Advancing CPU management in Kubernetes

ONS 2019 Europe
Levente Kálé
23-09-2019
Whoami

Product architect working on Kubernetes based edge cloud solutions within, and without Nokia (Akraino Radio Edge Cloud)

Specializes in satisfying 5G RAN requirements (both control plane and data plane) ... most of times related to compute resource management

LinkedIn: https://www.linkedin.com/in/levovar-045/

GitHub: https://github.com/Levovar

Not a Twitter guy 😊
Wait, we need to advance CPU management in Kubernetes?
Wait, we need to advance CPU management in Kubernetes?

Yes we do!*
Wait, we need to advance CPU management in Kubernetes?

Yes we do!*

Kubelet actually has a CPU manager, capable of provisioning exclusive cores

But special industries and workloads have special needs:
Wait, we need to advance CPU management in Kubernetes?

Yes we do!*

Kubelet actually has a CPU manager, capable of provisioning exclusive cores

But special industries and workloads have special needs:
  • Hard isolation for increased high-availability
Wait, we need to advance CPU management in Kubernetes?

**Yes we do!**

Kubelet actually has a CPU manager, capable of provisioning exclusive cores.

But special industries and workloads have special needs:

- Hard isolation for increased high-availability
- Sub-node partitioning and configuration of CPU cores
Wait, we need to advance CPU management in Kubernetes?

Yes we do!*

Kubelet actually has a CPU manager, capable of provisioning exclusive cores

But special industries and workloads have special needs:
• Hard isolation for increased high-availability
• Sub-node partitioning and configuration of CPU cores
• CPU pinning
  • I know, I know in an ideal world... but sometimes this is still needed
What did the community have to say about this?
What did the community have to say about this?

Paraphrasing:
„If you want pools, separate your workloads to different Nodes with nodeSelectors, and/or with taints and tolerations”
What did the community have to say about this?

Paraphrasing:
„If you want pools, separate your workloads to different Nodes with nodeSelectors, and/or with taints and tolerations”
„Soft isolation of workloads via limits and requests is enough for 80% of the workloads”
What did the community have to say about this?

Paraphrasing:

„If you want pools, separate your workloads to different Nodes with nodeSelectors, and/or with taints and tolerations”

„Soft isolation of workloads via limits and requests is enough for 80% of the workloads”

„Maybe Kubernetes is not the right choice for you.”
What did the community have to say about this?

Paraphrasing:

„If you want pools, separate your workloads to different Nodes with nodeSelectors, and/or with taints and tolerations”

„Soft isolation of workloads via limits and requests is enough for 80% of the workloads”

„Maybe Kubernetes is not the right choice for you.”

But what if you think Kubernetes **is** the right choice for you?
What did the community have to say about this?

Paraphrasing:

„If you want pools, separate your workloads to different Nodes with nodeSelectors, and/or with taints and tolerations”

„Soft isolation of workloads via limits and requests is enough for 80% of the workloads”

„Maybe Kubernetes is not the right choice for you.”

But what if you think Kubernetes is the right choice for you?

If you are also the 20%, CPU-Pooler is here to save your day!
Reservoir pools

x86 server

CPU
Reservoir pools

x86 server

• Mr. Green, The Platfrom
Reservoir pools

x86 server

- Mr. Green, The Platform
- Mr. Red, The Virtual Machine

Infra

VMs

CPU
Reservoir pools

- Mr. Green, The Platform
- Mr. Red, The Virtual Machine
- Mr. Purple, The Exclusive
Reservoir pools

- **Mr. Green, The Platform**
- **Mr. Red, The Virtual Machine**
- **Mr. Purple, The Exclusive**
- **Mr. Orange, The Shared**
Reservoir pools

- Mr. Green, The Platform
- Mr. Red, The Virtual Machine
- Mr. Purple, The Exclusive
- Mr. Orange, The Shared
- Mr. Grey, The Default
Pooling done right
Pooling done right

Instrumenting Kubernetes is easy, but making it feel like a native feature is a different story!
Pooling done right

Instrumenting Kubernetes is easy, but making an enhancement feel like a native feature is a different story!

CPU-Pooler is designed to work together with Kubernetes, not against it.
Pooling done right

Instrumenting Kubernetes is easy, but making an enhancement feel like a native feature is a different story!

CPU-Pooler is designed to work together with Kubernetes, not against it.

CPU-Pooler is designed to manage CPUs the same way as Kubelet does.
Pooling done right

Instrumenting Kubernetes is easy, but making an enhancement feel like a native feature is a different story!

CPU-Pooler is designed to work together with Kubernetes, not against it.

CPU-Pooler is designed to manage CPUs the same way as Kubelet does.

CPU-Pooler is designed to have the same user facing APIs as Kubernetes does.
Pooling done right

Instrumenting Kubernetes is easy, but making an enhancement feel like a native feature is a different story!

CPU-Pooler is designed to work together with Kubernetes, not against it.

CPU-Pooler is designed to manage CPUs the same way as Kubelet does.

CPU-Pooler is designed to have the same user facing APIs as Kubernetes does.

The end result?

A solutions which feels like Kubernetes, and works like Kubernetes – only with moar pools!
CPUs are technically devices, am I right?
CPU-Pooler in action
CPU-Pooler in action

cloudadmin@controller-1:~

poolconfig-compute-1.yaml:

nodeSelector:
  nodename: caas_worker1
pools:
  default:
    cpus: 1,13-14,29,41-42
  exclusive_caas:
    cpus: 9-12,24-27,37-40,52-55
  shared_caas:
    cpus: 2-8,15-23,30-36,43-51

poolconfig-controller-1.yaml:

nodeSelector:
  nodename: caas_master1
pools:
  default:
    cpus: 2,7,16,21,30,35,44,49
  exclusive_caas:
    cpus: 6,20,34,48
  shared_caas:
    cpus: 3-5,17-19,31-33,45-47
CPU-Pooler in action
CPU-Pooler in action

spec:
containers:
- name: exclusive-test
  image: registry.kube-system.svc.nokia.net:5555/caas/busybox:latest
  command: ["/bin/sh", "-c", "--"]
  args: ["while true; do sleep 1; done;"]
  resources:
    requests:
      nokia.k8s.io/exclusive_caas: 1
  limits:
    nokia.k8s.io/exclusive_caas: 1

- name: shared-test
  image: registry.kube-system.svc.nokia.net:5555/caas/busybox:latest
  command: ["/bin/sh", "-c", "--"]
  args: ["while true; do echo "\"Test\"; sleep 1; done;"]
  resources:
    requests:
      nokia.k8s.io/shared_caas: 200
  limits:
    nokia.k8s.io/shared_caas: 200

- name: default-test
  image: registry.kube-system.svc.nokia.net:5555/caas/busybox:latest
  command: ["/bin/sh", "-c", "--"]
  args: ["while true; do echo "\"Test\"; sleep 1; done;"]
  resources:
    requests:
      cpu: 1000m
    limits:
      cpu: 1000m
## CPU-Pooler in action

### Allocated resources:

(Total limits may be over 100 percent, i.e., overcommitted.)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Requests</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu</td>
<td>100m (1%)</td>
<td>1 (16%)</td>
</tr>
<tr>
<td>memory</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>ephemeral-storage</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>nokia.k8s.io/exclusive caas</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nokia.k8s.io/shared caas</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Allocated resources:

(Total limits may be over 100 percent, i.e., overcommitted.)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Requests</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu</td>
<td>1100m (18%)</td>
<td>2200m (36%)</td>
</tr>
<tr>
<td>memory</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>ephemeral-storage</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>nokia.k8s.io/exclusive caas</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>nokia.k8s.io/shared caas</td>
<td>200</td>
<td>200</td>
</tr>
</tbody>
</table>
CPU-Pooler in action

```yaml
poolconfig-compute-1.yaml:
  ---
  nodeSelector:
    nodename: caas_worker1
  pools:
    default:
      cpus: 1,13-14,29,41-42
    exclusive_caas:
      cpus: 9-12,24-27,37-40,52-55
    shared_caas:
      cpus: 2-8,15-23,30-36,43-51
```

```bash
$ kubectl exec cpu-pooling-demo-769fb5fb44-vftqx -c default-test cat /proc/1/status | grep Cpus_allowed_list
Cpus_allowed_list: 1,13-14,29,41-42

$ kubectl exec cpu-pooling-demo-769fb5fb44-vftqx -c exclusive-test cat /proc/1/status | grep Cpus_allowed_list
Cpus_allowed_list: 10

$ kubectl exec cpu-pooling-demo-769fb5fb44-vftqx -c shared-test cat /proc/1/status | grep Cpus_allowed_list
Cpus_allowed_list: 2-8,15-23,30-36,43-51
```
Closing remarks 1 - What about CPU socket alignment?
You should really attend „Topology Awareness in Kubernetes - The How and The Why” brought to you by Louise Daly & Feng Pan
(Wednesday, September 25 • 12:00 - 12:30, Marble Hall)

***Spoiler Alert ***

Guess which Kubernetes managed resources besides CPUs are automatically socket aligned by the Topology Manager? 😊

https://github.com/nokia/CPU-Pooler/issues/24
Closing remarks 2  - Shameless plugin

We are putting our money where our mouth is:

1. CPU-Pooler is used in production in Nokia edge cloud
2. CPU-Pooler is integrated into open source Akraino Radio Edge Cloud
   https://gerrit.akraino.org/r/gitweb?p=ta%2Fcaas-cpupooler.git;a=summary

If you liked what we did with CPU management in Kubernetes, wait until you have seen what we did to networks:
https://github.com/nokia/danm