

Project2 Report1

Nathaniel Leake, 424003778

Time to complete Task: **8hrs**

Time Tracking Information:

1st day: 7:00am – 9:08am

2nd day: 12:00pm – 2:00pm, 4:30pm-5:00pm

3rd day: 7:00pm – 10:20pm

Total: 7:58

Recommendation of Platform

From the first time I read over the provided information, my instincts told me that the TAMU-based system was a far better initial setup for the client. Given that TAMU is offering unlimited computer racks at a stunning discount^{9a} and free, limitless bandwidth/traffic^{9b} for the startup's servers plus a choice between two powerful clusters, this offer has a strong front-face appeal. The one concern is scalability, which to some extent is quantifiable since the system will initially be running only in College Station and Austin. Holding to the provided estimate that within a month of the initial release 5% of the student population of the two major universities will be using the app^{10a}, by the end of this period the app will have approximately 6,000 users (given that there are around 120,000 students between the two universities). Only 10% of these users generate a significant amount of content per day^{10b}, and this subset can be automatically recognized and categorized, meaning that TAMU-based servers will work very well for the initial release. Of course, the adaption and use of a social app by a society can expand exponentially, meaning that if the company is successful it may need to scale rapidly to a larger database. However, this explosion in popularity can be predicted, and a plan can be laid out ahead of time for a quick transition to a larger, more expensive (less power per dollar) service such as AWS or Azure, or the company could build their own datacenter. It is best to start with the cheap, fully controlled TAMU-based system to minimize being weighed down by steep rates during early stages of adaption and to have directly access and control the servers' platform(s).

Estimated Expenses

I am going to begin by declaring that the amount of video content projected by the company is completely unrealistic. Unless the company refrains from storing videos indefinitely or corrects its estimates for what percent of users are content-generators they will quickly be dealing with more data than they can afford. Thus, I am working under the assumption most videos are not stored indefinitely.

Cloud-based price ranges vary radically. If you need 1TB of storage, for example, you could find yourself paying anywhere from \$2 a month to \$100 a month, with variable bandwidth and performance metrics thrown in. For Amazon's AWS, the average cost of storage is \$21 per month, and the price of downloading is \$50 per GB, which does not bode well for an app that heavily utilizes upload/download transfer of video content. Even with some of the cheapest cloud-hosted options I could find, the price for transmitting data to/from the consumer averages at ~\$15 per GB. When a single 5 minute video at full-screen resolution (~1920x1080 averages about 1 gigabyte, current B2B consumer cloud services are simply a bad option. Of course, the app will presumably be running primarily on phone screens with a lower resolution (~1280x800) and video quality might be significantly reduced, but the prices here are still very hard to balance. However, the TAMU-based database is offering free unlimited upload/download and inter-machine communication, and TAMU's bandwidth reaches a decent 1000 Mbps. Overall, the TAMU-based setup isn't free but it might as well be when compared to

other options. My estimation of the price of information storage is around \$17 per TB per month. A rough approximation of the cost of the TAMU-based setup, including storage, plus ~\$100 of upkeep per month (not including backups done by TAMU), plus \$800 for purchasing servers, plus growing pains from platform evolution/changes, comes out to a total sum of under \$1500 per month for all database-related expenses, which might be manageable for the startup if they are as successful as they anticipate.

Recommendation of Cluster

If the business decides to take the TAMU datacenter option for their initial release, they need to decide which cluster to use for their intensive video processing algorithm⁵. This algorithm is the most computationally expensive task for the company, and is also the trademark secret that needs to be protected. Because this processing algorithm needs to be run on every video that is generated⁴, it is the primary bottleneck of the entire content generation process. According to the provided information, “even the simplest of the videos requires computationally intensive image processing” with “lots of memory” and “powerful CPUs” to generate the final output^{4, 6}. Because of this, the cluster described in 5a is more appropriate, because the machines are better suited to handle these kinds of tasks. Based on the minimal descriptions provided, I feel it is a safe guess that each of the powerful machines can perform about 10x faster than a corresponding “weak” machine, resulting in the powerful cluster being about 2x as fast overall than the smaller one. The parallelization provided by the 16 smaller machines in 5b wouldn’t necessarily benefit the process, because these machines have fewer cores, slower communication channels, and no SSD. This is not to be confused with the storage of user data, which would be fine with using a multitude of small, cheap machines in a cloud network, and benefit from the accompanying redundancy and provided backups. However, for the company’s trademark algorithm, persistence of data is secondary to the generation process and its performance.

Further Reasoning & Counterarguments

The biggest argument against my current recommendation is that it requires a higher level of preparedness for the potential need to scale rapidly. TAMU has granted “unlimited” access, but TAMU’s datacenter itself is limited in size (since this part of the provided information is theoretical, I cannot provide an estimate). For example, Facebook’s datacenter stores on the order of 100 PB. With a service like Azure, the company’s database size could grow “automatically” as more users join and begin generating content. Also, using a cloud-based system would likely enhance the safety of the data. Even with TAMU’s nightly backups, all the data is still stored in a single geographical location (the College Station area), whereas cloud services will likely duplicate the data across multiple datacenters. Overall, the TAMU-based option is still a better choice for the reasons argued above. If the company does attain this level of success, they will presumably have the resources required to make the transition to a cloud based service or to build their own datacenter, like what Facebook did in 2010.

Assumptions and Deductions

In this paragraph I will discuss the metrics from which I reached my conclusions, and the assumptions/steps I took to attain those metrics. As I briefly mentioned above, the startup states that 10% of the users generate 10 videos per day, 70% generate 0.3 videos per day, and 20% generate nothing. Thus, the total number of videos generated in a day is $1.21 * N$, where N is the number of users. For 68,625 students at Texas A&M and 51,331 students at UT, the total number of videos created in a single day if the proposed 5% of the students are using the app is 7,200. Assuming videos have an average length of 3 minutes, this is about 5 TB of data per day. This is a gigantic number, considering that Facebook reads in about 600 TB (possibly as high as 1 PB) per day and operates at a global scale.

If all students at both universities adopted this app, it would be generating data at about one tenth the rate of Facebook – and that’s with only two universities. The current largest video platform, YouTube, generates 3 PB per day (72 hours per minute), and even Google’s datacenters can’t hold much over 1 Exabyte combined, although this number is constantly growing. Assuming best-case video compression and V videos created per day at 5m each, the startup will be creating $0.6*V$ GB per day, which is still a massive amount of data. Of course, users can only import media files that have been uploaded through an existing online service¹, but since the app actually generates new videos using this media, it is not limited by the amount of existing content on these other platforms.

If the company chooses to use the TAMU system to maintain control of their servers, they can expect the following breakdown of relative expenses: 0% disaster recovery (covered by TAMU), 5% or less for storage (TAMU rack space is 5% of market price), 1% for networking (theoretically covered by TAMU, but might need work), 16% energy/facilities (not explicitly covered by TAMU), 32% people, 38% for purchasing servers and 8% just for the overhead of setup/paperwork/etc. A pie chart of this data for visual presentation is included at the bottom of this report.

Once the client surpasses the capabilities of the TAMU datacenter or when their partnership deal expires at the end of the 3-year period, whichever comes first, they may want to build their own datacenter. If this is the case, a generic estimate of datacenter expenses is as follows: 29% people, 22% software, 12% energy, 11% servers, 10% networking, 7% storage, 7% disaster recovery, 2% overhead (source: Gartner). Another pie chart is provided below for these estimations, along with a pie chart of expenses if they decide to move to AWS (calculated using the AWS website, screenshot provided).

Although mentioned in the provided information document¹¹, I never touched on the topic of reducing latency for popular videos. The answer to this is obviously to just keep several copies of these videos spread across multiple servers (and across multiple geographical locations, once the company expands), just like what YouTube does for similar videos. The reason I skipped over this is that the fraction of videos for which this is worth doing is very small ($<1\%$ according to YouTube), so it does not impact the order of magnitude of required storage space. Similarly, the daily song list sent in from the users’ devices is not a concern, since this information can be represented as a list of song titles or IDs and the actual music is only processed server-side during the video generation algorithm. Because video files are so much larger than text files, keeping track of which a user has listened to in their entire life will probably take up less space than the number of videos they watch in a single day.

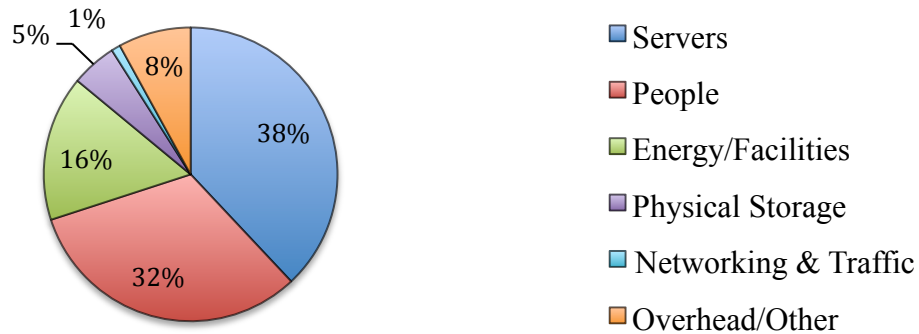
The provided information states that company’s code runs on the Ubuntu system⁸ and is written in Python. This statement indicates that not only do they have a working, functional prototype, but also they have a system setup for running this app’s backend (Ubuntu 16). This is why I left out cost estimations for developing the app. It is of course probable that the app will continue to evolve and that there could be resulting changes to the database architecture, but that doesn’t necessarily affect the choice of platform aside storage considerations and/or performance requirements.

Notes:

- Superscripted numbers are references to the **Provided Information** section of the homework PDF
- Sources are not referenced within the body of the report, but the order of URLs listed below matches the order in which sources are used in the report.
- It is fully possible that price estimates for storage requirements are significantly off due to the ambiguity of the provided information, especially considering that the values indicated in the assignment paper seem unrealistic.
- One of the difficulties I ran into during this assignment was when my editor crashed during day 3, resulting in about an hour of work lost.

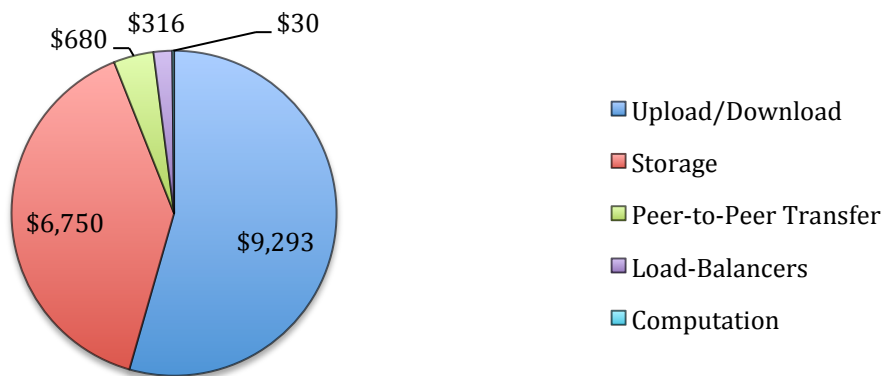
Expenses on TAMU System

Total: < \$1500



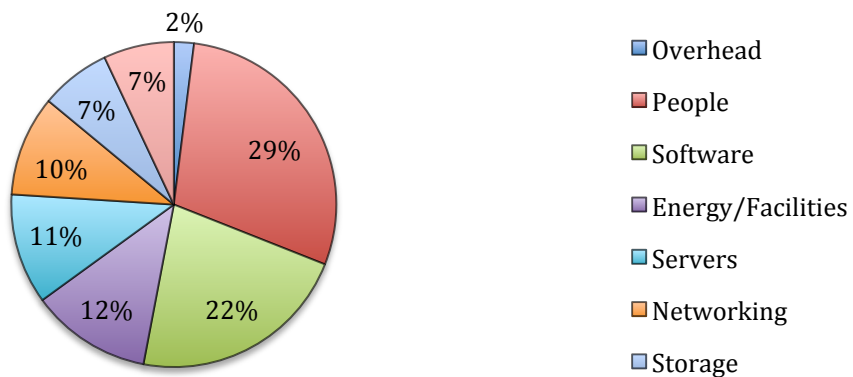
Amazon AWS Charges


Total: \$17,000 (after first month)



Gartner - Generic Datacenter Costs

Total: Depends on company growth rate



Services	Estimate of your Monthly Bill (\$ 16889.21)	
Choose region:	US-East / US Standard (Virginia)	Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free
 Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale for developers. Amazon Elastic Block Store (EBS) provides persistent storage to Amazon EC2 instances.		

Sources for Facts, Statistics, and Estimates

Most of the raw numbers and estimates that were grabbed off of random webpages or quickly calculated using known metrics. Below is a list of URLs of sites that I utilized during my research.

<http://www.tamu.edu/current-students/index.html>

<http://today.tamu.edu/2012/09/21/enrollment-surpasses-50000-milestone-includes-10000-grad-students/>

https://en.wikipedia.org/wiki/University_of_Texas_at_Austin

<http://calculator.s3.amazonaws.com/index.html>

<http://www.zdnet.com/article/cloud-storage-price-check/>

<https://www.backblaze.com/b2/cloud-storage-pricing.html>

<https://www.digitalrebellion.com/webapps/videocalc>

<http://www.quotecolo.com/new-york-cloud-services/>

<https://storageservers.wordpress.com/2013/07/17/facts-and-stats-of-worlds-largest-data-centers/>

<https://www.theguardian.com/technology/2007/jul/25/media.newmedia>

www.datacenterjournal.com/

Project2 Report2

Nathaniel Leake

424003778

Time to complete Task: **2hrs**

Time Tracking Information:

1st day: 10:00pm-11:50pm

Base System: Cloud9

Lately this semester I have been working a great deal in Cloud9 (c9.io). Cloud9 is a web-based platform that gives you free access to a virtual machine on which you can create projects, run Unix terminal commands, and run programs on a virtual local network. Thus, Cloud9 is more of an IaaS whereas Heroku is a PaaS. One of my team projects in another class involves building a Ruby web application in Cloud9; I have used this application for the Heroku usability task.

Note:

I had actually worked with Heroku (minimally) in the past, meaning that this process wasn't brand new to me; however, I did learn how to link a Cloud9 account with Heroku so that deployment was as simple as pushing to a master Heroku repository

Going Through Heroku Tutorial

I followed this tutorial for Heroku with Ruby:

<https://devcenter.heroku.com/articles/getting-started-with-ruby#introduction>

The first step of the process was to install the Bundler gem, which I had already done for this application (Bundler has a lot of useful packages). After confirming the existence of Bundler, I needed to install the Heroku CLI (command line interface), which as simple as just copying the install command from the tutorial into my Cloud9 terminal. By this point, I needed to log into my Heroku account and create an app under Dashboard > Personal apps. Then I had to clone this app over to my Cloud9 application folder, organize the files, run `heroku create`, and do `git push heroku master` to link the Heroku app with my repo. At this point, I was basically done with the setup.

Difficulties

After fiddling around with Heroku, I decided that I wanted change my app's name from the random one generated by Heroku when I created the app. However, after changing the App name, I couldn't get Cloud9 to commit to the Heroku repository, and had to follow their online guide to correct the local repository name before they were properly re-linked. Also, Heroku was running a newer version of Ruby than my App and I couldn't figure out how to downgrade Heroku to the right version because I kept getting build errors. In the end I found Cedar-14 (<https://blog.heroku.com/cedar-14-public-beta>), a Heroku mod that seemed to fix my build errors and run the older version of Ruby once I set it as the default for my Heroku app.

My Impression

Having set up lots of local/hosted servers and websites before, I don't think I have a huge need of Heroku, but I can definitely vouch for its easiness to learn, especially for someone new to website

hosting who doesn't want to have to deal with setting up dedicated servers with port-forwarding, expensive online hosting options (like Wordpress), or free online hosting options with severely limited control (like Google sites). The free version of Heroku is pretty fast from what I can tell, so for smaller website without a ton of database info it should work fine. From what I've read online, paid versions of Heroku scale well to balanced the load of heavy traffic and provide unlimited bandwidth, but do not scale cheaply for storage and don't have good enterprise-level discounts on anything. Still, I would definitely recommend Heroku as a cheap, easy way to get started for small or new businesses that needs full control over its website/application.

My team's app on Heroku:

<https://boat-rental-iter1.herokuapp.com/>