Architecture distribuée pour la gestion des ressources dans des grilles à grande échelle

Emmanuel Jeanvoine[†⋆], Louis Rilling[⋆#], Christine Morin[⋆‡], Daniel Leprince[†]

[†]EDF R&D, [⋆]IRISA Paris Project Team, [#]Université de Rennes 1, [‡]INRIA
{Emmanuel.Jeanvoine,Louis.Rilling,Christine.Morin}@irisa.fr,
Daniel.Leprince@edfgdf.fr

RÉSUMÉ. *Les grilles de calcul permettent de fournir la puissance de calcul nécessaire aux simulations numériques mais demeurent complexes à utiliser. Vigne est un système qui permet de simplifier l'utilisation d'une grille de grande échelle composée de ressources hétérogènes et non fiables. Dans cet article, nous décrivons l'architecture générale de Vigne et nous présentons plus particulièrement les services de gestion d'application et d'allocation de ressources. Nous présentons des résultats, obtenus avec un prototype fonctionnel de Vigne lors d'expérimentations réalisées sur la plate-forme Grid'5000, qui montrent l'efficacité du service d'allocation de ressources.*

ABSTRACT. *Grid computing provides enough computation power to perform numerical simulations but it is complex to use. Vigne is a system that ease the use of a large scale grid composed with heterogeneous and unreliable resources. In this paper, we describe the general architecture of Vigne and we present more particularly the application management service and the resource allocation service. We present results, obtained with a working prototype of Vigne during experimentations realized on the Grid'5000 testbed, that show the efficiency of the resource allocation service.*

MOTS-CLÉS : *Grille de calcul, système distribué, réseaux logiques pair-à-pair, allocation de ressources, auto-réparation*

KEYWORDS: *Grid computing, distributed system, peer-to-peer overlay networks, resource allocation, self-healing*

1. Introduction

La simulation numérique occupe une place prépondérante dans de nombreux domaines scientifiques où l'expérimentation réelle est coûteuse, voire impossible. Pour réaliser ces simulations, d'importants moyens de calcul doivent être employés. Afin de répondre à ces besoins, les entreprises et institutions se tournent vers le regroupement de leurs moyens de calcul pour former des grilles de calcul.

Une grille de calcul peut contenir un nombre important de ressources, grappes de calculateurs ou stations de travail, qui sont hétérogènes, dispersées géographiquement et qui ne peuvent donner aucune garantie de fiabilité. Ces contraintes font qu'actuellement une grille demeure complexe à utiliser, tant par les administrateurs que par les utilisateurs.

Dans cet article, nous présentons l'architecture de Vigne, un système d'exploitation pour grille. La contribution principale de cette architecture est qu'elle fournit un ensemble de services système qui simplifient l'utilisation des ressources d'une grille. En particulier, notre architecture offre aux utilisateurs une utilisation transparente de l'ensemble des ressources de la grille sans qu'ils n'aient à se soucier de la localisation, de la disponibilité ou des défaillances des ressources.

L'article s'organise de la façon suivante. Dans la partie 2, nous présentons Vigne, notre système d'exploitation pour la gestion des ressources d'une grille. Nous énonçons nos objectifs, nos choix de conception et présentons l'architecture de Vigne. Dans les parties 3 et 4, nous détaillons les services de gestion d'applications et d'allocation de ressources. Nous présentons les résultats des expérimentations que nous avons réalisées avec notre prototype Vigne et la grille d'expérimentation Grid'5000 [CAP 05a] dans la partie 5. Avant de conclure, nous présentons des travaux apparentés à Vigne pour ce qui concerne la gestion des ressources d'une grille dans la partie 6.

2. Vue générale de Vigne

2.1. *Objectifs*

Nous présentons dans cette partie les objectifs que nous avons souhaité atteindre pour la conception du système Vigne.

L'objectif principal est la simplicité d'utilisation de la grille pour les utilisateurs. Afin d'occulter la complexité induite par le grand nombre de ressources, nous souhaitons offrir à l'utilisateur une vision de type *système à image unique*, vision qui permet de donner l'illusion à l'utilisateur d'utiliser une seule machine et ainsi de masquer la distribution de l'ensemble des machines. Ainsi, un utilisateur doit pouvoir soumettre une tâche sur la grille depuis n'importe quelle ressource, sans se soucier des ressources qui seront utilisées pour l'exécution de son application.

Les utilisateurs réalisant des simulations numériques étant souvent habitués à utiliser des grappes gérées par un gestionnaire de travaux, le mode de soumission des

tâches doit se rapprocher du mode de soumission utilisé sur un gestionnaire de travaux de type OpenPBS [OPE] ou LSF [LSF]. Dans ce mode de soumission, l'utilisateur fournit au système une description contenant des informations pour que la tâche puisse être exécutée. Ces informations peuvent concerner par exemple des pré-requis matériels tels qu'une taille d'espace disque ou un nombre de processeurs.

Dans une grille les ressources sont exploitées de manières hétérogènes. En effet, il peut y avoir des ressources exploitées en mode interactif telles que des stations de travail utilisant l'ordonnanceur Linux ou des grappes fonctionnant avec un système à image unique comme Kerrighed [MOR 04] par exemple. Des ressources peuvent aussi être exploitées en mode non interactif comme des grappes fonctionnant avec un gestionnaire de travaux. Vigne doit être capable d'exploiter ces différent types de ressources pour en rationaliser l'accès. De plus, nous souhaitons conserver les systèmes existants sur les grappes ou les stations de travail afin que leur exploitation sans utiliser Vigne reste possible.

En terme d'échelle, Vigne doit pouvoir être utilisé avec une grille de grande taille qui comporte plusieurs milliers de ressources dispersées sur plusieurs sites géographiquement éloignés. De plus, nous ne souhaitons pas considérer les ressources utilisées comme fiables et toujours présentes dans la grille. Pour des raisons d'administration, une ressource peut être retirée de la grille. Dans ce cas le départ peut être anticipé. Dans le cas où une ressource est victime d'une défaillance, le départ ne peut pas être anticipé. Il faut ainsi que le système pour grille continue à fonctionner malgré la disparition de la ressource. Pour minimiser les conséquences de ces défaillances, notre système doit posséder des propriétés d'auto-réparation. Nous traitons des propriétés d'auto-réparation de Vigne dans [JEA 05].

Du point de vue de l'administrateur, nous souhaitons qu'une ressource puisse être jointe ou retirée de la grille très simplement, que le système pour grille ne soit pas trop intrusif et qu'il ne requiert pas une installation compliquée.

2.2. *Choix de conception*

Afin de traiter la problématique de la grande échelle, nous avons opté pour une conception complètement distribuée. Ainsi, la défaillance d'une ressource quelconque n'entraîne pas la défaillance complète du système.

Nous détaillons dans [JEA 05] l'utilisation de réseaux logiques pair-à-pair comme fondation pour les services de haut niveau dans Vigne. La principale raison d'utilisation des réseaux logiques réside dans leurs propriétés d'auto-organisation, d'auto-réparation et de passage à l'échelle.

Actuellement, nous ne proposons pas de solution efficace pour les problèmes de sécurité et pour les communications devant s'établir à travers des serveurs mandataires ou des pare-feux.

2.3. *Architecture de Vigne*

La figure 1 schématise l'ensemble des services du système Vigne discutés dans cet article. D'autres services sont présentés dans [RIL 05]. Ces services peuvent se classer en trois catégories qui sont les services de bas niveau, les services de haut niveau et les services d'interface à l'utilisateur.

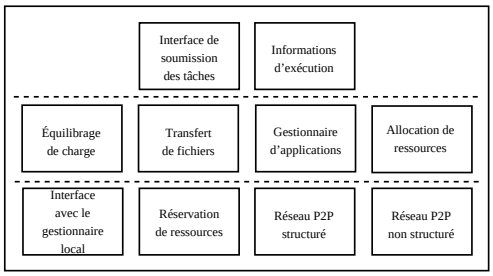

Figure 1. *Architecture du système Vigne*

2.3.1. *Services de bas niveau*

Dans Vigne, nous avons conçu un ensemble de services, dits de bas niveau. Comme dans un système d'exploitation classique, l'intérêt de ces services de bas niveau n'est pas de fournir directement des services à l'utilisateur mais plutôt de fournir des fonctionnalités pour les services de haut niveau.

Afin d'assurer les communications entre les différents nœuds de la grille et en dépit des défaillances pouvant intervenir, nous avons conçu deux services qui offrent des couches de communication à travers des réseaux logiques pair-à-pair.

Un premier service permet d'établir des communications à travers un réseau logique structuré. Les réseaux logiques structurés offrent des fonctionnalités de nommage et de localisation efficaces. Dans Vigne, nous avons mis en œuvre un réseau logique structuré fondé sur Pastry [ROW 01]. Ainsi, il est garanti que chaque nœud de la grille peut être identifié de manière unique et n'importe quel nœud peut contacter par routage n'importe quel autre nœud en un nombre limité de sauts. Avec Pastry, ce nombre de sauts est en moyenne $log_{16}(n)$ où n est le nombre de nœuds dans le réseau.

Un second service permet d'établir des communications à travers un réseau logique non structuré. Ce service permet de localiser des nœuds dans le système à partir d'une description fondée sur un ensemble d'attributs. Contrairement au service de réseau structuré, ce service n'offre pas la garantie de trouver un nœud correspondant à une description. La recherche dans un réseau non structuré s'effectue par inondation ou par des variantes d'inondation d'une requête dans le réseau. Dans Vigne, ce service est fondé sur le protocole Scamp [GAN 03].

Un troisième service est chargé de faire l'interface avec le gestionnaire local de chaque ressource. Nous avons évoqué dans la partie 2.1 que les ressources pouvaient être exploitées avec différent gestionnaires de ressources locaux. Le service d'inter-

face avec le gestionnaire local fournit une interface générique qui est spécialisée pour chaque type de gestionnaire de ressources.

Enfin, nous avons conçu un service de réservation de ressources. Ce service a pour but de gérer la concurrence d'accès à une ressource qui serait rare. Nous n'avons pas encore mis en œuvre ce service dans Vigne.

2.3.2. *Services de haut niveau*

Nous avons conçu quatre services de haut niveau, fondés en partie sur les services présentés dans la partie 2.3.1.

Un premier service appelé gestionnaire d'applications est chargé de fournir aux applications le moyen de s'exécuter de façon fiable. Nous revenons plus en détail sur ce service dans la partie 3.

Un second service, appelé allocateur de ressources, est chargé de découvrir et d'allouer des ressources libres de la grille aux applications. Ce service est détaillé dans la partie 4

Vigne possède aussi un service de transfert de fichiers entre nœuds qui permet de transférer tous les éléments nécessaires à l'exécution distante d'une application. Ce service permet à l'utilisateur de ne pas se soucier du déploiement de fichiers binaires de l'application et des fichiers d'entrée ou de sortie.

Nous avons aussi conçu un service d'équilibrage de charge. Ce service est le pendant dynamique de l'allocateur de ressources. En effet, le temps d'exécution d'une tâche est souvent imprévisible. Ainsi une ressource non choisie par l'allocateur de ressources puisqu'elle était occupée peut se retrouver libre après un court instant. L'objectif est de corriger les déséquilibres de charge qui peuvent se produire dans ces cas-là. Notre approche d'équilibrage de charge est fondée sur la migration de points de reprise d'applications. Étant donné que nous ne souhaitons pas modifier les applications des utilisateurs, cette fonctionnalité impose le support du système sous-jacent comme Kerrighed [MOR 04] par exemple. Nous n'avons pas encore mis en œuvre cette fonctionnalité dans Vigne.

2.3.3. *Services pour l'utilisateur*

Nous présentons maintenant deux services qui sont chargés de faire l'interface entre l'utilisateur et le système.

Le service de soumission de tâches permet à l'utilisateur de soumettre ses applications en fournissant une description comportant trois parties. La première partie comporte les chemins vers les binaires et les fichiers d'entrée/sortie de l'application. La seconde partie donne une description des éléments matériels et logiciels requis pour l'exécution de l'application. La troisième partie comporte des éléments optionnels qui améliorent le fonctionnement de l'application mais qui ne sont pas critiques. Lors d'une soumission, il faut fournir le fichier de description de la tâche et l'adresse d'une machine (quelconque) de la grille qui exécute le système Vigne.

Le service d'information d'exécution peut renseigner l'utilisateur sur l'état de son application. Il n'est pas possible de connaître l'état d'avancement d'une application sans en avoir instrumenté le code. Cependant le service est capable d'indiquer si une application a été exécutée sur une ressource, si une application a terminé son exécution ou encore si la ressource hébergeant l'application a subi une défaillance.

3. Gestionnaire d'application

Le service de gestion d'application a un rôle primordial dans Vigne. Ce service est responsable du cycle de vie complet d'une application. Lorsqu'une application est soumise dans Vigne, un gestionnaire d'application est automatiquement créé Ce gestionnaire possède un identifiant unique qui est obtenu en calculant le hachage SHA-1 du triplet *{MD5 du binaire de l'application, heure de soumission, nom de la machine de soumission}*. Pour des raisons d'équilibrage de charge, le gestionnaire d'application n'est pas créé sur la ressource où la soumission de la tâche a été effectuée. Nous utilisons ici le service de communication sur réseau logique structuré présenté dans le paragraphe 2.3.1. Ce service fournit un mécanisme de nommage fondé sur des clés pour tous les éléments du système. Dans notre implémentation, l'espace de nommage est de 256 bits. Le gestionnaire d'application est créé sur la ressource dont la clé est la plus proche numériquement de son identifiant unique. La propriété d'équilibrage de charge est assurée par les propriétés de SHA-1.

Une fois créé, le gestionnaire initie la découverte et l'allocation de ressources, transfère les fichiers de l'application sur la ressource choisie et surveille régulièrement l'état de l'application exécutée. Lorsque l'exécution est terminée, le gestionnaire d'application récupère les résultats produits dans un fichier de sortie et supprime les éventuels fichiers et processus créés sur la ressource qui a été utilisée. Le gestionnaire d'application n'est pas détruit tant que l'utilisateur n'a pas récupéré les résultats.

En cas de défaillance de la ressource choisie à l'issue de l'allocation de ressource, plusieurs politiques peuvent être employées. Par exemple, si la ressource possède un système permettant de sauvegarder des points de reprise, le gestionnaire d'application redémarre automatiquement l'application depuis son dernier point de reprise. Actuellement, seul le redémarrage automatique depuis le début de l'application est mis en œuvre dans Vigne.

Le gestionnaire d'application étant un point crucial dans la vie d'une application, des politiques de duplication peuvent être employées pour tolérer la défaillance d'une ressource hébergeant des gestionnaires d'application. Le réseau logique structuré fournit via le mécanisme de table de hachage distribuée des facilités pour effectuer de telles duplications. La duplication des gestionnaires d'application n'est toutefois pas encore mise en œuvre dans Vigne.

4. Allocateur de ressources

Afin de fournir à l'utilisateur une vision de type *système à image unique*, nous avons conçu un service permettant de trouver automatiquement des ressources convenant à la description fournie par l'utilisateur et disponibles dans la grille.

Le service d'allocation de ressources a deux rôles principaux qui sont la découverte des ressources et la sélection d'une ressource optimale parmi celles qui ont été trouvées et en fonction d'une politique donnée.

4.1. *Découverte de ressources*

Comme nous l'avons évoqué dans la partie 2.2, nous avons choisi d'utiliser une architecture complètement distribuée. La découverte de ressources repose sur le service de communication dans un réseau non structuré. Une découverte de ressources correspond à la propagation dans le réseau non structuré d'une partie de la description fournie par l'utilisateur à la soumission d'une tâche. Actuellement, la propagation de requêtes dans Vigne peut s'effectuer par inondation ou par marches aléatoires ; d'autres mécanismes sont en cours de développement.

À la suite de la propagation de la requête, les ressources convenant à l'exécution de la tâche répondent au gestionnaire d'application concerné. Étant donné la nature de la recherche, il est possible qu'aucune ressource ne convienne à la description ou que des ressources existent mais n'ont pas été trouvées. Dans ce cas, le gestionnaire d'application peut initier plusieurs découvertes de ressources successives pour trouver une ressource convenable.

4.2. *Sélection de ressources*

Une fois que la découverte de ressources est terminée et dans le cas où des ressources ont été trouvées, l'allocateur de ressources doit sélectionner la ressource la plus adaptée parmi celles trouvées. Plusieurs politiques peuvent être envisagées pour ce choix. Par exemple, il est possible de sélectionner la ressource la moins chargée, ou alors la ressource qui se situe le plus près d'un serveur de fichiers donné pour minimiser les coûts de transfert des fichiers d'entrée et de sortie. Actuellement, seule la politique du choix de la ressource la moins chargée a été mise en œuvre dans Vigne.

5. Évaluation

5.1. *État du prototype*

Afin d'évaluer le système Vigne, nous avons réalisé un prototype. Actuellement, ce prototype est fonctionnel, sans toutefois posséder toutes les fonctionnalités décrites dans la partie 2.3. Par rapport à nos objectifs initiaux, le prototype offre une vue de type *système à image unique* pour ce qui concerne la gestion d'un grand nombre de ressources de calcul qui peuvent être géographiquement éloignées. La soumission d'une tâche dans Vigne est comparable à la soumission d'une tâche dans un gestionnaire de travaux.

L'infrastructure de communication du prototype possède des propriétés d'auto-réparation et d'auto-organisation. Cela permet au système de survivre à des défaillances simultanées. Ces propriétés ont été évaluées dans [JEA 05].

5.2. *Utilisation du prototype*

L'implémentation du système Vigne se découpe en deux parties qui sont un démon et un client. Le démon doit être lancé sur chaque nœud de la grille, que ce soit le frontal

```
<Application>
  <Job_desc>
    <Exec>~ejeanvoi/app_bin</Exec>
    <Out>~ejeanvoi/output_file</Out>
  </Job_desc>
  <Required>
    <Arch>X86_64</Arch>
    <OS>LINUX</OS>
    <OS_Version>2.6.11+</OS_Version>
    <Net>Infiniband</Net>
    <Mem>2048</Mem>
    <Nodes>6</Nodes>
    <Libs>
      <Mpi>LAM_MPI</Mpi>
    </Libs>
  </Required>
</Application>
```

Figure 2. *Exemple de fichier de description d'une application*

d'une grappe exploitée en mode différé avec un gestionnaire de travaux, une machine quelconque d'une grappe exploitée en mode interactif avec un *système à image unique* ou une station de travail exploitée avec Linux. Pour lancer un démon, il faut lui fournir l'adresse de n'importe quel nœud du système (i.e. l'adresse d'un autre démon) pour qu'il puisse se joindre au système, sauf dans le cas du premier démon lancé.

Le client est ensuite utilisé pour effectuer toutes les interactions avec le système telles que : une requête de soumission de tâche, une requête d'interrogation d'état d'avancement ou encore une suppression de tâche. Voici un exemple de soumission de tâche :

```
./VigneClient -n ANY_NODE_RUNNING_VIGNE -t SUBMIT_TASK \
                                        -f TASK_DESCRIPTION_FILE
```

L'information `ANY_NODE_RUNNING_VIGNE` permet de spécifier l'adresse du nœud sur lequel la requête sera lancée. Une requête pouvant être effectuée depuis n'importe quel nœud de la grille, l'utilisateur a juste besoin de connaître un seul nœud pour lancer ses requêtes. L'information `SUBMIT_TASK` correspond au type de requête que l'on souhaite effectuer, ici une soumission de tâche. Finalement, l'information `TASK_DESCRIPTION_FILE` correspond au chemin vers un fichier comprenant une description de l'application qui doit être fournie par l'utilisateur.

La figure 2 présente un exemple de contenu d'un fichier pointé par l'information `TASK_DESCRIPTION_FILE` dans la requête de soumission. Ce fichier présente deux parties qui sont d'une part les chemins vers le binaire de l'application et les fichiers d'entrée/sortie, et d'autre part les éléments matériels et logiciels requis pour faire fonctionner l'application.

5.3. *Résultats préliminaires*

Nous présentons dans ce paragraphe les résultats des expérimentations réalisées avec le système Vigne sur la grille d'expérimentation nationale Grid'5000 [CAP 05a]. Nous avons utilisé 472 nœuds, soit 944 processeurs, répartis sur les sites de Bordeaux, Orsay, Rennes et Sophia-Antipolis. Les nœuds sont équipés de bi-processeurs Xeon à 2.4 GHz ou d'Opteron à 2 ou 2,2 GHz. Le système Vigne a été déployé sur tous les nœuds utilisés. Nous avons lancé 944 fois l'exécution d'un processus, afin que dans l'idéal chaque processeur de la grille soit utilisé. Le processus utilisé est un processus séquentiel qui modélise une application de calcul intensif et qui génère une charge processeur élevée.

Site Grid'5000	Nombre de processeurs	Temps d'exécution moyen
Bordeaux	82	1740 s
Orsay	344	1959 s
Rennes (grappe Paraci)	98	1594 s
Rennes (grappe Parasol)	62	2689 s
Rennes (grappe Paravent)	198	2062 s
Sophia	160	1905 s
Total	944	1962 s

Tableau 1. *Répartition des nœuds par site et temps d'exécution du processus séquentiel*

Le tableau 1 montre la répartition des nœuds par site ainsi que le temps d'exécution sur chaque site du processus séquentiel que nous avons utilisé.

Nous avons cherché à évaluer l'efficacité du service d'allocation de ressources dans Vigne. Pour cela nous avons mesuré le temps moyen d'exécution du processus séquentiel. Si chacun des 944 processus était lancé simultanément sur 944 processeurs distincts, le temps moyen d'exécution d'un processus serait 1962 secondes (cf. tableau 1). Cela correspond au temps d'exécution sur un processeur dont la puissance serait égale à la moyenne pondérée de toutes les puissances des processeurs de notre expérience.

Pour réaliser la mesure de l'efficacité du service d'allocation de ressources de Vigne, nous avons effectué cinq expériences dans lesquelles les soumissions des tâches ont été espacées de 1, 2, 5, 10 ou 20 secondes. Nous n'avons pas cherché à évaluer le temps global d'exécution des 944 processus mais le temps moyen d'exécution d'un processus. Ainsi, nous évaluons la capacité du service d'allocation de ressources a trouver les ressources les moins chargées pour placer les nouvelles tâches soumises.

Intervalle de soumission	1 s	2 s	5 s	10 s	20 s
Temps d'exécution moyen	2248 s	2143 s	2057 s	2042 s	2039 s
Surcoût	14,6%	9,2%	4,8%	4%	3,9%

Tableau 2. *Temps moyen d'exécution selon l'intervalle de soumission*

Le tableau 2 montre le temps d'exécution moyen pour chacune des expériences ainsi que le surcoût d'utilisation de notre prototype par rapport au temps d'exécution moyen de 1962 secondes. Nous pouvons remarquer que dans ces résultats, plus l'intervalle de soumission est élevé, plus le temps d'exécution est court. Nous expliquons cela par le fait qu'un court intervalle de soumission ne permet pas à l'allocateur de Vigne d'effectuer une mesure très précise de la charge des ressources utilisées. Ainsi, plusieurs processus peuvent être exécutés sur un seul processeur, ce qui allonge le temps d'exécution. Le surcoût d'exécution est toutefois relativement faible compte-tenu du gain en automatisation et en fiabilité obtenu. Ce surcoût pourra être réduit lorsque le service d'équilibrage de charge sera fonctionnel, puisque les déséquilibres pourront être corrigés après l'allocation initiale. De plus, le temps mesuré est supérieur au temps réel d'exécution. En effet, la détection de la fin de l'exécution d'un processus peut prendre jusqu'à 60 secondes.

6. Travaux apparentés

Globus [FOS 97] fournit un mécanisme de découverte de ressources, le *Monitoring and Discovery System* (MDS). Le MDS est fondé sur un ensemble de documents XML dans lequel il est possible de faire des requêtes complexes avec le langage Xpath. Les informations sur les ressources sont stockées dans le MDS et rafraîchies périodiquement. Les problèmes de passage à l'échelle sont résolus par une architecture hiérarchique du MDS. Toutefois, ce sont les administrateurs qui doivent déterminer de quel MDS de plus haut niveau dépendent leurs MDS. De plus, l'inconvénient du solution hiérarchique est que si un nœud de l'arbre est perdu, toute la sous-branche est perdue. De part sa conception totalement distribuée, l'architecture que nous proposons ne souffre pas de ce problème.

Condor [LIT 88] est un projet pionnier dans le domaine des gestionnaires de ressources pour grille. L'objectif était d'étendre le concept de gestionnaire de travaux pour le rendre distribué et ainsi pour pouvoir utiliser un grand nombre de machines. Condor-G [FRE 01] est fondé sur Globus et a pour objectif de fédérer des ressources de calcul de différent sites. Condor-G permet à différents domaines d'administration de la grille d'être utilisés simultanément puisqu'il utilise les mécanismes de sécurités de Globus. Condor-G ne met pas en œuvre de mécanisme de découverte de ressource. Le choix des ressources est fait dans une liste de serveurs *Grid Resource and Allocation Management* (GRAM) de Globus qui doit être fournie par les utilisateurs. XtremWeb [CAP 05b], comme Condor, est fondé sur un modèle où les nœuds de la grille sont soit clients, soit coordinateurs, soit travailleurs. Lorsque des travailleurs se joignent au réseau, ils se déclarent auprès d'un coordinateur. De cette façon la décou-

verte de ressources est simplifiée et très efficace. Dès qu'un client souhaite exécuter une tâche, il contacte un coordinateur qui choisit un travailleur libre convenant aux besoins de l'application. Toutefois, dans XtremWeb comme dans Condor, les coordinateurs reposent sur des nœuds stables du systèmes, ce que nous avons souhaité éviter dans la conception de notre architecture. En revanche, XtremWeb propose des solutions pour la sécurité et le passage de pare-feux.

Active Yellow Pages (ActYP) [ROY 01] est un projet dont l'objectif principal est de fournir un allocateur de ressources réactif dans le cas où beaucoup de requêtes similaires sont soumises. ActYP est fondé sur un ensemble de bases de données contenant une description des ressources du système. ActYP propose un système d'agrégation de ressources similaires afin d'optimiser le temps de réponse de certaines requêtes. Étant donné que les base de données peuvent être distribuées, le système possède de bonnes propriétés de passage à l'échelle. Connaissant la volatilité des ressources que nous considérons, nous avons souhaité que toutes les ressources aient le même rôle et donc nous avons exclu les approches de type base de données. Toutefois le concept d'agrégation pourrait être bénéfique à un système comme Vigne, dans le cas notamment des applications de type simulation paramétriques.

7. Conclusion

Dans cet article, nous avons décrit l'architecture de Vigne, un système auto-réparent pour la gestion des ressources d'une grille de grande échelle. Pour l'utilisateur, le système Vigne offre une interface simple de soumission des tâches, similaire à celle offerte par les gestionnaires de travaux. Vigne est conçu pour tolérer de multiples défaillances simultanées. Pour cela, il s'appuie sur un réseau pair-à-pair. Le gestionnaire d'application permet d'assurer l'exécution fiable des applications en présence de défaillances. De plus, nous avons mis en œuvre un prototype fonctionnel de Vigne qui a été expérimenté sur la grille de test Grid'5000 [CAP 05a]. Les expérimentations réalisées nous ont permis de valider l'architecture de Vigne en évaluant l'efficacité du service d'allocation de ressources dans une grille de grande taille. De futurs travaux seront consacrés à la duplication du gestionnaire d'applications et à la conception du service d'équilibrage de charge.

8. Bibliographie

[CAP 05a] CAPPELLO F., DESPREZ F., DAYDE M., JEANNOT E., JEGOU Y., LANTERI S., MELAB N., NAMYST R., PRIMET P., RICHARD O., CARON E., LEDUC J., MORNET G., « Grid'5000 : A Large Scale, Reconfigurable, Controlable and Monitorable Grid Platform », *Grid2005 6th IEEE/ACM International Workshop on Grid Computing*, 2005.

[CAP 05b] CAPPELLO F., DJILALI S., FEDAK G., HERAULT T., MAGNIETTE F., NÉRI V., LODYGENSKY O., « Computing on large-scale distributed systems : XtremWeb architecture, programming models, security, tests and convergence with grid », *Future Generation Computer Systems*, vol. 21, n° 3, 2005, p. 417–437.

[FOS 97] FOSTER I., KESSELMAN C., « Globus : A Metacomputing Infrastructure Toolkit », *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, n° 2, 1997, p. 115–128.

[FRE 01] FREY J., TANNENBAUM T., LIVNY M., FOSTER I., TUECKE S., « Condor-G : A Computation Management Agent for Multi-Institutional Grids », *Proc. of HPDC 2001*, San Francisco, CA, USA, août 2001, p. 55–63.

[GAN 03] GANESH A. J., KERMARREC A.-M., MASSOULIÉ L., « Peer-to-Peer membership management for gossip-based protocols », *IEEE Trans. on Computers*, vol. 52, n° 2, 2003.

[JEA 05] JEANVOINE E., RILLING L., MORIN C., LEPRINCE D., « Using Overlay Networks to Build Operating System Services for Large Scale Grids », Research Report n° INRIA-RR-5776, décembre 2005, INRIA, IRISA, Rennes, France.

[LIT 88] LITZKOW M. J., LIVNY M., MUTKA M. W., « Condor - A Hunter of Idle Workstations », *Proc. of ICDCS 1988*, Washington, DC, 1988, p. 104–111.

[LSF] « LSF », Web Page : http ://www.platform.com.

[MOR 04] MORIN C., GALLARD P., LOTTIAUX R., VALLÉE G., « Towards an Efficient Single System Image Cluster Operating System », *Future Generation Computer Systems*, vol. 20, n° 2, 2004, p. 505–521.

[OPE] « OpenPBS », Web Page : http ://www.openpbs.org.

[RIL 05] RILLING L., « Système d'exploitation à image unique pour une grille de composition dynamique : conception et mise en œuvre de services fiables pour exécuter les applications distribuées partageant des données », Thèse de doctorat, Université de Rennes 1, IRISA, Rennes, France, novembre 2005.

[ROW 01] ROWSTRON A. I. T., DRUSCHEL P., « Pastry : Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. », *Proc. of Middleware 2001*, nov 2001, p. 329–350.

[ROY 01] ROYO D., KAPADIA N. H., FORTES J. A. B., DE CERIO L. D., « Active Yellow Pages : A Pipelined Resource Management Architecture for Wide-Area Network Computing », *Proc. of HPDC 2001*, Washington, DC, USA, 2001, p. 147–157.