KubeCon | CloudNativeCon

North America 2017

# A Scheduling Simulator for Capacity Estimation of Kubernetes Clusters

Avesh Agarwal, Red Hat

# Introduction

- Work at Red Hat
- Core contributor to Kubernetes and OpenShift
- Descheduler
  - https://github.com/kubernetes-incubator/descheduler
- Cluster Capacity
  - https://github.com/kubernetes-incubator/cluster-capacity

# Agenda

- Motivation
- Design
- Features
- Usage
- Demo

# Motivation: Why?

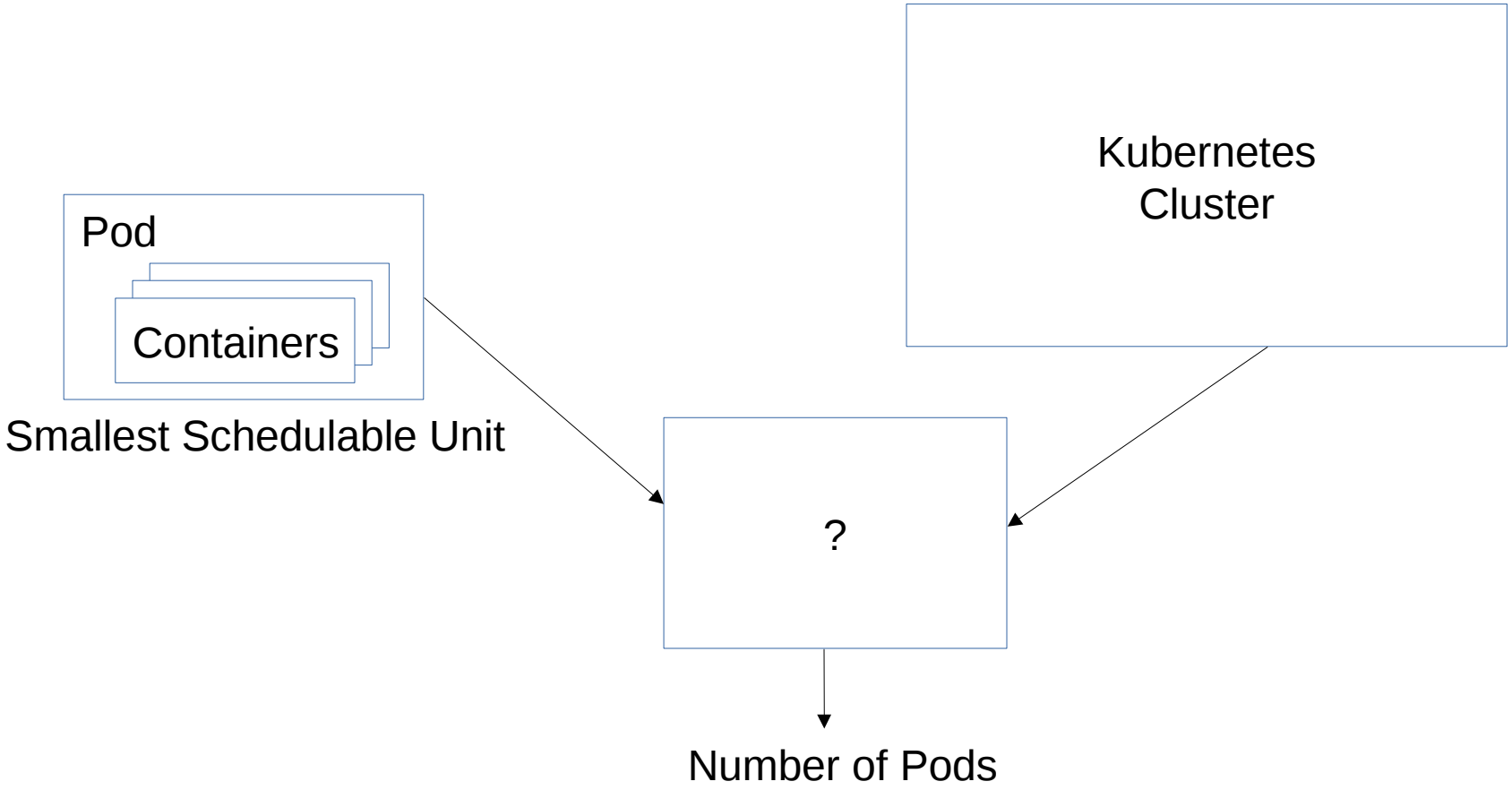
- A bit of history
- Capacity planning
  - How much capacity is left in a cluster?
  - When to add more capacity to a cluster?
- Workload types
  - Uniform and Variable
- Lightweight
  - Simple, easy to use

# Motivation: Why?

- Capacity
  - Consumable resources
    - CPU, Memory, Storage, GPUs etc.

Capacity in terms of individual resources ?

# Motivation: Why?

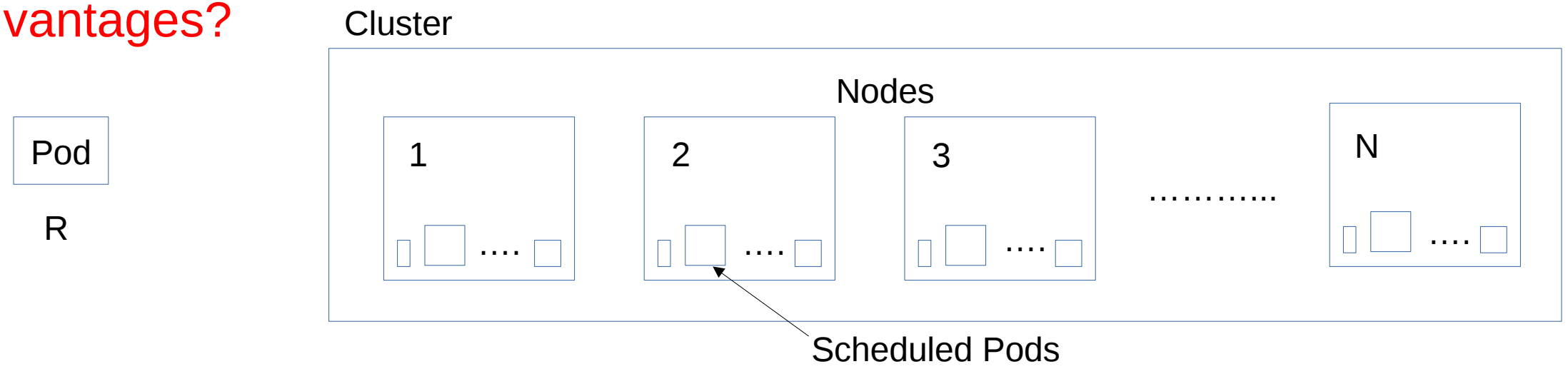


# Design

- Resource based approach
- Scheduler based approach

# Design: Resource based approach

Disadvantages?



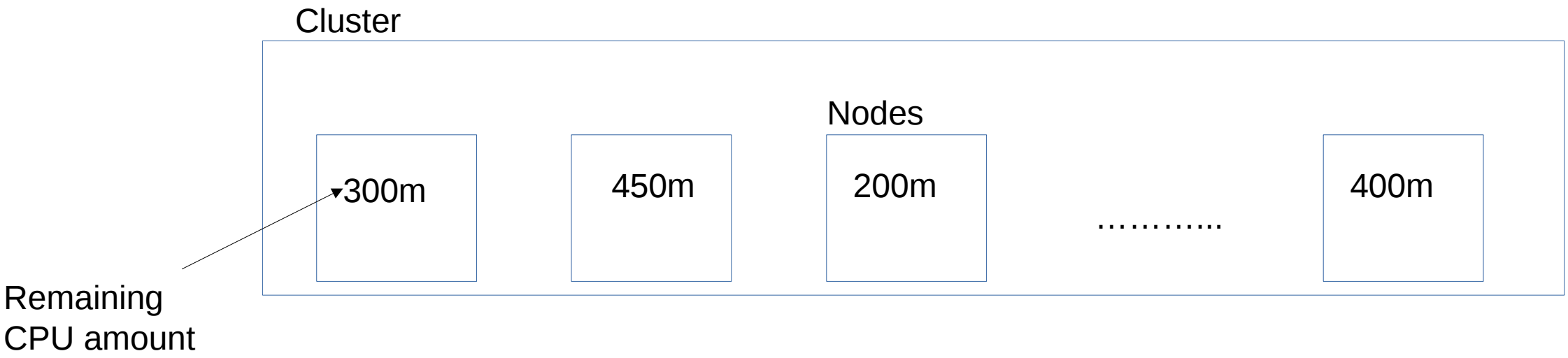

$$\text{Instances} = \frac{(A_1 - R_1) + (A_2 - R_2) + (A_3 - R_3) + \ldots\ldots\ldots + (A_N - R_N)}{R}$$

- Minimum among instances for all resources

Is it enough to consider resources only?

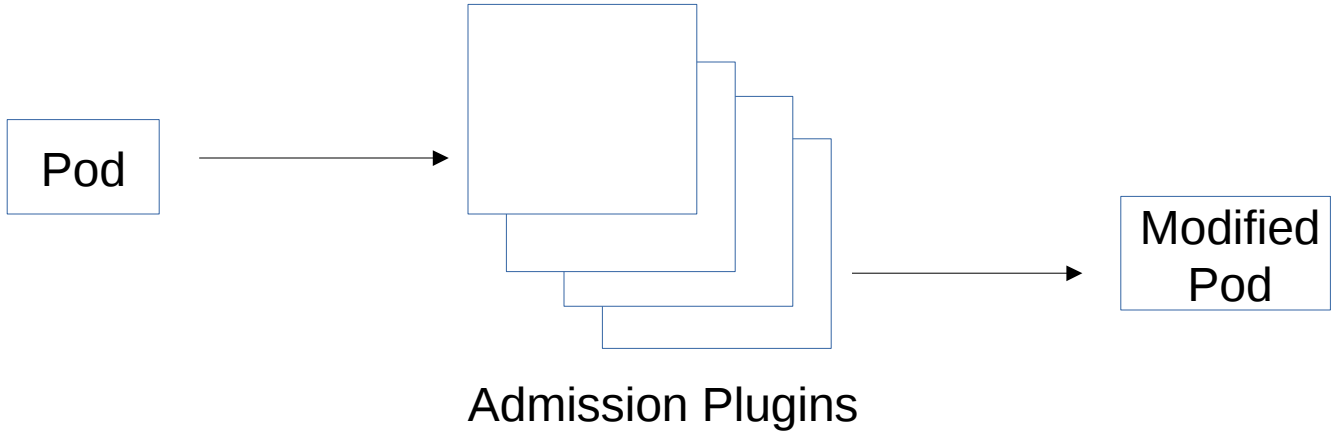
# Design: Impact on Scheduling

- Fragmentation
  - CPU example
  - Input pod’s CPU request: 500m



# Design: Impact on Scheduling

- Admission Plugins
  - PodNodeSelectors
  - PodTolerationRestrictions
  - Others



# Design: Impact on Scheduling

- Node selector and affinities
- Node taints and pod tolerations
- Pod affinities or anti-affinities
- Max number of pods on a node
- Pod’s storage requirements (PVC, Volumes)

# Design: Scheduler based

- Architecture
  - Creates an instance of the default scheduler
  - Maintains local resource store
  - Fetches initial state from cluster
  - Schedules input pod one by one
  - Does not bind to nodes
  - Stops when no more pods can be scheduled

# Features

- Proactive approach
- Run via command line or or as a pod
- Custom scheduler policy
    - Scheduler config file as input
- Max limit
    - Is it possible to schedule this many instances?
- Pod’s resource requirements from LimitRange
- Distribution on each node
- Currently considers uniform workloads

# Usage

- Running as a command line tool

*$ cluster-capacity --kubeconfig <path-to-kubeconfig> --podspec <path-to-pod-spec>*

  - Output is the number of pods that can be scheduled
  - Resource requests must be specified
    - Pod spec
    - LimitRange
  - *--verbose* flag
    - Shows distribution of pods among nodes

# Usage (Best Practices)

- Running as a pod in a Job
  - CC_INCLUSTER environment : true
  - Service account
    - Separate service account
    - Minimum required RBAC permissions
  - RBAC permissions: cluster role
    - *resources: ["pods", "nodes", "persistentvolumeclaims", "persistentvolumes", "services"]*
    - *verbs: ["get", "watch", "list"]*
  - Input pod spec
    - Mounted as a ConfigMap

# Status

- Tested with Kubernetes and OpenShift clusters
- Rebased to Kubernetes 1.7
- In OpenShift since 3.6

# Future

- Admission plugins
- ReplicaSets, Delpoyment, Jobs, StatefulSets
- Variable workloads
  - Accept a list (sequence) of pods
- Your use cases
- Multiple schedulers

# Demo

# Resources

- Design
  - https://github.com/kubernetes-incubator/cluster-capacity/blob/master/doc/cluster-capacity.md
- Kubernetes incubator repo
  - https://github.com/kubernetes-incubator/cluster-capacity/
- How to use with OpenShift
  - https://docs.openshift.com/container-platform/3.6/admin_guide/cluster_capacity.html
- Container image
  - docker.io/aveshagarwal/cluster-capacity
  - https://store.docker.com/community/images/openshift/origin-cluster-capacity

#sig-scheduling

Questions?