Mathieu Bivert, CSSR, bivert@essi.fr

PFE: Cahier des charges (DOW)

Placement constraints for a better QoS in clouds

POLYTECH NICE-SOPHIA

Entreprise Université de Nice-Sophia Antipolis

Lieu Sophia-Antipolis, France

Responsable Fabien Hermenier, équipe OASIS, fabien.hermenier@unice.fr

Résumé

Dans le cadre de la répartition de machines virtuelles sur un ensemble de serveurs physiques, ce projet vise à formaliser puis implémenter des contraintes de placement, portant sur le type de systèmes de virtualisation.

Abstract

Within the framework of placing virtual machines on a set of physical servers, this project aims to formalize and to implement placement constraints, relative to the type of different virtualization systems.

Contents

1 Description du projet

1.1 Contexte de travail

Le monde industriel étant de plus en plus informatisé, la qualité des réseaux s'améliorant, les sociétés informatiques tendent à louer des structures informatiques accessibles à distance.

Les hébergeurs de services informatiques étant spécialisées dans la maintenance de ces structures, elles sont plus performantes qu'une société spécialisée dans la construction d'automobiles par exemple. Cette dernière a alors tout interêt a déporter la charge de la conception et de la maintenance de ses systèmes d'informations à une entreprise de services. Cette dernière pourra offrir à ses clients des solutions personnalisées, et peu coûteuses par rapport à une gestion "manuelle".

À noter que cependant que toutes les entreprises n'ont pas forcément interêt à exporter leur centre de traitement de l'information : par exemple des structures reposant sur des données hautement confidentielles (recherche de pointe, armée, état, etc.).

Le terme de "cloud" correspond à un certain nombre de serveurs physiques et de logiciels, utilisés par une entreprise de services. Ces dernières se déclinent en plusieurs types selon le(s) service(s) qu'elles proposent:

SaaS Software As A Service, fournir un logiciel (eg. un client email : Gmail[1]);

PaaS Platform As A Service, fournir un ensemble de logiciels et services (eg. accès aux Google Apps[2]);

IaaS Infrastructure As A Service, fournir des ressources matérielles (eg. Amazon EC2[3]);

DaaS Data As A Service, fournir un accès à des données (eg. Dropbox[4]);

...

En particulier, un cloud *IAAS* fournit à l'utilisateur l'accès à un ensemble de systèmes d'exploitations. Ces derniers sont très souvent virtualisés, ce qui présente l'avantage de pouvoir faire tourner plusieurs OS sur un même serveur physique. La virtualisation repose sur un *hyperviseur*, c'est-à-dire un moniteur de machines virtuelles (VM), dont le but est réaliser l'isolation entre les différentes VMs et de les administrer. L'administration consiste à démarrer, arrêter, migrer, régler les ressources des VMs.

Par exemple, la figure 1 montre l'hyperviseur *Xen* [BDF+03] sous NetBSD en train de virtualiser un FreeBSD, deux NetBSDs et une Debian.

Une problématique pour les gestionnaires d'IAAS est donc de pouvoir placer correctement un ensemble donné de VMs sur un ensemble de serveurs physiques. Ce placement n'est pas libre : il est régit par un ensemble de contraintes qui doivent être satisfaites.

1.2 Motivations

La virtualisation présente plusieurs avantages:

- sur une application dite n-tiers, il est possible de placer chaque tiers sur une VM, et éventuellement d'en faire des duplications, ce qui améliore la robustesse de l'application. Par exemple, n cas de défaillance d'un serveur physique, il est stratégique d'avoir placé le réplicat d'un serveur de base de données sur un serveur différent de l'original;
- l'admninistration et la gestion des machines est simplifiée : il y a moins de hardware, donc moins de maintenance physique; la possibilité de pouvoir cloner/charger/décharger à la volée des VMs permet d'améliorer la QoS facilement. Notamment, dans le cas où l'administrateur doit effectuer une opération de maintenance sur un serveur physique, il va devoir migrer [CFH+05] les VMs sur un autre serveur;
- chaque application peut être répartie sur une VM différente. Ainsi, si une application est compromise, elle a moins de chances de pouvoir compromettre d'autres applications que si elles étaient toutes lancées sous un même OS;

[1] `https://gmail.com`
[2] `https://www.google.com/enterprise/apps/business/`
[3] `https://aws.amazon.com/ec2/`
[4] `https://www.dropbox.com/`

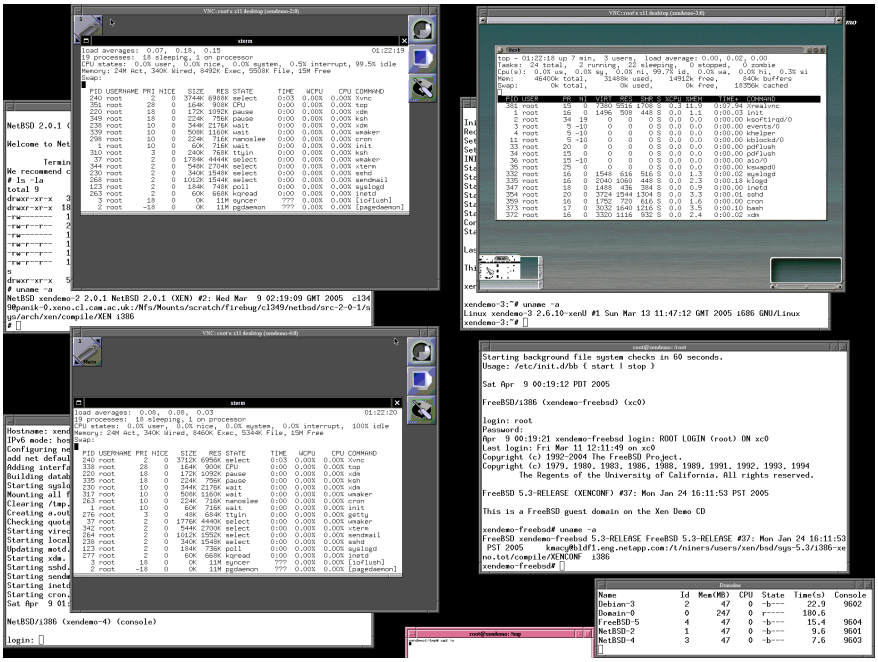


Figure 1: Virtualisation avec Xen; trois clients VNC [5]sont lancés pour accéder aux VMs

- utilisation plus performante du matériel, lorsqu'un ordinateur puissant peu être utilisé au maximum de ses performances en faisant tourner plusieurs systèmes d'exploitations. Cependant, utiliser un serveur à 99% de ses capacités n'est pas judicieux, puisque celui-ci sera alors incapable d'assurer une augmentation de charge. Il est donc impératif d'éviter une saturation;
- ...

La question de la répartition des machines virtuelles sur les machines physiques se pose alors pour des raisons diverses et variées:

maintenance un serveur physique peut tomber en panne, ou nécessiter une réparation, auquel cas les programmes tournant dessus doivent être migré ailleurs, afin de garantir au client une certaine qualité de service (QoS);

sécurité il peut s'avérer risquer pour un programme d'un client traitant des données sensibles (eg. données bancaires) de se retrouver au même endroit qu'un programme d'un autre client;

évolution des besoins où au cours d'un certain intervalle de temps, les besoins en puissance de calcul d'une entreprise peuvent augmenter (suite à une plus grande popularité par exemple), ou encore, augmentation brusque et irrégulière de la charge à des heures de pointes;

économie d'énergie où il peut être avantageux de réduire le nombre de serveurs physiques allumés, pour maximiser le rendement des autres machines physiques du cloud;

QoS où, à l'inverse de l'économie d'énergie, il est bon de garder des ressources supplémentaires disponibles immédiatement, de façon à ne pas perdre de temps (et donc en QoS) à redémarrer un autre serveur;

licence les entreprises fournissant les systèmes de virtualisation proposent des licences selon différents critères (eg. nombre de machines virtuelles lancées, utilisation de ressources (CPU, RAM, etc.));

plateforme plusieurs plateformes de virtualisations sont disponibles (eg. Xen, VMWare, Citrix); une autre contrainte sur la répartition des machines virtuelles se pose alors, un serveur physique ne faisant tourner qu'un seul type de plateforme;

...

[5]`http://www.hep.phy.cam.ac.uk/vncdocs/index.html`

1.3 Défis

Afin de pouvoir répondre aux besoins exprimés par l'un des domaines cité dans le paragraphe précédent, il est nécessaire de commencer par formaliser le problème. En d'autres termes, donner une définition mathématiques des contraintes impliquées par la problèmatique choisie, et s'assurer qu'elles sont envisageables en pratique. Finalement, cette représentation abstraite doit être implémentée sous forme de plugin Java pour Entropy [FH09], un manager de clusters reposant sur l'algorithme BtrPlace[6].

1.4 Objectifs

Actuellement, les trois/quatres derniers points cités ne sont pas forcément formalisés/implémentés complètement. Le projet consiste donc à choisir l'un de ces domaines et à l'ammener vers une forme satisfaisante.

Le dernier point est celui sur lequel se porte ce projet. Le travail sera d'autant plus original que les questions d'économies d'énérgies sont actuellement très prisées par les chercheurs au détriment des autres.

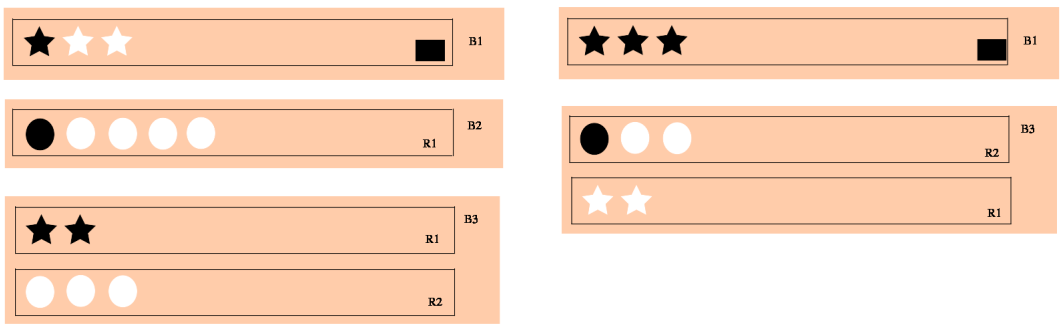
Dans un premier temps, il est nécessaire d'étendre le modèle actuel pour supporter le typage des infrastructures. C'est-à-dire, fournir une implémentation de base permettant d'associer un type à une machine virtuelle, et un ensemble de types possibles pour un serveur. En effet, ces derniers peuvent ne pas supporter n'importe quel hyperviseur.

Dans un second temps, on modélise et implémente un ensemble de contraintes correspondant à des besoins concrets. Par exemple, on peut limiter le nombre de plateformes d'un type donné, interdire un changement de type sur un serveur, etc.

1.5 Scénarios

1.5.1 Intuition

Pour faire simple, supposons dans un premier temps que comme en figure 2(a) on dispose de trois boîtes contenant une pile d'un ou plusieurs racks. Un rack peut contenir plusieurs objets d'une même forme (eg. rond, étoile). Par exemple la troisième boîte B_3 contient deux racks : R_1 qui est visible et contient deux étoiles, R_2 en dessous, pouvant stocker des ronds. En supposant que l'on doive prêter la deuxième boîte à un ami, il nous faut trouver une façon d'arriver à une configuration utilisant les deux boîtes restantes permettant de garder les mêmes objets visibles qu'au départ (figure 2(b)).



(a) Avant le prêt (b) Après le prêt

Figure 2: Simplification du problème

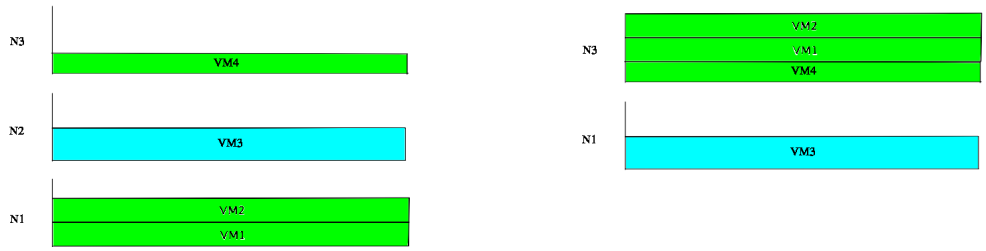
(XXX bug conversion PNG)

Pour y parvenir, on peut déplacer les deux étoiles de B_3 dans B_1, échanger les racks de B_3, mettre le rond de B_2 dans B_3, et finalement retirer B_2.

[6] `http://btrp.inria.fr/sandbox/about.html`

1.5.2 Cas réel

On reprend l'exemple précèdent sur la figure 3 avec des informations plus proches de ce qu'il se passe en réalité.



(a) Avant la reconfiguration (b) Après la reconfiguration

Figure 3: Exemple de changement de système de virtualisation. En vert des VMs Xen, en cyan une VM VMWare

Sur le diagramme 3(a), on souhait mettre le serveur physique N_2 hors-ligne pour des questions de maintenance. On utilise pour cela la contrainte *offline(N_i);*[7].

Comme aucun serveur VMWare n'est disponible, il est nécessaire de supprimer un serveur Xen, capable d'accueillir VM_3, par exemple N_1. Les machines virtuelles situées sur ce dernier, VM_1 et VM_2, doivent dans un premier temps être migrées sur un autre serveur.

Puis, N_1 doit s'éteindre et redémarrer en changeant son système de virtualisation. Enfin, la VM_3 est déplacée sur N_1, et N_2 peut être éteint, pour arriver dans l'état décrit par le diagramme 3(b).

1.6 Critère de succès

Pour que le projet aboutisse, il est nécessaire d'établir un formalisme mathématique correct pour représenter les différents types de VMs supportés par les machines physiques. L'implémentation dans BtrPlace doit aussi être réalisable.

2 État de l'art

(XXX pas lus, pas de citations)

2.1 Description générale

Historiquement, les gestionnaires de consolidation se sont concentrés sur le placement des VMs selon leurs besoins en ressources (CPU, RAM, BP, etc.), tout en cherchant à réduire la consommation énergétique

Traditinonellement, ces approches reposent sur des algorithmes ad'hoc rapides pour répartir les VMs. D'autres algorithmes plus flexible ont vu le jours suite au demande des clients, par exemple portant sur la fiabilité des applications.

Jung et al. proposent d'utiliser une fonction d'utilité pour les applications à haute disponibilité et performance. Leur solution porte sur la défaillance de serveurs physique, mais pas sur des changements de charge. Un algorithme glouton calcule le nombre de réplicats nécessaires pour chaque composant de l'application, de façon à satisfaire les besoins en temps de latence du réseau et en performance. Les réplicats peuvent être placés sur des racks ou des datacenters différents selon le degré voulu de fiabilité. (phrase pas claire). Ajouter de nouveaux paramètres, tels que la gestion des types de serveurs, demande

[7] `http://www-sop.inria.fr/members/Fabien.Hermenier/btrpcc/offline.html`

Id	Titre du livrable	Lot(s)	Nature	Date
D_0	Cahier des charges	1	Document	S_4
D_1	Gestion du typage et du déploiement	1	Document	?
D_2	Ensemble de contraintes	1	Document et Logiciel	?
D_3	Rapport de management	1	Document	S_{20}
D_4	Diaporama de présentation finale	1	Document	S_{20}

de revoir l'algorithme de reconfiguration. Les tests sont limités à un datacenter d'une douzaine de serveurs.

VMWare DRS donne à l'adminstrateur un accès à 4 contraintes similaire à *ban*, *fence*, *gather* et *spread*. Cependant, leur version de *spread* ne garantit pas que lors du déplacement d'une VM, l'ancienne version ne va pas tourner en même temps que la nouvelle. DRS n'est pas prévu pour être étendu, et 5 des contraintes de BtrPlace y sont manquantes. De plus, DRS ne choisit pas automatiquement en fonction des propriétés d'une VM entre une re-instantiation ou une migration à chaud. (induced re-location?) Enfin, DRS ne permet d'administrer que des clusters de 32 serveurs.

Entropy est à l'origine de BtrPlace. La programmation par contraintes est utilisée pour placer les VMs sur un nombre minimum de serveur. L'ordonnancement des actions nécessaires à la migration étant calculé avec une heuristique ad'hoc, il est impossible d'ajouter des contraintes telles que *spread*, qui affectent l'ordonnancement. Finalement, toutes les VMs en marche sont prises en compte lors de la résolution d'un mauvais placement, ce qui limite le nombre de serveurs gérés à quelques centaines.

Récemment, de nouvelles approches plus flexibles ont vu le jour; elles permettent d'intégrer des contraintes de placement à la demande. Bin et al. utilise aussi la programmation par contraintes pour fournir un gestionnaire de consolidation modulaire. Ils permettent une haute-disponibilité en guarantissant qu'à chaque instant, un certains nombre de serveurs sera disponible pour satisfaire la consommation de ressources des VMs et les contraintes de déplacement. Lorsqu'un serveur a de fortes chances de tomber en panne, les VMs sont migrés sur un autre capable de les satisfaire. Le modèle proposé ne supporte pas les contraintes portant sur la gestion de l'état des serveurs, l'ordonnancement des actions ou encore sur la façon dont les VMs sont migrées. Enfin, l'extensibilité n'a été vérifiée que pour 32 serveurs physiques et 128 VMs au maximum.

Quelques aspects théoriques de BtrPlace ont été étudié au préalable, avec un premier prototype prenant en compte les ressources allouées aux VMs, leur placement et l'étape de migration. [FH12] étends ce travail en montrant que BtrPlace est capable de gérer des datacenters de plusieurs centaines de serveurs. BtrPlace est donc à l'heure actuelle le gestionnaire le plus efficace en ce qui concerne le placement de VMs, tout en rendant possible de contrôler l'état des serveurs et les sur-allocations de ressources. L'addition d'une dizaine de nouvelles contraintes répondant à ces besoins démontre une extensibilité correcte. L'utilisation d'une heuristique basée sur une optimisation de filtre (XXX quésaco?) rends BtrPlace jusqu'à 20 fois plus rapide sur la gestion de datacenters composés de 2500 serveurs.

2.2 Formalisation du problème

3 Méthodologie et planification

3.1 Stratégie générale

3.2 Découpage en lots

bis repetita: trouver un bon formalisme; définir et implémenter

3.3 Plannification

gantt

Id	Jalon de fin de phase	Lot(s)	Date	Vérification
J_0	planification	1	S_4	D_0
J_1	formalisation	1	S_n	D_1 partiel
J_2	implémentation	1	S_{n+k}	D_2, D_1 partiel
J_3	projet	1	S_{20}	D_1, D_2, D_3 et D_4

3.4 Livrables associés au projet

3.5 Jalons

4 Description de la mise en œuvre du projet

4.1 Interdépendance des lots et tâches

bis repetita: trouver un bon formalisme; définir et implémenter

4.2 Description des lots

bis repetita: trouver un bon formalisme; définir et implémenter

4.3 Résumé de l'effort

4.4 Gestion du risque

5 Participants

5.1 Mathieu Bivert - CSSR

Étudiant à Polytech'Nice Sophia, spécialisé en Cryptographie, Systèmes Sécurité et Réseaux.

5.2 Fabien Hermenier - OASIS/INRIA

Fabien Hermenier a recu un doctorat en 2009 à l'université de Nantes. Depuis 2011, il enseigne en tant que Maître de conférence à l'université de Nice Sophia-Antipolis. Son travail de recherche s'articule autour des plateformes d'hébergement, de la virtualisation, du calcul autonome et de la gestion des ressources. Depuis 2006, il travaille sur des algorithmes de placement de machines virtuelles pour faire face à l'augmentation des SLA dans les plateformes d'hébergements.

Références

[BDF+03] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 164–177. ACM, 2003.

[CFH+05] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[FH09] Jean-Marc Menaud Gilles Muller Julia Lawall Fabien Hermenier, Xavier Lorca. Entropy : a consolidation manager for clusters. 2009.

[FH12] Gilles Muller Fabien Hermenier, Julia Lawall. Btrplace : A flexible consolidation manager for highly available applications. 2012.