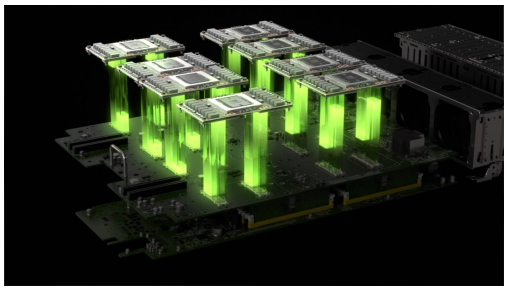
JADE Documentation

Release

Mozhgan K. Chimeh

Oct 09, 2017

Contents



This is the documentation for the Joint Academic Data science Endeavour (JADE) facility.

JADE is a UK Tier-2 resource, funded by EPSRC, owned by the University of Oxford and hosted at the Hartree Centre. The hardware was supplied and integrated by ATOS Bull.

A consortium of eight UK universities, led by the University of Oxford, has been awarded £3 million by the Engineering and Physical Sciences Research Council (EPSRC) to establish a new computing facility known as the Joint Academic Data science Endeavour (JADE). This forms part of a combined investment of £20m by EPSRC in the UK's regional Tier 2 high-performance computing facilities, which aim to bridge the gap between institutional and national resources.

JADE is unique amongst the Tier 2 centres in being designed for the needs of machine learning and related data science applications. There has been huge growth in machine learning in the last 5 years, and this is the first national facility to support this rapid development, with the university partners including the world-leading machine learning groups in Oxford, Edinburgh, KCL, QMUL, Sheffield and UCL.

The system design exploits the capabilities of NVIDIA's DGX-1 Deep Learning System which has eight of its newest Tesla P100 GPUs tightly coupled by its high-speed NVlink interconnect. NVIDIA has clearly established itself as the leader in massively-parallel computing for deep neural networks, and the DGX-1 runs optimized versions of many standard machine learning software packages such as Caffe, TensorFlow, Theano and Torch.

This system design is also ideal for a large number of molecular dynamics applications and so JADE will also provide a powerful resource for molecular dynamics researchers at Bristol, Edinburgh, Oxford and Southampton.

CHAPTER 1

JADE Hardware

JADE hardware consists of:

- 22 DGX-1 Nodes, each with 8 Nvidia P100 GPUs
- 2 Head nodes

Using the JADE Facility

If you have not used a High Performance Computing (HPC) cluster, the Linux operating system or even a command line before this is the place to start. This guide will get you set up using the JADE cluster fairly quickly.

Getting an account

As a regular user, getting started involves 3 steps:

1) Apply for a Hartree SAFE account

This is a web account which will show you which projects you belong to, and the accounts which you have in them.

Before applying for a SAFE account, you should first have an SSH key-pair, and be ready to provide your public key as part of the SAFE registration process. Information on generating and using SSH keys is available here: http://yukon.dl.ac.uk:8080/wiki/site/admin/SAFE%20User%20Guide.html#ssh but for any help you should contact your local university IT support staff.

Once you have your public SSH key ready, apply for your SAFE account by going here: https://um.hartree.stfc.ac.uk/hartree/login.jsp and providing all of the required information.

When your account has been approved, you will receive an email giving your initial password. When you login for the first time you will be asked to change it to a new one.

Further details on the registration process are available here: http://community.hartree.stfc.ac.uk/wiki/site/admin/safe%20user%20guide.html

2) Apply for a JADE project account

Once your SAFE account is established, login to it and click on "Request Join Project".

From the drop-down list select the appropriate **project**, enter the **signup code** which you should have been given by the project PI or manager, and then click "Request".

The approval process goes through several steps:

1. approval by the PI or project manager – once this is done the SAFE status changes to Pending
2. initial account setup – once this is done the SAFE status changes to Active
3. completion of account setup – once this is done you will get an email confirming you are all set, and your SAFE account will have full details on your new project account

This process shouldn't take more than 2 working days. If it takes more than that, check whether the PI or project manager is aware that you have applied, and therefore your application needs their approval through the SAFE system.

If your SAFE userid is xyz, and your project suffix is abc, then your project account username will be xyz-abc and you will login to JADE using the command:

ssh -l xyz-abc jade.hartree.stfc.ac.uk

Note that some users may belong to more than one project, in which case they will have different account usernames for each project, and all of them will be listed on their SAFE web account.

Each project account will have a separate file structure, and separate quotas for GPU time, filestore and other system resources.

Note also that JADE has multiple front-end systems, and because of this some SSH software operating under stringent security settings might give warnings about possible man-in-the-middle attacks because of apparent changes in machine settings. This is a known issue and is being addressed, but in the meantime these warnings can be safely ignored.

3) Apply for a Hartree ServiceNow account

This is a web account used for reporting any operational issues with JADE.

To obtain an account follow the directions here: http://community.hartree.stfc.ac.uk/wiki/site/admin/servicenow.html

Note the guidance which explains that the first time you try to login you will not have a password so you need to click on the link which says "reset your password here".

Due to a problem with synchronising userids between ServiceNow and JADE, it is possible that ServiceNow may say that your email address is not recognised. If this happens, please send an email to hartree@stfc.ac.uk and ask them to add you to the ServiceNow database.

Connecting to the cluster using SSH

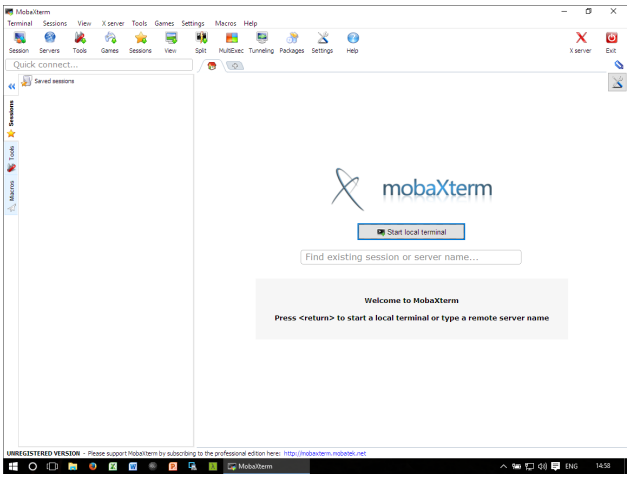
The most versatile way to **run commands and submit jobs** on the cluster is to use a mechanism called SSH, which is a common way of remotely logging in to computers running the Linux operating system.

To connect to another machine using SSH you need to have a SSH *client* program installed on your machine. macOS and Linux come with a command-line (text-only) SSH client pre-installed. On Windows there are various graphical SSH clients you can use, including *MobaXTerm*.

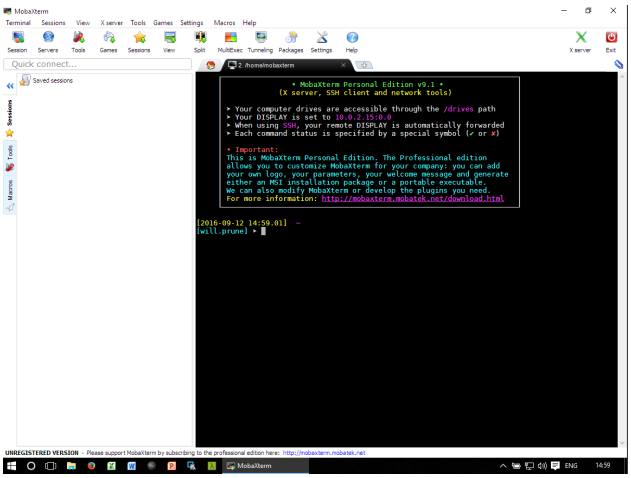
SSH client software on Windows

Download and install the *Installer edition* of mobaxterm.

After starting MobaXterm you should see something like this:



Click *Start local terminal* and if you see something like the following then please continue to *Establishing a SSH connection*.



Running commands from a terminal (from the command-line) may initially be unfamiliar to Windows users but this is the recommended approach for running commands on ShARC and Iceberg as it is the idiomatic way of interfacing with the Linux clusters.

SSH client software on Mac OS/X and Linux

Linux and macOS (OS X) both typically come with a command-line SSH client pre-installed.

If you are using macOS and want to be able to run graphical applications on the clusters then you need to install the latest version of the XQuartz *X Windows server*.

Open a terminal (e.g. *Gnome Terminal* on Linux or *Terminal* on macOS) and then go to *Establishing a SSH connection*.

Establishing a SSH connection

Once you have a terminal open run the following command to log in to a cluster:

```
ssh -l $USER jade.hartree.stfc.ac.uk
```

Here you need to replace `$USER` with your username (e.g. `te1st-test`)

Note: JADE has multiple front-end systems, and because of this some SSH software operating under stringent security settings might give **warnings about possible man-in-the-middle attacks** because of apparent changes in machine settings. This is a known issue and is being addressed, but in the meantime **these warnings can be safely ignored**

To ignore the warning, add the option *-o StrictHostKeyChecking=no* to your SSH command e.g.: *ssh -o StrictHostKeyChecking=no -l $USER jade.hartree.stfc.ac.uk*

Or in your *~/.ssh/config* file, add the line: *StrictHostKeyChecking no*

Note: **macOS users**: if this fails then:

- Check that your XQuartz is up to date then try again *or*
- Try again with `-Y` instead of `-X`

This should give you a prompt resembling the one below:

```
te1st-test@dgj223:~$
```

The Slurm Scheduler

Introduction

Running software on the JADE system is accomplished via batch jobs, *i.e.* in an unattended, non-interactive manner. Typically a user logs in to the JADE login nodes, prepares a job script and submits it to the job queue.

Jobs on JADE are managed by the Slurm batch system, which is in charge of:

- allocating the computer resources requested for the job,
- running the job and
- reporting the outcome of the execution back to the user.

Running a job involves, at the minimum, the following steps

- preparing a submission script and
- submitting the job to execution.

This guide describes basic job submission and monitoring for Slurm. The topics in the guide are:

- the main Slurm commands,
- preparing a submission script,
- slurm partitions,
- submitting a job to the queue,

- monitoring a job execution,
- deleting a job from the queue and
- environment variables.

Commands

The table below gives a short description of the most used Slurm commands.

Command	Description
`sacct`	report job accounting information about active or completed jobs
`salloc`	allocate resources for a job in real time (typically used to allocate resources and spawn a shell, in which the srun command is used to launch parallel tasks)
`sbatch`	submit a job script for later execution (the script typically contains one or more `srun` commands to launch parallel tasks)
`scancel`	cancel a pending or running job
`sinfo`	reports the state of partitions and nodes managed by Slurm (it has a variety of filtering, sorting, and formatting options)
`squeue`	reports the state of jobs (it has a variety of filtering, sorting, and formatting options), by default, reports the running jobs in priority order followed by the pending jobs in priority order
`srun`	used to submit a job for execution in real time

All Slurm commands have extensive help through their man pages *e.g.*:

```
man sbatch
```

shows you the help pages for the `sbatch` command.

Preparing a submission script

A submission script is a Linux shell script that

- describes the processing to carry out (e.g. the application, its input and output, etc.) and
- requests computer resources (number of cpus, amount of memory, etc.) to use for processing.

The simplest case is that of a job that requires a single node with the following requirements:

- the job uses 1 node,
- the application is a single process,
- the job uses a single GPU,
- the job will run for no more than 10 hours,
- the job is given the name "job123" and
- the user should be emailed when the job starts and stops or aborts.

Supposing the application run is called `myCode` and takes no command line arguments, the following submission script runs the application in a single job::

```
#!/bin/bash

# set the number of nodes
#SBATCH --nodes=1
```

```
# set max wallclock time
#SBATCH --time=10:00:00

# set name of job
#SBATCH --job-name=job123

# set number of GPUs
#SBATCH --gres=gpu:1

# mail alert at start, end and abortion of execution
#SBATCH --mail-type=ALL

# send mail to this address
#SBATCH --mail-user=john.brown@gmail.com

# run the application
myCode
```

The script starts with `#!/bin/bash` (also called a shebang), which makes the submission script a Linux bash script.

The script continues with a series of lines starting with `#`, which represent bash script comments. For Slurm, the lines starting with `#SBATCH` are directives that request job scheduling resources. (Note: it is important that you put all the directives at the top of a script, before any other commands; any `#SBATCH` directive coming after a bash script command is ignored!)

The resource request `#SBATCH --nodes=n` determines how many compute nodes a job are allocated by the scheduler; only 1 node is allocated for this job.

The maximum walltime is specified by `#SBATCH --time=T`, where `T` has format **h:m:s**. Normally, a job is expected to finish before the specified maximum walltime. After the walltime reaches the maximum, the job terminates regardless whether the job processes are still running or not.

The name of the job can be specified too with `#SBATCH --job-name=name`.

Lastly, an email notification is sent if an address is specified with `#SBATCH --mail-user=<email_address>`. The notification options can be set with `#SBATCH --mail-type=<type>`, where `<type>` may be `BEGIN`, `END`, `FAIL`, `REQUEUE` or `ALL` (for any change of job state).

The final part of a script is normal Linux bash script and describes the set of operations to follow as part of the job. The job starts in the same folder where it was submitted (unless an alternative path is specified), and with the same environment variables (modules, etc.) that the user had at the time of the submission. In this example, this final part only involves invoking the `myCode` application executable.

The `module` tool

Introduction

The Linux operating system makes extensive use of the *working environment*, which is a collection of individual environment variables. An environment variable is a named object in the Linux shell that contains information used by one or more applications; two of the most used such variables are `$HOME`, which defines a user's home directory name, and `$PATH`, which represents a list paths to different executables. A large number of environment variables are already defined when a Linux shell is open but the environment can be customised, either by defining new environment variables relevant to certain applications or by modifying existing ones (e.g. adding a new path to `$PATH`).

`module` is a Software Environment Management tool, which is used to manage the working environment in preparation for running the applications installed on JADE. By loading the module for a certain installed application, the environment variables that are relevant for that application are automatically defined or modified.

Useful commands

The module utility is invoked by the command `module`. This command must be followed by an instruction of what action is required and by the details of a pre-defined module.

The utility displays a help menu by doing:

```
module help
```

The utility displays the available modules by issuing the command:

```
module avail
```

or displays only the information related to a certain software package, *e.g.*:

```
module avail pgi
```

The avail instruction displays all the versions available for the installed applications, and shows which version is pre-defined as being the default. A software package is loaded with the load or the add instructions, *e.g.*:

```
module load pgi
```

If no version is specified, the default version of the software is loaded. Specific versions, other than the default can be loaded by specifying the version, *e.g.*:

```
module load pgi/2017
```

The modules that are already loaded by users in a session are displayed with the command:

```
module list
```

A module can be "unloaded" with the unload or rm instructions, *e.g.*:

```
module unload pgi
module load pgi/2017
```

Lastly, all modules loaded in a session can be "unloaded" with a single command::

```
module purge
```

Best practices

`module` can be used to modify the environment after login, *e.g.* to load the Portland compilers in order to build an application. However, most frequent usage will be to load an already built application, and the best way to do this is from within the submission script. For example, assuming a job uses the NAMD molecular dynamics package, the submission script contains:

```
module purge
module load NAMD
```

Using Containerised Applications

On entering the container the present working directory will be the user's home directory: `/home_directory`

Any files you copy into `/home_directory` will have the same userid as normal and will be available once exiting the container. The local disk space on the node is available at: `/local_scratch/$USERID`

This is 6.6TB in size but any data will be lost once the interactive session is ended. There are two ways of interacting with the containerised applications.

1.Interactive Mode

All the applications in containers can be launched interactively in the same way using 1 compute node at a time. The number of GPUs to be used per node is requested using the `gres` option. To request an interactive session on a compute node the following command is issued from the login node:

```
srun --gres=gpu:2 --pty /jmain01/apps/docker/caffe 17.04
```

This command will show the following, which is now running on a compute node:

```
================
==NVIDIA Caffe==
================

NVIDIA Release 17.04 (build 26740)

Container image Copyright (c) 2017, NVIDIA CORPORATION.  All rights reserved.
Copyright (c) 2014, 2015, The Regents of the University of California (Regents)
All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION.  All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying␣
↪project or file.

groups: cannot find name for group ID 1002
I have no name!@124cf0e3582e:/home_directory$
```

Note. The warnings in the last two lines can be ignored. To exit the container, issue the "exit" command. To launch the other containers the commands are:

```
srun --gres=gpu:8 --pty /jmain01/apps/docker/theano 17.04
srun --gres=gpu:4 --pty /jmain01/apps/docker/torch 17.04
```

2.Batch Mode

There are wrappers for launching the containers in batch mode. For example, to launch the Torch application change directory to where the launching script is, in this case called `submit-char.sh`:

```
cd /jmain01/home/atostest/char-rnn-master
```

A Slurm batch script is used to launch the code, such as:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH -p all
#SBATCH -J Torch
#SBATCH --gres=gpu:8
#SBATCH --time=01:00:00

/jmain01/apps/docker/torch-batch -c ./submit-char.sh
```

The output will appear in the slurm standard output file.

Each of the containerised applications has its own batch launching script:

```
/jmain01/apps/docker/torch-batch
/jmain01/apps/docker/caffe-batch
/jmain01/apps/docker/theano-batch
```

Information for PIs and Project Managers

Creating a Project

Access to the HPC systems at the Hartree Centre is available to groups of academic and industry partners. The PI (principal investigator) should complete a Project Proposal form in consultation with help desk staff and submit it for approval. Please contact hartree@stfc.ac.uk for a proposal form.

Getting an Account

After the proposal has been approved by the Help Desk, a SAFE account must be created which allows the management of **users** and **projects**. At registration, the PI provides an institutional email address which will be used as their SAFE login ID (please note, emails such as gmail or hotmail will not be accepted). An SSH Key is also required at this stage, and instructions on how to generate and upload this are provided in SAFE User Guide.

Once a project has been set up by Hartree staff, the PI can define associated project managers and then the PI and project managers are able to define groups (in a tree hierarchy), accept people into the project and allocate them to groups.

Projects and Groups on JADE

At the top-level there are **projects** with a PI, and within a **project** there are **groups** with resources such as GPU time and disk space. A **group must be created** in order to accept normal users in to the project.

The simplest approach is to create a single group for the Project for all users, but additional groups can be created if necessary.

To create a group:

1. From the main portal of SAFE, click the *Administer* button next to the name of the project that you manage.
2. Click the *Project Group Administration* button, then click *Add New*.
3. Type in a name and description and click *Create*.

Project Signup Code

The **project signup code is required** so that users can request to join your project.

To set a project signup code:

1. From the main portal of SAFE, click the *Administer* button next to the name of the project that you manage.
2. Click the *Update* button.
3. In the **Password** field, set the desired **project signup code** and press *Update*.

Adding Users to the Project

After creating a group, users can be added to a project.

To add users:

1. Provide your users with the **project name**, **project signup code** (see above section) and signup instruction for regular users at:

   http://jade-hpc.readthedocs.io/en/latest/jade/getting-account.html

2. Once a user has requested to join a project, there will be a "New Project Management Requests" box. Click the *Process* button and *Accept* (or *Reject*) the new member.
3. The PI can now add the member to the a **group** previously created.
4. Click on *Administer* for the Project then choose *Project Group Administration*.
5. Click on the *Group name*, then on *Add Account*, and then select any available user from a list (including the PI).
6. There is now a manual process, which may take up to 24 hours, after which the new Project member will be notified of their new userid and invited to log on to the Hartree systems for the first time.
7. The new project member will have an *Active* status in the group once the process is completed.

Note: The PI does NOT have to 'Request Join Project' as he/she is automatically a Project member. They must however, add themselves to a Group.

Project groups and shared file system area

Each Project group maps to a Unix group, and each Project group member's home file system is set up under a directory group structure. The example below starts from a user's home directory and shows that all other members of the Project group are assigned home directories as "peer" directories.

```
-bash-4.1$ pwd
/gpfs/home/training/jpf03/jpf26-jpf03
-bash-4.1$ cd ..
-bash-4.1$ ls
afg27-jpf03  bbl28-jpf03  cxe72-jpf03  dxd46-jpf03  hvs09-jpf03  jjb63-jpf03  jxm09-
↪jpf03  mkk76-jpf03  phw57-jpf03  rrr25-jpf03  rxw47-jpf03  sxl18-jpf03
ajd95-jpf03  bwm51-jpf03  cxl10-jpf03  dxp21-jpf03  hxo76-jpf03  jkj47-jpf03  kxm85-
↪jpf03  mxm86-jpf03  pxj86-jpf03  rrs70-jpf03  sca58-jpf03  tcn16-jpf03
axa59-jpf03  bxp59-jpf03  djc87-jpf03  fxb73-jpf03  ivk29-jpf03  jpf26-jpf03  lim17-
↪jpf03  nxt14-jpf03  rja87-jpf03  rwt21-jpf03  shared       txc61-jpf03
axw52-jpf03  bxv09-jpf03  dwn60-jpf03  gxx38-jpf03  jds89-jpf03  jrh19-jpf03  ltc84-
↪jpf03  pag51-jpf03  rjb98-jpf03  rxl87-jpf03  sls56-jpf03  vvt17-jpf03
```

Important to some, please note that for each Project group there is a "shared" directory which can be reached at

```
../shared
```

from each user's home directory. Every member of the Project group is able to read and write to this shared directory, so it can be used for common files and applications for the Project.

Once a Project has Finished

It is Hartree Centre policy that, after the agreed date of completion of a Project, all data will be made read-only and will then remain retrievable for 3 months. During this period, users are able to login to retrieve their data, but will be unable to run jobs. After 3 months have elapsed, all login access associated with the Project will be terminated, and all data owned by the Project will be deleted.

Software on JADE

The software initially installed on the machine is listed in the following table:

Application	Version	Note
GNU compiler suite	4.8.4	part of O/S
PGI compiler suite	17.4	
OpenMPI	1.10.2	Supplied with PGI
OpenMPI	1.10.5a1	Supplied with PGI
Gromacs	2016.3	Supplied by Nvidia
NAMD	2.12	

This software has been built from source and installed as modules. To list the source built applications do:

```
$ module avail
------------------------------------------------------- /jmain01/apps/modules ----------
↪--------------------
gromacs/2016.3           openmpi/1.10.2/2017      pgi/17.4(default)        pgi64/17.
↪4(default)
PrgEnv-pgi/17.4(default)      NAMD/2.12      openmpi/1.10.5a1/GNU     pgi/2017
↪     pgi64/2017
```

The applications initially supplied by Nvidia as containers are listed in the following table:

Application	Version
Caffe	17.04
Theano	17.04
Torch	17.04

To list the containers and version available on the system do:

```
$ containers
REPOSITORY                    TAG                 IMAGE ID                CREATED
↪          SIZE
nvidia/cuda                   latest              15e5dedd88c5        4 weeks ago
↪     1.67 GB
nvcr.io/nvidia/caffe          17.04               87c288427f2d        6 weeks ago
↪     2.794 GB
nvcr.io/nvidia/theano         17.04               24943feafc9b         8 weeks ago
↪     2.386 GB
nvcr.io/nvidia/torch          17.04               a337ffb42c8e         9 weeks ago
↪     2.9 GB
```

The following brief notes explain how to run the various applications.

Applications on JADE

Caffe

Caffe

URL http://caffe.berkeleyvision.org/

Caffe is a Deep Learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.

The Caffe Docker Container

Caffe is available on JADE through the use of a Docker container. For more information on JADE's use of containers, see *Using Containerised Applications*.

Using Caffe Interactively

All the contained applications are launched interactively in the same way within 1 compute node at a time. The number of GPUs to be used per node is requested using the "gres" option. To request an interactive session on a compute node the following command is issued from the login node:

```
# Requesting 2 GPUs for Caffe version 17.04
srun --gres=gpu:2 --pty  /jmain01/apps/docker/caffe 17.04
```

This command will show the following, which is now running on a compute node:

```
==================
== NVIDIA Caffe ==
==================

NVIDIA Release 17.04 (build 26740)

Container image Copyright (c) 2017, NVIDIA CORPORATION.  All rights reserved.
Copyright (c) 2014, 2015, The Regents of the University of California (Regents)
All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION.  All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying␣
↪project or file.

groups: cannot find name for group ID 1002
I have no name!@124cf0e3582e:/home_directory$
```

You are now inside the container where the *Caffe* software is installed. Let's check the version

```
caffe --version
```

You can now begin training your network:

```
caffe train -solver=my_solver.prototxt
```

Using Caffe in Batch Mode

There are wrappers for launching the containers within batch mode.

Firstly navigate to the folder you wish your script to lauch from, for example we'll use the home directory:

```
cd ~
```

It is recommended that you create a script file e.g. *script.sh*:

```
#!/bin/bash

# Prints out Caffe's version number
caffe --version
```

And don't forget to make your *script.sh* executable:

```
chmod +x script.sh
```

Then create a Slurm batch script that is used to launch the code, e.g. *batch.sh*:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH -p all
#SBATCH -J Caffe-job
#SBATCH --gres=gpu:8
#SBATCH --time=01:00:00

#Launching the commands within script.sh
/jmain01/apps/docker/caffe-batch -c ./script.sh
```

You can then submit the job using *sbatch*:

```
sbatch batch.sh
```

The output will appear in the slurm standard output file.

Gromacs

Job scripts

Gromacs is a versatile package for molecular dynamics simulations, which solves the Newtonian equations of motion for systems with hundreds to millions of particles. Although the software scales well to hundreds of cores for typical simulations, Gromacs calculations are restricted to at most a single node on the JADE service.

The following is an example Slurm script to run the code using one of the regression tests from the installation:

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH -J testGromacs
#SBATCH --time=01:00:00
#SBATCH --gres=gpu:4

module load gromacs/2016.3
```

```
mpirun -np $SLURM_NTASKS gmx_mpi mdrun -s topol.tpr -noconfout -resethway -nsteps␣
↪10000 -ntomp 10 -pin on &> run-gromacs.out
```

The example utilises half the resources on a JADE node, with a requests for a single node with 20 tasks and 4 GPUs. Gromacs is started with a number of processes to match the number of GPUs. Also, each process is multithreading, option which is set via *-ntomp*. Process pinning is requested via *-pin on*.

To read more about Gromacs processing on GPUs, please visit https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/gromacs/ .

Installation notes

The latest version of the source was used. The following build instructions were followed: http://www.nvidia.com/object/gromacs-installation.html

The code was compiled using OpenMPI v1.10.5a1 and GCC v4.8.4 with the following command:

```
CC=mpicc CXX=mpicxx cmake /jmain01/home/atostest/Building/gromacs-2016.3
-DGMX_OPENMP=ON
-DGMX_GPU=ON
-DGPU_DEPLOYMENT_KIT_ROOT_DIR=/usr/local/cuda-8.0/targets/x86_64-linux
-DCUDA_TOOLKIT_ROOT_DIR=/usr/local/cuda-8.0/targets/x86_64-linux
-DNVML_INCLUDE_DIR=/usr/local/cuda-8.0/targets/x86_64-linux/include
-DNVML_LIBRARY=/usr/lib/nvidia-375/libnvidia-ml.so
-DHWLOC_INCLUDE_DIRS=/usr/mpi/gcc/openmpi-1.10.5a1/include/openmpi/opal/mca/hwloc/
↪hwloc191/hwloc/include
-DGMX_BUILD_OWN_FFTW=ON
-DGMX_PREFER_STATIC_LIBS=ON
-DCMAKE_BUILD_TYPE=Release
-DGMX_BUILD_UNITTESTS=ON
-DCMAKE_INSTALL_PREFIX=/jmain01/home/atostest/gromacs-2016.3
```

NAMD

Job scripts

NAMD is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems. NAMD scales to hundreds of cores for typical simulations, however NAMD calculations are restricted to at most a single node on the JADE service.

Below is an example of a NAMD job script

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --ntassk-per-node=20
#SBATCH -J testNAMD
#SBATCH --time=01:00:00
#SBATCH --gres=gpu:4

module load NAMD/2.12

$NAMDROOT/namd2 +p$SLURM_NTASKS_PER_NODE +setcpuaffinity +devices $CUDA_VISIBLE_
↪DEVICES ./input.conf &> run.log
```

The above example utilises half the resources on a JADE node, with a requests for a single node with 20 tasks and 4 GPUs.

Because the job is run on a single node, NAMD can be started directly, thus avoiding the use of the launcher *charmrun*. The application is set to run on the resources allocated uwing the *+p* and *+devices* command line options. Additionally, affinity is requested using the option *+setcpuaffinity*.

The general recommendation is to have no more than one process per GPU in the multi-node run, allowing the full utilisation of multiple cores via multi-threading. For single node jobs, the use of multiple GPUs per process is permitted.

To read more about NAMD processing on GPUs, please visit https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/namd/ .

Installation notes

The latest version of the source code was used and built using OpenMPI v1.10.5a1 and GCC v4.8.4 following instructions from http://www.nvidia.com/object/gpu-accelerated-applications-namd-installation.html

Charm++ was built using:

```
./build charm++ verbs-linux-x86_64 gcc smp --with-production
```

NAMD was built using the following:

```
./config Linux-x86_64-g++ --charm-arch verbs-linux-x86_64-smp-gcc --with-cuda --cuda-
↪prefix /usr/local/cuda-8.0
```

For the decision on the number of threads to use per node, take a look at http://www.nvidia.com/object/gpu-accelerated-applications-namd-running-jobs.html

Tensorflow

Tensorflow

URL https://www.tensorflow.org/

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Tensorflow Docker Container

Tensorflow is available on JADE through the use of a Docker container. For more information on JADE's use of containers, see *Using Containerised Applications*.

Using Tensorflow Interactively

You can test that Tensorflow is running on the GPU with the following python code

```
import tensorflow as tf
# Creates a graph.
with tf.device('/gpu:0'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
  c = tf.matmul(a, b)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print(sess.run(c))
```

Which gives the following results

```
[[ 22.  28.]
 [ 49.  64.]]
```

Using Tensorflow in Batch Mode

Using multiple GPUs

Example taken from tensorflow documentation.

If you would like to run TensorFlow on multiple GPUs, you can construct your model in a multi-tower fashion where each tower is assigned to a different GPU. For example:

```
import tensorflow as tf
# Creates a graph.
c = []
for d in ['/gpu:2', '/gpu:3']:
  with tf.device(d):
    a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3])
    b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2])
    c.append(tf.matmul(a, b))
with tf.device('/cpu:0'):
  sum = tf.add_n(c)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print sess.run(sum)
```

You will see the following output.

```
Device mapping:
/job:localhost/replica:0/task:0/gpu:0 -> device: 0, name: Tesla K20m, pci bus
id: 0000:02:00.0
/job:localhost/replica:0/task:0/gpu:1 -> device: 1, name: Tesla K20m, pci bus
id: 0000:03:00.0
/job:localhost/replica:0/task:0/gpu:2 -> device: 2, name: Tesla K20m, pci bus
id: 0000:83:00.0
/job:localhost/replica:0/task:0/gpu:3 -> device: 3, name: Tesla K20m, pci bus
id: 0000:84:00.0
Const_3: /job:localhost/replica:0/task:0/gpu:3
```

```
Const_2: /job:localhost/replica:0/task:0/gpu:3
MatMul_1: /job:localhost/replica:0/task:0/gpu:3
Const_1: /job:localhost/replica:0/task:0/gpu:2
Const: /job:localhost/replica:0/task:0/gpu:2
MatMul: /job:localhost/replica:0/task:0/gpu:2
AddN: /job:localhost/replica:0/task:0/cpu:0
[[  44.   56.]
 [  98.  128.]]
```

Theano

Theano

URL http://deeplearning.net/software/theano/index.html

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano is most commonly used to perform Deep Learning and has excellent GPU support and integration through PyCUDA. The following steps can be used to setup and configure Theano on your own profile.

Theano Docker Container

Theano is available on JADE through the use of a Docker container. For more information on JADE's use of containers, see *Using Containerised Applications*.

Using Theano Interactively

Using Theano in Batch Mode

Reserving Memory to Improve Theano Performance

For optimal Theano performance, enable the CUDA memory manager CNMeM. To do this, create the .theanorc file in your HOME directory and set the fraction of GPU memory reserved by Theano. The exact amount of energy may have to be hand-picked: if Theano asks for more memory that is currently available on the GPU, an error will be thrown during import of theano module. Create or edit the `.theanorc` file with nano:

```
nano ~/.theanorc
```

Add the following lines and, if necessary, change the 0.8 number to whatever works for you

```
[lib]
cnmem=0.8
```

Run python and verify that Theano is working correctly

```
python -c "import theano;theano.test()"
```

Torch

Torch

URL http://torch.ch/

Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.

Torch Docker Container

Theano is available on JADE through the use of a Docker container. For more information on JADE's use of containers, see *Using Containerised Applications*.

Using Torch Interactively

Using Torch in Batch Mode

Development Tools on JADE

Libraries on JADE

More Information

JADE Web site: http://www.arc.ox.ac.uk/content/jade

Mike Giles' Web site: http://people.maths.ox.ac.uk/~gilesm/JADE/

STFC Web site: https://www.hartree.stfc.ac.uk/Pages/Hartree-Centre-welcomes-new-HPC-computing-facility-to-support-machine-lear
aspx

Troubleshooting

Univesities provide coordinated institutional technical support to assist local users and their developers with GPU applications and in accessing the JADE HPC facility. For more information, contact Research Software Engineers at your home univestrity.

The source is available on GitHub and contributions are always encouraged. Contributions can be as simple as minor tweaks to this documentation, the website or the core. To contribute, fork the project on GitHub and send a pull request.