Guix

Déploiement logiciel sur les supercalculateurs nationaux avec Guix : un retour d'expérience

7 novembre 2024
Rencontres Guix HPC

Romain Garbage
romain.garbage@inria.fr

À propos de moi

- CDD ingénieur de recherche au SED du centre INRIA de Bordeaux depuis le 1er décembre 2023
- Pas d'expérience de Guix avant cette date
- Utilisateur de logiciels libres depuis le début des années 2000
- Utilisateur d'une distribution Linux déclarative depuis fin 2022
- Première expérience professionnelle dans le secteur
- Pas de diplôme lié à l'informatique

Cas d'usage

- Test du support matériel de la pile logicielle HPC (SLURM/OpenMPI) : interconnexions réseau, GPUDirect, NCCL / RCCL...
- Déploiement et test "grandeur nature" de logiciels scientifiques nouvellement empaquetés (AVBP, Gysela...)

Machines utilisées

- Jean-Zay (IDRIS) : partition CPU et partition GPU (NVIDIA v100)
- Adastra (CINES) : partition CPU et partition GPU (AMD MI250)

Environnement logiciel

- ~~Guix~~
- Modules
- Pilotes matériels propriétaires (GPUs NVIDIA, interconnexion SLINGSHOT...)
- Singularity
  - Jean-Zay: possibilité de déployer une image Singularity personnalisée (documentation: `http://www.idris.fr/jean-zay/cpu/jean-zay-utilisation-singularity.html`)
  - Adastra: contacter le support pour déployer une image personnalisée
- Simple utilisateur (aucun droit administrateur)

Comment ?

Comment déployer et lancer des logiciels du gestionnaire de paquets Guix sur des machines où Guix n’est pas installé ?

Commande `guix pack`

```
$ guix pack --help
Usage: guix pack [OPTION]... PACKAGE...
Create a bundle of PACKAGE.
[...]
  -f, --format=FORMAT    build a pack in the given FORMAT
      --list-formats     list the formats available
  -R, --relocatable      produce relocatable executables
[...]
  -m, --manifest=FILE    create a pack with the manifest from FILE
[...]
```

Formats disponibles

```
$ guix pack --list-formats
The supported formats for 'guix pack' are:

  tarball       Self-contained tarball, ready to run on another machine
  squashfs      Squashfs image suitable for Singularity
  docker        Tarball ready for 'docker load'
  deb           Debian archive installable via dpkg/apt
  rpm           RPM archive installable via rpm/yum
```

Binaires repositionnables (Relocatable binaries)

Extrait de la documentation[1] :

```
--relocatable
-R

    Produire des binaires repositionnables - c.-à-d. des binaires que
    vous pouvez placer n'importe où dans l'arborescence du système de
    fichiers et les lancer à partir de là.

    Lorsque vous passez cette option une fois, les binaires qui en
    résultent demandent le support des espaces de nom utilisateurs
    dans le noyau Linux [...]
```

[1]:
https://guix.gnu.org/manual/devel/fr/html_node/Invoquer-guix-pack.html

Génération d'une archive contenant des binaires repositionnables

```
$ guix pack -R -S /bin=bin -S /etc=etc -C zstd hello
[...]
building /gnu/store/[...]-hello-tarball-pack.tar.zst.drv...
/gnu/store/iaqd0c0kqhps95blrnykgyr45ps33z8n-hello-tarball-pack.tar.zst
```

Détail des options

- `-C` : format de compression (`.gz` par défaut)
- `-S` : ajout de liens symboliques vers le profil

Contenu du pack

```
$ tar tf /gnu/store/[...]-hello-tarball-pack.tar.zst | tree --fromfile .
.
|-- bin -> gnu/store/bg7hf1f0j6bxl76zpdyp5jvb41k7rg6v-profile/bin
|-- etc -> gnu/store/bg7hf1f0j6bxl76zpdyp5jvb41k7rg6v-profile/etc
`-- gnu
    `-- store
        |-- [...]
        |-- bg7hf1f0j6bxl76zpdyp5jvb41k7rg6v-profile
        |   |-- [...]
        |   |-- etc
        |   |   |-- [...]
        |   |   `-- profile
        |   |-- manifest
        |   `-- [...]
        `-- [...]
```

Contenu de `etc/profile`

```
# Source this file to define all the relevant environment variables in Bash
# for this profile.  You may want to define the 'GUIX_PROFILE' environment
# variable to point to the "visible" name of the profile, like this:
#
#  GUIX_PROFILE=/path/to/profile ; \
#  source /path/to/profile/etc/profile
#
# When GUIX_PROFILE is undefined, the various environment variables refer
# to this specific profile generation.

export PATH="${GUIX_PROFILE:-/gnu/store/[...]-profile}/bin${PATH:+:}$PATH"
```

Cas 1

Test de la pile MPI de Guix sur Jean-Zay avec OSU benchmarks

Génération de l'archive sur la machine locale (avec Guix)

Génération de l'archive

```
$ guix time-machine -C channels.locked.scm \
    -- pack -R \
           -S /libexec=libexec \
           -S /bin=bin \
           -S /etc=etc \
           -C zstd \
           -m manifest-jean-zay.scm
[...]
/gnu/store/[...]-osu-micro-benchmarks-cuda-slurm-tarball-pack.tar.zst
```

Copie de l'archive sur Jean-Zay dans le répertoire $WORK

```
$ scp /gnu/store/[...]-osu-micro-benchmarks-cuda-slurm-tarball-pack.tar.zst \
   user@jean-zay:/path/to/$WORK/guix-pack.tar.zst
```

Déploiement sur Jean-Zay

Extraction de l'archive

```
$ cd $WORK && mkdir -p guix-pack \
           && zstd -d guix-pack.tar.zst \
           && tar xf guix-pack.tar -C guix-pack
```

Positionnement des variables d'environnement

```
$ export GUIX_PROFILE=$WORK/guix-pack
$ source $GUIX_PROFILE/etc/profile
$ unset LD_LIBRARY_PATH
```

Utilisation

Test de communication entre 2 GPUs sur le même noeud en utilisant NCCL

```
$ $GUIX_PROFILE/bin/srun -A project@v100 --time=00:05:00 \
  --nodes=1 --ntasks-per-node=2 --cpus-per-task=1 --gres=gpu:2 \
  env LD_PRELOAD="/usr/lib64/libcuda.so /usr/lib64/libnvidia-ml.so" \
      PSM2_GPUDIRECT=1 PSM2_CUDA=1 \
  $GUIX_PROFILE/libexec/osu-micro-benchmarks/xccl/pt2pt/osu_xccl_bw D D
[...]
#Using NCCL
[...]
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size      Bandwidth (MB/s)
[...]
2097152             35534.99
4194304             39434.92
```

Cas 2

Test de la pile MPI de Guix sur Adastra avec OSU benchmarks

Génération de l'archive

Génération de l'archive sur la machine locale avec Guix

```
$ guix time-machine -C channels.locked.scm \
    -- pack -R \
           -S /libexec=libexec \
           -S /bin=bin \
           -S /etc=etc \
           -C zstd \
           -m manifest-adastra.scm
[...]
/gnu/store/[...]-osu-micro-benchmarks-rocm-slurm-tarball-pack.tar.zst
```

Copie de l'archive sur Adastra dans $CCFRWORK

```
$ scp /gnu/store/[...]-osu-micro-benchmarks-rocm-slurm-tarball-pack.tar.zst \
      user@adastra:/path/to/$CCFRWORK/guix-pack.tar.zst
```

Déploiement sur Adastra

Extraction de l'archive

```
$ cd $CCFRWORK && mkdir -p guix-pack \
               && zstd -d guix-pack.tar.zst \
               && tar xf guix-pack.tar -C $CCFRWORK/guix-pack
```

Désactivation du plugin LUA dans la configuration de SLURM

```
$ cp /etc/slurm/slurm.conf ~
$ sed -i ~/slurm.conf -es'/^PlugStack/# PlugStack/g'
```

Positionnement des variables d'environnement

```
$ export GUIX_PROFILE=$CCFRWORK/guix-pack
$ source $GUIX_PROFILE/etc/profile
$ unset LD_LIBRARY_PATH
$ export SLURM_CONF=~/slurm.conf
```

Utilisation

Test de communication entre 2 GPUs sur le même noeud en utilisant RCCL

```
$ $GUIX_PROFILE/bin/srun -A user --time=00:05:00 --constraint=MI250 \
    --nodes=1 --ntasks-per-node=2 --cpus-per-task=1 --gpus-per-node=2 \
    --mpi=pmix \
    $GUIX_PROFILE/libexec/osu-micro-benchmarks/xccl/pt2pt/osu_xccl_bw \
      -d rocm D D
[...]
#Using RCCL
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size      Bandwidth (MB/s)
[...]
2097152             14611.61
4194304             14943.06
```

Cas 3

Test de la version C++ de Gysela sur la partition GPU de Jean-Zay en utilisant Singularity

Génération d'une image Singularity sur la machine locale

Génération de l'image

```
$ guix time-machine -C channels-gyselalibxx.locked.scm \
    -- pack -f squashfs \
           -S /bin=bin \
           --entry-point=/bin/bash \
           -m manifest-gyselalibxx.scm
```

Copie de l'image sur Jean-Zay dans le répertoire $WORK

⚠ à l'extension `.sif` ⚠

```
$ scp /gnu/store/[...]-osu-micro-benchmarks-cuda-slurm-tarball-pack.tar.zst \
    user@jean-zay:/path/to/$WORK/gyselalibxx.sif
```

Déploiement sur Jean-Zay

Copie de l'image Singularity dans l'espace d'exécution autorisé

```
$ idrcontmgr cp $WORK/gyselalibxx.sif
```

Chargement du module nécessaire à l'utilisation de Singularity

```
$ module load singularity
```

Documentation de Jean-Zay sur Singularity

http://www.idris.fr/jean-zay/cpu/jean-zay-utilisation-singularity.html

Utilisation du conteneur

Lancement de la simulation avec SLURM

```
$ srun -A user@v100 --ntasks=1 --gres=gpu:1 --cpus-per-task=1 \
   singularity exec --nv $SINGULARITY_ALLOWED_DIR/gyselalibxx.sif \
   env LD_PRELOAD=/.singularity.d/libs/libcuda.so \
   sheath_xperiod_vx config-gyselalibxx.yaml
```

Méthode générale

- Génération d'une archive avec Guix (machine locale ou mésocentre)
- Copie de l'archive sur le supercalculateur national (`scp`)
- Déploiement de l'archive sur le supercalculateur
- Lancement de l'exécutable avec SLURM (`srun` ou `sbatch`)

Points à garder en tête

Dans tous les cas

- utilisation de `LD_PRELOAD` pour CUDA

Avec les binaires repositionables

- compatibilité entre les versions de SLURM (archive/machine hôte)
- absence d'isolation de l'environnement de la machine hôte (variables d'environnement, en particulier `LD_LIBRARY_PATH` ; accès aux commandes de l'hôte…)

Limitations et perspectives

Limitations

- Avec les images Singularity, nécessité de passer par le support pour le déploiement d'une image personnalisée sur Adastra
- Avec les binaires repositionables, nécessité de l'activation de la fonctionnalité des espaces de nom utilisateurs (user namespaces) : activée sur Adastra et Jean-zay mais pas sur Irene (TGCC)

Perspectives

- Déploiement sur les machines du TGCC (`pcocc`)

Crédits

- Basé sur le travail et les conseils de (par ordre alphabétique):
  - Emmanuel Agullo
  - Ludovic Courtès
  - Florent Pruvost

Merci de votre attention

Questions ?

Utilisation de `sbatch` avec des binaires repositionnables

Génération d'un script pour Jean-Zay, partition CUDA

```
cat > job.sh <<EOF
#!/bin/bash
#SBATCH -A user@v100
#SBATCH --gres=gpu:1
#SBATCH [autres directives]
# Positionnement des variables d'environnement
export GUIX_PROFILE=$WORK/guix-pack
source \$GUIX_PROFILE/etc/profile
export LD_PRELOAD=/usr/lib64/libcuda.so
# Changement du répertoire courant
cd $WORK # ou #SBATCH--chdir=$WORK
# Commandes à exécuter pour la simulation
[...]
EOF
```

Utilisation de `sbatch` avec une image Singularity

Contenu d'un script pour Jean-Zay, partition GPU

D'après la documentation

```
#!/bin/bash
#SBATCH -A user@v100
#SBATCH --gres=gpu:1
# [...]
# on se place dans le répertoire de soumission
cd ${SLURM_SUBMIT_DIR}
# nettoyage des modules charges en interactif et herites par defaut
module purge
# chargement des modules
module load singularity
[...]
# exécution du code depuis espace d'exécution autorisé avec l'option --nv
# afin de prendre en compte les cartes NVIDIA
srun singularity exec --nv $SINGULARITY_ALLOWED_DIR/image.sif ma_commande
```

Instantiation d'un environnement logiciel

```
# Instantiation d'un environnement conteneurisé contenant
# le paquet hello
$ guix shell --container hello

# Instantiation d'un environnement conteneurisé contenant
# les paquets définis dans le fichier manifest.scm
$ guix shell --container --manifest=manifest.scm

# Version courte de la commande prédédente
$ guix shell -C -m manifest.scm
```

Provenance de la définition des paquets (canaux)

Fichier ~/.guix/config/channels.scm

```
(append (list
         (channel
          (name 'guix-hpc)
          (url "https://gitlab.inria.fr/guix-hpc/guix-hpc.git")
          (branch "master")))
        %default-channels)
```

État des canaux configurés pour l'utilisateur

```
$ guix describe
Generation 45 Oct 03 2024 12:31:46 (current)
  guix 308877b
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: 308877be17b2c2c92d972e105e8cc6a782ab4c82
  guix-hpc f5b49c8
    repository URL: https://gitlab.inria.fr/guix-hpc/guix-hpc.git
    branch: master
    commit: f5b49c82f590cf799267e8556ab694285645c4fc
```

Génération d'un manifeste

Commande

```
# Génération d'un fichier manifeste contenant le paquet hello
guix shell --export-manifest hello > manifest.scm
```

Contenu du fichier manifest.scm obtenu

```
;; What follows is a "manifest" equivalent to the command line you gave.
;; You can store it in a file that you may then pass to any 'guix' command
;; that accepts a '--manifest' (or '-m') option.

(specifications->manifest (list "hello"))
```

Plus de détails le manuel :

https://guix.gnu.org/manual/devel/fr/html_node/Ecrire-un-manifeste.html

Et dans . . . jours/mois/années ?

La commande `guix pull` permet de maintenir à jour ses définitions de paquets, mais que se passe-t-il si je recrée le même environnement à une date ultérieure ou sur une autre machine ?

Utilisation de canaux "verrouillés"

```
(list (channel
        (name 'guix)
        (url "https://git.savannah.gnu.org/git/guix.git")
        (branch "master")
[...]
        (commit
          "308877be17b2c2c92d972e105e8cc6a782ab4c82"))
      (channel
        (name 'guix-hpc)
        (url "https://gitlab.inria.fr/guix-hpc/guix-hpc.git")
        (branch "master")
        (commit
          "f5b49c82f590cf799267e8556ab694285645c4fc")))
```

Exemple de commande utilisant la machine à remonter le temps

```
$ guix time-machine -C channels.locked.scm -- describe
[...]
building package cache...
building profile with 2 packages...
  guix 308877b
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: 308877be17b2c2c92d972e105e8cc6a782ab4c82
  guix-hpc f5b49c8
    repository URL: https://gitlab.inria.fr/guix-hpc/guix-hpc.git
    branch: master
    commit: f5b49c82f590cf799267e8556ab694285645c4fc
```

Générer un fichier `channels.locked.scm`

```
# Sauvegarde l'état de la configuration globale des canaux
# à un instant donné
$ guix describe --format=channels > channels.locked.scm
```