HTTP persistent connection (Connection Pooling)

HTTP persistent connection, also called HTTP keep-alive, or HTTP connection reuse, is the idea of using a single TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair. The newer HTTP/2 protocol uses the same idea and takes it further to allow multiple concurrent requests/responses to be multiplexed over a single connection.

Advantages

- Reduced latency in subsequent requests (no handshaking and no slow start).
- Reduced CPU usage and round-trips because of fewer new connections and TLS handshakes.
- Enables HTTP pipelining of requests and responses.
- Reduced network congestion (fewer TCP connections).
- Errors can be reported without the penalty of closing the TCP connection.

According to RFC 7230, section 6.4, "a client ought to limit the number of simultaneous open connections that it maintains to a given server". The previous version of the HTTP/1.1 specification stated specific maximum values but in the words of RFC 7230 "this was found to be impractical for many applications... instead... be conservative when opening multiple connections". These guidelines are intended to improve HTTP response times and avoid congestion. If HTTP pipelining is correctly implemented, there is no performance benefit to be gained from additional connections, while additional connections may cause issues with congestion.[14]

Disadvantages

If the client does not close the connection when all of the data it needs has been received, the resources needed to keep the connection open on the server will be unavailable for other clients. How much this affects the server's availability and how long the resources are unavailable depend on the server's architecture and configuration.

Also a race condition can occur where the client sends a request to the server at the same time that the server closes the TCP connection.[15] A server should send a 408 Request Timeout status code to the client immediately before closing the connection. When a client receives the 408 status code, after having sent the request, it may open a new connection to the server and re-send the request.[16] Not all clients will re-send the request, and many that do will only do so if the request has an idempotent HTTP method.

NOTE

Python's requests library contains requests.Session(), which establishes a persistent HTTP connection, thereby allowing the underlying TCP connection to be reused, which can result in a significant performance increase.

Transmission Control Protocol

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP, which is part of the Transport Layer of the TCP/IP suite. SSL/TLS often runs on top of TCP.

TCP is connection-oriented, and a connection between client and server is established before data can be sent. The server must be listening (passive open) for connection requests from clients before a connection is established. Three-way handshake (active open), retransmission, and error detection adds to reliability but lengthens latency. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP) instead, which provides a connectionless datagram service that prioritizes time over reliability. TCP employs network congestion avoidance. However, there are vulnerabilities in TCP, including denial of service, connection hijacking, TCP veto, and reset attack.

References

https://en.wikipedia.org/wiki/HTTP_persistent_connection

https://en.wikipedia.org/wiki/Transmission_Control_Protocol