Tempesta FW
Yet Another Web-accelerator?

Alexander Krizhanovsky

Tempesta Technologies, Inc.

ak@tempesta-tech.com

Who am I?

- CEO & CTO at **NatSys Lab** & **Tempesta Technologies**
- **Tempesta Technologies** *(Seattle, WA)*
  - Subsidiary of NatSys Lab. developing **Tempesta FW** – open source Linux *Application Delivery Controller (ADC)*
- **NatSys Lab** *(Moscow, Russia)*
  - Custom software development in:
    - high performance network traffic processing
    - databases

Tempesta Technologies

Challenges

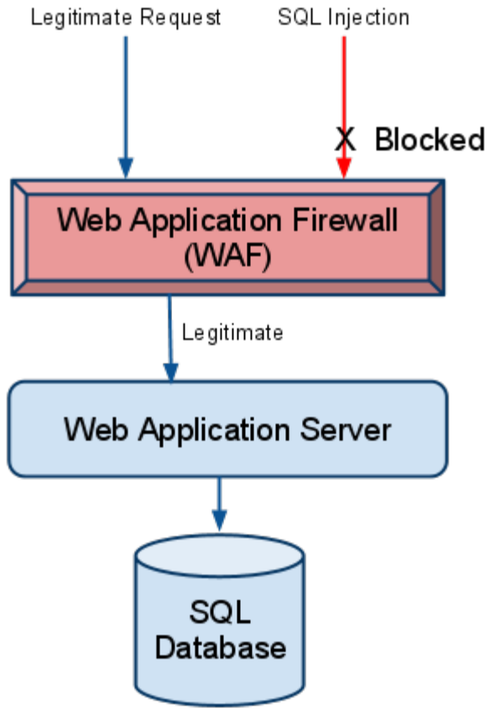
- Normal Web-servers deliver content

Challenges

- Normal Web-servers deliver content
- There are a lot of bad guys in modern Internet
- There are also good guys filtering bad guys out

Good Guys: WAF

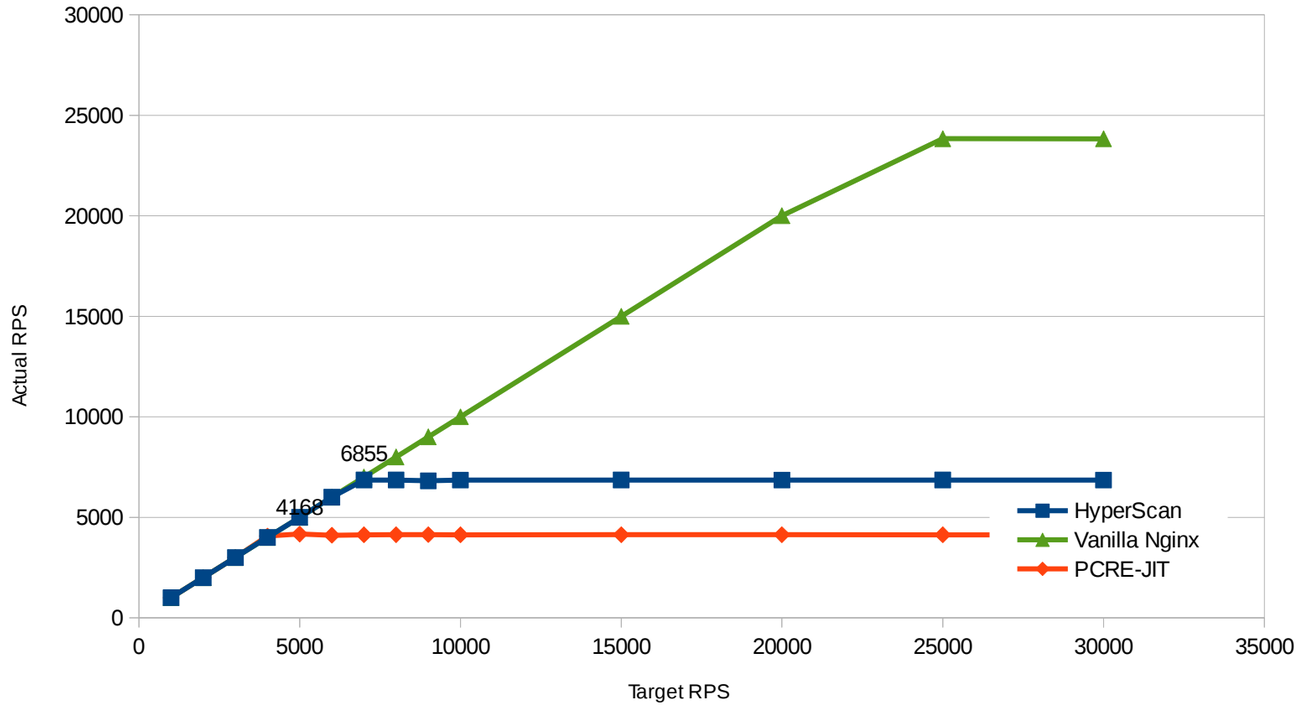
- **Technologies**: XHTML, WSDL, Machine learning, Regexps
- **Platforms**: Nginx, Apache Traffic Server etc.



Tempesta Technologies

WAF: Performance Issues

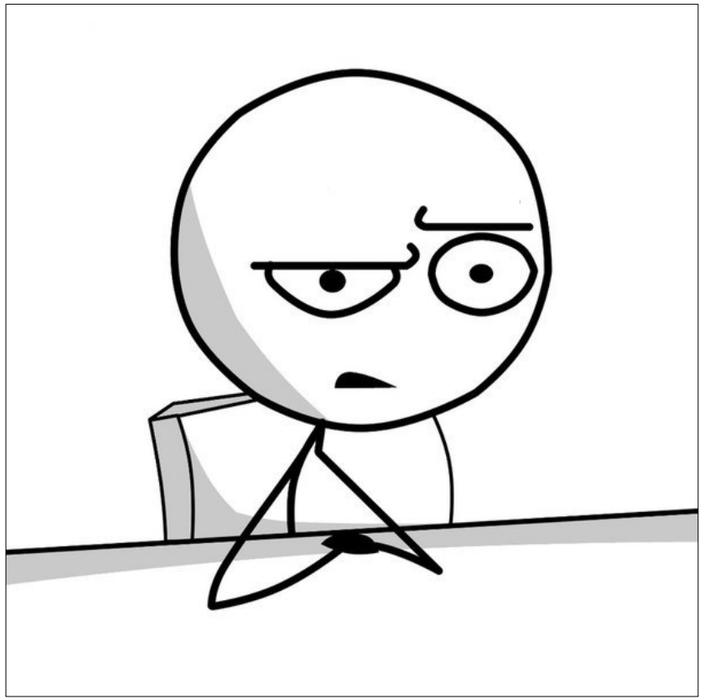
- Nginx + regexps



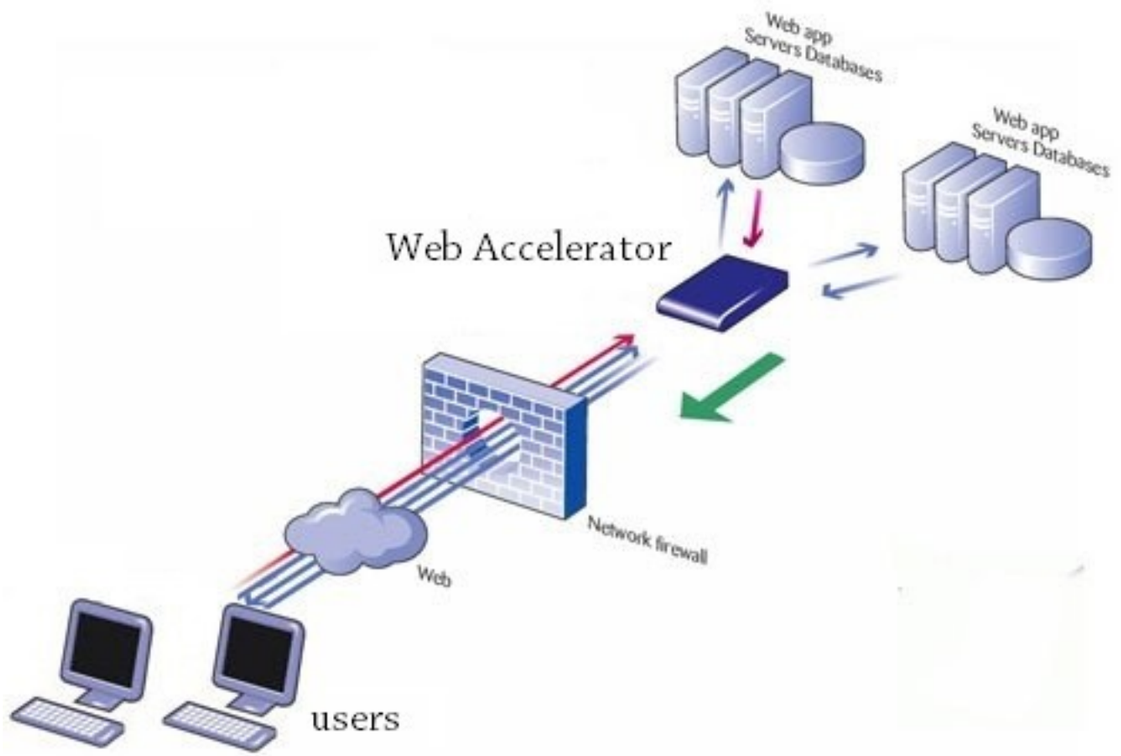
Tempesta Technologies

WAF: Performance Issues

- **15K** HTTP RPS on **24** cores *(but >100KRPS would be nice)*



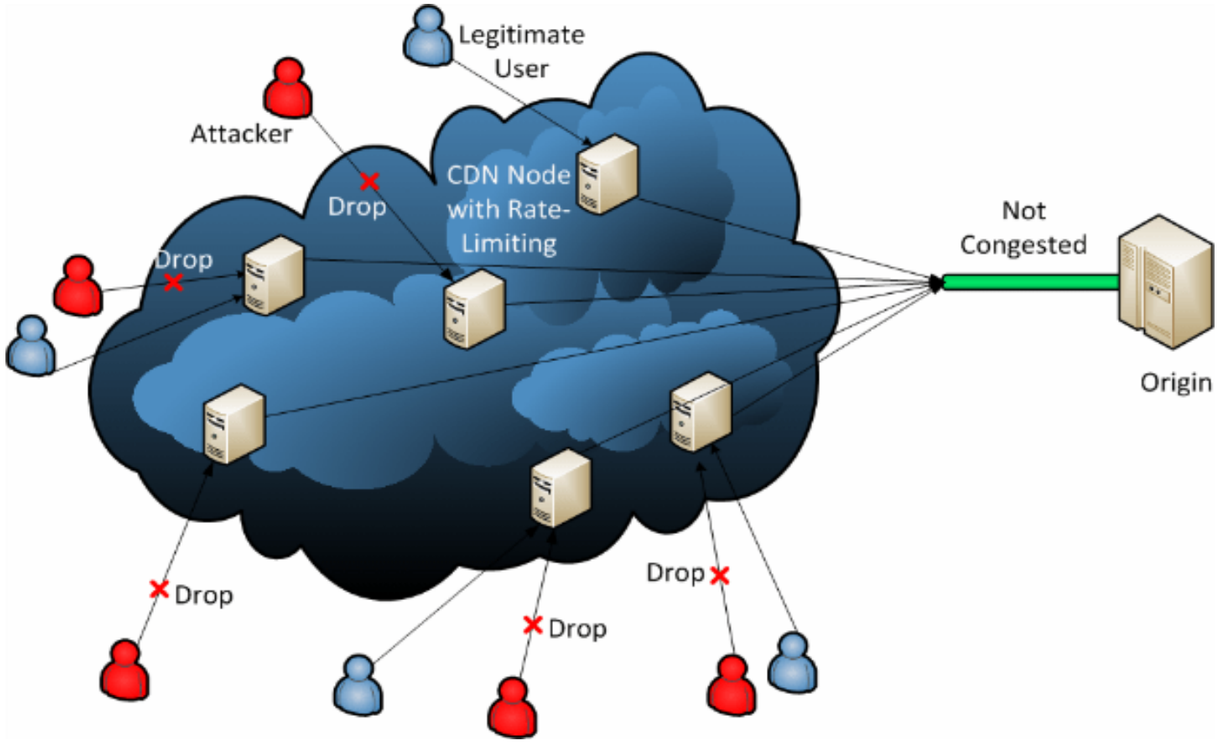
WAF: Acceleration Again



Tempesta
Technologies

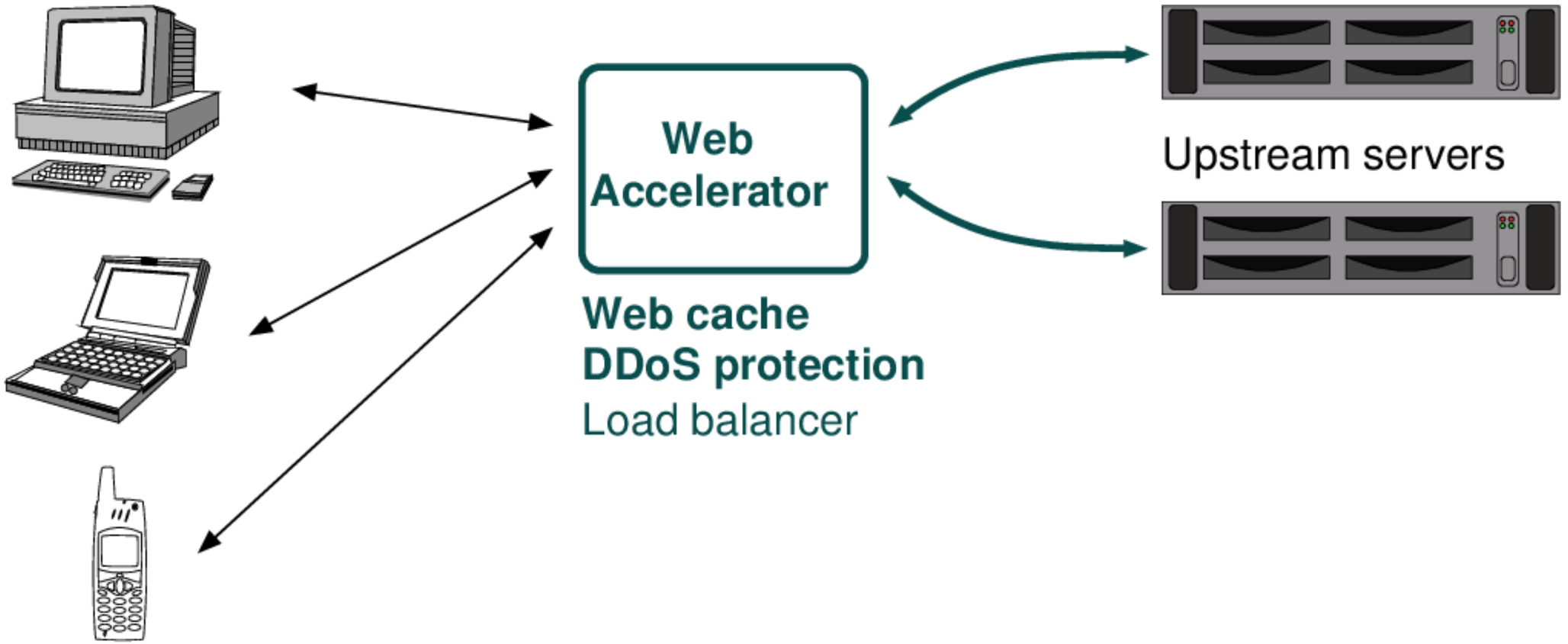
Good Guys: Anti-DDoS CDN

- **Technologies**: DPI or Firewall + Machine Learning
- **Platforms**: Nginx



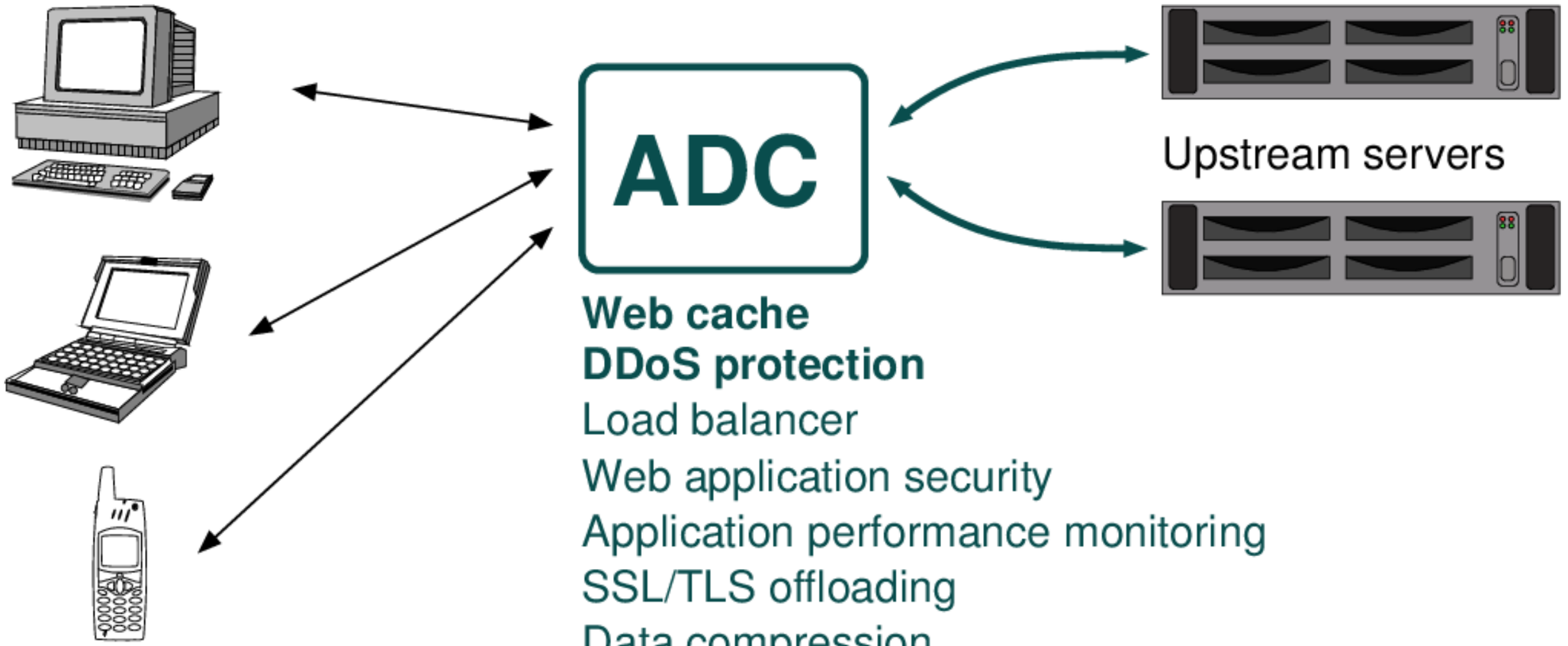
Tempesta Technologies

Anti-DDoS Web accelerator?



Tempesta
Technologies

Application Delivery Controller (ADC)



Web cache
DDoS protection
Load balancer
Web application security
Application performance monitoring
SSL/TLS offloading
Data compression
Connections QoS

Tempesta
Technologies

Application Layer DDoS

	Service from Cache	Rate limit
Nginx	22us	23us

Application Layer DDoS

	Service from Cache	Rate limit
Nginx	22us	23us

- **Fail2Ban**: write to the log, parse the log, write to the log, parse the log…

Application Layer DDoS

	Service from Cache	Rate limit
Nginx	22us	23us

- **Fail2Ban**: write to the *log*, parse the *log*, write to the *log*, parse the *log*… **- really in 21th century?!**

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, filtering etc.

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K**

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K –** *is it a problem for bot-net?*

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K –** *is it a problem for bot-net?* **SSL?**

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K –** *is it a problem for bot-net?* **SSL?**
  - what about tons of `'GET / HTTP/1.0\n\n'`?

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K –** *is it a problem for bot-net?* **SSL?**
  - what about tons of `'GET / HTTP/1.0\n\n'`?
- **Kernel-mode Web-accelerators**: TUX, kHTTPd

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K –** *is it a problem for bot-net?* **SSL?**
  - what about tons of `'GET / HTTP/1.0\n\n'`?
- **Kernel-mode Web-accelerators**: TUX, kHTTPd
  - basically the same sockets and threads

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K –** *is it a problem for bot-net?* **SSL?**
  - what about tons of `'GET / HTTP/1.0\n\n'`?
- **Kernel-mode Web-accelerators**: TUX, kHTTPd
  - basically the same sockets and threads
  - zero-copy → *sendfile()*

Web-accelerator Capabilities

- Nginx, Varnish, Apache Traffic Server, Squid, Apache HTTPD etc.
  - cache static Web-content
  - load balancing
  - rewrite URLs, ACL, Geo, **filtering?** etc.
  - **C10K –** *is it a problem for bot-net?* **SSL?**
  - what about tons of `'GET / HTTP/1.0\n\n'`?
- **Kernel-mode Web-accelerators**: TUX, kHTTPd
  - basically the same sockets and threads
  - zero-copy → *sendfile()* - not needed

Web-accelerators are Slow

```
    %                symbol name
1.5719          ngx_http_parse_header_line
1.0303          ngx_vslprintf
0.6401          memcpy
0.5807          recv
0.5156          ngx_linux_sendfile_chain
0.4990          ngx_http_limit_req_handler
```

+ довольно длинный хвост

Web-accelerators are Slow: syscalls

```
epoll_wait(.., {{EPOLLIN, ....}},...)
recvfrom(3, "GET / HTTP/1.1\r\nHost:...", ...)
write(1, “...limiting requests, excess...", ...)
writev(3, "HTTP/1.1 503 Service...", ...)
sendfile(3,..., 383)
recvfrom(3, ...) = -1 EAGAIN
epoll_wait(.., {{EPOLLIN, ....}}, ...)
recvfrom(3, "", 1024, 0, NULL, NULL) = 0
close(3)
```

Web-accelerators are Slow: HTTP parser

Start: *state = 1, *str_ptr = 'b'*

```
        while (++str_ptr) {
            switch (state) { <= check state
            case 1:
                switch (*str_ptr) {
                case 'a':
                    ...
                    state = 1
                case 'b':
                    ...
                    state = 2
                }
            case 2:
                ...
            }
            ...
        }
```

Web-accelerators are Slow: HTTP parser

Start: *state = 1, *str_ptr = 'b'*

```
        while (++str_ptr) {
            switch (state) {
            case 1:
                switch (*str_ptr) {
                case 'a':
                    ...
                    state = 1
                case 'b':
                    ...
                    state = 2 <= set state
                }
            case 2:
                ...
            }
            ...
        }
```

Tempesta Technologies

Web-accelerators are Slow: HTTP parser

Start: *state = 1, *str_ptr = 'b'*

```
while (++str_ptr) {
    switch (state) {
    case 1:
        switch (*str_ptr) {
        case 'a':
            ...
            state = 1
        case 'b':
            ...
            state = 2
        }
    case 2:
        ...
    }
    ... <= jump to while
}
```

Tempesta Technologies

Web-accelerators are Slow: HTTP parser

Start: *state = 1, *str_ptr = 'b'*

```
        while (++str_ptr) {
            switch (state) { <= check state
            case 1:
                switch (*str_ptr) {
                case 'a':
                    ...
                    state = 1
                case 'b':
                    ...
                    state = 2
                }
            case 2:
                ...
            }
            ...
        }
```

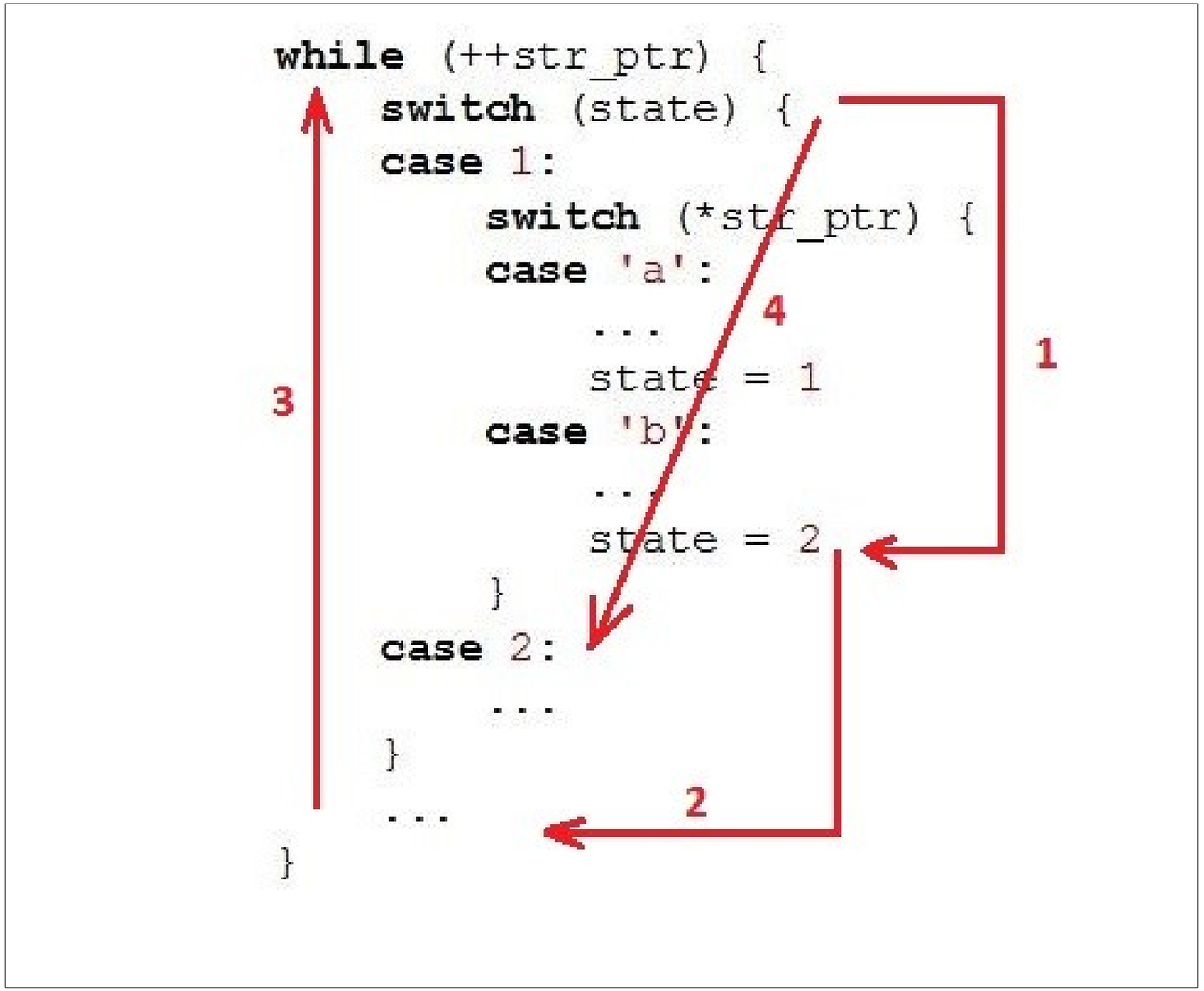
Web-accelerators are Slow: HTTP parser

Start: *state = 1, *str_ptr = 'b'*

```
        while (++str_ptr) {
            switch (state) {
            case 1:
                switch (*str_ptr) {
                case 'a':
                    ...
                    state = 1
                case 'b':
                    ...
                    state = 2
                }
            case 2:
                ...   <= do something
            }
            ...
        }
```

Tempesta Technologies

Web-accelerators are Slow: HTTP parser

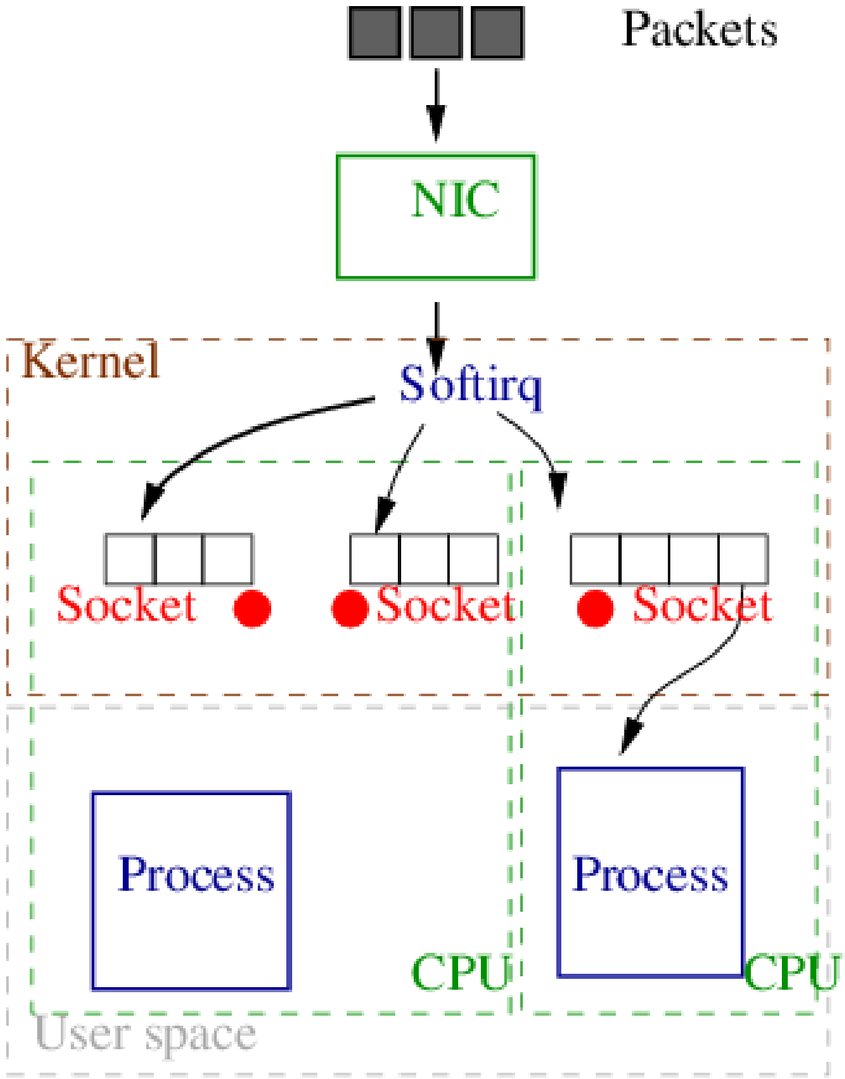


Tempesta Technologies

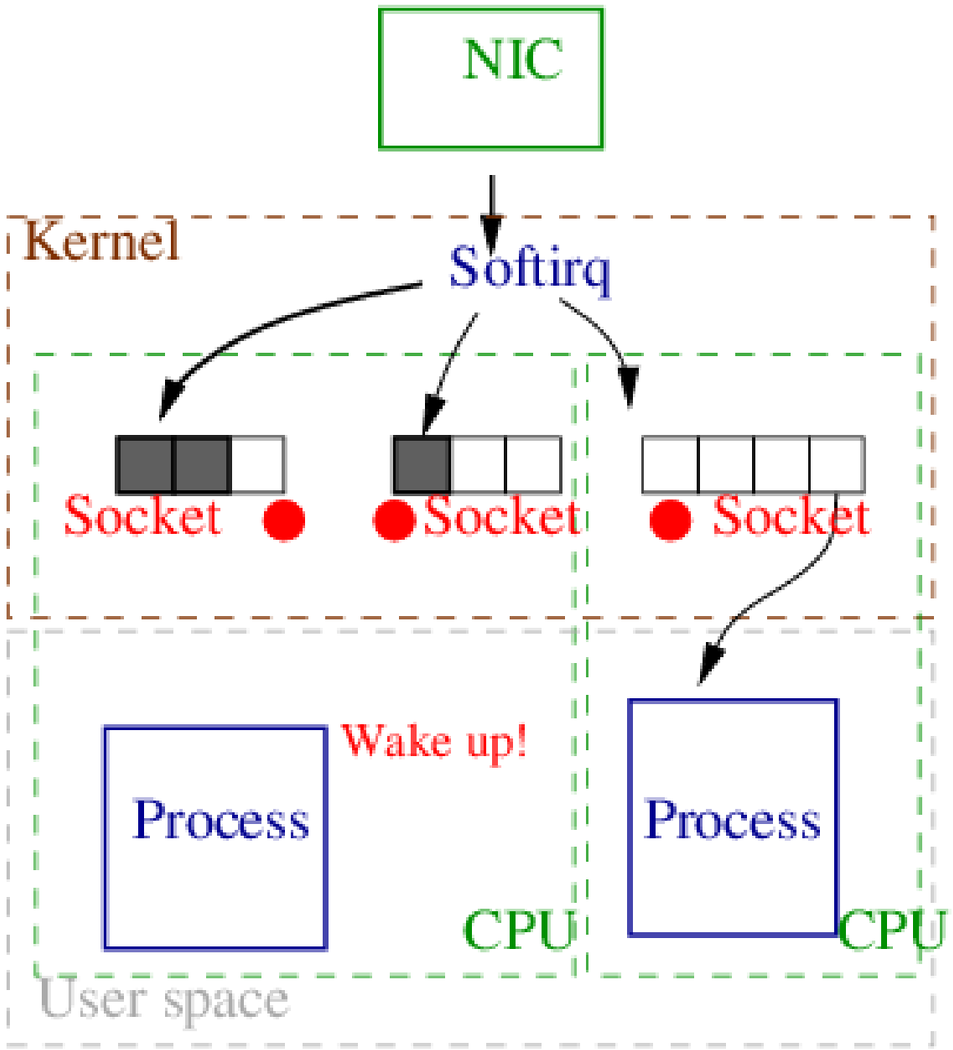
Web-accelerators are Slow: strings!

- We have AVX2, but GLIBC doesn't still use it
- HTTP strings are special:
  - No `'\0'`-terminatin (if you're zero-copy)
  - Special delimiters (`':'` or `CRLF`)
  - `strcasecmp()`: no need case conversion for one string
  - `strspn()`: limited number of accepted alphabets
- `switch()`-driven FSM is even worse
- **7x** performance improvement:
  http://natsys-lab.blogspot.ru/2016/10/http-strings-processing-using-c-sse42.html

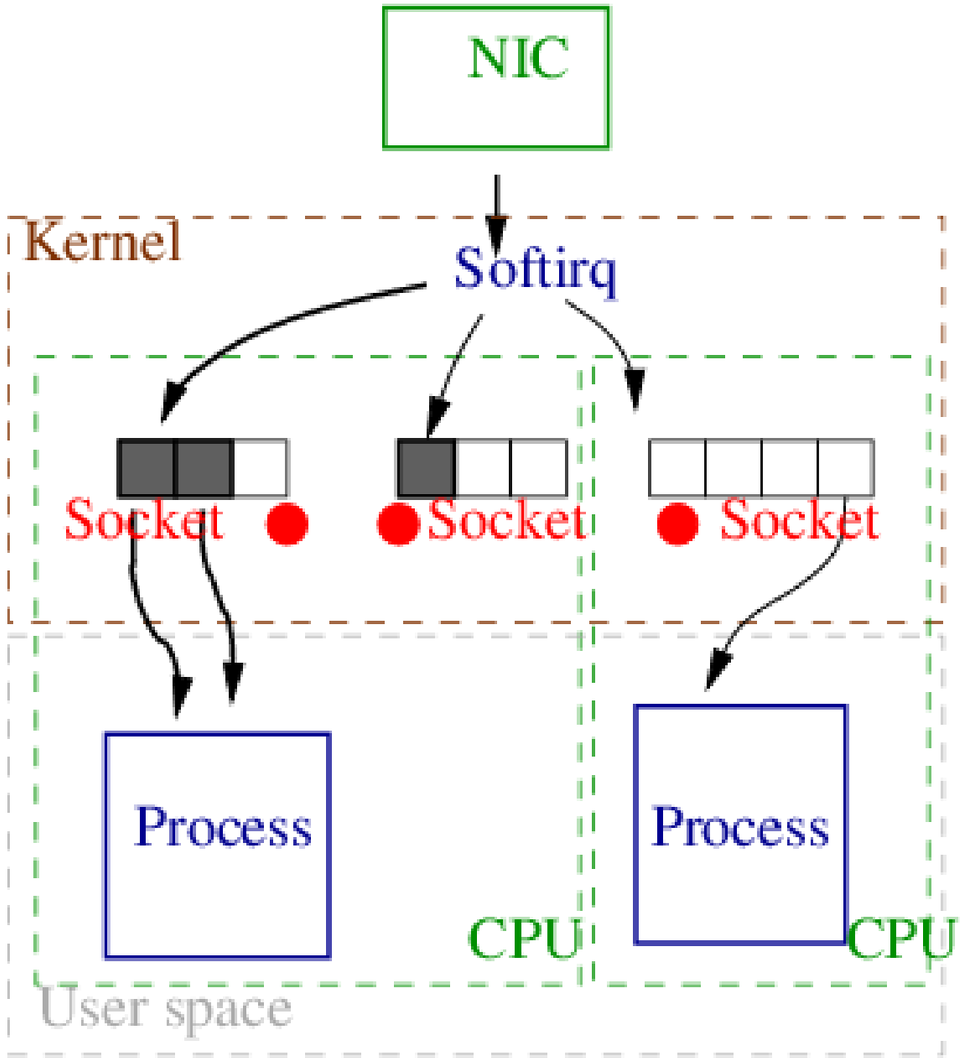
Web-accelerators are Slow: async I/O



Web-accelerators are Slow: async I/O

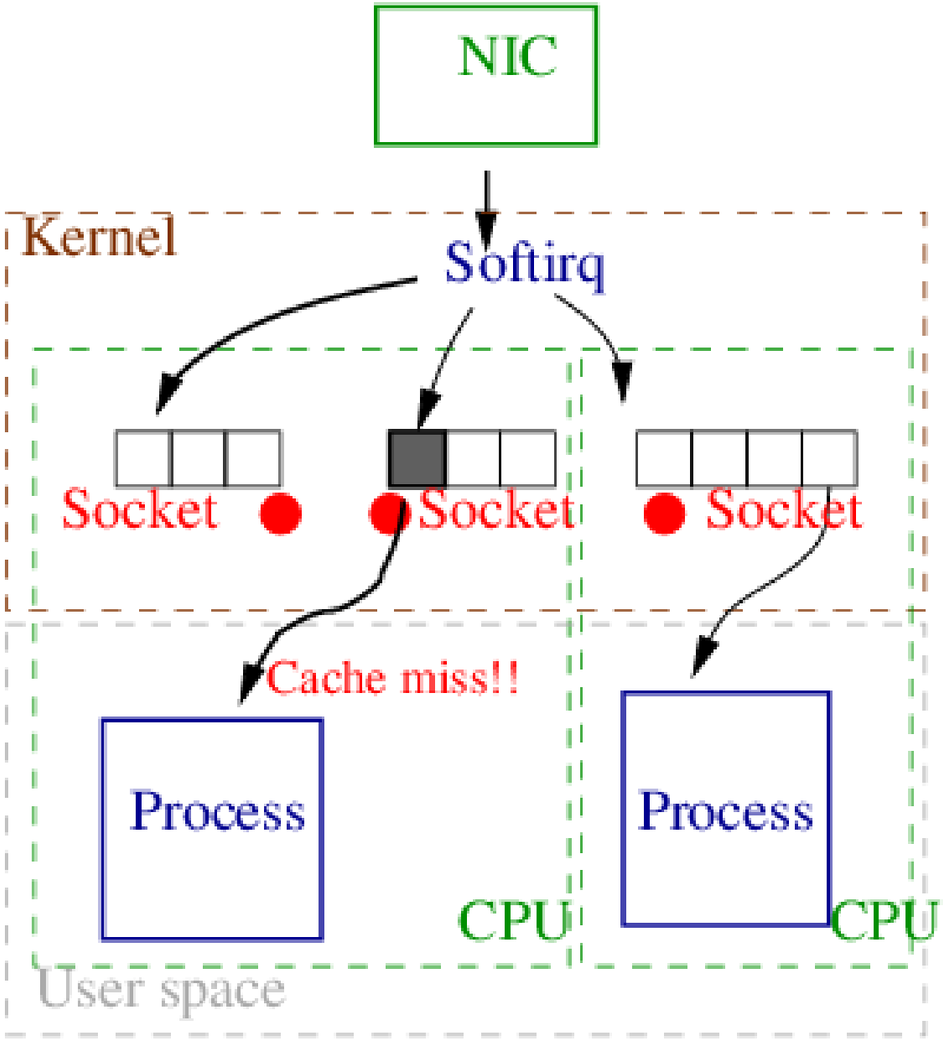


Web-accelerators are Slow: async I/O



Tempesta Technologies

Web-accelerators are Slow: async I/O



Tempesta Technologies

Tempesta FW

- First and only **hybrid** of HTTP accelerator and ***F***ire***W***all

Tempesta FW

- First and only **hybrid** of HTTP accelerator and ***F***ire***W***all
- ***F***ire***W***all: layer 3 (IP) – layer 7 (HTTP) filter

Tempesta FW

- First and only **hybrid** of HTTP accelerator and ***F***ire***W***all
- ***F***ire***W***all: layer 3 (IP) – layer 7 (HTTP) filter
- ***F***rame***W***ork: high performance and flexible platform to build intelligent DDoS mitigation systems and Web Application Firewalls (WAF)

Tempesta FW

- First and only **hybrid** of HTTP accelerator and ***F***ire***W***all
- ***F***ire***W***all: layer 3 (IP) – layer 7 (HTTP) filter
- ***F***rame***W***ork: high performance and flexible platform to build intelligent DDoS mitigation systems and Web Application Firewalls (WAF)
- **Directly embedded** into Linux TCP/IP stack

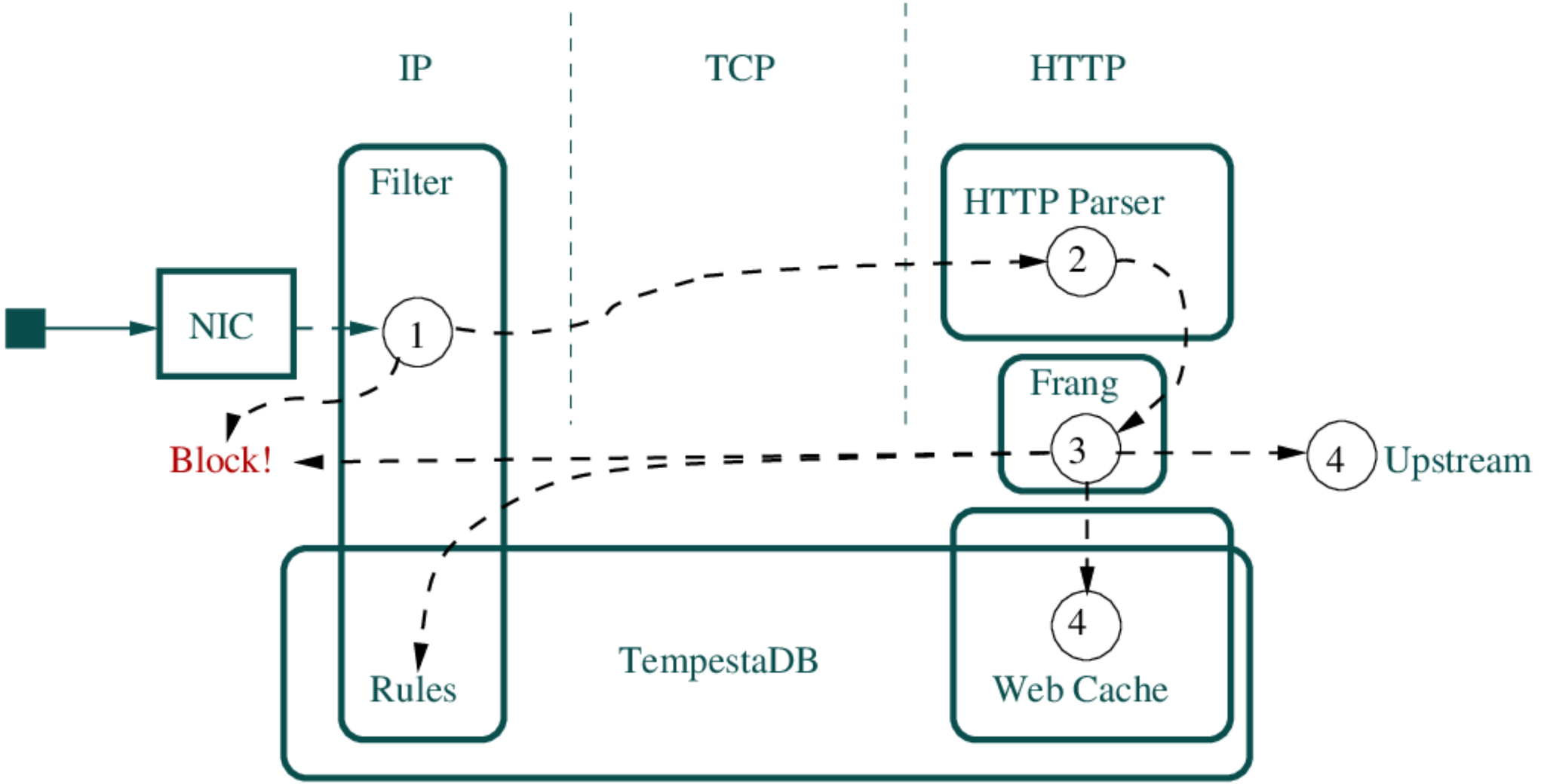
Tempesta FW

- First and only **hybrid** of HTTP accelerator and ***F***ire***W***all
- ***F***ire***W***all: layer 3 (IP) – layer 7 (HTTP) filter
- ***F***rame***W***ork: high performance and flexible platform to build intelligent DDoS mitigation systems and Web Application Firewalls (WAF)
- **Directly embedded** into Linux TCP/IP stack
- **NUMA-aware** x86-64 **cache conscious** Web-cache on **huge pages**

Tempesta FW

- First and only **hybrid** of HTTP accelerator and ***F***ire***W***all
- ***F***ire***W***all: layer 3 (IP) – layer 7 (HTTP) filter
- ***F***rame***W***ork: high performance and flexible platform to build intelligent DDoS mitigation systems and Web Application Firewalls (WAF)
- **Directly embedded** into Linux TCP/IP stack
- **NUMA-aware** x86-64 **cache conscious** Web-cache on **huge pages**
- This is **Open Source** (GPLv2)

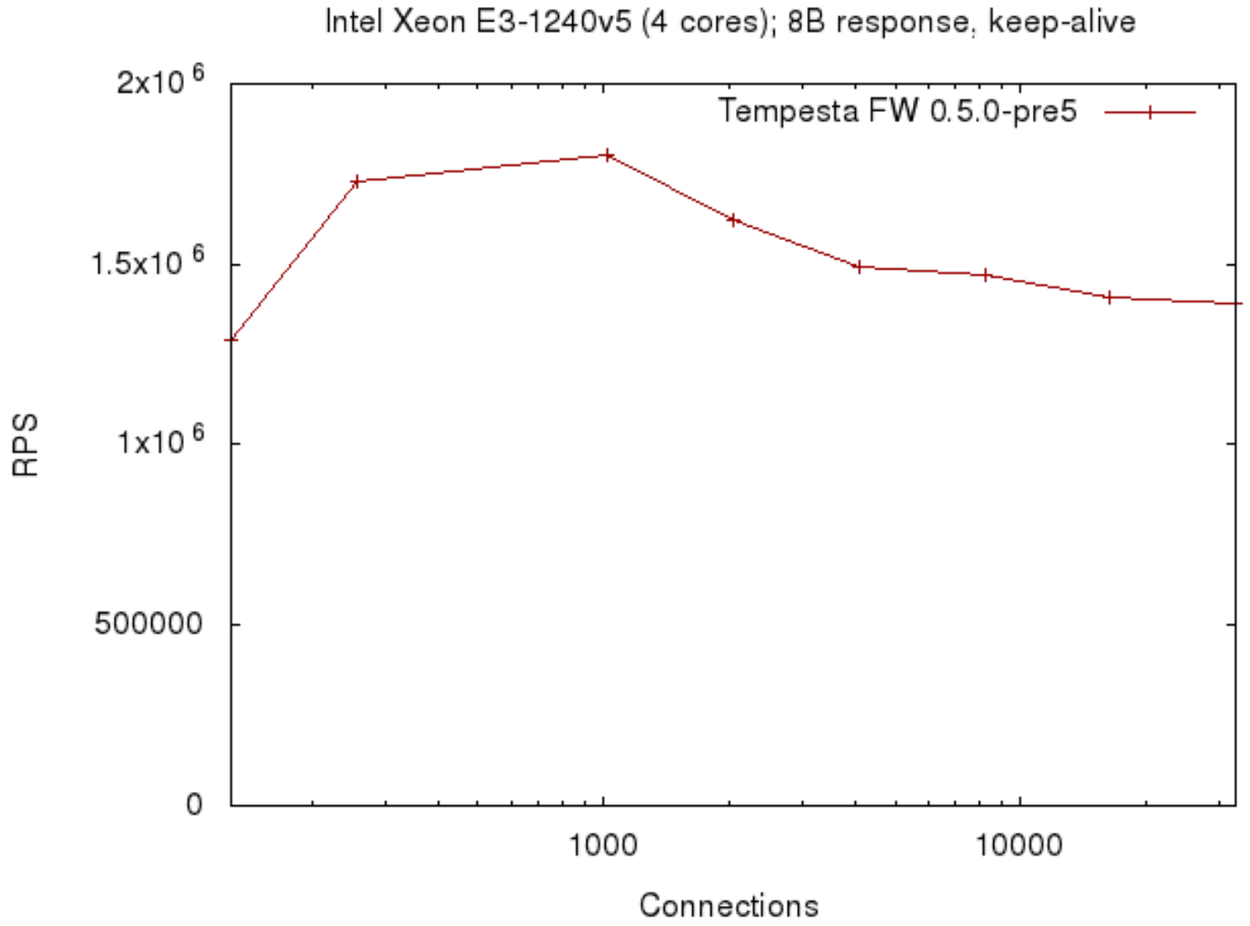
Tempesta FW



Tempesta Technologies

Performance

https://github.com/tempesta-tech/tempesta/wiki/Tempesta-FW-benchmark



Tempesta Technologies

Frang: HTTP DoS

- **Rate limits**
  - request_rate, request_burst
  - connection_rate, connection_burst
  - concurrent_connections

Frang: HTTP DoS

- **Rate limits**
  - request_rate, request_burst
  - connection_rate, connection_burst
  - concurrent_connections
- **Slow HTTP**
  - client_header_timeout, client_body_timeout
  - http_header_cnt
  - http_header_chunk_cnt, http_body_chunk_cnt

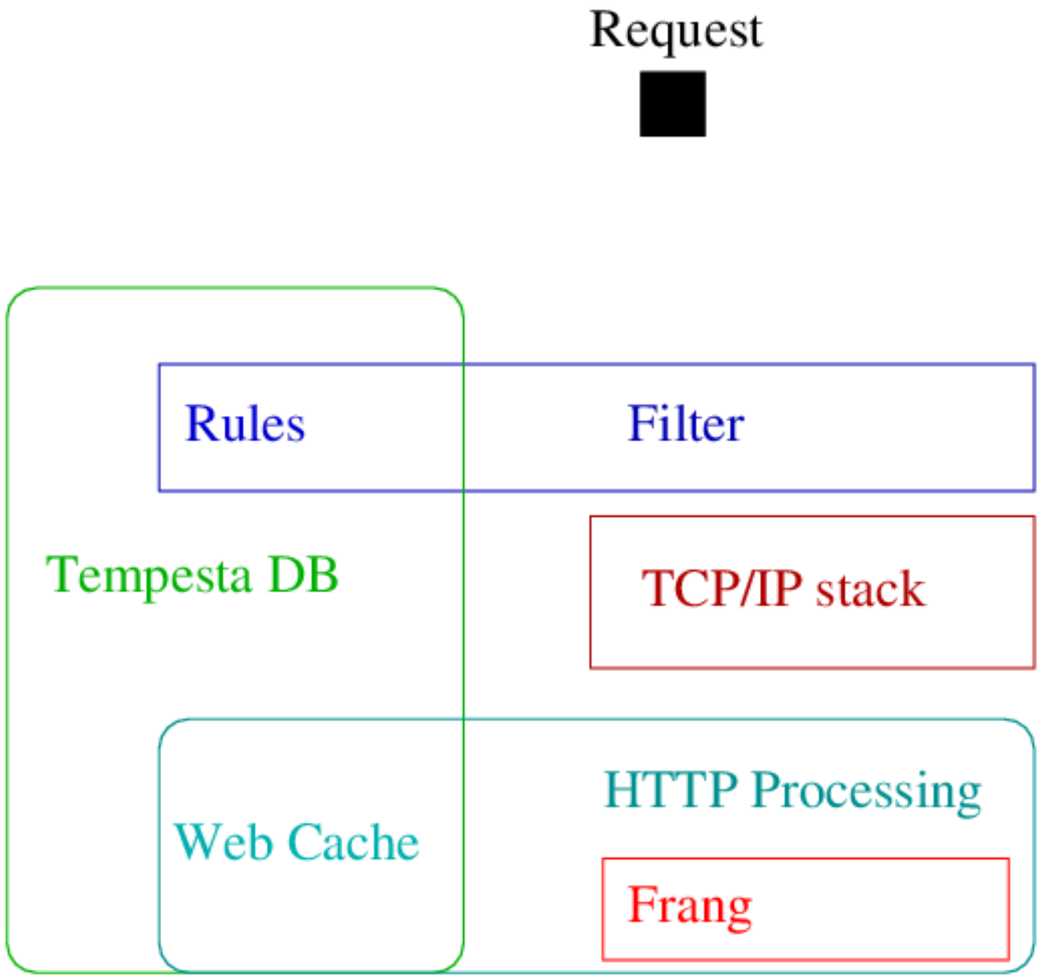
Frang: WAF

- **Length limits**
  - http_uri_len
  - http_field_len
  - http_body_len

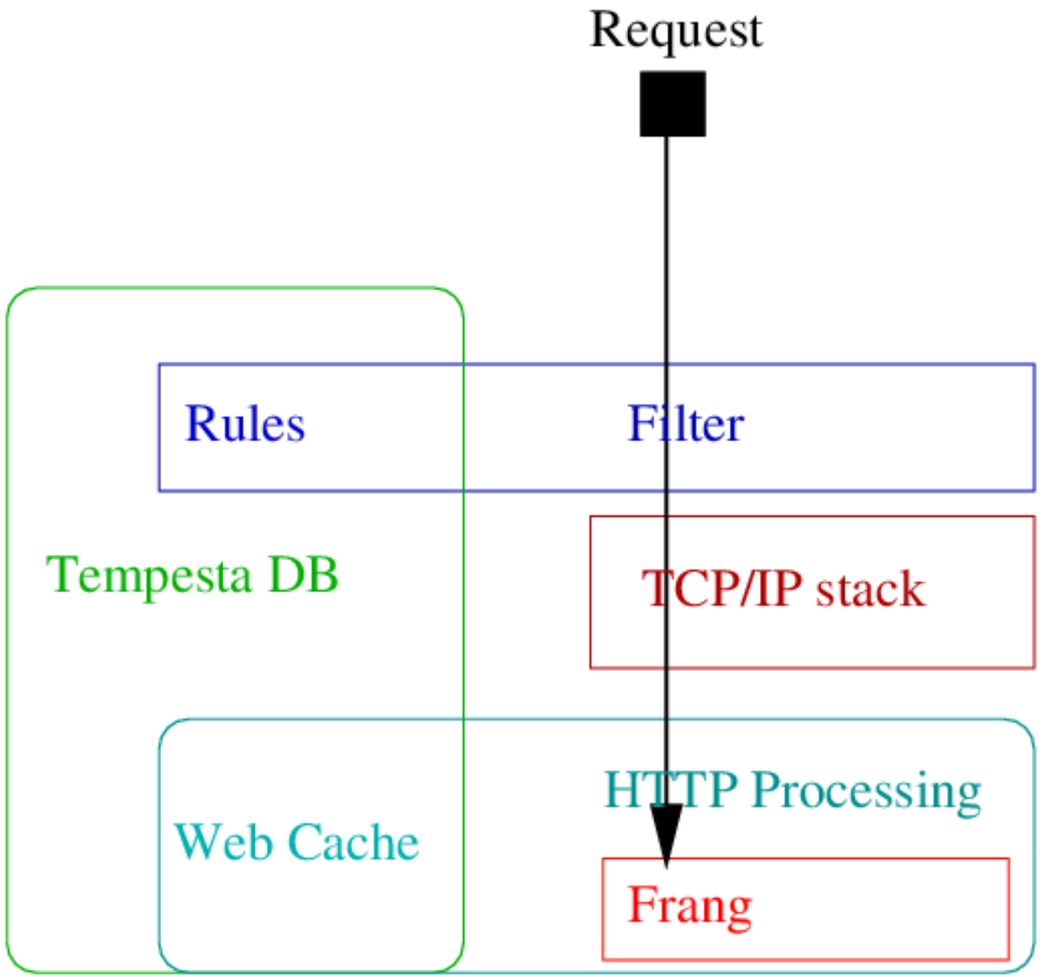
Frang: WAF

- **Length limits**
  - http_uri_len
  - http_field_len
  - http_body_len
- **Content validation**
  - http_ct_required
  - http_ct_vals
  - http_methods

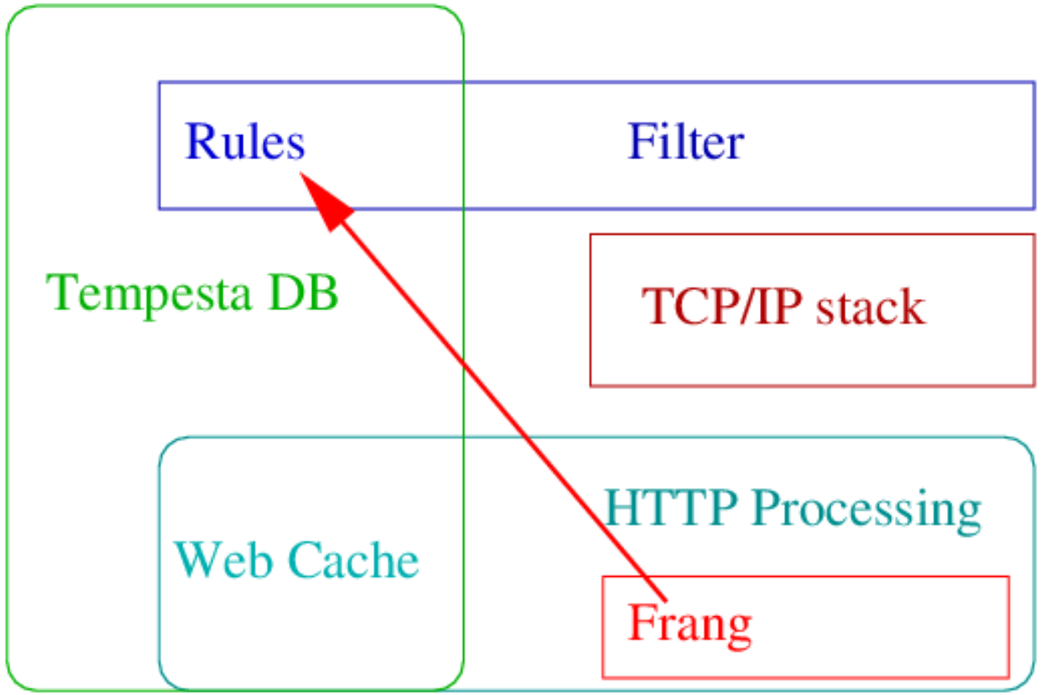
Frang: Filtering



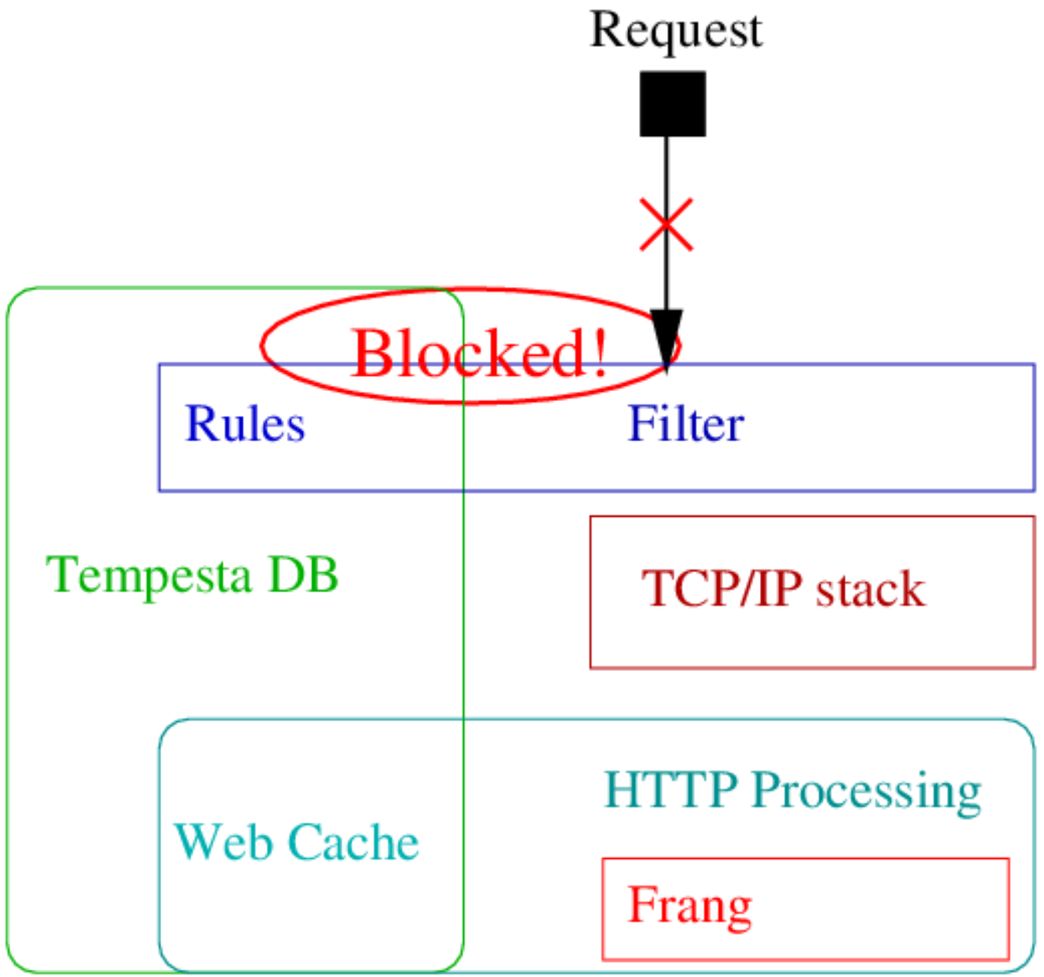
Frang: Filtering



Frang: Filtering



Frang: Filtering



Load Balancing

- Dynamic reconnections

Load Balancing

- Dynamic reconnections
- Configurable number of upstream connections

Load Balancing

- Dynamic reconnections
- Configurable number of upstream connections
- Schedulers
  - HTTP (server groups)
    - Wildcards, full match, prefix
    - Method, URI, Host & other headers

Load Balancing

- Dynamic reconnections
- Configurable number of upstream connections
- Schedulers
  - HTTP (server groups)
    - Wildcards, full match, prefix
    - Method, URI, Host & other headers
  - Round-Robin (inside server group)

Load Balancing

- Dynamic reconnections
- Configurable number of upstream connections
- Schedulers
  - HTTP (server groups)
    - Wildcards, full match, prefix
    - Method, URI, Host & other headers
  - Round-Robin (inside server group)
  - Rendezvous hashing (inside server group)

Load Balancing

- Dynamic reconnections
- Configurable number of upstream connections
- Schedulers
  - HTTP (server groups)
    - Wildcards, full match, prefix
    - Method, URI, Host & other headers
  - Round-Robin (inside server group)
  - Rendezvous hashing (inside server group)
  - **Adaptive & predictive** load balancing

Configuration Example

```
srv_group static { # sched=round-robin
    server 10.10.0.1:8080;
    server [fc00::2]:8081;
}

srv_group dynamic sched=hash {
    server 10.10.0.3:8080; # conns_n = 4
    server [fc00::4]:8081 conns_n=8;
}

srv_group black_hole { }

sched_http_rules {
    match black_hole  hdr_raw   prefix  "X-Bad:";
    match static      uri       prefix  "/static/";
    match dynamic     *         *       *;
}
```

Sticky Cookie

- User identification
- **Enforce:** HTTP 302 redirect

```
sticky name=__tfw_user_id__ enforce;
```

HowTo

Preprequisites:

- Haswell: AVX2, SSE 4.2 (“`avx2`”, “`sse4_2`” in `/proc/cpuinfo`)

HowTo

Preprequisites:

- Haswell: AVX2, SSE 4.2 (“`avx2`”, “`sse4_2`” in `/proc/cpuinfo`)
- Huge pages (“`pse`” in `/proc/cpuinfo`)

HowTo

Preprequisites:

- Haswell: AVX2, SSE 4.2 (“`avx2`”, “`sse4_2`” in `/proc/cpuinfo`)
- Huge pages (“`pse`” in `/proc/cpuinfo`)
- Custom Linux kernel (KVM or dedicated server)

HowTo

```
$ git clone https://github.com/tempesta-tech/linux-4.1-tfw.git
```

HowTo

```
$ git clone https://github.com/tempesta-tech/linux-4.1-tfw.git
$ cd linux-4.1-tfw
$ make && make modules && make modules_install && make install
$ reboot
```

HowTo

```
$ git clone https://github.com/natsys/tempesta.git
```

HowTo

```
$ git clone https://github.com/natsys/tempesta.git
$ cd tempesta && make
```

HowTo

```
$ git clone https://github.com/natsys/tempesta.git
$ cd tempesta && make
$ cat > etc/tempesta_fw.conf
  server 127.0.0.1:8080;  # upstream
  cache 1;                # cache sharding
  ^D
```

HowTo

```
$ git clone https://github.com/natsys/tempesta.git
$ cd tempesta && make
$ cat > etc/tempesta_fw.conf
  server 127.0.0.1:8080;  # upstream
  cache 1;                # cache sharding
  ^D
$ ./scripts/tempesta.sh --start
```

Thanks!

- Web-site: *http://tempesta-tech.com* ***(Powered by Tempesta FW)***
- Availability: *https://github.com/tempesta-tech/tempesta*
- Blog: *http://natsys-lab.blogspot.com*
- E-mail: *ak@tempesta-tech.com*

We are hiring!

Tempesta Technologies