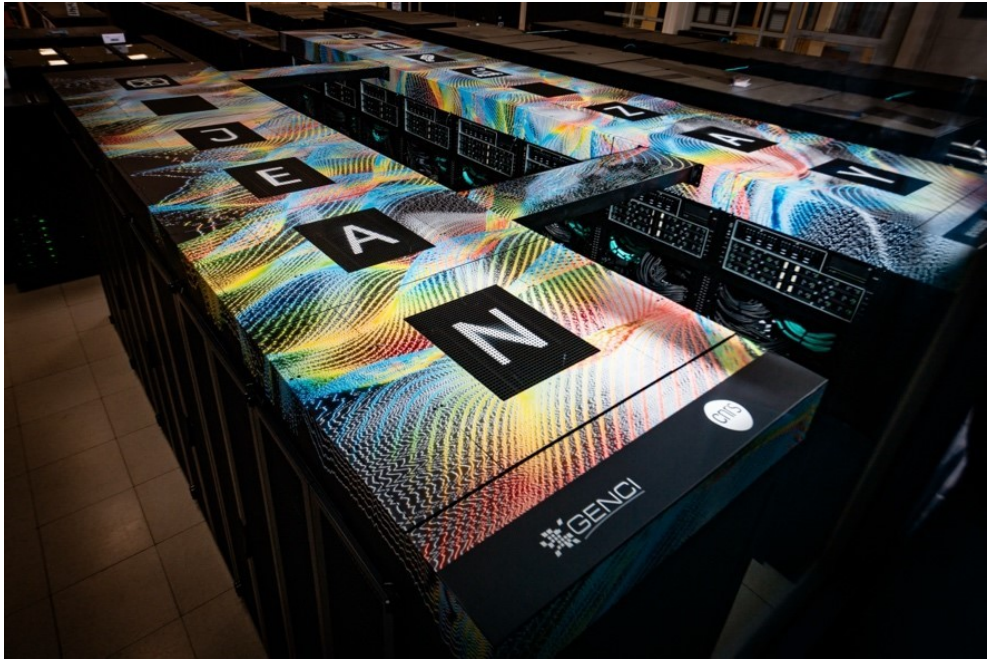
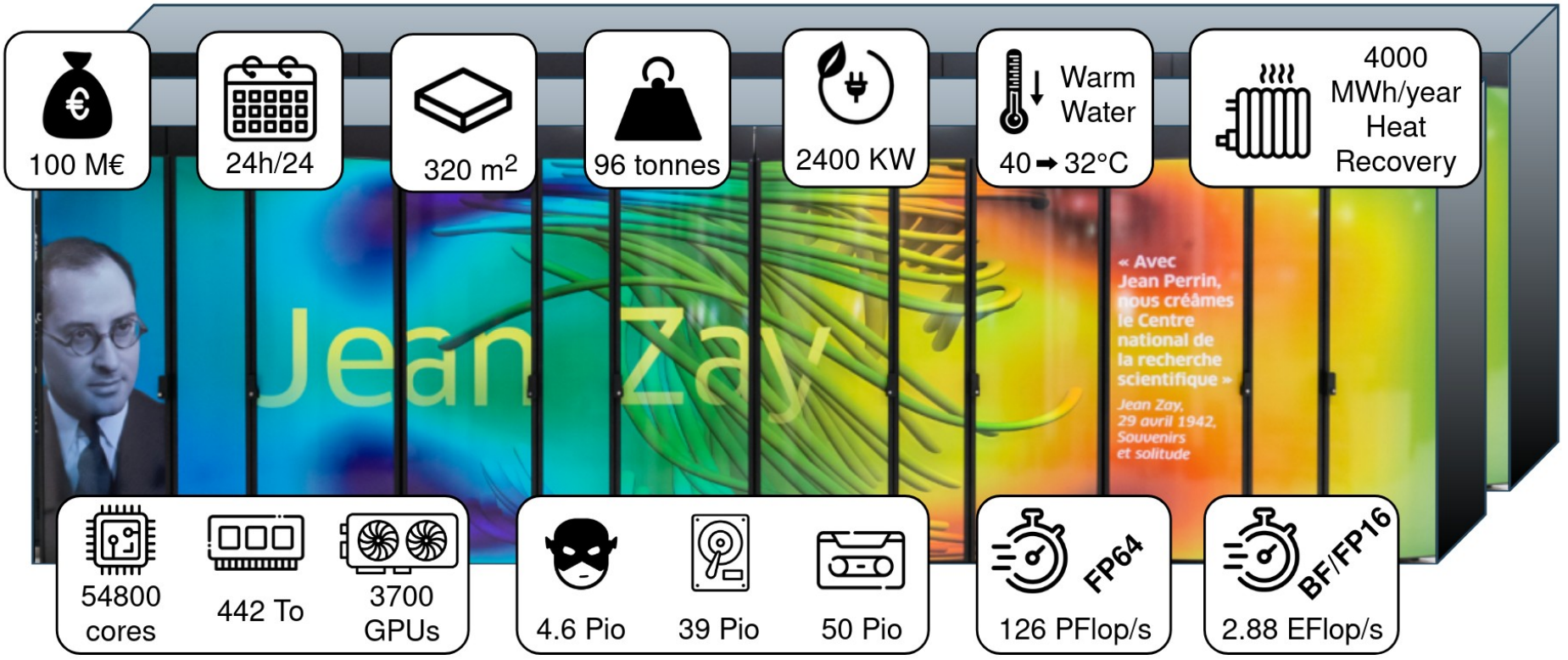
Software packaging and deployment at IDRIS



Rémi Lacroix – 24/02/2026

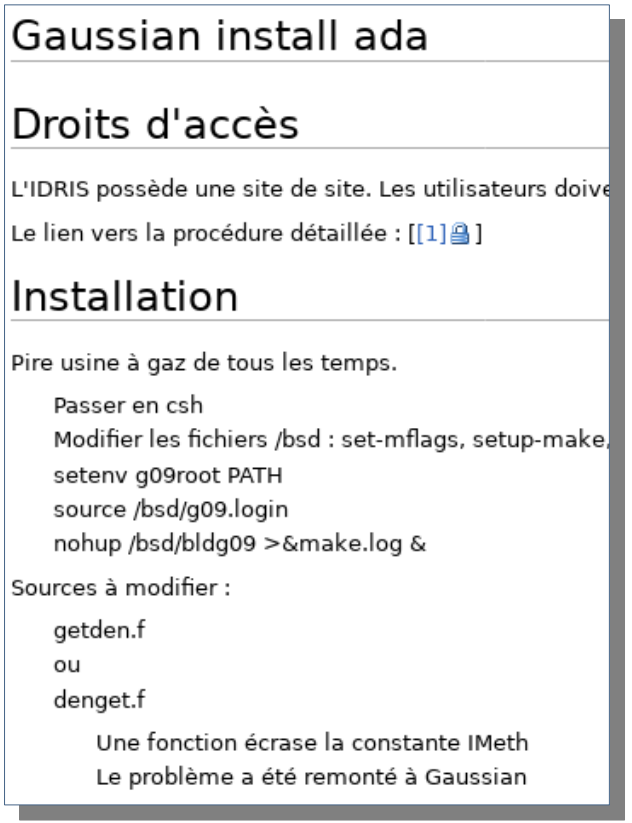
IDRIS

- One of the French national super-computing centers
- Operated by the CNRS
- ~4000 users:
  - High Performance Computing
  - Artificial Intelligence (since 2019: Villani report and IA For Humanity plan)
- Jean Zay: « Converged » supercomputer for HPC and AI


100 M€
24h/24
320 m²
96 tonnes
2400 KW
Warm Water
40 ➡ 32°C
4000 MWh/year Heat Recovery
« Avec Jean Perrin, nous créâmes le Centre national de la recherche scientifique »
Jean Zay, 29 avril 1942, Souvenirs et solitude
54800 cores
442 To
3700 GPUs
4.6 Pio
39 Pio
50 Pio
FP64
126 PFlop/s
BF/FP16
2.88 EFlop/s

Before Jean Zay

- Manual installations
- Recipes stored on a wiki
- No compiler/MPI consistency
- Environment Modules 3



Gaussian install ada

Droits d'accès

L'IDRIS possède une site de site. Les utilisateurs doive

Le lien vers la procédure détaillée : [[1]]

Installation

Pire usine à gaz de tous les temps.

Passer en csh
Modifier les fichiers /bsd : set-mflags, setup-make,
setenv g09root PATH
source /bsd/g09.login
nohup /bsd/bldg09 >&make.log &

Sources à modifier :

getden.f
ou
denget.f

Une fonction écrase la constante IMeth
Le problème a été remonté à Gaussian

Software deployment: Spack

- **S**upercomputer **PACK**age manager
- Package manager « from source »
- Developed at Lawrence Livermore National Laboratory since 2013
- Used in production at IDRIS since Summer 2019

Spack at IDRIS

- Currently not exposed to the users
- Pretty standard configuration:
  - Install directories layout
  - Default compilers
  - Default providers for MPI, BLAS, LAPACK, etc
  - Few external packages used
- Custom integration with Environment Modules 4

Environment Modules 4

- New developer: X. Delaruelle (CEA TGCC)
- Lot of fixes and optimizations
- New features:
  - Numerous usability improvements
  - **Automatic dependencies handling**
  - ...

Automatic dependencies handling

- Our goal: ensure a consistent environment
- Handled prerequisites:
  - Compiler
  - MPI library
  - CUDA
- Installation chosen based on the loaded environment
- Default environment used if none loaded

What about AI software stack?

- Now well integrated in Spack, no “environment” at the time
- New AI community:
  - Conda addicts
  - Not huge fans of not being root

  ⇒ Let’s use Conda
- In fact: Conda + Spack + manual installs :(

Spack: Conclusion

- No regret
- More consistent environment = less issues
- Contributions to Spack recipes
- Future?
  - Update our Spack install...
  - Make Spack available to users
  - Use Spack environments for AI stacks?

Containers at IDRIS

- No experience with containerization before Jean Zay
- Lots of constraints
  - Support for distributed filesystems like Lustre
  - Support for Omni-Path and Infiniband interconnects
  - Support for NVIDIA GPUs
  - Support for Slurm scheduling managers
  - Security constraints (no sudo, no setuid-bit if possible…)

Time to choose

- Initial choice: Podman
  - Made of sysadmins and security officers happy
  - Not compatible with GPFS/Lustre
- Alternative choice: **Singularity**
  - Widely used for HPC
  - Compatible with all our needs and constraints

Singularity at IDRIS

- Currently Singularity CE 4.0.2
- Setuid bit: default mode, installed by root
- SINGULARITY_ALLOWED_DIR used
- User namespace?

  ⇒ Forced the « sandbox » mode at the time:
  - Uncompressed image = 10-100k temporary files
  - Not acceptable on a distributed filesystem

Unavailable features

- Building images from a definition file
- Execution on the login nodes
- « Sandbox » mode
- Network virtualisation
- Fake root

Singularity: Conclusion

- Currently not that used
- Building a “good” container is not easy…
- Future?
  - Singularity CE or Apptainer?
  - Use Unprivileged User namespace with FUSE?

Thank you for your attention!

Comparative study of install tools

- To prepare for Jean Zay's installation
- Features comparison ⇒ Spack and EasyBuild stood out
  - Python based
  - Targetted for HPC with users in Europe
  - Environment Modules compatibility

  ⇒ Acceptability for both our team and the users
- Tested on our Power8+NVIDIA GPU prototype Ouessant

Outcomes of the tests

- EasyBuild:
  - Lot of files (toolchains, « easyconfig » files)
  - Perceived as complex

- Spack: easy win
  - Lot of enthusiasm for the specification language
  - Perceived as simpler to use

Renforcement de la sécurité

- « Protection du potentiel scientifique et technique de la nation »
- IDRIS = Zone à Régime Restreint physique et virtuelle
- Installation de produits = administration, même sans root
- Login avec mot de passe partagé plus acceptable

Renforcement de la sécurité

- Nouvelles contraintes en vigueur depuis Jean Zay :
  - Machine dédiée aux installations
  - Système de fichiers monté avec accès minimaux
  - Passage par un bastion avec journalisation
  - Authentification à plusieurs facteurs

- Un peu d'inquiétude mais finalement transparent !

Spack en équipe ?

- Principale question : un login ou plusieurs ?
- Système de verrouillage perfectionné
  ⇒ pas de soucis avec les accès concurrents
- Problème : garantir l'accès en écriture à tous
  - Répertoires d'installation : plutôt bien gérés par Spack
  - Répertoires et fichiers annexes (ex : .pyc) : pas si sûr…

  ⇒ Choix mono-login compatible sécurité : « sudo -i -u »

Mise à disposition des produits ?

- Utilisateurs habitués à Environment Modules
- Explosion combinatoire des modules !
- Étude de plusieurs solutions :
  - Modules classiques avec suffixes : netcdf/4.7.4-gcc-8.3.1
  - Approche hiérarchique : Lmod
    ⇒ charger un module en débloque d'autres
  - Approche pseudo-hiérarchique : Environment Modules 4
    ⇒ gestion automatique des dépendances

Gestion automatique des dépendances

- Chargement automatique des prérequis

```
#%Module1.0

module-whatis "App"

prereq lib1
prereq lib2a lib2b
```

```
$ module load app
Loading app
  Loading requirement: lib1 lib2a

$ module unload app
Unloading app
  Unloading useless requirement: lib1 lib2a

$ module load lib2b
$ module load app
Loading app
  Loading requirement: lib1

$ module unload app
Unloading app
  Unloading useless requirement: lib1
```

Intégration avec Spack

- Modules générés pour Environment Modules 3
  ⇒ pas de support de l'approche pseudo-hiérarchique
- Développement d'une extension pour Spack :
  - Changement de modèle
    ⇒ « 1 module pour 1 installation » à 1 pour N
  - Code pas propre mais fonctionnel
  - Volonté d'upstreamer mais manque de temps...

Exemples

```
$ module help netcdf-c/4.7.4
-------------------------------------------------------------------
Module Specific Help for /.../netcdf-c/4.7.4:

NetCDF (network Common Data Form) is a set of software libraries and
machine-independent data formats that support the creation, access, and
sharing of array-oriented scientific data. This is the C distribution.

Available software environment(s):
- intel-compilers/19.1.2
- intel-compilers/19.1.1
- intel-compilers/19.0.4
- nvidia-compilers/20.7
- pgi/20.4

If you want to use this module with another software environment,
please contact the support team.
-------------------------------------------------------------------
```

Exemples

```
$ module help netcdf-c/4.7.4-mpi
----------------------------------------------------------------
Module Specific Help for /.../netcdf-c/4.7.4-mpi:

NetCDF (network Common Data Form) is a set of software libraries and
machine-independent data formats that support the creation, access, and
sharing of array-oriented scientific data. This is the C distribution.

Available software environment(s):
- intel-compilers/19.1.2 intel-mpi/2019.8
- intel-compilers/19.1.1 intel-mpi/2019.7
- intel-compilers/19.1.1 openmpi/4.0.5
- intel-compilers/19.1.1 openmpi/4.0.4
- intel-compilers/19.0.4 intel-mpi/2019.4
- pgi/20.4 openmpi/4.0.4

If you want to use this module with another software environment,
please contact the support team.
----------------------------------------------------------------
```

Exemples

```
$ module load netcdf-c/4.7.4
Loading netcdf-c/4.7.4
  Loading requirement: intel-compilers/19.1.2
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.2-6qb4f4s4fszzyttlwbvvwui7iztx3bkg/bin/ncdump


$ module purge
$ module load nvidia-compilers/20.7
$ module load netcdf-c/4.7.4
$ which ncdump
/.../netcdf-c/4.7.4/nvhpc-20.7-5dwvxx727x7b62pcfatwpk7dy7vjsq57/bin/ncdump


$ module purge
$ module load netcdf-c/4.7.4-mpi
Loading netcdf-c/4.7.4-mpi
  Loading requirement: intel-compilers/19.1.2 intel-mpi/2019.8
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.2-55jlam4hjbzwj2kh2rdcw4lcd2qpnpqq/bin/ncdump
```

Exemples

```
$ module purge
$ module load intel-compilers/19.1.1 intel-mpi/2019.7 netcdf-c/4.7.4-mpi
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.1-kd7sjsicnhzgdmdbuvmol34szbhn7ytb/bin/ncdump


$ module purge
$ module load intel-compilers/19.1.1 openmpi/4.0.5 netcdf-c/4.7.4-mpi
$ which ncdump
/.../netcdf-c/4.7.4/intel-19.1.1-6bwguvlp26lyegv5lvjgnkmfg64lyzn7/bin/ncdump


$ module purge
$ module load intel-compilers/19.1.2 openmpi/4.0.5 netcdf-c/4.7.4-mpi
Loading intel-mpi/2019.8
  ERROR: intel-mpi/2019.8 cannot be loaded due to a conflict.
    HINT: Might try "module unload openmpi" first.

Loading netcdf-c/4.7.4-mpi
  ERROR: Load of requirement intel-mpi/2019.8 failed
```

Limitations

- Gestion des variantes :
  - Préfixes conservés pour MPI/CUDA
  - Approche alternative : « flavors »
- Gestion des compilateurs :
  - Pas d’exclusivité possible
  - Priorisation mais des problèmes potentiels

CI à l'IDRIS

- ZRR : traçabilité des actions

  ⇒ c'est compliqué…
- Une tentative avec Jenkins qui n'a pas dépassé le stade de prototype
- En pratique : zone grise...