Best Web Server

Luke Kennedy and Andrew Hopkins

# Added Features

* Changed provider to “Luke Kennedy and Andrew Hopkins”
* Added the 304 Not Modified Response
* Added the 403 Forbidden Response
* Added the 501 Not Implemented Response
* Added the 503 Service Not Available Response
* Added the 505 Not Supported Response
* Added in-memory caching of files requested by clients
* Limited connections
* Added blacklisting by IP address

# Availability

Our strategy for increasing availability was to decrease the amount of time needed to service each request. Our speed, and therefore availability, was increased by using a memory cache.

Limiting connections increases the availability by preventing service from degrading with a high number of requests. This also prevents the server from going down in the event of a DoS attack, leading to more uptime.

# Performance

Our server’s performance increases mainly stem from the implementation of an in-memory file cache. While this dramatically slows down the response for the first request, each subsequent does not require a hard disk access, which allows it to respond quicker.

In order to prevent performance from degrading, we limited the number of concurrent connections to the server. This allows us to ensure that too many connections will not slow down the response time.

If a request is made for a file in which the client has a local cached copy of the most recent version, our server will reply with a 304 Not Modified response. This improves performance by not sending extraneous data in the response.

# Security

By limiting connections, our server is able to thwart denial of service attacks by only allowing a limited number of requests. This does not allow a DoS attack to take down the server. We made the decision that rejecting a few legitimate requests while accepting DoS requests is the best solution if it prevents the server from completely crashing.

Our server utilizes a blacklist to reject connections from blocked IP addresses. This prevents unwanted users from using server resources and starting attacks.

# Improvement

To measure the improvements to performance on our server, we ran the thread-pooled DoS program against both our modified server and the original SimpleWebServer for roughly five minutes and recorded the Service Rate for each instance. Our modified server had a Service Rate of 11.392, while the unmodified server had a rate of 1.604. This shows an over 710% improvement in Service Rate.