Recent Trends in High Tail Latency

Aaron McCanty
The Ohio State University
Mccanty.1@osu.edu

1. INTRODUCTION

The amount of data stored by enterprises in the modern world is staggering. If every business were to handle all of their own data requests, their cost of operation would skyrocket. Because of this, many large enterprises across the world today utilize cloud infrastructures to process their requests and data transfers. These cloud infrastructures involve sending requests to large warehouses full of servers for processing. To speed up the data processing rates, these warehouses of servers utilize large-scale parallel computing. R

Parallel computation involves breaking requests up into many parts than can be executed simultaneously. These parts are then sent out to many different servers which perform the calculation, data retrieval, or other service and return the necessary values. The result of this type of speed up process is a drastic reduction in wait times. However, there is a down side. The nature of these parallel computations means that the request can only be completed once all of its respective parts have finished execution. So if one server has a drastically higher response time, the entire request is slowed down as well.

Servers typically have extremely fast average response times. For example, servers that have implemented an optimized Memcached system have an average response time of 100 µs [2]. A response time this low is excellent in terms of completing requests without a noticeable slowdown. Problems arise at the tail end of the request speeds. The 99.9th percentile latency in the same Memcached system discussed before is 5ms. This is an enormous slowdown. Obviously being the 99.9th percentile, servers only experience this slow a speed in 1 out of every 1000 requests. This doesn't seem to be very frequent, but when you consider that a single Facebook web request may access thousands of these Memcached servers [5] and a Bing search may access 10,000 index servers [6], these large latencies become very common and relevant to request response time.

1.

These large tail latencies can stem from several sources; including hardware, operating system, application level design, and configuration choices [2]. Large tail latency can have a huge impact on a customer's data retrieval time, which in turn can lead to a decrease in revenue for an enterprise. As a rule of thumb, delays exceeding 100 ms decrease total revenue by 1% [7]. As delays have the potential to negatively impact a business' bottom line, there has been a tremendous shift in the world of cloud infrastructure to reduce these large tail latencies.

2. RELATED WORK

There are several papers that attempt to take on the issue of extremely large tail latencies in cloud computing. Principle among them are: Tales of the Tail: Hardware, OS, and Application-level Sources of Tail Latency [2], C3: cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection [1], Adaptive Performance-Aware Distributed Memory Caching [3], and Zoolander: Efficiently Meeting Very Strict, Low-Latency SLOs. Each of these papers will now be examined in turn, and heir findings will be reported,.

2.1 Tails of the Tail [2]

The easiest place to start this examination is with Tails of the Tail: Hardware, OS, and Application-level Sources of Tail Latency. As the title would suggest, this paper examines some of the causes of the extremely high tail latencies experienced in the industry today. Once the causes of high tail latency are identified, steps can be taken to mitigate the impact they have on system performance.

Background Processes was the first cause of high tail latency times examined in this study. To simplify the problem, the researchers limited their experiments to a single server, using a single cpu with a single core. With this setup, there is absolutely no way two programs can run in parallel with each other. Despite only running the test application on each server, the scheduler still had to deal with default background processes that all servers encounter. When the scheduler allocates core time to one of these other processes, it can greatly increase system latency for the application. The versatility of Linux allowed the researchers to change the priority of their test application so that it was almost always given core resources. This reduced the additional slow down, but on a system less versatile this approach not be able to be used.

How the scheduling algorithm decides to allocate core time can also affect tail latency. This paper showed evidence that a First-in-First-Out(FIFO) scheduling

algorithm reduces tail latency, but increases median latency. The default scheduling algorithm on most linux servers is a non-FIFO algorithm which negatively affects tail latency.

Multicore processors were also tested and shown to decrease tail latency for some systems. Moving from single to multicore processors only reduced tail latency if the systems have a single-queue model (i.e. any processor can process any request). Some systems can be converted to this model easily, while others cannot.

The research also showed that dedicating certain cores in a multicore system to processing interrupts could also reduce tail latency.

Scaling out systems beyond a single CPU was also show to create problems with poor placement of threads and memory on NUMA systems. This led to a higher tail latency.

Lastly, the research showed that at low utilization levels, the system encounters a situation where there is a tradeoff between saving power and tail latency levels.

2.2 Adaptive Performance-Aware [3]

Adaptive Performance-Aware Distributed Memory Caching examines how one technique of dealing with large tail latencies can be improved upon to see noticeable results. Memcached systems are used to greatly increase performance on many web applications. Despite the gains seen by these systems, inefficient partitioning schemes can often lead to load imbalances, where one server or core is tasked with more work than others. This leaves the door open for further optimizations.

This paper presents an improvement for cache management. This method adapts to changing situations well, and ensures that the work load is evenly distributed among resources, helping reduce tail latency.

2.3 C3 [1]

C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection also explores some of the causes of large tail latencies, further reiterating the need for better optimized systems for handling large numbers of requests. It presents many of the current methods for sealing with these tail latencies but points out that these are often minor improvements on methods that have received a lot of attention in other studies.

C3 is an adaptive replica selection mechanism used to determine which replica server is chosen to service a request by a client. It is the authors view that using methods commonly used to reduce tail latency on top of a poor replica selection algorithm leaves a lot of room for more errors and problems to arise. C3 aims to create a solid replica selection foundation with which other techniques can be built on to greatly improve tail latencies.

2.4 Zoolander [4]

Zoolander: Efficiently Meeting Very Strict, Low Latency SLOs explores some of the ways computing services can attain service level objectives (SLOs) set by their clients. As responsiveness has been shown to impact revenue, it is very important that enterprises try to attain very fast response times for their customers, which is why they set strict low latency SLOs on their computing service providers.

This paper presents a technology called Zoolander, which was developed specifically to help computation service providers attain these strict SLOs. Zoolander tackles the idea of replication for predictability by utilizing a previously dismissed method: copying requests to multiple nodes. This may seem counter-intuitive, but these multiple requests reduce the risk of hitting a high tail latency and, thanks to advances in technology, it doesn't negatively impact throughput. Zoolander is also flexible enough to adjust its methodology when the situation calls for it to remain efficient. If more requests need to be distributed to more nodes, Zoolander adjusts accordingly, but during periods of low requests, it can also use under used nodes to reduce costly SLO violations, thereby reducing operating costs overall.

3. DISSIMMILARITES

As discussed previously, each and every one of these papers approached the issue of high tail latency in a different way, finding varying measures of success.

Tails of the Tail examined three different servers – a null RPC service, Memcached, and Nginx- trying to find the best optimizations of the Hardware, OS, an application-level issues that cause high tail latency. The paper examines what exactly causes high tail latency, and how each of these problems can be countered. By making changes to these causes on each machine, they were able to achieve a 99.9^{th} percentile latency of 32 µs and a median latency of 11 µs. These speedups were achieved without adding any real kind of middleware.

Adaptive Performance on the other hand explores an addition to a system currently in use in many distributed computer systems. The paper focuses on improving one particular issue causing high tail latency; namely load balancing. By implementing a new method of sending requests to servers, a higher hit rate and lower response time were achieved. Tests were carried out on Amazon EC2 servers.

C3 and Zoolander took yet another approach. Both are essentially middleware, which were developed to be installed on machines to help improve performance using their own algorithms. Zoolander focuses on using techniques to reduce high latency times to help cloud computing services meet the strict SLOs their clients require. Zoolander, similar to Adaptive Performance, tries to reduce tail latency by addressing which servers are sent specific request. It helps balance loads (like [3]) and also creates some redundancy in the requests sent, reducing the likelihood that long tail times are experienced. C3 also discusses a similar middleware.

4. SIMMILARITES

All of these papers tackle the issue of large tail latency by exploring the many ways of reducing tail latency to improve performance. Despite the different approaches taken in these papers, some similarities do appear.

First and foremost, all of these papers have identified a real problem in the computing world today that has not received much attention in the past. This is partly because the issues seen with large tail latencies have only become common in recent years with the advent of large scale cloud computing facilities and the heavy use of parallel computation. Regardless of how popular a subject it is, all of these papers identify why it is a major cause for concern in the industry today.

A recurring solution to high tail latencies was to examine how requests are sent to different servers. Often, many requests or very heavy requests are all sent to the same server, overloading its resources while other servers sit idle. Reducing these load imbalances is an easy way to reduce potentially damaging high latency times.

In addition, the tests carried out in these papers were performed on real machines that are used in the industry (Amazon's EC2 servers in particular). Some theoretical analysis was conducted as well, but the results were very concrete. The techniques or technologies studied in these papers can very easily be applied by companies in the very near future to tackle these problems.

Finally, despite the variety of techniques examined throughout these papers, all of them managed to see significant improvements in the tail latency. All four approaches succeeded in finding a solution to a very real problem, and the success experienced within the techniques is very encouraging for future research. If these early attempts at speedup can increase performance by orders of magnitude, then the future truly is bright when it comes to improving performance.

5. REFERENCES

[1] Canini, M., Feldmann, A., Schmid, S., and Suresh, L. 2015 C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection. NSDI, 2015.

[2] J. Li, N. Sharma, D. Ports and S. Gribble. Tales of the Tail: Hardware, OS, and Application-level Sources of Tail Latency, Symposium on Cloud Computing (SOCC) 2014

[3] J. Hwang and T. Wood. Adaptive Performance-Aware Distributed Memory Caching. International Conference on Autonomic Computing (ICAC), 2013

[4] C. Stewart, A. Chakrabarti and R. Griffith. Zoolander: Efficiently Meeting Very Strict, Low-Latency SLOs. International Conference on Autonomic Computing (ICAC), 2013.

[5] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani. Scaling memcache at Facebook. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13), Lombard, IL, USA, Apr. 2013.

[6]] V. Jalaparti, P. Bodik, S. Kandula, I. Menache, M. Rybalkin, and C. Yan. Speeding up distributed request-response work- flows. In Proceedings of ACM SIGCOMM 2013, Hong Kong, China, Aug. 2013

[7] rigor.com. Why performance matters to your bottom line. http://rigor.com/2012/09/ roi-of-web-performance-infographic.

[8] cpu-world.com. Microprocessor/Co-processor/Microcontroller families. http://www.cpu-world.com/CPUs/.