

Smart Internet Accessing System

Dipsy Kapoor, Sameer Maggon, Vikram Saxena, Deepak Malik
Computer Engineering
Dr. D.Y. Patil College Of Engineering
Pimpri, Pune - 411 018
Email: sameermaggon@rediffmail.com

Abstract

Clusters of servers, connected by a fast network, are emerging as a viable architecture for building a high-performance and highly available server. This type of loosely coupled architecture is more scalable, more cost-effective and more reliable than a single processor system or a tightly coupled multiprocessor system.

This paper presents information about Smart Internet Accessing System (SIAS). SIAS is a high-performance and highly available server built on a cluster of proxy servers. Scalability is achieved by transparently adding or removing a node in the cluster. High availability is provided by detecting node or daemon failures and reconfiguring the system appropriately.

1. Introduction

To provide Internet access to multiple users through a common Internet connection, a Proxy Server [1] is used. A Proxy Server has many functions and one of them is to cache the pages requested by the user, and hence improves the efficiency. As the number of users logged on to the net increases (say about 80-100 clientage), the access speed goes down.

To provide access to high clientage, Proxy Servers are connected in hierarchical fashion [2]. A number of proxy servers are taken with clients distributed among the Proxy Servers. The Proxy Servers are in turn connected as parent-child and siblings. A request for a page is searched in the immediate parent, and all other Proxies using the ICP [3] (Internet Cache Protocol). This process offers network delays because of waiting for ICP replies and timeouts in case the ICP reply does not come. Also whenever a page is fetched from a Proxy higher in the hierarchy, the page is replicated in all proxies in the path from the net to the client.

Therefore there are delays because of ICP and data redundancy in Proxy-Hierarchy Relationship. Also there is static assignment of clients to the Proxies because of which load is not balanced.

To provide access to high clientage, and to remove the drawbacks of Proxy-Hierarchy relationship, "Smart Internet Accessing System" can be used. In this system, a cluster of

proxies is used which are all at the same level (unlike in proxy-Hierarchy where we have parent-child relationship) and there exists a Load Balancer that is used to divert request to the proxies.

Logically, all clients are connected to the Load Balancer and in turn the Load Balancer maintains connections with all the Proxies in the network.

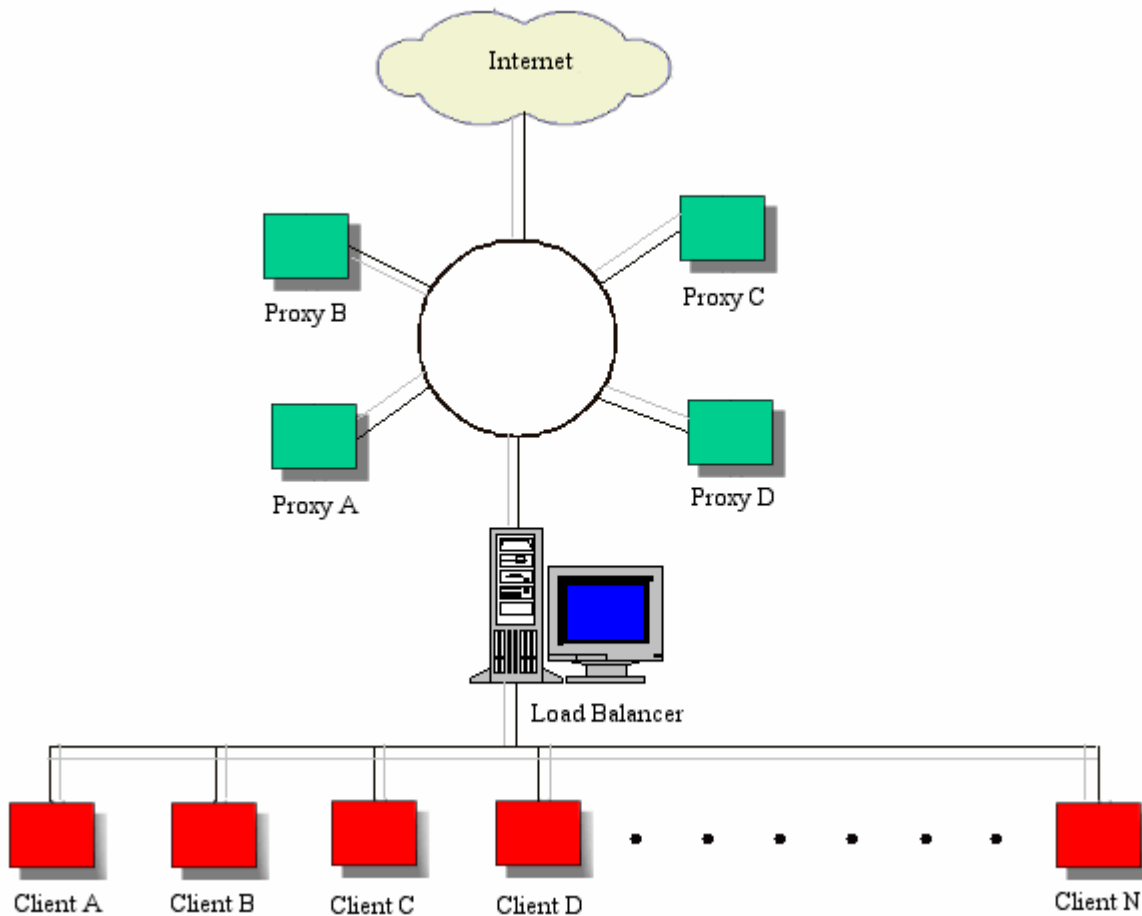


Figure 1.0 SIAS logical setup

The Load Balancer maintains a table called as a Cache-Digest that has information about the pages in all the proxies. Whenever any client requests a page, the Load Balancer looks up for the page in the Cache-Digest. If the page is present in the Cache-Digest of any of the Proxies, the request is diverted to that proxy. If the page is not present in any of the Proxy Caches, the request is diverted to the proxies in the network in a Round-Robin fashion.

Therefore, SIAS removes all drawbacks of Proxy-Hierarchy. SIAS provides a number of facilities like:

- Authentication support so that only authenticated users are allowed to access the Internet.
- User Interface for the administrator to monitor the request flows and users logged on.
- Logs for Administrative purposes

The whole system sits between the clients and the Internet gateway and is transparent to the user. The Load Balancer lies between the clients and the cluster of proxies. All the clients are connected to the Load Balancer and the Load Balancer is connected to all the proxies in the cluster. The Proxy Servers send their HTTP requests to the gateway and fetch pages what they want.

2. Related Works

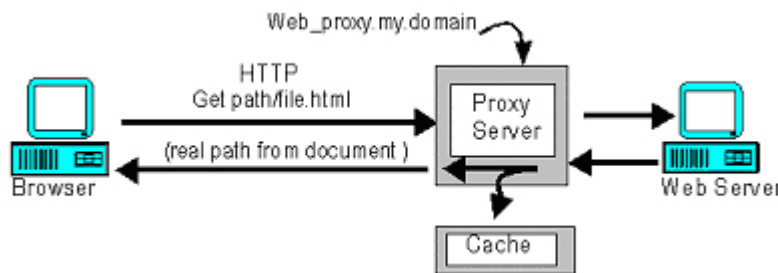
Proxy Server

At the client side, to make multiple users access the Internet simultaneously on the same line, and to schedule their request, a proxy server is used.

Proxy servers have two main purposes

- *Improve performance*

Proxy servers can dramatically improve performance for groups of users. This is because it saves the results of all requests for a certain amount of time. Consider the case where both User X and User Y access the World Wide Web through a Proxy Server. First User X requests a certain web page say `www.page1.com`. Sometime later, User Y requests the same page. Instead of forwarding the request to the web server where `www.page1.com` resides, which can be a time consuming operation, the proxy server simply returns the page it had fetched for User X from its cache. Since the proxy server is often on the same network as the user, this is much faster operation.



- *Filter requests*

Proxy servers can also be used to filter requests. For eg. A company might use a proxy server to prevent its employees from accessing a specific set of web sites. For transactions of a client with the proxy server, the client only uses HTTP, even when accessing a resource served by a remote server using another protocol, like FTP. When sending a request to a proxy, the full URL is specified and not just the pathname along with optional search keywords as with regular HTTP request. Proxy Cache Usually, the clients within a subnet access the same web proxy server. Some proxy servers let you cache Internet documents for clients within the LAN. Caching documents means keeping a local copy of Internet documents, so that the server doesn't need to request them over and over again.

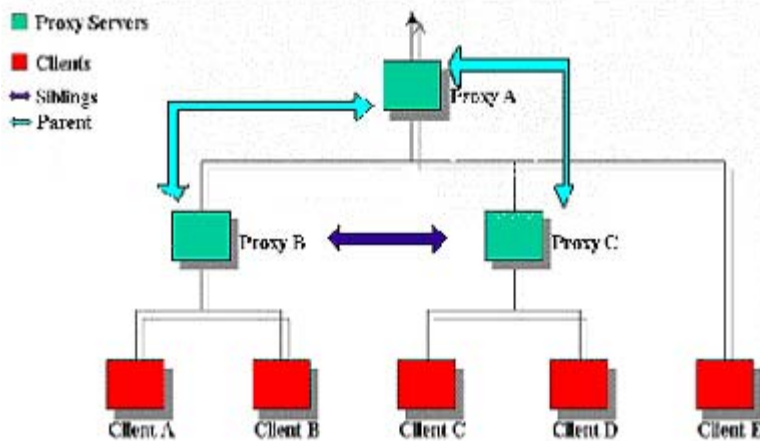
Proxy Hierarchy

Here the proxy servers are connected in hierarchical way. Users requests are first of all

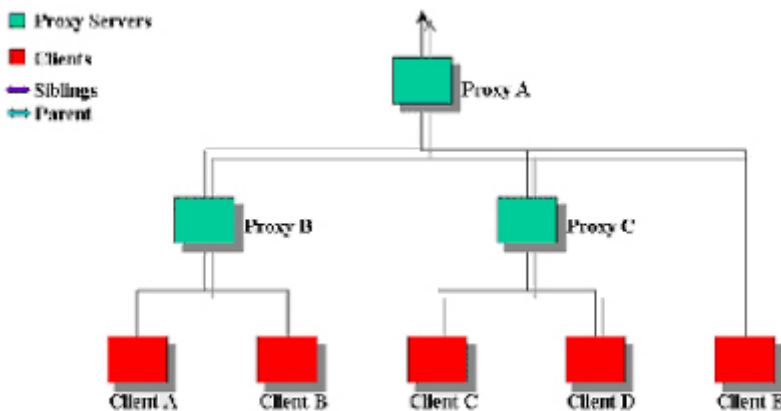
being processed by a local server, and then, if it could not find the requested information within the local cache area, it turns to another proxy server according to the hierarchy rules that were set. And if still no hit has occurred, the request will be sent to the original server.

In a cache hierarchy one cache establishes peering relationships with its neighbor caches: either parent or sibling. A parent is one level up, and a sibling cache is in the same level. The general flow of requests is up the hierarchy. A cache that doesn't hold the requested object refers to its neighbor caches to see if they might have the object. If one of the neighbors has it the cache will request it from them.. Else, the cache forwards the request either to a parent, or directly to the origin server. A "neighbor hit" can be cached from parent or sibling while "neighbor miss" can't be fetched from a sibling.

The hierarchical structure of Proxies is shown below:



The clients send a URL request to the Proxies. If the page is in the cache of the Proxy i.e. Cache Hit occurs, proxy sends the page to the client. On a cache miss, The Proxy would end ICP (Internet Cache Protocol) requests to the siblings and the parent to check for the page in their cache. If Cache hit occurs at any one of the siblings or parent, then an ICP reply is send. On a miss, the page is requested from the net.



3. SIAS Architecture

The project is divided into four modules.

1. Authentication Module

The Authentication Module provides the facility of restricted access to the Internet. This module is optional and can be turned on/off at compilation time. If the module is turned on, all clients run the client software through which they authenticate themselves to the Load Balancer. As soon as a user authenticates himself, his name is put in the ActiveUserList maintained at the Load Balancer. Access to the net is through the Load Balancer and hence any request is first checked to see if it is authenticated by checking its presence in ActiveUserList and only if it is, is it forwarded.

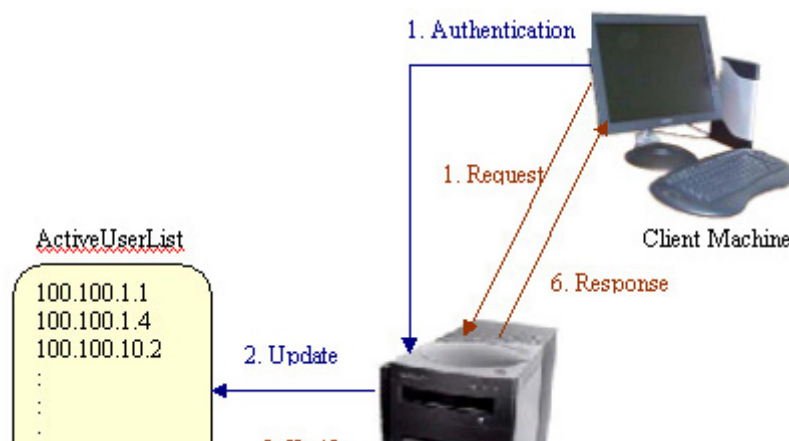
The Administrator is also provided with a tool to Add, Modify etc Users, to organize users into groups and to see the utilization of the net by the users through proper logs.

2. Cache Digest Module

The Cache Digest Module maintains the information of presence of web pages in each Proxy's cache in a table. It can be viewed as a simulation of the Hash-tables maintained by the Proxies to access their caches. The table is called as a Cache-Digest and is maintained at the Load Balancer. This table gives the Load Balancer the power to know about all proxy caches and hence it is able to divert the requests based on the information available in the Cache-Digest.

3. Squid Listen Module

This module handles the communication between the Load Balancer and the cluster of proxies. The proxies running in the cluster send messages to this module informing the cache updates and their status. The module takes care of the current number of proxies in the cluster. Whenever some new proxy is added to the system, a message is sent to Squid Listen module informing it about its presence so that the new Proxy server could be used for further requests. The deletion of the proxy is also taken care by removing the proxy from the Proxy list that in turn informs Load Balancer module to stop sending requests to that proxy. The module also informs the Cache Digest module about the new additions and deletions in the proxies' cache.



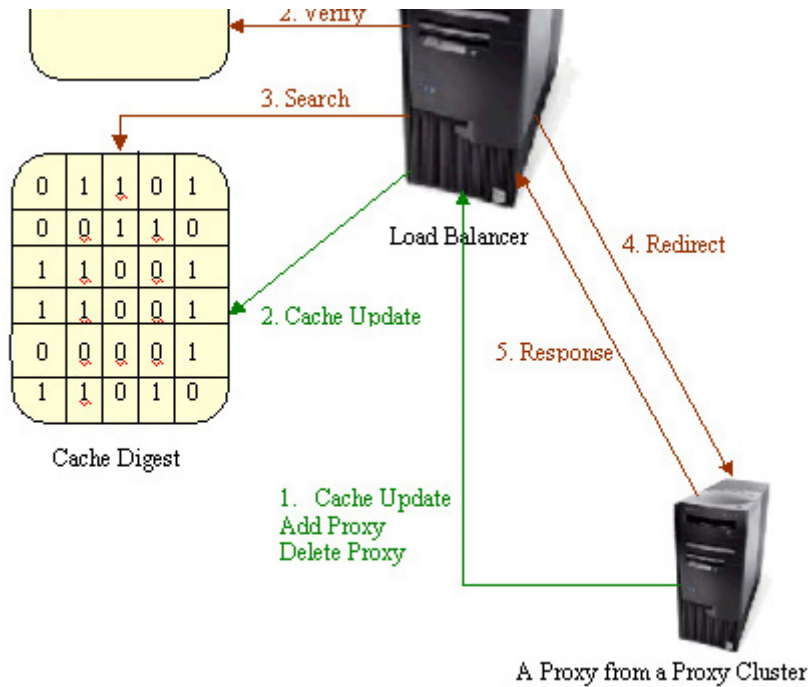


Figure 2.0 SIAS Work Flow

4. Load Balancer Module

The Load Balancer is the heart of the system. It interacts with all the modules and maintains the system-wide Data structures required by all the modules.

- The Load Balancer builds the ActiveUserList from the information given by the Authentication module.
- It builds up the Cache-Digest using the Cache-Digest module from the information given by the Squid-Listen module.

All client requests are diverted to the Load Balancer. The Load Balancer then performs a series of operations that are described below:

1. It checks if the user is authenticated by checking its presence in the ActiveUserList. If it is authenticated, the request goes to the stage 2 given below else user request is discarded as it is given appropriate message.
2. The Load Balancer extracts the URL name from the request. It then performs calculations on the URL and checks for its presence in the Cache-Digest.
3. If the URL is present in the Cache-Digest, it indicates the presence of the page in one of the proxy's cache. The proxy IP is found and the request is diverted to that proxy.
4. If the URL is not present in the Cache-Digest, the Load Balancer diverts the request to any one proxies in its cluster (actually in Round-Robin fashion).

4. Limitations

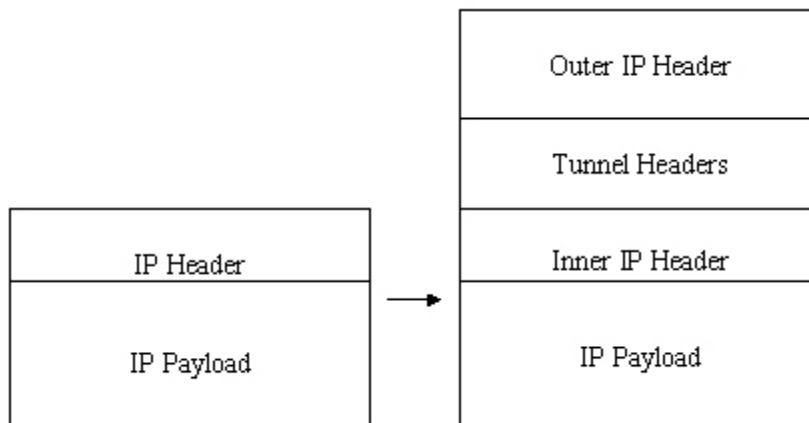
- Only HTTP[4] requests have been taken care of as squid normally caches HTTP

data.

- The software takes care of only HTTP GET method.
- The Load Balancer requires a system that supports POSIX threads. Linux kernel 1.3.56 onwards supports threads therefore the software requires kernel greater than 1.3.56.
- Modules like Authentication, Interfaces are not pluggable. The support for these modules has to be provided at compile time.

5. Future Work

- All requests and replies are forwarded through the Load Balancer. What can be done to further improve the performance is to use IP Tunneling [5].



IP tunneling (IP encapsulation) is a technique to encapsulate IP datagram within IP datagrams, which allows datagrams destined for one IP address to be wrapped and redirected to another IP address.

When a user accesses the system, a packet destined for virtual IP address (the IP address for the load balancer) arrives. The load balancer examines the packet's destination address and port. The Load Balancer chooses a Proxy Server according to a scheduling algorithm. Then, the load balancer encapsulates the packet within an IP datagram and forwards it to the chosen server. When the Proxy Server receives the encapsulated packet, it decapsulates the packet and processes the request, finally return the result directly to the user according to its own routing table.

- The proxy servers can be continuously monitored using ICMP echo request-reply so that whenever a proxy server goes down, the Load Balancer is informed and the Proxy Address is removed from the current cluster.
- In case the Load Balancer crashes, the entire network would shutdown. To avoid this situation, mirroring of the Load Balancer can be provided so that the mirrored machine can be put to use in such a critical situation.

Reference

[1] Proxy Server FAQ

<http://www.connect.com.au/docs/proxy/faq.html>

[2] Proxy Hierarchy

www.rad.com/networks/1998/proxy/proxy.htm

[3] RFC 2186 - Internet Cache Protocol v2

[4] RFC 1945 - HTTP

[5] RFC 1853 - IP in IP Tunneling