UDOCKER

BE ANYWHERE

Mário David <david@lip.pt>
Jorge Gomes <jorge@lip.pt>

INDIGO - DataCloud
Better Software for Better Science

Cofinanciado por:
COMPETE 2020
PROGRAMA OPERACIONAL COMPETITIVIDADE E INTERNACIONALIZAÇÃO
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Europeu de Desenvolvimento Regional
Lisb@20 20
FCT Fundação para a Ciência e a Tecnologia

Scientific computing and containers

UDOCKER

Running applications across infrastructures may require considerable effort

- **Computers**
  - Several computing systems
  - Laptops, Desktops, Farms, Cloud, HPC
- **OSes**
  - Several operating systems
  - Linux flavors, Distribution versions
- **Environments**
  - Specific computing environments
  - Compilers, Libraries, Customizations
- **Applications**
  - Multiple applications often combined
  - Portability, Maintainability, Reproducibility

Need a consistent portable way of running applications

Scientific computing and containers

UDOCKER

Running applications across infrastructures may require considerable effort

- **Computers**
  - Several computing systems
  - Laptops, Desktops, Farms, Cloud, HPC
- **OSes**
  - Several operating systems
  - Linux flavors, Distribution versions
- **Environments**
  - Specific computing environments
  - Compilers, Libraries, Customizations
- **Applications**
  - Multiple applications often combined
  - Portability, Maintainability, Reproducibility

We started to look at it in 2013

Need a consistent portable way of running applications

Containers for batch processing

UDOCKER
BE ANYWHERE

- Challenges of batch systems?
  - Integrate it with the batch system (how to start/stop etc) ?
  - Respect batch system policies (such as quotas/limits) ?
  - Respect batch system actions (job delete/kill) ?
  - Collect accounting ?

- Can we execute in a more basic way?
  - Can we download container images ?
  - Can we run without a layered filesystem ?
  - Can we run them as normal user ?
  - Can we still enforce container metadata ?

udocker

- Run applications encapsulated in docker containers:
  - without using docker
  - without using (root) privileges
  - without system administrators intervention
  - without additional system software
  - Does not require Linux namespaces

- Run:
  - as a normal user
  - with the normal process controls and accounting
  - in interactive or batch systems

udocker

udocker is open source

Developed under the **Indigo-Datacloud** and **DEEP Hybrid-Datacloud** projects

https://github.com/indigo-dc/udocker

- https://github.com/indigo-dc/udocker/tree/master
- https://github.com/indigo-dc/udocker/tree/devel

Documentation:
https://github.com/indigo-dc/udocker/tree/master/doc

udocker: install from github

```
$  curl
   https://raw.githubusercontent.com/indigo-dc/udocker/master/udocker.py
   >  udocker

$  chmod  u+rx  udocker
$  ./udocker  install
```

Does not require compilation or system installation
Tools are delivered statically compiled

udocker: pull images from repository

UDOCKER
BE ANYWHERE

```
$ udocker pull ubuntu:14.04
```

Search for names and tags at: https://hub.docker.com/

```
Downloading layer: sha256:bae382666908fd87a3a3646d7eb7176fa42226027d3256cac38ee0b79bdb0491
Downloading layer: sha256:f1ddd5e846a849fff877e4d61dc1002ca5d51de8521cced522e9503312b4c4e7
Downloading layer: sha256:90d12f864ab9d4cfe6475fc7ba508327c26d3d624344d6584a1fd860c3f0fefa
Downloading layer: sha256:a57ea72e31769e58f0c36db12d25683eba8fa14aaab0518729f28b3766b01112
Downloading layer: sha256:783a14252520746e3f7fee937b5f14ac1a84ef248ea0b1343d8b58b96df3fa9f
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
```

udocker: create container from image

```
$ udocker create --name=ub14 ubuntu:14.04
```

container name is an alias to container ID

container-id

```
9fe2f9e7-ce37-3be5-b12d-829a3236d2a6
```

udocker: run container

UDOCKER
BE ANYWHERE

```
$ udocker run ub14
```

udocker respects container metadata, if the container has a default cmd to run it will be run otherwise starts a shell

```
 ****************************************************************************
 *                                                                          *
 *               STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6              *
 *                                                                          *
 ****************************************************************************
 executing: bash
root@nbjorge:/# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.5 LTS"
root@nbjorge:/# apt-get install firefox
```

root emulation

udocker: run container as yourself

```
$ udocker run --user=jorge -v /home/jorge \
  -e HOME=/home/jorge --workdir=/home/jorge ub14
```

```
 Warning: non-existing user will be created
 ****************************************************************************
 *                                                                          *
 *                STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6             *
 *                                                                          *
 ****************************************************************************
 executing: bash
jorge@nbjorge:~$ id
uid=1000(jorge) gid=1000(jorge) groups=1000(jorge),10(uucp)
jorge@nbjorge:~$ pwd
/home/jorge
```

udocker: run commands in the prompt

```
$ udocker run --user=jorge --bindhome \
    --hostauth ub14 /bin/bash <<EOF
id; pwd
EOF
```

```
**************************************************************************
*                                                                        *
*              STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6             *
*                                                                        *
**************************************************************************
executing: bash
uid=1000(jorge) gid=1000(jorge) groups=1000(jorge),10(uucp)
/home/jorge
```

udocker

- Run time to execute docker containers:
  - search
  - pull
  - images
  - create
  - rmi
  - ps
  - rm
  - run
  - login
  - logout
  - load
  - save
  - import
  - export
  - setup
  - clone
  - verify
  - Inspect
  - mkrepo

UDOCKER
BE ANYWHERE

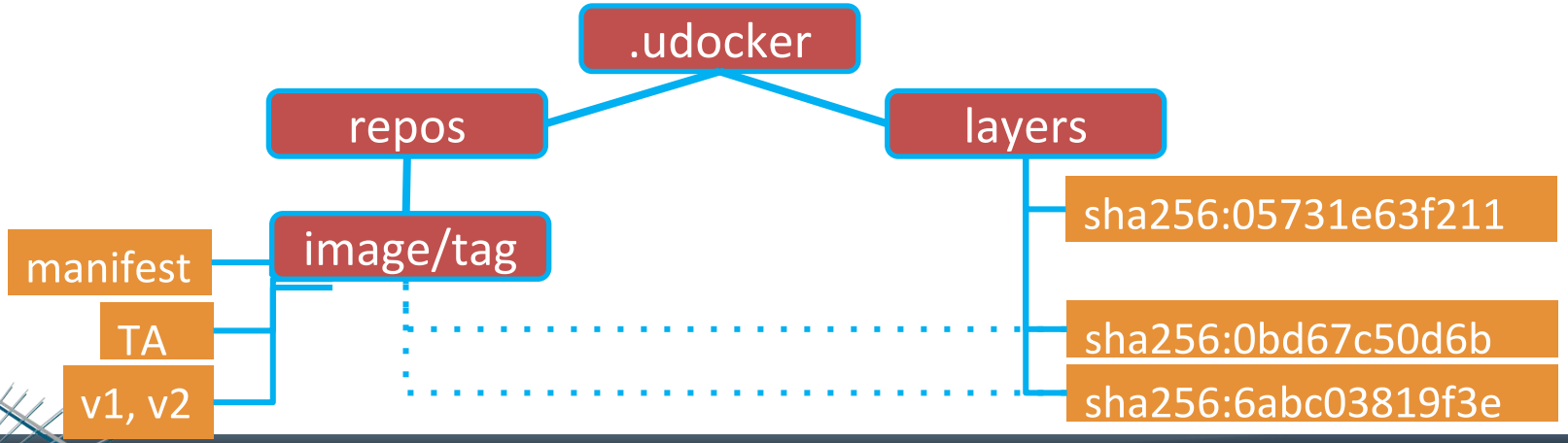
udocker
How does it work ...

udocker

- **Implemented**
  - python, C, C++, go
- **Can run:**
  - CentOS 6, CentOS 7, Fedora >= 23
  - Ubuntu 14.04, Ubuntu 16.04
  - Any distro that supports python 2.6 and 2.7
- **Components:**
  - Command line interface docker like
  - Pull of containers from Docker Hub
  - Local repository of images and containers
  - Execution of containers with modular engines

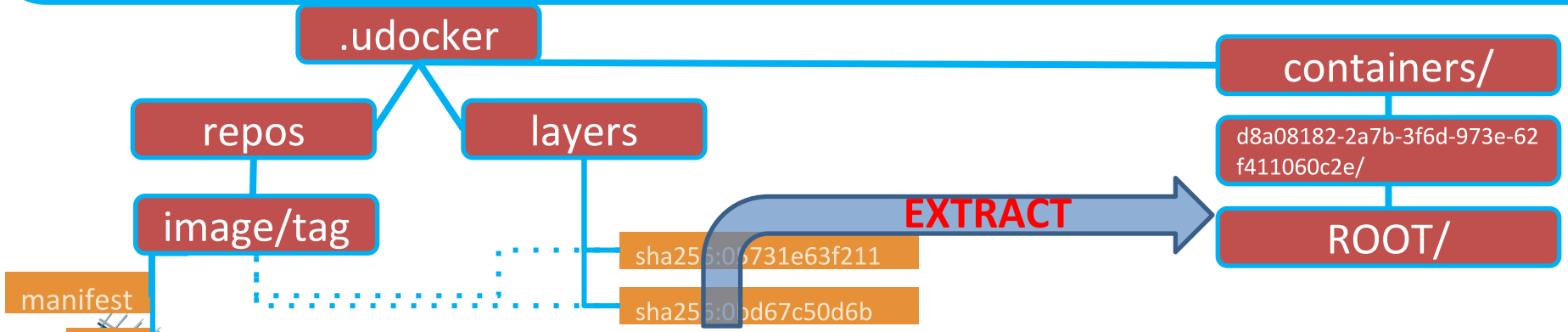
Udocker: pull - Images

- Layers and metadata are pulled with DockerHub REST API
- Image metadata is interpreted to identify the layers
- Layers are stored in the use home directory under ~/.udocker/layers so that can be shared by multiple images



Udocker: create - Containers

- Are produced from the layers by flattening them
- Each layer is extracted on top of the previous
- Whiteouts are respected, protections are changed
- The obtained directory trees are stored under ~/.udocker/containers in the user home directory

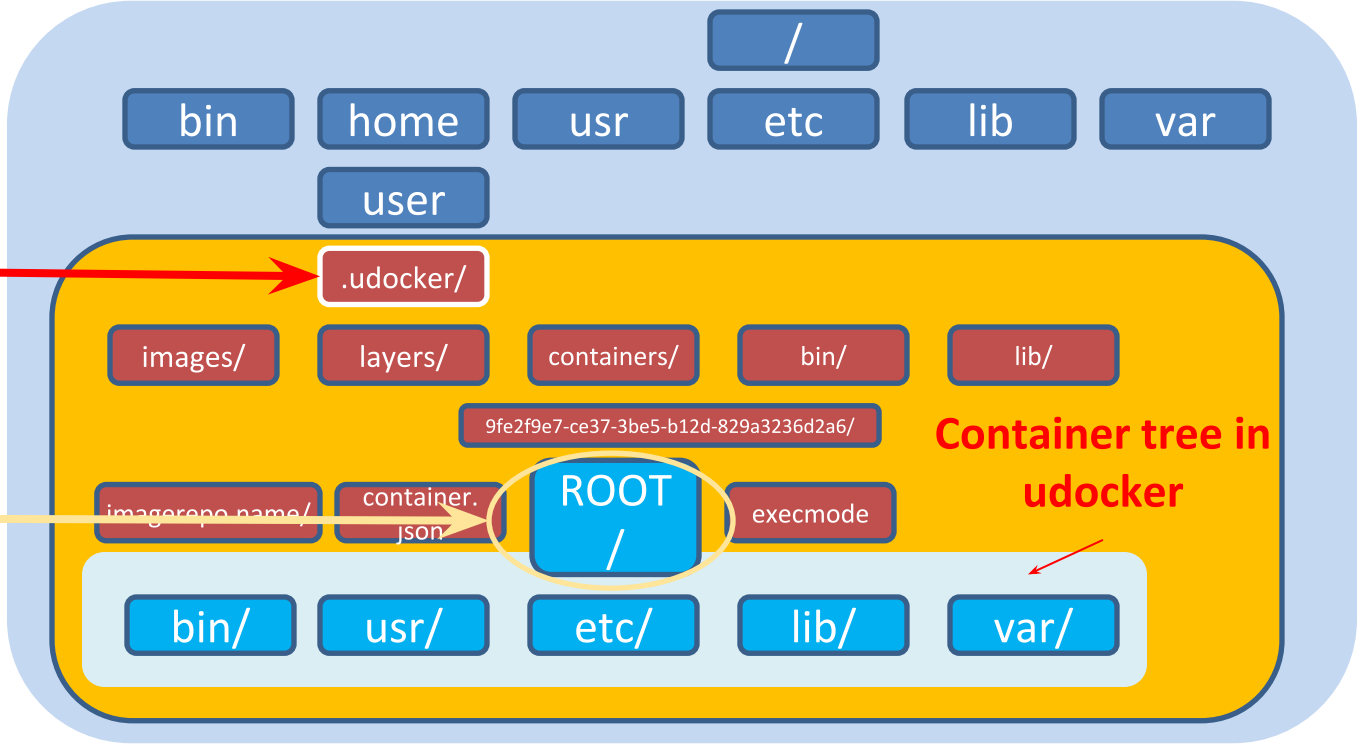


udocker: run - Container

- Execution
- chroot-like

udocker directory tree
$HOME/.udocker

chroot to this directory becomes the new root for container processes



udocker: Execution engines

- udocker supports several techniques to achieve the equivalent to a chroot without using privileges
  - They are selected per container id via execution modes

Mode	Base	Description
P1	PRoot	PTRACE accelerated (with SECCOMP filtering) DEFAULT
P2	PRoot	PTRACE non-accelerated (without SECCOMP filtering)
R1	runC	rootless unprivileged using user namespaces
F1	Fakechroot	with loader as argument and LD_LIBRARY_PATH
F2	Fakechroot	with modified loader, loader as argument and LD_LIBRARY_PATH
F3	Fakechroot	modified loader and ELF headers of binaries + libs changed
F4	Fakechroot	modified loader and ELF headers dynamically changed
S1	Singularity	where locally installed using chroot or user namespaces

udocker: PRoot engine

- PRoot uses PTRACE to intercept system calls
- Pathnames are modified before the call
  - To expand container pathnames into host pathnames
- Pathnames are modified after the call
  - To shrink host pathnames to container pathnames

udocker: PRoot engine (P1 and P2)

- The P1 mode uses PTRACE + SECCOMP filtering, to limit the interception to the set of calls that manipulate pathnames
  - We developed code to make it work on recent kernels
  - P1 is the udocker default mode

- The P2 mode uses only PTRACE therefore tracing all calls

- The impact of tracing depends on the system call frequency

udocker: runC engine (R1)

- runC is a tool to spawn containers according to the Open Containers Initiative (OCI) specification
  - In a very recent release 1.0 candidate 3, runC supports unprivileged namespaces using the user namespace
  - Unprivileged namespaces have many limitations but allow execution in a container Docker like environment
  - Only run as root is supported
  - Available devices are limited
- We added conversion of Docker metadata to OCI
- udocker can produce an OCI spec and run the containers with runC transparently

udocker: Fakechroot engine

- Fakechroot is a library to provide chroot-like behaviour
- Uses the Linux loader LD_PRELOAD mechanism to:
  - intercept library calls that manipulate pathnames
  - change the pathnames similarly to PRoot
- It was conceived to support debootstrap in debian
- The OS in the host and in the chroot must be the same
  - as the loader inside the chroot will by default load libraries from the host system directories
  - the loaders are statically linked and the pathnames inside are absolute and non changeable

udocker: Fakechroot engine

- The loaders search for libraries:
  - If the pathname has a / they are directly loaded
  - If the pathname does not contain / (no directory specified) a search path or location can be obtained from:
    1. DT RPATH dynamic section attribute of the ELF executable
    2. LD LIBRARY PATH environment variable
    3. DT RUNPATH dynamic section attribute of the ELF executable
    4. cache file /etc/ld.so.cache
    5. default paths such as /lib64, /usr/lib64, /lib, /usr/lib
- The location of the loader itself is encoded in the executables ELF header

udocker: Fakechroot engine (F1)

- The loader is encoded in the ELF header of executable
  - is the executable that loads libraries and calls the actual executable
  - also act as library providing functions and symbols

- Is essential that executables in the container are run with the loader inside of the container instead of the host loader

udocker: Fakechroot engine (F1)

- The mode F1 enforces the loader:
  - passes it as 1st argument in exec* and similar calls shifting argv
  - the loader starts first gets the executable pathname and its arguments from argv and launches it
  - Enforcement of locations is performed by filling in LD_LIBRARY_PATH with the library locations in the container and also extracted from the container ld.so.cache

udocker: Fakechroot engine (F2)

- The mode F2 changes the loader binary within the container:
  - A copy of the container loader is made
  - The loader binary is then edited by udocker
  - The loading from host locations /lib, /lib64 etc is disabled
  - The loading using the host ld.so.cache is disabled
  - LD_LIBRARY_PATH is renamed to LD_LIBRARY_REAL

udocker: Fakechroot engine (F2)

- Upon execution
  - Invocation is performed as in mode F1
  - The LD_LIBRARY_REAL is filled with library locations from the container and its ld.so.cache
  - Changes made by the user to LD_LIBRARY_PATH are intercepted and pathnames adjusted to container locations and inserted in LD_LIBRARY_REAL

udocker: Fakechroot engine (F3 and F4)

- The mode F3 changes binaries both executables and libraries
  - The PatchELF tool was heavily modified to enable easier change of
    - Loader location in ELF headers of executables
    - Library path locations inside executables and libraries
- When modes F3 or F4 are selected the executables and libraries are edited
  - The loader location is change to point to the container
  - The libraries location if absolute are changed to point to container
  - The libraries search paths inside the binaries are changed to point to container locations

udocker: Fakechroot engine (F3 and F4)

- The loader no longer needs to be passed as first argument
- The libraries are always fetched from container locations
- The LD_LIBRARY_REAL continues to be used in F3 and F4
- The mode F4 adds dynamic editing of executables and libraries
- This is useful with libraries or executables are added to the container or created as result of a compilation

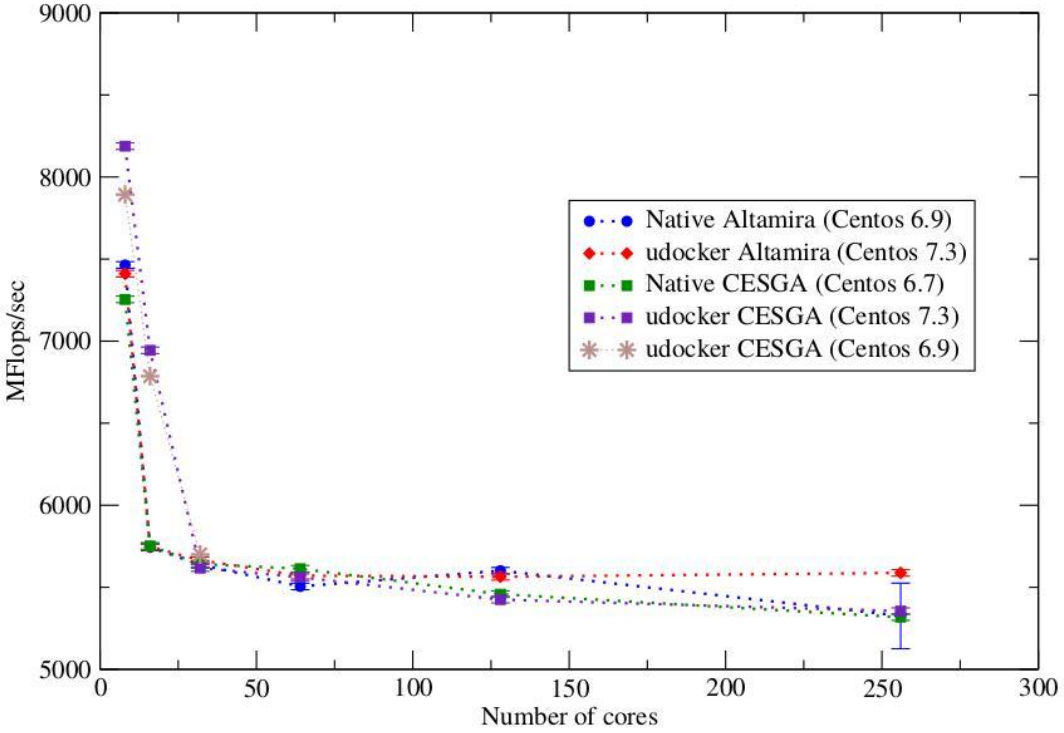
udocker: Fakechroot engine (F3 and F4)

- Containers in modes F3 and F4 cannot be transparently moved across different systems:
  - the absolute pathnames to the container locations will likely differ.
  - In this case convert first to another mode before transfer
  - or at arrival use: “`setup --execmode=Fn --force`”

UDOCKER
BE ANYWHERE

udocker
Running applications ...

udocker & Lattice QCD



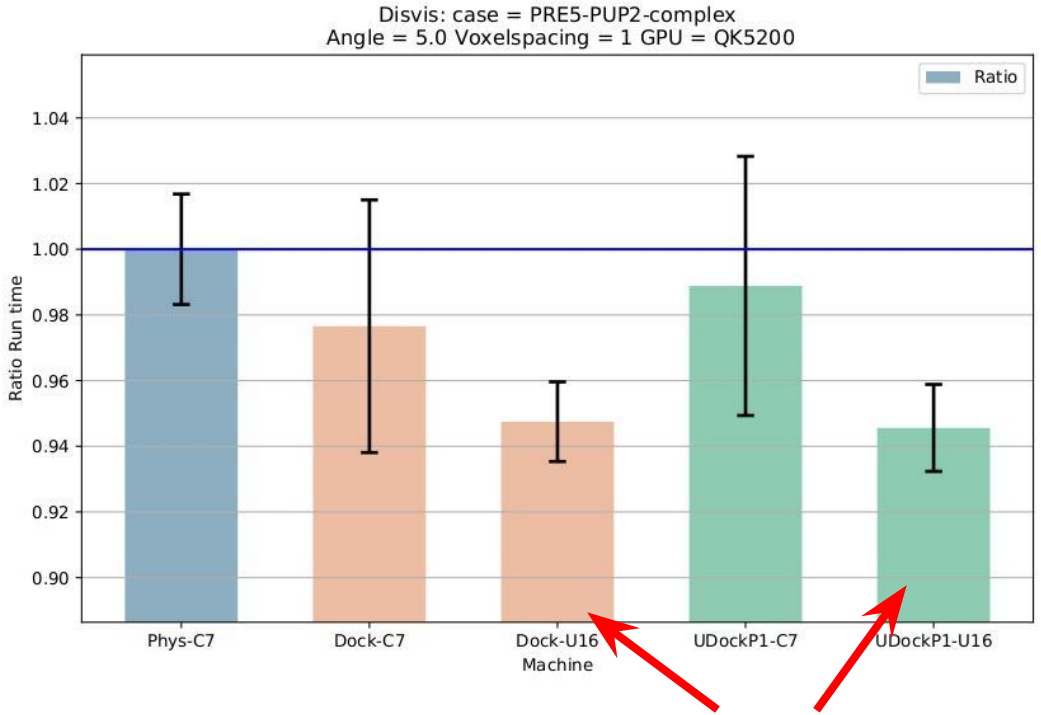
OpenQCD is a very advanced code to run lattice simulations

Scaling performance as a function of the cores for the computation of application of the Dirac operator to a spinor field.

Using OpenMPI

udocker in P1 mode

udocker & Biomolecular complexes



DisVis is being used in production with udocker

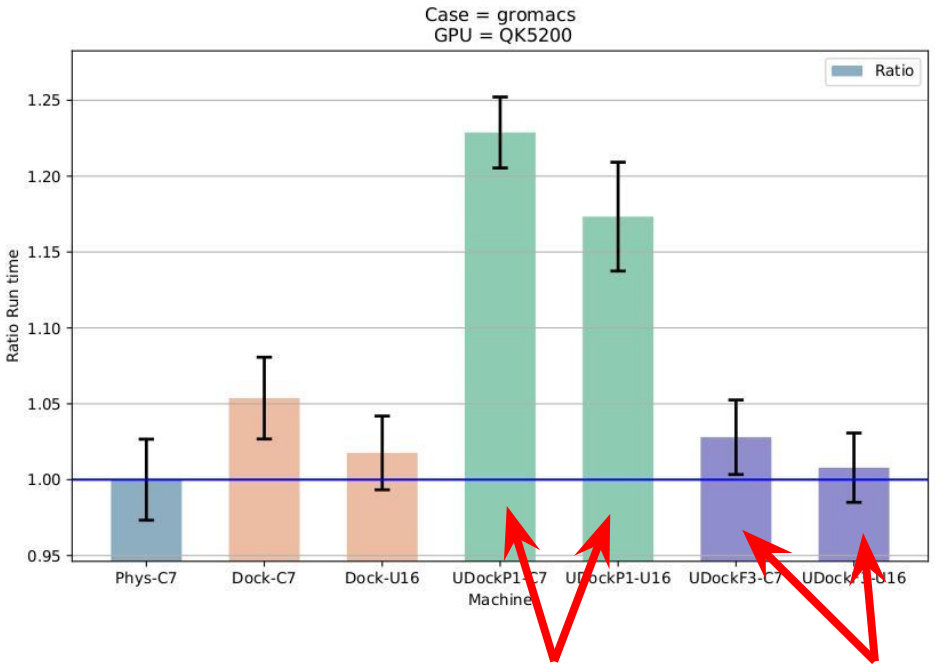
Performance with docker and udocker are the same and very similar to the host.

Using OpenCL and NVIDIA GPUs

Better performance with Ubuntu 16 container

udocker in P1 mode

udocker & Molecular dynamics



Gromacs is widely used both in biochemical and non-biochemical systems.

udocker P mode have lower performance
udocker F mode same as Docker.

Using CUDA and OpenMP

PTRACE **SHARED LIB CALL**

udocker in P1 mode
udocker in F3 mode

udocker & Phenomenology

Performance Degradation

	Compiling	Running
HOST	0%	0%
DOCKER	10%	1.0%
udocker	7%	1.3%
VirtualBox	15%	1.6%
KVM	5%	2.6%

udocker in P1 mode

MasterCode connects several complex codes. Hard to deploy.

Scanning through large parameter spaces. High Throughput Computing

C++, Fortran, many authors,

udocker & Phenomenology

```
export MASTERDIR=/gpfs/csic_users/userabc/mastercode
export UDOCKER_DIR=$MASTERDIR/.udocker

udocker.py run --hostauth \
    -v  /home/csic/cdi/ica/mcpp-master \
    -v  /home/csic/cdi/ica \
    -user=user001 \
    -w  /home/csic/cdi/ica/mcpp-master  mastercode \
    /bin/bash -c "pwd; ./udocker-mastercode.sh"
```

udocker
Next ...

udocker: Presently

- Current version - 1.1.3
- Run's with python 2.6 and 2.7
  - Centos6, Centos7, Ubuntu 14.04, 16.04 and 18.04 if with py2
    - Plus quite a few Fedoras, alpine
- You get it by "`wget`" or "`curl`" or "`git clone`"
  - Plus `rpm` and `deb`
- 1 python script - 7000+ LoC (plus the libs for exec engines)
  - (unit tests as well a single 7500+ LoC)

udocker: What's next

- Ongoing:
  - https://github.com/indigo-dc/udocker/tree/devel3
  - Modularization of udocker (and unit tests)
  - Porting to Python3
  - pip install - https://pypi.org

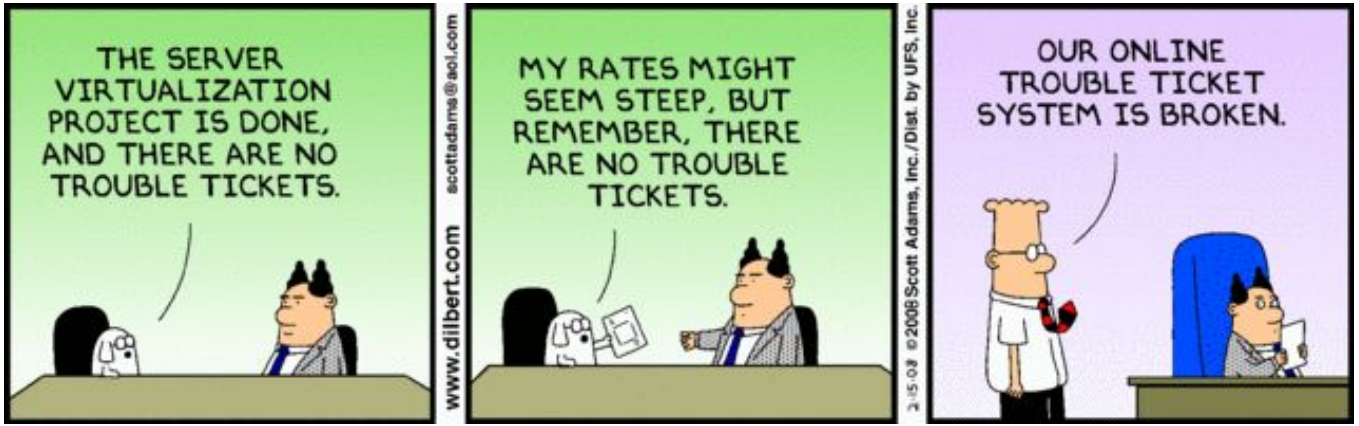
udocker: What's next

- Next
  - Increase automation for MPI/infiniband applications
    - OpenMPI and MPICH
  - Better translation of "volume" directories
  - Command line interface enhancements
  - Improve root emulation

Thank you

UDOCKER
BE ANYWHERE

https://github.com/indigo-dc/udocker



Backup

udocker: install from github

```
$ curl
  https://raw.githubusercontent.com/indigo-dc/udocker/master/udocker
  .py > udocker
$ chmod  u+rx  udocker
$ ./udocker  install
```


```
$ git  clone -b master https://github.com/indigo-dc/udocker.git

$ cd  udocker
$ chmod  u+rx  udocker
$ ./udocker  install
```

udocker: search images

```
$ udocker   search   ubuntu
```

```
NAME            OFFICIAL DESCRIPTION
ubuntu          [OK] Ubuntu is a Debian-based Linux operating system based on free
ubuntu-debootstrap [OK] debootstrap --variant=minbase --components=main,universe
ubuntu-upstart[OK] Upstart is an event-based replacement for the /sbin/init daemon
rastasheep/ubuntu-sshd ---- Dockerized SSH service, built on top of official Ubuntu images.
```

udocker: list containers

```
$ udocker ps
```

container-id

protected

write

alias/name

image

```
CONTAINER ID                          P M NAMES              IMAGE
9fe2f9e7-ce37-3be5-b12d-829a3236d2a6  . W ['ub14']           ubuntu:14.04
5c7bd29b-7ab3-3d73-95f9-4438443aa6d6  . W ['myoffice']       msoffice:lastest
676eb77d-335e-3e9a-bf62-54ad08330b99  . W ['fedora_25']      fedora:25
c64afe05-adfa-39de-bf15-dcd45f284249  . W ['debianold']      debian:oldstable
7e76a4d7-d27e-3f09-a836-abb4ded0df34  . W ['ubuntu16', 'S']  ubuntu:16.10
9d12f52d-f0eb-34ae-9f0e-412b1f8f2639  . W ['f25']            fedora:25
```

udocker: duplicate a container

```
$ udocker   clone   --name=yy   ub14
```

cloned container-id

```
9fe2f9e7-ce37-3be5-b12d-829a3236d2a6
```

udocker: remove

remove container by alias or id

```
$ udocker    rm   ub14
$ udocker    rm   9fe2f9e7-ce37-3be5-b12d-829a3236d2a6
```

remove image

```
$ udocker    rmi   ubuntu:14.04
```

udocker: export and import as image

export to tarball

```
$ udocker   export  -o ub14.tar  ub14

$ udocker   import  ub14.tar   myub14:latest
```

import from tarball

new image name

- Only the container files are exported, metadata is lost
- This is interoperable with docker

udocker: export and import as container

export to tarball

```
$ udocker   export   -o ub14.tar   ub14

$ udocker   import   --tocontainer   --name=xx   ub14.tar
```

import from tarball to container

new container alias

- Only the container files are exported, metadata is lost
- Export is interoperable with docker

udocker: export and import as container

export clone

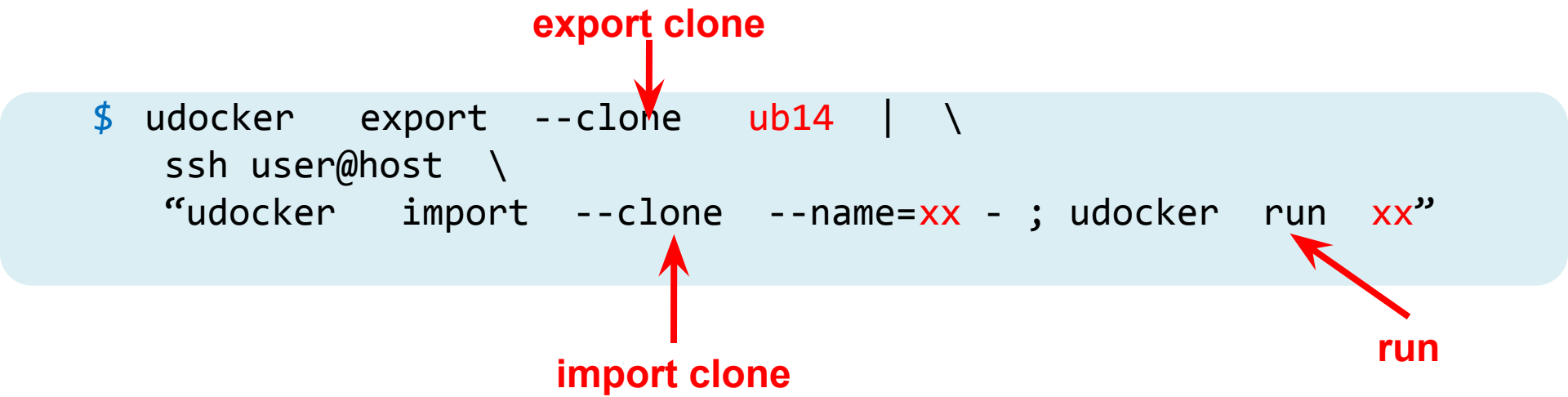
```
$ udocker   export  --clone  -o ub14.tar  ub14

$ udocker   import  --clone  --name=xx  ub14.tar
```

import clone

- Is imported as a container saving space and time
- Container metadata and execution mode are preserved
- This is NOT interoperable with docker

udocker: export and import as container



- Export and import across nodes
- Piping stdout to stdin and minimizing I/O

udocker: save and load images

save image with all layers and metadata

```
$ docker   save  -o  image.tar  centos:centos6

$ udocker   load  -i  image.tar
```

load image with all layers and metadata

- Docker saves the image as a tarfile containing layers
- Udocker loads the image
- Can be used to transfer images without having to pull them

udocker: save and load images

save image with all layers and metadata

```
$ docker   save   centos:centos6 | udocker   load
```

load image with all layers and metadata

- Save from docker and load with udocker
- Piping stdout to stdin

udocker: list local images

```
$ udocker  images
```

```
REPOSITORY
msoffice:lastest                    .
iscampos/openqcd:latest             .
fedora:25                         .
docker.io/susymastercode/mastercode:latest     .
ubuntu:14.04                      .
ubuntu:16.10                      .
ubuntu:latest                     .
```

udocker: list containers

```
$ udocker    ps
```

container-id alias image

```
CONTAINER ID                          P M NAMES          IMAGE
9fe2f9e7-ce37-3be5-b12d-829a3236d2a6  . W ['ub14']       ubuntu:14.04
5c7bd29b-7ab3-3d73-95f9-4438443aa6d6  . W ['myoffice']   msoffice:lastest
676eb77d-335e-3e9a-bf62-54ad08330b99  . W ['fedora_25']  fedora:25
c64afe05-adfa-39de-bf15-dcd45f284249  . W ['debianold']  debian:oldstable
7e76a4d7-d27e-3f09-a836-abb4ded0df34  . W ['ubuntu16', 'S']  ubuntu:16.10
9d12f52d-f0eb-34ae-9f0e-412b1f8f2639  . W ['f25']        fedora:25
```

udocker: run container as yourself


```
$ udocker run  --user=jorge  --bindhome  \
   --hostauth  ub14
```

```
 ****************************************************************************
 *                                                                          *
 *               STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6              *
 *                                                                          *
 ****************************************************************************
 executing: bash
jorge@nbjorge:~$ id
uid=1000(jorge) gid=1000(jorge) groups=1000(jorge),10(uucp)
jorge@nbjorge:~$ pwd
/home/jorge
jorge@nbjorge:~$
```

udocker: run commands in the prompt

```
$ udocker run --user=jorge --bindhome \
    --hostauth ub14 /bin/bash -c "id; pwd"
```

```
**************************************************************************
*                                                                        *
*               STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6            *
*                                                                        *
**************************************************************************
executing: bash
uid=1000(jorge) gid=1000(jorge) groups=1000(jorge),10(uucp)
/home/jorge
```

udocker: more quiet

```
$ udocker  -q  run  ub14  /bin/cat  /etc/lsb-release
```

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.5 LTS"
```

```
$ alias x=udocker  -q  run  --user=user --bindhome \
    --hostauth  ub14
$ x  /bin/ls
```