Restricted: CU Cloud case-studies

Three Cornell-based research-specific cloud case studies, from July 2014 through April 2016.

See also

- IaaS: Cloud computing, for research

Case study 1, July 2014

Cornell researcher, info from July 2014.

Brevity is not my strong point, but let me give it a go -

I predominantly use spot market instances offering the highest CPU power (either cc2.8xlarge or c3.8xlarge) with customized AMIs that I built off of Amazon standard AMIs for 64-bit Ubuntu. These AMIs are customized to mount my home filesystem from a file server in my office at Cornell via SSHFS over the Internet. Once they are booted and my home directory mounted, I can log into any of the Amazon spots I have running and they appear to be no different than my local workstation(s) at Cornell. I either use the systems interactively, or I farm out jobs to them using a simple dispatching system I wrote myself. If I'm using them interactively, I will typically start shells within "screen" so I can disconnect and reconnect without disrupting something that is running.

I start using Amazon when I have a lot of the same type of thing that I need to do. In parallel computing terms, this may be referred to under the term "query splitting". Specifically, I may have 1000 different samples that all need the same analysis process applied to them independently. I work out some of the details of the analyses first on my local system(s). Then, I farm out identical commands which operate on each of the different sample inputs. All of the data lives on my local system and is accessed over the network. In some cases, I have reprogrammed the software I have developed to do certain analyses in order to directly read pbzip2 (parallel multi-threaded version of bzip2) compressed files so that my jobs are not I/O bounded by the network mounted filesystem.

This has worked exceptionally well. The motivation for this rather involved time investment, done several years ago now, was primarily the refusal of my supervisor to pay for our large datasets to "live" in the cloud on EBS volumes where I could access them at faster I/O speeds uncompressed. At this time, I store very little data in the cloud long term, often just my custom AMI on a EBS volume and a few previous versions of it. This, in combination with using spot instances and timing the spot market well (weekends are good), has saved us thousands of dollars easily. Unfortunately though, the savings from this cost-effective use of the cloud is not something my supervisor can readily understand, she can only see the bill for the money that is spent and that always seems to her like an "extra" expense. On months where I am using the cloud I may run up a bill of $150-$300 dollars. Some months I don't start a single instance and only pay for the EBS store and it is as little as $25/month. Occasionally, when the spot market is unavoidably high priced and there is a lot I need to do urgently, I have run up bills over $500. Early on a few years back I hit $1000 a couple times, but it was after that that I worked out many cost savings approaches such as those I described above. Prices were also higher back then and instances less powerful than current offerings.

There - that was brief for me. Believe it or not, I edited that down a bit. :P

If you want, I can get as many past receipts as Amazon has available, I may have most of them saved somewhere even, and send them to you. It would show exactly what resources I used, for how long, and when, and what my monthly bills were like. One thing that is probably purposely not so clear in Amazon's docs but I discovered only after using AWS a bit is that for spot-instances you are billed at the fluctuating market rate in real time, not at your full bid price per hour. When the market rate exceeds your bid price, you lose your instance(s) immediately as if the power were cut. Often, I will bid something like $1.5/hour or $2/hour, but the market rate will be $0.29/hour and stay that way throughout the time I'm using the instances. That is where the massive savings come from. One has to be prepared for the possibility of losing the instance though. Often I can arrange my workloads such that that is ok. The spot market and spike and run up to several dollars an hour for just a few minutes which bumps a lot of people off and the price then falls. I wonder sometimes if somebody is doing that on purpose. When losing an instance will due to a short spike will be a problem, I will bid even $10/hour so that I don't get bumped, but then I'll watch the market price like a hawk. If it sustains high prices for more than just a few minutes, I'll drop the instances myself.

Let me know if you want any more details or if you want to chat in person it further or about what Cornell is brokering. I think it is an excellent idea for Cornell to broker a deal with Amazon.

Case study 2, March 2015

Cornell researcher, info from March 2015.

I am running Bayesian models on quantitative genetic data. Typically, when dealing with real data sets one workstation is enough. However, when I am analyzing multiple data sets (for example, currently 100 simulated data sets to be analyzed with a number of different models), I need to monopolize resources of multiple multi-core servers for 12 to 36 hours. In these kinds of cases, I use AWS. I have a disk image with my data stored in AWS (around 30G, at a cost of ~$0.50/month). I request c3.8xLarge or cc2.8xlarge compute-optimized instances on the spot market. I use the Amazon Linux AMI. This keeps the costs down (I wait for the spot price to fall below 30 cents/hour, which happens most of the time), at the expense of getting the instance shut off if the spot price goes above my bid price. I bid at the official price + 1 cent, which seems to be good enough. I have only lost a couple of instances over the last year and a half. When I run my instances, I activate them using the web interface, make a data disk for each instance from my data disk image (EBS volumes cannot be attached to more than one instance), and run a shell script in the instance (the shell script runs in the background). The shell script reads the data and unmounts the data disk (so I delete the data disks soon after the jobs start, again to save on cost). Several simulated data sets are processed at a time. Once a batch is done, the results are exported (via piping tar output to ssh — seems to be the most stable solution) to a "lab" server. This is so that if an instance is lost, not all results have to be re-generated. Data transfer costs are about 1/3 of the compute costs (1/4 of the total). I do not use more than 10 instances at a time (20 is the AWS limit) because it seems like running more puts too much pressure on the "lab" servers. The amount of data transferred is on the order of 20G per instance (some analyses it's only 2G), in something like 15 installments. Once all the analyses are done, my script shuts down the instance which then terminates. This way I only pay for the exact time I need.

Case study 3, April 2016

Cornell Lab of Ornithology, info from April 2016.

At the Cornell Lab of Ornithology, we are using data collected by citizen scientists to understand where bird populations are and how they move during migrations. The goal of this analysis is to produce high resolution (3km) estimates of the spatial and temporal abundance of species populations at weekly intervals across North America. A visualization of the estimated distribution for Tree Swallow can be seen here [http://ebird.org/content/ebird/wp-content/uploads/sites/55/TRES_1s.gif] The information we extract from this data is being used to identify the environmental drivers that shape species' distributions and for developing science-based management policies.

The observational data for this analysis come from eBird.org, a project run by the Cornell Lab of Ornithology that engages volunteers via the Internet and mobile apps to collect bird observations. We use statistical and machine learning models combining the eBird observations with remote sensing data from NASA to correct for biases in the data and fill in spatiotemporal gaps. Then we use the models to produce a comprehensive set of high-resolution estimates and summaries about the spatial and temporal abundance of species populations. Each job estimates a single species' population distribution across North America.

This is a CPU intensive modeling task (>5000 CPU hrs per job) with relatively small input data (~ a couple of GB) and larger, but still relatively small output data sizes (~ 20-50GB per job). Our workflow is a multistep implementation of map-reduce using Streaming Hadoop 2.6 with some Oozie and AWK scripts to automate the workflow. The statistical analysis is carried out in R, the statistical computing language. Using the cloud to run this workflow has allowed us to drastically reduce the wall-clock time to complete jobs. We have run this workflow on Amazon AWS/EMR using the Spot market (with Spot Fleets) with r3.4xlarge head node and (up to) 230 m3.2xlarge worker nodes. Currently, we are using Microsoft Azure HDIsight (LINUX Hadoop clusters) using clusters with D14 head nodes and 150 D4 worker nodes (1200 cores). The workflow takes 3-5 hours per job (species) on the Azure HDIsight.

N.B. Pricing information not included because pricing has been changing so much. Amazon's Spot Market is the classic example. But its more than the market price, Amazon continues to include new ways of bidding on the market - Spot Fleets made it possible to acquire larger blocks of processors and there is another option where you can pay a little more on the spot market and insure that you keep your processors for at least 6 hours (I forget what this is called!). Azure prices have changed too. For us, the critical pieces of information have been run time and resource requirements, and the demonstration that we can utilize these cloud resources at the necessary scale, etc. For example, it was useful for my group to demonstrate that we could we could get enough processors for a long enough time period to run our workflow using the spot market.