Transitioning from the desktop to cloud computing for teaching

Carl Boettiger
Ben Bolker
Julien Brun (brun@nceas.ucsb.edu)
Lauren Hallett
Matthew B. Jones (jones@nceas.ucsb.edu)
Mark Schildhauer (schild@nceas.ucsb.edu)
Mike Smorul
Tracy Teal
Ethan White

Abstract

Introduction (BMB)

MPS– definition of Cloud Computing from NIST (is this agreeable to all?): "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."–http://www.nist.gov/itl/cloud/

- Cloud computing enabled today due to advances in virtualization (VMs) and containerization (e.g., Docker) to create light weight virtual hosts
- now is the time to transition to teaching in the cloud
- current and past problems that motivate this transition,
- for 25 years there's been a computer in the corner that says 'Do Not Touch'
- we buy hardware that's soon obsolete, or there's not the staff to support locked down computer labs are no longer ...
- we can do better (Hannay et al. 2009)
- compute resources exist at reasonable cost
- we simply can't do many of our analyses on our desktops
- in the class room advantages in uniformity and transitioning to scalable computing for teaching
- we will be teaching skills that are useful in data driven research and industry
- logistical convenience for teaching
- there are many ways to access remote/scalable computing, cloud like AWS, local HPC resources, agency-sponsored HPC resources
- The path we are describing is one path to the cloud or other shared resources (locally hosted servers, HPC)
- This technique is designed to be technology agnostic, however the public cloud is almost universally accessible. Much of what we discuss below is equally applicable to other platforms such as HPC; we highlight distinctions between the public cloud and other platforms where they exist.

- In this paper we will describe four stages that go from accessing the internet on a computer to setting up and running your own custom environments. More info on stages.

Stages of the cloud transition

- Stage 1 ("teaching in the cloud")
  - use simple front ends (RStudio/Jupiter/etc.), users don't realize they are on the cloud
  - moving beyond the locked down computer lab and teaching using readily acessible images. We will show some of the images and describe the concepts an instructor needs to master in order to each in the cloud.
- Stage 2: ("teaching the cloud (basic)")
  - Empower via the command line, people consciously choose the cloud, possibly just run a new instance
  - Move to community standard images
  - when and how this happens is domain- and data-specific
  - and can be enabled through cloud systems like Whole Tale (Brinckman et al. 2018) and Binder
- Stage 3: ("teaching the cloud (advanced)"):
  - the power of multiple instances, configuring new VMs via containerized environments (e.g., via Docker)
  - New trainers should start at level 2 or 3
- Stage 4: ("parallel computation"):
  - distributed and parallelized computation, high-performance computation

Stage 1: Teaching in the cloud (msmorul)

Teaching "*in* the cloud" means using cloud/remote services for teaching (typically) introductory/undergraduate quantitative material. The goal is for the local vs. remote distinction to be largely transparent to the end-users (students); they only know that they have to go to URL (xxx) in order to use R/Python whatever. (Instructors of stage 1 courses need to start at stage 2 so they have basic administrative/troubleshooting skills required.)

Advantages

- make instructors' lives easier
- accessibility
- uniformity of the students' work environments
- usage via pre-defined systems (e.g., genomics, Rocker, etc.)
- improved pedagogy for students, learn to work in clean environment
- empower/lower barriers for later transition to the cloud/remote computing
- avoid command-line bullshittery, but get buy-in of remote use for later

Disadvantages

- licensing/usage/copyright restrictions by institutions costs?
- end-user systems not configured to work independently
- good set of "teaching images" not yet available

Learning objectives

- **mostly** not remote-computing-related: remote computing is a means to an end

- (some) awareness of remote computing (data staging, etc)

Examples

- Duke intro stats
- Berkeley: Jupyter/Python in the cloud
- Google Group of folks using JupyterHub for teaching https://groups.google.com/forum/#!forum/jupyter-education
- SESYNC viz course - local vm's

Stage 2: Teaching the cloud (basic)

This an intermediate stage; target audience includes (i) trainers who want to teach in the cloud; (ii) researchers who want to do *basic* scalable computing.

Advantages

- independence/self-support
- easy scaling for simple problems, lab-level scaling
- farewell to sticky notes
- leave GUI or batch job running on the instance

Disadvantages

- Cost
- Moving data to/from these resources
- Figuring out how it works

Examples

- point to DC "how to spin up your own instance" lesson (not taught)
  - https://github.com/JasonJWilliamsNY/cloud-genomics/blob/master/lessons/1.logging-onto-cloud.md (still needs updating)
- Whole Tale to spin up Docker instances via the web (Brinckman et al. 2018)
- Titus Brown's short courses http://angus.readthedocs.org/en/2014/day1.html
- iPlant Atmosphere on Jetstream
- EDAMAME course https://github.com/edamame-course/2015-tutorials/blob/master/final/2015-06-22-EC2_Startup.md, https://github.com/edamame-course/2015-tutorials/blob/master/final/2015-06-22-EC2_Connection_FileTransfer.md

Learning Objectives

- Standup own images
- the concept of on-demand resources (dynamic nature of cloud resources cpu/memory)
- Understand how to locate and acquire remote resources - local and public cloud
- understand pricing / cost tradeoffs

Stage 3: Teaching the cloud (intermediate/advanced scalable computing)

Target audience: researchers and educators, who need to customize images for their own use or do heavier/more complicated computing: postdocs, advanced students. Once students...

Multiple Goals: * Customizing or creating images for teaching or research * scaling processing across multiple images

Advantages

- Customization
- Enabling cross-platform testing for developers/maintainers
- parallelization is a benefit, but requires a higher level of understanding (can hide some of the difficulty for trivially parallel jobs, especially if libs are parallel enabled)
- Enables/enforces reproducibility/documentation
- HPC can be seamless when proper frontends are available (RStudio, Jupyter)
- cloud motivates the command line

Disadvantages

- logistics: administering instances, funding, hacking/security
- some will go away as stuff matures, but ...
- toolchain not installed locally; people don't learn how to do the installs
- HPC tools: different stacks, schedulers, data stores, etc.
- HPC and cloud barriers are different but both problems
- back-end technologies are rapidly evolving
- possibly not optimal for "Big Data" challenges

Learning objectives

- conceptual understanding (how does it work? how do I log in?)
- spinning (locating and choosing) up multiple instances
- choice of platforms (Amazon, Docker, Azure, iPlant, etc.)
- ssh and basic command line stuff
- security basics
- payment models
- `screen`
- transition to the cloud:
  - leave GUI running on the instance
  - single R CMD BATCH
  - multiple R CMD BATCH
  - multiple instances
  - security, key mgt, etc
  - ... ? learn to spawn multiple instances, distribute, etc.
  - distributed vs. parallel computation

What are the pre-req skills & concepts?

Examples

- NGS course
- Data Carpentry metagenomics

- need more practical, concrete examples/context

Stage 4: Parallel computing

Target audience:

Goals:

- ...

Advantages

- ...

Disadvantages

- ...

Learning objectives

- ...

Examples

- ...

Conclusion: exhortation to excel (not microsoft), rah rah rah

Prepare students for real-world problems; enable research collaboration multiple environments, but allow students to move relatively seamlessly between them.

Boxes (ideas)

- Box 1: basic glossary/concepts (instance, "spin up", HPC, ...) (do this later)
- Box 2: (MPS) scalable alternatives to the cloud. Availability is *highly* variable. Costs and benefits (cost, flexibility, admin load ...) have to be considered carefully.
- Box 3: Gaps (EW)
  - data storage solutions
    - temporary storage solution, not an archive
    - objections to S3: not well-integrated with EC2; specialized data transfer modes
    - need user-level answers/best practices/basic concepts
    - use KNB (NCEAS data repo)
    - Center for Open Science (workflow repository)
    - don't want to mandate openness?
    - security
  - all technologies are rapidly evolving
  - funding and prep for the cloud is lacking; need HOWTO teach and setup and acquire the cloud

- Box 4: recommendations (EW)
  - people should start doing this stuff in the classroom!
  - encourage campus/local IT to support virtual environments (not just classic HPC model)

References

Brinckman, Adam, Kyle Chard, Niall Gaffney, Mihael Hategan, Matthew B. Jones, Kacper Kowalik, Sivakumar Kulasekaran, et al. 2018. "Computing Environments for Reproducibility: Capturing the 'Whole Tale'." *Future Generation Computer Systems*. doi:https://doi.org/10.1016/j.future.2017.12.029.

Hannay, J. E., C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson. 2009. "How Do Scientists Develop and Use Scientific Software?" In *Software Engineering for Computational Science and Engineering, 2009. Secse '09. Icse Workshop on*, 1–8. doi:10.1109/SECSE.2009.5069155.