Identification des limites d'une base de données NewSQL distribuée dans un réseau étendu

Ronan-Alexandre Cherrueau, Adrien Lebre*

1er novembre 2017

Résumé

Une base de données *NewSQL* [3, 1] offre les mêmes avantages de distribution qu'une base de données NoSQL, mais garantit en plus les propriétés ACID nécessaires à la réalisation d'une transaction. Ceci permet aux bases de données NewSQL de viser un déploiement large échelle tout en interprétant les requêtes pareillement aux bases de données relationnelles. L'objectif de ce stage est d'identifier les limites de *CockroachDB* [1], une base de données NewSQL libre, dans un réseau étendu qui présente de fortes latences et des déconnexions.

Mots-clefs : Évaluation de performances, Cloud Computing, bases de données distribuées, 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 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.

*prénom.nom@inria.fr

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

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.

L'initiative Discovery travail actuellement sur le remplacement de la base de donnée relationnelle par une base de données dite NewSQL [3, 1]. Plus particulièrement, la base de données CockroachDB qui est une implémentation libre des

3. `https://openstack.org`

systèmes NewSQL. La question se pose, toutefois, de savoir comment se comporte cette base de données dans un réseau étendu. Un réseau étendu couvre une grande zone géographique, ce qui implique de fortes latences et des déconnexions. Ces latences et déconnexions ont alors un impacte sur l'algorithme de consensus [4] utilisé par les base de données NewSQL ce qui peut provoquer des problèmes de performances ou de cohérences.

3 Objectifs

3.1 Définir un protocole expérimental

Dans un premier temps, l'étudiant devra définir un protocole expérimental pour identifier les limites de CockroachDB. Pour se faire, l'étudiant pourra, par exemple, étudier le mécanisme de *basculement* (en anglais, failover) de CockroachDB. Le basculement est une action automatique qui corrige une erreur dans le système, *ex.*, une déconnexion réseau. CockroachDB repose sur l'algorithme de consensus Raft [4] et utilise celui-ci pour implementer son mécanisme de basculement. Raft est connu pour sa simplicité, mais aussi pour ses limites lors de fortes latences. L'étudiant pourra aussi s'inspirer des suites de tests sysbench [4], jepsen [5] et des récents travaux de recherches dans le domaine [5, 2].

3.2 Évaluer le comportement de CockroachDB dans un réseau étendu

L'initiative Discovery développe un outil qui se nomme *enos-lib* [6]. Enos-lib est une bibliothèque python qui repose sur la technologie des conteneurs pour déployer et évaluer un système sur n'importe quel banc de test. Il fournit aux chercheurs un petit langage pour exprimer simplement une configuration. Cette configuration décrit les services à installer, leur agencement sur un banc de test et des contraintes en terme de bande passante et débit sur les communications inter-services. À partir de cette configuration, la enos-lib récupère des ressources sur le banc de test (Grid'5000, VirtualBox ou Chameleon), déploie le système et applique les limitations réseaux. À partir de là, le chercheur peut exécuter des tests de performances dont les résultats sont automatiquement collectés par la enos-lib pour faire des analyses à posteriori.

L'étudiant devra utiliser la enos-lib pour implementer le protocole expérimental défini précédemment. Ces tests analyseront les performances et la cohérence du système dans différentes configurations réseaux : sans latences, avec de fortes latences entre certain nœuds, entre tous les nœuds... L'étudiant devra également modifier la enos-lib pour partitionner le réseau et ainsi déclencher le mécanisme basculement.

4. `https://github.com/akopytov/sysbench`
5. `https://jepsen.io`
6. `https://github.com/beyondtheclouds/enoslib`

3.3 Synthèse et étude d'une procédure d'ajout à chaud d'un nœud

À partir de la synthèse des résultats précédents, l'étudiant devra étudier une procédure d'ajout à chaud d'un nœud dans le contexte d'un OpenStack distribué. Cette dernière partie, prospective, peut aboutir à une thèse dans le cadre de l'initiative Discovery ou de l'équipe STACK.

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] Boris Bulanov. Benchmarking Google Cloud Spanner, CockroachDB, and NuoDB. dzone.com/articles/benchmarking-google-cloud-spanner-cockroachdb-and, 2017. Accès : 28/10/2017.

[3] 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.

[4] 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.

[5] Todd Warszawski and Peter Bailis. Acidrain : Concurrency-related attacks on database-backed web applications. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 5–20, 2017.