Rapport sur le manuscrit de Thèse présenté par Jonathan Pastor

pour l'obtention du Doctorat de l'École des Mines de Nantes

Rapporteur : Stéphane GENAUD,
Professeur des Universités,
ENSIIE.

Titre du document évalué :

Contributions à la mise en place d'une infrastructure de Cloud Computing à large échelle

Le document présenté parJonathan Pastor décrit les recherches qu'il a menées dans le cadre de sa thèse sous la direction conjointe de Adrien Lèbre, chargé de recherches à l'INRIA Rennes Bretagne-Atlantique, et Frédéric Desprez, directeur de recherches à l'INRIA Grenoble Rhône-Alpes.

Contexte de l'étude

Les infrastructures de Cloud Computing, dans lesquelles des fournisseurs hébergent des services informatiques, sont ajourd'hui hébergées dans des centre de données dont la taille ne cesse de croître afin de favoriser des économies d'échelles. Une alternative envisageable à cette tendance de centraliser les ressources, est au contraire de décentraliser ces centres de données vers des infrastrucures géographiquement réparties, comme celles dont dispoent les opérateurs de réseaux, afin de rapprocher les ressources des utilisateurs finaux. Cependant, les difficultés qu'engendrent la gestion décentralisée des ressources sont considérables. Le travail de la thèse s'inscrit dans ce contexte, en proposant des solutions logicielles aux problèmes posés.

Analyse du document présenté

Le document est organisé en deux parties comptant 8 chapitres hors les conclusions et perspectives.

Après une introduction globale de la structure du document, la première partie (II) présente le contexte de la thèse. Après une brève introduction historique de l'évolution de l'informatique jusqu'au cloud computing (chapitre 1), une description assez complète du paysage du cloud computing est faite (chapitre 2). Le chapitre suivant se focalise sur la description et comparaison de trois architectures possibles pour les centres de données et le modèle de cloud computing induits : méga-centres de données, fédération de centre de données et edge/fog computing. Le dernier chapitre d'analyse du contexte actuel (chapitre 4) traite du logiciel OpenStack largement utilisé aujourd'hui pour gérer les clouds de type IaaS. Ce chapitre devient plus technique et décrit utilement l'architecture du logiciel pour engager une réflexion sur les moyens envisageables pour décentraliser son fonctionnement. Différentes approches de déploiement hiérarchique, publiés dans la littérature, sont présentées de manière assez synthétique. Toutefois, une description plus précise des difficultés liées aux mécanismes de synchronisation et d'exclusion mutuelle auraient pu y trouver leur place (par exemple, page 64). Au final,

cette partie descriptive du contexte est très pédagogique et démontre une très bonne connaissance du domaine par l'auteur.

La deuxième partie (III) aborde les contributions propres du travail. Le chapitre 5 pose clairement les objectifs d'un système Locality based Utility Computing (LUC-OS) qui répondrait à l'objectif général de la thèse de pouvoir assurer la gestion de ressources décentralisées. Ce chapitre est court, car il ne présente que les fonctionnalités désirées et le choix d'adapter OpenStack pour y répondre. Deux éléments d'OpenStack sont identifiés comme problématiques pour son adaptation à un environnement distribué : le bus de messagerie (RabbitMQ qui utilise un agent de mise en relation centralisé) et son système de base de données (MySQL). L'obstacle du bus de messagerie est écarté car la communauté s'est emparée du problème, et c'est l'obstacle de la base de données centralisée qui est au coeur du travail et fait l'objet du chapitre suivant.

Le chapitre 6 (*Support des bases clé/valeur dans Nova*) est beaucoup plus conséquent. La proposition faite ici consiste à substituer un système de base de données de type clé/valeur au système de base de données existant dans le module Nova d'OpenStack. Il faut souligner que tenter d'insérer ses idées dans un logiciel répandu, en constante évolution et d'une complexité certaine, est ambitieux et requiert une approche méticuleuse. Le chapitre explique clairement les choix possibles d'un point de vue technique et leurs implications, pour justifier le choix fait de développer une interface ORM nommée Rome, permettant à OpenStack d'accéder au système de stockage de type clé/valeur Redis. Le chapitre détaille ensuite les développements nécessaires dans l'ORM Rome pour que les opérations relationnelles disponibles dans l'API, soient également fonctionnelles quand le moteur de gestion des données est Redis. Cet aspect couvre les jointures, les sessions qui doivent faire appel à un système de verrou distribué, et les optimisations des requêtes vis à vis du volume de données transférées. La fin du chapitre est consacrée à une évaluation expérimentale de cette réalisation sur la plate-forme Grid'5000. L'expérimentation en environnement réel doit être saluée, car c'est une étape chronophage mais cruciale pour la vérification des hypothèses faites, et malheureusement pas toujours présente dans les travaux menés par la communauté. Sur la forme, la présentation du protocole expérimental pourrait être améliorée concernant la description des matériels impliqués et les caractéristiques des machines virtuelles (VMs) créées. La description de l'expérience n'est ici pas assez complète pour en assurer sa reproductibilité, ce qui est paradoxal quand le document fait état de l'utilisation de l'outil Execo qui a précisément cet objectif. Néanmoins, l'expérience apparaît pertinente, comparant les comportements du prototype dans un environnement constitué d'un seul cluster, puis dans des configurations multi-sites (artificiellement créés par émulation d'une augmentation de la latence). Ces résultats expérimentaux montrent que le prototype utilisant un système de stockage clé/valeur permet la création de VMs dans un temps constant jusqu'à 8 sites, alors que la version originale avec base de données relationnelles voit un brusque quadruplement de temps de création en passant de 6 à 8 sites.

Au final, ce chapitre décrit une réelle contribution, dont le prototype est implanté dans le logiciel phare du domaine. La démarche expérimentale menée en environnement réel renforce grandement la proposition, même si une analyse plus fine des résultats me semble souhaitable.

Le chapitre 7 (*Prise en compte de la localité réseau : le cas de DVMS*) est la deuxième contribution de la thèse. L'algorithme DVMS, précédent travail de l'équipe, est un algorithme distribué destiné à calculer dynamiquement le placement des machines virtuelles afin de d'atteindre des critères d'équilibre de charge. Cet algorithme utilise des mécanismes pair-à-pair pour structurer l'ensemble des machines virtuelles en groupes logiques (overlay structuré de Chord). Cette structuration ne prend pas en compte la localité réseau (définie par la latence) des machines et le travail de thèse a pour objectif d'étendre DVMS pour minimiser les collaborations entre noeuds lointains en terme de latence réseau. Le chapitre commence par citer les efforts analogues faits au début des années 2000 dans la recherche sur les réseaux pair-à-pair. Ensuite est développé l'algorithme proprement dit, dont l'objectif est le

regroupement de machines virtuelles proches du point de vue réseau. L'information de localité réseau est obtenu ici grâce à Vivaldi.

Quelques points d'ombres me semblent subsister dans la description de l'algorithme concernant la terminaison et les enrôlements concurrents. Une description formelle de l'algorithme eût été la bienvenue. Il faut relever l'effort d'intégration logicielle fait ensuite en intégrant le mécanisme de structuration, Chord ou Vivaldi, comme un module spécialisé de DVMS. Le chapitre s'achève avec une validation expérimentale sur Grid'5000, avec cette fois, une configuration entièrement réelle, distribuée sur quatre sites géographiquement distants. DVMS calcule dynamiquement le placement des VMs pour satisfaire des critères d'équilibrage de charge, déclenchant des migrations de VMs pour maintenir la satisfaction des contraintes. L'expérience observe le nombre de migrations intra-site et inter-sites qui ont lieu, avec la version DVMS basée sur Chord et celle sur Vivaldi. La version Vivaldi tenant compte de la localité réseau remplit parfaitement l'objectif avec un taux de migration intra-site environ deux fois supérieure à la version originale. L'expérience est convaincante, bien que le seul jeu de paramètres choisi pour l'expérience concernant la distribution statistique de la charge n'est pas discuté alors qu'il correspond peut être à des situtations courantes.

Le chapitre 8 (*Contributions à VMPlaceS : un simulateur pour algorithme de placement de machines virtuelles*) est la troisième contribution de cette thèse. Le logiciel VMPlaceS est un simulateur s'appuyant sur le moteur de simulation SimGrid, bien connu de la communauté. L'objectif est de simuler les résultats d'algorithmes de placements de machines virtuelles, étant donné une charge de travail évolutive dans le temps et une infrastructure modélisée. L'orientiation du travail vers la simulation dans cette dernière partie du travail est pertinente. Les deux chapitres précédents faisaient appel à une expérimentation en environnement réel, à échelle réduite, et le document jsutifie bien que la simulation est un moyen complémentaire d'expérimenter à plus grande échelle. La démarche expérimentale suivie dans ce chapitre est tout à fait recommandable : les résultats de la simulation de VMPlaceS sont tout d'abord comparés à une exécution réelle sur une infrastructure réelle (réduite à 32 noeuds) afin de valider le comportement du simulateur. Ensuite, la validation à plus grande échelle est faite en simulation.

La comparaison expérimentale porte ici sur le comportement de trois algorithmes de calcul de placements des VMs : l'un centralisé, le deuxième hiérarchique, et le troisième distribué (DVMS présenté au chapitre précédent). La métrique de comparaison est le temps cumulé où les VMs n'ont pas pu satisfaire la contrainte de charge maximale imposée, c'est-à-dire qu'elles sont en surcharge. Les résultats montrent une nette réduction du temps de surcharge avec DVMS, validant ainsi l'approche.

Conclusion

Le document est bien rédigé et agréable à lire. Jonathan Pastor décrit le domaine de recherche dans lequel il a inscrit ses travaux avec un recul qui démontre ses connaissances poussées des techniques et technologies de l'état de l'art. Une partie importante du travail est à la mise en œuvre des idées présentées dans des logiciels existants, en particulier le logiciel OpenStack très répandu. C'est un choix ambitieux, qui a pour avantage de vérifier la faisabilité de idées dans un contexte proche de celui des environnements de production, mais le désavantage d'hériter de toute la complexité du logiciel utilisé, rendant plus difficile la compréhension fine du prototype proposé. Pour valider les objectifs visés, une évaluation expérimentale considérable est menée pour chacune des propositions. Les résultats expérimentaux confirment l'intérêt des contributions proposées.

Les travaux ont fait l'objet de plusieurs publications (dans 3 conférences internationales de très bon niveau et un chapitre de livre). Considérant tous ces éléments, j'émets un avis favorable à la soutenance de la thèse de Jonathan Pastor en vue de l'obtention du titre de Docteur en informatique de

l'École des Mines de Nantes. La thèse peut être soutenue en l'état.

Stéphane GENAUD,
fait à Strasbourg, le 20 septembre 2016.