ANF UST4HPC
Fréjus

Solutions de packaging et containers pour le HPC

GRICAD - Pôle Calcul
Bruno Bzeznik

May 15, 2018

Annonce:

Workshop GriCAD :
Méthodes de déploiement logiciel modernes pour le HPC
le 31 mai à Grenoble

https://deploymenthpc.sciencesconf.org/

Sommaire

Solutions de packaging et containers pour le HPC

Un point de vue...

- La problématique
- Packaging
- Containers
- Autres solutions pour le déploiement d'applications et librairies
- Conclusion

La problématique

- La commande "module", avec des builds (manuels) locaux d'applications et de librairies est aujourd'hui ce qui se fait le plus, MAIS:
  - Difficile: les systèmes de build des applications HPC sont souvent mal documentés, non standards, mal testés,...
  - Pas reproductible
  - Une upgrade système peut tout casser
  - Chacun fait ce qu'il peut, souvent seul dans son coin
  - Les utilisateurs doivent faire avec ce qui est installé dans le système ou dans le répertoire d'applications
- Questions qui vous agacent (aller..., avouez-le!)
  - "**Mais pourtant, ça compile très bien sur ma station**" -> "on n'a pas le même système ou les mêmes versions de librairies"
  - "**Mais il y a un paquet Debian/Redhat/Ma-distrib-a-moi-que-jaime, pourquoi vous ne l'installez pas?**" -> "quand on installe un paquet, il faut aussi le faire sur tous les noeuds et ce sera le même paquet pour tout le monde, et je ne parle même pas des dépendances..."
- "module" ne suffit plus...

Les solutions de packaging

Les solutions de packaging pour le déploiement des applications et des librairies dans le contexte du HPC

Les solutions de packaging

- But principal: automatiser les taches de compilation et faciliter leur répétition, avec pour effet de bord intéressant de consigner, voire documenter, les étapes
  - "**Mais pourtant, ça compile très bien sur ma station**" -> "ok, cherchons ensemble ce qui fait planter la compilation de ce paquet dans notre environnement particulier et contribuons à la portablité de ce paquet"
- Eventuellement fournir des binaires tout prêts (sous la forme de modules d'environnement ou de paquets)
- Permettre le partage et le développement collaboratif

Solutions de packaging - Quelques exemples
CONDA

- Initialement écrit pour Python, permet maintenant l'installation / build de paquets binaires et la gestion d'environnements
- Nécessite Anaconda ou Miniconda qui installe un répertoire de dépendances
- Généralement installé dans le home de l'utilisateur

Solutions de packaging - Quelques exemples
EASYBUILD

- Dévelloppé pour le HPC
- Fourni des recettes de compilation pour de nombreux logiciels scientifiques
- Modules python (easyblocks), fichiers de configuration (easyconfig) et notion de toolchain (compilateur + librairies + versions)
- Génère automatiquement des "modules"

Solutions de packaging - Quelques exemples
SPACK

- Principal concurrent de Easybuild
- Dévelloppé pour le HPC
- Fourni des recettes de compilation pour de nombreux logiciels scientifiques
- DAG et focus sur une bonne gestion de la combinatoire compilateur/version-compilateur/librairies/verions-librairies
- Se rapproche de NIX/GUIX (voir plus loin) avec une gestion d'identifiants uniques à base de checksums
- Génère automatiquement des "modules"

Solutions de packaging - Quelques exemples
NIX

- Système de paquets à part entière (pas spécifique HPC)
- Fourni des paquets binaires et leurs recettes, avec recompilation à la volée quand nécessaire
- Permet aux utilisateurs d'installer/développer eux-même des paquets dans leur environnement
- Très grande reproductibilité grace à un système de checksums identifiant de manière unique chaque recette
- Un même paquet binaire peut fonctionner sur plusieurs machines ayant des OS différents (Nix embarque ses propres glibc)
- Optimisation du partage (/nix) et des contributions
- NIXOS: un OS basé sur NIX

Solutions de packaging - Quelques exemples
GUIX

Guix

- Très proche de NIX (le Gnu NIX)
- Le langage de description des paquets n'est pas le même
- Quelques concepts supplémentaires intéressants pour le HPC (mais que NIX 2 a tendance à reprendre)

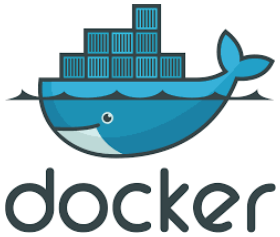
Les solutions de containers

Les solutions de containers, pour le déploiement des applications et des librairies dans le contexte du HPC

Les solutions de containers

- But principal: virtualiser pour s'affranchir du problème de la dépendance à l'OS de base et par effet de bord, permettre le partage facile d'images d'environnements d'éxécution
  - "**Mais il y a un paquet Debian/Redhat/Ma-distrib-a-moi-que-jaime, pourquoi vous ne l'installez pas?**" -> "OK, prenons une image Debian, installons-y ce paquet et faisons la tourner dans un container"
- Container = virtualisation légère
- Virtualiser uniquement le contexte d'exécution, mais pas forcément chercher l'isolation totale, qui amène souvent à perdre des performances (ex: virtualisation des accès réseau)
- MPI pas forcément simple, ou pas containerisé (ex: mpirun singularity ...)

Solutions de containers: quelques exemples
DOCKER



- Solution de containers très répandue, mais pas forcément adaptée au HPC
- Problème de sécurité en environnement multi-utilisateur: accès root nécessaire
- C'est néanmoins souvent la solution de base pour créer une image avant de l'éxécuter avec un autre mécanisme sur une machine de calcul
- DockerHub: partage d'images toutes prêtes. C'est un avantage mais aussi le risque de diffusion involontaire d'informations privées (mots de passe) et d'une mauvaise reconstructabilité (dépendance à une image de base qui peut changer et dont on ne sait pas forcément comment elle a été construite).
- Offre trop d'isolation par rapport au besoin (en particulier au niveau du réseau)

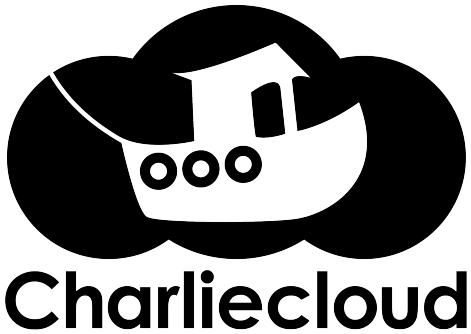
Solutions de containers: quelques exemples
SINGULARITY

- Développé spécifiquement pour le HPC
- Fourni un contexte d'éxécution plus léger (pas d'isolation réseau) et donc plus adapté
- Les images doivent être générées sur une machine sans restrictions de privilèges
- Exécution via SetUID ou maintenant Linux user namespaces

Solutions de containers: quelques exemples
SHIFTER

- Comme Singularity mais plus tourné vers la réutilisation directe des images Docker (avec une phase de conversion)

Solutions de containers: quelques exemples
CHARLIECLOUD



- Se concentre sur une techno: les Linux user namespaces
- Très léger, mais nécessite un noyau "assez" récent
- Probablement le plus proche d'une solution entièrement basée sur le noyau Linux

Solutions de containers: quelques exemples
UDOCKER

- Proche de Charliecloud
- Mais propose d'autres contextes d'exécution assez légers (tels que **proot** et une version patchée de proot) et une comparaison des performances entre eux

Autres solutions pour le déploiement d'applications et librairies

- Plateformes avec déploiement d'image à la demande : Grid5000
- Cloud
- Solutions Hybrides: par exemple Compute Canada utilise NIX comme base d'OS stable et Easybuild pour gérer une couche plus facile d'accès pour la mise en oeuvre des applications au dessus de NIX.

Conclusion
... non objective!

- Solutions de packaging
  - Avantages:
    - Excellente reproductibilité / reconstructibilité
    - Meilleures performances possibles
    - Travail communautaire facilité
  - Inconvénients: Pas toujours très facile d'accès, peuvent nécessiter un certain investissement de la part des utilisateurs
- Solutions de container
  - Avantages
    - Si un paquet existe dans une distribution, on peut l'utiliser tel quel
    - Pas de problèmes de dépendances
  - Inconvénients:
    - Souci d'organisation dû à la multiplicité des images: comment maintenir les images? qui les maintient?
    - La reproductibilité, bien que possible, n'est pas toujours garantie car la plupart des utilisateurs vont partir d'une image toute prête et la modifier sans forcément garder la trace des opérations
    - Solution peu adaptée aux jobs parallèles

Conclusion de la conclusion
... non objective!

Le meilleur des 2 mondes: proposer une ou plusieurs solutions de packaging et proposer un container quand le packaging devient trop compliqué.

Références

- Installing software for scientists on a multi-user HPC system - Keneth Hoste (Fosdem 2018)
  `https://fosdem.org/2018/schedule/event/installing_software_for_scientists/attachments/slides/2437/export/events/attachments/installing_software_for_scientists/slides/2437/20180204_installing_software_for_scientists.pdf`
- NIX as HPC packages management system - V. Reis (HUST 2017, Supercomputing)
  `https://hust17.github.io/2017_presentations/nix-paper.pdf`
- Conteneurs et HPC - Cédric Clerget et Laurent Philippe (10ème journées mesocentre)
  `http://calcul.math.cnrs.fr/IMG/pdf/presentation.pdf`
- `http://geekyap.blogspot.fr/2016/11/docker-vs-singularity-vs-shifter-in-hpc.html`
- Compute Canada NIX+Easybuild: `http://users.ugent.be/~kehoste/eum18/easybuild_nix_cvmfs_compute_canada_bart_oldeman.pdf`