INTRO TO LINUX SERVER & CCR

SESSION 02

By: Ningji Wei

October 15, 2018

University at Buffalo The State University of New York

About Last Session

Note1 : first time login CCR, you need to reset your password in CCR Identity Management Portal

About Last Session

Note1 : first time login CCR, you need to reset your password in CCR Identity Management Portal

Note2 : CCR Help Center (https://ubccr.freshdesk.com/support/home)

About Last Session

Note1 : first time login CCR, you need to reset your password in CCR Identity Management Portal

Note2 : CCR Help Center (https://ubccr.freshdesk.com/support/home)

Note3 : Video of session 1 has been uploaded in YouTube

About Last Session

Note1 : first time login CCR, you need to reset your password in
CCR Identity Management Portal

Note2 : CCR Help Center (https://ubccr.freshdesk.com/support/home)

Note3 : Video of session 1 has been uploaded in YouTube
Search: Linux Server CCR

Course Checklist

Course Checklist

Basics:
know how to finish common tasks in CCR

Course Checklist

Basics: know how to finish common tasks in CCR	**Advanced Tools:** more convenient and efficient

Course Checklist

Basics:
know how to finish common tasks in CCR

- [ ] Connect to CCR
- [ ] Filesystem Operations
- [ ] CCR Operations
- [ ] File Transfer

Advanced Tools:
more convenient and efficient

Course Checklist

Basics:
know how to finish common tasks in CCR

- [x] Connect to CCR
- [ ] Filesystem Operations
- [ ] CCR Operations
- [ ] File Transfer

Advanced Tools:
more convenient and efficient

Course Checklist

Basics:
know how to finish common tasks in CCR

- [x] Connect to CCR
- [ ] Filesystem Operations
- [ ] CCR Operations
- [ ] File Transfer

Advanced Tools:
more convenient and efficient

not now ...

Contents:

Basic Linux Commands Recap

Basic Linux Commands Recap

Usage	Command
Change directory	$ cd PATH
List contents	$ ls PATH
Print	$ echo STRING
Create file	$ touch FILE
Create folder	$ mkdir FOLDER
Remove file/folder	$ rm [-r] FILE/FOLDER
Move file/folder	$ mv [-r] FILE/FOLDER
Copy file/folder	$ cp [-r] FILE/FOLDER

Basic Linux Commands Recap

Usage	Command
Change directory	$ cd PATH
List contents	$ ls PATH
Print	$ echo STRING
Create file	$ touch FILE
Create folder	$ mkdir FOLDER
Remove file/folder	$ rm [-r] FILE/FOLDER
Move file/folder	$ mv [-r] FILE/FOLDER
Copy file/folder	$ cp [-r] FILE/FOLDER

Trick	Hotkey
Auto complete	tab
Show candidates	tab twice
Jump to the start of the line	C-a
Jump to the end of the line	C-e
Move cursor	left/right
Fast move cursor	C-left/C-right
Prev/Next command	up/down

Basic Linux Commands Recap

Usage	Command
Change directory	$ cd PATH
List contents	$ ls PATH
Print	$ echo STRING
Create file	$ touch FILE
Create folder	$ mkdir FOLDER
Remove file/folder	$ rm [-r] FILE/FOLDER
Move file/folder	$ mv [-r] FILE/FOLDER
Copy file/folder	$ cp [-r] FILE/FOLDER

Trick	Hotkey
Auto complete	tab
Show candidates	tab twice
Jump to the start of the line	C-a
Jump to the end of the line	C-e
Move cursor	left/right
Fast move cursor	C-left/C-right
Prev/Next command	up/down

Practice:

1. create folder prac1, prac2
2. create files test1.dat, test2.dat, test3.dat, na.info in folder prac1

New Trick: Wildcards

New Trick: Wildcards

Wildcards	Symbol	Meaning	Examples
Star Wildcard	*	string with arbitrary length	$ ls *.txt $ rm *exp*.dat
Question Mark Wildcard	?	exactly one character	$ ls *.???

Square Brackets Wildcard	[xyz]	any letter in brackets	$ ls *.[xyz]?
			$ ls [a-cst]*
			$ ls jones[0-9][a-z][A-Z]

New Trick: Wildcards

Wildcards	Symbol	Meaning	Examples
Star Wildcard	*	string with arbitrary length	$ ls *.txt $ rm *exp*.dat
Question Mark Wildcard	?	exactly one character	$ ls *.???

Square Brackets Wildcard	[xyz]	any letter in brackets	$ ls *.[xyz]?
			$ ls [a-cst]*
			$ ls jones[0-9][a-z][A-Z]

Practice:

3. copy all files with .dat format in folder prac1 to folder prac2
4. remove all files in prac1
5. remove folder prac1

TOO MANY COMMANDS!!!

TOO MANY COMMANDS!!!

How to remember?

TOO MANY COMMANDS!!!

How to remember?

- **DON'T try to remember, just use**

TOO MANY COMMANDS!!!

How to remember?

- **DON'T try to remember, just use**
- Just like learning how to type on keyboard

TOO MANY COMMANDS!!!

How to remember?

- **DON'T try to remember, just use**
- Just like learning how to type on keyboard
- If forget the command and parameters, just search it

TOO MANY COMMANDS!!!

How to remember?

- **DON'T try to remember, just use**
- Just like learning how to type on keyboard
- If forget the command and parameters, just search it

Where to search?

TOO MANY COMMANDS!!!

How to remember?

- **DON'T try to remember, just use**
- Just like learning how to type on keyboard
- If forget the command and parameters, just search it

Where to search?

- $ man CMD # manual for CMD

TOO MANY COMMANDS!!!

How to remember?

- **DON'T try to remember, just use**
- Just like learning how to type on keyboard
- If forget the command and parameters, just search it

Where to search?

- $ man CMD # manual for CMD
- $ CMD --help # help info for CMD

TOO MANY COMMANDS!!!

How to remember?

- **DON'T try to remember, just use**
- Just like learning how to type on keyboard
- If forget the command and parameters, just search it

Where to search?

- $ man CMD # manual for CMD
- $ CMD --help # help info for CMD
- Google

Text Editors

Text Editors

- There are many text editors in Linux

Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two

Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community

Text Editors

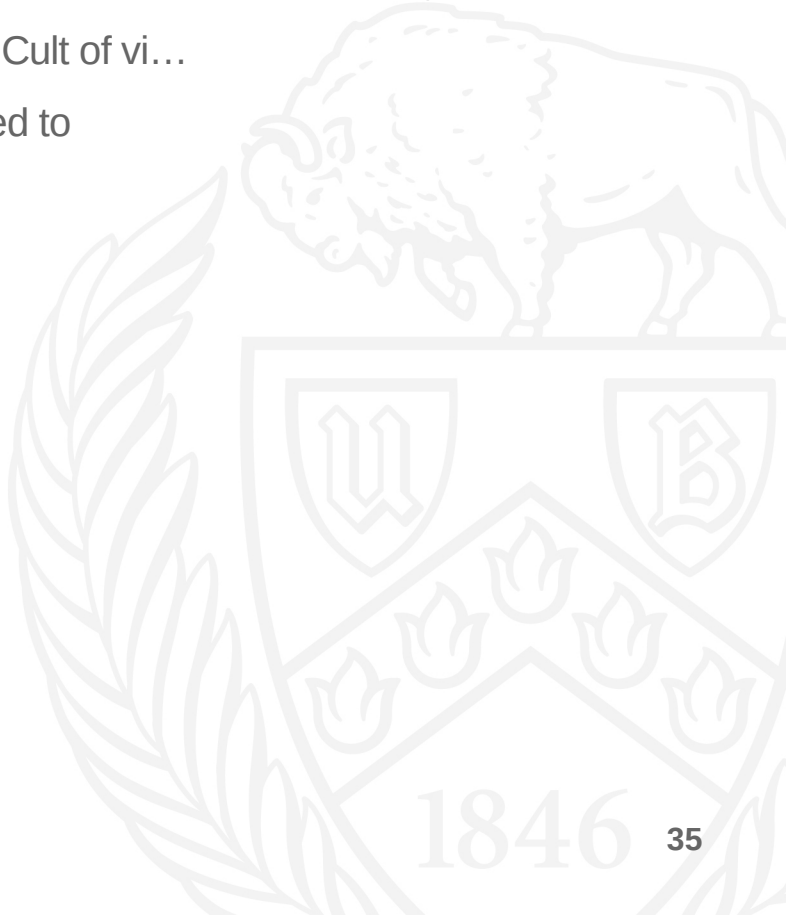
- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…

Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to

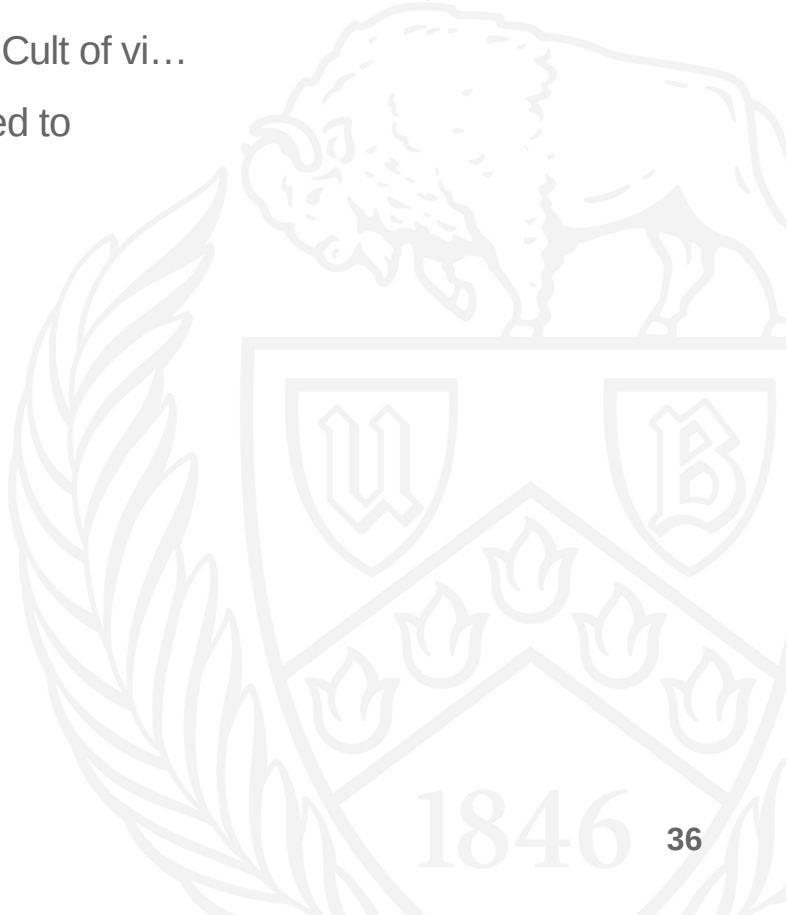
Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn



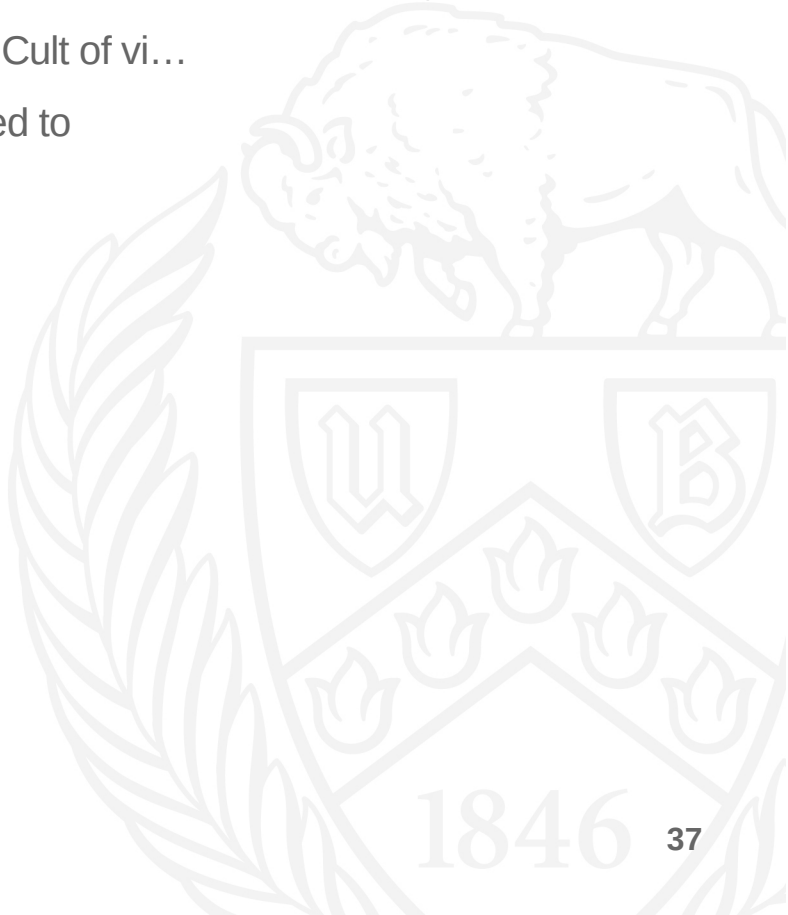
Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn
- Today we introduce nano



Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn
- Today we introduce nano
- In future sessions, vim will be introduced



Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn
- Today we introduce nano
- In future sessions, vim will be introduced
- If you are emacs supporers, don’t fight me…

Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn
- Today we introduce nano
- In future sessions, vim will be introduced
- If you are emacs supporers, don’t fight me…

```
$ nano                  # create new file
```

Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn
- Today we introduce nano
- In future sessions, vim will be introduced
- If you are emacs supporers, don’t fight me…

```
$ nano          # create new file
$ nano FILE     # create/open FILE
```

Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn
- Today we introduce nano
- In future sessions, vim will be introduced
- If you are emacs supporers, don’t fight me…

```
$ nano          # create new file
$ nano FILE     # create/open FILE
```

All the commands are in the bottom of the app, where “^” means ctrl

Text Editors

- There are many text editors in Linux
- Among all, vi and emacs are the most popular two
- Emacs vs vi debate was one of the original “holy wars” in free software community
- There are parody religions called Church of Emacs and Cult of vi…
- Both are heavyweight editors, take some time to get used to
- Nano on the contrary is the easiest to use and learn
- Today we introduce nano
- In future sessions, vim will be introduced
- If you are emacs supporers, don’t fight me…

```
$ nano          # create new file
$ nano FILE     # create/open FILE
```

All the commands are in the bottom of the app, where “^” means ctrl

Practice:

1. create a new file
2. write something
3. save as “first.conf” and close
4. open “first.conf”, do some changes
5. save and close

Well Done!

Basics:
know how to finish common tasks in CCR

- [x] Connect to CCR
- [ ] Filesystem Operations
- [ ] CCR Operations
- [ ] File Transfer

Well Done!

Basics:
know how to finish common tasks in CCR

- [x] Connect to CCR
- [x] Filesystem Operations
- [ ] CCR Operations
- [ ] File Transfer

Contents:

What is CCR

What is CCR

- A group of computers connected by network

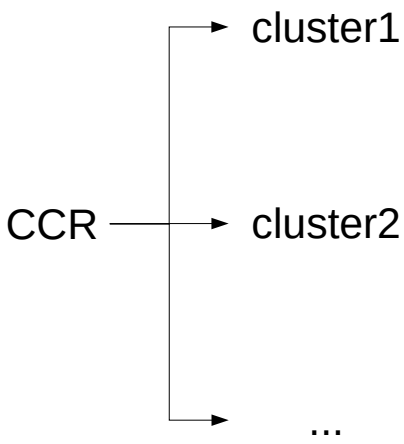
What is CCR

- A group of computers connected by network
- Form a more powerful machine

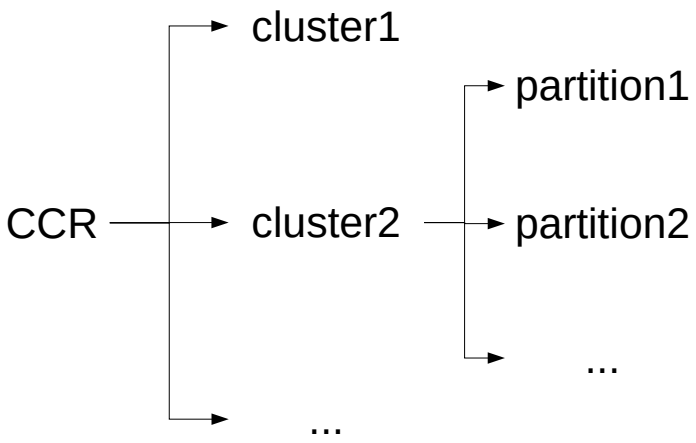
Logic Structure of CCR

CCR

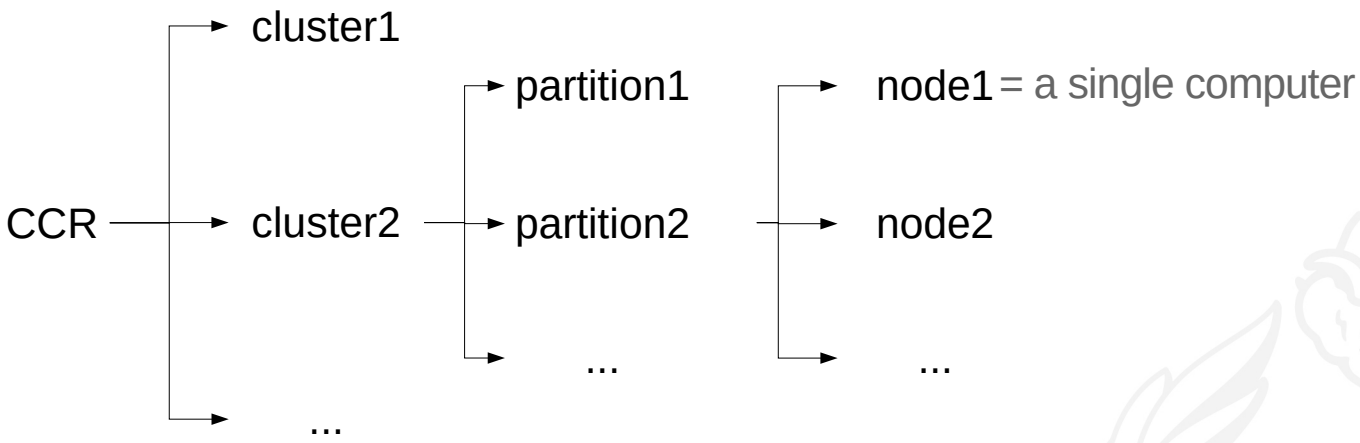
Logic Structure of CCR



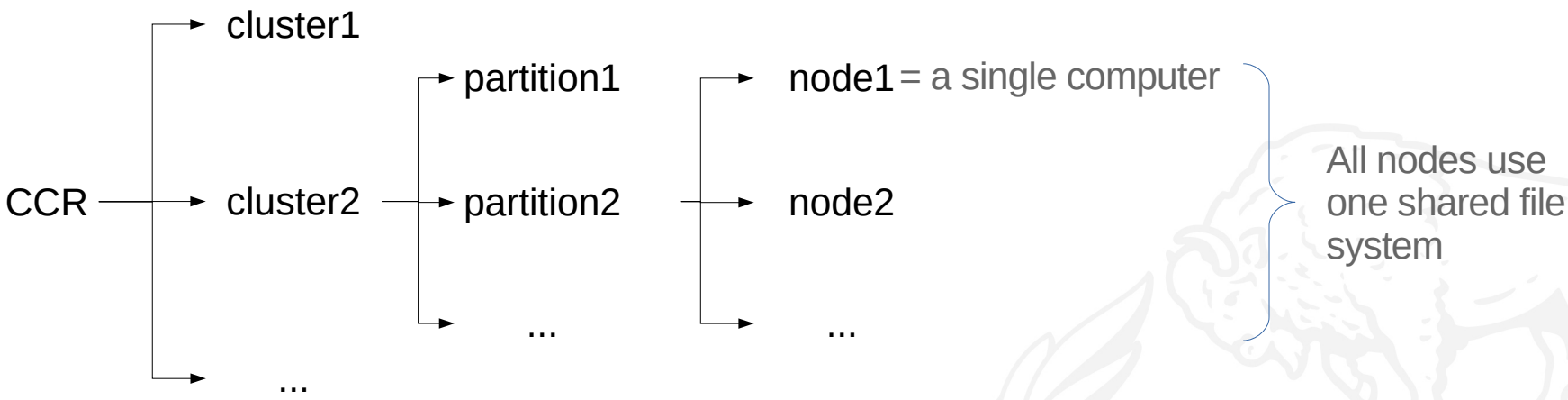
Logic Structure of CCR



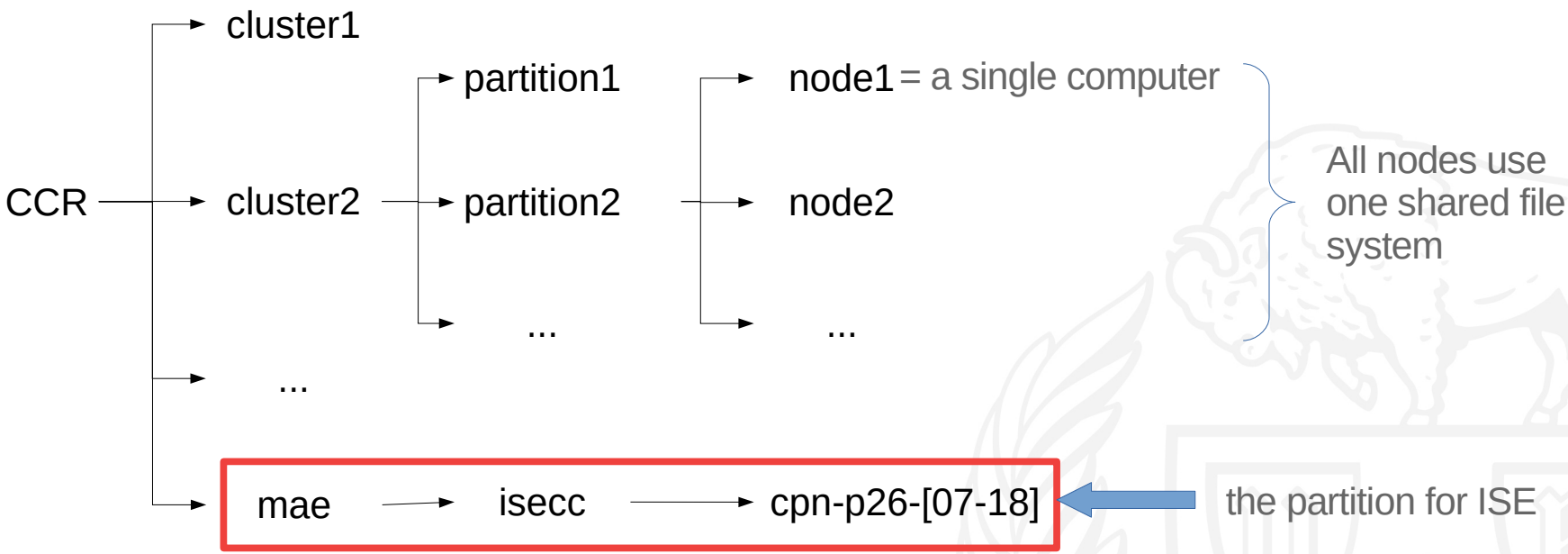
Logic Structure of CCR



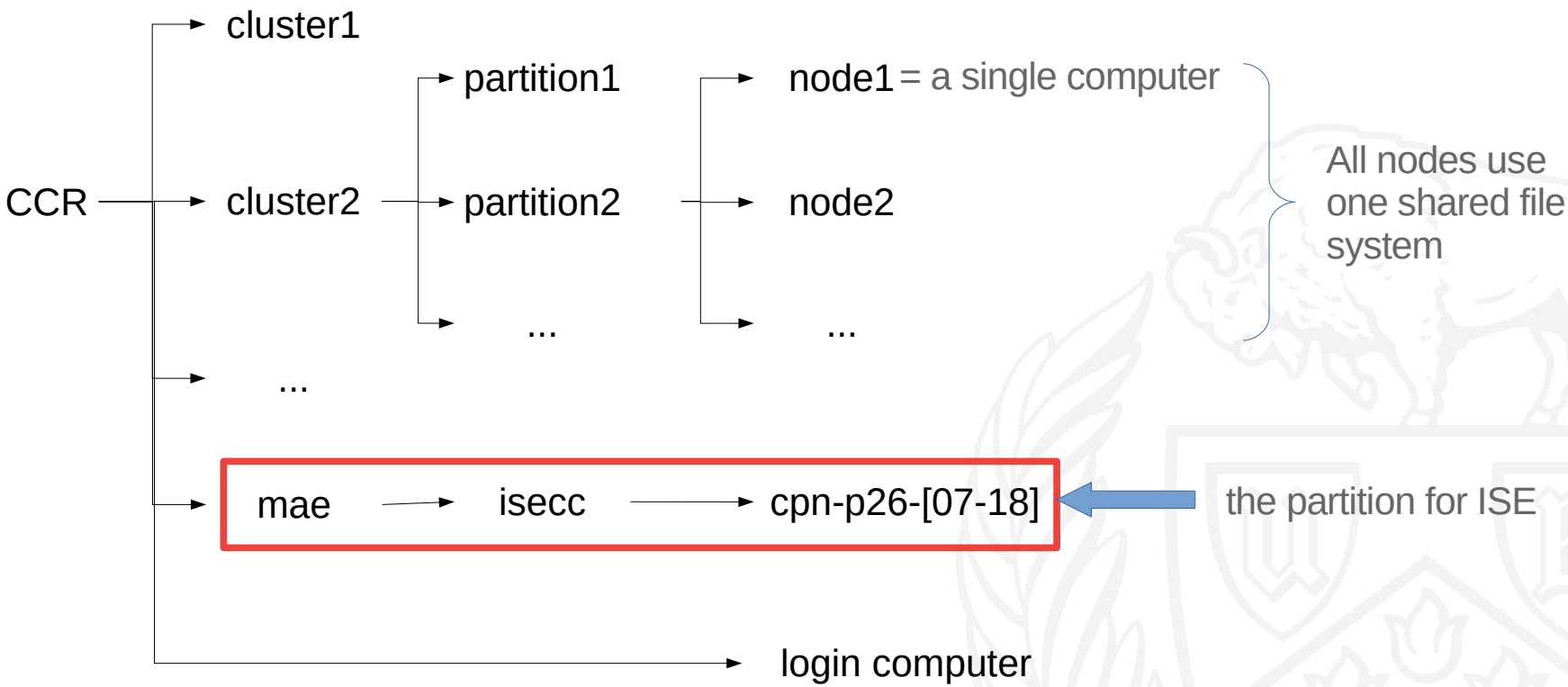
Logic Structure of CCR



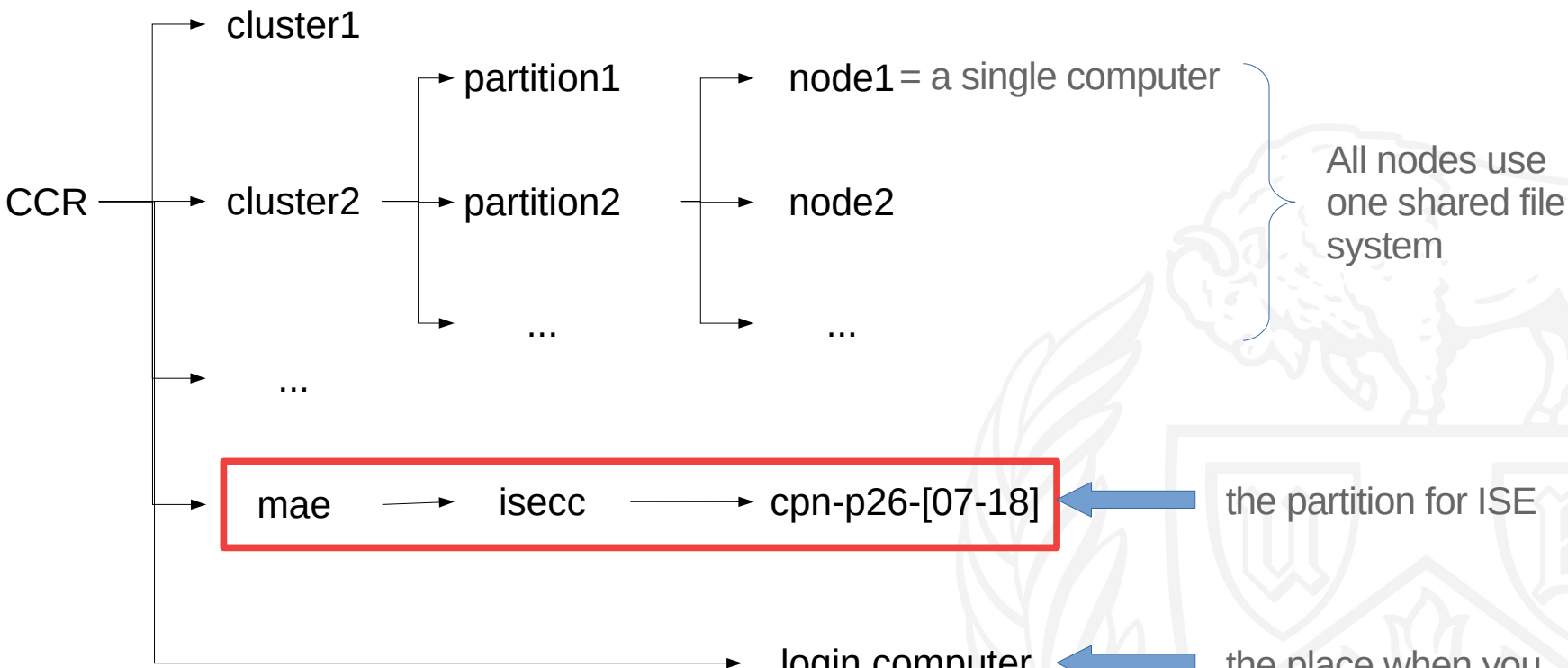
Logic Structure of CCR



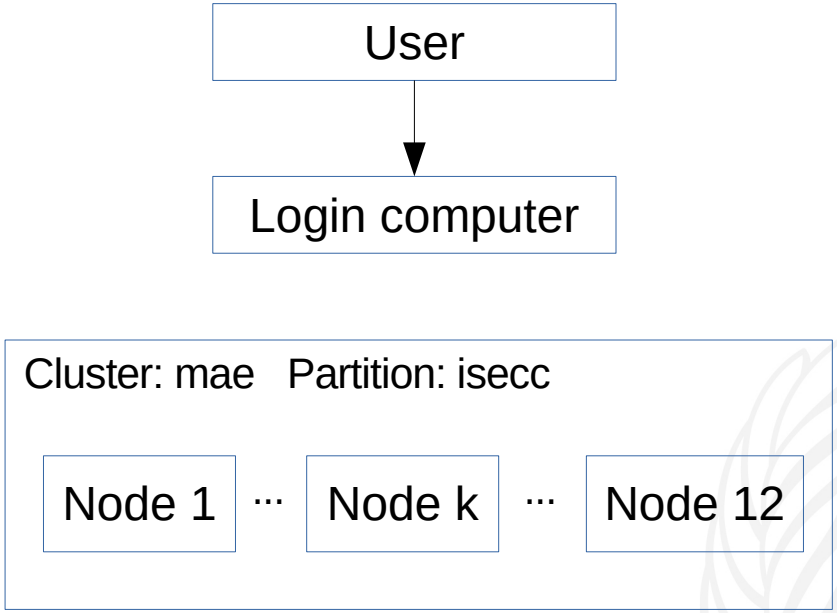
Logic Structure of CCR



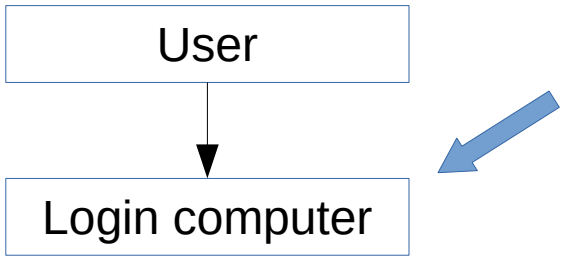
Logic Structure of CCR



Different Ways of Running Jobs in CCR

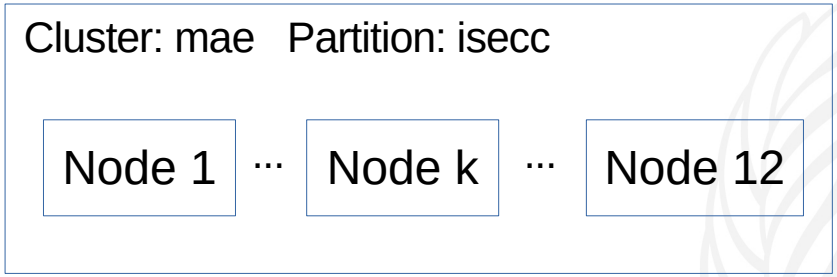


Different Ways of Running Jobs in CCR

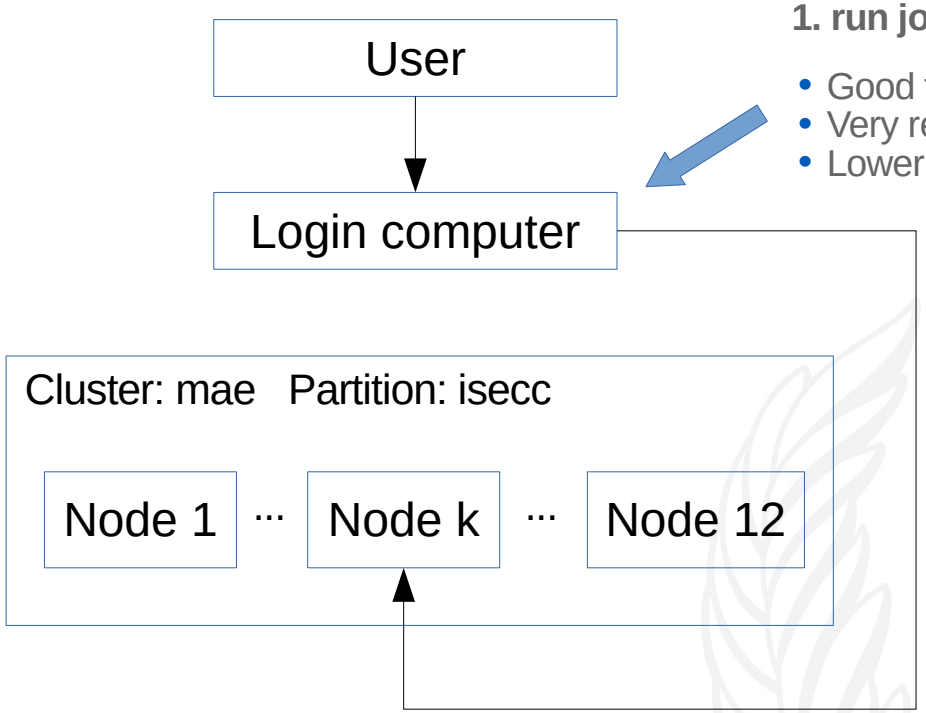


1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power



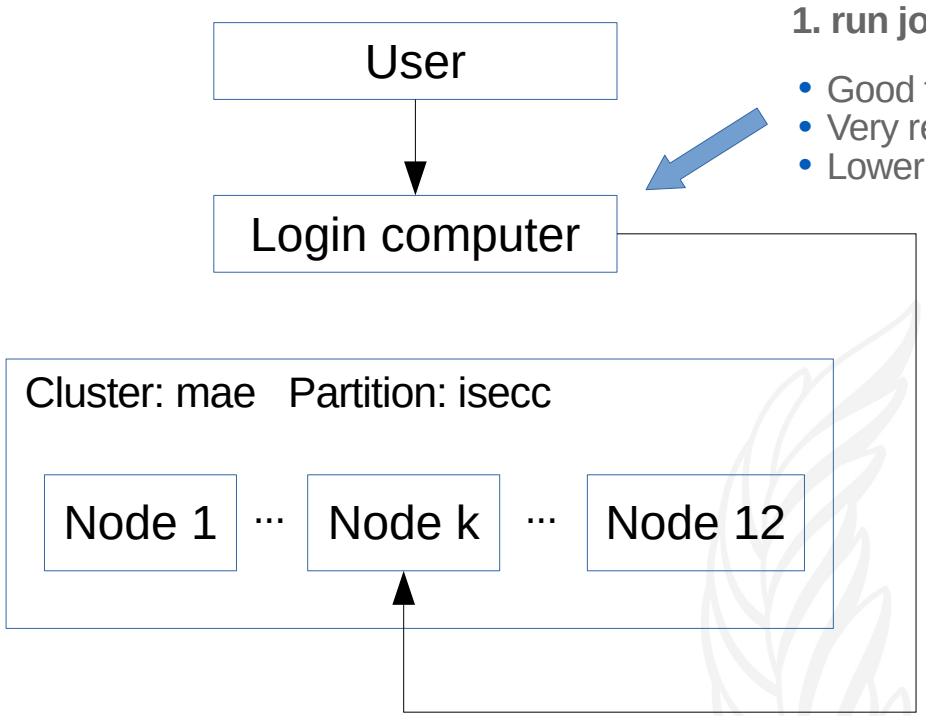
Different Ways of Running Jobs in CCR



1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

Different Ways of Running Jobs in CCR

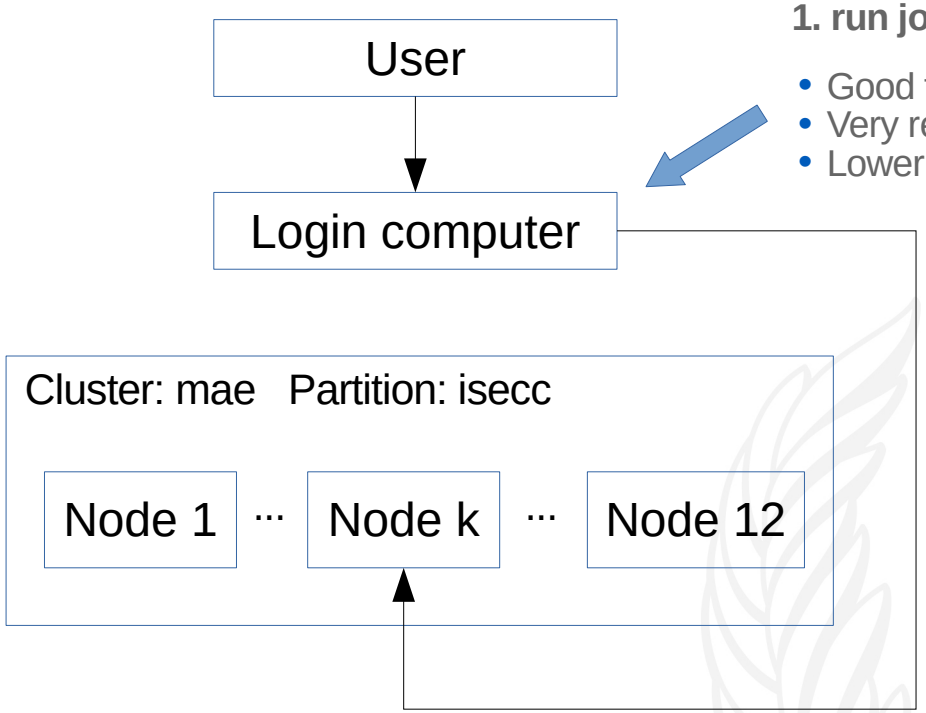


1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

Different Ways of Running Jobs in CCR



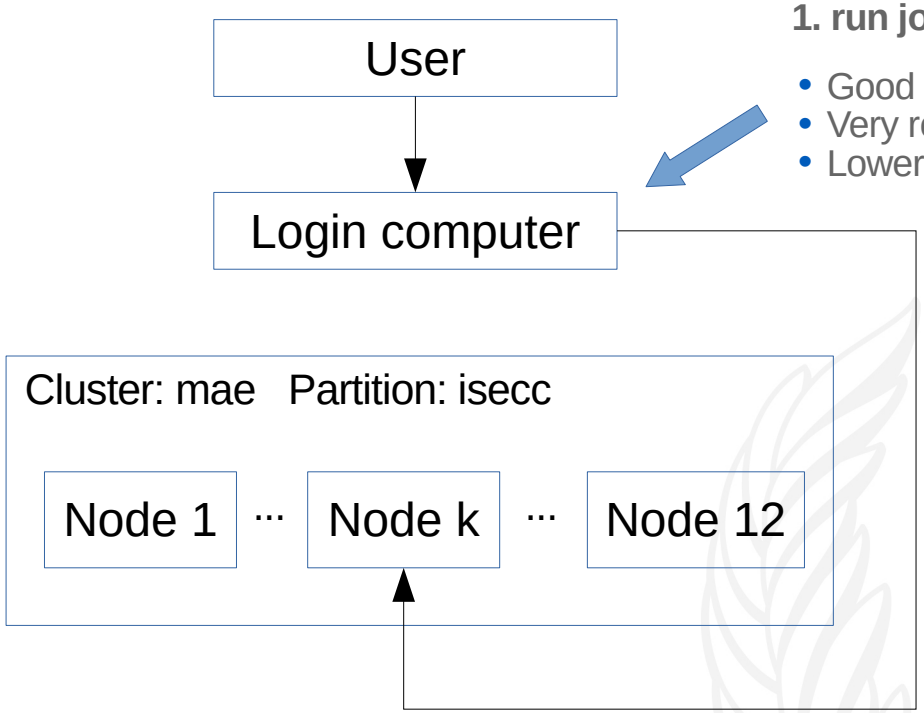
1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit

Different Ways of Running Jobs in CCR



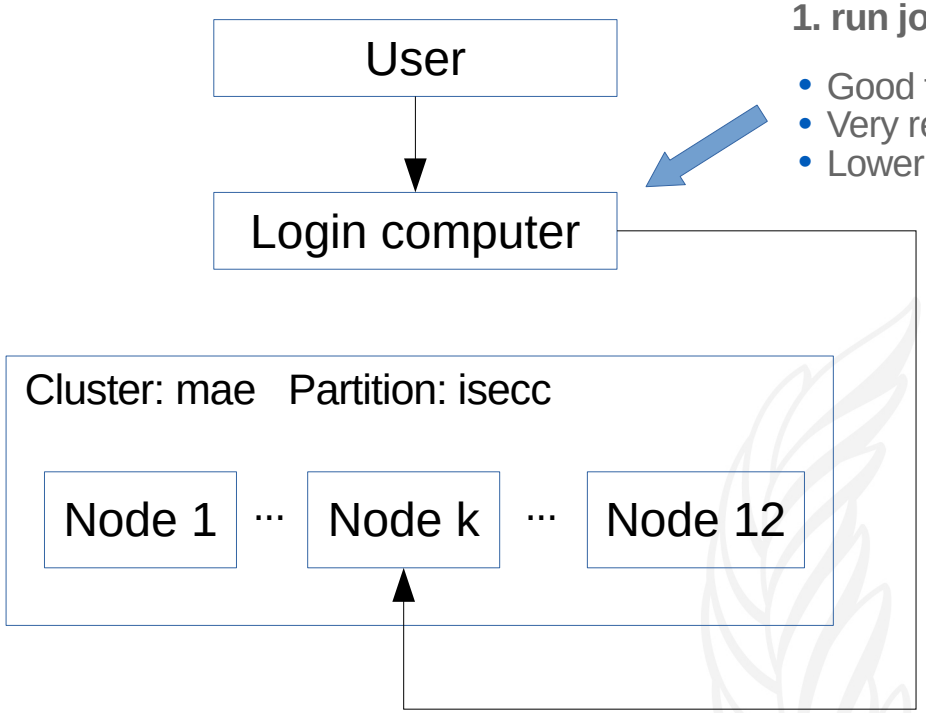
1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit
- Higher computation power

Different Ways of Running Jobs in CCR



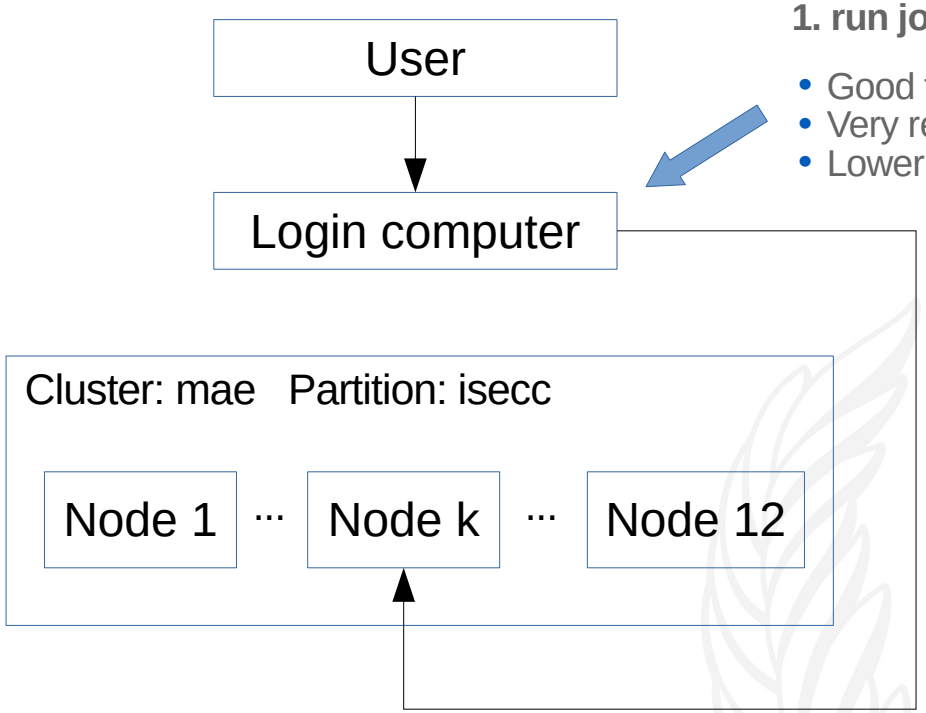
1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time

Different Ways of Running Jobs in CCR



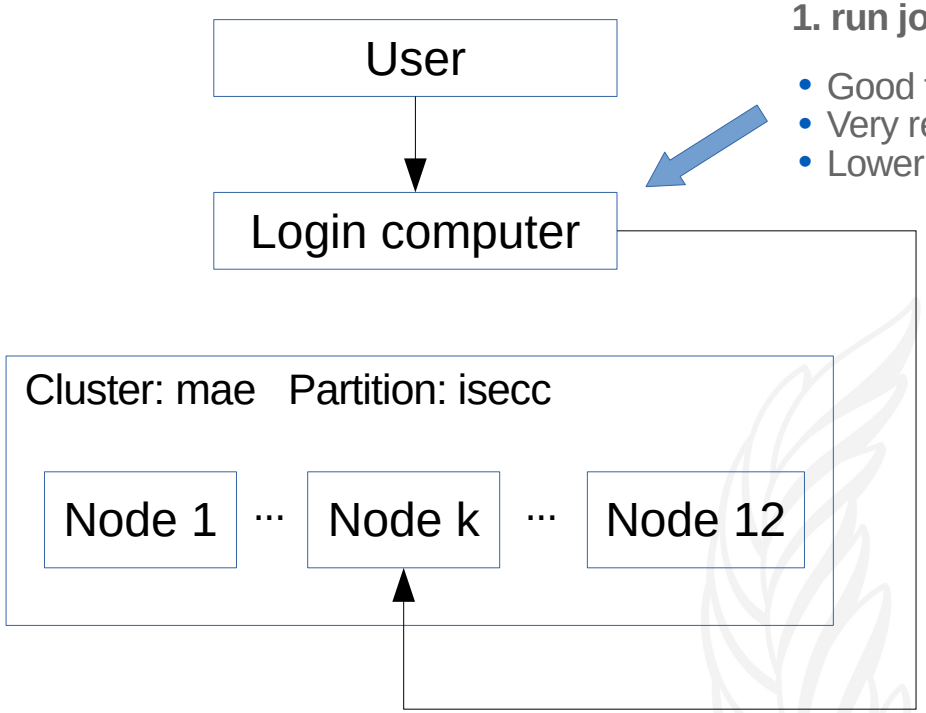
1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected

Different Ways of Running Jobs in CCR



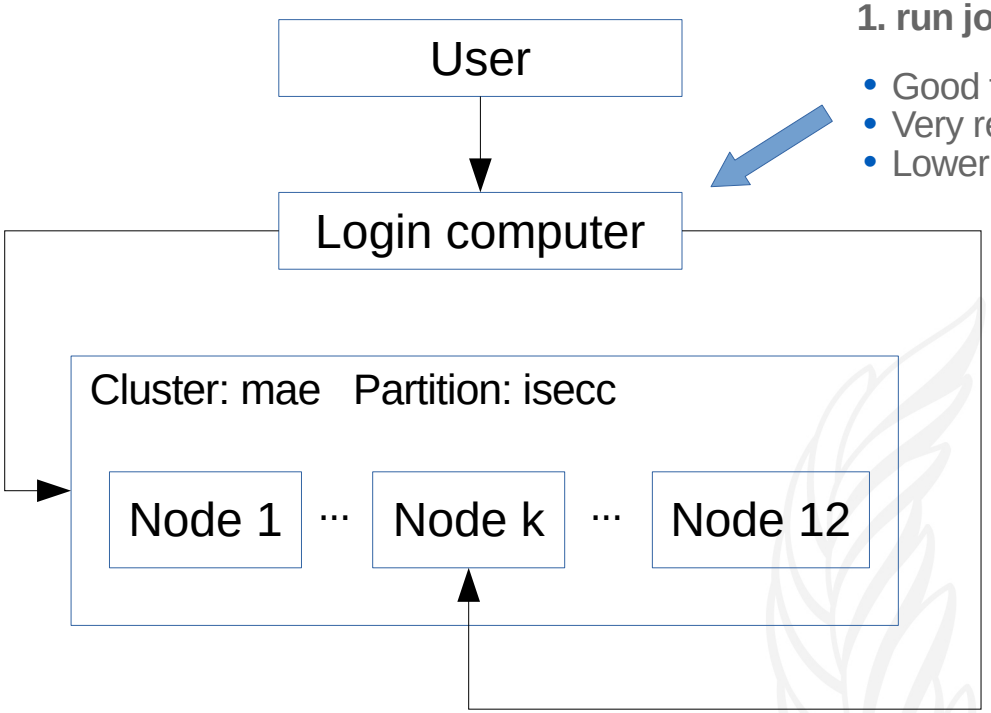
1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

Different Ways of Running Jobs in CCR



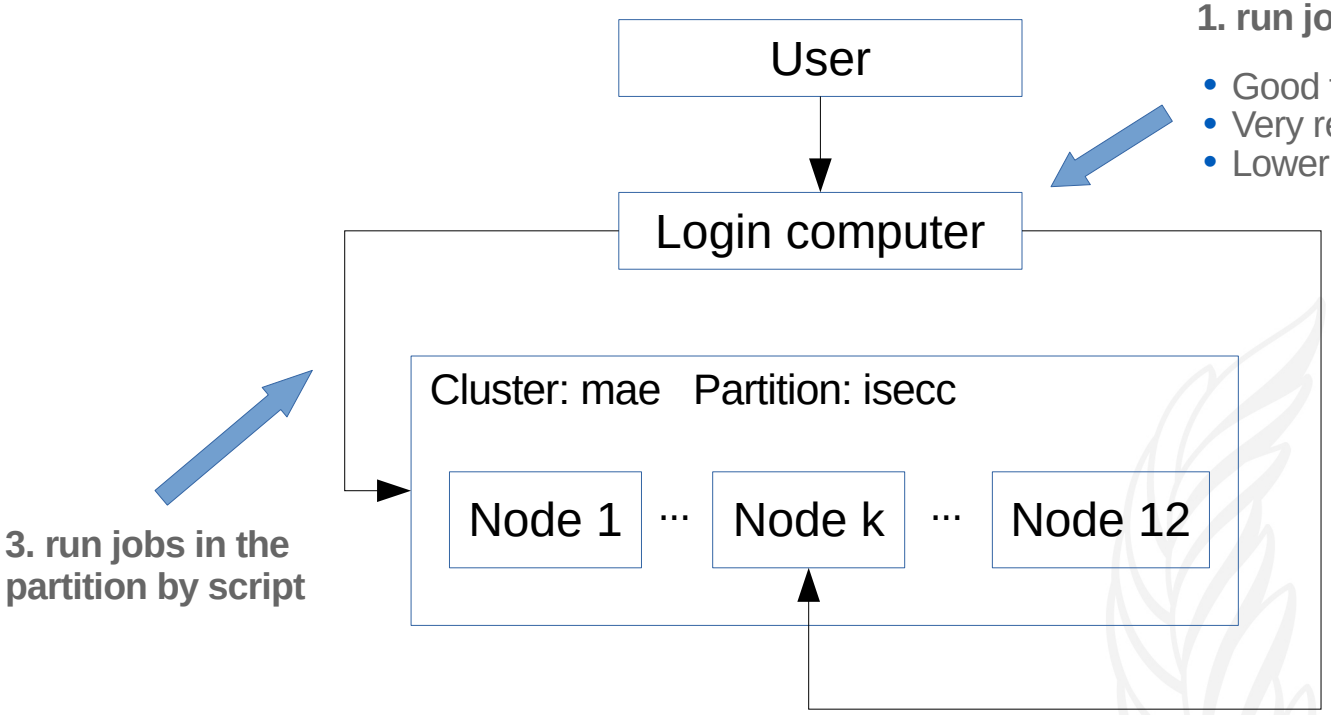
1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

Different Ways of Running Jobs in CCR



1. run jobs in the login computer

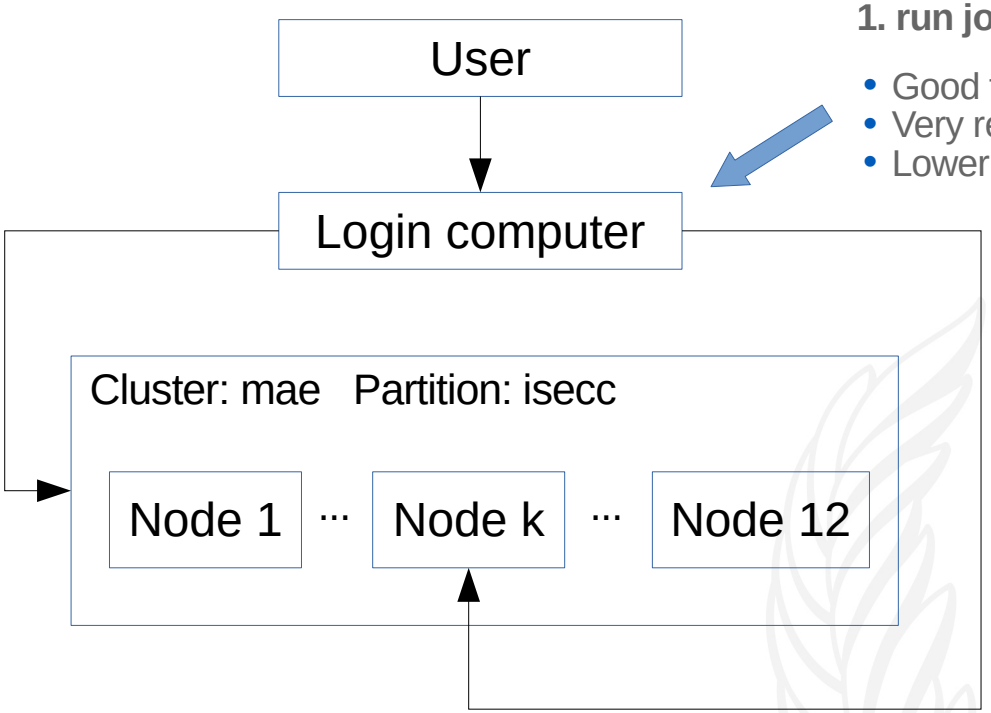
- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

3. run jobs in the partition by script

Different Ways of Running Jobs in CCR



1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

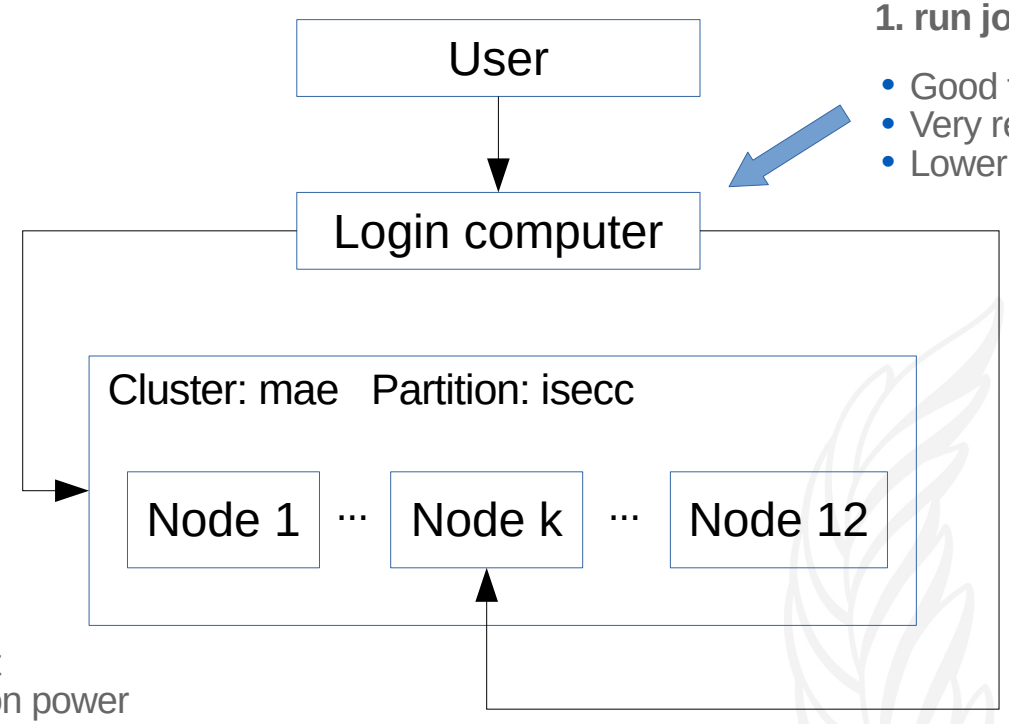
2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

3. run jobs in the partition by script

- Specify Time Limit

Different Ways of Running Jobs in CCR



1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

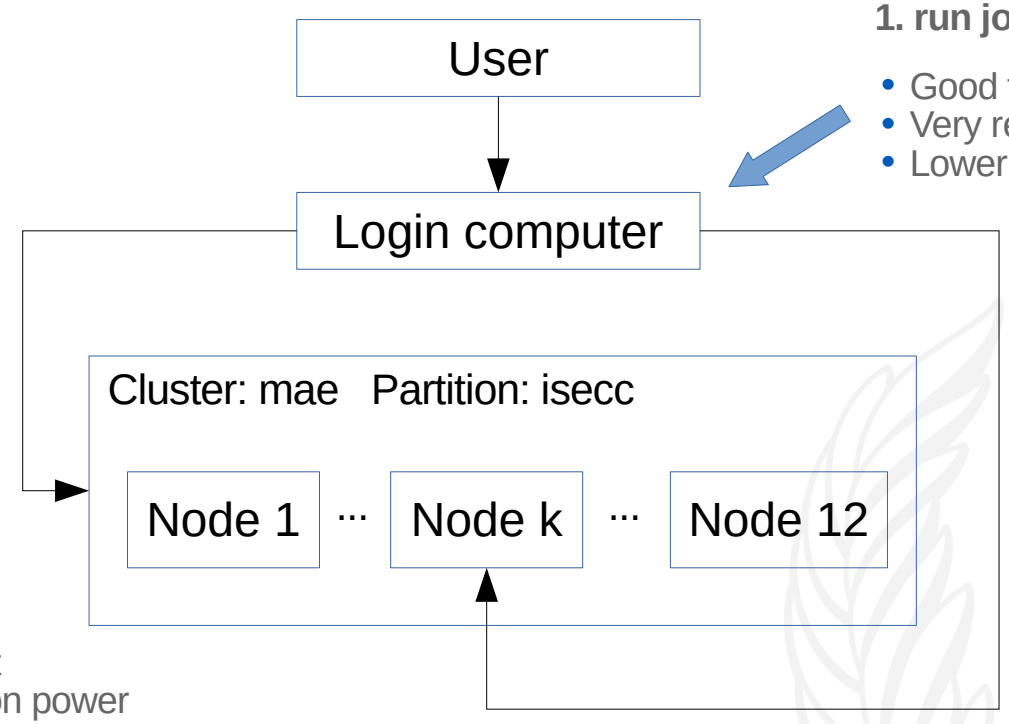
2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

3. run jobs in the partition by script

- Specify Time Limit
- Higher computation power

Different Ways of Running Jobs in CCR



1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

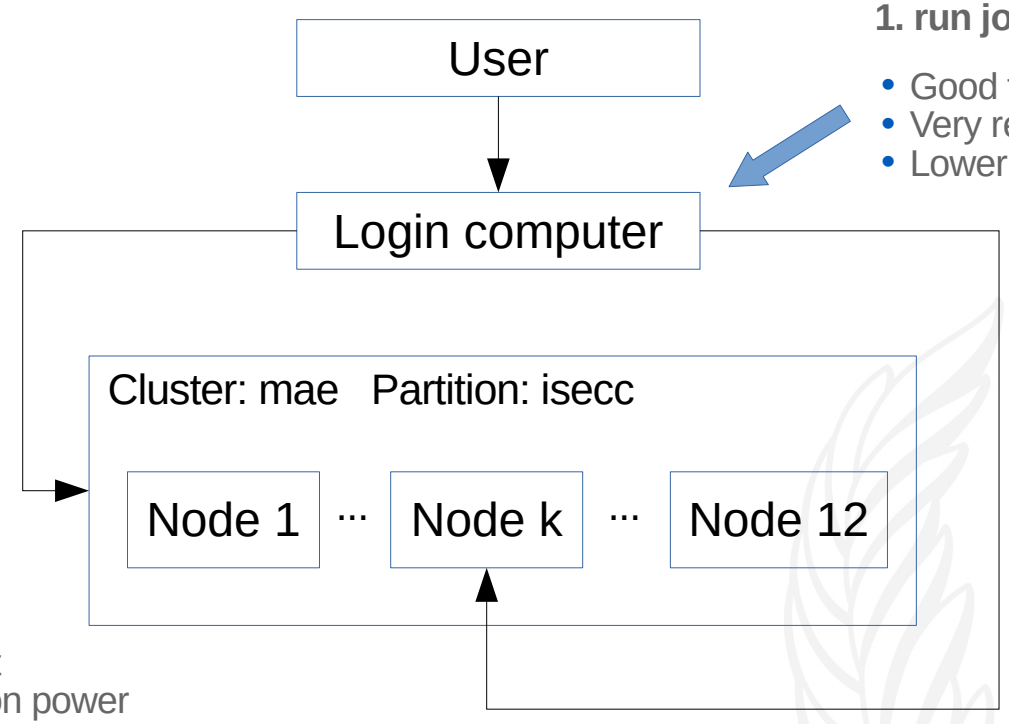
2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

3. run jobs in the partition by script

- Specify Time Limit
- Higher computation power
- Can submit an array of script

Different Ways of Running Jobs in CCR



1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

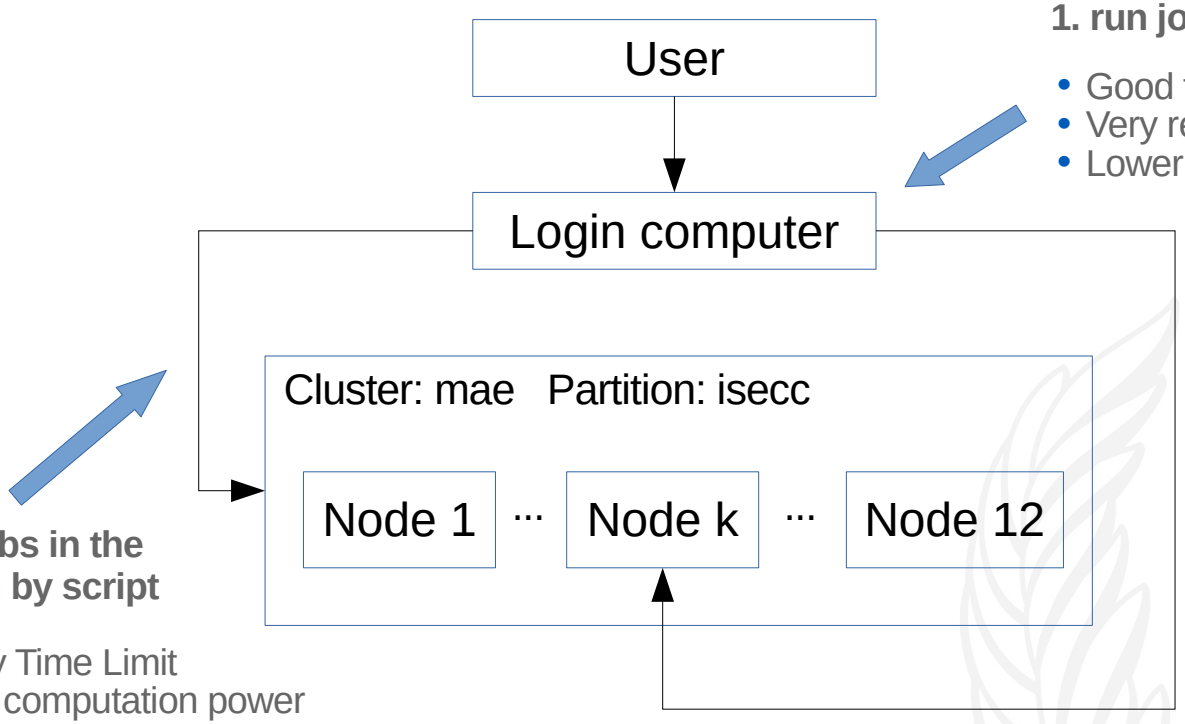
2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

3. run jobs in the partition by script

- Specify Time Limit
- Higher computation power
- Can submit an array of script
- Unaffected by the connection

Different Ways of Running Jobs in CCR



1. run jobs in the login computer

- Good testing environment
- Very restrictive time limit
- Lower computation power

2. run jobs in a node interactively

- Specify time limit
- Higher computation power
- Run single script a time
- Cancel job when disconnected
- Special testing environment

3. run jobs in the partition by script

- Specify Time Limit
- Higher computation power
- Can submit an array of script
- Unaffected by the connection
- Good running environment
- No interactive environment

Hello World

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print(‘Hello World!’)

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

2. Run HelloWorld.py in a node interactively

Step 1: jump to a computation node (computer):
$ fisbatch --clusters=mae --partition=isecc --nodes=1
--ntasks-per-node=12 --time=00:05:00 --exclusive
check indeed in a different environment: $ uname -n
Step 2: $ python HelloWorld.py

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

2. Run HelloWorld.py in a node interactively

Step 1: jump to a computation node (computer):
$ fisbatch --clusters=mae --partition=isecc --nodes=1
--ntasks-per-node=12 --time=00:05:00 --exclusive
check indeed in a different environment: $ uname -n
Step 2: $ python HelloWorld.py

3. Run HelloWorld.py in isecc partition by script

Step 1: create a batch script: task.bat
Step 2: $ sbatch task.bat

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

2. Run HelloWorld.py in a node interactively

Step 1: jump to a computation node (computer):
$ fisbatch --clusters=mae --partition=isecc --nodes=1 --ntasks-per-node=12 --time=00:05:00 --exclusive
check indeed in a different environment: $ uname -n
Step 2: $ python HelloWorld.py

3. Run HelloWorld.py in isecc partition by script

Step 1: create a batch script: task.bat
Step 2: $ sbatch task.bat

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorld"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
##SBATCH --exclude=cpn-p26-[13-18]
##SBATCH --array=1-4

python ./HelloWorld.py
```

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

2. Run HelloWorld.py in a node interactively

Step 1: jump to a computation node (computer):
$ fisbatch --clusters=mae --partition=isecc --nodes=1 --ntasks-per-node=12 --time=00:05:00 --exclusive
check indeed in a different environment: $ uname -n
Step 2: $ python HelloWorld.py

3. Run HelloWorld.py in isecc partition by script

Step 1: create a batch script: task.bat
Step 2: $ sbatch task.bat

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorld"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
##SBATCH --exclude=cpn-p26-[13-18]
##SBATCH --array=1-4

python ./HelloWorld.py
```

- "#!" means this line is a shebang, always first row

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

2. Run HelloWorld.py in a node interactively

Step 1: jump to a computation node (computer):
$ fisbatch --clusters=mae --partition=isecc --nodes=1 --ntasks-per-node=12 --time=00:05:00 --exclusive
check indeed in a different environment: $ uname -n
Step 2: $ python HelloWorld.py

3. Run HelloWorld.py in isecc partition by script

Step 1: create a batch script: task.bat
Step 2: $ sbatch task.bat

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorld"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
##SBATCH --exclude=cpn-p26-[13-18]
##SBATCH --array=1-4

python ./HelloWorld.py
```

- "#!" means this line is a shebang, always first row
- "#SBATCH" means this line is a sbatch option

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

2. Run HelloWorld.py in a node interactively

Step 1: jump to a computation node (computer):
$ fisbatch --clusters=mae --partition=isecc --nodes=1 --ntasks-per-node=12 --time=00:05:00 --exclusive
check indeed in a different environment: $ uname -n
Step 2: $ python HelloWorld.py

3. Run HelloWorld.py in isecc partition by script

Step 1: create a batch script: task.bat
Step 2: $ sbatch task.bat

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorld"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
##SBATCH --exclude=cpn-p26-[13-18]
##SBATCH --array=1-4

python ./HelloWorld.py
```

- "#!" means this line is a shebang, always first row
- "#SBATCH" means this line is a sbatch option
- in other cases, "#" means this line is a comment

Hello World

0. In the login computer, create file: HelloWorld.py

Step 1: use nano, add one line: print('Hello World!')

1. Run HelloWorld.py in the login computer

Step1: $ python HelloWorld.py

2. Run HelloWorld.py in a node interactively

Step 1: jump to a computation node (computer):
$ fisbatch --clusters=mae --partition=isecc --nodes=1 --ntasks-per-node=12 --time=00:05:00 --exclusive
check indeed in a different environment: $ uname -n
Step 2: $ python HelloWorld.py

3. Run HelloWorld.py in isecc partition by script

Step 1: create a batch script: task.bat
Step 2: $ sbatch task.bat

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorld"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
##SBATCH --exclude=cpn-p26-[13-18]
##SBATCH --array=1-4

python ./HelloWorld.py
```

- "#!" means this line is a shebang, always first row
- "#SBATCH" means this line is a sbatch option
- in other cases, "#" means this line is a comment
- **IMPORTANT:** create the folder for output & error

Using Array in Script

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorldArray"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
#SBATCH --exclude=cpn-p26-[13-18]
#SBATCH --array=1-4
##SBATCH --array=1,3-4

python ./insts/hw${SLURM_ARRAY_TASK_ID}.py
```

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorldArray"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
#SBATCH --exclude=cpn-p26-[13-18]
#SBATCH --array=1-4
##SBATCH --array=1,3-4

python ./insts/hw${SLURM_ARRAY_TASK_ID}.py
```

- ${VAR} is a variable

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorldArray"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
#SBATCH --exclude=cpn-p26-[13-18]
#SBATCH --array=1-4
##SBATCH --array=1,3-4

python ./insts/hw${SLURM_ARRAY_TASK_ID}.py
```

- ${VAR} is a variable
- --array=1-4 is an array of values to replace

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorldArray"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
#SBATCH --exclude=cpn-p26-[13-18]
#SBATCH --array=1-4
##SBATCH --array=1,3-4

python ./insts/hw${SLURM_ARRAY_TASK_ID}.py
```

- ${VAR} is a variable
- --array=1-4 is an array of values to replace
- array representation example:

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorldArray"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
#SBATCH --exclude=cpn-p26-[13-18]
#SBATCH --array=1-4
##SBATCH --array=1,3-4

python ./insts/hw${SLURM_ARRAY_TASK_ID}.py
```

- ${VAR} is a variable
- --array=1-4 is an array of values to replace
- array representation example:
  --array=1,5-9,11,14-18,32-99

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorldArray"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
#SBATCH --exclude=cpn-p26-[13-18]
#SBATCH --array=1-4
##SBATCH --array=1,3-4

python ./insts/hw${SLURM_ARRAY_TASK_ID}.py
```

- ${VAR} is a variable
- --array=1-4 is an array of values to replace
- array representation example:
  --array=1,5-9,11,14-18,32-99
- may occupy all 12 nodes

Using Array in Script

!!NOTE: Only in method 3, using sbatch with script

Step 1: create a list of instances whose names only differ in number, example:

hw1.py, hw2.py, … , hw4.py

Step 2: edit the script tasks.sh

Step 3: $ sbatch tasks.sh

```
#!/bin/bash
#SBATCH --clusters=mae
#SBATCH --partition=isecc
#SBATCH --qos=isecc
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --time=00:05:00
#SBATCH --mem=125000
#SBATCH --job-name="HelloWorldArray"
#SBATCH --mail-user=ningjiwe@buffalo.edu
#SBATCH --mail-type=ALL
#SBATCH --output=./console/console_%A_%a.out
#SBATCH --error=./console/console_%A_%a.err
#SBATCH --exclude=cpn-p26-[13-18]
#SBATCH --array=1-4
##SBATCH --array=1,3-4

python ./insts/hw${SLURM_ARRAY_TASK_ID}.py
```

- ${VAR} is a variable
- --array=1-4 is an array of values to replace
- array representation example:
  --array=1,5-9,11,14-18,32-99
- may occupy all 12 nodes
- use --exclude to specify nodes that **DON'T** use