Un intergiciel d'une base de données NewSQL qui considère la localité de l'application cliente

Ronan-Alexandre Cherrueau, Adrien Lebre*

1er novembre 2017

Résumé

Une base de données *NewSQL* [2, 1] fournit le passage à l'échelle tout en continuant de garantir les propriétés ACID nécessaires à la réalisation d'une transaction. Pour ce faire, une base de données NewSQL distribue ses entrées uniformément entre tous ses nœuds. Elle repose ensuite sur un algorithme de consensus [3] pour effectuer des requêtes et réaliser des transactions cohérentes sur les nœuds. L'objectif de ce stage est de modifier la manière dont *CockroachDB* [1], une base de données NewSQL libre, distribue ses entrées. L'idée est de favoriser les accès à un seul nœud, plutôt qu'a la totalité, lors des requêtes.

Mots-clefs : Bases de données distribuées, Cloud Computing, infrastructure Fog/Edge, OpenStack.

1 Contexte du stage

L'initiative Discovery [2] étudie les infrastructures d'*informatique en nuage* (en anglais, cloud computing) massivement distribuées dans de petits centres de données situés à la périphérie du réseau. Le dessein de telles infrastructures est double. Premièrement, de s'affranchir des problèmes de conditionnement des

*prénom.nom@inria.fr

1. `https://github.com/cockroachdb/cockroach`
2. `https://beyondtheclouds.github.io/`

centres de données lors du passage à l'échelle de l'infrastructure. Deuxièmement, de limiter la latence en rapprochant les unités de calculs avec les utilisateurs. Cette infrastructure désignée sous le nom d'*informatique en périphérie* (en anglais, edge computing) sert les besoins d'une nouvelle génération d'applications utilitaires. Des applications plus aptes à prendre en compte les demandes en ressources toujours croissantes et la dispersion géographique de ses utilisateurs. Les applications de conduite autonome, l'Internet des Objets, la diffusion de flux vidéos et la virtualisations des fonctionnalités réseaux (NFV) sont des exemples d'applications qui doivent être déployées en périphérie.

Pour étudier les infrastructures d'informatique en périphérie, l'initiative Discovery analyse et révise le gestionnaire d'infrastructures OpenStack [3]. En quelques mots, OpenStack est un ensemble de services libres pour déployer et opérer une infrastructure d'informatique en nuage. Celui-ci fournit des ressources de calculs (*c.-à-d.*, machines virtuelles avec Nova, conteneurs avec Magnum, machines physiques avec Ironic), des ressources de stockages (*c.-à-d.*, volumes disques avec Cinder et hébergement de fichiers avec Swift) et des ressources réseaux (*ex.*, routeurs, commutateurs, *etc.* avec Neutron) qu'il met à disposition de ses utilisateurs.

OpenStack est depuis quelques années la solution, de fait, pour gérer les infrastructures privées et publiques d'informatique en nuage. C'est aussi la solution qui se profile pour gérer les infrastructures d'informatique en périphérie. Malheureusement, bien que l'accent ait été mis sur le passage à l'échelle au cours du développement d'OpenStack, de nombreux services, comme la base de données relationnelle, ne satisfont pas ce critère. Ceci rend impossible, dans l'état actuel, la gestion de centaines de petits centres de données massivement distribués. De même, déployer et opérer des centres de données en périphérie oblige OpenStack à faire face à de fortes latences. Mais là encore, des services comme le bus de communications, deviennent inutilisables en présence de ces fortes latences. Il faut donc adapter le système OpenStack pour opérer de manière unifiée l'ensemble de ces centaines de petits centres de données situés en périphérie.

2 Problème

Concrètement, cinq services principaux forment OpenStack : Cinder, Glance, Keystone, Neutron et Nova. Chacun gérant un aspect particulier des infrastructures en nuages. Dans un déploiement typique d'OpenStack, le service Nova fournit des ressources de calculs à l'utilisateur. Ces ressources de calculs se comportent selon une image système définit par Glance. Elles utilisent également le service Neutron pour virtualiser les communication réseaux et le service Cinder pour gérer les ressources de stockage. Enfin, toutes ces interactions se font sous la supervision de Keystone qui s'assure des privilèges de l'utilisateur.

Chaque service d'OpenStack utilise une base de données relationnelle pour sauvegarder son état. Mais, cette base de données représente un point bloquant pour la création d'un OpenStack massivement distribué. Les solutions types MySQL et PostgreSQL sont difficile à distribuer, notamment dans un environnent avec de fortes latences réseaux.

3. `https://openstack.org`

L'initiative Discovery travail actuellement sur le remplacement de la base de donnée relationnelle par une base de données NewSQL [2, 1]. Plus particulièrement, la base de données CockroachDB qui est une implémentation libre des systèmes NewSQL. La question se pose, toutefois, de savoir comment sont distribuées les entrées de la base à travers tous les nœuds du système NewSQL. Une infrastructure massivement distribuée d'informatique en périphérie plaide pour un modèle de distribution des entrées qui considérerait la *localité* des utilisateurs. Est-il possible d'obtenir cette forme de distribution avec la base de données NewSQL CockroachDB ?

3 Objectifs

3.1 Étudier le modèle de distribution des bases de données NewSQL

Les bases de données NewSQL reposent sur un système de *tableaux associatifs* (en anglais, key-value store) pour distribuer les entrées à travers tous les nœuds. L'étudiant devra lire la littérature sur ces bases de données [4, 2, 1] pour comprendre comment les entrées sont distribuées. De même, l'étudiant devra étudier le mécanisme de *Réplication Zones*[4] chez CockroachDB. Un mécanisme qui influence où doit se produire la réplication des entrées. Ainsi que le mécanisme, plus traditionnel, de *partitionnement horizontal* (en anglais, sharding) comme proposé par MySQL[5] et Vitess[6].

3.2 Programmer un intergiciel qui considère la localité de l'application cliente

À partir des enseignements de l'objectif précédent, l'étudiant devra proposer un moyen de distribuer les entrées de la base de données en considérant la localité de l'application cliente. Ce moyen pourra être implémenté à la manière d'un *intergiciel* (en anglais, middleware) entre la base de données NewSQL et l'application cliente. Un intergiciel est un programme qui permet l'échange d'informations entre différentes applications informatiques. Ici l'intergiciel pourra modifier les requêtes SQL de l'application cliente pour leur donner une portée locale ou globale.

L'application cliente considérée est OpenStack. Il est important de noter que les APIs et services d'OpenStack n'intègrent pas la notion de localité. De ce fait, des modifications dans le code d'OpenStack seront peut-être nécessaires.

3.3 Évaluer l'intergiciel dans le contexte d'un OpenStack massivement distribué

L'initiative Discovery développe un outil qui se nomme *Enos*[7] (Experimental environment for OpenStack). Enos est un cadriciel qui repose sur la technologie des conteneurs pour déployer et évaluer un OpenStack de production sur

4. `https://www.cockroachlabs.com/docs/stable/configure-replication-zones.html`
5. `https://www.mysql.com/products/cluster/scalability.html`
6. `http://vitess.io/`
7. `https://github.com/beyondtheclouds/enos`

n'importe quel banc de test. Il fournit aux chercheurs un petit langage pour exprimer simplement une configuration d'OpenStack. Cette configuration décrit les services à installer, leur agencement sur un banc de test et potentiellement des contraintes en terme de bande passante et débit sur les communications inter-services. À partir de cette configuration, Enos récupère des ressources sur le banc de test (Grid'5000, VirtualBox ou Chameleon), déploie OpenStack et applique les limitations réseaux. Enfin, Enos peut effectuer des tests de performances et collecter les résultats de ces tests pour faire des analyses à posteriori.

L'étudiant devra rédiger des scénarios de tests pour Enos. L'objectif est de mesurer l'intérêt de considérer la localité dans une base de données distribuée.

4 Références et communications

[1] David F. Bacon, Nathan Bales, Nicolas Bruno, Brian F. Cooper, Adam Dickinson, Andrew Fikes, Campbell Fraser, Andrey Gubarev, Milind Joshi, Eugene Kogan, Alexander Lloyd, Sergey Melnik, Rajesh Rao, David Shue, Christopher Taylor, Marcel van der Holst, and Dale Woodford. Spanner : Becoming a SQL system. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 331–343, 2017.

[2] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson C. Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. Spanner : Google's globally-distributed database. In *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8-10, 2012*, pages 261–264, 2012.

[3] Diego Ongaro and John K. Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference, USENIX ATC '14, Philadelphia, PA, USA, June 19-20, 2014.*, pages 305–319, 2014.

[4] Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. Scalaris : reliable transactional p2p key/value store. In *Proceedings of the 7th ACM SIGPLAN workshop on ERLANG, Victoria, BC, Canada, September 27, 2008*, pages 41–48, 2008.