OAR : gestionnaire de ressources pour grandes grappes de calcul

Bruno Bzeznik, Joseph Emeras, Romain Cavagna, Richard Olivier

OAR: un gestionnaire de ressources pour grandes grappes de calcul

Bruno Bzeznik
Projet CIMENT / UJF
LJK, 51 rue des Mathématiques, 38400 Saint Martin d'Hères

Romain Cavagna
INRIA Rhone-Alpes
ENSIMAG, 51 avenue Jean Kuntzmann, 38330 Montbonnot Saint Martin

Joseph Emeras
INRIA Rhone-Alpes
ENSIMAG, 51 avenue Jean Kuntzmann, 38330 Montbonnot Saint Martin

Olivier Richard
LIG
ENSIMAG, 51 avenue Jean Kuntzmann, 38330 Montbonnot Saint Martin

Résumé

Les gestionnaires de ressources qui équipent en général les super-calculateurs parallèles dans les infrastructures de calcul assurent le lancement et veillent au bon déroulement des taches de calcul ainsi qu'au partage des ressources selon des règles établies. Ils offrent également des outils pour l'administration de ces machines. OAR est un gestionnaire de ressources pour grandes grappes de calcul, qui offre une grande polyvalence d'emploi, des fonctionnalités originales et qui a pour ambition de passer facilement à l'échelle. C'est un logiciel libre développé au sein du LIG (Laboratoire d'Informatique de Grenoble) dans un esprit de collaborations fortes entre recherche et production. Nous présentons ici les principales fonctionnalités de cet outil ainsi que son architecture. Ensuite, nous présenterons quelques retours d'expérience et les projets qui gravitent autour de OAR.

Mots clefs

Infrastructures de calcul, Super-calculateur, high performance computing, grappe de calcul, cluster, batch scheduling, ordonnancement, gestion de ressources, calcul parallèle

1 Introduction

Les gestionnaires de ressources (aussi appelés *Batch Scheduler*) sont un élément clé dans l'exploitation des infrastructures de calculs scientifiques. Ces infrastructures intégrent généralement une ou plusieurs grappes homogènes composées de quelques dizaines à quelques centaines de serveurs de calcul. Les utilisateurs sont majoritairement des scientifiques qui souhaitent exécuter une ou plusieurs applications séquentielles ou parallèles (aussi appelées tâches) sur un ou plusieurs jeux de données en entrée. C'est le gestionnaire de ressources qui va orchestrer les demandes de ressources nécessaires pour l'exécution des tâches, puis assurer leur lancement le moment venu sur les ressources choisies par l'algorithme d'ordonnancement, et finalement faire le suivi de l'ensemble du cycle de vie des tâches dans le système.

Les administrateurs par l'intermédiaire des gestionnaires de ressource, peuvent mettre en place des politiques d'exploitation spécifiques comme, par exemple, fixer des priorités entres les demandes des utilisateurs ainsi que surveiller la bonne utilisation globale des ressources de calcul (taux d'utilisation, surveillance globale de l'activité).

Les gestionnaires de ressources ont vu le jour dans les années 80, dès lors ils ont évolué au grès des évolutions technologiques, des besoins des utilisateurs et des administrateurs. A l'heure actuelle, les défis auxquels doivent faire face ces outils sont principalement: l'augmentation de la complexité des infrastructures (nombres de coeurs, structures hiérarchiques des machines), la variété des cadres d'utilisation, et l'apparition de nouveaux objectifs d'exploitation (notamment la maîtrise énergétique).

Dans cet article, nous présentons le logiciel OAR qui est développé par les équipes MESCAL et MOAIS du laboratoire LIG [1]. Les développements sont aussi fortement soutenus par la communauté Aladdin/Grid5000 [2] et le projet CIMENT [3] de l'Université Joseph Fourier qui sont actuellement les 2 plus importants utilisateurs.

Nous introduirons rapidement les motivations et l'historique du développement du logiciel, puis nous présenterons en détail les fonctionnalités de cet outil. Dans la section suivante nous exposons l'architecture du logiciel. Dans la section "retours d'expériences", nous évoquons 3 cas d'utilisation d'OAR. Enfin, nous terminerons par une conclusion qui fait état des perspectives d'évolution.

2 Motivations et Historique

Le développement du gestionnaire OAR a débuté en 2003. A cette époque l'INRIA Rhône-Alpes disposait d'une grappe, nommée Icluster1, composée de 225 noeuds monoprocesseurs. Le gestionnaire utilisé pour gérer les ressources était OpenPBS (qui est devenu par la suite Torque [6]). OpenPBS souffrait de problèmes liés au passage à l'échelle qui nécessitaient de le relancer régulièrement. Nous avons alors considéré d'autres alternatives libres comme Sun Grid Engine ou des logiciels propriétaires comme PBS-Pro. L'ensemble de ces logiciels nous posaient plusieurs problèmes. Tout d'abord les logiciels propriétaires nous empêchaient d'expérimenter et de développer de nouvelles fonctionnalités dans le domaine de la gestion de ressources. De plus les logiciels libres comme OpenPBS ou Sun Grid Engine [7], après examen de leur code source, nous ont semblé trop délicats à modifier et en désaccord avec notre vision des choix de conceptions que nous souhaitions étudier. Nous avons donc initié les développements d'un nouveau gestionnaire de ressource sous licence GPL nommé OAR, qui est construit autour d'une base de donnée et de composants logiciels de haut niveau ayant pour objectif d'être polyvalent, personnalisable et possédant de bonnes capacités de passage à l'échelle.

Il a est noté qu'à la même époque a débuté les développements du gestionnaire de ressources SLURM[8] , qui est spécialisé dans les très grandes grappes qui disposent généralement de caractéristiques matérielles spécifiques comme les machines BlueGene d'IBM.

Depuis les premiers développements jusqu'aux évolutions récentes, le gestionnaire OAR a su intégrer les fonctionnalités fondamentales des gestionnaires de ressources ainsi que celles répondant aux besoins apparus récemment.

3 Fonctionnalités

OAR est un gestionnaire de ressources ainsi qu'un ordonnanceur de tâches pour grandes grappes de calcul.

Son rôle est de planifier et de gérer les exécutions des tâches (jobs) sur les ressources ainsi que de manager les ressources elles-mêmes. Sa flexibilité lui permet d'être utilisé sur des grappes de production comme de recherche.

Son principe de fonctionnement est le suivant: un utilisateur fait une demande de ressources associée à une tâche. Le système ordonnance sa requête en fonction des disponibilités et lui retourne soit une date à laquelle sa tâche va demarrer soit un terminal sur une des ressources attribuées.

Comme beaucoup d'autres gestionnaires de ressources, OAR

- permet de lancer des taches en mode interactif (via un shell sur la ressource) ou en mode « batch »
- permet de soumettre par réservation; une tâche sera lancée à une date définie à l'avance (*advance reservation*)
- est multi files d'attente avec des priorités et un ordonnanceur différents pour chaque file (plusieurs ordonnanceurs sont fournis: fifo, fifo-with first-fit/backfilling, fairsharing)
- intègre des règles d'admission ce qui permet une grande souplesse de configuration de l'admission des tâches dans les files d'attente
- intègre la notion de tableaux de jobs: possibilité de soumettre en une seule fois de nombreux jobs aux propriétés similaires

dépendant éventuellement de paramètres

- permet de définir des dépendances d'exécution entre les tâches (Directed Acyclic Graph et Workflows)
- gère les propriétés des ressources: les ressources peuvent être de différents types (cpus, licences,...) et avoir de nombreuses propriétés qui vont permettre à l'utilisateur de les sélectionner finement
- supervise les noeuds: possibilité de paramétrer des "checks" pour activer ou désactiver automatiquement des ressources
- fait de la comptabilité: OAR maintient un résumé des consommations par utilisateur et par projet.

Cependant de nombreuses fonctionnalités sont plus spécifiques à OAR, comme

- l'utilisation d'un type de job spécifique: « best-effort » qui permet d'exploiter les ressources inutilisées

- la description dynamique d'une hiérarchie de ressources à la soumission : permet de définir finement la hiérarchie d'une grappe, par exemple "switch/noeud/socket/core" et facilite la gestion des grappes hétérogènes
- un algorithme de fairsharing avec définition de projets : partage intelligent des ressources en fonction du taux d'utilisation par utilisateur et par projet
- l'ordonnancement par diagramme de Gantt permettant la visualisation des décisions
- le support des licences comme types de ressources
- des scripts Prologue/Epilogue qui permettent de lancer des scripts personnalisés au démarrage et en fin de job, sur la frontale ou sur les noeuds
- l'exploitation des cpusets linux pour l'optimisation et le nettoyage efficace en fin de job
- l'optimisation de la chaine d'exécution sur les noeuds via Taktuk (optionnel)
- la possibilité d'intégration d'outil de déploiement (Kadeploy) pour l'exécution d'un job dans le but de déployer un OS spécifique
- la configuration minimale des noeuds; le seul prérequis indispensable sur chaque noeud de calcul est la présence de "Openssh" et de quelques scripts perl fournis
- la prise en compte de l'économie d'énergie : possibilité d'éteindre/allumer les noeuds à la demande
- la possibilité de deployer des senseurs sur les noeuds pour faire du monitoring.

Le principe du job « best-effort » est le suivant: l'utilisateur qui soumet dans ce mode sait ce qu'il fait. Il accepte le fait que si une ressource qui lui est allouée est explicitement requise par un autre utilisateur, son job sera terminé prématurement. Ce mode est particulierement bien adapté aux codes de calcul parallèle qui peuvent être facilement redémarrés. Ce mode présente le grand avantage de permettre de réserver une grande partie des ressources de la grappe sans pour autant monopoliser celle-ci.

Quand à la définition des ressources de manière hiérarchique, OAR permet non seulement de définir quelle est cette hiérarchie mais aussi le niveau de granularité que l'on veut. Par exemple, on pourra considérer que la plus petite ressource pourra descendre jusqu'au niveau du coeur, ou bien seulement au niveau machine. Ensuite c'est selon comment ont été initialisées les ressources que l'administrateur a défini une hiérarchie implicite. C'est ensuite à l'utilisateur lors de la soumission de definir quelle hiérarchie il veut utiliser.

Plusieurs outils périphériques à OAR sont aussi fournis, des outils de visualisation web très pratiques et configurables et un outil d'administration des ressources du cluster:

- **DrawGantt** : visualisation de l'occupation des ressources dans le temps (diagramme de Gantt). Ce dernier est écrit en Ruby.
- **Monika** : il permet la visualisation de l'état instantané de toutes les ressources de la grappe. Il est écrit en Perl.
- **Oaradmin**: outil d'administration pour faciliter la création initiale des ressources et la gestion des règles d'admission.

OAR propose aussi des outils de grille comme CiGri et OARGrid qui permettent l'utilisation d'OAR dans un contexte grille.

De plus une API REST est disponible depuis peu et permet d'interagir avec OAR à travers des URI. Cette api permet la soumission de jobs, la visualisation des jobs en cours et de l'êtat des ressources, la gestion des jobs, la gestion et la création des ressources. Le grand bénéfice de cette approche est de pouvoir, par exemple, construire aisément des interfaces web qui transcriront les clics des utilisateurs en appel à des URI qui elles mêmes feront des appels en REST à l'API débouchant ainsi à des actions sur OAR. Mais c'est aussi un moyen pour les utilisateurs de scripter plus facilement leur utilisation d'un cluster ou d'une grille. Enfin, cette API permettra certainement la réalisation d'interfaces vers des protocoles standards utilisés par les grilles.

4 Architecture

OAR connaît deux types de composants dans son architecture: un ensemble de modules et une base de données.

Le point central de l'architecture d'OAR est sa base de données. C'est elle qui contient toute la mémoire et l'état du système. Elle est aussi le centre nerveux des communications entre les différents modules d'OAR qui communiquent donc essentiellement à travers elle. OAR prend en charge deux types de SGBD: MySQL et Postresql.

Quant à la partie modulaire, c'est elle qui donne sa souplesse à OAR. Chaque module a été conçu dans un but spécifique (logging, surveillance de l'état des ressources ou des jobs, lancement, exécution ou terminaison des jobs...) et est appelé par un module chef d'orchestre.

Cette approche permet le développement de modules complémentaires écrits dans n'importe quel langage ayant une librairie d'interfaçage avec les bases de données.

Le fonctionnement général de OAR est présenté dans la figure ci-dessous. Un automate gère le lancement des differents modules d'OAR, qui, une fois leur tâche effectuée, vont se terminer. Ceci nous permet de rester sans état.

Un ensemble de commandes utilisateur (en ligne de commande) vont interagir avec la base de données pour permettre la soumission de taches, la visualisation des jobs en cours, de l'êtat des ressources... Il en est de même pour les commandes administrateur qui vont accéder à la base pour mettre à jour les propriétés des ressources, gérer les noeuds, …

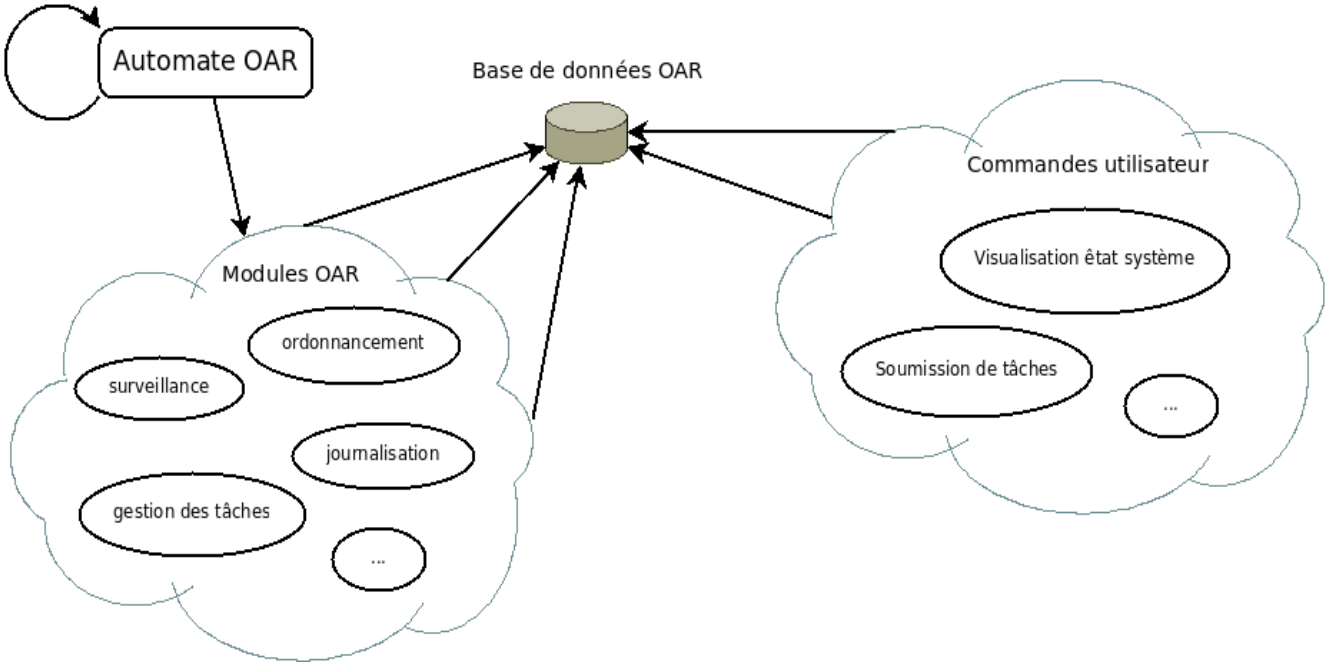

Figure 4.1 - *Architecture de OAR*

Le fait de centraliser toutes les informations dans la base de données a permis de créer facilement et indépendemment du reste du code des outils de gestion et d'interaction avec OAR ainsi que des interfaces web de visualisation de l'êtat du cluster. De plus, le couplage entre la base de données et le fait que OAR soit sans êtat permettent de facilement injecter des jobs dans la base pour que ceux-ci soient pris en compte par le scheduler et permet ainsi de réaliser aisement du rejeu de traces et des tests d'ordonnancement.

5 Retours d'expériences

5.1 CIMENT

CIMENT [3] (Calcul Intensif / Modélisation / Expérimentation Numérique et Technologique) est le mésocentre de calcul informatique de l'Université Joseph Fourier (Grenoble). Il réunit pratiquement toutes les disciplines de l'université qui ont des besoins en calcul haute performance: Chimie, Bioimagerie, Physique numérique, Environnement/Climatologie, Sciences de l'univers, Informatique distribuée,... Ce projet exploite aujourd'hui une dizaine de calculateurs représentant au total plus de 2000 coeurs de processeur.

Les histoires de OAR et de CIMENT sont étroitement liées et illustrées de collaborations fortes entre la recherche en informatique distribuée et les calculs de production à l'université de Grenoble. En effet, depuis le début des développements de OAR, les besoins des utilisateurs de CIMENT sont pris en compte et alimentent parfois des sujets de recherche intéressants,

principalement sur les problématiques d'ordonnancement. L'une des plus grandes collaborations concerne l'intergiciel de grille CiGri (cf 6.2) qui est très utilisé dans CIMENT. En effet, les calculateurs de CIMENT forment une grille légère opérée par cet outil qui est développé dans la même équipe que OAR. La grille de CIMENT a contribué à de nombreux calculs de production en donnant des résultats pour une vingtaines de projets de recherche scientifique dans différents domaines (physique, sciences de l'univers, biologie, informatique, ...). Outre ces usages, la grille de CIMENT est également une plateforme expérimentale pour la recherche en ordonnancement des applications de type « sac de tâches ».

Cette collaboration recherche/production dépasse maintenant le cadre de CIMENT. En effet, les utilisateurs de CIMENT peuvent exploiter, toujours grâce à la fonctionnalité best-effort de OAR, des machines habituellement réservées aux expérimentations (clusters Grid'5000 à Grenoble) ou des calculateurs de la communauté Lyonnaise (dans le cadre de collaborations régionales: projet CIRA [5])

Aujourd'hui, 9 clusters de CIMENT sur 11 sont équipés de OAR, ce qui fait de CIMENT l'un des deux plus gros utilisateurs de ce gestionnaire de ressources (l'autre étant Grid5000, cf chapitre suivant). Depuis 2006, plus de 200000 jobs standards et plus d'un million et demi de jobs best-effort (cf 4 et 6.2.) ont été éxécutés via OAR sur les machines de CIMENT. La plus grosse machine gérée par un serveur OAR dans CIMENT comporte près de 1000 coeurs sur 120 noeuds de calculs. Il s'agit en fait de 2 calculateurs ayant une administration commune: la possibilité « multi-cluster » de OAR est ici utilisée. CIMENT compte aussi des machines à mémoires partagées dont OAR s'accomode très bien.

Enfin, deux des derniers calculateurs acquis fonctionnent en mode « green computing », à savoir que lorsque les noeuds de calcul ne sont pas utilisés, ni par la grille, ni par des utilisateurs locaux (ce qui se produit parfois dans les périodes de congés), les noeuds de calcul sont automatiquement éteints par OAR afin d'économiser de l'énergie.

5.2 Grid'5000

Grid'5000 [2] est un instrument d'expérimentation scientifique pour l'étude des systèmes parallèles et distribués à grande échelle. Sa construction a démarré en 2003 suite au premier appel d'offre de l'ACI GRID [4]. L'objectif initial était de construire une infrastructure distribuée sur 10 sites avec un total de 5000 processeurs. Il a ensuite été recadré à 5000 coeurs de processeur , chiffre qui a été atteint pendant l'hiver 2008-2009.

Actuellement, Grid'5000 est composé de 9 sites en France et 1 au Brésil (Lille, Rennes, Orsay, Nancy, Bordeaux, Lyon, Grenoble, Toulouse, Sophia-Antipolis et Porto Alegre) et rassemble environ 5650 coeurs de processeur. 750 coeurs supplémentaires seront prochainement ajoutés dans Grid'5000 avec l'arrivée d'un nouveau cluster et l'intégration d'un site luxembourgeois. OAR est utilisé pour gérer l'ensemble des ressources des clusters Grid'5000.

La grande originalité de Grid'5000 est sa capacité de re-configuration logicielle. Un utilisateur peut allouer un ensemble de ressources et y télécharger si besoin l'ensemble de sa pile logicielle ; du système d'exploitation à l'application. Ces opérations sont rendues possibles par l'utilisation conjointe des outils Kadeploy [5] et OAR. En effet, OAR propose le type de job *deploy*, autorisant les utilisateurs à modifier ou ré-installer entièrement le système d'exploitation ainsi que les applications des machines de calcul avec Kadeploy. Dans le cas des jobs de type *deploy*, OAR cesse alors de vérifier le bon fonctionnement des machines allouées afin de ne pas les considérer comme suspectes ou défectueuses si leur état devient non standard au cluster.
A la fin des jobs *deploy*, l'épilogue de OAR ré-installe par le biais de Kadeploy le système ainsi que la configuration de base sur les noeuds modifiés et vérifie à nouveau périodiquement leur bon fonctionnement.

Une autre particularité de OAR qui est exploitée sur Grid'5000 est la possibilité d'exécuter des jobs en *best-effort*. Ce type permet d'autoriser des calculs provenant de grille de production comme CIMENT tout en laissant la priorité de la plate-forme pour les expériences scientifiques. C'est à dire que OAR terminera prématurément un job *best-effort* si un job « classique » est soumis alors que l'ensemble du cluster est déjà occupé ou si la ressource est explicitement demandée.

Afin de pouvoir effectuer des réservations sur plusieurs sites, un outils a été spécifiquement créé pour Grid'5000. Il s'agit de OARgrid, un outil permettant de gérer de manière centralisée les ressources de plusieurs serveurs OAR interconnectés entre eux. Il offre une interface en ligne de commande et propose désormais une API REST.

5.3 Cluster Virtuel de l'UFR IMAG

Une utilisation originale de OAR a été mise en oeuvre dans une UFR de l'université Joseph Fourier qui compte de nombreux postes de travail dans des salles de TP destinées aux enseignements. Il s'agit d'une implémentation de type « desktop computing », à savoir l'utilisation de ressources de type poste de travail pour effectuer des calculs lorsque ces derniers sont inutilisés (la nuit et le week-end dans le cas de l'UFR IMAG). La mise en oeuvre est rendue possible avec le logiciel **Computemode** (cf 6.4) qui permet de gérer un calendrier et des images de démarrage. Cette solution apporte aussi des ressources supplémentaires à la grille du projet CIMENT (cf 5.1) de l'Université de Grenoble qui voit le nombre de noeuds de calcul augmenter automatiquement dès que les machines de l'UFR IMAG apparaissent disponibles dans le mode dit « calcul ». Des optimisations ont été faites afin que les postes de travail ne soient pas allumés inutilement lorsqu'il n'y a pas de jobs, pour économiser l'énergie.

5.4 Autres usages connus

Outre ces usages, aujourd'hui, OAR semble de plus en plus utilisé. On a recensé des installations au Brésil, en Slovaquie, en Chine. Plus proches de nous, citons le BRGM à Orléans. Nous savons également que OAR est actuellement évalué par d'autres entités.

6 Outils connexes

6.1 TakTuk

TaktuK est un "lanceur parallèle" pouvant fonctionner à une très grande echelle. C'est un logiciel libre développé également au LIG (Laboratoire d'Informatique de Grenoble). Il décrit un arbre dans les ressources afin de lancer de façon efficace une commande sur un grand nombre de noeuds. OAR peut être configuré pour utiliser avantageusement cet outil pour toutes les opérations nécessitant d'accéder aux noeuds (vérification des noeuds, démarrage d'un job, nettoyage en fin de job, etc...).

Outre cette utilisation par OAR, Taktuk est un outil très puissant qui peut être utilisé pour toutes sortes de communications réseau lorsqu'il y a de nombreux noeuds. Il fonctionne avec un mécanisme de « vol de tache » lui permettant de s'adapter automatiquement à l'environnement. Il nécessite très peu de dépendances sur les systèmes et peut même s'autodéployer à la volée (aucune installation nécessaire sur les noeuds).

Enfin, en association avec l'outil Kanif (http://taktuk.gforge.inria.fr/kanif), TakTuk peut aider efficacement l'administrateur système d'un cluster dans les opérations de maintenance des noeuds.

Pour plus d'informations, visitez le site de taktuk: http://taktuk.gforge.inria.fr/

6.2 CiGri

CiGri est un logiciel libre, développé par les mêmes auteurs que OAR, permettant de constituer une grille de calcul dite "légère", exploitant les cycles cpu libres des grappes de calcul. Elle s'appuie sur le type de jobs "best-effort" que OAR propose. Avec CiGri, on peut optimiser la charge d'un ensemble de clusters et offrir de nombreuses ressources aux utilisateurs ayant des applications de type multi-paramétriques ou dites « sac de taches ».

CiGri est largement utilisé dans le projet CIMENT (cf 5.1) et a éxécuté plus de 3 millions de jobs depuis 2002 sur ce site. La charge des calculateurs d'un mésocentre est souvent faite de pics et de creux. L'utilisation de CiGri permet de lisser cette charge en extrayant localement les applications de type multi-paramétrique et en les diffusant à un niveau grille en best-effort.

Cette grille est quasi transparente vis à vis des utilisateurs locaux d'un calculateur donné puisque les jobs best-efforts sont d'une priorité nulle et immédiatement tués lorsque des jobs locaux ont besoin des ressources. CiGri se charge de re-soumettre automatiquement les jobs qui ont été tués. CiGri offre aussi un système de collecte automatique des résultats.

CiGri et OAR évoluent en parallèle. Certaines fonctionnalités de OAR sont parfois issues d'un besoin de CiGri, comme par exemple l'implémentation des tableaux de jobs (array jobs). Prochainement, CiGri tirera parti de la nouvelle API REST disponible dans les dernières versions de OAR, sécurisant, simplifiant et fiabilisant la couche de communication entre CiGri et OAR.

A noter que CiGri a récemment bénéficié d'un stage Google Summer Of Code 2009 au sein du projet OAR qui a permis de réécrire entièrement le mécanisme de scheduling.

6.3 Oargrid

Oargrid est un outil simple qui permet de soumettre des jobs simultanément sur plusieurs clusters exploités par OAR. Actuellement utilisé uniquement dans Grid'5000, il offre une API REST basée sur une librairie commune avec l'API de OAR. Ce n'est pas un intergiciel de grille, il n'y a aucune fonctionnalité d'ordonnancement dans Oargrid. Il s'agit d'un outil qui offre une vision globale d'un ensemble de clusters et qui permet de faire des réservations de ressources sur cet ensemble. Il exploite le mode « advanced reservation » de OAR.

6.4 ComputeMode

ComputeMode est un outil qui permet de créer très facilement un cluster virtuel composé de noeuds de calcul diskless (sans disque dur) à partir de PC de bureau se trouvant sur un même LAN. Typiquement, cela permet d'exploiter les PC de bureau d'un laboratoire ou les PC de salles de TP à des périodes où ils ne sont pas utilisés par leurs utilisateurs habituels (la nuit et les week-end par exemple). Un exemple d'utilisation est donné au chapitre 5.3. Signalons que cet outil est également utilisé au BRGM (un laboratoire de Géosciences basé à Orléans)

Un calendrier définit les périodes d'utilisation des machines et le mode dans lequel elles doivent fonctionner. Une image de démarrage correspond à chaque mode et les machines sont redémarrées sur l'image correspondante en fonction du calendrier. Dans le mode « calcul », Computemode fournit une image Linux fonctionnant en NFS-root (sans disque dur).

Dans la distribution standard de Computemode, fournie sous la forme d'une appliance virtuelle, un serveur OAR gère les ressources que Computemode a démarré en « mode calcul ». Computemode signale à OAR lorsqu'un nouveau noeud est opérationnel. Quant à lui, OAR propose une fonction qui permet de définir pour chaque ressource la durée pendant laquelle elle sera disponible. Cela permet un couplage avec le calendrier de computemode afin que les jobs ne soient lancés sur les ressources que si l'on sait qu'elles seront disponibles suffisament longtemps.

6.5 Kadeploy

Kadeploy est un système de déploiement rapide et évolutif conçu pour les clusters et les grilles de calcul. Il fournit un ensemble d'outils pour cloner, configurer et gérer un ensemble de noeuds. Actuellement, sur Grid'5000 il est utilisé pour déployer les environnements par défaut sur les noeuds ainsi que des systèmes Linux et * BSD personnalisés par les utilisateurs. OAR propose un couplage avec cet outil suivant deux points. Premièrement, il permet de gérer les droits de déploiement des noeuds alloués à une tâche. Et deuxièmement, les différents états du noeud relatifs aux phases de déploiements sont intégrés dans les cycles de gestion des ressources.

7 Conclusion

Les gestionnaires des ressources sont une pièce essentielle dans la gestion des infrastructures de calcul tel que les grappes et les grilles. Ces logiciels doivent continuellement s'adapter aux évolutions technologiques et aux usages. Dans cet article nous avons présenté OAR, qui est un gestionnaire de ressources moderne particulièrement polyvalent et personnalisable. En effet, il propose une large palette de fonctionnalité et, grâce à son architecture particulièrement souple, il est utilisé dans des contextes très variés comme des grilles de production de mésocentres, les grilles de recherche (comme Grid'5000) ainsi que dans des usages plus classiques (simple grappe). Les perspectives actuelles sont pour le court terme d'améliorer l'interfacage avec les intergiciels des couches supérieures via notamment des standards proposées par l'Open Grid Forum (organisme pour la promotion et la standardisation des grilles) et aussi de proposer une interface web plus complète. À plus long terme la question de la gestion de ressource à grande échelle reste posée et pourrait nécessiter des modifications importantes dans les structures de données représentant les ressources manipulées.

Finalement OAR est un projet très actif tant au niveau des développements que des partenariats en cours ou futurs. Nous pensons qu'il est un digne représentant des *Batch Schedulers* aptes à répondre aux besoins des utilisateurs et des administrateurs des infrastructures de calcul modernes.

Références

[1] Laboratoire d'Informatique de Grenoble: http://www.lig-lab.fr

[2] https://www.grid5000.fr

[3] https://ciment.ujf-grenoble.fr

[4] http://www-sop.inria.fr/aci/grid/public/acigrid.htm

[5] http://gforge.inria.fr/projects/kadeploy3/

[6] http://www.clusterresources.com/products/mwm/docs/pbsintegration.shtml

[7] http://en.wikipedia.org/wiki/Sun_Grid_Engine

[8] https://computing.llnl.gov/linux/slurm/

Pour en savoir plus sur OAR:
- Site web officiel: http://oar.imag.fr
- Wiki OAR: http://wiki-oar.imag.fr
- Fiche plume de OAR: http://www.projet-plume.org/fr/fiche/oar