Lesson 3.13

Following the Progression of a Job (SLURM)

Learning Objectives

Students will:

- Learn how to check the status of a job on a cluster
- Understand what is the checkpointing of a job
- Discuss how the allocation of different number resources affect computation time using data that they collect from the execution of a provided piece of code

squeue: Checking the Status of a Job

squeue - Displays all jobs currently in the queue

```
adev0: squeue
JOBID PARTITION  NAME  USER ST  TIME NODES NODELIST(REASON)
65646     batch  chem  mike  R 24:19     2 adev[7-8]
65647     batch   bio  joan  R  0:09     1 adev14
65648     batch  math  phil PD  0:00     6 (Resources)
```

By default it will report (in priority order) the jobs that are

- Currently running
- Pending to run

squeue: What the Output Means

```
adev0: squeue
JOBID PARTITION  NAME  USER ST  TIME NODES NODELIST(REASON)
65646     batch  chem  mike  R 24:19     2 adev[7-8]
65647     batch   bio  joan  R  0:09     1 adev14
65648     batch  math  phil PD  0:00     6 (Resources)
```

ST: Job state where “R” means running and “PD” means pending

TIME: How long the job has been running in days-hours:minutes:seconds

NODELIST(REASON): Where the job is running or the reason it is still pending

squeue: Customizing the Output

Some common squeue flags include:

-l: Reports more of the available information

-u USER: Will only load jobs submitted by user USER

--start: Reports the expected start times of pending jobs

See the manual for all available options

scontrol show job JOBNUMBER

For even more details about jobs you can use

scontrol show job JOBNUMBER

Where JOBNUMBER is optional

```
adev0: scontrol show job
JobId=65672 UserId=phil(5136) GroupId=phil(5136)
   Name=math
   Priority=4294901603 Partition=batch BatchFlag=1
   AllocNode:Sid=adev0:16726 TimeLimit=00:10:00 ExitCode=0:0
   StartTime=06/02-15:27:11 EndTime=06/02-15:37:11
   JobState=PENDING NodeList=(null) NodeListIndices=
   NumCPUs=24 ReqNodes=1 ReqS:C:T=1-65535:1-65535:1-65535
   OverSubscribe=1 Contiguous=0 CPUs/task=0 Licenses=(null)
   MinCPUs=1 MinSockets=1 MinCores=1 MinThreads=1
   MinMemory=0 MinTmpDisk=0 Features=(null)
   Dependency=(null) Account=(null) Requeue=1
   Reason=None Network=(null)
   ReqNodeList=(null) ReqNodeListIndices=
   ExcNodeList=(null) ExcNodeListIndices=
   SubmitTime=06/02-15:27:11 SuspendTime=None PreSusTime=0
   Command=/home/phil/math
   WorkDir=/home/phil
```

Checkpointing and Cancelling Jobs

Checkpointing

It is good practice for long calculations to save their state periodically so that if/when the job is terminated the calculation can be restarted a the last save point

Cancelling Jobs

If a job no longer needs to run, it can be cancelled by typing

scancel JOBNUMBER

SLURM Output File

SLURM Outputs information to an output file that the user can specify

In the example on the left the output file is named “my.stdout”

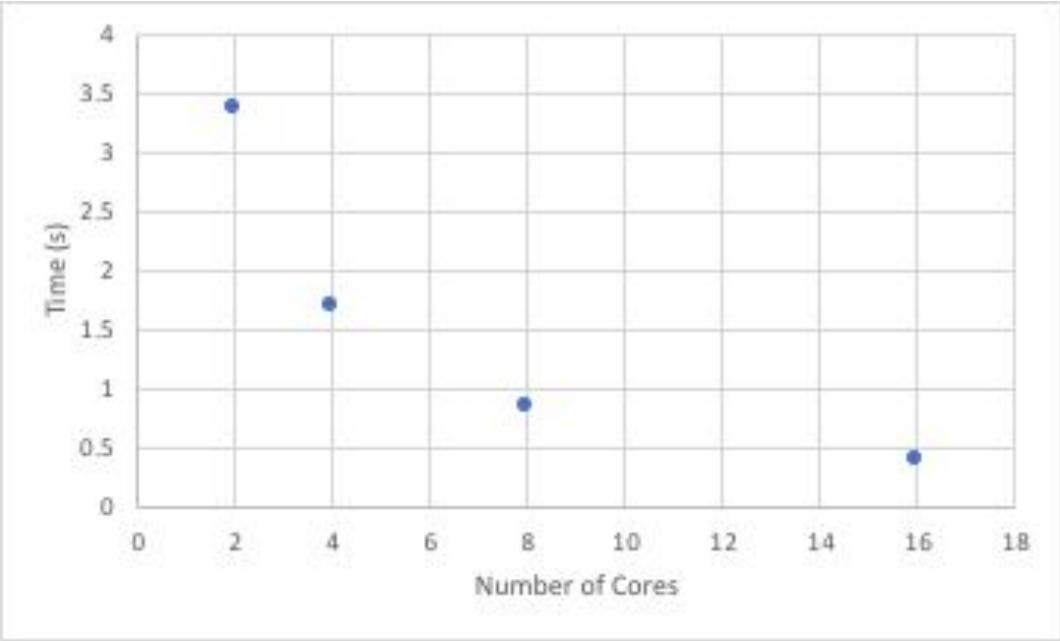
In it is the screen printout of the two commands in my.script

```
adev0: cat my.script
#!/bin/sh
#SBATCH --time=1
/bin/hostname
srun -l /bin/hostname
srun -l /bin/pwd

adev0: sbatch -n4 -w "adev[9-10]" -o my.stdout my.script
sbatch: Submitted batch job 469

adev0: cat my.stdout
adev9
0: adev9
1: adev9
2: adev10
3: adev10
0: /home/jette
1: /home/jette
2: /home/jette
3: /home/jette
```

Testing Effect of Resource Allocation

When parallelizing processes, there tends to be a limit on the speed-up when more processors are added.

We will explore this effect by numerically calculating π



Testing Effect of Resource Allocation: Calculating π

π can be calculated exactly using the following relationship:

$$\pi = 4 \arctan(1)$$

This expression above can be determined using polynomials by evaluating the following integral between 0 and 1.

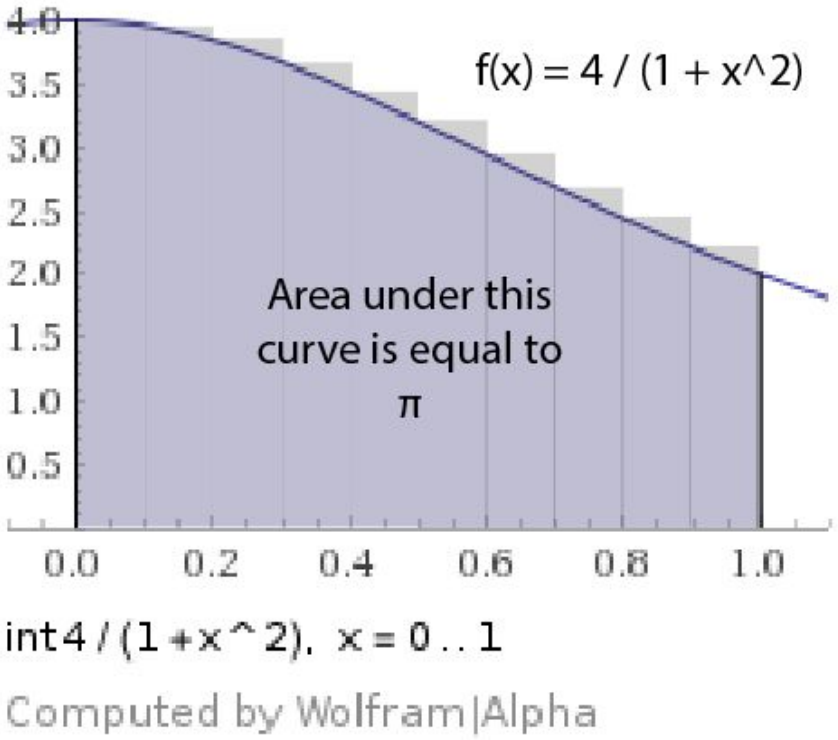
$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

Breaking the problem down

This polynomial based integral can be numerically solved by adding up the area of rectangles of height $f(x_i)$ and width Δx:

$$\pi = \sum_0^n f(x_i)\Delta x$$

This can be distributed between cores since they are independent operations.

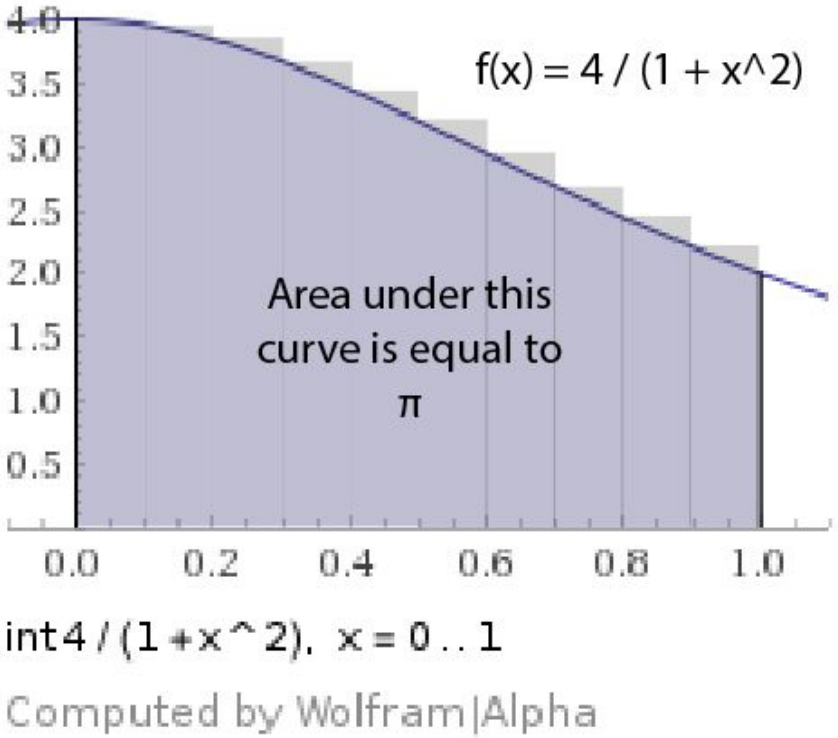


Activity Setup

In reality, to get an accurate representation of π, Δx must be very small.

$$\pi = \lim_{n\to\infty} \sum_0^n f(x_i)\frac{1}{n}$$

In the activity, you will plot how long this numerical integration takes when n is very large as a function of number of cores to illustrate how parallelization has speed-up limits.

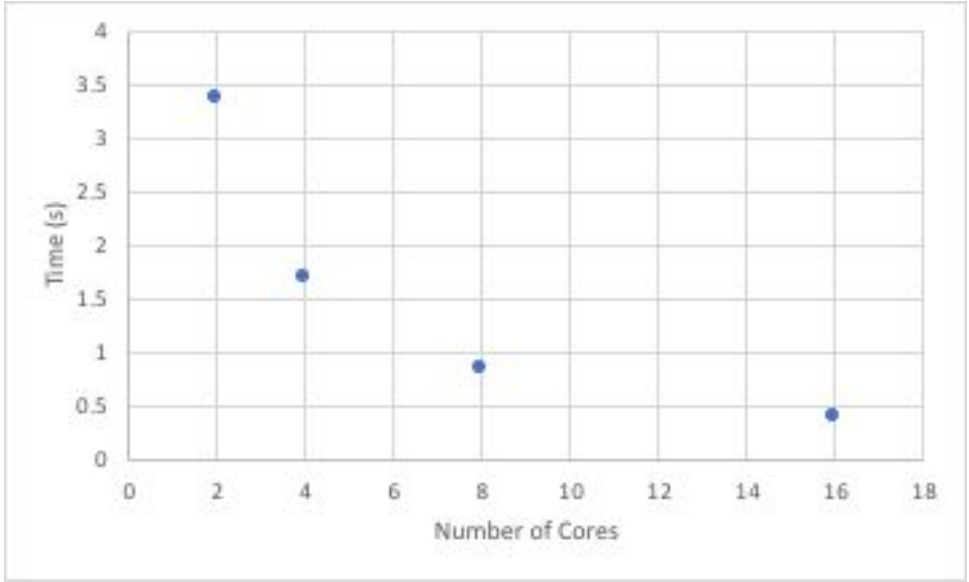


Summary

To check the status on a job type ‘squeue’

It is good practice for long calculations to save their state periodically so that if/when the job is terminated the calculation can be restarted a the last save point

There is a limit to the speed-up that parallelization of code can achieve



Lesson 3.13

Following the Progression of a Job (Torque/Maui)

Learning Objectives

Students will:

- Learn how to check the status of a job on a cluster
- Understand what is the checkpointing of a job
- Discuss how the allocation of different number resources affect computation time using data that they collect from the execution of a provided piece of code

qstat: Checking the Status of a Job

qstat - Displays all jobs currently in the queue

```
pcuser@h2ologin4:~/MPI_Calculate_Pi> qstat

bwsched.ncsa.illinois.edu: Blue_Waters

                                                                  Req'd       Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK  Memory      Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ --------- --------- - ---------
11328977.bw             chem        normal   Ag_graphene_00_i   5865     2     32        --  50:00:00 R  40:56:05
11328978.bw             math        normal   Ag_graphene_09_i  30155     2     32        --  50:00:00 R  40:56:05
11328979.bw             phys        normal   CoMeOTPP_ionic_r  28193     2     32        --  100:00:00 R  40:56:04
11331276.bw             bio         normal   CuI_BSE_CG         5141     1     32        --  25:00:00 R  00:45:27
```

By default it will report (in priority order) the jobs that are

- Currently running
- Pending to run

qstat: What the Output Means

```
pcuser@h2ologin4:~/MPI_Calculate_Pi> qstat

bwsched.ncsa.illinois.edu: Blue_Waters

                                                                  Req'd       Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK  Memory      Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ --------- --------- - ---------
11328977.bw             chem        normal   Ag_graphene_00_i   5865     2     32        --   50:00:00 R  40:56:05
11328978.bw             math        normal   Ag_graphene_09_i  30155     2     32        --   50:00:00 R  40:56:05
11328979.bw             phys        normal   CoMeOTPP_ionic_r  28193     2     32        --  100:00:00 R  40:56:04
11331276.bw             bio         normal   CuI_BSE_CG         5141     1     32        --   25:00:00 R  00:45:27
```

Job ID: System assigned identifier for the job

Queue: Resources/Priority assigned to the job

Jobname: Text identifier the user assigned to the job

S: Job state where “R” means running and “Q” means queued

Elap Time: How long the job has been running in hours:minutes:seconds

qstat: Customizing the Output

Some common squeue flags include:

-q: Reports on the available queues

-u USER: Will only load jobs submitted by user USER

-f JOB ID: Reports detailed information on the job assigned JOB ID

See the manual for all available options

qstat -f JOBID

For even more details about jobs you can use

qstat -f JOBID

Where JOBID is the JOB ID

the scheduler assigned to a job

```
adev0: scontrol show job
JobId=65672 UserId=phil(5136) GroupId=phil(5136)
   Name=math
   Priority=4294901603 Partition=batch BatchFlag=1
   AllocNode:Sid=adev0:16726 TimeLimit=00:10:00 ExitCode=0:0
   StartTime=06/02-15:27:11 EndTime=06/02-15:37:11
   JobState=PENDING NodeList=(null) NodeListIndices=
   NumCPUs=24 ReqNodes=1 ReqS:C:T=1-65535:1-65535:1-65535
   OverSubscribe=1 Contiguous=0 CPUs/task=0 Licenses=(null)
   MinCPUs=1 MinSockets=1 MinCores=1 MinThreads=1
   MinMemory=0 MinTmpDisk=0 Features=(null)
   Dependency=(null) Account=(null) Requeue=1
   Reason=None Network=(null)
   ReqNodeList=(null) ReqNodeListIndices=
   ExcNodeList=(null) ExcNodeListIndices=
   SubmitTime=06/02-15:27:11 SuspendTime=None PreSusTime=0
   Command=/home/phil/math
   WorkDir=/home/phil
```

Checkpointing and Cancelling Jobs

Checkpointing

It is good practice for long calculations to save their state periodically so that if/when the job is terminated the calculation can be restarted a the last save point

Cancelling Jobs

If a job no longer needs to run, it can be cancelled by typing

qdel JOBID

Torque/Maui Output and Error Files

The scheduler outputs information to an output file that the user can specify

In the example to the right, the output file used the form JOBID.out.

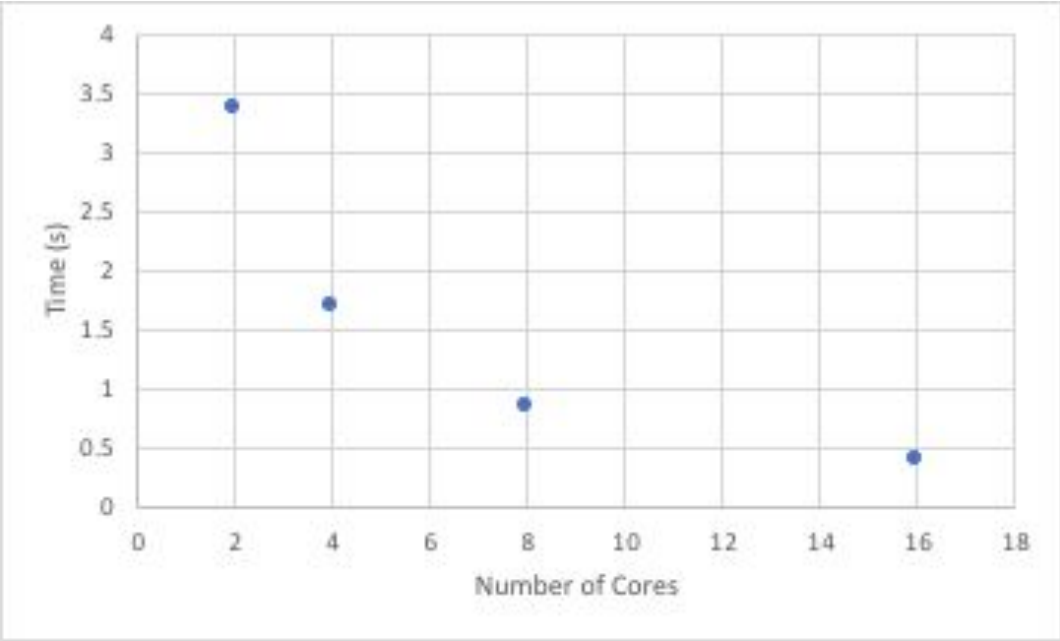
Error messages are also outputted to user defined error files

```
pcuser@h2ologin3:~/MPI_Calculate_Pi> head 11331309.bw.out
----------------------------------------
Begin Torque Prologue on nid27641
at Thu Jul  2 17:41:55 CDT 2020
Job Id:                 11331309.bw
Username:               pcuser
Group:                  EOT_bbct
Job name:               MPI_PI_Test
Requested resources:    nodes=1:ppn=8:xe,walltime=00:05:00,neednodes=1:ppn=8:xe
Queue:                  debug
Account:                bbct
```

Testing Effect of Resource Allocation

When parallelizing processes, there tends to be a limit on the speed-up when more processors are added.

We will explore this effect by numerically calculating π



Testing Effect of Resource Allocation: Calculating π

π can be calculated exactly using the following relationship:

$$\pi = 4 \arctan(1)$$

This expression above can be determined using polynomials by evaluating the following integral between 0 and 1.

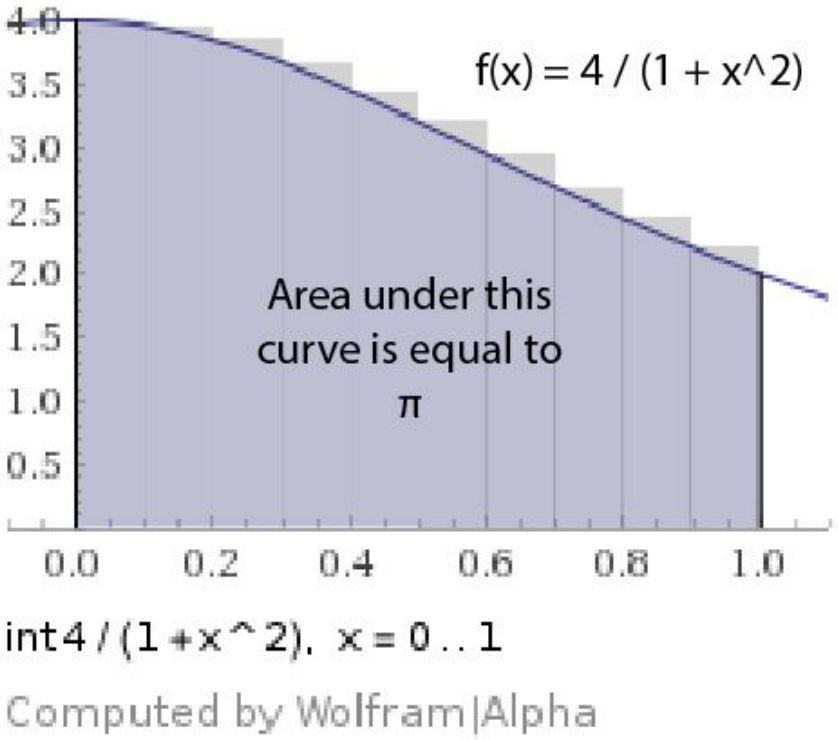
$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

Breaking the problem down

This polynomial based integral can be numerically solved by adding up the area of rectangles of height $f(x_i)$ and width Δx:

$$\pi = \sum_0^n f(x_i)\Delta x$$

This can be distributed between cores since they are independent operations.

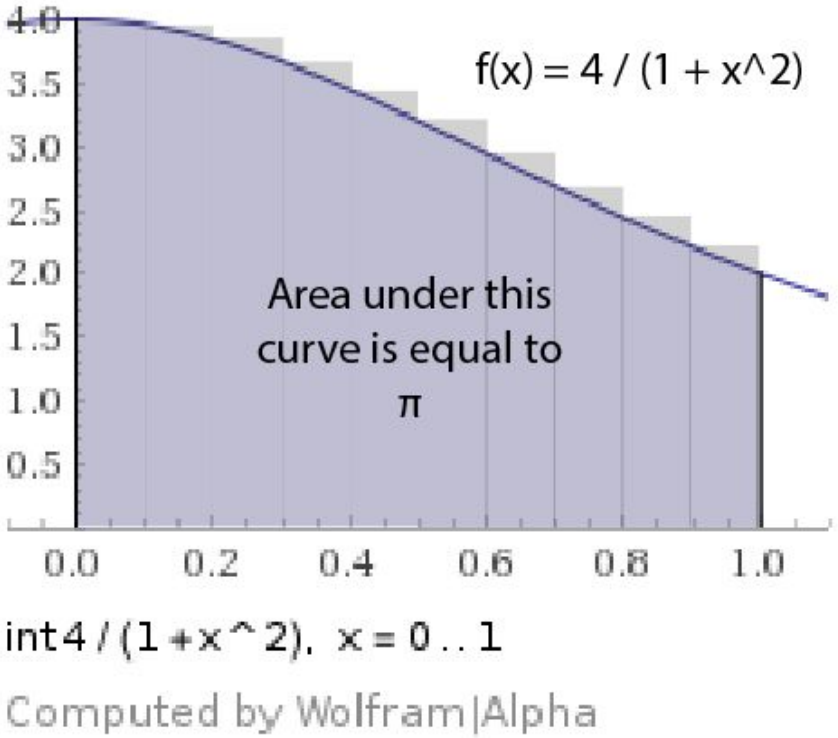


Activity Setup

In reality, to get an accurate representation of π, Δx must be very small.

$$\pi = \lim_{n\to\infty} \sum_0^n f(x_i)\frac{1}{n}$$

In the activity, you will plot how long this numerical integration takes when n is very large as a function of number of cores to illustrate how parallelization has speed-up limits.



Summary

To check the status on a job type ‘qstat’

It is good practice for long calculations to save their state periodically so that if/when the job is terminated the calculation can be restarted a the last save point

There is a limit to the speed-up that parallelization of code can achieve