Cirrus Documentation

Release 1.0

EPCC

Aug 03, 2021

Tesseract User Guide

The DiRAC Extreme Scaling service is an HPC service hosted and run by The University of Edinburgh and EPCC. It is part of the STFC DiRAC National HPC Service.

DiRAC Extreme Scaling (also known as Tesseract) is available to industry, commerce and academic researchers. For information on how to get access to the system please see the DiRAC website.

The Tesseract compute service is based around an HPE SGI 8600 system with 1476 compute nodes. There are 1468 standard compute nodes, each with two 2.1 GHz, 12-core Intel Xeon (Skylake) Silver 4116 processors and 96 GB of memory. In addition, there are 8 GPU compute nodes each with two 2.1 GHz, 12-core Intel Xeon (Skylake) Silver 4116 processors; 96 GB of memory; and 4 NVidia V100 (Volta) GPU accelerators connected over NVlink. All compute nodes are connected together by a single Intel Omni-Path fabric and all nodes access the 3 PB Lustre file system. As well as the fast, parallel Lustre storage, Tesseract also provides a tiered storage solution based on zero watt disk storage and tape storage built using HPE DMF.

This documentation covers:

- Tesseract User Guide: general information on how to use Tesseract
- Software Libraries: notes on compiling against specific libraries on Tesseract. Most libraries work *as expected* so no additional notes are required however a small number require specific documentation

Information on using the SAFE web interface for managing accounts and reporting on your usage on Tesseract (and DiRAC as a whole) can be found on the DiRAC SAFE Documentation

This documentation draws on the Cirrus Tier2 National HPC Service Documentation and the documentation for the ARCHER National Supercomputing Service.

CHAPTER 1

Introduction

This guide is designed to be a reference for users of the high-performance computing (HPC) facility: Tesseract. It provides all the information needed to access the system, transfer data, manage your resources (disk and compute time), submit jobs, compile programs and manage your environment.

1.1 Acknowledging Tesseract

You should use the following phrase to acknowledge Tesseract in all research outputs that have used the facility:

This work used the DiRAC Extreme Scaling HPC Service (https://www.dirac.ac.uk).

You should also tag outputs with the keyword *DiRAC* whenever possible.

1.2 Hardware

The current Extreme Scaling compute provision (Tesseract) consists of 1468 standard compute nodes and 8 GPU compute nodes connected together by a single Intel OPA fabric.

There are 2 login nodes that share a common software environment and file system with the compute nodes.

1.2.1 Standard Compute Nodes

Tesseract standard compute nodes each contain two 2.1 GHz, 12-core Intel Xeon Silver 4116 (Skylake) series processors. Each of the cores in these processors support 2 hardware threads (Hyperthreads), which are disabled by default.

There are 1468 standard compute nodes on Tesseract giving a total of 35,232 cores.

The compute nodes on Tesseract have 96 GB of memory shared between the two processors. The memory is arranged in a non-uniform access (NUMA) form: each 12-core processor is a single NUMA region with local memory of 48

GB. Access to the local memory by cores within a NUMA region has a lower latency than accessing memory on the other NUMA region.

There are three levels of cache, configured as follows:

- L1 Cache 32 KB Instr., 32 KB Data (per core)
- L2 Cache 1 MB (per core)
- L3 Cache 16.5 MB (shared)

1.2.2 GPU Compute Nodes

Tesseract GPU compute nodes each contain two 2.1 GHz, 12-core Intel Xeon Silver 4116 (Skylake) series processors. Each of the cores in these processors support 2 hardware threads (Hyperthreads), which are disabled by default. The nodes also each contain four NVIDIA Tesla V100-PCIE-16GB (Volta) GPU accelerators connected to the host processors and each other via [NVLink](https://www.nvidia.com/en-gb/data-center/nvlink/).

There are 8 GPU compute nodes on Tesseract giving a total of 192 cores and 64 GPU accelerators.

The compute nodes on Tesseract have 96 GB of memory shared between the two processors. The memory is arranged in a non-uniform access (NUMA) form: each 12-core processor is a single NUMA region with local memory of 48 GB. Access to the local memory by cores within a NUMA region has a lower latency than accessing memory on the other NUMA region.

There are three levels of cache, configured as follows:

- L1 Cache 32 KB Instr., 32 KB Data (per core)
- L2 Cache 1 MB (per core)
- L3 Cache 16.5 MB (shared)

1.2.3 OPA Interconnect

The system has a single Intel OPA fabric and every compute node and login node has a single OPA interface. The Lustre file system servers have two connections to the OPA fabric and all Lustre file system IO traverses the OPA fabric.

1.2.4 File systems and Data Infrastructure

There is currently a single Lustre parallel file system available on Tesseract: /tessfs1 is a Lustre parallel file system desgined to give high read/write bandwidth for parallel I/O operations.

The Lustre file system has a total of 3 PB available. The login and compute nodes mount the storage as /tessfs1, and all home and work directories are available on all nodes.

The compute nodes are diskless. Each node boots from a cluster management noded called the Rack Leader and NFS mounts the root file system from this management node.

Note: Data on the Lustre file system is automatically backed up to a separate tape library.

1.2.5 Parallel I/O

For a description of the terms associated with Lustre parallel file systems please see the description on Wikipedia:

- Lustre File Systems Description

The default striping on the Lustre filesystem is 1 stripe, and the default stripe size is 1 MiB.

CHAPTER 2

Connecting to Tesseract

On the Tesseract system, interactive access can be achieved via SSH, either directly from a command line terminal or using an SSH client. In addition data can be transferred to and from the Tesseract system using `scp` from the command line or by using a file transfer client.

This section covers the basic connection methods.

2.1 Access credentials

To access Tesseract, you need to use two credentials: your password and an SSH key pair protected by a passphrase. You can find more detailed instructions on how to set up your credentials to access Tesseract from Windows, macOS and Linux below.

2.1.1 SSH Key Pairs

You will need to generate an SSH key pair protected by a passphrase to access Tesseract.

Using a terminal (the command line), set up a key pair that contains your e-mail address and enter a passphrase you will use to unlock the key:

```
ssh-keygen -t rsa -C "your@email.com"
...
-bash-4.1$ ssh-keygen -t rsa -C "your@email.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Home/user/.ssh/id_rsa): [Enter]
Enter passphrase (empty for no passphrase): [Passphrase]
Enter same passphrase again: [Passphrase]
Your identification has been saved in /Home/user/.ssh/id_rsa.
Your public key has been saved in /Home/user/.ssh/id_rsa.pub.
The key fingerprint is:
03:d4:c4:6d:58:0a:e2:4a:f8:73:9a:e8:e3:07:16:c8 your@email.com
The key's randomart image is:
```

(continues on next page)

(continued from previous page)

```
+--[ RSA 2048]----+
|     . ...+o++++. |
| . . . =o..       |
|+ . . .......o o  |
|oE .   .          |
|o =      .   S    |
|.    +.+     .    |
|.  oo             |
|.  .              |
| ..               |
+-----------------+
```

(remember to replace "your@email.com" with your e-mail address).

2.1.2 Upload public part of key pair to DiRAC SAFE

You should now upload the public part of your SSH key pair to the DiRAC SAFE by following the instructions at:

- Upload your SSH key to SAFE

Once you have done this, your SSH key will be added to your Tesseract account.

Remember, you will need to use both an SSH key and password to log into Tesseract so you will also need to collect your initial password before you can log into Tesseract. We cover this next.

2.1.3 Initial passwords

The SAFE web interface is used to provide your initial password for logging onto Tesseract (see the DiRAC SAFE Documentation for more details on requesting accounts and picking up passwords).

Note: you may now change your password on the Tesseract machine itself using the *passwd* command or when you are prompted the first time you login. This change will not be reflected in the SAFE. If you forget your password, you should use the SAFE to request a new one-shot password.

2.2 SSH Clients

Interaction with Tesseract is done remotely, over an encrypted communication channel, Secure Shell version 2 (SSH-2). This allows command-line access to one of the login nodes of a Tesseract, from which you can run commands or use a command-line text editor to edit files. SSH can also be used to run graphical programs such as GUI text editors and debuggers when used in conjunction with an X client.

2.2.1 Logging in from Linux and MacOS

Linux distributions and MacOS each come installed with a terminal application that can be use for SSH access to the login nodes. Linux users will have different terminals depending on their distribution and window manager (e.g. GNOME Terminal in GNOME, Konsole in KDE). Consult your Linux distribution's documentation for details on how to load a terminal.

MacOS users can use the Terminal application, located in the Utilities folder within the Applications folder.

You can use the following command from the terminal window to login into Tesseract:

```
ssh username@tesseract.dirac.ed.ac.uk
```

You will first be prompted for the passphrase associated with your SSH key pair. Once you have entered your passphrase successfully, you will then be prompted for your password. You need to enter both correctly to be able to access Tesseract.

Note: If your SSH key pair is not stored in the default location (usually `~/.ssh/id_rsa`) on your local system, you may need to specify the path to the private part of the key wih the `-i` option to `ssh`. For example, if your key is in a file called `keys/id_rsa_tesseract` you would use the command `ssh -i keys/id_rsa_tesseract username@tesseract.dirac.ed.ac.uk` to log in.

Note: When you first log into Tesseract, you will be prompted to change your initial password. This is a three step process:

1. When promoted to enter your ldap password: Re-enter the password you retrieved from SAFE
2. When prompted to enter your new password: type in a new password
3. When prompted to re-enter the new password: re-enter the new password

Your password has now been changed

To allow remote programs, especially graphical applications to control your local display, such as being able to open up a new GUI window (such as for a debugger), use:

```
ssh -X username@tesseract.dirac.ed.ac.uk
```

Some sites recommend using the `-Y` flag. While this can fix some compatibility issues, the `-X` flag is more secure.

Current MacOS systems do not have an X window system. Users should install the XQuartz package to allow for SSH with X11 forwarding on MacOS systems:

- XQuartz website

2.2.2 Logging in from Windows using MobaXterm

A typical Windows installation will not include a terminal client, though there are various clients available. We recommend all our Windows users to download and install MobaXterm to access Tesseract. It is very easy to use and includes an integrated X server with SSH client to run any graphical applications on Tesseract.

You can download MobaXterm Home Edition (Installer Edition) from the following link:

- Install MobaXterm

Double-click the downloaded Microsoft Installer file (.msi), and the Windows wizard will automatically guides you through the installation process. Note, you might need to have administrator rights to install on some Windows OS. Also make sure to check whether Windows Firewall hasn't blocked any features of this program after installation.

Start MobaXterm using, for example, the icon added to the Start menu during the installation process.

If you would like to run any small remote GUI applications, then make sure to use -X option along with the ssh command (see above) to enable X11 forwarding, which allows you to run graphical clients on your local X server.

2.3 Making access more convenient using the SSH configuration file

Typing in the full command to login or transfer data to Tesseract can become tedious as it often has to be repeated many times. You can use the SSH configuration file, usually located on your local machine at `.ssh/config` to make things a bit more convenient.

Each remote site (or group of sites) can have an entry in this file which may look something like:

```
Host tesseract
  HostName tesseract.dirac.ed.ac.uk
  User username
```

(remember to replace `username` with your actual username!).

The `Host tesseract` line defines a short name for the entry. In this case, instead of typing `ssh username@tesseract.dirac.ed.ac.uk` to access the Tesseract login nodes, you could use `ssh tesseract` instead. The remaining lines define the options for the `tesseract` host.

- `Hostname tesseract.dirac.ed.ac.uk` - defines the full address of the host
- `User username` - defines the username to use by default for this host (replace `username` with your own username on the remote host)

Now you can use SSH to access Tesseract without needing to enter your username or the full hostname every time:

```
-bash-4.1$ ssh tesseract
```

You can set up as many of these entries as you need in your local configuration file. Other options are available. See the ssh_config man page (or `man ssh_config` on any machine with SSH installed) for a description of the SSH configuration file. You may find the `IdentityFile` option useful if you have to manage multiple SSH key pairs for different systems as this allows you to specify which SSH key to use for each system.

CHAPTER 3

Data Transfer Guide

This section covers the different ways that you can transfer data on to and off Tesseract.

3.1 scp command

The `scp` command creates a copy of a file, or if given the `-r` flag a directory, on a remote machine. Below shows an example of the command to transfer a single file to Tesseract:

```
scp [options] source_file user@tesseract.dirac.ed.ac.uk:[destination]
```

In the above example, the `[destination]` is optional, as when left out scp will simply copy the source into the users home directory.

3.2 rsync command

The `rsync` command creates a copy of a file, or if given the `-r` flag a directory, at the given destination, similar to scp above. However, given the -a option rsync can also make exact copies (including permissions), this is referred to as 'mirroring'. In this case the `rsync` command is executed with ssh to create the copy of a remote machine. To transfer files to Tesseract the command should have the form:

```
rsync [options] -e ssh source user@tesseract.dirac.ed.ac.uk:[destination]
```

In the above example, the `[destination]` is optional, as when left out rsync will simply copy the source into the users home directory.

3.3 Performance considerations

Tesseract is capable of generating data at a rate far greater than the rate at which this can be downloaded. In practice, it is expected that only a portion of data generated on Tesseract will be required to be transfered back to users' local

storage - the rest will be, for example, intermediate or checkpoint files required for subsequent runs. However, it is still essential that all users try to transfer data to and from Tesseract as efficiently as possible. The most obvious ways to do this are:

1. Only transfer those files that are required for subsequent analysis, visualisation and/or archiving. Do you really need to download those intermediate result or checkpointing files? Probably not.
2. Combine lots of small files into a single tar file, to reduce the overheads associated in initiating data transfers.
3. Compress data before sending it, e.g. using gzip or bzip2.
4. Consider doing any pre- or post-processing calculations on Tesseract. Long running pre- or post- processing calculations should be run via the batch queue system, rather than on the login nodes. Such pre- or post-processing codes could be serial or OpenMP parallel applications running on a single node, though if the amount of data to be processed is large, an MPI application able to use multiple nodes may be preferable.

Note: The performance of data transfers between Tesseract and your local institution may differ depending upon whether the transfer command is run on Tesseract or on your local system.

CHAPTER 4

File and Resource Management

This section covers some of the tools and technical knowledge that will be key to maximising the usage of the Tesseract system, such as the online administration tool SAFE and calculating the CPU-time available.

The default file permissions are then outlined, along with a description of changing these permissions to the desired setting. This leads on to the sharing of data between users and systems often a vital tool for project groups and collaboration.

Finally we cover some guidelines for I/O and data archiving on Tesseract.

4.1 The Tesseract Administration Web Site (SAFE)

All users have a login and password on the Tesseract Administration Web Site (also know as the 'SAFE'): SAFE. Once logged into this web site, users can find out much about their usage of the Tesseract system, including:

- Account details - password reset, change contact details
- Project details - project code, start and end dates
- CPUh balance - how much time is left in each project you are a member of
- Filesystem details - current usage and quotas
- Reports - generate reports on your usage over a specified period, including individual job records
- Helpdesk - raise queries and track progress of open queries

4.2 Checking your CPU-time allocation

Use the *Login accounts* menu to select the user account that you wish to query. The page for the login account will summarise the resources available to account.

You can also generate reports on your usage over a particular period and examine the details of how many CPUh individual jobs on the system cost. To do this use the *Service information* menu and selet *Report generator*.

4.3 Disk quotas

Disk quotas on Tesseract are managed via SAFE

4.4 File permissions and security

By default, each user is a member of the group with the same name as [group_code] in the `/lustre/home` directory path, e.g. `x01`. This allows the user to share files with only members of that group by setting the appropriate group file access permissions. As on other UNIX or Linux systems, a user may also be a member of other groups. The list of groups that a user is part of can be determined by running the `groups` command.

Default Unix file permissions can be specified by the `umask` command. The default umask value on Tesseract is 22, which provides "group" and "other" read permissions for all files created, and "group" and "other" read and execute permissions for all directories created. This is highly undesirable, as it allows everyone else on the system to access (but at least not modify or delete) every file you create. Thus it is strongly recommended that users change this default umask behaviour, by adding the command `umask 077` to their `$HOME/.profile` file. This umask setting only allows the user access to any file or directory created. The user can then selectively enable "group" and/or "other" access to particular files or directories if required.

4.5 DMF tiered storage

As well as the fast, parallel Lustre storage, Tesseract also provides a tiered storage solution based on zero watt disk storage and tape storage. This system is built on the HPE DMF solution and provides two functions:

- Backup of all data on the Tesseract Lustre file system
- Policy-based release of data that has not been accessed for a long period from Lustre to tape storage to free up space on the Lustre file system.

Files released from Lustre by the policy method are still visible on the Lustre file system but will suffer from long access times as they are retrieved from tape. Users can query the status of files to find out if they are on Lustre or tape and can request the retrieval of data on tape. This allows users to retrieve data before it is required by compute jobs so that they are not held up by the long access times.

Below we cover the commands for querying and retrieving data that has been released to tape via the DMF policies.

4.5.1 DMF data movement policies

Currently, the policies in place on Tesseract run daily to automatically release any files from Lustre (but retain on tape with a file stub on Lustre) based on **both** of the following criteria being true:

- The file is larger than 1 GiB in size; and
- the file has not been accessed in the past three months.

In addition, the policies only start running when the Lustre file system is more than 80% full and stop running when the Lustre file system reaches 75% full.

4.5.2 Checking the status of files: `lfs hsm_state`

Warning: You should not use any of the `lfs` HSM subcommands other than those documented in this User Guide (even though they may be documented elsewhere on the web). Use of other subcommands may result in loss of data with no way to get it back.

You use the `lfs hsm_state` (HSM == Hierarchical Storage Manager) command to check the state of a particular file. For example

```
[dc-user1@tesseract-login1 ~]$ lfs hsm_state testfile
testfile: (0x00000000)
```

There is no additional status associated with the file `testfile` so it is currently stored on the Lustre storage.

Looking at a file that has released from the Lustre file system via the DMF policies:

```
[dc-user1@tesseract-login1 ~]$ lfs hsm_state large_file.dat
large-file.dat: (0x0000000d) released exists archived, archive_id:1
```

Shows additional status information indicating that it has been archived to tape and the space released from Lustre to be reused for other data.

4.5.3 Retrieving files from tape

Warning: You should not use any of the `lfs` HSM subcommands other than those documented in this User Guide (even though they may be documented elsewhere on the web). Use of other subcommands may result in loss of data with no way to get it back.

You can retrieve a from tape simply by trying to access it; however, your terminal session or program will stall while waiting for the data to be retrieved from tape.

Usually, you will want to retrieve the data in the background ahead of when you want to use it. To retrieve in the background, first update the last access time (using `touch`) so the file will not be a prime candidate for release again and ask for it to be retrieved using the `lfs hsm_restore` command:

```
[dc-user1@tesseract-login1 ~]$ touch large-file.dat
[dc-user1@tesseract-login1 ~]$ lfs hsm_restore large-file.dat
```

After a while the `released` tag will disappear from the file so you know that it has been restored:

```
[dc-user1@tesseract-login1 ~]$ lfs hsm_state large-file.dat
large-file.dat: (0x00000009) exists archived, archive_id:1
```

As for the status command, the `lfs hsm_restore` command does not support globbing so to restore multiple files you will need to use the `xargs` command.

4.6 File system use

4.6.1 ASCII (or formatted) files

These are the most portable, but can be extremely inefficient to read and write. There is also the problem that if the formatting is not done correctly, the data may not be output to full precision (or to the subsequently required precision), resulting in inaccurate results when the data is used. Another common problem with formatted files is FORMAT statements that fail to provide an adequate range to accommodate future requirements, e.g. if we wish to output the total number of processors, NPROC, used by the application, the statement:

```
WRITE (*,'I3') NPROC
```

will not work correctly if NPROC is greater than 999.

4.6.2 Binary (or unformatted) files

These are much faster to read and write, especially if an entire array is read or written with a single READ or WRITE statement. However the files produced may not be readable on other systems.

GNU compiler `-fconvert=swap` compiler option. This compiler option often needs to be used together with a second option `-frecord-marker`, which specifies the length of record marker (extra bytes inserted before or after the actual data in the binary file) for unformatted files generated on a particular system. To read a binary file generated by a big-endian system on Tesseract, use `-fconvert=swap -frecord-marker=4`. Please note that due to the same 'length of record marker' reason, the unformatted files generated by GNU and other compilers on Tesseract are not compatible. In fact, the same WRITE statements would result in slightly larger files with GNU compiler. Therefore it is recommended to use the same compiler for your simulations and related pre- and post-processing jobs.

Other options for file formats include:

Direct access files Fortran unformatted files with specified record lengths. These may be more portable between different systems than ordinary (i.e. sequential IO) unformatted files, with significantly better performance than formatted (or ASCII) files. The "endian" issue will, however, still be a potential problem.

Portable data formats These machine-independent formats for representing scientific data are specifically designed to enable the same data files to be used on a wide variety of different hardware and operating systems. The most common formats are:

- netCDF: http://www.unidata.ucar.edu/software/netcdf/
- HDF: http://www.hdfgroup.org/HDF5/

It is important to note that these portable data formats are evolving standards, so make sure you are aware of which version of the standard/software you are using, and keep up-to-date with any backward-compatibility implications of each new release.

4.7 File IO Performance Guidelines

Here are some general guidelines

- Whichever data formats you choose, it is vital that you test that you can access your data correctly on all the different systems where it is required. This testing should be done as early as possible in the software development or porting process (i.e. before you generate lots of data from expensive production runs), and should be repeated with every major software upgrade.

- Document the file formats and metadata of your important data files very carefully. The best documentation will include a copy of the relevant I/O subroutines from your code. Of course, this documentation must be kept up-to-date with any code modifications.
- Use binary (or unformatted) format for files that will only be used on the Intel system, e.g. for checkpointing files. This will give the best performance. Binary files may also be suitable for larger output data files, if they can be read correctly on other systems.
- Most codes will produce some human-readable (i.e. ASCII) files to provide some information on the progress and correctness of the calculation. Plan ahead when choosing format statements to allow for future code usage, e.g. larger problem sizes and processor counts.
- If the data you generate is widely shared within a large community, or if it must be archived for future reference, invest the time and effort to standardise on a suitable portable data format, such as netCDF or HDF.

4.8 Backup policies

There are currently no backups of data on Tesseract as backing up the whole Lustre file system would adversly affect the performance of write access for simulations. We strongly advise that you keep copies of any critical data on different systems.

CHAPTER 5

Application Development Environment

The application development environment on Tesseract is primarily controlled through the *modules* environment. By loading and switching modules you control the compilers, libraries and software available.

This means that for compiling on Tesseract you typically set the compiler you wish to use using the appropriate modules, then load all the required library modules (e.g. numerical libraries, IO format libraries).

Additionally, if you are compiling parallel applications using MPI (or SHMEM, etc.) then you will need to load the MPI environment and use the appropriate compiler wrapper scripts.

By default, all users on Tesseract start with no modules loaded.

Basic usage of the `module` command on Tesseract is covered below. For full documentation please see:

- Linux manual page on modules

5.1 Using the modules environment

5.1.1 Information on the available modules

Finding out which modules (and hence which compilers, libraries and software) are available on the system is performed using the `module avail` command:

```
[user@tesseract-login1 ~]$ module avail
...
```

This will list all the names and versions of the modules available on the service. Not all of them may work in your account though due to, for example, licencing restrictions. You will notice that for many modules we have more than one version, each of which is identified by a version number. One of these versions is the default. As the service develops the default version will change.

You can list all the modules of a particular type by providing an argument to the `module avail` command. For example, to list all available versions of the Intel libraries, compilers and tools:

```
[user@tesseract-login1 ~]$ module avail intel

------------------------------ /tessfs1/sw/modulefiles -------------------------------
intel-cc-18/18.1.163    intel-fc-18/18.1.163    intel-tools-18
intel-cmkl-18/18.1.163  intel-mpi-18/18.1.163   intel-vtune-18/18.1.163
```

If you want more info on any of the modules, you can use the `module help` command:

```
[user@tesseract-login1 ~]$ module help intel-cmkl-18/18.1.163

----------- Module Specific Help for 'intel-cmkl-18/18.1.163' ------------------------
↪---

Sets up the paths for Intel Cluster Math Kernal Library 18.1.163
```

The simple `module list` command will give the names of the modules and their versions you have presently loaded in your envionment:

```
[user@tesseract-login1 ~]$ module list
Currently Loaded Modulefiles:
  1) intel-cc-18/18.1.163      4) intel-mpi-18/18.1.163
  2) intel-fc-18/18.1.163      5) intel-vtune-18/18.1.163
  3) intel-cmkl-18/18.1.163    6) intel-tools-18
```

5.1.2 Loading, unloading and swapping modules

To load a module to use `module add` or `module load`. For example, to load the Intel Fortran compilers into the development environment:

```
module load intel-fc-18
```

This will load the default version of the intel fortran compilers. If you need a specfic version of the module, you can add more information:

```
module load intel-fc-18/18.1.163
```

will load version 18.1.163 for you, regardless of the default. If you want to clean up, `module remove` will remove a loaded module:

```
module remove intel-fc-18
```

(or `module rm intel-fc-18` or `module unload intel-fc-18`) will unload what ever version of intel-fc-17 (even if it is not the default) you might have loaded. There are many situations in which you might want to change the presently loaded version to a different one, such as trying the latest version which is not yet the default or using a legacy version to keep compatibility with old data. This can be achieved most easily by using `module swap oldmodule newmodule`.

5.2 Available Compiler Suites

Note: As Tesseract uses dynamic linking by default you will generally also need to load any modules you used to compile your code in your job submission script when you run your code.

5.2.1 Intel Compiler Suite

The Intel compiler suite is accessed by loading the `intel-tools-*` module. For example:

```
module load intel-tools-18
```

Once you have loaded the module, the compilers are available as:

- `ifort` - Fortran
- `icc` - C
- `icpc` - C++

5.2.2 GCC Compiler Suite

The GCC 4.8.5 compiler suite is available by default without loading any modules.

The compilers are available as:

- `gfortran` - Fortran
- `gcc` - C
- `g++` - C++

5.3 Compiling MPI codes

Tesseract currently supports the Intel MPI library.

You should also consult the chapter on running jobs through the batch system for examples of how to run jobs compiled against MPI.

Note: By default, all compilers produce dynamic executables on Tesseract. This means that you must load the same modules at runtime (usually in your job submission script) as you have loaded at compile time.

5.3.1 Using Intel MPI

To compile MPI code with Intel MPI, using any compiler, you must first load the "intel-mpi-18" module:

```
module load intel-mpi-18
```

(If you loaded the `intel-tools-18` module then this automatically loads the Intel MPI module for you.)

This makes the compiler wrapper scripts available to you. The name of the wrapper script depends on the compiler suite you are using. In summary:

Language	Intel	GCC
Fortran	mpiifort	mpif90
C++	mpiicpc	mpicxx
C	mpiicc	mpicc

Further details on using the different compiler suites with Intel MPI are available in the following sections.

Using Intel Compilers and Intel MPI

You should make the Intel compilers and MPI environment available by loading the `intel-tools-18` module:

```
module load intel-tools-18
```

MPI compilers are then available as

- `mpiifort` - Fortran with MPI
- `mpiicc` - C with MPI
- `mpiicpc` - C++ with MPI

Note: Intel compilers with Intel MPI use non-standard compiler wrapper script names. If you use the standard names you will end up using the GCC compilers.

Using GCC Compilers and Intel MPI

Once you have loaded the `intel-tools-18` module, MPI compilers are then available as

- `mpif90` - Fortran with MPI
- `mpicc` - C with MPI
- `mpicxx` - C++ with MPI

5.4 Compiler Information and Options

Help is available for the different compiler suites

GCC Fortran `gfortran --help` , C/C++ `gcc --help`

Intel Fortran `man ifort` , C/C++ `man icc`

5.4.1 Useful compiler options

Note: For best performance on Tesseract we currently advise that you should use the Intel compilers wherever possible as the version of GCC available on the system is very old. We aim to install a more up to date version of GCC soon.

Whilst difference codes will benefit from compiler optimisations in different ways, for reasonable performance on Tesseract, at least initially, we suggest the following compiler options:

Intel `-O2`

GNU `-O2 -ftree-vectorize -funroll-loops -ffast-math`

When you have a application that you are happy is working correctly and has reasonable performance you may wish to investigate some more aggressive compiler optimisations. Below is a list of some further optimisations that you can try on your application (Note: these optimisations may result in incorrect output for programs that depend on an exact implementation of IEEE or ISO rules/specifications for math functions):

Intel `-fast`

GNU `-Ofast -funroll-loops`

Vectorisation, which is one of the important compiler optimisations for Tesseract, is enabled by default as follows:

Intel At `-O2` and above

GNU At `-O3` and above or when using `-ftree-vectorize`

To promote integer and real variables from four to eight byte precision for Fortran codes the following compiler flags can be used:

Intel `-real-size 64 -integer-size 64 -xAVX` (Sometimes the Intel compiler incorrectly generates AVX2 instructions if the `-real-size 64` or `-r8` options are set. Using the `-xAVX` option prevents this.)

GNU `-freal-4-real-8 -finteger-4-integer-8`

5.5 Using static linking/libraries

By default, executables on Tesseract are built using shared/dynamic libraries (that is, libraries which are loaded at run-time as and when needed by the application) when using the wrapper scripts.

An application compiled this way to use shared/dynamic libraries will use the default version of the library installed on the system (just like any other Linux executable), even if the system modules were set differently at compile time. This means that the application may potentially be using slightly different object code each time the application runs as the defaults may change. This is usually the desired behaviour for many applications as any fixes or improvements to the default linked libraries are used without having to recompile the application, however some users may feel this is not the desired behaviour for their applications.

Alternatively, applications can be compiled to use static libraries (i.e. all of the object code of referenced libraries are contained in the executable file). This has the advantage that once an executable is created, whenever it is run in the future, it will always use the same object code (within the limit of changing runtime environemnt). However, executables compiled with static libraries have the potential disadvantage that when multiple instances are running simultaneously multiple copies of the libraries used are held in memory. This can lead to large amounts of memory being used to hold the executable and not application data.

To create an application that uses static libraries you must pass an extra flag during compilation, `-Bstatic`.

Use the UNIX command `ldd exe_file` to check whether you are using an executable that depends on shared libraries. This utility will also report the shared libraries this executable will use if it has been dynamically linked.

CHAPTER 6

Running Jobs on Tesseract

The Tesseract facility uses PBSPro to schedule jobs.

Writing a submission script is typically the most convenient way to submit your job to the job submission system. Example submission scripts (with explanations) for the most common job types are provided below.

Interactive jobs are also available and can be particularly useful for developing and debugging applications. More details are available below.

If you have any questions on how to run jobs on Tesseract do not hesitate to contact the DiRAC Helpdesk at dirac-support@epcc.ed.ac.uk.

6.1 Using PBSPro

You typically interact with PBS by (1) specifying PBS directives in job submission scripts (see examples below) and (2) issuing PBS commands from the login nodes.

There are three key commands used to interact with the PBS on the command line:

- `qsub`
- `qstat`
- `qdel`

Check the PBS `man` page for more advanced commands:

```
man pbs
```

6.1.1 The qsub command

The qsub command submits a job to PBS:

```
qsub job_script.pbs
```

This will submit your job script "job_script.pbs" to the job-queues. See the sections below for details on how to write job scripts.

6.1.2 The qstat command

Use the command qstat to view the job queue. For example:

```
qstat
```

will list all jobs on Tesseract.

You can view just your jobs by using:

```
qstat -u $USER
```

The `-a` option to qstat provides the output in a more useful format.

To see more information about a queued job, use:

```
qstat -f $JOBID
```

This option may be useful when your job fails to enter a running state. The output contains a PBS `comment` field which may explain why the job failed to run.

6.1.3 The qdel command

Use this command to delete a job from Tesseract's job queue. For example:

```
qdel $JOBID
```

will remove the job with ID `$JOBID` from the queue.

6.2 Queue Limits

Queues on Tesseract are designed to enable users to use the system flexibly while retaining fair access for all.

The queues on Tesseract are linked to the physical layout of the hardware on the interconnect to help ensure that jobs have access to nodes with the optimal layout on the interconnet topology. In practice, this means that jobs are limited to specific numbers of nodes and will be rejected unless those numbers of nodes are selected using PBS as described below. The node numbers currently supported on Tesseract are:

- 16 nodes (384 cores)
- 32 nodes (768 cores)
- 64 nodes (1536 cores)
- 128 nodes (3072 cores)
- 256 nodes (6144 cores)
- 512 nodes (12288 cores)
- 1024 nodes (24576 cores) - available by request only, contact dirac-support@epcc.ed.ac.uk

The exception to this layout are the 8 GPU compute nodes where jobs can range from 1 to 8 nodes.

- The maximum runtime for jobs in the standard CPU queues on Tesseract is currently 48 hours

- The maximum runtime for jobs in the standard GPU queues on Tesseract is currently 24 hours

6.3 Low priority queues

Each of the queues on Tesseract has a low priority counterpart. Jobs that run in the low priority queues are not charged against the user's budget but have a much lower priority than charged jobs so will only run if there are no eligbile charged jobs waiting for resources.

6.4 Output from PBS jobs

PBS produces standard output and standard error for each batch job can be found in files `<jobname>.o<Job ID>` and `<jobname>.e<Job ID>` respectively. These files appear in the job's working directory once your job has completed or its maximum allocated time to run (i.e. wall time, see later sections) has ran out.

6.5 Running Parallel Jobs

This section describes how to write job submission scripts specifically for different kinds of parallel jobs on Tesseract.

All parallel job submission scripts require (as a minimum) you to specify three things:

- The number of nodes you require via the `-l select=[Nodes]` option. Each node has 24 cores (2x 12-core processors). For example, to select 16 nodes (384 cores in total) you would use `-l select=16`. Remember that only certain node counts are permitted (see the Queue Limits section above for more details).
- The walltime. This can be specified in two ways: as a hard `walltime` which sets the maximum walltime for the job or as a "shrink-to-fit" job where a `min_walltime` and a `max_walltime` are set and the scheduler chooses a walltime between these extremes to start the job as soon as possible. + The maximum length of time (i.e. walltime) you want the job to run

  for via the `-l walltime=[hh:mm:ss]` option. To ensure the minimum wait time for your job, you should specify a walltime as short as possible for your job (i.e. if your job is going to run for 3 hours, do not specify 12 hours). On average, the longer the walltime you specify, the longer you will queue for.

  – Shrink-to-fit jobs specify a `min_walltime` and `max_walltime` and the scheduler picks a walltime between these valuse that starts the job as soon as possible. Once a job starts, you can find the chosen walltime using the `qstat -f <jobid>` command.
- The placement option '-l place=scatter:excl' to ensure that parallel processes/threads are scheduled to the full set of compute nodes assigned to the job.
- The project code that you want to charge the job to via the `-A [project code]` option

In addition to these mandatory specifications, there are many other options you can provide to PBS. The following options may be useful:

- The name of the job can be set with the `-N <name>` option. The name will be used in various places. In particular it will be used in the queue listing and to generate the name of your output and/or error file(s). Note there is a limit on the size of the name.
- If you wish to use the GPU compute nodes you should specify the `QGPU` queue with the `-q QGPU` option.

Note: All compute nodes on Tesseract are run in exclusive mode. This means that only one job at a time can run on any compute node.

6.6 Shrink-to-fit jobs

Shrink-to-fit jobs specify a `min_walltime` and `max_walltime` and the scheduler picks a walltime between these valuse that starts the job as soon as possible. Once a job starts, you can find the chosen walltime using the `qstat -f <jobid>` command.

For example, suppose your typical job requires 48 hours of wall time. If there are fewer than 48 hours available in an upcoming slot then the job cannot run. However, if you know that your job can do enough useful work running for 24h days or longer, you can submit it in the following way:

```
qsub -l min_walltime=24:00:00,max_walltime=48:00:00 job_script
```

When PBS attempts to run your job, it will initially look for a time slot of 48 hours. When no such time slot is found, it will look for shorter and shorter time slots, down to the minimum wall time of 24 hours and, if it finds an acceptable time slot, the job will start.

6.7 Running MPI parallel jobs

When you running parallel jobs requiring MPI you will use an MPI launch command to start your executable in parallel.

6.7.1 Intel MPI

Intel MPI is accessed at runtime by loading the `intel-mpi-18`.

```
module load intel-mpi-18
```

but it is usually added through the `intel-tools-18` module which sets up the Intel compilers and asscociated libraries.

Intel MPI: parallel job launcher `mpirun`

The Intel MPI parallel job launcher on Tesseract is `mpirun`.

Note: This parallel job launcher is only available once you have loaded the `intel-mpi-18` module (usually via the `intel-tools-18` module).

A sample MPI launch line using `mpirun` looks like:

```
mpirun -n 384 -ppn 24 ./my_mpi_executable.x arg1 arg2
```

This will start the parallel executable `my_mpi_executable.x` with arguments "arg1" and "arg2". The job will be started using 384 MPI processes, with 24 MPI processes placed on each compute node (this would use all the physical cores on each node). This would require 16 nodes to be requested in the PBS options.

The most important `mpirun` flags are:

-n [total number of MPI processes] Specifies the total number of distributed memory parallel processes (not including shared-memory threads). For pure MPI jobs that use all physical cores this will usually be a multiple of 24. The default on Tesseract is 1.

-ppn [parallel processes per node] Specifies the number of distributed memory parallel processes per node. There is a choice of 1-24 for physical cores on Tesseract compute nodes (1-48 if you are using Hyper-Threading) For pure MPI jobs, the most economic choice is usually to run with "fully-packed" nodes on all physical cores if possible, i.e. `-ppn 24` . Running "unpacked" or "underpopulated" (i.e. not using all the physical cores on a node) is useful if you need large amounts of memory per parallel process or you are using more than one shared-memory thread per parallel process.

Documentation on using Intel MPI (including `mpirun`) can be found online at:

- Intel MPI Documentation

Intel MPI: running hybrid MPI/OpenMP applications

If you are running hybrid MPI/OpenMP code using Intel MPI you need to set the `I_MPI_PIN_DOMAIN` environment variable to `omp` so that MPI tasks are pinned with enough space for OpenMP threads.

For example, in your job submission script you would use:

```
export I_MPI_PIN_DOMAIN=omp
```

You can then also use the `KMP_AFFINITY` enviroment variable to control placement of OpenMP threads. For more information, see:

- Intel OpenMP Thread Affinity Control

Intel MPI: Process Placement

By default, MPI processes are placed on nodes in a round-robin format. For example, if you are using 4 nodes, 16 MPI processes in total and have 4 MPI processes per node, you would use the command:

```
mpirun -n 16 -ppn 4 /path/to/my/exe
```

the processes would be placed in the following way:

```
MPI process 0: placed on Node 1
MPI process 1: placed on Node 2
MPI process 2: placed on Node 3
MPI process 3: placed on Node 4
MPI process 4: placed on Node 1
MPI process 5: placed on Node 2
MPI process 6: placed on Node 3
MPI process 7: placed on Node 4
MPI process 8: placed on Node 1
...
MPI process 15: placed on Node 4
```

The alternative way to place MPI processes would be to fill one node with processes before moving onto the next node (this is often known as *SMP placement*). This can be achieved within a PBS job on Tesseract by using the `-f` flag to pass the node list file explicity. For example:

```
mpirun -n 16 -ppn 4 -f $PBS_NODEFILE /path/to/my/exe
```

The processes would be placed in the following way:

```
MPI process 0: placed on Node 1
MPI process 1: placed on Node 1
MPI process 2: placed on Node 1
MPI process 3: placed on Node 1
MPI process 4: placed on Node 2
MPI process 5: placed on Node 2
MPI process 6: placed on Node 2
MPI process 7: placed on Node 2
MPI process 8: placed on Node 3
...
MPI process 15: placed on Node 4
```

Intel MPI: MPI-IO setup

If you wish to use MPI-IO with Intel MPI you must set a couple of additional environment variables in your job submission script to tell the MPI library to use the Lustre file system interface. Specifically, you should add the lines:

```
export I_MPI_EXTRA_FILESYSTEM=on
export I_MPI_EXTRA_FILESYSTEM_LIST=lustre
```

after you have loaded the `intel-tools-18` module.

If you fail to set these environment variables you may see errors such as:

```
This requires fcntl(2) to be implemented. As of 8/25/2011 it is not. Generic MPICH
Message: File locking failed in
ADIOI_Set_lock(fd 0,cmd F_SETLKW/7,type F_WRLCK/1,whence 0) with return value
FFFFFFFF and errno 26.
- If the file system is NFS, you need to use NFS version 3, ensure that the lockd
 daemon is running on all the machines, and mount the directory with the 'noac'
 option (no attribute caching).
- If the file system is LUSTRE, ensure that the directory is mounted with the 'flock'
 option.
ADIOI_Set_lock:: Function not implemented
ADIOI_Set_lock:offset 0, length 10
application called MPI_Abort(MPI_COMM_WORLD, 1) - process 3
```

6.8 Example parallel MPI job submission scripts

Example job submssion scripts are included in full below. They are also available via the following links:

- Intel MPI Job: `example_mpi_impi.bash`
- Intel MPI Hybrid MPI/OpenMP Job: `example_hybrid_impi.bash`
- Intel MPI Array MPI Job: `example_array_impi.bash`

6.8.1 Example: Intel MPI job submission script for MPI parallel job

A simple MPI job submission script to submit a job using 4 compute nodes (maximum of 144 physical cores) for 20 minutes would look like:

```
#!/bin/bash --login

# PBS job options (name, compute nodes, job time)
#PBS -N Example_MPI_Job
# Select 16 full nodes
#PBS -l select=16
#PBS -l walltime=00:20:00
#PBS -l place=scatter

# Replace [budget code] below with your project code (e.g. t01)
#PBS -A [budget code]

# Change to the directory that the job was submitted from
cd $PBS_O_WORKDIR

# Load any required modules
module load intel-tools-18

# Set the number of threads to 1
#   This prevents any threaded system libraries from automatically
#   using threading.
export OMP_NUM_THREADS=1

# Launch the parallel job
#   Using 384 MPI processes and 24 MPI processes per node
mpirun -n 384 -ppn 24 ./my_mpi_executable.x arg1 arg2 > my_stdout.txt 2> my_stderr.txt
```

This will run your executable "my_mpi_executable.x" in parallel on 384 MPI processes using 16 nodes (24 cores per node, i.e. not using hyper-threading). PBS will allocate 16 nodes to your job and mpirun will place 24 MPI processes on each node (one per physical core).

See above for a more detailed discussion of the different PBS options

6.8.2 Example: Intel MPI job submission script for MPI+OpenMP (mixed mode) parallel job

Mixed mode codes that use both MPI (or another distributed memory parallel model) and OpenMP should take care to ensure that the shared memory portion of the process/thread placement does not span more than one node. This means that the number of shared memory threads should be a factor of 12.

In the example below, we are using 16 nodes for 6 hours. There are 32 MPI processes in total and 12 OpenMP threads per MPI process. Note the use of the `I_MPI_PIN_DOMAIN` environment variable to specify that MPI process placement should leave space for threads.

```
#!/bin/bash --login

# PBS job options (name, compute nodes, job time)
#PBS -N Example_MixedMode_Job
#PBS -l select=16
#PBS -l walltime=6:0:0
#PBS -l place=scatter

# Replace [budget code] below with your project code (e.g. t01)
#PBS -A [budget code]

# Change to the directory that the job was submitted from
```

(continues on next page)

(continued from previous page)

```
cd $PBS_O_WORKDIR

# Load any required modules
module load intel-tools-18

# Set the number of threads to 12
#   There are 12 OpenMP threads per MPI process
export OMP_NUM_THREADS=12

# Set placement to support hybrid jobs
export I_MPI_PIN_DOMAIN=omp

# Launch the parallel job
#   Using 32 MPI processes
#   2 MPI processes per node
#   12 OpenMP threads per MPI process
mpirun -n 32 -ppn 2 ./my_mixed_executable.x arg1 arg2 > my_stdout.txt 2> my_stderr.txt
```

6.9 Interactive Jobs

When you are developing or debugging code you often want to run many short jobs with a small amount of editing the code between runs. This can be achieved by using the login nodes to run MPI but you may want to test on the compute nodes (e.g. you may want to test running on multiple nodes across the high performance interconnect). One of the best ways to achieve this on Tesseract is to use interactive jobs.

An interactive job allows you to issue `mpirun` commands directly from the command line without using a job submission script, and to see the output from your program directly in the terminal.

To submit a request for an interactive job reserving 16 nodes (384 physical cores) for 20 minutes you would issue the following qsub command from the command line:

```
qsub -IVl select=16:ncpus=24,walltime=0:20:0 -A [project code]
```

..note :: Unlike non-interactive jobs, you must specify the number of cores you want to use per node by adding *:ncpus=24* to the select statement.

When you submit this job your terminal will display something like:

```
qsub: waiting for job 436.tesseract-services1 to start
```

It may take some time for your interactive job to start. Once it runs you will enter a standard interactive terminal session. Whilst the interactive session lasts you will be able to run parallel jobs on the compute nodes by issuing the `mpirun` command directly at your command prompt (rememberyou will need to load the `intel-tools-18` module before running) using the same syntax as you would inside a job script. The maximum number of cores you can use is limited by the value of select you specify when you submit a request for the interactive job.

If you know you will be doing a lot of intensive debugging you may find it useful to request an interactive session lasting the expected length of your working session, say a full day.

Your session will end when you hit the requested walltime. If you wish to finish before this you should use the `exit` command.

CHAPTER 7

Using the Tesseract GPU Nodes

Tesseract has eight compute nodes equipped with GPGPU accelerators. This section of the user guide explains how to compile code and submit jobs to the GPU nodes.

Note: The GPU accelerators on Tesseract are only available in TCC (Tesla Compute Cluster) mode and so do not support graphics rendering tasks, only computational tasks.

7.1 Compiling software for the GPU nodes

7.1.1 CUDA

CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs).

To use the CUDA toolkit on Tesseract, you should load the *cuda* module:

```
module load cuda
```

Once you have loaded the `cuda` module, you can access the CUDA compiler with the `nvcc` command.

As well as the CUDA compiler, you will also need a compiler module to support compilation of the host (CPU) code. The CUDA toolkit supports GCC. You should load your chosen compiler module before you compile.

Note: The `nvcc` compiler currently supports versions of GCC up to 7.x and does not support Intel compilers.

Using CUDA with GCC

By default, `nvcc` will use the system version of GCC. We recommend that you load a more recent version of GCC than the system default to support the CUDA compiler, e.g.

```
module load gcc
```

You can now use `nvcc` to compile your source code, e.g.:

```
nvcc -o cuda_test.x cuda_test.cu
```

7.2 Submitting jobs to the GPU nodes

One additional option is needed in GPU job submission scripts over those in standard jobs:

- `-q QGPU` This option is required to submit the job to the `gpu` queue on Tesseract

---

Note: GPU compute nodes are only available in exclusive mode. This means your minimum

---

request ia 1 GPU compute node with 24 CPU cores and 4 V100 GPU accelerators.

7.2.1 Job submission script using multiple GPUs on a single node

---

Note: Remember that there are a maximum of 4 GPU accelerators per node and a maximum of 24 CPU cores per node.

A job script that requires 1 GPU compute node (4 GPU accelerators and 24 CPU cores) for 20 minutes could look like:

```
#!/bin/bash
#
#PBS -N cuda_test
#PBS -q QGPU
#PBS -l select=1
#PBS -l walltime=0:20:0
# Budget: change 't01' to your budget code
#PBS -A t01

# Load the required modules
module load cuda
module load gcc

cd $PBS_O_WORKDIR

./cuda_test.x
```

The line `#PBS -l select=1` requests 1 node and the line `#PBS -q QGPU` requests GPU compute nodes.

CHAPTER 8

References and further reading

8.1 Online Documentation and Resources

- GNU compiler online documentation: http://gcc.gnu.org/onlinedocs/
- MPI Home pages: http://www-unix.mcs.anl.gov/mpi/
- Free MPI implementation useful for testing: http://www.open-mpi.org/software/ompi/v1.2/
- Various HPC Workshops by NCCS: http://www.nccs.gov/user-support/training-education/workshop-archives/
- An HPC tutorial: http://www.llnl.gov/computing/hpc/training/
- An MPI tutorial: http://www-unix.mcs.anl.gov/mpi/tutorial/gropp/talk.html
- HPC tutorials by NCSA http://www.citutor.org/login.html

8.2 MPI programming

- MPI: The Complete Reference. Snir, Otto, Huss-Lederman, Walker and Dongarra. MIT Press. ISBN 0 262 69184 1
- MPI: The Complete Reference, volume 2. Gropp et al. MIT Press. ISBN 0262571234
- Using MPI. Gropp, Lusk, Skjellum. MIT Press. ISBN 0 262 57104 8

8.3 OpenMP programming

- Parallel Programming in OpenMP. Chandra, Kohr, Menon, Dagum, Maydan, McDonald. Morgan Kaufmann. ISBN: 1558606718

8.4 Parallel programming

- Practical Parallel Programming. Gregory V. Wilson. MIT Press. ISBN 0 262 23186 7
- Designing and Building Parallel Programs. Ian Foster. Addison-Wesley. ISBN 0 201 57594 9 http://www.mcs.anl.gov/dbpp/
- Parallel Computing Works! Roy D. Williams, Paul C. Messina (Editor), Geoffrey Fox (Editor), Mark Fox Morgan Kaufmann Publishers; ISBN: 1558602534
- Parallel programming with MPI. Peter S. Pancheco. The complete set of C and Fortran example programs for this book are available at: http://www.cs.usfca.edu/mpi

8.5 Programming languages

- Fortran90/95 Explained. Metcalf and Reid. Oxford Science Publications. ISBN 0 19 851888 9
- Fortran 90 Programming. Ellis, Philips, Lahey. Addison-Wesley. ISBN 0-201-54446-6
- Programmers Guide to Fortran90. Brainerd, Goldberg, Adams. Unicomp. ISBN 0-07-000248-7
- The High Performance Fortran Handbook. Koelbel, Loveman, Schreiber, Steele, Zosel. ISBN 0-262-11185-3 / 0-262-61094-9
- Parallel Programming using C++. G.V.Wilson and P Lu. MIT Press. ISBN 0 262 73118 5

8.6 Programming skills

- Debugging and Performance Tuning for Parallel Computing Systems, Simmons et al.
- Foundations of Parallel Programming, A Machine-independent Approach, Lewis.

CHAPTER 9

Intel MKL: BLAS, LAPACK, ScaLAPACK

The Intel MKL libraries contain a variety of optimised numerical libraries including BLAS, LAPACK, and ScaLAPACK.

9.1 Intel Compilers

9.1.1 BLAS and LAPACK

To use MKL libraries with the Intel compilers you first need to load the Intel tools module:

```
module load intel-tools-18
```

To include MKL you specify the `-mkl` option on your compile and link lines. For example, to compile a single source file, Fortran program with MKL you could use:

```
ifort -c -mkl -o lapack_prb.o lapack_prb.f90
ifort -mkl -o lapack_prb.x lapack_prb.o
```

The `-mkl` flag without any options builds against the threaded version of MKL. If you wish to build against the serial version of MKL, you would use `-mkl=sequential`.

9.1.2 ScaLAPACK

The distributed memory linear algebra routines in ScaLAPACK require MPI in addition to the compilers and MKL libraries.

```
module load intel-tools-18
```

Once you have the modules loaded you need to specify `-mkl=cluster` to build against the MPI parallel version of MKL (including BLACS and ScaLAPACK). Remember to use the MPI versions of the compilers:

```
mpiifort -c -mkl=cluster -o linsolve.o linsolve.f90
mpiifort -mkl=cluster -o linsolve.x linsolve.o
```

9.2 GNU Compiler

9.2.1 BLAS and LAPACK

To use MKL libraries with the GNU compiler you first need to load the Intel tools module:

```
module load intel-tools-18
```

To include MKL you need to explicitly link against the MKL libraries. For example, to compile a single source file, Fortran program with MKL you could use:

```
gfortran -c -o lapack_prb.o lapack_prb.f90
gfortran -o lapack_prb.x lapack_prb.o -L$MKLROOT/lib/intel64 -lmkl_gf_lp64 -lmkl_core␣
↪-lmkl_sequential
```

This will build against the serial version of MKL, to build against the threaded version use:

```
gfortran -c -o lapack_prb.o lapack_prb.f90
gfortran -fopenmp -o lapack_prb.x lapack_prb.o -L$MKLROOT/lib/intel64 -lmkl_gf_lp64 -
↪lmkl_core -lmkl_gnu_thread
```

9.2.2 ScaLAPACK

The distributed memory linear algebra routines in ScaLAPACK require MPI in addition to the compilers and MKL libraries.

```
module load intel-tools-18
```

Once you have the modules loaded you need to link against two additional libraries to include ScaLAPACK. Remember to use the MPI versions of the compilers for GCC:

```
mpif90 -c -o linsolve.o linsolve.f90
mpif90 -o linsolve.x linsolve.o -L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64 -lmkl_
↪intel_lp64 -lmkl_sequential -lmkl_core -lmkl_blacs_intelmpi_lp64 -lpthread -lm -ldl
```

9.2.3 ILP vs LP libraries

If you look in the *$MKLROOT/lib/intel64* directory then you will see ILP and LP libraries, in the above we were linking against the LP libraries and you can choose either. ILP use a 64 bit integer type, whereas LP use a 32 bit integer type. For very large arrays then ILP is the best choice (as it can index far more data), but there are some limitations. For more information see the Intel documentation here.

CHAPTER 10

Terms and Conditions of Access

These Terms and Conditions of Access are presented to each user when they register. We anticipate that they will change rarely, if at all. Note that the Terms and Conditions include an undertaking to observe the Code of Conduct. It also includes an agreement to the Service processing the user's personal data; in return, the Service undertakes to follow the *Personal Data and Privacy Policy*.

Access to Tesseract is also subject to abiding by the University of Edinburgh Computing Regulations and Security Policy:

- University of Edinburgh Computing Regulations
- University of Edinburgh Security Policy

10.1 You agree:

- only to use the service for the purposes for which you were given access, for example as specified in your research project's grant award
- not to disrupt the working of the service, for example by knowingly introducing malicious software into it, nor to try to breach its security or use resources which aren't assigned to you
- not to interfere with other users' work, corrupt their data or invade their privacy
- not to infringe copyright or other intellectual property rights
- not to take data from any database or dataset without the explicit or implied permission of its owner
- not to use another person's account, nor to let other people use your accounts, except by agreement with us;
- to keep your SAFE and machine passwords confidential, and to inform us if someone else learns any of them or if you become aware that the security of our systems is compromised in any way
- not to misuse the Internet, for example by sending spam or malicious software, by pretending to be someone else or by doing anything that might hinder or prevent someone else from using the Internet legitimately

- not to use the service for illegal or immoral purposes, such as theft, fraud, drug-trafficking, money-laundering, terrorism, pornography, violence, cruelty, incitement to racial hatred, prostitution, paedophilia, or for offensive, obscene, abusive, menacing, defamatory or insulting behaviour
- to observe the JANET Acceptable Use Policy
- to comply with any special conditions that may apply to particular software packages

10.2 We agree:

- as far as we reasonably can, to provide a 24-hour service as described at <http://tesseract-dirac.readthedocs.io>, it being understood that there will be times when the service is unavailable, for example as a result of unexpected failures, maintenance work or upgrades
- to take reasonable steps to protect your data from being lost or corrupted.
- to protect the security and privacy of the data we hold about you, as described in our *Personal Data and Privacy Policy*
- that we will acquire no intellectual property rights over your software and data
- to respond promptly to any complaints or suggestions you make about the service

10.3 You accept:

- that the service is provided "as is" and we can't guarantee 100% perfection. In legal terms, this means that we are excluding all warranties and conditions applying to the service, including those implied by law. What is this?. We are not liable if things go wrong and you suffer damage as a result, although if our negligence results in anyone's death or personal injury we do not limit or exclude our liability for that.
- that you are responsible for your use of any advice or information we may give you. We will take reasonable steps to ensure that it's true and useful, but we can't guarantee this. In legal terms, this means that we expressly disclaim any and all liability for all representations, statements, conditions or warranties to that or any other effect except to the extent that such liability may not be lawfully excluded.
- that we may make changes in the service
- that we will use the personal details which you supply to us, together with records of your use of the service, as described in our *Personal Data and Privacy Policy*
- that we may suspend your access to the service and discuss this with your PI if it seems to us that you are breaking these Terms and Conditions; that if it is necessary to protect the service or other users' work or data, we can halt the execution of any program which you start; and that we have the right to close your accounts
- that you alone are responsible for what you do when using the service. If you break the law you alone must answer for it, and if you cause damage to anyone else, you alone are liable, not us
- that you will acquire no intellectual property rights over the software or any information we provide
- that the use of the service by nationals of certain countries is controlled by special regulations laid down by the UK Government in connection with the Wassenaar Arrangement
- that we may make reasonable changes to these Terms and Conditions at any time, and, once we have posted those changes on our website, the new version will then apply to you

These Terms and Conditions are governed by the laws of Scotland and the Scottish courts.

If you have any questions about these Terms and Conditions, please mail the DiRAC Helpdesk.

CHAPTER 11

Personal Data and Privacy Policy

Information about you: how we use it and with whom we share it

This policy applies to personal data about users of the Tesseract service. It relates to data gathered on the service itself. The details that you provide on registration are stored in the SAFE and will be processed according to the SAFE data privacy policy.

We may store details of any aspects of service use, including, for example, the amount of processor time and storage space used by user accounts. This may include some personally identifiable data such as the network addresses you use to connect to the system. We will use this information to help us manage and administer the service, to review, analyse and improve its performance, security, its patterns of use, and to plan for the future.

This information will be available to appropriate members of the staff who are working on the Tesseract service. Resource usage information will be uploaded to the SAFE where it will be processed according to the SAFE data privacy policy.

Periodically, the service may be examined by auditors acting on behalf of the system owners, The University of Edinburgh. These may see your personal data. They will be acting under conditions of professional confidentiality.

We reserve the right to monitor use of the service by your accounts, including anything transmitted over the Internet, and any data or software stored on our systems, in order to ensure that you and all the other users are complying with the Terms and Conditions of Access and not breaking the law. We must allow any court or other competent authority to inspect our records of your use of the system, or your data, and to take copies of it, if this is legally required; and we must report your activities to the competent authorities if we know or suspect that you are breaking the law. These are legal obligations for us.

We would not be able to fulfil our contract to administer the service properly, nor adequately account to our funding bodies for our conduct of this project, without processing your personal data in this way.

We may retain the data that we gather on the system for the duration of the Tesseract service. Data that could be used to identify you directly will not be retained longer than two years after the end of the service.

If you have any questions about the treatment of your personal data, please contact The DiRAC Helpdesk.