Rapport d'avancement : Alexandre LITHAUD - LIG

Ce fichier est un mini rapport qui va condenser et synthétiser les tâches réalisés ainsi que les difficultés rencontrées pendant mon stage au Laboratoire Informatique de Grenoble (LIG) dans l'équipe DATAMOVE.

Ce fichier est séparé en grande phase du stage qui va correspondre au grande missions que j'ai eu à réaliser.

Ce fichier n'est qu'un résumé très succin des taches réalisées. Pour voir les informations précisément, il est préférable de regarder le LOGBOOK qui est mis à jour quotidiennement. [**LIEN**]

Mise en contexte

Le sujet de mon stage consiste a étudier l'écosystème Nix et NixOS et de contribuer au projet de NixOS Compose qui permet de deployer des noeuds (où machines) dans différent écosystèmes comme dans un VM ou dans Grid5000. De plus il consiste à aider l'équipe DATAMOVE en maintenant certain outils ou en réalisant divers démonstrateurs de composition pour **Nxc** (NixOS Compose)

Ce qui à été fait

Monté en Compétences

Pendant les premières semaines de mon stage il a été important de prendre connaissance des différentes technologies qui ont été essentiel durant l'intégralité du stage.

Travaillant sur Nix il a été important de bien comprendre tous les différents aspect de Nix.

Afin de ce faire j'ai réalisé de nombreux programmes utilisant le langage de programmation fonctionnel qu'est Nix. Ces programmes ont été réaliser en suivant de nombreux tutoriels disponible sur internet, le wiki de NixOs et plus particulièrement les **Nix Pills**.

J'ai de plus, lors de mon stage installé sur une machine l'OS NixOS afin de pouvoir analyser son comportement et son fonctionnement. J'ai rapidement été passionné par le fonctionnement de la configuration system de NixOS et sur les différentes possibilités quelles entraînes. Si bien que j'ai réaliser dans mon temps libre un configuration Nix et Home manger utilisant les différents outils que j'ai utilisé lors du stage et que je continue d'utiliser aujourd'hui. (*ranger, zsh, tmux, neovim, fzf*) [**LIEN**]

Enfin, j'ai aussi étudié le fonctionnement ainsi que créé des fichiers spéciaux Nix appelé les ***flakes***. C'est fichiers, qui sont toujours expérimentaux bien que massivement utilisé, permettent de facilement et fonctionnellement créer des environnement ou des "compositions" qui sont parfaitement reproductible car ciblant en input des fichier ou dépôts distants en assurant le téléchargement de la bonne version en utilisant des hash. Cette partie à été l'une des plus importante car elle traite d'un fondement de base de nix aujourd'hui et qui essentiel dans de nombreux dépôts comme NixOs Compose par exemple.

(L'outils des flakes sont si puissant selon moi que j'ai créé quelque flake.nix dans des dépôts git afin de pouvoir automatiquement utiliser les bonne version des mise a jours grâce au Nix présent dans mon ordinateur Ubuntu)

NixOS Compose

Suite à la phase de "formation" au différentes technologies nécessaire j'ai commencé à étudier le programme et le fonctionnement de NixOS Compose pour ce faire j'ai suivie un git tutoriel permettant de comprendre rapidement comment utiliser l'outils et les différentes possibilités. De plus, j'ai pu demandé directement des informations à son mainteneur et co-créateur Quentin Guilloteau.

J'ai créé quelques compositions classiques en utilisant des dépendances simple afin de tester le fonctionnement de l'outils. J'ai déployé ses applications sur plusieurs "saveur" différentes comme VM, RamDisk, Grid5000-Image ou Grid5000-nfs-store.

Les "saveurs" ou flavours est le nom donné pour les différentes options de déploiement dans grid5000.

J'ai donc réalisé des applications simple permettant de deployer plusieurs machine possédant différents comportement en utilisant le principe de role inhérent à NixOS Compose. Ces machines communiquai entre elle a travers le réseaux Grid-5000 en utilisant le protocole NFS.

Afin de pouvoir efficacement utiliser grid-5000 j'ai du apprendre des outils comme tmux (un multiplexeur de terminal très utile pour utiliser ssh), ssh et rsync. Ces outils se son révélé essentiel afin d'accroître ma productivité sur Nxc dans le cas de déploiement de compositions de G5K.

Nur-Kapack

Nix utilise un git spécial afin de stocker toutes les dérivations nécessaire afin d'installer un paquets ce git est nommé Nixpkgs [**LIEN**]. Ce dépôt est au centre du gestionnaire de paquet qu'est Nix. Cependant nixpkgs n'est pas le seul moyen de télécharger des paquets. Il existe aussi le principe de NUR. NUR est un répertoire de package Nix gérer par le communauté, les paquets sont build a partir de la source contrairement a nixpkgs et surtout les paquets de NUR ne sont pas review par des membres de nixpkgs. Tous le monde est donc libre de partager des paquets Nix sur NUR.

J'ai été amenées à étudier le fonctionnement de NUR en utilisant un dépôt distant de l'équipe OAR nommé Nur-Kapack. [**LIEN**] J'ai pu tester le fonctionnement et faire des choses qui n'était pas possible avec nixpkgs. Comme par exemple, lancer les phases de compilations d'un outils les une après les autres. Ce qui est très utile dans le cadre de mise a jour et/ou de création de paquets cela assure une bonne capacité de debug en ciblant directement les phases de compilations qui sont problématiques.

J'ai donc pu tester toutes ces fonctionnalités avec le paquet OAR dans le NUR-Kapack.

BeegFS

Ma première grosse mission à été de créer une composition permettant de faire fonctionner et de déployer un système de fichier parallèle différent dans les différentes saveurs de NixOS Compose, BeegFS. Ce travail à été de longue haleine car de nombreux problèmes ont été rencontrés durant cette mission.

BeegFS était présent dans nixpkgs pendant un moment mais il a été supprimer du dépôt car considéré comme deprecated. J'ai donc récupéré les fichiers présent dans le dépôt avant leur suppression et les ai rajouter dans nur-kapack afin de pouvoir les modifier et l'utiliser facilement. BeegFS a été enlevé de Nixpkgs a raison car il était peux fonctionnel. Tous était a modifier. Par exemple, le fichier de création de la dérivation de base utilisai une "extra library" qui avais changé de nom il a donc fallut codifier directement un grand nombre de fichier .c et .h en remplaçant a la volé le nom de la librairie en utilisant la commande *substituteInPlace*.

Après cela c'est le module a du être massivement modifier car il utilisait un certain nombre d'attribut nix n'existant plus. Il m'a fallu réécrire la plupart des services systemd de beegfs à la fin afin de pouvoir lancer directement les différents services au bon moment.

Beegfs a besoin d'un certain nombre de service afin de fonctionner : **mgmtd** (Management) le service central qui s'occupe d'organiser tous les transfert et les requêtes, **meta** (Métadonnée) le service qui s'occupe de stocker les

métadonnées de chaque client et chaque transfert de données, **storage** (Stockage) le service qui s'occupe de stocker les différentes fichiers partagé, **client** (Client) la machine client qui va utiliser le Système de fichier.

Le service meta avais un problème car il utilisait des disque temporaire mais avais besoin d'un système de métadonnée. Ce problème à été résolu en utilisant les disque laisser a disposition par Grid5000 pour créer une partition pour les métadonnées.

Enfin, le service client utilisai un driver kernel qui a du être modifier à la main afin de le faire fonctionner. Il nécessitait une certaine version du kernel Linux (4.14) afin de fonctionner il a donc fallu modifier la version du kernel par défaut. Le résultat était fonctionnel mais il n'est pas possible du a un bug dans NXC de changer la version du kernel (car nxc lui meme en dépend). J'ai donc créé des issues git afin de montrer les problèmes et comment les reproduire.

Regale Upgrade

Ma mission actuelle consiste à mettre à jour les paquets regale. Regale est un autre dépôt qui stocke les différents outils scientifique important pou rle laboratoire. Tous les outils sont mis a jour régulièrement dans des git respectif mais presque aucun n'utilise directement les dernières version. En effet, afin de s'assurer de la reproductibilité il est commun de "pin" directement un certain commit d'un git dans le flake.nix ainsi nous somme assurer que toutes les personnes lançant le flake ai la même version. De plus, ces dépôt "pin" aussi sur une version de Nixpkgs (généralement la 22.05) qui est donc la version de nixpkgs qui a été créer en mai 2022 qui est donc loin d'être a jour.

J'ai donc changé la version de Nixpkgs de 22.05 à 22.11 (la version 23.05 étant trop instable pour le moment). De plus, comme regale utilise nur-kapack j'ai du créé une branche sur nur-kapack afin de pouvoir utiliser le dernier commit récent de oar. Ces petits changements ont entraîné un certain nombre de bug qu'il ma fallu corriger en faisant quelques changements dans le la composition ou dans les modules de oar sur nur-kapack.

La version 23.05 à changer le comportement d'une bibliothèque python essentiel à oar (poetry). Ce changement (de 1.0.8 en 22.11 à 1.1.0 en 23.05) à modifier l'architecture de la librairie et du module poetry-core. La correction nécessite quelques changements dans le code python de oar (modification d'import).

Ces changements demandent une connaissance significative du fonctionnement de Nix, de son paradigme, et des modules qui les composes. La correction des différentes erreurs est une tâche complexe qui me permet de comprendre de mieux en mieux le fonctionnement de Nix, NixOS, NXC et des nombreux autres outils qui les composes.

Enfin, après avoir corrigé oar j'ai pu exécuter des tests unitaires (grâce à la bibliothèque mpi) ou simplement grâce à des test déjà présent dans la composition nix que j'ai utilisé.

Ce qu'il reste a faire

Il me reste donc à continuer de mettre à jour et de tester le fonctionnement des outils de régale (il en reste encore 7) : **bdpo, bdpo-oar, bebida, ear, examon, melissa-oar-ear et melissa-oar**.

De plus, Nix étant très vaste je continue de m'améliorer et de comprendre de mieux en mieux son fonctionnement en pratiquant et en surmontant des problèmes de fonctionnement.

Résumé

Durant mon stage au LIG qui a commencé le 17 avril 2023, j'ai installé et utilisé Nix et NixOS. J'ai aussi utilisé et contribué au projet NixOS Compose et Nur-Kappack en rajoutant le système de fichier parallèle Beegfs dans nur. De plus, j'ai créé des démonstrateurs de beegfs et autres afin de pouvoir les faire fonctionner facilement et rapidement dans NixOS Compose et donc de pouvoir les deployer sur de nombreuse différentes "saveurs" comme VM, Docker ou Grid5000. Enfin, j'aide au quotidien l'équipe en mettant à jours les différents outils scientifique et les compositions NXC qui les déploient. En les mettant a jour avec la dernière version de Nixpkgs (23.05) qui est entré en mesure au milieu de mon stage le 1 juin 2023.