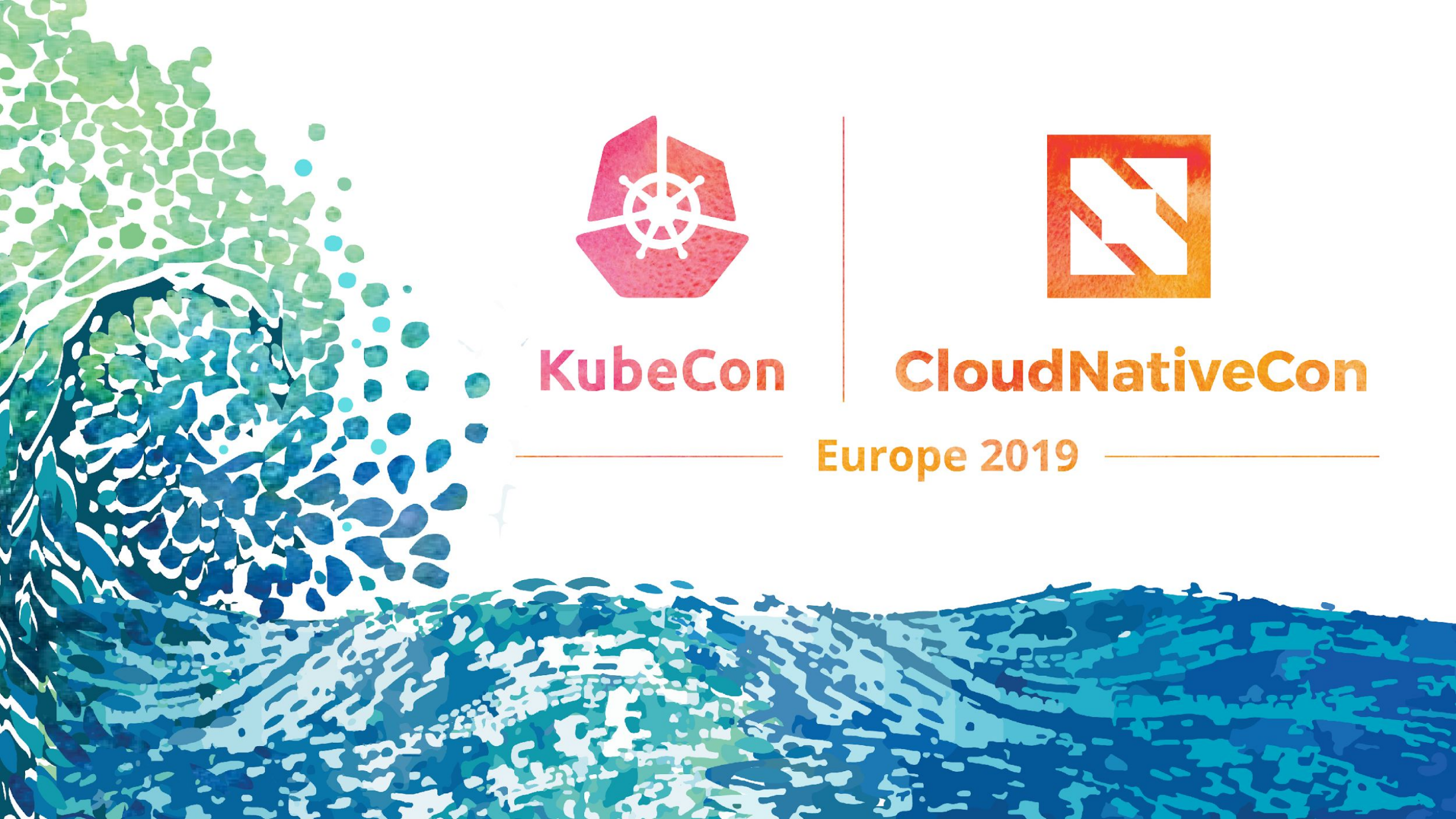
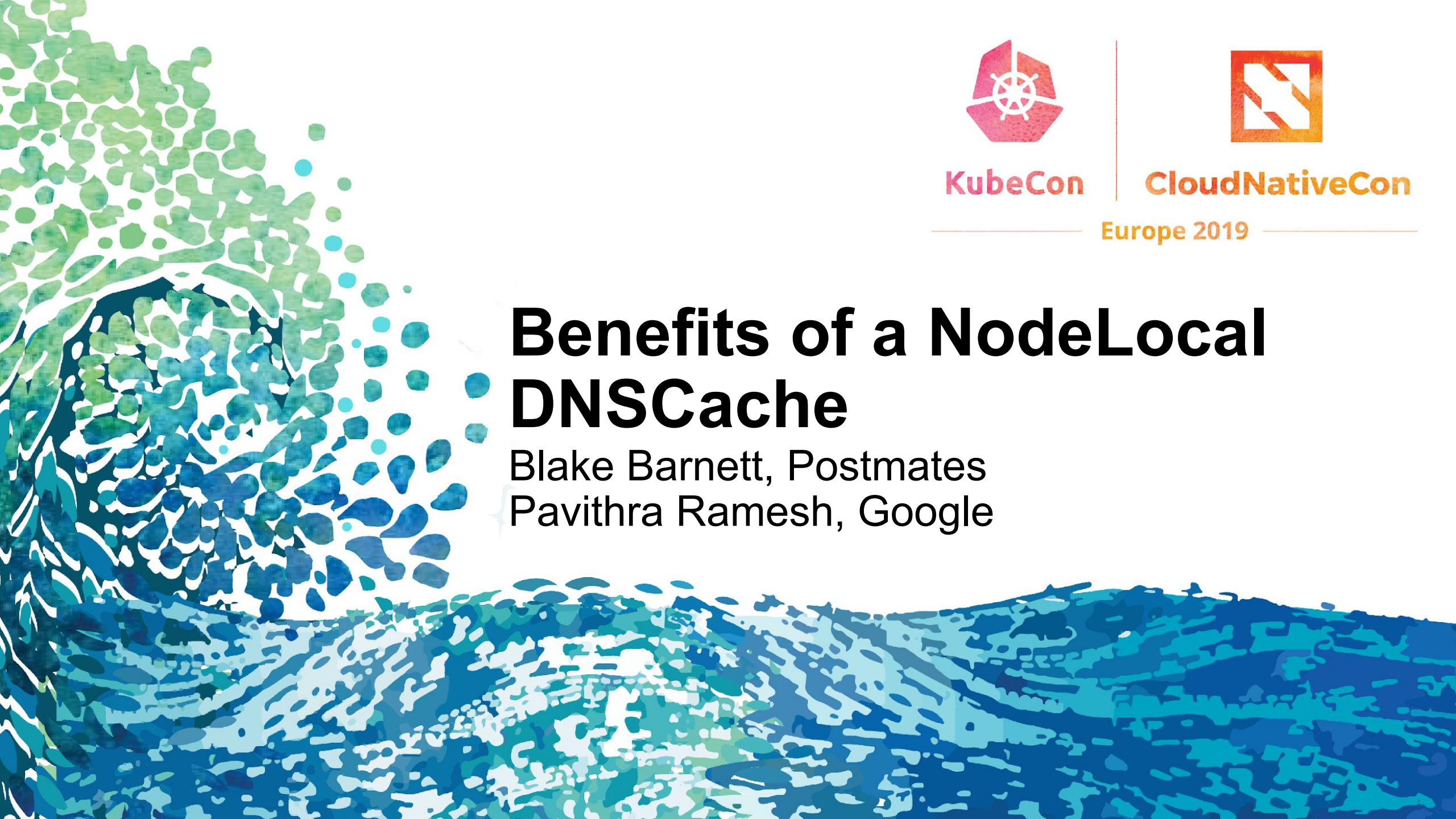

KubeCon
CloudNativeCon
Europe 2019



KubeCon | CloudNativeCon

Europe 2019

Benefits of a NodeLocal DNSCache

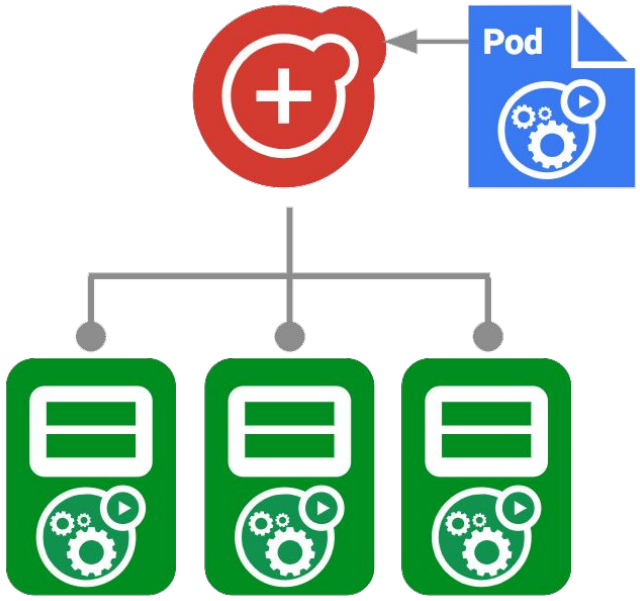
Blake Barnett, Postmates
Pavithra Ramesh, Google

Agenda

- What is NodeLocal DNSCache? Why do we need it?
  - Current K8s DNS setup
  - NodeLocal DNSCache deep dive
- Metrics exposed by NodeLocal DNSCache
- Latency Improvements
- Steps to deploy NodeLocal DNSCache
- Future work
- Questions

What is NodeLocal DNSCache?

- Addon that runs a DNS cache on each node as a Daemonset. Runs CoreDNS as a cache.
- Pods in the same node talk to the local cache instance.



Motivation

For DNS we'd like the following things to be considered

1. Set multiple DNS servers via kubelet
2. Setup a DNS server/cache per host

https://github.com/kubernetes/kubernetes/issues/7470#issuecomment-248912603

https://github.com/kubernetes/kubernetes/issues/45363

kube-dns per node #45363

consider running dnsmasq on nodes instead of kube-dns #32749

- https://github.com/kubernetes/kubernetes/issues/32749

Possible causes for DNS latency

1. Parallel v4 and v6 queries + auto searchpath expansion = 10x the initial number of queries. This increases the chances of hitting netfilter race conditions

foo.com ->

```
$ns.svc.cluster.local
svc.cluster.local
cluster.local
… host suffixes
```

= 5 A, 5 AAAA queries

Possible causes for DNS latency

2. Too many DNS queries overflowing conntrack tables

3. dnsmasq concurrent connections limit(applies to kube-dns)

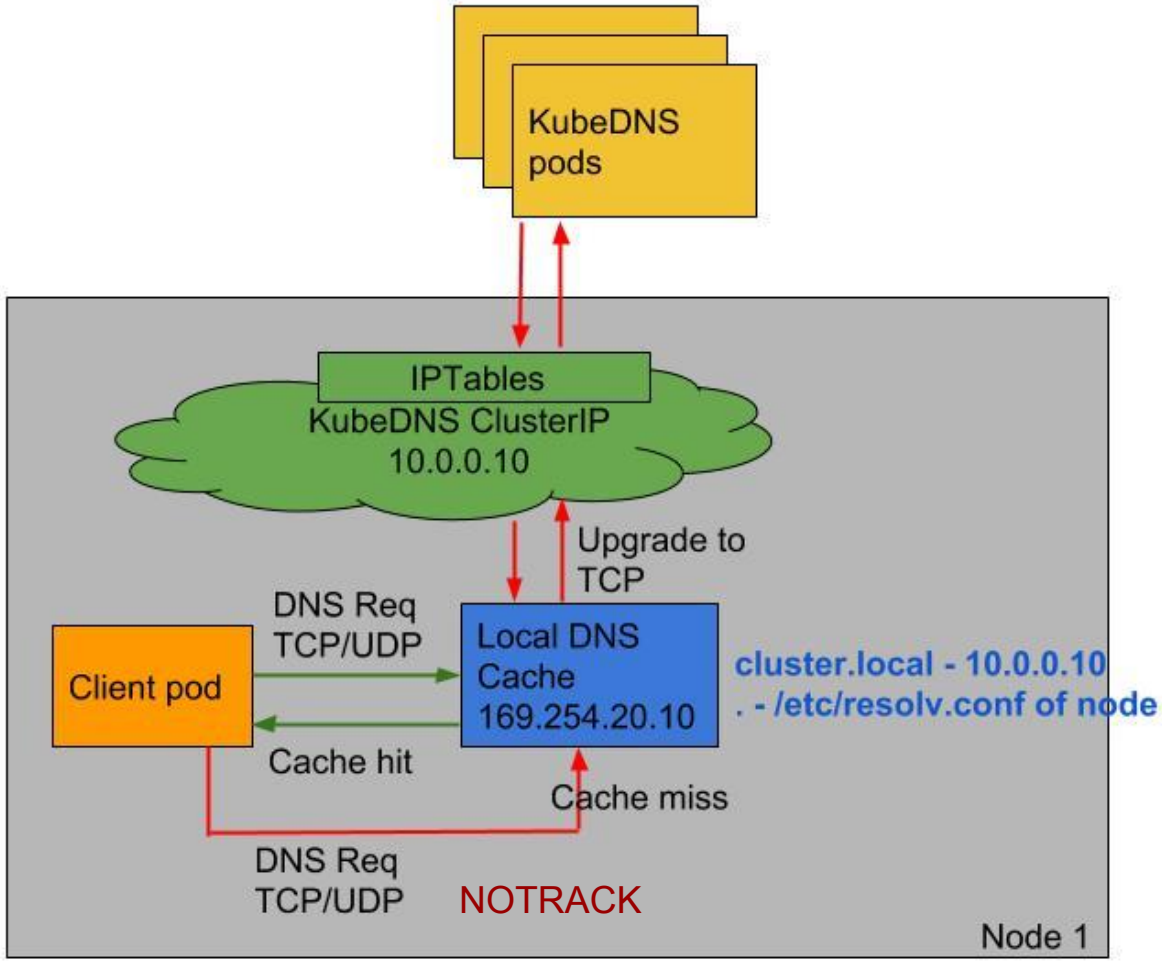
4. Additional cloud provider limits for dns lookups

5. UDP being unreliable, client needs to wait for timeout in case of packet drops

Existing solutions/workarounds

1) `single-request-reopen` to avoid parallel queries.
2) Reduce ndots value, modify timeout value
3) Run kube-dns as daemonset, dnsmasq as daemonset
4) Modify dnsmasq parameters to support more concurrent connections.
5) use-vc option to query via TCP instead of UDP
6) autopath plugin (in CoreDNS) to reduce number of client-initiated queries.

Enter NodeLocal DNSCache



Source code : https://github.com/kubernetes/dns/tree/master/cmd/node-cache

Enter NodeLocal DNSCache

Previously listed problems are addressed in NodeLocal DNSCache

1) Too many queries causing conntrack table to fill up, netfilter race.

Addressed by skipping conntrack and caching to reduce the number of trips upstream which will still use DNAT and conntrack

2) dnsmasq concurrent connections limit

By having an instance per node, the lookups are localized and fewer queries will go to dnsmasq. Also external hostname queries don't go to dnsmasq anymore.

Enter NodeLocal DNSCache

3) Additional cloud provider limits for dns lookups

External queries don't need to go through clusterDNS pods, better use of per-node limits

4) UDP being unreliable, client needs to wait for timeout in case of packet drops

Upstream queries are sent over TCP. The forward plugin also reuses TCP sockets, so number of TCP connections will not grow with each request.

Metrics

Since NodeLocal DNSCache uses CoreDNS as a cache, metrics exported by the different coreDNS plugins are available.

These can give insight into per-node request/response statistics.
Some of the metrics are:

```
coredns_cache_size
coredns_cache_hits_total
coredns_cache_misses_total
```

```
coredns_forward_request_count_total
coredns_forward_request_duration_seconds_bucket
```

And many more

Validation tests for NodeLocal DNSCache

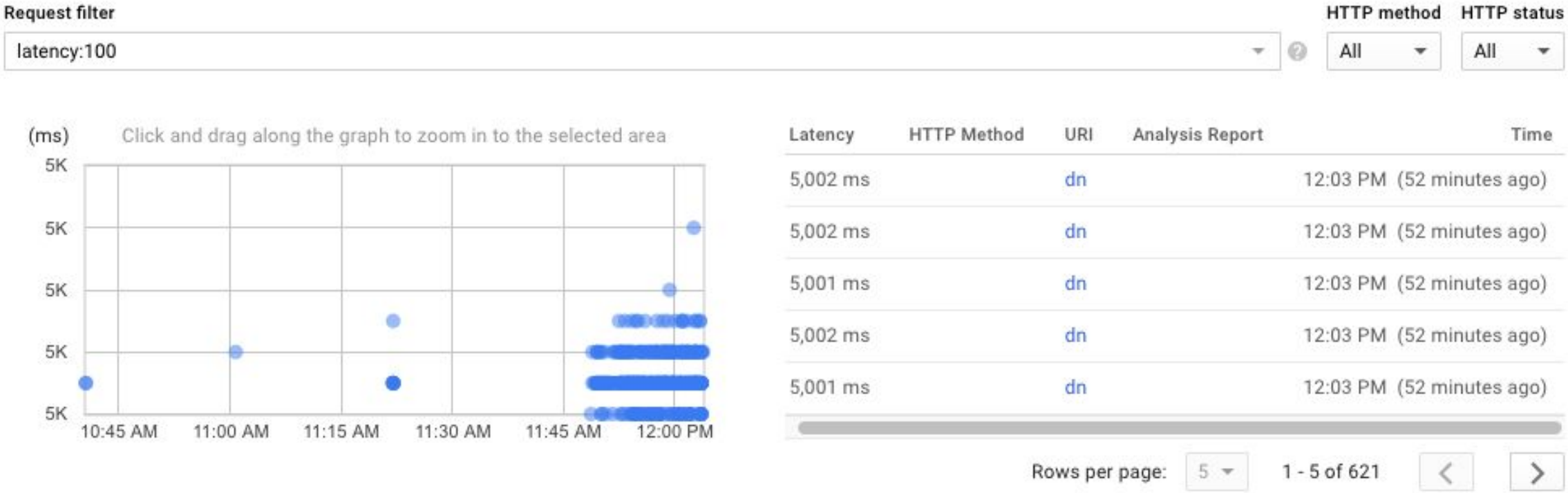
Test setup

Test description:
Uses konfirm-dns setup, source code here(Thanks Justin Santa Barbara!)
- https://github.com/kubernetes/dns/pull/281
In the process of moving this to perf-tests/dns repo.

Client pod looks up “kubernetes.svc.cluster.local” in a loop, 200 QPS(A + AAAA)
240 test pods were spun up, all pointing to the same kube-dns service IP.

Same tests were run on different cloud providers, similar results.
NodeLocal DNSCache improvements were seen in all cases.

Tests without NodeLocal Cache



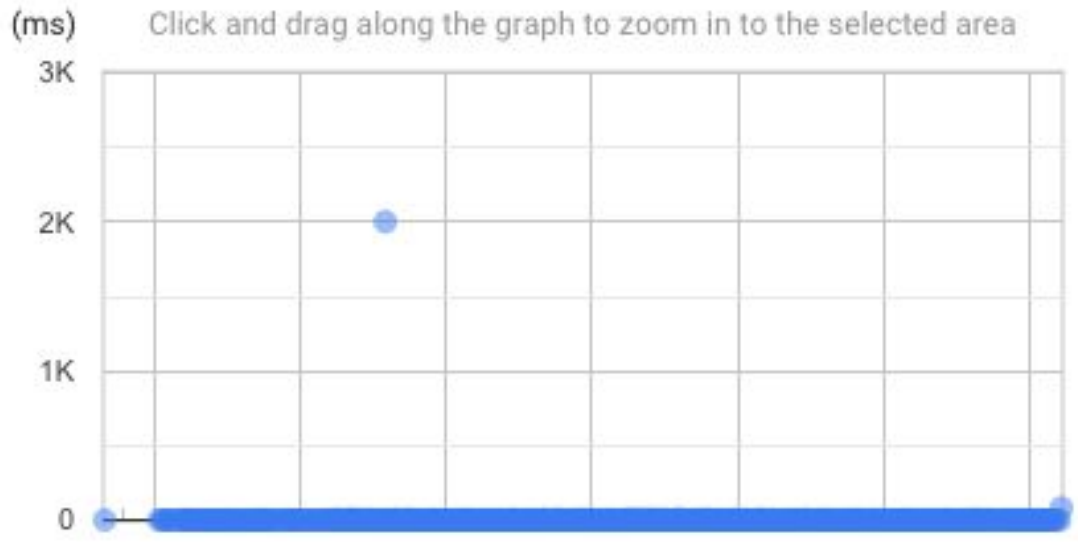
Using default kube-dns settings, pods sending ~200 QPS each to 2 kube-dns pods.
240 konfirm-dns pods across 3 nodes.
Conntrack limit of `net.nf_conntrack_max = 524288`
5k latency indicates 5s timeouts.

Improved latencies

Request filter

HTTP method: All

HTTP status: All



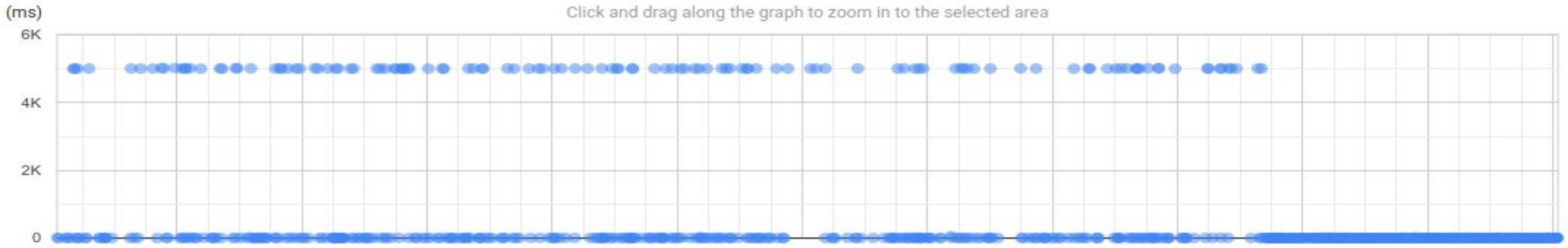
Latency	HTTP Method	URI	Analysis Report	Time
82 ms		dns looku		1:41 PM (Just now)
0 ms		dns looku		1:41 PM (Just now)
1 ms		dns looku		1:40 PM (Just now)
0 ms		dns looku		1:40 PM (Just now)
0 ms		dns looku		1:40 PM (Just now)

Rows per page: 5 1 - 5 of 1000

node-local-dns + kube-dns

- 3 nodes
- 2 kube-dns pods
- 240 konfirm-dns pods

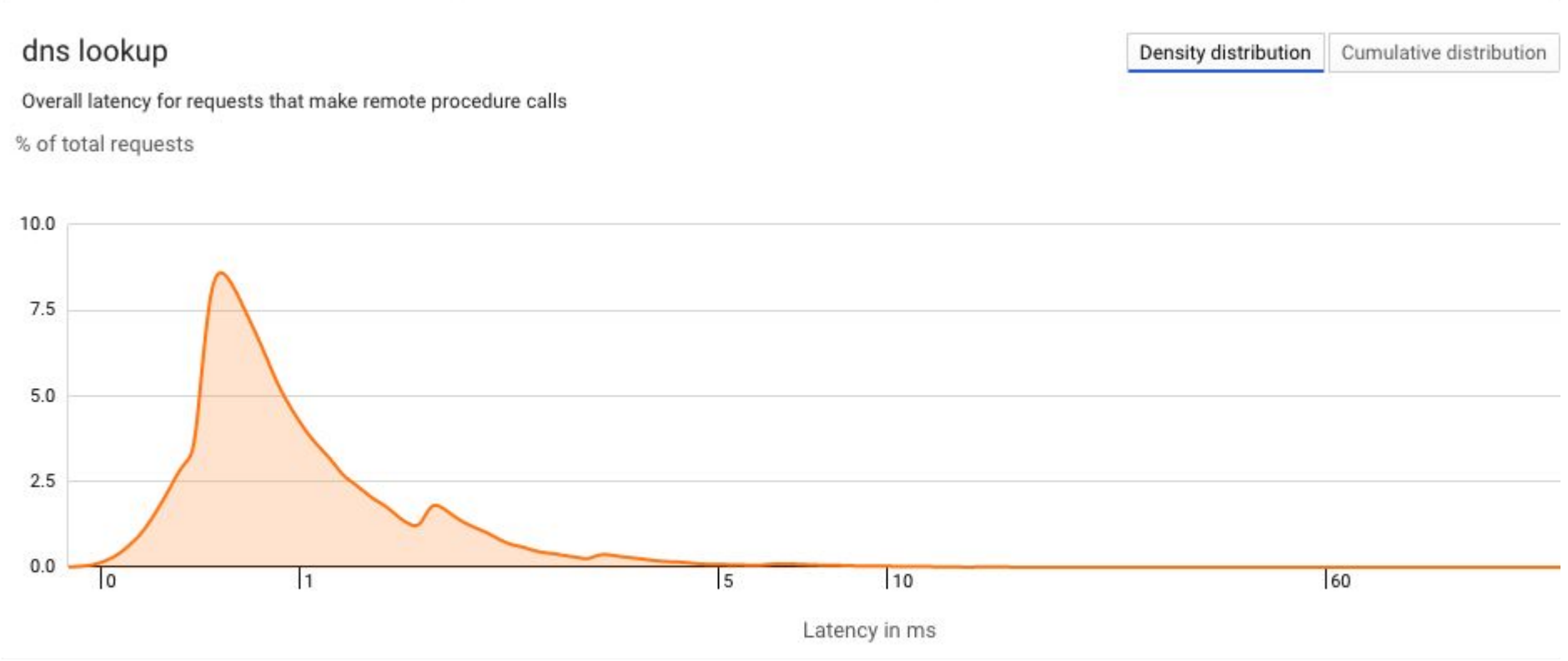
Improved latencies - test 2



node-local-dns + kube-dns(on the right)

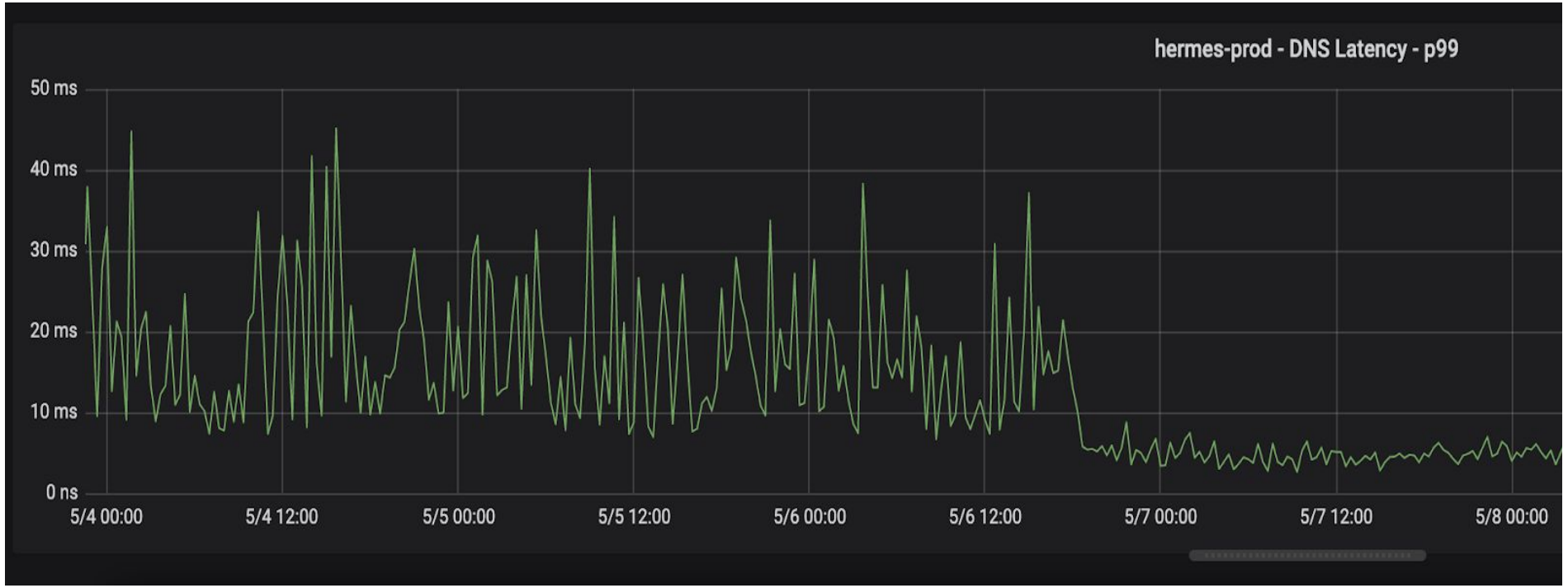
- 3 nodes (size: 2 cores, 7.5GB memory)
- 2 kube-dns pods
- 5 konfirm-dns pods, each doing 10k QPS + searchpath(100k QPS)
- Queries 5 different hostnames(external + within cluster)
- Test image: gcr.io/pavithrar-k8s-dev/dns-stress-test:latest

Report view


dns lookup
Density distribution
Cumulative distribution
Overall latency for requests that make remote procedure calls
% of total requests
10.0
7.5
5.0
2.5
0.0
0
1
5
10
60
Latency in ms

Cache settings improvements

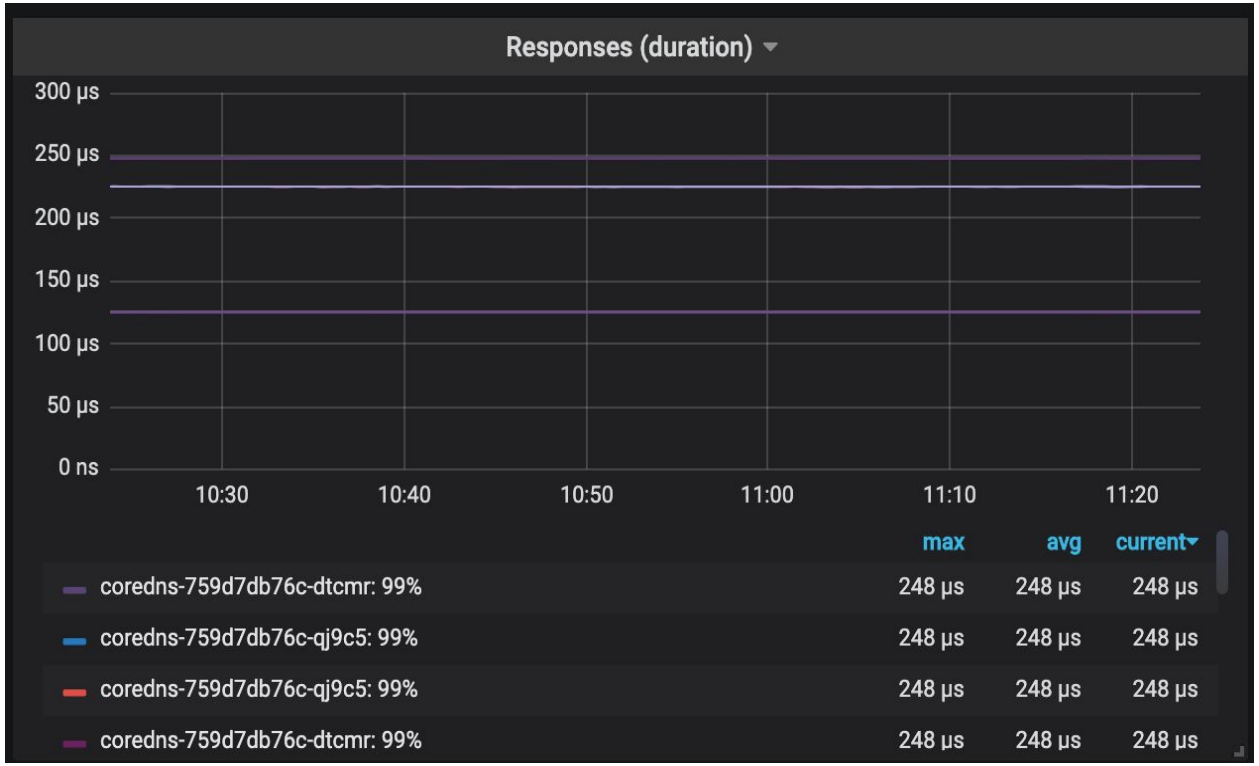
Before prefetch on the left, after on the right:

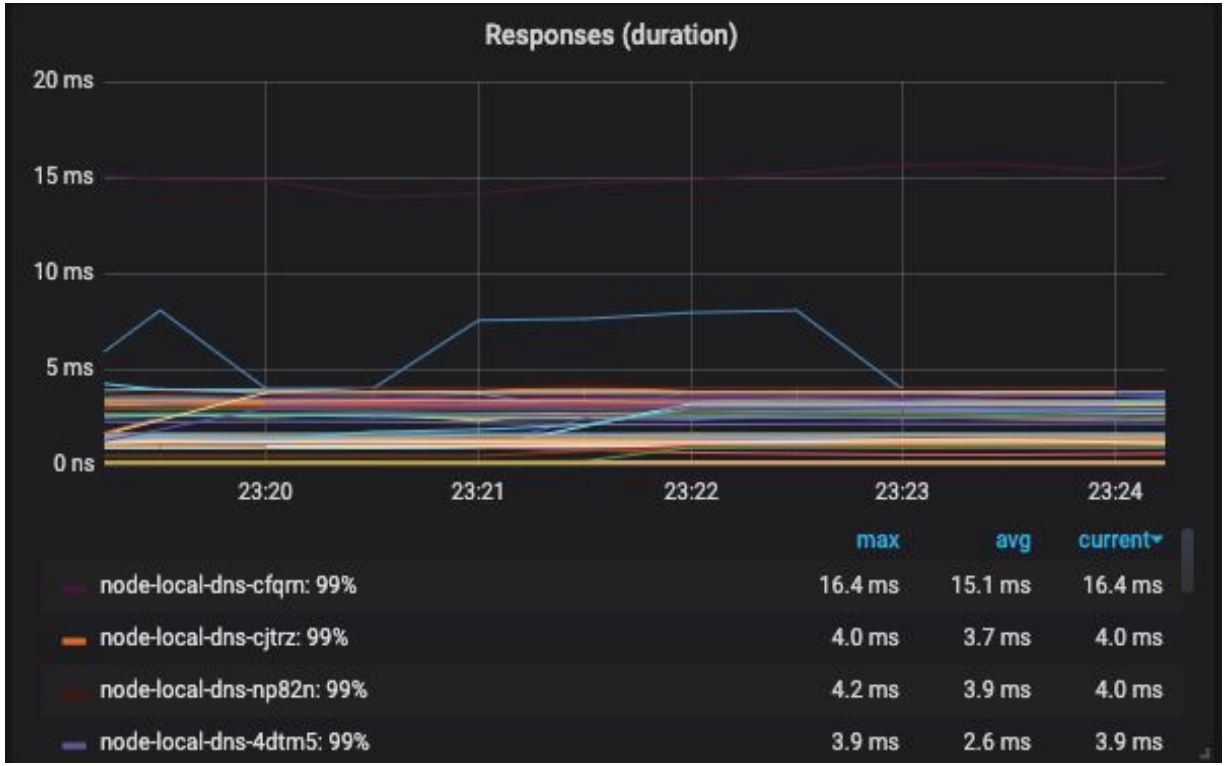


example config:

```
.:53 {
    errors
    prometheus :9153
    forward . /etc/resolv.conf
    cache {
        success 9984
        denial 9984
        prefetch 1 1h 50%
    }
    loop
    reload
}
```

Production stats - CoreDNS + nodelocal





Production deployment

- cluster-dns: coredns
- client images - mix of alpine, debian, distroless
- ~50,000 queries per second at peak
- Researching shared cache to reduce node startup time slowness.

Steps to run NodeLocalDNS

1) On GCE, the 1.13 or above cluster can be created using:
   KUBE_ENABLE_NODELOCAL_DNS=true go run hack/e2e.go -v --up
2) On any setup(any cluster version), deploy the yaml via kubectl.
   Example:
   https://github.com/kubernetes/kubernetes/issues/56903#issuecomment-485353223
   Latest yaml here:
   https://github.com/kubernetes/kubernetes/blob/8ae998ceb69ae83afe730795aea3bd44913ad868/cluster/addons/dns/nodelocaldns/nodelocaldns.yaml
3) Requires changing the --cluster-dns flag to kubelet.
   Can be run without cluster-dns flag change - https://github.com/kubernetes/dns/pull/280 (Thanks Justin Santa Barbara!)

Future Work

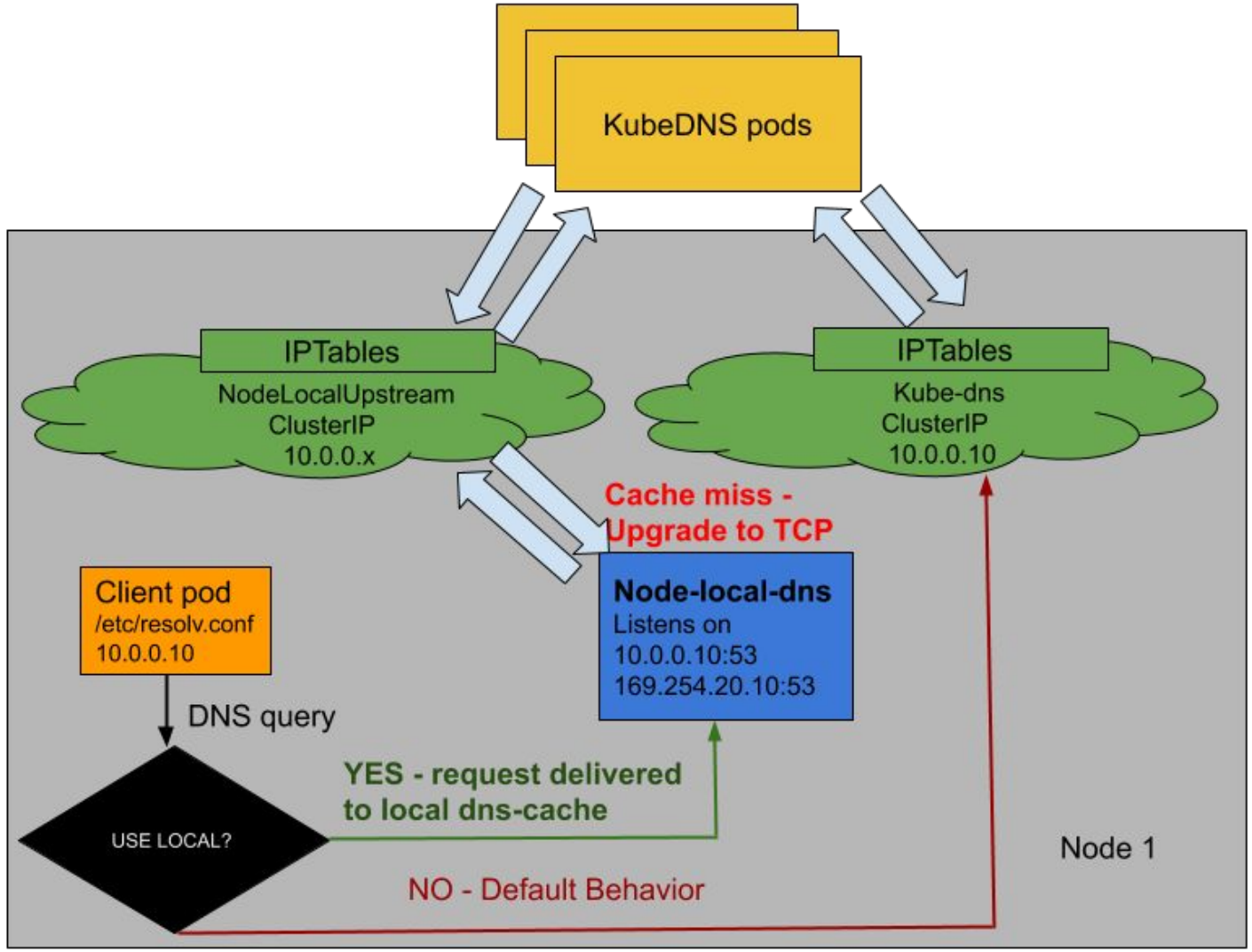
1) Feature graduating to beta in 1.15.

2) Got great feedback from users, please share your experience.

3) HA, Autopath

Future Work - HA

Link to KEP: https://github.com/kubernetes/enhancements/blob/master/keps/sig-network/20190424-NodeLocalDNS-beta-proposal.md



Future Work - Autopath

Proposal to move searchpath completion to server side.
Link to KEP in progress :
https://github.com/kubernetes/enhancements/pull/967

Benefits are:

- Reduced number of queries originated by client pod.
- Client pod also does not need to know about the searchpath schema that Kubernetes follows.
- Can be applied outside of Kubernetes clusters as well.
- CoreDNS has similar functionality in the autopath plugin, the proposed approach is less resource-intensive.

Autopath - Proposal

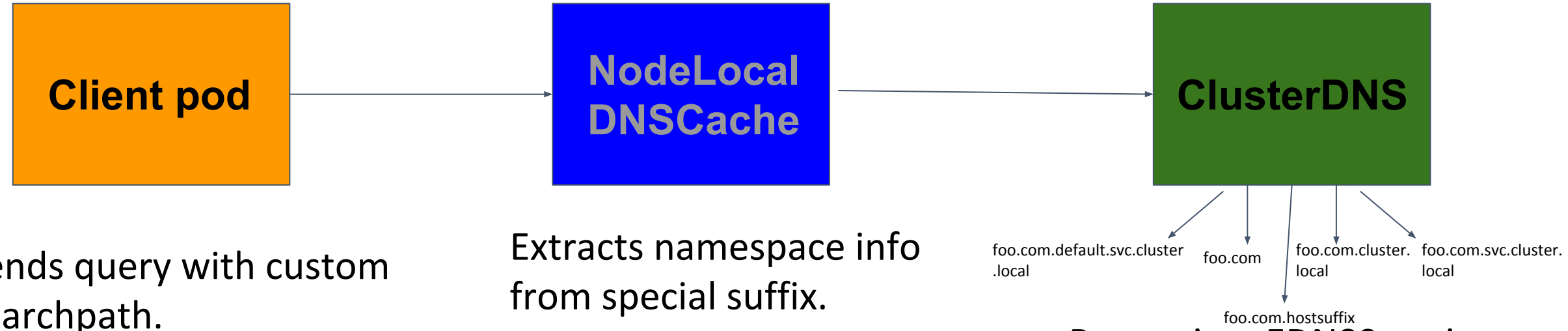
Introduce a new dnsPolicy “ClusterFirstWithAutopath” *

In this mode, kubelet generates a single searchpath*:
`search.$NS.$SUFFIX.k8s-v1`

$NS is the namespace of the pod,
$SUFFIX is the cluster suffix(cluster.local by default).

** work in progress, to be finalized*

Autopath - Workflow



Sends query with custom searchpath.
Example:
“foo.com.default.cluster.local.k8s-v1”

Extracts namespace info from special suffix.

Rewrites the query with just the base query name “foo.com” and searchpath strings as an EDNS0 option.

Recognizes EDNS0 option and performs multiple queries and returns all non-empty responses.

Questions

KubeCon

CloudNativeCon

Europe 2019

Thank You!