Using Containers to develop and port Salome

First experiments and feedback

C. Trophime, LNCMI CNRS, JUS 2018
Lncmi
a French User Magnet Facility

Water Cooled Resistive Magnets
31 KA, 24 MW, 36 T

Research Areas
• Metals and Superconductors
• Magnetism
• Semiconductors
• Soft Matter and Magnetosciences
• Applied Superconductivity
**HifiMagnet**

a software tool Chain

3D Qualification
- Electromagnetism
- Thermics
- Mechanics
- Hydraulics

Design
- Optimization
- Mech. Structure

Magnets
- Validations
- Commissioning

3D NL MultiPhysics
- 70 Billions Elem.
- 1 T RAM
- 48 CPUs
MSO4SC
HPC Cloud Computing service

A Portal to access:

- software catalogue (Hifimagnet),
- data repository,
- workflow management
- pre/post visualization of data (Salome)
Containers: a key ingredient
What are containers?

from freight transport to Software:
• can be (un-)loaded/stacked efficiently
• can be loaded on ships, trains, trucks, . . .
• can be handled without being opened

Light Virtual Environment
• isolate app from OS
• ensure app run “everywhere”
• share system resources host
Docker Workflow

Prerequisites
FROM ubuntu:xenial

ENV LC_ALL=C.UTF-8
ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get -qq update && \
    apt-get install -y wget firefox module-init-tools …

COPY Salome.run /tmp
RUN sh /tmp/Salome.run -t /opt/salome -d

COPY NVIDIA-DRIVER.run /tmp
RUN sh ./NVIDIA-DRIVER.run -a -N --ui=none --no-kernel-module

CMD ["/bin/bash"]

docker build -t trophime/salome:nvidia .
Running Docker images

<table>
<thead>
<tr>
<th></th>
<th>Start <strong>VxSrv, PowerShell</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Windows" /></td>
<td>docker run -it --rm -e DISPLAY=$IP:0 \ trophime/salome-8.2.0:nvidia</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Start <strong>XQuartz, xterm</strong>:</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Mac" /></td>
<td>xhost +$ip \ docker run -it --rm -e DISPLAY $ip:0 \ trophime/salome-8.2.0:nvidia</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Start a <strong>terminal:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Ubuntu" /></td>
<td>xhost local:root \ docker run -ti --rm -e DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix \ -v $HOME/.Xauthority:/home/feelpp/.Xauthority --env \ QT_X11_NO_MITSHM=1 trophime/salome-8.2.0:nvidia</td>
</tr>
</tbody>
</table>
However...

**Docker containers**
- need to be “root” to run container
  - alternative use uDocker
- bad performance issue on HPC

**HPC containers** [Singularity](https://www.singularityuniversity.edu/)
- need to be “root” *only* to build container
- good performance on HPC provided
  - “specific network driver” loaded within container (aka InfiniBand)
  - same MPI within container and on the host
Singularity Containers

Same workflow as Docker:

- **sudo** singularity build ...
- singularity push/pull ...
- singularity run ...

“Native” support for Nvidia

Bootstrapping:

```bash
Bootstrap: docker
From: debian:jessie

%setup
mkdir -p $SINGULARITY_ROOTFS/tmp
cp path/Salome-V8_4_0-univ_public.run $SINGULARITY_ROOTFS/tmp
exit 0

%post
apt-get update
apt-get -y install lsb-release wget curl firefox ...
sh /tmp/Salome-V8_4_0-univ_public.run -t /opt/salome -d
```
Live demos

- Connect to Cesga HPC (Santiago, Spain) with MSOPortal
- Start in Salome in Web browser (NoVnc/VirtualGl)

You need credentials to log
# Use of Salome

## Native

| Plugin for Hifimagnet from scratch, SalomeTools | Lncmi (1 person) |

## Virtualisation

<table>
<thead>
<tr>
<th>VM (Vagrant)</th>
<th>Master II Univ. Strasbourg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker</td>
<td>Master II Univ. Strasbourg</td>
</tr>
<tr>
<td>Singularity</td>
<td>Lncmi, Cemosis, Mso4sc (Cesga HPC, ...)</td>
</tr>
</tbody>
</table>
**Conclusions & Perspectives**

**Salome** deployment:
- easier (since maintenance limited to an OS)
- custom to your need
- available on all platforms (!! Graphic Cards !!)

Need for Official containers
- Docker/Singularity Registry
- *Windows store* ([WSL](https://wsl.io)) aka LXC container

Go for a **Salome** container *a la* **CMB (KitWare)**, **CAELinux**
Critics

- Installation simpler than in the past
  - still lot of questions (see forum)
  - several executables with different modules
  - no explicit nor standard way to rebuild modules

- Merging the installation process
  - Moving to Ubuntu/Debian package
- Using containers to “standardize”
  - Using Salome as a “black-box”
  - Keeping containers as light as possible
Thank you for your attention!

Contact information:
Scientific Coordinator: Zoltán Horváth (SZE) horvathz@math.sze.hu
Project Coordinator: Javi Nieto (ATOS) javi.nieto@atos.net
Website: www.mso4sc.eu

Grant agreement No. 731063
To go further

● Docker
  ○ Docker Nvidia
  ○ Docker with OpenGl (MacOs X)
● Singularity
  ○ SRegistry (CLI)
● Windows Subsystem Linux (Windows 10)
  ○ https://blogs.msdn.microsoft.com/commandline/
● Feelpp
● HiFiMagnet
### Benchmarks for 8.4.0
Meshing a Cube with Hex. (test_box.py)

**Debian/Testing**
- **CPU:** i7-4790 CPU @ 3.60GHz
- **RAM:** 32 Go
- **Mesh:** 300x300x300
- **Nvidia Quadro K2200**

<table>
<thead>
<tr>
<th></th>
<th>Native</th>
<th>Docker “Nvidia”</th>
<th>Docker “noVNC”</th>
<th>Singularity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compute</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>24 s</td>
<td>23.7 s</td>
<td>22.0 s</td>
<td>23.5s</td>
</tr>
<tr>
<td></td>
<td>8725 MiB</td>
<td>8666 MiB</td>
<td>8698.0 MiB</td>
<td>8679.7 MiB</td>
</tr>
<tr>
<td><strong>Display</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>86.7 s</td>
<td>83.1 s</td>
<td>68.0s</td>
<td>73.4 s</td>
</tr>
<tr>
<td></td>
<td>11105 MiB</td>
<td>10623.4 MiB</td>
<td>11009.8 MiB</td>
<td>11067.7 MiB</td>
</tr>
</tbody>
</table>
## Benchmarks for 8.4.0
Meshing a Cube with Hex. (test_box.py)

<table>
<thead>
<tr>
<th>Windows 10 Pro Build 1709</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU:</strong> i7-6820HQ @ 2.70GHz</td>
</tr>
<tr>
<td><strong>RAM:</strong> 16 Go</td>
</tr>
<tr>
<td><strong>Mesh:</strong> 100x100x100</td>
</tr>
<tr>
<td><strong>MobaXTerm 10.5</strong></td>
</tr>
<tr>
<td><strong>IntelHD 530</strong></td>
</tr>
<tr>
<td><strong>Quadro M1000M</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th><strong>Native (8.3.0)</strong></th>
<th><strong>Docker “Nvidia”</strong></th>
<th><strong>Docker noVNC</strong></th>
<th><strong>WSL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compute</strong></td>
<td>1.5 s 390.1 MiB</td>
<td>1.1 s 607.5 MiB</td>
<td>12.0 s 458.4 MiB</td>
<td></td>
</tr>
<tr>
<td><strong>Display</strong></td>
<td>4.5 s 439.7 MiB</td>
<td>3.6 s 724.4 MiB</td>
<td>3.8 s 526.5 MiB</td>
<td></td>
</tr>
</tbody>
</table>
Benchmarks for 8.4.0
Meshing a Cube with Hex. (test_box.py)

<table>
<thead>
<tr>
<th>Windows 10 Pro Build 1709</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU: 7-6700HQ CPU @ 2.60GHz</td>
</tr>
<tr>
<td>RAM: 32 Go</td>
</tr>
<tr>
<td>Mesh: 100x100x100</td>
</tr>
<tr>
<td>MobaXTerm 10.5</td>
</tr>
<tr>
<td>VxSrv 1.19.6.2</td>
</tr>
<tr>
<td>IntelHD 530</td>
</tr>
<tr>
<td>GeForce GTX 960M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Native (8.3.0)</th>
<th>Docker “Nvidia” (8.3.0)</th>
<th>Docker noVNC</th>
<th>WSL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compute</strong></td>
<td><strong>Display</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2 s 123.6 MiB</td>
<td>6.3 s 702.1 MiB 4.4 s 723.1 MiB</td>
<td>82.6 s 8036.0 MiB 4.8 s 520.7 MiB</td>
<td></td>
</tr>
<tr>
<td>1.9 s 608.4 MiB</td>
<td>4.4 s 723.1 MiB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8 s 606.5 MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>233.6 s 6722.9 MiB 2.0 s 451.0 MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SMDS_MemoryLimit disabled
Known issues

● 8.3.0:
  ○ Linux: Xorg 1.17 at least
  ○ Windows: VxSrv or MobaXterm
  ○ MacOs X: XQuartz 2.7.10

● 8.4.0:
  ○ Linux:
  ○ Windows: MobaXterm 10.5 with Nvidia
  ○ MacOs X:

● OpenGL support with remote rendering
● Singularity only available for Unix
● NoVNC container : !! memory allocated !!
● Windows WSL:
  ○ SMDS_Memorrylimit may induce perf issue