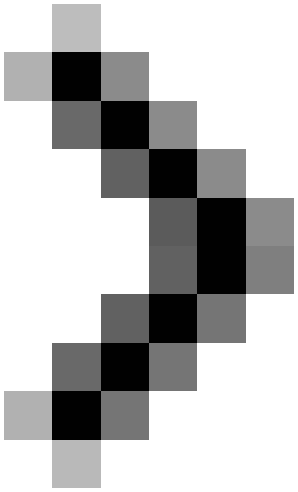
Load balancing in Web applications (end part)

The previous section we studied the load balancing method based on DNS round robin. However, this method can balance a number of users on each server, but it does not completely balance server load. Therefore this section we will consider load balancing d

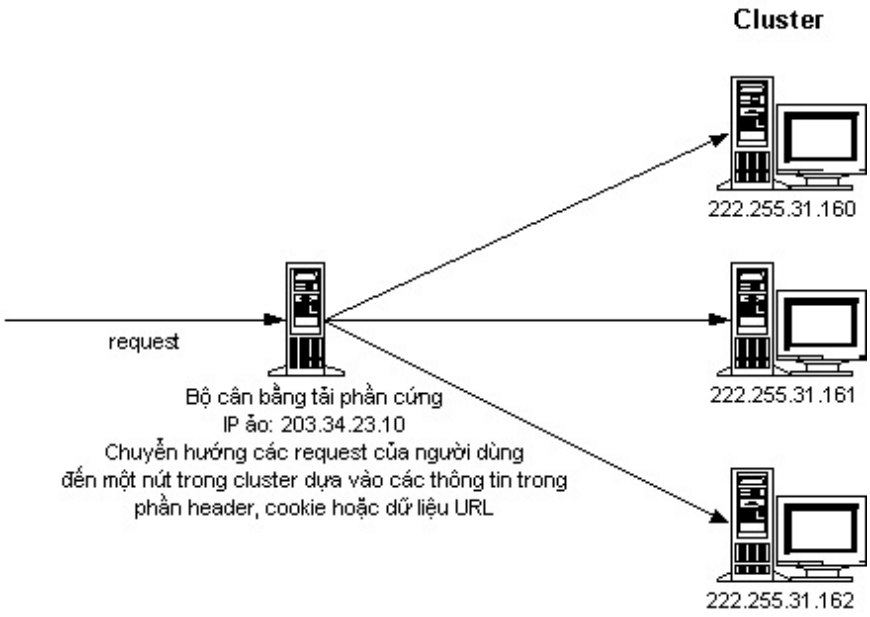


Part I: DNS round rotation

Vivek Viswanathan

Hardware-based load balancing

Hardware load balancers solve many of the problems that we have just faced in the above DNS round robin software method through virtual IP addresses. The load balancer will display a virtual IP address for the external network, which maps to the addresses of each machine in a cluster. Therefore, this conveyor belt needs to provide an IP address of all the computers in the cluster to the outside world.



When a request arrives at the load balancer, it will record the request header to point to the other machines in the cluster. If a machine is removed from the cluster, the request will not run at risk of 'hit' on this dead server, since all other servers in the cluster will appear to have the same IP address. This address remains the same even if a node in the cluster fails. When a response is returned, the client will see the response coming from the hardware load balancer. In other words, the client will deal with a computer that is a hardware equalizer.

Advantages

• **Relationship between servers** . The hardware load balancer reads cookies or URLs are being read on each request by the client. Based on this information, it can record the header information and send the request to the appropriate node in the cluster, where its session is maintained.

These load balancers can provide server-to-server relationships in HTTP communications, but not through secure channels like HTTPS. In secure channels, messages are SSL encrypted and can avoid load balancing from reading session information.

• **High availability via automatic failover system** . Failover occurs when a node in the cluster cannot process a request and redirect it to another node. There are two types of failover:

1. *Failover level required* . When a node in the cluster cannot process a request (usually because of a failure), it will move this request to another node.
2. *Transparent session redundancy* . When a call fails, it will be routed smoothly to another node in the cluster to complete execution.

This type of equalizer provides request level failover; ie when it detects that a node has a problem, this equalizer will redirect all subsequent requests sent to this node to another active node in the cluster. However, any session information on the dead node will be lost when requests are redirected to a new node.

Transparent session failover requires some knowledge of execution for a process in a node, because the hardware load balancer can only detect network level problems, no errors. To execute transparently on failover, nodes in the cluster must combine with other nodes and have shared memory or shared databases to store all

session data. Also, if a node in the cluster has a problem, a session can continue in another node.

• **Metrics** . Since all requests to a web application must go through the load balancing system, the system can determine the number of active sessions, the number of active sessions connected in different cases, the time periods. Response time, maximum voltage time, the number of sessions throughout the maximum voltage range, the number of sessions throughout the minimum voltage range . All this test information is used to refine the entire system. System to optimize performance.

Defect

The downside to hardware routing is its cost, the complexity of the setup issue, and the possibility of a problem in the future. Because all requests are routed through a hardware load balancer, the hardware requirement error affects the entire site.

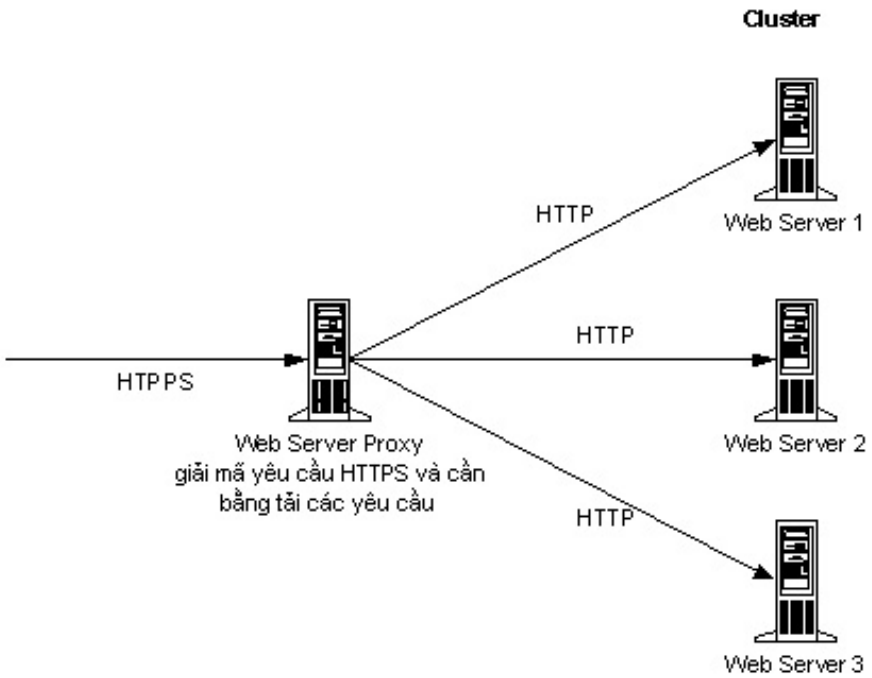
Balancing HTTPS requests

As mentioned above, it is very difficult to load balance and maintain session information of requests using HTTPS protocol, since they are encrypted. The hardware load balancer cannot forward requests based on information in the header, cookie or URL. There are two ways to solve this problem:

1. Proxy of Web Server
2. Hardware SSL decoder

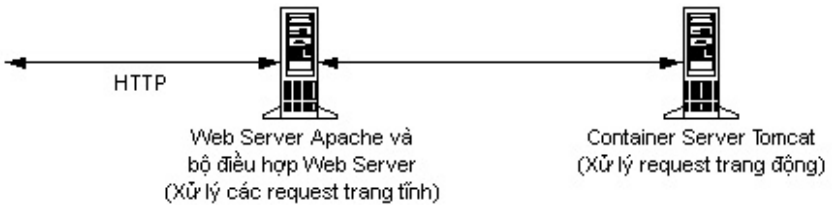
Additional Web Server proxies

A Web server proxy is in front of a cluster of Web servers taking all requests and encrypting them. It then redirects them to an appropriate node, based on header information in the header, cookie and URL.

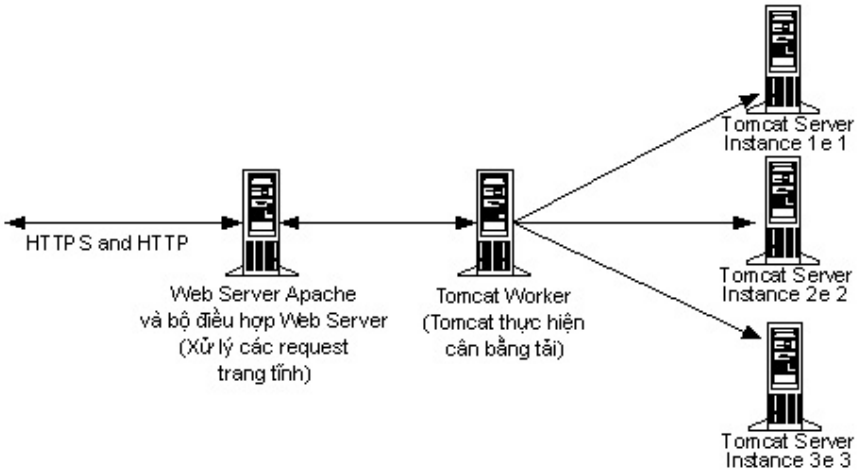


The advantage of Web server proxies is that they provide a way to get server-to-server communications for SSL-encrypted messages, without any extra hardware. But the extended SSL process requires an extended load on the proxy.

Apache and Tomcat . In many server systems, Apache and Tomcat servers work together to manage all HTTP requests for dynamic pages (JSSP or servlets). Tomcat servers also manage static pages, but in coordinated systems, they are usually set up to manage dynamic requests.



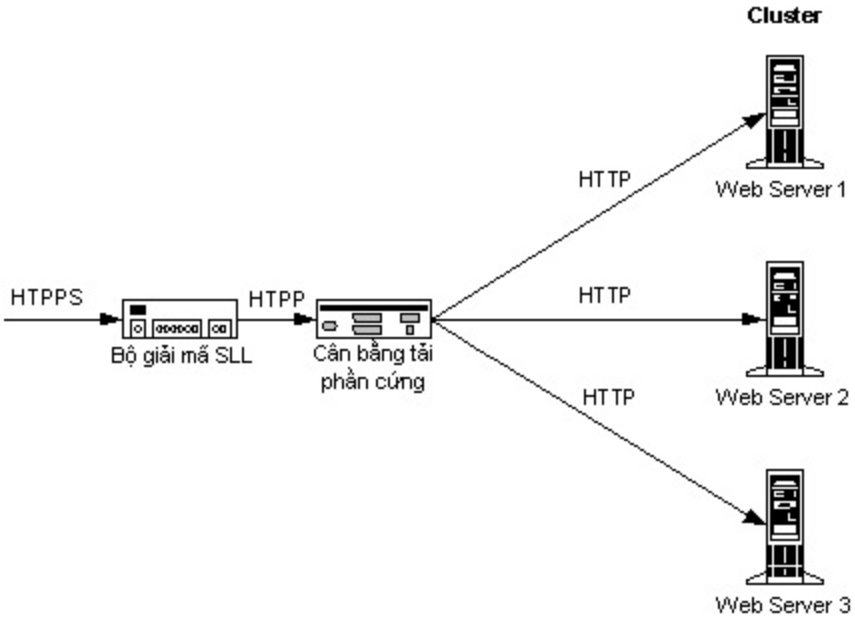
You can also configure Apache and Tomcat to control HTTPS requests and in load balances. To do this, run the multi-Tomcat server instance on one or more machines. If all Tomcat servers are running on one machine, they should be configured to listen on different ports. To execute load balancing, you create a special Tomcat template called Tomcat Worker.



As you can see on the image above, the Apache Web server receives HTTP and HTTPS requests from clients. If the request is HTTPS, then the Apache Web server encrypts the request and sends it to the Web server adapter, the server sends the request to Tomcat Worker, including the load balancing algorithm. Similar to the Web server proxy, this algorithm also balances the load between machines like Tomcat.

Hardware SSL decoder

Finally, we should mention here that there are some hardware devices that can decode SSL requests. Detailed instructions about them are not covered in this article, but can be briefly stated, they are placed in front of the hardware load balancer and allow decoding information in cookies, headers and URLs.



These hardware SSL decoders often operate faster than Web server proxies and are highly scalable. However, as most hardware solutions, they still cause many complex problems in the setup and configuration.