Exercise 01-Getting Started on the HPC

Peter S. Hovmand

1/11/2022

Introduction

CWRU's High Performance Computing (HPC) cluster provides a powerful tool for conducting computational intensive system dynamics research using RStudio and Stella Simulator. These tools are now more accessible than every before using the OnDemand web portal enabling shared resources to be used more efficiently with higher levels of utilization than ever before. For Community Based System Dynamics, this technology closes the gap between computationally intensive analyses and interpreting the meaning of results.

Learning objectives

In this exercise, we will:

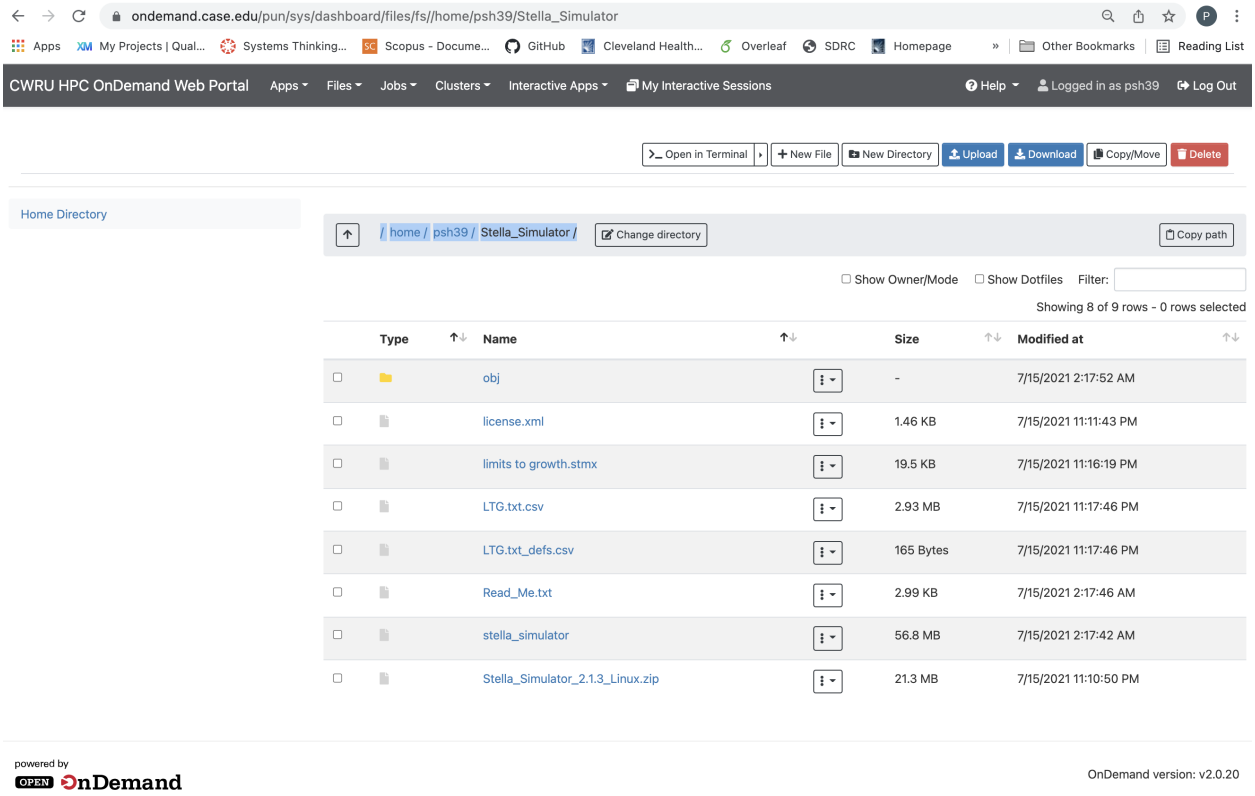
1. Check HPC access and permissions to the CBSD Lab directors and software.
2. Learn how to set up paths, directories, and manage simulations in a way that is scalable in parallel computing.
3. Introduce and learn how to run a Stella model using Stella Simulator on the HPC.
4. Learn how to export parameter values and initial conditions to a Stella model.
5. Learn how to import and plot simulation results in R.

Accessing the High Performance Computing (HPC) cluster

The easiest way to get started with the HPC is to go to the OnDemand web portal, where you can develop and test code interactively using RStudio server and submit/manage batch jobs that require more computationally intensive resources in terms of memory and cores.

You can access the the OnDemand web portal by going to https://ondemand.case.edu, where you may be prompted to enter your user name, password, and 2-factor authentication. Once you have been added to the CBSD Lab group under psh39, you should be able to access the CBSD Lab Stella Simulator license.

To check this, change your directory to /home/psh39/Stella_Simulator, and you should see the following files:



Stella Simulator

isee Systems Stella Simulator is a "headless" version of Stella Architect without the graphical user interface and runs on the CWRU HPC cluster under the Linux operating system. Stella Simulator runs considerably faster than Stella Architect since it does not have to manage the visualization of output and can support up to 20 parallel simulations for each license. Stella Simulator runs any valid Stella model, sensitivity analysis and optimization. And, Stella Simulator (from version 2.1.3+) exports the loop score analysis for each run.

Stella Simulator is run through the terminal or system command, which in R is the `system2()` command. To check access to being able to run Stella Simulator and see the latest options, run the following code in RStudio:

```
system2("/home/psh39/Stella_Simulator/stella_simulator", stderr = TRUE)
```

```
## Warning in system2("/home/psh39/Stella_Simulator/stella_simulator", stderr =
## TRUE): setting stdout = TRUE
```

```
## Warning in system2("/home/psh39/Stella_Simulator/stella_simulator", stderr =
## TRUE): running command ''/home/psh39/Stella_Simulator/stella_simulator' 2>&1'
## had status 1
```

```
##  [1] "Stella Simulator version 2.1.3, Copyright (C) 2021 by isee systems, inc."
##  [2] "Registered to: Peter S Hovmand, Case Western Reserve University"
##  [3] "Usage: stella_simulator <options> model_file"
##  [4] "  -0 arg               Set specified variable to 0 before first run"
##  [5] "  -1 arg               Set specified variable to 1 before first run"
##  [6] "  -d arg               Seconds to delay between successive runs (default: 0)"
##  [7] "  -h arg               Name of file to use for handshakes (default: none)"
##  [8] "  -i                   Import now before each run"
##  [9] "  -ld arg              Load data from the file as a run"
## [10] "  -ltm arg             Run LTM and put results in file (and file_defs.txt)"
```

```
## [11] "  -ltmn arg            Exhaustive loop search threshold"
## [12] "  -p arg               Pause at the given interval (0: DT; default: no pause)"
## [13] "  -pd arg              Seconds to delay before resuming after a pause "
## [14] "                        (default: 0)"
## [15] "  -pe arg              Off|Model|Interface Observe pause events from the "
## [16] "                        mdodel or interface (default off)"
## [17] "  -ph arg              Name of file to use for pause/resume handshakes "
## [18] "                        (changes -pd to maximum wait/overrides -pf; default: "
## [19] "                        none)"
## [20] "  -pi                  Pause immediately after initialization"
## [21] "  -pir                 Pause immediately after initialization, then "
## [22] "                        reinitialize on resume"
## [23] "  -ps arg              Name of the status file (default: none). Use with -ph"
## [24] "  -q                   Quiet mode (only errors are output)"
## [25] "  -r                   Run the model once (default)"
## [26] "  -rn arg              Run specified number of times"
## [27] "  -ro                  Perform optimization (requires optimization be set up "
## [28] "                        in the model)"
## [29] "  -rs                  Perform sensitivity (requires sensitivity be set up in "
## [30] "                        the model)"
## [31] "  -x                   Export now after each run"
## [32] ""
## attr(,"status")
## [1] 1
```

Running a simulation

Stella simulations run in the folder of the model where all paths to are relative to the directory of the model. So, for example, if you run the SIR.stmx model in your home directory in Stella Architect and save the results, you'll see that Stella Architect created the SIR.isdb file in the same directory.

Stella Simulator *does not* create a .isdb file for the simulation run, so you will need to set up the model in Stella Architect with a file for exporting the data from each run. And, typically, you'll also want to set up a file for sending the parameter values and initial conditions for importing data into a model before each run. The paths for these files are also relative to the path of the model.

Since Stella Simulator can simulate multiple models in parallel, conflicts can arise if the models and import/export files are all in the same directory. The easiest and safest way to manage this is to set up a separate folder for each simulation, copy the model to that folder along with import/export files, and then run the model in that folder.

To make this easier and more generalizable, we'll set variables that we can use to reference the paths for Stella Simulator and our simulation runs.

First, set the path to Stella Simulator:

```
stella_path <- "/home/psh39/Stella_Simulator/stella_simulator"
```

Next, create the directory to copy the model and import/export files to:

```
run_path <- "SDRUG_EX1_tmp"
dir.create(run_path)
```

```
## Warning in dir.create(run_path): 'SDRUG_EX1_tmp' already exists
```

Once we've set up the paths and directories, copy the files associated with a model into the `run_path` folder.

We're just running one simulation in a single thread, so there are many ways to do this, but to take full advantage of the HPC and Stella Simulator, we'll want to write programs that can run multiple models in parallel. Hence, we need an efficient way to copy a set of files associated wtih a model to one or more folders for each run. The R `unzip()` function does this nicely.

```
zipped_model <- "/home/psh39/Desktop/Projects/SD_R_Users_Group/Exercise_01/SIRmodel.zip"
unzip(zipped_model, exdir=run_path)
```

Finally, we'll call the `system2()` function and run the model:

```
system2(stella_path,
        args=paste0(" ",run_path,"/SIR.stmx"," -r"), stdout = TRUE)
```

```
## [1] "Stella Simulator version 2.1.3, Copyright (C) 2021 by isee systems, inc."
## [2] "Registered to: Peter S Hovmand, Case Western Reserve University"
## [3] "Begin run.End run!"
```

Changing parameters and initial values

Setting and changing the parameters and initial values in R is done by creating a named list and saving the list to the .csv file that the model is looking for as an input file:

```
parms<-c(Contact_Rate=50, Infectivity=0.1)
write.csv(parms,file=paste0(run_path,"/Parms.csv"))
```

Run the model again with the new parameters:

```
system2(stella_path,
        args=paste0(" ",run_path,"/SIR.stmx"," -r"), stdout = TRUE)
```

```
## [1] "Stella Simulator version 2.1.3, Copyright (C) 2021 by isee systems, inc."
## [2] "Registered to: Peter S Hovmand, Case Western Reserve University"
## [3] "Begin run.End run!"
```

Importing and plotting results

To import the data from the simulation run into R and plot the data, read the Stella export file which has been set up:

```
results <- read.csv(paste0(run_path,"/Results.csv"))
head(results)
```

```
##   months Contact.Rate Infected Infection.Rate Infectivity Population  Recovered
## 1    1.0           50 1.000000       2.000000         0.1       2500 0.00000000
## 2    1.1           50 1.166667       2.332867         0.1       2500 0.03333333
## 3    1.2           50 1.361064       2.720949         0.1       2500 0.07222222
## 4    1.3           50 1.587791       3.173341         0.1       2500 0.11759104
## 5    1.4           50 1.852198       3.700608         0.1       2500 0.17051739
## 6    1.5           50 2.160519       4.315020         0.1       2500 0.23225733
##   Recovery.Rate Recovery.Time Susceptible
## 1     0.3333333             3   1000.0000
## 2     0.3888889             3    999.8000
## 3     0.4536881             3    999.5667
## 4     0.5292635             3    999.2946
## 5     0.6173994             3    998.9773
```

```
## 6     0.7201731              3    998.6072
```

Results can then be plotted using R plotting functions.

```
plot(Infected~months, data=results, type="l")
```