



UM | Faculty of Agricultural
and Food Sciences

Essentials of High Performance Computing for New Users: A Practical Example Using Hydrological Models

Befekadu Taddesse Woldegiorgis & Marcos Cunha Cordeiro

October 31, 2023

Preview

💧 Why HPC

→ Make a case using a hydrological example

💧 Typical workflow for running a hydrological modelling job on HPC

💧 Demonstration

Why HPC for Hydrological Modelling?

- 💧 Modelling exercises usually involve several steps

 - Sensitivity analysis, parameterization, uncertainty quantification

- 💧 Simulations typically require large computational resources

 - e.g. Global sensitivity analysis

 - Requires a large sample size for convergence

 - e.g., for $n_{pars} = 40$, $n_{simulations} = 84000$, and, one minute per simulation,
~60 days are required for serial sensitivity analysis

Why HPC ? (cont'd)

💧 Parallel computation is required for practical use

→ Large memory and many CPUs may be required

▪ Usually impossible to accomplish on local computers


→ Using HPC is an optimum solution


▪ e.g., Cedar cluster: up to 48 cores per node, 28 days

▪ Multiple nodes can be run (independently) to finish jobs in short time

Workflow to Run a Job on HPC

Common steps

- Acquire HPC cluster information 
- Develop your script/code
- Log into HPC cluster
- Transferring files
- Testing and submitting a job
- Monitoring jobs and retrieving results



The screenshot shows the website for the Digital Research Alliance of Canada. The header includes the logo and the text "Digital Research Alliance of Canada" and "Alliance de recherche numérique du Canada". There are links for "English" and "Français". A navigation bar contains "Home" and "Support". The main content area has a welcome message and a "Please sign in" section with fields for "Login:" and "Password:". Below the fields are links for "Sign in", "Forgot Password", and "Register". A footer contains the copyright notice "© 2008-2023 Digital Research Alliance of Canada" and a link for "email Support".

Workflow to Run a Job on HPC (1/14)

💧 Transferring files to HPC cluster

→ Using WinSCP, FileZilla, Globus

Name	Size	Changed	Rights	Owner
..		2023-10-12 3:48:33 PM	rwxr-xr-x	root
Sensitivity_theoretical		2023-08-18 2:23:57 PM	rwxr-x---	mehariy
Five_parameter_Logistic_curve		2023-08-17 3:35:11 AM	rwxr-x---	mehariy

→ Using the command prompt (PowerShell)

▪ Using the scp command

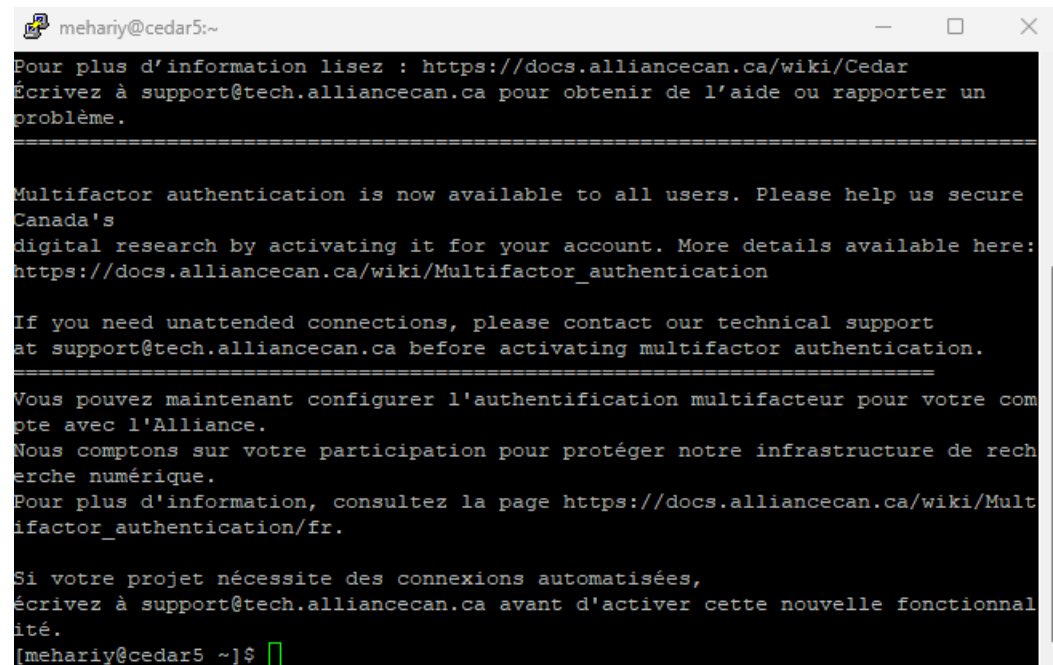
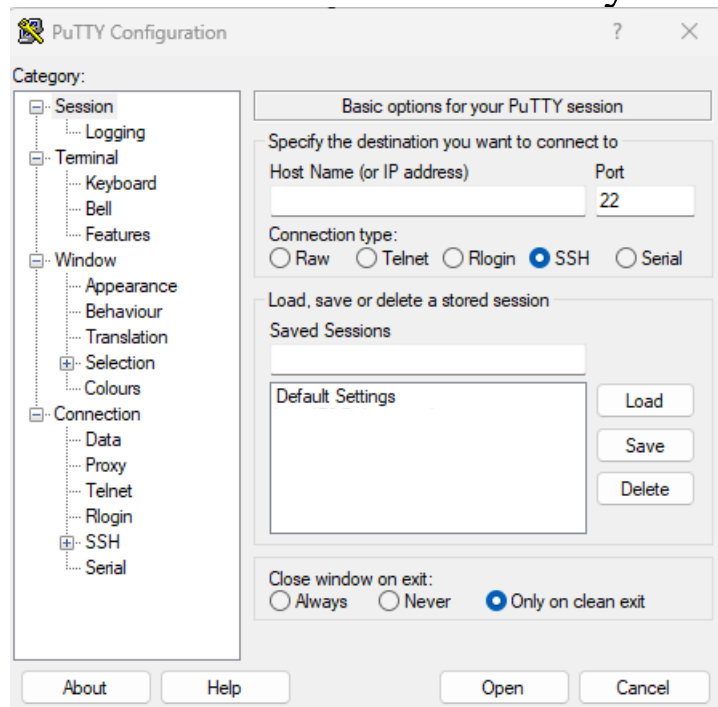
```
scp -r F:\BCRC_project\HYPE_AB_BTW1 username@cedar.computecanada.ca:/scratch/username/HPC/
```

Workflow to Run a Job on HPC (2/14)

💧 Logging into an HPC cluster using an SSH client

→ Install Putty

▪ an SSH client that securely connects to the remote machine and displays the screen



Workflow to Run a Job on HPC (3/14)

💧 You're in. What now?

→ Is the software you need available on the HPC cluster?

- Remember: HPC clusters run Linux!

- Common software available

 - ✍ e.g., R, Python, etc.

```
module spider r/
```

→ Load the software and install the required libraries from the HPC

```
module load r/4.2.1  
R -e "remotes::install_local('/localPath/sensitivity_1.29.0.tar.gz', dependencies=T)"
```


Workflow to Run a Job on HPC (4/14)

→ A high level scripting language (e.g. R, Python) is usually required

- To call the software

 - ✍ e.g., hydrological model

```
system("myExecutable.exe")
```

- To use available packages

 - ✍ e.g. for sensitivity analysis, optimization, parallelization of jobs, data manipulation, etc.

- To post-process results

Workflow to Run a Job on HPC (5/14)

→ Do you need a custom piece of software?

- Compilation might be required

- ✍ For example, to compile a Fortran application

```
module load intel/2022.1.0
```

```
make -f makefile
```

Workflow to Run a Job on HPC (6/14)

💧 Preparing the project directory

- Transfer (copy) all the files needed to successfully run your model
- Create subfolders for parallel computations
 - Usually each process updates the parameters and writes to the output file
 - ✍ Subfolders are required to avoid overwriting and errors
 - Use a shell script to generate subfolders and create symbolic links
 - ✍ create symbolic links for repetitive files that do not update during a simulation
 - ✍ A symbolic link keeps a single copy of the input file and 'points' to original file
 - ✍ A symbolic link saves space

Workflow to Run a Job on HPC (7/14)

→ Create “duplicateSubfolders.sh” in the main folder

```
for n in $(seq 1 16)
do
mkdir "MdCrk$n"
touch "MdCrk$n"/filedir.txt
echo "$(pwd)/MdCrk$n/" > "MdCrk$n"/filedir.txt
ln -sf $(pwd)/info.txt $(pwd)/MdCrk$n/info.txt
ln -sf $(pwd)/TMAXobs.txt $(pwd)/MdCrk$n/TMAXobs.txt
ln -sf $(pwd)/TMINobs.txt $(pwd)/MdCrk$n/TMINobs.txt
ln -sf $(pwd)/Tobs.txt $(pwd)/MdCrk$n/Tobs.txt
ln -sf $(pwd)/Pobs.txt $(pwd)/MdCrk$n/Pobs.txt
ln -sf $(pwd)/Hype_ParEdit.exe $(pwd)/MdCrk$n/Hype_ParEdit.exe
ln -sf $(pwd)/HYPE_5_5_1_IE.exe $(pwd)/MdCrk$n/HYPE_5_5_1_IE.exe
cp -r $(pwd)/CropData.txt $(pwd)/MdCrk$n/CropData.txt
done
```

→ Make it executable and execute the shell script

```
chmod +x duplicateSubfolders.sh && ./duplicateSubfolders.sh
```

Workflow to Run a Job on HPC (8/14)

→ Running multiple Fortran executables for each process

▣ Using a shell script wrapper for each process

- ✍ Passing default arguments to binaries or 3rd party apps. (e.g. remove unnecessary outputs)
- ✍ Changing path and launch the executable(s)

```
#!/usr/bin/bash
# Specifying the bash path. /bin/bash` works everywhere that has bash but /usr/bin/bash only works on a
exec > /dev/null 2>&1 # Equivqlent of #@echo off of Windows operating system
set -e # It is used to exit immediately if a command exits wit
cd '/scratch/mehariy/HPC_presentation/MdCrk1' #Changes directory for each parallel process
./Hype_ParEdit.exe # Run the parameter editor
PID=$! #catch the last PID (process identification number), he
wait $PID #wait for command1, in background, to end
./HYPE_5_5_1_IE.exe

bfname_parEdit_sim<-as.character(paste0("./run_par_edit_and_run_the_model_",id, ".sh"))
system(bfname_parEdit_sim)
```

Workflow to Run a Job on HPC (9/14)

💧 Testing the code before submitting a job

→ Test run your code at a small scale

▪ To avoid failing of jobs after a long queue

```
salloc --nodes=1 --ntasks-per-node=16 --mem=8G --time=00:30:00 --account=def-cordeiro salloc.sh
```

Workflow to Run a Job on HPC (10/14)

💧 If your code doesn't run

→ e.g., error returned for a parallel R code calling a Fortran app.

```
Error in { : task 1 failed - "cannot open the connection"  
Calls: %dopar% -> <Anonymous>  
Execution halted
```

```
no_cores <- 16  
cl<-parallel::makeCluster(no_cores)  
registerDoParallel(cl)  
  
clusterEvalQ(cl, {  
  .libPaths("/home/mehariy/R/x86_64-pc-linux-gnu-library/4.2")  
  library(bigmemory, help, pos = 2, lib.loc = "/home/mehariy/R/x86_64-pc-linux-gnu-library/4.2")  
})  
for (i in seq(1,dim(par_set)[1]/no_cores)) {  
  print(paste("Simulation",i*no_cores,"of",dim(par_set)[1],sep = " "))  
  for (j in seq(1,no_cores,1)) {  
    par_set_no_cores[j,] <- par_set[(i-1)*no_cores+j,]  
  }  
  paralle_runs <- foreach(id = 1:no_cores, .combine = rbind) %dopar%  
  {  
    bigmemory_shared_object_par <- attach.big.matrix("bigmemory_par_uncert_hydr_HPC_presntation.desc")  
    bigmemory_shared_object_pred <- attach.big.matrix("bigmemory_pred_uncert_hydr_HPC_presntation.desc")
```

Workflow to Run a Job on HPC (11/14)

→ Debugging a parallel R code

```
cl<-parallel::makeCluster(no_cores,outfile="")
```

```
Error in { : task 1 failed - "cannot open the connection"  
Calls: %dopar% -> <Anonymous>  
Execution halted  
Error in unserialize(node$con) : error reading from connection  
Calls: <Anonymous> ... doTryCatch -> recvData -> recvData.SOCKnode -> unserialize  
Error in unserialize(node$con) : error reading from connection  
Calls: <Anonymous> ... doTryCatch -> recvData -> recvData.SOCKnode -> unserialize  
In addition: In addition: Warning message:  
Warning message:  
In file(file, "rt") :  
  In file(file, "rt") : cannot open file '/scratch/mehariy/HPC_presentation/Observed_Q_Meadow_Creek_AB.txt  
Error in unserialize(node$con) : error reading from connection
```

▣ Alternatively, you can serialize the job

```
for (i in seq(1,dim(par_set)[1]/no_cores)) {  
  #paralle_runs <- foreach(id = 1:no_cores, .combine = rbind) %dopar%{  
    id <- 1
```


Workflow to Run a Job on HPC (12/14)

💧 Submitting a job to the HPC cluster

→ Preparing a submission script

- Define parameters in the HPC cluster
- Execute the job
 - ✍ Call the script that will run your software
 - R calls a hydrological model

Submission script >> runs the R script >> calls the hydrological model

Workflow to Run a Job on HPC (13/14)

💧 Submitting a job to the HPC cluster

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --tasks-per-node=16
#SBATCH --mem=8G
#SBATCH --time=0-00:30:00
#SBATCH --job-name=HPC_job
#SBATCH --account=def-cordeiro
echo "Starting run at: `date`"
module load r/4.2.1
echo "r/4.2.1 is loaded"
echo "Working in directory $PWD"
echo "Started running the job"
R CMD BATCH HPC_demonstration.R
echo "Program finished with exit code $? at: `date`"
exit 0
```

sbatch submitJob.sh

Workflow to Run a Job on HPC (14/14)

💧 Check the status of (a) job(s)

```
squeue -u YourUsername
```

→ Helps to identify whether a job has started or pending (failed)

```
scontrol show job 14761467
```

→ Helps to know when a specific job in a queue starts

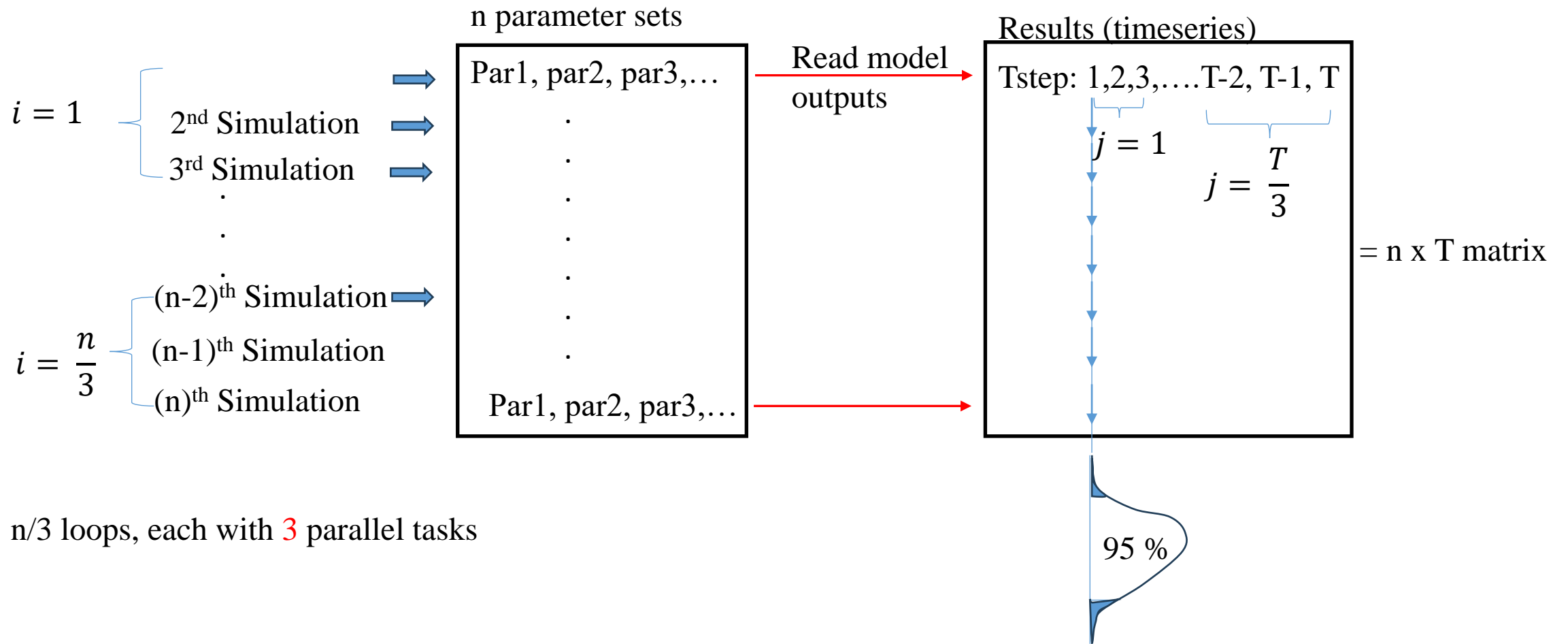
→ Helps to distinguish the path from which a specific job was submitted

Demonstration

Questions ?

Befekadu.woldegiorgis@umanitoba.ca

The parallel computing job



$n/3$ loops, each with 3 parallel tasks