

Wordpress on HipHop

White Paper – October 2010

When we started out thinking about TBB (Threading Building Blocks) and its various implementations in script languages, PHP was a strong contender from day one.

PHP is very widely used, and some projects that started out as small pet projects on the side grew to massive sites that have to deal with all the usual scalability problems. Some of them decided to rewrite the whole platform in a new language that scales better -Amazon is one of them; others stick to the language, but try to optimize the performance.

To optimize the performance of PHP there are a number of screws that can be adjusted and the real execution performance of PHP is often only a minor concern. Database performance was a major problem for Yahoo! for a while. The improved art of caching is becoming more and more relevant, with Memcached, Varnish and Redis three very strong products complementing the older version of running Squid in front of the web server farm.

PHP execution speed has for a long time been an area of tuning as well, and the Zend optimizer was for a long time pretty much the only way to speed things up. The Zend optimizer is commercial and does not have a deep market penetration as far as we can tell. The other version of running PHP faster was phc, a PHP compiler that generated an optimized binary. Phc does not have a deep market penetration either and the project is not the most active.

Facebook is one of the largest shops running on a PHP stack in the moment and they were trying to speed up the execution time of PHP with a project called HipHop.

HipHop is Facebook's answer to the lack of high performance optimizers for PHP and comes in two versions. The HipHop engine can either statically compile the entire PHP project to a binary that can be run with a HipHop wrapper, or run in a just-in-time compiler mode. The JIT mode is especially handy for development and debugging and is the mode we used for most of our development and testing.

HipHop takes a PHP source tree and compiles it first to optimized c++ and then compiles c++ to a binary that runs on Linux and some UNIX versions. It uses the GNU tool chain for compilation and makes use of TBB for memory management in the moment only.

As our focus was to improve the execution speed of a well-known PHP project, we picked WordPress and ported it over to HipHop.

Our initial tests looked very promising and after further testing the community started patching WordPress as well and we saw patches in the WordPress root that made it possible to compile it on HipHop. As the WordPress team was moving forward new incompatibilities were introduced, but the support for up to date patch sets is improving on a daily basis.

We started measuring performance on a traditional Apache based LAMP stack and compared it to a HipHop based stack. Our development machines were standard Core 2 Duo machines running virtual box with the Ubuntu 10.04 LTS release and either Apache or HipHop and a second VM with our test code. We ran all the tests from the same box between VMs to reduce network latency and then on different physical machines to measure the impact of network overhead and reduce the impact of our monitoring. The network was standard wireless with 54MBit and no other significant traffic than the two VMs communicating with each other.

The test code was using Tsung and generated a Digg style attack with a max of 250 concurrent connections. The emitting side was timed to a total of 70 seconds starting with an interval of 1 second for new users then dropping down to an interval of a quarter of a second and then rising to a second again. Tsung is written in Erlang and can handle lots of concurrent connections in a very

efficient way. In fact the impact of running the test code on the same physical machine was not as significant as we had expected, and the results via network were not too different to the ones run locally CPU and memory wise.

The problem with our test set-up was the lack of big enough machines to complete the Apache test entirely to the end. The development VMs ran on 1GB of memory and ran out of memory and started swapping pretty much instantly after starting the test. Even running Apache on a dedicated machine with 2GB of ram the attack maxed out the box pretty fast and the tests were cancelled after about 6 minutes as the box running Apache was not responding anymore.

The results we saw from running Apache was that initially the connection setup is really fast, but then the amount of resources Apache needs to process PHP are stacking up too fast and in the end the server swaps to death.

The next test was to run WordPress on HipHop and we had success with our tests. The time to complete the entire test was 4 minutes 20 seconds and we had at all times memory left on the server. The limiting factor here was the CPU, as far as we can tell.

The results are pretty obvious. Apache cannot be as optimized as a compiled binary version of the PHP project and we expected the HipHop version to run faster than the Apache version. The interesting bit was how fast Apache ran out of resources (which was after about the 100th connection) and how badly it ground to halt. The VM had to be killed after each test run and the testing code was not able to run through an entire test.

HipHop is a considerable improvement for running PHP projects and we can confirm a significantly reduced memory footprint. The problem seems to be CPU-bound now and we will address this in the next test.

© **Open Parallel** White Paper
October 2010

www.OpenParallel.com

Correspondence: Nicolas.Erdody@OpenParallel.com