fusionnés, par exemple, les sites virtuels (Vsites) d'UNICORE et les organisations virtuelles (VO) de Globus, *Target System Interfaces* (TSI) et *Globus Resource Allocation Manager* (GRAM) dénotent le même concept dans les deux technologies. Quelques classes ont été intégrées par la fusion de leurs caractéristiques communes dans un super-type commun. À titre d'exemple, le portail du grid peut être

[4]http ://www.unicore.org
[5]http ://www.globus.org
[6]http ://legion.virginia.edu
[7]http ://www.gridbus.org/

un site Web ou une interface utilisateur. Ce concept a été modélisé par une classe généralisée (*Gateway*) avec les sous-classes spécifiques *User Interface* et *Web Portal*. Les concepts qui sont trop spécifiques ou trop techniques ont été enlevés quand leurs fonctionnalités ne sont pas essentielles. Quelques uns ont été "cachés" à l'intérieur des connecteurs. Le choix des entités a été fait, dans la mesure du possible, avec un respect des définitions standards (par exemple le *Open Grid Services Architecture*[8]).

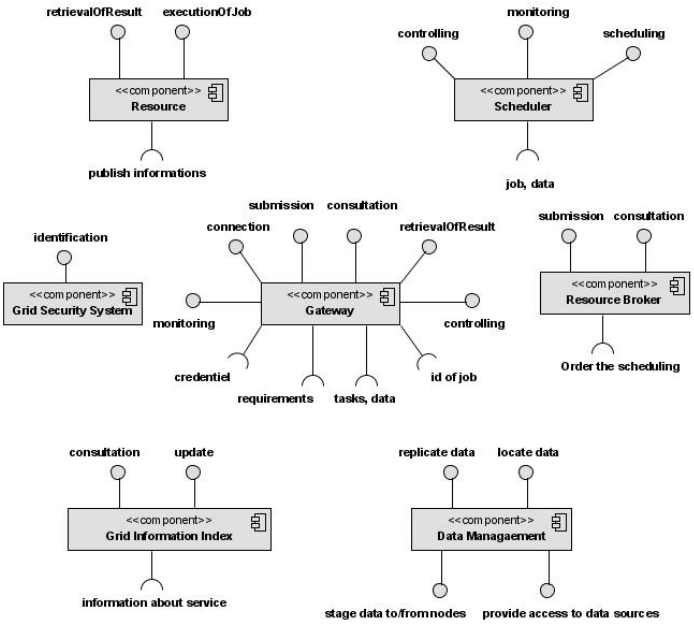

FIG. 3 – *Diagramne générique de déploiement du meta-grid*

Enfin, le modèle statique a été organisé en un modèle de composants (FIG. 3). Il représente l'API principale avec laquelle notre système MAS devra communiquer. Ces composants sont autonomes et peuvent être distribués à travers le grid.

3.3 Modèle integré

Cette section documente les concepts et les relations principaux modélisés dans le modèle statique intégré (FIG. 2). Les termes concernant des entités du modèle statique intégré apparaissent en style `typewriter`. Les attributs des entités sont omis dus à la limitation de l'espace et pour rendre le modèle plus lisible.

Nous suivrons une approche *top-down* pour commenter notre modèle. Le grid (`Grid`) se compose d'une ou plusieurs organisations virtuelles (`Virtual Organization`). Une VO est constituée d'une ou plusieurs organisations physiques (`Physical Organization`) (PO), comme elles peuvent contenir d'autres VO d'une manière hiérarchique. Chaque VO

[8]http ://www.globus.org/ogsa/

est caractérisée par un ensemble de politiques (`Policy`)[9]. Le méta-ordonnanceur (`Meta Scheduler`) prendra ses décisions sur base de ces politiques pour ordonnancer les tâches (`Task`). Le grid est considéré comme un ensemble de type de services (`Type of Service`) qui ne sont autre que des abstractions des services concrets (`Service`). Ces services sont réalisés par les ressources (`Resource`) offertes par les PO. Ces ressources peuvent être classifiées comme des données (`Data`), des mémoires (`Memory`), des liens physiques (`Physical link`), etc. Chaque ressource peut, à son tour, être composée d'autres ressources selon une sémantique bien définie qui dépend de l'agrégat : utilise (`use`), besoin (`need`), ou contenir (`contain`) (par exemple, une application (`Application`) utilise des CPU (`CPU`), les clusters (`Cluster`) contiennent des noeuds (`Node`). Les ressources sont enregistrées dans l'annuaire du grid rassemblant toutes leurs informations (`Grid Information Index`) (GII).

À cet endroit nous pouvons stocker des informations sur les services disponibles (`Grid Information Service` (GIS)) ainsi que sur les données (`Grid Information Data` (GID)) stockées sur ce grid. La notion du service comme considérée ici est très générique. Elle contient le matériel et le logiciel, les applications et les détecteurs,comme elle peut contenir des tâches effectuées par des agents humains, etc.

L'autorité de certification (`Certification Authority`) fournit des certificats aux utilisateurs (`Users`). Les utilisateurs peuvent alors soumettre des tâches qui sont décrites dans un langage de description de job (`Job Description Language`) (JDL) déterminé par le grid. Chaque tâche (`Task`) se compose d'un ensemble de jobs (`Job`) exprimés en termes de besoins et des conditions (`Requirement`) concernant leur exécution. En se connectant à un point d'entrée (`Gateway`), l'utilisateur peut soumettre ses jobs au courtier (`Resource Broker`) (RB) responsable de cette tâche. Ce dernier identifie les ressources adéquates en consultant le GII. Par la suite, la main sera passée à l'ordonnanceur local (`Local Management Scheduler`) (LMS) ou au gestionnaire grid des ressources et des jobs (`Grid Resource and Job Management`) (GRJM) pour le déploiement et le contrôle des jobs sur les ressources.

3.4 Scénarios

Cette section illustre plusieurs scénarios qui peuvent être joués avec notre modèle de composants (FIG. 4).

3.4.1 1^{er} scénario — Authentification

Quand un utilisateur (`User`) souhaite profiter des services (`Type of Service`) proposés par le grid (`Grid`), il doit d'abord obtenir un certificat de la part de son CA (`Certification Authority`). Cette dernière instance vérifie alors la validité de sa demande, et, si les conditions de délivrance d'une telle certificat sont satisfaites[10]. À la fin, la CA fournit un certificat

[9] Une politique est un ensemble de règles imposées par les POs, par les VO, par les administrateurs du grid, etc. La notion de la politique est cruciale, parce que, selon les règles suivies, nous pouvons décider si le grid est académique, ou commercial. Plus précisément, si entre les clauses de la politique, nous trouvons la notion du "coût de services" et du "mode de paiement" alors le grid est commercial. Encore plus loin, les agents qui vont exploiter la plateforme doivent respecter et prendre en considération les clauses de ces politiques durant la réalisation de leurs buts et objectifs

[10] Parmi les conditions vérifiées, nous pouvons trouver : est-ce qu'il est bien membre de la communauté dont il se déclare appartenir ? est-ce que sa communauté à le droit de bénéficier des services offerts par le grid en question ?, etc

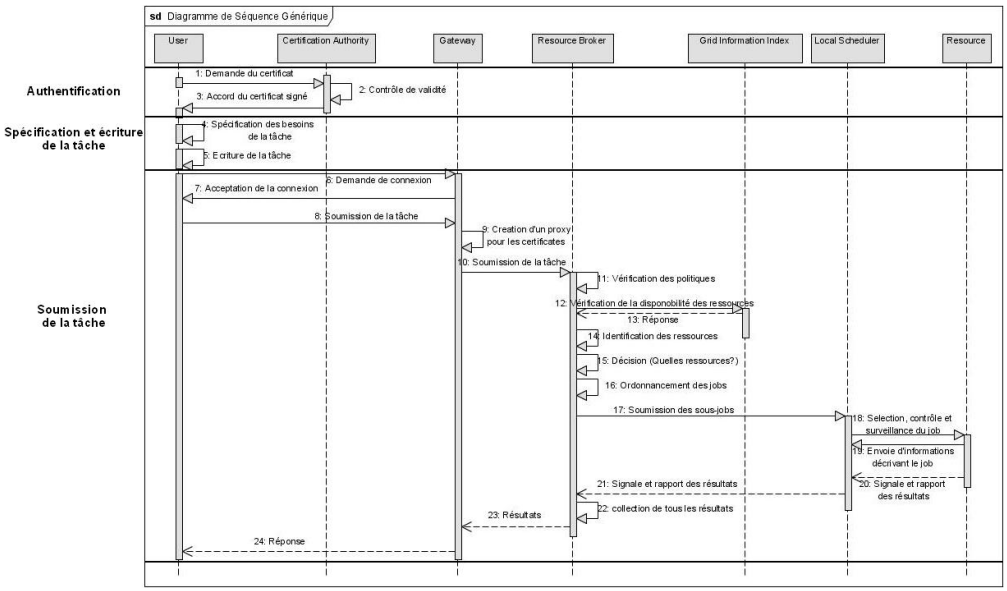

FIG. 4 – *Diagramme générique de séquence du meta-grid*

signé à l'utilisateur. Les certificats ne sont pas représentés dans le modèle statique, ils ont été modélisés en tant qu'attribut de l'utilisateur.

3.4.2 2^e scénario — Consultation des services disponibles

Un autre scénario qui est simple aussi. Il s'agit de la consultation des services disponibles. L'utilisateur (`User`) doit se connecter au grid par l'intermédiaire du point d'entrée (`Gateway`). Après, il consulte la base de données des informations (GII). Ceci afin de voir les informations sur les services (`Grid Information Service`) (GIS) et sur les données (`Grid Information Data`) (GID) stockées sur le site.

3.4.3 3^e scénario — Soumission d'une tâche

Le scénario le plus fréquent est probablement "la soumission d'une tâche". L'utilisateur (`User`) commence par spécifier les besoins (`Requirement`) de ses tâches (`Task`) et ensuite, il les encode, en les écrivant dans un langage de description de job (`Job Description Language`). Après, il doit se connecter au grid via le point d'entrée (`Gateway`). Une fois à ce stade, le point d'entrée soumet le job au courtier des ressources (`Resource Broker`). Sur base des spécifications (`Requirement`) et de l'information disponible dans GII, le RB identifie les ressources (`Resource`) adéquates. Une fois les ressources trouvées, le LS ou GRJM prend le contrôle du job sur la ressource en question. C'est ainsi que l'ordonnanceur peut choisir, contrôler, et surveiller les jobs.

3.5 Composants

Cette section se focalise sur un composant (`Gateway`) et clarifie son idée. Les autres composants sont régis par les mêmes règles.

Comme expliqué ci-dessus, les composants sont un mécanisme employé par le système MAS pour gérer et contrôler le grid physique. Ce mécanisme offre des points spécifiques d'entrée ; il s'agit des interfaces bien spécifiées.

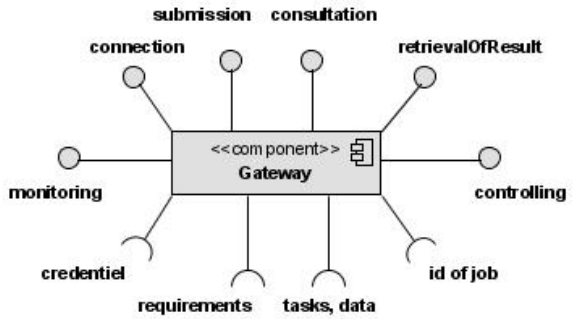

FIG. 5 – *Le composant "Gateway" du meta-grid*

Le composant "`Gateway`", FIG 5, fournit six fonctionnalités. À travers ces fonctionnalités, l'agent peut se connecter (*connection*) au grid. Pour que cette connexion soit acceptée, il faut que le certificat soit valide. Il peut, également, consulter (*consultation*) les services offerts par ce grid. Il peut soumettre (*submission*), surveiller (*monitoring*), et contrôler (*controlling*) un job, à travers ce composant. Une fois que le job est terminé, il peut récupérer (*retrievalOfResult*) ses résultats, toujours au moyen de cette interface.

De l'autre côté, ce composant fournit quelques informations. Cette information peut être utilisée par d'autres composants, d'autres agents ou être exploitée directement par les connecteurs.

Dans le cas du "`Gateway`", l'information distribuée, qui en sort, (*output*) peut être des *credentials* à authentifier par le composant qui s'occupe de la sécurité du grid, ou des conditions qui doivent être matchées avec le contenu du GII. Comme elle peut être une tâche ou/et ces données pour qu'elle soit traitée par le composant qui implémente le RB. À la fin, l'information peut être un identificateur de job pour vérifier le statut de la tâche en utilisant le composant ordonnanceur, ou pour récupérer les résultats d'un job terminé.

En d'autres termes, ces informations présentent les messages échangés entre les modèles.

4 La validation

Pour valider notre modèle, nous avons choisi de le soumettre à des spécialistes, des administrateurs, et des utilisateurs des technologies grid. Le premier groupe d'utilisateurs est une équipe de développeurs dans une unité de recherche en bioinformatique à l'Université Libre

de Bruxelles (Belgique). Cette équipe travaille avec Gria[11] qui est un middleware orienté business. La deuxième équipe était composée d'administrateurs du projet BEgrid[12], qui est le réseau belge du grid pour la science et la recherche. Ils utilisent Glite[13] comme middleware.

Le processus de validation a subi trois phases :

- Construction de nos premiers modèles sur base de notre expérience ainsi que sur base du bagage théorique issu de la bibliographie ;
- Soumission de nos premières ébauches à des gens du domaine pour une première correction et une amélioration ;
- Une dernière soumission aux experts pour une ultime validation.

5 Conclusion et perspectives

Dans cet article, nous avons motivé l'utilisation de technologies et des concepts orientés agents afin de construire des fédérations élargies de grids, incluant divers fournisseurs de services. À cette fin, il s'est avéré nécessaire de définir un framework de grid pivot, neutre et indépendant. Cette infrastructure a pour objectif de permettre le développement d'une couche orientée agents, capable d'intéragir avec des grids implementés dans diverses technologies non nécessairement compatibles. Elle permet également d'exposer des services, généralement cachés dans les interfaces business, afin d'opérer une médiation plus fine si nécessaire.

Le travail présenté dans cet article représente un résultat intermédiaire dans l'élaboration d'un grid orienté agent. Ce résultat préliminaire permettra à notre travail de thèse de sŠaffranchir des contraintes technologiques tout en restant interopérable avec les solutions industrielles. Nous poursuivons l'implémentation de la plate-forme meta-grid ainsi que du framework en question. Nous poursuivons, aussi, le développement de connecteurs entre notre meta-grid et quelques grids physiques du monde réel.

Références

Asadzadeh, P. et al. (2004). Global grids and software toolkits: A study of four grid middleware technologies. *Journal of Grid Computing citeseer.ist.psu.edu/735630.html*.

Cao, I., D. Kerbyson, et G. Nudd (2001). Performance evaluation of an agent-based resource management infrastructure for grid computing. *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, 311–318.

Cao, J. et al. (2002). Arms: An agent-based resource management system for grid computing. *Scientific Programming, IOS Press 10*, 135–148.

Catlett, C. et L. Smarr (1992). Metacomputing. *Commun. ACM June*, 44–52.

Chao, I., R. Sangüesa, et O. Oardaiz (2004). Grid resource management using software agents. *ERCIM News 59*.

Duvert, F. et al. (2006). Agent-grid integration ontology. *OTM Workshops 2006, LNCS 4277*, 136–146.

[11] http ://www.gria.org/
[12] http ://www.begrid.be/
[13] http ://glite.web.cern.ch/glite/

Forestiero, A., C. Mastroianni, et G. Spezzano (2006). Agent-based logical organization of resource in grids. *Proc. of the Workshop on the Use of P2P, GRID and Agents for the Development of Content Distribution Networks*, 135–149.

Foster, I., N. Jennings, et C. Kesselman (2004). Brain meets brawn: why grid and agents need each other. *3rd International Joint Conference on Autonomous Agents and Multiagent Systems 1*, 8–15.

Foster, I. et C. Kesselman (1997). Globus: A metacomputing infrastructure toolkit. *Int. J. Supercomput Appl. 11*, 115–128.

Foster, I. et C. Kesselman (1999). *The Grid : Blueprint for a New Computing Infrastruture*. Morgan Kaufmann.

Foster, I., C. Kesselman, et S. Tuecke (2001a). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal on High Performance Computing Applications 15*, 200–222.

Foster, I., C. Kesselman, et S. Tuecke (2001b). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications 15*, 200–222.

Gradwell, P. (2003). Grid scheduling with agents. Technical report, http://peter.gradwell.com/phd/writings/computing-economy-review.pdf.

Grimshaw, A. et al. (1997). The legion vision of a worldwide virtual computer. *Commun. ACM 40*, 115–128.

Jennings, N. (2001). An agent-based approach for building complex software systems. *Communications of the ACM 44*, 35–41.

Junwei, C. et al. (2005). Grid load balancing using intelligent agent. *Future Generation Computer Systems 21*, 135–149.

Manola, F. et C. Thompson (1999). Characterizing the agent grid. Object services and consulting, http://www.objs.com/agility/tech-reports/990623-characterizing-the-agent-grid.html.

Nemeth, Z. et V. Sunderam (2003). Characterizing grids: Attributes, definitions, and formalisms. *Journal of Grid Computing*.

Shao, J. et al. (2004). Supporting formation and operation of virtual organisations in a grid environment. *Proceedings of UK e-Science All Hands Meeting, IOS Press*.

Shi, Z. et al. (2006). Agent-based grid computing. *Applied Mathematical Modelling, Elsevier 30*, 629–640.

Summary

With the increasing focus on grid development there is a need for proper abstraction for modelling grids. In this paper a platform for agent-based grid framework is presented. This platform takes the form of a meta-grid that consists of an abstract grid representing “ideal” kinds of grids. This layer is developed to deal with problems raised by the differences between grids, and by other technical details specific to each one by hiding them and specifying an uniform model. The creation of such platform will dispense the programmer of applications of caring about problems specific to the environment and let him concentrate only on the develop-

ment of the applications themselves. Therefore, this structure will help us to conceptualize and abstract the environments of grids. The multiagent system that we want to create will interact with this meta-grid to solve the real world problems without caring about specificities proper to each physical grid.