Intro to R Programming & Parallelism
Overview

• Intro to R
• Using R on the CBI Cheetah Cluster
• Intro to RStudio
• Multicore parallelism in R
• Multisystem parallelism in R
• One hour hands on session
Intro to R

• R is an implementation of the S/Scheme languages (1993, 1995 (open sourced))
• S language originally created at Bell Labs (1976)
• R in family of tools such as SAS, SPSS, Stata, Matlab + Statistics toolbox
• R is open source, excellent graphics, extensible, multi-platform, excellent multiprocessing parallelism support
• Usable within HPC environments
  – User based package setup control
  – Interactive Jobs: Command Line & Rstudio
  – Batch Jobs: Grid Engine + R
  – Parallel Jobs with R w/ MPI based packages + Grid Engine
    • parallel, doParallel, foreach
    • Rmpi, doMPI, foreach
    • Rmpi, snow, snowfall
Intro to R

• Key Differentiator: R is open source (GNU GPL)
  – http://cran.r-project.org/src/base/R-3/R-3.0.2.tar.gz

• Additional ecosystem tools: Rstudio® provides an excellent Graphical Development environment.
  – Available on the CBI Cheetah cluster.
  – R is <a language + programming environment + data analysis + data visualization + parallel processing environment + applied & general purpose software development environment + software deployment framework + tools & ecosystem…>

• 5000+ packages are available (e.g. Rmpi, doMPI,…)
  – http://cran.r-project.org/web/packages/

• Commercial support: www.revolutionanalytics.com
# Matlab vs. R

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scripting Language</td>
<td>Scripting Language</td>
</tr>
<tr>
<td>Commercial</td>
<td>Open-Source (GPL), commercial support from <a href="http://www.revolutionanalytics.com">www.revolutionanalytics.com</a></td>
</tr>
<tr>
<td>Mathworks Inc., Natick, MA, (Founded 1984)</td>
<td>Ross Ihaka &amp; Robert Gentleman (University of Auckland, New Zealand); R Development Core Team (GNU Project); [1993(binary), 1995(open source)]</td>
</tr>
<tr>
<td>Originally Built for Linear Algebra</td>
<td>Originally Built for Statistics, derives from S language, with Scheme influences</td>
</tr>
<tr>
<td>Integrated IDE + Visualization</td>
<td>Multiple IDE’s &amp; Visualization environments (Rstudio, Rcmdr,... ggplot2)</td>
</tr>
<tr>
<td>Toolboxes + MatlabCentral</td>
<td>cran.r-project.org</td>
</tr>
<tr>
<td>C&amp;C++ &amp; Fortran Integration</td>
<td>C&amp;C++ &amp; Fortran Integration</td>
</tr>
<tr>
<td>Parallel Computing Toolbox, MDCS, Auto-multithreading</td>
<td>parallel, Rmpi, doParallel, doMPI, foreach, iterators, snow, snowfall (multi-processing)</td>
</tr>
<tr>
<td>Symbolic Math Toolbox</td>
<td>Ryacas Interface (Package ‘Ryacas’)</td>
</tr>
</tbody>
</table>
## Matlab vs. R

<table>
<thead>
<tr>
<th>MATLAB</th>
<th>R</th>
</tr>
</thead>
</table>
| Simulink (Control Theory, DSP, Model Based Design) targets Engineering Applications | ...
| Many domain specific toolboxes                                        | May not be a direct 1 to 1 mapping in R for all Matlab Toolboxes     |
| Statistics Toolbox                                                     | R was built for Statistics                                          |
| Built in Java support                                                  | rJava                                                               |
| Web: Matlab Builder JA                                                | Shiny                                                               |
| Matlab Parallel Compute Toolbox (PCT) GPU                              | Rpud package for R + GPU                                            |
| Matlab Parallel Compute Toolbox Distributed Arrays                     | RHIPE Package                                                       |

[cbi.utsa.edu](http://cbi.utsa.edu)
Intro to R: Language Overview

• Interpreted language
  – Interfaces to C, C++, Fortran
• Data structures: Vectors, Matrices, Lists, Factors, Data Frames
• 2-D & 3-D plotting features (grid, lattice, ggplot2, scatterplot3d,...)
• Statistical data analysis functions
• Similarity in code structure to Matlab
• GUI interface: RStudio®, Rcmdr
• Reactive Web Programming in R: Shiny (e.g. spreadsheets...)
• Access to GPUs possible & Hardware accelerators
• **Multi-core & Multi-system parallelism enabled via key packages:** Rmpi, parallel, Snow, Snowfall, doMPI, doParallel, foreach packages
R environment@ CBI Lab

- **Interactive Command Line:**
- Interactive GUI
- Batch
R environment@ CBI Lab

- Interactive Command Line
- **Interactive GUI:**
- Batch

![RStudio](image)
**Interactive GUI:**

Additional Packages can be easily installed: e.g. scatterplot3d package

```
install.packages("scatterplot3d")
library('scatterplot3d')
```

```r
z <- seq(-100,100,0.1);
x <- cos(z);
y <- sin(z);
scatterplot3d(x,y,z);
```
R environment@ CBI Lab

Interactive 3D Visualization

cbi.utsa.edu
R environment@ CBI Lab

3d interactive visualization:

install.packages("rgl")
install.packages("Rcmdr")
library("rgl")
library("Rcmdr")
plot3d(rnorm(100), rnorm(100), rnorm(100))
• **Batch Mode**: Submit both serial & parallel programs as batch jobs that can be simply submitted to the Cheetah Cluster Scheduler.

• Frees you from having to have your computer on for long periods of time.

• After you submit the job, you can exit and turn off your laptop, the job will be on the Cheetah Cluster queue and will run as soon as an open slot is available on the cluster.

• Good plan is to develop in interactive mode; then submit the jobs in batch mode when ready.
**Batch:** Submitting R jobs to the Cheetah Cluster via the Grid Engine.
- Serial R job (default)
  - The default is 1 R operating system process per job.
  - This maps to 1 processor core.
- High throughput using Grid Engine Job Arrays
  - Many fully independent R jobs, e.g. parameter scans.
- Multi-core, single node R jobs
  - With parallel packages, developer can create multi-core parallel applications with R; parallel package
- Multi-core & multi-system R parallel jobs
  - With Rmpi + doMPI, foreach / snow, snowfall: Developer can create multi-system distributed/parallel applications.
R + HPC Frameworks

- **The foreach framework:** Separates parallel algorithm from backend parallel runtime (abstract for loop)
- **MPI + Rmpi + doMPI + foreach:** Multicore+Multisystem
  - *Other Approaches:*
    - *parallel + doParallel + foreach:** Multicore
    - **MPI + Rmpi + snow + snowfall:** Multicore+Multisystem

- Interactive development mode (serial/multicore)
- Batch job (serial/multicore)
- Batch MPI enabled job (multisystem)
R + HPC

• **Rmpi + doMPI + foreach:** A bit simpler to use versus snowfall, scales to multi-system parallelism

• **Rmpi + snow + snowfall:** More flexibility e.g. load balancing, scales to multi-system parallelism

• **parallel + doParallel + foreach:** Multi-core parallelism
Parallel Infrastructure Initialization

```r
# doParallel initialization
myCluster <- makeCluster(numCores)
registerDoParallel(myCluster)

# doMPI initialization
myCluster <- startMPIcluster(count=numCores)
registerDoMPI(myCluster)  # register with foreach

# snowfall initialization
sfInit(parallel=TRUE, cpus=numCores, type="MPI")
```
R+HPC

```r
# Compute work (foreach + (doParallel/doMPI) approach)

x <- foreach(i = 1:200, .combine="c") %dopar% {
  set.seed(i)  # make tests repeatable
  testData <- rnorm(10000)  # make some data
  #print(testData)
  # create computational workload, j controls runtime
  for ( j in 1:2000 ) {
    testData <- rnorm(1000) + j
    #print(testData)
  }
  # summarize data
  x <- mean(testData)
}

result = x
```

doParallel/doMPI+ foreach
# Compute work (snowfall approach)

calcPar <- function(i) {
  set.seed(i) # make tests repeatable
  testData <- rnorm(10000) # make some data
  #print(testData)
  # create computational workload, j controls runtime
  for (j in 1:2000) {
    testData <- rnorm(1000) + j
    #print(testData)
  }
  # summarize data
  x <- mean(testData)
  return(x)
}

result <- sfLapply(1:200, calcPar) # Non-load balanced
#result <- sfClusterApplyLB(1:200, calcPar) # Load balancing
result <- as.matrix(result) # convert from list to matrix for plotting
R+HPC

Multi-core (foreach)

Multi-system (snowfall)

Multi-system (foreach)
Demo

An example R parallel code running on the Cheetah Cluster
Hands-On Session ( 1 hr )

• Logging into Cheetah Cluster @ CBI Lab
• Running R & Rstudio Session Interactively (qlogin)
• Trying out a sample R batch serial program
• Trying out a sample R parallel program (interactive development mode)
• Trying out a sample R parallel program( batch mode)
Appendix

More detailed info on R parallel environment.
**Example 1: Interactive mode, multi-core (single system)**

- ssh -Y user.name@cheetah.cbi.utsa.edu
- qlogin
- cd ~/Rworkshop/example1
- ./runInteractive.sh 4
- 4 cores ~ 17 seconds (doParallel, Interactive development node) vs. **1 min @ 1 core**.

**Example 1: Batch mode, multi-core (single system)**

- ssh -Y user.name@cheetah.cbi.utsa.edu
- qlogin
- cd ~/Rworkshop/example1
- qsub ex1.job
- 4 cores ~ 15 seconds (doParallel, Batch mode)
Example 2: Interactive mode, Rmpi+snowfall (single system)

• ssh -Y user.name@cheetah.cbi.utsa.edu
• qlogin
• cd ~/Rworkshop/example2
• ./runInteractive.sh 4
• 4 cores ~ 16 seconds

Example 2: Batch mode, Rmpi+snowfall (multi-system)

• ssh -Y user.name@cheetah.cbi.utsa.edu
• qlogin
• cd example2
• qsub ex2.job (edit job file to control # MPI workers)
• 16 cores ~ 8 seconds
Multi-system parallel R job using Rmpi+snow+snowfall running on the Infiniband Queue:
**Example 3: Interactive mode, Rmpi+doMPI+foreach (single system)**

- `ssh -Y user.name@cheetah.cbi.utsa.edu`
- `qlogin`
- `cd example3`
- `./runInteractive.sh 4`
- 4 cores ~ 17 seconds

**Example 3: Batch mode, Rmpi+doMPI+foreach (multi-system)**

- `ssh -Y user.name@cheetah.cbi.utsa.edu`
- `qlogin`
- `cd example3`
- `qsub ex3.job ( edit job file to control # MPI workers )`
- 16 cores ~ 8 seconds
Other Aspects of R

Debugging
Profiling
Package Building
Package Deployment
C/C++/Fortran Integration: Rcpp
Documentation with Knitr
R+GPU programming with rpuD
Application Areas

- Many, many application areas...
- Regression data analysis
- PCA: Principal Component Analysis
- Monte Carlo Simulations
- Machine Learning/Predictive Analytics (Random Forests,...)
- Bootstrap methods
- Data visualization
- Bayesian Inference, SVM
- GPU algorithms
- Parallel/Distributed data analysis in HPC clusters
- ... and many more application areas... <A general purpose software development environment>
Parallel Software Design Points

• Focus on dependencies
• Try to first parallelize at highest level possible
  – System Level then processor/core level
• Minimize communications between workers
• Maximize compute work per unit of communication
• Compute(ALU), Network Communication, Memory, Disk, Cache, I/O issues...
• R programming environment gives you tools to investigate & design parallel programs.
• Optimal # workers ~ algorithm&system dependent
Useful References

[27] http://www.ats.ucla.edu/stat/r/library/functions_no_loop.txt
Useful References

[31] https://idre.ucla.edu/stats (Statistical Computing resources)
[34] http://cran.r-project.org/web/packages/foreach/vignettes/foreach.pdf (parallel quicksort demo)
[36] http://www.stat.wisc.edu/~bates/JuliaForRProgrammers.pdf (Julia is an upcoming programming language with many similarities to R)
[37] https://www.coursera.org/course/rprog
[38] http://www.cbi.utsa.edu/faq/sge/infiniband
[40] http://www.cbi.utsa.edu/faq/sge/strategies
[42] https://gist.github.com/steveharoz/172ed1b1825f101be0f8 (Mandelbrot example program in R +foreach)
Acknowledgements

- This project received computational, research & development, software design/development support from the **Computational System Biology Core/Computational Biology Initiative**, funded by the National Institute on Minority Health and Health Disparities (**G12MD007591**) from the National Institutes of Health. URL: [http://www.cbi.utsa.edu](http://www.cbi.utsa.edu)
Contact Us

http://cbi.utsa.edu