Introduction on Slurm Job Submission

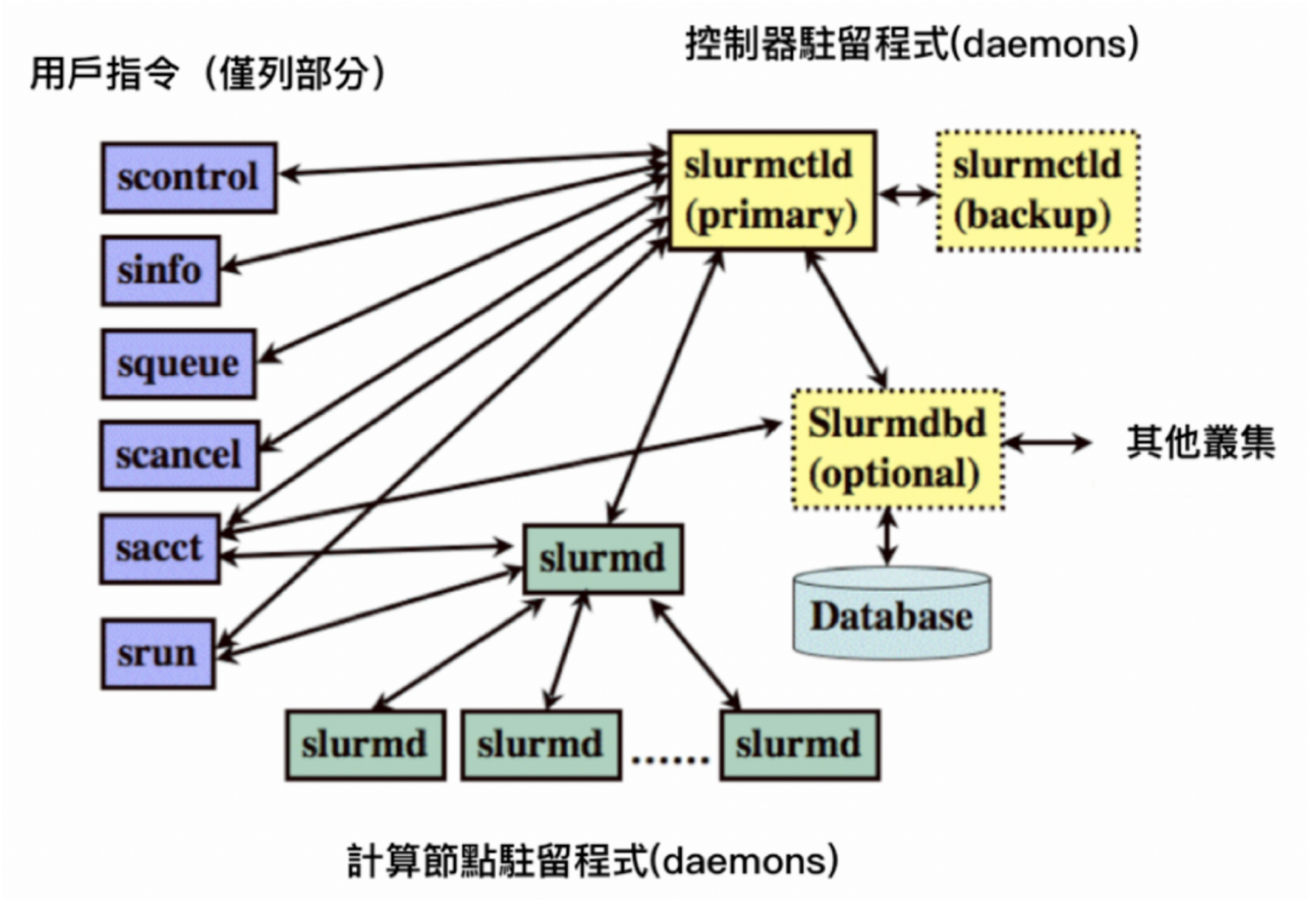
Rudy 陳侑廷
rudy.chen@twgrid.org
Academia Sinica Grid-computing Centre (ASGC)

Introduction of Slurm

Overview

Slurm is an

- Open source
- Fault-tolerant
- Highly scalable cluster management
- Job scheduling system



Introduction of Slurm - Computing

- Computing Machine Specifications Computing Nodes:

Cluster	CPU	Nodes	RAM-Per-Node	Cores-Per-Node	Total Cores
FDR5	Intel® Xeon® CPU E5-2650 v4@2.20GHz	92	128 GB	24	2208
HDR1	AMD Rome 7662 @2.0GHz	6	1.5 TB	128	768

GPU Cluster	GPU Model	Nodes	GPU-Boards (each node)
GPU-A100	NVIDIA A100	2	8
GPU-V100	NVIDIA V100	5	8

User Interfaces (Login Nodes)

Login into Slurm User Interface

- The user interface node for slurm are:
  slurm-ui.twgrid.org
- Login in user interface:
  ssh jack@slurm-ui.twgrid.org
- You will be prompted with the relative information of your account when login into the slurm user interfaces

- For Windows users can download and install SSH client software (e.g. PuTTY, MobaXterm, VScode, etc.).
- For macOS users, you can open the built-in terminal directly.

Basic Usage of Slurm System

Basic Usage of Slurm System

- Query cluster information

  sinfo

- Query the jobs submitted by you

  sacct or sacct -u jack

- Submit your job with bash script (recommended)

  sbatch your_script.sh

- Submit your job (binary executable) with srun

  srun your_program arg1 arg2

Basic Usage of Slurm System

- Show queue information

  squeue
- Show your job in the queue

  squeue -u jack
- Show the detailed job information

  scontrol show job your_jobid
- Cancel your job

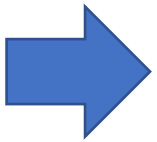
  scancel your_jobid

```
(base) [rudy@slurm-ui02 rudy_slurmcode]$ scontrol show job 3877007
   JobId=3877007 JobName=Rudy_test
   UserId=rudy(5013) GroupId=ASGC(525) MCS_label=N/A
   Priority=23159 Nice=0 Account=asgc QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:08 TimeLimit=00:30:00 TimeMin=N/A
   SubmitTime=2023-11-02T07:02:25 EligibleTime=2023-11-02T07:02:25
   AccrueTime=2023-11-02T07:02:25
   StartTime=2023-11-02T07:02:26 EndTime=2023-11-02T07:32:26 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2023-11-02T07:02:26
   Partition=qdr6 AllocNode:Sid=slurm-ui02:15411
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=as-wn[629-630]
   BatchHost=as-wn629
   NumNodes=2 NumCPUs=40 NumTasks=40 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=40,node=2,billing=40
   Socks/Node=* NtasksPerN:B:S:C=20:0:*:* CoreSpec=*
   MinCPUsNode=20 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/dicos_ui_home/rudy/rudy_slurmcode/test_QDR6.sh
   WorkDir=/dicos_ui_home/rudy/rudy_slurmcode
   StdErr=/dicos_ui_home/rudy/rudy_slurmcode/job.%J.err
   StdIn=/dev/null
   StdOut=/dicos_ui_home/rudy/rudy_slurmcode/job.%J.out
   Power=
   NtasksPerTRES:0
```

Partitions/Queues of Slurm

- Slurm Partitions (Queues)
- The default queue is “short”. Users could submit to different partitions by assigning partition parameters, e.g.

  sbatch -p large myscript.sh



PARTITION	Nodes	GPU-Boards (each node)	can use at a time
a100	2	8	4 (each node)
a100_long			
a100_short			
v100	5	8	4 (each node)
v100_long			
v100_short			

PARTITION	TIMELIMIT	NODELIST
Large (MinNodes=2)	14-00:00:0	smwn[001-092]
long_serial	14-00:00:0	smwn[081-090]
short	3:00:00	smwn[001-092]
moderate_serial	2-00:00:00	
short_serial	4:00:00	
development	1:00:00	smwn[091-092]
v100	5-00:00:00	hp-teslav[01-05]
v100_short	6:00:00	
v100_long	7-00:00:00	
a100	5-00:00:00	hp-teslaa[01,03]
a100_long	7-00:00:00	
a100_short	6:00:00	
amd	5-00:00:00	hpa-wn[01-04], sma-wn[01-02]
amd_short	4:00:00	
amd_devel	1:00:00	

Environment Modules

Environment Modules Introduction

- In DiCOS Slurm system, we have environment modules installed in user interfaces and worker nodes
- Detailed information please refer to the original document:
  https://modules.readthedocs.io/en/latest/
- Environment-modules help user to setup environment and environment variables properly for specific software environments
  - User doesn’t need to worry about the complex settings of the environments

Basic Usage of Environment Modules

- Show available modules in slurm-ui

```
$ module avail
------ /cvmfs/cvmfs.grid.sinica.edu.tw/hpc/modules/modulefiles/Core ---------

aomp/17.0-2          app/anaconda3/4.10.3  app/binutils/2.35.2  app/git/2.37.1  app/make/4.3          app/R/4.0.5
app/anaconda3/4.9.2  app/anaconda3/4.12.0  app/cmake/3.20.3     app/julia/1.8.0 app/paraview/5.8.0    app/R/4.2.1
app/root/6.24        gcc/4.8.5             gcc/9.3.0            gcc/10.3.0      gcc/11.1.0            gcc/12.1.0
intel/2017           intel/2020            nvhpc_sdk/20.11      python/3.9.5    pgi/20.11
```

- Load module

```
module load intel/2020
```

- Unload module

```
module unload intel/2020
```

- Show currently loaded modules

```
module list
```

- Unload all loaded modules

```
module purge
```

You can use ml instead of the above module commands

Python, Compilation and MPI Environment

Python

- The default system python on CentOS 7 is python 2.7.4
- Use python 3, please consider using anaconda with python3 first

  module load app/anaconda3/4.12.0
- Install additional applications

  → /ceph/work/<groupname>/
- Before installing a special python package, use the virtual environment:

```
wget https://repo.anaconda.com/archive/Anaconda3-2023.07-1-Linux-x86_64.sh
bash Anaconda3-2023.07-1-Linux-x86_64.sh
eval "$(/ceph/work/<groupname>/anaconda3/bin/conda shell.bash hook)"
conda create --name <myenv>      ## Create a virtual environment called myenv.
conda env list                   ## List the current state of the virtual environment.
conda activate <myenv>           ## Starting a new virtual environment.
conda install <your_package>     ## Install the required packages in this virtual environment.
conda deactivate                 ## Leaving the Virtual Environment.
```

Compilation

- Intel compiler

  module load intel/2020

- AMD compiler

  module load aomp/17.0-2

  It can be used on our HDR1 (hpa-wn[01-04], sma-wn[01-02])

- GCC

  module load gcc/12.1.0

- nvidia development kit (nvcc, for GPU program development)

  module load nvhpc_sdk/20.11

MPI

- Load compiler first, e.g. intel compiler
  - module load intel/2020
  - Or source /cvmfs/cvmfs.grid.sinica.edu.tw/hpc/compiler/scripts/centos7_intel_2020.sh
- Load different MPI implementation
  - mpich

    module load mpich
  - openmpi

    module load openmpi/4.1.0
  - mvapich2

    module load mvapich2

```
(base) [rudy@slurm-ui02 Hands-on_SLURM_2023]$ module avail
------------------------------------------------------------------------------------------------------------------------ /cvmfs/cvmfs.gr
lammaps/jct/3Mar2020  lammps/jct/3Mar2020  mpich/3.4.1  mvapich2/2.3.5  openmpi/2.1.6  openmpi/3.1.6  openmpi/4.1.0

------------------------------------------------------------------------------------------------------------------------ /cvmfs/c
aomp/17.0-2          app/anaconda3/4.10.3  app/binutils/2.35.2  app/git/2.37.1   app/make/4.3        app/R/4.0.5  app/r
app/anaconda3/4.9.2  app/anaconda3/4.12.0  app/cmake/3.20.3     app/julia/1.8.0  app/paraview/5.8.0  app/R/4.2.1  gcc/4

Key:
<module-tag>  <L>=loaded
```

Slurm Job Submission Examples - Hands on

Preparation

- Help me download the code that will be used first.

  git clone https://github.com/ASGCOPS/Hands-on_SLURM_2023.git

- Go to that folder.

  cd Hands-on_SLURM_2023

- Change Permission

  chmod 755 stress

Example 1 - Simple Job Submission (Hello World)

- Prepare a user defined shell script hello_world.sh

- Submit the job with sbatch

  sbatch hello_world.sh

```
#!/bin/bash

date
echo "Hello World DiCOS Users!"
hostname
```

Example 2 - Submit a MCORE job

- You will need to assign in your preamble of the script for the requesting resources. E.g. mcore.sh
- Submit job:

  sbatch mcore.sh
- This example will submit a job which requesting 10 CPU cores

```
#!/bin/bash

#SBATCH --job-name=My_MCORE_Job # define the name of your job
#SBATCH --time=01:00:00         # specify the required time
#SBATCH --nodes=1               # number of nodes allocated to the job
#SBATCH --ntasks-per-node=1     # number of tasks to invoke on each node
#SBATCH --cpus-per-task=10      # number of CPUs required per task
#SBATCH --mem-per-cpu=128GB     # set the memory limit for each task to 128GB
#SBATCH -D /ceph/work/<group>   # switch to the working directory and execute the command
#SBATCH --error=job.%J.err      # job error. By default, both files are directed to a file of the name slurm-%j.err
#SBATCH --output=job.%J.out     # job output. By default, both files are directed to a file of the name slurm-%j.out

srun stress -c 10 -t 100
```

Example 3 - Submit a python job using anaconda3 python3

- Prepare a python script that calculate pi number: calculate_pi.py

```
# Initialize denominator
k = 1
# Initialize sum
s = 0
for i in range(1000000000):
    # even index elements are positive
    if i % 2 == 0:
        s += 4/k
    else:
        # odd index elements are negative
        s -= 4/k
    # denominator is odd
    k += 2
print(f"{s}")
```

Example 3 - Submit a python job using anaconda3 python3

- Prepare a shell script that wrapping the environment modules and run python script: calculate_pi.sh

```
#!/bin/bash

module load app/anaconda3/4.9.2
python calculate_pi.py
```

- Submit job using sbatch

  sbatch calculate_pi.sh

Problem Report and FAQ

- Online documents: https://dicos.grid.sinica.edu.tw/wiki/
- Email channel to ASGC admins: DiCOS-Support@twgrid.org
- Regular face-to-face (on-site) video conferences: ASGC DiCOS user meetings (held every Wednesday at 13:20 (UTC+8)), please ask our staff for meeting information.