

TexasChromeshop.com Performance Audit Report

I've completed the full performance audit of the Texas Chrome Shop website. Below is a breakdown of the issues I found, along with some suggestions to resolve them and improve the overall performance and stability of the site.

1) Server Stack Bottleneck (Primary Issue)

The site is currently hosted on an Apache + PHP-FPM environment, which is a poor fit for high-performance and plugin-heavy WooCommerce setups. Apache tends to consume more server resources, slowly processes dynamic requests, and doesn't handle concurrency as efficiently as NGINX. This is the #1 contributor to slow page loads and backend delays.

- Average page load time reaches **up to 30 seconds**
- Saving or updating products/pages often takes **30–40 seconds**, which is unacceptable for a live store

2) Outdated PHP Version

The website is currently running on **PHP 7.4**, which is now outdated. I understand the codebase currently relies on PHP 7.4 and may not yet be compatible with PHP 8.x, but sticking with 7.4 long-term will limit both performance and security.

For now, we can continue working with PHP 7.4, but I recommend preparing the codebase to support **PHP 8.1+** in the future.

3) Misconfigured Redirects

In .htaccess file, there is no rule to redirect <http://www.texaschromeshop.com> to <https://www.texaschromeshop.com>. So we get an SSL error when visiting the website.

4) Object Cache Not Working

The object cache appears to be enabled on the site, but it's actually not connected to Redis. WordPress detects the drop-in caching file (`object-cache.php`) and assumes caching is working, but Redis is not installed on the server, so the system is falling back to inefficient PHP-based in-memory caching. Redis caching must be properly installed and configured on the server to benefit website performance.

5) WordPress Memory Limit Is Too Low

The current WordPress memory limit is 256MB, which is below the recommended value for WooCommerce sites. Ideally, this should be increased to at least 512MB to ensure smoother performance, especially in the admin area and on pages with lots of product variations or heavy plugins.

6) Broken Asset URLs & Mixed Content

I found several instances of assets trying to load from a development domain (`texaschromeshop.vnddev.com`). For example, the favicon is attempting to load from a non-existent path:

```
https://texaschromeshop.vnddev.com/wp-content/themes/porto/images/logo/favicon.png
```

These broken references are not only causing browser security warnings but also affecting how certain resources are rendered on the live site.

Additionally, there are multiple database entries that still contain the development domain (`texaschromeshop.vnddev.com`), causing mixed content issues and redirect loops. These references should be carefully searched and replaced across the database.

7) Missing PHP Modules

Some important PHP extensions are also missing, such as `imagick` for image handling and `php-redis` for proper cache integration. Additionally, OPcache which is critical for speeding up PHP execution is not working at all.

Installing and configuring these modules correctly will improve both backend and frontend performance.

8) Redundant/Heavy Plugins

Jetpack is installed but disconnected from WordPress.com, and none of its features are in use. It's a heavy plugin and should be removed to improve performance.

In addition, plugins like LC Scripts Optimizer, Smush, WP Deferred JavaScript, and WP-Optimize can be replaced with a single, more efficient premium plugin: **WP Rocket**, which handles caching, minification, deferred JS, and lazy loading all in one place, and much more efficiently.

9) Unoptimized PHP.ini

The current `php.ini` file on the server is using default, unoptimized values. Key settings like `memory_limit`, `max_execution_time`, and OPcache parameters need to be tuned for production use.

10) My Recommendation: Switch to LEMP Stack

To truly resolve the performance issues and give the site a more stable and scalable foundation, we must migrate the site to LEMP stack, which uses Nginx+PHP-FPM-FastCGI+OpCache+MariaDB.

This setup is significantly faster, more resource-efficient, and better suited to WooCommerce stores. It handles static files more efficiently, supports modern caching strategies like Redis and FastCGI, and works better with PHP optimizations like OPcache.

If a staging environment is available, I strongly suggest that we first migrate the site to a LEMP-based staging server. Once all settings (SSL, caching, PHP config) are tested and verified, we can mirror those changes to the live site with minimal risk.

If no staging environment is available, it's still possible to migrate safely to the live environment by running NGINX alongside Apache temporarily. Once the NGINX setup is fully tested in the background, we can switch over with minimal downtime.

11) Immediate Fixes to Implement

While we prepare for the full migration and optimization, there are a few important fixes I recommend applying immediately to improve user experience.

- **Fix misconfigured redirect**
Set up a redirect from <http://www.texaschromeshop.com> to <https://www.texaschromeshop.com> to eliminate SSL warnings and ensure consistent traffic handling.
- **Remove unused or heavy plugins**
Uninstall Jetpack and consolidate multiple optimization plugins using WP Rocket for better performance and fewer conflicts.
- **Fix broken asset links and remove references to the development domain**
Clean up all references to [texaschromeshop.vnddev.com](https://www.texaschromeshop.com) in the database to prevent mixed content and broken resources.
- **Increase WordPress memory limit and max execution time**
This will improve backend performance, especially during product editing or plugin use.
- **Install and configure Object Cache**
This will immediately reduce the load on the database and speed up queries.