Speeding into the Bright Future of Legal Tech

By

Joan Yoon

Department of Computer and Mathematical Sciences

Chicago, IL

May 2017

TABLE OF CONTENTS

ABSTRACT

Logikcull is a cloud-based e-discovery software with a mission to democratize discovery. This product is an amazing tool for legal workers to sort and filter effectively to organize the important documents for their cases. After providing a thorough description of the industry and Logikcull, I look into the specific issue of speed that the application currently has, especially as a result of a recent rapid and heavy flood of users and data. The main part of the paper goes over suggestions on how to resolve or ease the slowness, and backs it up with general research accounts.

CHAPTER 1

INTRODUCTION

Technology has come a long way from its historical forms, and it is still growing at an impressive rate. With a majority of our devices and tools being technologically tied, it is clear that there is a heavy reliance on such tools. Today, it is even touching upon the legal industry and software like Logikcull allows the members working in this sector to be more resourceful in their work.

No technology can be perfect. Each product is a constant work in progress. Logikcull is no exception. Our program is very powerful in its current state, but as indicated by shortcomings experienced as a result of our customer base skyrocketing in a short amount of time, it is apparent that we can always produce improvements. A major challenge that came up is on the topic of delays in speed, which had a serious rolling effect on the overall performance of the system and inflicted the usual workflow.

There are several points that we can cover to bring Logikcull back up to speed. The suggestions I make will be further illustrated in application with research already done on similar instances. We hope to implement these kind of solutions to make improvements on speed and continue to present a most effective tool for our customers.

1.1 Industry Overview

1.1.1 Legal Industry

The legal industry bases their work on traditions and history, hard facts and evidence, for the purpose of asserting their case statements. This extends to their general functions as well, as most still review and produce their case files – consisting of hundreds of thousands of documents

– manually! This means that multiple people are engaged, and long extensions of time to go over them are required as well. Being that there is a lot of sensitive information concerned, it is even necessary to make double checks; we cannot neglect the fact that it is also a risk that people not directly involved in the cases (e.g. third-party vendors assisting in organizing files) are included for creating a production.

1.1.2 Technology industry

On the other hand, the technology industry is accustomed to being very innovative and moving at a faster pace. The work here appeals to being quicker, more efficient, more modern. We have come a long way from the past ways; the Information Age affected all aspects of life that we see and experience today: transportation, communication, education, healthcare, social interactions, etc. The related products aim to make things automated and customized in their fields to gratify the demands of the savvy consumers.

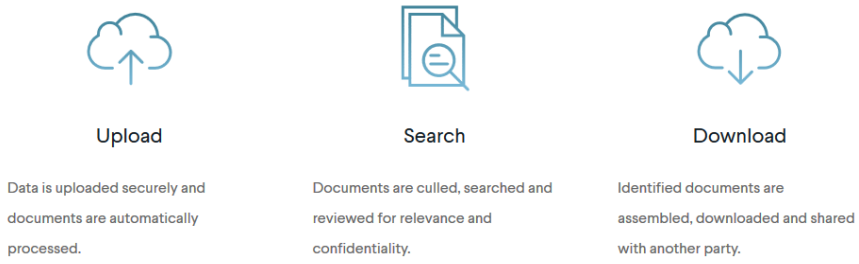
1.1.3 Legal tech industry

Opposites. That is a word that somewhat describes the relationship between the legal and technology industries. Despite the seeming discrepancy of the two fields, I somehow came to enter the "legal tech" world. Although the adoption process may meet some initial skepticism from the legal workers, this industry is growing. A startup called "Logikcull," we cater a program to improve the productivity of legal workers and provide an inventive way to perform their tasks in an easier, more resourceful manner. This program would automate much of their processing and offer a greatly streamlined workflow with a highly user friendly interface, to boot. Smart algorithms and formulas remove trivialities and broadly complete the crux of the normal labor. What would usually take weeks and months to complete, would then be accomplished with simple steps and rather only take a few days, or even mere hours and minutes!

1.2 Logikcull

What is Logikcull? Essentially, it is a discovery tool that allows for the democratization of discovery! It is an intelligence application that simplifies the legal process with the power of automation down to the simple steps of: upload, search, and download.

Image 1.2 The Basic Steps of Logikcull



The upload process is one of the major aspects that makes our program so useful for our clients. Files of various types are supported, and even database files can be inserted as well, although there are stricter guidelines that must be followed to allow for proper processing of the delimiters. This sector can be further broken down into the following stages:

- Transfer: The part of the process where the data is copying over from the original file source (preferably a local drive, as opposed to a network or shared drive, CD, etc.). It is being entered into our secure and encrypted storage space.
- Processing: This is where the magic takes place. During processing, languages are detected, the transferred data gets scanned for viruses, text is extracted or optical character recognition (OCR) is applied, and these documents are rendered to web-viewable formats. It is important to note that this list mentioned is not all-inclusive; there is a lot going on here!
- Post-processing: The program ties up ends by applying any applicable QC tags, analyzing duplicates, indexing text into the search engine, and syncing with previous uploads.

After uploading, the customer can cull, tag, and search to progress their review. The power search is available to build advanced searches to isolate certain metadata conditions and syntax. The embedded filter carousel has a wide collection of categories to assist in identifying the relevant documents for the case. They can even coordinate with their team to assign sets of documents and share notes and comments.

Finally, download encompasses all aspects of exporting, load files, and productions. With the highly customizable templates, a production can be built out of a specific search criteria, and that also meets the requirements that the opposing has requested [2].

CHAPTER 2

THE INVESTIGATION

2.1 Speed

This is obviously a very powerful tool for the legal world. However, once these tools are placed in the customers' hands, they can become greedy and continue to demand even more features. A faster, stronger, more customized program – the requests are endless. Of course we want to deliver as well, to maintain our high standards of service with our product. Our program is already considerably quick in its broad functions, but this is still an important factor for all of our customers. In this case, slow and steady doesn't always win the race. The users have to work within certain deadlines, and yet must still be meticulous on the details. As a result, this ties together the issues that we face on the current application speed and the success of its operations.

2.1.1 Variables

Logikcull is cloud-based, which may set off as nearly a whole new topic, but we will focus on this factor in relation to speed. It was realized that between regions and even between different operating devices, speed and optimization may range – Amazon's cloud performance has been reported as varying as much as 200 times! Various factors impact the issue of latency apart from simply geographical location or the physical servers and wires that the cloud reduces down to. John Koetsier of VentureBeat actually pins Amazon Web Services (AWS) system design as a more plausible reason, which suggests the idea that there are potential resolutions to similar issues that are not being satisfied [11]. It is unfortunate that Koetsier did not elaborate on this idea, but analysis does show that Amazon "values cheap, elastic computing power above all else" [7]. Dr. Liu's research presents that AWS was built without a focus on cost so attempts flexibility, as its performance "displayed high variation in the response time and availability for

queries" [6]. If the problem is traced to exist at the base of the original host, then eyebrows are certainly raised on the effectiveness of Logikcull working like a large database center as it does for our customers.

Of the several variables that contribute to the decrease in speed of Logikcull, internet connection may potentially be the most significant to note. The most common cause of slow speed on the internet is the host server. This is in turn connected to the host's own internet connection. Being hosted on the cloud such as AWS emphasizes the reliance on a good internet connection. If there is a spotty connection, there is limited access to the resources, or in this case, the applications within Logikcull. A stable internet connection, especially with the proper bandwidth, can be a major game-changer in the speed at which the computer produces the desired actions and results.

Documents are entailed at every stage of the use case in Logikcull, and I speak of it in relation to the number of documents in a file and the types of documents included. Two sets of 1 GB files can have very different types and amounts of matter (i.e. any and all materials related to a specific case), which could significantly alter the time it takes to process them. As a result, it is difficult to determine an average running time. As a rule-of-thumb, however, is about 2 GB per hour per project, or about 5,000 documents per hour, depending on how compact the container is. During processing or production, to ensure optimal transfer conditions, we make sure to recommend a wired connection, a local drive being the transfer source (as opposed to a network or shared drive), and sleep mode set to off to prevent interruption.

Another variable would be the number of users. Please note that this program is cloud-based on AWS to scale, but the technical reality is that there may still be a bottleneck with the limits of virtual workers available for the span of each task and file. This connects to the internet

connection, because there is that distribution of service. Currently, business is exploding for our company – we saw the biggest spike in matter over just this past month than we saw over the total beginning months of this year! This means that we need a platform that can support all of this data. We currently have a set amount of service allotted by AWS, but we must find an adequate method to match the exponential growth of users, and its subsequent number of matter uploaded.

Finally, the last variable to be mentioned will be a long stretch and should be taken with a grain of salt. This partly corresponds to the host we can even place some blame on the structure of Logikcull as a cause of weakness leading to slowness in speed. More specifically, I want to pinpoint the maintenance of Logikcull to be somewhat at fault. In this way I am tying the learning from the course about testing and formal verifications to this proposed variable. Currently, I understand that the team conducts a lot of bug checks and testing (especially if a new feature is rolled out); this is a more reactive, short-term outlook that does not consider the extent problem codings that may be overlooked and more intricate issues that may arise. The best formal verification methods may not be actively in play to keep up to the design of Logikcull and its changes.

2.1.2 Challenges to the Investigation

Without data, progress cannot be made. Koetsier contends that if Amazon would release its data statistics, there may be actual solutions to the slowness in system speed. Server debugging company Takipi's co-founder Shoor reproaches the company on neglecting the potential to conduct proper research and apply its proposals to significantly improve speed [11]. However, for now, we can at least assume that the two variables are related and also can be controlled to some extent to decrease latency.

It is curious as to what Koetsier was exactly referring to when he stated his belief that there is a problem within the Amazon system design as a whole. This factor may be significant in proposing resolutions to slowness, to an extent. Even if there really are problems within AWS itself that can limit our product, we will have to take it for granted for the time being for the purpose of our investigation. If we view this in terms of the cloud at the most general sense, we know that it has extended scale that can support our variables. However, in our case there is still a limit based on the type of plan contracted with AWS.

Another limit connected to the cloud is the variable of internet connection. We support various operating systems and modern browsers, but we do typically recommend the latest version of Chrome for the best results, as this is what we usually program the infrastructure of Logikcull on. We do also advise on meeting certain conditions for optimal performance, but we unfortunately cannot force someone to provide that perfect state.

2.2 Solutions for the Current Day

2.2.1 A Look into the Current Day

Logikcull is a startup, and it is definitely taking off. Being in a place of growth, many of the variables cannot be directly addressed in a way that will allow immediate increase in system speed. By this I mean that some variables will actually have to be increased in a "negative" manner for desired results. This in turn may initially lead to slower response times, but will have to be tackled in terms of other variables.

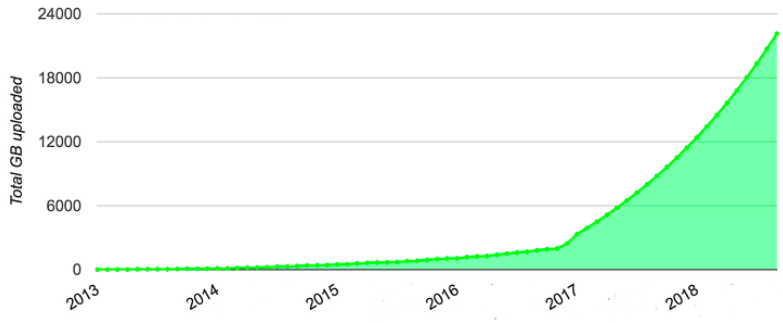
We are currently experiencing a high volume of demand. We went from only one customer per week to about 15 customers per week! This month clearly had an astounding jump, compared to the total first couple months of the year. The following image provides a single

day's instance of data, and just a sector that shows an overall picture of the total jobs affecting latency of the system. Let's say that the system was overrun with a total of 345,000 jobs to be processed between window and linux. As shown by the red color coding (good status would be colored green), this was already a strain. The latency graph at this same time shows that there were big spikes at multiple points. The delays can be a contributing factor in extended holdup of the queue completion rate.

Figure 2.2.1 (a) Job Queue and Latency Graph

I want to reiterate that this is just one simple example of what we are experiencing now. Although from the beginning I don't believe the queue was at green, and this is actually difficult to achieve with the demand, but it is significant when I notice the jump from around 100,000 jobs queued to this number shown. The highest I have seen it was at around 500,000 jobs queued, but it held for only a couple of hours. It is important to note that the report is actually displayed *queued* jobs, as the system has a bottleneck and can only process so many tasks at a time. This further indicates that there is a serious lag in the system because the unexpected jump in jobs. The default attempts to evenly distribute the virtual workers to tasks, but the increase in jobs causes a lack of stability while also possibly stocking out the virtual workers. This makes for disorganization and the consequent slow running speeds within general Logikcull functions.

Figure 2.2.1 (b) Graph of Exponential Growth



As you can see in the figure above, we are expecting exponential growth and are preparing for the data tsunami. The area from the results of the graph covers the explosion of data. The company, and especially the engineers, had to deal reactively by putting out recent fires that occurred as a result of the sudden large influx. API channels and basic monitoring tools alerted them of issues so that they could respond to them as they came in. The capacity on the store server was increased, and it was observed that latency dropped quickly. Manual pushes were even made to move things along. They also are working proactively by building the additional foundation to help meet the quickly approaching challenges. Although we had to rearrange our priorities to meet the demand, we have settled to focus on the customers and their data, as more appropriate to the situation.

2.2.2 Proposed Solutions

The simpler it is to use a technology, typically the more complex it is in the background. Logikcull is still a start-up product but fairly established. Despite this honor, it is important to note that we placed customer feature requests temporarily on the backburner so that we can put all hands on deck towards optimizing performance above all. The most important success and challenge is to make sure Logikcull functions as we say it will, and assure the customers that they will be able to commit a smooth workflow. Playing “traffic cop” with the assignment of

VMs is inefficient and should rather be automatic as according to the coding built in; we want to avoid having to make manual pushes as much as possible.

The monitoring process now is somewhat disconnected and is spread across different regions of Logikcull. We use Datadog, Trello, Logikbot/Lita, Pager Duty, and possibly more. They are definitely useful, but the analytics sometimes do not match up. Slower response times are logged as a result of these limits. It would be ideal to find a communication channel that casts a wider net. We would then want to use this towards locating fires earlier and putting them out with solutions we know would undoubtedly work, achieved through stable data sets collected from a global view.

Troubleshooting can be performed more efficiently. This would tie to more accurate forecasting, but one-off cases must be taken into account just in case as well. We have algorithms in place to make sure Logikcull works competently during normal states. However, because we were caught off guard at the upsurge, we resorted to trial-and-error type scenarios. For example, with the jump we determined that unicorn scaling would help ease the latency levels. But we cannot cast it easily for all similar occurrences, as the release process requires starting a new master and set of workers. You would need to have at least the amount of free memory as the total RSS of unicorn and its sub-processes, otherwise a release could be another problem. We assumed that releasing a certain amount of unicorns would help us in the short-term, but it is imperative that we do not rely so heavily on our current programming to work. We cannot just be reactive; we must arrange backup plans and guards to attend to the issues directly and properly. Jumping into action, I am assuming that a lot of work will be done on the backend with focuses on code increasing scope of functions. This should reflect on the front-end in forms of better streamlined tasks and response times.

According to the architecture, the application flexes to the infrastructure of the cloud. Yes, we are aware of the issue of geography to latency. However, this is a small setback, and the cloud is still considered the best platform to establish Logikcull. Research already shows that many users already work around the weaknesses of the cloud, as the benefits still outweigh the weaknesses at this point [6]. The database system like ours was made with the cloud in mind, and serves the best purposes for our customers. We must then largely abide to the "rules" that the cloud has established.

Being more specific, a challenge is that there is a bottleneck of our current uses within AWS. The solution here would then be to find that balance of bandwidth and scale to make sure that response times are quick and steady for our customers. The cloud is known to be very accessible and most importantly here, scalable. The AWS cloud allows for adjustment of CPU, memory size, etc., as needed [10]. This basically implies that if we scale up with our cloud provider, we can increase our resources and easily improve the speed. We are additionally expecting that AWS will someday respond to competition by delivery better response times, and performance.

We can always improve our production speed with our virtual machines (VM). Still referring to the same queue of 345,000 jobs as shown in Figure 2.2.1, Figure 2.2.2 (a) presents a graph of online to active workers. The maximum number of VMs is indicated with the red dotted line; the blue area shows the active workers. Although the queue is so large, it is strange to see the occurrence of a drop in the workers. It is worrying that it may very well be that they failed as a result of being unable to handle the flood. It also seems that it seems the cap is almost reached and signifies another risk factor. The corresponding image (b) shows an example of what the

VMs are tasked to (the blackouts are a result of preserving levels of privacy and confidentiality of company data).

Figure 2.2.2 (a) Worker Count

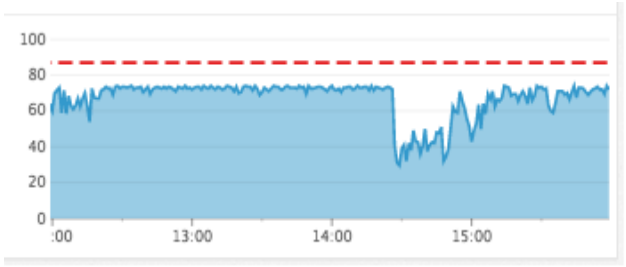
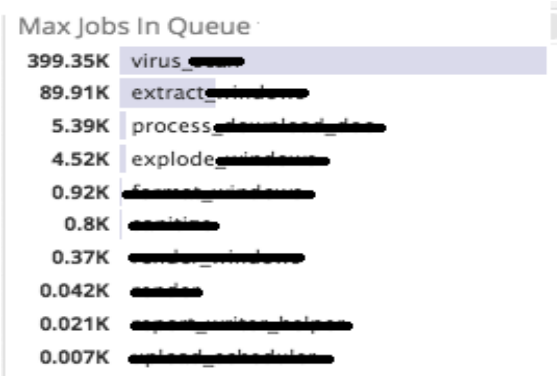


Figure 2.2.2 (b) Types of Jobs in Queue



Obviously, the numbers are quite uneven. There must be some kind of ruling to determine that some jobs can be prioritized over others; overruling should especially be permitted in times of high traffic. Increasing the number and the range of the VMs would be very helpful to help move jobs along. Furthermore, assigning their jobs distinctly would also be very helpful. By this I mean to uphold an organization where the majority of VMs work on a specific job and move on to the next, rather than being distributed across all jobs. This would depend on the current activity and stages of the users within their workflow, but an example of this instance is shown in figure (b).

The ultimate proposal I would like to make to improve speed of Logikcull involves the knowledge that I have gained with the course materials. Formal verification methods as an addition to traditional checking methods will allow the program that it is applied to, to be sound in its complete workings. These tools look at the details of codes written and executes through to warrant a smooth, full-circle course. Bug checks and testing are standard to confirming that our speed level is up to par. ATP and model checking will make a point to improving current statuses and arrange preventive measures as well. We do have some in place now, but taking a more

detailed look may be one of the most important to hold as a core unit to secure the platform and still allow for expansion.

2.2.3 Review of Research Supporting Proposed Solutions

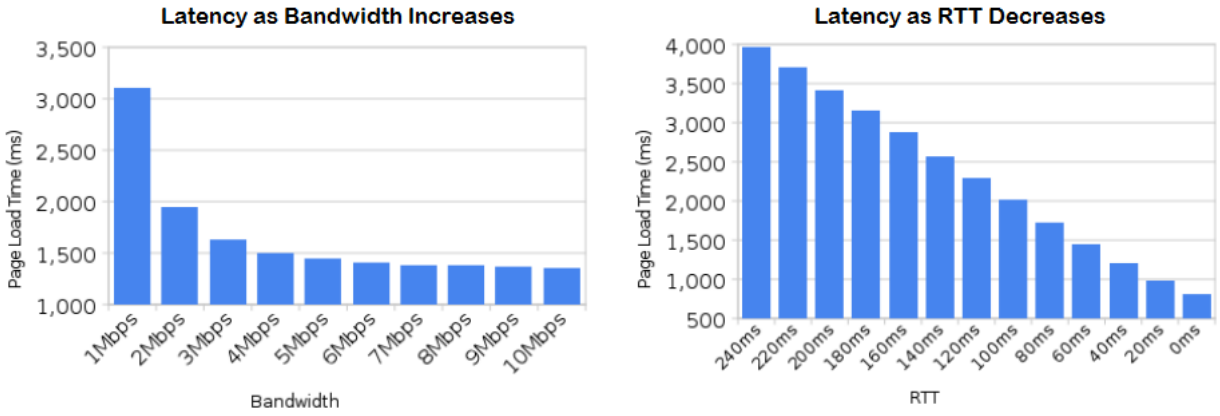
Variations in performance must be factored into software application design to be effective. As mentioned previously, we cannot be troubleshooting solely on a reactive basis. We want to make our software flexible to be able to handle the worst case scenario with a best case output. For example, we may consider the Big O notation, as it can be used to describe the space used by an algorithm or the time required for it to execute. This would be helpful because from there we can determine what variables were affecting the writing in a negative manner. Essentially, the foundations must be dug into to fill any cracks in the seams that can weaken the basic features of a cloud-based software, i.e. flexibility, accessibility, security, and speed.

Verification is very important in making sure that everything runs as it should, including keeping up speed. We must combine various methods to achieve this as there is unfortunately not yet a single way to prove all layers of algorithms. We need to test to make sure that the bugs are hacked out and doesn't have any breaks in behavior. Clarke et al found that bounded model checking (BMC) can be used for bug checking, in addition to generating a satisfiable propositional formula. Model checking is typically better applied to finite models, but in the case of an infinite-state system, BMC can be abstracted to attest our system and also avoid a state explosion. It was actually discovered that BMC was the most successful in preventing a state explosion with the ability to "find counterexamples in circuits with thousands of latches and inputs" [8]. Furthermore, if we implement problem-oriented automated theorem provers (ATP) such as that Bibel and his associates advocate, we would be able to cover all of our bases. A

system that uses both classical and non-classical logics offers an all-encompassing solution within the mathematics, programming, and planning [5].

It is critical for networks in cloud computing to provide high performance that is also consistent to maintain customer happiness. Mike Belshe and Phillip Kent outline in their work that one of the main problems when pinpointing the cause of slow services is API latency. Bandwidth plays a part, as a reason for slowness would be that the capacity is reached or exceeded. However, it cannot be narrowed down to just megabytes and file sizes [4]. The throughput (i.e. bandwidth) of a network can be expanded to allow larger sets of data to be passed through, but the round trip time (RTT) is not so elastic. This ties to the claims of Koetsier that speed would depend on the geographic location. "For a service delivered to users through a web server the target response time should be below 200 milliseconds … any more than this will introduce a noticeable, uncomfortable delay for the user" [10].

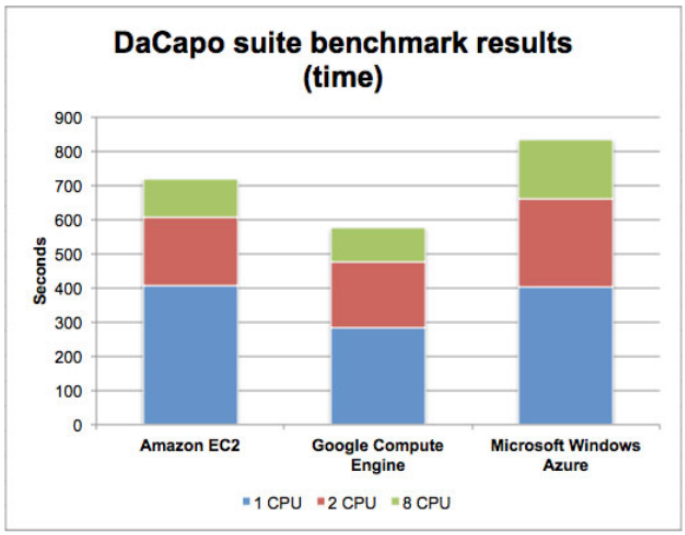
Figure 2.2.3 (a) Bandwidth and RTT on Page Load Time



The following figures provide data on each effect: as bandwidth is increased, latency also decreases, but this begins to taper off at a certain value; however, there is a stronger striking impact on latency (decreasing) as RTT increases. It would be ideal to have all of our customers within reasonable distances to consider this, but is not a reality. It is difficult to adjust a network,

but we can at least minimize latency by other means, such as the foundational programming and a larger scale.

Figure 2.2.3 (b) Speed According to CPU Size



Peter Wayner of InfoWorld compared speed between three leading cloud providers, and one of his tests displayed that increasing CPU may help to certain extents [13]. We need to find a good balance between these factors and forecast accurately to secure satisfactory speeds.

Whether or not we scale up, we have to be smart about how we assign our supply of workers. Wu et al observed that oftentimes VM allocations focus on maximizing the efficiency of resources, while neglecting the service guarantee for users. They emphasize that both parts need to be met by introducing OGWAR. OGWAR is defined as "a novel model-free virtual machine allocation … characterized by an online greedy algorithm … [which is coupled] with non-increasing reserving algorithms to deal with flexible jobs and inflexible jobs " [3]. Leveraging a priority function for allocation, the OGWAR mechanism is fast and has a tight competitive bound with strong performance for when maximum demand of customers is close to the capacity available. Assignment algorithms managing VMs allow for efficiency, minimizing the number of workers for each task and yet with adequate post to complete.

As displayed by the research into the generality of these variables that affect speed, we observe what forms would be best applied to Logikcull. We want to be efficient with our resources and forward-thinking when establishing our infrastructure. To sustain customer satisfaction for our technology, performance, and specifically speed, can be stabilized by with the proper programming, foundationally and by checks. Despite our software being created conformed to the cloud, we must take advantage of its own flexibility to serve our purpose.

CHAPTER 3

CONCLUSION

There is a bright future in legal tech. Andrew Ross, CEO of ROSS Intelligence, firmly believes in this outlook. The emergence of this field challenged the risk-averse psychology of the people in this field, and made them face the reality. Artificial intelligence simplifies and empowers "human lawyers to do more than ever before possible" [12]. Following along with this viewpoint, we are centering ourselves amidst the hubbub. If not a complete machine that can procure the proper arguments for our desired functions and results, then at least one that is able to trump as a sort of checks.

Logikcull was built to be scalable. Our mission to "Democratize Discovery" cannot be accomplished if we cannot serve all the peoples. This leads to the loaded engineering question of if it really can scale. There are some single point of failures (SPOFs) that need to be distributed for better fault tolerance. Data processing scaling issues need to be optimized so we can increase the processing speed per customer, by at least a multiple of ten. We are aware of the problems to solve. We are looking at this with a positive outlook, as the high number of problems can be attributed to the high demand, which is a great case for a business. The engineering team is making fundamental improvements on the infrastructure continuously, towards increasing speed and global optimal performance. The findings in technology that supports our software are being integrated into our systems so that we can continue to pave our future and advocate our values towards our clients.

BIBLIOGRAPHY

[1] (2013, November 15). What are AWS Cloud Design Patterns? Retrieved from http://en.clouddesignpattern.org/index.php/Main_Page

[2] Automated eDiscovery Features. Retrieved from http://logikcull.com/features/

[3] Arjona Aroca, J., Fernanez Anta, A. (2015, April 14). *Empirical comparison of power-efficient virtual machine assignment algorithms*. Madrid, Spain: Universidad Carlos III de Madrid.

[4] Belshe, M. (2010, April 8). More Bandwidth Doesn't Matter (much). Retrieved from https://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2

[5] Bibel, W., Korn, D., Kreitz, C., Schmitt, S. *Problem-Oriented Applications of Automated Theorem Proving*. Darmstadt, Germany: Technical University in Darmstadt.

[6] Brooks, C. (2009, September 30). Performance woes the next big hurdle for cloud. Retrieved from http://searchcloudcomputing.techtarget.com/news/1369815/Performance-woes-the-next-big-hurdle-for-cloud

[7] Cimino, S. (2010, November). Cloud computing performance FAQ: What makes the cloud go? Retrieved from http://searchcloudcomputing.techtarget.com/feature/Cloud-computing-performance-FAQ-What-makes-the-cloud-go

[8] Clarke, E. M., Klieber, W., Novacek, M., Zuliani, P. (2012). Model Checking and the State Explosion Problem. *Tools for Practical Software Verification*. Retrieved from https://pdfs.semanticscholar.org/a425/7cd9abde220d55d777da6c061671b63f8bf2.pdf

[9] Grigorik, I. (2012, July 19). Latency: The New Web Performance Bottleneck. Retrieved from https://www.igvita.com/2012/07/19/latency-the-new-web-performance-bottleneck/

[10] Kent, Phillip. (2017, January 26). How to improve cloud latency and throughput: How good is your cloud provider's network? Retrieved from https://cloudstore.interoute.com/knowledge-centre/blog/improve-cloud-latency-throughput-how-good-providers-network

[11] Koetsier, J. (2013, June 3). Amazon Web Services speeds can vary by up to 200X depending on region. Retrieved from https://venturebeat.com/2013/06/03/amazon-web-services-speeds-can-vary-by-up-to-200x-depending-on-region/

[12] Sullivan, Casey. (2017, May 1). A New Era: ROSS's Andrew Arruda on A.I., Law, and the Bright Future of Legaltech. Retrieved from http://logikcull.com/blog/new-era-rosss-andrew-arruda-law-bright-future-legaltech/

[13] Wayner, Peter. (2014, February 26). Ultimate cloud speed tests: Amazon vs. Google vs. Windows Azure. Retrieved from http://www.infoworld.com/article/2610403/cloud-computing/ultimate-cloud-speed-tests--amazon-vs--google-vs--windows-azure.html?page=3

[14] Wu, X., Gu, Y., Li, G., Tao, J., Chen, J., Ma, X. (2014, September). Online Mechanism Design for VMs Allocation in Private Cloud. In *11th IFIP International Conference on Network and Parallel Computing (NPC).* Retrieved from https://hal.inria.fr/hal-01403088/document