HHVM

general concepts and operations

What is HHVM?

A virtual machine that is able to run PHP code and is almost 100% compatible with PHP 5.x (for some value of x > 3). It features JIT compilation of bytecode.

It is also a multithreaded FastCGI server.

Command line

```
$ hhvm --php some_evil_script.php
```

- Slow compared to traditional php
- “php” on the CLI will point to hhvm, via an installed alternative

$ php some_evil_script.php

- Uses /etc/hhvm/php.ini as a config file

Server mode

```
$ hhvm -m server --config /etc/hhvm/fcgi.ini
```

Via init:

```
$ sudo service hhvm start
```

HHVM in server mode only speaks FastCGI, by default it listens on port 9000; It can also listen on a UNIX socket.

The FastCGI server

Main process binds to socket/port and spawns some threads.

It starts an event loop that dispatches request to said workers and fetches the responses (think of a sane version of what we do with parsoid).

More are spawned on request, to a max of `hhvm.server.thread_count`

The FastCGI server

If all threads are busy, requests are queued.

If the queue becomes long, requests wait forever, and apache will soon-ish reach MaxClients.

(Do “HHVM busy threads” and “HHVM queue” alerts make more sense now?)

How HHVM works

All PHP code gets compiled to bytecode and stored in the bytecode repo

```
hhvm.repo.central.path =
/run/hhvm/cache/fcgi.hhbc.sq3
```

Yes, it’s a sqlite3 file. You can take a peek at its structure if you feel like that.

How HHVM Works

Execution of code is accelerated thanks to use of a jit compiler

```
hhvm.jit = true
hhvm.jit_a_cold_size = 34603008.0
hhvm.jit_a_frozen_size = 104857600
hhvm.jit_a_size = 104857600
```

How HHVM Works

The JIT compiling means the first requests take a *serious* penalty (and jit is disabled for the first N requests). “HHVM is slow upon startup”

If any of the caches is exhausted, HHVM will simply refuse to work anymore. Luckily it’s mostly easy to size them.

The admin server

Exposed via HTTP on localhost on port 9002

```
$ curl localhost:9002/
$ curl localhost:9002/check-health
{
 "load":0, "queued":0, "hhbc-roarena-capac":0,
"tc-hotsize":0, "tc-size":2801221, "tc-profsize":12326486, "tc-coldsize":
1612350, "tc-frozensize":6225585,
"targetcache":81712, "rds":81712, "units":647, "funcs":14264
}
```

The admin server

Can do all the sort of potentially useful and surely dangerous things:

```
$ curl localhost:9002/stop
(guess what it does?)
$ curl localhost:9002/stats-sql
On
$ curl localhost:9002/stats-sql
Off
```

Puppet

The puppet classes for HHVM are basically:

- The hhvm module (modules/hhvm)
- modules/mediawiki/manifests/hhvm.pp

Most things you want to tune can be tuned via hiera

Puppet

The puppet classes for HHVM are basically:

- The hhvm module (modules/hhvm)
- modules/mediawiki/manifests/hhvm.pp

Most things you want to tune can be tuned via hiera

We have canary pools!

Salt

Act on canaries:

```
# salt -G 'canary:appserver' test.ping
```

On all but canaries:

```
# salt -C 'G@cluster:appserver and not
    @Gcanary:*' test.ping
```

How to debug when things go wrong

1) Where are the logs?
2) curl/furl
3) Use the admin server
4) Get a stack trace
5) Perf
6) restart HHVM/clean the bc cache
7) Ask Tim to rewrite the relevant HHVM code :)

How to debug when things go wrong

1) Where are the logs?

Upstart logs HHVM's stdout `/var/log/upstart/hhvm.log`

Errors go to `/var/log/hhvm/error.log`, NOT to `/var/log/apache.log!!!`

Errors also go to fluorine, ~~logstash~~

The apache log is full of scary FCGI errors. They are BOGUS

How to debug when things go wrong

2) curl/furl

Curl speaks to apache (HTTP)

```
curl -H 'Host: en.wikipedia.org' localhost/wiki/Main_Page -v
```

Sometimes things get lost in the Apache/HHVM communication (OAUTH is an example)

Furl is Ori's creation, speaks FastCGI directly to HHVM

```
furl --headers --docroot /srv/mediawiki/docroot/wikipedia.org \
     --script /w/index.php http://en.wikipedia.org/wiki/Main_Page
```

How to debug when things go wrong

3) Use the admin server

The `/check-health` endpoint tells you if the Jit cache is full, the number of busy threads, the request queue length.

It is usually a quick way to see what is the state of the system and if it needs respawning

How to debug when things go wrong

4) Get/analyze a stack trace

```
$ sudo quickstack -p `pgrep hhvm`
```

If HHVM already crashed, a core dump you can analyze with gdb is in `/var/log/hhvm`

It's not unusual to get a deadlock where all or most threads are blocked on a futex. Just find out what is this futex blocking on, report it and proceed to 6)

`HPHP::SynchronizableMulti::wait()` means the thread is idle and waiting connections

How to debug when things go wrong

5) Perf

```
$ sudo perf top -p `pgrep hhvm`
```

Perf is usually a good way to understand why things continue to be too slow/HHVM is consuming a lot of resources. We do maintain perf pid maps that should help understanding the output. Ex. the disabling of Xhprof because of a spike of system resource usage.

How to debug when things go wrong

6) Restart HHVM / clean the BC cache

How to debug when things go wrong

6) Restart HHVM / clean the BC cache

After you have determined why HHVM is not responding/slow you will probably want to restart it. The “recommended way” is:

```
# service apache2 stop; \
    service hhvm restart; \
    sleep 5;
    service apache2 start;
```

How to debug when things go wrong

6) Restart HHVM / clean the BC cache

What you will actually do:

```
$ sudo service hhvm restart
```

How to debug when things go wrong

6) Restart HHVM / clean the BC cache

It may happen that strange errors (usually 500 errors originating from the PHP interpreter, not Mediawiki) persist between restarts. In that case clean the BC cache too (as in, delete it).

```
$ sudo service hhvm stop
$ sudo rm -f /run/hhvm/cache/…
$ sudo service hhvm start
```

How to debug when things go wrong

You didn’t expect me to explain in greater detail option 7, right?

Question Time