CCP PETMR Tcon 16Dec2016

Attendees: Ben Thomas (UCL), Casper da Costa-Luis (KCL), Charalampos Tsoumpas (Leeds), Christoph Kolbitsch (PTB), Evgueni Ovtchinnikov (STFC), James Hetherington (UCL), Kris Thielemans (UCL), Martin Turner (STFC/Manchester)

This t-con concentrated on deployment of SIRF.

Ben illustrated vagrant (https://www.vagrantup.com/) to create a (Virtualbox) VM. With a simple text file, you can configure a VM with e.g. Ubuntu and put in lines for package installation, cloning of git-repo and building etc. Typing "vagrant up" starts the VM, which will first time download relevant VM etc and build. Those VMs can be stored. You can then "startx" to get to a running Linux system with everything preinstalled. Everything in the "vagrant" folder on the VM is actually on the host (where the original Vagrantfile was). Presumably more sophisticated sharing is possible. James comments important advantage of vagrant is that you can have versioning of VMs this way.

Some discussion if it'd be possible to then "export" the VM for distribution. Ben&James think so (it's just a VirtualBox VM after all). Main advantage of this would be to be able to create a VM with a 'script', as opposed to hand-crafting. Kris is a bit worried about the "vagrant" shared folder.
Action (added after the meeting): Ben to try/investigate this.

Ben then also illustrated the use of docker (https://www.docker.com/), which allows for more lightweight "containers". Similar scripts to vagrant (or indeed shell-scripts) can be used to configure the docker container. Casper showed example of docker-script for SIRF (not complete yet). Advantage of docker over a VM is that it's very fast to start, so you can use a single executable in the container as if it's a native applications. Disadvantages: some extra learning for the user, no out-of-the-box GUI capabilities (ok on Linux, possible on Mac, on Windows probably needs VNC). Kris notes that Docker on Windows 10 is only easy on Pro and Enterprise (see https://www.docker.com/products/docker#/windows)

Some discussion on distribution of MATLAB on a VM or docker-image. No clear conclusions. Possible avenues for more info: James via "a friend", Kris via Robert Silk (UCL software licenses), but we're not optimistic to be able to do this such that it'd work at every university etc.

Discussion on (dis)advantages of virtualisation vs native. General feeling was that if we want to target the basic-user who doesn't know about VMs etc, a Windows native installer would be best. Some issues/comments:

- Installation of Gadgetron on Windows is painful.
  https://github.com/gadgetron/gadgetron/wiki/Windows-Installation-VS2013
  could be scripted up (Casper is willing to look at this).
  Action: Kris to ask NIH people about progress and willingness to help
  Action: Evgueni to check where he was stuck when trying 1 year ago
- Casper claims making a Windows installer is not hard and recommends iexpress (preinstalled in Windows). Create a folder, put everything in there, run iexpress. Done!
- We have no idea how easy this will be with different Windows/MATLAB versions etc
- Depends on Gadgetron for windows to work ok (although we can still use the VM for Gadgetron itself, and installer for the rest).

James recommends brew for Mac, RPM and deb for Linux. Kris is doubtful of having to support so many, with different OS versions etc.

James mentions VMs etc very useful for classroom environment. Also mentions https://github.com/jupyterhub/jupyterhub for Python deployment (users only need a web-browser, you need a server with jupyterhub and all software preinstalled). Trade-off between giving users software on their machine vs easy start. UCL is setting up Jupyterhub server with access to UCL home drives, but this will not be useful for the CCP.

Current conclusion on deployment:

High Priority: we will keep the VM option going as 'easy' solution, but make a new one with higher disk space etc, possibly via vagrant.

Medium priority: We will attempt to make our build system easier.

Medium priority: We will spend some time on investigating the Windows solution.

Appendix: Vagrant and Docker demo (Ben Thomas)

(see also separate PDF with nicer formatting)

Vagrant

'Vagrant is an open-source software product for building and maintaining portable virtual development environments.'

Home page: vagrantup.com

VagrantFile:

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/wily64"
  #config.vm.provision :shell, path: "bootstrap.sh"

  config.vm.provider "virtualbox" do |vb|
    # Display the VirtualBox GUI when booting the machine
    vb.memory = 2048
    vb.cpus = 1
    vb.gui = true
  end

  # Install xfce and virtualbox additions
  config.vm.provision "shell", inline: "sudo apt-get update"
  config.vm.provision "shell", inline: "sudo apt-get install -y xfce4
virtualbox-guest-dkms virtualbox-guest-utils virtualbox-guest-x11"
  # Permit anyone to start the GUI
  config.vm.provision "shell", inline: "sudo sed -i
's/allowed_users=.*$/allowed_users=anybody/' /etc/X11/Xwrapper.config"
  config.vm.provision "shell", inline: "sudo apt-get install -y build-
essential g++"
```

```
end
```

To start the environment:

```
vagrant up
```

By default the host directory containing the VagrantFile is mounted to `/vagrant`

To pause the environment:

```
vagrant suspend
```

To remove the environment:

```
vagrant destroy
```

Docker

'Docker is an open-source project that automates the deployment of Linux applications inside software containers.'

Home page: docker.com

Image repository: hub.docker.com

My Docker image repositories

Docker examples

Pulling an image to local machine:

```
docker pull ubuntu:14.04
```

Show the images that are locally available:

```
docker images
```

Run a `Bash` shell within the container:

```
docker run -it ubuntu:14.04 /bin/bash
```

Show containers:

```
docker ps -a
```

Run `Bash` without an image locally available:

```
docker run -it debian:jessie /bin/bash
```

Run STIR tests inside a Ubuntu container:

```
docker run -it benthomas1984/stir:latest /bin/bash
cd /opt/STIR-BUILD
make test
```

Attach host volume to container and compile:

`hostCodeDir/helloWorld.c`

```
#include<stdio.h>

int main()
{
    printf("Hello World\n");
    return 0;
}
docker run -v /Users/bathomas/Documents/DEV/DOCKER/hostCodeDir:/mycode \
-it benthomas1984/stir:latest /bin/bash

cd /mycode
cat helloWorld.c
gcc helloWorld.c
./a.out
```

Run command inside container from host command line:

```
docker run -it benthomas1984/stir:latest stir_math
```

Remove all containers that are not in use:

```
docker rm `docker ps --no-trunc -aq`
```

Remove all images:

```
docker rmi $(docker images -q)
```

License

either express or implied. See the License for the specific language governing permissions and limitations under the License.