



KubeCon



CloudNativeCon

North America 2019

Is There a Place for Performance Sensitive Workloads in Kubernetes?

Levente Kálé, Gergely Csatári





Levente Kálé

Nokia
TelCo edge cloud architect

 : [levovar-045/](https://www.linkedin.com/in/levovar-045/)

 : [@Levovar](https://github.com/Levovar)



Gergely Csatári

Nokia
Open Source Specialist

 : [gergely-csatári-76591020/](https://www.linkedin.com/in/gergely-csatari-76591020/)

 : [@CsatariGergely](https://github.com/CsatariGergely)

What do we do?



KubeCon



CloudNativeCon

North America 2019

Surprise: Kubernetes-based TelCo grade clouds and containerized TelCo applications 😊

Clouds

Private edge, central, and public clouds
Kubernetes-only or multi-orchestrator



Apps

5G Radio Access Network (DU, CU, RIC)

Mobile core

Fixed network

IT/Enterprise



Open source entanglements



KubeCon



CloudNativeCon

North America 2019



CNCFTUG



CNF Testbed



CNTT



SIG-Node / WG-Resource Management



DANM

CPU-Pooler



Akraino



Is There a Place for **Performance Sensitive Workloads** in Kubernetes?



KubeCon



CloudNativeCon

North America 2019

How does K8s perform when it is used to run the **worst possible** workloads?

Our worst possible is running compute heavy workloads:

Using all the available CPU of a Node without much idling

With high network throughput needs: 25Gbps+

With real-time latency requirements: <1 μ s RTT

Facing challenging high-availability ("five-nines") and legal (government laws) requirements



Is There a Place for Performance Sensitive Workloads in Kubernetes?



KubeCon



CloudNativeCon

North America 2019

YES!!!

BUT...



"Solutions" out-of-scope and question



KubeCon



CloudNativeCon

North America 2019

Before deep-dive, some things **You** should also say no to:

- Proprietary hardware
- proprietary operating systems
- special kernel patches (including preemptive RT kernel)
- proprietary kernel drivers
- privileged applications

With a few native enhancements even the worst applications can be kept cloud-native*!

(*ish) 😊

```
00008E10: 85 C6 00 00 00 66 83 7C|6E FE 30 75 01 45 45 0F | ...Ć...F.|nt0u.EE.
00008E20: B7 4C 6E FE 8B C1 83 C0|D0 66 83 E8 0A 72 14 83 | -Lnt<Á.R0F.č.r..
00008E30: C0 F9 66 83 E8 06 72 13|83 C0 E6 66 83 E8 06 72 | RûF.č.r..RčF.č.r.
00008E40: 15 EB 71 0F B7 F9 83 EF|30 EB 14 0F B7 F9 83 EF | .eq.·û.d0è...û.d
00008E50: 41 83 C7 0A EB 09 0F B7|F9 83 EF 61 83 C7 0A 83 | A.Ć.ë...û.da.Ć..
00008E60: 7C 24 0C 00 75 09 83 7C|24 08 00 72 47 EB 02 7C | |$.u..|$.rGè.|
00008E70: 43 81 7C 24 0C FF FF FF|0F 75 09 83 7C 24 08 FF | C.|$.·.·.u..|$.·
00008E80: 76 04 EB 30 7F 2E 8B C7|99 52 50 8B 44 24 10 8B | u.è0M.<Ć™RP<D$.<
00008E90: 54 24 14 0F A4 C2 04 C1|E0 04 03 04 24 13 54 24 | T$.·*Â.ÂF...$.T$
00008EA0: 04 83 C4 08 89 44 24 08|89 54 24 0C 45 33 DB E9 | ..Ă.%.D$.%.T$.E3Ué
00008EB0: 6B FF FF FF 80 7C 24 10|00 0F 84 D0 00 00 00 8B | k'·'·|$....0...<
00008EC0: 44 24 08 8B 54 24 0C F7|D8 83 D2 00 F7 DA 89 44 | D$.<T$.÷R.Ñ.÷Ú%Ð
00008ED0: 24 08 89 54 24 0C E9 B4|00 00 00 0F B7 44 6E FE | $.%T$.é'....·Dnt
00008EE0: 83 C0 D0 66 83 E8 0A 73|5F 0F B7 7C 6E FE 83 EF | .R0F.č.s_..|nt.d
00008EF0: 30 83 7C 24 0C 00 75 09|83 7C 24 08 00 72 49 EB | 0.|$.u..|$.rIè
00008F00: 02 7C 45 81 7C 24 0C CC|CC CC 0C 75 0C 81 7C 24 | .|E.|$.ĚĚĚ.u..|$.
00008F10: 08 CC CC CC CC 76 04 EB|2F 7F 2D 6A 00 6A 0A 8B | .ĚĚĚĚv.ë/■-j.j.<
00008F20: 44 24 10 8B 54 24 14 E8|D0 FD FF FF 52 50 8B C7 | D$.<T$.čDý'·RP<Ć
00008F30: 99 03 04 24 13 54 24 04|83 C4 08 89 44 24 08 89 | "·.$.T$.·.Ă.%.D$.%.
00008F40: 54 24 0C 45 33 DB EB 93|80 7C 24 10 00 74 17 8B | T$.E3Ué"■|$...t.<
00008F50: 44 24 08 8B 54 24 0C F7|D8 83 D2 00 F7 DA 89 44 | D$.<T$.÷R.Ñ.÷Ú%Ð
00008F60: 24 08 89 54 24 0C 83 7C|24 0C 00 75 05 83 7C 24 | $.%T$...|$.u..|$.
00008F70: 08 00 74 1B 83 7C 24 0C|00 75 0A 83 7C 24 08 00 | ..t..|$.u..|$...
00008F80: 0F 92 C0 EB 03 0F 9C C0|3A 44 24 10 74 01 4D 66 | .'Rè..sR:D$.t.MF
00008F90: 83 7C 6E FE 00 0F 95 C0|0A D8 74 09 45 4D 8B 04 | .|nt...R.Rt.EM<.
00008FA0: 24 89 28 EB 07 8B 04 24|33 D2 89 10 8B 44 24 08 | $%(è.<.$3N%.<D$.
00008FB0: 8B 54 24 0C 83 C4 14 5D|5F 5E 5B C3 88 14 CD FC | <T$.·.Ă.]_^[Ă.İü
00008FC0: FB 4A 00 89 04 CD F8 FB|4A 00 C3 90 53 56 57 55 | ůJ.%.ÍřŮJ.Ă.SUVU
00008FD0: 51 8B D8 C6 04 24 01 85|DB 74 7F 3B 1D F8 FB 4A | Q<RĆ.$...Ůt■;řŮJ
00008FE0: 00 75 0C 0F B6 05 FC FB|4A 00 88 04 24 EB 6F 0F | .u..q.üŮJ...$ëo.
00008FF0: B6 03 2C 0D 74 06 FE C8|74 0F EB 5E 0F B6 43 01 | q.,.t.tĈt.è^·qC.
00009000: 03 C3 8B 40 0A 8B 18 EB|D2 0F B6 73 01 03 F3 83 | .Ă@.<.ëÑ.qs..ó.
00009010: 7E 06 00 76 45 8B 6E 06|4D 85 ED 7C 3D 45 33 FF | ~..vE<n.M.Í|=E3'
00009020: 83 7C FE 0A 00 74 37 8B|44 FE 0A 8B 18 80 3B 0D | .|t..t7<Dt.<..■;.
00009030: 75 14 0F B6 43 01 03 C3|8B 40 0A 8B 00 E8 8A FF | u..qC..Ă@.<..čŠ'
00009040: FF FF 84 C0 75 18 80 3B|0E 75 0B 8B C3 E8 7A FF | '·.Ru.■;.u.<Ăčz'
00009050: FF FF 84 C0 75 08 47 4D|75 C6 C6 04 24 00 0F B6 | '·.Ru.GMuĆĆ.$..q
00009060: 04 24 5A 5D 5F 5E 5B C3|53 56 8B F0 E8 2B 33 00 | .Z]_^[ĂSV<dč+3.
00009070: 00 3B B0 08 00 00 00 75|0E E8 1E 33 00 00 0F B6 | .:°.....u.č.3...q
```

Realtime Scorecard for K8s 1.16



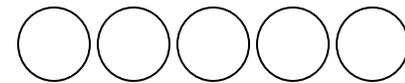
KubeCon



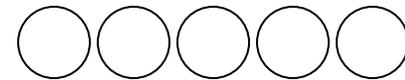
CloudNativeCon

North America 2019

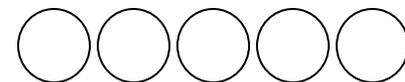
CPU management



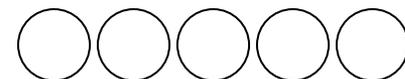
HW acceleration



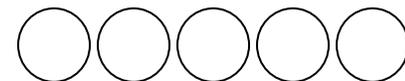
Memory management



Network management



Topology awareness



CPU management baseline

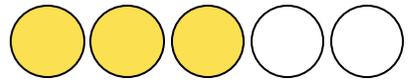


KubeCon



CloudNativeCon

North America 2019



Kubernetes has a CPU manager!

The Good:

- Uses cgroups

- Supports multiple policies: none, static

- Possible to allocate exclusive CPU cores

The Bad:

- No interworking with non-K8s managed processes

- Only supports node-level separation



CPU management – Need moar pools!



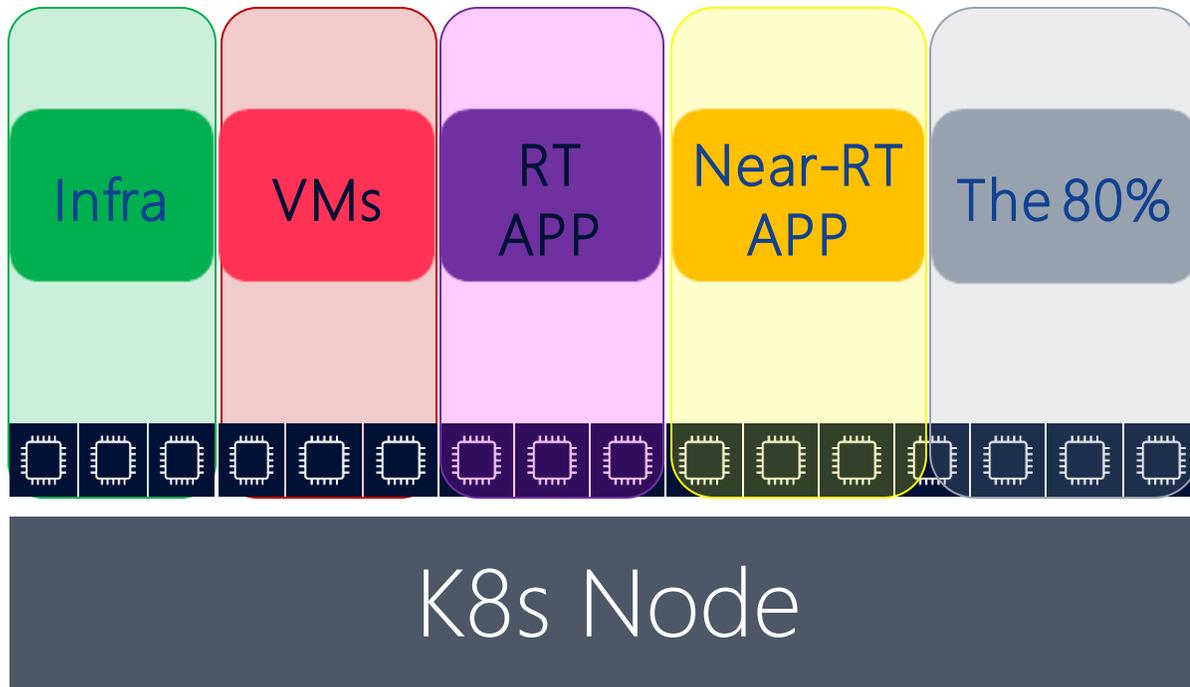
KubeCon



CloudNativeCon

North America 2019

Reservoir pools



[Mr. Green, The Platform](#)

[Mr. Red, The Virtual Machine](#)

[Mr. Purple, The Exclusive](#)

[Mr. Orange, The Shared](#)

[Mr. Grey, The Default](#)

CPU management – CPU-Pooler



KubeCon



CloudNativeCon

North America 2019

NO LOGO YET

<https://github.com/nokia/CPU-Pooler>

Native extension to CPU Manager
Uses core resources API

Uses same kernel features

Just supports more pools – and
makes it possible to tune them
differently

CPU management – Pooler in action



KubeCon



CloudNativeCon

North America 2019

```
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
```

```
[cloudadmin@controller-1 ~]$ 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
```

```
[cloudadmin@controller-1 ~]$ kubectl exec cpu-pooling-demo-769fb5fb44-vftqx -c exclusive-test cat /proc/1/status | grep Cpus_allowed_list  
Cpus_allowed_list:      10
```

```
[cloudadmin@controller-1 ~]$ 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
```

CPU management enhanced



KubeCon



CloudNativeCon

North America 2019

Support non-K8s managed pools by excluding cores from its own

CRI resource manager

Why not take the best of both worlds -> ***dynamic pooling!***

Natively support sub-node pools like CPU-Pooler

But allocate cores to a pool dynamically, on-demand like CPU Manager

A „CPU Pool” would become an abstraction, describing a **set of characteristics** or **configurations** CPU Manager would apply ***when a workload is instantiated!***

HW acceleration baseline



KubeCon



CloudNativeCon

North America 2019



HW acceleration: delegate recurring and costly computing operations to specialized hardware
Examples: (SR-IOV,) FPGA, GPU

The Good:

- Flexible and extensible (gRPC)
- Device Plugin API to plugin HW device managers

The Bad:

- Absence of fine-grained control



HW acceleration devices



KubeCon



CloudNativeCon

North America 2019

F(ield)P(rogrammable)G(ate)A(rray): better for single-thread/serialized, high-volume, low-complexity computations

FEC computation in L1 of RAN

(https://en.wikipedia.org/wiki/Forward_error_correction)

Offloading network, crypto(encryption/decryption), storage management etc.

GPU: better for multi-threaded computations

(Predictive) rendering: cloud gaming, AR/VR

High performance general purpose computing ([CUDA/GPGPU](#)): AI/ML, image recognition, neural networks, physical simulations, cryptocurrency

HW acceleration enhanced



KubeCon



CloudNativeCon

North America 2019

„Release” DPAPI call

Passing parameters to Allocate() DPAPI call

Sharing the same physical device (e.g. queue of the same FPGA, GPU cores/lanes of the same card etc.)

Memory management baseline



KubeCon



CloudNativeCon

North America 2019



The Good:

- Native support for allocating normal memory
- Supports huge memory pages
- Supports different sizes (2M, 1Gi)

The Bad:

- Need better isolation
- Lack of topology awareness hurts performance



Memory management enhanced



KubeCon



CloudNativeCon

North America 2019

Topology aware accounting of hugepages

Topology aware resource management of hugepages

Manage hugepages on the container level

Topology aware allocation of RAM

Isolating memory and hugepages from non-K8s managed workloads?

Network management baseline

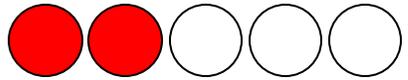


KubeCon



CloudNativeCon

North America 2019



"Network management" in vanilla
Kubernetes == CNI

The Good:

Perfect for small, single tenant, IT
apps

The Bad:

Perfect for *small, single* tenant, *IT*
apps

Cannot satisfy TelCo functional,
standard, and legal requirements



Network management requirements in production

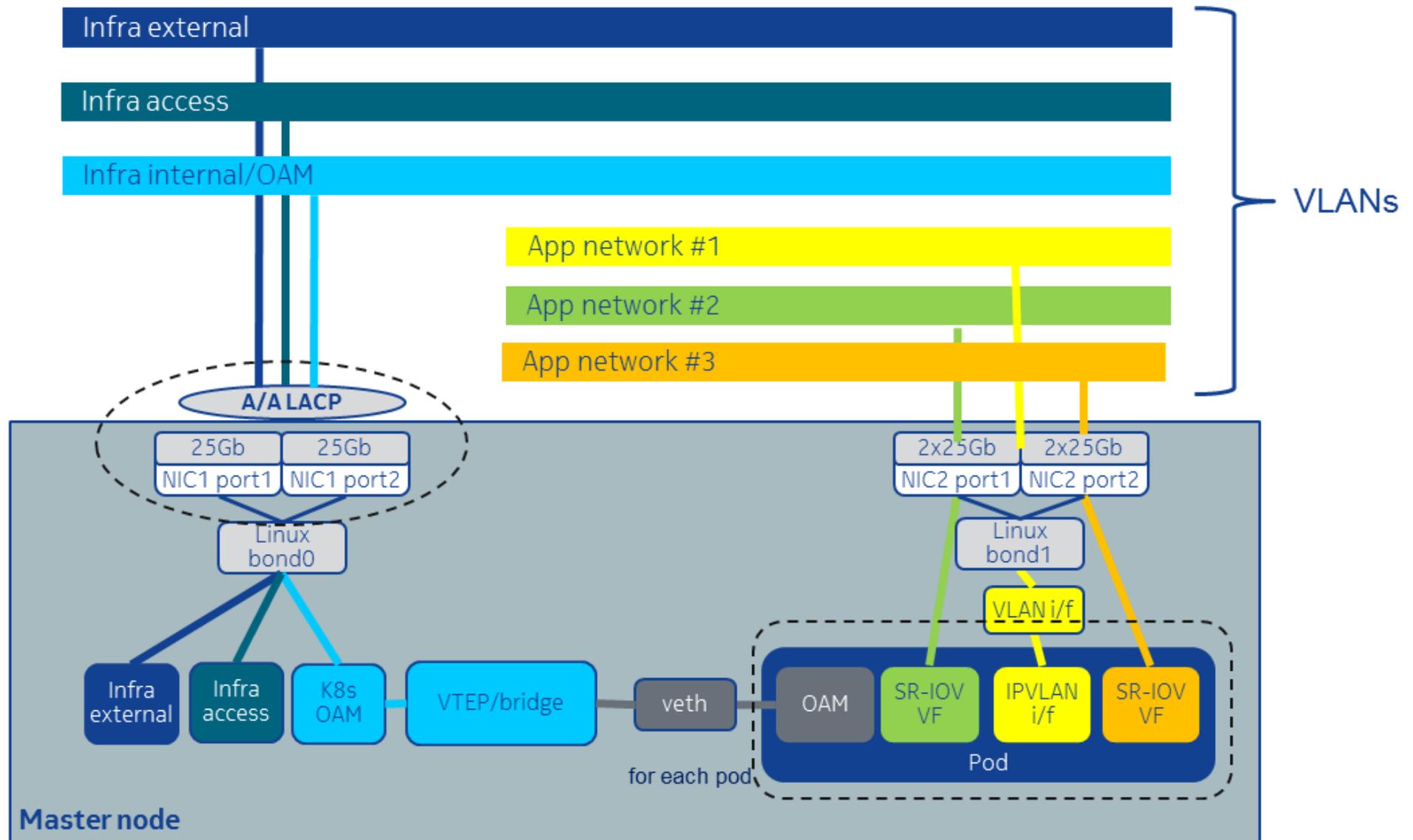


KubeCon



CloudNativeCon

North America 2019



Network management eco-system



KubeCon



CloudNativeCon

North America 2019

Industry consensus: K8s needs a centralized network manager in production
To quote NSM documentation: „An API should exist that allows the **VNF*** to specify its networking intent through an abstract API dynamically.”

Same problem description, different solutions:



[DANM](#) (presenter's choice): hides –but implements- CNI, provides multi-tenant, ***user role specific*** network management APIs to **operators** and **applications**, implements common features in a backend-agnostic manner



[Multus](#): directly exposes CNI to operators



[Network Service Mesh](#): works independently from normal CNI, extra interfaces provisioned post-deployment, targeting specific use-cases

Networking enhanced



KubeCon



CloudNativeCon

North America 2019



<https://github.com/nokia/danm>

API-driven, multi-tenant, multi-role network management integrated to bare-metal cloud underlay network fabric ("host-to-leaf")

Supports multiple, physically separated interfaces

Natively supports multiple, varying network provisioning backends with different characteristics

Centralized, API-driven features (e.g. K8s Services, IPAM, IP routes, VLAN, VxLAN etc.) extended for all interfaces, agnostic of their type

Don't believe the hype



KubeCon



CloudNativeCon

North America 2019

[DANM User Guide](#)

[Akraino REC](#)

[Multiple networks for Kubernetes workloads](#)

Topology awareness baseline

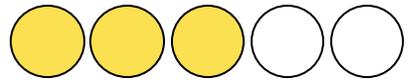


KubeCon



CloudNativeCon

North America 2019



There is now a [Topology Manager](#) in K8s!

The Good:

- Aligns CPUs, and Devices

- Multiple policies available from the get-go ("strict" included)

The Bad:

- Unnecessary restriction on alignment

- Does not align hugepages

- No topology aware *scheduling*



Topology awareness enhanced



KubeCon



CloudNativeCon

North America 2019

Fortunately Topology Manager is a major community focus, so most of the shortcomings are already being addressed!

[History and future](#)

[1.17 issue tracker](#)

Notable and important enhancements:

[Removing restriction of when alignment can happen](#) (1.17)

Hugepage support (1.18)

Topology aware scheduling (under discussion)

The final, definitive, never-to-be-changed K8s 1.16 realtime scorecard is...



KubeCon



CloudNativeCon

North America 2019



References



KubeCon



CloudNativeCon

North America 2019

- DANM: <https://github.com/nokia/danm/>
- CPU Pooler: <https://github.com/nokia/CPU-Pooler>
- CNTT: <https://github.com/cntt-n/CNTT>
- CNCF TUG: <https://github.com/cncf/telecom-user-group>
- REC: <https://www.lfedge.org/projects/akraino/release-1/telco-appliance-radio-edge-cloud/>



KubeCon



CloudNativeCon

North America 2019

Q&A