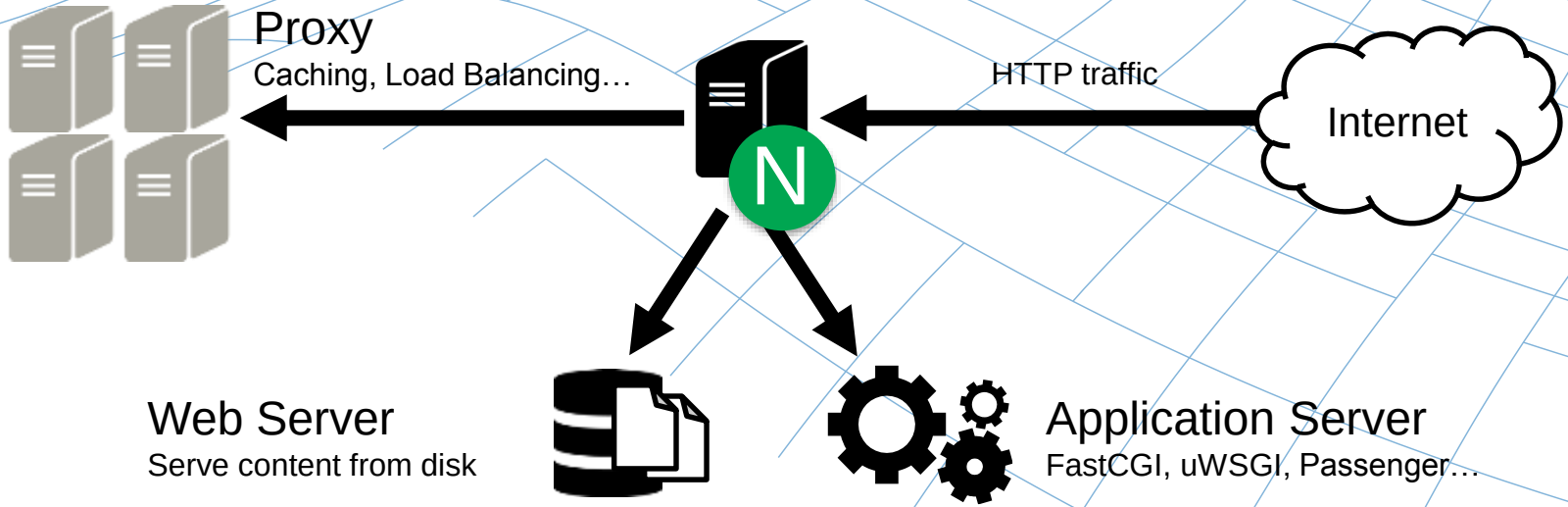
5 things you didn't know NGINX could do

Sarah Novotny
Nginx, Inc.

Many people know NGINX as an HTTP request and load balancing server that powers many of the world's busiest websites. But, there are a lot of ancillary pieces that go into the software to make it a whole web application accelerator.

What is NGINX?



NGINX Accelerates

143,000,000

Websites

Advanced Features

- ☑Bandwidth Management
- ☑Content-based Routing
- ☑Request Manipulation
- ☑Response Rewriting
- ☑Application Acceleration
- ☑SSL and SPDY termination
- ☑Authentication
- ☑Video Delivery
- ☑Mail Proxy
- ☑GeoLocation
- ☑Performance Monitoring
- ☑High Availability

22%

Top 1 million websites

37%

Top 1,000 websites

Those 5 things --

1. Compress assets for delivery
2. Stop form spamming
3. Protect Apache from thread exhaustion attacks
4. Rewrite content inline
5. Online updates

Bonus: determine a nearly complete command
for the configure script

1. Compress data to reduce bandwidth

- Reduce bandwidth requirements per client
  - Content Compression reduces text and HTML
  - Image resampling reduces image sizes

HTTP gzip module

- Provides Gzip capabilities so that responses from NGINX are compressed to reduce file size
- Directives can be used in the http, server and location contexts
- Key directives
  - `gzip`
  - `gzip_types`
  - `gzip_proxied`

Gzip example

Enable gzip

```
gzip   on;
```

Apply gzip for text, html and CSS

```
gzip_types text/plain text/html text/css;
```

Enable gzip compression for any proxied request

```
gzip_proxy    any;
```

It is not advisable to enable gzip for binary content types such as images, word documents or videos

HTTP image filter

- Provides inline image manipulation to transform images for optimal delivery
- Directives can be used in the location context
- Key directives
  - `image_filter size;`
  - `image_filter resize` *`width height`*`;`
  - `image_filter crop` *`width height`*`;`

HTTP image filter example

```
location /img/ {
    proxy_pass   http://backend;
    image_filter resize 150 100;
    image_filter rotate 90;
    error_page   415 = /empty;
}
location = /empty {
    empty_gif;
}
```

We talk about the ‘N second rule’:

- 10 seconds
  (Jakob Nielsen, March 1997)
- 8 seconds
  (Zona Research, June 2001)
- 4 seconds
  (Jupiter Research, June 2006)
- 3 seconds
  (PhocusWright, March 2010)

2. Stop brute force retries

- Stop brute force password attacks
- Stop form spamming
  - Use the NGINX limit request module

HTTP limit req module

- Allows granular control of request processing rate
- Directives an be used in http, server and location contexts
- Key directives
  - `limit_req_zone`
  - `limit_req`

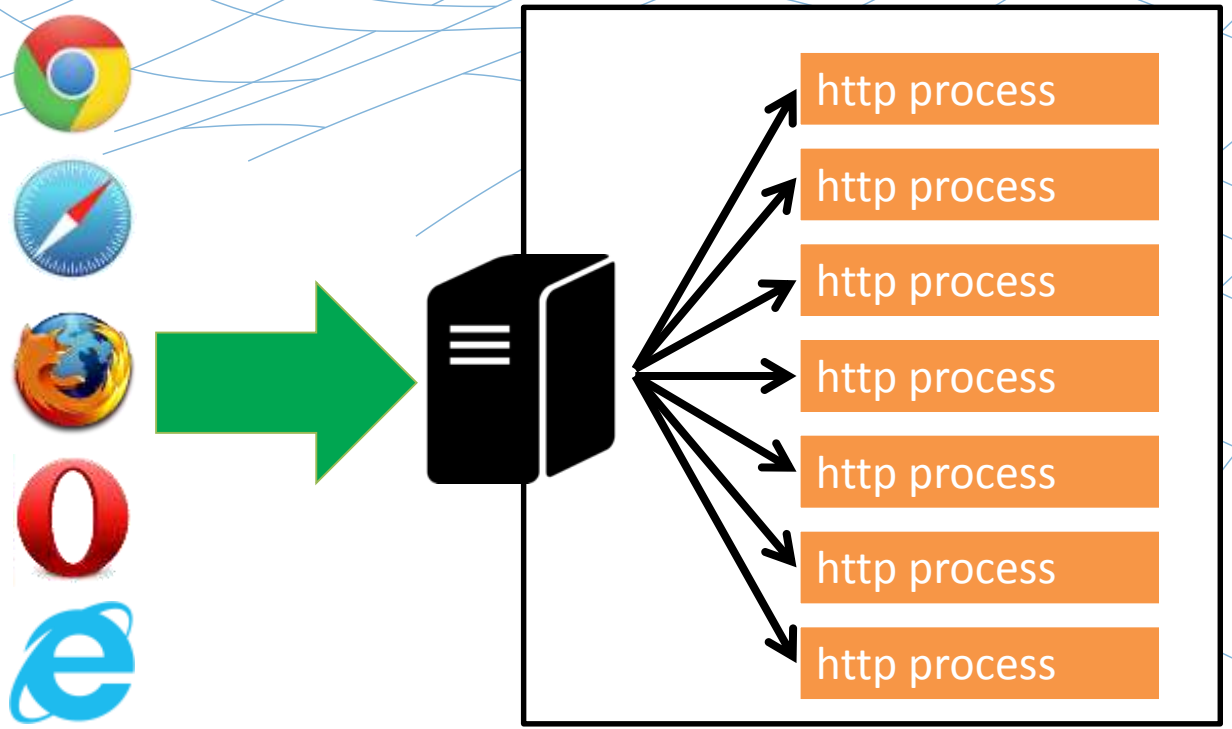
HTTP limit req module

```
http {
    limit_req_zone $binary_remote_addr zone=one:10m
rate=1r/s;
    …
    server {
        …
        location /search/ {
            limit_req zone=one burst=5;
        }
    }
}
```

3. Protect Apache from thread exhaustion attacks

- Use NGINX in front of Apache
- Mitigates ‘slow loris’, ‘keep dead’ and ‘front page of hacker news’ attacks

What is thread exhaustion?



Client-side:

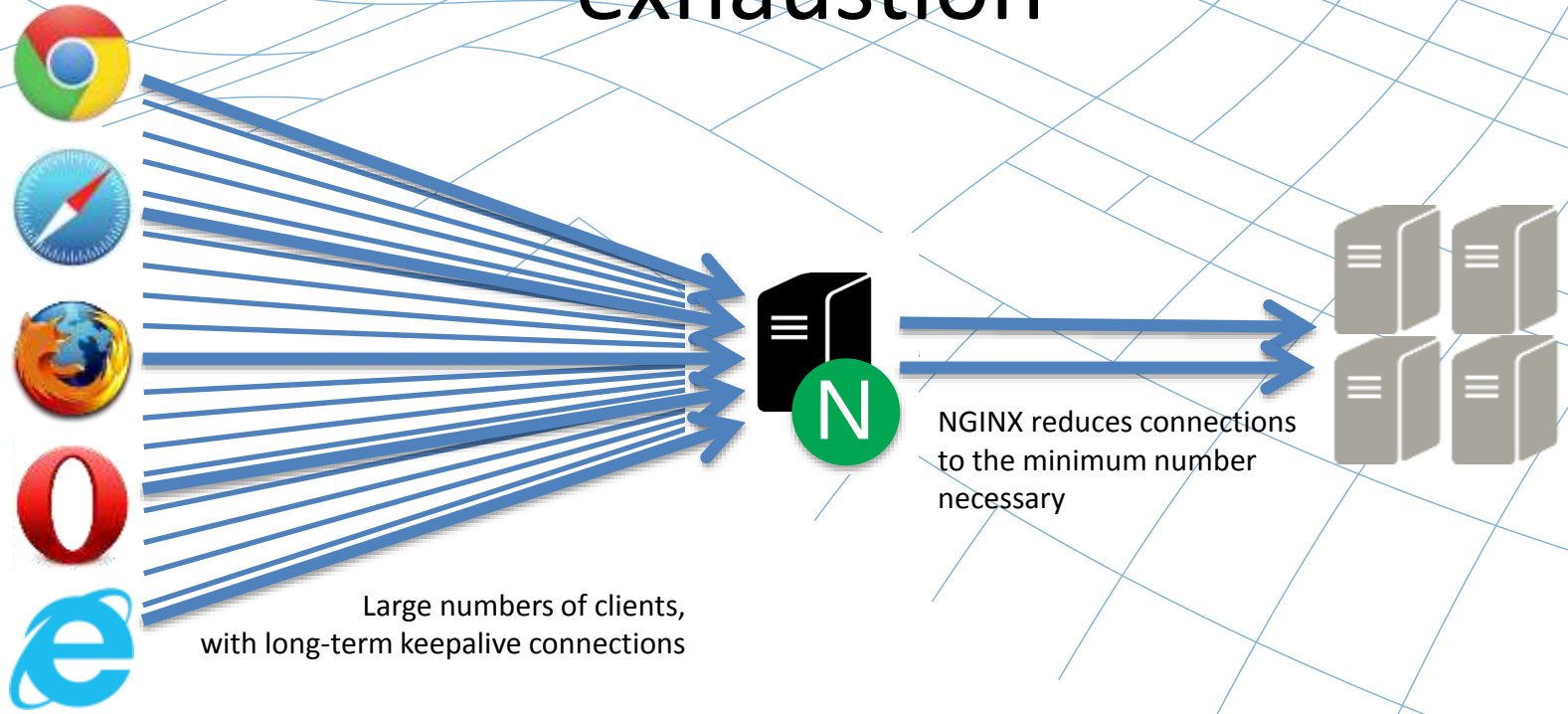
Multiple Connections

HTTP Keepalives

Server-side:

Limited concurrency

How NGINX mitigates thread exhaustion



4. Rewrite content inline

- Use the power of substitution to simplify updates
- Directives can be used in the http, server and location contexts
- Key directives
  - `sub_filter_once`
  - `sub_filter`
  - `sub_filter_types`

HTTP sub module example

```
location / {
        sub_filter_once off;
        sub_filter_types text/html;
        sub_filter “__copyright_date__” “2014”;
}
```

5. Online Binary updates and configuration changes

- Update either the configuration files or the binary without losing any connections

Configuration file update

```
[root@localhost ~]# nginx -s reload
[root@localhost ~]#
```

Yep. It’s that simple

Binary update

- Choose your method of binary installation
- Replace the binary

```
[root@localhost ~]# cat /var/run/nginx.pid
1991
[root@localhost ~]# kill -USR2 1991
```

Binary update

```
[root@localhost ~]# ps -ef |grep nginx
root      1991     1  0 08:06 ?        00:00:00 nginx: master
process /usr/sbin/nginx -c /etc/nginx/nginx.conf
nginx     2974  1991  0 08:22 ?        00:00:00 nginx: worker
process
nginx     2975  1991  0 08:22 ?        00:00:00 nginx: worker
process
root      3123  2948  0 08:43 pts/0    00:00:00 grep nginx
root      3124  1991  0 08:43 ?        00:00:00 nginx: master
process /usr/sbin/nginx -c /etc/nginx/nginx.conf
```

Binary update

```
[root@localhost ~]# kill -WINCH 1991
```

- Verify things are working as expected
  (you can still back out gracefully at this point)

```
[root@localhost ~]# kill -QUIT 1991
```

Bonus:

nginx –V gives a nearly complete configuration script for compiling

```
[root@localhost ~]# nginx -V
nginx version: nginx/1.5.7
built by gcc 4.4.7 20120313 (Red Hat 4.4.7-3) (GCC)
TLS SNI support enabled
configure arguments: --prefix=/etc/nginx/ --sbin-
path=/usr/sbin/nginx --conf-path=/etc/nginx/nginx.conf --error-
log-path=/var/log/nginx/error.log --http-log-
path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --
lock-path=/var/run/nginx.lock --http-client-body-temp-
path=/var/cache/nginx/client_temp --http-proxy-temp-
path=/var/cache/nginx/proxy_temp --http-fastcgi-temp-
path=/var/cache/nginx/fastcgi_temp --http-uwsgi-temp-
path=/var/cache/nginx/uwsgi_temp --http-scgi-temp-
path=/var/cache/nginx/scgi_temp --user=nginx --group=nginx --with-
http_ssl_module --with-http_spdy_module --with-http_realip_module
--with-http_addition_module --with-http_sub_module --with-
http_dav_module
```

--etc

More resources

- Check out our blog on nginx.com
- Webinars: nginx.com/webinars

Try NGINX F/OSS (nginx.org) or NGINX Plus (nginx.com)

Thanks for your time!

@sarahnovotny
Evangelist, NGINX
Program Chair, OSCON