CMPSCI 677 Operating Systems Spring 2013 **Lecture 22: April 16**

Lecturer: Prashant Shenoy *Scribe: Supriya Jagannath*

1. WWW Principles

WWW is a standard client server System . The client is a web browser and accesses content from the web server . The web server can access a database to get some data requested by the client.

Server is not a single logical entity . It has a multi tier architecture . It has a front end web tier , application tier and a database tier . Front end is a HTTP server (Apache/IIS). The second tier is where the application code is written . Applications can be written using Java (Servelets/J2EE), Python,PHP,Ruby on Rails. The HTTP requests from the front end are processed by the application tier and it makes use of the database tier to fetch any data required. On processing the requesting it builds an HTML document and sends it back to the front end.

To handle a large number of requests each of these tiers are replicated . In replication , there is a front end node . The browsers connect to the front end node. There can be any number of replicas . The front end acts as a dispatcher and redirects the request to a replica.

In case of stateful services the use of multiple replicas can lead to problems . The switch therefore has to be stateful and send the requests from a given user to the same replica again. Hence the load balancing is on a session basis .

A detailed Schematic of the switch has been given in the slide. The first time a request comes in from a user the dispatcher decides which replica handles the users request . Once the mapping is done the distributor is informed about the user mapping and a state is established .

The dispatcher is load balancer written by the application developer and hence includes application logic whereas the switch is a standard request forwarding switch and if it is not programmable then the dispatcher and the switch exist as separate entities . If the switch is programmable then can coexist .

The load balancing can be done at a DNS level . If the IP address of the replica is available then the loading balancing can be done using a round robin DNS . If the application is geographically replicated then an intelligent DNS can be used however within a Data center a dispatcher is preferred . Content Distribution Network makes use of intelligent DNS for load balancing .

2. Web Services Fundamentals

The basic idea is that some client application can call upon the services as provided by a server application. Standardization takes place with respect to how those services are described such that they can be looked up by a client application. Web services is now more than just a user-site interaction: sites could offer services to other sites. Standardization is badly required.

3. Processes

3.1 Clients

Browsers form the Web's most important client-side sofware. One of the problems that Web browser designers have to face is that a browser should be easily extensible. Another client-side process that is often used is a Web proxy. Originally, such a process was used to allow a browser to handle application- level protocols other than HTTP.

For example, to transfer a file from an FTP server, the browser can issue an HTTP request to a local FTP proxy, which will then fetch the file and return it embedded as HTTP.

3.2 Apache Web server

The server is internally organized more or less according to the steps needed to process an HTTP request. Most popular Web server is Apache, which is estimated to be used to host approximately 70% of all Web sites. Apache can also handle module dependencies by letting a developer specify an ordering in which functions from different modules should be processed

3.3 Web Server Clusters

To improve performance and availability, WWW servers are often clustered in a way that is transparent to clients. The front end may easily get overloaded, so that special measures need to be taken. Transport-layer switching: Front end simply passes the TCP request to one of the servers, taking some performance metric into account. Content-aware distribution: Front end reads the content of the HTTP request and then selects the best server.

4. HTML and Web Documents

The web browsers display HTML web pages . 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 .

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 :** For every object a new connection is started. Initally a TCP connection is established and the HTTP connection runs over it .This protocol fetchs the object from the server gives it to the browser and tear down the connection. It is called *Non persistent HTTP*
- **HTTP 1.1:** This is called *Persistent Connection* Instead of many connections , a single TCP connection is established and multiple HTTP requests are sent over it.

In case of HTTP 1.0 there is no need for cleanup however the overhead involved in establishing a TCP connection makes it less efficient.

5. Communication

Communication in the Web is generally based on HTTP; a relatively simple client-server transfer protocol having the following request messages. When two machines communicate with each other without an intervention of an user they make use of web service protocols. Each of the machine exposes an API over HTTP . One machine communicates with another using a RPC call.
Some of the examples include the search box in the browser . In this case the search query is sent as a web service request . A dropbox client on the users machine communicates with the dropbox server using web services.

5.1 HTTP methods

There are 5 HTTP methods . The Get method is the most commonly used methods and is used to return the document from the client. The Post method is used in the case of forms . The put and the delete methods are not commonly used. Put stores a document and the delete removes the document. Arbitrary users are not generally allowed to remove content unless a scenario where the file transfers are involved . The head method returns the header of the document

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.
- **RESTful services:** Since SOAP is a heavy weight protocol , RESTful services are more commonly used . These are lightweight protocols and are used ony for point to point XML communications . They have the Get, post, put and the Delete methods . They are much closer the web and are much simpler when compared to RPC-style SOAP

The slide shows an example for RESTful Service request and response and the corresponding SOAP messages .The SOAP message is more complex when compared to the RESTful service. In RESTful services the arguements are a part of the URL whereas in SOAP they are a part of the message making RESTful simple and effective.

5.2 Comparison between SOAP and RESTful services

- SOAP is transport agnostic- independent of it's 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

6. 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.
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.
Cooperative caching- By which you first check your neighbors on a cache miss

Cache-consistency protocols:

- Always verify validity by contacting server
- Age-based consistency:

  $T_{expire} = \alpha \cdot (T_{cached} - T_{last\ modified}) + T_{cached}$

6.1 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 differing update frequencies making cache consistency complex .

Two messages involved :

- Invalidate message: Invalidate messages can be sent from the server to the caches to mark the pages inconstent .
- Update message: The update message , updates the web page at the cache . There is a overload involved in sending the new page to the cache and is preferred only if the page is popular.

6.2 Push based approach

In the push based approach the server tracks all the proxies that have requested the objects . If the web page is modified then each of the proxies are notified. The notification can be an invalidate or an update message. The approach used is to send updates for frequent objects and send invalidates to the less frequent used objects . This techniques provides tight consistency and is very effective when the proxies are not intelligent . However , 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 .

6.3 Pull based approach

In the pull based approach the proxies are responsible for maintaining consistency . They periodically poll the server to see if the object has changed. The frequency of polling however has to be determined . The time-to-live(TTL) assigned by the server gives a hint however there is no guarantee that the object will not be changed in the interim One technique is to use Intelligent polling. In this technique the refresh rates are computed based on the past observations . Starting from a conservative refresh interval the rates are modified based on the change between two successive polls. This is an adaptive approach and require

7. Replication of Web applications

Replication becomes more difficult when dealing with databses 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).