Compute Servers for Teaching Big Data

Clark Fitzgerald

February 4, 2025

Abstract

We compare and contrast different server options for providing undergraduate students with an authentic experience analyzing large, real world data sets located on powerful remote servers. These server options include institutionally supported academic computers, commercial cloud services, public cloud services, and personal physical machines. We briefly describe the course, and offer suggestions on when to choose which server option, based on the background of the instructor and the level of institutional support. Looking to the future, we suggest requirements for a broadly shared educational compute service that will allow more instructors to teach similar classes without the overhead of having to manage and administer their own server.

1 The Course

Stat 129, 'Analyzing and Processing Big Data', is an upper division elective course offered by the Mathematics and Statistics Department at CSU Sacramento, and is required for students pursuing a bachelor's degree in Statistics with an Emphasis in Data Science. 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.

Stat 129 uses technologies including bash, Python, and SQL to analyze large data sets, with the intention of exposing students to the common tools in data science. Students tend to be highly motivated, because many of them hope to land jobs requiring these skills after graduation. Stat 129 requires a programming prerequisite, since it is too advanced for a first programming course. Statistics majors take Stat 129 after taking 'Statistical Computing', an upper division course focused on data analysis using the R programming language, with content similar to Data Science in a Box, R4DS TODO: Cite.

The philosophy behind this class is to provide students with real world data sets and ask open ended questions, following in the footsteps of our teachers and mentors, Deborah Nolan and Duncan Temple Lang [3]. We have been teaching data science programming technologies in an academic setting since 2015, and we developed Stat 129 and have taught it each year since 2021. Feedback from this data forward approach in general, and this class in particular, has been overwhelmingly positive. In Stat 129 students ask real questions based on real data that resides on a real server, and for this reason several alumni have reported that this has been the single most practical and valuable class that they have taken in college.

1.1 Which Data?

For the purposes of Stat 129, 'big data' means a data set that is large enough such that one cannot load the entire data set into a laptop's memory at once. In a typical introductory data analysis class, a student reads a single data file into R or Python and proceeds with the analysis steps [1]. This works great if the data fits into memory, but if the data does not fit into memory, then it simply doesn't work. Data sets that are larger than memory require entirely different approaches, such as streaming through to filter out a subset of interest. Our goal is to choose data sets that are comfortably over this memory threshold, such that the techniques we're learning are motivated, but not gratuitously large such that answering a basic question takes an excessive amount of time. Practically speaking, at the time of this writing, a typical student laptop will have between 4 and 32 GB of memory, and data sets that are on the order of 10-100 GB on disk work well to motivate the techniques. We assign common data analysis tasks that typically take 1 to 10 minutes if done in serial, and are an order of magnitude faster if done in parallel.

When teaching Stat 129, we look for data sets that satisfy the criteria below.

1. **Size** The files should be large enough to motivate the computational techniques we're learning, but no larger, as described above.

2. **Publicly Available** The data should be readily available for legal public download. Special permissions, non disclosure agreements, and licensing all take

time away from teaching.

3. **Format** The storage format should be a common one that students will encounter in data science work, so that students can directly apply what they learn in class. Avoid proprietary, novel, or highly specialized data formats. Each semester of Stat 129, we work with data in a plain text tabular format such as CSV or TSV, and nested formats such as XML or JSON.

4. **Background Knowledge** A college student should be able to understand what the data actually represents. If it takes longer than a single class period to explain the core idea of what the data captures, then that data is probably too complex.

These criteria don't necessarily apply to more domain specific classes, for example, a bioinformatics class studying genomes would do well to focus on genomic data. Amazon Open Data[1] has provided a reliably good source of large data sets fitting these criteria. The US government[2] is another good source.

We often find interesting data sets that are close, but not quite what we're looking for. In this case, it may be possible to process the data set before providing it to students to make it better suit the goals of the course. As a specific example, consider the tax return data on nonprofits provided by the Internal Revenue Service of the United States[4]. We removed the XML namespace information in these files so that students learning XPATH can write `"//ActivityOrMissionDesc"` to find the 'activity or mission description' for a particular nonprofit rather than the cum-

[1] `https://registry.opendata.aws/`

[2] `https://data.gov/`

bersome `//*[local-name()='ActivityOrMissionDesc']`. The latter is distracting for students just learning XPATH.

The large data sets are the heart of Stat 129. To properly analyze them we need to provide our students with a compute server.

1.2 Why a Server?

Students must access data and run code on remote servers for a class on 'Big Data', because that's where big data is typically found: on remote servers. There is a qualitative difference between the experience of running code on a laptop with a monitor that's sitting in front of you, and on a machine that you never actually see, that you can only access through `ssh`, secure shell. Our goal with embracing the shell is to move students beyond the initial unfamiliarity and discomfort that users experience when they first move away from a Graphical User Interface (GUI). To access the large data sets and complete the assignments, quizzes, and exams, the students must login to the class server through `ssh`.

We teach the classic shell bash, because it has proven to be a durable technology. If one hopes to do anything with more advanced tools such as larger compute clusters or frameworks such Hadoop, then a foundational knowledge of bash is essential. TODO: cite In general, students in technology oriented class such as Stat 129 benefit when instructors focus on ubiquitous and proven core technologies, rather than on what happens to be popular this year, because then they come away with skills that they can apply in many different contexts.

2 Server Options

Which server option is best depends on the goals of the course, the level of institutional support, the number of students enrolled, the background of the students, and the background of the instructor. The overarching goal of Stat 129 is to do real things on real data sets, using professional software. Students use the server for every assignment and assessment for 15 weeks, so it makes sense to invest some time teaching them skills and tools with steeper learning curves, such as terminal based text editors. If a different class only used a server for two weeks of the semester to examine one particular data set, then it probably makes more sense to use a more approachable graphical user interface to the server such as Jupyter Notebooks [2], which will inform the best choice of server options.

Institutional support for teaching a class like Stat 129 varies widely. For us, there were no servers available to use for teaching at the department, college, or university level; however, the college did provide us with startup funds, which funded the experimentation that led to this paper. The Mathematics and Statistics Department at Sacramento State University 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, 15-30 students, and no staff, teaching assistants, or other support. We don't need a large machine for any purpose other than teaching.

We have some professional background in programming, and our server administration experience has proven particularly useful for teaching Stat 129. What follows are the server options, ordered by our personal preference. There's no one size fits all solution, and what's best for you may well differ from our prefernces.

2.1 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!

We've used more traditional supercomputers with job queueing software (SLURM) for teaching this class, but we 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 we try to teach the tools that are the most common and widely applicable, to better serve the students in their future careers.

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

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

The major advantages of this approach are cost and simplicity. Regarding cost, We'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. We myself am the systems administrator, so we can install whatever we want, whenever we want. We didn't need any specialized knowledge beyond the basics. We don't have to worry about any creating and maintaining cloud images. The downside of this is that we am our own support. If we 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, we 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 we intended. We 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 we had warned them many times and they knew to expect this. We've used a 16 core CPU in previous classes, and found that it was not sufficient. With more students, we 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. We 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`. We 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.

It's relatively easy and direct to automate certain grading tasks on a physical server. For example...

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. We'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 Public Cloud

We 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. We 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 we 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. We 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. We 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 our use case is the uncertainty of how many years I'll be able to rely on it to continue offering the course, as we need to renew it annually. Jetstream2 is not designed as a long term solution for institutions offering the same course year after year. We 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 Commercial Cloud

Several companies offer commercial cloud services. We 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. Our 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, we 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. We 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 our 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 we 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.

We 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 we 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. We 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 we list some of them here.

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] Mine Cetinkaya-Rundel. Data science in a box. `https://datasciencebox.org/`. Accessed: 2025-01-28.

[2] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks–a publishing format for reproducible computational workflows. In *Positioning and power in academic publishing: Players, agents and agendas*, pages 87–90. IOS press, 2016.

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

[4] Internal Revenue Service. Form 990 series downloads. `https://www.irs.gov/charities-non-profits/form-990-series-downloads`. Accessed: 2025-01-28.

4 Scratch

We'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. Our 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 our office (current solution!)