In collaboration with Massachusetts Institute of Technology

Masdar INSTITUTE

Introduction to High Performance Computing

Dr. Sergio Martinez

Topics

1. Introduction to parallel computing
2. Parallel machines
3. Masdar Institute – HPC cluster
4. Using the HPC cluster

First Topic – Introduction to parallel computing

- What is parallel computing
- Why do parallel computing
- Real life scenario
- Types of parallelism
- Limits of parallel computing
- When to do parallel computing

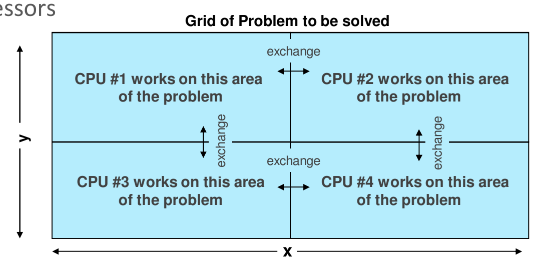
What is Parallel Computing?

- Consider your favorite computational application
  - One processor can give me results in N hours
  - Why not use N processors
  - ✓ and get the results in just one hour?

The concept is simple:
Parallelism = applying multiple processors to a single problem

Parallel computing

- Parallel computing: the use of multiple computers or processors working together on a common task.
  - Each processor works on its section of the problem
  - Processors are allowed to exchange information with other processors

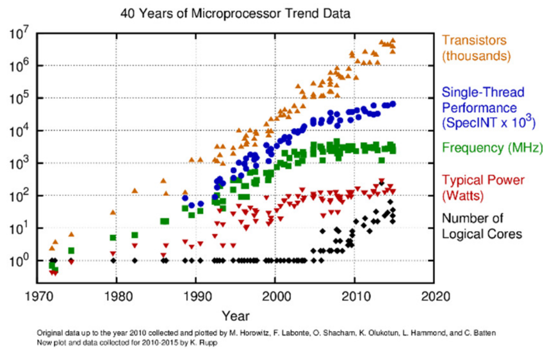

Grid of Problem to be solved
CPU #1 works on this area of the problem
CPU #2 works on this area of the problem
CPU #3 works on this area of the problem
CPU #4 works on this area of the problem
exchange
exchange
exchange
exchange
y
x

Why Do Parallel Computing?

- Limits of single CPU computing
  - ✓ Available memory
  - ✓ Performance/Speed
- Parallel computing allows:
  - ✓ Solve problems that don't fit on a single CPU's memory space
  - ✓ Solve problems that can't be solved in a reasonable time
- We can run…
  - ✓ Larger problems
  - ✓ Faster
  - ✓ More cases
  - ✓ Run simulations at finer resolution
  - ✓ Model physical phenomena more realistically

Why Do Parallel Computing?

- Chip density is continuing increase ~2x every 2 years
- Clock speed is not
- Number of processor cores may double instead
- Power is under control, no longer growing



Parallel Computing – Real Life Scenario

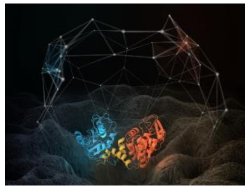
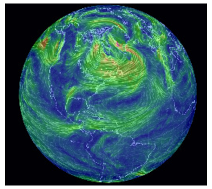
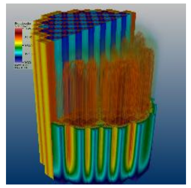
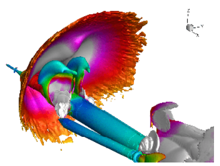
- What is the best strategy ?
  - ✓ Simple way is to divide the total books equally among workers. Each worker stacks the books one at a time. Worker must walk all over the library.
  - ✓ Alternate way is to assign different sections to each worker. Each worker is assigned equal # of books arbitrarily. Workers stack books in their section or pass to another worker responsible for the section it belongs to.

Parallel Computing – Real Life Scenario

- Parallel processing allows to accomplish a task faster by dividing the work into a set of sub-stacks assigned to multiple workers.
- Assigning a set of books to workers is task partitioning. Passing of books to each other is an example of communication between subtasks.
- Some problems may be completely serial; e.g. digging a post hole. Poorly suited to parallel processing.
- All problems are not equally amenable to parallel processing.

Application areas

- Vehicle design and dynamics
- Analysis of protein structures
- Human genome work
- Quantum chromodynamics
- Astrophysics
- Earthquake wave propagation
- Molecular dynamics
- Climate, ocean modeling
- CFD
- Imaging and Rendering
- Petroleum exploration
- Nuclear reactor, weapon design
- Database query
- Ozone layer monitoring
- Natural language understanding
- Study of chemical phenomena
- And many other scientific and industrial simulations

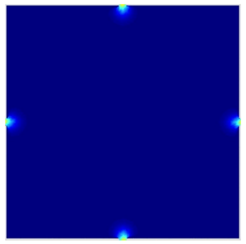





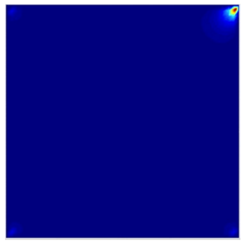
Types of Parallelism : Two Extremes

- Data parallel
  - ✓ Each processor performs the same task on different data
  - ✓ Example – heat propagation
- Task parallel
  - ✓ Each processor performs a different task
  - ✓ Example – image processing (edge detection)
- Most applications fall somewhere on the continuum between these two extremes

Typical Data Parallel Program

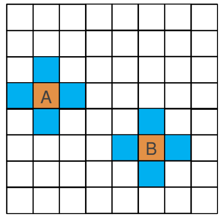
- Example: heat propagation 2D problem:
  - ✓ $T_{NEW} = T_{OLD} + k \cdot (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGTH} - 4 \cdot T_{OLD})$


Sources on margins


Source on top-right corner

Basics of Data Parallel Programming

- One code will run on N CPUs
- Each CPU will calculate the temperature of one cell
  - ✓ $T_{NEW} = T_{OLD} + k \cdot (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGTH} - 4 \cdot T_{OLD})$



CPU A

```
program:
...
Told = getTemp(A);
Ttop = getTopTemp(A);
Tbottom = getBottomTemp(A);
Tleft = getLeftTemp(A);
Tright = getRightTemp(A);
Tnew= Told+k*(Ttop+Tbottom …
   +Tleft+Tright-4*Told);
end program
```

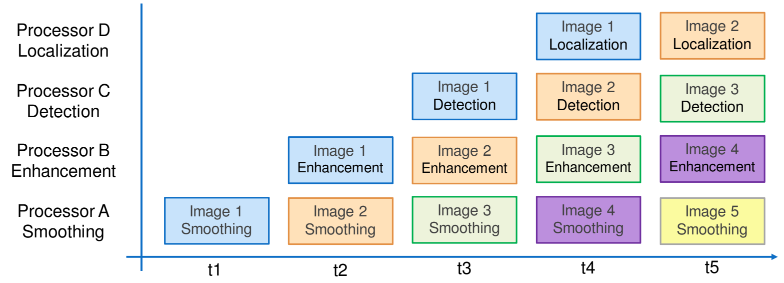
CPU B

```
program:
...
Told = getTemp(B);
Ttop = getTopTemp(B);
Tbottom = getBottomTemp(B);
Tleft = getLeftTemp(B);
Tright = getRightTemp(B);
Tnew= Told+k*(Ttop+Tbottom …
   +Tleft+Tright-4*Told);
end program
```

13

Typical Task Parallel Application

- Example: Image Processing, edge detection.
- Use one processor for each task
- Can use more processors if one is overloaded



14

Basics of Task Parallel Programming

- One code will run on 4 CPUs
- Program has 4 tasks to be done by 4 CPUs (A, B, C and D)

```
program.c:
…
initialize
...
if CPU=A then
   do Smoothing
elseif CPU=B then
   do Enhancement
elseif CPU=B then
   do Detection
elseif CPU=B then
   do Localization
end if
….
end program
```

CPU A

```
program:
...
Initialize
...
do Smoothing
...
end program
```

CPU B

```
program:
...
Initialize
...
do Enhancement
...
end program
```

CPU C

```
program:
...
Initialize
...
do Detection
...
end program
```

CPU D

```
program:
...
Initialize
...
do Localization
...
end program
```

15

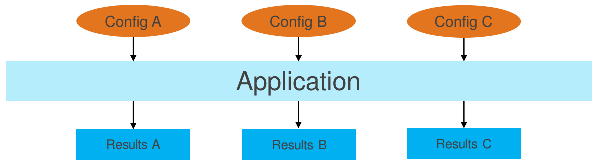
How Your Problem Affects Parallelism

- The nature of your problem constrains how successful parallelization can be
- Consider your problem in terms of
  - ✓ When data is used, and how
  - ✓ How much computation is involved, and when
- Importance of problem architectures
  - ✓ Embarrassingly parallel
  - ✓ Fully synchronous

16

Embarrassingly Parallel Problem

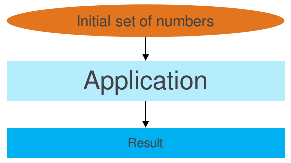
- Scenario: parameter sweep
  - ✓ Same application is run on data with different parameter configuration
  - ✓ Parallelism comes from having multiple data sets processed at once
  - ✓ Could be done on independent machines



- This is the simplest style of problem
- Key characteristic: calculations for each data set are independent
  - ✓ Could divide/replicate data into files and run as independent serial jobs
  - ✓ (also called "job-level parallelism")

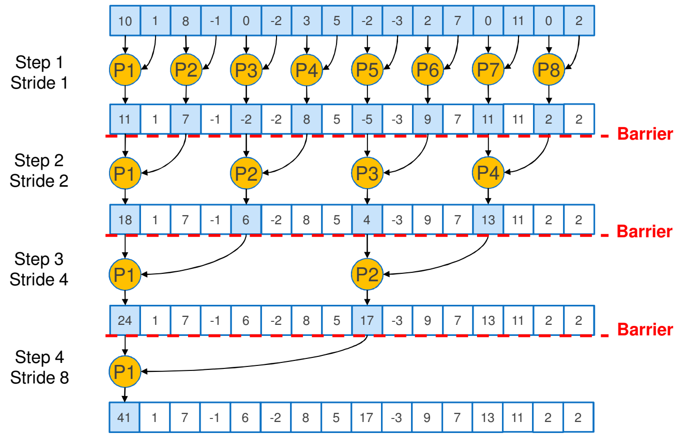
Fully Synchronous Problem

- All the processes synchronized at regular points
- Scenario: Reduction operation
  - ✓ Operations which are associative and commutative can be reduction operations
  - ✓ Some of them are addition, multiplication, logical AND/OR/XOR, finding maximum/minimum amongst a given set of numbers.



- Key characteristic: Requires the use of barriers
- Potential problems
  - ✓ Serial bottlenecks force other processors to "wait"

Fully Synchronous Problem
Reduction operation



Limits of Parallel Computing

- Theoretical Upper Limits
  - ✓ Amdahl's Law
- Practical Limits
  - ✓ Load balancing
  - ✓ Non-computational sections (I/O, system ops etc.)
- Other Considerations
  - ✓ Time to re-write code

Theoretical Upper Limits to Performance

- All parallel programs contain:
  - ✓ Serial sections
  - ✓ Parallel sections
- Serial sections – when work is duplicated or no useful work done (waiting for others) - limit the parallel effectiveness
  - ✓ Lot of serial computation gives bad speedup
  - ✓ No serial work "allows" perfect speedup
- Speedup is the ratio of the time required to run a code on one processor to the time required to run the same code on multiple (N) processors - Amdahl's Law states this formally

Amdahl's Law

- Amdahl's Law places a strict limit on the speedup that can be realized by using multiple processors.
  - ✓ Effect of multiple processors on run time

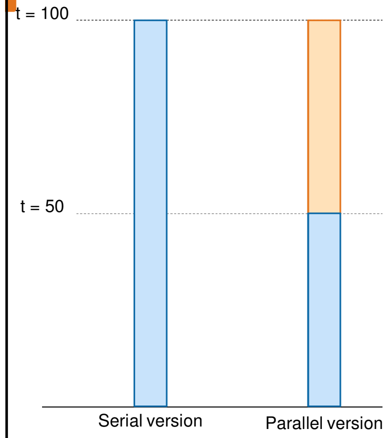
$$t_n = \left(f_p / N + f_s\right) t_1$$

  - ✓ Effect of multiple processors on speed up (S = t1/tn)

$$S = \frac{1}{f_p / N + f_s}$$

  - ✓ Where
    - fs = serial fraction of code
    - fp = parallel fraction of code
    - N = number of processors
    - tn = time to run on N processors

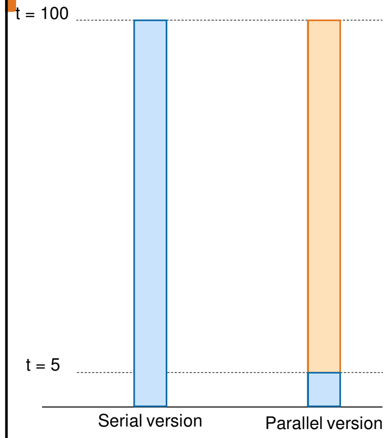
Limits of Parallel Computing



$$\text{Speedup} = \frac{\text{Time serial}}{\text{Time parallel}} = \frac{100}{50}$$

2x

Limits of Parallel Computing

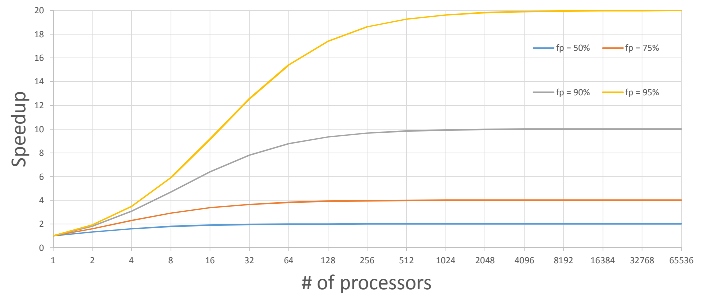


$$\text{Speedup} = \frac{\text{Time serial}}{\text{Time parallel}} = \frac{100}{5}$$

20x

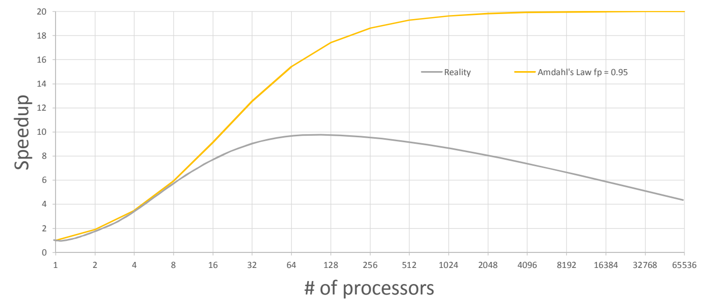
Illustration of Amdahl's Law

- It takes only a small fraction of serial content in a code to degrade the parallel performance.



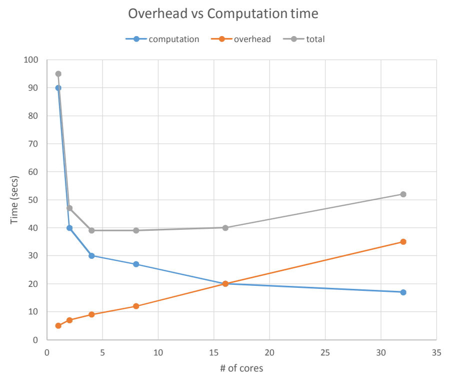
Amdahl's Law vs. Reality

- Amdahl's Law provides a theoretical upper limit on parallel speedup assuming that there are no costs for speedup, assuming that there are no costs for communications. In reality, communications will result in a further degradation of performance.



Practical Limits: Amdahl's Law vs. Reality

- In reality, Amdahl's Law is limited by many things:
  - ✓ Communications
  - ✓ I/O
  - ✓ Load balancing (waiting)
  - ✓ Scheduling (shared processors or memory)

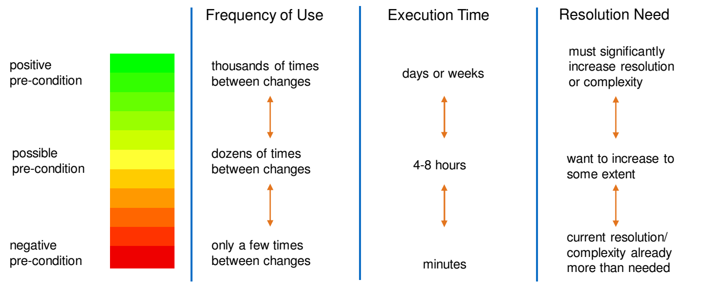


When do you do parallel computing?

- Writing effective parallel application is difficult
  - ✓ Communication can limit parallel efficiency
  - ✓ Serial time can dominate
  - ✓ Load balance is important
- Is it worth your time to rewrite your application?
  - ✓ Do the CPU requirements justify parallelization?
  - ✓ Will the code be used just once?

Will the investment of your time be worth it?

Test the "Preconditions for Parallelism"


Frequency of Use
Execution Time
Resolution Need
positive pre-condition
possible pre-condition
negative pre-condition
thousands of times between changes
dozens of times between changes
only a few times between changes
days or weeks
4-8 hours
minutes
must significantly increase resolution or complexity
want to increase to some extent
current resolution/ complexity already more than needed

- According to experienced parallel programmers:
  - ✓ no green ⟶ Don't even consider it
  - ✓ one or more red ⟶ Parallelism may cost you more than you gain
  - ✓ all green ⟶ You need the power of parallelism (but there are no guarantees)

29

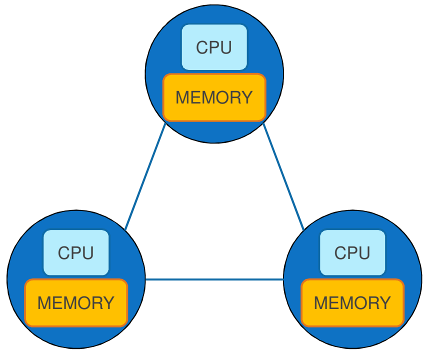
Second Topic – Parallel Machines

- Simplistic architecture
- Types of parallel machines

30

Simplistic architecture

- Processors
- Memory
- Interconnect Network



31

Processor Related Terms

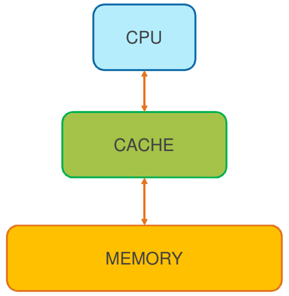
- RISC: Reduced Instruction Set Computers
- PIPELINE : Technique where multiple instructions are overlapped in execution
- SUPERSCALAR: Multiple instructions per clock period

IF	ID	EX	MEM	WB				
IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB
				IF	ID	EX	MEM	WB

32

Memory/Cache Related Terms

- CACHE : Cache is the level of memory hierarchy between the CPU and main memory. Cache is much smaller than main memory and hence there is mapping of data from main memory to cache.



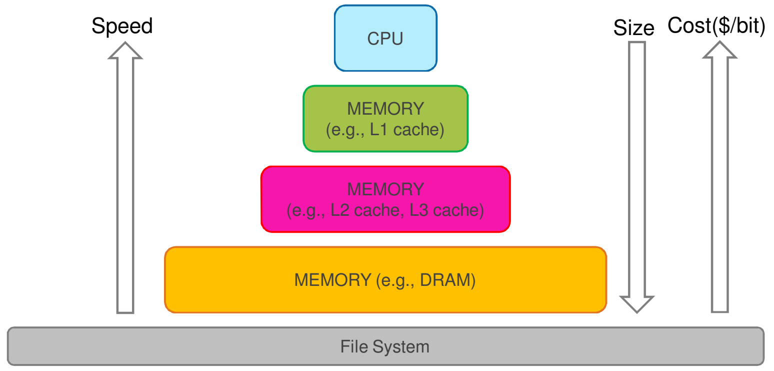
33

Memory/Cache Related Terms

- ICACHE : Instruction cache
- DCACHE (L1) : Data cache closest to registers
- SCACHE (L2) : Secondary data cache
  - ✓ Data from SCACHE has to go through DCACHE to registers
  - ✓ SCACHE is larger than DCACHE
- L3 cache

34

Memory/Cache Related Terms



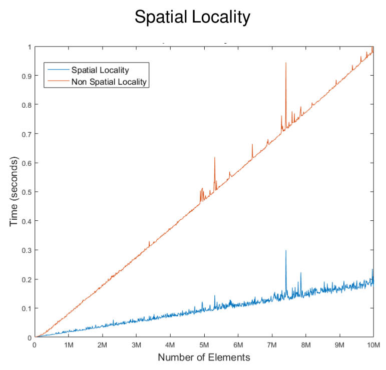
35

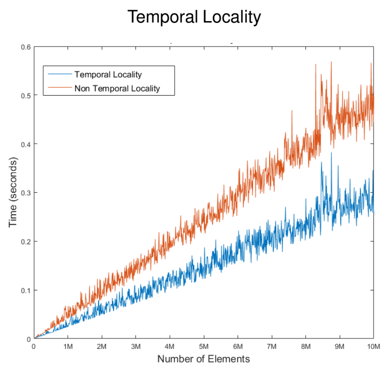
Memory/Cache Related Terms

- The data cache was designed with two key concepts in mind:
- Spatial Locality
  - ✓ When an element is referenced its neighbors will be referenced too
  - ✓ Cache lines are fetched together
  - ✓ Work on consecutive data elements in the same cache line
- Temporal Locality
  - ✓ When an element is referenced, it might be referenced again soon
  - ✓ Arrange code so that date in cache is reused as often as possible

36

Memory/Cache Related Terms





37

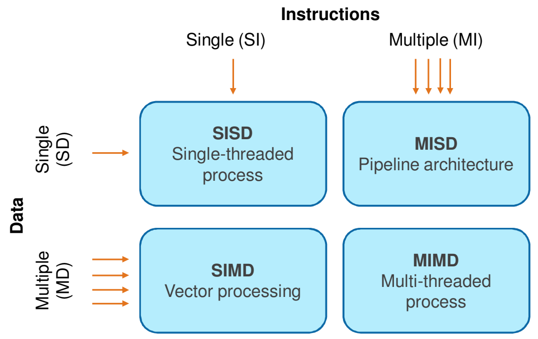
Network Interconnect Related Terms

- LATENCY : How long does it take to start sending a "message"? Units are generally microseconds now a days.
- BANDWIDTH : What data rate can be sustained once the message is started? Units are bytes/sec, Mbytes/sec, Gbytes/sec etc.
- TOPLOGY: What is the actual ‘shape’ of the interconnect? Are the nodes connect by a 2D mesh? A ring? Something more elaborated?

38

Types of Parallel Machines

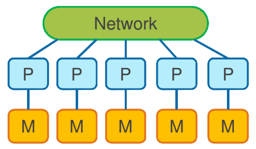
- Flynn's taxonomy has been commonly use to classify parallel computers into one of four basic types:

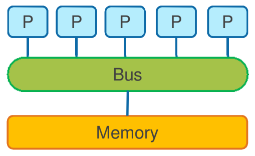


39

Shared and Distributed memory

- Distributed memory
  - ✓ Each processor has its own local memory. Must do message passing to exchange data between processors.
  - ✓ Examples: CRAY T3E, XT; IBM Power,Sun and other vendor made machines
- Shared memory
  - ✓ Single address space. All processors have access to a pool of shared memory.
  - ✓ Examples: CRAY T90, SGI Altix

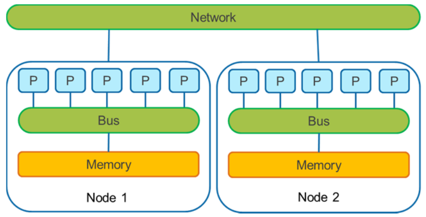




40

Hybrid machines

- Each node has its own memory (Distributed memory)
- Memory on each node is shared within the processors (Shared memory)
- Programming
  - ✓ Threads like methods within the same node (OpenMP)
    - Explicitly Start multiple tasks
    - Each given own section of memory
    - Use shared variables for communication
  - ✓ Message passing within different nodes (MPI)



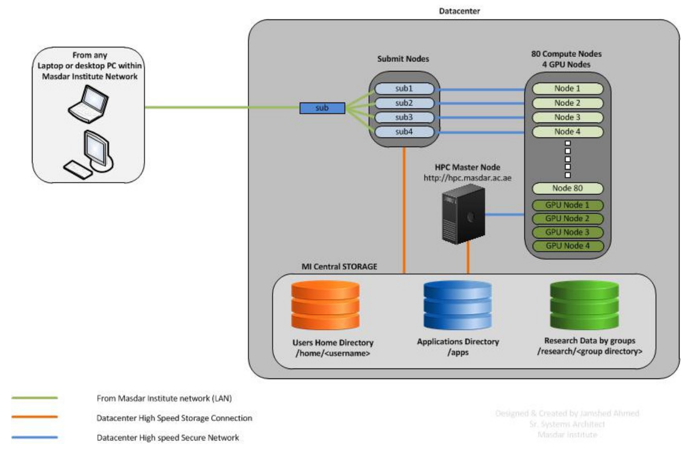
ThirdTopic – Masdar Institute – HPC

- Nodes, processors and cores
- High Level Network Layout – Masdar Institute
- Compute Nodes
- GPU Nodes

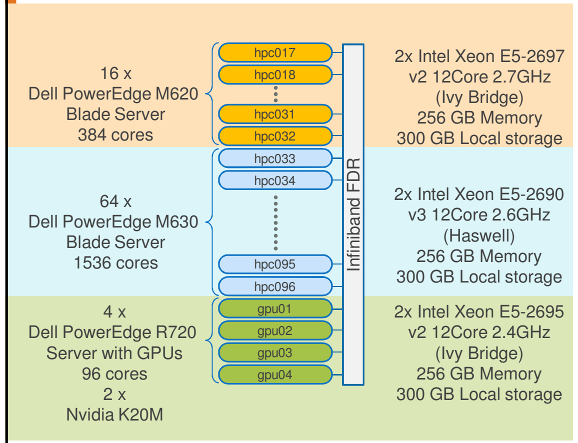
Nodes, processors and cores

- Important to know what is meant by nodes, processors and cores.
  - ✓ **Nodes (hosts)**: Refers to the physical machine/server. In current systems, a node would typically include one or more processors, as well as memory and other hardware.
  - ✓ **Processor**: Refers to the central processing unit (CPU), which contains one or more cores.
  - ✓ **Cores**: Refers to the basic computation unit of the CPU. This is unit that carries out the actual computations.
- So in essence, each compute node contains one or more processors/CPUs and each CPU will typically consist of one or more cores.

HPC Cluster
High Level Network Layout – Masdar Institute


Datacenter
From any Laptop or desktop PC within Masdar Institute Network
Submit Nodes
sub
sub1
sub2
sub3
sub4
80 Compute Nodes 4 GPU Nodes
Node 1
Node 2
Node 3
Node 4
Node 80
GPU Node 1
GPU Node 2
GPU Node 3
GPU Node 4
HPC Master Node
http://hpc.masdar.ac.ae
MI Central STORAGE
Users Home Directory /home/<username>
Applications Directory /apps
Research Data by groups /research/<group directory>
From Masdar Institute network (LAN)
Datacenter High Speed Storage Connection
Datacenter High speed Secure Network
Designed & Created by Jamshed Ahmed
Sr. Systems Architect
Masdar Institute

HPC Cluster
Compute Nodes

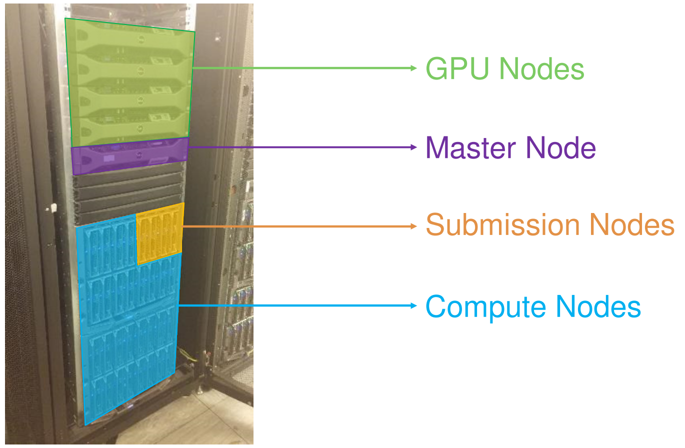


	Nodes	Specs
16 x Dell PowerEdge M620 Blade Server 384 cores	hpc017, hpc018, ⋮, hpc031, hpc032	2x Intel Xeon E5-2697 v2 12Core 2.7GHz (Ivy Bridge) 256 GB Memory 300 GB Local storage
64 x Dell PowerEdge M630 Blade Server 1536 cores	hpc033, hpc034, ⋮, hpc095, hpc096	2x Intel Xeon E5-2690 v3 12Core 2.6GHz (Haswell) 256 GB Memory 300 GB Local storage
4 x Dell PowerEdge R720 Server with GPUs 96 cores 2 x Nvidia K20M	gpu01, gpu02, gpu03, gpu04	2x Intel Xeon E5-2695 v2 12Core 2.4GHz (Ivy Bridge) 256 GB Memory 300 GB Local storage

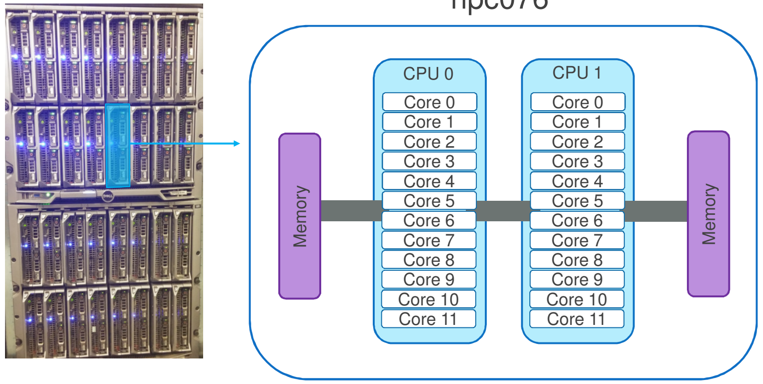
TOTAL

84 Nodes
168 Processors
2016 Cores
21.5 TB Mem
23+ TB NFS

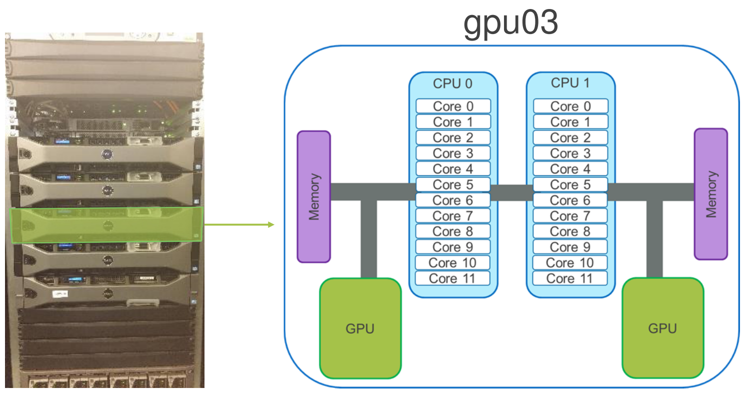
Datacenter images



Schematic of a compute node



Schematic of a GPU node



Nvidia Tesla K20M

Specifications	Tesla K20M
CUDA Driver Version / Runtime Version	6.5 / 6.5
CUDA Capability Major/Minor version number	3.5
Total amount of global memory	4800 MBytes (5032706048 bytes)
(13) Multiprocessors, (192) CUDA Cores/MP	2496 CUDA Cores
GPU Clock rate	706 MHz (0.71 GHz)
Memory Clock rate	2600 Mhz
Total amount of constant memory	65536 bytes
Total amount of shared memory per block	49152 bytes
Total number of registers available per block	65536 (32 bits each)
Warp size	32
Maximum number of threads per multiprocessor	2048
Maximum number of threads per block	1024
Max dimension size of a thread block (x,y,z)	(1024, 1024, 64)
Max dimension size of a grid size (x,y,z)	(2147483647, 65535, 65535)

49

Fourth Topic – Using the HPC cluster

- Basic use: Logging on/off and file transfer
- Modular environment
- Job submission

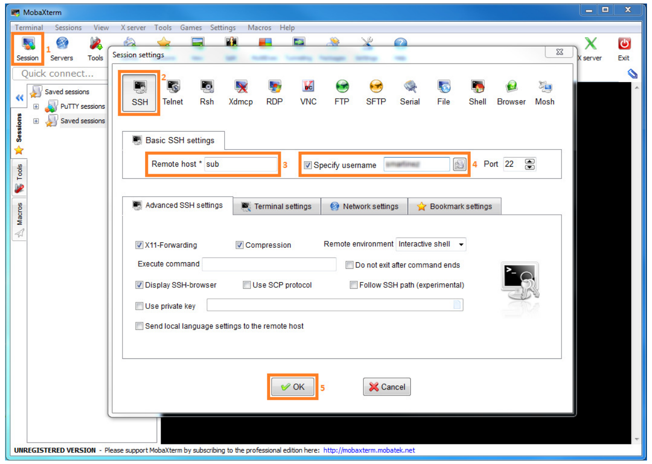
50

Basic use: Logging on/off

- Logging on:
- Use secure shell from UNIX:
  - ✓ $ ssh –Y <username>@sub
- From Windows use MobaXterm which has an inbuilt X-server
  - ✓ MobaXterm: http://mobaxterm.mobatek.net/
- To log off, type "logout"

51

Basic use: Logging on/off (MobaXterm)



Remote host * sub

[✓] Specify username

Port 22

[✓] X11-Forwarding [✓] Compression Remote environment Interactive shell

Execute command [ ] Do not exit after command ends

[✓] Display SSH-browser [ ] Use SCP protocol [ ] Follow SSH path (experimental)

[ ] Use private key

[ ] Send local language settings to the remote host

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

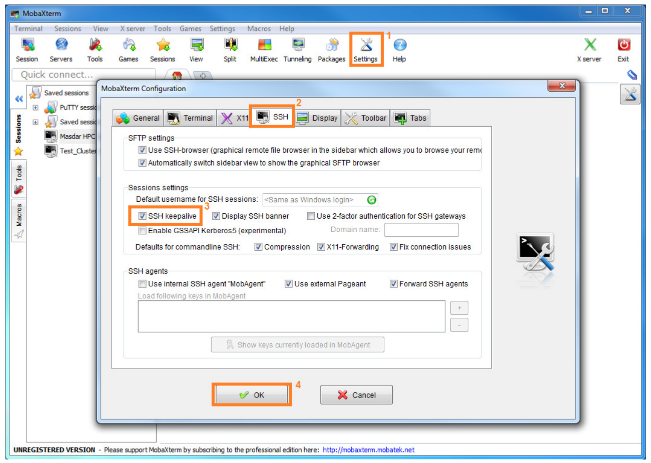
52

Basic use: Keepalive

- A **keepalive** (KA) is a message sent by one device to another to check that the link between the two is operating, or to prevent this link from being broken use secure shell from UNIX:
- Linux
  - ✓ Open the file /etc/ssh/ssh_config with your text editor.
  - ✓ At the end of the file add this text in a new line:
    - ServerAliveInterval 180
  - ✓ Save the file

53

Basic use: Keepalive (MobaXterm)


MobaXterm Configuration

SFTP settings
- ☑ Use SSH-browser (graphical remote file browser in the sidebar which allows you to browse your remo
- ☑ Automatically switch sidebar view to show the graphical SFTP browser

Sessions settings
- Default username for SSH sessions: <Same as Windows login>
- ☑ SSH keepalive
- ☑ Display SSH banner
- ☐ Use 2-factor authentication for SSH gateways
- ☐ Enable GSSAPI Kerberos5 (experimental)
- Domain name:
- Defaults for commandline SSH: ☑ Compression ☑ X11-Forwarding ☑ Fix connection issues

SSH agents
- ☐ Use internal SSH agent "MobAgent"
- ☑ Use external Pageant
- ☑ Forward SSH agents
- Load following keys in MobAgent

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net
MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
1
Quick connect...
Saved sessions
PuTTY sessi
Saved sessi
Masdar HPC
Test_Cluster
Sessions
Tools
Macros
General Terminal X11 SSH Display Toolbar Tabs
2
3
Show keys currently loaded in MobAgent
OK
4
Cancel

54

Basic use

- To change password, use the passwd command.
- Use regular UNIX commands for file listing and directory access : ls, cd, pwd, etc.
- Default shell is the Bourne (bash) shell.
- Environment is set by .bash_profile file.
- Files can be edited with either: vi, nano using the command line.
- Files can be edited with your favorite editor using MobaXterm or Linux graphic interface.

55

Exercise 1: LOGGING IN

- Go to Exercises document:
  - ✓ Exercise 1: Logging in

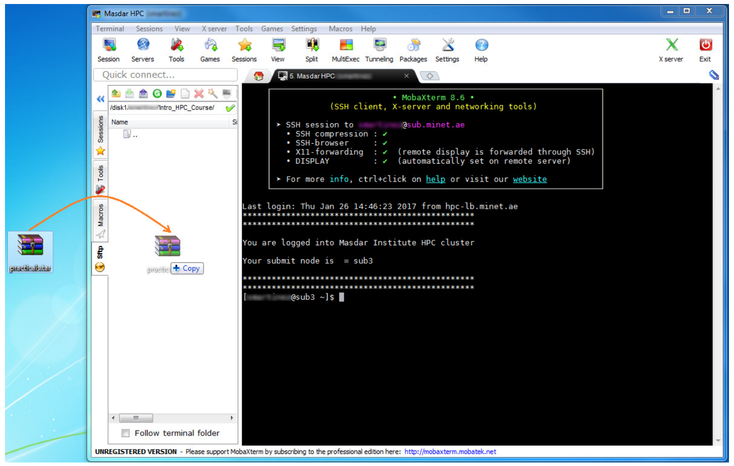
56

Basic use: File transfer

- File transfer via scp:
  - ✓ From local machine to upload a file:
    - $ scp <file> <username>@sub:<directory>
    - $ scp –r <directory> <username>@sub:<directory>
  - ✓ From HPC to transfer a file back:
    - $ scp <file> <localhostusername@<localmachine>:<directory>
    - $ scp –r <directory> <localhostusername@<localmachine>:<directory>
- File transfer via rsync:
  - ✓ From local machine to upload a file:
    - $ rsync –v –e ssh <file> <username>@sub:<directory>
    - $ rsync –v –e ssh –r <directory> <username>@sub:<directory>
  - ✓ From HPC to transfer a file back:
    - $ rsync –v –e ssh <file> <localhostusername@<localmachine>:<directory>
    - $ rsync –v –e ssh –r <directory> <localhostusername@<localmachine>:<directory>

57

Basic use: File transfer (MobaXterm)


Masdar HPC
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
5. Masdar HPC
Name
Copy
Follow terminal folder

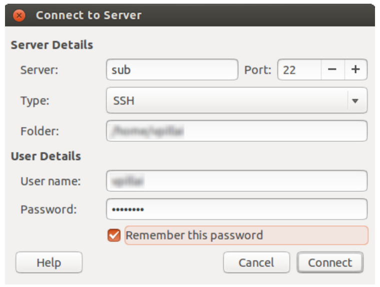
```
                    • MobaXterm 8.6 •
       (SSH client, X-server and networking tools)

 ➤ SSH session to [redacted]@sub.minet.ae
   • SSH compression : ✓
   • SSH-browser     : ✓
   • X11-forwarding  : ✓  (remote display is forwarded through SSH)
   • DISPLAY         : ✓  (automatically set on remote server)

 ➤ For more info, ctrl+click on help or visit our website

Last login: Thu Jan 26 14:46:23 2017 from hpc-lb.minet.ae
************************************************
************************************************

You are logged into Masdar Institute HPC cluster

Your submit node is  = sub3

************************************************
************************************************
[[redacted]@sub3 ~]$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

58

Basic use: File transfer (Ubuntu)

- Open your home folder
- File -> Connect to server…
- Fill the information with your credentials



Connect to Server

Server Details

Server:	sub
Port:	22
Type:	SSH
Folder:	[redacted]

User Details

User name:	[redacted]
Password:	••••••••

[✓] Remember this password

Help Cancel Connect

59

Exercise 2: FILE TRANSFER

- Go to Exercises document
  - ✓ Exercise 2: File transfer

60

Modular environment

- On a complex computer system, on which it is necessary to make available a wide choice of software packages in multiple versions, it can be quite difficult to set up the user environment so as to always find the required executables and libraries.
- This is particularly true where different implementations or versions use the same names for files.
- Environment modules provide a way to selectively activate and deactivate modifications to the user environment which allow particular packages and versions to be found.

Modular environment

- Module command is used to alter user environment:
- module list - lists currently loaded modules.
- module avail – lists all available modules.
- module load <module_name> - loads the module <module_name>. e.g.:
  - ✓ $ module load matlab/R2015b
- module switch <old_mod> <new_mod> - switch similar/conflicting modules. e.g.:
  - ✓ $ module switch matlab/R2015a matlab/R2015b

Exercise 3: MODULAR ENVIRONMENT

- Go to Exercises document:
  - ✓ Exercise 3: Modular environment

Scheduler and Policies

- Job: is the unit of work or execution. It can be interactive or batch.
- In distributed HPC environment, jobs are managed by job schedulers.
- The scheduler can manage multiple queues where jobs are submitted to.
- In our cluster, the scheduler is Platform Load Sharing Facility (or Platform LSF), from IBM

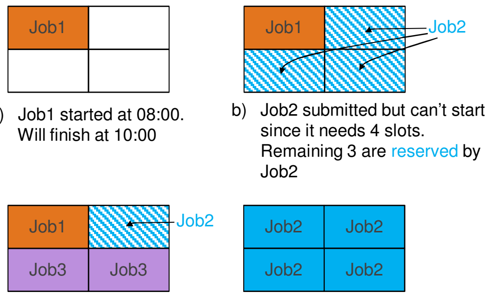
Scheduler and Policies – Queues

- Queues:
  - ✓ general [hpc021-hpc096]
    - ✓ Batch and Interactive
    - ✓ Exclusive mode
    - ✓ Default wall-time: 2 hours
    - ✓ Maximum wall-time: 48 hours
    - ✓ Fairshare configured for users
    - ✓ Backfill
    - ✓ Advance reservation
  - ✓ long [hpc017-hpc021]
    - ✓ Batch and Interactive
    - ✓ Exclusive mode
    - ✓ Default wall-time: 2 days
    - ✓ Maximum wall-time: 7days
    - ✓ Fairshare configured for users
    - ✓ Backfill
    - ✓ Max one job (running) per user
  - ✓ gpu[gpu01-gpu04]

Scheduler and Policies – Backfill Scheduling

- Allows the reserved job slots to be used by small jobs that can run and finish before the large job starts.

08:00 Job1: 1 slot, 2 hours
08:00 Job2: 4 slots, 1 hour
08:30 Job3: 2 slots, 1 hour



a) Job1 started at 08:00. Will finish at 10:00

b) Job2 submitted but can't start since it needs 4 slots. Remaining 3 are reserved by Job2

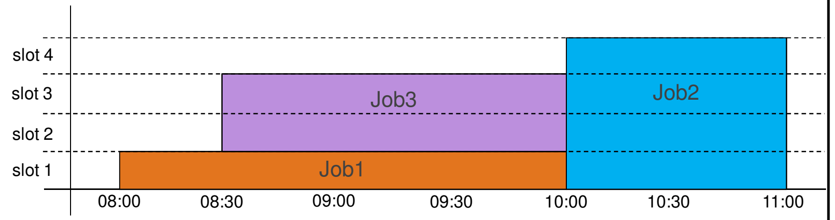
c) At 08:30 Job3 is submitted. Job3 backfills Job2

d) At 10:00, Job2 starts

Scheduler and Policies – Backfill Scheduling

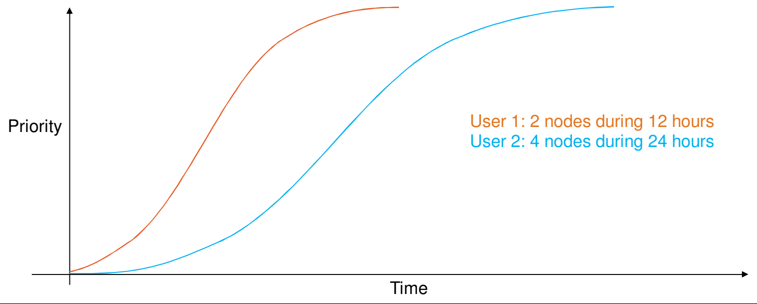
- Allows the reserved job slots to be used by small jobs that can run and finish before the large job starts.

08:00 Job1: 1 slot, 2 hours
08:00 Job2: 4 slots, 1 hour
08:30 Job3: 2 slots, 1,5 hour



Scheduler and Policies – Fairshare

- Fairshare feature calculates and assigns users' or groups' priority. This priority is inversely proportional to the utilization of the resources.
- This avoids users or groups to monopolize the resources.



Job submission and control

- Both interactive and batch jobs may be submitted
  - ✓ In addition, can monitor jobs and status of the job queue, cancel/delete jobs.
  - ✓ Below are listed the important batch scheduler commands along with a brief description. More information is available via the man pages, e.g. man bsub.

Command	Example	Description
`bsub`	`bsub < myscript.sh`	Submits a job to the queue
`bhosts`	`bhosts -l`	Displays hosts and their static and dynamic resources
`bjobs`	`bjobs –u all`	Displays information about jobs
`bkill`	`bkill 10254`	Sends a signal to a job (kill the job)
`bhist`	`bhist –l 10254`	Displays historical information about jobs

Exercise 4: PLATFORM LSF

- Go to Exercises document:
  - ✓ Exercise 4: Platform LSF

Job Submission – Configuring job

- bsub takes several options:

Option	Description
`-n <cores>`	Requests a number of CPU cores.
`-q <queue-name>`	Submits job to the specified queue. By default this is "normal".
`-o <file_name>`	Writes output to <file name>, "%J" can be used to include the JOBID, e.g. "result.J%.out".
`-e <file_name>`	Writes errors to <file_name>, "%J" can be used to include the JOBID, e.g. "result.J%.err".
`-J <job_name>`	Assigns the specified name "<job_name>" to the job.
`-W <HH:MM>`	Requests wall time limit of HH hours and MM minutes.
`-R "rusage[mem=XXX]"`	Requests a specific amount of memory for your job. XXX is the memory size in MB.
`-R "span[ptile=m]"`	Requests m CPU cores on each node.
`-R "span[hosts=1]"`	Indicates that all the processors allocated to this job must be on the same host.
`-I`	Submits an interactive job. Use `-XF` with `-I` to submit an *interactive* job using SSH X11 forwarding.
`-x`	Puts the host running your job into exclusive execution mode.

Job Submission – Batch jobs

- Batch jobs are submitted using bsub:
  - ✓ `bsub –n 4 myjob`
- This will return a jobid:
  - ✓ `Job <93446> is submitted to default queue <general>.`
- Execution status can be checked with bjobs: "PEND" pending, "RUN" running, "SUSP" suspended. Use bjobs –u all to display all jobs
- Jobs are deleted with bkill <jobid>, e.g.:
  - ✓ `$ bkill 10254`

Job submission methods – Example program

- Circle Area (python): The program gets the radius of a circle as argument and calculates its circumference and its area.
- We will use the same example to explain the different submission methods.

circle_area.py

```python
import sys
from math import pi

print ("Radius is = %s" % sys.argv[1])
radius=float(sys.argv[1])
circumference = 2*pi*radius
print ("Circumference is = %.5f" % circumference)
area = pi * pow(radius,2)
print ("Area is = %.5f" % area)
```

Job submission methods – Command-line

- In its simplest form a job can be submitted to the default queue using a command such:
  - ✓ `bsub -n 1 -q general -J circle_area -o circle_area.%J.out -e circle_area.%J.err "python circle_area.py 5"`
- This will run the specified command and write standard output and error to a file with the job number %J in the filename.
- There are many additional options that can be used to control the resources allocated to an individual job including the queue to submit to, the wall time limit (-W option), and any processor or memory allocation/limitations for your job.
- Not indicated for complex jobs.

Job submission methods – Script file (spooling)

- For more complex jobs, running a sequence of commands or requiring more complicated environment configuration, a script can be constructed, such as:

config.sh

```
#BSUB -n 1
#BSUB -q general
#BSUB -J circle_area
#BSUB -o circle_area.%J.out
#BSUB -e circle_area.%J.err

python ./circle_area.py 5
```

- This bash script contains a series of directives, prefixed with `#BSUB` which provides the same functionality as the command line arguments above.
- To submit a job using a job script file it is important to use a redirect such as: bsub < config.sh

Job submission – Multiple submission

- Command-line
  - ✓ `bsub -n 1 -q general -J circle_area -o circle_area.%J.out -e circle_area.%J.err "python circle_area.py 3“`
  - ✓ `bsub -n 1 -q general -J circle_area -o circle_area.%J.out -e circle_area.%J.err "python circle_area.py 4“`
  - ✓ `bsub -n 1 -q general -J circle_area -o circle_area.%J.out -e circle_area.%J.err "python circle_area.py 5“`
- Script files

config_3.sh

```
#BSUB -n 1
#BSUB -q general
#BSUB -J circle_area
#BSUB -o circle_area.%J.out
#BSUB -e circle_area.%J.err

python ./circle_area.py 3
```

config_4.sh

```
#BSUB -n 1
#BSUB -q general
#BSUB -J circle_area
#BSUB -o circle_area.%J.out
#BSUB -e circle_area.%J.err

python ./circle_area.py 4
```

config_5.sh

```
#BSUB -n 1
#BSUB -q general
#BSUB -J circle_area
#BSUB -o circle_area.%J.out
#BSUB -e circle_area.%J.err

python ./circle_area.py 5
```

  - ✓ `bsub < config_3.sh`
  - ✓ `bsub < config_4.sh`
  - ✓ `bsub < config_5.sh`

Exercise 5: JOB SUBMISSION METHOD

- Go to Exercises document:
  - ✓ Exercise 5: Job submission methods

Job Submission - Interactive jobs

- Can allow input and output from terminal.
- E.g. Matlab, Lumerical, data analysis or debugging
- Commands will only be run if resources are available
- Both serial and parallel use.
- Same defaults and constraints as batch jobs.
- Commands, take identical options to bsub:
  - ✓ `bsub -n 24 –W 120 -R "span[hosts=1]" -I -q general -XF fdtd-solutions`
    - `-I` for interactive
    - `-XF` for an X11 job
- Appear in bjobs and can be controlled through bkill.

Exercise 6: INTERACTIVE JOBS

- Go to Exercises document:
  - ✓ Exercise 6: Interactive jobs

Job submission – Array jobs

- Array syntax is an efficient way of submitting multiple jobs simultaneously
- Useful when submitting a large number of jobs
  - ✓ Ensemble forecasting
  - ✓ Parameter sweep
- Avoids “looping” and unnecessarily submitting a large number of jobs with different numbers
- Most batch-submission platforms have this feature

Job submission – Array jobs
Configuration

- How to configure an array of jobs:

```
bsub < config_typical.sh                bsub < config_array.sh
```

config_typical.sh

```
#BSUB -n 1
#BSUB -J circle_area
#BSUB -o circle_area.%J.out
#BSUB -e circle_area.%J.err

python ./circle_area.py 5
```

config_array.sh

```
#BSUB -n 1
#BSUB -J circle_area[1-6]
#BSUB -o circle_area.%I.%J.out
#BSUB -e circle_area.%I.%J.err

python ./circle_area.py $LSB_JOBINDEX
```

- Indicate the array in the job name: #BSUB -J circle_area[1-6]
- "%I" refers to the job element
- When a job array is submitted, each array value is stored as variable $LSB_JOBINDEX

Job submission – Array jobs
bjobs output

- Typical:

```
JOBID   USER       STAT  QUEUE    FROM_HOST  EXEC_HOST  JOB_NAME     SUBMIT_TIME
93794   smartinez  RUN   general  sub4       hpc070     circle_area  Nov 29 10:25
```

- Array:

```
JOBID   USER       STAT  QUEUE    FROM_HOST  EXEC_HOST  JOB_NAME        SUBMIT_TIME
93795   smartinez  RUN   general  sub4       hpc070     circle_area[1]  Nov 29 10:26
93795   smartinez  RUN   general  sub4       hpc034     circle_area[2]  Nov 29 10:26
93795   smartinez  RUN   general  sub4       hpc056     circle_area[3]  Nov 29 10:26
93795   smartinez  RUN   general  sub4       hpc074     circle_area[4]  Nov 29 10:26
93795   smartinez  RUN   general  sub4       hpc027     circle_area[5]  Nov 29 10:26
93795   smartinez  RUN   general  sub4       hpc083     circle_area[6]  Nov 29 10:26
```

- All elements have same JOBID, but individual ones can be specified using brackets (e.g., 93795[5])

Job submission – Array jobs
Killing an array of jobs

- To kill the whole array, you only need to specify the single job number:
  - ✓ bkill 123 kills all jobs in the array
- You can also kill a single entry of an array:
  - ✓ bkill "123[5]" kills job array element 5

Job submission – Array jobs
Limiting the number of jobs that can run

- Sometimes it may be desirable to limit how many array members can run simultaneously
- This can be limited by adding "%val" in the submission
  - ✓ #BSUB -J circle_area[1-10]%3
- In this example, a maximum of 3 jobs will run simultaneously

Exercise 7: ARRAY JOBS

- Go to Exercises document:
  - ✓ Exercise 7: Array jobs

Job submission – Array jobs Advanced

- We don't need to submit "continuous numbers":
  - ✓ #BSUB -J circle_area[1-10] passes values of 1,2,3,4,5,6,7,8,9,10 to $LSB_JOBINDEX
  - ✓ #BSUB -J circle_area[1,3-6,42,90] passes values of 1,3,4,5,6,42, and 90 to $LSB_JOBINDEX
  - ✓ #BSUB -J circle_area[1,10:2] passes values of 1,3,5,7, and 9 to $LSB_JOBINDEX
- Advanced Array Job:

radius.txt

```
3.7861
8.1158
5.3283
3.5073
9.3900
8.7594
5.5016
6.2248
5.8704
2.0774
```

```
bsub < config_advanced.sh
```

config_advanced.sh

```
#BSUB -n 1
#BSUB -J circle_area[1-10]
#BSUB -o circle_area.%I.%J.out
#BSUB -e circle_area.%I.%J.err

radius=`head -n$LSB_JOBINDEX radius.txt | tail -n1`
python ./circle_area.py $radius
```

Job submission – Array jobs Dependency conditions

- Arrays can easily be used in dependency conditions:
  - ✓ #BSUB -w "ended (square_area[1-10])
  - ✓ #BSUB -J circle_area[1-10]
- In this case, "circle_area" won't execute until all elements of "square_area" have ended

Exercise 8: ARRAY JOBS ADVANCED

- Go to Exercises document:
  - ✓ Exercise 8: Array jobs advanced

Thank You

Ver. 1

89