
KubeCon
CloudNativeCon
North America 2019

SIG Scalability

Intro + Deep Dive

Matt Matejczyk, Google (mm4tt@)

Wojtek Tyczyński, Google (wojtek-t@)

About the speakers

- Matt (mm4tt@) - SIG Scalability
- Wojtek (wojtek-t@) - SIG Scalability Chair

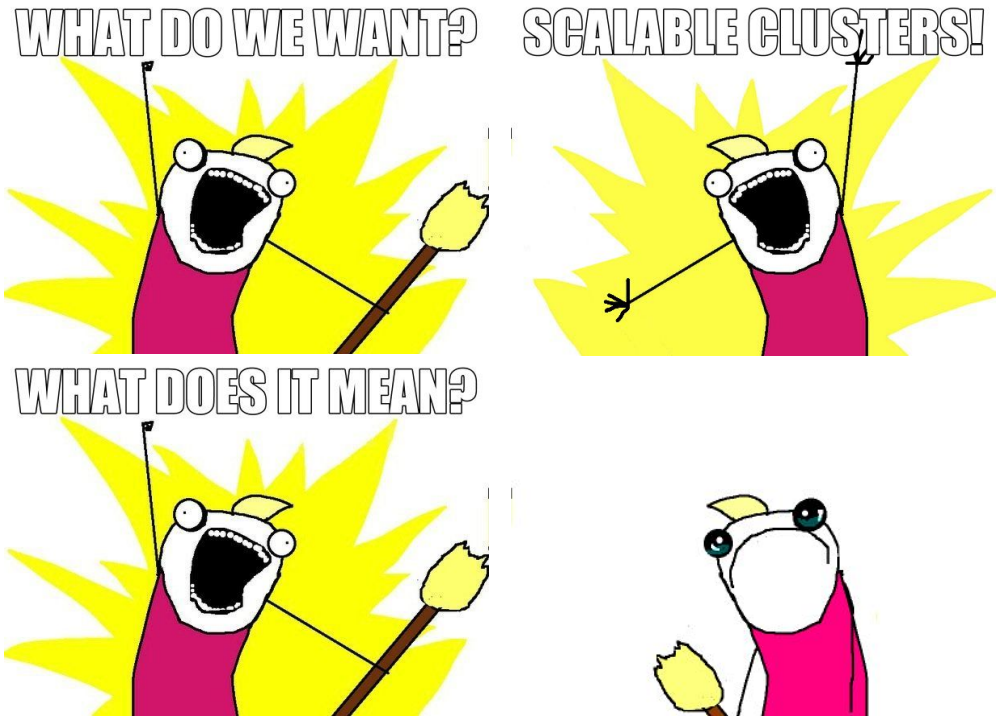
TL;DR

We:

- Define & Drive - scalability definition & goals
- Monitor & Measure - performance of the system
- Coordinate & Contribute - performance improvements
- Persevere & Protect - from scalability regressions
- Consult & Coach - community about scalability

Not to confuse with SIG Autoscaling!

What is Kubernetes Scalability?



What is Scalability?

"**Scalability** is the property of a system to handle a growing amount of work by adding resources to the system."

"In computing, scalability is a characteristic of computers, networks, algorithms, networking protocols, programs and applications. An example is a search engine, which must support increasing numbers of users, and the number of topics it indexes."

Wikipedia contributors, "Scalability," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Scalability&oldid=892100604 *(accessed Nov 05, 2019).*

What is (not) K8s Scalability?

Scalability is **not a single number** (like 5000).

Yes, we *"support"* up to 5000 nodes in k8s.

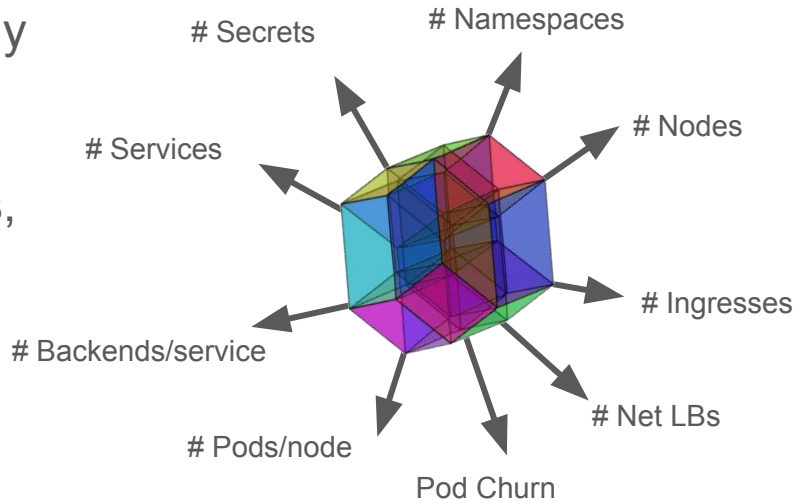
But that's not even close to the whole story!

K8s Scalability Envelope

In fact, scalability needs to be analyzed in many more dimensions.

This subscape has many interesting properties, e.g. it's NOT a cube, it's NOT convex.

Scalability Envelope - a *safe zone*, if you're within it, your cluster is *happy*.



Source of hypercube image:
http://www.gregegan.net/APPLETS/29/29.html

Scalability Envelope

Precisely computing the envelope boundaries is too ***hard*** problem.

Even if we could do that (test all possible configurations) we still need to define what it means that **kubernetes scales** for a given configuration.

To test scalability we need to define it first...

Scalability Definitions & Goals

Scalability - how to define it?

SLI - Service Level Indicator

SLO - Service Level Objective

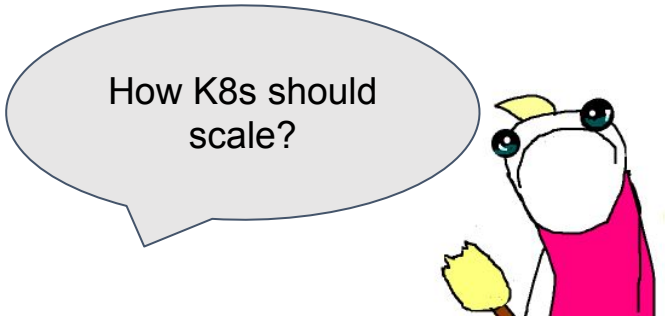
Scalability - how to define it?

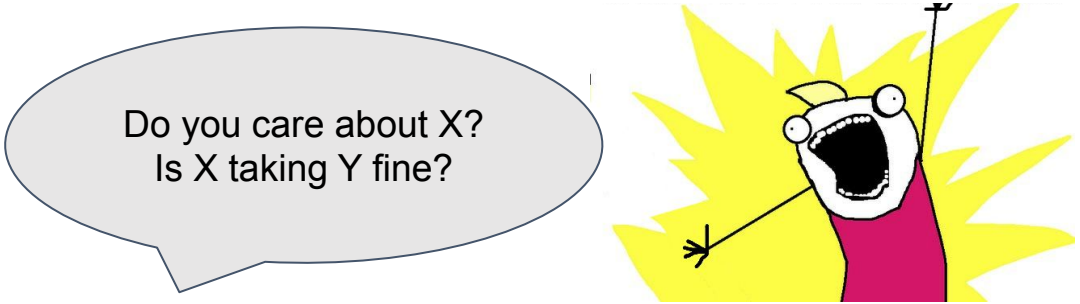
Cluster Scales

=

All Scalability SLOs satisfied

Scalability - SLO Coverage





SLI/SLO Principles

- User-oriented
- Testable
- Precise
- Well defined

Scalability SLIs / SLOs

Approved Scalability SLIs / SLOs

1. API Call Latency
    a. 99% of write API calls <= 1s latency
    b. 99% of read API calls:
        i. <= 1s latency (for GET)
        ii. <= 5s latency (for LIST in namespace)
        iii. <= 30s latency (for LIST across namespaces)
2. Pod Startup Latency (stateless pods)
    a. 99% startup latency <= 5s

Scalability SLIs / SLOs

WIP Scalability SLIs / SLOs

1. Pod Startup Latency (stateful pods)
2. In-Cluster Network Programming Latency
3. DNS Programming Latency
4. In-Cluster Network Latency
5. DNS Latency

More at github.com/kubernetes/community/tree/master/sig-scalability/slos

Scalability SLO Framework

You promise to

Correctly configure cluster
Stay within Scalability Envelope

We promise

Satisfied Scalability SLOs

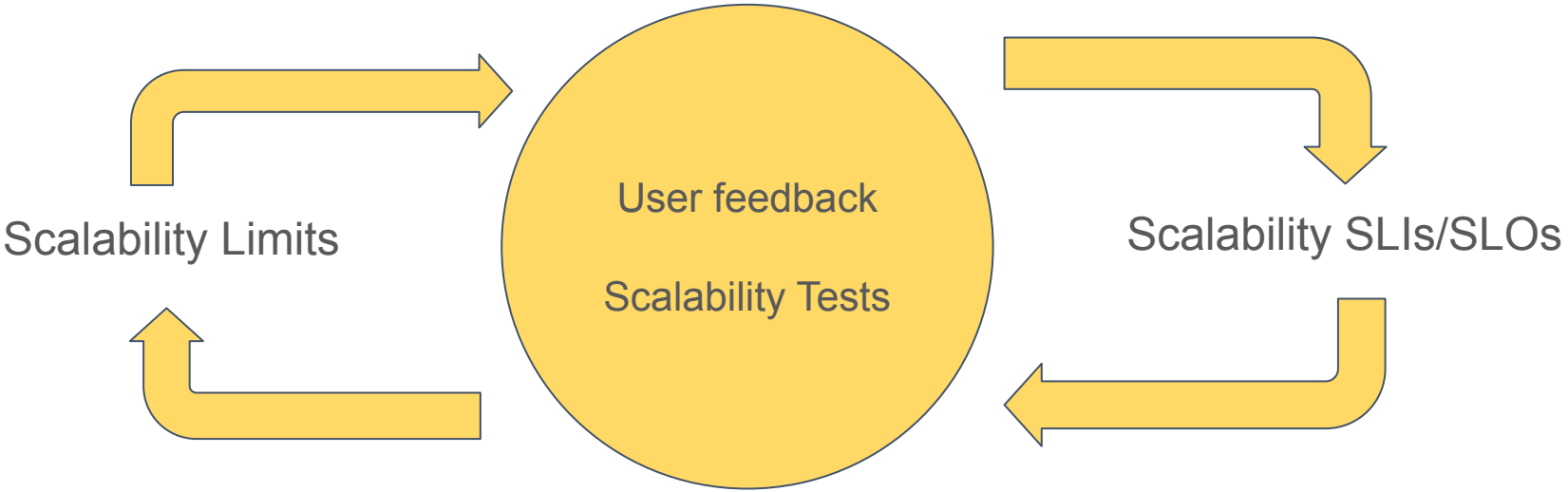
Scalability Limits

It’s hard to precisely compute the envelope boundaries but it can be decomposed into smaller envelopes, e.g.

- # Nodes <= 5K
- # Pods <= 30 * #Nodes
- # Pods per Node <= min(110, 10*#cores)
- ...

More at
github.com/kubernetes/community/tree/master/sig-scalability/configs-and-limits/thresholds.md

Feedback Loop



Keeping K8s Scalable

Perf-test Infrastructure #1

KubeCon CloudNativeCon North America 2019

Automated Scalability Testing

sig-scalability-gce | sig-scalability-node | sig-scalability-kubemark | sig-scalability-perf-tests | sig-scalability-benchmarks | sig-scalability-experiments

Summary | gce-cos-1.14-scalability-100 | gce-cos-1.15-scalability-100 | gce-cos-1.16-scalability-100 | gce-cos-1.17-scalability-100 | gce-master-scale-correctness | gce-master-scale-performance | gce-cos-master-scalability-100

gce-cos-1.14-scalability-100: FLAKY
28 of 196 tests (14.3%) and 10 of 14 runs (71.4%) failed in the past 7 days
Last update: 11-05 10:39 CET
Tests last ran: 11-05 00:00 CET
Last green run: ac756284b

gce-cos-1.15-scalability-100: FLAKY
8 of 280 tests (2.9%) and 3 of 14 runs (21.4%) failed in the past 7 days
Last update: 11-05 10:41 CET
Tests last ran: 11-05 08:00 CET
Last green run: 670c78109

gce-cos-1.16-scalability-100: FLAKY
3 of 280 tests (1.1%) and 2 of 14 runs (14.3%) failed in the past 7 days
Last update: 11-05 11:01 CET
Tests last ran: 11-05 04:01 CET
Last green run: 3d9fbfdd6

gce-cos-1.17-scalability-100: PASSING
6 of 494 tests (1.2%) and 3 of 26 runs (11.5%) failed in the past 7 days
Last update: 11-05 10:50 CET
Tests last ran: 11-05 10:00 CET
Last green run: 1aae77ada

gce-master-scale-correctness: FAILING
27 of 7035 tests (0.4%) and 5 of 7 runs (71.4%) failed in the past 7 days
Last update: 11-05 10:53 CET
Tests last ran: 11-05 04:02 CET
Last green run: 51d891ff3

- Show Alerts -

gce-master-scale-performance: PASSING
3 of 98 tests (3.1%) and 1 of 7 runs (14.3%) failed in the past 7 days
Last update: 11-05 10:35 CET
Tests last ran: 11-05 09:01 CET
Last green run: 6a19261e9

gce-cos-master-scalability-100: FLAKY
46 of 4218 tests (1.1%) and 24 of 222 runs (10.8%) failed in the past 7 days
Last update: 11-05 10:32 CET
Tests last ran: 11-05 10:09 CET
Last green run: de56c9054

Scalability Tests

Kinds of e2e scalability tests

1. **Performance** = “load” + “density” test
2. **Correctness** = regular functional tests run at scale
3. **Other** - storage, benchmarks, ...

Types of Scalability Tests

Periodic tests

1. Release blocking (K8s on GCE)
    a. Performance 100 nodes
    b. Performance 5000 nodes
    c. Correctness 5000 nodes
2. Non-release blocking
    a. Kubemark
    b. Storage
    c. Benchmarks (also microbenchmarks)
    d. ....

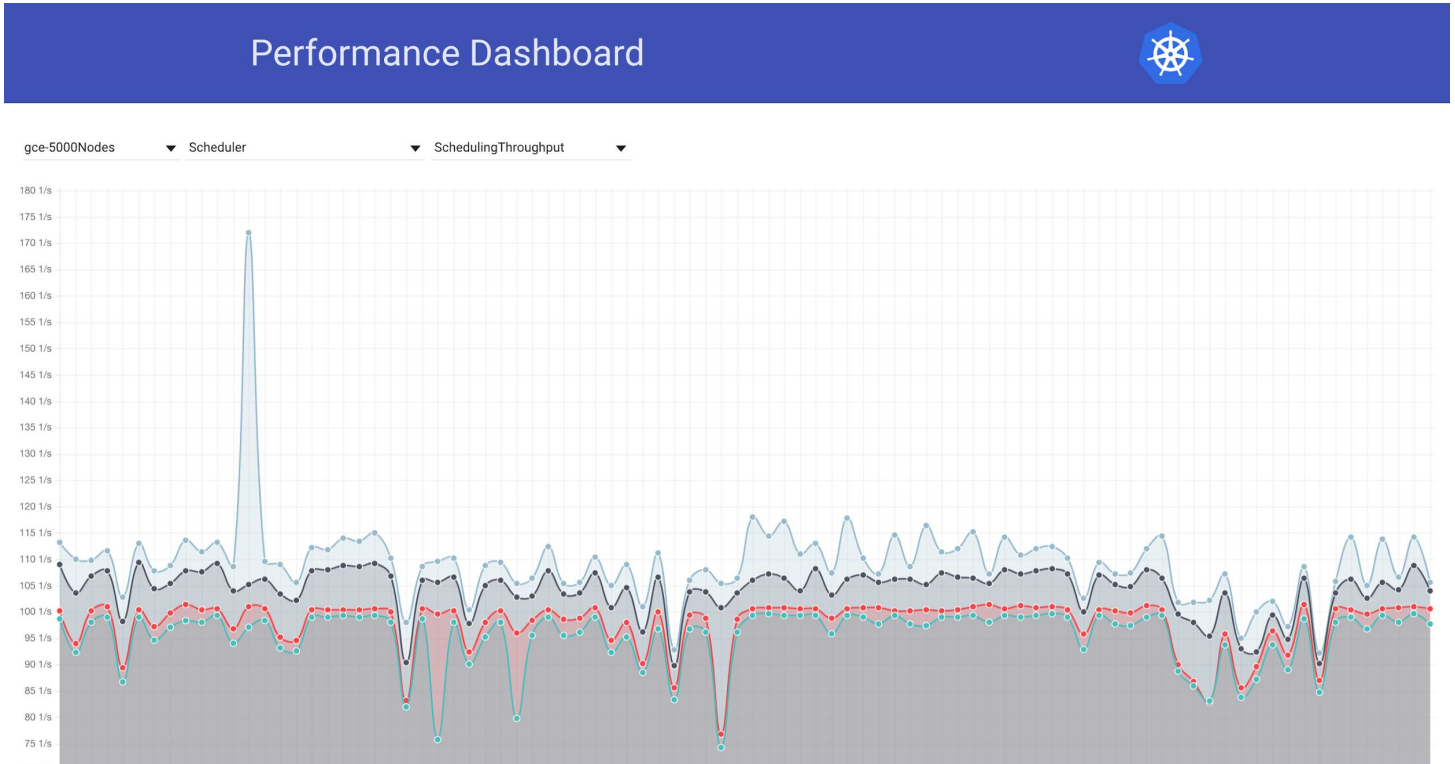
Types of Scalability Tests

Presubmits

1. k8s.io/kubernetes
    a. Performance K8s on GCE 100 nodes
    b. Performance Kubemark 500 nodes
2. k8s.io/perf-tests
    a. Performance K8s on GCE 100 nodes
    b. Performance Kubemark 500 nodes

Perf-test Infrastructure #2

Performance Dashboard - perf-dash.k8s.io



Perf-Dash

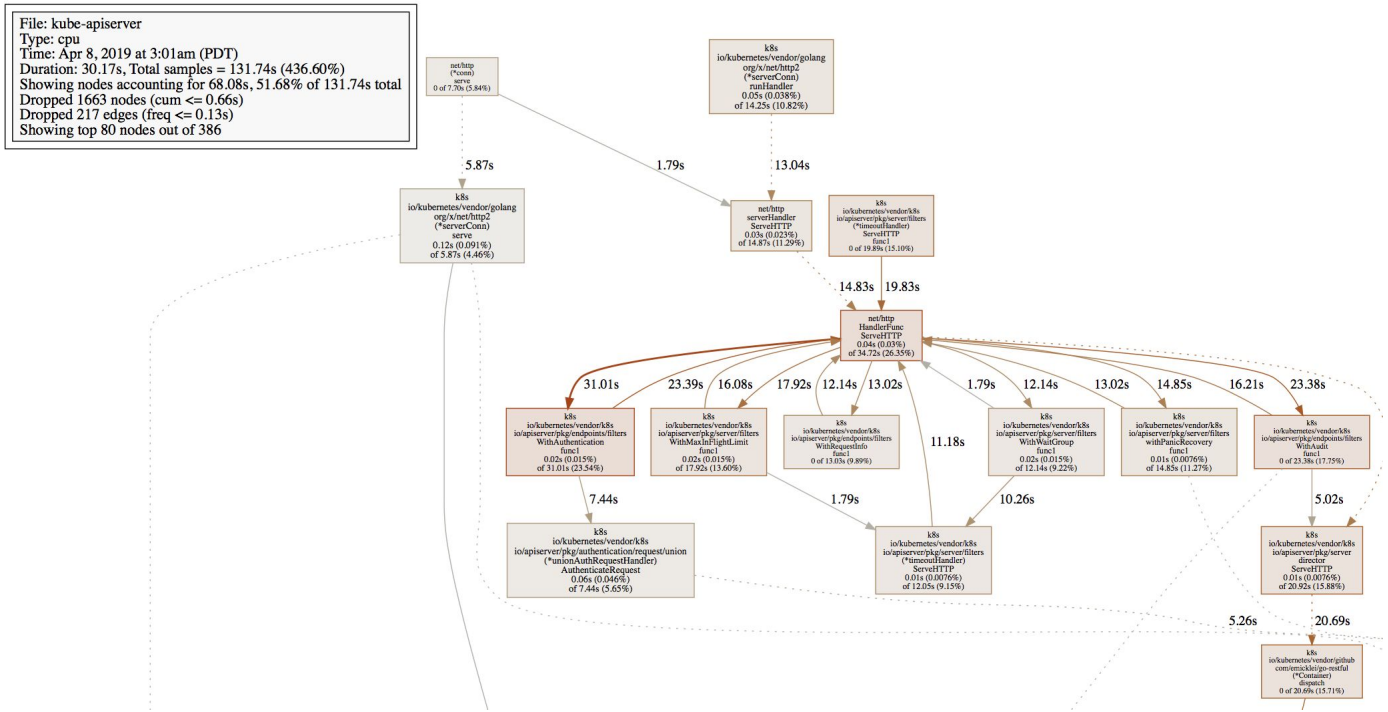
Dashboard visualizing various metrics from CI tests across different runs.

Primitive but powerful tool for debugging performance regressions and for finding various perf related k8s characteristics, e.g.

- What is current scheduler throughput in 5K node cluster?
- API Server memory usage?
- Performance regression debugging?

Perf-test Infrastructure #3

Profile Gathering



Profile Gathering

What profiles?

- CPU Profiles
- Memory Profiles
- Mutex Profiles

What components?

- kube-apiserver
- etcd
- kube-scheduler
- kube-controller-manager

Perf-test Infrastructure #4

ClusterLoader2

"Bring your own Yaml!"

```yaml
tuningSets:
- name: Uniform5qps
  qpsLoad:
    qps: 5

steps:
- name: Starting measurements
  measurements:
  - Identifier: APIResponsivenessPrometheus
    Method: APIResponsivenessPrometheus
    Params:
      action: start
  - Identifier: WaitForRunningMyDeploymentPods
    Method: WaitForControlledPodsRunning
    Params:
      action: start
      apiVersion: apps/v1
      kind: Deployment
      labelSelector: app = my-deployment
      operationTimeout: 5min

- name: Creating deployments
  phases:
  - namespaceRange:
      min: 1
      max: $namespaces
    replicasPerNamespace: 2
    tuningSet: Uniform5qps
    objectBundle:
    - basename: my-deployment
      objectTemplatePath: deployment.yaml
      templateFillMap:
        Replicas: {{$podsPerDeployment}}
        CpuRequest: 10m
        MemoryRequest: 10M

- name: Waiting for pods to be created
  measurements:
  - Identifier: WaitForRunningMyDeploymentPods
    Method: WaitForControlledPodsRunning
```

ClusterLoader2

Declarative paradigm - test defines a state in which a cluster should be and CL2 brings the cluster to that state.

A test also specifies **how** it should happen (e.g. throughput) and what should be measured during the execution (e.g. SLIs).

In addition, CL2 provides extra **observability** of the cluster during the test.

ClusterLoader2 API

A **test** is a list of **steps** (executed **serially**).

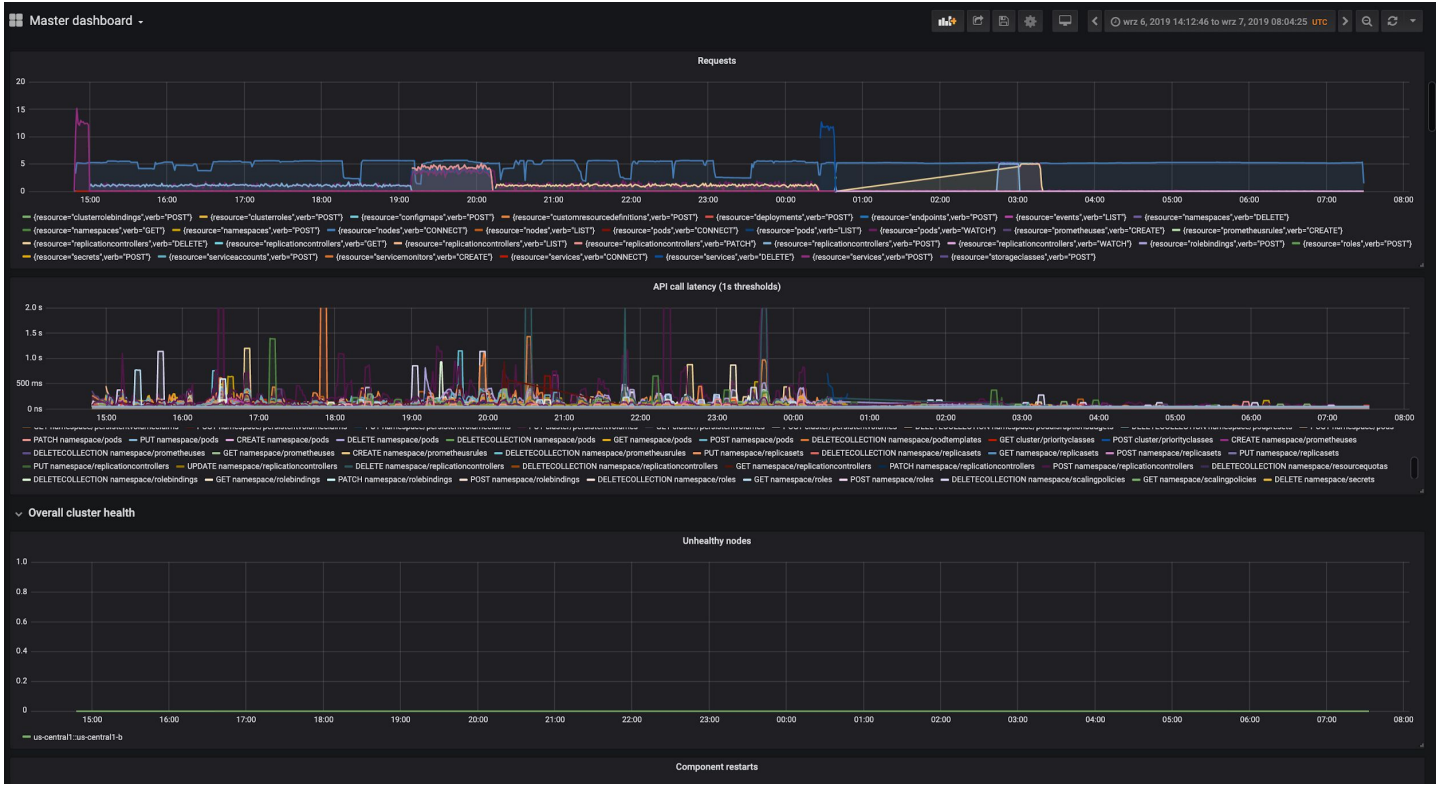
A **step** can be either a collection of **phases** or **measurements** (execute **parallely**).

A **phase** defines a state the cluster should reach.

A **measurement** allows to measure something or wait for something.

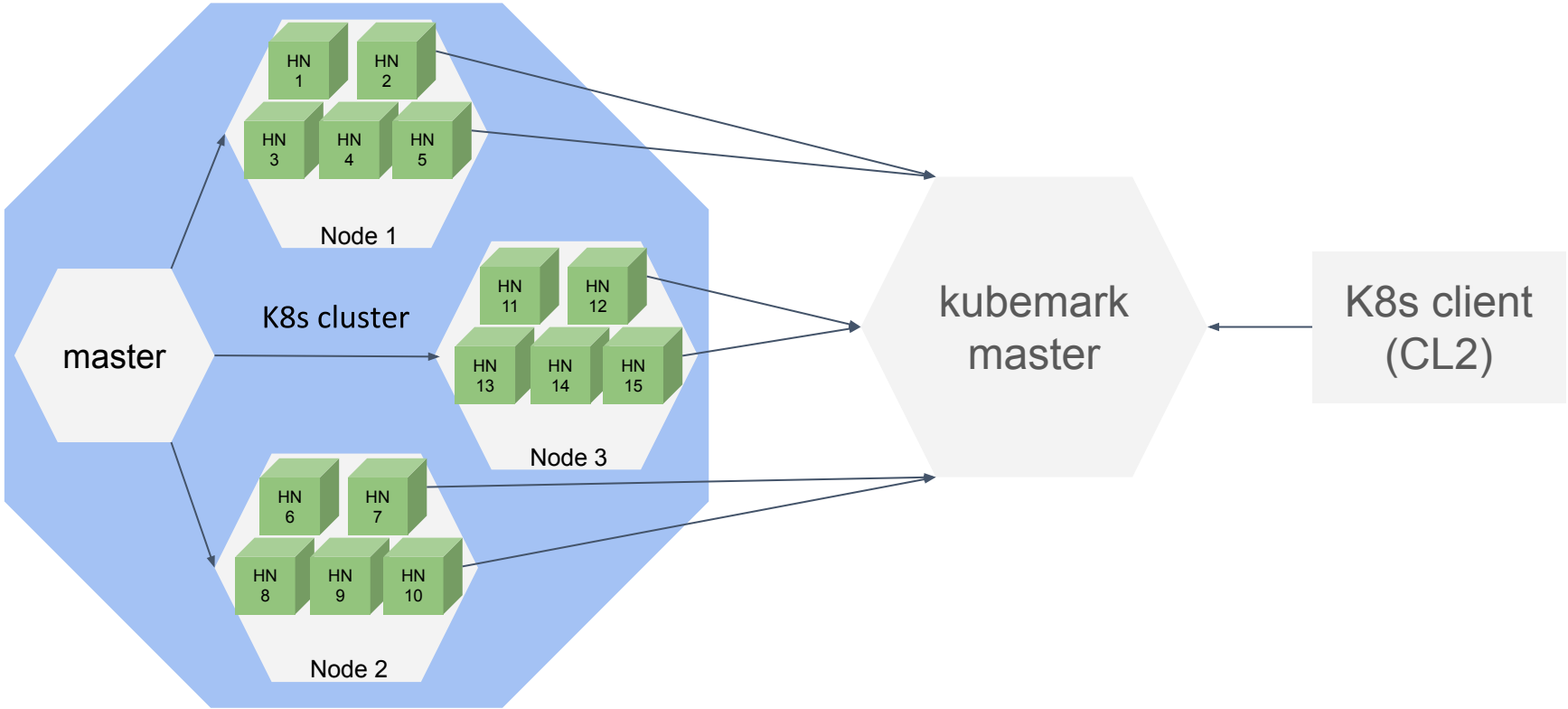
Perf-test Infrastructure #5

Prometheus & Grafana



Perf-test Infrastructure #6

Kubemark



Kubemark

Kubemark - tool for simulating large K8s clusters for scale testing purposes.

Motivation: Cheaper scale tests!

How much cheaper? ~5000 vCPUs vs ~700vCPUs to run 5K node scale tests!

Idea:

- Control plane needs to stay the same as that's what we exercise in the tests
- Look for the savings on the node side = **Hollow Node**

Protecting From Regressions

Scalability Regressions

- Scalability is *sensitive*
- We've seen regressions come from pretty much everywhere:
  - Golang
  - Operating System
  - Controllers
  - API machinery
  - Scheduler
  - Etcd
  - Kubelet
  - …
- We often debug/fix them ourselves, or triage to relevant SIGs

Scalability Regressions

Some interesting regressions

- Golang - kubernetes/kubernetes/issues/75833
- Cos - kubernetes/kubernetes/issues/83020
- CoreDNS - kubernetes/kubernetes/issues/78562
- Klog - kubernetes/kubernetes/issues/78734
- NodeLifecycleController - kubernetes/kubernetes/issues/77733
- Many, many more ...

Driving Scalability Improvements

Scalability Improvements

Some recent improvements:

- Kubelet config polling → watching
- New events API to reduce spamming
- Cheaper node heartbeats
- EndpointSlice API
- NodeLifecycleController, TaintManger, GC-Controller improvements
- Watch serialization mechanism improvements - #81914
- Watch Bookmarks
- Scheduling algorithm (rank only subset of nodes)
- IPVS as alternative for iptables
- Etcd concurrent reads

Scalability Improvements

Other improvements

- Scalability approval process
- Migrated scalability tests off public IPs
- Kubemark Improvements: HA support, system-pods, ...
- CL2 Improvements: monitoring, crashlooping pods detection, ...
- Load test extended to cover more resources: DaemonSets, Jobs, ...
- Implemented new SLIs in tests: NetworkProgrammingLatency, Dns PRogramming Latency, Network Latency, DNS Latency

Want to get involved?

How can you get involved?

kubernetes/perf-tests help-wanted

kubernetes/kubernetes help-wanted

Ping us on #sig-scalability Slack for other stuff

Where to find us?

- Home page: README
- Public Meetings: Thursdays 18.30 Warsaw time (bi-weekly)
- Slack channel: https://kubernetes.slack.com/messages/sig-scalability
- List: https://groups.google.com/forum/#!forum/kubernetes-sig-scale


KubeCon
CloudNativeCon
North America 2019
Q & A


KubeCon
CloudNativeCon
North America 2019
Thanks!