MATLAB on the clusters

The full version of MATLAB is only installed on the login node so in order to run MATLAB jobs on the cluster you first need to compile your .m files then run them using the MATLAB runtime.

Let's start with our MatrixCAB.m file

```
disp("Matrix A:");
A = [1, 2; 3, 4];
disp(A);

disp("Matrix B:");
B = [5, 6; 7, 8];
disp(B);

disp("Matrix C = A * B:");
C = A * B;
disp(C);
```

On the DCSR clusters MATLAB is in the default path so we also have access to the MATLAB Compiler (mcc).

We now compile the MatrixCAB.m file:

```
$ mcc -v -m MatrixCAB.m

Compiler version: 7.0.1 (R2019a)
Dependency analysis by REQUIREMENTS.
Parsing file "/users/ulambda/MatrixCAB.m"
	(referenced from command line).
Generating file "/users/ulambda/readme.txt".
Generating file "MatrixCAB.sh".
```

The compiler documentation can be found at https://ch.mathworks.com/help/compiler/mcc.html

Note that there are now 3 new files:

`readme.txt`

`run_MatrixCAB.sh`

`MatrixCAB`

If we take a look at the last file we see

```
$ file MatrixCAB
MatrixCAB: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses
shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=ad76a4654419e7968208a77a172f103afe2d77c2,
stripped
```

The `readme.txt` explains in great detail how to run the compiled object and the `run_MatrixCAB.sh` script is for launching the job.

In order to make use of the executable we need to load the MATLAB runtime environment using module

```
source /dcsrsoft/spack/bin/setup_dcsrsoft
module load matlab-runtime/2019a
```

Please note that the runtime has to correspond to the version of mcc used to compile the .m file. Please see the following page for the corresponding runtime and compiler versions:

https://ch.mathworks.com/products/compiler/matlab-runtime.html

The runtime module sets the `MCR_PATH` variable which is needed by the `run_MatrixCAB.sh` script.

To launch the compiled MatrixCAB object we need to put all the elements together:

`sh run_MatrixCAB.sh $MCR_PATH`

Obviously this should be done on a compute node using a job script:

```
#!/bin/bash

#SBATCH --time 00-00:05:00
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 1
#SBATCH --mem 4000M

source /dcsrsoft/spack/bin/setup_dcsrsoft
module load matlab-runtime/2019a

MATLAB_SCRIPT=MatrixCAB
```

```
sh run_$MATLAB_SCRIPT.sh $MCR_PATH

echo "Finished - next time I'll port my code to python"
```

Task farming with Matlab

When processing numerous Matlab jobs in parallel on the clusters, you will likely encounter stability issues with some jobs failing randomly, other hanging (see below the explanations from Matlab support). To solve the issue, you must set the MCR_CACHE_ROOT environment variable (see https://ch.mathworks.com/help/compiler_sdk/ml_code/mcr-component-cache-and-ctf-archive-embedding.html) in order that the same location (by default in your home directory) is not used by all jobs.

For job arrays, you can adopt the following:

```
#! /bin/bash

#SBATCH --array=1-5
#SBATCH --partition wally
#SBATCH --mem=1G
#SBATCH --time=00:15:00

source /dcsrsoft/spack/bin/setup_dcsrsoft
module load matlab-runtime/2019a

# Create a task-specific MCR_CACHE_ROOT directory
mcr_cache_root=/tmp/$USER/MCR_CACHE_ROOT_${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}
mkdir -pv $mcr_cache_root
export MCR_CACHE_ROOT=$mcr_cache_root

### YOUR MATLAB ANALYSIs HERE

# Tidy up the place
rm -rv $mcr_cache_root
```

Explanations from Matlab support

>

> When running a MATLAB Compiler standalone executable, the MCR_CACHE_ROOT location is used by the standalone executable to extract the deployable archive into. As the name suggests, the extracted archive is cached in this location, meaning the archive is extracted the very first time you run the application and then for consecutive runs the already extracted data from the cache is used.
>
> There are mechanisms in place which try to ensure that when you run multiple instances of the same application at the same time, you do not run into any concurrency issues with this cache (e.g. a second instance should not also try to extract the archive if the first instance was already in the process of doing this). However, there are some limitations to these mechanisms; they were designed to deal with concurrency issues which might occur if an interactive user would run a handful of concurrent instances of the application; when doing this interactively this implies that you are not starting all those instances at exactly the same point in time and there are at least a few seconds between starting each instance. If you are somehow starting *a lot* of instances at *virtual the same time* (through some shell script, or possible even some cluster scheduler), this mechanism may break down. The likelihood of running into issues increases even more if the cache is in located on a shared network drive, shared by multiple machines (which can definitely be the case for a home directory), and all these machines are running instances of the same application.
>
> This is probably what you are running into then. Giving each instance its own cache location would prevent those issues altogether as there would be no concurrency in the first place.

Revision #20
Created 13 January 2020 15:58:14 by Ewan Roche
Updated 8 March 2021 07:27:11 by Etienne Orliac