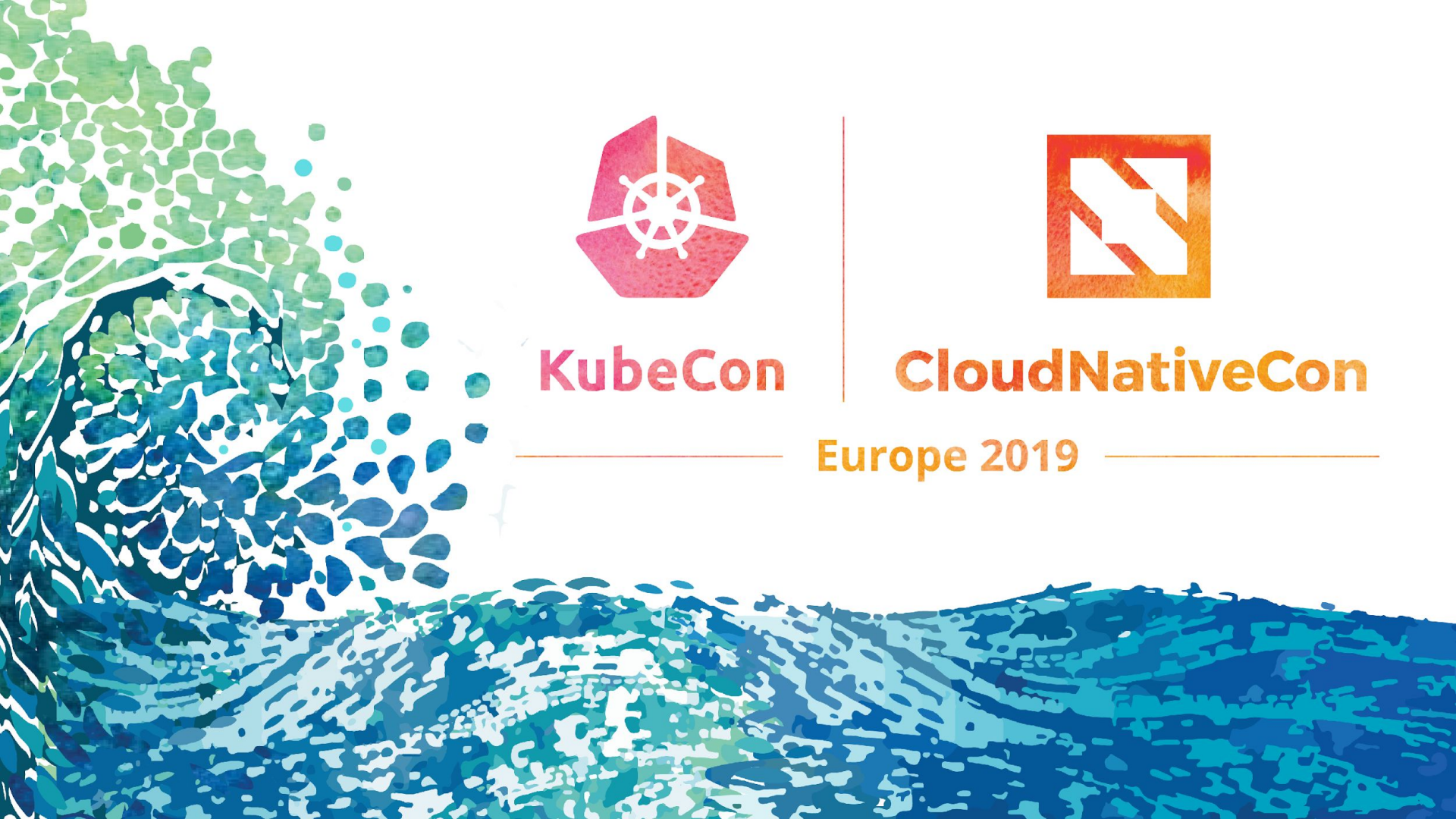

KubeCon
CloudNativeCon
Europe 2019

KubeCon | CloudNativeCon

Europe 2019

Latest Kubernetes Scalability Improvements

Yassine Tijani, VMware
(@yastij)

Shyam Jeedigunta, AWS
(@shyamjvs)

Background

Kubernetes *STARTED SCALING* to large clusters a while ago

Bigger clusters gained popularity

Usage patterns exposed some newer bottlenecks

So...

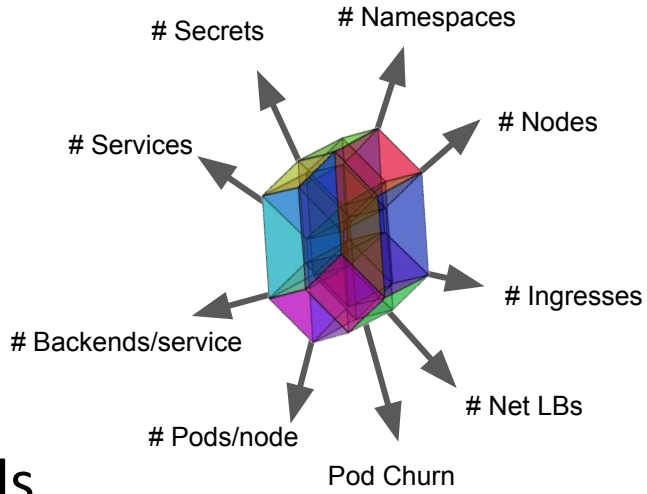
Background

We moved to scalability definition 2.0,

“Scalability is a multi-dimensional problem”
(for more see this past talk - https://sched.co/GrXy)

And…

Started working on improving various bounds



So.. What did we improve?

Too many node revisions

Problem:

- Too many node revisions in large clusters due to node heartbeats
- Made worse if too many images or volumes on the node
- Etcd disk fills up, causing NoSpace alarm
- Writes can’t happen anymore

Dimensions that suffer:

- #Nodes
- #Images/node
- #Volumes/node

Too many node revisions

How we solved?

- Split node objects from heartbeats
- Split node status updates from heartbeat updates
- Use new lease API to signal node heartbeats
- Continue using node status update also as liveness signal
- [Long-term] Reduce node-status update frequency to 10m

Feature availability:
1.13 (alpha)
1.14 (beta)

Node Lease API


```
apiVersion: coordination.k8s.io/v1
kind: Lease
metadata:
  creationTimestamp: 2019-04-16T13:12:35Z
  name: node-foo
  namespace: kube-node-lease
…
…
spec:
  holderIdentity: node-foo
  leaseDurationSeconds: 40
  renewTime: 2019-05-03T15:19:32.136799Z
```

heartbeats

```
status:
  …
  conditions:
    -       lastHeartbeatTime: "2019-05-05T18:30:46Z"
            lastTransitionTime: "2019-05-05T18:30:46Z"
            message: NodeController create implicit route
            reason: RouteCreated
            status: "False"
    -       lastHeartbeatTime: "2019-05-05T18:31:15Z"
            lastTransitionTime: "2019-05-05T18:30:46Z"
            message: kubelet is posting ready status. AppArmor enabled
            reason: KubeletReady
            status: "True"
            type: Ready
    …
  images:
    -       …
  volumesInUse
    -       …
  volumesAttached
    -       …
```

node status

Kubelet polling configs

Problem:

- Kubelet periodically polls secrets/configmaps it needs
- Can lead to many `GET secret/configmap` API calls
- Can eat away significant chunk of apiserver request queue
- Caching and reducing poll frequency, used as stopgaps

Dimensions that suffer:

- #Nodes
- #Secrets/node
- #Configmaps/node

Kubelet polling configs

How we solved?

- Switch kubelet to watch individual secrets/configmaps

Feature availability:

Enabled till 1.12.6 (but disabled from 1.12.7 due to golang bug)
Enabled till 1.13.4 (but disabled from 1.13.5 due to golang bug)
Enabled from 1.14.2 (with bug fixed by updating golang to 1.12)

Note: The bug is with kubelet TCP streams exhaustion if there are many (~250) configmaps/secrets needed by it

Iptables performance

Problem:

- Iptables maintains a linear list of rules, so:
  - Packet routing can be slow
  - Updating the rules can be slow
- Networking performance affected with number/size of services

Dimensions that suffer:

- #Services
- #Pods/services

Iptables performance

How we solved?

- IPVS as an alternative to iptables for kube-proxy
- IPVS uses hash tables, much more efficient than linear list
- Gives very high scale for load-balancing
- Missing some functionality and still to mature

Feature availability:
1.11 (GA)

Scheduling performance

Problem:

- Scheduling throughput is low on large clusters:
  - ~90/s in 2k-node cluster
  - ~30/s in 5k-node cluster
- Scheduling throughput is very low when using pod anti-affinity:
  - < 5 pods/min in 5k-node cluster

Dimensions that suffer:

- Pod churn
- #Nodes

Scheduling performance

How we solved?

- Score only a percentage of nodes that were found feasible
- Improvement on affinity computation
- Pod scheduling Latency improved
  - improving the way we snapshot scheduler's cache

Feature availability:

- scoring/affinity improvements in 1.12
- cache snapshot improvements in 1.14

Events overload

Problem:

- Multiple scalability issues over the time
  - Events filling up etcd
  - Events overloading the apiserver
- Poor usability (more of UX problem)

Dimensions that suffer:

- #API calls
- #Nodes
- Pod churn

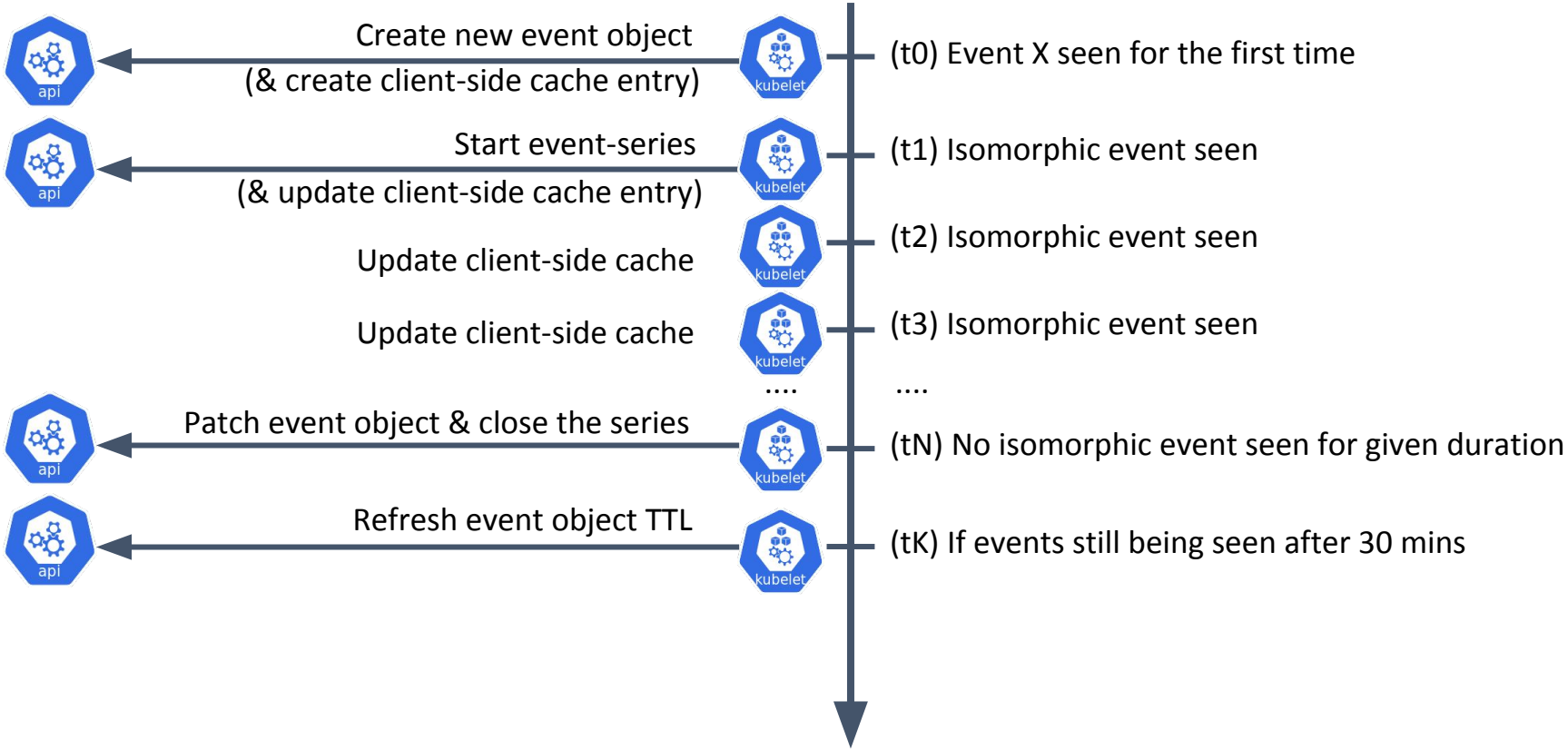
Events overload

How we solved?

- New Event API
- New deduplication logic that makes use of the new API
  - Concept of isomorphic Events
  - Avoid aggregation using an Event object

Reporting controller:	Kubelet
Reporting Instance:	node-foo
Regarding:	pod-bar
Related:	daemonset-foo
Type:	Warning
Action:	FailedCreatePodSandBox
Reason:	cannotAssignIP
Note:	Failed to create sandbox : ...

Events overload



Watch restart cost

Problem:

- Watch restart overloads apiserver if ResourceVersion is stale
- Users tend to get the famous “watch of v1.Foo ended with: too old resource version”

Dimensions that suffer:

- #Watches
- #API calls

Watch restart cost

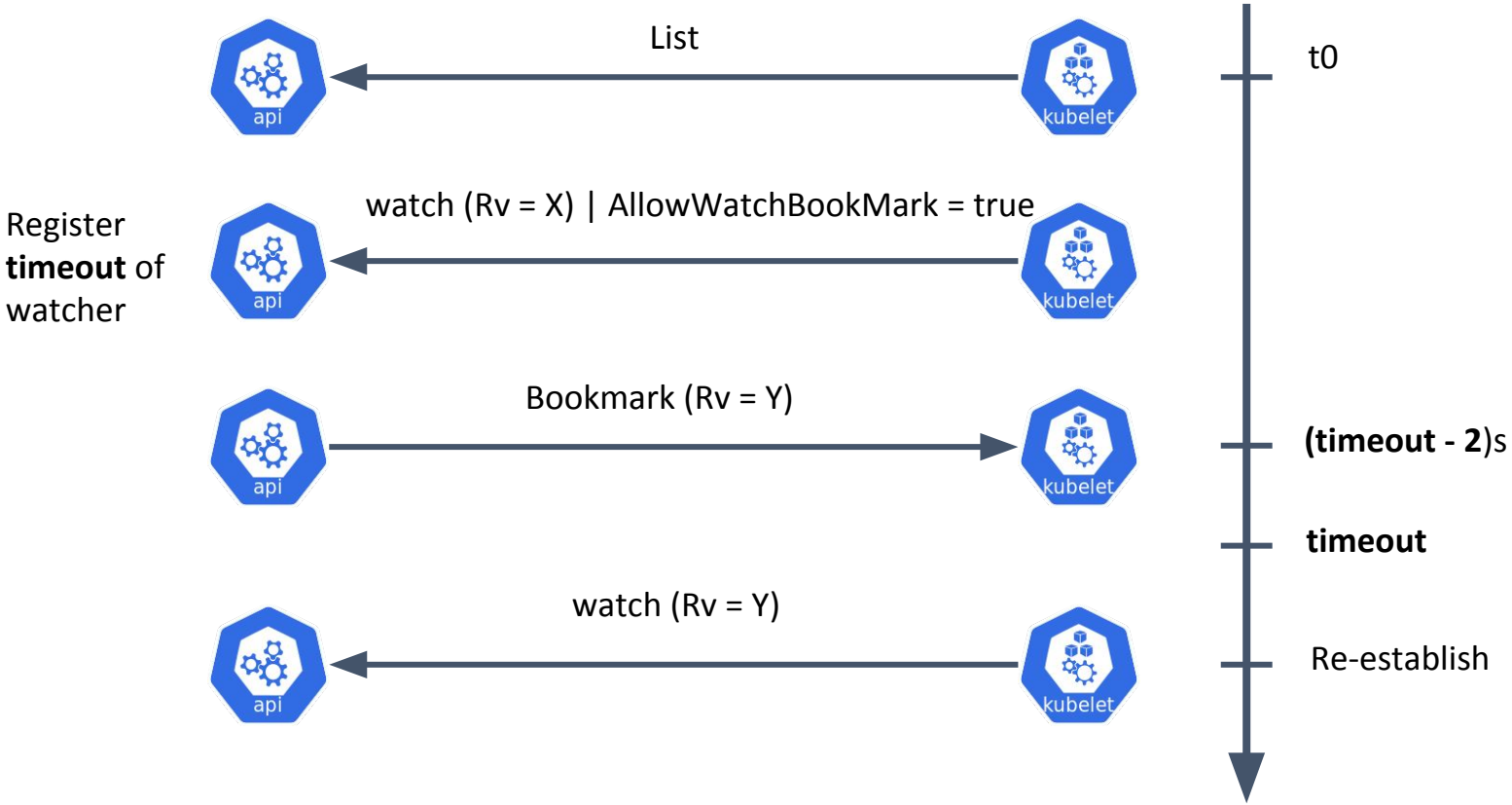
How we solved?

- Introduces a new Event type watch called “Bookmark”
- Tells client which resourceVersion watch has processed till
- Clients need to opt-in to receive bookmarks (backward compat)
- If opted in, they get bookmark 2s before watch timeout
- Initial testing shows 40x decrease in wasteful event processing

Feature availability:

- 1.15 (alpha)

Watch restart cost



What we plan to do next

Peep into the future

- New Endpoints API
- Allow for higher pod density
- Rethink affinity/anti-affinity
- Watch for arbitrary fields
- …


KubeCon
CloudNativeCon
Europe 2019
Thank you!