Compile and run RAPID on the Microsoft Azure Cloud

By Cédric H. David (cedric.david@jpl.nasa.gov),

Introduction

Foreword
This tutorial explains how to compile RAPID (http://rapid-hub.org/) on the Microsoft Azure Cloud (http://azure.microsoft.com/).

Knowledge prerequisites
It is assumed here that the reader has some familiarity with Unix-like operating systems.

Account prerequisites
In order to run RAPID on the Microsoft Azure Cloud, one needs to have an Azure account. More information on Azure can be found at: http://azure.microsoft.com/.

Hardware prerequisites
Completion of this tutorial requires access to a computer with an Internet connection.

Software prerequisites
The computer needs an Internet browser, and FTP client and a Linux terminal (this type of software is either readily available or can be installed on Windows, Mac and Linux computers).
Create a Linux Virtual Machine on Azure

The easiest way to run RAPID is to do so in a Linux environment. One can create a Linux Virtual Machine on the Azure Cloud. Such can be done by logging in on https://manage.windowsazure.com/.

We will start by first creating a storage account because everything else will be stored in there. This is optional because a storage account would be created automatically anyway, but it allows picking the name of the storage (Azure automatically-generated names are quite lengthy and mysterious). To create a storage account, click on New and select Data Services/Storage/Quick Create. For the purpose of this tutorial, pick a Storage Name: storagename, Location: West US, Replication: 6Geo-Redundant.

A storage account is now created:
We will now create a virtual machine using a distribution of Linux is based on CentOS version 6.5. To do so, click on New and select Compute/Virtual Machine/From Gallery.

Then choose CentOS (The Community Enterprise Operating System).
Here several options can be chosen. For the purpose of this tutorial, pick a Virtual Machine Name: machinename, Tier: Standard, Size: A6 (4 cores 28 GB memory), New User Name: username, Authentication: Password.

And even more... Pick Create a new cloud service, Cloud service DNS name: machinename, Region: West US, Storage Account: storagename.
And leave the option **Install the VM agent** selected.

Now you should see the storage, virtual machine and cloud service all available:
The hard drive of this machine is located on the storage account.
Logging in and out of the virtual machine

In order to log in the virtual machine, one needs to open a Linux terminal window and type:

\texttt{ssh username@machinename.cloudapp.net}

(followed by the corresponding password)

Logging out is done by typing:

\texttt{Exit}
Basic set up of the Linux Virtual Machine running on Azure

Install compilers
The distribution of CentOS used here does not have compilers installed. Typing the following commands in the terminal after logging in will install the GNU Compiler Collection (GCC) Fortran and C++ compilers:

```
sudo yum install gcc
sudo yum install gcc-c++
sudo yum install gcc-gfortran
```

Install source code management
The following command will install the source code management software called GIT (http://git-scm.com/), which is helpful – though not mandatory – for using RAPID:

```
sudo yum install git
```

CentOS 6.5 has git version 1.7.1 by default.

Alternatively, one can do:

```
cd ~
mkdir installz
cd installz
wget http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.3-1.el6.rf.x86_64.rpm
sudo rpm -Uvh rpmforge-release-0.5.3-1.el6.rf.x86_64.rpm
sudo yum-config-manager --enable rpmforge-extras
sudo yum update git
```

Create an image of this virtual machine
Go to Virtual Machine/Instance/machinename/Dash board and select Shutdown. This is because it is better to turn a virtual machine off before saving an image of it.

Now do Virtual Machine/Instance/machinename/Dash board and select Capture. Pick a name like CentOS-gcc-git to remember what’s in this image.
The image should be now available for later:

Finally do **Virtual Machine/Instance/machinename/Dash board** and select **Start**.
Install the libraries needed by RAPID

Create basic directories
Create one directory for installation of software (installz) and one directory for running simulations (work) by typing:

```bash
mkdir installz
mkdir work
```

Uploading and installing the scientific libraries needed by RAPID
In order to compile and run RAPID, one needs to have some scientific libraries installed on the virtual machine. The version of RAPID in this tutorial uses the following:

- The netCDF data library ([http://www.unidata.ucar.edu/software/netcdf/](http://www.unidata.ucar.edu/software/netcdf/)) version 3.6.3

Download the corresponding files from the internet (netcdf-3.6.3.tar.gz, petsc-3.3-p7.tar.gz, and tao-2.1-p2.tar.gz) and upload them on the virtual machine in the folder `/home/username/installz` using an FTP client (the same `username` and `password` used for logging on using the terminal are required).

Install the netCDF library
To install the netCDF library, log on the virtual machine using the terminal and type:
cd installz
tar -xzf netcdf-3.6.3.tar
mkdir netcdf-3.6.3
 cd netcdf-3.6.3
 ./configure --prefix=/home/username/installz/netcdf-3.6.3-install
make check > check.log
make install > install.log

Install the PETSc library
To install the PETSc library in regular mode, open the Linux terminal and type:

cd installz
tar -xzf petsc-3.3-p7.tar.gz
 cd petsc-3.3-p7
 ./configure PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx --download-f-blas-lapack=1 --download-mpich=1 --with-cc=gcc --with-cxx=g++ --with- fc=gfortran --with-clanguage=cxx --with-debugging=0
make PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx all
make PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx test

To install the PETSc library in debug mode, open the Linux terminal and type:

cd installz
cd petsc-3.3-p7
 ./configure PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx-debug --download-f-blas-lapack=1 --download-mpich=1 --with-cc=gcc --with-cxx=g++ --with- fc=gfortran --with-clanguage=cxx --with-debugging=1
make PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx-debug all
make PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx-debug test

To install the PETSc library with third-level optimization, open the Linux terminal and type:

cd installz
cd petsc-3.3-p7
 ./configure PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx-03 --download-f- blas-lapack=1 --download-mpich=1 --with-cc=gcc --with-cxx=g++ --with- fc=gfortran --with-clanguage=cxx --with-debugging=0 COPTFLAGS='-O3 - march=native -mtune=native' CXXOPTFLAGS=''-O3 -march=native -mtune=native' FOPTFLAGS=''-O3 -march=native -mtune=native'
make PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx-03 all
make PETSC_DIR=$PWD PETSC_ARCH=linux-gcc-cxx-03 test

Notes: adding PETSC_DIR and PETSC_ARCH in the configure and make commands allows not to have to worry about the environment variables being set while installing so that such can be done afterwards. If environmental variables are set before-hand, the installation commands above use command line
variables and ignores environment variables. The way PETSc deals with different architectures is by creating various folders at $\text{PETSC_DIR}/\text{PETSC_ARCH}$.

**Install the TAO library**

To install the TAO library in regular mode, open the Linux terminal and type:

```
cd installz
tar -xzf tao-2.1-p2
cd tao-2.1-p2
make TAO_DIR=$PWD PETSC_DIR=/home/username/installz/petsc-3.3-p7
PETSC_ARCH=linux-gcc-cxx all > make.log
make TAO_DIR=$PWD PETSC_DIR=/home/username/installz/petsc-3.3-p7
PETSC_ARCH=linux-gcc-cxx tao_testfortran > fortran.log
```

To install the TAO library in debug mode, open the Linux terminal and type:

```
cd installz
cd tao-2.1-p2
make TAO_DIR=$PWD PETSC_DIR=/home/username/installz/petsc-3.3-p7
PETSC_ARCH=linux-gcc-cxx-debug all > make-debug.log
make TAO_DIR=$PWD PETSC_DIR=/home/username/installz/petsc-3.3-p7
PETSC_ARCH=linux-gcc-cxx-debug tao_testfortran > fortran-debug.log
```

To install the TAO library with third-level optimization, open the Linux terminal and type:

```
cd installz
cd tao-2.1-p2
make TAO_DIR=$PWD PETSC_DIR=/home/username/installz/petsc-3.3-p7
PETSC_ARCH=linux-gcc-cxx-O3 all > make-O3.log
make TAO_DIR=$PWD PETSC_DIR=/home/username/installz/petsc-3.3-p7
PETSC_ARCH=linux-gcc-cxx-O3 tao_testfortran > fortran-O3.log
```

**Notes:** the way TAO deals with different architectures is by adding various folders at $\text{TAO_DIR}/\text{PETSC_ARCH}$.

**Set environmental variables**

Environment variables are used in Linux to set some user preferences. Such is usually done in a file called `.bashrc` that is located in the user’s home directory `/home/username`. Here we will specify where the libraries were installed, and make sure Linux knows where the MPI executables for parallel computing (installed with PETSc) are located. Edit the `.bashrc` file by adding:

```
export TACC_NETCDF_LIB='/home/username/installz/netcdf-3.6.3-install/lib'
export TACC_NETCDF_INC='/home/username/installz/netcdf-3.6.3-install/include'
export PETSC_DIR='/home/username/installz/petsc-3.3-p7'
export PETSC_ARCH='linux-gcc-cxx-O3'
```
Notes: One can now switch between regular and debug mode in PETSc/TAO by commenting out either export PETSC_ARCH='linux-gcc-cxx' or export PETSC_ARCH='linux-gcc-cxx-debug' in the .bashrc file.

**Basic setup for git**

RAPID uses git for source code management. The basic setup of git is the following:

```
git config --global user.name "Your Name"
git config --global user.email your@email.com
git config --global color.ui true
```

**Create an image of this virtual machine**

The steps are similar as earlier in this tutorial.
**Downloading and compiling RAPID**

Downloading the entire RAPID repository on your virtual machine can be done easily with:

```bash
cd work/
git clone https://github.com/c-h-david/rapid.git
```

In order to make sure the version of RAPID you’re using is the same as the one used in this tutorial, type:

```bash
cd rapid/
git checkout 20140709
```

**Now compile RAPID:**

```bash
cd src/
make rapid
```

That’s it!!!
Additional notes on deleting disks
The disks from the Azure cloud are located in:

storage / storagename / containers / vhds /

However, before deleting the OS hard drives, one needs to make sure the virtual machines are deleted from:

virtual machines / instances / machinename

And in order to delete the disk images, one needs to do so in:

virtual machines / images

Further information
RAPID website: http://rapid-hub.org/
RAPID source code: https://github.com/c-h-david/rapid/