ISP Server Farm

Better Surfing Through Caching

Jason Goggin, Erik Nordlander, Daniel Loreto, and Adam Oliner

April 1, 2004

1 Introduction

The Internet places heavy demands on servers to provide a continuous, reliable, and efficient service to clients. Servers should be resilient to failures, able to handle very large numbers of requests, and able to answer those requests quickly. A common way to provide these features is by having multiple networked servers share their resources for the benefit of all. These servers are usually all housed in a single location and are commonly called a *server farm*.

Of particular interest to us are ISPs: client requests come through ISP routers and must be handled appropriately. The service provider is responsible for maintaining the connection to the Internet, and managing the infrastructure that guarantees fast, reliable access. The ISP can achieve this, in part, through caching.

The aim of this project is to provide the following deliverables for an ISP system:

1. A specification for the system.
2. High-level code implementation of the system, emphasizing the tricky and novel parts of the system.
3. The abstraction function and key invariants for the correctness of the code.
4. Proofs of correctness for the code whenever appropriate.

Among the different aspects we will consider in the design of our system are:

- Scaling: How easy is it for our system to grow? Can we add new servers easily? How many servers can we support?
- Initialization: How is the system brought to its starting state? If a server needs to be restarted, how do we initialize it so that it is in agreement with the rest of the system?
- Partitioning of Data: How will the data in our farm be divided? Is data to be distributed among several servers, or will each server be used for a specific type of data? Which servers will hold what information?
- Replication for Availability: If there is very important data for which we must guarantee availability, should we replicate it? How and where should we do that?
- Persistent Caching: Should frequently requested data be "cached" on multiple servers to increase performance? Where do we place copies of the data in order to avoid bottlenecks and balance the load?
- System Failures: How can our system fail? How can we recover from failed system components? What are the consequences of a failure?
- Automation: The system should perform with minimal manual intervention. In particular when it comes to load balancing and handling failures.

In the next section, we describe our technical approach to the problem and discuss some basic ideas behind our design.

2 Technical Approach

The system can be viewed as consisting of four components: routers, management servers, caching servers, and persistent content servers. This section describes each component. An overview of the system is shown in Figure 1.

2.1 Router Level

The router level is composed of the routers and network connections that bridge between the exterior network and the caching servers. The routers forward incoming requests for data to the appropriate caching servers. Some such requests are sent around the caching servers and directly to the content servers. Their routing tables can be changed dynamically and are controlled by a set of management servers.

2.2 Caching Level

The cache level is the heart of our system. It consists of a number of stateless servers that cache data requested by clients. These servers receive requests from the router level and determine, using time stamps, whether they should return their locally cached copy of the data to the client or retrieve a fresh copy from the server. Multiple caching servers may contain the same data, and some content servers may not have any caching server assigned to them.

There are a variety of replacement policies, but one ultimate goal of the strategy should be to minimize a cost function; most likely, this will relate to latency or bandwidth costs. The caching servers will cache data in their persistent storage, as well, because disk latencies may be exceeded by network latencies.

2.3 Content Level

This level consists only of the persistent servers. The job of each of these servers is simply to store the data on a stable medium and serve material back to clients and cache servers. These machines are not necessarily under our control, and some may be web servers across the world. Indeed, those are the kinds of machines for which our system will be most beneficial.

2.4 Management Level

The management servers control aspects of the routers and caching servers. The management servers sample requests as they are sent to the routers. The goal of the management servers is to manage routing tables to ensure that the caching servers are not overloaded, and that the appropriate data is cached. Periodically, the management servers update routing tables and instruct the caching servers to move data among themselves. The management server is aware of what caching servers are responsible for what storage servers.

Each management server is mirrored by a number of hot backup servers that process all incoming requests alongside the primary server. Therefore, these backups always contain the current state of the system. In the event of a failure, the system can fail over quickly to another backup management server. When a management server is restored, it can copy the current state over from one of the active backups. The system will not be able to function optimally for very long without any management servers, but will still function.

In a more geographically disperse system, there can be a number of management servers active at the same time. For example, an ISP with locations in New York and Boston may wish to have cache servers active in each location, which could each be controlled by their own collection of management servers.

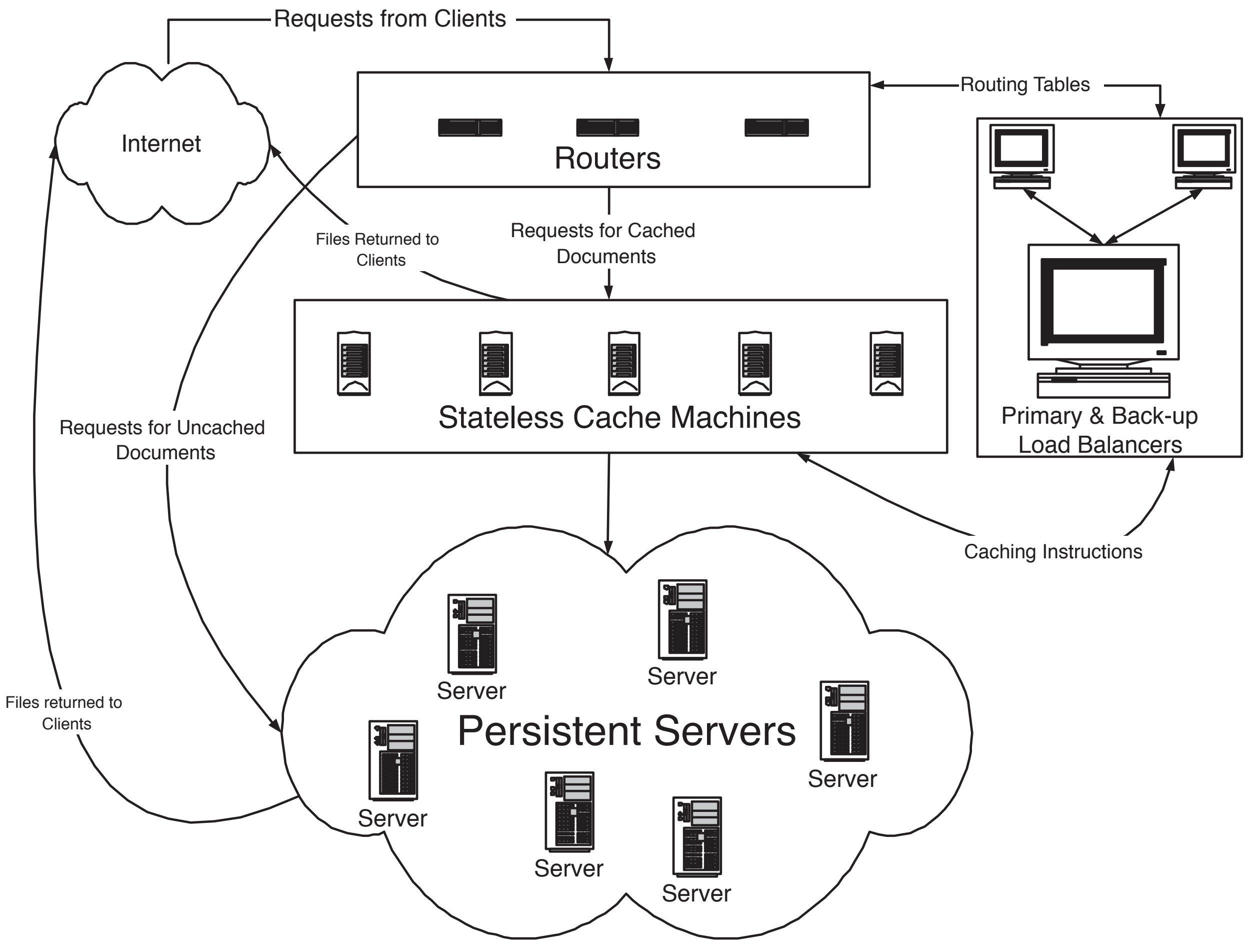

Figure 1: An overview of the network architecture. The system is composed of three main components: the routers, the stateless cache machines, and the load balancers that manage the other two. Requests are made by clients of the ISP. The persistent servers may either be local to the ISP or may be arbitrarily far away. The distant servers are the ones for which caching is most beneficial.