Vers une Architecture de Type Agent BDI pour un Ordonnanceur de Grille Adaptatif

Inès Thabet[*], Chihab Hanachi [**], Khaled Ghédira[*]

*Laboratoire LI3 (ENSI) Université de Manouba, Manouba 2010, Tunis, Tunisie
Ines_thabet@yahoo.fr
Khaled.Ghedira@isg.rnu.tn
[**]Institut de Recherche en Informatique de Toulouse IRIT, UMR 5505,
Université Toulouse 1, 2 rue du Doyen-Gabriel-Marty, 31042 Toulouse Cedex 9
Chihab.Hanachi@irit.fr

Résumé. Un Ordonnanceur de Grille (OG) a pour objectif d'optimiser l'utilisation des ressources mises à sa disposition. Le plus souvent, ce composant manque de flexibilité dans sa conception et a des difficultés à s'adapter aux diverses perturbations (fluctuation de la qualité de service, arrivée ou départ de nouvelles ressources, arrivée de travaux prioritaires, etc). L'objectif de cet article est de proposer une architecture orientée agents de type BDI (Belief Desire Intention) pour permettre à l'OG de s'adapter à ces perturbations et de travailler en coopération avec ses accointances. L'OG proposé fonctionne en boucle selon un schéma perception-délibération-action. La perception de l'environnement et de ses perturbations est rendue possible ici à l'aide d'un modèle conceptuel qui décrit à la fois la structure de la Grille et son fonctionnement. La délibération décide des actions à entreprendre (migration d'application, changement de plan d'ordonnancement, changement de politique d'attribution des tâches,..) pour s'adapter aux perturbations sur la base de règles qui proposent différents types d'adaptation selon les événements détectés. L'action correspond à la mise en oeuvre des décisions prises lors de la délibération.

1 Introduction

La Grille de calcul (Foster et Kesselman, 1999; Foster et al., 2001) est reconnue comme un concept indispensable pour supporter l'exécution d'applications diverses (scientifiques, médicales, physiques,etc.) nécessitant une grande puissance de calcul ou une importante capacité de stockage. Un de ses composants logiciels essentiels est l'Ordonnanceur de Grille (OG) qui a pour objectif d'utiliser au mieux les ressources mises à sa disposition. Lors d'une demande d'exécution d'une application sur la grille, l'OG a pour rôle de répartir dans le temps l'exécution des tâches constituant l'application, en tenant compte des règles de précédence entre les tâches, des critères de qualité de service (performance) et des informations sur l'état des ressources.

De nombreux travaux et systèmes tels que Nimrod (Abramson et al. 2000), Legion (Chapin et al., 1999), Netsole (Casanova et Dongarra, 1997) se sont intéressés à la conception et à l'implémentation de ce composant, mais très peu d'entres eux ont pris en

considération la complexité et le caractère dynamique et instable de l'environnement de la Grille. Cette complexité provient notamment du fait que les ressources sont hétérogènes, géographiquement distribuées sur de multiples domaines, et autonomes dans la mesure où elles disposent de leurs propre politique de traitement des tâches et de contrôle d'accès. La dynamique est due au caractère ouvert de la Grille qui autorise les ressources à joindre ou quitter la grille à tout moment. L'instabilité découle de cette complexité, de la dynamique et du fait que les ressources ont des performances d'exécution hétérogènes et non uniformes.

Afin de profiter des avantages de la Grille, l'OG doit tenir compte des contraintes décrites ci-dessus imposées par l'environnement et être adaptatif. Le problème traité dans cet article est la définition d'un OG adaptatif c'est à dire capable de percevoir son environnement et ses perturbations, raisonner sur ces éléments et décider des réactions à entreprendre afin de maintenir ou améliorer la performance de la Grille.

L'examen de l'état de l'art montre que très peu d'OG sont adaptatifs et, quand des solutions sont proposées, elles sont ad-hoc, difficilement généralisables ou contraignantes pour les utilisateurs. Nous avons notamment étudié les systèmes GraDs (Berman et al., 2001 ; Mellor-Crummey et al., 2004 ; Dail et al., 2002), AppLeS (Berman et al., 2003) et les techniques d'ordonnancement basés sur les contrats de performance (Vadhiyar et Dongarra, 2005 ; Reed et Mendes, 2005). Le système AppLeS (Berman et al., 2003) est un système à base d'agents qui met en oeuvre un ordonnancement adaptatif en incorporant des informations statiques et dynamiques sur les ressources, des prédictions de performances, des informations spécifiques aux utilisateurs ainsi que des techniques d'ordonnancement qui adaptent l'exécution de l'application en ligne. Dans GraDs (Vadhiyar et Dongarra, 2005), un composant logiciel est chargé de l'adaptation qui consiste à migrer l'exécution d'une tâche de manière opportuniste ou parce que la ressource exécutant la tâche ne satisfait pas les critères de performance attendus. Cependant ces systèmes présentent une limitation majeure dans la mesure où l'Ordonnanceur doit être intégré dans l'application à exécuter, ce qui rend ces systèmes difficilement maintenables et généralisables à un large spectre d'applications. De plus, ils imposent aux utilisateurs d'exprimer les besoins en ressources, de fournir un modèle de performance spécifique à l'application et de prévoir des plans de transfert des données, ce qui est très contraignant pour des utilisateurs non informaticiens.
En ce qui concerne les contrats de performance (Reed et Mendes, 2005), ils permettent de formaliser, qualifier et quantifier la relation entre les besoins en performance des applications et les capacités des ressources. En cas de violation d'un contrat, l'adaptation se fait soit par la révision du contrat soit par le réordonnancement de l'application sur un ensemble de nouvelles ressources qui satisfont les spécifications du contrat. L'avantage essentiel du contrat est sa souplesse dans la mesure où les modalités de l'adaptation ne sont pas figées mais peuvent faire l'objet d'une négociation entre les applications et les ressources. Les faiblesses de ce travail résident dans la lourdeur de la gestion des contrats qui, par ailleurs, est spécifique à chaque application. Une dernière limite que l'on peut adresser aux systèmes et techniques présentés ci-dessus est que l'adaptation se limite à du réordonnancement (réaffectation des tâches à de nouvelles ressources). Or, d'autres types d'adaptation peuvent être envisagés tel que le changement de politique d'ordonnancement, l'adaptation du processus d'exécution, l'adaptation de la topologie de la Grille, etc.

L'objectif de cet article est de proposer une architecture pour un Ordonnanceur de Grille adaptatif, faiblement couplé aux applications mais capable de coopérer, et permettant d'exprimer plusieurs modes d'adaptation. Cette architecture est basée sur la technologie agent (Wooldridge et Jennings 1995) qui permet de prendre assez naturellement en compte la complexité du contexte de la Grille et notamment, l'autonomie, la distribution et la nature dynamique de ses composants. Elle permet également de concevoir des composants logiciels dotés de capacités sociales et cognitives, ce qui explique son utilisation dans de nombreux travaux de gestion de Grille (Cao et al., 2002, 2002 ; Shi et al., 2007 ; Chunlin et Layuan, 2004).

Pour permettre à l'OG de raisonner, de collaborer et de prendre des décisions d'adaptation dirigées par des buts (maintenir une qualité de service, maximiser l'utilisation des ressources) en tenant compte de l'état de l'environnement, nous avons choisi une architecture de type BDI (Belief Desire Intention ou Croyance Désir Intention en français) (Rao et Georgeff, 1995) qui permet un fonctionnement cyclique selon un schéma perception-délibération-action.

Pour permettre la perception de l'environnement et de ses perturbations, nous proposons également un modèle conceptuel qui décrit à la fois la structure de la Grille et son fonctionnement. Afin de permettre la délibération et l'action, nous avons : i) identifié un ensemble de situations nécessitant une adaptation et qui porte sur les objets du modèle conceptuel ii) proposé un ensemble d'actions à entreprendre (migration d'application, changement de plan d'ordonnancement, changement de politique d'attribution des tâches,..) pour s'adapter aux perturbations sur la base de règles qui proposent différents types d'adaptation selon les événements détectés.

Cet article est organisé comme suit. La section 2 présente notre modélisation des concepts impliqués dans l'ordonnancement dans la Grille et détermine ainsi les objets sur lesquels porte l'adaptation. Dans la section 3, nous présentons une classification des types d'événements perturbateurs et les types d'adaptation à entreprendre. La section 4 décrit une architecture orientée agents de type BDI qui exploite les modèles conceptuels (proposés en section 2) et les règles d'adaptation (proposées en section 3) pour permettre à l'OG de s'adapter aux perturbations de l'environnement. Enfin, nous concluons par un aperçu des perspectives offertes par ce travail.

2 Modélisation conceptuelle de l'ordonnancement dans la Grille

Dans cette section, nous présentons les concepts impliqués dans l'ordonnancement ainsi que les relations qu'ils entretiennent entre eux. Les concepts définis ainsi que les propriétés citées ont été choisis suite à une étude approfondie du domaine (Pugliese et al., 2006; Dong et Akl, 2006 ; Schopf, 2004 ; Fibich et al., 2005). Deux points de vue sont à envisager : le point de vue structurel et le point de vue fonctionnel. Chacun de ces points de vue donne lieu à un modèle décrit ci-dessous. Les instances de ces modèles définissent les objets sur lesquels peut porter l'adaptation et ils permettent à l'OG d'une part de percevoir et de caractériser l'état de la Grille et d'autre part de délibérer pour décider des actions à entreprendre pour l'adapter.

2.1 Modèle conceptuel de l'aspect structurel

Le Service d'ordonnancement de la grille est assuré par la coopération de deux composants qui sont l'ordonnanceur de Grille (OG) et le gestionnaire local de ressources. L'OG utilise à la fois des informations statiques et dynamiques. Les informations statiques ont une qualité qui ne change pas au cours du temps et dont la mesure est effectuée une seule fois. Les informations dynamiques évoluent dans le temps et concernent l'ensemble des données qui peuvent être recueillies sur la progression de la soumission d'une application ou l'exécution d'une tâche et sur le suivi des disponibilités. Le modèle proposé intègre ces deux types d'informations ainsi que l'ensemble des entités participantes à l'exécution d'une application de Grille.

Les concepts les plus importants de ce modèle (cf. figure 1) sont :

- *Ressource*. Elle représente la classe centrale de notre modèle. Il s'agit d'un dispositif de base qui peut être un réseau, un noeud de stockage ou un calculateur. Chaque ressource est identifiée de manière unique et possède des caractéristiques statiques telles que le nom, la capacité mémoire, la performance crête de calcul, le système d'exploitation, etc.
- *Organisation*. Une ou plusieurs ressources de la grille peuvent former une organisation offrant ainsi un nouveau ou une meilleure qualité de service. Chaque organisation dispose de son propre protocole d'entrée/sortie de ressources.

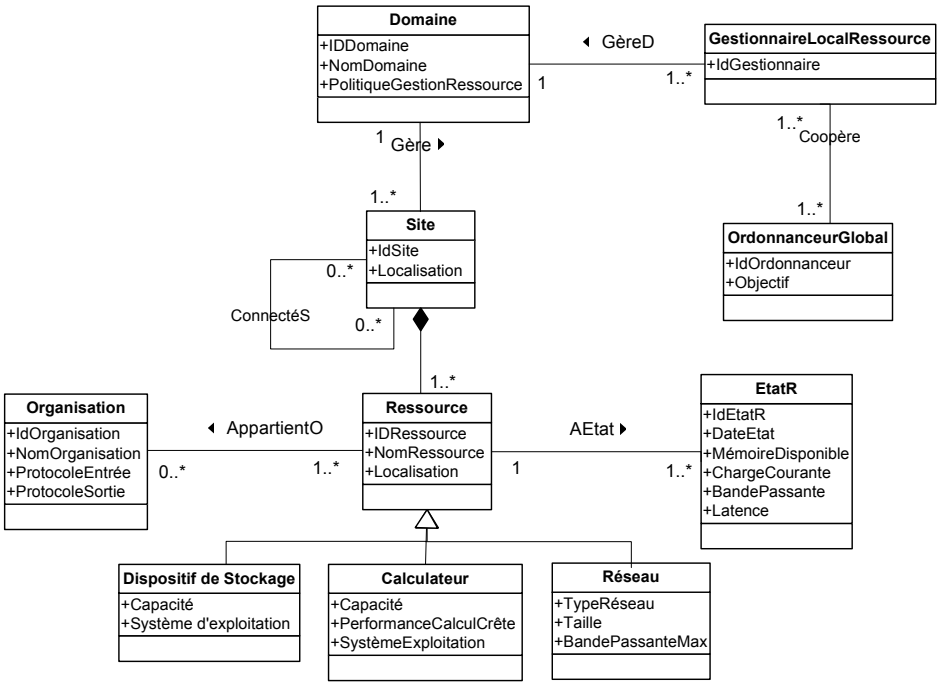

FIG. 1 – *Modèle Conceptuel de l'aspect Structurel.*

- *Site*. Il est défini par un ensemble composé d'une ou de multiples ressources et géré par un domaine administratif permettant de définir la politique locale de gestion de ressource.
- *EtatR*. Il représente les informations dynamiques d'une ressource telles que la mesure de bande passante, la latence, la charge processeur, la mémoire disponible, etc.
- *Ordonnanceur de Grille (OG)*. Il reçoit les requêtes des utilisateurs et les ordonnance selon un objectif qu'il vise à atteindre (maximisation de l'utilisation des ressources, minimisation des temps d'exécution des applications, etc). L'ordonnancement peut être assuré par plusieurs ordonnanceurs déployés sur la Grille et organisés de manière décentralisée ou hiérarchique.
- *Gestionnaire local de ressource*. Il gère seul ou en collaboration avec d'autres gestionnaires l'ordonnancement des travaux dans un domaine. Ces travaux peuvent être soumis par l'OG mais aussi par des utilisateurs locaux. Il fournit aussi les informations sur l'état des ressources.

2.2 Modèle conceptuel de l'aspect applicatif

La figure 2 représente l'ensemble des concepts impliqués lors de l'exécution d'une application sur la Grille. Nous distinguons à droite la description des applications à exécuter ainsi que les ressources à disposition et, à gauche, les éléments faisant l'objet d'une exécution ainsi que leurs états.

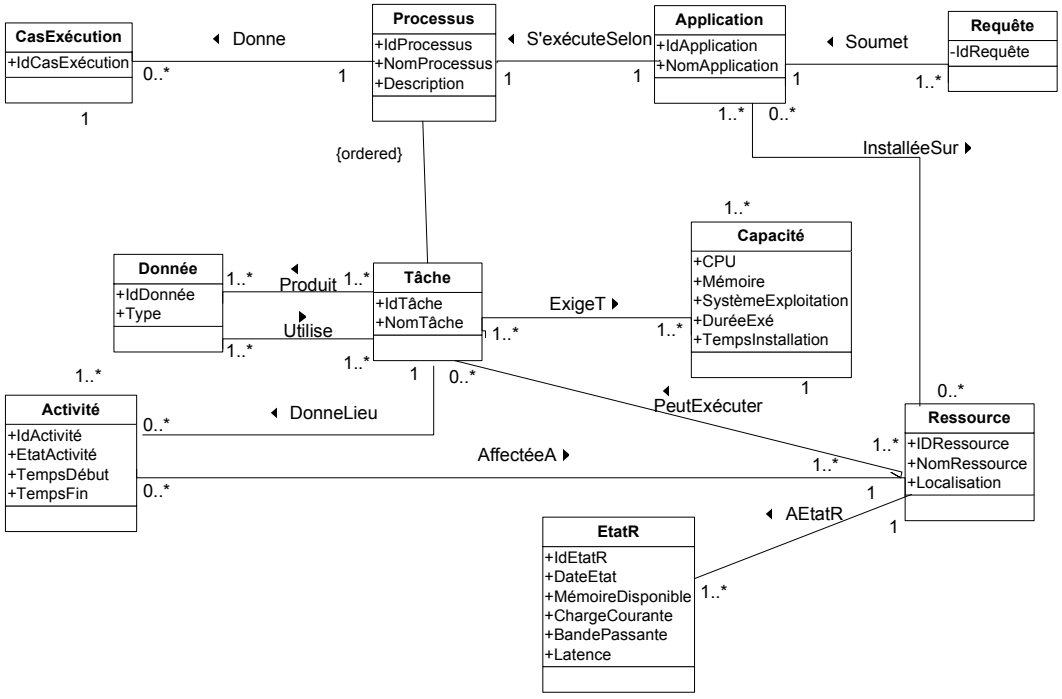

FIG. 2 – *Modèle Conceptuel de l'aspect applicatif.*

- *Requête.* Elle représente une demande extérieure d'exécution d'une application (médicale, biologique, météorologique, etc.).
- *Application (travail).* Elle est définie par un ensemble de tâches coordonnées selon un processus et qui seront exécutées sur un ensemble de ressources. Une application a des besoins nécessaires pour son exécution tels que les types et les quantités de ressources, ou l'intervalle de temps requis sur ces ressources (CPU, mémoire requise, durée d'exécution, temps d'installation, etc.). Ces informations sont enregistrées dans l'entité capacité.
- *Ressource*. En plus des caractéristiques décrites dans le modèle précédent, la ressource offre un ou plusieurs services dont la mise à disposition du code de l'application (*lien « installée sur »*) et l'exécution locale d'applications (*lien « affectée à »*) pour peu que ses capacités et son état le permettent.
- *Capacité*. Elle peut définir les exigences d'une application ou d'une tâche ou bien les capacités d'une ressource. L'OG a pour rôle la mise en correspondance entre les capacités des ressources et les besoins des applications.
- *Processus*. Il décrit l'ordre dans lequel l'exécution des tâches d'une application doit être réalisée. Un processus peut engendrer plusieurs plans d'exécution alternatifs.
- *Tâche*. Elle désigne une unité de traitement atomique qui sera exécutée sur une ressource. La tâche comporte de la même manière que l'application un certain nombre de besoins en ressources (CPU, mémoire, date de début, date de terminaison, etc.) enregistrées dans l'entité capacité.
- *CasExécution*. L'exécution d'une application selon un processus donne lieu à un cas d'exécution.
- *Activité*. L'exécution d'une tâche sur une ressource de la Grille donne lieu à une activité. Cette entité enregistre un ensemble d'informations nécessaires à l'adaptation de l'ordonnancement tels que l'état de l'activité (affectée, en attente d'exécution, en cours d'exécution, interrompue ou terminée) et les résultats d'exécution sur une ressource donnée. Ces résultats sont utilisés pour la prédiction de performance d'une ressource donnée.

En plus des concepts définis, notre modèle présente une contrainte d'intégrité que nous spécifions textuellement. Elle concerne la durée d'exécution d'une application et la somme des durées des tâches qui la composent. Ces durées sont définies à travers les exigences des applications et des tâches et sont enregistrées dans la classe capacité. La contrainte est la suivante : « la durée d'exécution d'une application doit être supérieure ou égale à la somme des durées d'exécution de ses tâches ». Ceci est dû au temps nécessaire à la synchronisation des tâches de l'application et notamment la transmission des données entre elles.

3 Ordonnancement adaptatif dans les Grilles de calcul

L'environnement de Grille peut être perturbé par des événements divers : apparition/disparition de ressources, panne d'une ressource, arrivée d'une tâche prioritaire, surcharge ou demandes concurrentes d'une ressource, etc. L'adaptation consiste à réagir à ces perturbations pour maintenir ou améliorer un fonctionnement nominal. Alors que les perturbations concernent principalement les ressources, l'adaptation peut concerner différentes entités constituantes de la Grille telles que décrites dans les modèles conceptuels précédemment présentés (figure 2 et 3).

Dans la suite, nous définissons d'une part une typologie des événements perturbateurs, et d'autre part, une typologie des adaptations. Ces deux éléments servent à définir des règles d'adaptation utilisées par notre architecture (section 4). Ces règles font correspondre aux événements les actions d'adaptations.

3.1 Typologie des événements perturbateurs

- *Apparition/Disparition de ressources*. L'environnement de Grille est un environnement ouvert que les ressources peuvent librement joindre ou quitter et, de ce fait, perturber l'ordonnancement. L'arrivée d'une nouvelle ressource perturbe dans un premier temps l'organisation de la Grille mais peut, par la suite être utile pour l'amélioration des performances.
- *Arrivée de tâches prioritaires*. La soumission d'une tâche (prioritaire, plus rentable, locale, etc.) peut venir modifier l'ordre d'exécution des tâches.
- *Panne d'une ressource*. Correspond à la situation où une ressource est momentanément indisponible. Elle continue à faire partie du système, mais seul son état change. Ce problème peut se produire au niveau d'un noeud de stockage qui conserve les données nécessaires à l'exécution, au niveau d'un noeud de calcul sur lequel l'application est entrain de s'exécuter, mais aussi au niveau des réseaux de connexion, où une bande réseau devient non disponible car affectée par un très grand trafic ou par une congestion. La panne d'une ressource peut dégrader la qualité de service et avoir aussi des conséquences en chaîne que l'on peut détecter à l'aide de notre modèle (changement de l'état tâche, de l'état ressource, restructuration de l'organisation, etc.).
- *Surcharge ou Concurrence des applications pour les ressources au moment de l'exécution*. Les tâches peuvent s'exécuter en parallèle sur une même ressource ou être en attente de la disponibilité de la ressource. Cependant une tâche peut se conduire de manière inattendue en consommant plus de ressources ou de temps que prévu, bloquant ainsi l'exécution d'autres tâches concurrentes.

3.2 Typologie des adaptations

Nous avons présenté dans la section précédente les événements perturbateurs. Ici nous identifions les types de réactions possibles. Les types d'adaptation se différencient par plusieurs critères : temporel (réactif, pro-actif), la dynamique (statique ou émergente), l'objet sur lequel porte l'adaptation (plan, processus, organisation des ressources, politique d'allocation des ressources aux tâches, etc.).

3.2.1 Adapatation réactive/pro-active

- *Adaptation réactive.* Elle est mise en oeuvre dès que l'ordonnanceur détecte un évènement perturbateur. Il exécute alors un ensemble d'actions correctives.
- *Adaptation pro-active.* Elle suppose une surveillance permanente de l'environnement qui permet à l'ordonnanceur d'intervenir de manière opportuniste par anticipation. Par exemple, si une ressource a été trop longtemps sollicitée, il pourrait être judicieux d'alléger sa charge. Si deux ressources fonctionnent de

manière optimale quand elles sont associées il pourra être bénéfique d'en faire une organisation, etc.

3.2.2 Adaptation statique/émergente

- *L'adaptation statique.* Elle correspond à une adaptation dont les règles sont prédéfinies. A titre d'exemple, nous citons la règle suivante: **Si** « Evénement perturbateur = panne ressource » **Alors** « calculer nouveau plan d'affectation des tâches aux ressources ».
- *Adaptation émergente.* Elle suppose que l'ordonnanceur a la capacité de produire de nouvelles règles d'adaptation.

3.2.3 Différenciation des adaptations selon l'objet sur lequel elles portent

- *Adaptation des processus*. Pour illustrer nos propos, nous pouvons représenter formellement un processus par un réseau de Petri où les tâches correspondent aux transitions, les données nécessaires à une tâche aux places d'entrée de la transition correspondante et les données produites par une tâche aux places de sortie de la transition correspondante. La structure du réseau exprime la coordination de ces tâches et peut décrire des structures de contrôles diverses : séquence, alternation et parallélisme. De ces structures de contrôle, on peut déduire plusieurs plans d'exécution du processus. L'adaptation d'un processus revient à privilégier un plan par rapport à un autre, à supprimer certains plans qui ne sont plus réalisables dans le contexte courant, etc. Ces adaptations auront pour effet de modifier la structure du réseau sous-jacent.

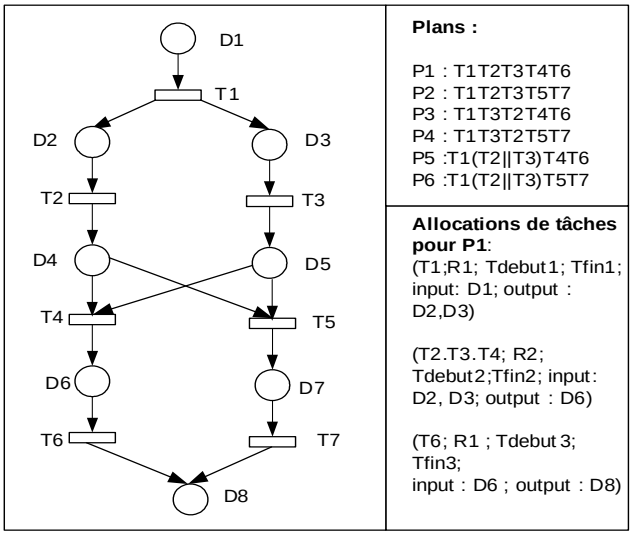

FIG.3 – *Modélisation du processus d'application par un réseau de Petri.*

La figure 3 montre un processus d'exécution d'une application modélisée par un réseau de Petri. Cette application est composée de sept tâches T1à T7. La structure du réseau exprime les contraintes de précédences entre les tâches. Considérons maintenant l'hypothèse où le marquage initial du réseau de Petri est un jeton dans D1. Les tâches T2 et T3 ont besoin respectivement des données D2 et D3 en entrée et peuvent s'exécuter en parallèle. Les tâches T4 et T5 ont besoin des données D4 et D5 en entrée et s'excluent mutuellement. T5 et T6 s'exécutent séquentiellement. L'ensemble des exécutions possibles de ce processus est un ensemble de plans (P1, P2, P3, P4, P5 et P6). L'ordonnancement va consister à choisir un de ces plans et à allouer à ces tâches des ressources. Dans l'exemple, le plan P1 est choisi, et ses tâches sont affectées aux ressources R2 et R1 en précisant le temps de début et le temps de terminaison, les données à utiliser et celles à produire. On remarque que des regroupements de tâches ont été opérés sur la ressource R2.

- *Adaptation de plan d'allocation de tâches.* En cas d'un événement perturbateur sur une ressource, l'allocation des tâches aux ressources doit être remise en cause. Un nouveau plan d'allocation est alors calculé et exécuté par l'OG. Cette adaptation exige que l'infrastructure de Grille sous-jacente offre des mécanismes de migration de tâches, des points de contrôle, etc. La plupart des systèmes d'ordonnancement adaptatifs supportent ce type d'adaptation (Mellor-Crummey et al., 2004 ; Berman et al., 2003; Vadhiyar et Dongarra, 2003, 2005; Reed et Mendes, 2005).
- *Adaptation de la politique d'allocation de tâches.* L'OG met en oeuvre des politiques d'allocation qui peuvent être fixes ou variables et dépendantes des ressources et de leurs états. L'adaptation consiste ici à modifier cette politique suite à une perturbation. A titre d'exemple, on peut citer le passage de l'utilisation d'un algorithme d'ordonnancement statique qui utilise des informations sur les prédictions des performances, à un algorithme dynamique qui va équilibrer les résultats de l'ordonnancement statique et considérer l'état courant des ressources. Un autre exemple est celui où, suite à l'arrivée de nouvelles tâches prioritaires, l'OG décide de lancer une enchère pour déterminer le plus offrant en terme de qualité de service ou de coût, au lieu de suivre une politique classique de type push ou pull.
- *Adaptation du niveau de qualité de service* : suppose d'abord de définir des critères de qualité (profit économique, taux d'utilisation de la ressource, durée d'exécution, coût d'exécution de l'application) qui peuvent être quantitatifs (100 Mb/s ou 10 ms) ou qualitatifs (haut, moyen, faible), une échelle de valeurs pour chacun des critères ou pour des combinaisons de ces critères et un seuil de tolérance en deçà duquel le système doit s'adapter pour maintenir ce seuil ou entrer dans une négociation pour relaxer ce seuil comme dans (Li et Yahyapour, 2006).
- *Adaptation de structure organisationnelle* : certaines ressources ne sont pas isolées mais appartiennent à une organisation dans laquelle elles tiennent un rôle (fonction ou type de service). Dail (Dail et al., 2003) utilise un algorithme d'ordonnancement qui a pour objectif la recherche d'un groupe de machines candidates afin de réaliser une correspondance entre les tâches de l'application et les ressources du groupe de machines. L'organisation se base sur le groupement des ressources d'un même site ou d'un même domaine administratif (délai de communication dans chaque sous ensemble inférieur au délai entre les sous ensembles). Cependant, les ressources peuvent disposer de suffisamment d'intelligence pour pouvoir former des coalitions (organisation virtuelle) afin de pouvoir exécuter collectivement une application, que

la ressource seule ne pourrait exécuter. Dans ce contexte, l'adaptation consiste à réorganiser ces structures organisationnelles (ajout, suppression de ressources, modification des rôles…). Au sein d'une même organisation, des groupes de ressources peuvent être formés sur la base d'observations de leur performance collective. Par exemple, on peut remarquer que les ressources R et S réunies exécutent de manière plus performante l'application A que les ressources S et U. Cela peut amener l'OG à donner une préférence au premier groupe de machines quand il s'agira d'exécuter l'application A. Dans le cas de coalitions, la ressource qui en prend l'initiative peut modifier ses accointances en fonction des coopérations passées et de la qualité des résultats obtenus avec chacune de ses accointances. Ceci peut être géré par des mécanismes de réputations tels que les mécanismes utilisés dans le projet CONISE-G (Dail et al., 2003).

4 Architecture de l'Ordonnanceur de Grille à base d'agents BDI

L'objectif de cette section est de proposer une architecture capable de percevoir son environnement et de s'adapter à ses perturbations. La perception se fera à l'aide des modèles définis en section 2 et les actions d'adaptation se font conformément aux événements et types d'adaptation décrits en section 3. Pour cela, nous avons opté pour une architecture Belief Desire Intention (Croyance Désir Intention) (Widom et Ceri, 1996) dans laquelle i) les croyances permettent de capturer l'état de l'environnement y compris les demandes d'exécution d'applications provenant des utilisateurs, ii) les désirs formalisent les objectifs assignés à l'OG et, iii) les intentions sont les plans (ensemble d'actions) mis en oeuvre par l'OG. On distinguera par la suite les plans d'allocation des tâches aux ressources qui ont pour objectif de répartir l'exécution des tâches sur les ressources, des plans d'adaptation qui ont pour objectif d'adapter le fonctionnement de la Grille face à des perturbations. Les plans sont rattachés aux objectifs (désirs) qu'ils permettent d'atteindre. Examinons plus en détail ces différents composants et leurs interactions.

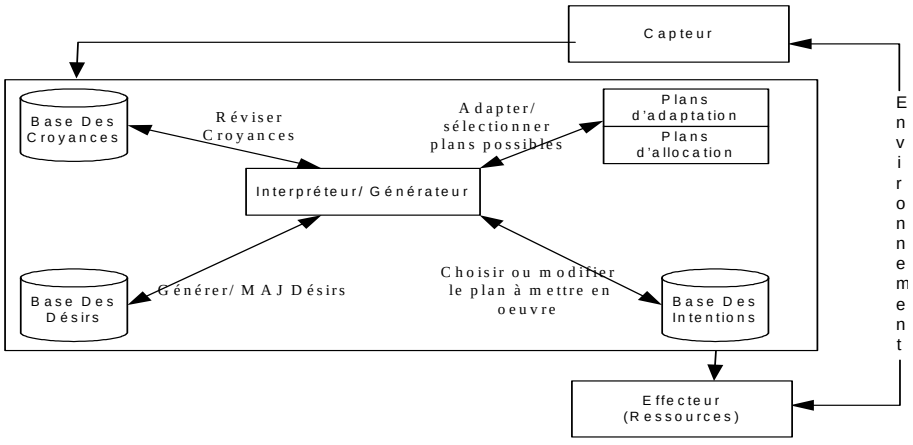

FIG.4 – *Architecture BDI de l'Agent Ordonnanceur.*

- *Les Croyances* : sont des informations sur l'environnement que l'agent possède ou acquiert via des capteurs. Dans le modèle proposé, les croyances représentent l'ensemble des informations sur les ressources, leurs disponibilités (mémoire disponible, charge processeur, largeur bande réseau…), leurs états (en attente/ libre/ en panne…), les informations sur les exigences des applications et des tâches (mémoire requise, système d'exploitation, temps de début, temps de terminaison…) et sur leurs états (affectée/ en attente/ en cours d'exécution/ terminée…). Ces éléments sont enregistrés conformément au modèle conceptuel présenté dans la section 2 et ils sont rafraîchis à partir des événements perturbateurs présentés en section 3 (apparition ou disparition de ressources, panne ressource, arrivée de tâche prioritaire…) qui peuvent subvenir et causer des changements de l'état des ressources et des applications.
- *Les Désirs* : sont les objectifs que l'OG vise à atteindre. Il s'agit d'objectifs de haut niveau de la Grille. Ces objectifs peuvent être la maintenance ou l'amélioration de la qualité de service offerte aux applications, la maximisation des profits des ressources ou la minimisation des coûts d'exécution des applications dans un contexte économique.
- *Les plans d'allocation des tâches aux ressources* : L'OG reçoit des demandes d'exécution d'applications soumises par des utilisateurs, sélectionne les ressources nécessaires pour leurs exécutions en se basant sur ses croyances (exigences des applications et informations sur les ressources), sur une prédiction de performance (calcul d'une estimation du temps pris par chacune des ressources potentielles pour exécuter une tâche). L'OG utilise ces données en entrée, génère un ensemble de plans faisables d'allocation des tâches aux ressources, choisit celui qui maximise son objectif et l'ajoute à sa base des intentions pour l'exécuter.
- *Les plans d'adaptation* : sont des actions à entreprendre pour s'adapter aux événements perturbateurs. Les réactions, comme précédemment définies peuvent correspondre à l'un des types suivants : adaptation de l'allocation des tâches aux ressources, adaptation de la structure, adaptation de la qualité de service, adaptation de la politique d'allocation des tâches, etc. Les plans peuvent s'exprimer sous la forme de règles du type (Widom et Ceri, 1996) : Quand <**E**vénement> Si <**C**ondition> Alors <**A**ction>. La partie <**E**vénement> concerne la situation qui doit prévaloir dans l'environnement et correspond à des perturbations. La partie <**C**ondition> correspond à des propriétés des objets de notre modèle (tâches, ressources…) et la partie <Action> à des adaptations. Voici deux exemples de règles[1]:

  R1 : **Quand** « panne ressource R» **Si** « Tâche T en cours sur R » **Alors** « Réaffecter T sur une ressource S ayant au moins la même capacité que R».

  R2 : **Quand** « Arrivée ressource R » **Si** « il existe une ressource S ayant au plus les mêmes capacités que R et s'il existe une tâche T en attente d'exécution sur S» **Alors** « réaffecter T sur R ».

  Ce formalisme a de nombreux avantages procurés par le style déclaratif des règles. Ces règles offrent un pouvoir d'expression qui permet de décrire aisément le

[1] Bien que rédigées informellement, ces règles ont une formulation possible en logique des prédicats.

comportement d'agents BDI comme précisé dans (Rao et Georgeff, 1995) et elles sont facilement maintenables et évolutives du fait de leur relative indépendance les unes vis-à-vis des autres. Enfin, l'interprète chargé d'exploiter ces règles -l'OG dans notre cas- peut être conçu et modifié indépendamment de ces règles.
- *Les intentions* : correspondent aux plans choisis et à exécuter parmi un ensemble de plans possibles via des effecteurs (ressources). Les plans choisis doivent être conformes à la réalisation des désirs (objectifs). La sélection de plans peut se baser sur une hiérarchie des plans selon leur degré d'efficacité (estimé à priori ou mesuré) dans la réalisation d'un objectif.

L'OG fonctionne selon un cycle composé de 3 étapes :

1. *Perception de l'environnement et Mise à jour des croyances* (nouvelles pannes, entrée nouvelle ressource, baisse des performances, arrivée de tâches prioritaires)

2. *Délibération*
   2.1 *Révision des Désirs* (Objectifs d'ordonnancement : minimiser le temps d'exécution, équilibrer les charges, maximiser l'utilisation,…)
   2.2 *Génération et sélection de plan(s) à exécuter* (affecter les tâches sur des ressources oisives, adapter le processus d'une application, réorganiser la grille, etc.).

3. *Action* : Exécution d'un Plan

L'exécution modifie l'état de l'environnement et relance le cycle.

5 Conclusion

Nous avons mis en avant le caractère hétérogène et dynamique de la Grille et la variation imprévue de la performance de ses ressources qui rend difficile le service d'ordonnancement. Ceci nous a naturellement conduit à envisager le problème de l'adaptation de l'ordonnancement. Nous avons proposé pour cela une architecture d'ordonnanceur de Grille (OG) basée sur le modèle orienté agent Belief Desire Intention. L'OG fonctionne selon un cycle Perception-Delibération-Action. Nous avons donné un schéma conceptuel de l'ensemble des données statiques et dynamiques qui permettent de réaliser sa fonction de perception. Nous avons donné les principes du processus de raisonnement ou de délibération et précisé le format des plans à mettre en oeuvre pour assurer l'adaptation. Une typologie des plans a également été proposée qui permet de mettre en correspondance des événements perturbateurs et des actions à entreprendre pour permettre à l'OG de réagir et de s'adapter.

Bien que ce travail reste à un niveau conceptuel, dans la mesure où l'Ordonnanceur n'a pas encore été implémenté, il comporte d'ores et déjà plusieurs avantages qui découlent en partie de l'architecture BDI. L'OG est doté d'un raisonnement qui lui permet d'avoir un contrôle sur son comportement puisqu'il décide des actions à entreprendre en fonction de l'état de l'environnement, des événements et d'un objectif. Par ailleurs, l'expression des plans d'adaptation sous forme de règles permet d'exprimer plusieurs types d'adaptation et d'obtenir une base de plans d'adaptation facilement maintenable et évolutive. Les règles s'expriment sous la forme Quand <**E**vénement> Si <**C**ondition> Alors <**A**ction>, la partie action se base

sur une typologie des adaptations que nous avons identifiée et présente aussi une des originalités de notre travail. La qualité du raisonnement dépend de la qualité de la fonction de perception, elle-même basée sur la qualité du modèle de l'environnement. Nos modèles conceptuels (section 2 et 3) contribuent à cet objectif et, à notre connaissance, de tels modèles n'ont jamais été explicités dans d'autres travaux. Par ailleurs, étant en interaction avec son environnement, l'OG reconsidère en permanence son comportement et peut agir de manière coopérative. L'OG a également l'avantage d'être séparé des autres composants de la Grille dans la mesure où il est un composant à part entière donc plus facile à modéliser, maintenir et adapter. Étant découplé des applications à ordonnancer, l'OG leur impose très peu de contraintes.

Ce travail ouvre des perspectives. Pour chaque concept faisant l'objet d'une adaptation, il est essentiel de définir des paramètres qui permettent de mesurer l'efficacité effective des plans d'adaptation afin de pouvoir les comparer et les hiérarchiser.

L'architecture BDI pourrait également être validée par une simulation afin de mesurer des métriques d'efficacité notamment la complexité, la robustesse et le passage à l'échelle face aux perturbations. L'utilisation d'un outil tel que (Bellifemine et al, 2006) qui intègre l'architecture BDI est d'ores et déjà envisagée dans cette perspective.

L'architecture BDI est connue pour faciliter la coopération et un de nos objectifs à long terme est de progresser vers un système multi-agents où les ressources de la grille sont des agents et les gestionnaires locaux des agents BDI pouvant s'adapter localement.

Références

Abramson, D., J. Giddy and L. Kotler (2000). *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid*. Mexico: 14th International Symposium on Parallel and Distributed Processing (IPDPS): 520-528.

Bellifemine, F. L., G.Cairo and D.Greenwood (2006). *Developping multi-agent systems with JADE*. Wiley Edition.

Berman, F., A.Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski (2001). *The GrADS Project: Software Support for High-Level Grid Application Development*. International journal of supercomputer Application, 15(4): 324-344.

Berman, F., R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, D. Zagorodnov (2003). *Adaptive Computing on the Grid Using AppLeS*. IEEE Transactions on Parallel and Distributed Systems, 14(4): 369-382.

Cao, J., P. Spooner, J. D. Turner, S. A. Jarvis, D. J. Kerbyson, S. Saini, and G. R. Nudd (2002). *Agent-based Resource management for Grid Computing*. Berlin, Germany: 2nd IEEE/ACM International Symposium : 350-350.

Cao, J., S.A. Jarvis, S. Saini, D.J. Kerbyson, and G.R. Nudd (2002). *ARMS: an Agent-based Resource Management System for Grid Computing*. In Scientific Programming (special issue on Grid Computing), 10(2): 135-148.

Casanova, H. and J. Dongarra (1996). *NetSolve: A Network Server for Solving Computational Science Problems*. Pittsburgh, USA: Proceedings of the 1996 ACM/IEEE conference on Supercomputing.

Chapin, S. J, D. Katramatos, J. Karpovich and A. S. Grimshaw (1999). *The Legion Resource Management System*. San Juan, Puerto Rico: 5th Workshop on Job Scheduling Strategies for Parallel Processing.

Chunlin, L. and L. Layuan (2004). *Agent framework to support the computational grid*. Journal of systems and software, 70 (1-2): 177-187.

Dail, H., H. Casanova and F. Berman (2003). *A modular scheduling approach for Grid Application Development Environment*. Journal of Parallel and Distributed Computing.

Dail, H., H. Casanova, and F. Berman (2002). *A Decoupled Scheduling Approach for the GrADS Environment*. Baltimore, Maryland USA: ACM/IEEE conference on Supercomputing: 1-14.

Dong, F., and S.G. Akl (2006). *Scheduling Algorithms for Grid Computing: State of the Art and Open Problem.* Technical Report No. 2006-504, Queen's University, Canada.

Fibich, P., L. Matyska and H. Rudovà.(2005) *Model of Grid Scheduling Problem*. California, USA: In Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing,, pp. 17-24.

Foster, I. and C. Kesselman (editor) (1999). *The Grid: Blueprint for a future computing Infrastructure*. USA : Morgan Kaufmann.

Foster, I., C. Kesselma, and S. Tuecke (2001). *The Anatomy of the Grid: Enabling Scalable Virtual Organization*. International J. Supercomputer Applications, 15(3): 200-220.

Li, J. and R.Yahyapour (2006). *Negotiation Model Supporting Co-Allocation for Grid Scheduling*. Barcelona: The 7th IEEE/ACM International Conference on Grid Computing: 254-261.

Mellor-Crummey. J., F. Berman, H. Casanova, A. Chien, H. Dail, X. Liu, A. Olugbile, O. Sievert, H. Xia, L. Johnsson, B. Liu, M. Patel, D. Reed, W. Deng, C. Mendes, Z.Shi, A. YarKhan and J. Dongarra (2005). *New Grid Scheduling and Rescheduling Methods in the GrADS Project*. International Journal of Parallel Programming, 33(2): 199-206.

Patel, J., W T Luke Teacy, N. R. Jennings, S. Chalmers, N. Oren, T. J. Norman, A.Preece, P. M D Gray, M. Luck, G.Shercliff, P. J. Stockreisser, J. Shao, W.A Gray, N. J Fiddian and S. Thompson (2005*). Agent based formation of virtual organisation*. Knowledge-based systems, 17: 103-111.

Pugliese, A., D.Talia and R. Yahyapour (2006). Modeling and supporting grid scheduling. CoreGRID Technical Report Number TR-0056.

Rao, A. S. and M. P. Georgeff (1995). *BDI Agents: From Theory to Practice*. The First International Conference on Multiagent Systems: 312-319.

Reed, D.A. and C.L. Mendes (2005). *Intelligent Monitoring for Adaptation in Grid Applications*. Proceedings of the IEEE, 93(2): 426- 435.

Schopf, J.M (2004). *Ten actions when Grid scheduling: the user as a Grid scheduler*. Grid resource management: state of the art and future trends: 15-23. USA, Kluwer Academic.

Shi, Z., H. Huang, J. Luo, F. Lin, H. Zhang (2007). *Agent based Grid computing*. Rio de Janeiro, Brazil: 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007).

Vadhiyar, S. and J. Dongarra (2003). *A Performance Oriented Migration Framework for the Grid*. Tokyo, Japan: The 3rd International Symposium on Cluster Computing and the Grid (CCGrid'03): 130-139.

Vadhiyar, S. S. and J. J. Dongarra (2005). *Self Adaptivity in Grid Computing*. In Principles and Practice of Parallel Programming: 235 – 257.

Widom, J. and S. Ceri (1996). *Active Database system*. Morgan Kaufmann Publishers.

Wooldridge, M. and N. R. Jennings (1995). *Intelligent Agents: Theory and Practice Knowledg*. Knowledge Engineering Review, 10(2): 115-152.

Summary

The Grid Scheduler (GS for short) aims at optimizing the use of resources at its disposal. More often, this component lacks flexibility in its design and has difficulties to adapt its behaviour to the various perturbations (quality of service degradation, arrival or departure of new resources, priority work submissions, etc). The objective of this paper is to propose a BDI-based (Belief Desire Intention) agent architecture allowing the GS to adapt to these perturbations and to work in cooperation with its acquaintances. The proposed GS performs a cycle made of three steps: perception-deliberation-action. The perception of the environment and its perturbations is made possible here using a conceptual model which describes at the same time the structure of the Grid and its functioning. The deliberation step determines actions to undertake (migration of application, rescheduling, scheduling policy change...) to adapt to the perturbations. This step is based on rules proposing various types of adaptation according to detected events. The action step corresponds to the execution of the actions determined by the deliberation.