Sig-node is planning on having a Face to Face meeting in Seattle right before KubeCon.

Location : Menzies Suite at the Grand Hyatt Seattle
Schedule: 11/7/16; 1 pm to 4 pm Pacific Time
VC: https://zoom.us/j/6229641857

Original sig-node mailing list thread here.

If you are interested in attending either in person or via VC, add yourself to the list below

Attendees: **[Limit: 25]**

1. Dawn Chen (In Person)
2. Derek Carr (In Person)
3. Paul Morie (In Person)
4. Harry Zhang (In Person)
5. Euan Kemp (In Person)
6. Jeremy Eder (In Person)
7. Brandon Philips (In Person) 2:30pm+ b/c of OSC Panel. Remove me if you need the room
8. Yifan Gu (In Person)
9. Caleb Miles (In Person)
10. Balaji Subramaniam (In Person)
11. Nicholas Weaver (In Person)
12. Connor Doyle (In Person)
13. Niklas Nielsen (In Person)
14. Dan Gillespie (In Person)
15. Pengfei Ni (In Person)
16. Xu Wang (In Person)
17. David Oppenheimer (In Person)
18. Mrunal Patel (In Person)
19. Piotr Siwczak (TBD)
20. Jędrzej Nowak (In Person)
21. Mike Goelzer (In Person)
22. Stephen Walli (In Person)
23. Ihor Dvoretskyi (In Person)

Proposed discussion topics:

1. Jeremy Eder: Performance-Sensitive Application Platform
2. Brandon Philips: CRI blockers and progress
3. Vish: Runtime agnostic debugging tools
4. Nodespec standardization (Dawn)
    a. Image validation, kernel validation (Dawn)

5. Packaging/running Kubelet and containers (Euan)
    a. Sig-cluster-lifecycle related work
6.

Notes

Performance-Sensitive Application Platform

What is it? Laundry list of features if we could agree on: “Do we want to run ‘high performance applications’ on Kubernetes?”

Running things like telecom, HFT, animation studios, batch jobs. High performance applications usually use specific cpu features, system topology, etc.

Taints and tolerations, vlans vs other things, PCI passthrough of things, separate control plane and data plane.

We are working on some features related to this (like GPU, sysctls, etc). Things like hugepages support, cpusets, etc.

Is it actually obvious that we should support these applications? Derek recalls when discussing “do we add cpusets” Eric Tune had a nice answer of “Yes, but not yet”.

Tradeoffs here of deciding whether we fit out podspec to represent everything possible, or require some super-specialized thing to be unbeknownst to k8s

Maybe we need to support two different ideas of podspec, nodespec, etc.
How do we build that abstraction layer? Represent things like ssd, cpu, node.

One thing we came up with something like a bunch of opaque scheduling via annotations + node flags. Related to a “NodeCell” proposal.
With some special kubelet and so on supports additional knobs.

Lots of this is covered by tains of tolerations, we have the 80%.

We don’t have a yeah lets do it consensus though.

Derek: How do we make this all coexist? We don’t want to have “special kubelets/nodes”. Doe sthis go into the scheduler? At what point is this a wrong knob? Thockin’s point of “we can’t take away knobs once they’re there”.

Dawn: We have low level knobs, like cpus, namespaces, etc. This isn't what the user wants, the user wants performance. User-facing concepts related to that. If you give the user knobs at cgroup, you can start having kernel abstractions leak up (e.g. kernel version, etc). Ref to things like nodespec/kernelspec to make sure it works.

Jeremy: Propose: We break it into two pieces. An admin controlled side. User says "I want my SLO". The admin carves up the cluster. The idea being e.g. "a user wants 'prod' level performance", the cluster operator setup these things. Scheduler is aware, e.g. with taints.

Derek: Similar to NodeClass, you have a StorageClass. But at the end of the day, something has to tell the kubelet something. Does the kubelet understand this? At the end of the day the kubelet has to understand it.

Derek: At the end of the day, one thing that happens here is nodeName -> NumaNode. Do you send that down to the kubelet? How does this actually work? Is the Pod API going to have these knobs? Is that "broken" in some way? Is this a natural extension of this?

DavidOp: Does the scheduler really need to know this stuff e.g. Numa?

Dawn: User does at least need to know this, e.g. for eviction and scheduling.

Euan: Taints and toleration works for most of this "does it fit", we already have those sorts of knobs mostly. Opaque resources + tains/tolerations gets you most of the way. CoreOS TPM admission does sometihng related.
The hard problem is exposing more knobs and figuring out the best way to do that, testing, it, and so on.

Dawn: More levels of abstraction for api -> kubelet -> kernel. Those abstractions and API and harder.

Caleb: Operators! Those could be used to add these concepts.

Derek: We can't get around the kubelet having to do everything.

Jermey: What's the next unit of work?

Derek: Same to you..

Jeremey: cpuset is next up probably… E.g. hugepages is less valueable without this.
Also, single pod per numa node?

Is it exclusive? Depends on the app.
Can we have one solution to all of this, or will there be multiple depending on app?

Derek: What's wrong with the kubelet as it is now that shouldn't be there for this workload? E.g. cfs doesn't matter.

Dawn: Have a daemonset helper to achive this sorta thing (NodeAllocatable and so on)? (Jeremy: We have this prototyped).

Derek: Run kubelet + daemonset schedule pods, have daemonset sees pods and??? Am I understanding? Launches the container?
Dawn: It runs on the node and then edits the cgroup.

Jeremy: Less value if it's not part of the product. We can prototype this stuff..

CRI blockers and progress

- 1.5 - What's the state, what do we expect
    -
- 1.6 - What's the goal there? What's broken? What are blockers? How will we rollout? (In place, disruptive, what have you).
- What are the requirements to stabilize the alpha api? How do we version it?
- Open quesitons as well: Host networking? Monitoring? Promoting alpha -> beta

Dawn: Docker integration problems
- Monitoring has been punted. Do we need container-level stats; and should it be runtime?
- Proposal assumed per-pod cgroup with 1.5 (and cadvisor). But we haven't rolled out pod-cgroup; that could relate to this.
- Do we want to roll this out as the default? What are we missing from the test suite. E.g. kubenet. Still combined as one binary.. How will we rollout cri validation (and cri binaries I guess).
- API Versioning? Do we start from v1alpha1, when / how do we change?
- Feature parity
    - Hostport network
    - (Sorta metrics) (Can't do containers)
        - Filesystem usage (inode, rootfs)

DavidOp:
- I can imagine container-level resource usage use-cases

    - I guess you can run cadvisor (e.g. maybe VPA would want this for per-container basis?)
- This could be useful

Derek: Most people run a small number of containers in a pod, so… though there are exceptions. When we're talking about v1, we only need per-container fs, not memory/cpu/etc.

We all agree that this isn't essential to start with.

Hostport: That's a current miss of the CRI api we think?
Bandwidth shaping: Not included in CRI currently. Do we want to punt? Do we want to get rid of it?
- Derek: We should have a firstclass way of supporting bandwidth shaping eventually. Once it's supported, ew can plumb it into CRI
- Dawn: This was intentionally not included in CRI; killing bandwidth shaping

Dawn: We want to build a CRI validation test that we can use to validate all the runtimes..

Derek: We have experimental features on kubelet that will not work for CRI implementations. Do we have a good list of kubelet-features <-> CRI? Have we vetted the list?
- Dawn: What's hte prereq for CRI? We have skipped some things (like userns). We need the community to know whether we cut the right things?
- Dawn: API, nothing blocker. Note, that v1 there can be no version skew for the compiled in CRI bits.
- Dawn: How will we make the switch/rollout for 1.6

Derek: Do we delete the old integration when we turn it on? Do we have a flag to switch back?
Do kubelet developers need to support both codepaths for longer?
Do we stop adding new features in the old version?

Dawn: If we switch to new api in 1.6? If we do, 1.6 and 1.7 we probably add it in both place and keep both. Keep feature parity in both places for that duration.
Dawn: Too early to actually know for sure.

Dawn: What features work for the other runtimes?

Mrunal:
- Working out image api, getting it to run
- Working on kubelet -> cri-o integration

Euan:
- Things in general do work

- Missing: Have’nt vendored in (scary because of deadline, in-flight), metrics, logging, attach (logging/attach WIP),

Dawn:

- 1.5: have alpha api, don’t actually use it, but treat it as a mostly stable api. Document any known issues
- 1.6 work on missing things like test suite and hostport and monitoring and so on. Think about api-version, decide how to move it to beta.

We’re shipping alpha for 1.6? Dawn :Yes
Reminder 1.6 is going to be a stabalization release. Maintenance.
Derek: Not as urgent to improve quality of code as actually finish our missing features.

Security concerns: Auth stuff for exec/streaming handlers is lots of TBD, alpha, scary. Also has problems with charging to a given pods.

Brandon: Target a release for switching over?

- 1.6… Hopefully. Dawn and Derek are both agreeing.

Derek:

- But doesn’t it have to be beta etc for that? Dawn: we don’t need to.

Dawn:

- If we want to switch in 1.6, we need to figure out features we need to support, what features we won’t need or will rewrite. Need to do it case-by-case.

We need to talk about api versioning: Decouple docker default integration / internal refactor vs CRI as a streaming api.

Do we have a list of things we need to do?

- E.g. auth

Testing:

- Cluster level?
- Node level?
- CRI validation?

Runtime agnostic debugging tools

Different runtimes, different storage on disk. You're forced to go through the actual implementation

Fundamental tension between kubelet knowing information, CRI, and so on.

What is this feature we're talking about?

Brandon: Any time in the docs you use docker effectively, we need to have an alternative tool that owrks for arbitrary runtimes. E.g. image management, container management / listing.

Brandon: We're right now pushing parsing onto the user for things like docker ps.

Would parsing checkpointed data help?

Dawn: As soon as we checkpoint we'll need to have versioning, backwards compatibility, migrations, etc.
We can come back to checkpointing after we have CRI and so on to make sure we know exactly what we want.

Derek: We need to think through the actual use cases here more, e.g. there's ones where you ned the runtime, and things where you do have a healthy kubelet. Do we also need to think about tools that do work for healthy kubelet vs runtime.

Philips: Introspection happens too, e.g. etcd we can turn on live tracing.. But customers are still string-parsing.

Derek: Warstories, painful to debug (worst include nfs basically blocked volumes, destroys nodes).. For that, we would tell the customer to turn on things to get better debugging tools. Don't want to have to tell them to restart kubelet with different flags, etc.
Are we all thinking through the right thing for debugging usecases ,etc? Are we thinking of the ways to ask things from the kubelet like how to do this or that, I want tihs or that?
(another warstory): Node was suddenly going to not reday, but events were expired. We're bad at logging.. Events are okay, but it's not obvious. It's not always logged, etc.

Brandon: We could at least for 1.6 audit logs.
Wish kubelet would integrate with systemd watchdog; kick the kubelet for certain events.
Derek: Overcommitted nodes are another common debuggability issue.. Reboot all together and suddenly it starves itself effectively.
Is tihs CRI or kubelet even for staggering this?

Volume subsystem visability:
Identifying those and figuring out the kubelet's state would be valuable.
Something we did for the controller is publish metrics which lets you ask your queue depth, process, rate, and so on.

Dawn: We now have node problem detector, etc. Some of these problems are historical (e.g. old dockerclient etc).
We have to use these things and give feedback; we don't know all the problems etc.
Derek: NodeProblemDetector didn't work on RH, not deployed. Rhel has abrt which is a sorta similar thing too.
Jeremy: abrt matches core patterns and backtraces, but does'nt mark nodes as bad, drain, etc.
Dawn: Can integrate abrt with it. "By design it's extensible and can integrate into that".

Brandon:
Euan: We can at least all agree this isn't something to solve perfectly until after CRI, for now incrementally improve.

Nodespec standardization