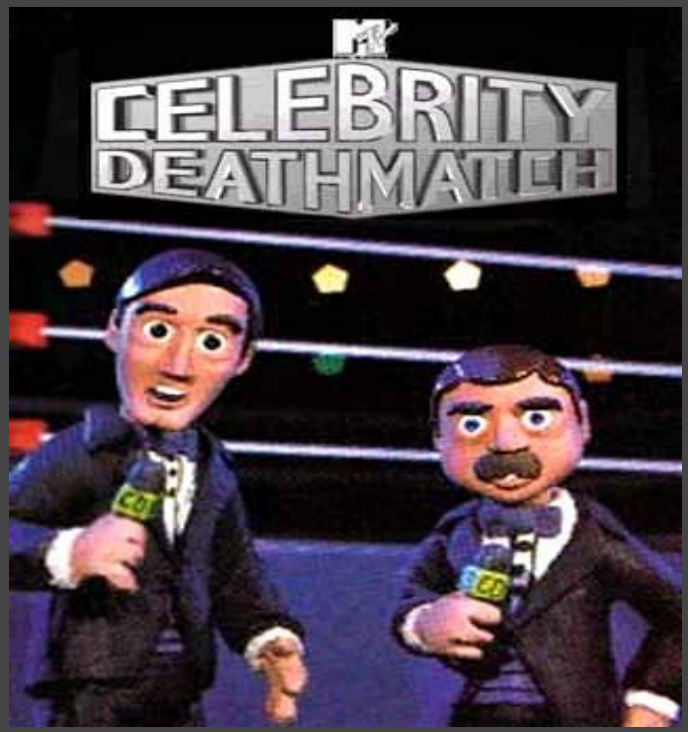
Web Server Deathmatch



Joe Williams
Cloudant

@williamsjoe on Twitter
http://www.joeandmotorboat.com/

Overview

The Contenders

The Systems, Test Setup, Environment and Configuration

The Results
- Base Tests
- Experiements
- 13A vs 12B-5

Take Home

The Contenders

Apache (Prefork)
Apache (Worker)
Erlang INETS:HTTP
Lighttpd
MochiWeb
Nginx
Yaws

The Systems

Server
- Intel Core 2 Duo T7500 2.20GHz
- 4GB RAM
- 7200 RPM Hard Disk
- Ubuntu 8.10 x86_64

httperf Client
- Intel Core 2 Duo P7350 2.0GHz
- 2GB RAM
- 5400 RPM Hard Disk
- OSX x86_64

GigE link between machines

The Test

Used httperf to test performance of serving static files

Request payload 10k PNG file

Six total test runs
- Three tests then reboot of server then three more

Httperf configuration
- Request rates, 1000 to 5000/sec
  - Incremented by 1000 each test
- Based on connections and calls
  - 1000 requests = 100 connections x 10 calls per connection
  - 2000 requests = 200 connections x 10 calls per connection
    - etc ..

Environment & Configuration

Ran many iterations of the test to find optimal configurations for each server

httperf via autobench

httperf compiled with
-DFD_SETSIZE=10000
-D_DARWIN_ULIMITED_SELECT

autobench is a wrapper script for automating httperf tests

http://www.xenoclast.org/autobench/
http://www.hpl.hp.com/research/linux/httperf/

Environment & Configuration

ulimits

File limits (ulimit -n) were increased on both the client and server

sysctl.conf

```
kernel.sem = 250 32000 100 128
kernel.shmall = 2097152
kernel.shmmax = 2147483648
kernel.shmmni = 4096

fs.file-max = 65536

vm.min_free_kbytes = 65536
vm.swappiness = 5
vm.vfs_cache_pressure = 50
```

```
net.core.rmem_default = 524288
net.core.rmem_max = 16777216
net.core.wmem_default = 524288
net.core.wmem_max = 16777216
net.core.netdev_max_backlog = 2500

net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.tcp_rfc1337 = 1
net.ipv4.tcp_sack = 1
net.ipv4.tcp_fack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_ecn = 0
net.ipv4.route.flush = 1
net.ipv4.ip_no_pmtu_disc = 0
```

Environment & Configuration

Apache 2.2.11

Prefork and Worker

Compiled with "--enable-nonportable-atomics=yes"

```
Timeout 300
 KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
UseCanonicalName Off
ServerTokens Full
ServerSignature On
HostnameLookups Off
MaxClients        10000
```

Environment & Configuration

- INETS:HTTP
  - Erlang 12B-5 and 13A
  - erl +K true +h 1600
    - Kernel polling and heap size per process
  - Max Clients set to 300
    - Increasing beyond this seemed to degrade performance

Environment & Configuration

Lighttpd 1.4.22

```
server.max-keep-alive-requests = 16
server.max-keep-alive-idle = 5
server.max-read-idle = 60
server.max-write-idle = 360
server.event-handler = "linux-sysepoll"
server.network-backend = "linux-sendfile"
server.max-fds = 10000
server.max-connections = 10000
server.max-worker = 2
```

Environment & Configuration

MochiWeb

- Erlang 12B-5 and 13A
- erl +K true +h 1600

Code Changes

- Added garbage_collect() to cleanup function in mochiweb_request.erl
- Set max=5120 to socket server record in mochiweb_socket_server.erl

Environment & Configuration

Nginx 0.7.43

```
worker_processes  2
worker_connections  5120
sendfile on
keepalive_timeout 65
```

Environment & Configuration

Yaws 1.81

- Erlang 12B-5 and 13A
- erl +S 4 +h 1600
  - Four schedulers
  - Kernel polling is on by default

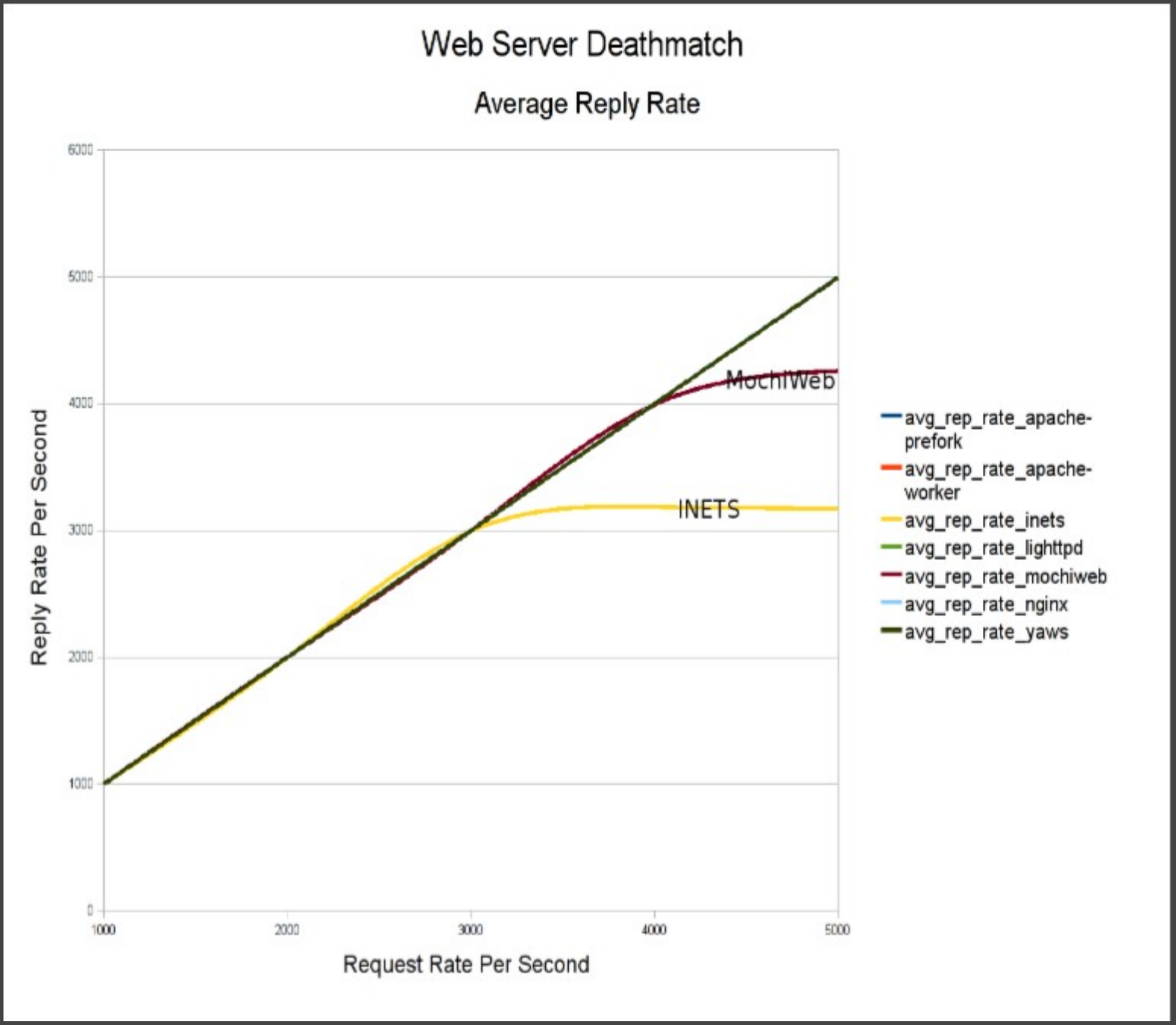
- Turning off logging improves performance
  - access_log = false

- Sendfile serves small files (<4k) in memory

Results

INETS maxes out just over 3000 replies/s

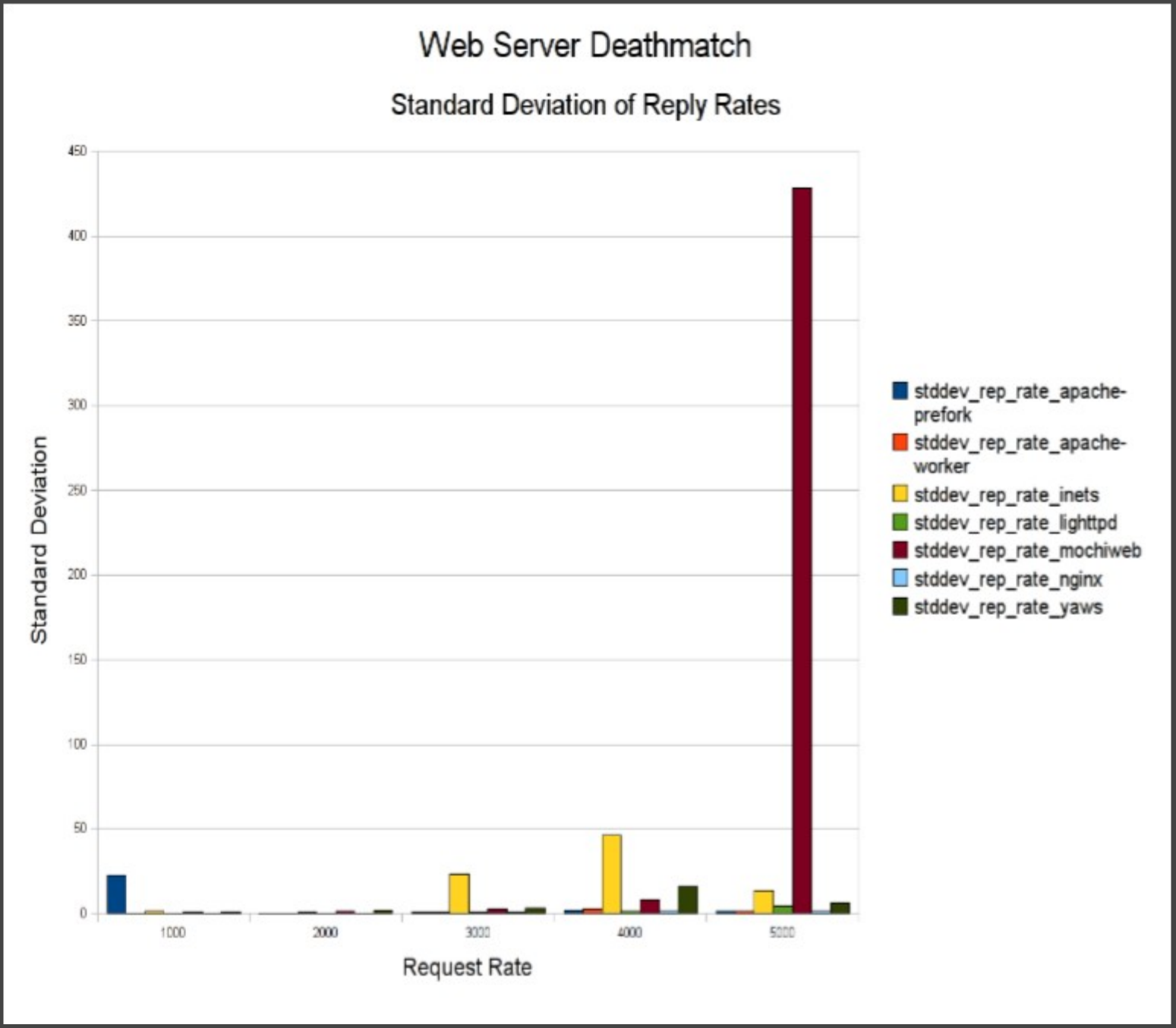
MochiWeb maxes out around 4300 replies/s



Results

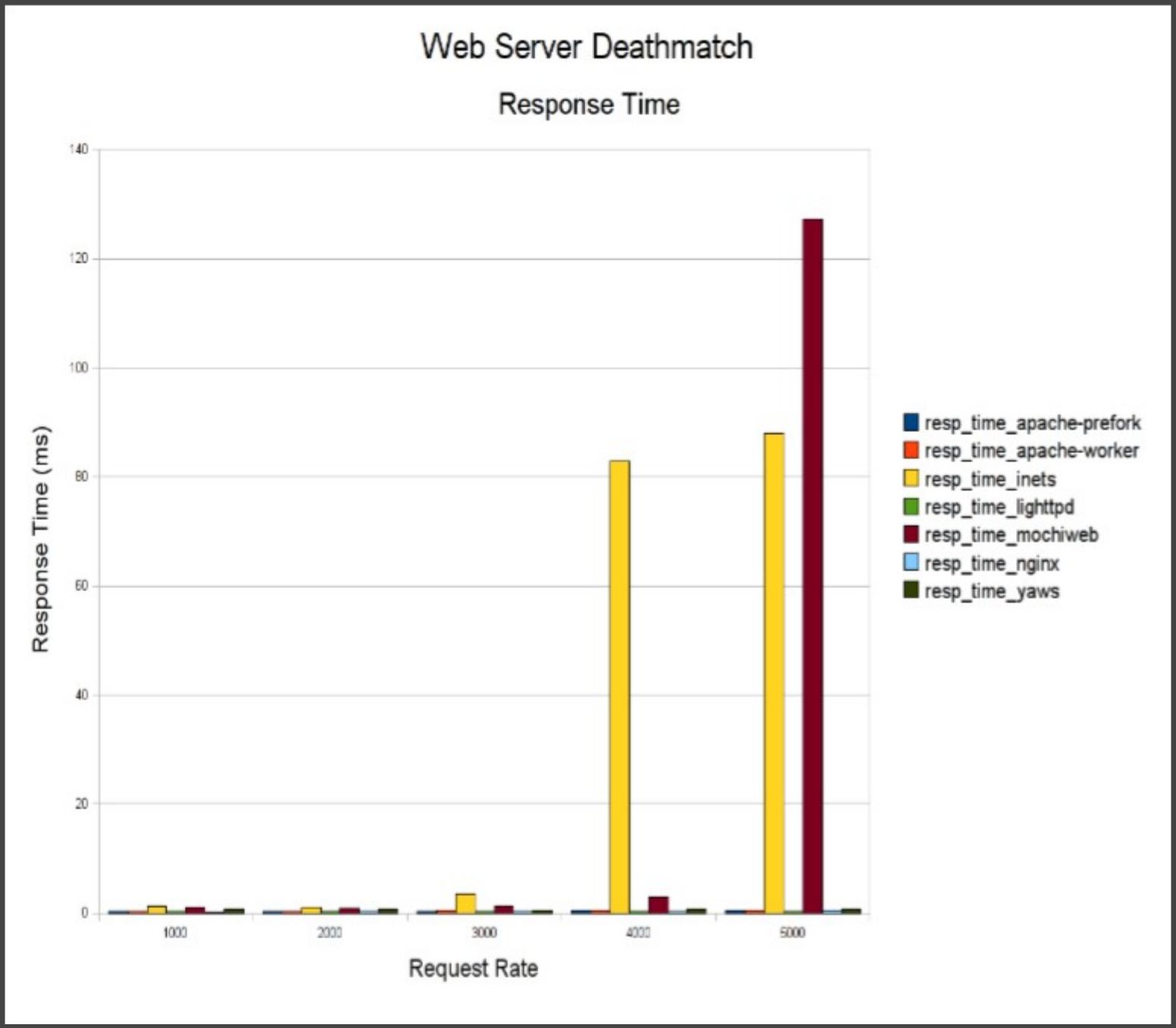
INETS seems to be consistent albeit slow

MochiWeb is inconsistent at 5000 replies/s



Results

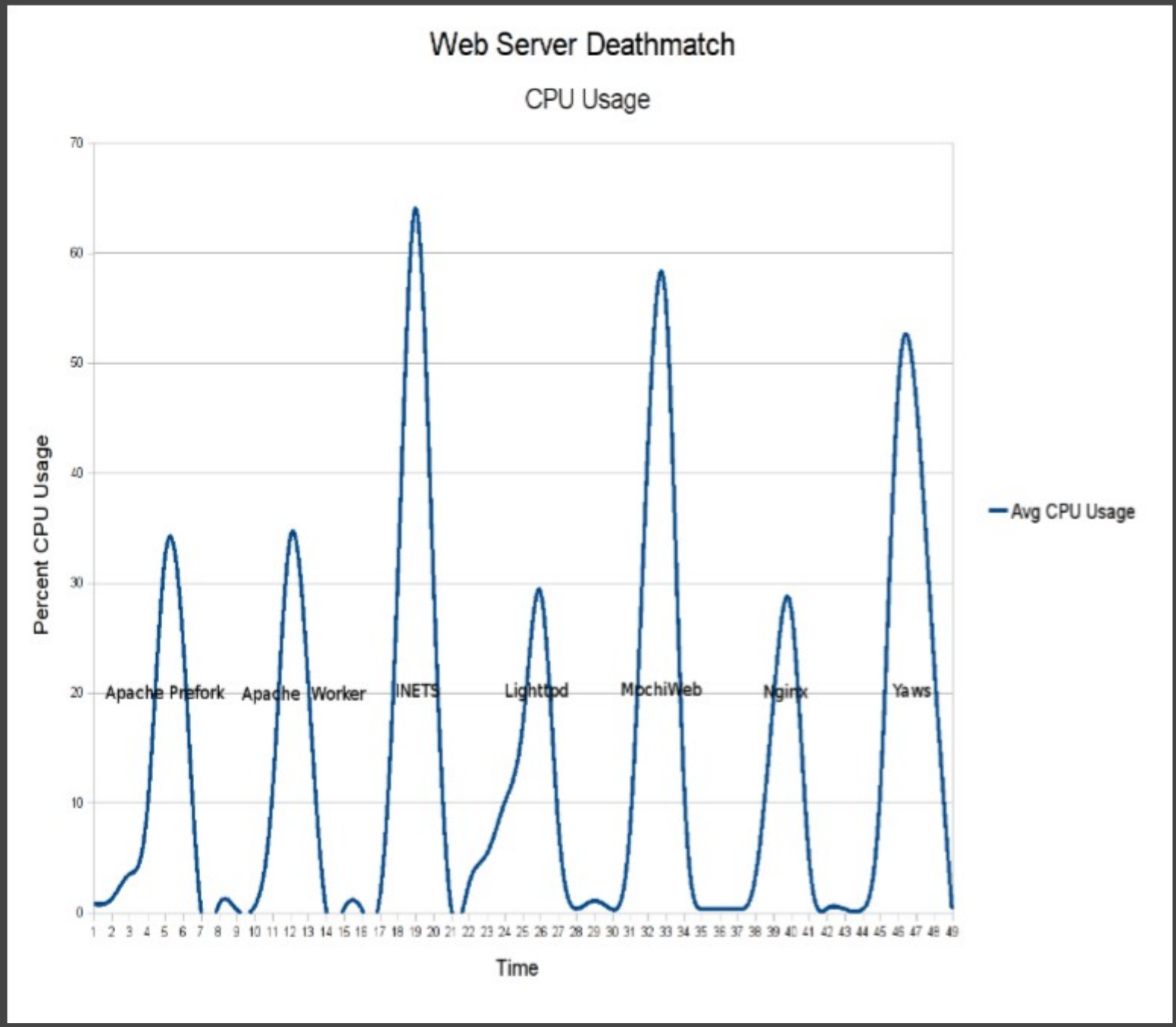
Similar to the other metrics INETS and MochiWeb have issues at the 4000 and 5000 level



Results

Nginx and Lighttpd use the least CPU

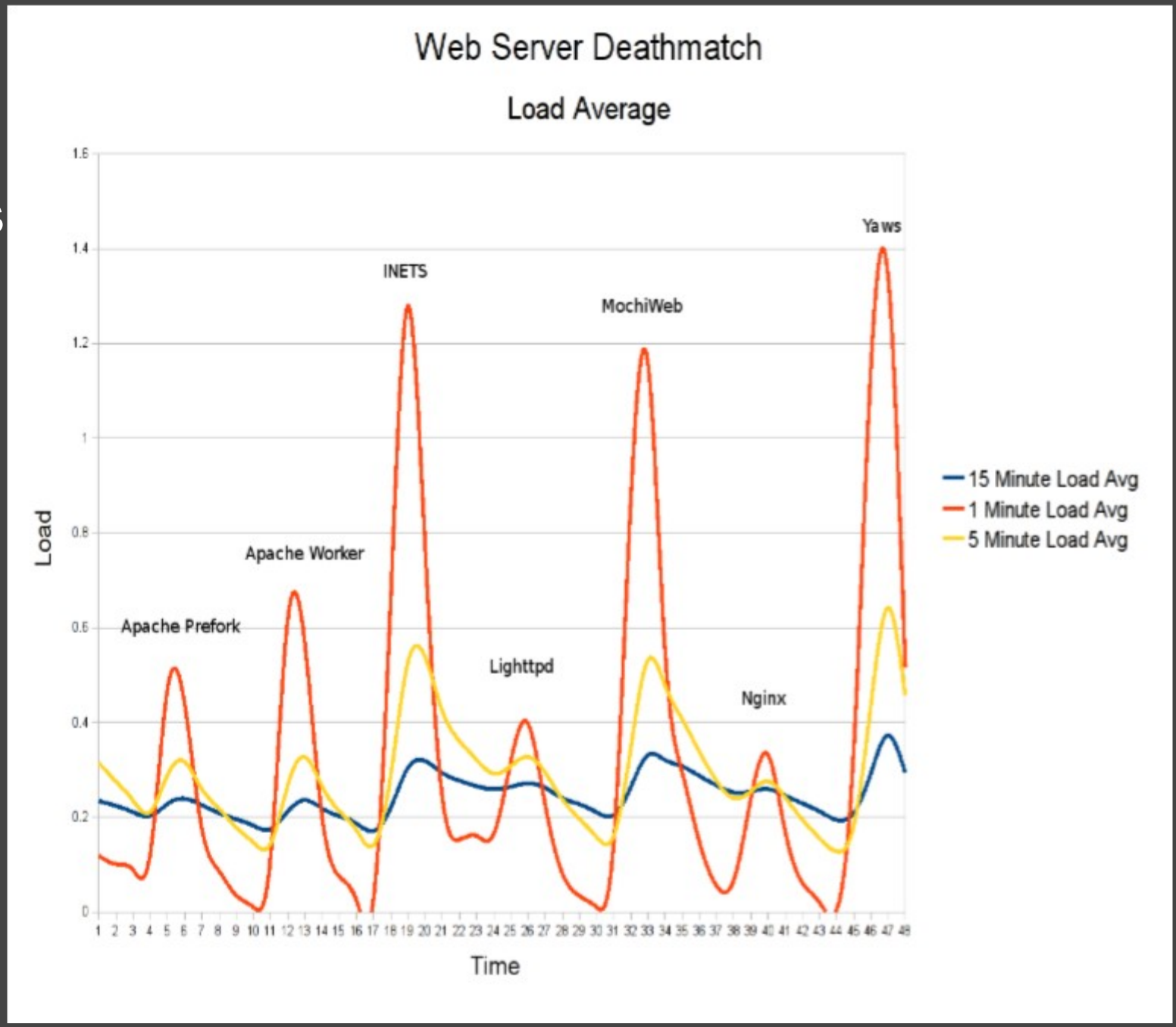
The Erlang servers seem to use more



Results

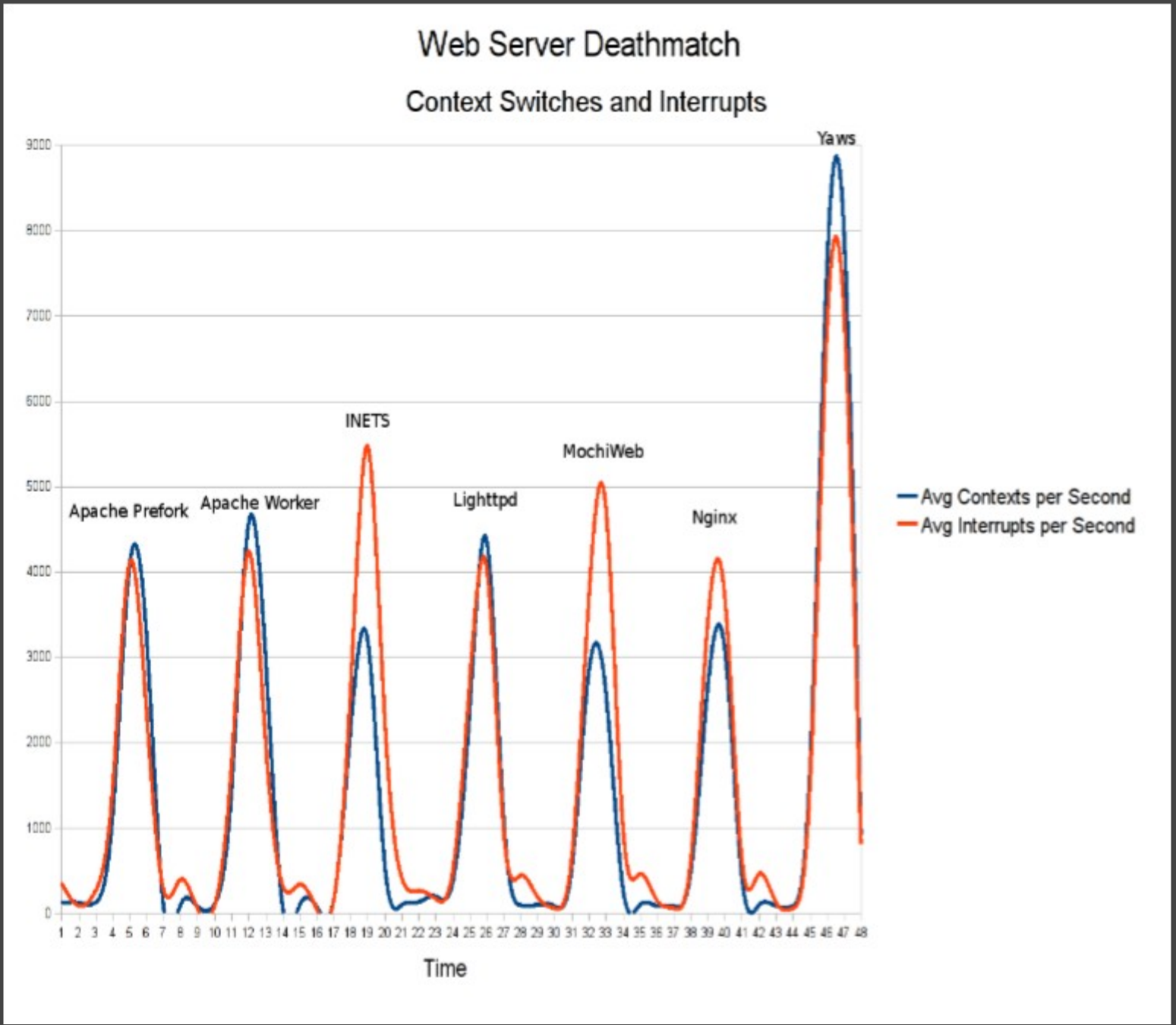
Load was low with all servers

Yaws was the highest with 1.4



Results

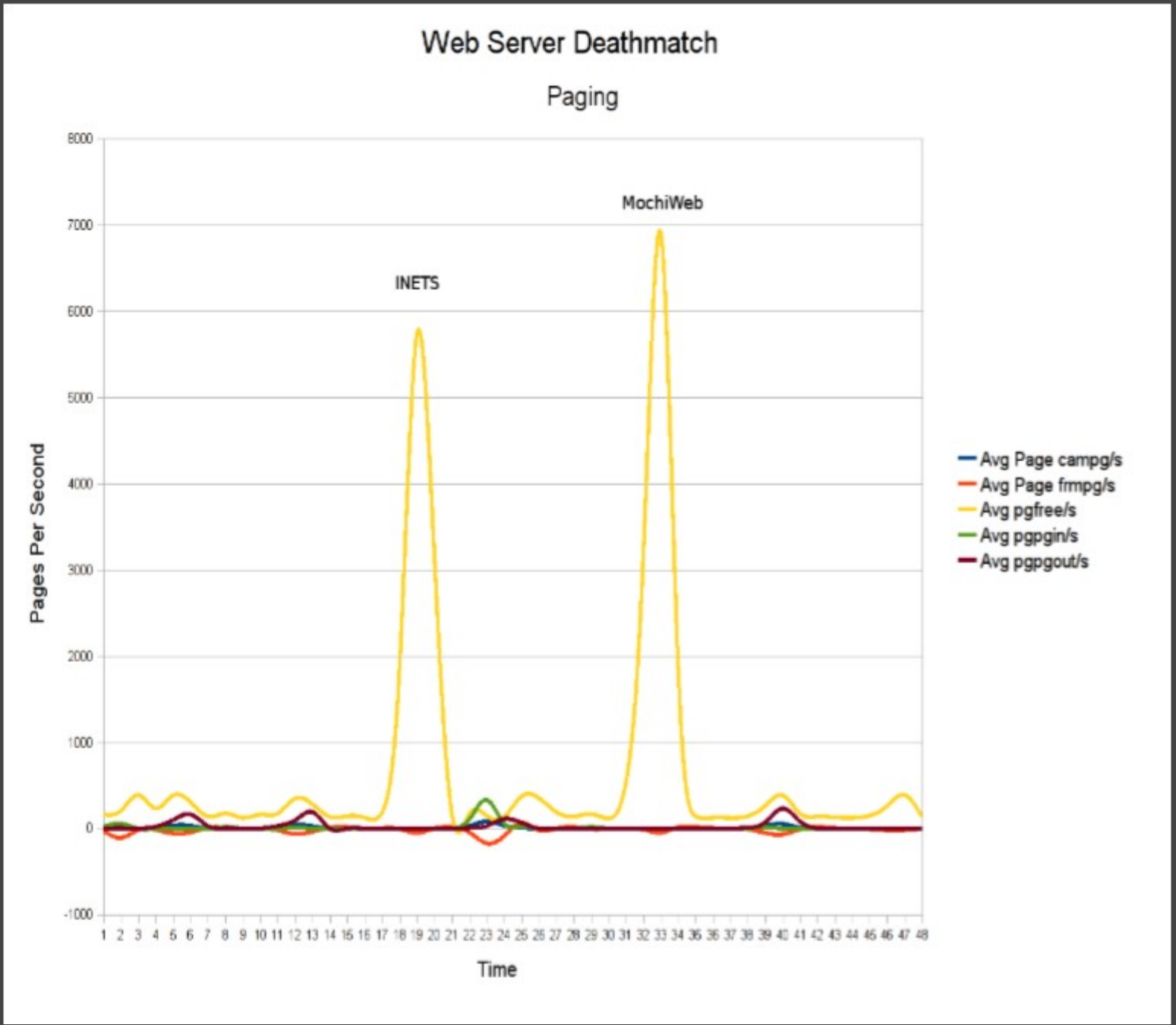
Yaws causes a high level of interrupts and context switches



Results

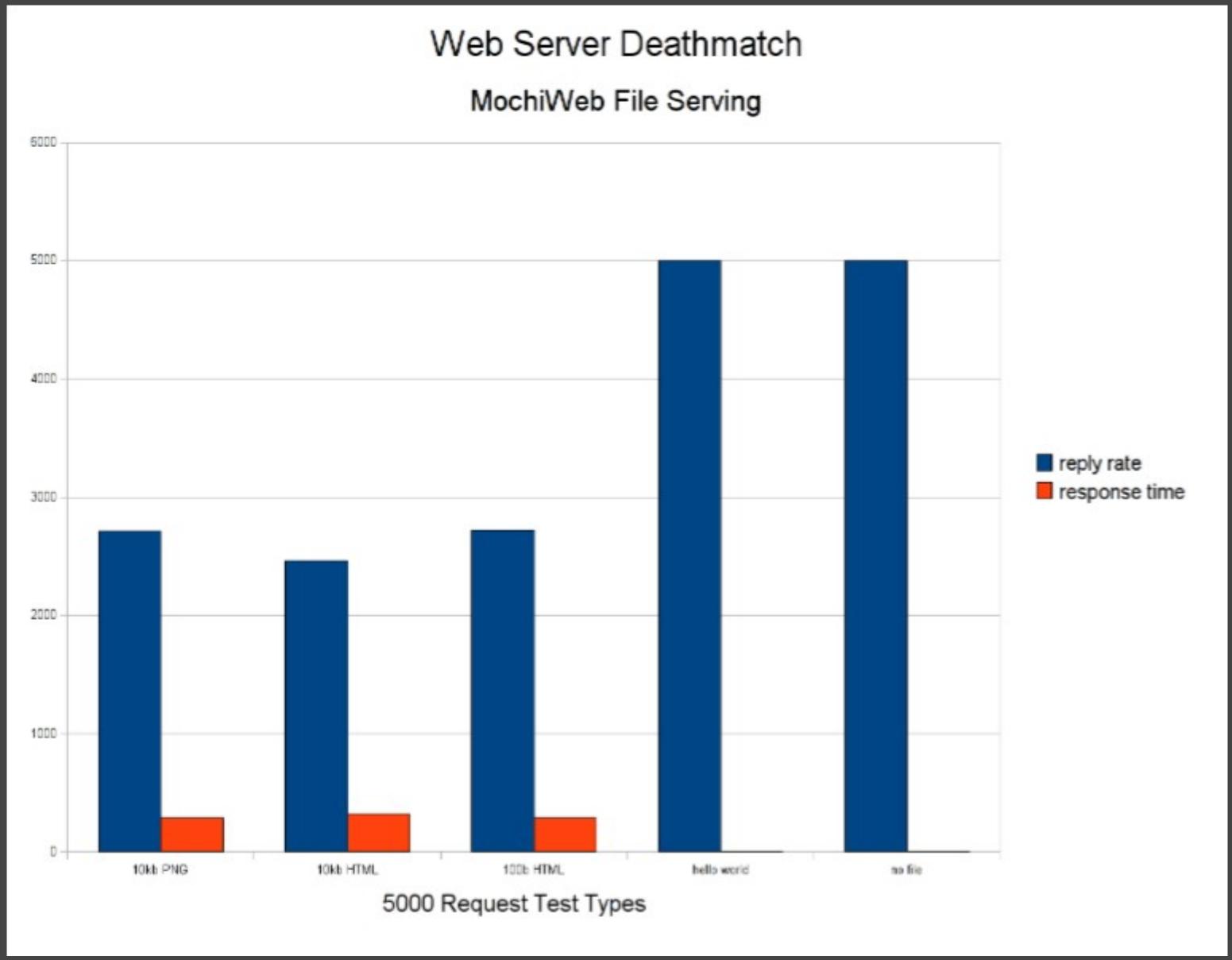
MochiWeb and INETS had very high pgfree/s

Usually the sign of creating and destroying processes



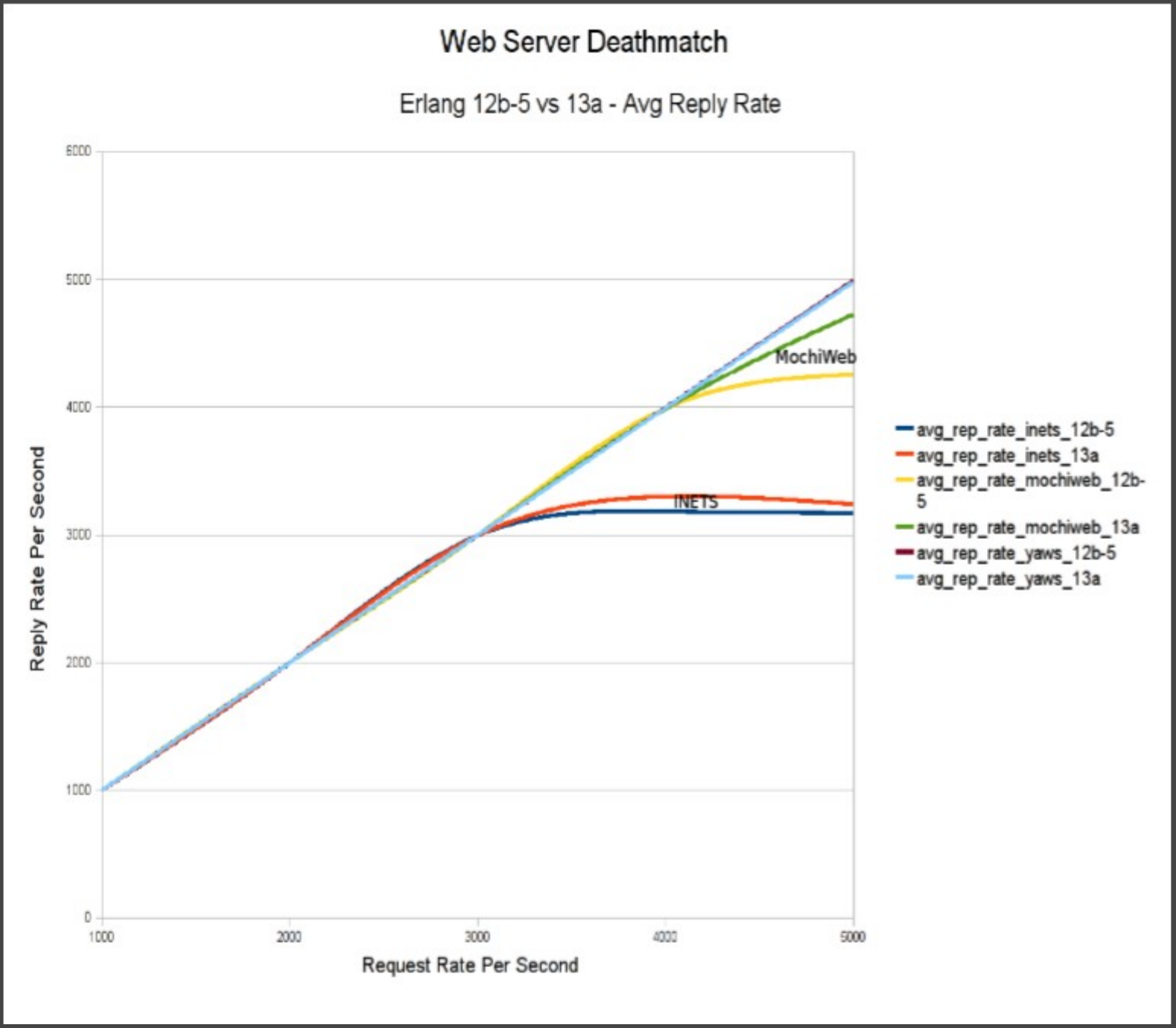
Experimental Tests

- MochiWeb seems to have issues reading files from the filesystem



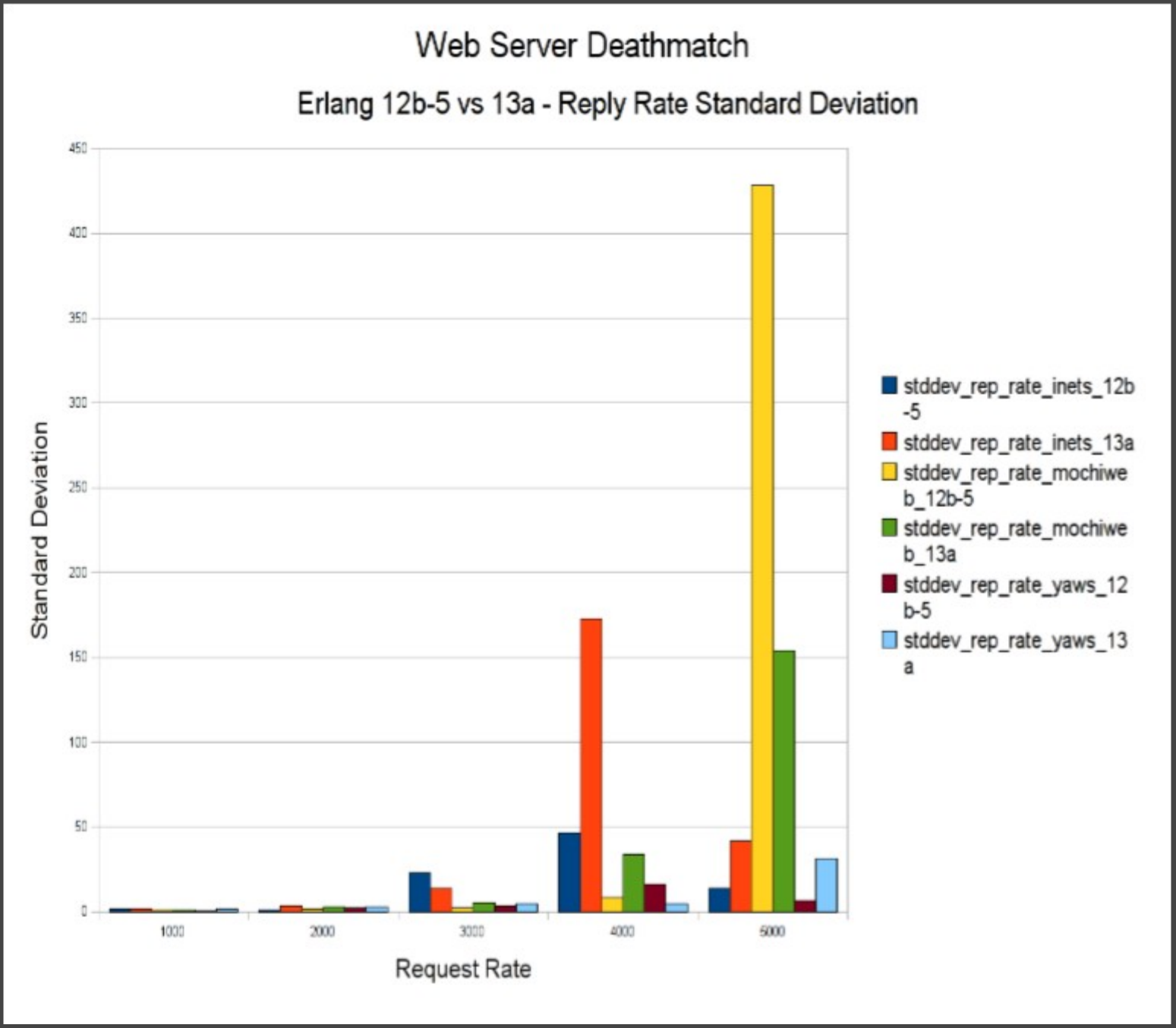
Results

MochiWeb and INETS improve with 13A



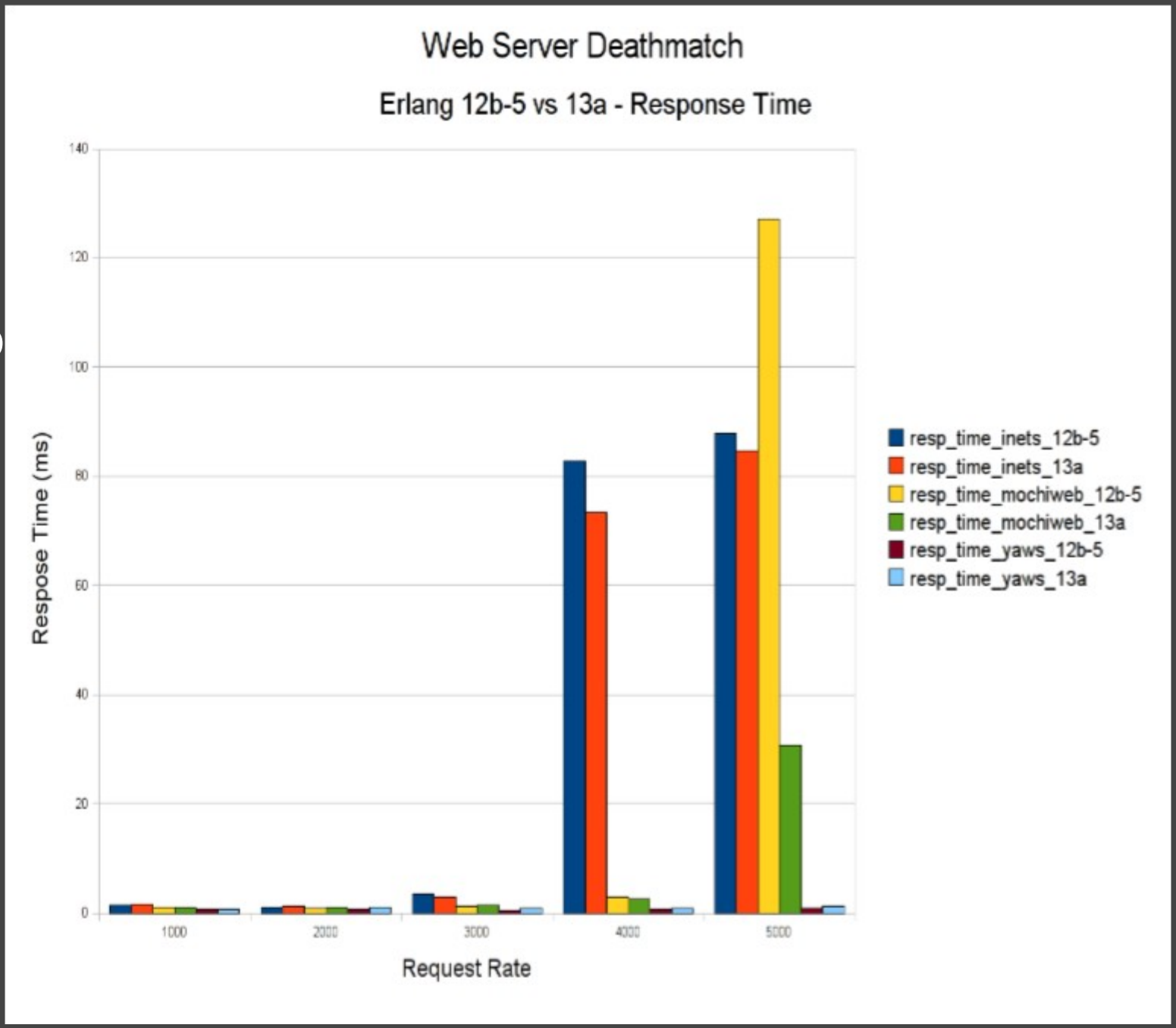
Results

Consistency improved with 13A for MochiWeb



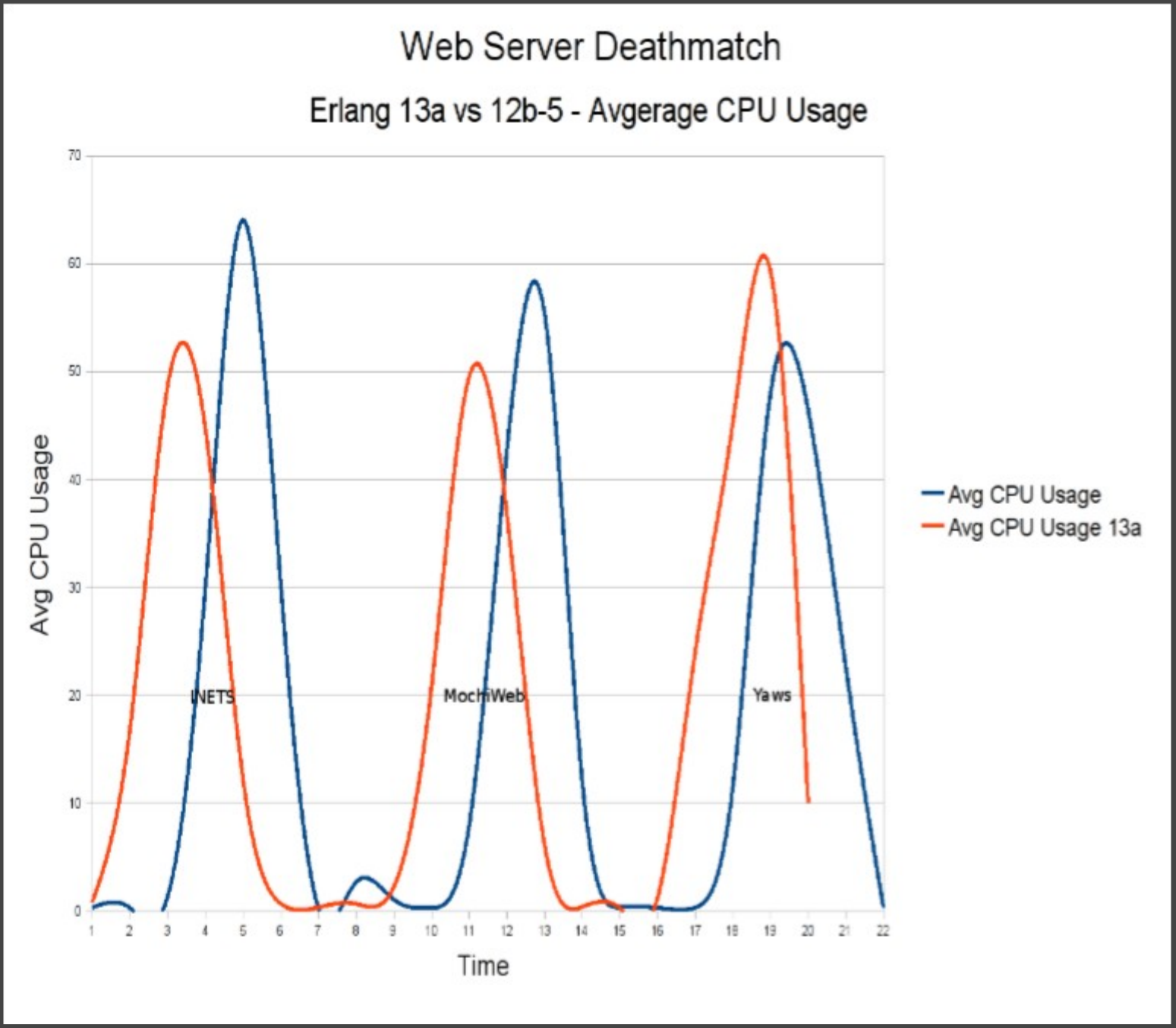
Results

Response times dropped for both INETS and MochiWeb



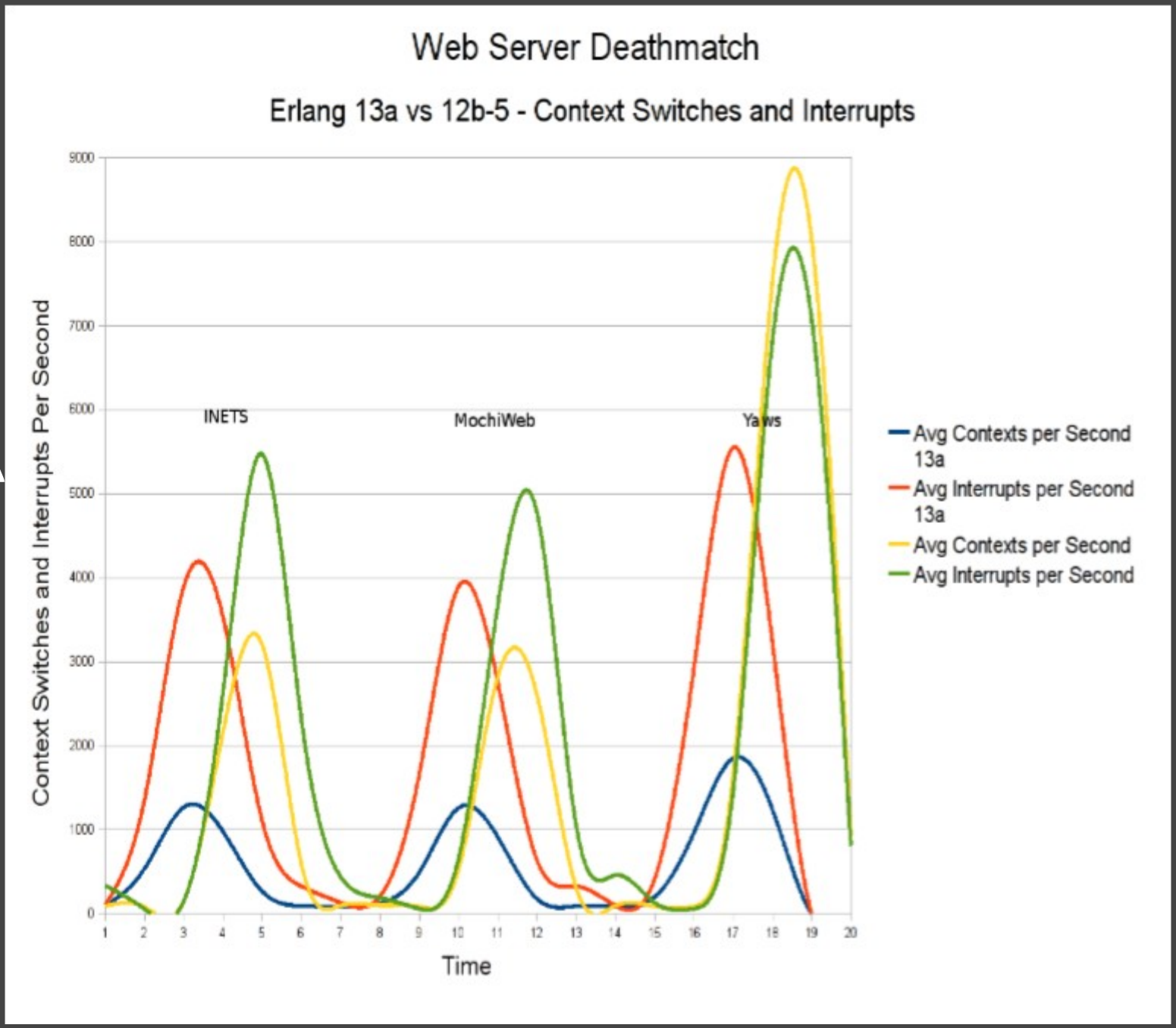
Results

- Yaws CPU usage increases with 13A
- MochiWeb and INETS do well with 13A



Results

- Decreased context switches and interrupts across the board with 13A



Take Home

Yaws, Nginx, Lighttpd and Apache all perform well in this test

Erlang based servers used more CPU than I expected

Overall 13A helps performance across the board

Test them out in your environment and work load

Take Home

Apache
- Compile with non-portable atomics
- Make sure max clients is set high enough

INETS
- Increasing max clients causes more harm than good
- Maxes out around 3500 requests a second

Lighttpd
- Performs well with low CPU usage
- Epoll, send file, FD's and keep alives are key for performance

Take Home

MochiWeb
- Not built for serving file serving
- Otherwise performs very well
- Setting GC in clean up function and high connection limit help performance

Nginx
- Performed best, fast by default

Yaws
- Turning off access logs helps greatly

Thanks

Steve Vinoski and Claes Wikstrom with the Yaws project

Bob Ippolito creator of MochiWeb