Globule: a User-Centric Content Delivery Network

Swaminathan Sivasubramanian Berry van Halderen Guillaume Pierre

Dept. of Computer Science, Vrije Universiteit, Amsterdam
{swami,berry,gpierre}@cs.vu.nl

1 Introduction

Web users often experience slow document transfers caused by poorly performing servers and overloaded network links. A classical solution to this problem is to replicate documents across the Internet and serve client requests at a nearby replica. Content Delivery Networks (CDNs), such as Akamai, offer such replication services to content providers by offering them a worldwide distributed platform to host their content.

This poster presents *Globule*, a content delivery network to be operated by end-users themselves. Instead of delegating content delivery to an external company, content providers can organize together to trade their (relatively cheap) local resources against (valuable) remote resources. As a result, a user participating in the Globule network is offered a distributed set of servers in which his/her Web content can be replicated.

Globule is designed in the form of an add-on module for the Apache Web server. To replicate their content, content providers only need to compile an extra module into their Apache server and edit a simple configuration file. Globule automatically replicates the site's content and redirects clients to a nearby replica. Servers also monitor each other's availability, so that client requests are not redirected to a failing replica. Future versions will allow automatic replica placement, replication of dynamic content and security measures. Globule prototypes are available under an open-source licence at `http://www.globule.org/`.

2 Replica Placement

The current prototype of Globule requires that content providers manually configure the list of servers where documents should be replicated. In future versions, however, replica placement will be realized automatically in order to maintain client-to-replica latencies as low as possible.

Automatic replica placement is realized in three stages: first, servers passively evaluate their latency to each client. This operation is completely transparent to the client: round-trip delays are measured by the servers upon establishing incoming TCP connections as the delay between the emission of the SYN/ACK packet and the reception of the corresponding ACK. These latency estimates allow servers to position each client in an N-dimensional space [7]. The second step consists of dividing this N-dimensional space into network regions and selecting a set of regions where replicas must be placed. The third step requires to choose the best-suited replica server(s) in each selected network region, and dynamically create replicas. For more details on automatic replica placement, please refer to [6].

3 Dynamic Adaptation of Replication Strategies

Replicating Web documents permits to balance the load among multiple servers as well as reducing the client-to-server latencies. However, it introduces consistency problems when a document is updated. To prevent client from accessing stale information, whenever a document is updated all existing copies of that document should be updated or destroyed. There exists a wide range of replication strategies which are able to maintain consistent replicas. In our earlier research, however, we showed that there is no single best performing strategy [1]. The best performing strategy must be decided on per-document basis based on its unique access and update patterns.

Globule servers support four different strategies. Based on analysis of past access pattern, the system decides which policy performs best for each individual document. To account for changes in access patterns, replication strategies are evaluated periodically. If necessary, the replication strategy of a document is switched dynamically. For more details, please refer to [3].

4 Client Redirection

Replicating Web content is useless if clients do not access it from a replica close to them. However, we consider it unacceptable to require clients to manually select their best replica. Instead, Globule automatically redirects clients to the best possible replica server that has a replica of the requested content.

Automatically redirecting clients requires to select a *redirection policy* and a *redirection mechanism*. The former dictates the choice of a server among all the replica servers to serve a client, while the latter is the means by which the client is informed of the choice. We have implemented two different redirection policies: *round-robin* redirects requests to replica servers in a round-robin fashion; *AS-path length* redirects requests to the closest replica server, with the proximity between the Web clients and the servers defined in terms of number of Autonomous System (AS) hops between them. Clients can be redirected to the selected replica using any of two mechanisms: one uses HTTP redirection to instruct browsers fro which server each particular document should be fetched; the other one uses a custom DNS server to resolve the name of a Web site into the IP address

of the replica closest to that client. All aspects of client redirection (including the custom DNS server) are implemented as part of the Globule module for Apache [5].

5 Disconnected operations

Globule allows replica servers to be operated by end-users. A consequence is that these machines cannot be considered as highly available. Unavailable replica servers may create problems when combined with automatic client redirection because clients cannot override the automatic replica selection in case something goes wrong. In such cases requests will fail, even though there may have been other available replicas capable of serving the request correctly. Dealing with unavailable servers is not only relevant when a server fails, but also to allow for replica server maintenance, software upgrades, network outages, etc.

In Globule, the origin server (i.e., owner) of a document periodically checks the availability of its replica servers. When it detects that one of them is unavailable or misconfigured, it stops redirecting clients to this replica until the replica has recovered. This simple scheme allows to support disconnected operation of replica servers, but not that of the origin server of a given document. We plan to address the disconnection of origin servers by splitting the client redirection mechanisms from the origin server. This allows for the construction of highly available redirectors, for example based on redundant redirecting DNS servers. Such mechanisms will allow a site to function correctly, even in the case where its origin server is unreachable.

6 Dynamic document support

The past few years have seen a tremendous increase in the usage of dynamic document technologies by Web content providers. Such documents are generated on the fly by arbitrary code written in languages such as PHP of JSP, which in turn may access relevant data stored in files or databases. Replicating dynamic documents requires to replicate both the code and the data, so that documents can be generated at any replica server.

Replicating dynamic document's code and data is relatively straightforward provided that the code does not modify the underlying data. However, if it does, then the system must maintain consistency among the replicated data. This problem is harder than maintaining consistency of static documents because concurrent updates can originate from any replica. In addition, traditional consistency strategies incur a large performance overhead due to synchronization between replicas.

We propose to reduce replication overhead by not replicating all data at all replica servers. Instead, data should be replicated only at servers that access them often. Simulations have shown that this provides important performance gains in document generation time, while reducing the data update traffic exchanged between the replicas. For more information, please refer to [4].

7 Security

Replicating content on unknown and untrusted hosts creates a security problem, as for example a malicious replica server may serve modified versions of documents. However, traditional techniques such as digitally signing documents do not suffice because most existing Web browsers would not check these signatures.

Globule addresses this challenge by taking both preventive and punitive measures. First, it defines a trust metric amonst servers. A given server will only replicate its content at servers that it trusts. Second, random checks are performed on the replica servers, by comparing secure hashes (e.g., SHA-1) of documents delivered by the replicas to that of the origin server. If the hashes are found to be different, then the owner of document is informed of the incident and the malicious server can be banned from the network of trusted servers. For more information, please refer [2].

8 Conclusion

We presented Globule, an open source content delivery network formed out of a distributed set of co-operating replica servers. Globule aims to allow Web content providers to organize together and operate their own world-wide hosting platform. It handles replication, consistency, client redirection, and disconnected operation automatically. Future versions will incorporate automatic replica placement, dynamic document support and security. Globule is available under an open-source license at `http://www.globule.org/`.

References

[1] G. Pierre, M. van Steen, and A.S. Tanenbaum. Dynamically selecting optimal distribution strategies for Web documents. *IEEE Transactions on Computers*, 51(6):637–651, June 2002.

[2] B. Popescu, B. Crispo, and A.S. Tanenbaum. Secure data replication over untrusted hosts. In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS 2003)*, 2003.

[3] S. Sivasubramanian, G. Pierre, and M. van Steen. A case for dynamic selection of replication and caching strategies. In *Proceedings of the Eighth International Workshop on Web Content Caching and Distribution (WCW'03)*, Hawthorne, NY, USA, September 2003.

[4] S. Sivasubramanian, G. Pierre, and M. van Steen. Replicating web applications on-demand. Submitted for publication, April 2004. `http://www.globule.org/publi/RWAOD_draft.html`.

[5] M. Szymaniak, G. Pierre, and M. van Steen. Netairt: A DNS-based redirection system for Apache. In *Proceedings of the IADIS International Conference on WWW/Internet*, Algarve, Portugal, November 2003.

[6] M. Szymaniak, G. Pierre, and M. van Steen. Latency-driven replica placement. Submitted for publication, May 2004. `http://www.globule.org/publi/LDRP_draft.html`.

[7] M. Szymaniak, G. Pierre, and M. van Steen. Scalable cooperative latency estimation. In *Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS)*, Newport Beach, CA, USA, July 2004.