Get this presentation:

git clone https://github.com/RBigData/R4HPC.git

- Open
  
  R4HPC_Part1.html

  in your web browser

- Navigation help is the question mark: ?

Many thanks to my colleagues and former colleagues who contributed to the software and ideas presented here and who are listed in the RBigData Organization on Github: https://github.com/RBigData. Also, many thanks to all R developers of packages used in this presentation.

Any errors are mine alone.
Using R on HPC Clusters Webinar

- A basic workflow for how to use R on an HPC cluster
- Speed up R scripts with parallel computing concepts
- Many packages in R offer parallel computing abstractions, yet they use a much smaller set of underlying approaches:
  - multithreading in compiled code, the unix fork, and MPI
- We take a narrow path to focus on the direct approaches
- Targeted for current users of OLCF, CADES, ALCF and NERSC
- Others are welcome to the lecture portions but will not be able to participate in all of the hands-on activities

Objectives

- Learn a workflow to edit R code on your laptop and run it on an HPC cluster
- Learn how to use multicore and distributed parallel concepts in R on an HPC cluster system
The Clusters

ORNL OLCF Andes

- 704 nodes, each with two 16-core 3.0 GHz AMD EPYC processors

ORNL CADES SHPC Condos

- ~650 nodes, a mix of x86_64 processors with 32 to 128 cores
- New LMOD software stack (see https://docs.cades.ornl.gov/#condos/software/bash-env/#new-software-stack)

LBL NERSC Cori

- 9,688 nodes, each an Intel Xeon Phi with 64 cores
- 2,388 nodes, Intel Xeon "Haswell" processor with 32 cores

ANL ACLF Theta KNL- Intel-Cray XC40

- 4,392 nodes, each with a 64-core, 1.3-GHz Intel Xeon Phi 7230
Access to HPC Clusters

- DOE OLCF https://docs.olcf.ornl.gov/accounts/accounts_and_projects.html
- DOE ORNL CADES https://cades.ornl.gov/
- DOE ALCF https://www.alcf.anl.gov/support-center/account-and-project-management/allocations
- DOE NERSC https://www.nersc.gov/users/accounts/allocations/
- NSF XSEDE to ACCESS https://www.xsede.org/
- EU PRACE https://prace-ri.eu/hpc-access/ (for example IT4I.cz https://www.it4i.cz/en/for-users/computing-resources-allocation)

- Institutional Clusters
Section I: Environment and Workflow

Section II: Parallel Hardware and Software Overview

Section III: Shared Memory Tools

Section IV: Distributed Memory Tools
Working with a remote cluster using R
Laptop RStudio (Posit in October, 2022)

- Familiar custom editing environment (Windows, Mac, Unix)
- Interactive Syntax checking

GitHub/GitLab

- Portability to remote computing
- Version control
- Collaboration

Cluster unix

- Same environment for all
- Batch job submission

**Advanced: interactive multinode development and debugging**

- Available now (packages: launchr, pbdCS, pbdRPC, remoter)
Running Distributed on a Cluster
Software Needed on Laptop

- **Mac**
  - R, RStudio
  - terminal, git (in Xcode)

- **Windows**
  - R, RStudio
  - putty
  - git
  - WinSCP
Software on Cluster

- OpenBLAS
- FlexiBLAS
- OpenMPI
- HDF5 (for parallel I/O)
- R (>= 4.0)

Packages:
Day 1: flexiblas, remotes, RBigData/pbdMPI, randomForest, mlbench
Day 2: RBigData/kazaam, RBigData/pbdDMAT

R vs conda-R Deployment

- Direct R is preferred
- CRAN and Anaconda differ in package management philosophy
- Can end up with conflicts if mixing
- Conda adds a layer of complexity
- If already used to Conda, you may find it useful
GitHub and git (laptop to cluster)
Making git easy: set ssh keys

A message encrypted by public key is decrypted by private key

Works like a single-use password generator and authenticator

Your private keys are protected in your account (laptop and cluster)

Put your public key on GitHub to enable easy access
Clusters are Linux systems

- Linux is one of many descendants of original Unix. MacOS is another.
- Like all file systems, Linux files are organized as a tree.
- When in a terminal, you are talking to a shell program (bash is most common)
  - Each command can have a list of options and a list of arguments
  - Standard input and standard output of a command is the terminal but can be redirected
  - `<, <<, >, >>` redirect standard input and output
  - `command1 | command2` pipes standard output1 to standard input2
  - Commands are looked up in directories listed in your PATH variable (try "echo $PATH")
  - `$` means substitute variable value
  - `export` lists (or sets) shell variables and their values
- There are many resources on the web to learn Linux basics
Job Submission on Cluster

- Command line submission
- Shell script submission (preferred)

**Slurm (Andes, CADES, )**

- `sbatch your-shell-script.sh`
- `squeue -u uid`
- `scancel jobnumber`

**PBS (Theta, )**

- `qsub your-shell-script.sh`
- `qstat -u uid`
- `qdel jobname`

**module** to set software environment (PATH)

- `module list` - list what is loaded
- `module avail` - list what is available
- `module load r`
Hands-on Session 1 - Fork and clone your R4HPC

- Fork R4HPC to your GitHub account
  - Login to GitHub
  - Navigate to RBigData/R4HPC repository
  - Click Fork button near top-right
  - Copy forked repo green Code url
- Clone to New Project in RStudio
- Open Terminal window (ssh or putty)
- Login to cluster
- Clone your R4HPC (git clone ...)
- You are ready for the development loop:
  - edit -> commit -> push -> pull -> run -> examine output
Hands-on Session 1 - On Login Node

- Go to R4HPC/code_1 directory
- `cat hello_MACHINE_slurm.sh` to see what modules to load and do so
- Start R and install needed packages:
  - `install.packages("remotes")`
  - `install.packages("flexiblas")`
  - `remotes::install_github("RBigData/pbdMPI")`
- Submit the `hello_MACHINE_slurm.sh`
- Examine output in `hello.e` and `hello.o` and notice that:
  - 4 nodes are involved
  - 4 R sessions were running on each node
  - Each R session ran `mclapply` on several cores
  - All `mclapply` process id's are reported
  - The code figured out how many cores in total
  - Only one R session wrote the output
Section II:

Parallel Hardware
Three Basic Concepts in Hardware

Shared Memory

Multicore Processor

- core
- core
- core
- core
- network
- memory

Shared Memory Co-Processor

- network
- memory

Manycore

- node
- node
- node
- node
- network fabric

Distributed Memory Cluster

- network
- memory

GPU
Three Basic Concepts in Hardware

Shared Memory
Multicore Processor

Shared Memory
Co-Processor

Manycore

GPU

Distributed Memory Cluster
Your laptop

Shared Memory
Multicore Processor

Shared Memory
Co-Processor

Manycore

Distributed Memory Cluster

node
node
node
node

network fabric

network
memory

network
memory

network
memory

network
memory
A Cluster of Multicore nodes with GPU co-processors
A Cluster of Multi-core nodes with Many-core co-processors
Section III:

Parallel Software
Native Programming Mindset

Default is serial: which tasks should be made parallel?

Shared Memory

Multicore Processor

Default is parallel: how to partition the data and what to share?

Offload data and tasks: We are slow but many!

Shared Memory Co-Processor

Distributed Memory Cluster

GPU
Native Programming Models and Tools

Shared Memory Multicore Processor

- OpenMP
- OpenACC

Offload data and tasks: We are slow but many!

Shared Memory Co-Processor

- CUDA
- OpenCL

Distributed Memory Cluster

- Sockets
- SPMD (MPI)
- MapReduce (Shuffle)

Default is parallel: how to partition the data and what to share?

Default is serial: which tasks should be made parallel?
35+ Years of Practical Parallel Computing

Default is serial: which tasks should be made parallel?

Shared Memory Multicore Processor

- OpenMP
- OpenACC

Shared Memory Co-Processor

- CUDA
- OpenCL

GPU

Distributed Memory Cluster

- Sockets
- SPMD (MPI)
- MapReduce (Shuffle)

Default is parallel: how to partition the data and what to share?
Last 15+ years of Advances

Default is serial: which tasks should be made parallel?

Shared Memory Multicore Processor

- OpenMP
- OpenACC

Offload data and tasks: We are slow but many!

Shared Memory Co-Processor

- Cuda
- OpenCL

pthreads
fork

Distributed Memory Cluster

Sockets
SPMD(MPI)
MapReduce(Shuffle)

Default is parallel: how to partition the data and what to share?
Distributed Programming Works in Shared Memory

Default is serial: which tasks should be made parallel?

OpenMP
OpenACC

Offload data and tasks: We are slow but many!

Cuda
OpenCL

Shared Memory Co-Processor

 pthreads
fork

Multicore Processor

Default is parallel: how to partition the data and what to share?

Sockets
SPMD(MPI)
MapReduce(Shuffle)

Distributed Memory Cluster

node
node
node
node

network fabric
R Interfaces to Low-Level Native Tools

Default is serial: which tasks should be made parallel?

Shared Memory Multicore Processor

Shared Memory Co-Processor

Offload data and tasks: We are slow but many!

OpenMP OpenACC

Foreign Language Interfaces
.C .Call Rcpp OpenCL inline

Gpu

pthreads
fork

parallel = multicore + snow

Distributed Memory Cluster

Default is parallel: how to partition the data and what to share?

Sockets SPMD(MPI)
MapReduce(Shuffle)

Cuda OpenCL

node network fabric

network memory

node core core core core

http://127.0.0.1:8080/R4HPC_Part1.html#1
Section IV:

Shared Memory Tools

Working with a single node
Working with a single node

Default is serial: which tasks should be made parallel?

Offload data and tasks: We are slow but many!

Shared Memory Co-Processor

OpenMP
OpenACC

Foreign Language Interfaces
C
.Call
Rcpp
OpenCL
inline

Shared Memory
Multicore Processor

pthreads
fork

parallel = multicore + snow

Distributed Memory Cluster

Default is parallel: how to partition the data and what to share?

Sockets
SPMD(MPI)
MapReduce(Shuffle)

node	node	node

network fabric

node

network
memory

Cuda
OpenCL

GPU

pbdMPI
Rmpi

RHadoop
SparkR

http://127.0.0.1:8080/R4HPC_Part1.html#1
Using R on HPC Clusters

Part 1

8/17/22, 2:27 PM

http://127.0.0.1:8080/R4HPC_Part1.html#1
Unix fork

- A memory efficient parallelism on shared memory devices
- Copy-on-write: copy page if forked process tries to write
- R: `parallel` package `mclapply` and friends
  - Use for numerical sections only
  - Avoid GUI, I/O, and graphics sections
- Convenient for data (not modified)
- Convenient for functional languages like R
- Careful with nested parallelism
  - OpenBLAS takes all cores by default
  - data.table switches to single threaded mode upon fork

A deeper discussion of fork memory (if you have interest) on YouTube by Chris Kanich (UIC)
Copy-on-write
Mapping Threads to Cores

Theory and Reality

- Operating system manages core affinity
- OS tasks can compete and core switching occurs frequently
R: Drop-in replacements (almost) for \texttt{lapply, mapply, and Map}

\begin{verbatim}
mclapply(X, FUN, ..., mc.preschedule = TRUE, mc.set.seed = TRUE, mc.silent = FALSE, mc.cores =getOption("mc.cores", 2L), mc.cleanup = TRUE, mc.allow.recursive = TRUE, affinity.list = NULL)

mcmapply(FUN, ..., MoreArgs = NULL, SIMPLIFY = TRUE, USE.NAMES = TRUE, mc.preschedule = TRUE, mc.set.seed = TRUE, mc.silent = FALSE, mc.cores =getOption("mc.cores", 2L), mc.cleanup = TRUE, affinity.list = NULL)

mcMap(f, ...)
\end{verbatim}
Hands-on Session 2 - Multicore Random Forest

- Go to R4HPC/code_2 directory
- Look at the rf_serial.R and rf_mc.R codes
Hands-on Session 2 - Example Random forest Code

Letter recognition data (20 000 × 17)

Figure 1: Letter Recognition data (image: [Frey and Slate, 1991], description: mlbench package).

*Parallel Statistical Computing with R: An Illustration on Two Architectures
arXiv:1709.01195
Hands-on Session 2 - Random Forest Classification

Build many decision trees

Each tree built from

- random subset of variables: subset of columns
- resampled (with replacement) data: same number of rows

Use their majority votes to classify
Hands-on Session 2 - R4HPC/code_2/rf_serial.R

suppressMessages(library(randomForest))
data(LetterRecognition, package = "mlbench")
set.seed(seed = 123)

n = nrow(LetterRecognition)
n_test = floor(0.2 * n)
i_test = sample.int(n, n_test)
train = LetterRecognition[-i_test, ]
test = LetterRecognition[i_test, ]

rf.all = randomForest(lettr ~ ., train, ntree = 500, norm.votes = FALSE)
pred = predict(rf.all, test)
correct = sum(pred == test$lettr)
cat("Proportion Correct:", correct/(n_test), "\n")
library(parallel)
library(randomForest)
data(LetterRecognition, package = "mlbench")
set.seed(seed = 123, "L'Ecuyer-CMRG")

n = nrow(LetterRecognition)
n_test = floor(0.2 * n)
i_test = sample.int(n, n_test)
train = LetterRecognition[-i_test, ]
test = LetterRecognition[i_test, ]

nc = as.numeric(commandArgs(TRUE)[2])
ntree = lapply(splitIndices(500, nc), length)
rf = function(x, train) randomForest(lettr ~ ., train, ntree=x,
  norm.votes = FALSE)
rf.out = mclapply(ntree, rf, train = train, mc.cores = nc)
rf.all = do.call(combine, rf.out)

crows = splitIndices(nrow(test), nc)
rfp = function(x) as.vector(predict(rf.all, test[x, ]))
cpred = mclapply(crows, rfp, mc.cores = nc)
pred = do.call(c, cpred)
correct <- sum(pred == test$lettr)
Hands-on Session 2 - Assignment

Time the random forest code rf_mc.R for 1 through 32 cores by modifying the rf_MACHINE_slurm.sh script.
Libraries via compiled language interfaces

Default is serial: which tasks should be made parallel?

Shared Memory
Multicore Processor

OpenMP
OpenACC

Shared Memory
Co-Processor

Cuda
OpenCL

Foreign Language Interfaces

network

memory

parallel = multicore + snow

Distributed Memory Cluster

Default is parallel: how to partition the data and what to share?

http://127.0.0.1:8080/R4HPC_Part1.html#1
R-LAPACK-BLAS

- **BLAS**: Basic Linear Algebra Subroutines - A matrix multiplication library
  - `%*%`, `crossprod()`, `sweep()`, `scale()`, and many more
- **LAPACK**: dense and banded matrix decomposition and more
  - `svd()`, `La.svd()`, `prcomp()`, `princomp()`, `qr()`, `solve()`, `chol()`, `norm()`, and many more
  - But not `lm()`, careful with `qr(x, LAPACK = TRUE)`: column pivoting

- Implementations: OpenBLAS, Intel MKL, Nvidia nvBLAS, Apple vecLib, AMD BLIS, Arm Performance Libraries

- **FlexiBLAS**: A BLAS and LAPACK wrapper library with runtime exchangeable backends
  - Great for benchmarking implementations
  - Great for dynamic core assignment
OpenBLAS

OpenBLAS is an optimized BLAS library based on GotoBLAS2 (2010, Kazushige Goto).

- [openblas.net](http://openblas.net)
- Optimizes algorithm to chip microarchitecture details of memory hierarchies (L1 cache, L2 cache, etc.) and register vector length
- IT4I FlexiBLAS: "OPENBLAS" backend

Wang Qian, Zhang Xianyi, Zhang Yunquan, Qing Yi, AUGEM: Automatically Generate High Performance Dense Linear Algebra Kernels on x86 CPUs, In the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'13), Denver CO, November 2013.
FlexiBLAS

flexiblas_setup.r

```r
library(flexiblas)
flexiblas_avail()
flexiblas_version()
flexiblas_current_backend()
flexiblas_list()
flexiblas_list_loaded()

getthreads = function() {
  flexiblas_get_num_threads()
}
setthreads = function(thr, label = "") {
  cat(label, "Setting", thr, "threads\n")
  flexiblas_set_num_threads(thr)
}
setback = function(backend, label = "") {
  cat(label, "Setting", backend, "backend\n")
  flexiblas_switch(flexiblas_load_backend(backend))
}

https://github.com/Enchufa2/r-flexiblas
https://cran.r-project.org/package=flexiblas
```
Hands-on Session 3 - FlexiBLAS

- Go to code_3 directory
## Default BLAS from Netlib

```r
> x = matrix(rnorm(1e7), nrow = 1e4)
> system.time(crossprod(x))
user  system elapsed
6.752  0.023  6.801
```

## vecLib

```r
> system.time(crossprod(x))
user  system elapsed
0.666  0.003  0.120
```

## OpenBLAS

```r
> system.time(crossprod(x))
user  system elapsed
0.822  0.042  0.121
```
Appendix: FlexiBLAS For BLAS Control on your macOS Laptop

- Install Xcode and command line tools
- Install Homebrew: [https://brew.sh/](https://brew.sh/)
- In a terminal window:
  - `brew install cmake`
  - `brew install openblas`
- cmake needs to be told about OpenBLAS:
  - `export CMAKE_PREFIX_PATH=/usr/local/opt/openblas:$CMAKE_PREFIX_PATH`
- Install FlexiBLAS: [https://www.mpi-magdeburg.mpg.de/projects/flexiblas](https://www.mpi-magdeburg.mpg.de/projects/flexiblas)
  - See Install section in its README.md
- After installation, link to R (terminal window):
  - `ln -sf /usr/local/lib/libflexiblas.dylib /Library/Frameworks/R.framework/Resources/lib/libRblas.dylib`
- In R, `install.packages("flexiblas")` and test if it works:
  - `flexiblas_avail()`
  - `flexiblas_list()`

R can now swap OpenBLAS and APPLE vecLib dynamically.
Appendix: For faster R on your Windows laptop

Assessing R performance with optimized BLAS across three operating systems [link](#)

Building R 4+ for Windows with OpenBLAS [link](#)