

Mathieu Bivert, CSSR, bivert@essi.fr

PFE: Cahier des charges (DOW)

Placement constraints for a better QoS in clouds



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. Avant de s'intéresser à ces contraintes, il est nécessaire de doter BtrPlace d'un tel système de typage.

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. Before thinking about those constraints, BtrPlace needs to be extended in order to support such a type system.

Contents

1	Description du projet	3
1.1	Contexte de travail	3
1.2	Motivations	4
1.3	Défis	5
1.4	Objectifs	5
1.5	Scénario	5
1.6	Critère de succès	6
2	État de l'art	6
2.1	Description générale	6
3	Méthodologie et planification	7
3.1	Stratégie générale	7
3.2	Découpage en lots	7
3.3	Plannification	7
3.4	Livrables associés au projet	7
3.5	Jalons	7
4	Description de la mise en œuvre du projet	7
4.1	Interdépendance des lots et tâches	7
4.2	Description des lots	7
4.3	Résumé de l'effort	7
4.4	Gestion du risque	7
5	Participants	7
5.1	Mathieu Bivert - CSSR	7
5.2	Fabien Hermenier - OASIS/INRIA	8

Description du projet

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.

Une société spécialisée dans l'automobile par exemple, a tout intérêt à déporter la charge de conception et de maintenance de ses systèmes d'informations à un hébergeur de services, spécialisée dans ce domaine. En raison d'une plus grande expertise, cette dernière sera en effet beaucoup plus performante, engendrant alors une baisse des coûts pour l'entreprise cliente.

On notera cependant que toutes les entreprises n'ont pas forcément intérêt à exporter leur centre de traitement de l'information : par exemple des structures reposant sur des données hautement confidentielles, comme la recherche de pointe, l'armée ou les états.

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, par exemple:

Acronyme	Service fourni	Description	Exemple
SaaS	Software	Accès à un logiciel	Gmail ¹
PaaS	Platform	Une pile logicielle pour le développement	Google Apps ²
IaaS	Infrastructure	Accès à des ressources matérielles	Amazon EC2 ³
DaaS	Data	Accès à des données	Dropbox ⁴

En particulier, un cloud *IAAS* fournit à l'utilisateur un 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 de réaliser l'isolation entre les différentes VMs ainsi que de les administrer. Cette tâche d'administration consiste essentiellement à démarrer, arrêter, migrer, régler les ressources desdites 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.

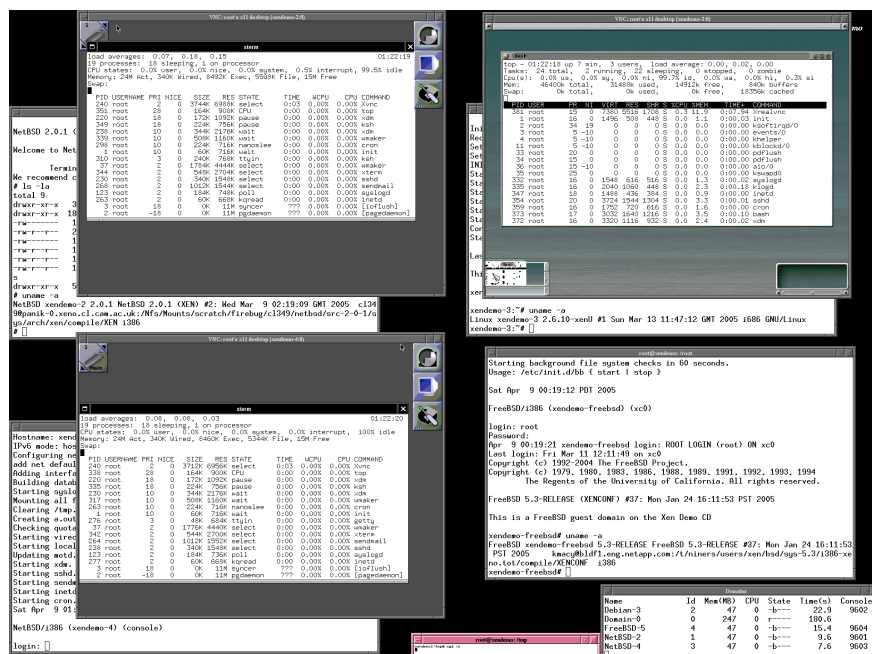


Figure 1: Virtualisation avec Xen; trois clients VNC ⁵sont lancés pour accéder aux VMs

Une problématique pour les gestionnaires d'IAAS est donc de pouvoir placer correctement un ensemble

de VMs sur un ensemble de serveurs physiques. Ce placement n'est pas libre : il est régit par un ensemble de contraintes, éventuellement variables, devant 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, en 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'administration 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. 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;
- utilisation plus performante du matériel, lorsqu'un ordinateur puissant peut 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 toujours 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 la qualité de service (QoS) qui lui est due;

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 croître (suite à une plus grande popularité par exemple), ou encore la charge pouvant augmenter de façon brusque, voire irrégulière, aux heures de pointe;

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, comme le nombre de machines virtuelles lancées, ou l'utilisation de ressources (CPU, RAM et bande passante principalement);

plateforme plusieurs plateformes de virtualisations sont disponibles (eg. Xen, VMWare, Citrix); comme à un instant donné un serveur physique ne peut faire tourner qu'un seul type de plateforme, une nouvelle contrainte sur la répartition des machines virtuelles se pose.

...

Enfin, on notera que ces besoins fluctuent au cours du temps; les systèmes de placement doivent donc être flexibles, au moins à deux niveaux. D'abord, ils doivent être *configurables*, afin de prendre en compte les spécificités et les propriétés, éventuellement variables, des applications et du datacenter. Une deuxième propriété d'*extensibilité* est nécessaire, pour être capable de supporter de nouvelles fonctionnalités au fur et à mesure de l'évolution des besoins.

⁵<http://www.hep.phy.cam.ac.uk/vncdocs/index.html>

1.3 Défis

Cette extensibilité passe par un ajout aussi aisé que possible de nouvelles contraintes. Le contexte fortement combinatoire (eg. le placement d'une VM influe sur celui des autres) rend cette tâche difficile.

BtrPlace [FH12] est un algorithme de placement de machines virtuelles, implémenté dans le manager de clusters Entropy [HLM⁺09]. Il utilise la programmation par contraintes [RvBW06] à travers la bibliothèque Java Choco [cho10] pour trouver une solution au placement. L'ajout de nouvelles contraintes en quelques dizaines de lignes de code sous forme de plugins garanti le besoin d'extensibilité précédemment formulé. Enfin, par rapport aux autres avancées dans ce domaine, BtrPlace trouve des solutions de placement assez rapidement, par exemple 3 minutes pour 5000 serveurs et 30000 VMs.

1.4 Objectifs

Actuellement, plusieurs des points besoins cités au niveau de la répartitions des VMs ne sont pas formellement implémentés ou formalisés entièrement. Le projet consiste à choisir l'un de ces domaines et à l'ammener vers une forme satisfaisante.

Le dernier point est celui sur lequel porte ce projet, à savoir rendre BtrPlace capable de gérer plusieurs types de VMs. Le travail sera d'autant plus original que les questions d'économies d'énergies sont historiquement très prisées par les chercheurs, au détriment des autres.

Au sein d'une infrastructure, plusieurs types d'hyperviseurs, et donc de VMs, doivent être supportés. Afin de satisfaire le critère de flexibilité, on peut avoir besoin de changer le système de supervision des VMs d'un serveur à la volée. Par exemple, on peut désirer vouloir garantir une certaine proportion d'une plateforme donnée, un nombre maximum de VMs d'un certain type.

De ce point de vue, BtrPlace est incomplet : il ne permet pas de gérer le type des VMs/plateformes, ni les étapes de reconfiguration nécessaires à un changement d'hyperviseur. Dans ce sujet, on étendra donc dans un premier temps 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 modélisera et implémentera un ensemble de contraintes pour contrôler les plateformes selon leur type. Par exemple, on peut limiter le nombre de plateformes d'un type donné ou encore interdire un changement de type sur un serveur.

1.5 Scénario

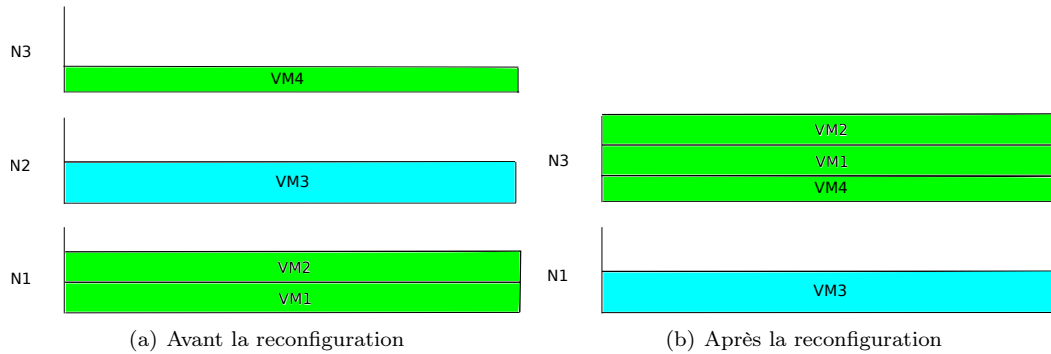


Figure 2: Exemple de changement de système de virtualisation. En vert des VMs Xen, en cyan une VM VMWare

Sur le diagramme 2(a), on souhaite mettre le serveur physique N_2 hors-ligne pour des questions de maintenance. On utilise pour cela la contrainte $offline(N_i)$.⁶

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.

⁶<http://www-sop.inria.fr/members/Fabien.Hermenier/btrpcc/offline.html>

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. L'état final est décrit par le diagramme 2(b).

1.6 Critère de succès

L'extension apportée à BtrPlace devra supporter correctement les problèmes de dépendances sur le placement des VMs, le typage des serveurs et les éventuels changement dynamique de type de ceux-ci. Enfin, les contraintes doivent être censées, bien définies, et répondre à des besoins concrets.

2 État de l'art

2.1 Description générale

Historiquement, les gestionnaires de placement 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

Traditionnellement, ces approches reposent sur des algorithmes ad'hoc, rapides, pour répartir les VMs. D'autres algorithmes plus flexibles ont vu le jour suite aux demandes des clients, par exemple portant sur la fiabilité des applications.

L'approche prise par [GJ10] consiste à utiliser une fonction d'utilité pour les applications à haute disponibilité. Leur solution est prévue pour fonctionner dans le cas d'une défaillance d'un serveur physique, mais n'est nullement adaptée pour répondre à des changements de charge. De plus, l'ajout de nouveaux paramètres tels que la gestion des types des serveurs nécessite de revoir l'algorithme de reconfiguration. Enfin, les tests sont limités à un datacenter d'une douzaine de serveurs.

VMWare DRS [Epp10] suit une approche similaire à celle de BtrPlace, en donnant à l'administrateur un accès à quelques contraintes (plus ou moins équivalentes à *ban*, *fence*, *gather* et *spread*). Cependant, leur version de *spread* ne garantit pas que lors du déplacement d'une VM, l'ancienne et la nouvelle version de celle-ci ne vont pas se superposer. DRS n'est pas prévu pour être étendu, et il lui manque 5 contraintes par rapport à BtrPlace. Enfin, l'administration est limitée à 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, l'algorithme prend en compte toutes les VMs en marche lorsqu'il cherche à résoudre un mauvais placement, ce qui en pratique 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. utilisent aussi la programmation par contraintes pour fournir un gestionnaire de placement modulaire. Ils permettent une haute-disponibilité en garantissant qu'à chaque instant, un certain 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 migrées sur un autre, capable de satisfaire leurs besoins. 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 étudiés au préalable, avec un premier prototype prenant en compte les ressources allouées aux VMs, leur placement et l'étape de migration. [FH12] étend 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 le contrôle de l'état des serveurs et les sur-allocations de ressources. L'ajout 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 rends BtrPlace jusqu'à 20 fois plus rapide sur la gestion de datacenters composés de 2500 serveurs.

Enfin, plusieurs plateformes expérimentales permettent de changer l'OS tournant sur un nœud. Grid'5000 [CCD⁺05], reposant sur Kadeploy⁷, un ensemble d'outils permettant de manager un en-

⁷<http://kadeploy.imag.fr/>

semble de nœuds, utilise OAR, un algorithme de réservations de ressources. Une autre plateforme, Emulab [HRS⁺08] utilise Assign [RAL03] comme algorithme de placement.

Aucun des deux algorithmes utilisés par ces plateformes ne supporte des contraintes, la gestion de licence, ou encore la préparation de plateforme à l'avance.

Donc pour résumer, aucun système existant hormis BtrPlace ne scale correctement, et aucun ne supporte le typage des VMs/nœuds, ni même les moyens de l'exprimer telle quelle avec leur modèle.

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

3.4 Livrables associés au projet

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}

3.5 Jalons

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

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 reçu 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.
- [CCD⁺05] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid’5000 : A large scale and highly reconfigurable grid experimental testbed. *IEEE/ACM Int. Workshop on Grid Computing*, 2005.
- [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.
- [cho10] Choco : an open source Java constraint programming library. Research report 10-02-INFO, Mines de Nantes, 2010.
- [Epp10] Epping, Duncan and Denneman Frank. *VMware vSphere 4.1 HA and DRS technical deepdive*. CreateSpace, 2010.
- [FH12] Gilles Muller Fabien Hermenier, Julia Lawall. Btrplace : A flexible consolidation manager for highly available applications. 2012.
- [GJ10] M.A. Hiltunen R.D. Schlichting C. Pu G. Jung, K.R. Joshi. Performance and availability aware regeneration for cloud based multitier applications. 2010.
- [HLM⁺09] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy : a consolidation manager for clusters. In *VEE*, 2009.
- [HRS⁺08] Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, and Jay Lepreau. Large-scale virtualization in the emulab network testbed. In *USENIX ATC*, 2008.
- [RAL03] R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network testbed mapping problem. *ACM SIGCOMM Computer Communication Review*, 33(2) :65–81, 2003.
- [RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming*. Elsevier Science Inc., 2006.