Compute Servers for Teaching Big Data

Clark Fitzgerald

January 14, 2025

Abstract

I've taught an upper division statistics "Big Data" course at UC Davis and CSU Sacramento and have gained some perspective on what works well for this course. My goals with the course are to get students comfortable with using remote machines and to do realistic analysis of data sets that don't fit in memory, typically on the order of 100GB or so.

I've tried what feels like an excessive number of ways to run the server for the students to get the 'server experience': Campus supported Linux cluster (SLURM) AWS EC2 - student accounts and Jupyter Notebooks Google colab NSF Jetstream cloud (Access) Physical server in the corner of my office (current solution!)

1 Introduction

I developed and have taught the course Stat 129, titled 'Analyzing and Processing Big Data'. The catalog course description is:

Statistical analysis of large, complex data sets. Topics include memory efficient data processing, the split-apply-combine strategy, rewriting programs for scalability, handling complex data formats, and applications such as statistical learning, dimension reduction, and efficient data representation. Students will access data and run code on remote servers.

Our general approach to data science education is to provide students with real world data sets and ask open ended questions, squarely in the tradition of Deborah Nolan and Duncan Temple Lang [1]. Indeed, they were our teachers and mentors.

Many of the techniques that we teach in this class work equally well on a modern laptop as on a server.

It's essential that students access data and run code on remote servers, because there is a qualitative difference between running code on a laptop that's sitting in front of you and a machine that you never actually see, that you can only access through ssh, secure shell. I teach bash, the classic command line, because it has proven to be a durable technology; it's been around since the 1970's, and it isn't going away any time soon. In general, students benefit when instructors focus on proven core technologies, rather than on what happens to be popular this year, because they come away with skills that they can apply in many different contexts.

My goal with the command line is for them to do it enough that they get over the initial fear that comes from moving from a Graphical User Interface (GUI), so they will be more comfortable in the future. Thus we need an actual remote server that students can log in to so that they can perform the tasks. The problem is that my job doesn't provide such a server. If I want students to have this experience, then it is on me to find and manage such a server. You may be in a similar situation of wanting to teach a class that relies on a server, and are wondering how should you do it? If so, read on.

1.1 Which Data?

The class is called 'Big Data'. For this term to have any meaning, we need to define what we mean by 'big'. For the purposes of this class, a 'big' data set is one that is large enough that one cannot load the entire data set into a laptop's memory at once. Once a data set is this large, it's not possible to simply load the whole thing and start working using R or Python. My goal is to choose data sets that are comfortably over this threshold, such that the techniques we're learning are motivated, but not gratuitously large such that answering a basic question takes many hours. Practically speaking, from 2019 until the present I try to choose data sets that are on the order of 10 or 100 GB. The data analysis tasks will typically take 1 to 10 minutes if done in serial, and will be an order of magnitude faster if done in parallel.

While teaching this course, I have to select data sets. Everything that we do in the course is based on working with a legitimate, unwieldy, data set. In general, I'm looking for the following characteristics:

1. **Publicly Available** data should be available for legal public download.
2. **Size** large enough to motivate the computational techniques we're learning, but no larger.
3. **Format** common format, at least one tabular and another nested. Typically we will do a CSV/TSV for the tabular, and XML or JSON for the nested.
4. **Background Knowledge** a college student should be able to understand what the data actually represents with minimal effort.

Most of these principles I've violated at one time or another, and the difficulties became clear later.

Amazon Open Data `https://registry.opendata.aws/` has provided a reliably good source of large data sets fitting these criterion. Another good source is the US government `https://data.gov/`.

Once I find a data set to work with, I download it to the server and massage it into the form where I want my students to see. A large part of the course is 'cleaning' the data, preparing it for analysis by filtering, imputing, etc. TODO: cite However, it requires judgement as to which cleaning tasks are good for the students learning, and which are merely tedious or too uncommon.

2 server options

I'm an Assistant Professor in the Mathematics and Statistics Department at CSU Sacramento, a regional university. I have some professional background with computers and programming. In particular, I've done some server administration, and that experience has definitely proven useful when teaching this class.

Our department plans to offer Stat 129 once a year for the foreseeable future, so we need a long term solution for a server. In a particular year we'll have one instructor, 15-25 students, and no TA's or other support. I don't need a large machine for any purpose other than teaching. Please keep in mind the context I'm coming from when making the following recommendations, as they may not be appropriate for your situation.

What follows are the server options, ordered by my personal preference.

What do you actually need to teach a class like this?

2.1 Best option - Dedicated Support and Infrastructure

If you're in the lucky position of having compute infrastructure with dedicated support in your department, college, or university, then this is an ideal option for teaching classes. You simply ask the system administrator for what you need, and they'll make it happen. The only hard part is knowing what it is you need.

While teaching at UC Davis, the system administrator's helped me with the following tasks related to the class:

- Creating user accounts from a student roster.
- Installing R, Python, along with various packages and libraries of software.
- Setting up and configuring an open source database (Postgres) for use by students.

A systems administrator can do many other things, of course. Just ask!

I've used more traditional supercomputers with job queueing software (SLURM) for teaching this class, but I feel that this is not ideal for the class, because of the amount of overhead, and the abstraction. For example, one topic that we cover is reading the output of `top/htop` to identify a bottleneck in a shell pipeline that's streaming hundreds of GB of data. It's possible to do this with job queueing software where the job is running on a worker node, but it takes either more advanced knowledge, or something specific to that system. Whenever possible I try to teach the tools that are the most common and widely applicable, to better serve the students in their future careers.

I feel that timely support is much more important the physical compute power, especially if the class is small.

2.2 Option 1 - Personal Physical Server

A personal physical server is either in a rack in a server room somewhere on campus, or else a large box in an office. If it's in a server rack then you can generally get a machine with higer specs, better internet connectivity, and it's not cluttering up your space. If the machine is in your office, then when you launch a big job the machine will heat up and the cooling fan will turn on, which provides a very real connection to the amount of data being processed for the students in office hours, which can be fun.

This is my current solution, after having explored and experienced the others described in this article. I find that it is the most pragmatic solution, requiring the least effort on my part.

The major advantages of this approach are cost and simplicity. Regarding cost, I've found it easier to fund a one time purchase of a few thousand dollars compared to finding a sponsor who will commit to the ongoing variable subscription fees of the cloud services, even if the total cost would be lower for the cloud subscription. It's simple because you can set it up one time, and then use the same machine for several years. I myself am the systems administrator, so I can install whatever I want, whenever I want. I didn't need any specialized knowledge beyond the basics. I don't have to worry about any creating and maintaining cloud images. The downside of this is that I am my own support. If I spill coffee on your machine, then I'm in trouble. If something goes wrong, I'm the only one who's going to fix it.

For a class of 15 students, I found that a single 32 core CPU with 128 GB of memory was just sufficient for students to experience the class assignments in the way I intended. I always recommend that students do their homework well before the due date, because if a handful of users are simultaneously trying to use parallel jobs with all available CPU cores, then they won't see the speedup they are expecting. In practice it happened occassionaly that students who waited until the last hour to do an assignment didn't get to see their final result, but they weren't surprised, because I had warned them many times and they knew to expect this. I've used a 16 core CPU in previous classes, and found that it was not sufficient. With more students, I believe that students would feel crowded on the server.

One technical detail to be aware of is that you must provide remote access so that others can log on to the machine. Campus IT handled this for me. I simply asked them for a hostname, and they took care of it. Our students need to be either on campus or at home on the campus VPN (Virtual Private Network), and they can login via ssh, for example: `ssh fitzgerald@nsm-stats.csus.ed`. I would recommend against setting up a physical server at your home, because this step could be tough depending on your internet provider, and there's no reason you should pay for the electricity instead of campus.

The physical server feels out of date, and lacks the cachet of being 'the cloud'. Despite that, it provides a legitimate and simple way to get students using a remote server. From a student's perspective, there's no difference between `ssh` to this machine, or to a virtual machine that you own that's running on the cloud.

With either cloud option, you have two ways to provide students with access to a server: they can create their own virtual machines, or they can use a machine that you provide. I've used both of these options, and they both have a place, depending on the goals of the class.

The first option provides more true 'cloud' experience. The students can have their own account with the cloud provider, they create their own virtual machines which they have complete control over, and they delete them when they are done. They are the administrators to these machines with `sudo` access, so they can do anything they wish, and any mistakes they make won't affect anyone else. This would be a good option if the course is designed to teach them more about infrastructure, for example, you could allow them to implement their own web applications.

The second option allows students to focus more on the data analysis rather than the infrastructure. You, the instructor, create a virtual machine that's sufficiently large to accommadate all of the students simultaneously. You can either give the students SSH access, or allow them to access through a web interface, for example with JupyterHub or RStudio Cloud. The downside here is that you need to pay for a large enough machine all the time, or else manually load balance by switching to smaller or larger machines as the demand changes.

2.3 Option 2 - Public Cloud

I have had generally an excellent experience using the publicly funded cloud, Jetstream2. The two big benefits of this model are that it's free for the students, and it's a relatively simple way for the students to get real cloud experience.. In this model, you apply for an allocation, Jetstream reviews and awards you a certain number of compute hours, you set up the accounts. To do the work, each student creates their own virtual machine and uses it to do the data analysis. I applied, was approved, and had everything set up in less than a week.

The website `https://jetstream-cloud.org/index.html` states: TODO: cite

> Jetstream2 makes cutting-edge high-performance computing and software easy to use for research regardless of your project's scale, even if you have limited experience with supercomputing systems.
>
> Cloud-based and on-demand, the Jetstream2 environment also includes discipline-specific apps. You can even create virtual machines that look and feel like your lab workstation or home machine—but with thousands of times the computing power.

The Jetstream2 user interface and options are far simpler than commercial cloud providers.

Two disadvantages of Jetstream2 for teaching are the security model, and the lack of safeguards. Jetstream2 seems to be focused on allowing individual researchers and small groups, and enabling them to do projects with more compute than they would otherwise be able to. What I did, teaching a class where every student gets access, is certainly not the primary use case. When you're awarded an allocation, every user gets complete access to every part of the account. Students have administrative access to everything the instructor creates, and every other students machine. There are no limits on how much compute an individual can use, so it's possible for one user to use up all the credits. In other words, the only thing that's stopping them from doing something bad is their own awareness and respect for others. I mostly got around these issues by setting very clear expectations for behavior, with the understanding that this whole idea is based on trust. Even so, a student unintentionally left a large machine running for several weeks. I wasn't dilligently monitoring, so this used up around 25% of our compute credits. Fortunately, we still had plenty to get through the course.

A technical detail you'll have to handle is how students will access the large data sets. Jetstream2 provides data storage through the Manila Filesystems as a service, and this worked well for sharing large data sets across several different running instances.

Another disadvantage for my use case is the uncertainty of how many years I'll be able to rely on it to continue offering the course, as I need to renew it annually. Jetstream2 is not designed as a long term solution for institutions offering the same course year after year. I expect they would renew every year, but there's no guarantee of this happening. If they choose not to renew, then I'll need to look for another solution.

2.4 Option 3 - Commercial Cloud

Several companies offer commercial cloud services. I focused mainly on Amazon Web Services (AWS), the biggest and most popular vendor, so that our students can get marketable experience that they can then put on their resume. My favorite feature of AWS is the direct access to Amazon's high quality open data sets through the `aws s3` command line tools. From an instructor's perspective, this eliminates the step of downloading the data and making it available to students.

The financial aspect is not ideal for students. These are commercial services, so a student must have a credit card and be prepared to pay for any compute time and storage that they use. In Spring 2022, I found that students on AWS can generally get the data processing experience for $20 or less in one semester. If they make a mistake, say using an expensive service and forgetting to shut it off, then they can easily run up an expensive and unexpected bill that can be tough for a college student to cover. In the past, AWS offered an educational program with free credits for students. The big selling point for me was that students didn't need to even provide a credit card. I used this program, but they discontinued it, leaving me searching for other options. For a busy faculty member, this is a waste.

Complexity and vendor specificity are also problems when teaching with cloud services. Amazon's EC2 (cloud compute server) "provides a wide selection of over 750 instance types optimized to fit different use cases." There are also hundreds of different services and several world regions to choose from. The services often have company specific names, which makes it difficult for an instructor to understand what they are. Indeed, the main purpose of the 'AWS Cloud Practitioner' qualification seems to be to understand all of these proprietary names and services. This vendor specific education is all well and good if Amazon is providing the training, however, at a public institution my goal is to focus on the overarching concepts and portable technologies that a student can take and apply anywhere, regardless of the specific vendor.

While I feel that the commercial cloud services aren't currently the best choice for me, they might be the best choice for you. If you already have experience with a particular cloud provider, or you wish to build that experience, then go for it. If your institution has some connection with a cloud provider, then it may be mutually beneficial to leverage those connections. For example, the company may provide cloud services for free in exchange for the ability to recruit graduates into that company.

I do believe that viable options can be built using commercial cloud services. For example, RStudio, Anaconda, and Snowflake offer cloud services that are essentially a higher level abstraction and user interface catered towards the data science R, Python, or SQL user.

3 Future Steps

If it was easier to set up and run a server, then I believe that more instructors would be interested in teaching this kind of class. For a lone instructor with no support, the barriers to entry are significant. I would welcome a solution that provides this capability publicly. It must have the following features:

1. **command line access** If this is present, then it should be possible to install and use all manner of command line software, including R, Python, and SQL.
2. **100 GB data storage**
3. **free or low cost** Free for students is ideal, of course. If that's not possible, then the cost over the semester should not exceed that of a reasonably priced textbook.

Based on our observations above, there are several other features that would be nice to have, and I list them here in order of preference.

1. **longevity** It's much easier for instructors to invest in a particular solution if they can be sure that it will be around in 5 years.

References

[1] Deborah Nolan and Duncan Temple Lang. Computing in the statistics curricula. *The American Statistician*, 64(2):97–107, 2010.