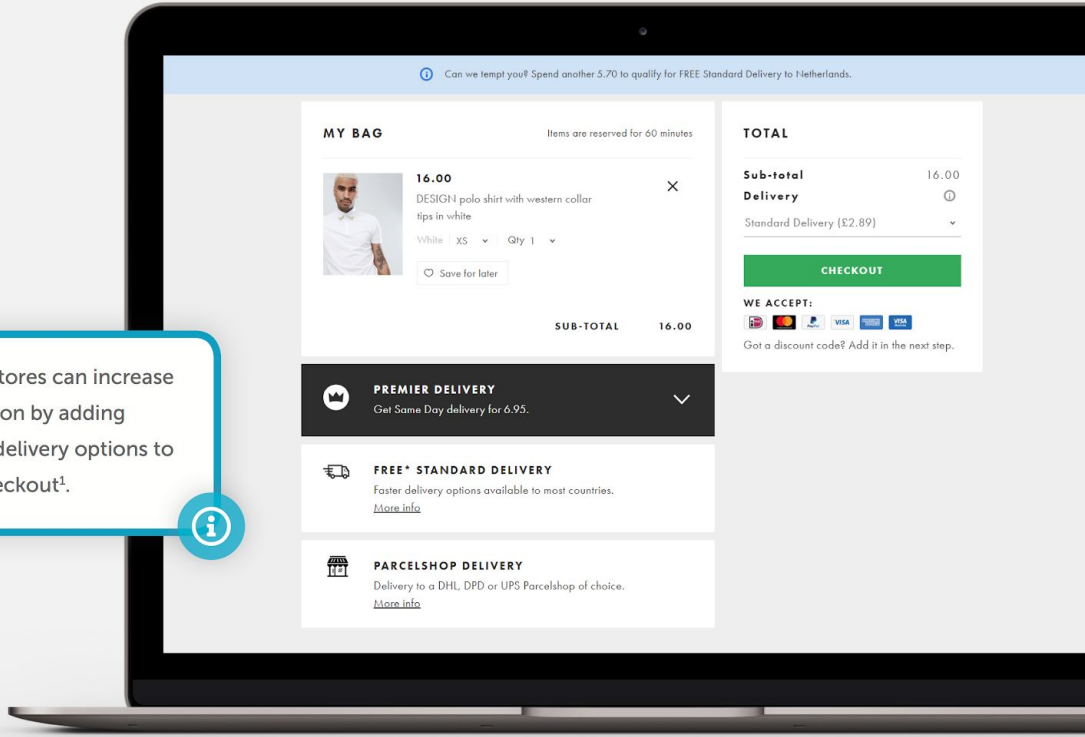
Scaling blocking APIs for WSGI frameworks

Product - Shipping checkout integration

We created an all-in-one shipping solution based on five key elements:

1. Checkout
2. Picking
3. Shipping
4. Notify
5. Returns

Online stores can increase conversion by adding flexible delivery options to their checkout[1].

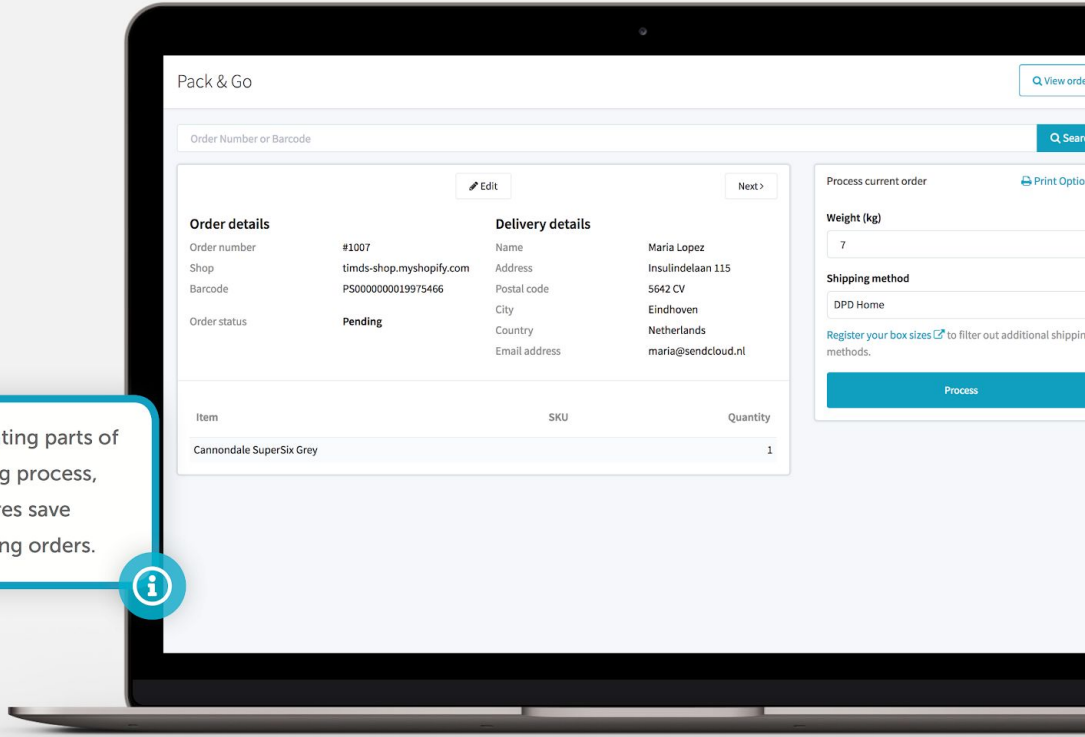


[1] AlixPartners | Research E-commerce Cart Abandonment

Product - Order processing

We created an all-in-one shipping solution based on five key elements:

1. Checkout
2. Picking
3. Shipping
4. Notify
5. Returns

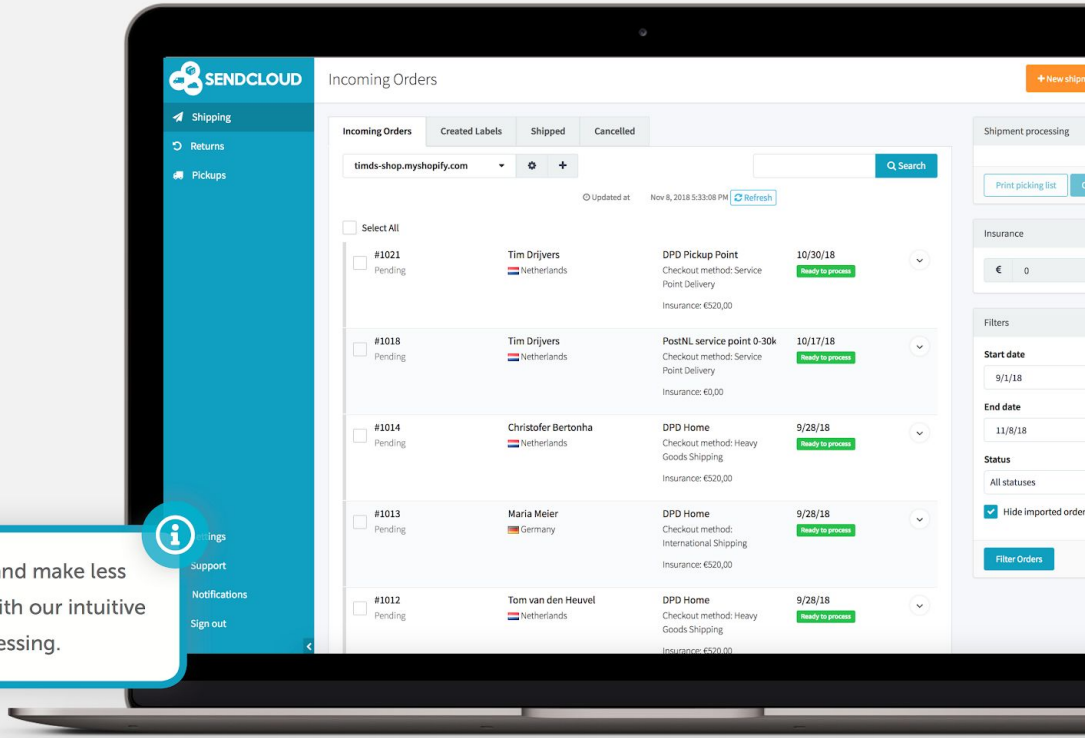


By automating parts of the packing process, online stores save time packing orders.

Product - Shipping

We created an all-in-one shipping solution based on five key elements:

1. Checkout
2. Picking
3. Shipping
4. Notify
5. Returns



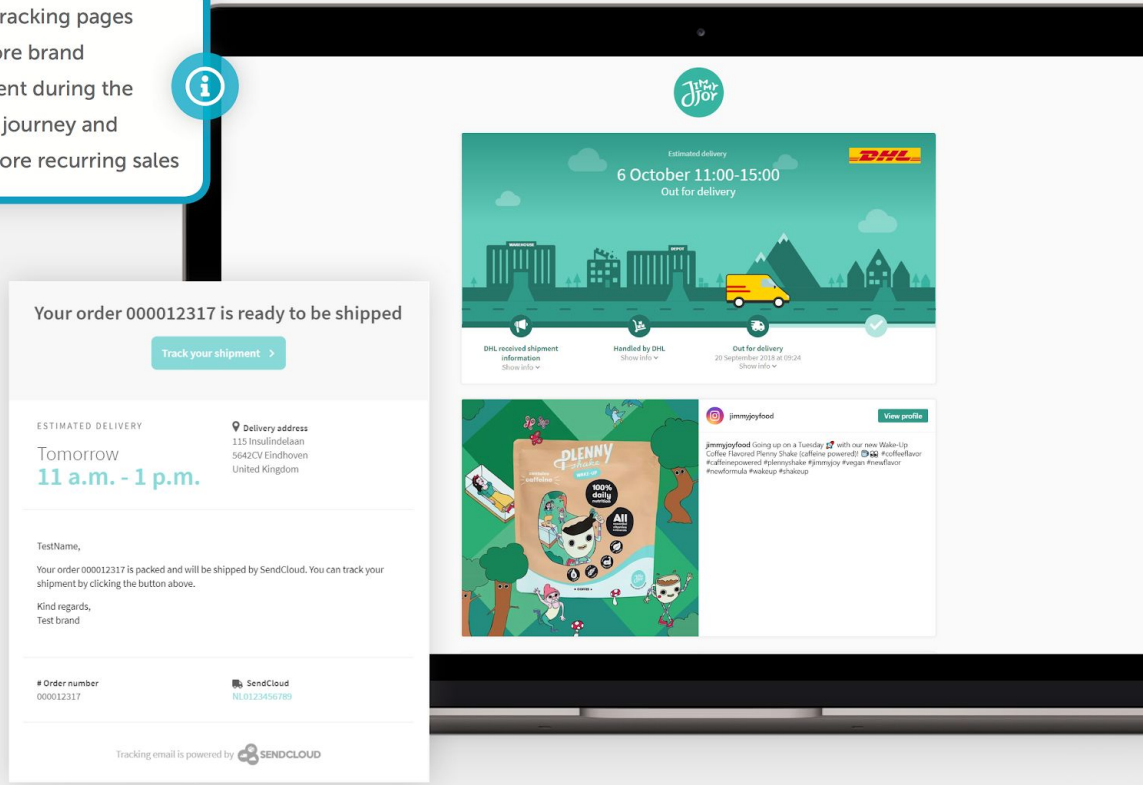
Save time and make less mistakes with our intuitive order processing.

Product - Customer communication

We created an all-in-one shipping solution based on five key elements:

1. Checkout
2. Picking
3. Shipping
4. Notify
5. Returns

Branded tracking pages create more brand engagement during the customer journey and creates more recurring sales



Your order 000012317 is ready to be shipped

Track your shipment

ESTIMATED DELIVERY

Tomorrow
11 a.m. - 1 p.m.

Delivery address
115 Insulindelaan
5642CV Eindhoven
United Kingdom

TestName,

Your order 000012317 is packed and will be shipped by SendCloud. You can track your shipment by clicking the button above.

Kind regards,
Test brand

Order number
000012317

SendCloud
NL0123456789

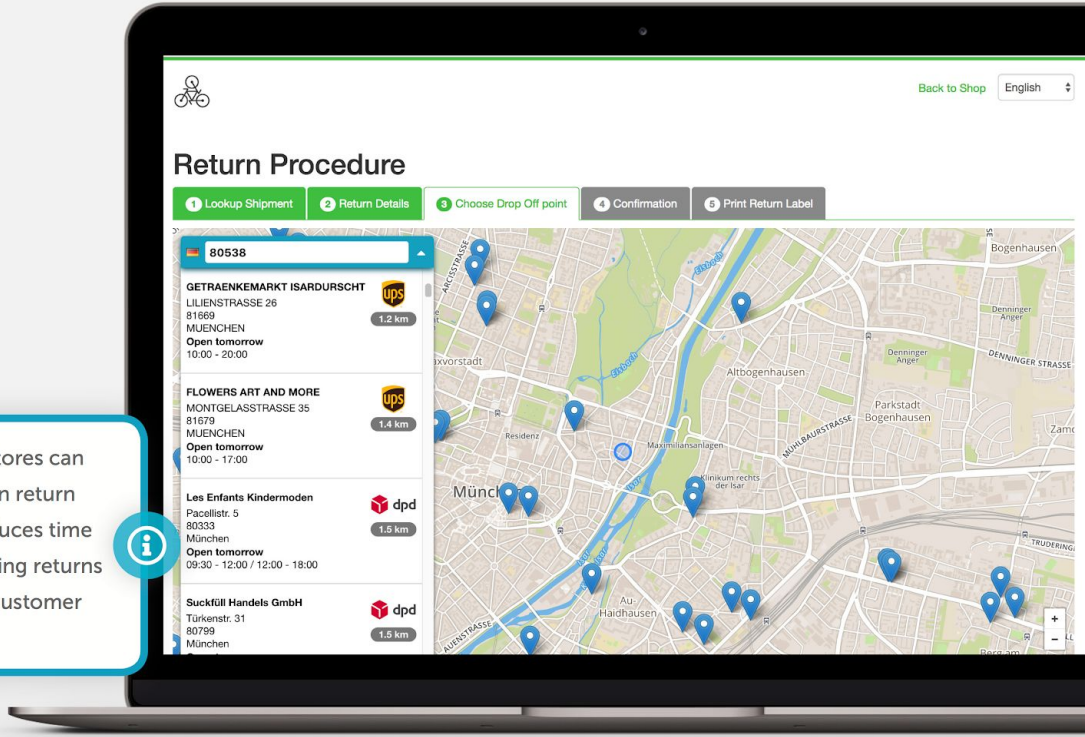
Tracking email is powered by SENDCLOUD

Product - International Returns

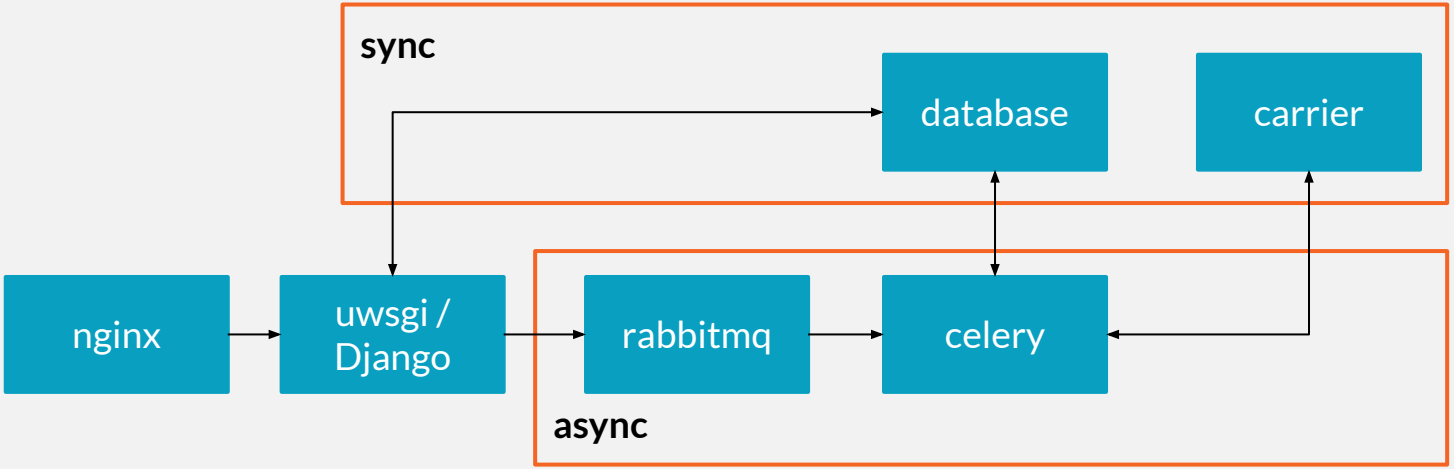
We created an all-in-one shipping solution based on five key elements:

1. Checkout
2. Picking
3. Shipping
4. Notify
5. Returns

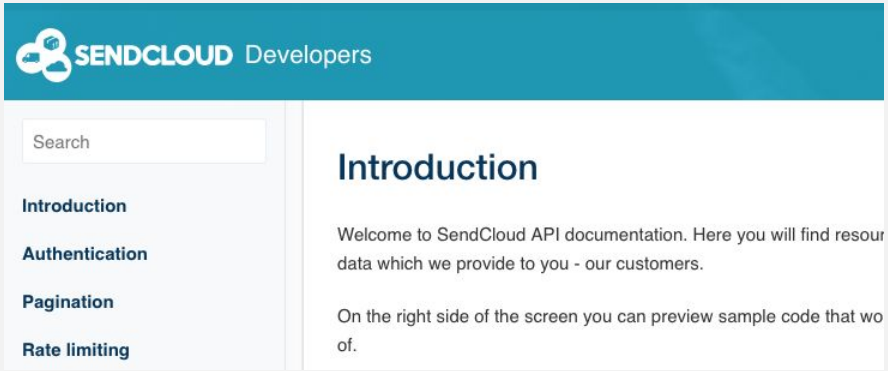
E-commerce stores can set up their own return portal. This reduces time spent on handling returns and improves customer satisfaction.



Architecture



SendCloud's Achilles' heel



- Blocking calls
- Synchronous announcement required by some shop integrations / legacy
- Performance directly coupled to carrier's performance

Announcement

request_label `boolean`	No	Should the parcel request a label. *This property used to be called `requestLabel`. We kept it backwards compatible by still accepting the previous name.*
request_label_async `boolean`	No	Makes sure that the label is requested asynchronously. The parcel is returned, but without label. You will need to poll for status changes on the parcel.

The december 2017 hack

- Scale with machines
- AWS ELB

The 2018 hack

- gevent

Onwards

Gevent is blocking us to upgrade

- Python 3.6 requires new Gevent
- Celery workers are hanging (4.2.x known bug, 4.3?)
- Compatibility issues

Other options

Why not a proxy?

- Replication of business logic
- Both sync and async announcements would pass through

Rewrite of the API

Test setup

5 processes

Demo project + example code:
https://github.com/timdrijvers/offload-blocking-requests

Profiling tool: Baton
https://github.com/americanexpress/baton

- 200 open connections
- 2000 requests

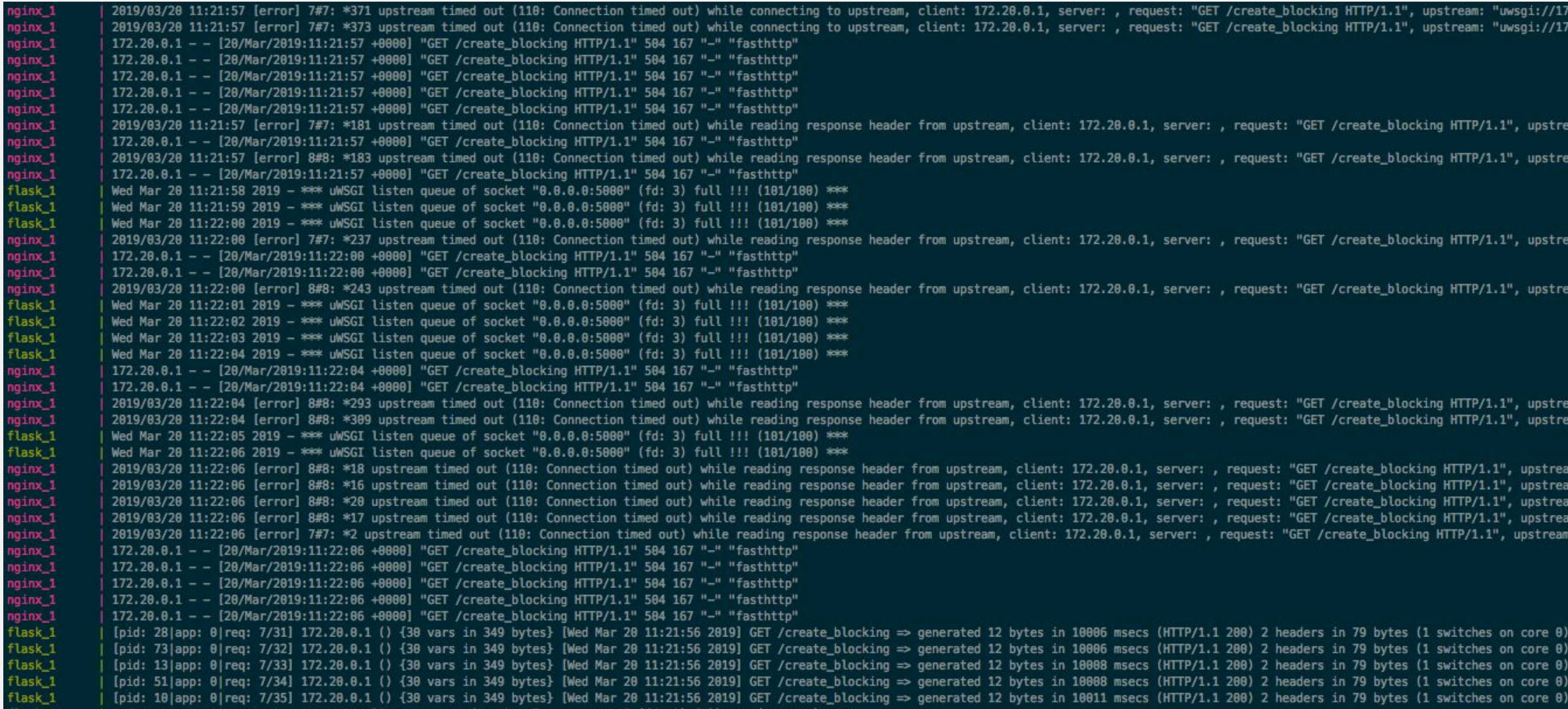
Test setup


```
DURATION = 10

@app.route('/create_blocking')
  def create_blocking():
      time.sleep(DURATION)
      return 'Hello world!'
```

Base case - test results

```
Time taken to complete requests:     10m31.510501781s

Requests per second:                                3


Number of connection errors:                       60

Number of 1xx responses:                            0

Number of 2xx responses:                           25

Number of 3xx responses:                            0

Number of 4xx responses:                            0

Number of 5xx responses:                         1915
```

Base case - test results



```
nginx_1      | 2019/03/20 11:21:57 [error] 7#7: *371 upstream timed out (110: Connection timed out) while connecting to upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstream: "uwsgi://17
nginx_1      | 2019/03/20 11:21:57 [error] 7#7: *373 upstream timed out (110: Connection timed out) while connecting to upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstream: "uwsgi://17
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:21:57 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:21:57 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:21:57 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:21:57 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:21:57 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 2019/03/20 11:21:57 [error] 7#7: *181 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstre
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:21:57 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 2019/03/20 11:21:57 [error] 8#8: *183 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstre
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:21:57 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
flask_1      | Wed Mar 20 11:21:58 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
flask_1      | Wed Mar 20 11:21:59 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
flask_1      | Wed Mar 20 11:22:00 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
nginx_1      | 2019/03/20 11:22:00 [error] 7#7: *237 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstre
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:00 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:00 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 2019/03/20 11:22:00 [error] 8#8: *243 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstre
flask_1      | Wed Mar 20 11:22:01 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
flask_1      | Wed Mar 20 11:22:02 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
flask_1      | Wed Mar 20 11:22:03 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
flask_1      | Wed Mar 20 11:22:04 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:04 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:04 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 2019/03/20 11:22:04 [error] 8#8: *293 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstre
nginx_1      | 2019/03/20 11:22:04 [error] 8#8: *309 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstre
flask_1      | Wed Mar 20 11:22:05 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
flask_1      | Wed Mar 20 11:22:06 2019 - *** uWSGI listen queue of socket "0.0.0.0:5000" (fd: 3) full !!! (101/100) ***
nginx_1      | 2019/03/20 11:22:06 [error] 8#8: *18 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstrea
nginx_1      | 2019/03/20 11:22:06 [error] 8#8: *16 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstrea
nginx_1      | 2019/03/20 11:22:06 [error] 8#8: *20 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstrea
nginx_1      | 2019/03/20 11:22:06 [error] 8#8: *17 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstrea
nginx_1      | 2019/03/20 11:22:06 [error] 7#7: *2 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 172.20.0.1, server: , request: "GET /create_blocking HTTP/1.1", upstream
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:06 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:06 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:06 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:06 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
nginx_1      | 172.20.0.1 - - [20/Mar/2019:11:22:06 +0000] "GET /create_blocking HTTP/1.1" 504 167 "-" "fasthttp"
flask_1      | [pid: 28|app: 0|req: 7/31] 172.20.0.1 () {30 vars in 349 bytes} [Wed Mar 20 11:21:56 2019] GET /create_blocking => generated 12 bytes in 10006 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches on core 0)
flask_1      | [pid: 73|app: 0|req: 7/32] 172.20.0.1 () {30 vars in 349 bytes} [Wed Mar 20 11:21:56 2019] GET /create_blocking => generated 12 bytes in 10006 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches on core 0)
flask_1      | [pid: 13|app: 0|req: 7/33] 172.20.0.1 () {30 vars in 349 bytes} [Wed Mar 20 11:21:56 2019] GET /create_blocking => generated 12 bytes in 10008 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches on core 0)
flask_1      | [pid: 51|app: 0|req: 7/34] 172.20.0.1 () {30 vars in 349 bytes} [Wed Mar 20 11:21:56 2019] GET /create_blocking => generated 12 bytes in 10008 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches on core 0)
flask_1      | [pid: 10|app: 0|req: 7/35] 172.20.0.1 () {30 vars in 349 bytes} [Wed Mar 20 11:21:56 2019] GET /create_blocking => generated 12 bytes in 10011 msecs (HTTP/1.1 200) 2 headers in 79 bytes (1 switches on core 0)
```

Gevent 5 uwsgi processes / 10 gevent

```
Time taken to complete requests:       6m1.756349245s

Requests per second:                                6


Number of connection errors:                        0

Number of 1xx responses:                            0

Number of 2xx responses:                         1708

Number of 3xx responses:                            0

Number of 4xx responses:                            0

Number of 5xx responses:                          252
```

Gevent 10 uwsgi processes / 10 gevent

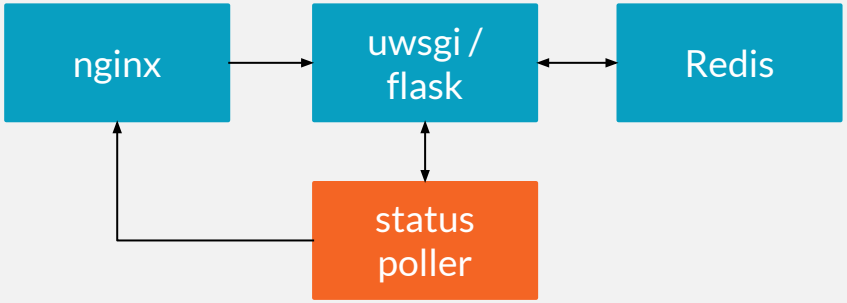
```
Time taken to complete requests:      3m23.359332252s

Requests per second:                              10


Number of connection errors:                       0

Number of 1xx responses:                           0

Number of 2xx responses:                        2000

Number of 3xx responses:                           0

Number of 4xx responses:                           0

Number of 5xx responses:                           0
```

Uwsgi offloading

- Threads to offload tiny tasks
- Increased concurrency
- Non-blocking
  - Serve big files
  - Internal routing

- https://uwsgi-docs.readthedocs.io/en/latest/OffloadSubsystem.html

Test setup



uwsgi configuration

```
offload-threads = 100

response-route-if = equal: ${OFFLOAD_TO_POLLER};y disableheaders:

response-route-if = equal: ${OFFLOAD_TO_POLLER};y
http:$(POLLER_IP):5500,,/poll/ ${POLLER_ID}
```

uwsgi/Flask - Python bindings

```
@app.route('/create_uwsgi')

def create_uwsgi():

    new_id = Data.new()

    uwsgi.add_var("OFFLOAD_TO_POLLER", "y")

    uwsgi.add_var("POLLER_ID", str(new_id))

    return make_response('id: {}'.format(new_id), 200, {})
```

uwsgi/Flask - poll endpoint

```
@app.route('/poll/<int:poll_id>/<call_try>', methods=['GET', 'POST'])

def poll(poll_id, call_try):

    ttl = Data.get(poll_id)

    if ttl is None:
        return ('', 404)
    elif time.time() - ttl < DURATION:
        return ('ttl: {}'.format(time.time() - ttl), 400)
    else:
        return 'Poll response {}'.format(poll_id)
```

asyncio Starlette

```
@app.route('/poll/{id}')

async def single_poller(request):

    status, text = await poll(request.path_params['id'])

    return Response(text, status)
```

asyncio Starlette

```
async def poll(poll_id):
    max_end_time = datetime.now() + POLL_TIMEOUT
    while datetime.now() < max_end_time:
        async with aiohttp.ClientSession(timeout=conn_timeout) as session:
            status, text = await fetch(session, poll_id)
            if 200 <= status < 300:
                return status, text
        await asyncio.sleep(POLL_INTERVAL)
    return 423, 'failed'
```

asyncio Starlette

```
async def fetch(session, poll_id):

    url = 'http://nginx/poll/{}/0'.format(poll_id)

    async with session.get(url) as response:

        return response.status, await response.text()
```

uwsgi offloading stats

```
Time taken to complete requests:      1m53.147095702s

Requests per second:                              18


Number of connection errors:                       0

Number of 1xx responses:                           0

Number of 2xx responses:                        2000

Number of 3xx responses:                           0

Number of 4xx responses:                           0

Number of 5xx responses:                           0
```

uwsgi issues

https://github.com/unbit/uwsgi/issues/1894

Using `max-worker-lifetime` kills workers without waiting for pending requests to finish #1894

Open cipriancraciun opened this issue on 14 Oct 2018 · 2 comments

Nginx njs offloading

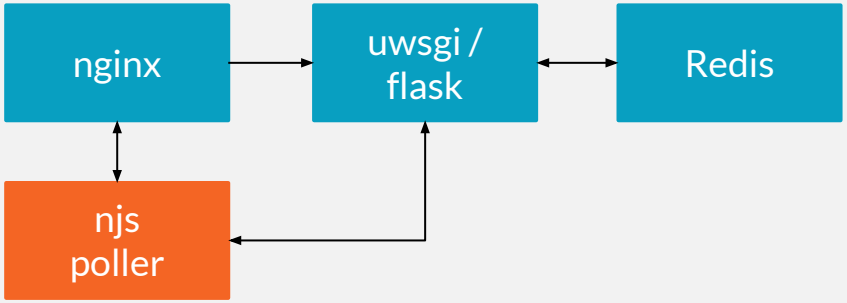
http://nginx.org/en/docs/njs/index.html

“njs is a subset of the JavaScript language that allows extending nginx functionality”

Use cases

- Complex access control and security checks in njs before a request reaches an upstream server
- Manipulating response headers
- Writing flexible asynchronous content handlers and filters

Test setup



nginx offload - Flask

```
@app.route('/create_nginx', methods=['GET', 'POST'])
def create_nginx():
    new_id = Data.new()
    return make_response(
        'id: {}'.format(new_id),
        418,
        {
            'X-OffloadToPoller-Url': '/poll',
            'X-OffloadToPoller-Args': str(new_id)
        }
    )
```

nginx offload - nginx config

```
location @app {
    uwsgi_intercept_errors  on;
    error_page 418 = @handle_redirect;

    include uwsgi_params;
    uwsgi_pass flask:5000;
}

location @handle_redirect {
    subrequest_output_buffer_size 0;
    set $offload_args '$upstream_http_x_offloadtopoller_args';
    set $offload_url '$upstream_http_x_offloadtopoller_url';
    js_content offloader;
}
```

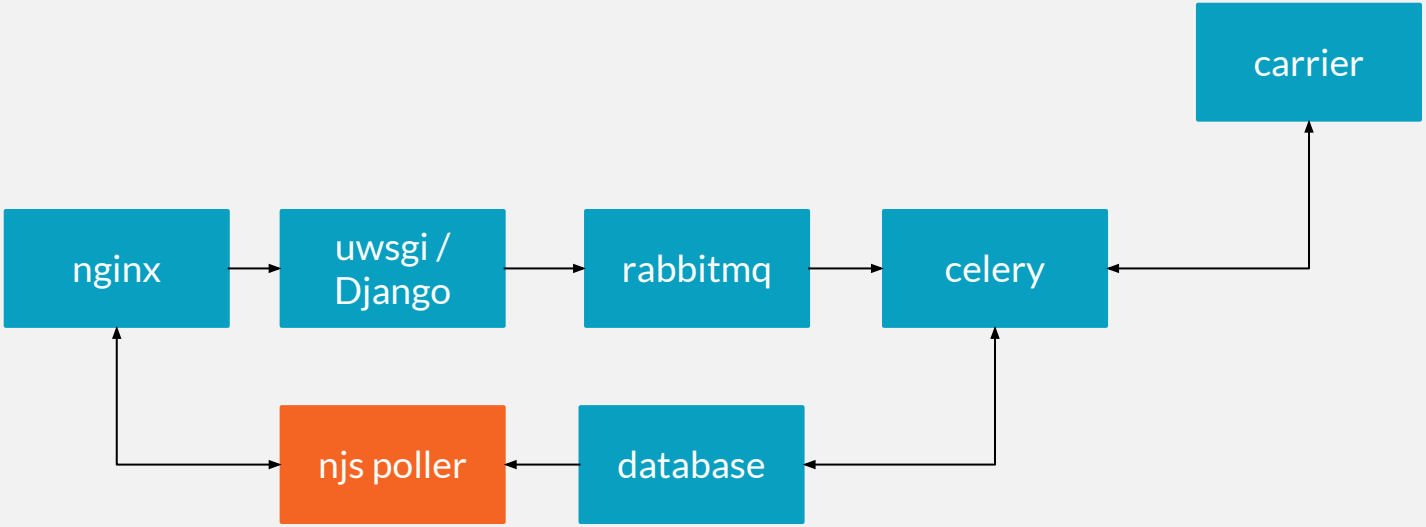
nginx offload - njs

```
function offloader(r, call_try) {
    offload(r, r.variables["offload_args"], 0);
}

function offload(r, poll_id, call_try){
   r.subrequest(
        "/poll/"+poll_id+'/'+call_try, {method: 'POST'},
        function(res) {
            if (res.status >= 200 && res.status < 300) {
                return r.return(res.status, res.responseBody);
            }
            setTimeout(offload, 1000, r, poll_id, call_try + 1);
        }
    );
}
```

nginx offloading stats

```
Time taken to complete requests:      1m47.236563893s

Requests per second:                              19


Number of connection errors:                       0

Number of 1xx responses:                           0

Number of 2xx responses:                        2000

Number of 3xx responses:                           0

Number of 4xx responses:                           0

Number of 5xx responses:                           0
```

Nginx offloading at SendCloud



Stats

Scenario	Request / sec	Error rate
Base	3	96%
gevent 5 workers	6	13%
gevent 10 workers	10	0%
uwsgi offloading	18	0%
nginx	19	0%

Thanks!