

Getting Started with NSCC Supercomputing on ASPIRE 1

Getting Started with NSCC Supercomputing on ASPIRE 1

Updated: 16 December 2020



Table of Contents

1	Inti	oduct	ion	3
2	Acc	ess to	ASPIRE 1	4
	2.1	Enro	ll as an NSCC User	4
		2.1.1	Go to NSCC User Login Page	4
		2.1.2	Login with Your Institution's Account	5
		2.1.3	Set Your NSCC Account Password	7
	2.2	Conr	ect to NSCC Server using VPN	9
		2.2.1	Setup and Use NSCC VPN	9
	2.3	Setup	and Use SSH Tools	11
		2.3.1	Connect to NSCC ASPIRE 1	15
3	Rui	ı Inter	active MPI Parallel Programs	16
	3.1	Mod	ule Package	16
	3.2	Intro	duction to MPI	17
	3.3	Exer	cise 1: First MPI Program	
		3.3.1	Writing your First MPI Program – mpi_hello_world.c	
		3.3.2	Compile and Run your First MPI Program	20
		3.3.3	Understanding the Output	20
		3.3.4	Takeaways	21
4	Rui	ı Inter	active Batch MPI Jobs	22
	4.1	NSC	C Job Scheduler - PBS Pro	22
		4.1.1	PBS Queue	22
		4.1.2	Submitting Jobs to PBS Queue	23
	4.2	Tran	sferring files between PBS Compute Manager and Local Machine	23
	4.3	Exer	cise 2: Interactive Batch MPI Program	25
		4.3.1	Write your Interactive Batch MPI Program – mpi_prime.c	25
		4.3.2	Run your MPI Prime Program	29
		4.3.3	Understanding the Output	30
		4.3.4	Takeaways	30
5	Rui	1 Auto	mated Batch MPI Jobs	31
	5.1	Input	s and Outputs	
	5.2	Runn	ning jobs in your directory	
	5.3	Exer	cise 3 (Part 1): Shell Script for Automated Batch Job	31
		5.3.1	Write a Batch Script – batch_hello_world.pbs	32
		5.3.2	Understanding your Batch Script – batch_hello_world.pbs	32
		5.3.3	Submit and monitor your Batch Job	33
		5.3.4	Check your Output	33
	5.4	Exer	cise 3 (Part 2): Shell Script for Automated Batch Job	33
		5.4.1	Abnormal Job Termination	33
		5.4.2	Write a Batch Script – batch_prime.pbs	34
		5.4.3	Understanding your Batch Script – batch_prime.pbs	35
		5.4.4	Check your Output	
		5.4.5	Takeaways	
6	Sun	nmary	r ••••••••••••••••••••••••••••••••••••	37
7	Ref	erence	25	38



1 Introduction

The National Supercomputing Centre (NSCC) Singapore was established in 2015 and manages Singapore's first national petascale facility with high-performance computing (HPC) resources to support the science and engineering computing needs for academic, research and industry communities.

ASPIRE 1, an NSCC HPC system, provides total compute capacity of 1 petaflop [1]. Each Fujitsu PRIMERGY compute server node consisted of dual Intel E5-2690v3 (2.60GHz, 12 cores) processors with 128GB of memory. In total, ASPIRE 1 consists of 31,392 cores and 229 terabytes of memory. Some compute nodes have attached NVIDIA Tesla K40 GPU accelerator. ASPIRE 1 uses the latest EDR interconnect technology for high-throughput and low-latency inter-node communication. The storage of ASPIRE 1 consists of 3 tiers with I/O bandwidth up to 500 GB/s. The total storage of ASPIRE 1 is 14 PByte. The Operating System (OS) on ASPIRE 1 is Linux [2].



Figure 1: Compute Node Architecture [3]

This starter guide provides a quick introduction for users to run their first program on ASPIRE 1. This guide is organized into five sections. Section 1 introduces NSCC supercomputing resources and the objective of this lab manual. Section 2 explains the steps on enrolling as an NSCC user and setting up the login environment. Section 3 discusses how to run interactive MPI parallel programs. Setting up and running of batch MPI programs interactively are shown in section 4 and finally, section 5 introduces shell scripting to automate the running of multiple batch jobs.



2 Access to ASPIRE 1

The objective of this section is to learn how to access the NSCC cluster from your computer [4]. First, we will show the steps to register as an NSCC user and get the credentials for accessing the NSCC ASPIRE 1 cluster. Second, we will show you how to connect to NSCC VPN (only for non-institution users) and access NSCC cluster for Linux/MacOS/Windows user via SSH [5]. VPN stands for Virtual Private Network, which provides access to a private network over the public network. SSH, known as Secure Shell, provides a secure tunnel over an unsecured network.

2.1 Enroll as an NSCC User

2.1.1 Go to NSCC User Login Page

In this step, we will guide you on how to register as an NSCC user [6].

To enroll as an NSCC user, log in using your institution's account at <u>https://user.nscc.sg/saml/</u> and click 'Login'. If you are not from the institutions listed in the page, please email <u>contact@nscc.sg</u> to obtain your account [7].

user.nscc.sg/saml/		1 ☆	3	00	•
	Welcome to NSCC!		- Course	T Provide Street	
	 The Login here is for NUS, NTU, A*STAR, SUTD and SMU users only. You will select your organisation and be directed to your organisatic authentication server to sign in with your own user ID and password First-time login will enrol you to NSCC system, and an NSCC account will be created for you automatically. You may login subsequently to reset your NSCC account password of SSH Key. By clicking Login here, you agree to abide by <u>NSCC Acceptable Use</u> 	on's 1. t or			
	Policy (AUP). 6. If you are not from any of the organisations mentioned above, pleas contact us to register your interest to use NSCC services. Login Note: • Please nable cookies in your browser for the login to work • You may try to clear browser cache if there's something wrong	ie H			
1.	Please don't use "Back" button from the browser. Copyright © 2018 NSCC NSCC NSC NSCC NSCC NSC NS	uting		RRI RRI ARRI ARRI	-
			125 C		

Figure 2.1: NSCC User Login



2.1.2 Login with Your Institution's Account

Choose your institution and log in using your institution's account.

Select your identity provider
Afrikaans Català Čeština Dansk Deutsch ελληνικά English Español eesti keel Euskara Suomeksi Français אריבע Hrvatski Magyar Bahasa Indonesia Italiano 日本語 Letzebuergesch Lietuvių kalba Latviešu Nederlands Nynorsk Bokmål Język polski Portugués Portugués brasileiro Românește pyccxий язык Sámegiella Slovenščina Srpski Svenska Türkçe isiXhosa 简体中文 繁體中文 IsiZulu
You have previously chosen to authenticate at SUTD - Singapore University of Technology and
Design Login at SUTD - Singapore University of Technology and Design
Singapore Institutions
Incremental search
SUTD - Singapore University of Technology and Design
A*STAR - Agency for Science, Technology and Research
NTU - Nanyang Technological University
NUS - National University of Singapore
SMU - Singapore Management University
SP - Singapore Polytechnic
TP - Temasek Polytechnic
Copyright © 2007-2019 UNINETT AS

Figure 2.1.1: NSCC User Login (Choose Your Institution)



© 2013 Microsoft Change Password Forgot Password? Help On 2FA

Figure 2.1.2: NSCC User Login (NUS)



Getting Started with NSCC Supercomputing on ASPIRE 1

NTU:



Figure 2.1.3: NSCC User Login (NTU)

SUTD:

\leftrightarrow \rightarrow C \triangleq saturn.sutd.edu.sg/adfs/ls/?SAMLRef	equest=pZJBT%2BMwEIX%2FSuR74jiBlqy2UqFCVGL Suest
	SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
	Sign in with your organizational account
	someone@example.com
	Password
	Sign in
	For students, please sign in with "SUTDSTU" followed by your network ID. Example SUTDSTU\1000001 For staff, please sign in with "SUTDSTF" followed by your network ID. Example SUTDSTF\John_tan
	© 2013 Microsoft

Figure 2.1.4: NSCC User Login (SUTD)



2.1.3 Set Your NSCC Account Password

After logging in to your institution, you will be redirected to the NSCC page with your NSCC account. On the first login, you will have to set a new password for your account.

Click 'Set/Reset Password' to set the new password for your NSCC account.

iscc.sg/sami/index.pnp	N K PO
- Charles	and strategy.
Welcome to NSCC User Portal!	Log out Back
Hello	
Your NSCC account is	
Set/Reset SSH Key	
Set/Reset Password	A DESCRIPTION OF A DESC
	A REAL PROPERTY AND A REAL PROPERTY OF A DESCRIPTION OF A
https://help.nscc.sg/ - User guides and FA	Qs
https://servicedesk.nscc.sg/ - Service Desl	k to log support tickets
https://projects.nscc.sg/ - Submit projects	s/resource requirements
netp@nscc.sg - Email to seek technical ass	sistance
	G
Copyright © 2018 NSCC	National Supercomputing
	NSCC Centre
and it is a second s	A STATEMENT REPORT OF THE OWNER
and a second sec	CALL OF CALL O

Figure 2.2.1: NSCC Account Password Reset

Ensure that your password is strong enough and meet the following requirements:

- Minimum 8 characters
- Mixture of upper and lower-case characters
- Must contain numbers (0-9)
- Must include at least one special char: !@#\$%^&*+=?><

Your NSCC account and password also works as your credential for accessing ASPIRE 1.



Once ready, you may click the Set Password button.



Figure 2.2.2: NSCC Account Password Reset



2.2 Connect to NSCC Server using VPN

2.2.1 Setup and Use NSCC VPN

In this step, we will show you how to connect to the NSCC cluster.

NSCC server is only accessible from a secured network which includes institutions' network or via NSCC VPN. To use NSCC server, please connect to the VPN provided by your institution or directly connect from the network within your institution.

If you are not from any institutions listed in the login page, you should use NSCC VPN [6].

i. To use NSCC VPN, go to <u>https://vpn.nscc.sg/</u>. Enter your username and password to log in.

Login to	UserPortal
Username: Password:	
Remember my	login (uses cookie)

Figure 2.3.1: NSCC VPN

ii. Upon logging in, you will see the following page:

Configuration for your OTP Software Token

To enter the User Portal you have to authenticate with an One Time Password. Scan the QR code below with Google Authenticator on your phone. The app will then generate a new passcode every 30 seconds. From then on, your password, directly followed by the passcode displayed, is the one-time password you have to enter to login. The *Details* link helps you to install Google Authenticator and shows token information in plain text.





Figure 2.3.2: OTP Verification



- iii. Download and install "Sophos Authenticator" on your mobile phone from Apple Store (iOS) or Google Play (Android). Then scan the QR code from the page above.
- iv. Once you have logged in successfully, go to Remote Access and download the corresponding version according to your Operating System.

SSL VPN	
Click here to download a complete installation package including client software, keys and automatic configuration for Windows XP / Vista / 7 / 8.	Download
Click here to download an installation file which updates all keys and configuration on your system, without re-installing the client software (Windows XP / Vista / 7 / 8).	Download
Click here to download a ZIP archive which contains all necessary files to set up SSL VPN on Linux, MacOS X, BSD or Solaris.	Download
Click here to install the SSL VPN configuration on your Android™ or iOS™ device. The client software is available for download on <u>Google Play</u> or the <u>App Store</u> .	Install

Figure 2.3.3: SSL VPN Download

v. After downloading, install the Sophos SSL VPN. Once the installation is completed, start the client and log in using your username and password.

Username:	1
Password:	
OK	Cancel

Figure 2.3.4: Sophos SSL VPN

vi. You will be able to access the NSCC cluster via NSCC VPN now.



2.3 Setup and Use SSH Tools

In this step, we show you how to use SSH tools to connect to NSCC cluster.

SSH, known as Secure Shell, is a network protocol that gives users a secure way to access a computer over an unsecured network [5]. SSH is widely used by HPC users for managing systems and applications remotely, enabling them to log in to another computer over a network, execute commands and move files from one computer to another.

For users who prefer using the terminal: use terminal (Linux/MacOS) or PowerShell (Windows), log in via SSH on port 22. Your login server is dependent on your institution. Your account credential is your NSCC ID with NSCC password.

If you are logging in from	Your Hostname is	Port
NUS	nus.nscc.sg	22
NTU	ntu.nscc.sg	22
A*STAR	astar.nscc.sg	22
SUTD	sutd.nscc.sg	22
Everywhere else (via NSCC VPN)	aspire.nscc.sg	22

Figure 2.3.4: SSH Login Hosts

The login command for SSH is: ssh YOUR_NSCC_ID@NSCC_HOSTNAME

Here, YOUR_NSCC_ID is your NSCC account ID. Refer to the table above for your NSCC_HOSTNAME based on the institution you are from.

For users who prefer SSH client:

- i. Download and install an SSH client, eg. Putty, XShell. Here we take XShell as an example.
- ii. XShell can be downloaded from https://www.netsarang.com/en/all-downloads/



Figure 2.4.1: XShell Download



Getting Started with NSCC Supercomputing on ASPIRE 1

iii. For research purpose, please choose free licensing.

CHOOSE YOUR LICENSE TYPE

By downloading our evaluation software, you agree to receive follow-up emails regarding your evaluation as well as occasional patch notes and notices. You can unsubscribe from these emails at any time by clicking the "Unsubscribe Me" button at the bottom of any email. To opt-out of promotional emails such as occasional promotional discounts or special events, please uncheck the box below before submitting your information for a 30 day evaluation. We DO NOT sell your information to third-parties.							
Existing User Product Key (required)	(a) 30 Day Evaluation Your Name (required)	Free License for Home and School Users Free Licensing Page					
Choose Your Version (required)	Your Email (required) A download link will be emailed to you Company	Related Downloads • EULA • User Manual • Datasheet					
	□ I agree to receive emails related to occasional promotional discounts or special events.						

Figure 2.4.2: XShell Download

iv. Enter your name and e-mail. Check your e-mail for downloading and installing. Here we download both XShell and XFTP. XShell is used as an SSH tool to connect to ASPIRE 1 server. XFTP helps us to upload or download files from the remote server.

	Terms of Free Use
Free for Non-Commercial Use Only Your Name (required)	NetSarang Computer, Inc. prides itself in providing our powerful SSH and SFTP/FTP client for free for the past 10 years. Our free licenses are not just fr in price, they are free of advertisements, spyware, or any other method of exploitation of our userbase. We believe users from all backgrounds and circumstances should have access to a powerful and feature-rich SSH and SFTP/FTP client whether it be to learn, teach, or to even just supplement a hobby.
Your Email (required)	NetSarang Computer, Inc.'s free licensing of Xshell and Xftp is for non- commercial use only. Any use of our free licensing for business purposes is
Both 🗆 Xshell Only 📄 Xftp Only	strictly against the terms laid out in our Free End User License Agreement. If would like to use Xshell or Xftp for commercial use, we encourage you to purchase a license and help us further develop our software.
DOWNLOAD	By downloading our free licensing software, you agree to receive emails rela to occasional promotional discounts or special events as well as occasional
NOTE: A valid email address is required. A download link will be sent to your email.	patch notes and notices. You can unsubscribe from these emails at any time clicking the "Unsubscribe Me" button at the bottom of any email. We DO NOT sell your information to third-parties.

Figure 2.4.3: XShell Download



Getting Started with NSCC Supercomputing on ASPIRE 1

v. After installing, open XShell and set up your login server properties. Click 'New' to create a new profile for ASPIRE 1.

🔮 Sessions 🛛 🕹 🗙						
📮 New 🔻 🕄 🛛 🔀 💼	📑 💀 Prop	erties		-		Q
All Sessions						0
Name 🔺	Host	Port	Protocol	User N	Description	
🕑 aliyun	tmhtbd.com	22	SSH	root		·
🕑 alligator	alligator-sa	22	SSH	lab		
🕑 Aragorn	aragorn.d2	22	SSH	celia		
√ S AM2	54.255.192	22	SSH	ubuntu		
Second Se	3.16.75.202	22	SSH			
SUSTC CS	cstu.ustc.e	22	SSH	colt		
Show this dialog box at startup						

Figure 2.4.4: Creating a New Profile for XShell

vi. Proceed to connect to the NSCC cluster using the host you are associated with. Refer to Figure 2.3.4 for the SSH login host.

Properties of nscc			?	\times
Category:				
Connection	Connection			
Authentication	General			
Login Scripts	Name:	nscc	- - T	
	Protocol:	SSH V	-	
- Security	Host:	nus nscc. sq		
SFTP	Port Number			
··· TELNET	Port Number:	22		
RLOGIN	Description:	A 1		
Proxy				
Keep Alive				
E Terminal	Reconnect			
···· Keyboard ···· VT Modes	Reconnect a	utomatically if connection is terminated unexpec	tedly	
Advanced	Interval:	0 A sec Limit: 0	in min	
Appearance	incervan		Y	
Window Hiahliaht	TCP Options			
- Advanced	Use Nade's a	laorithm		
Trace				
Logging				
File Transfer				
X/YMODEM				
ZMODEM				
		Connect OK	Cance	1

Figure 2.4.5: Connecting to NSCC Cluster on XShell



vii. Set up your login authentication. Your account credential is your NSCC ID and password.

Properties of nscc			?	×
<u>C</u> ategory:				
Connection	Connection > Auth	nentication		
Authentication Login Prompts Unclosed Scripts SSH SSH Tuppeling	Select an authenticat Use this section to sa we recommend you le	ion method and other related paramete ve time when logging in. However, for r ave this section blank if security is a co	rs. naximum securi ncern.	ity,
SFTP	Method:	Password 🗸	Setup	
TELNET RLOGIN	<u>U</u> ser Name:]	
SERIAL	Password:	•••••		
Proxy Keep Alive	User <u>K</u> ey;	<none> ~</none>	Browse.	
Terminal Keyboard Keybboard Keyboard Keyboard Keyboard Keybboard Keyboard	P <u>a</u> ssphrase:			
		Connect OK	Cance	ł

Figure 2.4.6: Connecting to NSCC Cluster on XShell



2.3.1 Connect to NSCC ASPIRE 1

Connect to NSCC ASPIRE 1. You should be able to see the following welcome message (Figure 2.5).

```
*****************
# Welcome to ASPIRE1
                                                                #
# _____
                                                               #
                                                                #
# - To list the available environment modules, do "module avail"
                                                               #
# - To purge the loaded modules, do "module purge"
# - To see list of all jobs in the queues, run "gstat"
# - The "myquota" command is available to view your quota limits for \setminus #
   "home" and "scratch"
#
                                                                #
*****
Usage in 7 days to 2020/12/08:
Number of jobs on CPU nodes: 0
Core hours on CPU nodes: 0.0
Number of jobs on GPU nodes: 0
Core hours on GPU nodes:
                          0.0
Usage in month to 2020/11/30:
Number of jobs on CPU nodes: 6
Core hours on CPU nodes:
                          10.6
Number of jobs on GPU nodes: 1
Core hours on GPU nodes:
                          2.6
Usage since system start to 2020/11/30:
Number of jobs on CPU nodes: 100
Core hours on CPU nodes:
                          227.8
Number of jobs on GPU nodes: 4
Core hours on GPU nodes:
                          23.6
Project status as of 2020/12/10 00:14
ASPIRE1 Core hours remaining for project: 1001
Total Grant: 10000.0
Total Used: 3000.0
Total Pending: 0.0
Total Avail: 7000.0
AI GPU hours remaining for project: 1001
Total Grant: 1000.0
Total Used: 791.8
Total Pending: 0.0
Total Avail: 208.1
Purging policy is implemented on scratch directory
and files which have not been accessed within the last 30 days will be
purged automatically. For more info visit https://help.nscc.sg -> FAQs
```

hpcuser@nscc04 ~ \$

Figure 2.5: NSCC ASPIRE 1



3 Run Interactive MPI Parallel Programs

The objective of this section is to run your first program on ASPIRE 1. We will first understand the organization of software on the cluster. Next, we shall cover a brief MPI introduction to facilitate the understanding of the MPI programs that we will be writing subsequently. Lastly, we will walk you through in writing your first MPI program.

3.1 Module Package

The NSCC cluster is running with the Linux operating system [8]. Unlike computers used by one or a few users, HPC system is shared by many users. Different users use different software, but also share some software like compilers and computing engines. To save users the hassle of downloading software, some of the common software have been preinstalled. These software are managed through modules. [9]. Modules enable the user to pick and choose what software or version of the software they wish to use. When there are several versions available of the same software package, the user can select which version to use with the load command.

To display all available software installed on the cluster, at the system prompt enter: module avail

Here is the sample output of module avail:

hpcuser@nscc04 ~ \$ module	avail	
/a	.pp/modules/dev-gnu	
autoconf/2.69	gcc/4.9.3(default)	gmp/6.1.2
leveldb	mkl/gcc	protobuf/2.6.1
autogen/5.18.7	gcc/5.1.0	gsl/2.1
libgd/2.1.1	mpc/1.0.3	readline/6.3
automake/1.15	gdb/7.10	guile/2.0.11
libiberty/4.9.3	mpfr/3.1.4	snappy/1.1.3
binutils/2.26	gflags	
hdf5/1.8.16/gcc493/serial	libtool/2.4.6	mpiP/3.4.1/gcc493
yasm/1.3.0		
boost/1.59.0/gcc493/serial	glog/0.3.3	isl/0.14
libunistring/0.9.5	openmpi/gcc493/1.10.2	zlib/1.2.8
dejagnu/1.5.3	gmp/6.1.0	isl/0.22.1
lmdb	pcre/8.39	
/ap	p/modules/dev-gpu	
caffe	cuda/9.2	cuda91/profiler/9.1.85
openmpi/gcc493_gpu/3.0.1	tensorflow/1.4	
caffe-nv	cuda80/toolkit/8.0.44	cuda91/toolkit/9.1.85
tensorflow/1.0	tensorflow/1.4-cpu	
cuda/10.1	cuda91/blas/9.1.85	digits
tensorflow/1.0-cpu	tensorflow/cpu	
cuda/7.0	cuda91/fft/9.1.85	opencv/2.4.3
tensorflow/1.0+keras	tensorflow/gpu(default)	
cuda/7.5	cuda91/nsight/9.1.85	
openmpi/gcc493_gpu/1.10.4	tensorflow/1.0-python3	torch/2016-08-02
Figure 3.1: mo	dule avail Command on NSC	C ASPIRE 1

Figure 3.1 shows several software preinstalled on the cluster. The software highlighted in red is OpenMPI, which we will be using in the next few steps.



3.2 Introduction to MPI

MPI, known as Message Passing Interface, is a widely used parallel computing library [10]. Unlike normal tasks running with a single process, many processes are working together to perform a task in parallel computing [11]. To enable processes to work together, we need a mechanism such as MPI to help exchange information between processes.

The key concept in MPI is the notion of a communicator. A communicator defines a group of processes that communicates with one another. In this group of processes, each process is assigned a unique ID called rank, and they explicitly communicate with one another by the ranks.

The message passing is achieved by sending and receiving operations among processes. A process may send a message to another process by providing the rank of the process and a unique tag to identify the message. The receiver can post a receive for a message with a given tag, and handle the data accordingly. This kind of communication is known as point-to-point communication.

Another kind of communication is called collective communication, where processes communicate with everyone else. For example, when a master process needs to broadcast information for all its worker processes. It is not efficient if we make each worker process do a point-to-point communication with a master process.

A typical MPI program can be structured as follows:

MPI include file	<pre>#include "mpi.h" #include <stdio.h></stdio.h></pre>
Brogram Begins	<pre>#include <stdlib.h></stdlib.h></pre>
	int main (int argc, char *argv[]) {
Serial code	<pre>int numtasks, rank, dest, source, rc, count, tag=1; char inmsg, outmsg='x'; MPI_Status Stat;</pre>
Initialize MPI environment Parallel code begins	MPI_Init(&argc,&argv); MPI_Comm_size(MPI_COMM_WORLD, &numtasks); MPI_Comm_rank(MPI_COMM_WORLD, &rank);
Do work & make message passing calls	<pre>if (rank == 0) { dest = 1; source = 1; rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD); rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat); }</pre>
Terminate MPI environment Parallel code ends	<pre>else if (rank == 1) { dest = 0; source = 0; rc = MPI Recv(&inmsg, 1, MPI CHAR, source, tag, MPI COMM WORLD, &Stat); rc = MPI Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD); } </pre>
Serial code	MPI_Finalize();
Program Ends	

Figure 3.2: MPI Program Structure [10]



3.3 Exercise 1: First MPI Program

In this exercise, we will apply what we have learnt about MPI to writing a basic MPI program so that you have a better understanding on how an MPI program is executed, as well as the spawning of processes facilitated by MPI.

We will first ensure that the NSCC Cluster has the MPI Module required to run our program. Then we will proceed to write our MPI program with the necessary commands needed to set up the MPI execution environment. The functions of each command used will also be explained. Finally, you can compile, run your program and observe the results.

3.3.1 Writing your First MPI Program – mpi_hello_world.c

The program, mpi_hello_world.c is a basic Hello World program that prints the processor name the program is executed on, as well as the rank of each process in the communicator.

Step 1: Load MPI Module to NSCC ASPIRE 1 Cluster

By using the module avail command, we can see the different MPI module available on the cluster.

MPI is a standard for message passing interface. There are different implementations available. In this guide, we will use OpenMPI with the gcc compiler.

Load the OpenMPI module via the following command: module load openmpi/gcc493/1.10.2

To check whether your module is successfully loaded, use the command: module list

```
hpcuser@nscc04 ~ $ module load openmpi/gcc493/1.10.2
hpcuser@nscc04 ~ $ module list
Currently Loaded Modulefiles:
1) binutils/2.26 3) mpfr/3.1.4 5) is1/0.14 7) openmpi/gcc493/1.10.2
2) gmp/6.1.0 4) mpc/1.0.3 6) gcc/4.9.3
Figure 3.3: List of Modules Loaded
```

Figure 3.3 shows that other modules have been loaded. These are modules that OpenMPI needed as dependencies, which we do not need to deal with when loading modules.

Step 2: Creating a New File

Open a text editor and create a file called mpi_hello_world.c [12]. Common editors on Linux include emacs, vi, joe, etc. For example: vi mpi_hello_world.c



This creates and opens a file named mpi_hello_world.c under the current path. Press 'i' to enter the editing mode in vi.

Step 3: Include Header Files

```
//
// Exercise 1 – mpi_hello_world.c
//
#include <mpi.h>
#include <stdio.h>
```

In every MPI program, we will start with the MPI header file #include <mpi.h>.

Step 4: Setting the MPI Execution Environment

```
int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);
    // Get the number of processes
    int world size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    // Get the rank of the process
    int world rank:
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name len;
    MPI_Get_processor_name(processor_name, &name_len);
    // Print off a hello world message
    printf("Hello world from processor %s, rank %d out of %d processors\n",
            processor_name, world_rank, world_size);
    // Finalize the MPI environment.
    MPI_Finalize();
```

Next, MPI_Init() constructs all of MPI's global and internal variables. For example, a communicator has formed around all processes spawned, and unique ranks are assigned to each process.

MPI_Comm_size() returns the size of a communicator and MPI_COMM_WORLD encloses all of the processes in the job. In other words, this call returns the number of processes that were requested for the job.



Getting Started with NSCC Supercomputing on ASPIRE 1

MPI_Comm_rank() returns the rank of a process in a communicator. Each process inside a communicator is assigned an incremental rank starting from zero. The ranks of the processes are primarily used for identification purposes when sending and receiving messages.

MPI_Get_processor_name() obtains the actual name of the processor on which the process is executing.

Finally, MPI_Finalize() is used to clean up the MPI environment and no more MPI calls can be made after this command.

To save your file and exit vi, use 'Esc' to exit from editing mode, followed by: :wq

Here, 'w' means written and 'q' means quit.

3.3.2 Compile and Run your First MPI Program

Like any other programs, you need to compile your MPI program before running it. mpicc -o mpi_hello_world mpi_hello_world.c

You should see the executable mpi_hello_world after compiling.

```
hpcuser@nscc04 hello $ mpicc -o mpi_hello_world mpi_hello_world.c
hpcuser@nscc04 hello $ ls
mpi_hello_world mpi_hello_world.c
Figure 3.4: Compiling MPI Program
```

Do note that MPI programs cannot be compiled using the 'gcc' command alone as there are a few other steps required which are beyond the scope of this guide. Therefore mpicc is a shortcut to all the steps required for compiling MPI programs.

Run your MPI program using the following command: mpirun -np 2 ./mpi_hello_world

-np indicates how many processes you would like to run your program with.

hpcuser@nscc04 hello \$ mpirun -np 2 ./mpi_hello_world Hello world from processor nscc04, rank 0 out of 2 processors Hello world from processor nscc04, rank 1 out of 2 processors Figure 3.5: Running MPI Program

3.3.3 Understanding the Output

Now let's use 10 processes instead. mpirun -np 10 ./mpi_hello_world

This gives you an output of 10 lines (Figure 3.6).



Getting Started with NSCC Supercomputing on ASPIRE 1

hpcuser@nscc04 hello \$ mpirun -np 10 ./mpi_hello_world Hello world from processor nscc04, rank 1 out of 10 processors Hello world from processor nscc04, rank 3 out of 10 processors Hello world from processor nscc04, rank 5 out of 10 processors Hello world from processor nscc04, rank 7 out of 10 processors Hello world from processor nscc04, rank 9 out of 10 processors Hello world from processor nscc04, rank 6 out of 10 processors Hello world from processor nscc04, rank 8 out of 10 processors Hello world from processor nscc04, rank 8 out of 10 processors Hello world from processor nscc04, rank 8 out of 10 processors Hello world from processor nscc04, rank 0 out of 10 processors Hello world from processor nscc04, rank 2 out of 10 processors Hello world from processor nscc04, rank 4 out of 10 processors Hello world from processor nscc04, rank 4 out of 10 processors Hello world from processor nscc04, rank 4 out of 10 processors

By indicating 10 processes, the program spawns 10 processes in the communicator that each produces an output 'Hello world' along with their rank.

3.3.4 Takeaways

From this exercise, you should have a better understanding of how to run an MPI program, the various MPI commands or functions needed for execution including its rationale and its function in the program. Most of the MPI commands or functions covered in the example need to be present in every MPI program, and hence it is important that you understand its purpose. Lastly, we hope that you have learnt the basics of writing an MPI program and we will build on this knowledge in the next section.



4 Run Interactive Batch MPI Jobs

The objective of this section is to run your MPI program in batch mode **interactively** on a job scheduler (PBS Pro) via submitting to the PBS queue [14]. First, we will learn how a batch job is implemented by the queues in the cluster. Then we will learn how to transfer files between your local machine and PBS Compute Manager, before learning how to submit your program in batch to the PBS queue.

4.1 NSCC Job Scheduler - PBS Pro

NSCC uses PBS Pro to schedule jobs on the cluster. This scheduler provides a workload management solution that maximizes the efficiency and utilization of high-performance computing (HPC) resources and improves job turnaround. Several queues (PBS Queue) have been created to satisfy the resource requirements of the various workloads which use the system.

4.1.1 PBS Queue

The PBS Queue allows users to share resources on supercomputers. As such, you can run jobs that require large amounts of different resources by submitting your jobs to the queues below, which will be subsequently scheduled to run by PBS Pro. The table below lists the queues available to all NSCC users. There may be special queues on the system which are not normally available to everyone.

Queue Name	Resources Available	Remarks		
normal 1160 Compute nodes 24 cores per server 4GB/core memory or 96 GB per server		The queue normal is a routing queue and does not execute any jobs. It only routes the job to the internal queues based on the resource requirement		
gpu 128 GPU nodes 24 Cores per server 4GB/core memory or 96 GB memory per server		This queue is a routing queue and does not execute any jobs This queue routes the jobs to the internal queues based on resource requirement		
largemem	9 Compute nodes 24/48 cores per server 1TB/4TB/6TB memory configurations	This is an execution queue which can take large memory jobs requiring more than 96GB of memory per server		
iworkq	Special queue for interactive or visualization jobs	Jobs submitted from the Display manager to be dispatched to this queue		



4.1.2 Submitting Jobs to PBS Queue

When Batch Jobs are submitted, they are assigned to a queue and wait for its turn to run by the scheduler.

echo COMMAND | qsub -q normal -l select=1:ncpus=2,walltime=00:13:00

This submits a job to the normal queue, which requires 2 CPU cores and 13 minutes of wallclock time.

\$ echo sleep 10 | qsub -q normal -l select=1:ncpus=2,walltime=00:13:00
INFO: As you have not specifed whether this job as a personal or project
run,
INFO: the system will count this as a personal run by default.
INFO:
INFO: Please use -P Personal or -P <project_id> to properly account for
your job.
INFO:
INFO: Alternatively, in your job submission script please add
INFO: #PBS -P Personal or #PBS -P <project_id>
INFO: submitting job...
2123248.wlm01

Figure 4.1: Submission of Job Interactively

You will see the output as above (Figure 4.1). Use 'ls' command to check your directory. You will see two files called STDIN. e<<u>YOUR_JOB_ID></u> and STDIN.o<<u>YOUR_JOB_ID></u>.

hpcuser@nscc04 ~ \$ ls STDIN.e2123248 STDIN.o2123248 Figure 4.2: Output and Error Files of Job

PBS recognises your job name as STDIN. In general, your job outputs and any PBS messages can be found in the file <Job-name>.o<Job-ID>. If there are any MPI or system errors in running the program the error messages will be found in <Job-name>.e<Job-ID>. In this case, STDIN.o2123248 will contain your outputs while STDIN.e2123248 will contain the error messages (if any).

4.2 Transferring files between PBS Compute Manager and Local Machine

The PBS Compute Manager is a simple web interface where users can submit jobs, monitor jobs which are running and transfer files.



i. You may log in to PBS Compute Manager at <u>https://aspireweb.nscc.sg</u>.



Figure 4.3.1: PBS Compute Manager Login Page

ii. Upon login, click on Remote Files on the bottom left.

MONITORING Connect Terminate Move to Queue * Remove More Actions * Image: Applications Job Id State Application Name Queue Name Server Name Image: Applications No jobs to show No jobs to show No jobs to show Server Name Image: Applications Image: Applications No jobs to show Server Name Server Name Image: Applications Image: Applications No jobs to show Server Name Server Name Image: Applications Image: Applications No jobs to show Server Name Server Name Image: Applications Image: Applications No jobs to show Server Name Server Name Image: Applications Image: Applications No jobs to show Server Name Server Name Image: Applications Image: Applications Image: Applications Server Name Server Name Image: Applications Image: Applications No jobs to show Server Name Server Name Image: Applications Image: Applications Image: Applications Server Name Server Name Image: Applications Image: Applications <t< th=""><th>🛆 Compute Manager</th><th></th><th></th><th></th><th></th><th></th><th>e0201019 0 Messages</th><th>0</th></t<>	🛆 Compute Manager						e0201019 0 Messages	0
Image: Server		🖹 Download 🖉 Resul	omit 🛛 🎑 Connect 🗋 ⊘ Terr	ninate	🗶 Remove		More Actions	•
P Applications No jobs to show P Image: Sources P P Image: Sources P<	🗹 🤱 My Jobs	Job Id	State	Application	Name	Queue Name	Server Name	۲
 ■ Servers ■ ①	Applications			No jobs te	o show			^
	Servers							
€ □ Time	Job States							
	🔿 🔲 🛅 Time							
								÷
m Meetong	Monitoring							
Remote Files	Remote Files	Overview Properties	Files Actions					

Figure 4.3.2: PBS Compute Manager

iii. You will find the upload and download buttons in the bar.

	😤 C		G		Filter by Name	
Name 📥	/	1	Туре	Size	Date Modified	
ignome2			Folder		05-10-2020 22:09:33	*
imozilla .mozilla	Download	Upload	Folder		05-10-2020 22:09:33	Ε
🛅 .ssh			Folder		05-29-2020 15:50:20	
intel			Folder		05-22-2020 14:32:20	
.bash_history			bash_history Fi	22.837 KB	06-17-2020 14:24:01	
.bash_logout			bash_logout Fil	18 bytes	05-10-2020 22:09:33	
.bash_profile			bash_profile Fil	176 bytes	05-10-2020 22:09:33	
.bashrc			bashrc File	190 bytes	05-10-2020 22:09:33	_
				5001		Y

Figure 4.3.3: PBS Compute Manager Files



4.3 Exercise 2: Interactive Batch MPI Program

Now that you have understood how to submit jobs to the PBS queues to execute large jobs, we will proceed to write a program that utilizes more than one node to execute. Subsequently, we will show you how to submit the batch job interactively and request for a specific amount of resources to execute it.

4.3.1 Write your Interactive Batch MPI Program – mpi_prime.c

The program, mpi_prime.c [15], counts the number of prime between 1 and N where N starts from 1 and increases by 2 times every iteration until the user input, and prints the time taken to count. The program only reads an input that is a power of 2.

The counting is done across P processes, as indicated by the number of MPI process the user has chosen. To divide the work among P processes evenly, the processor ID starts at 2 + ID and increases by P each time.

Step 1: Include Header Files

Create a file called mpi_prime.c.

```
//
// Exercise 2 - mpi_prime.c
//
#include <math.h>
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <time.h>
```

Firstly, include the MPI header file and the all necessary files required. Since we will be using mathematical formulae, we would require <math.h>. We need <time.h> as we will be tracking the time taken to count the number of primes. Finally, we also need <stdlib.h> for atoi() function.

Step 2: Write Function Declarations

```
int main (int argc, char *argv[]);
int prime_number (int n, int id, int p);
void timestamp ();
```

Next, we will have 3 functions in this example. In addition to the main function, prime_number() is used to count the number of primes, while timestamp() returns the date and time.



Step 3: Define all Functions

```
int main ( int argc, char *argv[]) {
    int id, ierr, n, n_factor, n_hi, n_lo, p, primes, primes_part;
    double wtime, start, end;
    n_{lo} = 1;
    n_{factor} = 2;
    if (argc < 2) {
        fprintf(stderr, "State upper limit");
        exit(1);
    }
   n_hi = atoi(argv[1]);
    if (ceil(log2(n_hi)) != floor(log2(n_hi))) {
        fprintf(stderr, "Please enter an input that is a power of 2.");
        exit(1);
    }
    ierr = MPI_Init ( &argc, &argv );
    if (ierr != 0) {
        printf ( "n" );
        printf ( "PRIME_MPI - Fatal error!\n" );
        printf ( " MPI_Init returns nonzero IERR.\n" );
        exit (1);
    }
    //Get the number of processes.
    ierr = MPI_Comm_size ( MPI_COMM_WORLD, &p );
    //Determine this processes's rank.
    ierr = MPI_Comm_rank ( MPI_COMM_WORLD, &id );
    if (id == 0) {
        printf("Running program PRIME_MPI\n");
        printf("Start: ");
        timestamp ( );
        start = MPI_Wtime();
        printf ( "n" );
        printf ( " This is an MPI example program to count the number of primes.\n" );
        printf (" The number of logical processes is %d.\n", p);
        printf ( " The input entered is %d.\n", n_hi);
        printf ( "\n" );
        printf ("
                          Ν
                                    Prime
                                                    Time(s)n'');
        printf ( "\n" );
```



}

```
}
    n = n_lo;
    while (n \le n_h)
        if (id == 0) {
            wtime = MPI_Wtime ();
        }
        ierr = MPI_Bcast ( &n, 1, MPI_INT, 0, MPI_COMM_WORLD );
        primes_part = prime_number ( n, id, p );
        ierr = MPI_Reduce ( &primes_part, &primes, 1, MPI_INT, MPI_SUM, 0,
                MPI COMM WORLD );
        if (id == 0) {
            wtime = MPI_Wtime () - wtime;
            printf ( " %8d %9d %18f\n", n, primes, wtime );
        }
        n = n * n_factor;
    }
    //Terminate MPI.
    ierr = MPI_Finalize ();
    //Terminate.
    if (id == 0)
        printf ("\n");
        printf ( "PRIME_MPI - Master process:\n");
        printf ( " Normal end of execution.\n");
        end = MPI_Wtime();
        printf ( "n" );
        printf("End: ");
        timestamp ( );
        printf("PRIME_MPI ran for a duration of %fs.\n", end-start);
        printf ( "n" );
    }
    return 0;
int prime_number ( int n, int id, int p) {
    int i, j, prime, total;
    total = 0;
    for (i = 2 + id; i \le n; i = i + p) {
```



```
prime = 1;
        for (j = 2; j < i; j++) {
            if ((i \% j) == 0) {
                 prime = 0;
                 break;
             }
         }
        total = total + prime;
    }
    return total;
}
void timestamp() {
# define TIME SIZE 40
    static char time_buffer[TIME_SIZE];
    const struct tm *tm;
    time_t now;
    now = time ( NULL );
    tm = localtime ( &now );
    strftime ( time_buffer, TIME_SIZE, "%d %B %Y %I:%M:%S %p", tm );
    printf ( "%s\n", time_buffer );
    return;
# undef TIME_SIZE
```

Lastly, we will define all functions that are required for our program.

As discussed before in section 3.3.2, the basics of the MPI commands needed applies here similarly. We will explain the new MPI commands used in this program.

MPI_Wtime() returns the elapsed time in seconds since an arbitrary time in the past. MPI_Bcast() broadcasts a message from the process with rank root to all other processes of the communicator.

MPI_Reduce() reduces values on all processes to a single value.

Step 4: Compile your MPI Prime Program

Compile your program: module load openmpi mpicc -o mpi_prime mpi_prime.c -lm

The program can also be seen in the folder.



Step 5: Request Computation Resources

Request resources for your program: qsub -I -q normal -l select=2:ncpus=24:mpiprocs=12 -l walltime=1:00:00 -P Personal -N mpi_prime

This issues the PBS directives to run your MPI Prime program. In this example, it selects the normal queue, requests for 2 server nodes with 24 CPUs and 12 MPI processes each, and with a walltime of 1 hour. The -P directive indicates the project as a personal project and -N assigns the name mpi_prime to it.

Once the resources have been acquired and the job is ready to run, you will see the following:

```
hpcuser@nscc04 prime $ qsub -I -q normal -l select=2:ncpus=24:mpiprocs=12 -
l walltime=1:00:00 -P Personal -N mpi_prime
qsub: waiting for job 2123249.wlm01 to start
qsub: job 2123249.wlm01 ready
Figure 4.4: Message for a Job Ready to Run
```

4.3.2 Run your MPI Prime Program

Next, load your MPI module and run the program: module load openmpi cd \$PBS_O_WORKDIR mpirun -np 12 ./mpi_prime 1024

This runs the program with 12 processes and input of 1024.

Note: in this example, we have requested for 48 cores, but the program is only using 12. In this case, we seem to have "wasted" 36 cores. However it is quite common to request for more cores than MPI processes when making use of OpenMP or pthreads in very compute-intensive mathematical applications or when a large memory footprint is required, but these issues are beyond the scope of this guide.



4.3.3 Understanding the Output

```
Running program PRIME MPI
Start: 10 December 2020 05:25:51 PM
 This is an MPI example program to count the number of primes.
 The number of logical processes is 12.
 The input entered is 1024.
        Ν
                Prime
                              Time(s)
                  0
                              0.001251
        1
        2
                  1
                             0.000006
                 2
                             0.00003
        4
       8
                             0.000004
            4
6
11
18
31
54
                 4
       16
                              0.00003
                              0.00003
       32
                             0.000013
       64
      128
                             0.000009
      256
                             0.000009
      512
                 97
                              0.000029
          97
172
     1024
                              0.000363
PRIME MPI - Master process:
 Normal end of execution.
End: 10 December 2020 05:25:52 PM
PRIME MPI ran for a duration of 0.210847s.
                  Figure 4.5: Output for MPI Prime Program
```

Each row represents the time taken to count the number of primes (Prime) that are present between 0 and N, up to 1024 which is our input. At the end of the program, the time taken to run the entire program is also shown so that we can keep track of the efficiency of the job executed.

4.3.4 Takeaways

From this exercise, we hope that you have a deeper understanding of the purpose and benefits of MPI in a program. In addition, you should have learnt how to request computation resources for your program which will influence the speed and thus the efficiency of the execution.

With the new MPI commands that were introduced, you should be able to write your MPI program and eventually explore more advanced MPI usage.



5 Run Automated Batch MPI Jobs

The objective of this section is to learn how to **automate** the submission of a batch job using a shell script.

In this section, we will first introduce how inputs and outputs are indicated when running a program and then ensuring your jobs are executed in your directory. Next, we will write a shell script to submit a job to the queue. Following this, we will monitor the job progress and check the output of the program.

At the end of this section, we will introduce another batch script with more functionalities involving exception handling to ensure that your batch job runs smoothly.

5.1 Inputs and Outputs

In some cases, your program may be required to read in the input and redirect the output to an output file.

mpirun -np 24 ./program_name 100 > output_file

As seen above, the input should always come after the program name, while the output file's name should be indicated after an arrow. When a single arrow > is used, the output will overwrite any existing content in the output file. If you would like to redirect the output to an existing output file, you may use double arrows >> instead.

5.2 Running jobs in your directory

When submitting a batch job, computation resources are shared between many users and hence it is important to ensure that the jobs you have submitted are executed in your directory.

cd \$PBS_O_WORKDIR

This executes the commands in the current working directory. Ensure that the command line above is present in every batch script you write.

5.3 Exercise 3 (Part 1): Shell Script for Automated Batch Job

Exercise 3 is split into 2 parts. In this part, we will first learn the basics of writing a simple script to automate the submission of our batch job. It will include a single run for our program. After understanding the structure of a script, Part 2 will expand our script to include checks for abnormal termination of jobs and subsequent error handling.



5.3.1 Write a Batch Script – batch_hello_world.pbs

Create a new script named 'batch_hello_world.pbs'. Input the following code:

```
#!/bin/bash
#
# Exercise 3 – batch_hello_world.pbs
#
#PBS -q normal
#PBS -P Personal
#PBS -1 select=1:ncpus=10:mpiprocs=10
#PBS -1 walltime=00:05:00
#PBS -j oe
#PBS -o JobOutput
module load openmpi
cd $PBS_O_WORKDIR
mpicc -o mpi_hello_world ./mpi_hello_world.c
mpirun -np 10 ./mpi_hello_world > output
```

5.3.2 Understanding your Batch Script – batch_hello_world.pbs

#!/bin/bash tells the computer that this is a bash shell.

#PBS -q normal selects the "normal" queue.

#PBS - **P** Personal specifies that this is a "Personal" job.

#PBS -1 select=1:ncpus=10:mpiprocs=10 and **#PBS** -1 walltime=00:05:00 specifies that this job is going to use 1 set of 10 CPU cores and be allocated a time of maximum 5 minutes.

#PBS - j oe combines the output and error files into a single file.

#PBS -o JobOutput specifies the name of the output file.

module load openmpi loads the OpenMPI module to allow for compilation and execution of MPI jobs.

cd **\$PBS_O_WORKDIR** changes the directory you are currently working on to the PBS directory

mpicc -o mpi_hello_world ./mpi_hello_world.c compiles the program to mpi_hello_world. mpirun -np 10 ./mpi_hello_world > output runs mpi_hello_world and redirects the output to output.



5.3.3 Submit and monitor your Batch Job

After writing the script, you may submit to PBS Queue via the command: qsub batch hello world.pbs

You will then receive a Job ID:

```
hpcuser@nscc04 hello $ qsub batch hello world.pbs
2123250.wlm01
```

Figure 5.1.1: Job ID from Job Script Submission

In this example, 2123250 is the Job ID for executing batch hello world.pbs.

You may use the following command to track your jobs in PBS Queue: qstat

Figure 5.1.2: Job Monitoring using qstat Command						
2123250.wlm01	batch_hello_wor	hpcuser		0	Q	dev
Job id	Name	User	Time	Use	S	Queue
hpcuser@nscc04	hello \$ qstat					

5.3.4 Check your Output

Once the job is done, find your output in the file named 'output' in your working directory.

```
Hello world from processor std1669, rank 1 out of 10 processors
Hello world from processor std1669, rank 2 out of 10 processors
Hello world from processor std1669, rank 4 out of 10 processors
Hello world from processor std1669, rank 6 out of 10 processors
Hello world from processor std1669, rank 9 out of 10 processors
Hello world from processor std1669, rank 0 out of 10 processors
Hello world from processor std1669, rank 3 out of 10 processors
Hello world from processor std1669, rank 5 out of 10 processors
Hello world from processor std1669, rank 7 out of 10 processors
Hello world from processor std1669, rank 8 out of 10 processors
```

Figure 5.2: Output from Hello World Job

5.4 **Exercise 3 (Part 2): Shell Script for Automated Batch Job**

In this part, we will learn how to check for abnormal job termination in our script and the subsequent steps to successfully handling these errors.

Abnormal Job Termination 5.4.1

We may want to be informed when our program terminates abnormally while executing a job such that we are able to detect errors and account for it.

When a program or a command terminates, we will receive a return code from it. For example, there is always 'return 0' at the end of the main function in a C program as shown below:

int main()



	return 0;			
}				

As bash scripts are executed serially, each command will not be executed until the previous part has finished. Hence, we can insert 'echo \$?' between commands to obtain each command's exit code and detect any abnormal termination.

```
YOUR_PROGRAM
echo $?
```

5.4.2 Write a Batch Script – batch_prime.pbs

When running a batch job, we would encounter the need to run a job multiple time. Hence, this would require conditional execution and exception handling in order to run a batch job smoothly, which we will be covering in the following sections.

Create a new script named 'batch_prime.pbs'. Input the following code:

```
#!/bin/bash
#
# Exercise 4 – batch_prime.pbs
#
#PBS -q normal
#PBS -l select=1:ncpus=24:mem=4G:mpiprocs=24:ompthreads=1
#PBS -1 walltime=1:00:00
#PBS -m ae
#PBS -M youremail@gmail.com
#PBS -P Personal
#PBS -N prime
module load openmpi
cd $PBS_O_WORKDIR
mpirun -np 24 ./mpi_prime 64 > output_test
if [[ $? -eq 0 ]]; then
 for i in $(seq 1 3); do
   mpirun -np 24 ./mpi_prime 64 >> output_test
   if [[ $? -eq 1 ]]; then
     continue
   fi
  done
  else
   for i in 128 1024 8192 65536; do
     mpirun -np 24 ./mpi_prime $i >> output_test
     if [[ $? -eq 1 ]]; then
       continue
```



fi			
don	e		
fi			

5.4.3 Understanding your Batch Script – batch_prime.pbs

As discussed in Part 1, the PBS directives used in the script similarly applies here. We will explain the additional directives used in this script.

#PBS -m ae indicates you would like an email alert when the job is aborted or has finished execution.

#PBS -M <u>youremail@gmail.com</u> indicates your email address.

mpirun -np 24 ./mpi_prime 64 > output_test

- Runs the program with an input of 64.
- The single arrow after the input of 64 redirects the output to the output file named output_test.

if [[\$? -eq 0]]; then

This checks if each program run is executed successfully through the return code \$? before proceeding to the next run. If it returns 0, the run is successful. If it returns 1, the run is unsuccessful.

```
for i in $(seq 1 3); do
    mpirun -np 24 ./mpi_prime 64 >> output_test
    if [[ $? -eq 1 ]]; then
        continue
        fi
        done
```

- Upon successful execution of the first run, the same job is run 3 more times and the output is appended to the same output file.
- If any of the 3 runs is unsuccessful, it will simply proceed to the next run.

```
for i in 128 1024 8192 65536; do
    mpirun -np 24 ./mpi_prime $i >> output_test
    if [[ $? -eq 1 ]]; then
        continue
        fi
        done
```

- On the other hand, if the first run is unsuccessful, the job will be run with 24 MPI processes with different inputs (128, 1024, 8192, 65536) each time.
- Again, if any of the runs is unsuccessful, it will simply proceed to the next run.



5.4.4 Check your Output

Once the job is done, you can find 'output_test' in your working directory.

You will notice that the program will either run with an input of 24 for 4 times, or with an input of 24, 128, 1024, 8192 and 65535, but not both. This demonstrates exception handling of the job such that your job will not stop running midway, which achieves the efficiency of a batch job.

5.4.5 Takeaways

From this exercise, we hope that you have a better understanding of how a batch script works and the need to account for errors in your job run to prevent jobs from halting midway. Ultimately, the goal of batch jobs is to reduce manual monitoring and increase the efficiency of executing a program multiple time.



6 Summary

You should now be familiar with running jobs on the NSCC ASPIRE 1 both interactively, batch and through the submission of scripts. With the basic understanding of the various MPI commands needed to execute an MPI program as well as knowing the rationale and function behind, you should be able to get started with writing more advanced MPI programs.

Lastly, we have introduced shell scripting to help in running of automated batch jobs and ensuring that errors can be handled elegantly for a smooth job flow. Moving forward, you should be able to develop scripts more specific to your program or your batch jobs such that it can execute efficiently.

What you have learnt from this guide are just the rudimentary of MPI and PBS. You should learn how to make your MPI programs more efficient so that they can run faster, and how to make efficient use of PBS to schedule your jobs more optimally. Besides the many freely-available online guides, NSCC also conducts regular workshops on advanced programming (mostly MPI) and advanced job scheduling (PBS). You are strongly encouraged to attend these workshops to learn how to use NSCC's resources more efficiently.



7 References

- [1] "NSCC Software/Hardware Information", National Supercomputing Centre Singapore, [Online]. Available: <u>https://help.nscc.sg/softwarehardware-information/</u> [Accessed 10 Dec 2020].
- [2] "Basic Linux Tutorial", National Supercomputing Centre Singapore, [Online]. Available: <u>https://help.nscc.sg/wp-content/uploads/2016/03/BasicLinuxTutorial-v0.1.pdf</u> [Accessed 10 Dec 2020].
- [3] NSCC, "Software/Hardware Information", [Online]. Available: https://help.nscc.sg/softwarehardware-information/ [Accessed 10 Dec 2020].
- [4] "NSCC Help User Guides", National Supercomputing Centre Singapore, [Online]. Available: <u>https://help.nscc.sg/user-guide/</u> [Accessed 10 Dec 2020].
- [5] D. J. Barrett and R. E. Silverman, "Introduction to SSH", O'Reilly & Associates, [Online]. Available: <u>https://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch01_01.htm</u> [Accessed 10 Dec 2020].
- [6] "NSCC New User Starter Guide", National Supercomputing Centre Singapore, [Online]. Available: <u>https://help.nscc.sg/wpcontent/uploads/2017/06/NSCC_New_User_Starter_Guide_v0.1.pdf</u> [Accessed 10 Dec 2020].
- [7] "User Enrollment Guide", National Supercomputing Centre Singapore, 16 March 2016.
 [Online]. Available: <u>https://help.nscc.sg/wp-content/uploads/2016/03/NSCC-UserEnrollmentGuide-v0.1.pdf</u> [Accessed 10 Dec 2020].
- [8] "UNIX / LINUX Tutorial", [Online]. Available: <u>https://www.tutorialspoint.com/unix/index.htm</u> [Accessed 10 Dec 2020].
- [9] X. Delaruelle, "Environment Modules", [Online]. Available: <u>http://modules.sourceforge.net/</u> [Accessed 10 Dec 2020].
- [10] B. Barney, "Message Passing Interface (MPI)", Lawrence Livermore National Laboratory, 26 June 2020. [Online]. Available: <u>https://computing.llnl.gov/tutorials/mpi/#Getting_Started</u> [Accessed 10 Dec 2020].
- [11] B. Barney, "Introduction to Parallel Computing Tutorial", Lawrence Livermore National Laboratory, 26 June 2020. [Online]. Available: <u>https://computing.llnl.gov/tutorials/parallel_comp/</u> [Accessed 10 Dec 2020].
- [12] W. Kendall, "MPI Hello World", [Online]. Available: <u>https://mpitutorial.com/tutorials/mpi-hello-world/</u> [Accessed 10 Dec 2020].
- [13] J. Brockmeier, "Vim 101: A Beginner's Guide to Vim", 20 November 2009. [Online]. Available: <u>https://www.linux.com/training-tutorials/vim-101-beginners-guide-vim/</u> [Accessed 10 Dec 2020].
- [14] "NSCC PBSPro Quickstart Guide", National Supercomputing Centre Singapore, [Online]. Available: <u>https://help.nscc.sg/pbspro-quickstartguide/</u> [Accessed 10 Dec 2020].



- [15] J. Burkardt, "Count Primes Using MPI", Florida State University, 26 June 2020. [Online]. Available: <u>https://people.sc.fsu.edu/~jburkardt/c_src/prime_mpi/prime_mpi.html</u> [Accessed 10 Dec 2020].
- [16] NSCC, "NSCC Supercomputing: A Beginner's Guide to Running AI Jobs".