

Cluster API scalability - Introducing the In-memory provider

Lennart Jern
28.11.2023

Ensure scale of 1000 clusters – CAPI&Metal3

Insanely expensive if done on real hardware

Can we do it some other way?

Bare Metal Operator has a "test mode" (fake
Ironic)

Workload cluster API?

Iterative approach – solve the biggest issue and
move forward

Faking workload clusters

Step
1

- Run API server + etcd with certificates expected by CAPI
- Just a data store -> Add Nodes and Pods as needed

Step
2

- 1:1 mapping too heavy
- Shared etcd; shared API?

Step
3

- Distribute and run remotely (e.g. 100 API servers per VM)

Collaboration and help from maintainers



Issues:

<https://github.com/kubernetes-sigs/cluster-api/issues/8052>

<https://github.com/kubernetes-sigs/cluster-api/issues/8602>



Extremely helpful to get insights from maintainers



Hints and ideas about how to build the test environment



Bug fixes and performance improvements as soon as we discovered them

Results

Enabled scaling to 1000 clusters in reasonable time

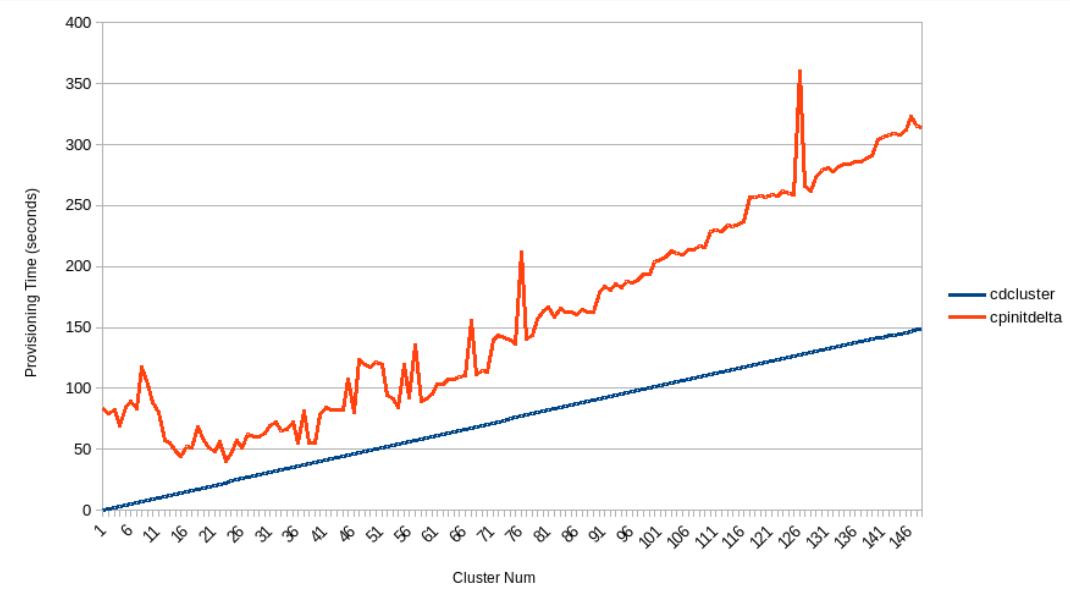
- Initially stopped at ~300 due to concurrency and test env issues
- Progressed to ~600, blocked by rate limits

KubeadmControlPlane controller performance

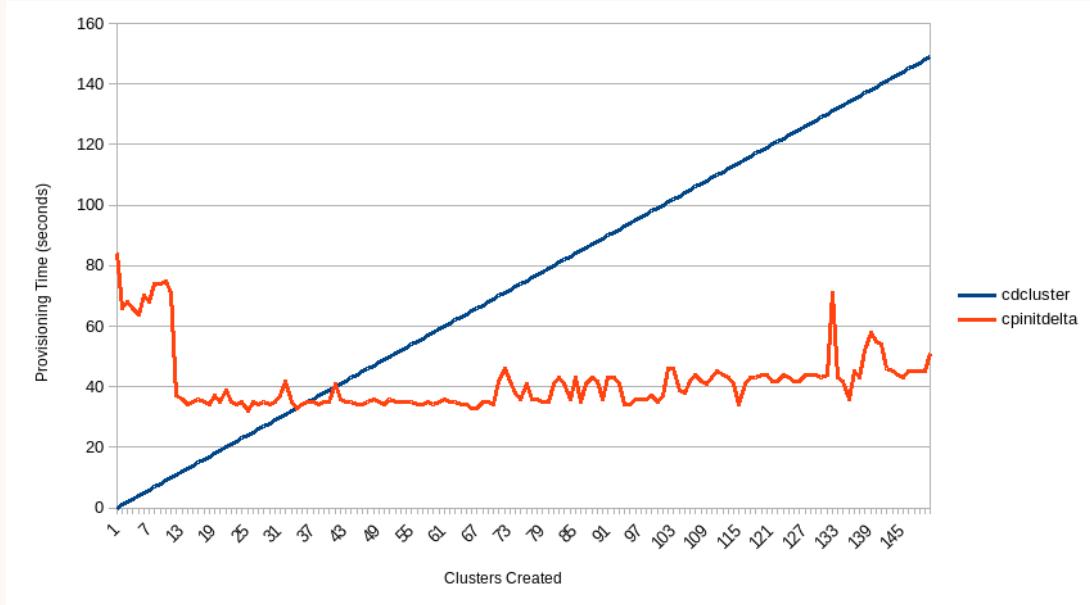
- 80 % reduction in CPU usage
- Cluster creation time reductions

Results

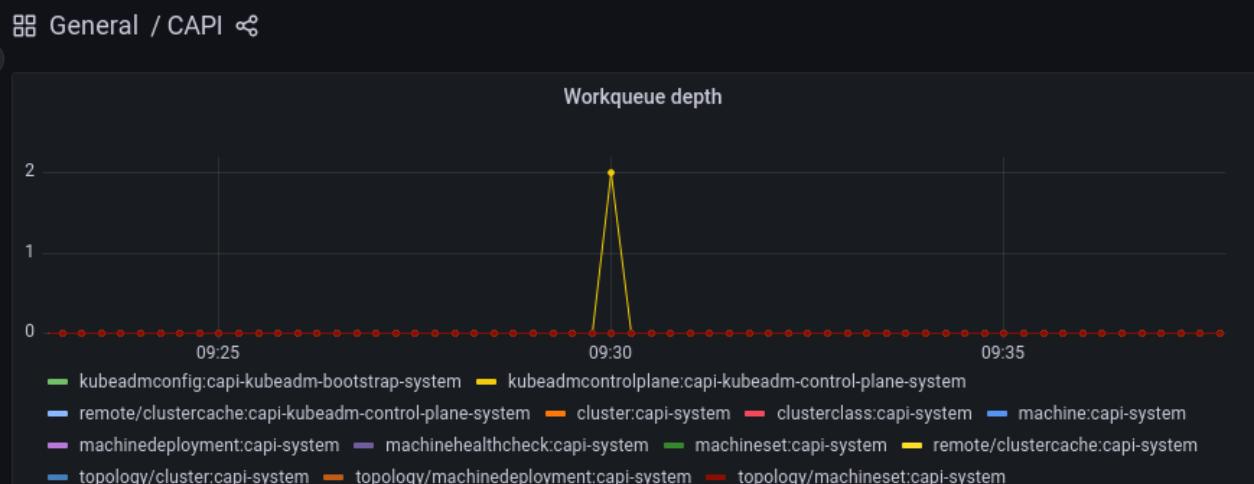
Before



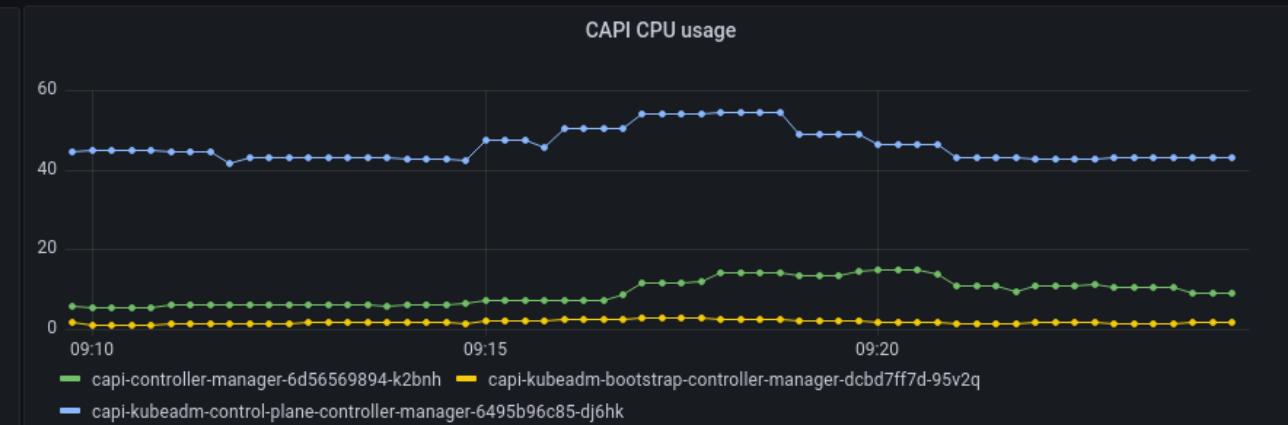
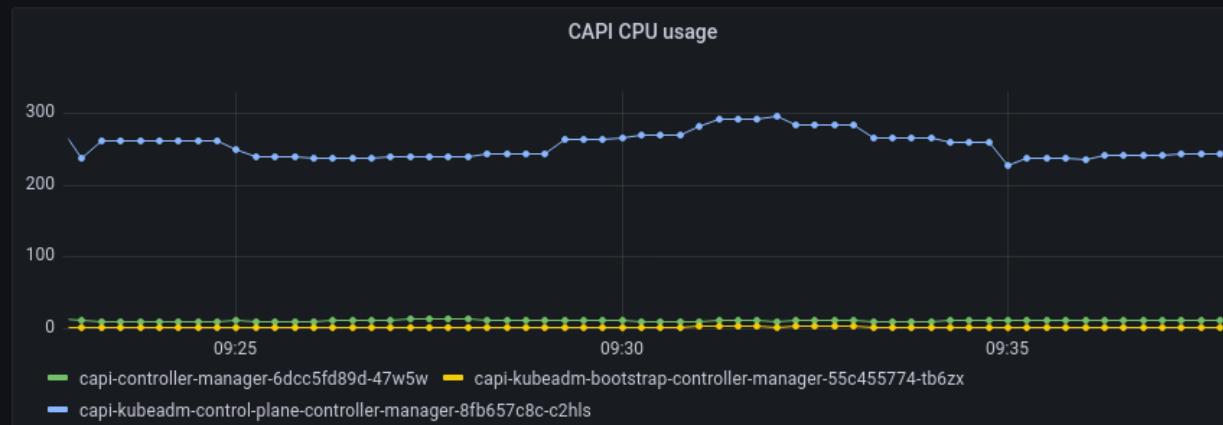
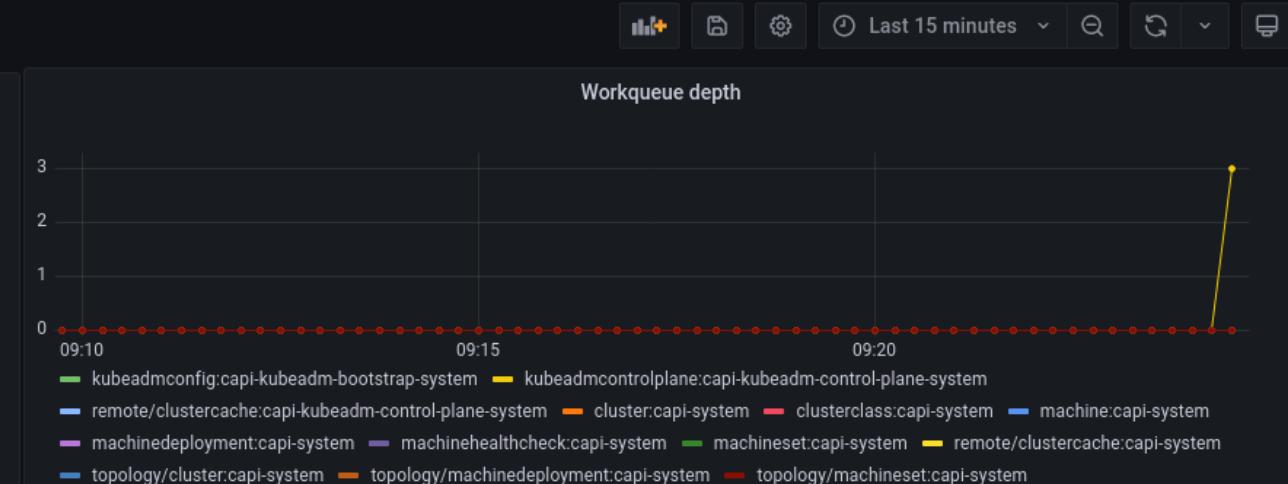
After



Before



After



Introducing the In-memory provider

Scale testing CAPI like a Pro

In-memory provider for Cluster API

- Cluster API has implemented an in-memory provider for scaling tests
- Tests CAPI (not providers like Metal3)
- Demo recording: <https://asciinema.org/a/607646>

```
s (246 kB/s)
unselected package asciinema.
  373385 files and directories currently installed.

.../ascinema_2.1.0-1_all.deb ...
(2.1.0-1) ...
(2.1.0-1) ...
for man-db (2.10.2-1) ...

pace/playground/in-memory took 5s
usters.sh

pace/playground/in-memory
sh 20
s.io "test-1" deleted
s.io "test-2" deleted
s.io "test-3" deleted
s.io "test-4" deleted
s.io "test-5" deleted
s.io "test-6" deleted
s.io "test-7" deleted
s.io "test-8" deleted
s.io "test-9" deleted
s.io "test-10" deleted
s.io "test-11" deleted
s.io "test-12" deleted
s.io "test-13" deleted
s.io "test-14" deleted
s.io "test-15" deleted
s.io "test-16" deleted
s.io "test-17" deleted
s.io "test-18" deleted
s.io "test-19" deleted
s.io "test-20" deleted

pace/playground/in-memory took 1m48s
sh 20
s.io/test-1 created
s.io/test-2 created
s.io/test-3 created
s.io/test-4 created
s.io/test-5 created
s.io/test-6 created
s.io/test-7 created
s.io/test-8 created
s.io/test-9 created
s.io/test-10 created
s.io/test-11 created
s.io/test-12 created
s.io/test-13 created
s.io/test-14 created
s.io/test-15 created
s.io/test-16 created
s.io/test-17 created
s.io/test-18 created
s.io/test-19 created
s.io/test-20 created

pace/playground/in-memory took 3s
```

2:~*

```
Every 2,0s: kubectl get machine
NAME          CLUSTER   NODENAME   PROVIDERID
test-1-tzzqc-fd2rt test-1      in-memory://test-1
test-10-rlsg-czzr2 test-10     in-memory://test-1
test-11-rb7zn-zhmnx test-11     in-memory://test-1
test-12-g19bg-s5vbm test-12     in-memory://test-1
test-13-t8g6l-rb7zq test-13     in-memory://test-1
test-14-q9cww-stf2h test-14     in-memory://test-1
test-15-mclcv-52w4m test-15     in-memory://test-1
test-16-t96bg-s84hg test-16     in-memory://test-1
test-17-61588-gmw4q test-17     in-memory://test-1
test-18-j6bnd-f8spd test-18     in-memory://test-1
test-19-q8m7v-xgmcg test-19     in-memory://test-1
test-2-m5966-fmf8q test-2      in-memory://test-2
test-20-sxgdg-2qjv6 test-20     in-memory://test-2
test-3-vrtwq-8642w test-3      in-memory://test-3
test-4-gb8p9-fw5cj test-4      in-memory://test-4
test-5-jcshc-492xg test-5      in-memory://test-5
test-6-s5brh-lqxzc test-6      in-memory://test-6
test-7-k67ts-v7pdh test-7      in-memory://test-7
test-8-p6pmp-bqr2f test-8      in-memory://test-8
test-9-ttvb2-4lxz7 test-9      in-memory://test-9
```

```
Every 2,0s: kubectl get kcp
NAME          CLUSTER   INITIALIZED   API SERVER AVAILABLE
test-1-tzzqc test-1        test-1
test-10-rlsg test-10       test-10
test-11-rb7zn test-11      test-11
test-12-g19bg test-12      test-12
test-13-t8g6l test-13      test-13
test-14-q9cww test-14      test-14
test-15-mclcv test-15      test-15
test-16-t96bg test-16      test-16
test-17-61588 test-17      test-17
test-18-j6bnd test-18      test-18
test-19-q8m7v test-19      test-19
test-2-m5966 test-2        test-2
test-20-sxgdg test-20      test-20
test-3-vrtwq test-3        test-3
test-4-gb8p9 test-4        test-4
test-5-jcshc test-5        test-5
test-6-s5brh test-6        test-6
test-7-k67ts test-7        test-7
test-8-p6pmp test-8        test-8
test-9-ttvb2 test-9        test-9
```

Scale testing CAPI like a Pro

- An "infrastructure" provider
- Controller just fakes the minimum necessary
- Each control plane gets an endpoint with unique port
- Each Machine results in a Node object in the API
- A fully operational workload cluster from CAPI perspective

Getting started with the in-memory provider

- Recording: <https://asciinema.org/a/618386>
- Initialize with clusterctl like any other provider
- Check cluster template and cluster class in CAPI releases
- Note that timings can be configured in the InMemoryMachineTemplate

```
1 ---  
2 apiVersion: infrastructure.cluster.x-k8s.io/v1alpha1  
3 kind: InMemoryMachineTemplate  
4 metadata:  
5   name: in-memory-quick-start-default-worker-machinetemplate  
6 spec:  
7   template:  
8     spec:  
9       behaviour:  
10      vm:  
11        provisioning:  
12          startupDuration: "30s"  
13          startupJitter: "0.2"  
14      node:  
15        provisioning:  
16          startupDuration: "10s"  
17          startupJitter: "0.2"  
18    apiServer:  
19      provisioning:  
20          startupDuration: "10s"  
21          startupJitter: "0.2"
```

The work goes on



- CAPI efforts to automate scale tests in CI + metrics and observability
- Ensure scalability of full Metal3 stack (Ironic and Bare Metal Operator)
- More details on CAPI work in the KubeCon deep dive 🎊

