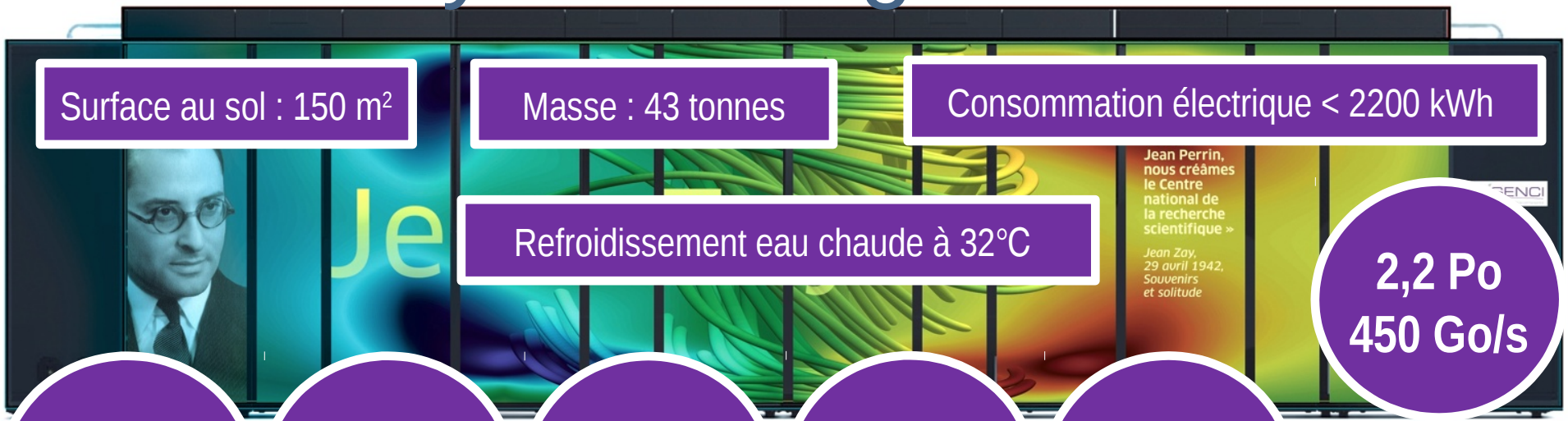
INSTITUT DU
DÉVELOPPEMENT ET DES
RESSOURCES EN
INFORMATIQUE
SCIENTIFIQUE

www.idris.fr

Conteneurisation à l'IDRIS : Retour d'expérience

Jean Zay : convergence HPC/IA



28 Pflop/s
Puissance crête cumulée

2696
GPU NVIDIA V100 16-32 Go

86344 cœurs
de calcul Intel Cascade Lake 2.5 GHz

427 To
de mémoire

100 Gb/s
Réseau d'interconnexion Omni-Path

2,2 Po 450 Go/s
Full Flash

35 Po 150 Go/s

L’IDRIS et les conteneurs

- Historiquement : communauté HPC
  - Maîtrise de la compilation
  - Quelques utilisateurs intéressés pour la portabilité
- Ouverture à la communauté IA :
  - Habitudes très différentes au niveau logiciel
  - Attrait nettement plus marqué pour les conteneurs
- Une seule expérience : prototype Ouessant (IBM Power8+/V100)

Des besoins spécifiques

- Support des applications HPC et IA
- Support du système de fichiers parallèle GPFS
- Support du réseau d'interconnexion Omni-Path
- Support des GPU NVIDIA
- Intégration avec le gestionnaire de travaux Slurm

⇒ Pas une configuration typique...

Contraintes de sécurité

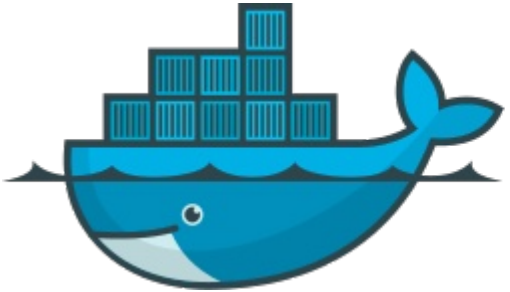
- « Protection du potentiel scientifique et technique de la nation »
- IDRIS = Zone à Régime Restreint physique et virtuelle

⇒ Attention particulière à la sécurité

⇒ *sudo*, *setuid-bit* interdits ou limités au maximum

⇒ Pas les contraintes habituelles des usages historiques des conteneurs (déploiement de services)

Docker



- Outil historique
  ⇒ quasi-synonyme de « conteneurs » pour les utilisateurs
- Répond aux besoins au niveau technique
- Nécessite des droits particuliers (sudo) même pour démarrer une image

  ⇒ Inenvisageable sur Jean Zay

Shifter

- Développé par le NERSC
  ⇒ orientation HPC
- Utilise principalement des images Docker
- Exécutable privilégié (setuid bit)
- Support des GPU via nvidia-docker
  ⇒ déprécié et adhérence à Docker
- Pas de version stable publiée depuis 2018

Singularity 

- Lien historique avec Lawrence Berkeley National Lab
  ⇒ orientation HPC
- Images au format SIF avec compatibilité Docker
- Plusieurs modes d'installation : root-less possible
- Construction d'image pouvant nécessiter d'être root
- Exécution sans privilège particulier

Podman

- Compatibilité complète avec Docker (alias docker=podman)
- Nécessite un kernel Linux récent
- Remplace Docker dans la distribution Red Hat 8
- Pas de *daemon*
- Pas de besoins de privilèges particuliers

L’heure du choix

- Choix initial : Podman
  - Plaisait beaucoup à nos équipes système et sécurité
  - Abandonné pour cause d’incompatibilité avec GPFS
- Choix alternatif : **Singularity**
  - Probablement le plus utilisé pour le HPC
  - Compatible avec nos besoins et contraintes

Modes de fonctionnement

- Setuid bit : mode par défaut, installation par root
- User namespace :
  - Nécessite un kernel récent
  - Aucun privilège particulier requis

⇒ Tentant mais rend le mode « sandbox » obligatoire :

  - Image décompressée = 10-100k fichiers temporaires
  - Pas acceptable sur un système de fichiers parallèle

Configuration mise en place

- Singularity 3.6.4
  ⇒ une seule version : suivi des failles de sécurité facilité
- Compromis fonctionnalités/contraintes système/sécurité :
  - Mode privilégié (setuid bit)
  - Contrainte de sécurité supplémentaire
    ⇒ Restriction sur l'emplacement des images exécutées

```
$ singularity run test.sif
FATAL: singularity image is not in an allowed configured path
```

$SINGULARITY_ALLOWED_DIR

- Espace dédié pour les images exécutables
- Propre à chaque utilisateur
- Jusqu'à 20 conteneurs sans limite de taille
- Accès restreint en écriture
  ⇒ passage obligatoire par une commande spécifique

idrcontmgr

- Permet d'imposer des vérifications de sécurité complémentaires sur les images
- Exemple : Scanner d'images comme Stools (Clair)
- Usage :

```
$ idrcontmgr cp test.sif
1 file copied.

$ idrcontmgr ls
test.sif
1 file(s) found.

$ ls -l $SINGULARITY_ALLOWED_DIR
-rw-r----- 1 cont cont 6762291200 janv. 15 18:24 test.sif

$ idrcontmgr rm test.sif
File 'test.sif' was succesfully deleted.
```

Fonctionnalités disponibles

- Construction d'images depuis un dépôt
- Exécution d'une image sur un nœud de calcul :
  - singularity run / exec
  - singularity shell possible en interactif
- Montage de dossiers dans le conteneur

```
$ singularity exec --bind /chemin/sur/jz:/chemin/dans/cont \
                   $SINGULARITY_ALLOWED_DIR/test.cif       \
                   ls /chemin/dans/cont
```

Fonctionnalités non disponibles

- Construction d'images à partir d'un fichier de définition
- Exécution sur les frontales
- Exécution en mode « sandbox »
- Virtualisation de réseau
- Mode « fakeroot »

⇒ Pour l'instant pas un problème

Utilisation des GPU

- Option `--nv` :
  - Rend visible les périphériques `/dev/nvidiaX`
  - Monte les bibliothèques/exécutables liés au driver (`nvidia-container-cli` ou `etc/singularity/nvbliblist`)
  - Positionne le `LD_LIBRARY_PATH` en conséquence
- Runtime CUDA à intégrer à l'image du conteneur

Utilisation de MPI

- Exécuter MPI à l'intérieur du conteneur :

```
$ singularity exec [...] mpirun –n 40 ./app_mpi.exe
```

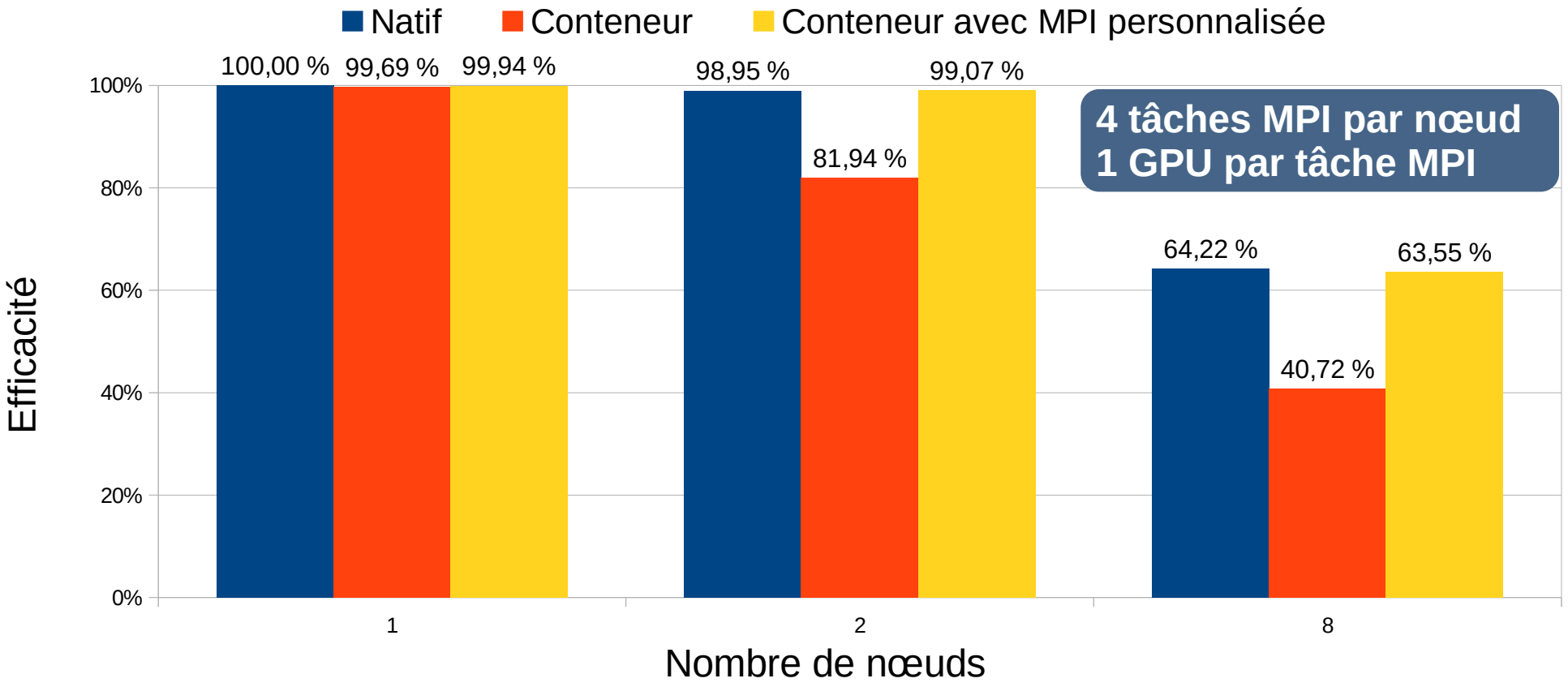
  - Fonctionne en mono-nœud

- Exécuter MPI à l'extérieur du conteneur :

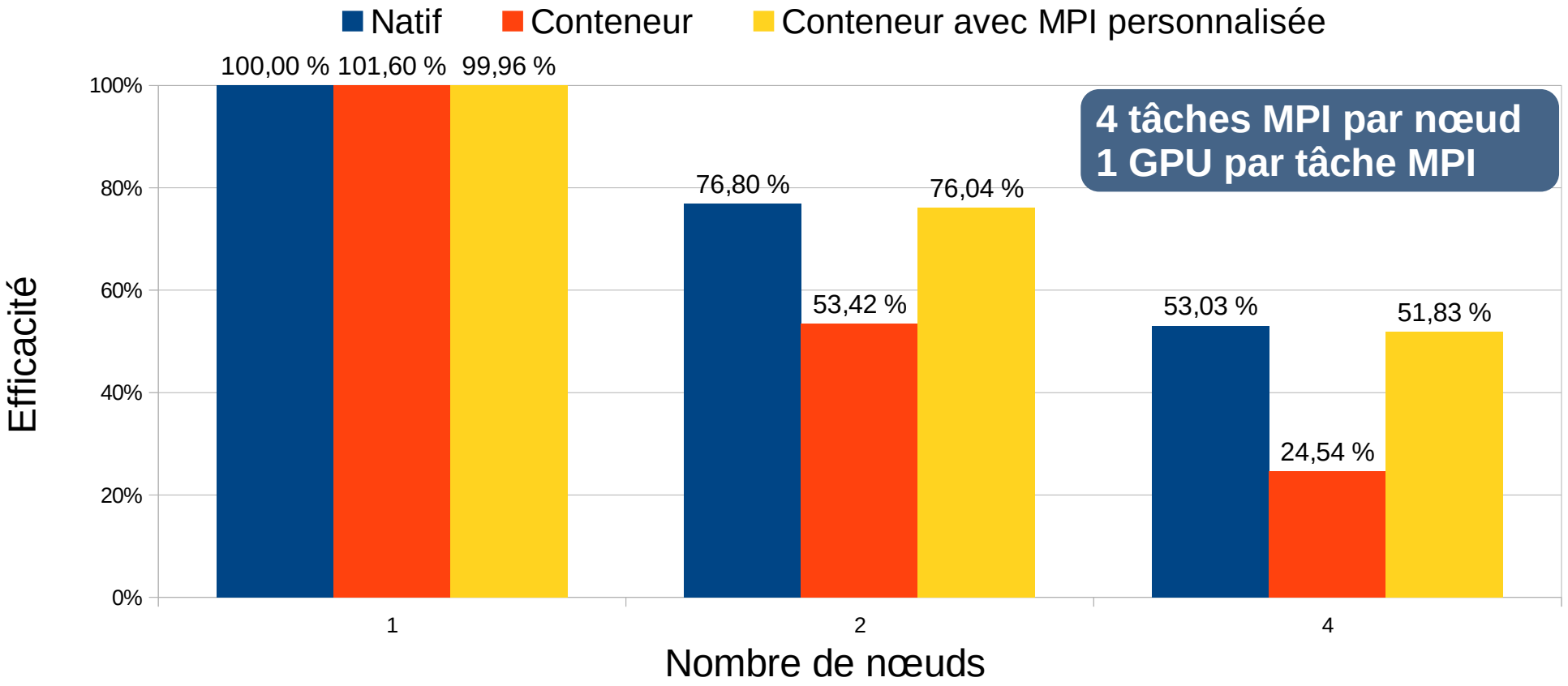
```
$ srun/mpirun –n 40 singularity exec [...] ./app_mpi.exe
```

  - Utiliser deux bibliothèques compatibles
  - Monter une bibliothèque externe compatible avec la bibliothèque utilisée pour compiler

Performances : Heat3D



Performances : DYNAMICO



Conclusion

- Singularity : probablement le plus mature pour du HPC
- Compromis entre contraintes techniques et sécurité
- Possibilité de vérifications de sécurité sur les images
- Performances MPI multi-nœuds : pas si simple