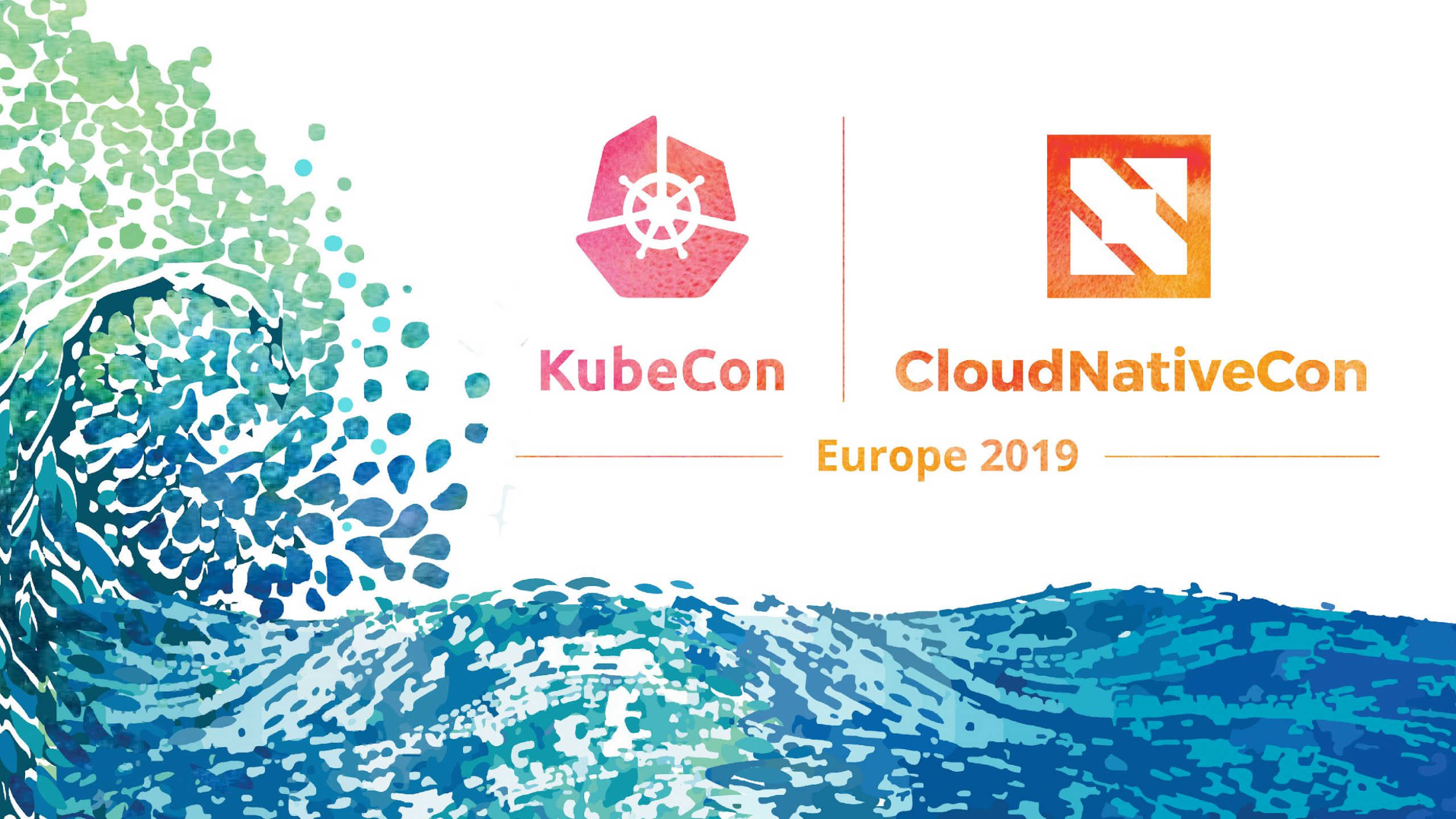

KubeCon
CloudNativeCon
Europe 2019

Scale Kubernetes Service Endpoints 100X

Wojciech Tyczynski, Staff Software Engineer, Google
Minhan Xia, Software Engineer, Google

Current pain points

1. Limit for # of endpoints in a service

2. Performance degradation in large clusters

Existing Endpoints API


```
type Endpoints struct {
        metav1.TypeMeta `json:",inline"`
        // Standard object's metadata.
        // More info: https://git.k8s.io/community/contributors/devel/api-conventions.md#metadata
        // +optional
        metav1.ObjectMeta `json:"metadata,omitempty" protobuf:"bytes,1,opt,name=metadata"`

        // The set of all endpoints is the union of all subsets. Addresses are placed into
        // subsets according to the IPs they share. A single address with multiple ports,
        // some of which are ready and some of which are not (because they come from
        // different containers) will result in the address being displayed in different
        // subsets for the different ports. No address will appear in both Addresses and
        // NotReadyAddresses in the same subset.
        // Sets of addresses and ports that comprise a service.
        // +optional
        Subsets []EndpointSubset `json:"subsets,omitempty" protobuf:"bytes,2,rep,name=subsets"`
}
```

Endpoints per service limit

# of backend pods: P

Size of Endpoints object: O(P)

Max size of etcd object

1.5 MB ≈ O(5000) endpoints

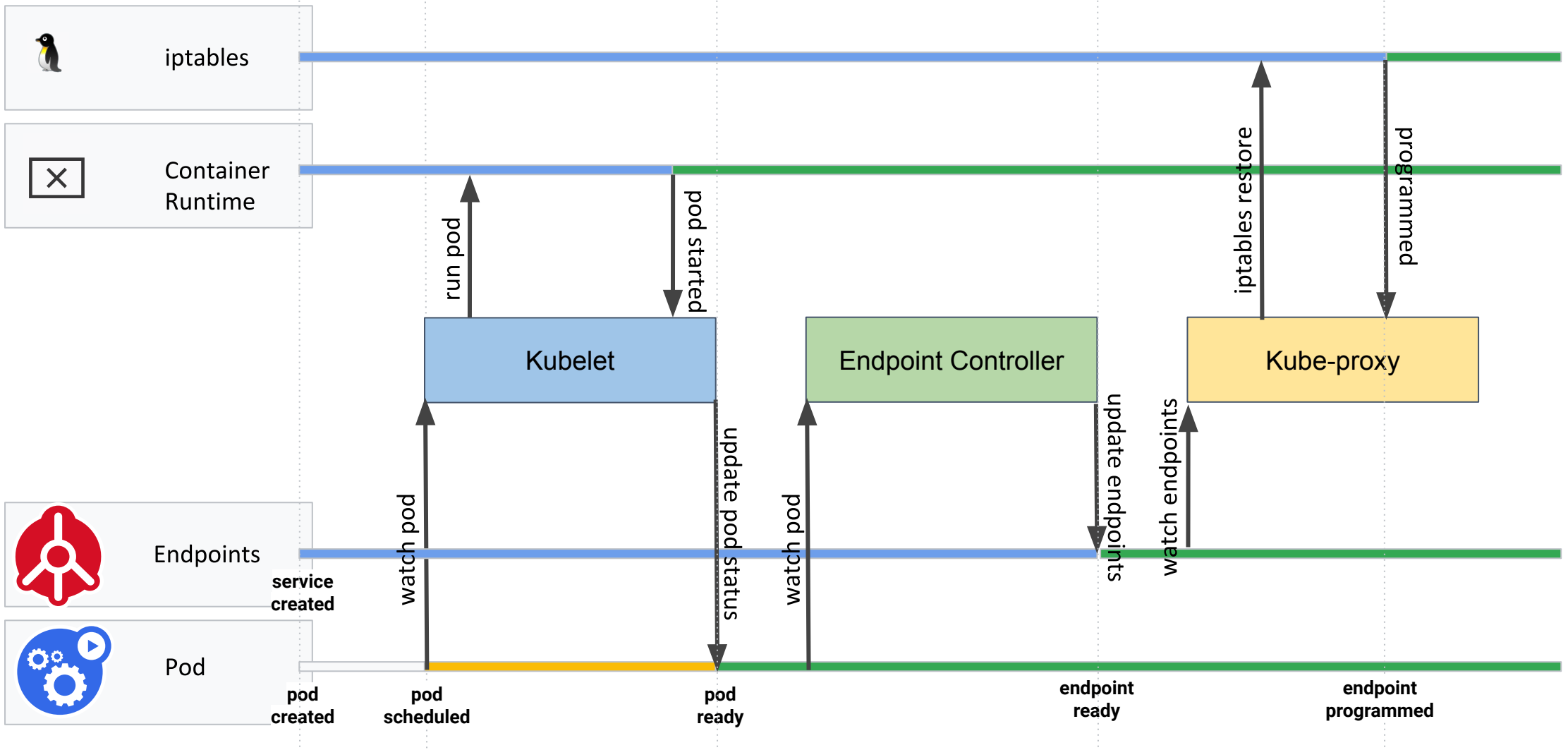
Max size of etcd object

if endpoints object > 1.5 MB:

Endpoints per service limit

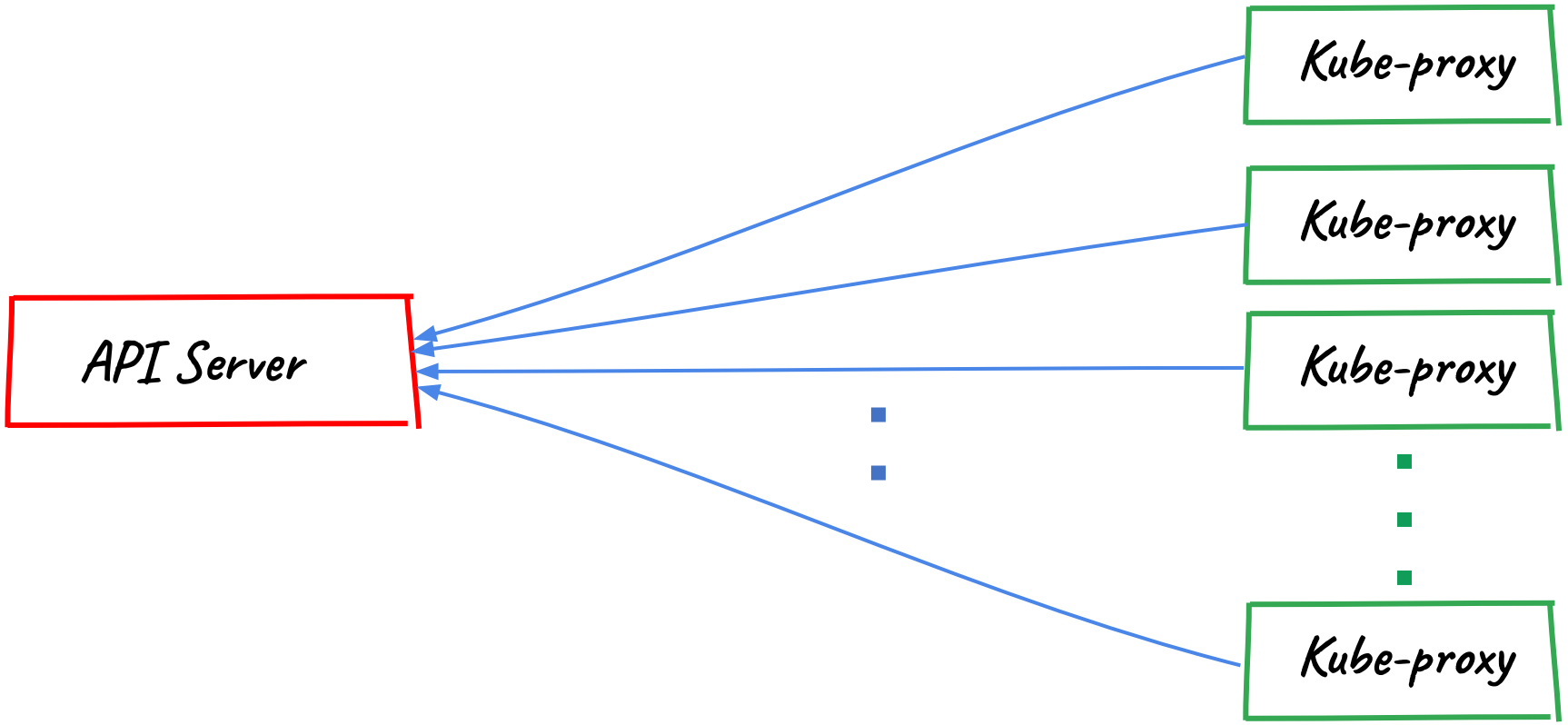
- Size of object grows linearly with #endpoints
- Reaching default limit for object size in etcd

Service Control Flow



Performance Degradation

GET /api/v1/endpoints?watch=true&...



Performance Degradation

# of nodes: N

# of watchers: N

# of object copies per update: N

Performance Degradation

# of backend pods: P

Size of Endpoints object: O(P)

# of object copies per update: N

total bytes transmitted per update: O(NP)

Estimation

# of nodes: 5000

Size of Endpoints object: 1 MB

total bytes transmitted per update:

5000 X 1 MB = 5GB

DVD?

Estimation

total bytes transmitted per update: 5GB

rolling update?

~5000 X 5 GB = 25TB !

User Expectations

- 10k+ endpoints/service
- Large Cluster
- High churn within a service

Just Works!

Solution

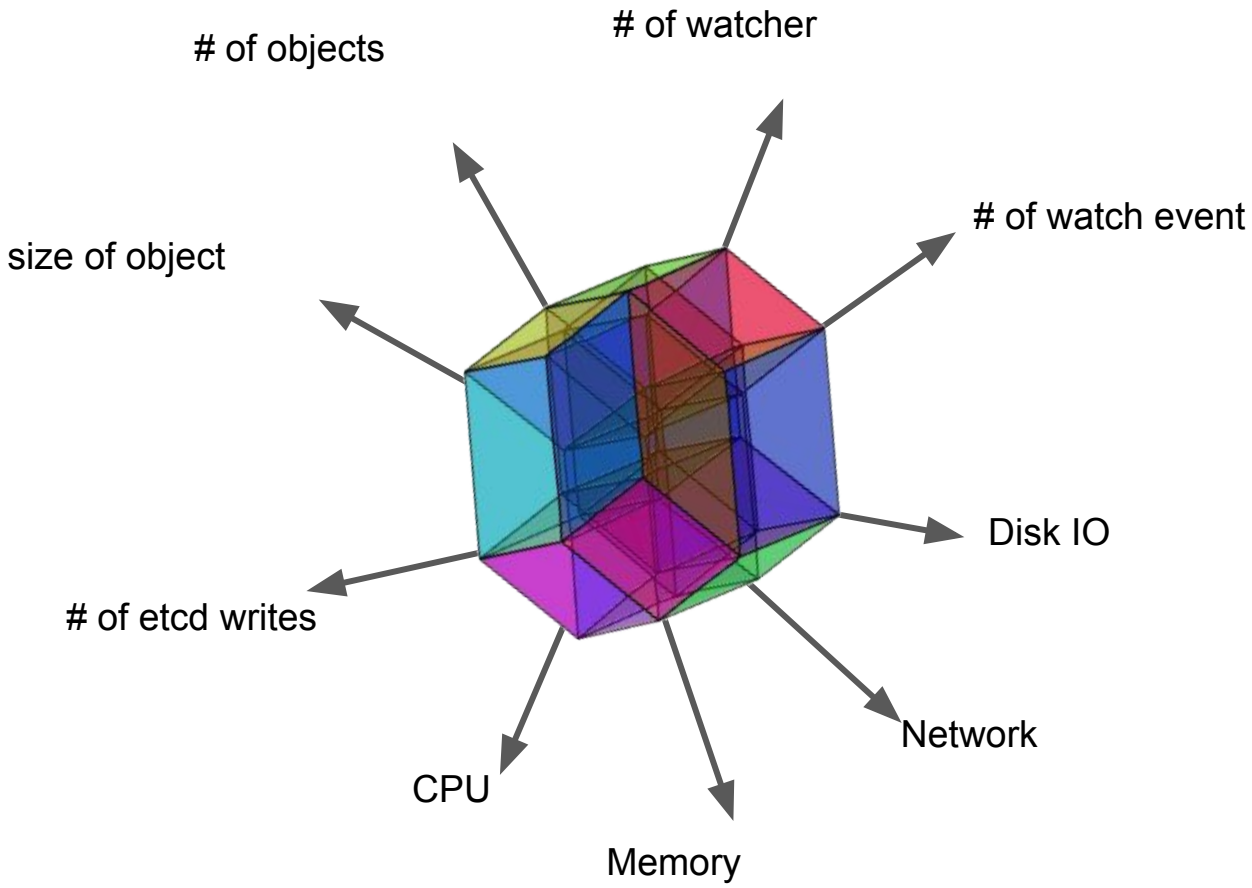
Redesign the API

Goals of the redesign

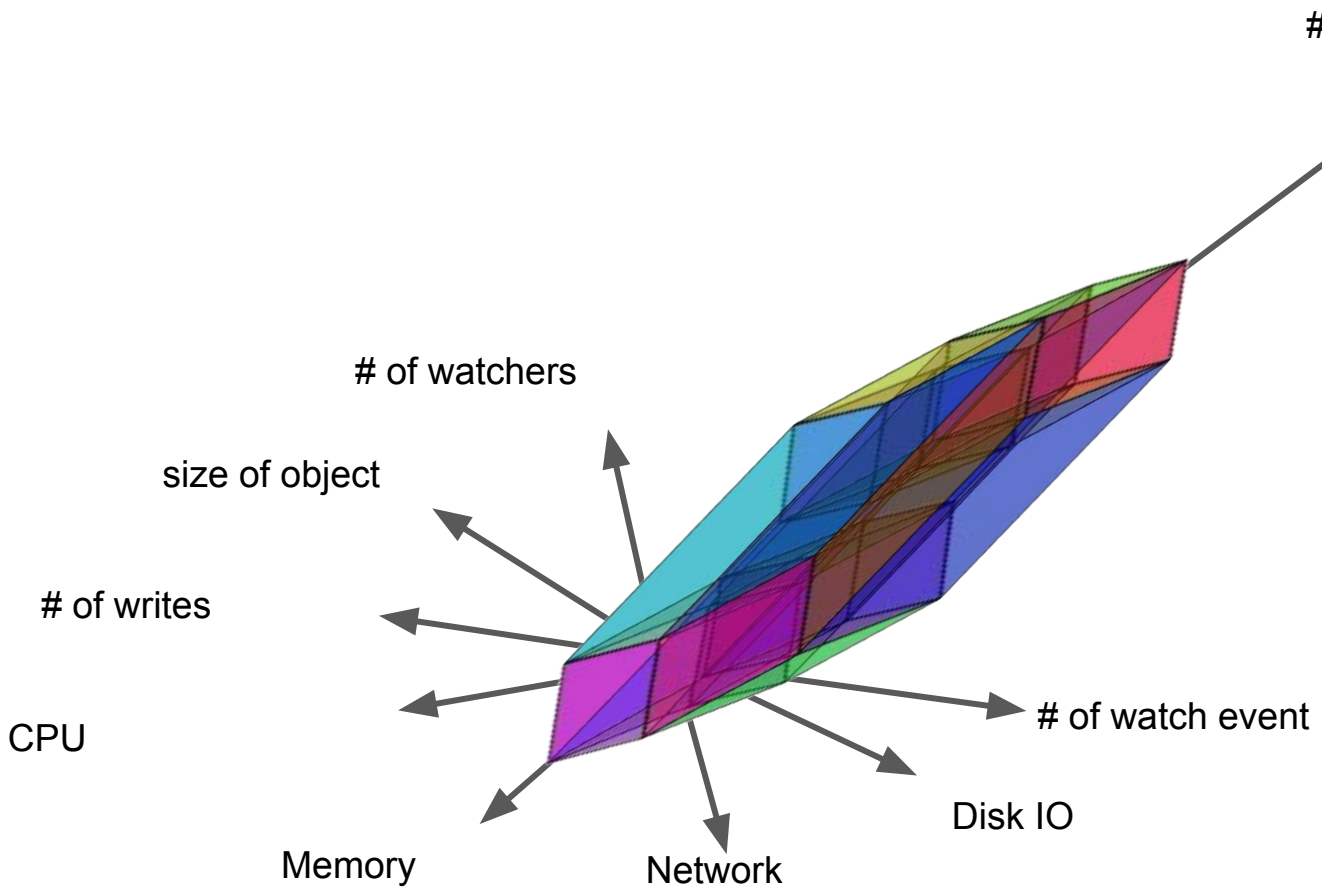
- Support **tens of thousands** of backend endpoints in a cluster with **thousands** of nodes

- Enable future extensions, e.g.
  - Dynamic endpoints subsetting

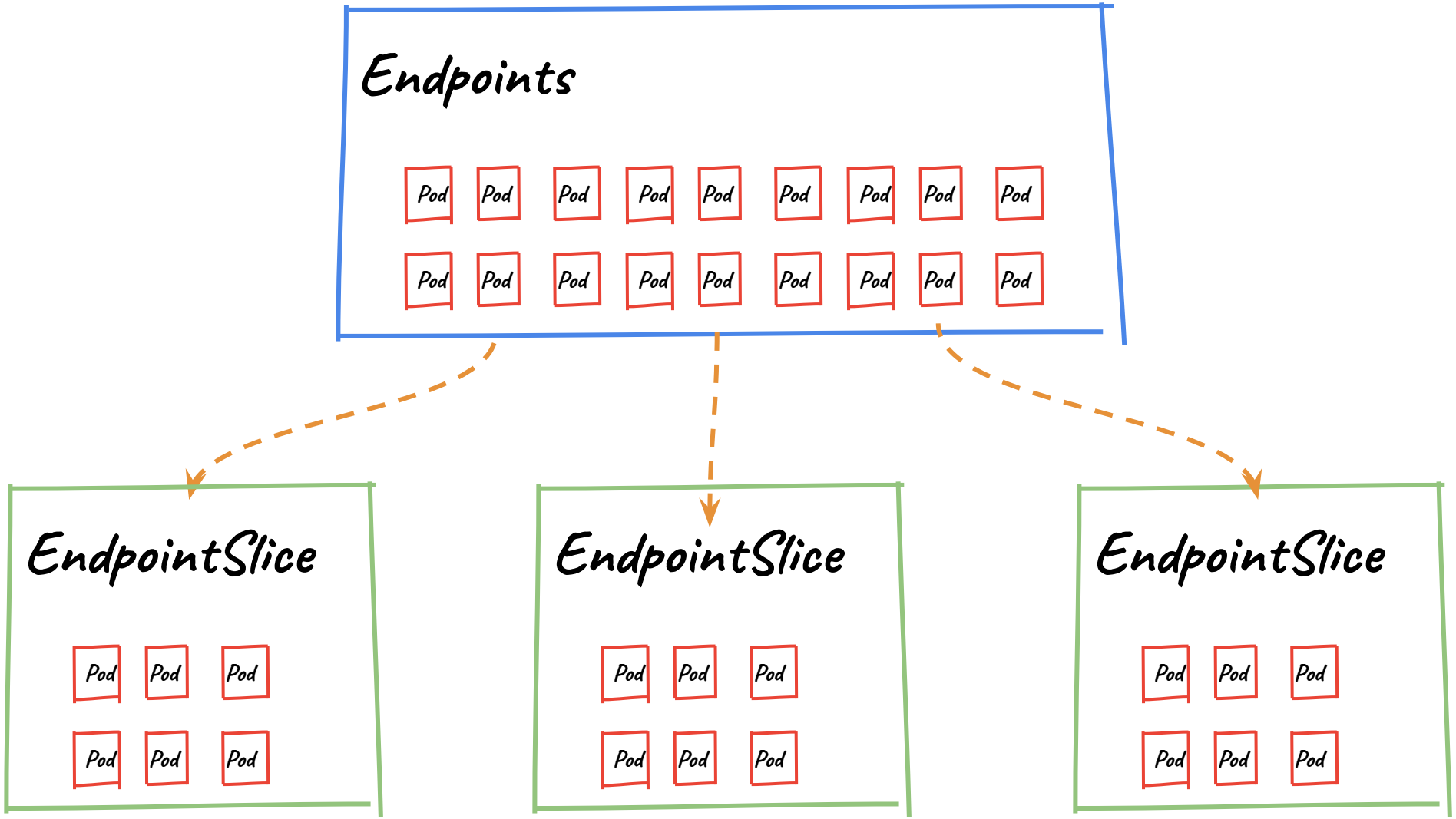
Scalability Constraints



Over Optimization



High-level idea



High-level idea

EndpointSlice contains **100 endpoints**

Configurable

EndpointSlice naming:

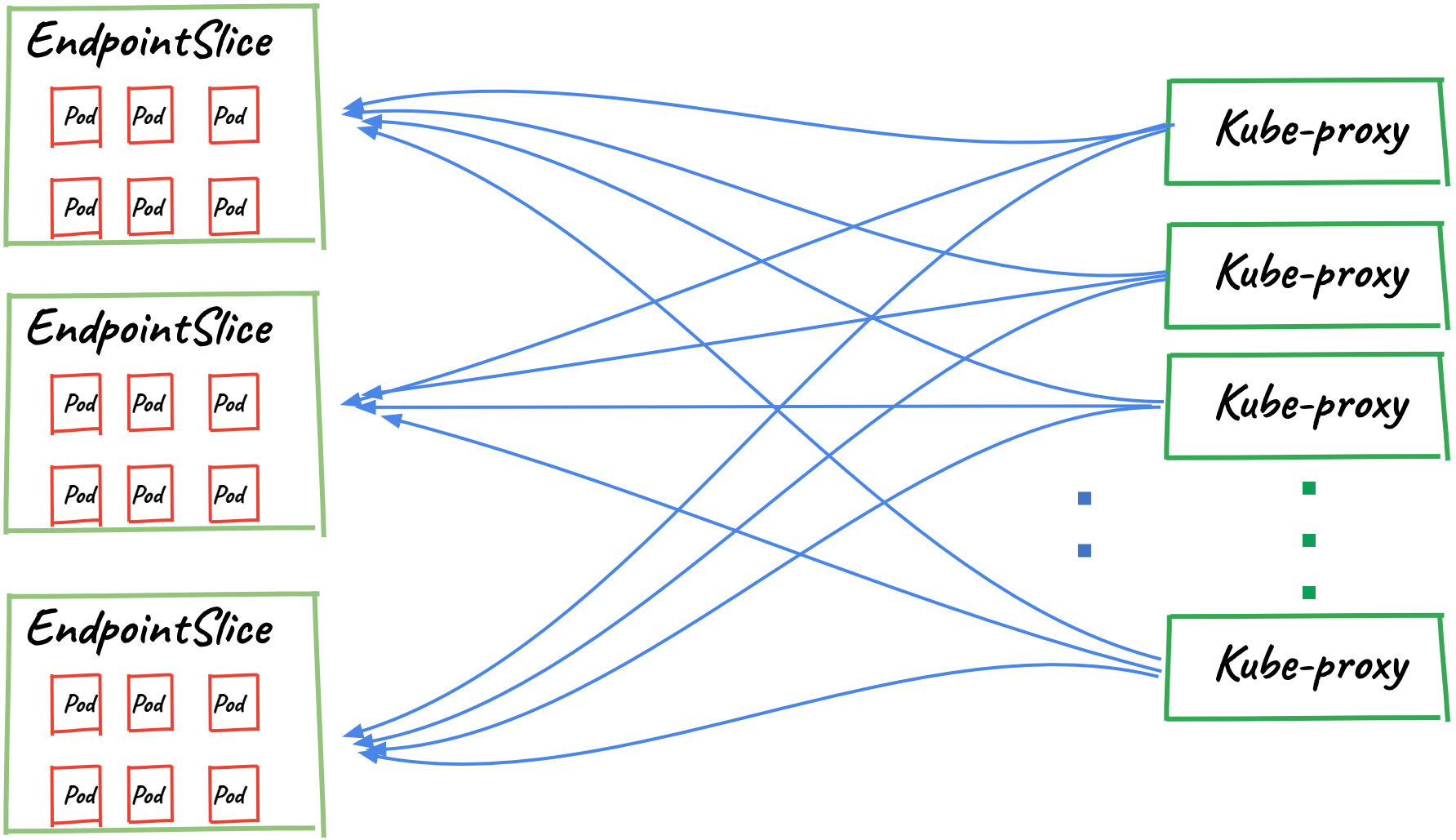
${service-name}-<random>

Service <-> EndpointSlice mapping

key: k8s.io/service

value: ${service-name}

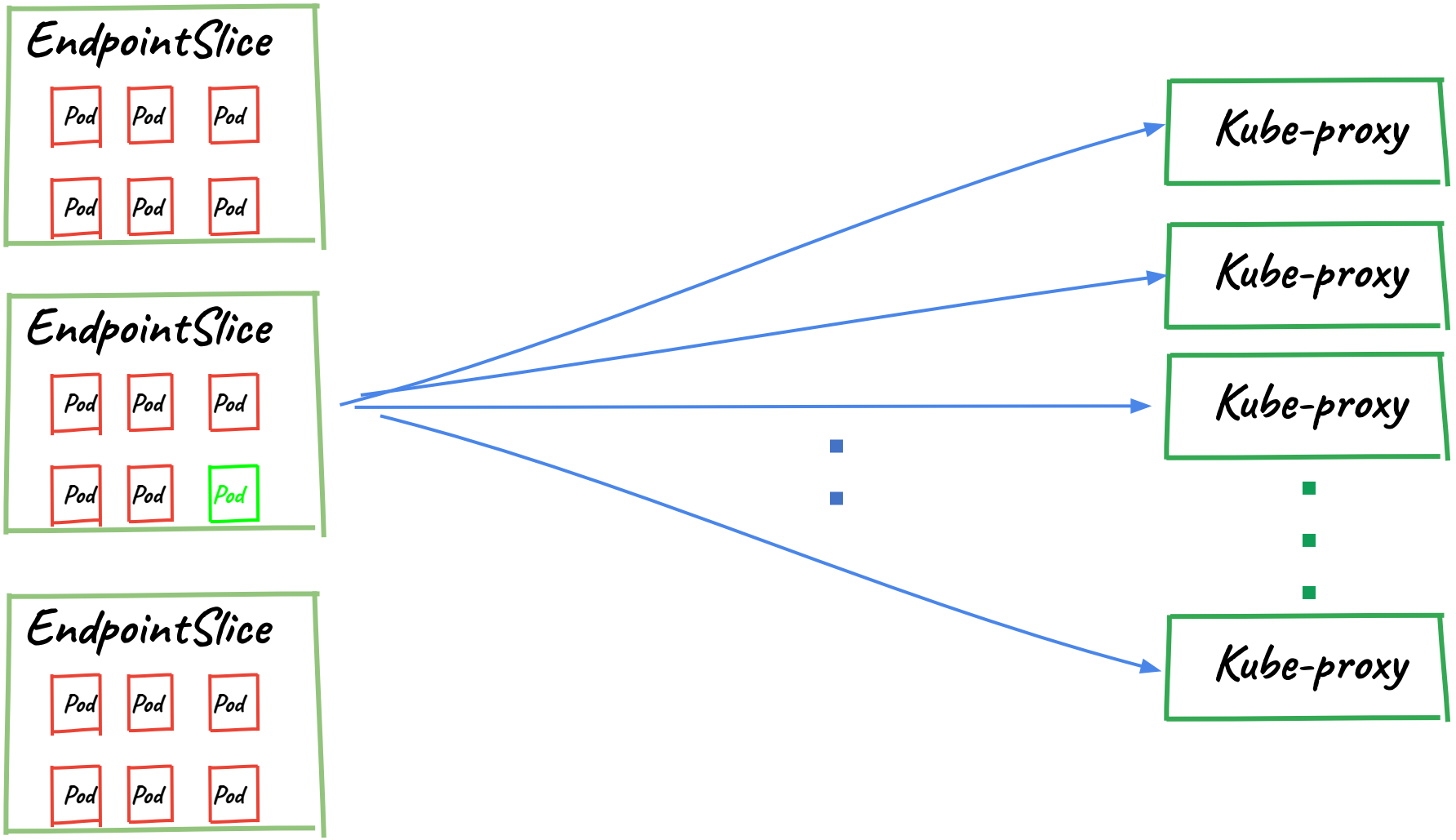
Endpoint Update



Endpoint Update

Endpoint Update

Endpoint Update



Evaluation scenarios

- Single endpoint update
- Rolling upgrade of a service
- Service creation/deletion

Evaluation metrics

- # of mutating API calls
- size of single api object

- # of watch events per watcher
- total # of watch events
- total # bytes transmitted

Example Evaluation: Service Creation

Sample Case: **20,000 endpoints, 5,000 nodes**

# of Backend Pod: ***P***
# of Node: ***N***
# of Endpoint Per EndpointSlice: ***B***

Single Endpoint Update

	Endpoints	**100 Endpoints per EndpointSlice**	**1 Endpoint per EndpointSlice**
# of writes # of watch events per watcher	*O(1)* *1*	*O(1)* *1*	*O(1)* *1*
Size of API object Size of each watch event	*O(P)* *20k * const = ~2.0 MB*	*O(B)* *100 * const = ~10 KB*	*O(1)* *< ~1KB*
# of watchers per object	*O(N)* *5000*	*O(N)* *5000*	*O(N)* *5000*
# of total watch event	*O(N)* *5000*	*O(N)* *5000*	*O(N)* *5000*
Total Bytes Transmitted	*O(PN)* *~2.0MB * 5000 = 10GB*	*O(BN)* *~10k * 5000 = 50MB*	*O(N)* *~1KB * 5000* = ~5MB

Rolling Update

	Endpoints	**100 Endpoints per EndpointSlice**	**1 Endpoint per EndpointSlice**
# of writes # of watch events per watcher	*O(P)* *20k*	*O(P)* *20k*	*O(P)* *20k*
Size of API object Size of each watch event	*O(P)* *20k * const = ~2.0 MB*	*O(B)* *100 * const = ~10 KB*	*O(1)* *< ~1KB*
# of watchers per object	*O(N)* *5000*	*O(N)* *5000*	*O(N)* *5000*
# of total watch event	*O(NP)* *5000 * 20k*	*O(NP)* *5000 * 20k*	*O(NP)* *5000 * 20k*
Total Bytes Transmitted	*O(P^2N)* *2.0MB * 5000 * 20k = 200 TB*	*O(NPB)* *10KB * 5000 * 20k = 1 TB*	*O(NP)* *~1KB * 5000 * 20k = ~100 GB*

Service creation/deletion

	Endpoints	**100 Endpoints per EndpointSlice**	**1 Endpoint per EndpointSlice**
# of writes # of watch events per watcher	*O(1)* *1*	*O(P/B)* *200*	*O(P)* *20000*
Size of API object Size of each watch event	*O(P)* *20k * const = ~2.0 MB*	*O(B)* *100 * const = ~10 KB*	*O(1)* *< ~1KB*
# of watchers per object	*O(N)* *5000*	*O(N)* *5000*	*O(N)* *5000*
# of total watch event	*O(N)* *5000*	*O(NP/B)* *5000 * 200 = 1,000,000*	*O(NP)* *5000 * 20000 = 100,000,000*
Total Bytes Transmitted	*O(PN)* *2.0MB * 5000 = 10GB*	*O(PN)* *10KB * 5000 * 200 = 10GB*	*O(PN)* *~10GB*

Why not singular endpoint?

- Small service == single EndpointSlice
  - small conceptual change for users
- No way to batch updates
- Higher amplification of mutating API calls
  - one of limiting factors for scalability

Generic API

Change in Kubernetes scalability characteristic

=

Ability to change the logic
with **no API changes**

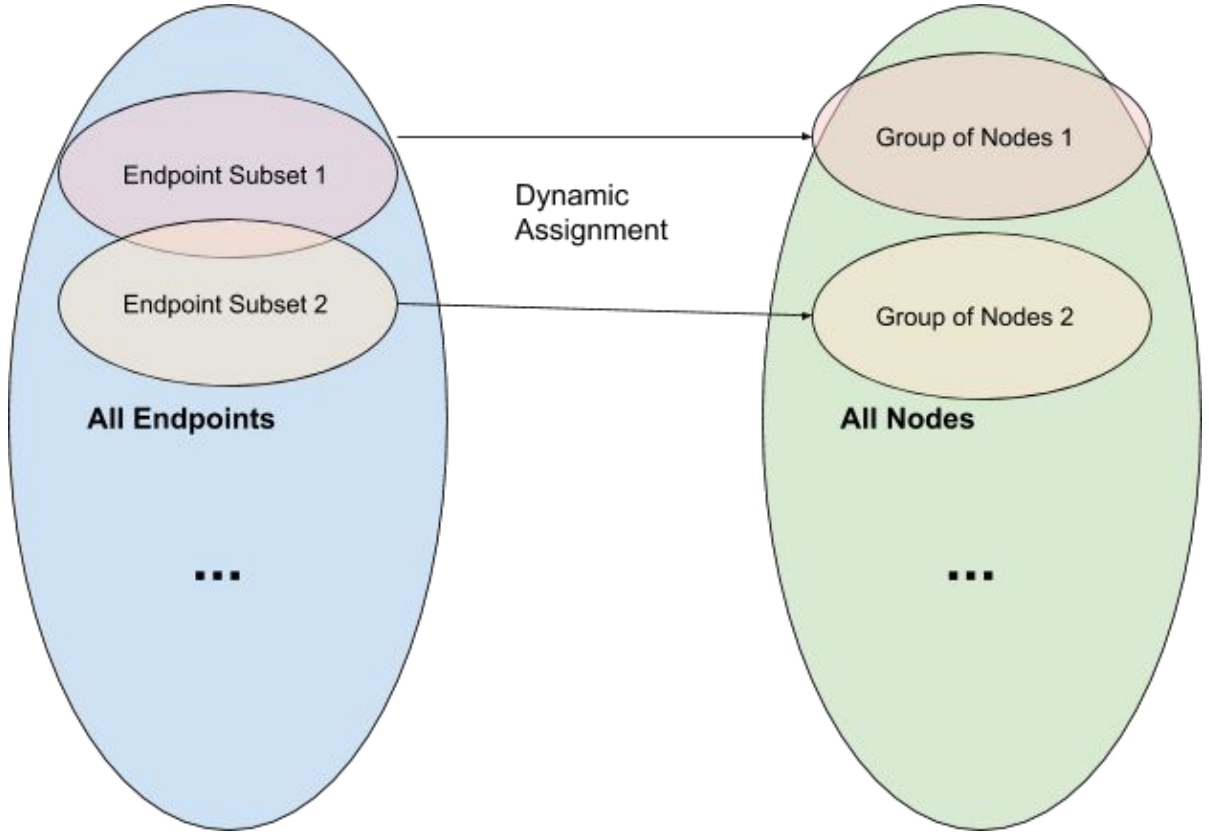
Future extensions

But other door are opening...

Dynamic Endpoints Subsetting

Avoid distributing all endpoints to all nodes

- Reduce per-node overhead
- Further reduce transmission overhead on endpoint update

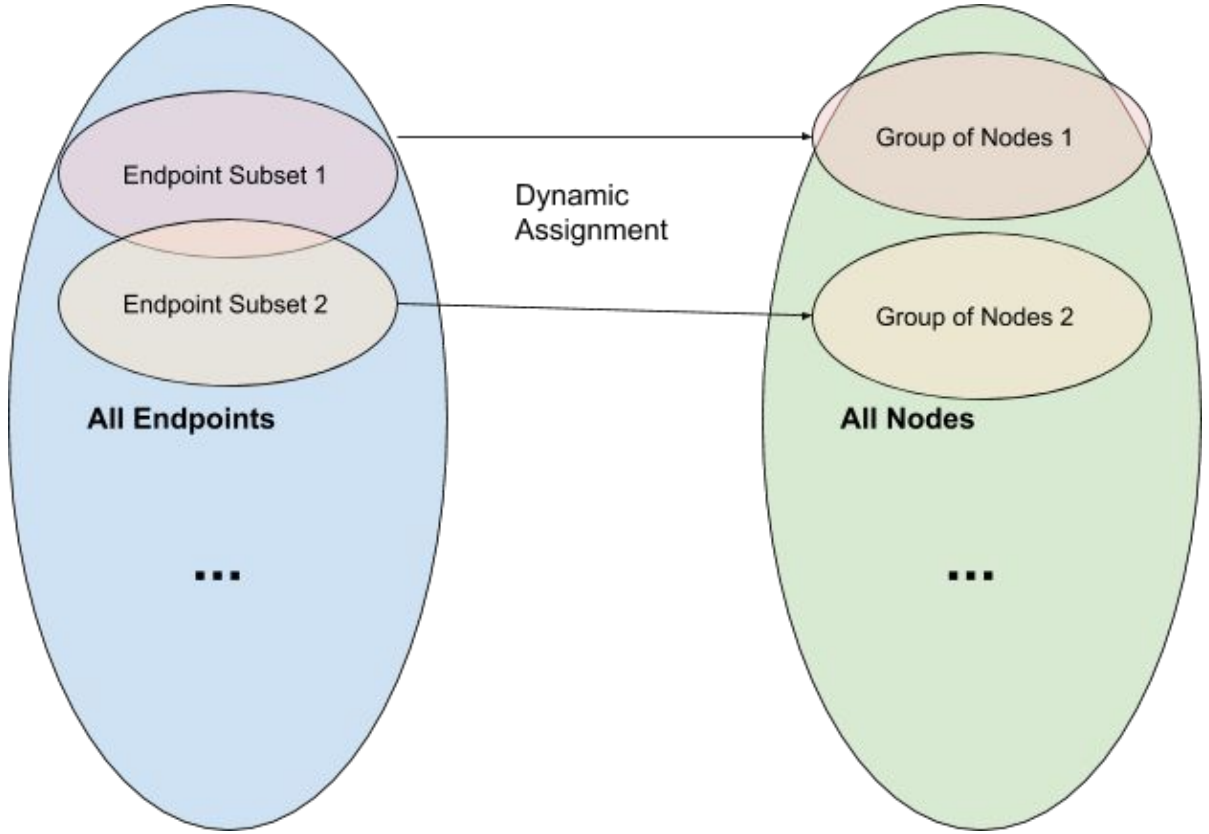


Dynamic Endpoints Subsetting

Not urgent, and hard

- Unknown client source
- Mutating API calls amplification

API will be ready,
we will get to it later.

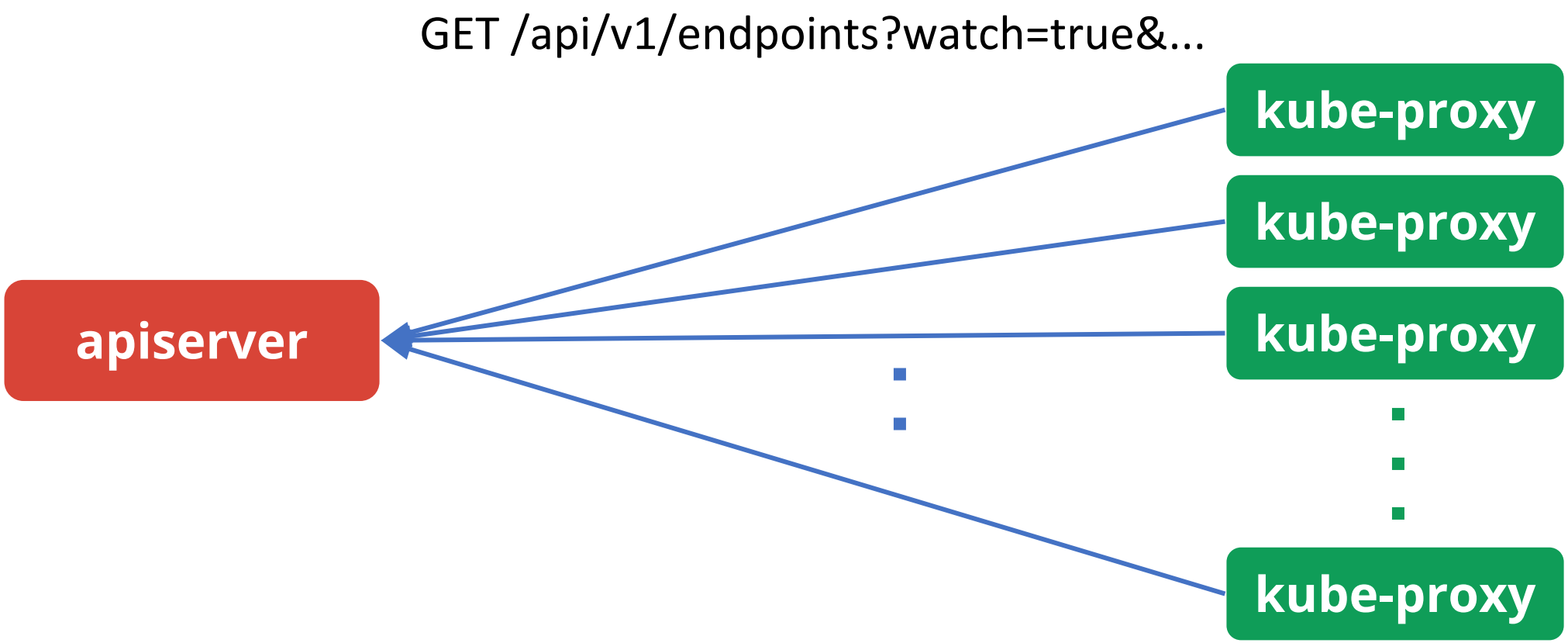


Join us to help!

Future extensions

TODO: This isn't finished, I think...

Performance degradation



High-level idea