Apache optimization: KeepAlive On or Off?

Source : http://abdussamad.com/archives/169-Apache-optimization:-KeepAlive-On-or-Off.html

Apache is the most widely used web server on the Internet. Knowing how to get the most out of Apache is very important for a systems administrator. Optimizing Apache is always a balancing act. It's a case of sacrificing one resource in order to obtain savings in another.

What is KeepAlive?

HTTP is a session less protocol. A connection is made to transfer a single file and closed once the transfer is complete. This keeps things simple but it's not very efficient.

To improve efficiency something called KeepAlive was introduced. With KeepAlive the web browser and the web server agree to reuse the same connection to transfer multiple files.

Advantages of KeepAlive

- **Improves website speed**: It reduces latency associated with HTTP transfers and delivers a better user experience.
- **Reduces CPU usage**: On the server side enabling KeepAlive reduces CPU usage. Consider that a typical web page has dozens of different files such as images, stylesheets, javascript files etc. If KeepAlive is disabled a separate connection must be made for each of those files. Creating and closing connections has an overhead and doing it for every single file wastes CPU time.

Disadvantages of Keepalive

- **Increases memory usage**: Enabling KeepAlive increases memory usage on the server. Apache processes have to keep connections open waiting for new requests from established connections. While they are waiting they are occupying RAM that could be used to service other clients. If you turn off KeepAlive fewer apache processes will remain active. This will lower memory usage and allow Apache to serve more users.

When should you enable KeepAlive?

Deciding whether to enable KeepAlive or not depends on a number of different factors:

- **Server resources**: How much RAM vs. how much CPU power you have? RAM is often the biggest limiting factor in a webserver. If you have little RAM you should turn off KeepAlive because having Apache processes hanging around while they wait for more requests from persistent connections is a waste of precious memory. If CPU power is limited then you want KeepAlive on because it reduces CPU load.
- **Types of sites**: If you have pages with a lot of images or other files linked into them, KeepAlive will improve the user experience significantly. This is because a single connection will be used to transfer multiple files.

- **Traffic patterns**: The type of traffic you get. If your web traffic is spread out evenly throughout a day then you should turn on KeepAlive. OTOH, if you have bursty traffic where a lot of concurrent users access your sites during a short time period KeepAlive will cause your RAM usage to skyrocket so you should turn it off.

Configure Apache KeepAlive settings

Open up apache's configuration file and look for the following settings. On CentOS this file is called httpd.conf and is located in /etc/httpd/conf. The following settings are noteworthy:

- **KeepAlive:** Switches KeepAlive on or off. Put in "KeepAlive on" to turn it on and "KeepAlive off" to turn it off.
- **MaxKeepAliveRequests**: The maximum number of requests a single persistent connection will service (0 = no limit). A number between 50 and 75 would be plenty.
- **KeepAliveTimeout:** How long should the server wait for new requests from connected clients (= next request from the same client on the same connection). The default is 15 seconds which is way too high. Set it to between 1 and 5 seconds to avoid having processes wasting RAM while waiting for requests.

Other settings

KeepAlive affects other settings in your Apache configuration file even though they are not directly related. Here are the settings in an Apache prefork MPM webserver:

- **MaxClients**: MaxClients is the maximum number of child processes launched by Apache to service incoming requests. With KeepAlive enabled you have will have a higher number of child processes active during peak times. So your MaxClients value may have to be increased.
- **MaxRequestsPerChild**: The number of requests a child process will serve before it is killed and recreated. This is done to prevent memory leaks. When KeepAlive is turned on each persistent connection will count as one request. That effectively turns MaxRequestsPerChild into a maximum connections per child value. As a result you can set a lower MaxRequestsPerChild value if you allow KeepAlive. If you don't allow KeepAlive you should increase the MaxRequestsPerChild value to prevent excessive CPU usage.

Final thoughts

There is no one universal solution to tuning Apache. It all depends on the resources at your disposal and the type of sites you have. When used properly KeepAlive can improve the user experience at minimal cost in terms of server resources. But it can also be a liability when you are faced with a large number of concurrent users.