Projet d'option GSI Vivaldi
Rapport final

Nicolas Joseph, Raphaël Gaschignard
Guillaume Blondeau, Cyprien Quilici, Jacob Tardieu

20 février 2014

1 Contexte

Le projet est en réalité un sous-projet au projet `DisCoVEry`. Il est né suite à l'impulsion du département de recherche en informatique de l'École des Mines de Nantes et consiste en une amélioration du modèle de cloud computing courrant. En effet, il apparaît que ce modèle est construit autour de centre de données centralisés qui sont excentrés géographiquement des zones d'utilisation. De cette remarque vient le modèle proposé par `DisCoVEry` : rapprocher les serveurs des utilisateurs. Plus précisément les réduire à un réseau de mini-data center présents au niveau des différents noeuds de réseau, de manière à construire un cloud complètement décentralisé dont les serveurs sont proches des lieux d'utilisation.

Rappelons que le principe du cloud computing est la virtualisation de machines sur des machines physiques. Dans un système de cloud il faut pouvoir migrer les machines virtuelles d'une machine physique à l'autre. Dans un centre de donnée cette migration se fait sur le réseau local et est donc tranparente. Dans le système proposé par l'équipe de recherche de `Discovery` il faut pouvoir effectuer des migrations au sein du réseau sur de longues distances. La notion de localité devient donc prépondérante pour améliorer les performances du systèmes. On préfére migrer une machine virtuelle sur un autre serveur d'un même cluster plutôt que sur un serveur distant de plusieurs centaines de kilomêtres.

2 Problématique

Il faut donc être capable de determiner quels sont les machines les plus proches de la machine courante au sein du réseau. Une première approche consisterai à pinger l'ensemble des noeuds pour determiner les distances relatives. Cependant, cette approche est très coûteuse en terme de communication réseau et c'est pour répondre à cette problématique notre projet

intervient. Grâce à un algorithme prédéfini, Vivaldi, notre brique logicielle doit être capable de fournir les nœuds les plus proches de la machine courante sans avoir contacter toutes les machines du réseau.

Cette brique logicielle devait être complètement indépendante des autres systèmes `DisCoVEry` de manière à pouvoir être remplacée facilement le cas échéant. Quatre tâches ont été définies au debut du projet par le client pour définir notre travail.

- Mise en œuvre du réseau logique Vivaldi en suivant un modèle de programmation de type acteur
- Mise en œuvre d'une API permettant d'exploiter cette notion de distance depuis les mécanismes de plus haut niveau
- Mise en œuvre d'un mécanisme permettant le parcours du réseau logique de manière efficace (i.e., sur une notion de plus court chemin) en s'appuyant sur l'API bas niveau
- Validation du mécanisme de parcours au sein de la proposition `DVMS`.

3 Méthodologie

3.1 Gestion de Projet

Ce projet a été réalisé en autonomie avec des réunions régulières suivant une méthode de développement agile. Nous reviendrons plus en détail sur le détail de la méthodologie de développement dans la partie 3.2.

Pour la gestion du projet, nous avons choisi un chef de projet qui avait un rôle de management administratif du projet. Il a géré une partie des relations de l'équipe avec les tuteurs et les éléments de gestion de projet comme le suivi d'avancement des différentes tâches, la répartition des tâches ...

Nous avons aussi utilisés plusieurs outils en ligne pour la gestion de projet :

- Trello - Gestionnaire de taches
- Propulse - Gestionnaire de tâches
- GitHub - Repository et gestionnaire de bugs
- Google Groups - Mailing lists

Nous avons commencé par utiliser Trello pour mettre en place le cahier des charges et les premières briques de base de Vivaldi et mettre en place différentes dates de rendu. Nous avons ensuite basculé sur Propulse, qui nous semblait un outil un peu plus conventionnel et utile que Trello pour la fin du projet.

Nous avons aussi régulièrement fait des réunions avec les différents acteurs du projet, à la fois les clients et les tuteurs école.

`GitHub`, en plus de nous servir de gestionnaire de version, nous a aussi servit de gestionnaire de bug. Jonathan Pastor a pu l'utiliser pour nous signaliser des bugs potentiels qu'il a pu rencontrer dans le code. Les notifications associées nous ont permis d'être réactifs sur la correction des bugs.

3.2 Développement

Le projet a commencé par une phase de courtes spécifications. Cette phase a abouti à l'élaboration d'une architecture logicielle permettant de répondre aux contraintes imposées, à un choix de technologie (Scala + Akka) et à la répartition de différentes tâches de développement parallélisables.

Le développement a donc pu commencer pour un premier sprint dans un cadre définit par les règles suivantes :

- Chaque développeur code les tests pour toutes les fonctions et fonctionnalités qu'il développe.
- Chaque développeur travaille sur une branche spécifique par fonctionnalité.
- Une fois le développement terminé, chaque développeur crée une pull request GitHub de manière à ce que le code soit relu et approuvé par quelqu'un d'autre
- Une fois le code approuvé, la branche est fusionnée dans la branche principale du projet

Une fois toutes les fonctionnalités développées et testées de manière indépendante, nous avons développé un test local global de manière à valider le bon fonctionnement et la stabilité de notre brique logicielle.

Il s'en est suivi une série de sprints de test - fix de manière à corriger les bugs restant sur la plateforme.

En parallèle nous avons aussi décidé de développer un système de monitoring nous permettant d'agréger des données sur le système distribué que nous étions en train de développer. Cela nous a permis de vérifier que le comportement du réseau Vivaldi était celui attendu.

4 Les Résultats et perspectives

Vivaldi a pu être intégrée à `DVMS` et testée sur le réseau `Grid 5000`. A l'issue de ces tests, nous avons pu observer que les migrations de machines virtuelles sont effectuées à 80% dans le même cluster alors qu'avec l'ancien algorithme `Chord` les délocalisations se faisaient à hauteur de 50%.

En l'état actuel, certains paramêtres du projet peuvent encore être optimisés pour permettre des gains de performances notables. Certaines fonctionnalités peuvent aussi être ajoutées. Il peut être ajouté des notions de localité pour, par exemple, prendre en compte les débits differents ou encore les perfomances "dures" des différents sytèmes.

5 Le Bilan du projet

Nous avons pu réaliser ce projet en 380h hommes environ. La charge de travail est restée relativement équilibrée entre les membres du groupe et est restée stable dans le temps. Nous avons aussi pu acquerir un certain nombre de compétences au cours de ce projet :

- Scala
- Modèle d'acteur avec Akka
- Tests au cours du développement
- Intégration continue
- Travail collaboratif avec l'utilisation de GitHub
- Git

De manière plus générale, l'équipe est globalement satisfaite du déroulement projet et de son résultat. Le Client semble aussi satisfait par notre travail qui répond au cahier des charges initialement établit.

Références

[1] Frank Dabek, Russ Cox, M. Frans Kaashoek, Robert Morris, *Vivaldi : a decentralized network coordinate system*. SIGCOMM 2004.