Alternative Infrastucture

if we do it, we might as well do it right

Experiments

Twitter clone "retwis":
- Key Value Store: redis
- Web Framework: sinatra

Mediawiki:
- Webserver: Nginx (!= Apache)
- PHP: php-fpm (!= mod_php)

Retwis-RB

"An example Twitter application using the Redis key-value database" --> http://github.com/danlucraft/retwis-rb

sinatra	redis
require 'rubygems' require 'sinatra' get '/hi' do "Hello World!" end	require 'rubygems' require 'redis' r = Redis.new puts "set foo to bar" r['foo'] = 'bar'

Redis Benchmark

```
root@keyvalue:~# redis-benchmark -q

SET: 105273.69 requests per second
GET: 107526.88 requests per second
INCR: 95238.10 requests per second
LPUSH: 121987.80 requests per second
LPOP: 108728.26 requests per second
PING: 133386.66 requests per second
```

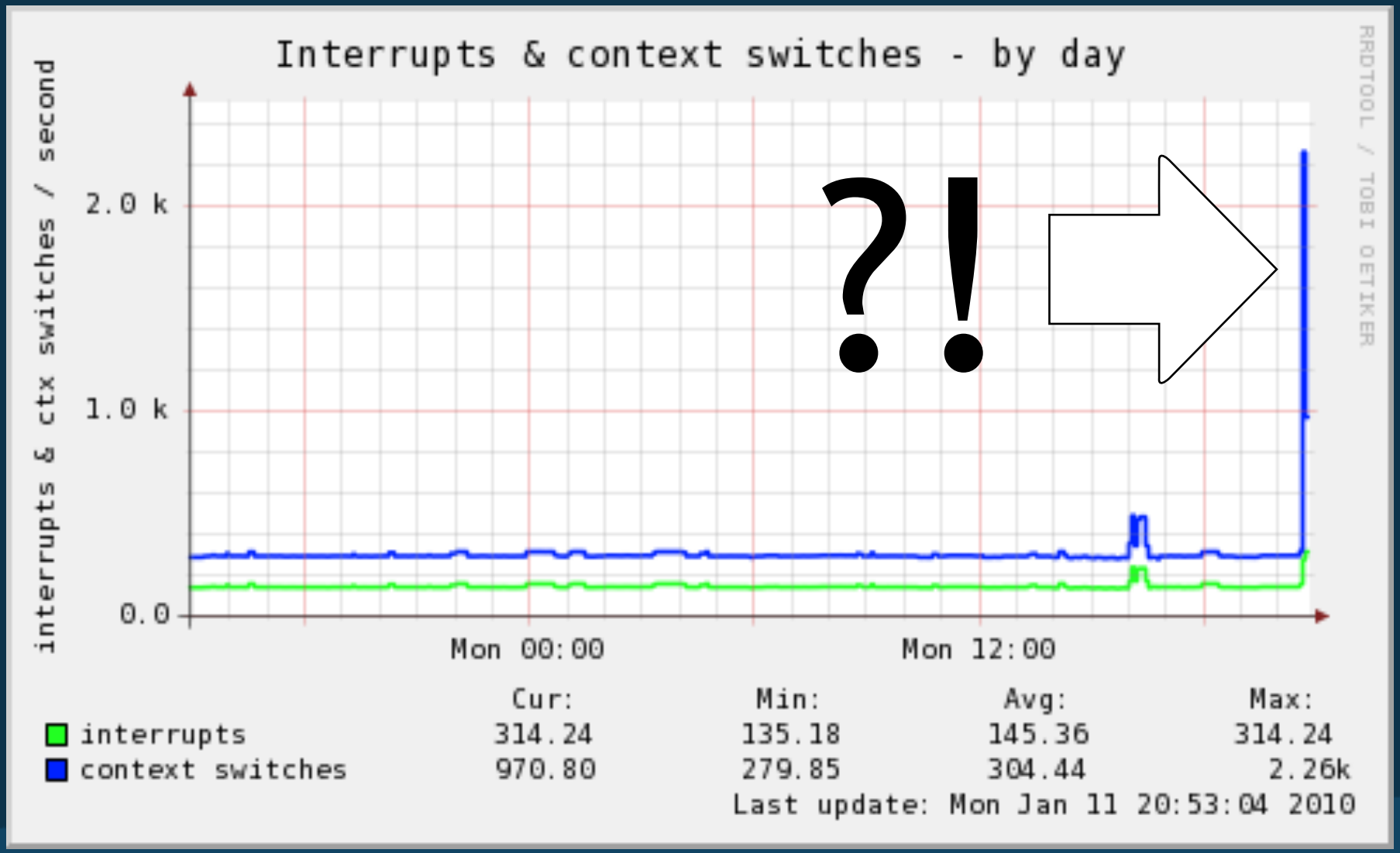
testing @ HdM

CPU:
100% httperf
57% redis-server
40% ruby1.9.1

Durchsatz: 6 req/s

Ò_ó

Context Switches?



server ->

praktikum@keyvalue: ~

File Edit View Terminal Help

```
praktikum@keyvalue:~$ vmstat 5
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 1  0    180  74700 162160 3009088    0    0     2    22    1    1  3  5 92  0
 0  0    180  74692 162160 3009088    0    0     0     0  303  661  1  2 97  0
 0  0    180  74692 162160 3009088    0    0     0     0  134  377  0  0 100  0
 0  0    180  74692 162160 3009088    0    0     0     0  148  390  0  0 100  0
 0  0    180  74692 162160 3009088    0    0     0     0  230  472  0  0 100  0
 0  0    180  74692 162160 3009088    0    0     0     0  234  477  0  0 100  0
 0  0    180  74692 162160 3009088    0    0     0     0  233  476  0  0 100  0
 0  0    180  74468 162160 3009088    0    0     0    13  194  552 19 12 69  0
 0  0    180  74468 162160 3009088    0    0     0    13  134  376  0  0 100  0
 3  0    180  73964 162160 3009092    0    0     0     0  220  495 17 11 72  0
 4  0    180  73940 162164 3009100    0    0     0    13  317  949 61 37  2  0
 4  0    180  73824 162164 3009100    0    0     0    11  320  931 58 40  1  0
 6  0    180  73072 162164 3009108    0    0     0    11  318  933 60 37  2  0
 3  0    180  72892 162164 3009116    0    0     0    21  342  981 60 40  1  0
 4  0    180  73824 162164 3009120    0    0     0    11  314  919 60 39  1  0
 9  0    180  52480 162164 3009124    0    0     0    14  318  954 59 39  1  0
 6  0    180  52348 162164 3009132    0    0     0    11  313 1007 60 39  1  0
 5  0    180  52216 162164 3009140    0    0     0    11  313 1011 60 40  0  0
^C
praktikum@keyvalue:~$ httperf --server=localhost --uri=/praktikum --num-conns=10
0 --num-calls=50
```

home ->

marc@xps: ~

File Edit View Terminal Help

```
              count is the number of updates.
marc@xps:~$ vmstat 5
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  0      0 1614328 264496 1039968    0    0    47    43  215  495  6  2 86  5
 0  0      0 1614536 264504 1039972    0    0     0     3  760 1113  3  1 95  1
 0  0      0 1614888 264512 1039976    0    0     0     8  879 1621  3  3 94  0
 0  0      0 1614872 264524 1039976    0    0     0    45  790 1185  2  1 95  2
 0  0      0 1614808 264524 1039976    0    0     0     4  761 1111  1  1 98  0
 5  0      0 1614132 264536 1039996    0    0     0    21  662 3965 32 10 57  1
 2  0      0 1613692 264552 1039988    0    0     0   112  700 4903 41 10 47  2
 6  0      0 1613868 264580 1039992    0    0     0    41  635 5898 54 17 27  1
 6  0      0 1614536 264600 1039984    0    0     0    49  540 7424 70 26  5  0
 3  0      0 1613824 264604 1040008    0    0     0    13  523 7179 71 28  1  0
 6  0      0 1613880 264616 1039984    0    0     0    26  550 6194 67 28  5  0
 8  0      0 1613356 264632 1039984    0    0     0    53  585 5656 60 27 12  1
 2  0      0 1612484 264636 1039984    0    0     0    26  498 5901 65 32  3  0
 7  0      0 1614580 264640 1039988    0    0     0    17  525 5768 64 28  7  1
 1  0      0 1615112 264656 1039992    0    0     0    23  768 2450 19  8 71  1
 0  0      0 1614488 264664 1039992    0    0     0     4  770 1122  2  1 96  0
^C
marc@xps:~$ httperf --server=localhost --uri=/praktikum --num-conns=100 --num-ca
lls=50
httperf --client=0/1 --server=localhost --port=80 --uri=/praktikum --send-buffer
```

Home Tests:

```
httperf --server=localhost --port=4567 --uri=/test --num-conns=100 --num-calls=50
```

Webserver: "thin"

EventMachine:
a library for Ruby, C++, and Java programs. It provides event-driven I/O using the Reactor pattern

single connection

Total:
connections 1
requests 500
replies 500

	home	
Request rate	56.7 req/s (17.6 ms/req)	
Reply rate [replies/s]	min 41.6 avg 41.6 max 41.6	
Reply time [ms]	response 17.6 transfer 0.0	
Reply status:	1xx=0 2xx=500 3xx=0 4xx=0 5xx=0	
test-duration	8.824 s	

100 connections

Total:
connections 100
requests 5000
replies 5000

	home
Request rate	48.5 req/s (20.6 ms/req)
Reply rate [replies/s]	min 16.2 avg 48.7 max 71.6
Reply time [ms]	response 20.6 transfer 0.0
Reply status:	1xx=0 2xx=5000 3xx=0 4xx=0 5xx=0
test-duration	103.160 s

2nd Experimental Setup

Replacing
Apache+mod_php by
Nginx+PHP-FPM

Nginx ...

is...

- a lightweight Web Server
- a Reverse Proxy
- an IMAP/POP3 proxy

- used by 14,988,610 domains today
- implmented by larges sites as WordPress, Github, SourceForge etc.

Nginx vs Apache

- Apache
  - process based
  - each connection requires a new thread
  - high concurrency
  -> high memory usage
  -> CPU overhead (e.g. context switches)
  - PHP is usally included in Apache Web Server as module (mod_php)

- Nginx
  - fork of apache 1.3 with the multi-processing ripped out in favor of an event loop
- asynchronous model (event based)
- uses only one thread for all connections (master thread)
- PHP is used as seperate process over FastCGI (PHP-FPM)
  - Web Server and PHP-FPM are used as seperate applications
  - communication via TCP-connetions or Unix-sockets
  -> little overhead due to communication costs

Event Loop

What is an event loop?

usually you write code like:

```
var result = db.query("select..");
result.do_something();
```

.

.

but an event loop looks like:

```
db.query("select..", function (result) { result.do_something()});
```

Motivation

Apache+mod_php compared to **Nginx+php-fpm**

(comparison made by Boštjan Škufca - http://blog.a2o.si))

5 Different scenarios

- *HelloWorld.php – simple echo of* “Hello, World!” (13 bytes),
- *HelloWorld.txt – static file with* “Hello, World!” (also 13 bytes)
- *100KB.txt – static content*
- *1MB.txt* - static content
- *index.php – more complex site with several DB-queries, HTML template parsing…*

Tests with keepalive-feature [-k] and without keepalive
(same socket can be used for several requests)

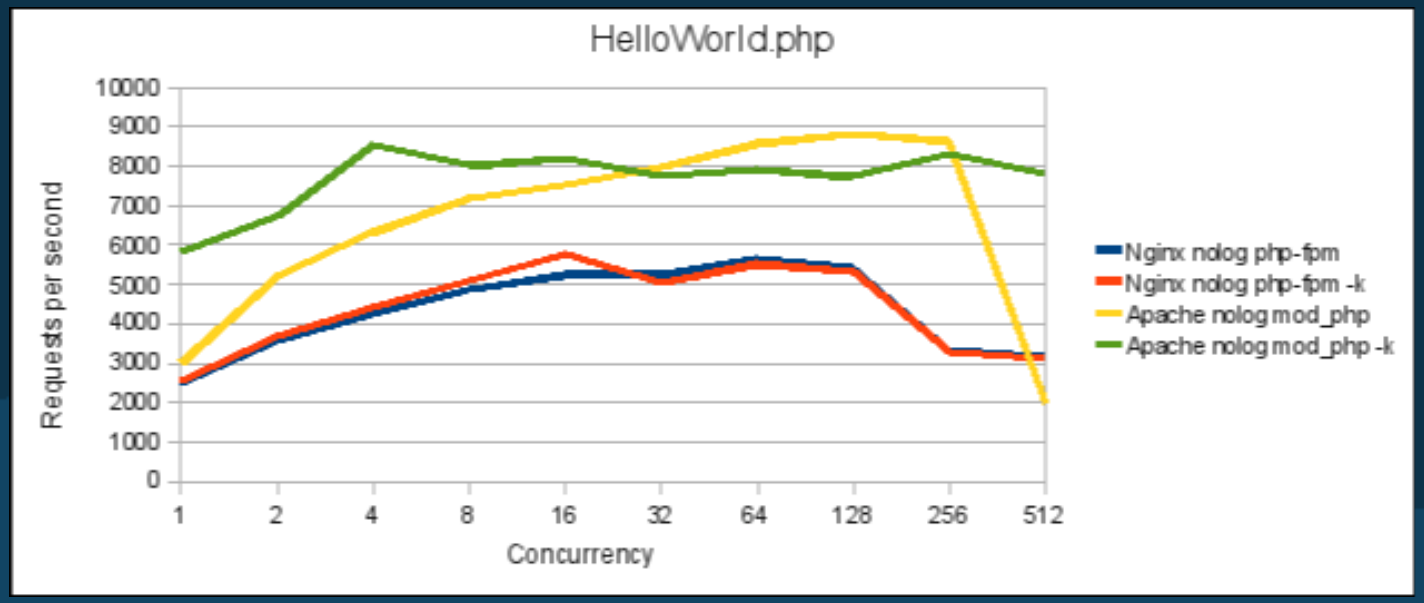
Benchmark Setup

- Benchmark tests conducted using ApacheBench
- ab -n *NREQ* -c *NCONC* [-k] http://server.domain.com/bench/FileName
  NREQ is the number of requests:
  - HelloWorld.php: 500000
  - HelloWorld.txt: 500000
  - 100KB.txt: 500000
  - 1MB.txt: 50000
  - AppFront: 5000
- NCONC = number of concurrent requests

1 ,2,4,8,16,32,64,128,256,512

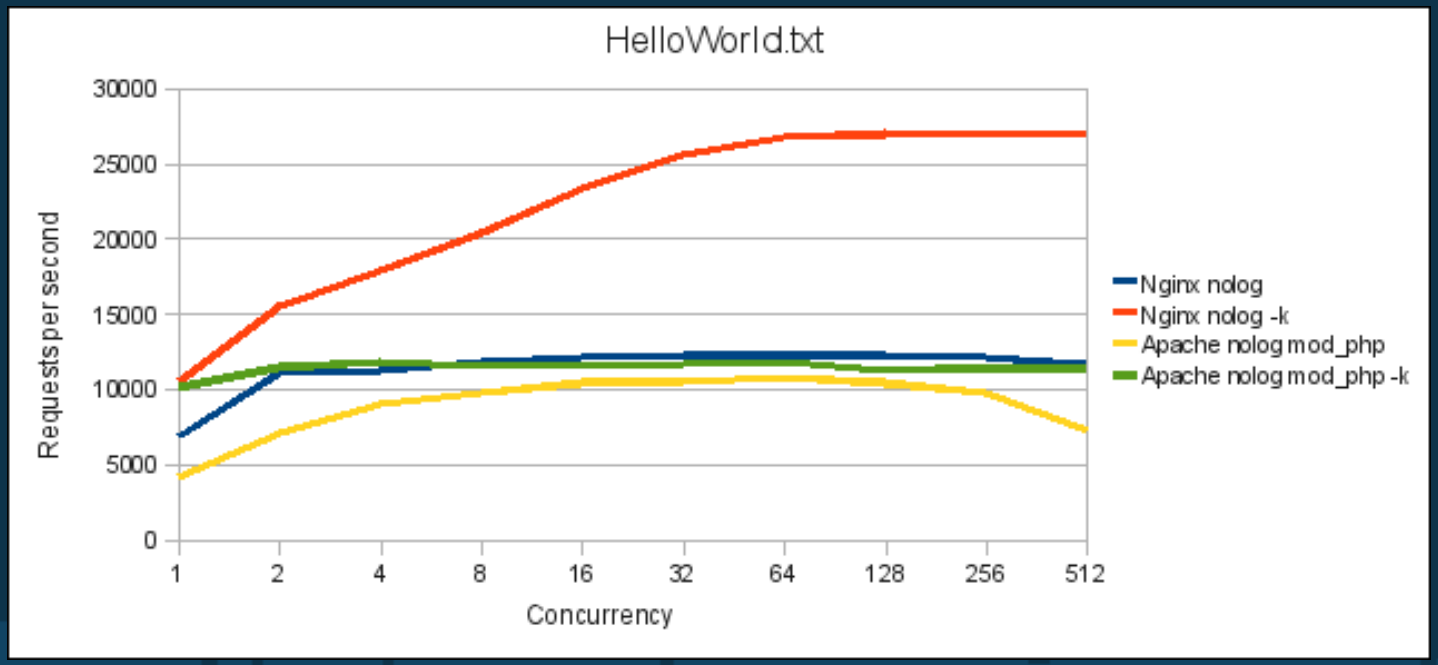
PHP-generated Hello World!

- Apche is always faster than Nginx
- This demonstrates the overhead of the communication between Nginx and PHP-FPM



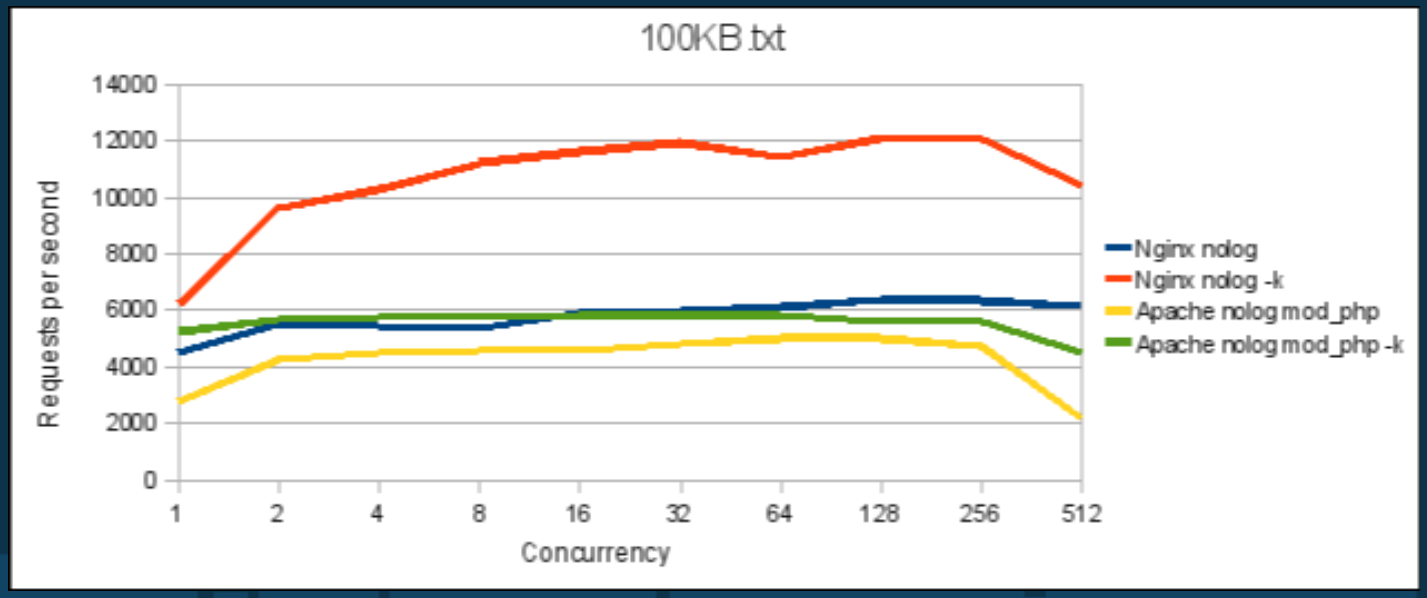
Static Hello World!

- Nginx with keepalive is more than twice as fast as Apache
- This demonstrates the overhead that is caused by creating TCP-connections



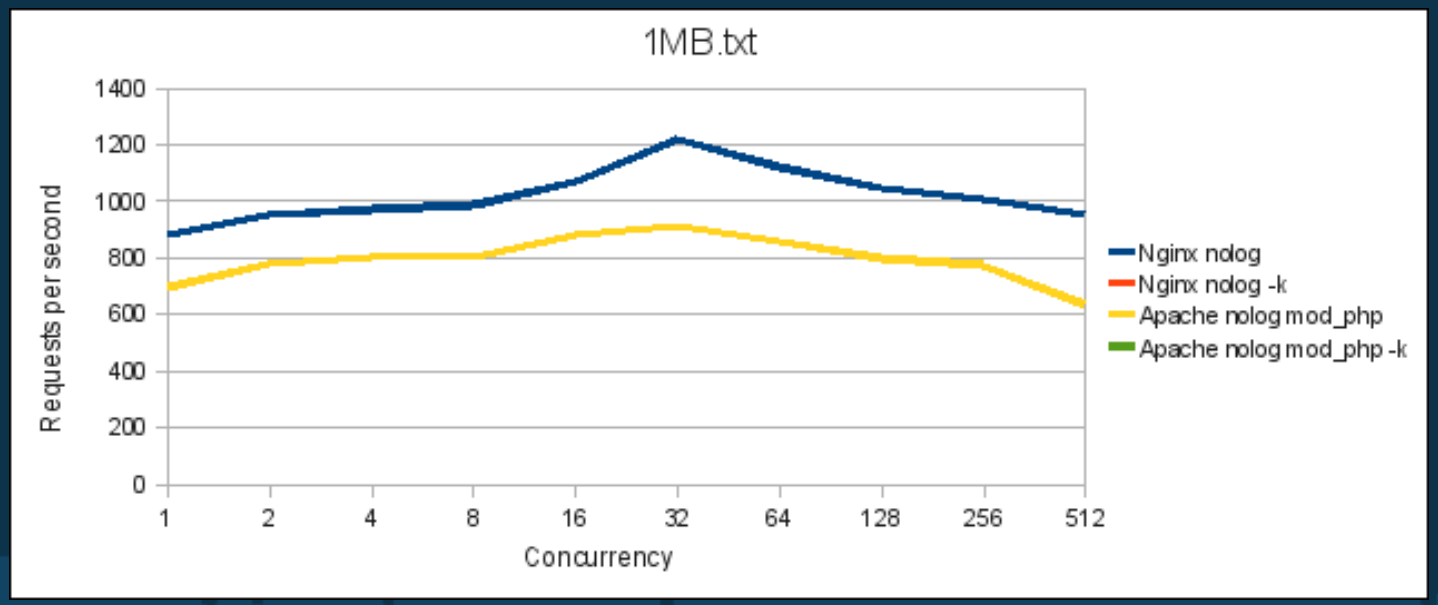
Static 100kb.txt File

- This test should demonstrate a „real world “ example of a static page request
- Again, Nginx is twice as fast as Apache



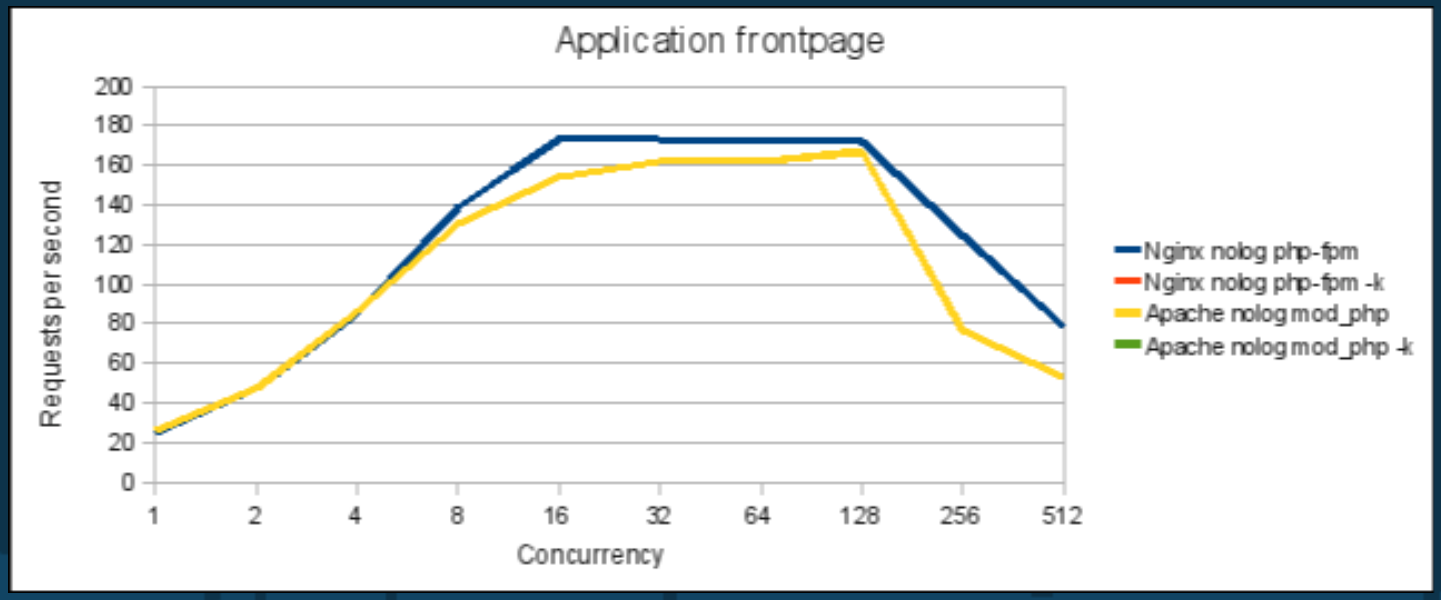
1MB.txt File

- This test demonstates a more complex file transfer
- Keepalive was not tested, because the file size is so large that TCP-connections aren‘t important
- Nginx is just slightly better



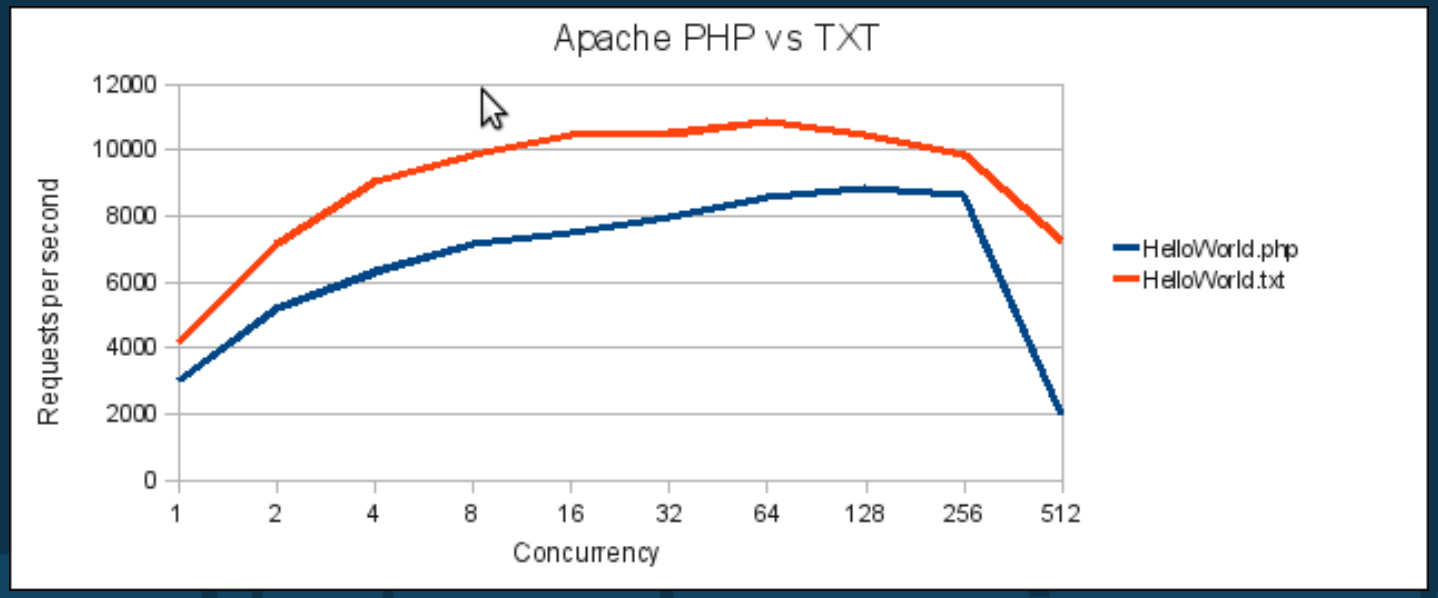
Application Frontpage (index.php)

- Again a „real world“ example with a more complex PHP-site
- Nginx is just slightly better



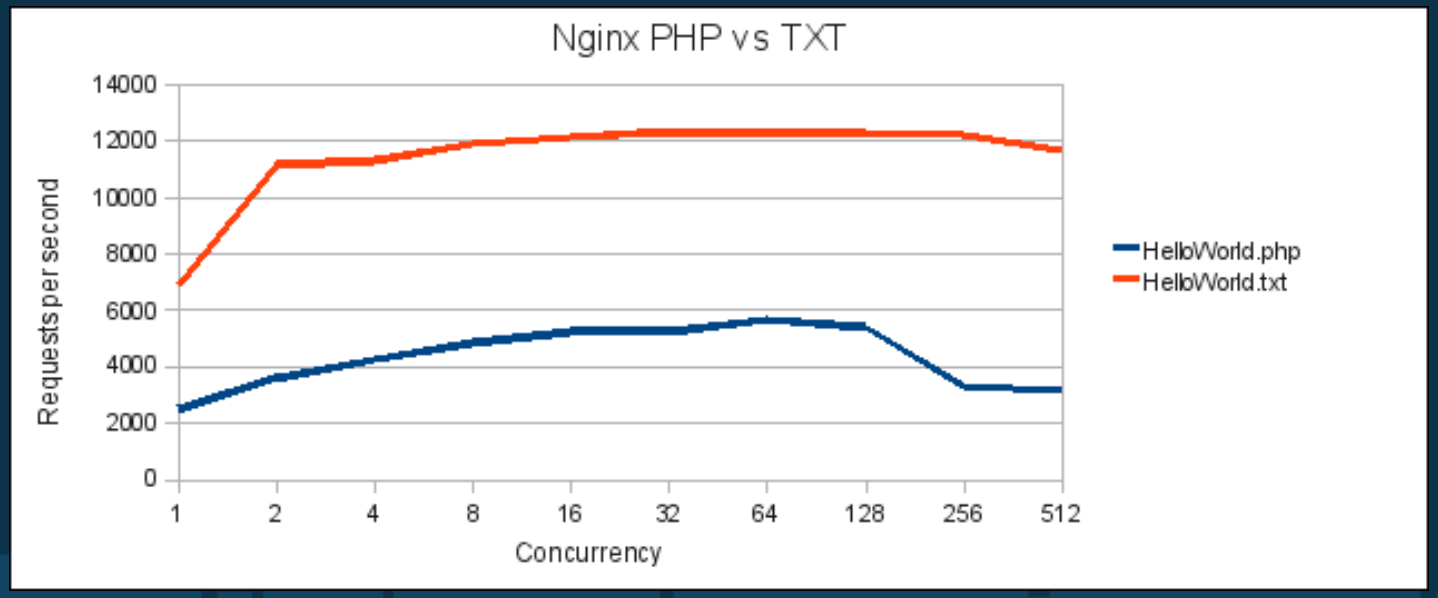
Apache PHP vs TXT

- Dynamically generated content and static content are nearly equally fast

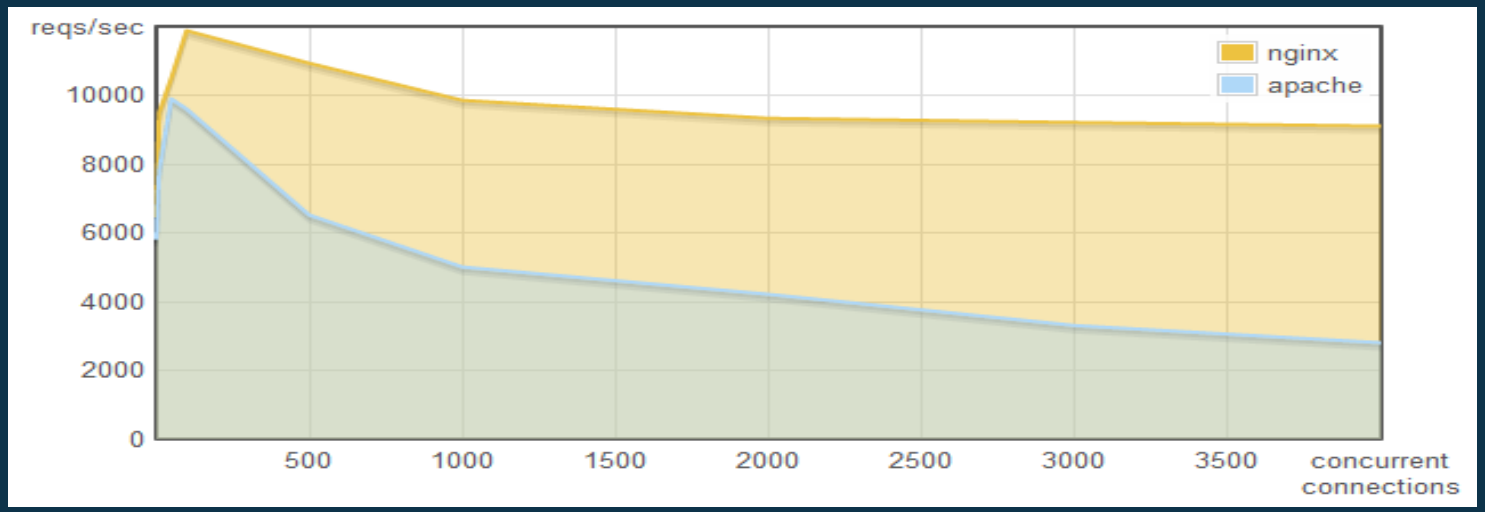


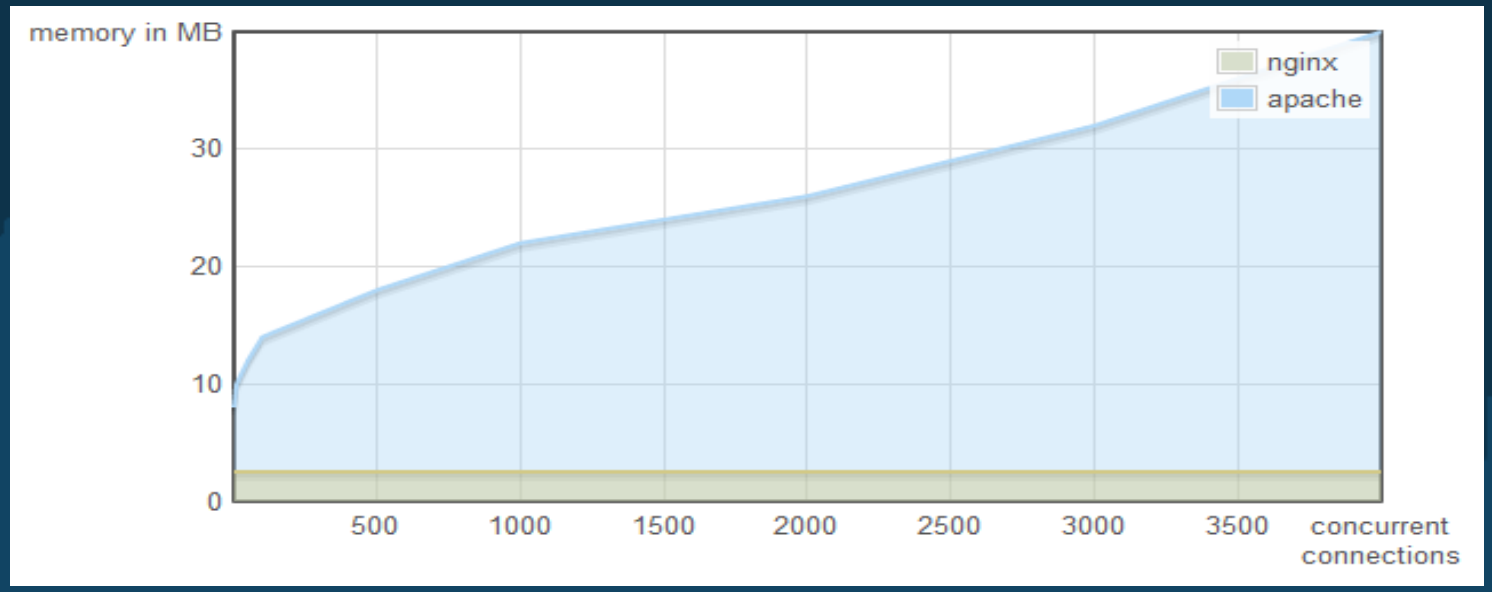
Nginx PHP vs TXT

- Nginx serves static content twice as fast as dynamic content



Additional Comparison





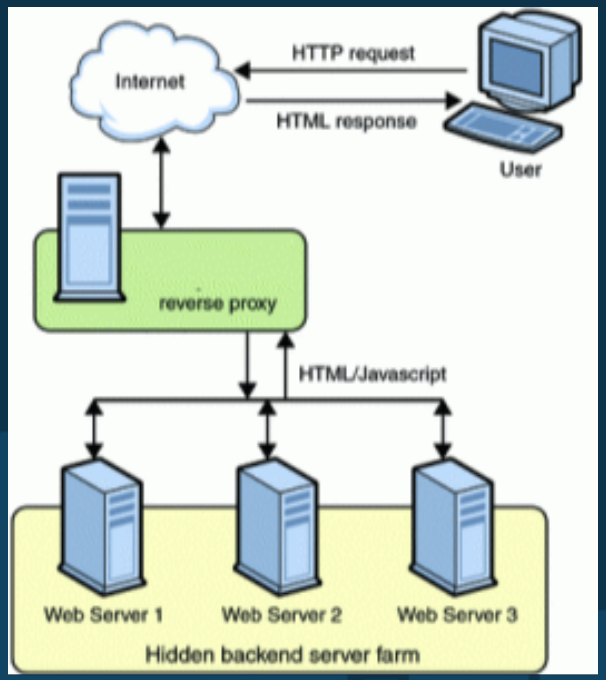
Conclusion

- When is it worth to use Nginx?
  - If you have limited hardware resources (e.g. on VPS)
  - If you have a lot of static content

Further Alternative

- Nginx works as reverse proxy
- static content is passed by Nginx (e.g. Pictures)
- dynamic Content will be forwarded to an Apache behind the proxy

- advantages:
  - static content will be returned very fast
  - slow user connections do no longer hold resources, because the "blocking" connection is now between Nginx and Apache; not the user and Apache



our problems

mysql dumps:
create dump + copy dump + insert dump = hours

loadtesting:

- client == server

--> no testing for high concurrency, no isolation of variables.

- client too slow
- different configurations
  - keepalives
  - Nginx workers/processes vs apache threads/clients
  - ...