Throttling on a multi-layer Varnish setup

Peter van der Spek

bluebillywig

Varnish User Day
November 28th 2013

About Us

bluebillywig

Two businesses

- Online Video Platform
- News websites using Escenic CMS

Online Video Platform

Market leader in Benelux countries.
150 European Media websites in 8 countries.

News Websites using Escenic CMS

Independent.ie

Belfast Telegraph.co.uk

Herald.ie

DAGBLAD VAN HET NOORDEN

LEEUWARDER COURANT LC.nl

lphen.cc compact compleet

De Gooi- en Eemlander

Haarlems Dagblad
Oprechte Haerlemse Courant 1656

Noordhollands Dagblad

Leidsch Dagblad

IJmuider Courant
Waarin opgenomen de Kennemer Courant en het Dagblad voor IJmuiden

Almere Vandaag

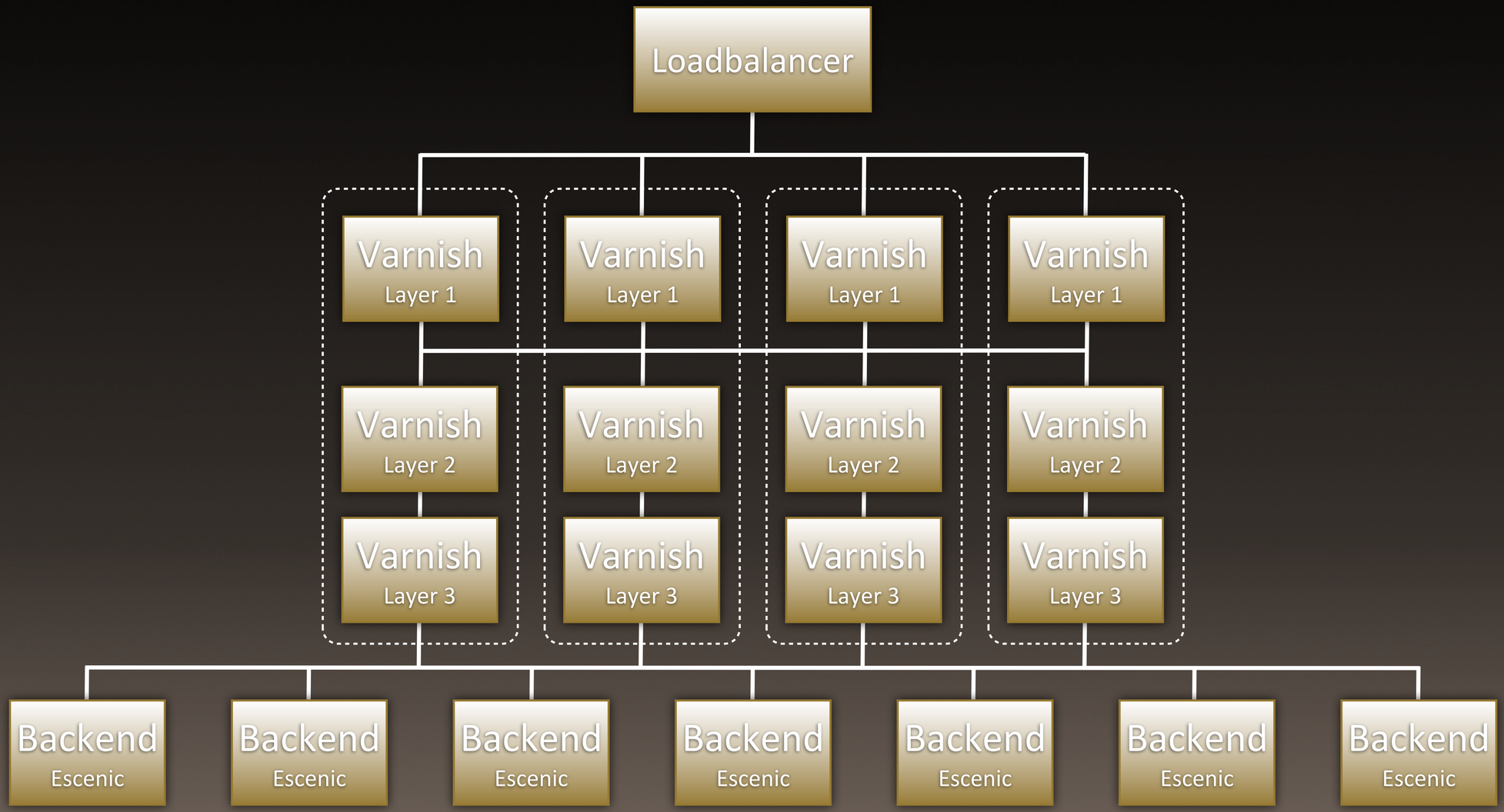
Varnish for Online Video Platform

Varnish is used for:

- Video players
- Video metadata
- Video API calls

Video streams themselves are distributed using a CDN.

Varnish for INM websites



Varnish for INM websites

Requirements

Availability
Scalability
Efficient backend utilization
Responsiveness
Robustness

Availability & Scalability

- Use more Varnish and Escenic instances than we strictly need.
- We need at least two Varnish stacks to run reliably. Normally we run four stacks.
- Use Varnish backend pools.
- Tooling to “administratively” enable and disable components.

Availability & Scalability

in VCL – director pool to layer 2

```
probe varnish_l2_online_healthcheck {
    .url = "/healthcheck/l2";
    .interval = 5s;
    .timeout = 2s;
    .window = 1;
    .threshold = 1;
    .initial = 1;
    .expected_response = 200;
}
```

Availability & Scalability

in VCL – director pool to layer 2

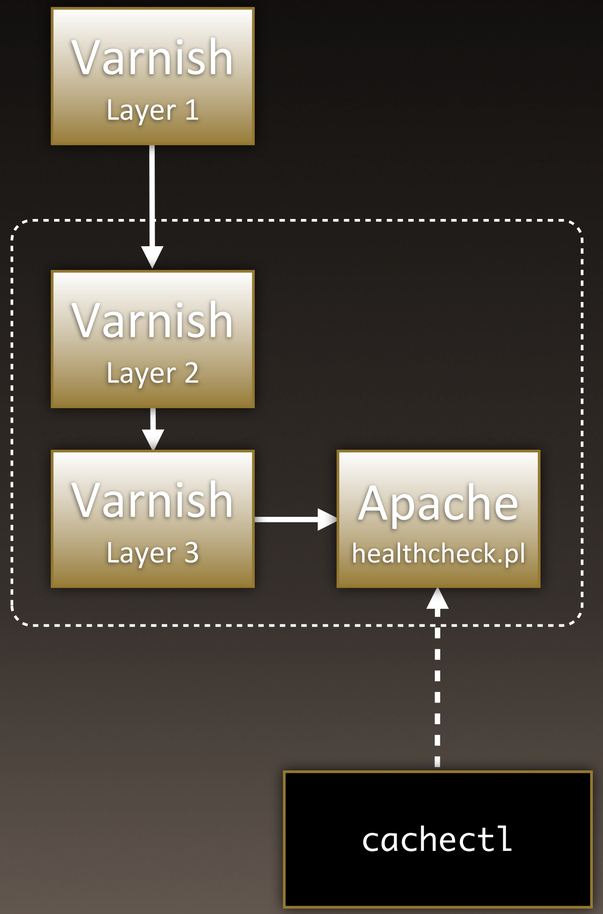
```
backend varnish_l2_node_01 {
    .host = "172.16.0.1";
    .port = "8087";
    .first_byte_timeout = 600s;
    .probe = varnish_l2_online_healthcheck;
}

...

backend varnish_l2_node_30 {
    .host = "172.16.0.30";
    .port = "8087";
    .first_byte_timeout = 600s;
    .probe = varnish_l2_online_healthcheck;
}
```

Availability & Scalability

in VCL – director pool to layer 2

```
director varnish_l2_director hash {
    {
        .backend = varnish_l2_node_01;
        .weight = 100;
    }
...
    {
        .backend = varnish_l2_node_30;
        .weight = 100;
    }
}
```

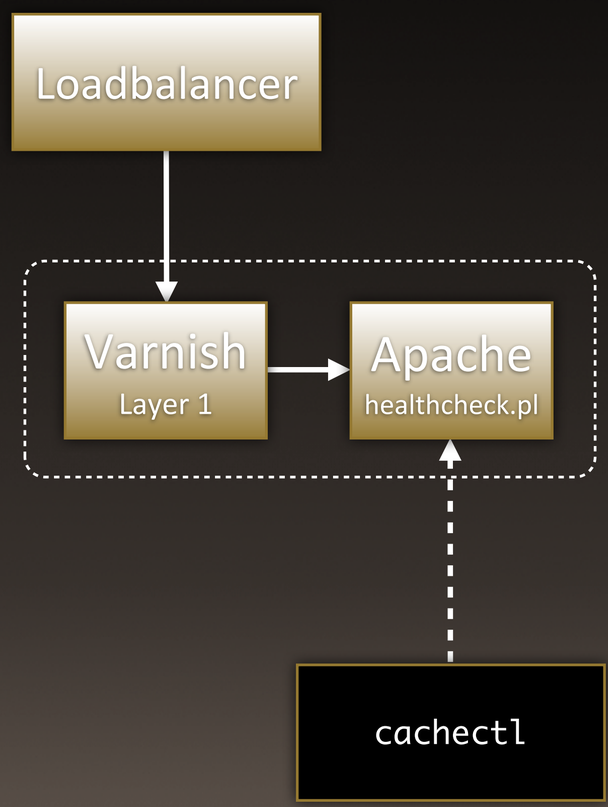
Availability & Scalability

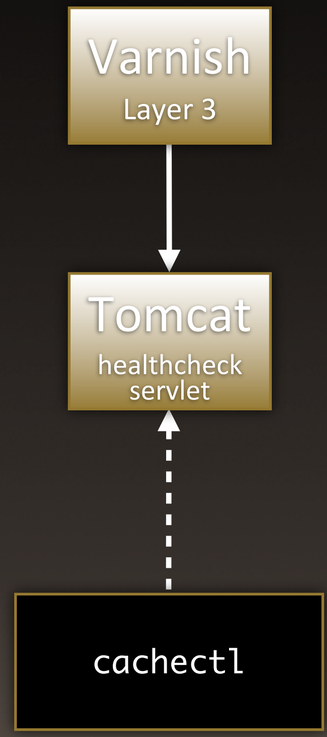


- Varnish layer 1 healthchecks `/healthcheck/l2` on Varnish layer 2. Expects `200 OK` response.
- Request is passed on to Varnish layer 3, healthcheck is handled by Apache.
- Healthcheck is controlled by custom `cachectl` tool:

```
$ cachectl online varnish
$ cachectl status
  varnish       : ONLINE  (pools: l1, l2)
```

Availability & Scalability





Availability & Scalability

- Tooling allows us to restart Escenic in a controlled way:
- `cachectl offline engine` takes Escenic out of the Varnish pool.
- Escenic instance can be deployed, restarted and pre-heated while being “offline”.
- `cachectl online engine` takes Escenic into the Varnish pool again.

Availability & Scalability

- Adding an Escenic instance is just as easy:
- Deploy Escenic instance, start and pre-heat.
- `cachectl online engine` adds Escenic instance to the Varnish pool.

Varnish for INM websites

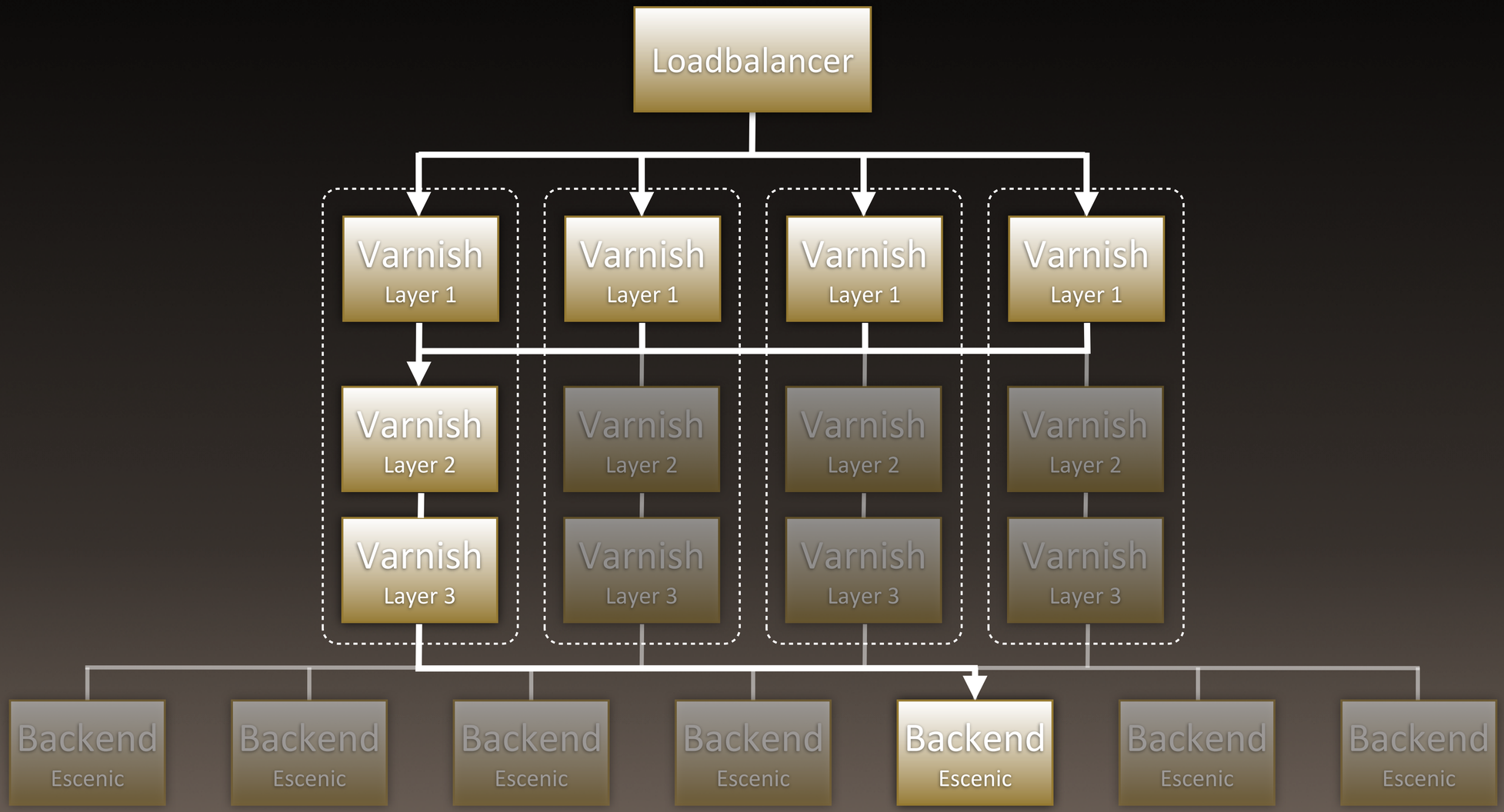
Requirements

Availability
Scalability
Efficient backend utilization
Responsiveness
Robustness

Efficient backend utilization

- Generating pages is expensive, so: We want every unique URL to be generated on only one Escenic instance.
- For any unique URL, only one Varnish layer 3 instance may go the backend.
- Direct any incoming request in Varnish layer 1 to layer 2 using the hash director.

Varnish for INM websites



Efficient backend utilization

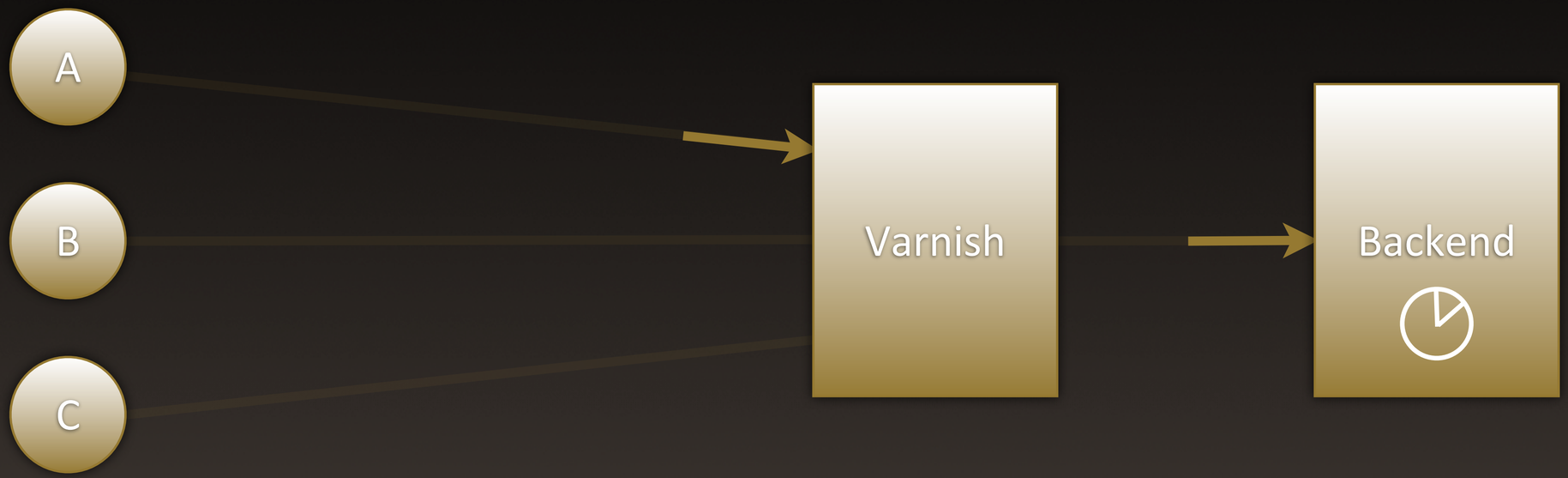
- But... any change in the number of layer 2 backends will cause the hash director in layer 1 to redistribute the requests.
- This leads to an excessive number of cache misses which impacts the backends.
- Could be solved by using a consistent hash director, but we don’t have that (yet).
- Maybe in Varnish 4 as a VMOD?

Varnish for INM websites

Requirements

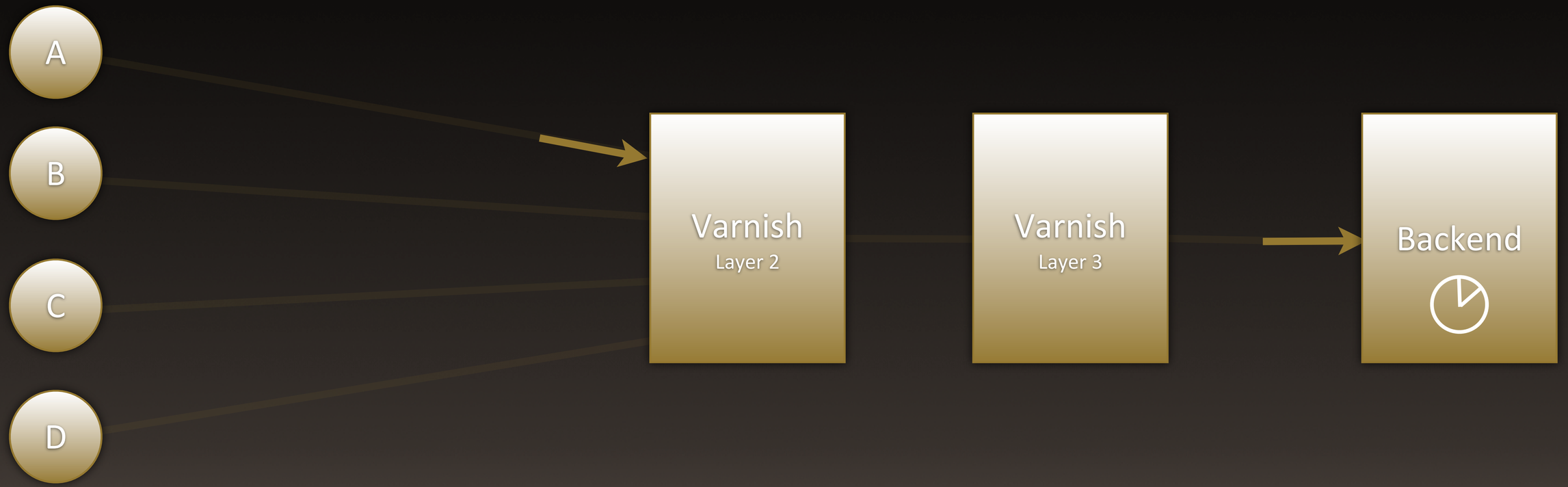
Availability
Scalability
Efficient backend utilization
Responsiveness
Robustness

Responsiveness



“Traditional” setup with grace enabled.

Responsiveness



Two layer setup

Responsiveness

Implementation in VCL

Key tasks:

- Failfast layer 3 backend
- Retry on failfast timeout
- Fix TTL on layer 2
- Fix Age on layer 2

Responsiveness

in VCL – layer 2

```
backend varnish_l3_node_failfast {
    .host = "127.0.0.1";
    .port = "8088";
    .first_byte_timeout = 0.4s;
    .between_bytes_timeout = 0.4s;
}

backend varnish_l3_node {
    .host = "127.0.0.1";
    .port = "8088";
}
```

Responsiveness

in VCL – layer 2

```
sub vcl_recv {
    set req.grace = 0s;

    if (req.restarts == 0) {
        set req.backend = varnish_l3_node_failfast;
    } else {
        set req.backend = varnish_l3_node;
    }

    return (lookup);
}
```

Responsiveness

in VCL – layer 2

```
sub vcl_error {
    if ((obj.status == 503) && (req.restarts == 0)) {
        return (restart);
    }
}
```

Responsiveness

in VCL – layer 2

```
sub vcl_fetch {
    if (beresp.http.X-L3-TTL) {
        set beresp.ttl = std.duration(beresp.http.X-L3-TTL + "s", 60s);

        if (beresp.http.X-L3-Age) {
            set beresp.ttl = beresp.ttl -
                    std.duration(beresp.http.X-L3-Age + "s", 0s);
        }
    }

    return (deliver);
}
```

Responsiveness

in VCL – layer 2

```
sub vcl_deliver {
    unset resp.http.X-L3-TTL;

    if (resp.http.X-L3-Age) {
        set resp.http.Age = "" + (
                std.integer(resp.http.X-L3-Age, 0) +
                std.integer(resp.http.Age, 0)
        );
        unset resp.http.X-L3-Age;
    }
}
```

Responsiveness

in VCL – layer 3

```
sub vcl_fetch {
    set beresp.grace = 120m;

    set beresp.http.X-L3-TTL = beresp.ttl;

    return (deliver);
}

sub vcl_deliver {
    set resp.http.X-L3-Age = resp.http.Age;
    unset resp.http.Age;
}
```

Responsiveness

But how about purging?

`PURGE` requests are handled by setting the TTL to zero on layer 2 and on layer 3.

Varnish for INM websites

Requirements

Availability
Scalability
Efficient backend utilization
Responsiveness
Robustness

Robustness

- End-user and bot traffic behave very differently. We added an option to separate both types of traffic to different sets of backends.
- We regularly see remote scripts scraping the websites. We have added protection against high levels of cache misses caused by a single IP.

Robustness

in VCL – Throttling. Determine whitelist on layer 1

```
sub vcl_recv {
    unset req.http.X-No-Throttle;

    if (client.ip ~ local || client.ip ~ inm) {
        set req.http.X-No-Throttle = "true";
    }
}
```

Robustness

in VCL – Apply throttle on layer 2

```
import throttle;
import var;

sub vcl_miss {
    var.set("client_ip", regsub(req.http.X-Forwarded-For, "^([^,]*), .*$", "\1"));

    if (req.restarts == 0 && !req.http.X-No-Throttle) {
        var.set("throttle", "5req/1s,15req/30s,50req/5m");

        if (throttle.is_allowed(var.get("client_ip"), var.get("throttle")) > 0s) {
            error 429 "Too Many Requests";
        }
    }
}
```

Conclusion

- One or multiple layers: best choice depends on requirements.
- For us the multi layer setup helped us achieve our requirements.

Questions?

Thank You

Peter van der Spek
p.vanderspek@bluebillywig.com

http://nl.linkedin.com/in/pvdspek