Configuration automatique d'un cluster de calcul avec Puppet

Riwan Blondé, André Dimitri et Quentin Dexheimer1

21 février 2012

Table des matières

Chapitre 1

Ennoncé du Projet Tuteuré

1.1 Le Sujet

L'énoncé tel que défini est le suivant : l'équipe devait réaliser une série de modules Puppet permettant de configurer un cluster de calcul de manière automatisée

1.2 Le Travail Attendu

Le travail attendu consiste en un guide d'installation et d'utilisation de l'outil utilisé, des scripts permettant l'installation des modules puppet et de lancement de la configuration, plus une démonstration en temps réel sur la plate-forme Grid5000.

1.3 Constitution d'un site Grid5000

Un site Grid5000 est constitué de plusieurs services : s'y trouvent un DNS, un serveur DHCP, une base de données MySQL ainsi que les outils de déploiement et de réservation OAR et kadeploy. S'y ajoutent également le service NFS, qui permet de stocker les dossier personnels d'utilisateurs. Au niveau physique, un site géographique Grid5000 est représenté par une machine de frontend, plusieurs noeuds et les machines qui serviront pour le stockage NFS.

Chapitre 2

La plate-forme Grid5000

2.1 Objectif de la création

L'objectif de la création de Grid5000 est d'atteindre 5000 processeurs sur les différents sites. Cela permettra d'avoir les moyens d'effectuer des recherches dans le domaine des grilles. Il sera ainsi possible d'effectuer des tests réels plutôt que de ne présenter que des résultats de simulation.

2.2 Organisation

Grid5000 a vu le jour en 2003 suite au besoin exprimé par l'Etat Français de disposer d'une grille informatique destinée à la recherche.

Le projet Grid5000 est réparti sur différents sites qui sont interconnectés avec le réseau Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche (RENATER).

Les sites sont composés de un ou plusieurs clusters, comme le site de Nancy avec Griffon et Grelon. Chaque site est géré par son propre administrateur, ce qui entraîne une hétérogénéité de configuration sur les différents sites. Par exemple, les frontends des différents sites qui ne disposent pas des mêmes librairies.

A l'heure actuelle, la plate-forme Grid5000 comporte au total 7244 coeurs,disponibles sur 1500 machines réparties sur les différents sites comme suit : - Bordeaux : 154 machines - Grenoble : 118 machines - Lille : 100 machines - Lyon : 135 machines - Nancy : 92 machines - Orsay : 340 machines - Reims : 44 machines - Rennes : 162 machines - Sophia : 101 machines - toulouse : 80 machines

2.3 Les outils du Grid5000

2.3.1 OAR

OAR est une collection d'outils développée par le laboratoire IMAG1 de l'Institut National de Recherche en Informatique et Automatique (INRIA) permettant de gérer un cluster de machines. Il fonctionne avec Perl à l'aide de MySQL et prend en charge le mécanisme de réservation ainsi que l'utilisation massivement simultanée des machines en cluster. Grid5000 fait partie des cinq partenaires de OAR.

Lorsqu'un utilisateur souhaite accéder au projet depuis l'extérieur, il doit passer par une machine pare-feu d'un des neufs sites de France, parmi celles qui acceptent ce genre de connexion. Une fois ce pare-feu passé grâce à l'identification par clé SSH, c'est un frontal qui s'offre à l'utilisateur. Celui-ci est pourvu d'un espace qui lui est dédié.

Cet espace seretrouvera partout où l'utilisateur ira grâce à un montage Network File System (NFS), dans n'importe quelle machine de n'importe quel cluster, tant qu'il ne quitte pas le site. En effet, les espaces personnels ne sont pas encore partagés d'un site à l'autre, ce qui pose divers problèmes d'harmonisation et de réplication des données. Sur ce frontal, les possibilités sont très limitées et se cantonnent

surtout à utiliser les outils OAR.

La première chose que ces outils lui permettront de faire, c'est de réserver des nœuds. Dans le jargon des clusters, cela signifie réserver un ensemble de machines physiques, pour pouvoir exécuter des actions dessus et de façon parallèle. Une réservation se fait avec son nom d'utilisateur et pour un temps donné.

Il est également possible de préciser le nombre de nœuds ou de processeurs souhaités. Par le biais de filtres SQL, d'autres options sont offertes : réservation sur un cluster en particulier, sélection de nœuds possédant un matériel spécifique, etc. Si la requête trouve les machines correspondantes et que celles-ci sont libres dans l'intervalle de temps demandé, l'utilisateur est automatiquement connecté sur la première des machines de sa réservation. . L'environnement proposé par défaut est appelé l'environnement de développement. Celui- ci lui permettra d'exécuter des programmes qui concerneront l'ensemble des machines de sa réservation, auxquelles il peut accéder sans limite par SSH. Le principal intérêt de cet environnement est d'exécuter des programmes MPI. Nous verrons cette norme par la suite, mais retenez que c'est un moyen de faire travailler toutes les machines de la réservation en les faisant communiquer facilement.

Cet environnement de développement, différent sur chaque site et parfois même d'un cluster a l'autre, est stocké sur une partition réservée sur chacun des nœuds. L'espace de l'utilisateur est monté en NFS sur les nœuds réservés.

2.3.2 Kadeploy

Le déploiement consiste à installer son propre système d'exploitation sur tous les nœuds de la réservation. Dans ce cas, la réservation doit être prise en indiquant ses intentions lors de l'appel de la commande. Des systèmes préparés sont ensuite disponibles dans la base de données des sites (qui diffère d'un site à l'autre).

Il est possible d'installer un de ces systèmes, de le copier-coller en ajoutant ou en supprimant des programmes, des données, et de compresser le tout pour en faire un environnement personnalisé, qui pourra être déployé sur une future réservation. Lorsque l'utilisateur réserve des nœuds pour un déploiement, il est redirigé vers un environnement spécial de déploiement sur le frontal. Il pourra alors demander le déploiement d'un système sur toutes les machines, qui redémarreront sur une partition différente après avoir effectué l'installation du système. Ce déploiement est géré principalement par kadeploy3, un outil également développé par l'INRIA.

Pour constater le redémarrage de la machine et éventuellement détecter les problèmes qui peuvent survenir à cause de l'environnement, il est possible de visualiser directement la séquence de lancement du BIOS, avant même que le noyau Linux ne soit démarré. Kaconsole3 permet ceci, grâce à un contrôleur matériel et l'outil telnet.

Chapitre 3

Le logiciel Puppet

3.1 Qu'est ce que Puppet Labs ?

uppet est un logiciel open source comportant les outils nécessaires à la configuration de systèmes informatiques. Il est basé sur le langage de programmation Ruby , et est sous licence GPL v2. Puppet a principalement été développé par Luke Kanies et son entreprise Puppet Labs.

Kanies a développé puppet grâce à son expérience dans les systèmes Unix et les systèmes d'administration depuis 1997. Non satisfait des outils de configuration existants, il a commencé à travailler avec des outils de développement en 2001, et a fondé Puppet Labs en 2005, une entreprise de développement open source basée sur les outils d'automatisation. Peu de temps après, Puppet Labs sort son nouveau produit phare : Puppet. Il peut être utilisé pour gérer la configuration d'application sous Unix et OSX, mais aussi de Linux et Windows depuis peu de temps.

Son modèle est basé sur 3 piliers :

1. Le déploiement ,
2. langage de configuration et une couche d'abstraction ,
3. couche transactionnelle .

3.2 Le Contexte

La gestion au quotidien des configurations systèmes et applicatives d'une entreprise représente un travail très fastidieux. Puppet simplifie grandement la vie des administrateurs : plus de contrôles et d'interventions à réaliser régulièrement. Puppet se charge d'imposer sur les machines des utilisateurs les configurations « type » définies par l'administrateur. Puppet est un outil de déploiement et de gestion centralisée de configurations pour les environnements Linux, Unix et Windows ; les machines gérées pouvant être physiques ou virtualisées.

3.3 Fonction principale

Puppet est un outil de déploiement et de gestion automatisés de configurations et de systèmes informatiques (serveurs, postes de travail..). Il repose sur un modèle client/serveur : un serveur central sert de dépôt de configurations, les systèmes clients (nœuds) se mettant à jour de manière manuelle ou automatique.

Avec Puppet, l'administrateur n'écrit pas un ensemble d'opérations à exécuter sur les différents nœuds : sous la forme d'un script, l'administrateur décrit l'état final de la machine dans un Manifest, ce qui l'affranchit de la connaissance des commandes propres à chaque système d'exploitation pour arriver à cet état.

Le client Puppet peut être exécuté à chaque fois, mais les changements ne seront effectifs que si l'état de la machine ne correspond pas aux attentes de l'utilisateur.

Chapitre 4

Le travail réalisé

4.1 Choix techniques

Il a été décidé, d'après ce que nous dit l'énoncé, d'utiliser le logiciel de gestion centralisé Puppet. Il existe deux raisons à cela : tout d'abord, nous connaissions le logiciel (par le biais d'exposés techniques), d'autre part par sa grande simplicité de mise en oeuvre par le biais de modules à rajouter.

De plus, comme vous le verrez plus bas, les scripts ont été rédigés en langage Bash et non Ruby, dans la mesure où nous n'avions pas encore les connaissances (et les cours) en début de projet pour réaliser les scripts dans ce langage. Bash était également le langage le plus utilisé entre les membres du projet.

Nous avions comme contraintes d'installer les services cités en partie 1b. , ce qui impliquait notamment de prendre en main les outils proposés commme OAR et Kadeploy. Deux choix se sont offerts à nous concernant la gestion des ressources. Nous avons préféré ne pas utiliser d'image de nos travaux, qui nous aurait simplifié la tâche mais qui n'était pas forcément dans l'esprit du projet. Cela nous a permis également de repartir de machines fraîchement installées et de pouvoir tester nos scripts sans avoir à désinstaller ce qui avait été déjà fait avant tout test.

4.2 État actuel des travaux.

4.2.1 Historique

Nous avons appris la première semaine à utiliser la plate-forme Grid5000 par le biais de tutoriels.

La deuxième semaine, nous avons pris en main Puppet pour une partie du groupe. Nous avons réussi à installer manuellement l'intégralité d'une architecture Puppet (Master Puppet + 2 clients) sur 3 noeuds réservés manuellement. En parallèle, un script de réservation de machines, d'installation du maître Puppet et de l'installation des clients Puppet a été écrit. Pour permettre d'utiliser Puppet, nous avons écrit des modules Puppet pour l'installation de la base de données MySQL et d'un serveur D.N.S.

La troisième semaine, nous avons continué la création de modules Puppet avec l'écriture d'un module pour l'installation du N.F.S. . Nous avons également pris en main l'outil de gestion de réseaux virtuels kavlan, et terminé les scripts automatiques de réservation des noeuds (reserv.sh), ainsi que l'installation et configuration de Puppet sur le serveur Maître Puppet et sur les clients Puppet.

La quatrième semaine nous a permis d'inclure le module MySQL et de rédiger le rapport de mi-projet.

4.2.2 Scripts réalisés

Reserv.sh

Le premier script utilise l'architecture et les outils mis à disposition par la plate-forme Grid5000 pour déployer un ensemble de noeuds sous distribution Debian.

```
#!/bin/bash

#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#Formulaire
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
echo -n "Nom de la réservation : "
read nom
echo -n "Nombre de machine à réserver : "
read nbr_nodes
echo -n "Temps de la réservation (h:mm:ss) : "
read tmps

#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#Réservation des machines
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
echo "Réservation de "$nbr_nodes" machine(s) pour "$tmps" heure(s)."
oarsub -I -t deploy -l {"type='kavlan-local'"}/vlan=1+/nodes=$nbr_nodes,walltime=$tmps -n "$nom"
```

FIGURE 4.1 – reserv

Nous avons décidé de simplifier la réservation en proposant un outil plus simple à prendre en main dans la mesure où l'utilisateur n'a pas besoin de chercher dans les pages de manuel pour apprendre la commande.

Install.sh

Intéressons nous maintenant au script le plus important, celui d'installation de l'ensemble des noeuds réservés.

```


#Définition des chemins vers les fichiers exploités.
##Chemin du fichier liste des nodes réservées.
list_nodes="$HOME/scripts/fichiers/list_nodes"
##Chemin des fichiers listes des clients et des masters
puppetclients="$HOME/puppet/ressources/fichiers/puppetclients"
puppetmasters="$HOME/puppet/ressources/fichiers/puppetmasters"
##Chemin du dossier des scripts de configuration puppet(master/cl
scripts="$HOME/puppet/scripts"
##Chemin du dossier ressources des modules puppet
modules="$HOME/puppet/ressources/modules"

```

FIGURE 4.2 – install part 1

Cette étape nous permet de personnaliser les chemins.

Ici, une étape de mise en place du DHCP vlan pour le cluster, puis une vérification que l'utilisateur puisse bien se connecter via ssh aux nodes du vlan.

```
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#Mise en place de KaVLAN
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
##Activation du dhcp
kavlan -e
##Récupération du numéro VLAN
jobid=`oarstat | grep $USER | cut -d' ' -f1`
vlan=`kavlan -V -j $jobid `


#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#Vérification et configuration SSH de l'utilisateur
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
##On regarde si le fichier .ssh existe
echo "Vérification de la configuration ssh de l'utilisateur "$USER
config_ssh >> $HOME/.ssh/config
```

Figure 4.3 – install part 2

Nous déployons les images disques sur les différents noeuds, puis nous créons les tunnels SSH avec l'utilitaire taktuk fourni par Grid5000.

```
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#Préparation des nodes (KaVLAN-edit)
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
##Déploiement du système d'exploitation Debian Squeeze-x64-base sur les nodes
echo "---"
echo "Liste des machines réservé:"
kavlan -l
if [ $choix == "n" ]; then kavlan -l > $list_nodes; fi
echo "---"
echo "Déploiement de l'environnement Linux, Debian Squeeze 64bit, sur toutes les machines réservé (Peut prendre plusieurs minutes)
kadeploy3 -e squeeze-x64-base -f $OAR_FILE_NODES --vlan $vlan -d -V4 -k $HOME/.ssh/id_dsa.pub
echo "Copie des clés SSH vers toutes les machines."
cat $HOME/.ssh/id_dsa.pub >> $HOME/.ssh/authorized_keys
echo $USER > $HOME/username
for node in $(cat $list_nodes)
do
   scp $HOME/.ssh/id_dsa* root@$node:~/.ssh/
   scp $HOME/username root@$node:~/
   taktuk -l root -s -m $node broadcast exec [ 'cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys' ]
done
echo "---"
```

Figure 4.4 – install part 3

Par la suite, nous installons ensuite la première machine en tant que Puppet Master, puis les autres machines comme clients.

Nous nous assurons enfin que les machines disposeront chacune d'un service avec Puppet.

```
80  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
81  #Installation du master puppet
82  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
83  ##récupération du node master (premier de la liste)
84  if [ $choix == "n" ]; then sed -n "1 p" $list_nodes > $puppetmasters; fi
85  puppetmaster=`cat $puppetmasters `
86  ##installation via APT des paquets serveur et agent de puppet. Notre serveur sera aussi son propre client
87  echo "Installation des paquets sur le serveur: "$puppetmaster
88  taktuk -l root -s -m $puppetmaster broadcast exec [ apt-get -q -y install puppet facter puppetmaster ]
89
90
91  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
92  #installation des clients puppet
93  #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
94  ##récupération des nodes clientes
95  if [ $choix == "n" ]; then cat $list_nodes | tail -n +2 >> $puppetclients; fi
96  ##installation via APT des paquets agent de puppet pour les clients
97  echo "Installation des paquets sur les machines clientes"
98  taktuk -l root -s -f $puppetclients broadcast exec [ apt-get -q -y install puppet facter ]
99
```

FIGURE 4.5 – install part 4

```
121 #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
122 #Déploiement des catalogues
123 #-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
124 ##synchronisations clients <-> master
125 ###envoie des demandes de certificat
126 echo "Déploiement des catalogues."
127 taktuk -l root -f $puppetclients broadcast exec [ puppet agent --test ]
128 ###signature des certificats
129 taktuk -l root -m $puppetmaster broadcast exec [ puppet cert --sign --all ]
130 ##rapatriement des catalogues/modules/manifests sur le master
131 echo "Rapartriement des catalogues/modules/manifests sur "$puppetmaster
132 scp $modules/ root@$puppetmaster:/etc/puppet/
133 ##attribution des rôles aux clients et ajout des clients dans nodes.pp
134 echo "Attribution des rôles :"
135 echo "- "`sed -n '2 p' $list_nodes `" : bind9,"
136 taktuk -l root -m $puppetmaster broadcast exec [ "bind=`sed -n '2 p' /root/list_nodes`; echo node '$bind' { include bind } >> /etc/puppet/manifests/nodes.pp" ]
137 echo "- "`sed -n '3 p' $list_nodes `" : MySQL,"
138 taktuk -l root -m $puppetmaster broadcast exec [ "mysql=`sed -n '3 p' /root/list_nodes`; echo node '$mysql' { include mysql } >> /etc/puppet/manifests/nodes.pp" ]
139 echo "- "`sed -n '4 p' $list_nodes `" : NFS."
140 taktuk -l root -m $puppetmaster broadcast exec [ "nfs=`sed -n '4 p' /root/list_nodes`; echo node '$nfs' { include nfs } >> /etc/puppet/manifests/nodes.pp" ]
141 ###récupération des catalogues
142 taktuk -l root -m $puppetmaster broadcast exec [ "echo import 'nodes' >> /etc/puppet/manifests/site.pp" ]
143 taktuk -l root -f $puppetclients broadcast exec [ puppet agent --test ]
```

FIGURE 4.6 – install part 5

4.2.3 Poursuite du projet, ce qui est à venir...

A la moitié du projet, il nous reste encore beaucoup de tâches à réaliser. tout d'abord, il nous reste à intégrer tous les modules comme le DNS, le DHCP, OAR et kadeploy. Il nous manquera également une récupération de toutes les recettes Puppet afin de pouvoir consulter l'état des machines. Lorsque toutes ces tâches seront réalisées, l'idéal serait de pouvoir installer un Puppet Dashboard afin de pouvoir administrer l'intégralité des services en temps réel. Comparé aux attentes du projet, nous réalisons déjà une installation sans image (en se basant tout de même sur l'existant).