Will it blend?

A comparison of oVirt, OpenStack® and kubernetes schedulers

Martin Sivák
Principal Software Engineer
Red Hat Czech

3th of Feb 2018

Agenda

Anatomy of a scheduler

- Goals
- Design considerations
- The three schedulers

Architecture similarities and differences

- Resource tracking
- Scheduling algorithm
- Balancing and preemption

Highlights and ideas to share

Goals of a scheduler

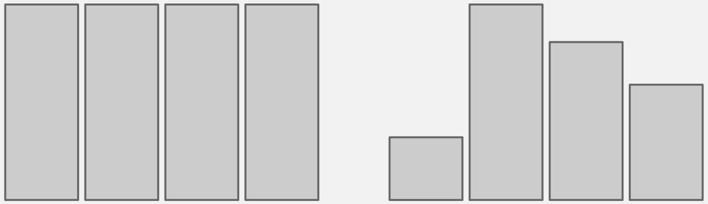
Find a place with enough resources
to start the given VM[1] ...

… and make sure it keeps running
… and make sure it handles the load
… and keep the power consumption low
… and ...

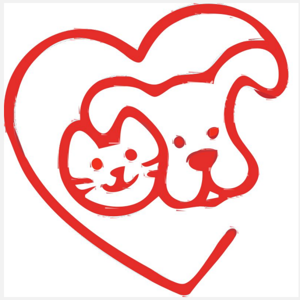
[1] or container

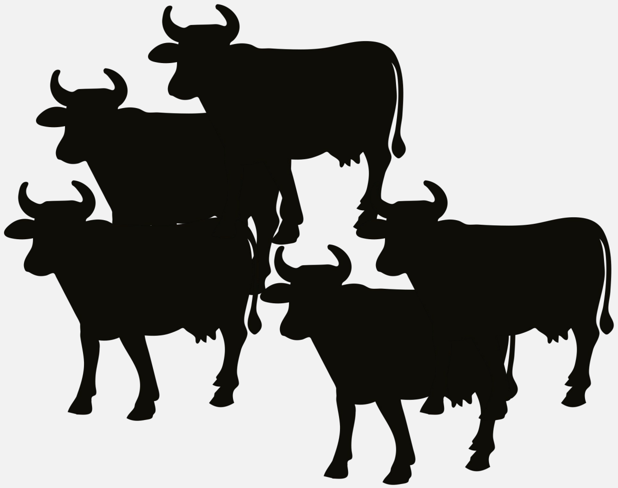
Design considerations

- Size of cluster (~ hundreds of nodes)
- Deterministic algorithms
- Migrations and balancing
- Homogeneous cluster vs. heterogeneous cluster

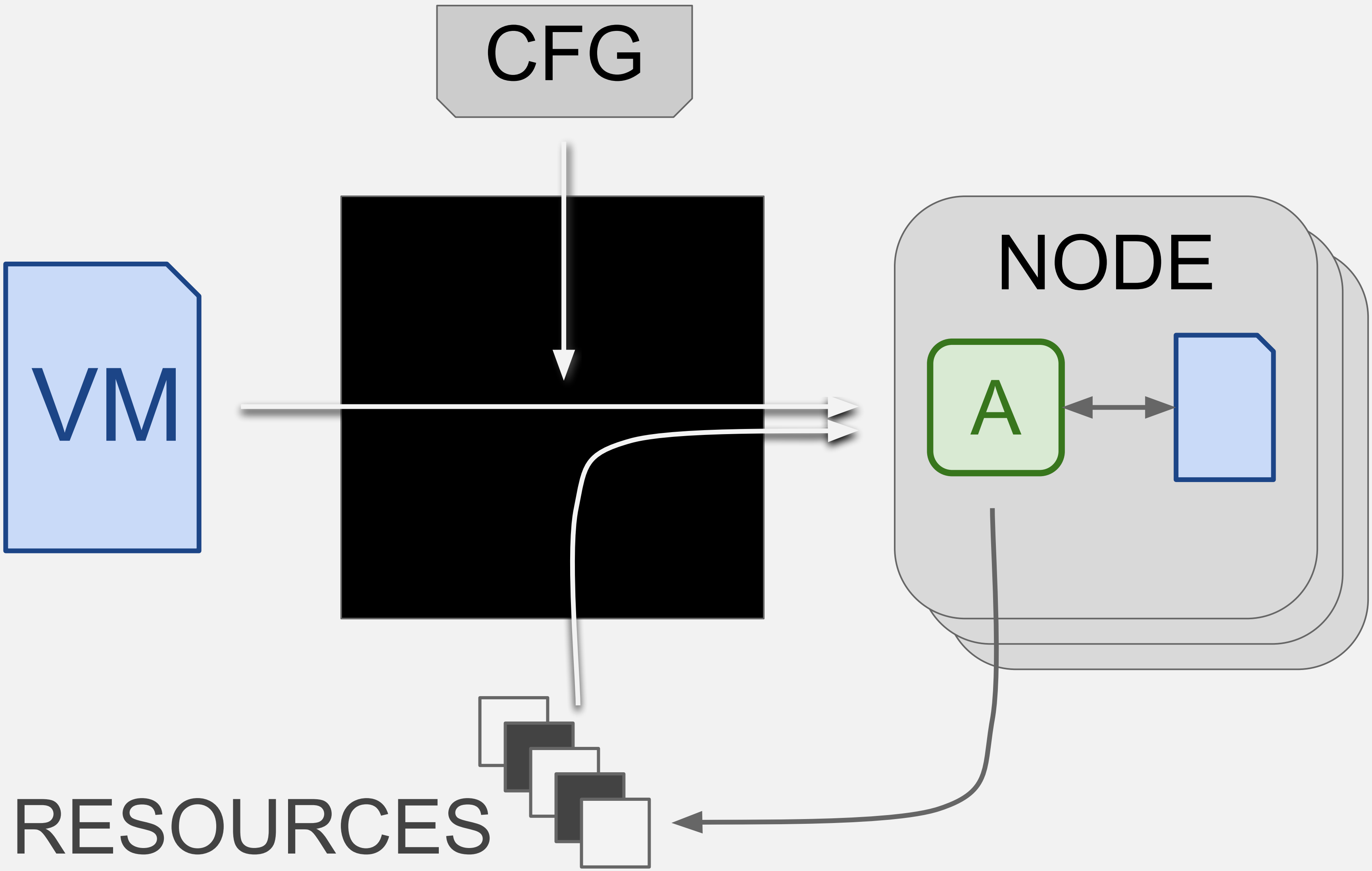


- Pet vs. cattle





Scheduler as a function



The schedulers

oVirt

openstack®

Number comparison

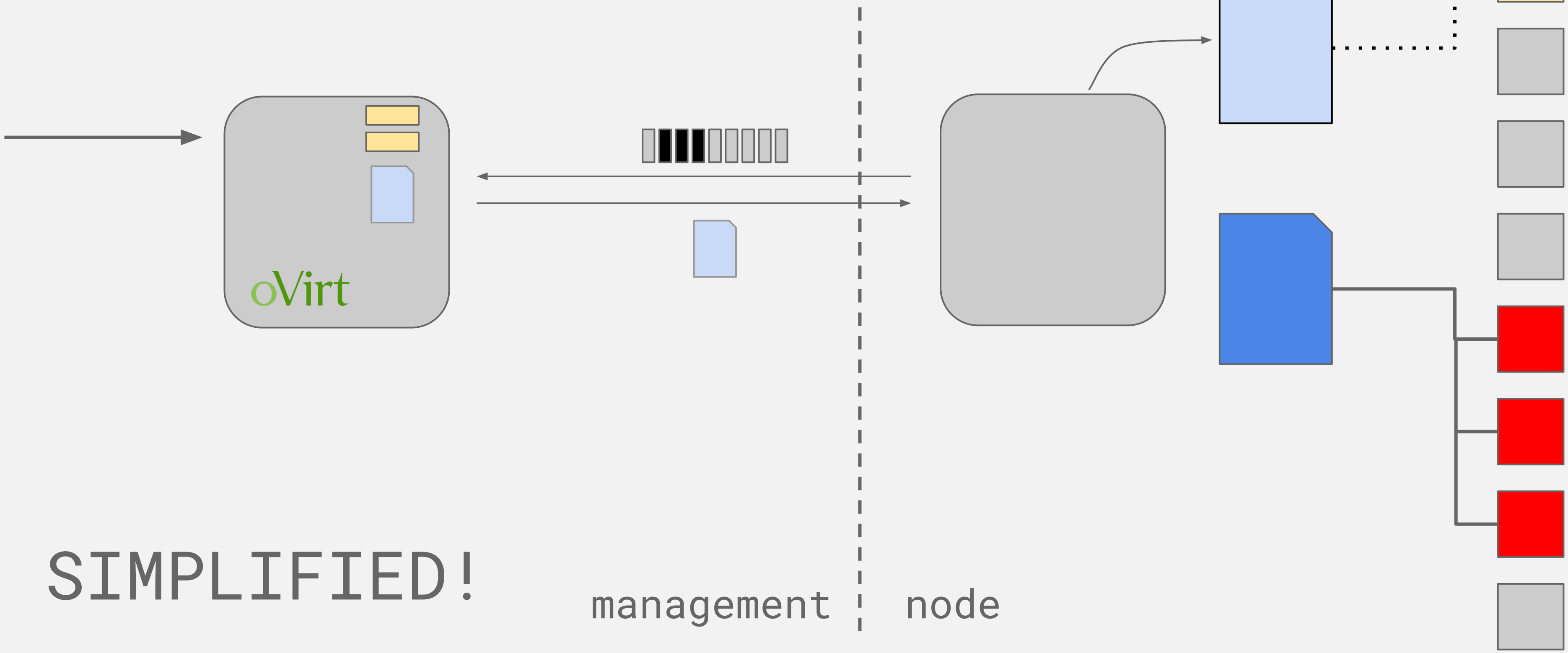
	oVirt	OpenStack	kubernetes
~ Max nodes	200	~300	5000
Language	Java	Python	Go
Load type	pet VMs	cattle VMs	containers
Resource tracking	pending + stats	placement service	pod spec in etcd
Active schedulers	1	1 or more	1 or more

Resource tracking

Resource tracking

oVirt

- pending resources are tracked,
  free resources come from reports

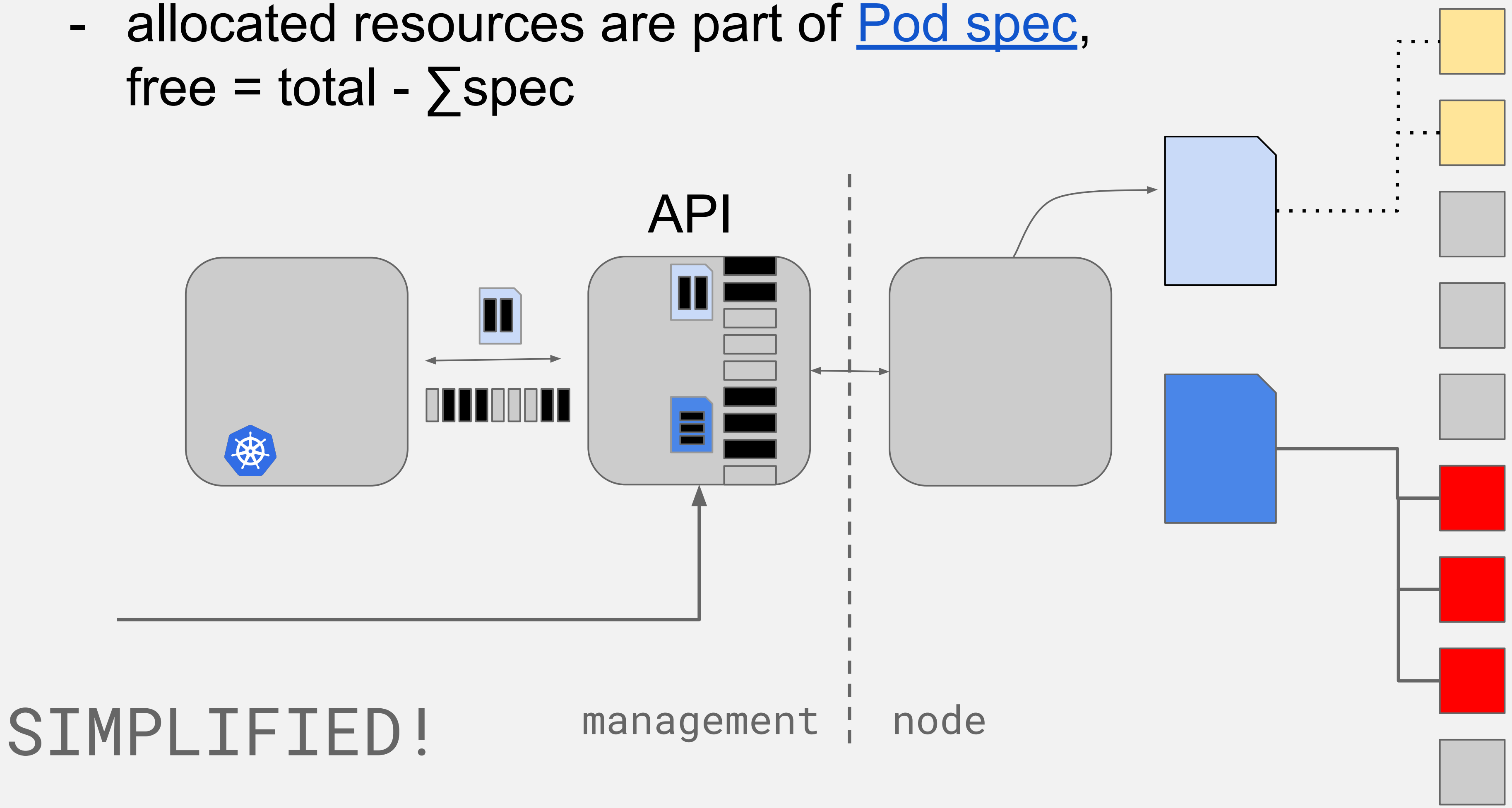


SIMPLIFIED!

Resource tracking

kubernetes

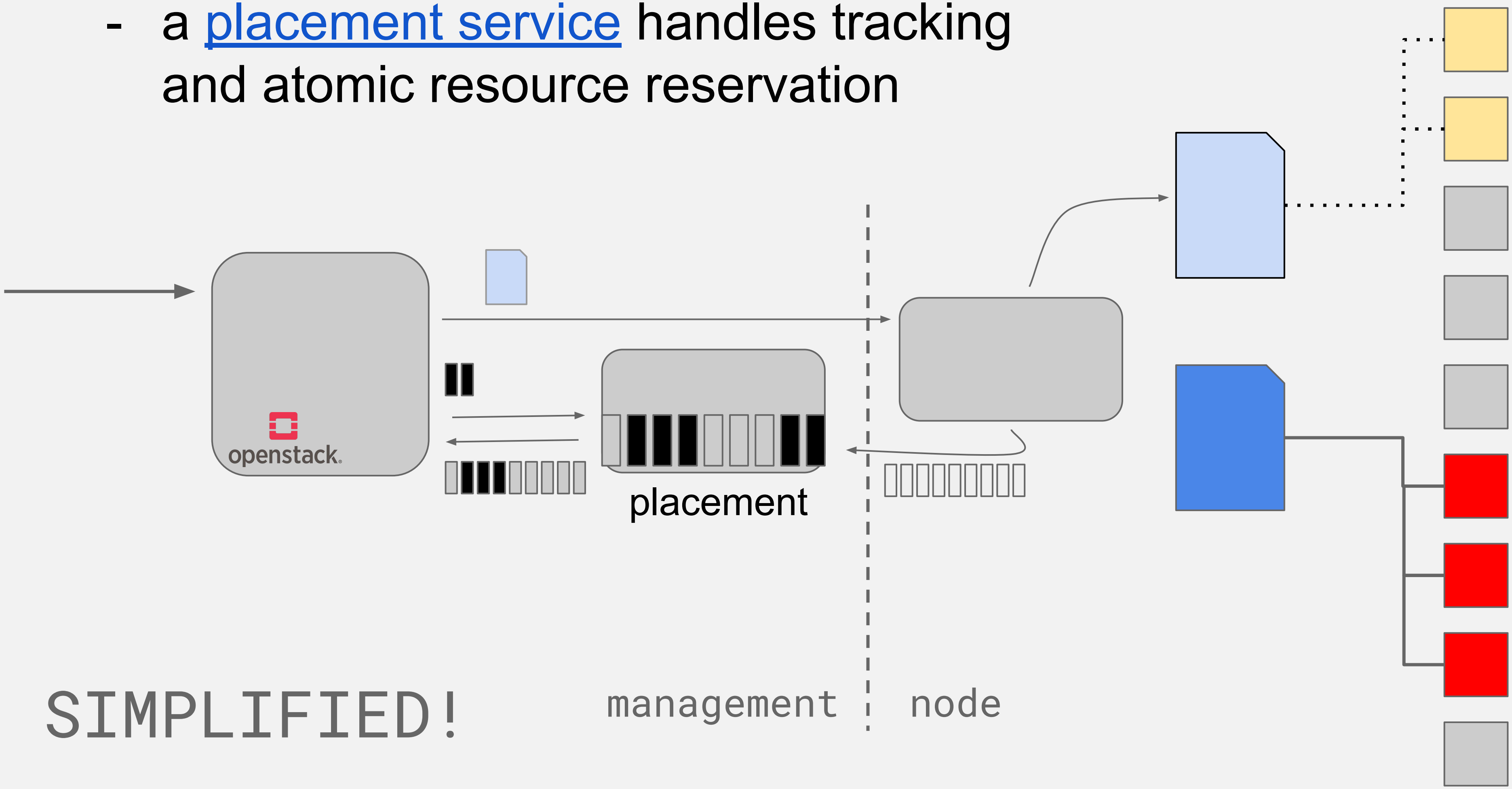
- allocated resources are part of [Pod spec](), free = total - ∑spec



Resource tracking

OpenStack

- a placement service handles tracking and atomic resource reservation



SIMPLIFIED!

The Algorithm

Algorithm - not rocket science

yes / no

Filter — Remove all nodes that do not satisfy hard constraints

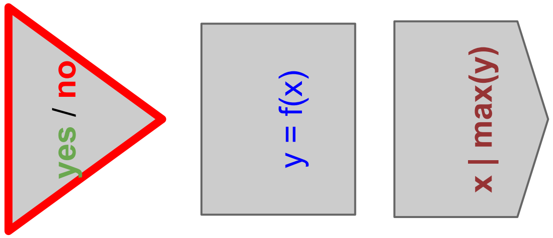
y = f(x)

Map — Compute score, typically based on node load and free resources

x | max(y)

Reduce — Select the best node

Filtering



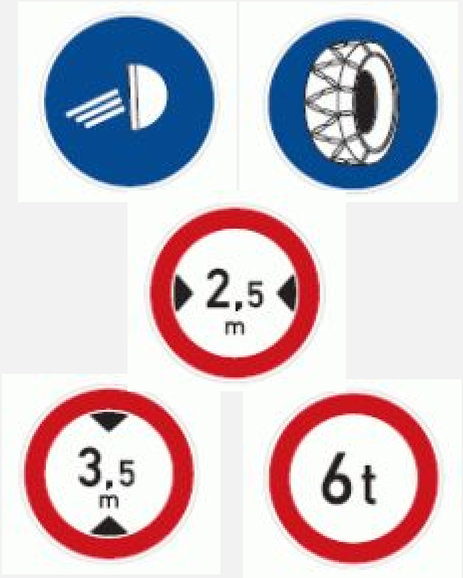
Filter out incompatible nodes

Typical filters:

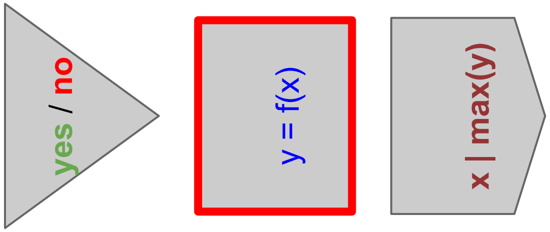
- CPU compatibility
- Free RAM
- Network presence
- Storage connectivity

Highlights:

- Affinity
- Load isolation and trust
- **Labels**

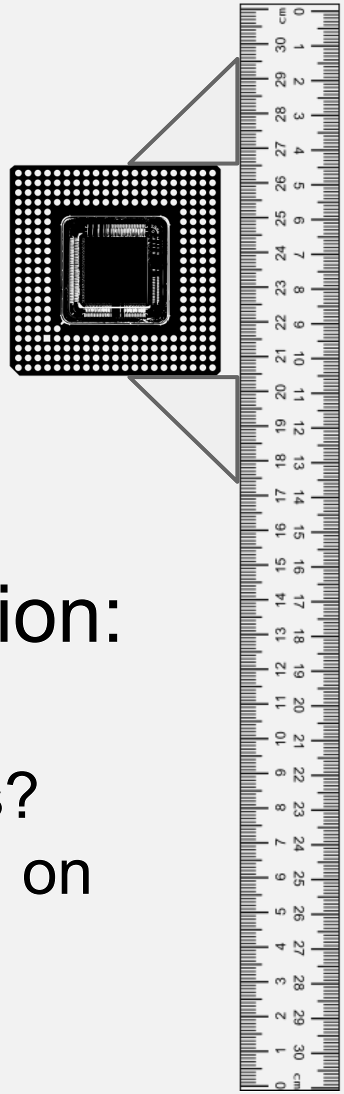


Scoring



Map a metric to a score

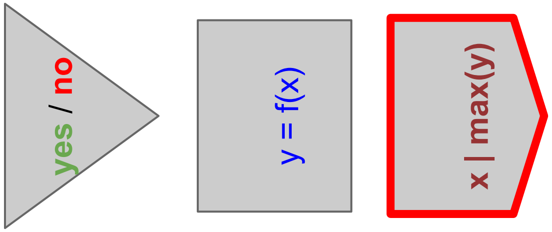
like CPU load 10% to 10.



Different metrics require different representation:

- CPU cores, running VM count - absolute number
- Free memory vs used memory - absolute or percents?
- CPU load vs “free” CPU - percents, something based on frequency? SMP?
- Label presence - boolean

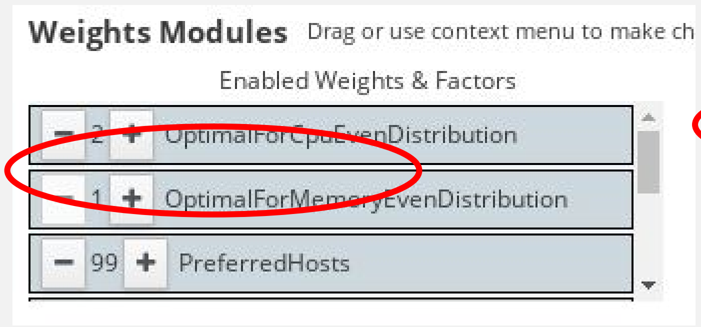
Selecting the destination



Which node is the best? … it depends on the **goal**

- Maximizing performance, saving power or upgrade process?

Multiple metrics need **multipliers** for importance




```
nova.conf:

weight_setting =
"metric1=ratio1,metric2=ratio2"
```

```
kind: "Policy"
version: "v1"
predicates:
...
priorities:
...
 - name: "RackSpread"
   weight: 1
```

So which node is the best then?

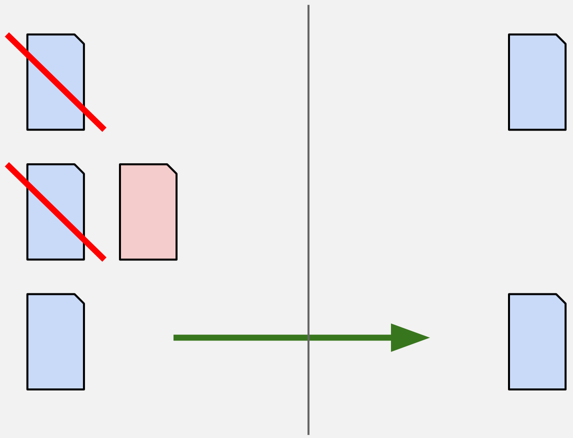
- How do you sum 10%, 3.5GiB and 16 together?
- **Normalization!**

Score normalization

Project	Algorithm	To	Note
oVirt	rank	-	compresses differences
OpenStack	scale / maximum over all hosts	0 - 1	depends on filter results
kubernetes	scale / single host	0 - 10	incorrect on heterogeneous clusters

Balancing and preemption

Balancing and Preemption



Methods

- offline migration (kill & re-start)
- preemption (kill & start other)
- live migration (move)

“Situations” emerging at runtime

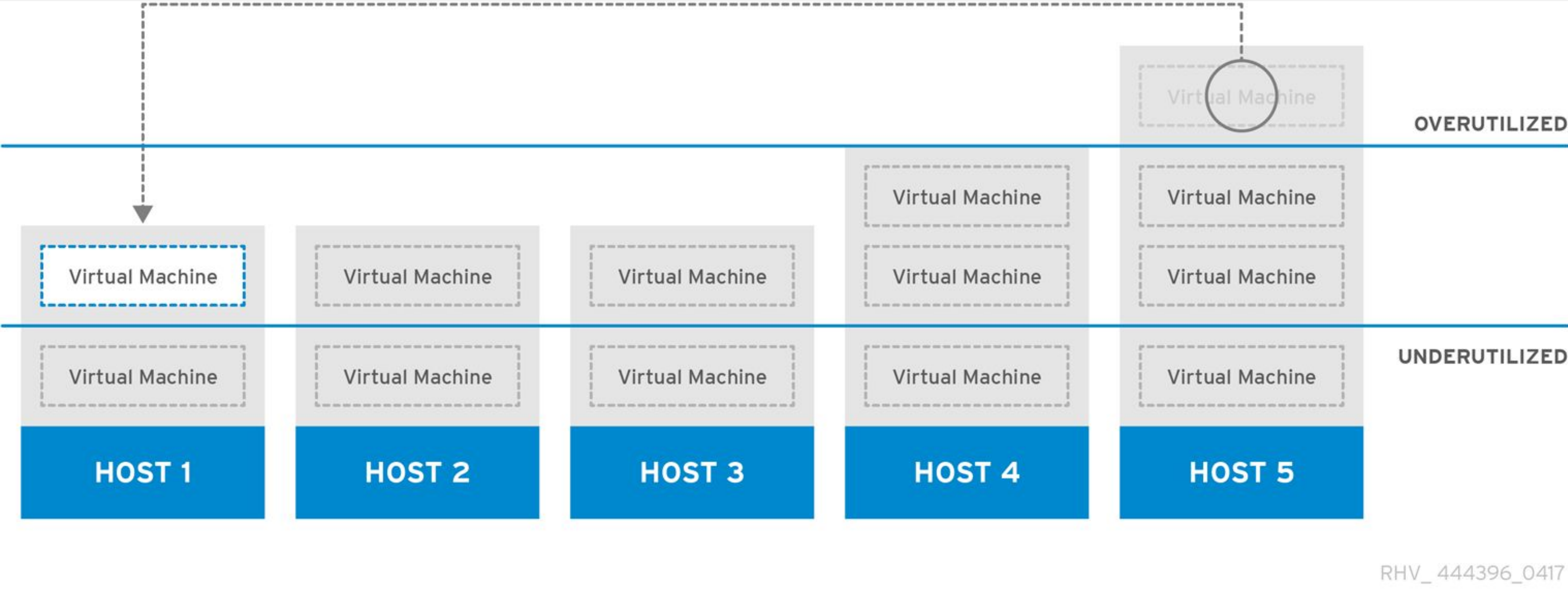
- overload
- rule violations (eg. new affinity defined)

Selecting the best move

- select the object and select the move
- remember the **deterministic assumption**
- **HARD!**

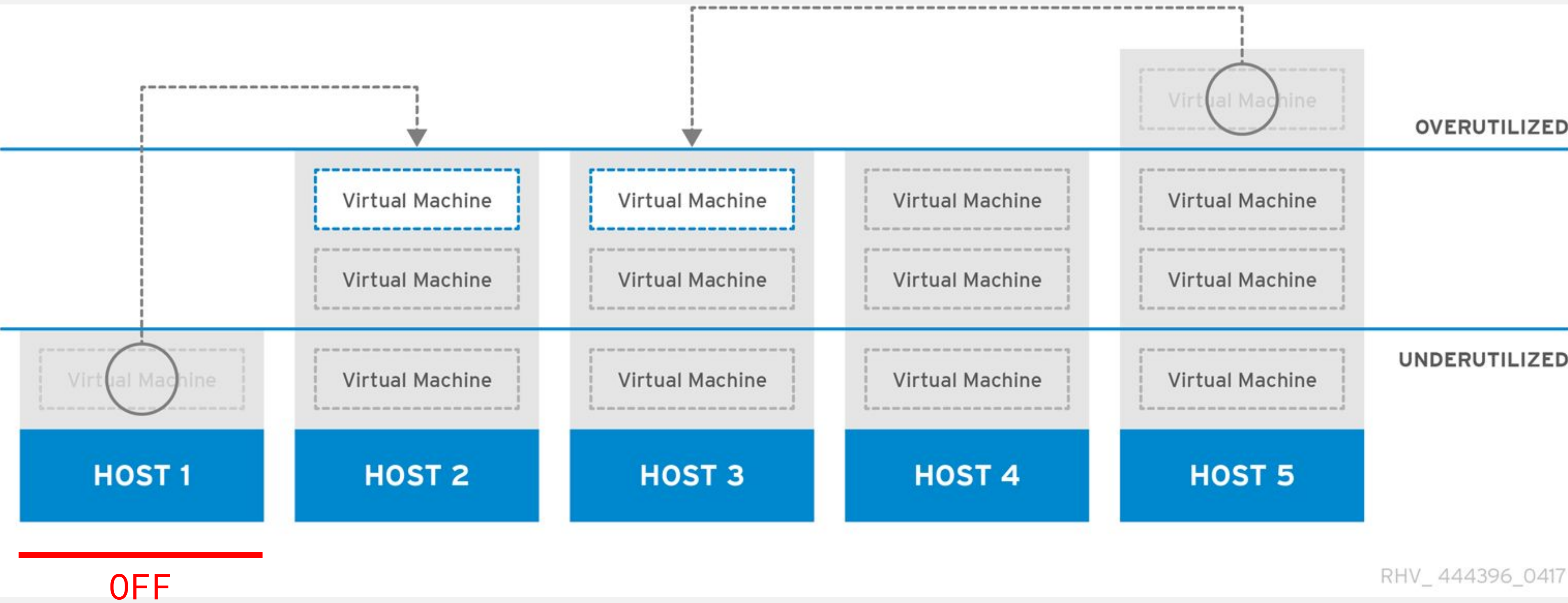
Balancing - oVirt

Load balancing - equally balanced policy



Balancing - oVirt

Load balancing - power saving policy



Preemption - kubernetes

Can we kill low priority load when needed?

- Guaranteed load scheduling (DNS, network controller)
- Eviction policy (Help! I am overloaded)
- Disruption budget (Feel free to use one of mine)

Preemption in use elsewhere:

- AWS spot instances - money based priority

Highlights and good ideas

Interesting highlights

Scheduling:

- oVirt optimizer (probabilistic scheduling service)
- Chance scheduler (random selection)
- Arbitrary filtering rules in spec (booleans, operators)

Host devices:

- resource hierarchy and host device aliases

Resource tracking

- declarative and reactive - scheduler fills in data to Pod spec

Good ideas

- **labels**
- **normalization** methods
- **atomic** resource tracking and **reservation**

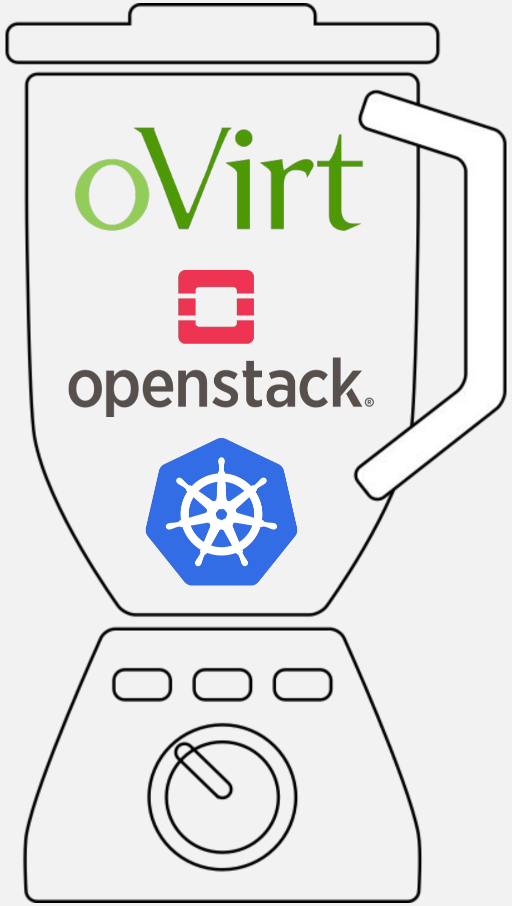
- multiple schedulers and split-brain protection
- balancing and preemption

Summary

All three schedulers are very similar in concept

Differences are small and based on the needs of the typical workload

There are ideas worth sharing!



THANK YOU !

Martin Sivák
msivak@redhat.com

with thanks to Red Hat's OpenStack team