Recommended Practices at U Chicago HPC

Trung Nguyen, Ph.D.

Computational Scientist

Research Computing Center, University of Chicago

University-shared cluster: Midway3

(several others are dedicated to research centers and federal funded projects)

- 2 login nodes
- Shared compute nodes:
  - 400+ CPU-only nodes: 48-core Intel Cascade Lake CPUs, or 64-core AMD EPYC CPUs, 192 GB – 1.5 TB RAM, 960 GB SSD
  - 10 GPU nodes (NVIDIA V100 and A100)
- Dedicated compute nodes:
  - 100+ CPU-only nodes
  - 50+ GPU nodes (NVIDIA RTX6000, A40, A100, H100)
- Interconnect: Infiniband HDR
- Storage: 2 PB (GPFS) mounted on all the nodes

Our users

- 10+ schools and divisions: Science, Engineering, Economics, Medical, Humanities
- 100+ research groups
- ~1000 active user accounts within and outside U Chicago: students, postdocs, staff, PIs and their collaborators
- 100+ tickets/week submitted to Help Desk

Our team

- 3 HPC admins + 10 computational scientists + 4 part-time students
- Overlapping expertise in Computational Chemistry and Physics, Machine Learning/AI, Geographic Information System, Quantum Computing, Climate, Scientific Computing, Visualization
- Responsibilities
  - provide support: from user account creation to technical troubleshooting
  - develop internal tools and backend services
  - consult: computing and storage purchases, code optimization and development
  - organize workshops and tutorials

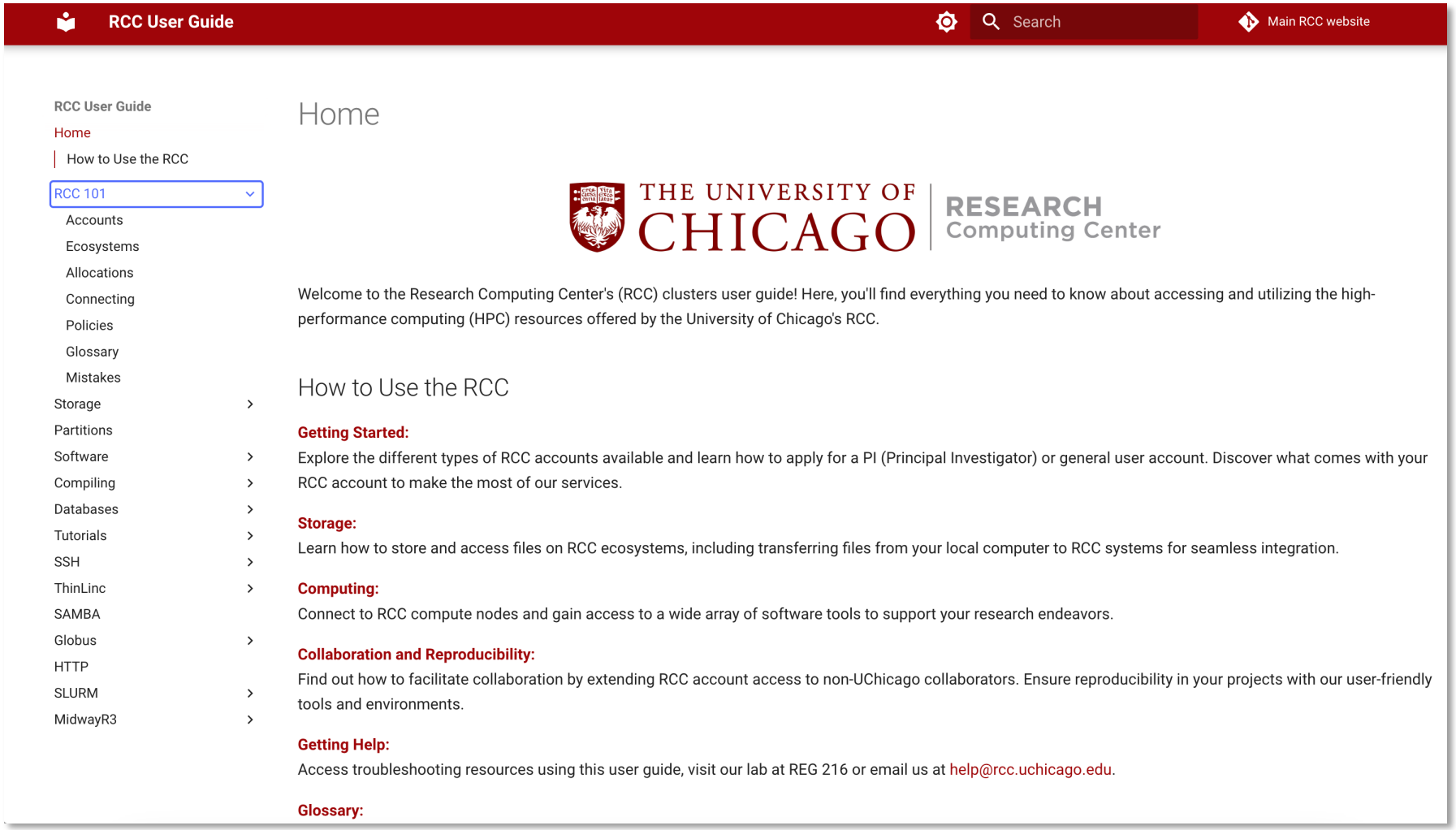
Our GitHub-hosted User Guide

https://rcc-uchicago.github.io/user-guide/

Search bar

Step-by-step instructions

- ✓ Quick user feedback
- ✓ Regular updates and fixes
- ✓ Changes made with pull requests and peer reviews



(Some) frequently asked questions

- How to install custom software packages?
- How to get code and data in and out the cluster?
- How to run my calculations on the cluster?
- Why do my simulations run so slow on these new nodes?
- Why do my calculations burn my PI's allocation so fast, yet I am not getting my stuff done, while frustrating others?
- My code would take months to complete, can you help me?

and my recommendation …

How to install custom software packages?

- Examples: QE 7.0 with libxc, AlphaFold v2.3.2, Ovito, pytorch
- Check if the software is already installed (module avail)
- Read the software package's Installation doc page
  - Python packages: install into your own env

    module load python

    python –m venv myenv

    conda create –prefix=/path/to/your/space/myenv –clone scicomp

    source activate myenv

    pip install …

    conda install …
  - Legacy packages written in Fortran/C/C++

    module load openmpi/4.1.2+gcc-10.2.0 mkl/2023.1 cuda/11.5 cmake/3.26

    mkdir build; cd build; cmake /path/to/CMakeList.txt/; make –j4; make install
- Recommendation: Read the manual carefully.

How to get code and data in and out of the cluster?

- Check our RCC User Guide, search for “data transfers”
- Login nodes have Internet access:
  - Codes: git clone/pull
  - Container images: singularity pull/apptainer pull
  - Datasets: wget/curl
  - Data to and from your laptops and workstations (scp/rsync)
- Compute nodes don’t have Internet access:
  - Dedicated to doing calculations when all the required input data is available
  - Accessible to on-premises storage spaces only
- Recommendation: Get your input data ready from the login node.

How to run the calculations on the cluster?

- Example: My python code runs on my laptop, how to run it on the cluster?
- Check our RCC User Guide, search for “jobs”
- General idea:
    1. You submit a script to the resource manager (SLURM) to ask for resource: number of nodes, CPU cores, RAM, GPUs, and walltime.
    2. Within the script, you specify where the input and output data are located, which software packages to be used, and the commands you wanted to run.
    3. SLURM puts your request (now called a job) into the queue and allocates the resource when available (some patience needed).
    4. The job lands on some compute node(s) and the commands inside your script get executed.
- Recommendation: Submit your jobs to the queue; search for the SLURM doc pages.

Why do my simulations run so slowly on these new nodes?

- Example: User reported their same code and input run 1.5-2x slower on their newer dedicated nodes
- How did we resolve this one?
  - Asked for the input and binary to reproduce the issue (old-fashioned MPI)
  - Narrowed down the issue to single node runs
  - Compared the CPU configure on the new (Intel Ice Lake) vs on the older nodes (Cascade Lake).
  - Found that the newer ones got hyperthreading (HT) turned on (the older ones don't).
  - Turning off HT gives the performance up to expectation on the newer nodes.
- Recommendation: Hardware configuration matters.

Why do my calculations burn my PI's allocation so fast, while frustrating others in the queue?

- Example: A user ran 10s jobs, each requesting 6 GPU nodes (24 GPUs and 192 CPUs in total), taking 4-6 hours, burning 10K+ core-hours per week.
- How did we help with this one?
  - Reached out to ask what they were trying to do (i.e. a typical job script): NAMD 2.14 with GPU acceleration, 200K atoms, standard equilibration
  - Suggested the user request **1** node, **1** GPU (--gres=gpu:1) and **4-8** CPUs via multithreading (Reason: Too small a system for multi-GPUs and multiple nodes)
  - Calculation per run reduced a bit, but occupying fewer GPUs and CPUs by 24 times, shorter waiting times for jobs to run, burning SUs more efficiently.
- Recommendation: Understand your tools and calculations.

My code would take months to complete, can you help me?

- Example: a PI has a Python code that loops through a long list of data files, does some work on each of them…
- How did we help?
  - Asked for the job script, asked the user to explain what they want to achieve
  - Turned the (serial) loop to a parallel for loop inside the job script

```
#SBATCH --ntasks-per-node=32
for i in {0..31}
do
  taskset –c $i python3 run.py data-$i.txt > out-$i.txt &
done
wait
```

  - The PI's work completed within a week, instead of 3 months.

- Recommendation: Maximize the concurrency at the task level.

Summary

To make best use of the HPC environment:

- Learn the specification of the compute nodes
- Read the software documentation and user guide
- Understand your calculations and how your tool(s) operate