2nd Fantastic Futures Conference
Pre-Conference Workshop: Cloud Services Primer
2019-12-03, Zac Painter, Stanford University

Lesson Material Credits:
The Carpentries (Data Carpentry for Genomics)
Various Lectures from CS349D at Stanford
Discussions with IT and HPC staff at Stanford

What is Cloud Computing?
Essentially, computing with large systems without direct user management
You define your parameters and requirements, and the system takes it from there

Have we done this before?
Sort of! Mainframe computing used to be standard before personal computers were mainstream
Despite how powerful personal computers are now, our modern computing needs can outstrip them
For large computing jobs now, you probably need some kind of extra power than your local machine

What are my remote computing choices?
Local Machines/Clusters (e.g., Farmshare/Sherlock at Stanford, MGHPCC in Massachusetts)
Open Research Compute Networks (e.g., XSEDE, Open Science Grid)
Commercial Cloud Providers (e.g., Amazon Web Services, Google Cloud, Microsoft Azure)

What can I get from a cloud? Some or all of the following stack…

Software as a Service (SaaS)
Centrally-hosted, Subscription-based licensing model (Communication Services, GitHub, etc.)

Platform as a Service (PaaS)
Allow you to work with applications without other barriers (Web Applications and Servers, Databases, etc.)

Infrastructure as a Service (IaaS)
APIs that allow customers to access physical resources (Virtual Machines, Storage, Networks, etc.)

Cloud/Cluster Architecture
Servers with CPU/GPU, RAM, Disks live on…
Racks of about 40-80, which are combined into…
Clusters of those for a data center

Data Lakes
Repositories where raw data is stored and manipulated
More important as compute is moving to where data is

Other Cloud Differentials

Edge Computing brings high traffic compute locally
Fog Computing works locally, using Internet to route

Public Clouds open to anyone to rent
Private Clouds only for members of specific group

General Commercial Cloud Pricing Schema
Compute pricing usually pay-as-you-go (especially IaaS)
No minimum/up-front fee
Most tenants not at 100% use – can be oversold?
Economies of scale for purchase/power/manage machines
Not all machines will be the same, some old – some new

TCO = capital (CapEx) + operational (OpEx) expenses

Operators perspective (Hardware dominates the cost)
CapEx: building, generators, A/C, compute/storage/net HW
Including spares, amortized over 3 – 15 years
OpEx: electricity (5-7c/KWh), repairs, people, WAN, insurance, …

Users perspective
CapEx: cost of long term leases on HW and services
OpeEx: pay per use cost on HW and services, people

The $64,000 Question… Can you build it cheaper?

The Answer? Possibly, depending on your scale and scope
GPU time is a limiting factor; those resources tend to be expensive
Need to factor in some cost for local maintenance and support

HPC Clusters
Often the cheapest solution if you have it
Usually fast local help and support options
Generally fewer privacy/security concerns than public cloud

Most put some kind of limits on what you can do; less flexible
Storage, Power you can use at once, or Process time often limited
Most are shared, so you need to use a scheduler (e.g., Slurm)

Open Researcher Clouds

Often supported by national research directives (e.g., NSF)
Usually free or low-cost for researchers with those grants

Extreme Science and Engineering Discovery Environment (XSEDE)
Works similar to HPC Clusters: you need to run schedulers
You need to bid for time and allocations like a grant
Upshot is that you can bid for almost anything you need

Open Science Grid (OSG)
Runs off of shared systems at partner institutions; no scheduling
More limited in overall ability, but a strong choice for some tasks

Commercial Comparison

Overall:
Compute prices roughly the same across all three majors
Service offerings (e.g., computing, storage) broadly similar
Very hard to support infrastructure for multiple vendors
Transfer from one platform to another is slow and/or difficult
You pay for renting the computer/whatever, whether you use it or not

Amazon Web Services:
Probably largest provider at Stanford
Probably the provider who has invested the most in education/research

Google Cloud Platform:
First to sign at Stanford
Tends to be more business or third-party service driven
Probably easiest to use with Kubernetes?

Microsoft Azure:
Generally business/enterprise driven
Usually slightly more expensive instances than the other two
If you are Windows driven, this might be cheaper/easier

Useful Unix commands for you to know

ssh user_name@address : How you log into most environments
whoami : Shows what user and computer you are
pwd : Shows you where you currently are
ls : Lists the contents of a directory

df -h : Shows the space on the hard drive
top : Checks stats of the current system
ps : Shows all processes in current system

tmux … : Allows you to keep running jobs when disconnected
screen … : An older alternative to tmux on some machines

sudo apt upgrade : Common command to get things ready to install
sudo apt install … : Common command to install programs

CAP Theorem
In distributed systems, choose 2 out of 3*
No network system safe from partition failure
Choice is really between C & A in the event of partition-failure
Partitions less common now than before, but still a question
Serverless computing is increasingly popular
BASE and ACID are similar ideas

Consistency
Every read returns data from most recent write
Failure can be from bad updates or multiple locations
Availability
Every request executes & receives a (non-error) response
High vs Perfect is generally the major choice
Partition-tolerance
The system continues to function when network partitions occur
If you have partitions, you can't have perfect and consistent

Containers and Orchestration

PaaS Services that allow you to replace VMs with portable "containers"
Allows for some transfer of items from one user/cluster to another
If you need to move from one environment to another, this is how to do it

Orchestration gathers containers and deploys them automatically
Docker, Mesos, and Kubernetes are the three largest current players

Security & Privacy

Theoretically, service provider can access anything uploaded at any time
Encrypting data is a solution, but adds extra work

Terms of service sometimes silent on question of "who owns the data?"
What happens if hardware collapses, or the provider shuts down?
Clouds can have insecure APIs or Infrastructure, creating risk