CMPSCI 677 Operating Systems Spring 2015

Lecture 22: April 16

Lecturer: Prashant Shenoy *Scribe: Rose Tharail John*

1. WWW Principles

The world wide web is a conglomeration of several client server systems. The most common type of client we see is the web browser. The web browser sends a HTTP request to the web server and accesses content from it. The web server can access a database to get some data requested by the client and send its response to the client. Essentially a Request Response methodology is followed.

1.1 Clients

Browsers contain a User Interface which the user sees, a Browser engine which is used to construct requests to the server, a Rendering engine which is responsible for taking images, audio, HTML and other content and rendering it. A multi threaded or multi process architecture can be employed to implement an efficient browser.

1.2 Servers

Server can be a monolithic entity but it is often organized into multiple tiers handling different aspects of the HTTP request. It has a front end web tier, application tier and a database tier. Front end is a HTTP server (Eg: Apache Server). It is responsible for handling the incoming HTTP request. The second tier is where the application logic code is written. Applications can be written using Java (Servlets/J2EE), Python, PHP, Ruby on Rails. The last tier is the Database tier which stores the content and also any other data helpful in processing requests. If the request is for a static HTML page, the page is fetched from the file system of the server and returned as response. Otherwise, the HTTP requests sent to the front end will be dispatched to the application tier which will process the request with the help of the database tier. On processing the requesting it builds an HTML document and sends it back to the front end.

1.3 Client Proxy Server Architecture

These days instead of a simple Client Server Architecture most systems follow the Client Proxy Server Architecture. There exists a Web proxy between the Client and the Web Server. The client here will be sending the request to the Proxy server. The proxy can be set up to do many tasks depending on the functionality of the proxy.

Eg 1: Protocol Translation - Client sends a HTTP request to the Proxy. The proxy does protocol translation from the HTTP to an FTP request, fetches the data from the FTP server and sends the response back to the Client.

Eg 2: Caching - Proxies can cache the data of the Server within itself. When a request arrives, it will deliver the response from its own cache rather than going to the server. This way latency is reduced.
Eg 3: Content Translation - Proxies can be configured to do transcoding. If the response is a video file and is meant to be displayed on large screens but the client requesting the video has a smaller screen, the Proxy can be used to perform on the fly encoding of the file to be rendered on a smaller screen.

1.4 Apache Web server

It is the most popular Web server is Apache, which is estimated to be used to host approximately 70% of all Web sites. It is an open source web server. Originally it was a multi process architecture. Now it is multi threaded with a dynamic pool of threads which is adjusted depending on the number of requests coming in.

The server is internally organized as a set of modules. Thus the basic functionality of Apache can be extended by plugging in these modules. Apache can also handle module dependencies by letting a developer specify an ordering in which functions from different modules should be processed. The request processing pipeline then will follow this order of modules.

1.5 Scaling up Web servers

1.5.1 Web Server Clusters - I, invisible server replicas

To handle a large number of requests we can have a cluster of web servers to handle the requests. Each web server can have its own set of tiers for handling the processing. The server itself is replicated on multiple nodes on a LAN. There is a front end node which is visible to the Clients. The clients will send request to this Front end node. It is the front end node which performs load balancing and request dispatching to a server in the cluster. If the application is stateful, then the front end has to ensure that it sends the subsequent requests to the same server as the earlier requests. The response from these servers are sent back to the front end node, which will deliver it to the clients.

1.5.2 Web Server Clusters - II, visible server replicas

In this architecture, the request is sent to an intelligent switch. It will decide which server to send the request to. After the server processes the request the response is delivered directly to the client which sent the request. This way, we can ensure that the Switch does not become a bottleneck. All the subsequent communications are handled between the server and the client. The session is established between this server and the client. Thus the presence of multiple servers(Replication) is not transparent to the clients.

1.6 HTML and HTTP

1.6.1 HTML

The web browsers display HTML web pages. HTML is Hypertext Markup Language which allows you to encode different types of content using this mark up. It is the standard for web documents. The table given in the slides includes different kinds of objects like images, audio, video that can be included in a document. The documents are encoded with these objects . A web browser first fetches a web page and parses it to find the page contains any objects like images or multimedia. If it contains objects it has to fetch them . HTML5 has encoding for audio/video and no plug-ins are required.

1.6.2 HTTP

HTTP(Hypertext Transfer Protocol) is used to fetch the web documents. It runs over TCP and the HTTP requests and responses are used for communication between the browser and the web servers.

There are two HTTP versions

• HTTP 1.0 : (Non-persistent HTTP) For every object a new connection is started. Initially a TCP connection is established and the HTTP connection runs over it. This protocol fetches each object from the server on a new TCP connection and gives it to the browser. After the request, it tears down the connection. Thus each request will incur the over head of setting up and tearing down a TCP connection, which highly wasteful.

• HTTP 1.1 : (Persistent HTTP) Instead of creating multiple connections, a single TCP connection is established and multiple HTTP requests are sent over it. Once a response is received the connection is kept alive for a period of time to process any more requests. This is the most widely used form of HTTP nowadays.

1.6.3 HTTP Methods

There are 5 HTTP methods.

- Get method - is the most commonly used methods and is used to return the document from the client.
- Post method - is used in the case of forms to submit the information of forms to the server.
- Put - stores a document
- Delete - removes the document.

Arbitrary users are not generally allowed to remove content or upload files into a server unless a scenario where the file transfers are involved. Thus Put and the Delete methods are not commonly used.

- Head method returns the header of the document.

2. Web Services Fundamentals

Web services allow servers to make RPC calls to other servers using HTTP. The basic idea is that some client application can call upon the services as provided by a server application. Web services is now more than just a user-site interaction: sites could offer services to other sites. Interfaces are registered which can be looked up by other services.
Eg: Credit card authentication and charging. In an online store for payment the bank exposes the credit card operations as a web service. The online store can connect to this credit card service and charge the credit card for the purchase.

2.1 Protocols

Two Commonly used protocols are

• **Simple Object Access Protocol (SOAP)** : It is essentially RPC over HTTP. The slide gives an example of an XML based SOAP message. Here the name of the function called is alert control and a simple alert message is sent from one machine to another. The entire request is sent as a text XML document.

• **RESTful (Representative State Transfer) services** : Since SOAP is a heavy weight protocol, RESTful services are more commonly used. These are lightweight protocols and are used only for point to point XML communications. They use the HTTP's Get, Post, Put and Delete methods for reading, creating/updating and deleting the contents in the other server. They are much closer to the web and are much simpler when compared to RPC-style SOAP.
The slide shows an example for RESTful Service request where we are requesting for the Stock price of a company using a HTTP GET method. In RESTful services the arguments are a part of the URL whereas in SOAP they are a part of the message. The response coming back will be XML over HTTP. Thus you can consider RESTful web services to be a simplified form of SOAP.

2.2 Comparison between SOAP and RESTful services

• SOAP is transport agnostic- independent of its carrier whereas the RESTful services work only over HTTP.

• SOAP supports n party communication while RESTful services work only in case of point to point communication.

• RESTful services have no standard and the application designer can design it unlike SOAP which has standards and set of prebuilt tools to work on. Restful web services are simpler to use and easier to deploy. Thus they have become more popular.

3. Web Proxy Caching

Sites install a separate proxy server that handles all outgoing requests. Proxies subsequently cache incoming documents. The client-server system is modified to become a client-proxy-server system. Proxies cache the data from the server and the client can access the data from the local proxies instead of requesting the server.

3.1 Why use Web Caching?

- Latency - If the proxy is near by to the clients, than the server it can help in reducing the response latency.
- Load reduction - The proxies can handle some of the requests coming to the server. This will ensure that the server is not a bottle neck.

Content Distribution Networks are created mainly for these two reasons.

Cache hit: If the document requested by the client is present in the proxy then the document is returned to the client

Cache miss: If the requested document is not present in the proxy , the proxy requests the server for the document and on returns the obtained document to the client.

3.2 Consistency Issues

The major challenge faced by proxies is to provide consistent data to the client . The web pages at the proxy have to be updated over time to stay consistent with the servers. However, web pages have different update frequencies making cache consistency a complex issue.

Two methods are used to maintain consistency involved :

3.2.1 Invalidate/Update Messages

• **Invalidate message**: Invalidate messages can be sent from the server to the caches to mark the pages inconsistent. The proxy will discard the page in its cache and later if the page is requested it is fetched afresh from the server.

• **Update message**: The update message, updates the web page at the proxy. The proxy will replace the page in its cache with the new page. There is a overload involved in sending the new page to the cache and is preferred only if the page is popular.

3.2.2 Push, Pull and Lease Based Method

Push based approach:

In the push based approach the server tracks all the proxies that have requested the objects. The server is stateful. If the web page is modified then each of the proxies are notified by the Server. The notification can be an invalidate or an update message. The approach used is to send updates for frequently accessed objects and send invalidates to the less frequently used objects.

Advantages -

This techniques provides tight consistency and is very effective when the proxies are not intelligent. Proxies can be passive.

Disadvantages -

There is a lot of overhead involved in keeping track of all the proxies that have requested the objects and in case of a server crash all the information is lost. HTTP is a stateless protocol. So as to maintain state other mechanisms have to be followed. The state may need to be maintained indefinitely. The server has to keep track of every proxy and what pages each proxy requested. This state can keep growing and will become too huge to maintain and handle.

Pull based approach:

In the pull based approach the proxies are responsible for maintaining consistency. It uses a HTTP method called **conditional GET** which will include an IF clause in the GET method. For eg: GET <page> if modified since <timestamp>. They periodically poll the server to see if the object has changed. The frequency of polling however has to be determined. The server can assign a time-to-live(TTL) value to each resource it delivers to the Proxy. The proxy now needs to poll the server only after this expiration time. However there is no guarantee that the object will not be changed in the interim period. Thus it gives lesser consistency than the Push based approach.

One technique is to use Intelligent polling. In this technique the refresh rates are computed by the Proxy based on the past changes in the page. Starting from a conservative refresh interval the rates are modified based on the change between two successive polls. So this adaptive rate of poll will try to match the rate of polling to the rate of updates in the page.

Advantages -

Server can now be stateless. It can work even when there is a server or proxy failure.

Disadvantages -

Consistency guarantee provided is weaker. Sophisticated code needed to handle adaptive polling in proxies and there can be a higher rate of messages exchanged thus increasing the load on the server.

Lease based(Hybrid) approach:

A contract is agreed upon between the Server and the Proxy. Lease is nothing but a duration of time during which the server agrees to notify the Proxy of any modification. The first time a request is sent from the Client to the Proxy, the proxy sends the request for the page to server along with a Lease request. The server decides on how long the lease has to be provided and sends the response to the Client. During the duration of the lease it will follow a Push based approach. If the lease expires and no requests arrive for this page, then it turns the page to a Pull based page. Because it is not a very popular page. But if there are many requests for this page, then Lease can be renewed so as to always deliver fresh content for this popular page. If the Lease is of infinite duration, it becomes a pure Push based approach. If the Lease is of 0 duration, it becomes a pure pull based approach. For

deciding how long the Lease duration should be the server can follow any of the Age based, Renewal frequency based, Server load based approaches.

4. Cooperative caching

When the proxy receives a request, if there is a Cache miss, then rather than asking a Server for the page, it requests another proxy for this page. This means you first check your neighbors on a cache miss. Proxies are pooling their caches in order to look like a larger cache. If the proxies are nearby, and the associated server is far away, fetching it from a nearby proxy will reduce latency. Effectiveness of the caching mechanism becomes higher as the Cache hits become more. In hierarchical cooperative caching, the request is sent up the hierarchy to search for the page. If it is a cache miss in the entire hierarchy(all parents) only then send the request to the server. The protocol used is called **Internet Cache Protocol.** Disadvantage is a **global cache miss**. Here if the page is not found anywhere in the hierarchy many messages will have to be exchanged in order to ensure it is not available and then fetch the resource from the server. This will increase the overall latency. Solution: Keep an extra metadata in order to ensure that a hit/miss is restricted to only 2 hops. Each node will keep a table of what resources it caches and what its neighbors cache. For a request, if there is a hit, it will fetch the content from the proxy which has it(itself or neighbor). If it is a miss in this table, then the request is directly forwarded to the server instead of propagating it all the way up the hierarchy. This will increase the performance.

5. Content Delivery Networks

Network implementing a set of cooperative caches. A global network of caches are maintained by commercial services. Eg: Akamai. Typically the proxies are used to server large files - huge images, videos etc, which are popular so that the main server does not get overloaded. The request is forwarded to a nearby cache using the DNS service. The browser need not know which proxy exactly to contact. The DNS server of the CDN provider will provide you the IP address of the Proxy cache transparent to the browser. It is the CDN which will handle consistency in the background.

6. Replication of Web applications

Replication becomes more difficult when dealing with databases and such. No single best solution. Assumption made here is that updates are carried out at origin server, and propagated to edge servers. In a normal case, Web clients request data through an edge server, which, in tum, gets its information from the origin server associated with the specific Web site referred to by the client.
Other Alternative solutions are:-

- Full replication: high read/write ratio, often in combination with complex queries.
- Partial replication: high read/write ratio, but in combination with simple queries.
- Content-aware caching: Check for queries at local database, and subscribe for invalidations at the server. Works good with range queries and complex queries.
- Content-blind caching: Simply cache the result of previous queries. Works great with simple queries that address unique results (e.g., no range queries).