The Effect of Unpredictable Client Access Patterns on a Grid-Quorum CDN*

Christopher B. Mayer, K. Selçuk Candan, and Venkatesh Sangam

Computer Science and Engineering Department,
Arizona State University
{chris.mayer,candan,venkatesh.sangam}@asu.edu

1 Introduction

A *content delivery network* (CDN) replicates content and services to a multitude of servers strategically placed throughout the Internet, intelligently redirecting client requests to an appropriate server. Attractive properties of CDNs include reductions in network traffic and network delays, fast response times, reduced load at the origin server, and fault tolerance in case of server failures.

We consider replicating large objects, such as video, in a CDN environment. Large objects present a problem since their size makes it costly to maintain replicas at many servers. Because objects are large, read response time is dominated by the time needed to download an object. Thus, all other overhead, such as time to locate a server with the object, can be ignored. A second concern is maximizing CDN availability. Assuming servers are located close to clients, the main bottleneck in serving large objects is server capacity, not network resources. Since a server appears to fail when its capacity is exceeded and it becomes unresponsive, careful data placement and request redirection strategies that prevent servers from overloading are essential to CDN availability.

In this extended abstract, we first review a previously developed grid-quorum CDN system for replicating large objects and then present new results showing how a system behaves when request loads differ from that for which the system was designed.

2 A Grid-Quorum CDN

Many quorum-based object replication algorithms use grids as an organizational structure [1, 2, 4]. We use a basic grid structure for placing replicas and for load balancing where rows of the grid are *write quorums* and columns are *read quorums* (Fig. 1). To read an object, a client first selects a CDN server (a *proxy*) using a probabilistic strategy (the *client strategy*) and transmits an object request (an *initial-request*) to the proxy. The proxy then selects a read quorum according to a predetermined, probabilistic *proxy strategy*. The proxy interrogates each server in the read quorum and selects the one with the most recent copy. The proxy then redirects the client to the selected server for downloading.

To minimize the likelihood of server overloading, each server should experience a load proportional to its contribution to the system's total capacity. In

* This research funded by NSF grant 998404-0010819000.

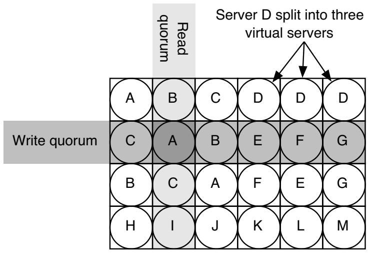

Fig. 1. Example grid-quorum CDN. Note that the highlighted write and read quorums intersect at servers A, B, and C, not just at A.

other words, system load should be distributed *fairly*. To help ensure fairness, each CDN server is split into one or more equally powerful *virtual servers* . The quorum structure is formed by filling a grid with virtual servers (Fig. 1). Since each virtual server represents about same amount of capacity, load fairness can be acheived by evenly distributing read load to grid cells. The load on each grid cell, on the other hand, depends on (1) the arrangement of virtual servers (the quorum structure), (2) the CDN policy for selecting write quorums, (3) initial-request loads at the proxies induced by the client strategies, and (4) the proxy strategies.

A detailed analysis of our approach was performed in [3]. The analysis identified two parameters, σ and N, indicating the quality of a CDN design. σ of r represents the fraction of read requests directed to read quorum r by the proxies. N of s indicates the degree of read and write quorum intersections for server s with respect to other servers. A server's load is closely tied to the σs of its read quorums and its own N value. For instance, the higher N is, the higher s's load will be. Likewise, the higher the σs of a server's read quorums (*average server σ*) are, the more load the server will receive. A CDN will have fair load distribution when each server's N value equals the number of virtual servers the server has *and* each read quorum's σ equals $\frac{1}{number\ of\ read\ quorums}$.

Producing ideal Ns and σs is a non-linear problem which is difficult to solve. Thus in [3] we presented a two-stage, heuristic approach for constructing CDNs that turns the non-linear problem into a linear one. In the first stage, we build the quorum structure, controlling the critical N values, by filling a grid with rectangular clusters of virtual servers where the servers in each cluster have the same capacity. In the second stage, a linear program (LP) is solved in order to determine proxy strategies for achieving optimal σs.

3 Impact of Unpredictable Initial-Request Loads

In the work of [3], client strategies for selecting proxies are an input to the LP. *Thus the CDN replication strategy depends on expected initial-request loads at the proxies at the time the strategy's formulation.* Unfortunately, in a real CDN, client strategies, and hence initial-request loads at proxies, may change over time. We need to explore system behavior when initial-request loads vary from that for which the CDN was designed. This is done experimentally as described below.

Twenty server sets were randomly created such that the virtual servers in each set fill an 8x8 grid and each server has between one and five virtual servers. Virtual servers in each server set were mapped to grids using (1) the clustering heuristic for good N values and (2) randomly to produce bad N values. For each resulting grid, σ-optimizing proxy strategies were formulated using the LP approach. Non-σ-optimized proxy strategies, in which a proxy chooses read quorums equiprobably from those in which it has virtual servers, were also created for each grid. The result is four combinations of mappings and proxy strategies for each server set. Experiments using the four strategies for each server set were performed using a testbed system of actual Web servers. The metric of interest, *server load*, is the average number of downloads a server supported in each second of an experiment. Objects had a uniform download time. Experiments were run long enough to capture long-term server loadings. A *baseline load* of 256 intitial-requests per server represents proxy intitial-request loads expected at the time of a CDN's design. To view the effects of varying client strategies, subsets ($\lceil 10\% \rceil$, $\lceil 20\% \rceil$, and $\lceil 40\% \rceil$ of servers in a set) were subjected to extra initial-requests of 25% then 50% above the baseline load.

Experiment results answered six questions about the influence of σ and N on system behavior when initial-request loads vary. The questions and short answers to each are given below. Unless otherwise noted, results are for cases where 40% of servers received 25% and 50% extra initial-requests as similar results were observed for other cases.

Q1: *If a server's N value increases or decreases between different quorum structures, does the extra read load experienced by the server also increase or decrease?* Comparing grids with bad N (randomly-mapped) to those with good N (clustered), we observed that if a server's N value increases (decreases) between mappings, then around 83%–86% of the time its load increases (decreases) as well, regardless of extra initial-request load. This behavior indicates that N has a strong influence on where extra load is distributed.

Q2: *If the σs of a server's read quorums increases or decreases under different proxy strategies, does the extra read load experienced by the server also increase or decrease?* In a similar manner to Q1, we see that if a server's average σ increases (decreases) between proxy strategies, its loading increases (decreases) as well. However, the influence of σ is not as pronounced as for N as only around 61%–65% of the time does a change in average server σ correspond to a change in load in grids mapped for bad N. However, in mappings with good N, the correllation is in the range of 81%–85%. This indicates that the impact of optimizing for σ is greatest only after controlling for N.

Q3: *How is extra read load distributed in clustered versus randomly-mapped systems?* In randomly mapped systems, a few servers receive the bulk of the extra load. When cluster mapping is used, 64% of the servers get higher amounts of extra load than they get when randomly mapped. This indicates that systems with good N do a better job of distributing extra load to servers.

Q4: *How is extra read load distributed in systems that are σ-optimized versus non-σ-optimized?* Load distribution is minimally affected by optimizing for σ, with observed load changes ranging from -4% to 3.5% for randomly mapped grids and from -5% to 3% for clustered grids. Comparitively, load shifts in the range of -12.5% to 6% were observed for randomly-mapped grids versus clustered grids. Combined with the behavior seen in the answers to Q1 through Q3, we conclude that N (quorum structure), and not σ (proxy strategies), is the main factor in determining where extra read load is distributed.

Q5: *What does a server's N value indicate about its extra read load?* This question has been partially addressed by Q1 and Q3. Further analysis of server load and N values show that extra read load is highly variable for small N and becomes less variable as N increases. For instance, in randomly mapped grids swings in extra read load ranged from -22% to 43% when $N = 1$ but only from -15% to 18% when $N = 5$. This implies that the system could be made robust to variances in initial-requests by making N values large. However, doing so could destroy load fairness under expected initial-request loads. We also observed that read loads can remain fairly level as initial-requests increase, but dramatically jump beyond some threshold, regardless of mapping strategy.

Q6: *What do the σs of a server's read quorums indicate about the server's extra read load?* When N errors are small, as in cluster-mapped grids, server load is proportional to average server σ. When N errors are large, σ's influence is negligible.

4 Conclusion

This extended abstract briefly reviewed the design and operation of a grid-quorum-based CDN for replicating large objects. Since CDNs are designed for expected client strategies for contacting servers with read requests, variations in client strategies, specifically when some servers receive extra numbers of requests, affect load fairness. Experimental results indicate that quorum structure, as represented by the parameter N, has the most influence on extra load distribution. Also, server loading shows some resilience to increased request loads, but quickly deteriorates past some threshold.

References

[1] A. Kumar. An efficient supergrid protocol for high availability and load balancing. *IEEE Transactions on Computers*, 49(10):1126–1133, 2000.

[2] A. Kumar, M. Rabinovich, and R. K. Sinha. A performance study of general grid structures for replicated data. In *International Conference on Distributed Computing Systems*, pages 178–185, 1993.

[3] C. B. Mayer, K. S. Candan, and V. Sangam. Constraints, parameters, and strategies for replicating large content for web delivery. Technical Report TR-02-007, Computer Science Department, Arizona State University, 2002.

[4] O. E. Theel and H. Pagnia-Koch. General design of grid-based data replication schemes using graphs and a few rules. In *International Conference on Distributed Computing Systems*, pages 395–403, 1995.