Imperial College London

RCS RESEARCH COMPUTING SERVICE
Gateway to Discovery

Introduction to HPC at Imperial

Katerina Michalickova – kmichali@imperial.ac.uk

The Graduate School, Research Computing and Data Science Programme

Outline for today

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs

Video

Video (12 min) short introduction to the HPC resource:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=4d5cc349-00a0-4889-bc53-abc600f32b44

This video (and all subsequent ones) covers the slides until the next “Outline for today slide”.

The videos are from 2020. Please make sure to always check the slides for up-to-date technical details.

Imperial College London

Why are we here today?

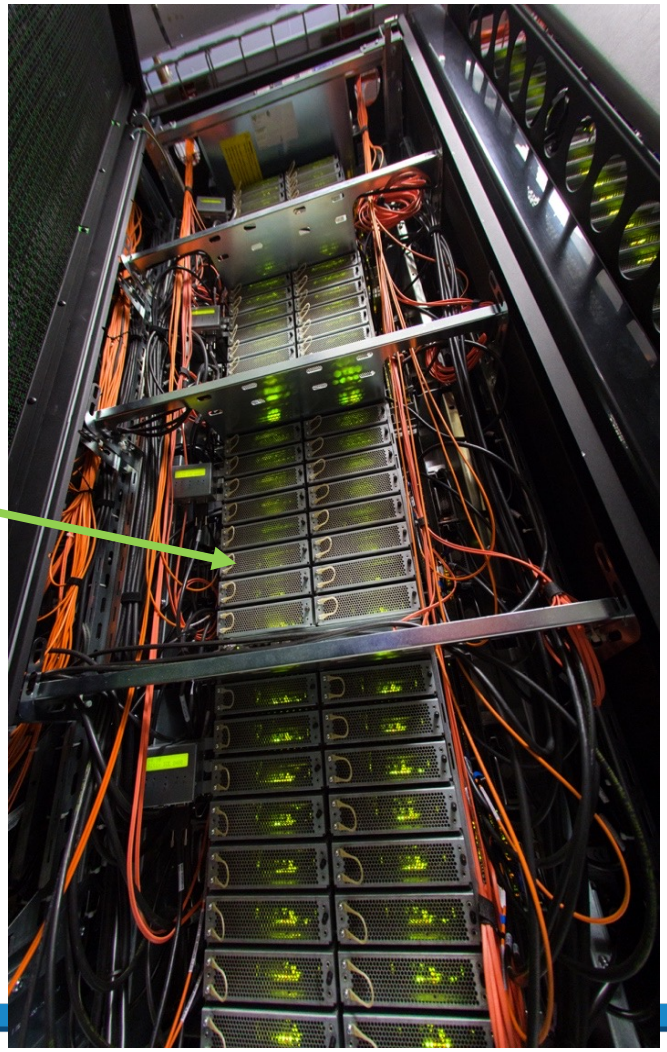
- Pillars (legs) of science today – theory, experimentation *and computing*.
- Much of scientific experimentation such as modeling, simulations or data management and mining is made possible by computing.
- If you need to use a central resource, it is very likely it will come in a form of **high performance computing cluster**.

Computer cluster

- A **computer cluster** consists of a set of connected computers that work together so that, in many respects, they can be viewed as a single system.

Inside a compute rack

a single computer
aka a compute node



Inside a disk rack

All compute nodes see one central file system – the Research Data Store

Imperial College London

What can clusters do?

- the cluster can serve to offload code execution from your laptop/workstation
  - code that runs too long or needs too much memory or disk space
- clusters are particularly useful for executing parallel code
  - on one compute node
  - on multiple compute nodes at once

- Note on speed of execution:
- the compute nodes have similar architecture to your desktop - they are not much faster
- the main advantage of cluster computing lies in parallel code execution

HPC terminology

I will try to avoid or explain again during the class. In case I forget myself, here are few terms that need explanation:

- **Job** = your program on the cluster
- **Submit job** = instruct the cluster to run your program
- **Node** = compute node = group of cores that can access the same memory (I may inadvertently also say a computer or a machine)
- **Memory** = main memory or RAM - fast memory directly connected to the processor, when your program is running it is stored in RAM together with needed data
- **Core** = the basic computation unit inside a processor that can run a single process
- **Serial code** = runs on one core
- **Parallel code** = program that runs on two or more cores

Workload types

The resource contains hardware components suited to different type of computations:

- High-throughput tasks – large number of relatively small jobs
- Parallel jobs on a single node
- Parallel jobs on multiple nodes
- Large single node memory jobs (up to TBs of memory)
- Jobs using GPUs (such as machine learning tasks)

Key contacts

- RCS website - https://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/
- User support - https://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/get-support/contact-us/
- Technical wiki - https://wiki.imperial.ac.uk/display/HPC/
- Weekly drop-in clinics - https://wiki.imperial.ac.uk/display/HPC/Attend+a+clinic
- Service status (under VPN/Zscaler)- https://selfservice.rcs.imperial.ac.uk/service-status
- Self-service (under VPN/Zscaler, interactive access, group management, job monitoring and extensions etc.) - selfservice.rcs.imperial.ac.uk

Outline for today

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs

Video

Video (5 min) showing how to log into the HPC resource:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=645bf8af-ced6-4e36-bfed-abc601539b66

Imperial College London

Before you login

- use **VPN or Zscaler** when off-campus

https://www.imperial.ac.uk/admin-services/ict/self-service/connect-communicate/remote-access/virtual-private-network-vpn/
https://www.imperial.ac.uk/admin-services/ict/self-service/connect-communicate/remote-access/unified-access/

Special instructions for Linux setup: https://wiki.imperial.ac.uk/display/HPC/Using+the+VPN

- on campus - use **Imperial-WPA wifi**

- for **regular HPC access**, register as follows
  https://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/get-access/

- some of you have only temporary access for this course - it expires on the first of each month

Log in using your terminal

Mac, Windows (with Bash) and Linux – start a terminal and type

`ssh your_username@login.hpc.ic.ac.uk`

```
katerina — ssh kmichali@login.hpc.ic.ac.uk — 99×29
-bash-4.2$ ssh kmichali@login.hpc.ic.ac.uk
Last login: Mon May 25 21:19:10 2020 from dyn1232-97.vpn.ic.ac.uk

               Imperial College London Research Computing Service
               --------------------------------------------------

  ***** Continue ONLY if you are an authorised user, otherwise EXIT now *****

Communications, including personal communications, made on or through  Imperial
College's computing and telecommunications systems may be monitored or recorded
to  secure effective  system operation  and for other lawful purposes.  See the
Information Systems Security Codes of Practice for more information.

                For any queries or assistance, please visit:

https://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/support/help/

   Service status https://selfservice.rcs.imperial.ac.uk/service-status


            Remember the logins are SHARED RESOURCES.
                        Programs that are
          long-running, memory-intensive or multithreaded
                   MUST BE RUN AS BATCH JOBS.

       Clinics are now held online 14:00 - 16:00 every Tuesday
         Book a consultation:  https://tinyurl.com/ycf4fvns

     Next scheduled maintenance Monday 29th and Tuesday 30th June
```

HPC environment

- Linux operating system
- Bash shell

```
dyn3184-184:~ katerina$ ssh kmichali@login.hpc.ic.ac.uk
Last login: Fri Mar 29 22:02:43 2019 from dyn1251-214.vpn.ic.ac.uk

                Imperial College London Research Computing Service
                --------------------------------------------------

  ***** Continue ONLY if you are an authorised user, otherwise EXIT now *****

Communications, including personal communications, made on or through  Imperial
College's computing and telecommunications systems may be monitored or recorded
to  secure effective  system operation  and for other lawful purposes.  See the
Information Systems Security Codes of Practice for more information.

For any queries or assistance, please visit:

https://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/support/help/

Follow our Twitter channel for updates: https://www.twitter.com/imperialRCS

Attend our weekly drop-in clinics held  Tuesday afternoons 14:00-16:00
in the ICT Training Room 204, Central Library, South Kensington Campus


 RDS usage at 15:50 on 3/4/2019

 Individual allocation /rds/general/user/kmichali

   Home       Data:  369GB of 1.00TB (37%)
              Files: 382K of 10.00M (4%)
   Ephemeral  Data:  0GB of 109.95TB (0%)
              Files: 0K of 20.97M (0%)


-bash-4.2$
```

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs

Video

Video (12 min) on software management on the cluster:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=c3a07ebb-c7c2-4517-a3ec-abc6015abd75

Module system for centrally installed packages

- Loading modules sets required environment to use a software package
- For the latest info, check

https://wiki.imperial.ac.uk/display/HPC/EasyBuild#EasyBuild-production

- List all modules available
  - module avail
- Find a module for your software
  - module avail matlab
- Load a module
  - module load matlab
- Load a specific version
  - module load matlab/R2018a
- List all loaded modules
  - module list
- Swap modules
  - module switch matlab matlab/R2018a
- Unload a module
  - module unload matlab/R2018a
- Get rid of all loaded modules
  - module purge

```
anaconda3/2.4.1
anaconda3/4.1.1
anaconda3/4.3.1(default)
anaconda3/personal
angsd/609
angsd/915
annovar/2013-03-18
annovar/2013-08-05
annovar/2015-06-17(default)
ansys/16.2
ansys/17.1
ansys/18.1
ansys/18.2
ansys/19.1-fluids
ansys/cfx/14.0(default)
ansys/cfx/15.0
ansys/cfx/16.1
ansys/fluent/14.0(default)
ansys/fluent/15.0
ansys/workbench/16.1
ant/1.6.1
ant/1.6.5
ant/1.8.1(default)
ants/2.2.0
ANTs/2015-02-23
aracne/2
armadillo/1.1.90
armadillo/1.2.0
armadillo/2.0.1
armadillo/3.2.4
armadillo/3.4.2
armadillo/6.200.2
armadillo/7.200.2(default)
arpack/96-patch(default)
aspect/1.4.0
aspect/1.5.0(default)
aster/11.3
atat/2.71(default)
ATK/2015.1
magma/2.18-5
magma/2011-07-05
magma/devel-modules(default)
make/3.82(default)
maker/2.10
maker/2.31.9
mamba/1.0.0
mantra/2012-11-19
maple/2016
mapsembler/2.2.4
mapsplice/2.2.0
maq/0.7.1(default)
marv/1.0.4
masurca/2.3.2
masurca/3.2.1
masurca/3.2.1_01202017
matlab/R2017a(default)
matlab/R2017b
matlab/R2018a
matplotlib/0.90.1(default)
mauve/2015-02-13
maxima/5.40.0(default)
mayavi/1.5(default)
mcarts/1.2.0
mcl/14.137
mcr/717(default)
mcscan/0.8
mcstas/2.1(default)
meep/0.10
meep/0.10.1
meep/0.20.3
meep/0.20.4
meep/1.0.3
meep/1.1.1(default)
meep/1.3
meep/1.4.3
meme/4.3.0
meme/4.3.0-nompi
meme/4.9.0
```

Cannot find what you're looking for?

- check if the package is available for Anaconda and install yourself
- submit a request for installation (variable waiting times): https://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/get-support/contact-us/
- you can install packages in your home directory (take advantage of various pre-installed libraries via module load)

Setting up python with Anaconda

We distribute personal version of Anaconda environment. Install on the command line using:

```
module load anaconda3/personal
anaconda-setup
```

This is done only once. Afterwards, using “module load ananconda3/personal” will setup your personal python environment.

Anaconda enables you to create separate package environments for your projects. This helps to avoid version and dependency conflicts.

For example, installing scipy in a separate environment:

```
conda create -n projectx          #create new environment
conda activate projectx           #activate env. (conda/source)
conda install scipy               #install scipy
conda deactivate                  #deactivate current env.
```

https://wiki.imperial.ac.uk/display/HPC/Python

Setting up Python with Easybuild

- https://wiki.imperial.ac.uk/display/HPC/Python#Python-EasybuildPython

Setting up R with Anaconda

R and libraries can be installed using personal Anaconda.

Setup an a new environment and install R inside it. Packages can be installed using conda install or from within R (plus other libs, e.g. bioconductor-teqc)

```
conda create -n Renv
conda activate Renv                    # (conda or source)
conda install r -c conda-forge        #install R in new env.
                                       #using conda-forge channel
conda install bioconductor-teqc -c conda-forge
conda deactivate
```

https://wiki.imperial.ac.uk/display/HPC/R

Setting up R with Easybuild

- https://wiki.imperial.ac.uk/display/HPC/R

Application-specific guidance

Information for selected popular applications:

https://wiki.imperial.ac.uk/display/HPC/Application+Guides

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs

Video

Video (15 min) on transferring files from and to the cluster:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=2e03e4d4-4b34-4aa2-a973-abc60147d00b

Video (30s) on mapping network drive on Windows:
https://vimeo.com/302461989

Imperial College London

Before you start

- If not on campus, use Imperial **VPN**

https://www.imperial.ac.uk/admin-services/ict/self-service/connect-communicate/remote-access/virtual-private-network-vpn/

Special instructions for Linux setup:

https://wiki.imperial.ac.uk/display/HPC/Using+the+VPN

- On campus - use **Imperial-WPA wi-fi**

File management

All computers in the HPC resource are connected to one parallel filesystem – Research Data Store (RDS). You get access to:

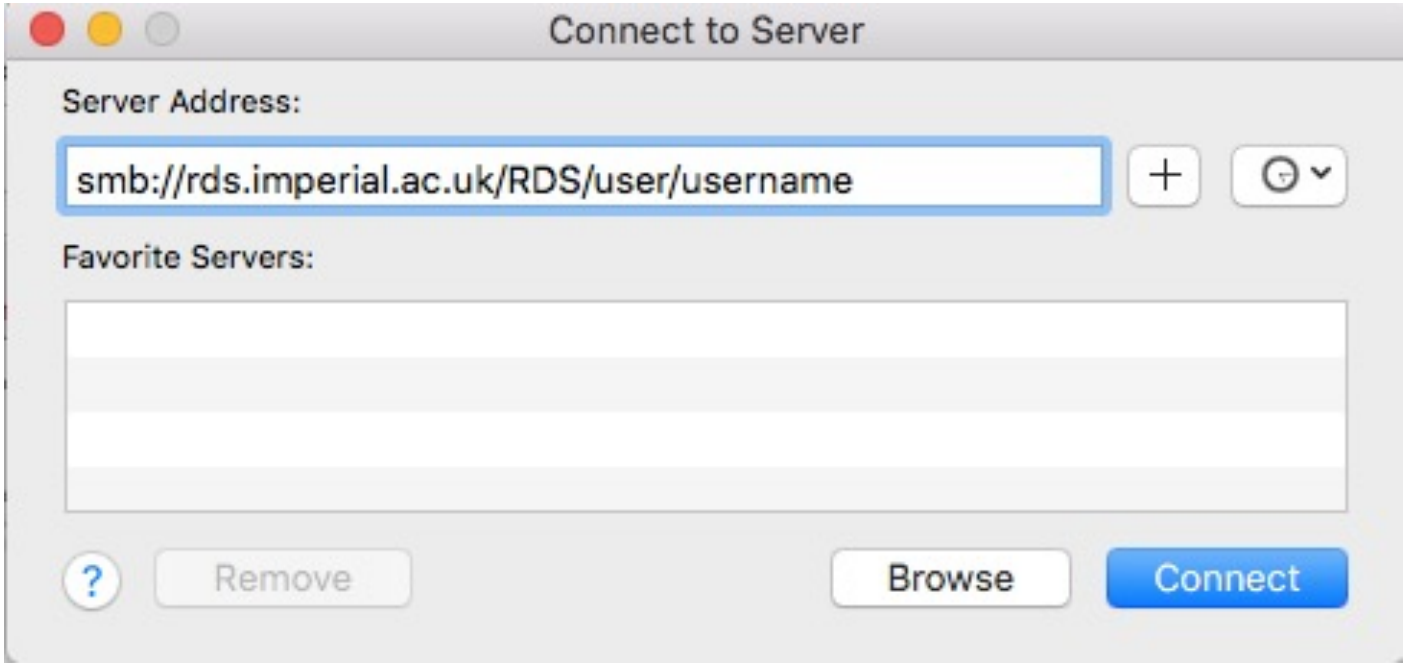
your home directory ($HOME)	• personal working space • 1 TB allocation (up to 10 million files)
temporary storage ($EPHEMERAL)	• additional individual working space • unlimited allocation • files deleted after 30 days
allocated project space ($RDS_PROJECT)	• your project allocations (if your supervisor has any)

check your usage with “quota -s”

documentation: https://wiki.imperial.ac.uk/display/HPC/Research+Data+Store

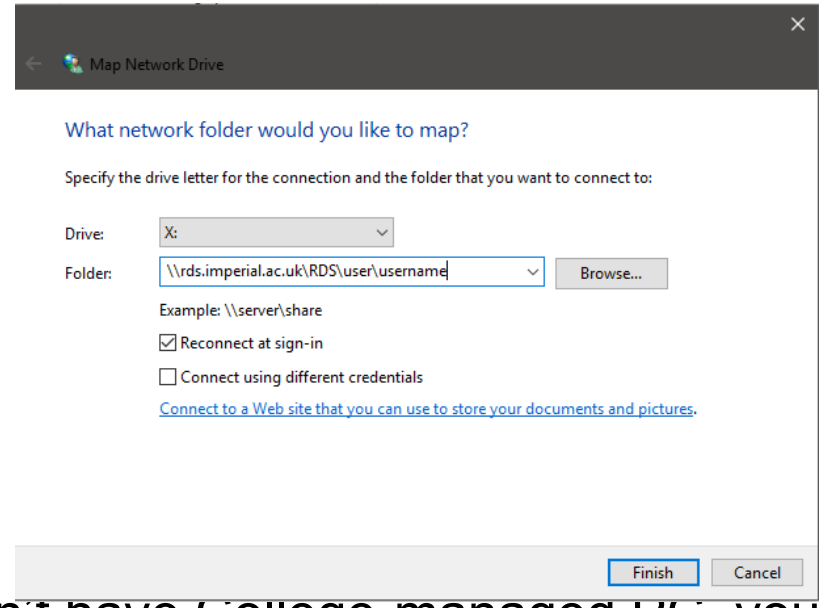
1a. Connect to RDS with Finder on your Mac

- **on a Mac (recent macOS version)**
- Finder -> Go -> Connect to Server -> Server Address
- **`smb://rds.imperial.ac.uk/RDS/user/your_username`**



1b. Connect to RDS with File Explorer on Windows

- **recent update of Windows 10**
- File Explorer -> My PC -> Map network drive
- **`\\rds.imperial.ac.uk\RDS\user\your_username`**



- if you don't have College-managed PC, you may need to use "**IC\username**" when loging in.

1b. Connect to RDS with File Explorer on Windows

Detailed description of connecting to RDS from Windows File Explorer:
https://github.com/ImperialCollegeLondon/RCS_UserSupport_public/tree/main/RDS_FSmounting/MS_Windows

2. Copying files between your laptop to RDS with FileZilla

- if on laptop, use **Imperial-WPA wifi** or **VPN** outside the College
- **FileZilla** file transfer client: https://filezilla-project.org

  type **sftp://login.hpc.ic.ac.uk** into the host window,

  (add username, password, leave port empty, click Quickconnect)

- **WinSCP – good alternative for Windows** https://winscp.net/

3. Copying files on the command line

Secure copy (scp) command will copy files between remote systems

- **scp file.txt username@login.hpc.ic.ac.uk:~**

this copies file.txt from the current directory on your machine to your home directory on RDS

- **scp username@login.hpc.ic.ac.uk:~/file2.txt .**

this copies a file2.txt from your home directory on RDS to the current directory on your machine

- The scp command can be used with any directory path, for example:

scp /Users/katerina/Desktop/file3.txt username@login.hpc.ic.ac.uk:/rds/general/user/kmichali/home/projectx

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs

Video

Video (16 min) on using the cluster to run a job:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=98193a17-af24-47c8-8fd9-abc701032f13

Your command

Assembling the command line is your responsibility, it requires some knowledge of the particular package. We can help with making sure the application runs correctly and advise you on managing your files. We can comment on individual applications to some degree (depending on having the expertise in the group).

Examples:

- **python** python_code.py –i input.txt –o output.txt
- **matlab** –batch matlab_script
- **R** CMD BATCH r_script.R
- **bash** shell_script.sh
- **g09** water.com > log
- **blastall** -i query -d fasta_db -o output
- mpiexec **nwchem** input.nw > log

Running software

Do not execute your program from the command line.

Why?

When you log into the cluster, you find yourself on one of four login nodes. If all users compute here, the nodes would be overwhelmed and the rest of the cluster would be idle. Instead, instruct the **cluster resource manager** to execute your job.

Cluster resource manager

The resource manager takes care of:

- keeps track of available hardware resources and how busy they are
- sorting your requests into job classes
- scheduling your requests
- starting jobs when compute resources become available
- monitoring jobs
- producing logs

X

Talking to the resource manager

- Write **a shell script (or job script)** instructing the resource manager what to do for you. Passing your finished script to the resource manager is called “submitting a job”.

- The resource manager needs to know your **command** and also what **resources** your job needs.

  - Every job script must specify **run time, memory (RAM) and number of cores** needed to run the job.

  - If the requested resources are exceeded during the job run, the job is terminated.

  - The bigger the resource, the longer the wait for a job to start.

A job script

```
#!/bin/bash
#PBS -l walltime=00:10:00
#PBS -l select=1:ncpus=1:mem=1gb

module load hellohpc

hellohpc.py
```

job params

load software

your command

Submit and monitor a job

katerina — ssh kmichali@login.hpc.ic.ac.uk — 100×40

```
-bash-4.2$ cat submit.pbs
#PBS -l walltime=00:10:00
#PBS -l select=1:ncpus=1:mem=1gb

module load hellohpc

cd $PBS_O_WORKDIR

hellohpc.py
sleep 30


-bash-4.2$
-bash-4.2$
-bash-4.2$ qsub submit.pbs
1066263.pbs
-bash-4.2$
-bash-4.2$
-bash-4.2$ qstat
   Job ID           Class            Job Name        Status     Comment
-------------- --------------- --------------------- -------- -------------
1066263        Debug           submit.pbs            Running  finishing in 10 minutes
-bash-4.2$
```

Resource manager commands

submit	**qsub** your_script	system returns a jobid
monitor	**qstat** qstat job_id	short overview of your jobs one job overview
delete	**qdel** jobid	caution: it takes a bit of time to see the job disappear from qstat output

Job status using selfservice

This page will list all your active jobs (queuing and running).

https://selfservice.rcs.imperial.ac.uk/jobs/qstat

Every job produces two log files

```
-bash-4.2$ ll *1037925
-rw------- 1 kmichali hpc-kmichali   0 Feb  5  2020 submit.pbs.e1037925
-rw------- 1 kmichali hpc-kmichali 462 Feb  5  2020 submit.pbs.o1037925
-bash-4.2$
```

error messages

screen output + res. manager messages

Imperial College London

Troubleshooting your jobs

All standard output (screen and errors - STDOUT and STDERR) from your job plus other useful info is written into log files. **Read the log files** carefully even if all seems fine.

Default names:

STDOUT - e.g. `submit.pbs.o9795758`

STDERR - e.g. `submit.pbs.e9795758`

If asking for help, use the ICT ASK form https://imperial.service-now.com/ask (search for “RCS Job problem”)

please provide as much information as your can to help us troubleshoot:

- job id
- steps to reproduce the error
- your script
- the logs (expected and observed)

Example of a python script (using personal anaconda)

```
#!/bin/bash
#PBS -l walltime=01:00:00
#PBS -l select=1:ncpus=1:mem=1gb

module load anaconda3/personal
source activate projectx

cd $PBS_O_WORKDIR
python myscript.py
```

Example of a R script (using personal anaconda)

```
#!/bin/bash
#PBS -l walltime=01:00:00
#PBS -l select=1:ncpus=1:mem=1gb

module load anaconda3/personal
source activate Renv

cd $PBS_O_WORKDIR
R CMD BATCH myscript.R
```

Matlab script (no anaconda involved)

```
#!/bin/bash
#PBS -l walltime=01:00:00
#PBS -l select=1:ncpus=1:mem=1gb

module load matlab/R2019a

cd $PBS_O_WORKDIR
matlab -batch myscript
```

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs

Video

Video (13 min) on job parameters:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=b17d7c23-9a1c-4246-a55c-abc8009cf49e

- Note: The video may show outdated information regarding job sizing. Please always check the latest job sizing on the RCS webpage. https://wiki.imperial.ac.uk/display/HPC/New+Job+sizing+guidance

Requirements syntax

The system expects the following syntax at the top of your job script:

lower case L

job duration requirement

```
#PBS -l walltime=24:00:00
#PBS -l select=1:ncpus=1:mem=1gb
```

number of cores on one node

number of compute nodes

memory per node

Job duration requirement

- This specifies how long you expect the program to run.
- Sensible values are 8, 24 hours, 48 hours or 72 hours. There is little benefit in selecting intermediate values.
- In some cases, specified time can be extended using the RCS self-service page. If your job is eligible for an extension, the option will appear on: https://selfservice.rcs.imperial.ac.uk/jobs
- If possible, use checkpointing, i.e. save intermediate results that can be used to restart your program.

Memory requirement

- When you run a program, the program and data are read into the main memory (RAM). The data is processed in memory and eventually written out (usually into a file on the disk).

- Majority of programs that do not deal with lots of data need no more than 2gb of RAM. The data usually determines the bulk of memory requirement. If your program reads large data files, ask for more memory.

- If in doubt, select 2GB to start with. If this is not enough, the resource manager will terminate your program and place a message (including how much RAM you tried to use) in a **log file**. In this case, increase the memory requirement and run again.

Number of parallel processes

- You program may be capable of parallel execution. If you are not sure, read the manual, ask your colleagues or check the HPC logs that specify how many cores your program tried to use. Failing that, visit the RCS clinic.

- On the HPC cluster, we generally see two types of parallelism:
  - shared memory (jobs that run on one node using threads)
  - distributed memory (jobs that run on multiple nodes using MPI)

Submit script for threaded parallel jobs

Threaded (OpenMP) jobs run on multiple cores on a single node - they require shared memory. In your script, increase ncpus count, memory (to accommodate more cores) and add ompthreads parameter (ompthreads is always equal or less than ncpus).

```
#PBS -l walltime=24:00:00
#PBS -l select=1:ncpus=8:mem=16gb
```

number of cores per node

memory per node

Submit script for MPI jobs

MPI jobs don't require shared memory and run on multiple cores on multiple nodes. State the number of nodes using "select". The rest of the params (ncpus, mem and mpiprocs) state the requirement per one node; mpiprocs are always equal (or less than) ncpus.

```
#PBS -l walltime=24:00:00
#PBS -l select=2:ncpus=32:mem=16gb
```

number of cores on one node

number of compute nodes

memory per node

Imperial College London

RCS RESEARCH COMPUTING SERVICE
Gateway to Discovery

Job sizing (as of May 2023)

Queue	Use cases	Nodes (select)	Number of cores (ncpus)	Memory in gb (mem)	Walltime in hrs (walltime)
interactive	Small interactive test jobs and interactive debugging	4	1 - 48	1 - 100	0 - 8
short8	Short running jobs	1	1 - 256	1 - 920	0 - 8
pqjupyter	Queue for JupyterHub jobs	1	1, 2, 4, 8	8, 16, 32, 64	8
pqood	Queue for Open OnDemand (RStudio) jobs	1	1, 2, 4, 8	8, 16, 32, 64	8
throughput72	Low core jobs	1	1- 8	1 - 100	9 - 72
general72	General compute queue	1 - 16	9 - 32	1 - 124	0 - 72
medium72	Single-node jobs	1	9 - 127	1 - 920	9 - 72
large72	Whole node jobs	1	128 - 256	1 - 920	9 - 72
largemem72	Large memory jobs	1	1 - 256	921 - 4000	0 - 72
gpu72	Main queue for GPU jobs	1 - 4	1 - 256	1 - 920	0 - 72
capability48	Multi-node jobs 48h	2 - 8	64 - 128	1 - 920	24 - 48
capability24	Multi-node jobs 24h	2 - 8	64 - 128	1 - 920	0 - 24

Check the latest details at: https://wiki.imperial.ac.uk/display/HPC/New+Job+sizing+guidance

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs

Video

Video (8 min) on managing input and output files:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=427f191d-d5cf-41ac-b2f1-abc7011de699

Current working directory for a job

When a job starts running, **the current working directory changes.** It runs in a temporary directory that is created for it ($TMPDIR).

This is somewhat counter-intuitive and can lead to file management errors.

For example, if you place your input file in the same directory as your job script and expect your program to read it in a current directory, the file will not be found.

The next few slides show different ways of managing this situation.

Hellohpc output

```
-bash-4.2$ more submit.pbs.o1037925
Hello from r5i2n2 on Wed Feb  5 13:57:14 2020.
I'm job 1037925.pbs submitted by kmichali from
/rdsgpfs/general/user/kmichali/home/intro_hpc_jan22
and executed in
/rds/general/ephemeral/tmpdir/pbs.1037925.pbs.

==========================================

        Job resource usage summary

                 Memory (GB)    NCPUs
 Requested  :         1             1
 Used       :         0 (peak)   0.00 (ave)

==========================================
-bash-4.2$
```

Note: The submit and execute directories are different

Job script with input and output files

This will not work because the default current directory for a job is not the same as the directory where the job was submitted (and where you expect to find your input file).

```
#PBS -l walltime=01:00:00
#PBS -l select=1:ncpus=1:mem=1gb

module load myprog

# input and output will not be found
# the current dir changes when the job is running

myprog input output
```

Environment variable pointing to your submit directory

$PBS_O_WORKDIR

- will exist when your job is running
- this is where you submitted your job
- usually a directory where your job script resides
- you can use this variable in your script

Job script for with $PBS_O_WORKDIR

```
#PBS -l walltime=01:00:00
#PBS -l select=1:ncpus=1:mem=1gb

module load myprog


cd $PBS_O_WORKDIR
myprog input output
```

Job script with an output file

```
#PBS -l walltime=00:10:00
#PBS -l select=1:ncpus=1:mem=1gb

module load hellohpc

cd $PBS_O_WORKDIR
hellohpc.py > mylog
```

Introduction to HPC at Imperial

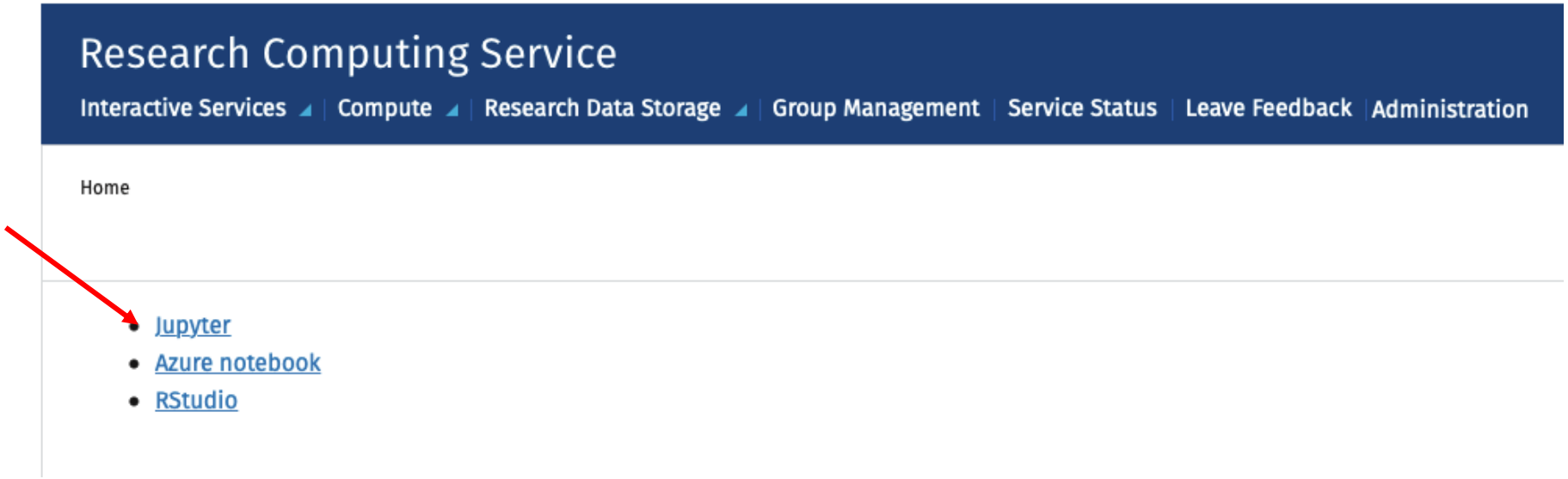
- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism – array jobs
- parallel programs

Interactive resources

- It is possible to use the HPC cluster via:
  - Jupyter Notebooks
  - RStudio

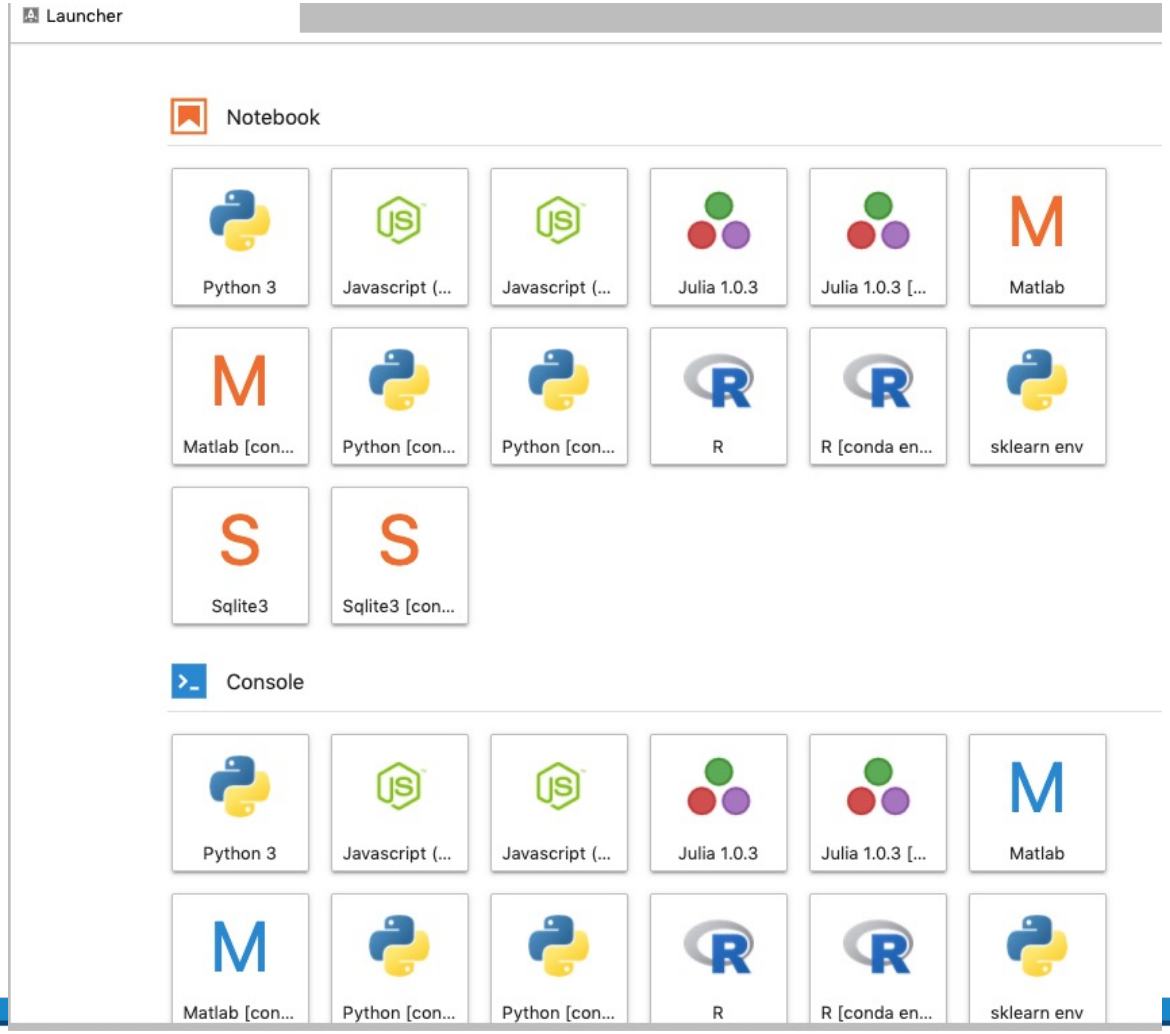
Using the HPC resource via Jupyter Lab

https://selfservice.rcs.imperial.ac.uk/interactive

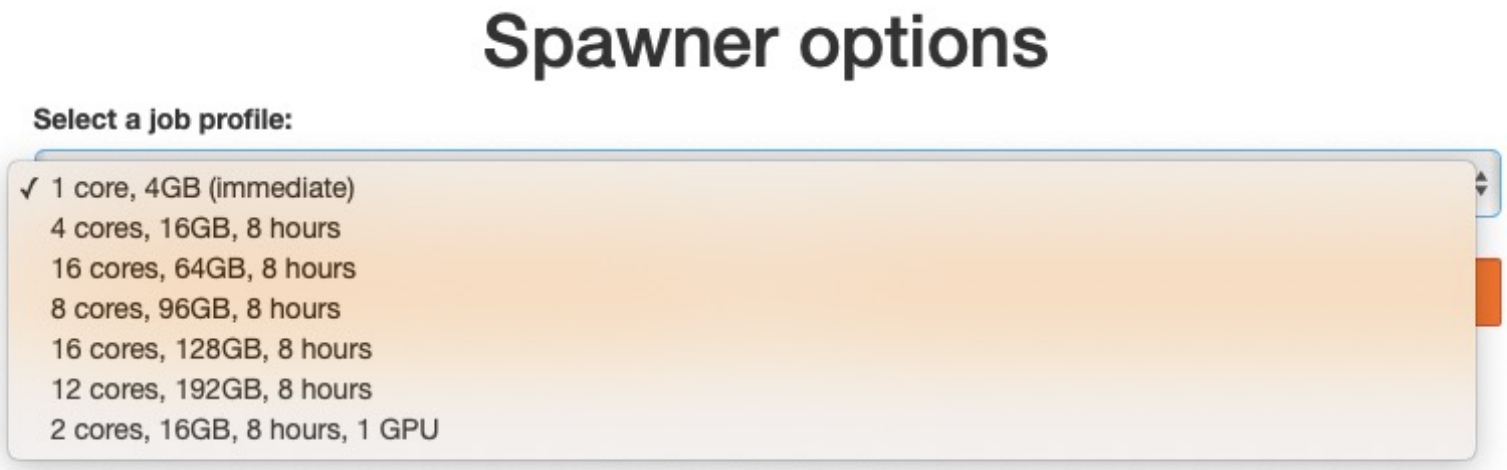


Jupyter Lab

https://jupyter.rcs.ic.ac.uk



Jupyter lab options

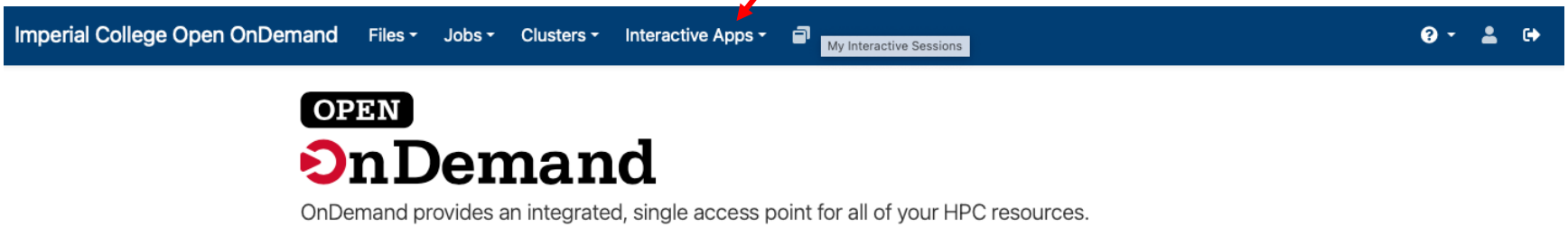


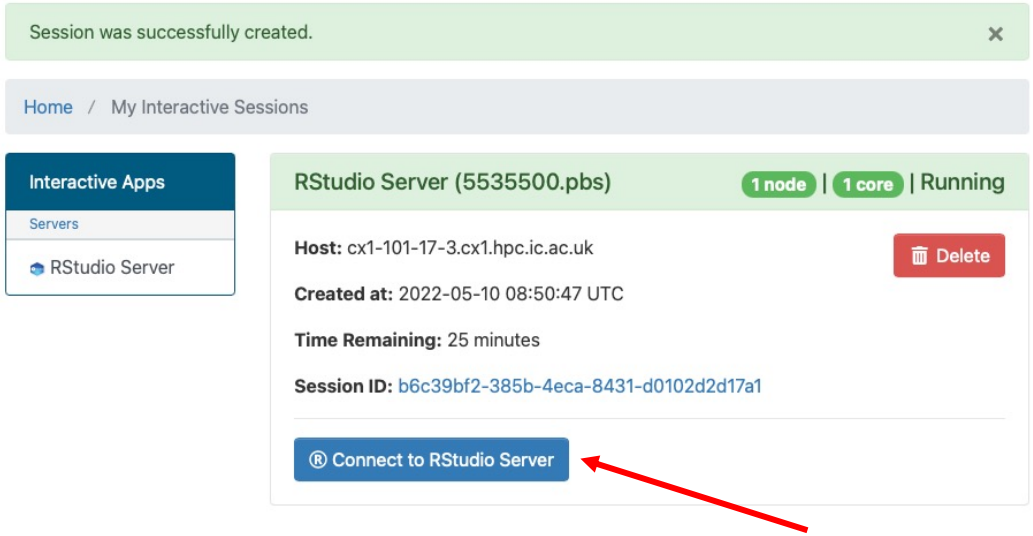
Imperial College London

RCS RESEARCH COMPUTING SERVICE
Gateway to Discovery

RStudio through Open OnDemand

https://openondemand.rcs.ic.ac.uk/





Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism – array jobs
- parallel programs

Video

Video (14 min) on using the cluster to run simple array jobs:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=339125b5-f1ce-42d3-a8c8-abc900f8219a

Parallel computing

Data parallelism (array jobs)

- concurrent processing of data on an HPC cluster

Parallel programs

- parallelism is implemented on code level

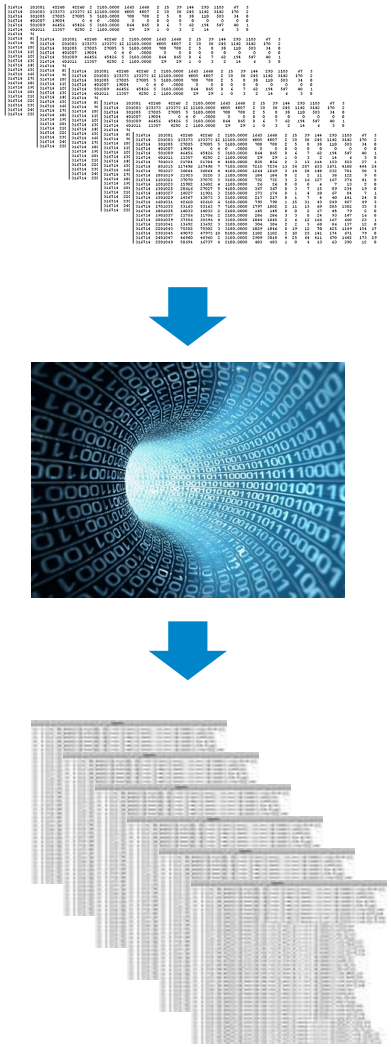
Data parallelism

Data parallelism examples:

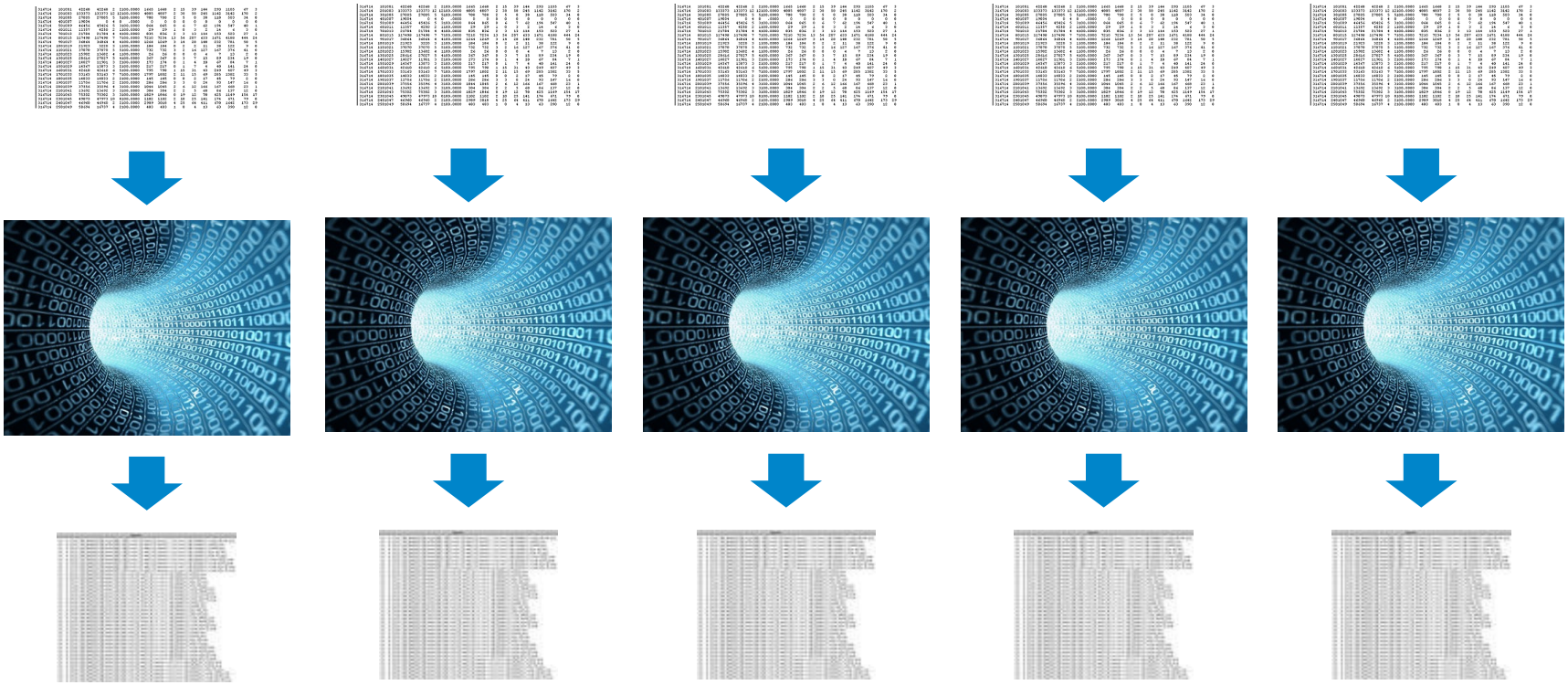
- processing multiple files
- processing large files that can be split
- simulation replicates
- parameter sweeps

Individual cases have to be independent:

- they do not exchange data
- input for any case doesn't depend on output of the other



Data parallelism – array jobs

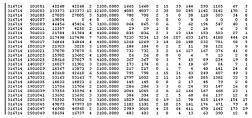


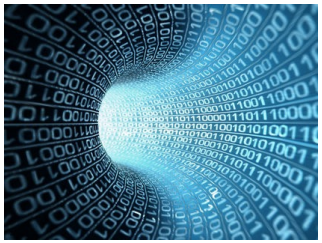
Advantages of data parallelism

- jobs run concurrently (in semi-parallel fashion)
- relatively easy to implement
- jobs run independently when a resource becomes available; no need to wait until a whole compute node is available (program-level parallel jobs would have to)

Single job (before parallelisation)

```
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=1:mem=1gb
```

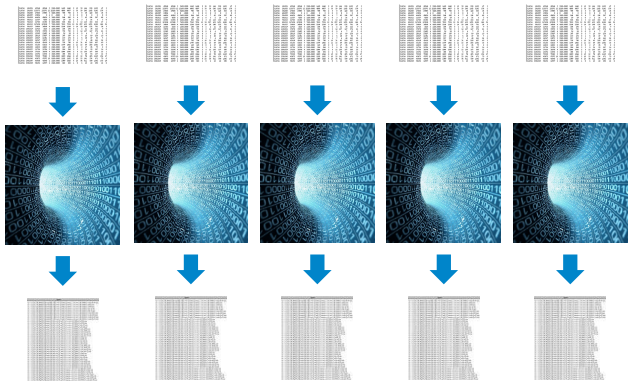




```
module load myprog

myprog input output
```

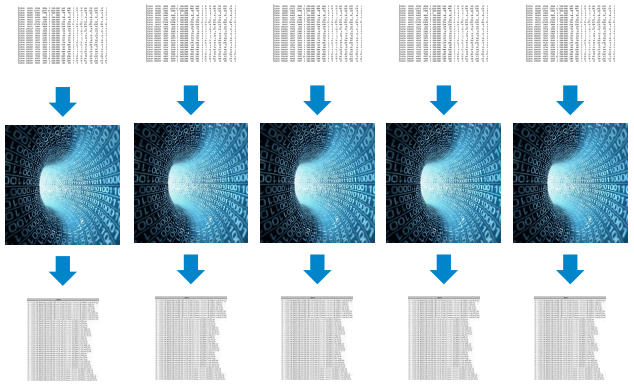
Parallelisation - array job script

```
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -J 1-5

module load myprog

myprog  input? output?
```



Parallelisation - array job script

```
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -J 1-5

module load myprog

myprog  input? output?
```



$PBS_ARRAY_INDEX takes values from 1 to 5 (or any other value specified by –J)

Array job scenario 1 – numbered files

Task: **run filter.py on infile;** BUT the file is too big and takes a long time to process.
Solution: Split infile to smaller chunks and process them concurrently.

```
-bash-4.2$ ll
total 32
-rw-r--r--. 1 kmichali hpc-kmichali 24471 Apr  4 13:56 infile
-bash-4.2$
-bash-4.2$
-bash-4.2$ split -n 3 -a 1 --numeric-suffixes=1 infile infile.
-bash-4.2$
-bash-4.2$
-bash-4.2$ ll
total 32
-rw-r--r--. 1 kmichali hpc-kmichali 24471 Apr  4 13:56 infile
-rw-r--r--. 1 kmichali hpc-kmichali  8157 Apr  4 14:13 infile.1
-rw-r--r--. 1 kmichali hpc-kmichali  8157 Apr  4 14:13 infile.2
-rw-r--r--. 1 kmichali hpc-kmichali  8157 Apr  4 14:13 infile.3
-bash-4.2$
```

Imperial College London

Hands-on – array jobs

Download and run array run case scenario 1:

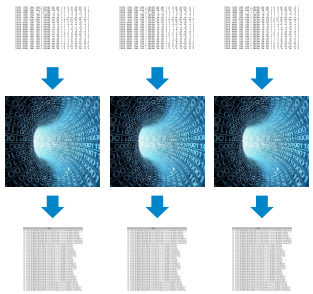
```
git clone https://github.com/kmichali/array_test.git
cd array_test/filter
qsub submit_filter_array.pbs
```

Array job script – numbered files (before parallelisation)

```
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -N class


module load anaconda3/personal
cd $PBS_O_WORKDIR
python filter.py infile outfile
```

Array job script – numbered files

```
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -J 1-3

module load anaconda3/personal
cd $PBS_O_WORKDIR
python filter.py infile.$PBS_ARRAY_INDEX outfile.$PBS_ARRAY_INDEX
```



$PBS_ARRAY_INDEX takes values from 1 to 3
(or any other value specified by –J)

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
  - working remotely and login
  - software management on the cluster
  - copying files to and from the cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism – array jobs 2
- parallel programs

Video

Video (16 min) on using the cluster to run array jobs:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=89740912-67a5-41fb-9750-abc900fd7602

Array job scenario 2 – file list

Task: **run stats.py on many molecules;** there are too many and it takes too long.
Solution: Process the molecules concurrently on the HPC cluster.

```
-bash-4.2$ ll
total 5
-rwxr-xr-x 1 kmichali hpc-kmichali   19 May 28 16:11 clean
-rw-r--r-- 1 kmichali hpc-kmichali 1158 May 28 16:11 cubane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali  622 May 28 16:11 ethane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali   69 May 28 16:11 list
-rw-r--r-- 1 kmichali hpc-kmichali  422 May 28 16:11 methane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali 1828 May 28 16:11 octane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali 1226 May 28 16:11 pentane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali  825 May 28 16:11 propane.pdb
-rwxr-xr-x 1 kmichali hpc-kmichali  176 May 28 16:11 stats.py
-rw-r--r-- 1 kmichali hpc-kmichali  242 May 28 16:11 submit_list.pbs
-bash-4.2$
```

Imperial College London

Hands-on – array jobs

Download and run array run case scenario 2:

```
git clone https://github.com/kmichali/array_test.git
cd array_test/molecules
qsub submit_list.pbs
```

Array job script for one file (before parallelisation)

```
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=1:mem=1gb

module load anaconda3/personal
cd $PBS_O_WORKDIR

INFILE=cubane.pdb

python stats.py $INFILE ${INFILE}.out
```

Leveraging $PBS_ARRAY_INDEX to choose input

```
katerina — ssh kmichali@login.hpc.ic.ac.uk — 89×36
-bash-4.2$ ls -l
total 5
-rwxr-xr-x 1 kmichali hpc-kmichali   19 Mar  6 15:24 clean
-rw-r--r-- 1 kmichali hpc-kmichali 1158 Mar  6 15:24 cubane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali  622 Mar  6 15:24 ethane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali  422 Mar  6 15:24 methane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali 1828 Mar  6 15:24 octane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali 1226 Mar  6 15:24 pentane.pdb
-rw-r--r-- 1 kmichali hpc-kmichali  825 Mar  6 15:24 propane.pdb
-rwxr-xr-x 1 kmichali hpc-kmichali  176 Mar  6 15:24 stats.py
-rw-r--r-- 1 kmichali hpc-kmichali  250 Mar  6 15:36 submit_list.pbs
-bash-4.2$ ls -1 *pdb
cubane.pdb
ethane.pdb
methane.pdb
octane.pdb
pentane.pdb
propane.pdb
-bash-4.2$ ls -1 *pdb | head -n 3
cubane.pdb
ethane.pdb
methane.pdb
-bash-4.2$ ls -1 *pdb | head -n 3 | tail -n 1
methane.pdb
-bash-4.2$ INFILE=$(ls -1 *pdb | head -n 3 | tail -n 1)
-bash-4.2$ echo $INFILE
methane.pdb
-bash-4.2$
```

The screen shows Linux commands that list the input files, pick the n-th file from the list, and store the file name in a variable.

These commands can be used in an array run job script to process a list of files.

The example picks the third file from the list and stores it in a variable called $INFILE.

Array job script – file list

```
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -J 1-6

module load anaconda3/personal
cd $PBS_O_WORKDIR

INFILE=$(ls -1 *pdb | head -n $PBS_ARRAY_INDEX | tail -n 1)

python stats.py $INFILE ${INFILE}.out
```

Array job scenario 3 – for loops and matrices

- if you are running simulation replicates or parameter sweep, your program might contain a very time-consuming for loop.
- If the loop iterations are independent, there is a good chance that you can deploy it as an array run.
- The following Python example uses $PBS_ARRAY_INDEX value directly in the Python code.
- The code iterates through a matrix and uses individual elements for a calculation. In the original code, the calculations happen in serial manner.

Imperial College London

Hands-on – array jobs

Download and run array run case scenario 3:

```
git clone https://github.com/kmichali/array_test.git
cd array_test/matrix
qsub submit_matrix.pbs
```

Array job scenario 3 – for loops

Task: Iterate though a large matrix and use each element for a calculation using an array run.

Solution:

The following Python example uses $PBS_ARRAY_INDEX value directly in the Python code.

The code uses $PBS_ARRAY_INDEX to execute the correct iteration of the loop that processes just one element of the matrix.

All array runs combined process all elements of the matrix in parallel.

Python template for matrix iteration

```
input = np.loadtxt(”matrix.txt", dtype='f’,
delimiter=',’)
counter = 0

for i in range(0,input.shape[0]):
    for j in range(0, input.shape[1]):
        print('processing element', input[i,j])
        # your code for processing goes in here
```

Python template for matrix iteration with array run

```
input = np.loadtxt(”matrix.txt", dtype='f’,
delimiter=',’)
counter = 0
array_index = int(os.environ['PBS_ARRAY_INDEX'])

for i in range(0,input.shape[0]):
    for j in range(0, input.shape[1]):
        counter = counter+1
        if counter == array_index:
            print('processing element', input[i,j])
            # your code goes in here
```

Submit script for 3x3 matrix

```
#PBS -l walltime=00:10:00
#PBS -l select=1:ncpus=1:mem=1gb
#PBS -J 1-9


module load anaconda3/personal

cd $PBS_O_WORKDIR
python matrix_array.py
```

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs on the cluster
  - OpenMP
  - MPI
  - python

Video

Video (12 min) on parallel codes and a code example:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=1fb03429-7347-418e-88ed-abca00b6a4e7

Parallel programs

So far, we have implemented parallelism using array runs on the cluster; the software that was used remained unchanged.

The next section is about parallelism that is achieved by writing parallel code.

Two types of parallel code:

- OpenMP or multiprocessing – requires shared memory
- Message Passing Interface – on multiple compute nodes at once

Parallel codes

- As it has become difficult and expensive to produce faster and faster processors, the focus has shifted in combining the power of multiple CPUs to work on a single task. We supports two main parallel programming paradigms.

- **OpenMP (C,C++,Fortran) and multiprocessing (Python)**

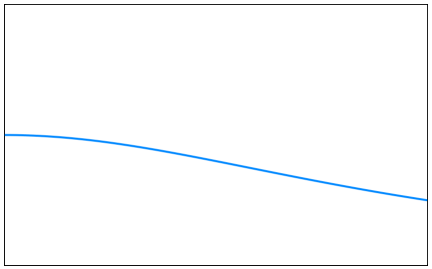
  OpenMP jobs require shared memory. The parallelization is achieved by using compiler directives around individual blocks of code, often loops or functions. Converting your serial program into OpenMP is “relatively” easier than attempting MPI programming.

- **Message Passing Interface (C, C++,Fortran, Python)**

  MPI jobs do not require shared memory and can be executed on separate cluster nodes. MPI standard consists of library routines that create parallel processes and pass necessary data (messages) between them. The messages are passed across the network if needed.

Real problem - approximate Pi with numerical integration

$$\frac{\pi}{4} = \int_0^1 \frac{dx}{1+x^2} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{1}{1+\left(\frac{i-\frac{1}{2}}{N}\right)^2}$$

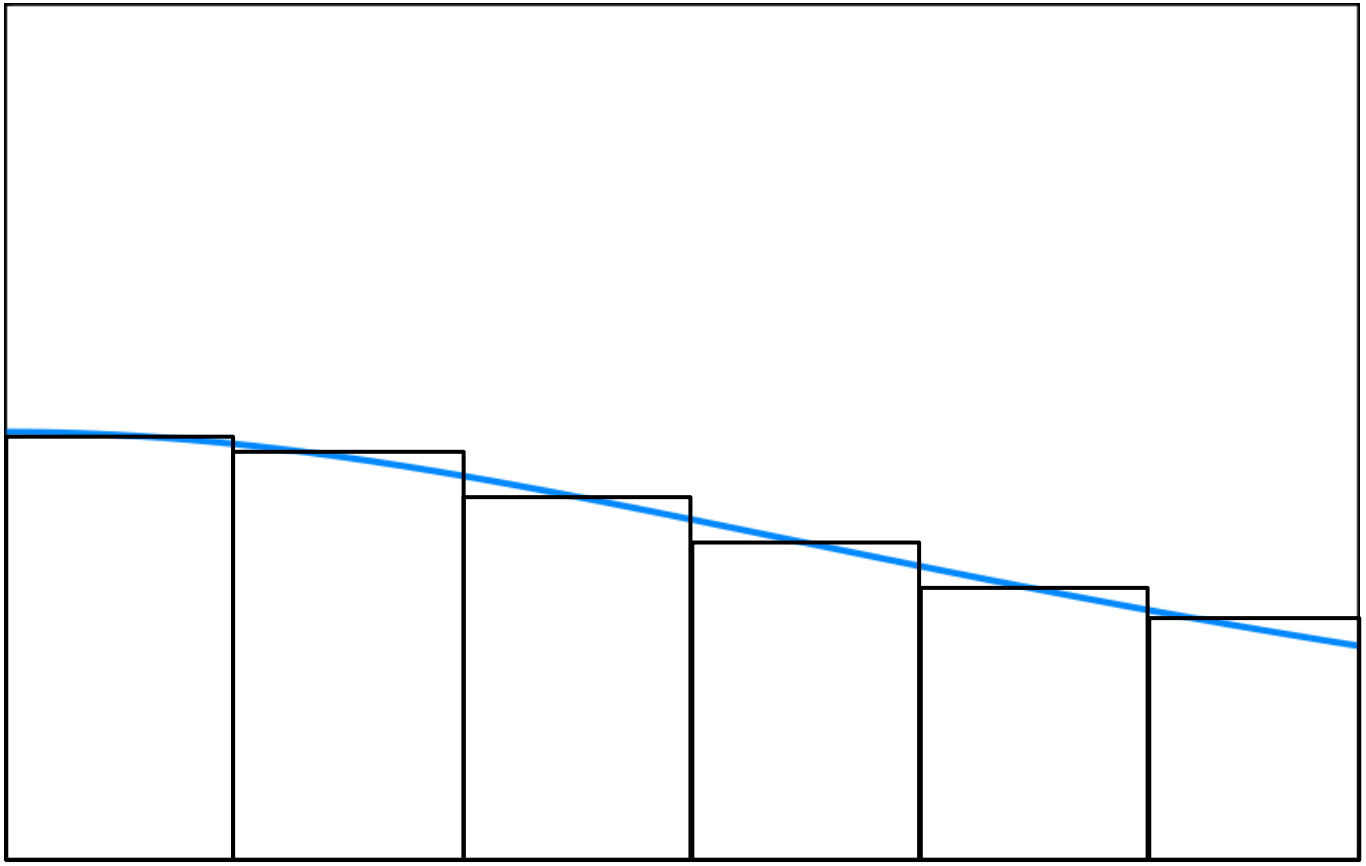


The answer becomes more accurate with increasing N

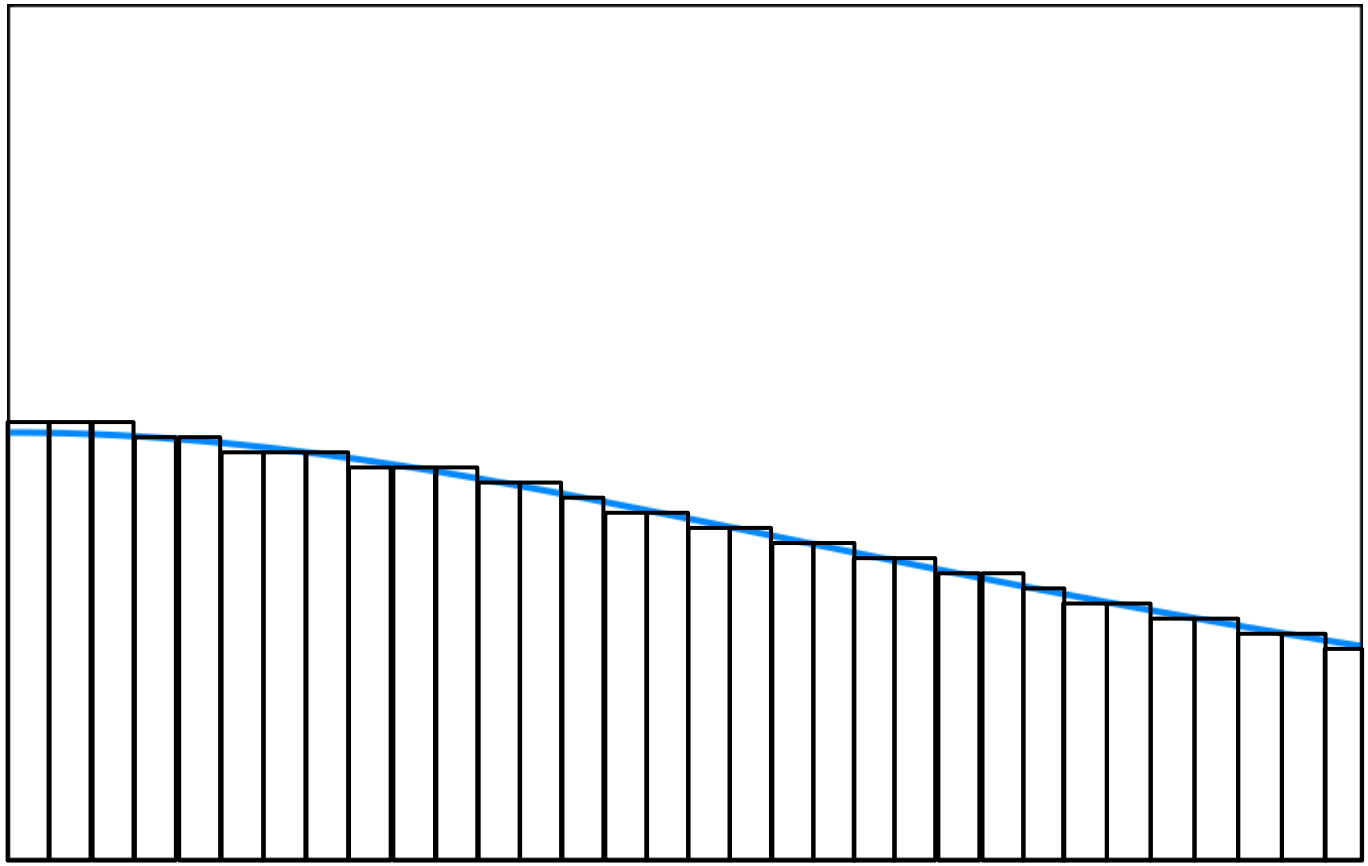
Iterations over N are independent, the calculation can be parallelised

We'll discuss pi code examples using serial, OpenMP, MPI and various python approaches.

Numerical integration



Numerical integration



Imperial College London

Serial Pi

$$\frac{\pi}{4} = \int_0^1 \frac{dx}{1+x^2} \approx \frac{1}{N}\sum_{i=1}^{N} \frac{1}{1+\left(\frac{i-\frac{1}{2}}{N}\right)^2}$$

```
program pi_serial

implicit none

integer(KIND=8)  :: n, i

double precision :: w, x, sum, pi, a, start, finish

double precision :: duration, timef

n=1E10

start=timef()

w=1.0/n

sum=0.0

do i=1,n

   x=w*(i-0.5)

   sum=sum+4.0/(1.0+x*x)

end do

pi=w*sum

finish=timef()

duration=finish-start

print*,n,"  ",pi,"  ",duration

end
```

Imperial College London

Hands-on – get codes, compile and run

```
#clone repository
git clone https://github.com/kmichali/parallel_ex.git

# go to repository
cd parallel_ex

# load compiler and parallel library
module load intel-suite mpi

# compile all codes
make

#run serial pi code
qsub pi_serial.pbs
```

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs on the cluster
  - OpenMP
  - MPI
  - python

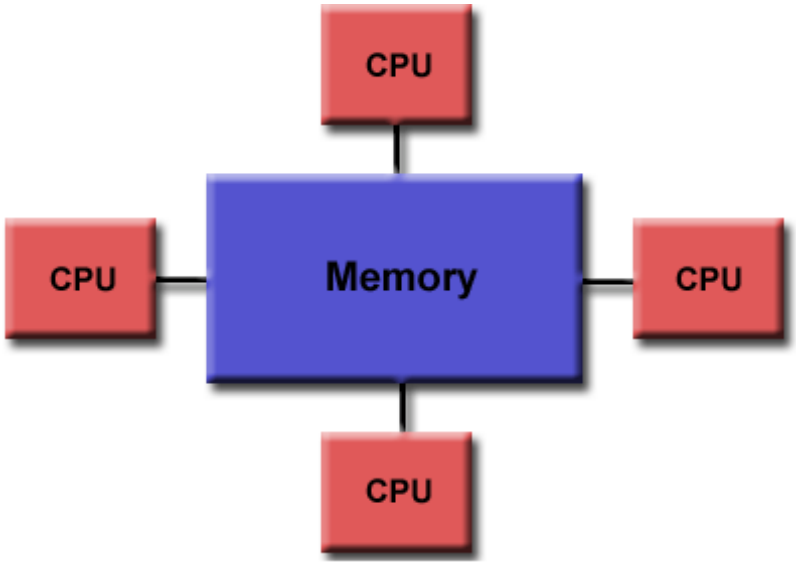
Video

Video (10 min) on OpenMP jobs on the cluster:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=9b7407fc-9f94-48fe-bb4b-abca00be7c3b

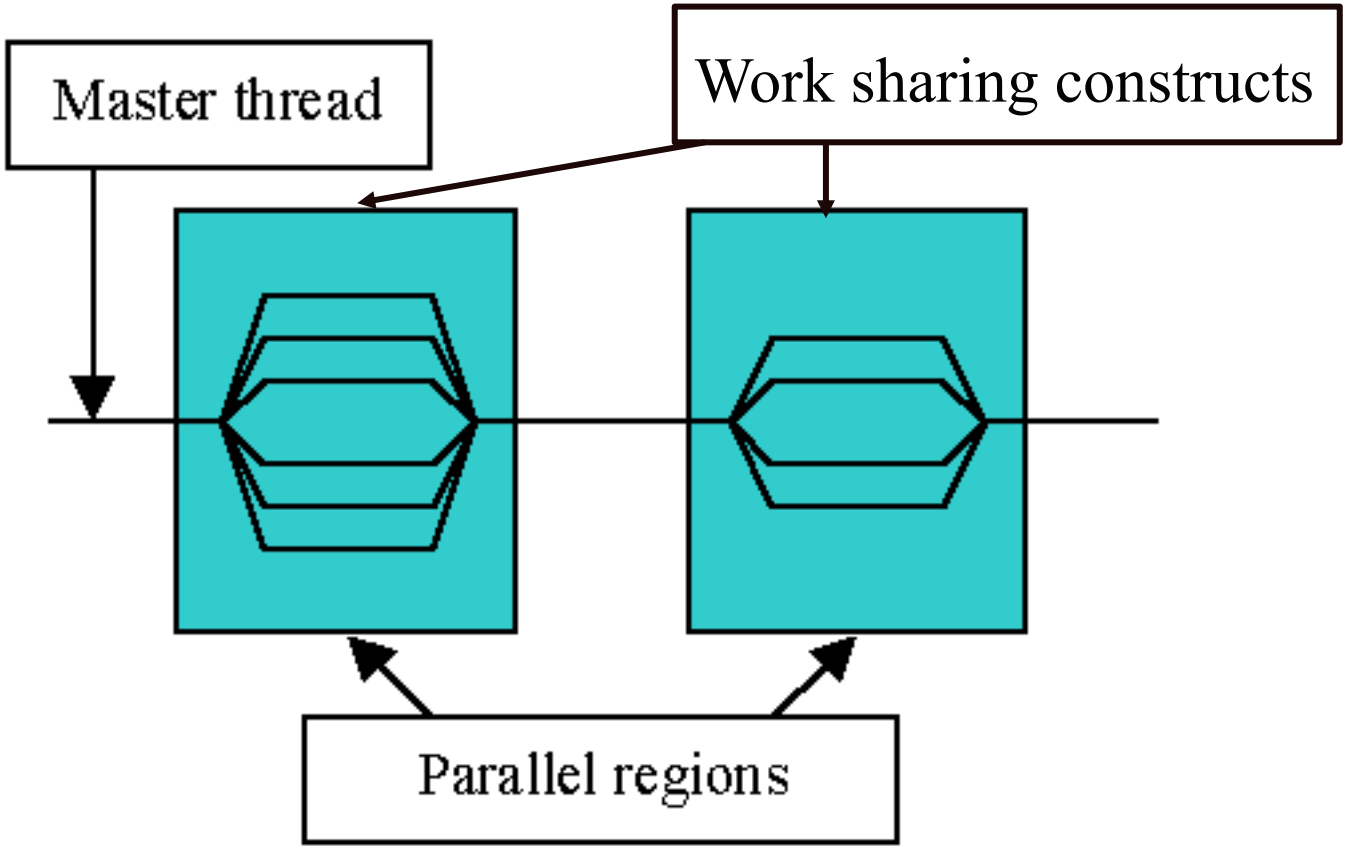
Parallel programming paradigms – OpenMP

OpenMP (open multiprocessing)

- set of extensions (compiler pragmas and API calls) to provide Fortran/C/C++ with the ability to run certain parts of the code in parallel, without explicitly managing (creating, destroying, assigning) threads
- requires shared memory



OpenMP



Hello world in C and OpenMP

```
#include <omp.h>
#include <stdio.h>
main (int argc, char *argv[])
{

  #pragma omp parallel
   {
       printf("Hello world!");
   }
}
```

OpenMP submit script

```
#PBS -l walltime=01:00:00
## Use 1 compute node with 8 cores and 4gb of memory
#PBS -l select=1:ncpus=8:mem=4gb:ompthreads=8

module load intel-suite

$PBS_O_WORKDIR/hello_openmp
```

Imperial College London

Hands-on – calculate Pi with serial and OpenMP code

```
#clone repository and compile code
git clone https://github.com/kmichali/parallel_ex.git
cd parallel_ex
module load intel-suite mpi
make

# have a look at pi_openmp.pbs and pi_openmp.f90
# run pi_openmp with and examine the timing
qsub pi_openmp.pbs
```

3.14159265358979323846264 3

Introduction to HPC at Imperial

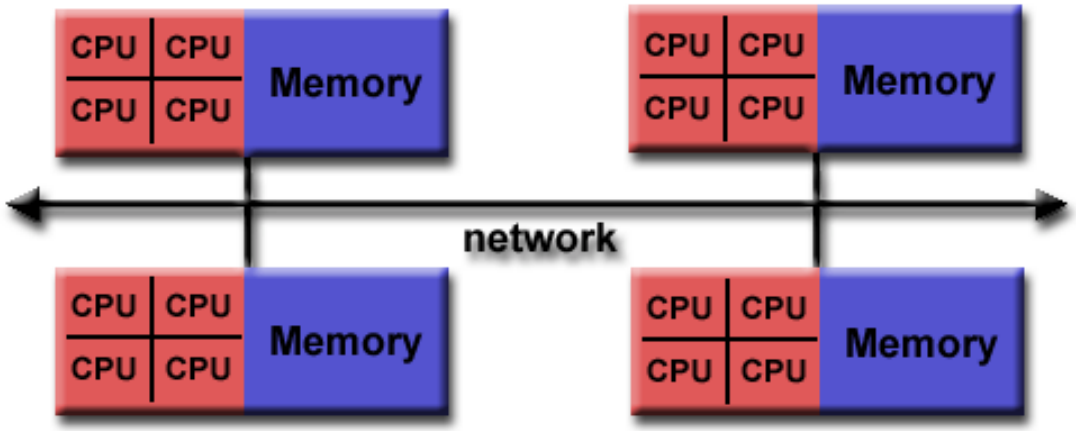
- introduction, key links and contacts
- preparing to use the HPC cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs on the cluster
  - OpenMP
  - MPI
  - python

Video

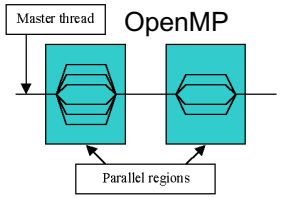
Video (11 min) on MPI jobs on the cluster:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=85ef9410-cf0d-440f-ac23-abca00c25bac

Message Passing Interface - MPI

- message passing parallel programming model
- one program is deployed on multiple nodes with distributed memory
- messages (data) are sent over the network
- MPI is a specification for a library with different implementations (Open MPI, Intel, MVAPICH)



MPI program



Hello world in C and MPI

```
int main(int argc,char **argv) {
  MPI_Comm comm;
  int nprocs,procid;

  MPI_Init(&argc,&argv);
  comm = MPI_COMM_WORLD;
  MPI_Comm_size(comm,&nprocs);
  MPI_Comm_rank(comm,&procid);

  int name_length = 100;
  char proc_name[name_length];
  MPI_Get_processor_name(proc_name,&name_length);


  printf("Hello from process %d out of %d running on %s\n",
       procid, nprocs, proc_name);


  MPI_Finalize();
}
```

MPI job submit script

```
#PBS -l walltime=01:00:00
## Use 2 nodes with 24 cores each and 4 gb of memory per node.
#PBS -l select=2:ncpus=24:mem=4gb:mpiprocs=24

module load intel-suite mpi

mpiexec $PBS_O_WORKDIR/hello_mpi
```

Imperial College London

Hands-on 4 – calculate Pi with MPI code

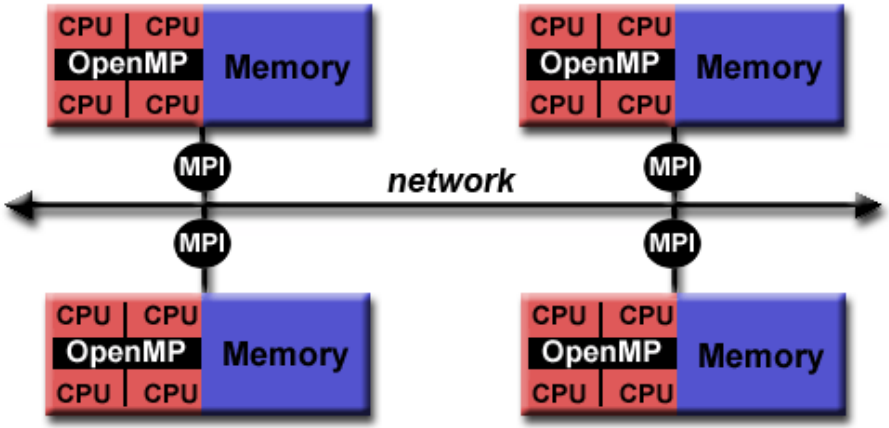
```
#clone repository
git clone https://github.com/kmichali/parallel_ex.git
cd parallel_ex
module load intel-suite mpi
make

# have a look at hello_mpi.c and hello_mpi.pbs
# submit and look at the log files
qsub hello_mpi.pbs

# have a look at pi_mpi.pbs and pi_mpi.f90
# run pi_mpi and examine the log files
qsub pi_mpi.pbs
```

Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs on the cluster
  - OpenMP
  - MPI
  - python

Hybrid jobs

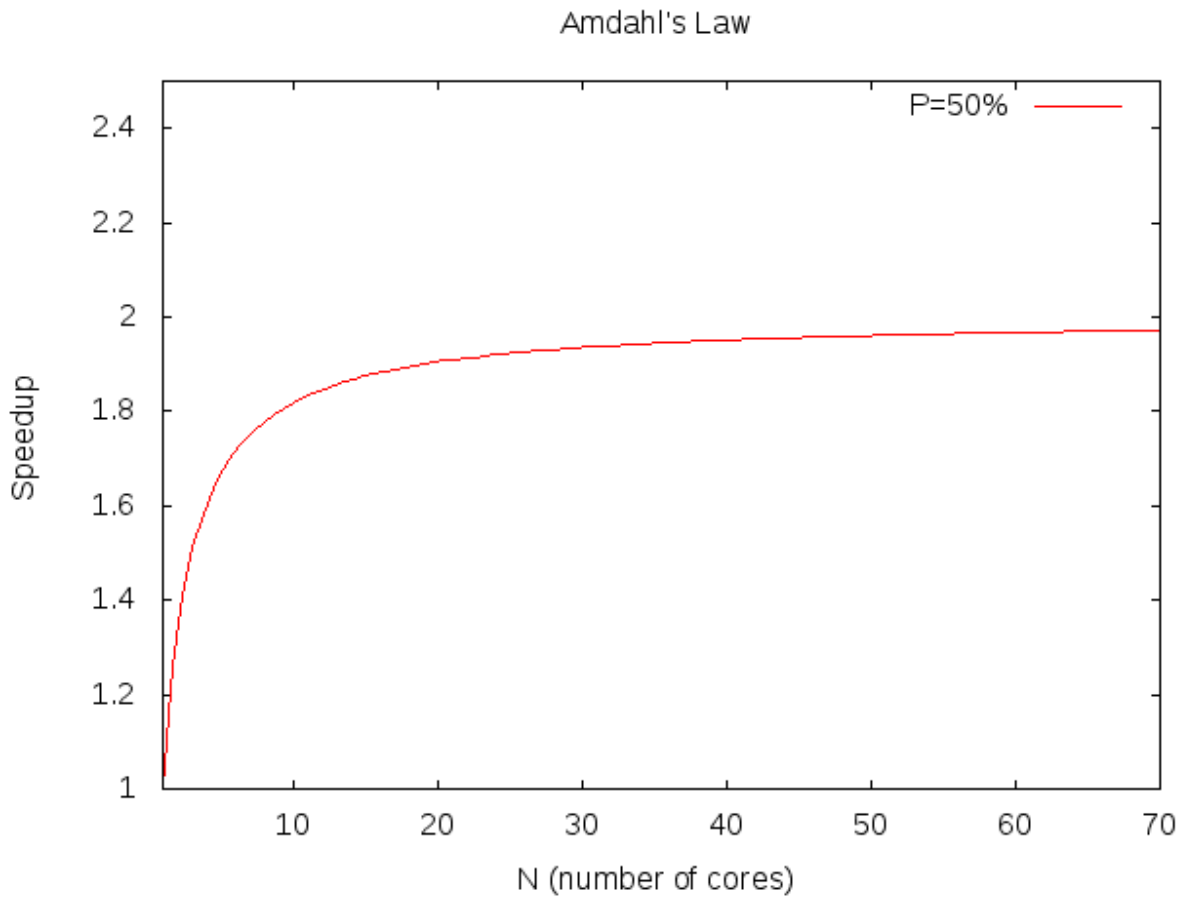
- hybrid code combines MPI with OpenMP
- MPI provides distributed memory parallelism
- OpenMP provides on-node shared memory parallelism
- the model reduces data movement within a node
- way to combine vectorisation and large scale parallel code



Hybrid submit script

```
#!/bin/sh
#PBS -l walltime=01:00:00
#PBS -l select=2:ncpus=24:mem=4gb:mpiprocs=1:ompthreads=24
## Use 2 nodes with 24 cores each and 4 gb of memory per node.
## Each node will host 1 mpirank and 24 threads.

module load intel-suite mpi

mpiexec $PBS_O_WORKDIR/myprog
```

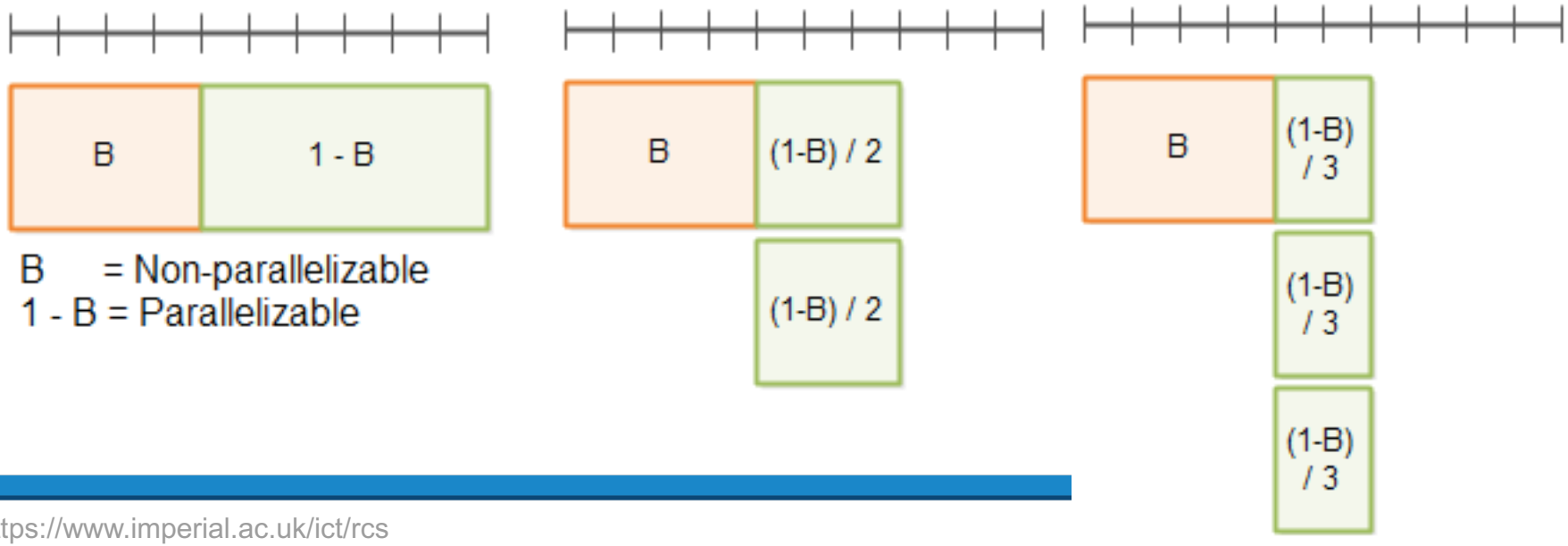
Amdahl's law


Amdahl's Law
P=50%
Speedup
2.4
2.2
2
1.8
1.6
1.4
1.2
1
10
20
30
40
50
60
70
N (number of cores)

Amdahl's law

Amdahl's law states that in parallelization, if 1-B is the proportion of a system or program that can be made parallel, and B is the proportion that remains serial, then the maximum speedup that can be achieved using N number of processors is 1/(B+((1-B)/N).

If N tends to infinity then the maximum speedup tends to 1/B.



Introduction to HPC at Imperial

- introduction, key links and contacts
- preparing to use the HPC cluster
- software execution on the cluster
- job parameters and job scripts
- managing input and output files in your script
- interactive resources
- data parallelism
- parallel programs on the cluster
  - OpenMP
  - MPI
  - python

Video

Video (14 min) on parallel python codes:
https://imperial.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=d281164a-84a6-4b9b-9ac1-abca00d5bec3

Parallel python?

- It is possible but not straightforward.
- The usual suspects – multiprocessing and MPI – were not the best way to go.

Pi code with python

- serial with python interpreter
- multiprocessing with Pool
- python MPI with mpi4py
- serial with pypy3 interpreter
- serial with just in time compiling (numba and jit)
- parallel with numba and jit

Imperial College London

Hands-on – get codes


```
#clone repository
git clone https://github.com/kmichali/parallel_ex.git
cd parallel_ex/python

# if you want to run the codes yourself, set up
anaconda environments for mpi4py (name it mpi), numba
(numba) and pypy (pypy); see link below

#submit all with
bash submit_all.sh
```

#installation help:
https://numba.pydata.org/numba-doc/latest/user/installing.html
https://anaconda.org/conda-forge/pypy
https://anaconda.org/anaconda/mpi4py

Serial pi in Python

```
import time

n = int(10e9)
start_time = time.time()
w = 1.0/n
psum = 0.0

for i in range(1,n+1):
    x = w*(i - 0.5)
    psum = psum+4.0/(1.0 + x*x)

pi=w*psum
duration = time.time() - start_time
print(f"{n:,d}", "   ", pi, "   ", duration)
```

Multiprocessing with Pool

```
import time
import multiprocessing as mp
import functools
import operator

def calc_sum(i):
    psum = 4.0/(1.0 + w*w*(i - 0.5)*(i - 0.5))
    return psum


n = int(10e9)
w = 1.0/n
start_time = time.time()

pool = mp.Pool(24)
result = pool.map(calc_sum, range(1,n+1))
total = functools.reduce(operator.add, result)
pi=w*total
print(f"{n:,d}", "   ", pi, "   ", time.time() - start_time)
```

MPI with mpi4py

```
import numpy as np
from mpi4py import MPI

n = int(10e9)

comm = MPI.COMM_WORLD
myrank = comm.Get_rank()
nproc = comm.Get_size()

if myrank == 0:
    start_time = time.time()

start = 1+myrank * int(n/nproc)
finish = (myrank+1) * int(n/nproc)

psum = 0.0
pi = 0.0
w = 1.0/n
```

```
for i in range(start, finish + 1):
    x = w*(i - 0.5)
    psum = psum+4.0/(1.0 + x*x)
```

```
psum = np.asarray(psum)
pi = np.asarray(pi)
comm.Reduce(psum, pi, op=MPI.SUM)
pi=w*pi

if myrank == 0:
    duration = time.time() - start_time
    print(n, "   ", pi, "   ", duration)
```

Numba and jit implemenation

```
import time
from numba import jit

@jit(nopython=True)
def calc_sum(n,w):
    psum = 0.0

    for i in range(1,n+1):
        psum += 4.0/(1.0 + w*w*(i - 0.5)*(i - 0.5))
    return psum


n = int(10e9)
start_time = time.time()
w = 1.0/n
sum = calc_sum(n,w)
pi=w*sum
print(f"{n:,d}", "   ", pi, "   ", time.time() - start_time
```

Parallel Numba

```
import time
from numba import jit, prange

@jit(nopython=True, parallel=True)
def calc_sum(n,w):
    psum = 0.0

    for i in prange(1,n+1):
        psum += 4.0/(1.0 + w*w*(i - 0.5)*(i - 0.5))
    return psum


n = int(10e9)
start_time = time.time()
w = 1.0/n
sum = calc_sum(n,w)
pi=w*sum
print(f"{n:,d}", "   ", pi, "   ", time.time() - start_time)
```

Serial and parallel python – pi with 10 billion iterations

- serial Fortran at 16s and 24 core OpenMP at 1.3s

- serial with python interpreter – 40-50 min
- python MPI with mpi4py – 61s (32 cores)
- python multiprocessing with Pool and Queue – does not scale
- modified multiprocessing with Pool – 117s (8 cores)

- serial with pypy3 interpreter – ~2 min
- serial python with just-in-time compiling – numba and jit – ~18s
- python with parallel numba and jit – ~4s (24 cores)

Thank you!