Mathieu Bivert, CSSR, bivert@essi.fr

PFE : Rendu intérmédiaire D_3

7 Mars 2013

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

L'ajout d'un type représentant les hyperviseurs dans D_2 permet l'implémentation de nouvelles contraintes dans BtrPlace. Ce document en présente quelques unes. Le lecteur est renvoyé à la première section de D_2 pour des définitions précises du vocabulaire employés ici.

Abstract

The addition of a type to represent the available hypervisors for a node in D_2 allows to implement new constraints in BtrPlace. This documents presents some of those possible constraints. The reader may refer to D_2 to find definitions of the vocabulary being used here.

Table des matières

1 Motivations **3**

2 Implémentation du modèle de D_2 **3**

3 Nouvelles contraintes de placement **3**
3.1 Motivations . . . 3
3.1.1 Licences . . . 3
3.1.2 QoS . . . 3
3.2 Validation . . . 3

4 Exemples de contraintes **3**
4.1 Nombre maximum de machines virtuelles . . . 4
4.2 Nombre minimum de nœud d'un type donné . . . 4

1 Motivations

BtrPlace est un logiciel permettant de trouver efficacement une solution à un problème de placement de machines virtuelles sur des serveurs physiques. Pour ce faire, l'utilisateur utilise des contraintes fournies par BtrPlace. Celles-ci permettent de rendre compte de certaines limitations (eg. ressources mémoire/cpu/réseau), d'une topologie particulière des composants d'une application (eg. réplicat de serveurs pour augmenter la disponibilité), etc.

Dans D_2, le modèle théorique de BtrPlace est étendu afin de permettre le typage des nœuds et des plateformes. Les contraintes qui en découlent portent sur la compatibilité entre machines virtuelles et nœuds, sur la capacité à un serveur physique de faire tourner un hyperviseur donné, ou encore sur le nombre de machines virtuelles d'un type donné pouvant tourner sur un hyperviseur.

2 Implémentation du modèle de D_2

Deux premières contraintes *Platform* et *TypedPlatform* permettent respectivement de changer le type d'un ensemble de nœud, et de s'assurer que les VMs d'un type donné sont bien placés sur un nœud du même type.

Le lecteur est renvoyé au D_2 pour une description plus précise.

3 Nouvelles contraintes de placement

3.1 Motivations

3.1.1 Licences

Comme mentionné dans les autres documents, les licences des hyperviseurs limitent les ressources pouvant être utilisées. Par exemple, VMWare[1] limite le nombre de machines virtuelles pouvant tourner sur un même nœud ; Xen limite la mémoire utilisable pour un hyperviseur, ou encore le nombre d'interfaces réseaux pouvant être connectées simultanément, voire la puissance CPU.

Des contraintes portant sur le type ajoutée peuvent rendre compte de ces limitations.

3.1.2 QoS

Dans le cas d'un déploiement réel, on peut chercher à optimiser le temps d'indisponibilité d'un service. Par exemple, supposons qu'un nœud sous un hyperviseur Xen tombe en panne : il est impératif que les VMs situées sur celui-ci soient migrées au plus vite. Avoir sous la main un nœud « vide » tournant sous Xen permet d'éviter d'avoir à redémarrer un nœud.

Une contrainte s'assurant qu'il existe au moins un certain nombre de nœud tournant sous un hyperviseur donné permet de palier à ce problème. On pourrait raffiner la contrainte, et s'assurer que pour un type, il existe au moins un nœud capable de recevoir telle ou telle VM en cas de problèmes.

3.2 Validation

De même que pour la contrainte du cas général, ces contraintes ne sont pas validées par manque de temps. Il faudrait définir des cas d'utilisation précis avec l'encadrant, puis implémenter des tests unitaires les mettant en œuvre.

4 Exemples de contraintes

L'implémentations est réduite au strict minimum, c'est-à-dire à l'injection de contraintes dans choco et au test de leur satisfiabilité.

[1] `https://www.vmware.com/support/licensing/per-vm/`

4.1 Nombre maximum de machines virtuelles

La contrainte *MaxVM*[2] implémente le cas décrit dans la section précèdente. Son constructeur prends en argument un ensemble de nœud, le type sur lequel porte la limitation, et enfin le nombre maximum de VMs pour ce type. Pour chaque nœud, on regarde si une action de retypage est prévue. Si c'est le cas, et que le nouveau type correspond à celui spécifié par l'utilisateur, on s'assure que le nombre de VMs sur ce nœud est inférieur à la borne.

4.2 Nombre minimum de nœud d'un type donné

La contrainte *MinPlatform*[3] spécifie que pour un type donné, au moins *nType* nœuds tournent sous un hyperviseur *type*. Pour ce faire, une variable est incrémenté à chaque fois qu'un nœud du type requis est trouvé, puis, on renseigne choco sur le fait que cette variable doit être inférieur à la borne donnée par l'utilisateur.

[2]https ://github.com/Heaumer/pfe/blob/master/entropy-fh/src/main/java/entropy/vjob/MaxVM.java
[3]https ://github.com/Heaumer/pfe/blob/master/entropy-fh/src/main/java/entropy/vjob/MinPlatform.java