CDNs: troubleshooting and optimisation

Jaakko Ruutiainen
Aalto University School of Science and Technology
jruutiai@niksula.hut.fi

Abstract

Content delivery networks (CDNs) is an efficient mechanism for reducing delivery costs, increasing availability and improving performance of large-scale content retrieval. Today there is a number of commercial CDNs, such as Akamai, Google and Limelight, offering their services to media and computer companies.

Due to their scale, CDNs are hard to keep efficient. Researchers have been evaluating CDN performance and proposed various mechanisms to improve it. This study summarises troubleshooting and optimisation efforts on CDNs.

KEYWORDS: CDN, Performance, Simulation, Optimisation, Troubleshooting

1 Introduction

Content delivery networks (CDNs) are overlay networks in the Internet. CDN surrogate servers are distributed around the globe and clients are routed to the server that can best serve them. CDNs are widely used to save costs and to offer better performance to customers. Furthermore, CDNs are considered to be resilient to server failures, network outages and denial-of-service attacks [15].

Currently many commercial large-scale CDNs exist, such as Akamai [1], Google [7] and Limelight [9]. In addition to commercial CDNs there are also non-commercial CDNs, such as CoDeeN [4] and CORAL [6]. Large-scale CDNs consist of hundreds or thousands of servers and currently CoDeeN Statistics [5] lists over 400 servers participating in CoDeeN CDN. This high number of servers means that CDNs are hard to keep efficient and that it is not easy to predict how possible changes affect the performance nor plan how to efficiently increase the performance of the CDN. Further, it is obvious that building such a large system is not cheap and that CDN owners will want to achieve best possible results from their investments.

Several options for increasing the CDN performance exist, such as adding more servers to current data centers, adding more data centers and upgrading network connectivity. Unfortunately, all of these actions are somewhat permanent, e.g. one can not build a data center and then easily move it to some other location just to see if such a move increases, or decreases, performance. To evaluate these situations, tools, including CDNSim [14], WISE [16] and heuristic algorithms [3], exist allowing CDN operators to predict how their actions are likely to affect the CDN performance.

On the other hand, CDN performance issues could be caused by factors outside the CDN, such as inefficient routing between clients and the CDN. These conditions are not directly controllable by the CDN operators, but tools, such as WhyHigh [8], help finding those issues and make it possible for the CDN operators to contact parties that can fix them.

CDNs are used to provide a better experience for customers and one key aspect of the experience is the latency clients experience. Routing clients to surrogate servers is usually done based on latency, i.e. the client is directed to the CDN node with the lowest latency [8], but other CDN redirection policies do exist [14].

Most often used method to minimise latency is to increase the number of surrogate servers [16] and add servers to new locations geographically closer to clients. But optimisation of surrogate server latency does not necessarily optimise latencies experienced by the clients [8]. Also, other delays such as network queuing affect CDN performance perceived by clients.

Besides latency, CDN performance is measured by defining surrogate server utility [14]. The surrogate server utility is used to measure the relation between the amount of data served to clients against the amount of data fetched from the original provider server. The utility corresponds to the idea that surrogate server is most useful when it increases the performance of the original provider server by serving more copies of data than it fetches.

While improving CDN performance is important, CDNs face a situation where the CDN performance actually drops if no new capacity is added as the number of Internet users increases rapidly and new applications, such as high definition video streaming, need more bandwidth than older applications. To keep up with these developments CDNs need to increase capacity just to even maintain their current performance. Further, the Internet topology is constantly changing and making previously optimal CDN node placement less optimal than it could be [3]. As the Internet connection speeds have improved the users have become used to good performance and expect it also from CDNs [15].

Surrogate server placement effectiveness is evaluated by the figure of merit. Figure of merit indicates how close the server is to the clients. Optimal placement is the one with the lowest figure of merit [3], i.e. the node is as close to the client as possible.

Distributing static files such as web pages and file downloads are among traditional CDN services [14]. As Internet services have evolved and client connection speeds have increased, new rich web content has become big part of Internet traffic. Residential Internet traffic has already been moving to more streaming and video content, and stream-

ing protocols, such as RTSP, RTMP and SHOUTcast, add up to 5% of residential Internet traffic [12]. Streaming is also possible over the HTTP protocol and the overall amount of streaming is estimated to be around 20% of all Internet traffic. For content providers this means that they need to have lots of service capacity to provide sufficient performance.

As a result of the rising popularity of streaming content the leading CDNs are already offering services for distributing streaming media. The streaming media quality is badly degraded by even small delays or network outages which cause artifacts and freezes [15]. Furthermore, consumers are very sensitive to streaming performance and might change to a competing service if not satisfied by performance [2]. As achieving acceptable streaming performance takes considerable resources, the media companies should be very interested in streaming CDN solutions instead of building their own systems.

Because users are very sensitive to even small disruptions, harming streaming CDNs, or the stream source, is possible, even with relatively low resources [15]. For example opposing political parties might be tempted to disrupt streams from political events which are frequently streamed over the Internet.

In this paper, we go through some of the issues CDNs are having as well as some solutions others have proposed to these problems. Further, we compare these proposed solutions and their results.

2 Google's WhyHigh

Google has developed a system called WhyHigh [8] to analyse latency issues in their CDN. This system considers multiple attributes such as routing information, router geolocation and round trip times (RTT). In the Google CDN, two factors are found to cause significantly increased latencies. First, many network prefixes are routed inefficiently. Second, a significant amount of latency comes from queue delays. WhyHigh system concentrates on the first factor and the system has been able to find four different cases that cause increased latency. Namely these cases are lack of peering, limited bandwidth, routing misconfiguration and traffic engineering.

WhyHigh system focuses on RTT between end users and Google's CDN nodes. In their paper Krishnan et al. [8] present the classification of high RTTs and most cases of high RTTs are caused by traffic engineering or lack of peering between ASs. These are not in the hands of CDN operators to solve directly, but some measures, such as advertising different routing prefixes to the Google network, can be taken. Furthermore, limited bandwidth links between ASs or misconfigurations in the networks cause high RTTs, although much less often than traffic engineering or lack of peering. Even though CDN operators can not directly fix these issues, identifying them still proves that the CDN itself is not at fault.

Identifying problems in other networks is also important as ASs might not be aware of the problems, but are still very much willing to correct them because establishing peering links between ASs is expensive and ASs want to utilize those links efficiently [8].

In all of the four cases that are found to cause increased latency it is more effective, and most likely cheaper, to solve these networking issues than it is to just deploy new surrogate servers. However, as WhyHigh does not fully know all the routing configurations in use the cause for high latency can not be always identified, which shows that CDN operators cannot solve all performance issues simply by adding nodes. Further, it should be noted that WhyHigh observes the traffic paths only from one end, which makes it harder to identify the cause for high RTTs as many routing details are unknown. Observing traffic also at the client end would provide much more information, but that is not possible with WhyHigh.

3 Node placement

Somewhat related to WhyHigh, and to the problems it can identify, is the CDN node placement. Ideally, CDN nodes should be placed so that they are equally close to all clients. Bassali et al. [3] describe a heuristic algorithm for proxy placements. Further, they also present experimental studies of their heuristic algorithms. In total they define and evaluate three different CDN node placement algorithms. First is the highest-degree-first algorithm, which positions nodes in the Autonomous Systems (ASs) with highest amount of neighbours. Second is the farther-first algorithm, which positions nodes in the ASs with highest amount of neighbours and are far from other ASs with nodes. Third is the optimised-hybrid algorithm, which runs the hybrid algorithm for several times and selects the results with lowest figure of merit. The hybrid algorithm positions the first n nodes in the ASs with highest amount of neighbours, then it places the remaining nodes in the ASs with high amount of neighbours that are not neighbours of the already placed nodes and finally, if necessary, places remaining nodes to ASs with high amount of neighbours, even if those are neighbours with ASs that already have a node.

For evaluating their algorithms the authors [3] use Internet topology snapshot and then continue to apply the changes on the Internet topology to demonstrate algorithm performance over time. Optimal node placements would be achieved by selecting new node locations after each topology update, but in practice it is much more convenient not to move nodes to new locations every time the Internet changes. However, the figure of merit quickly begins to rise as the Internet topology evolves, while the already placed nodes remain in fixed positions. Therefore the amount of CDN nodes should be relative to the number of ASs in the Internet. In this changing environment a simple algorithm, e.g. highest-degree-first, works reasonably well. Overall the authors conclude that one simple algorithm can not produce the best results when the Internet topology is constantly evolving.

The algorithms have only a partial view of the Internet routing and the node placement efficiency could be improved with more complete and detailed view of the Internet. The algorithms treat all ASs as equal, while in reality some ASs are more likely to have users interested in the content distributed through CDN than others. For example, the consumer ISP's AS is more likely to have users downloading videos from the CDN than the AS of some big corpora-

tion. Nevertheless, positioning the CDN nodes in the right places is important as relocating servers is costly. Increasing the algorithms knowledge about network topology might be useful and provide even more accurate results. Such network topology information could, for example, be collected by the WhyHigh system. However, the WhyHigh system can only collect information from ASs where it already has nodes, which might limit the usefulness of such data unless one needs to add more capacity to the CDN by adding more surrogate servers in to an already existing data center.

4 Simulating CDNs

4.1 CDNSim

Stamos et al. [14] present a complete CDN infrastructure simulation environment, called CDNSim. The simulation system considers all CDN networking matters such as "surrogate server selection, propagation, queuing, bottlenecks and processing delays". Further, the CDNSim simulates the TCP/IP protocol, packet switching, packet retransmissions and it is aware of the network topology. Additionally the delivered content and content request patterns are generated artificially, but they resemble realistic models. They have simulated four different cases, namely CDN utility vs. network topology, CDN utility vs. popularity distribution, CDN utility vs. size distribution and CDN utility vs. CDN redirection policy. In other words they "have evaluated the CDN utility under different network topologies, traffic models and Web site models".

Regarding CDN utility vs. network topology they find that CDN utility has a peak at certain cache size, which occurs when cache size is set to 10%, and that the peak is independent of network topology. Important finding is that replicating small size of total content guarantees satisfactory performance. Cache size is important factor as it directly affects the CDN pricing.

For CDN utility vs. popularity distribution they observe that again the performance peak is at cache size of 10% regardless of popularity distribution. If the popular content is only small portion of total content, then the CDN can focus on caching those few popular objects and gain higher utility.

CDN utility vs. size distribution simulation shows that CDN utility is higher if large files are more popular than smaller ones. Again the utility has a peak at cache size of 10% for all tested values of correlation between size and popularity.

CDN utility vs. CDN redirection policy is the only simulated case where CDN utility does not have the performance peak with all tested parameters. The peak is only present with the "closest surrogate server with cooperation" policy. Which the authors conclude to mean that "poorly designed redirection policy would not exhibit the desired CDN utility peak" [14] .

Using CDNSim the authors have shown, among other findings, that CDNs have a performance peak in terms of CDN utility, which is constant regardless of the network topology, the traffic model and the web site model. This means that the CDN can be tuned to work optimally, which of course affects performance positively. CDN utility is also shown to be valid metric for expressing the usefulness of the CDN and the traffic activity in the CDN. The CDNSim is a useful tool for determining how various parameters of the CDN affect the CDN utility.

4.2 WISE

Tariq et al. [16] present a "What-If Scenario Evaluator (WISE), a tool that predicts the effects of possible configuration and deployment changes in content distribution networks". The evaluation focuses on service response times. What-if scenario could for example be a change in the number or location of CDN nodes, a change in cache size, or a change in network connectivity. The result from such scenario evaluation would be the effects on the service response time. WISE uses machine learning to model CDNs because the number of variables is large and relationships between variables are complex, but still the underlying properties can be observed as correlations. By observing these correlations the machine learning algorithms can find the functions affecting response times. The machine learning algorithms are also used to adapt source datasets for each scenario as obtaining real datasets is not possible.

In their paper [16] the authors use real usage and response time data, collected with previously existing network monitoring infrastructure, from Google's global CDN for Websearch service to test WISE. First, data from the week of June 17-23, 2007 is used to train the system. Second, the WISE system is used to predict response times for the week of June 24-30, 2007. Finally, the predictions are compared against real usage data from that time period. The median error for the predictions is found to be between 8-11%, which they find to be noticeably better than predictions obtained from simpler models.

Further, the WISE system is evaluated with data generated using Emulab emulation testbed as the Google's live production CDN only produces dataset that are similar to each other. However, real usage traces are used to run the emulations. Tariq et al. [16] have done two different experiments with this emulation environment. Namely, changing the resource size, where resource size is halved for the emulation, and changing the cache hit ratio, where 50% of the resources are cached for the emulation instead of the normal 10%. For the first emulation WISE has error of only 4.7% and for the second emulation the error is 4.9%.

With evaluation of WISE the authors have shown that WISE is an effective tool for predicting the effect of specific network deployments on the CDN performance. This is very useful when forecasting how the CDN can handle outages or when determining the best location to add more capacity in order to maximise the gained performance increase. Further they show WISE to be fast enough to use, even with frequent infrastructure changes.

5 Combining P2P and CDN

CDNs use high numbers of servers and data centers. Data centers also need reliable high bandwidth network connectivity. Neither servers, data centers nor network connectivity are cheap, which means that CDNs have limits on how many

clients they can server at once, and content originators need to pay more if they want to provision more capacity. Peer-to-Peer (P2P) systems on the contrary provide high scalability with minimal server resources [17].

P2P networks use connected nodes as both clients and servers thus solving the scalability issues without investing into servers and network connectivity. Popular P2P streaming services such as PPLive [13] are used by large number of users. However service performance can not be guaranteed as participating nodes are frequently removed from network. P2P traffic is blocked or throttled in some networks which means that some users might not able to connect at all to the service or they might experience unacceptable performance. Further, P2P architectures do not work well over NATs as the users behind NATs might not be able to upload at all because of connectivity issues. Further unfair network use occurs as users with good reachability and high upload bandwidth contribute by uploading significant amounts of data. P2P systems suffer from long startup delays, because establishing connections to peers capable of providing data takes time. For streaming data long buffers are needed as clients must have time to find new peers without causing disruptions to the stream in case of many peers leaving the network.

A hybrid CDN-P2P architecture combines the best aspects of CDNs and P2P systems and offsets the problems. Yin et al. [17] describe and evaluate one such system, namely LiveSky [10] which is used to stream live video. In LiveSky clients connect using special software. Clients are directed to closest, lightly loaded CDN node with DNS redirection. CDN node provides client a list of potential peers for P2P usage and small amount data to begin streaming. After the initialization clients download from peers in the P2P network. However, in situations where P2P can not provide sufficient performance clients can stream directly from the CDN node. The CDN nodes also act as seeds for the P2P network. The CDN nodes form a normal CDN and the non-P2P aware clients use it as a traditional CDN only without the advantages of P2P. High quality streams are available only to the P2P aware clients to encourage P2P use. [17]

CDN capacity needed is reduced as clients help the CDN by uploading to other peers by becoming small scale surrogate servers and therefore taking away some of the load. In a example provided by Yin et al. [17] the P2P network handles roughly one third of all the CDN-P2P traffic. In addition Lu et al. [11] prove that a CDN-P2P network can serve much more simultaneous clients than central server while maintaining acceptable performance. However, it is still possible to serve popular streaming content for many viewers without the P2P network component [15].

Nevertheless, by using P2P networks the CDN network operators lose the ability to carefully choose places where data is distributed from and malicious peers could disturb others by uploading false content. CDN-P2Ps also face problems of network engineering as P2P traffic shaping, or even total blocking, is used by some ASs. However, those problems might become less common once network operators realise that P2P is used for legitimate content delivery instead of piracy. CDN-P2P networks must still deal with P2P specific problems such as startup delays and rebuffering under extreme peer disconnection rates.

In addition to a hybrid CDN-P2P system is CDN peering. A hybrid CDN-P2P system is tightly coupled and combining heterogeneous CDN and P2P from different vendors is not easy. To overcome this a Web Services-based Content Delivery Service Peering Scheme (WS-CDSP) is proposed by Lu et al. [11]. In their scheme CDNs use Web Services to communicate and support peering. This system allows different CDNs, or CDN-P2Ps, to cooperate or to create a loosely coupled CDN-P2P architecture. The scheme allows flexible combining of a traditional CDN and a separate P2P architecture thus easily creating a CDN-P2P architecture instead of tightly coupling those two solutions together. The easiness of creating a CDN-P2P hybrid network should make it easy to experiment with different combinations.

6 Conclusions

In this paper we have reviewed several different solutions proposed for optimising CDN performance. Directly comparing these different solutions to each other is difficult. While all the solutions are ultimately trying to solve the same problem and help improve CDN performance, they take very different perspectives on this broad and complicated issue. Further, none of the papers referenced include direct comparisons to others. Because all of the solutions reviewed are trying to solve various different subproblems in the CDN performance, it would be unreasonable to conclude that one solution would be much better than the others. However, not all problems are alike thus they need slightly different solutions.

In their paper, Krishnan et al. [8] describe Google's WhyHigh system and they find that various problems cause elevated latencies and degraded performance in the Google CDN. The authors find four different causes for high RTTs, namely, lack of peering, limited bandwidth, routing misconfiguration and traffic engineering. WhyHigh is demonstrated to be an effective tool for troubleshooting problems with high, or higher than usual, RTTs [8], especially when the problems are outside of the CDN.

On the other hand the WISE is shown to be an accurate and fast tool for predicting the CDN performance changes as a result of changes to the CDN infrastructure [16]. Also with simulations Stamos et al. [14] find that CDN utility is a valid metric for measuring CDNs effectiveness and also that CDNSim works well for finding CDN utility performance peaks. Compared to CDNSim WISE uses machine learning and it does not need complex knowledge about underlying infrastructure making it easier to use with different CDN architectures. CDNSim however contains detailed implementation of TCP/IP protocol, packet switching and packet retransmission thus allowing more detailed simulations of varying network conditions. Results of the WISE system are more directly related to the performance perceived by the clients retrieving the content while the results provided by the CDNSim are more related to the internal efficiency of the CDN.

With CDNs one key aspect affecting performance is the location of CDN proxy servers. Bassali et al. [3] describe heuristic algorithms for selecting ASs as the CDN node locations. The CDN proxies must be wisely located in order to

provide sufficient performance for clients all over the world. They also show that new CDN nodes must be deployed as the Internet expands or the CDN performance begins to decrease.

Hybrid CDN-P2P networks on the other hand do not necessarily need as detailed network simulations because the P2P protocols dynamically choose the best peers. Dynamic nature of the P2P networks also make CDN node placement less important as data can be retrieved from different peers. However, the CDN still needs to create the P2P networks so that peers in one network are reasonably close to each other to enhance the network friendliness. Maybe the biggest obstacle when considering CDN-P2P solutions is the need for specially made client software which currently makes it impossible to use CDN-P2P for web content. However, CDN-P2P architectures decrease the capacity needed from the CDN [17, 11] because clients are acting as peers and providing content for the other peers. As a downside CDN-P2P networks require special client software capable of P2P for the system to function. On the other hand the web services based scheme for establishing CDN peering [11] between otherwise incompatible CDNs, or P2P networks allows CDN operators to easily combine, and evaluate different combinations, CDNs and P2P systems. Further, CDN peering allows two, or more, CDNs to combine capacity, if one bigger and more powerful CDN is needed.

We find the CDN-P2P architectures to be the most promising in improving the CDN performance, especially the streaming performance, with reasonable costs. Even when Akamai currently has one of the largest CDNs and is able to serve nearly one million simultaneous streams without any P2P functionality in the clients [15].

References

[1] Akamai Technologies. `http://www.akamai.com/`.

[2] Akamai Technologies. Akamai Study Uncovers Critical Link Between Video Quality and Audience Retention, Revenue Opportunities. `http://www.akamai.com/html/about/press/releases/2007/press_080707.html`.

[3] H. S. Bassali, K. M. Kamath, R. B. Hosamani, and L. Gao. Hierarchy-aware algorithms for CDN proxy placement in the Internet. *Computer Communications*, 26(3):251 – 263, 2003.

[4] CoDeeN. `http://codeen.cs.princeton.edu/`.

[5] CoDeeN Statistics. `http://fall.cs.princeton.edu/codeen/`.

[6] CORAL. `http://www.coralcdn.org/`.

[7] Google. `http://www.google.com/`.

[8] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. [moving beyond end-to-end path information to optimize cdn performance.

[9] Limelight Network. `http://www.limelightnetworks.com/`.

[10] LiveSky. `http://en.chinacache.com/`.

[11] Z. Lu, J. Wu, C. Xiao, W. Fu, and Y. Zhong. WS-CDSP: A Novel Web Services-Based Content Delivery Service Peering Scheme. In *SCC '09: Proceedings of the 2009 IEEE International Conference on Services Computing*, pages 348–355, Washington, DC, USA, 2009. IEEE Computer Society.

[12] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 90–102, New York, NY, USA, 2009. ACM.

[13] PPLive. `http://www.pplive.com/en/`.

[14] K. Stamos, G. Pallis, A. Vakali, and M. D. Dikaiakos. Evaluating the utility of content delivery networks. In *UPGRADE-CN '09: Proceedings of the 4th edition of the UPGRADE-CN workshop on Use of P2P, GRID and agents for the development of content networks*, pages 11–20, New York, NY, USA, 2009. ACM.

[15] A.-J. Su and A. Kuzmanovic. Thinning akamai. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 29–42, New York, NY, USA, 2008. ACM.

[16] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering what-if deployment and configuration questions with wise. *SIGCOMM Comput. Commun. Rev.*, 38(4):99–110, 2008.

[17] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 25–34, New York, NY, USA, 2009. ACM.