Exacloud: An Overview

Stephan Lindner

6/7/2016

Goals of this presentation

- Give an overview of Exacloud and how we can use it.
- Provide a foundation for discussion regarding how to move forward with Exacloud server.
- Stay away from more complicated, technical aspects.

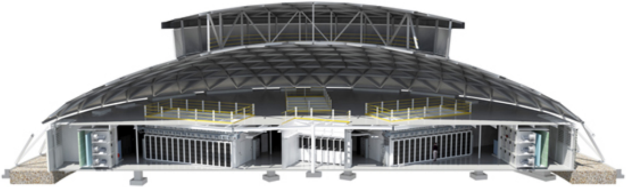
Outline

1. What is Exacloud?
2. Accessing and navigating Exacloud.
3. Interactive and non-interactice use of Exacloud.

1. What is Exacloud? And why is it on a Linux server?

What is Exacloud?

- Exacloud a server run by OHSU's Advanced Computing Center to support large-scale, computational and data intense workflows.
- Currently more than 35 Terabytes of memory and more than 750 Terabytes of usable storage.
- Housed at the Data Center at OHSU's West Campus.



Exacloud and Linux

- Exacloud uses Linux as operating system.
- By contrast, the CHSE server (and our computers) use Windows as operating system.
- An operating system is the software that manages a computer's basic functioning — the "habitat of your programs".
- Linux and Windows get along OK, but they do not particularly like each other.
- Most programs (such as R, Stata) are developed for both operating systems.

Why does Exacloud uses Linux?

Linux is . . .

- Very stable.
- Slim and scalable and therefore has less hardware requirements.
- Designed as a multi-user system.
- More secure than Windows.
- FOSS (Free and Open Source Software).

What this mean for us

- Many programs we use for our analysis are open-source, have been developed on Linux and work very well on Exacloud: `R`, `git`, `markdown`.
- `RStudio` and `Stata` are not open-source and native to Linux. They are currently not installed on Exacloud.
- Accessing Exacloud is different from accessing the CHSE server.

2. Accessing and navigating Exacloud

Accessing Exacloud via ssh

- Remote access of CHSE server: through Windows desktop.
- Remote access of Exacloud: through `ssh` (secure shell).
- `Shell` is a command prompt to interact with the computer (e.g., start or terminate programs).
- Bare-bone, 1970 technology that requires very little memory.

MobaXterm: ssh for Windows

- Install `MobaXterm` on desktop.
- Initiate ssh session with
  - Remote host: `exacloud.ohsu.edu`
  - User name: `your OHSU user name`.
- Prompts for password and then connects to server.

Navigating Exacloud

A couple of useful commands:

- Printing the working directory: `pwd`
- Listing files in current directory: `ls (-lh / -a)`
- Start `R`: `R`
- Start `Stata`: `stata` (currently not installed).
- Work with `HTcondor`: `condor_submit`, `condor_q`

Switch to MobaXterm

3. Interactive and non-interactive use of Exacloud

Interactive versus non-interactive R / Stata session:

1. **Interactive:**

   - Workflow: Develop some code chunk in script file → evaluate code in `R` / `Stata` → revise / debug code → . . .

   - Setup: Umbrella program that integrates editor with statistical program: `RStudio`, `Stata`'s GUI, etc.

   - Requirement: Umbrella program needs to be able to send code chunks to `R` / `Stata` and display results.

Interactive versus non-interactive R / Stata session:

2. Non-interactive mode:

- Workflow: Write full script file → run full script in `R` / `Stata` → revise / debug → ...
- Setup: Call script file through umbrella program / shell (in `R`: `Rscript`).
- Requirements: some way to call `R` / `Stata` from command line.

Interactive mode on servers:

Option 1:

Run umbrella program and R / Stata on server.

- This is how we use the CHSE server.
- Requires a lot of data traffic between remote server and local computer.
- Not possible for Exacloud server because it does not have a desktop environment.

Interactive mode on servers:

Option 2:

Run umbrella program locally, R / Stata on server.

- Requires little data traffic between remote server and local computer.
- Umbrella program needs to be able to transfer code / results back and forth between local computer and server.
- Possible for Exacloud, depending on umbrella program:
  - Rstudio: No
  - Stata: ?
  - Emacs: Yes :)

Non-interactive mode: a simple example

1. Write `.R` file (*example1.R*):

   ```r
   x <- 1:1000
   summary(x)
   ```

2. Evaluate `.R` file using `Rscript`:

   ```
   Rscript example1.R
   ```

Non-interactive mode: processing R markdown file

- An `.Rmd` file has text and source code.
- `knitR` evaluates the `R` source blocks and creates `markdown` file.
- In interactive mode, simply call `knit(file.Rmd)` in `R`.
- Does not work in non-interactive mode because `Rscript` does not accept `.Rmd` as input file.
- Solution: write `R` script file that calls `knit` to evaluate `.Rmd` file.

Non-interactive mode: processing R markdown file

1. Write .Rmd file (*example1.Rmd*):

~~~
Example markdown file

```{r}
 x <- 1:1000
 summary(x)
```
~~~

2. Write .R file that evaluates .Rmd file (*master-knitr.R*):

```
library(knitr)
library(rmarkdown)
knit(commandArgs(TRUE)[1])
```

3. Evaluate .R file using Rscript:

```
Rscript master-knitr.R example1.Rmd
```

Non-interactive mode on Exacloud: HTCondor

- Purpose: Efficiently allocate resources to processes that run on decentralized computing system such as Exacloud.
- Basic usage is pretty simple:
  1. Write a submit file that tells `HTCondor` which program to run.
  2. Submit the request to `HTCondor`.
- There is a lot we could do with `HTCondor`:
  - Request memory for job.
  - Run script files in different directories.
  - Use macros, conditionals, ...

Non-interactive mode using HTCondor

1. Write .Rmd file (*example1.Rmd*):

~~~
Example markdown file

```{r}
 x <- 1:1000
 summary(x)
```
~~~

2. Write .R file to evaluate .Rmd file (*example1.R*):

```
library(knitr); library(rmarkdown)
knit(commandArgs(TRUE)[1])
```

3. Write HTCondor file to evaluate .R file (*examples.htc*):

```
Executable          = /usr/bin/Rscript
Arguments           = "master-knitr.R example1.Rmd"
```

4. Run file:

```
condor_submit examples.htc
```

Complete sequence to evaluate .Rmd script on Exacloud:

1. Submit file to `HTCondor`.
2. `HTCondor` calls `Rscript`.
3. `Rscript` evaluates `master-knitr.R`.
4. `master-knitr.R` calls `knit` to evaluate `.Rmd` file.
5. Resulting markdown file can be downloaded and exported to html / pdf.