Web-Server: KeepAlive - Ein Mysterium?

Frage:

Wenn man bei Google sucht oder in Foren fragt, findet man tausende von verschiedenen Erklärungen und Ansätzen und Theorien über die Apache-Einstellungen `KeepAlive`, `KeepAliveTimeout` und `MaxKeepAliveRequests`.

Viele Meinungen dazu sind absoluter Humbug und an den Haaren herbei gezogen. Z.B.:

- "Bei vielen Websites auf einem Server sollte `KeepAlive Off` gesetzt werden."
- "KeepAlive ist für AJAX gut aber braucht dann einen hohen `KeepAliveTimeout`."
- "Der `KeepAliveTimeout` gilt als maximale Zeit der Datenübertragung."
- "Stelle `KeepAliveTimeout` auf einen möglichst hohen Wert."
- "Sehr hohe `MaxKeepAliveRequests` sind gut."
- "Sehr niedrige `MaxKeepAliveRequests` sind gut."

Diese Aussagen sind alle totaler Quatsch!

Hier nun die Wahrheit über den Mythos *KeepAlive*:

KeepAlive erklärt:

KeepAlive ist lediglich für das Laden einer Page gedacht.

Definition von "Page": eine HTML-Seite (statisch oder dynamisch generiert) mit all seinen Inhalten (CSS, JavaScript, Bilder, etc.).

Funktionsweise:

Der TCP-Handshake um eine Connection aufzubauen dauert eine kleine Zeit. Diese Zeit wird gespart wenn eine bestehende Connection für einen zweiten, dritten, vierten, ... Request genutzt wird.

Im ersten Request wird die HTML-Seite selbst geladen. Die Browser bauen ihren DOM-Baum bereits während dem Laden auf und erkennen weitere zu ladende Dateien (CSS, Bilder, JavaScript, etc.). Je nach Konfiguration startet der Browser nun eine gewisse Anzahl an Connections (bei Firefox lauten die std. Einstellung network.http.max-connection-per-server 15 (für `KeepAlive Off` und network.http.max-persistent-connection-per-server 6 (bei aktivierten KeepAlive) und versucht diese Dateien - auf diese Anzahl von Connections verteilt - nach zu laden.

Solange die Seite selbst nur aus 6 Elementen (HTML + 5 Bilder/CSS) besteht, wird KeepAlive gar nicht zum Zuge kommen. Aber bei mehr als 6 Request werden diese Connections dank KeepAlive aufrecht erhalten und eine Datei nach der Anderen nachgeladen.

Man kann davon ausgehen, dass der Browser bereits ausreichend viele Elemente kennt, während die HTML-Seite weiter geladen wird und nicht einmal die 2 Sekunden Timeout bräuchte. Denn er feuert einen Request nach dem Anderen durch die offenen Connections.

Warum aber überhaupt ein KeepAliveTimeout?

Der KeepAlive wird vom Server dem Browser angeboten. Ob der Browser davon Gebrauch macht, und wenn ja, wie oft, kann der Server nicht von sich aus erkennen. Ergo wartet er einfach diesen Timeout ab und schließt dann von sich aus die Connection.

Was ist das Gefährliche am `KeepAliveTimeout`?

Ein hoher Timeout (std. Einstellung 15 Sekunden) kann von Bösewichten ausgenutzt werden um alle vorhandenen Apache-Threads an sich zu binden ohne Requests zu schicken. Eine simple, aber oft effektive Art eines DDoS-Angriffs.

Ein kurzer Timeout (1-2 Sekunden) gibt in diesem Falle die ersten Connections schon wieder frei bevor der letzte Apache-Thread vereinnahmt ist.

An dieser Stelle sei auch die Verringerung des allgemeinen `TimeOut` für leerstehende Verbindungen erwähnt. Dieser ist mit der Standard-Einstellung von 300 Sekunden (5 Minuten) viel zu hoch. 10 bis 15 Sekunden reichen hier ebenfalls vollkommen aus.

Was passiert bei "`KeepAlive Off`"?

Der Server bietet keinen KeepAlive an uns schließt jede Connection nach seinem abgeschlossenen Response. Der Browser muss also für jede Datei eine neue Verbindung aufbauen. Dadurch erhöht sich z.B. beim Firefox die Anzahl der Verbindungen von 6 auf 15 (s.o.). Bei umfangreichen Seiten auf einem unausgelastete Server erhöht sich die Ladezeit nur minimal.

Nachteil bei ausgelasteten Servern: Hier kann es passieren, dass jeder Request erst in der Queue

landet und wartet bis ein Apache-Thread frei ist. (Dies ist bei eingeschalteten KeepAlive nicht der Fall!) und die gesamte Page (siehe o.g. Definition) lädt deutlich langsamer

Ebenfalls bei schlecht konfigurierten Servern: der Server muss intern einiges an Arbeit verrichten: Connection aufbauen, Apache-Childs generieren, uvm.

Welcher Wert ist für `MaxKeepAliveRequests` optimal?

Hier kann getrost die Vorgabe von `100` stehen bleiben. Ein zu niedriger Wert könnte den ganzen KeepAlive-Effekt zerstören. Ein zu hoher Wert schadet aber nicht.

Eindeutige ID: #1410
huschi
2011-05-31 11:22