Ananta: Cloud Scale Load Balancing

Nitish Paradkar, Zaina Hamid

EECS 589 Paper Review

1 Full Reference

Patel, P. *et al.*, "Ananta: Cloud Scale Load Balancing," *Proc. of ACM SIGCOMM '13*, 43(4):207-218, Oct. 2013.

2 Paper Summary

This paper presents Ananta, a horizontally scaled layer-4 load balancer. It presents a shift in traditional hardware load balancers by combining ideas from software defined networking and distributed systems. Ananta is made up of three major components, the Ananta Manager (which makes up the control plane), and the Ananta Mux and Host Agent (which make up the data plane). Through the use of the Host Agent and the Ananta Manager, the system is able to perform direct server return, Fastpath and NAT, enabling the VM to bypass the Mux, resulting in less load on the load balancer. The authors begin the paper with a summary of the nature of traffic in some of eight data centers. After observing these data centers for a week, they discovered that greater than 80% of VIP traffic (data that needs load balancing or SNAT) is either outbound or contained within the same data center. With these requirements in mind, they design Ananta to offload all of this work to the host, thus only handling about 20% of the total VIP traffic. Through the use of Ananta, the authors are able to add many features that traditional load balances can't apply, such as horizontal scaling, reliability, tenant isolation, and some DoS protection. They describe the architecture in detail of the three components, and how each of them contributes to the bigger picture. Consequently, they describe the practical implementation, and modifications made on it on top of existing knowledge. The authors also test Ananta in the public cloud and find that their system has good SNAT response latency (99% of the SNAT port allocations occur within 2 seconds), high availability (99.95% on average), and discover that they have a median VIP configuration time of 75ms.

3 Highlights

The first thing we appreciate about the author's method is creating a load balancer in software. With their system, they were not only able to bypass the load balancer for VIP traffic 80% of the time, they were also able to horizontally scale their system such that multiple network elements simultaneously process packets for the same VIP without per flow state synchronization, which enables it to grow more while still being relatively cheap. This represents a break from intuition, as intuitively, one would be very likely to create a load balancer in hardware in order to make the load balancer as fast as possible. Another key difference from existing load balancers is the offload of a significant chunk of data and control plane functionality to hypervisors, that handle state only for the VMs hosted on it. While distribution and division of labour has its own benefits it also adds complications of state maintenance and availability.

By creating their load balancer in software, they are also able to implement techniques like Fastpath and direct server return, which help the Mux avoid up to 80% of all VIP traffic. These techniques thus reduce the bottleneck on the load balancer a lot more than a hardware-based load balancer would. In addition to reducing the amount of traffic sent through the load balancer, we also appreciate some of the benefits of the algorithms implementing packet rate fairness and SNAT port allocation fairness. With a traditional hardware load balancer, both of these benefits are very hard to achieve; however, the inclusion of the components of Ananta Manager and Host Agent make this easier. While traditionally thinking about a data center, we were always used to thinking of a system where one entity controls all of the components in the data center (for example, like Facebook, who uses their entire data center to run all of their different applications). However, in the case of the authors, it is often a different user that is controlling different sets of VM's. This makes it especially important to give each tenant equal access to both SNAT ports and bandwidth regardless of other tenants. While we didn't think of these kinds of issues of fairness in data centers before, we appreciate that the authors explained the multi-tenant nature of their data centers, and helped underline the importance of these kinds of

issues. In addition to catering for scalability, reliability, and availability, ensuring Tenant isolation in times of attacks / cross usage and load balancing across services is a fairly interesting and unique requirement to cater to. We believe that the inclusion of these benefits makes a very strong case for a system like Ananta where a data center is going to be letting different tenants control various VM's.

With respect to the architecture, classification of idle timeouts by a Mux on the basis of trusted and untrusted flows to counter state exhaustion attacks to a large extent is an interesting approach. Additionally, another aspect that compliments the offloading practice is checking for DIP health monitoring at the Host Agent level therefore avoiding unwanted traffic at the Mux level.

We also appreciate how the authors took the time to do some preliminary research into the kinds of traffic that their data centers experience. The authors show that their load balancers handle about 44% of VIP traffic, and that around 80% of this VIP traffic can be avoided through the use of Ananta. The collection of this data is no doubt made easier by the fact that they have easy access to these Microsoft data centers; however, it still helps give the reader a more clear picture of a quantitative benefit that something like Ananta can provide (in addition to some of the fairness benefits discussed above). On a similar note, we also appreciate how Ananta was deployed and data was collected to show the responsiveness and availability of the Ananta Manager. It comes across as instead of reinventing the wheel at different places, Ananta very efficiently works on top of and around existing implementations. While reading the paper, we understand that scalability of the number of muxes with respect to the number of AM replicas is independent, we often felt as though the Ananta Manager could have become a new bottleneck in this system (as it had to keep track of all the Muxes and monitor all of the health of all the Host Agents), and it does help alleviate this concern by showing the reader that it had a very high availability for most of the month of January when the data was collected.

Finally, we also appreciate how the authors compare Ananta to both DNS based load balancing and OpenFlow based load balancing. Both of these are relatively common ways of performing load balancing today, and it is nice to see the authors take some time to address some of the pitfalls of these methods, and showing how Ananta does not repeat these same flaws. This also helps make for a strong case of adoption for a system like Ananta. Overall, as discussed further apart from the extensions that could have been examined with respect to metric calculations, we feel the organization of testing the framework was effective in covering base requirements, and highlighting pros and cons based on experimentation. Understanding the cases that work as expected in the case of SYN-flood and SNAT performance isolation, highlighting what's in the works for future improvements, and depicting results distinctively is essential. In the case of real world data on a public cloud with fairly high bandwidth, the authors make an attempt to justify Ananta in regard to the basic requirements set: on response latency, availability and scale, bringing to light overall base adeptness and shortcomings. Also clearly explaining the algorithms with run-throughs at the grass root level is beneficial towards fitting in the bigger picture, while ensuring to highlight disadvantages during the course of the paper.

4 Possible Improvements and Extensions

While this paper does a good job in general of explaining Ananta and going over how each of the three components interact with one another, we feel as though there are a few things in this paper that could be improved upon or extended.

One of the things that we believe could be improved is the deployability of Ananta. While the system itself provides huge benefits by decreasing the amount of VIP traffic a load balancer has to process, there are a lot of components that need to be deployed. The authors themselves address this issue when they say that upgrading Ananta is a complex process, as it involves three phases which include installing the Ananta Manager, installing the Muxes and finally installing the Host Agents. The Host Agent is present in every single physical machine that is served by Ananta. As one can imagine, for large data centers, this process can get very time consuming and

complex, and can present a huge barrier to more data centers adopting a system like Ananta. In addition such a complex process can lead to mistakes and downtimes for data centers. Based on some quick research [1], around 22% of data center outages were caused by human error. If such a complex task were to take place, it invites more room for error, and can lead to a lot of companies choosing to avoid adopting Ananta.

Another thing that we feel could have been addressed by the authors includes testing how Ananta performs on various different data sets. At the beginning of the paper, the authors observe data in some Azure data centers, and based on this, they set up the requirements for Ananta. However, research has also been done in traffic patterns in different data centers, and there tends to be a high variability in these patterns. In [2], the authors talk about how traffic for Facebook's data center is often not rack local. In that same paper, the authors also reference other works done at Microsoft that directly contradicts this statement by mentioning how the majority of traffic tends to be rack local. In the rack local case, a lot of Ananta's benefits would disappear as a lot of traffic would not be intra-DC or go to the internet. If this is the case, it would end up that Ananta would often not be very useful. Based on intuition, traffic patterns change based on the type of application that is used. This can be seen through [2] where it shows that Hadoop cluster traffic tends to be rack local, whereas the servers performing tasks for the Facebook news feed tend to have a lot of intra-DC communication. In this paper, however, the authors don't mention what kind of applications were running on the data centers they collected their data from. We feel like the inclusion of what kind of applications the data centers were running would have been useful information to the reader. In addition to this, we feel like the authors could have also made a stronger case for Ananta by measuring how much VIP traffic goes through the load balancer for different kinds of applications. This kind of information would give the reader more insight into how useful Ananta could be across different kinds of applications, and would also help provide motivation to adopt a system like Ananta in their data centers.

In addition to this, we feel like the Fastpath technique discussed by the authors could have been more efficient. With the method described by the authors, there are 7 steps of handshaking

necessary before two host agents can even begin communicating with each other. The Fastpath technique definitely saves time on the whole as this traffic does not have to travel through the Mux every single time it needs to communicate between two hosts. However, it seems that some of the handshaking in this technique could be optimized. For example, the host agent could communicate with the Mux and the Mux could forward the data to the recipient host agent with some header information about the host agent trying to contact it. Using this, the recipient host could simply start up a direct connection with the sender host. This could result in a handshake that only takes two additional steps, resulting in even less traffic that has to travel through the Mux. One potential downside to this alternative method, however, could be in the case of a malicious host. The sender Host Agent could send only a SYN packet and do nothing after receiving a SYN-ACK packet from the recipient Host Agent. This would offload some of the DoS avoidance techniques into the Host Agent (as opposed to the regular Fastpath technique, where the direct communication messages are only sent after the full TCP handshake). Another potential downside to this alternative method puts more complexity in each Host Agent it would have to listen for communication from other Host Agents all the time (as opposed to the normal Fastpath method, where it would only have to listen to the Mux, which then informs it to listen for communication from a specific Host Agent).

Finally, all of the evaluation in the paper of Ananta is either verification results to make sure their system works correctly, or is just testing its availability and responsiveness of the Ananta Manager. However, there is no comparison to how well it works against the load balancers the Azure cloud previously used. As it currently is, this does not give the reader much of a clear picture as to how effective Ananta truly is as a system. We feel like the authors could have made a stronger case for Ananta by including more concrete numbers showing how much less traffic the load balancer has to process (as opposed to leaving the reader to guess that around 80% of VIP traffic will not have to go through the Mux). Building on top of this, the authors also don't mention how much additional work each server has to do to have the Host Agent on its machine and do more communication. This data could have been interesting to see as it would show how

much additional work every single physical machine ends up performing with the adoption of Ananta (as there is Host Agent on every physical machine used in Ananta).

5 References

[1] https://gcn.com/Articles/2016/02/09/data-center-outages.aspx

[2] Roy, A. *et al.*, "Inside the Social Network's (Datacenter) Network," *Proc. of ACM SIGCOMM '15*, 45(4):123-137, Oct. 2015.