Building the Next-Gen Edge at Expedia

Rick Fast
@tortiepoint
Distinguished Software Engineer
Expedia Inc.
Building the Next-Gen Edge at Expedia

Rick Fast
@tortiepoint
Distinguished Software Engineer
Expedia Inc.
Overview

• The Problem

• Re-platforming “the edge” - Styx

• Architecture
  • Routing + Bot Management
  • Resilience
  • Self Service + Distributed Configuration
  • Traffic Data
The Problem
Brand Expedia

- White Label Platform
- 1000+ developers (global)
- Millions of (public) requests per minute, billions per day
- 100s of backend microservices
- Growing number on the front end
- Constantly re-building

…and many more
1994

Microsoft Travel Guide
CD-ROM
1998

eMain

#VelocityConf
2005

Web

Hotels

Flights

“E3”
2013

Exp Web

Hotels
  - Pricing
  - Geography

Flights
  - Search
  - GDS

Checkout
  - User
  - Finance

#VelocityConf
2015

Exp Web

Landing Pages

/Hotels
/Hotel-Search
/Flights
/Vacation-Packages
/Chicago-Hotels.d178248.Travel-Guide-Hotels

#VelocityConf
2015

/Chicago-Hotels.d178248.Travel-Guide-Hotels

50%  

Bucket 0  
(Control)

Exp Web

Landing Pages

50%  

Bucket 1  
(Treatment)
2015

/Landing Pages/Chicago-Hotels.d178248.Travel-Guide-Hotels

50% 50%

CVR 8% Bounce 22% CTR 68%

Exp Web

Landing Pages

CVR -2% Bounce -8% CTR -12%

#VelocityConf
2015

/Chicago-Hotels.d178248.Travel-Guide-Hotels

CVR 8%
Bounce 22%
CTR 68%

Exp Web

50%

50%

Landing Pages

CVR +1%
Bounce +0%
CTR +2%
2015

Exp Web

Landing Pages

Another Landing Page Platform

/Hotels
/Hotel-Search
/Flights
/Vacation-Packages
/Chicago-Hotels.d178248.Travel-Guide-Hotels

/Chicago-Hotels.d178248.Travel-Guide-Hotels

#VelocityConf
2017, Actually

- Exp Web
- Landing Pages
- Another Landing Page Platform
- Another Landing Page Platform (AMP)
- Home Page
- Hotel Search
- Hotel Details
- Check Out

(and many more)
Stitching it all together

• Lots of common functionality from monolith lost
• How to route between pages when they’re separate applications?
• How to “test in” new experiences and pages?
• How to scale all of this to hundreds of developers?
Re-platforming “the edge”
At the “edge”

Outer Edge

- us-west-2
- us-east-1
- eu-west-1
- ap-northeast-1
At the “edge”

Outer Edge

us-west-2

Inner Edge

Home Page  Hotel Search  Hotel Details  etc.
Before

Outer Edge

us-west-2

Inner Edge

Home Page  Hotel Search  Hotel Details  etc.
NGINX

• Industry standard reverse proxy
• Extensible via Lua scripting
• Expressive configuration language
• For simpler websites, might be all you need
NGINX

- Certain pages served from different application based on criteria
- Criteria gets complex
- NGINX config doesn’t cleanly support complex logic (e.g. nested if-s)
- Proprietary functionality difficult to port to Lua
- Core/platform libraries written in Java
- Each routing change was a code change by a central team
- Work items piled up quickly
- Codebase huge

```conf
# Always set a default for the variable.
set $homepageweb_route $chandler_upstream;
set $homepageweb_forceroute 0;

# Cookie based override for / route
if ($http_cookie ~* 'hopscotch_route_override=[^;]*homepage:true') {
    set $homepageweb_route $homepageweb_upstream;
    set $homepageweb_forceroute 1;
}

location ~* ^/homepage/.* {
    set $route_taken $homepageweb_upstream;

    # Redirect all http to https
    if ($protocol = "http") {
        return 301 https://$host$request_uri;
    }

    include conf.d/proxy_include.txt;
}

location ~* ^/globalcontrols/.* {
    location ~* ^/homepage/.* {
```

# VelocityConf

Expedia®
A New Approach

- NGINX scales from a performance/throughput standpoint
- We need to scale from a **people** standpoint
- Changes as **data** vs. **code**
- Self-service is key
A New Approach (JVM)
Choosing Technology

- JVM? Why
- Customizability
- Compatibility with internal libraries
- Blocking vs. Non-blocking IO
Blocking IO

Inbound Request

Thread/Process
BLOCKED

Request Processing

Response Processing

Proxied Request

Upstream App

Request Thread

#VelocityConf
Non-blocking IO
Styx

• JVM (Java 8) Reverse Proxy
• Written at hotels.com (part of Expedia)
  • Based on hotels.com’s contribution to Netflix Cloud Prize Competition in 2013 (Won best performance contribution)
• Built on Netty
• Plugin based
  • RxJava API
• Newly open sourced!
class MyPlugin implements Plugin {
    Observable<HttpResponse> intercept(HttpRequest request, Chain chain) {
        HttpRequest.Builder builder = request.newBuilder();

        builder.header("X-VelocityConf", "London");
        chain.context().add("OReilly", "London");

        Observable<HttpResponse> response = chain.proceed(builder.build());

        return response.map(resp -> {
            return resp.newBuilder().status(HttpResponseStatus.CREATED);
        });
    }
}
Styx

Inbound Request

GET www.expedia.com/

Plugin .intercept(…)

Plugin .intercept(…)

Plugin .handle(…)

Router

200 OK <html>…</html>

backend service
Styx

Plugin Chain

GET www.expedia.com/
200 OK
<html>…</html>

Plugin  .intercept(…)
Plugin  .intercept(…)
Plugin  .handle(…)
Router

backend service

#VelocityConf
Styx

Load balancing (choose origin)

Extension point: LoadBalancingStrategy
Implementations: BusyConnections, RoundRobin

GET www.expedia.com/

Plugin .intercept(…)

Plugin .intercept(…)

Plugin .handle(…)

Router

200 OK
<html>…</html>

backend service
Styx

Router (performs upstream call)

GET www.expedia.com/

Plugin .intercept(...) Plugin .intercept(...) Plugin .handle(...) Router

200 OK <html>...</html>

#VelocityConf
Styx

GET www.expedia.com/

Plugin \[.intercept(\_)\]

Plugin \[.intercept(\_)\]

Plugin \[.handle(\_)\]

Router \[\text{backend service}\]

200 OK
<html>…</html>

Rx operators applied on response
\texttt{Observable<HttpResponse>}

#VelocityConf
Styx

GET www.expedia.com/

200 OK
<html>...</html>

Response sent

Router

Plugin

Plugin

Plugin

Plugin

backend service

#VelocityConf
Other extension points

- BackendServicesRouter
  - Matches path, and performs backend call
- RetryPolicy
- BackendServicesRegistry
  - Supports configuring static, file based origins
  - Hot reloadable (via file or other mechanism)
  - Can be implemented to use dynamic service discovery
Get Styx

https://github.com/HotelsDotCom/styx
Architecture
#VelocityConf

cloud-gate-proxy

lobot

PostgreSQL

exp-web (legacy monolith)

Akamai

ELB

cloud-gate-proxy

NGINX (legacy)

SQS

various front end apps

Kinesis

Expedia

#VelocityConf
Outer Edge: Akamai

- Static assets
- Caching
- Geo-routing
- WAF
- DDoS protection
- Configuration
  - Monolithic
  - Change sync slow (~5-15 minutes)
#VelocityConf
cloud-gate-proxy -> lobot

ELB -> cloud-gate-proxy

SQS

lobot

PostgreSQL

exp-web (legacy monolith)

various front end apps

NGINX (legacy)

Kinesis

#VelocityConf
Inner Edge: Cloud Gate Proxy

- Named after “The Bean” in Chicago
- Built on Styx
- Primary Functions:
  - Routing
  - Bot management
  - Edge authorization
  - Distributed/global configuration system
  - Proxies millions of requests per minute

Cloud Gate
Cloud Gate Proxy

- site resolution
- user identification
- device identification
- bot score
- bot treatment
- routing
- kinesis logger
Context Building

```
{
  "name": "Rick",
  "loggedIn": true,
  "deviceId": "a123"
}
```

- Site resolution
- User identification
- Device identification
- Bot score
- A/B test evaluation
- Routing/bot treatment
- Kinesis logger

Expedia

*Travelocity

Wotif

Orbitz

#VelocityConf
Context Building

```
{
  "name": "Rick",
  "loggedIn": true,
  "deviceId": "a123"
}

{
  "mobile": true
}

{
  "My_Test": 1
}
```

```
{
  "user": {
    "name": "Rick",
    "loggedIn": true,
    "deviceId": "a123"
  },
  "site": {
    "name": "expedia.com"
  },
  "locale": "en_US",
  "experiments": {
    "My_Test": 1,
    "Experiment_0": 1
  },
  "bot": {
    "classification": "Human",
    "confidence": 100.0
  },
  "device": {
    "mobile": true
  }
}
```
Request Handling

```json
{
    "rule": {
        "and": [
            {
                "site": "orbitz",
                "experiment": {
                    "My_Test": 1
                }
            }
        ],
        "application": "homepage-web"
    }
}
```
Request Handling

{  
  "request": {  
    ...
  },
  "response": {  
    ...
  },
  "context": {  
    ...
  }
}
Routing

• Single path can be served by multiple (1-5) applications

• Complicated routing logic required
  • By site (Orbitz, Travelocity, etc)
  • By locale (en_US, etc)
  • By A/B test bucketing
  • By bot classification
  • Query parameters, headers, and cookies
Routing

• Multi-region
  • One cloud gate proxy farm per region
  • “Vegas rules”
  • Manual failover / maintenance mode
• 2 phases
  • Endpoint (path) resolution
  • Application resolution - using rules engine
Routing - Endpoint Resolution

```json
GET https://www.expedia.com/Hotels

{  
  "id": "hotel-home",
  "matchType": "EXACT",
  "path": "/Hotels",
  "experiments": [ 
    "My_AB_Test",
    "Another_AB_Test"
  ]
}
```
Routing - Rules Engine

- Embedded rules engine: Clara
- Written in Clojure
- Input: “Facts” (Request + Context)
- Output: Application ID
- Rules represent logic, but are data
- Website: http://www.clara-rules.org/

```json
{
  "id": 1,
  "name": "homepage-header-rule",
  "rule": {
    "and": [
      {
        "site": "travelocity"
      },
      {
        "locale": "en_US"
      },
      {
        "experiment": {
          "My_Test": 1
        }
      }
    ]
  },
  "result": {
    "app": "homepage-web"
  },
  "version": 1,
  "created_by": "creator",
  "created": "2016-09-20T23:55:49.940Z"
}
```
GET https://www.expedia.com/Hotels

endpoint: hotel-home

```
"rules": [{
  "rule": {
    "and": [
      "site": "expedia",
      "experiment": {
        "My_AB_Test": 1
      }
    ]
  }
},
  "application": "homepage-web",
},
{ "rule": {
  "and": [
    "site": "expedia",
    "experiment": {
      "My_AB_Test": 0
    }
  ]
},
  "application": "exp-web",
}]
```
Routing - Application Resolution

GET
https://www.expedia.com/Hotels

endpoint: hotel-home

```

"rules": [{
  "rule": {
    "and": [
      "site": "expedia",
      "experiment": {
        "My_AB_Test": 1
      }
    ]
  },
  "application": "homepage-web",
},
  {
    "rule": {
      "and": [
        "site": "expedia",
        "experiment": {
          "My_AB_Test": 0
        }
      ]
    },
    "application": "exp-web",
  ]
}
```

#VelocityConf
Routing - Application Resolution

GET
https://www.expedia.com/Hotels

endpoint: hotel-home
application: homepage-web

{
  "id": "homepage-web",
  "origins": {
    "us-west-2": [
      "homepage-web.prod.expedia.com",
      ...
    ],
    ...
  },
  "circuitBreaker": {
    "maxConnections": 300,
    "timeout": 2000,
    "threshold": 0.30,
    ...
  }
}
Routing - Upstream Request

- GET https://www.expedia.com/Hotels
  - **endpoint**: hotel-home
  - **application**: homepage-web

- GET https://10.10.10.10/Hotels
Circuit Breakers

- Defend proxy + rest of site from bad upstreams
- Netflix circuit breaker library
  - Also uses RxJava
- Allows us to relieve pressure when upstreams back up
  - Upstream latency = site killer
- Configurable fallbacks
- Isolates individual upstream applications
- Configurable timeout + threshold per upstream app
Routing - Circuit Breaker

GET
https://www.expedia.com/Hotels

endpoint: hotel-home
application: homepage-web
circuitBreaker:
  threshold: 0.30

200 OK

(router) (closed) (homepage-web)

failure rate: 10%
Routing - Circuit Breaker

GET https://www.expedia.com/Hotels

endpoint: hotel-home
application: homepage-web
circuitBreaker:
  threshold: 0.30

200 OK

(homepage-web) (closed)

failure rate: 20%
Routing - Circuit Breaker

- GET https://www.expedia.com/Hotels
- **endpoint**: hotel-home
- **application**: homepage-web
- **circuitBreaker**: threshold: 0.30
- **failure rate**: 30%

503 SERVICE UNAVAILABLE
Routing - Circuit Breaker

GET
https://www.expedia.com/Hotels

endpoint: hotel-home
application: homepage-web
circuitBreaker:
  threshold: 0.30

200 OK

(open)

failure rate: 30%

(exp-web (fallback))
Routing - Circuit Breaker

GET
https://www.expedia.com/Hotels

endpoint: hotel-home
application: homepage-web

circuitBreaker:
threshold: 0.30

(half-open)

failure rate: 0%

200 OK
Routing - Circuit Breaker

GET
https://www.expedia.com/Hotels

endpoint: hotel-home
application: homepage-web
circuitBreaker:
threshold: 0.30

200 OK

(router)

(closed)

homepage-web

failure rate: 0%
Preventing conflicts

• Legacy system was code

  • Used Cucumber - all changes had corresponding tests

• New concept of “owned requests”

  • Routing rules have request signatures associated

  • When rules change, all owned requests are applied to ensure no encroachment

• “Headless” proxy runs plugins without proxying calls
Bot Detection + Treatment

- Bot classification
- Bot treatments
  - RecapTCHA (for suspects)
  - Tarpit (good for bad scrapers)
  - Blackhole (for malicious bots)
  - DOM mangling (good for bad scrapers)
Bot Classification

GET https://www.expedia.com/Hotel-Search
Bot Classification

GET
https://www.expedia.com/Hotel-Search

classification: Suspect
confidence: 90

search-web

cassandra
Bot Treatment

GET
https://www.expedia.com/Hotel-Search

bot score

search-web

recaptcha
Bot Treatment

GET https://www.expedia.com/Hotel-Search

bot score

search-web

recaptcha

#VelocityConf
Bot Treatment

GET https://www.expedia.com/Hotel-Search

bot score

search-web

recaptcha

#VelocityConf
Bot Treatment

GET
https://www.expedia.com/Hotel-Search
Bot Treatment

GET
https://www.expedia.com/Hotel-Search
Bot Treatment

GET
https://www.expedia.com/Hotel-Search

bot score

classification: Human
certainty: 100

cassandra

search-web

recaptcha

#VelocityConf
#VelocityConf

Cloud-gate-proxy

ELB

Akamai

Kinesis

various front end apps

lobot

PostgreSQL

exp-web (legacy monolith)

globally

NGINX (legacy)

#VelocityConf
Configuring the Fleet

• Legacy system copied files around

• Configuration read from Consul
  • Using Consul watches
  • Multi-DC synchronization
  • Routing config, bot config, auth config

• Source of truth - Lobot + PostgreSQL

• Acknowledgements
Consul

Server Quorum
(3, 5, or 7)
One per region
Consul

cloud-gate-proxy

cloud-gate-proxy

cloud-gate-proxy
GET /v1/kv/routing?index=100&wait=10s 200 OK

Consul

X-Consul-Index: 102
[

- { "Key": "/routing/application/homepage-web", "ModifyIndex": 123432, "Value": "{...}" },

- { "Key": "/routing/endpoint/hotels-home", "ModifyIndex": 918234, "Value": "{...}" },

- { "Key": "/routing/rules/hotel-home", "ModifyIndex": 10200, "Value": "{...}" }

]
```java
public void onConfigChange(ConsulResponse response) {
    // deserialize and atomically update state
    updateState(applications, endpoints, rules);

    // acknowledge
    ack(response.consulIndex());
}
```
Consul

```
GET /v1/kv/routing?index=102&wait=10s
```
cloud-gate-proxy

lobot

PostgreSQL

NGINX (legacy)

various front end apps

Kinesis

exp-web (legacy monolith)

ELB

Akamai

#VelocityConf
Lobot

- Self service traffic portal
- Source of truth for all edge configuration
- Stores normalized config data in PostgreSQL
  - Use JSONB format for Clara rules
  - Configuration is versioned w/ audit trail
- Fine grained authorization
  - Critical functionality locked down
  - Line of business teams own their endpoints
Lobot

- Configuration edited in database
  - Routing config
  - Bot classification/treatment rules
  - Feature flags
- Changes published (denormalized) to Consul
  - Leveraging Consul transaction
  - Published to all active regions
  - Environment promotion (manual changes in test, promote to prod)
- Lobot API
  - UI
  - CLI tools
#VelocityConf

cloud-gate-proxy

lobot

(virtualized)

PostgreSQL

ELB

various front end apps

Kinesis

SQS

exp-web (legacy monolith)

#VelocityConf
Ensuring Consistency

• Lots of Cloud Gate Proxy instances deployed worldwide
  • Use Consul to discover instances
  • Track acknowledgements from each
    • Global SQS queue
ACKs

```json
{
  "id": "cgp_10-204-223-23",
  "region": "ap-southeast-1",
  "ok": true,
  "index": 102,
  "errors": []
}
```
Streaming Traffic Data

- Edge traffic data is useful
- Limit IO on Cloud Gate Proxy
- Request, Response and Context written to Kinesis in Avro format
- Kinesis
  - Write data once, many consumers
  - Almost zero management
Consumer: Bot Detection

- voight-kampff
- Applies bot rules to each request/response
- Same rules engine + rule format as routing
- Rules managed manually
- Automated detection
- Tensorflow

```
"rules": [
  {
    "rule": {
      "and": [
        {
          "cidr": "192.168.100.14/24"
        },
        {
          "userAgent": "SEO_Bot"
        }
      ]
    },
    "result": {
      "classification": "Good_Bot",
      "confidence": 100.0
    }
  },
  ...
]
```
Consumer: Monitoring

- Cloud Gate Proxy uses Dropwizard Metrics
- Short term (days), fine grained metrics
  - Spark streaming
  - Slices + aggregates across shards by many dimensions
  - Emits to Graphite
- Long term (years)
  - AWS Firehose to S3
  - Athena for queries
Consumer: Tracing

- Inner edge is start of each request trace
- Performed offline to eliminate extra IO on Cloud Gate Proxy
- Uses Haystack
  - Opentracing compliant
  - Kinesis consumer packages and sends trace information
- Open source in progress

Haystack
Things we learned

• While tempting, don’t build a monolith edge layer
• Remain IO bound, avoid rendering
• Don’t block!
• Configuration files don’t scale
• Be resilient!
thx