Heterogeneous Resource Scheduling Using Apache Mesos for Cloud Native Frameworks

Sharma Podila
Senior Software Engineer
Netflix

Aug 20th
MesosCon 2015

Agenda

- Context, motivation
- Fenzo scheduler library
- Usage at Netflix
- Future direction

Why use Apache Mesos in a cloud?

Resource granularity

Application start latency

A tale of two frameworks

Reactive stream processing

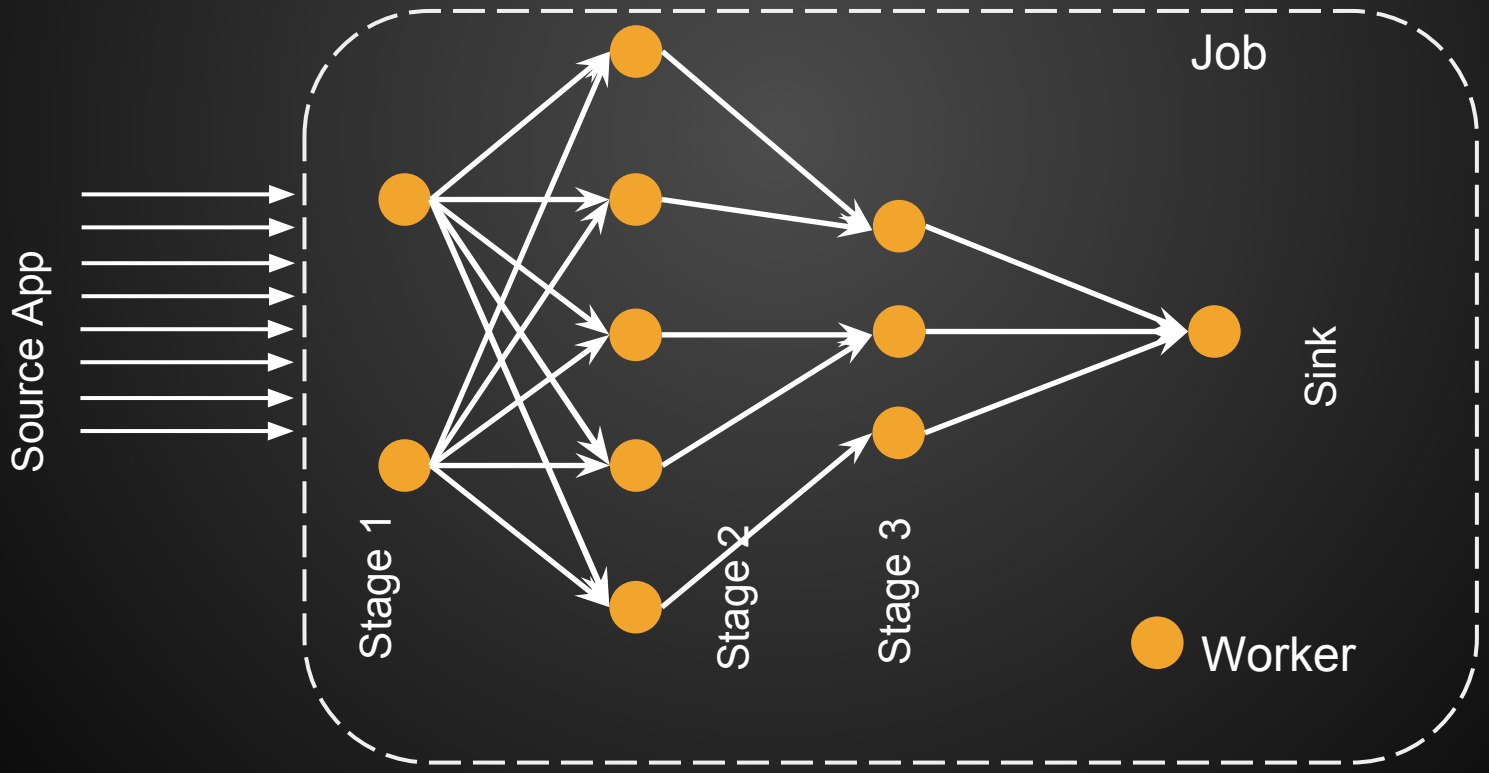
Container deployment and management

Reactive stream processing, Mantis

- Cloud native
- Lightweight, dynamic jobs
  - Stateful, multi-stage
  - Real time, anomaly detection, etc.
- Task placement constraints
  - Cloud constructs
  - Resource utilization
- Service and batch style

Mantis job topology

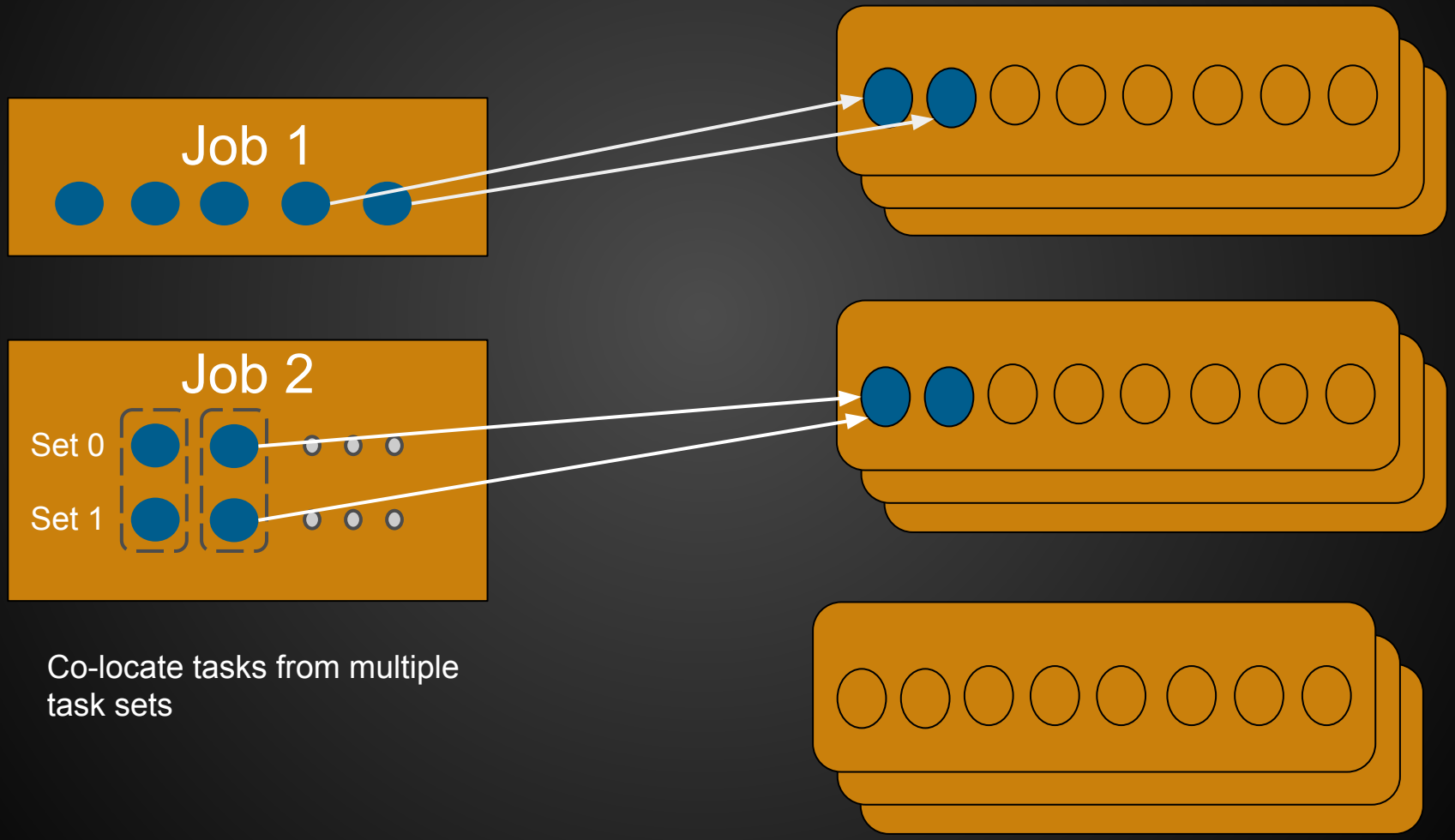
A job is set of 1 or more stages
A stage is a set of 1 or more workers
A worker is a Mesos task



Container management, Titan

- Cloud native
- Service and batch workloads
- Jobs with multiple sets of container tasks
- Container placement constraints
  - Cloud constructs
  - Resource affinity
  - Task locality

Container scheduling model



Co-locate tasks from multiple task sets

Why develop a new framework?

Easy to write a new framework?

Easy to write a new framework?

What about scale?

Performance?

Fault tolerance?

Availability?

Easy to write a new framework?

What about scale?

Performance?

Fault tolerance?

Availability?

And scheduling is a hard problem to solve

Long term justification is needed to create a new Mesos framework

Our motivations for new framework

- Cloud native
  (autoscaling)

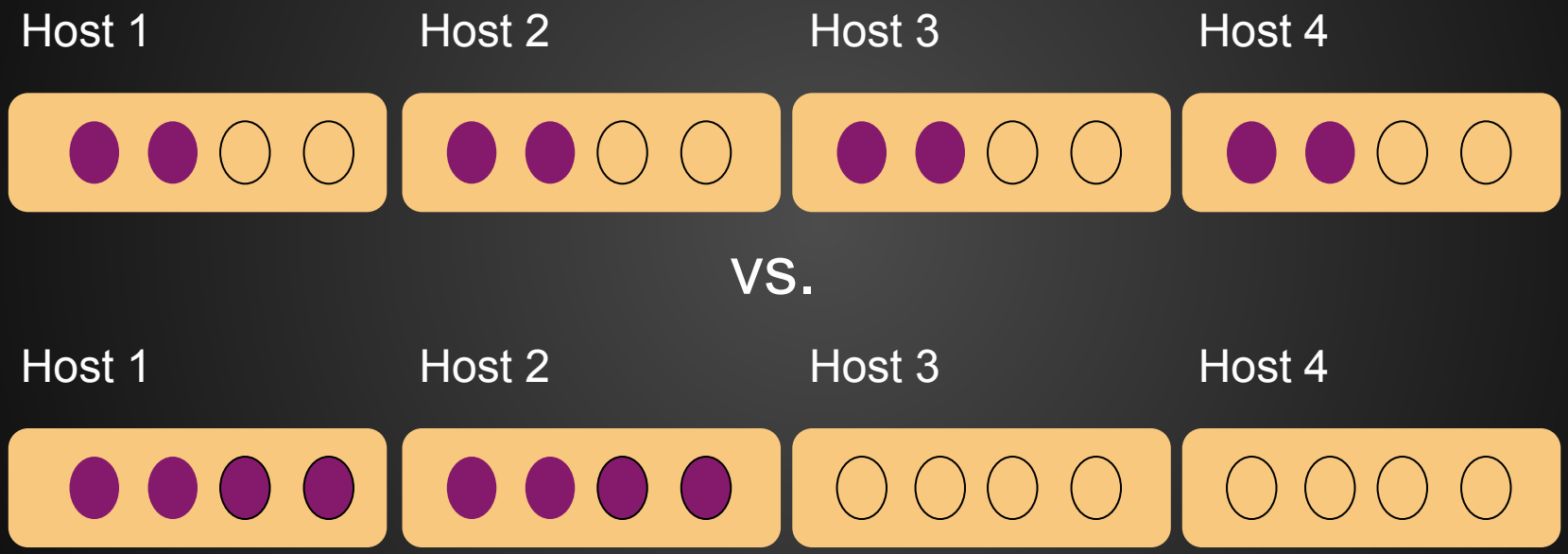
Our motivations for new framework

- Cloud native
  (autoscaling)

- Customizable task placement optimizations
  (Mix of service, batch, and stream topologies)

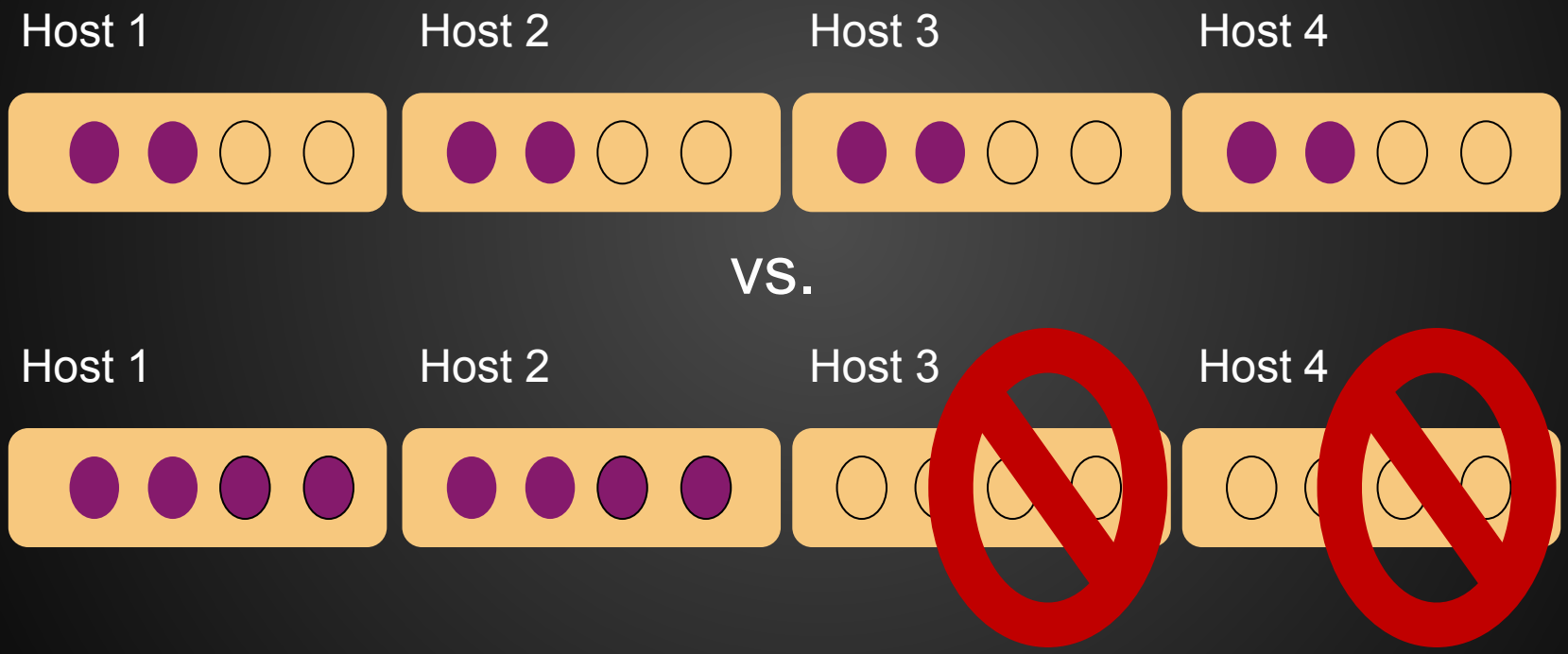
Cluster autoscaling challenge

For long running stateful services



Cluster autoscaling challenge

For long running stateful services



Components of a mesos framework

API for users to interact

Components of a mesos framework

API for users to interact

Be connected to Mesos via the driver

Components of a mesos framework

API for users to interact

Be connected to Mesos via the driver

Compute resource assignments for tasks

Components of a mesos framework

API for users to interact

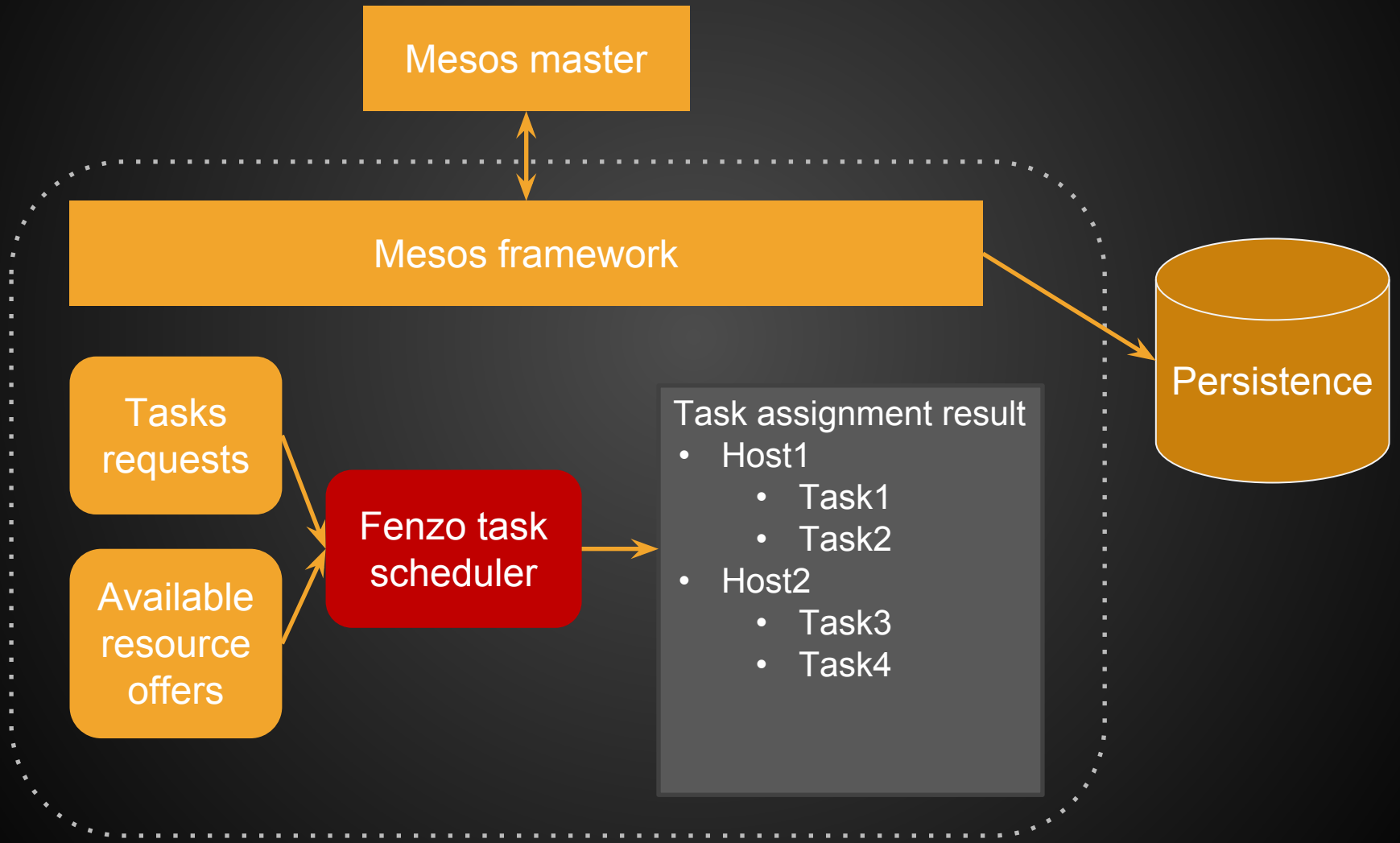
Be connected to Mesos via the driver

Compute resource assignments for tasks

Fenzo

A common scheduling library for Mesos frameworks

Fenzo usage in frameworks



Fenzo scheduling library

Heterogeneous task requests

Heterogeneous resources

Plugins for Constraints, Fitness

Autoscaling of cluster

Visibility of scheduler actions

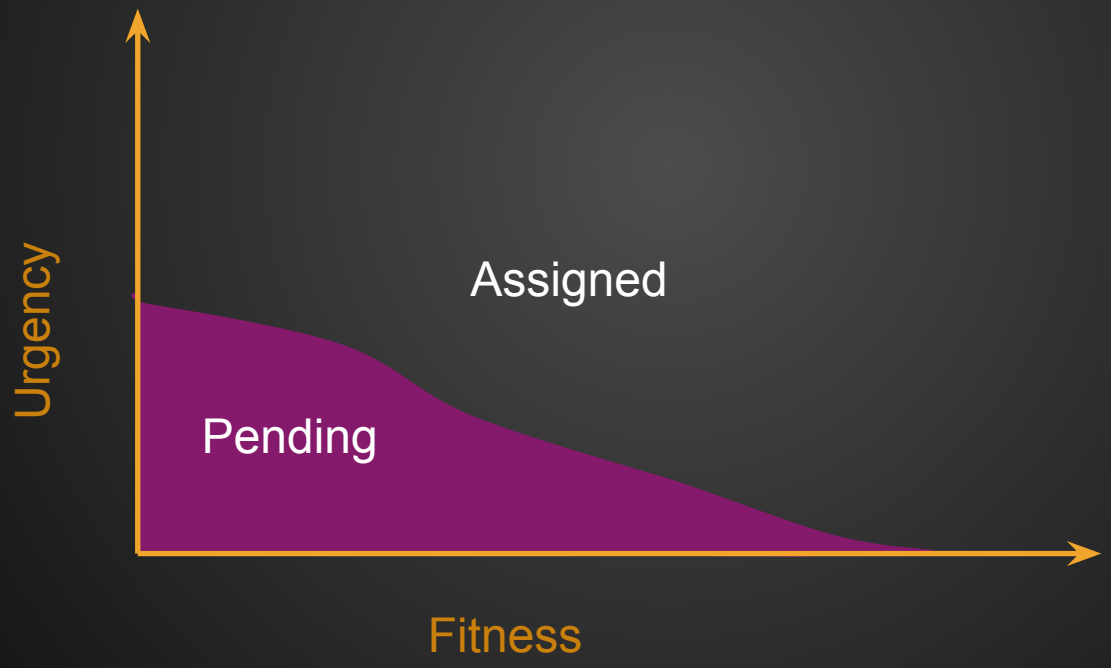
High speed

Announcing availability of Fenzo in Netflix OSS suite

Fenzo details

Scheduling problem

N tasks to assign from *M* possible slaves



Scheduling optimizations

First fit assignment

Optimal assignment

~ O *(1)*

Real world trade-offs

~ O *(N * M)*[1]

[1] Assuming tasks are not reassigned

Scheduling strategy

```
For each task
   On each host
      Validate hard constraints
      Eval fitness and soft constraints
   Until fitness good enough, and
      A minimum #hosts evaluated
```

Task constraints

Soft

Hard

Task constraints

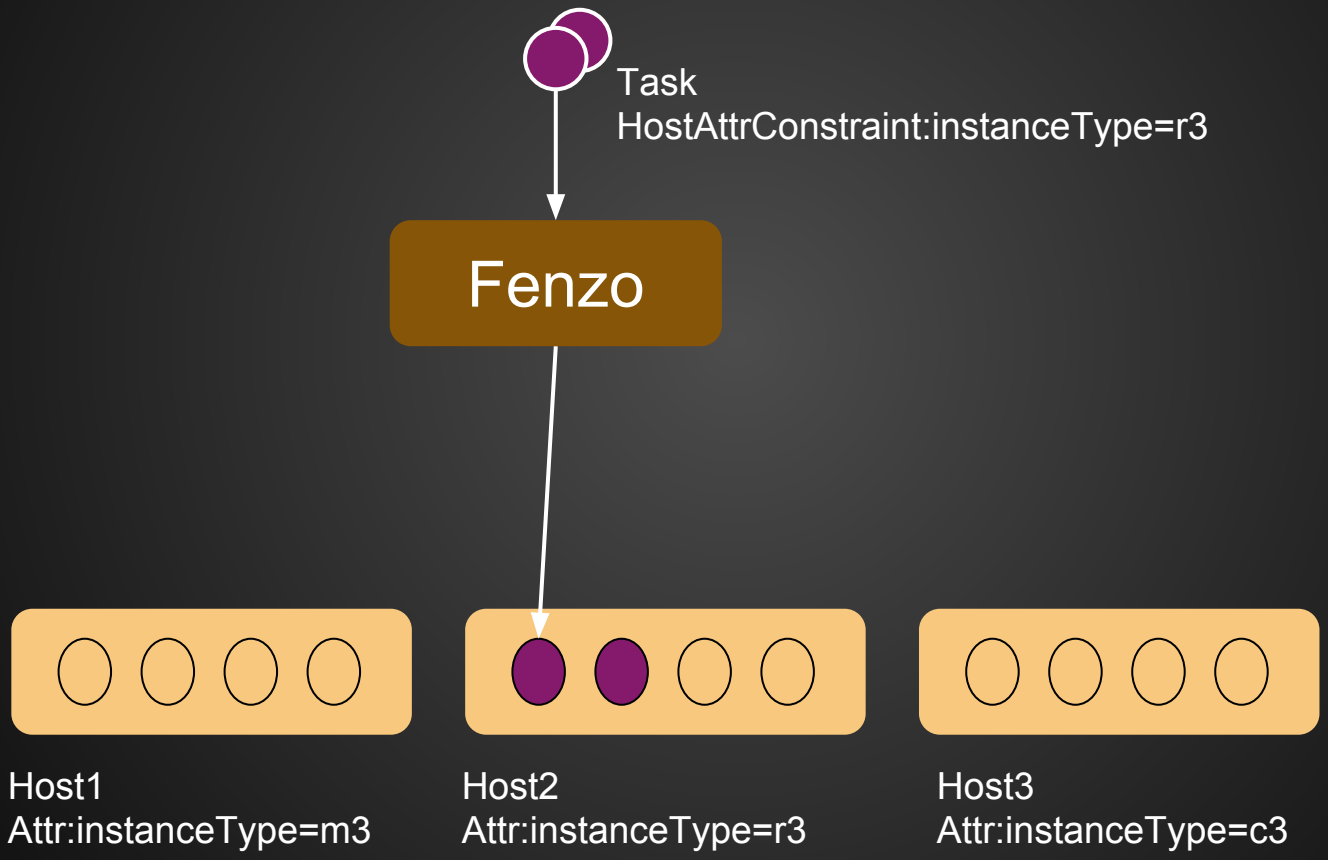
Soft

Hard

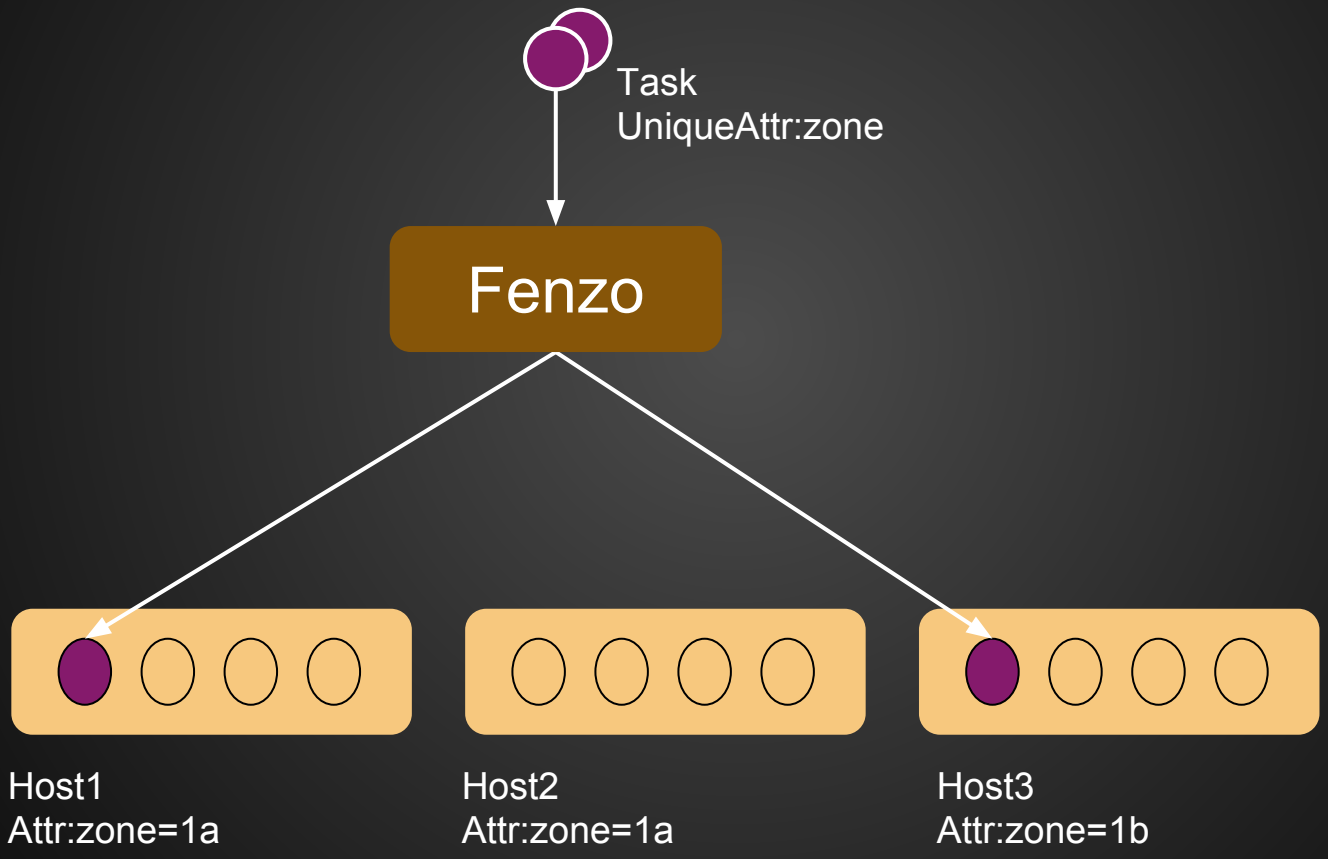
Extensible

Built-in Constraints

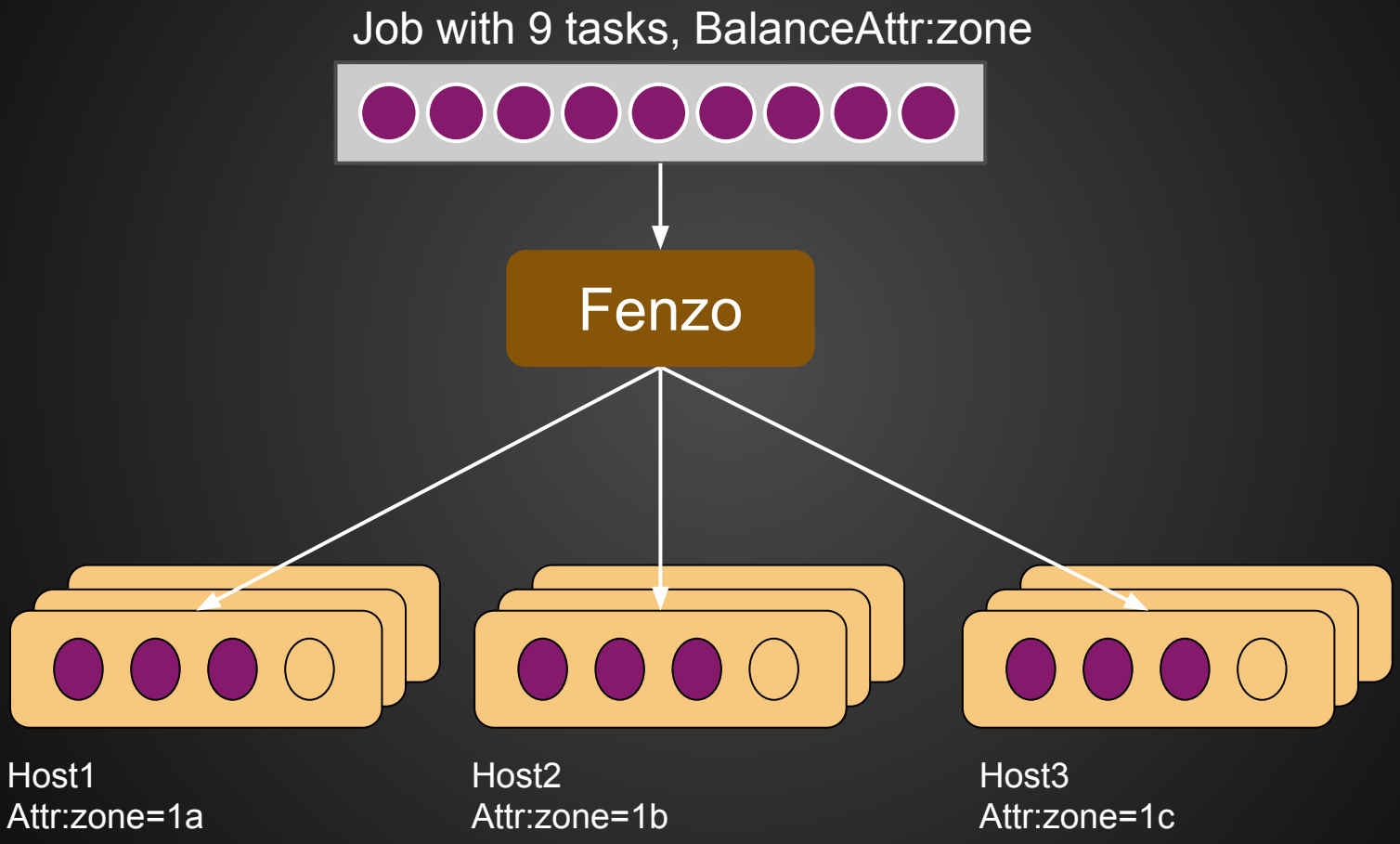
Host attribute value constraint



Unique host attribute constraint



Balance host attribute constraint



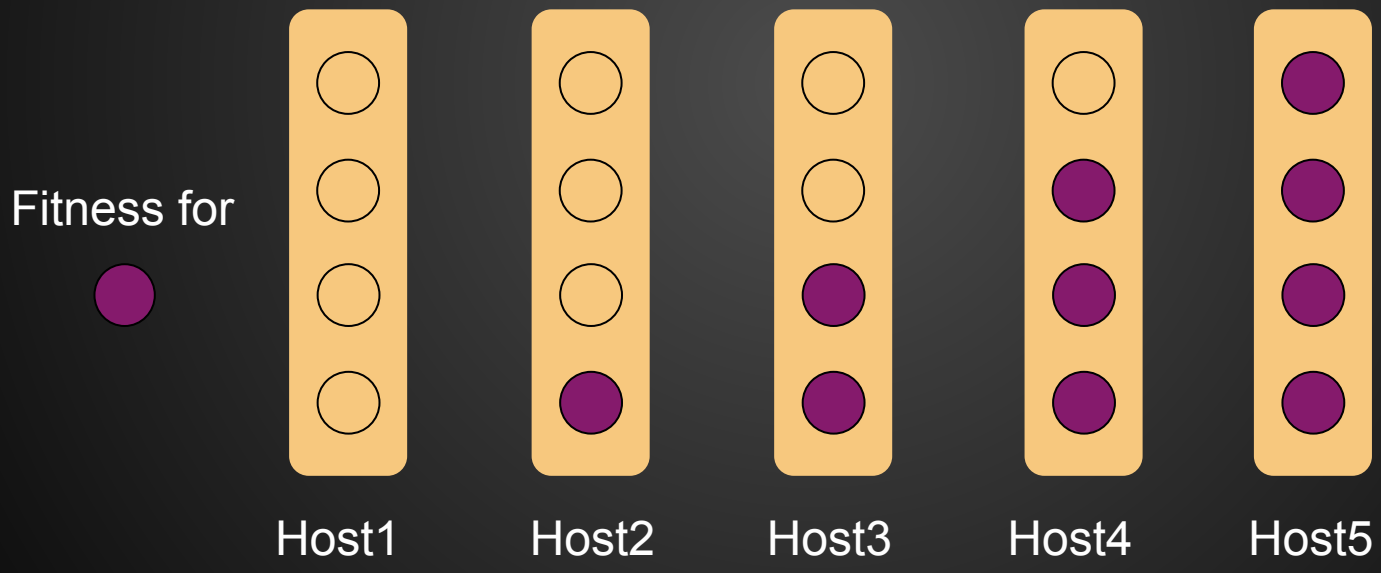
Fitness evaluation

Degree of fitness

Composable

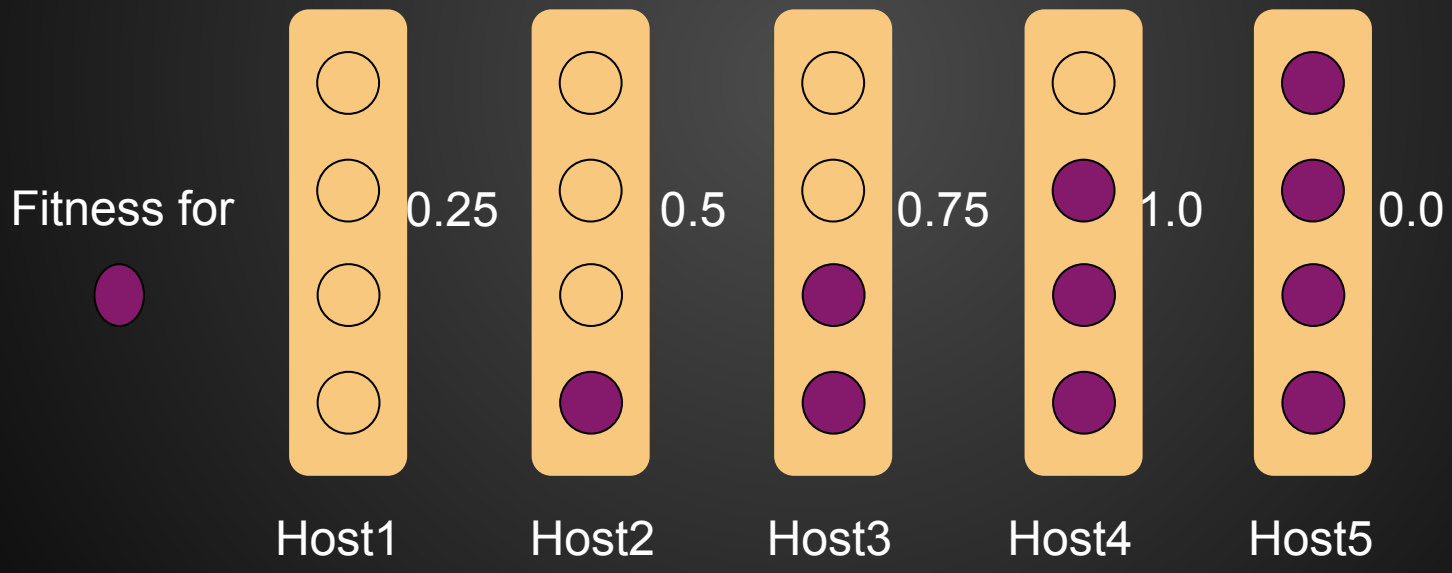
Bin packing fitness calculator

```
fitness = usedCPUs / totalCPUs
```



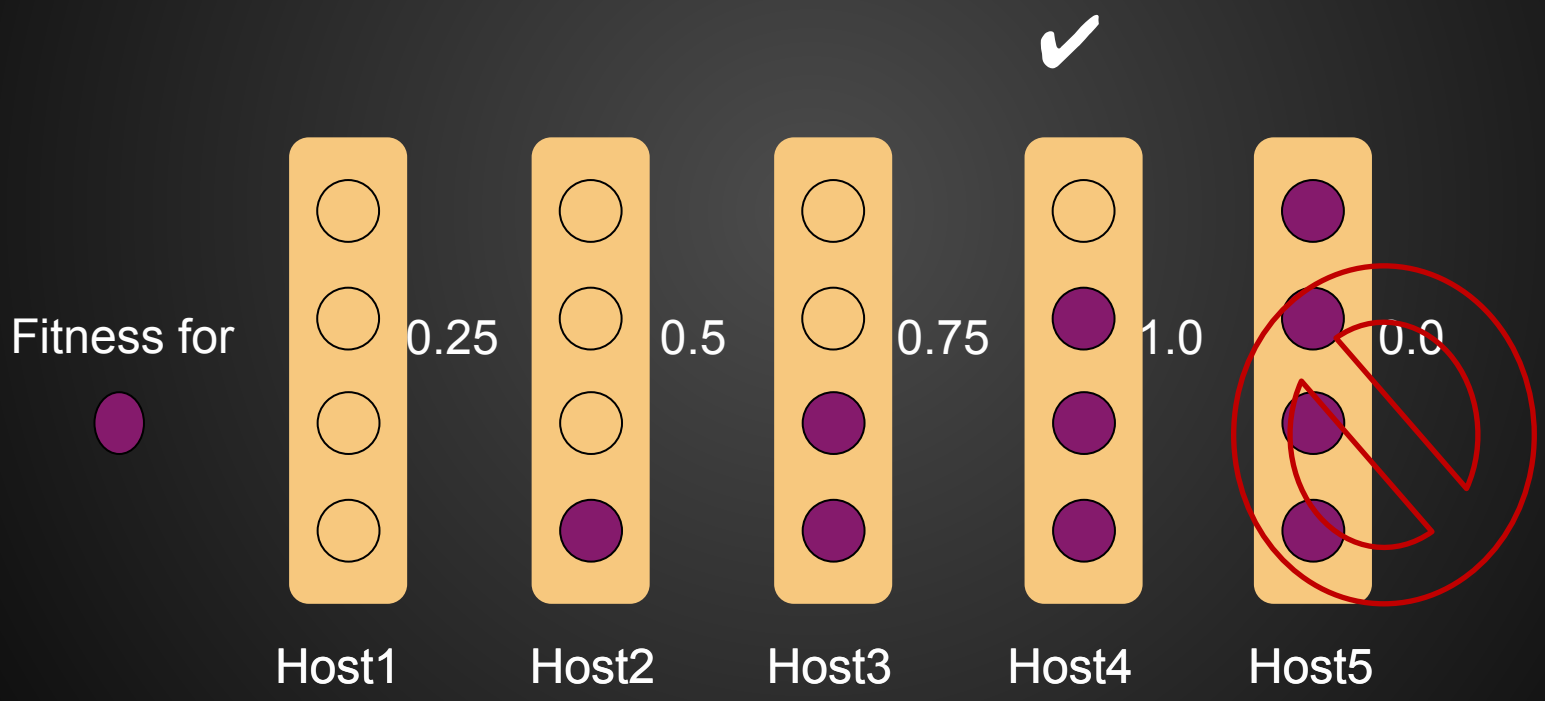
Bin packing fitness calculator

```
fitness = usedCPUs / totalCPUs
```



Bin packing fitness calculator

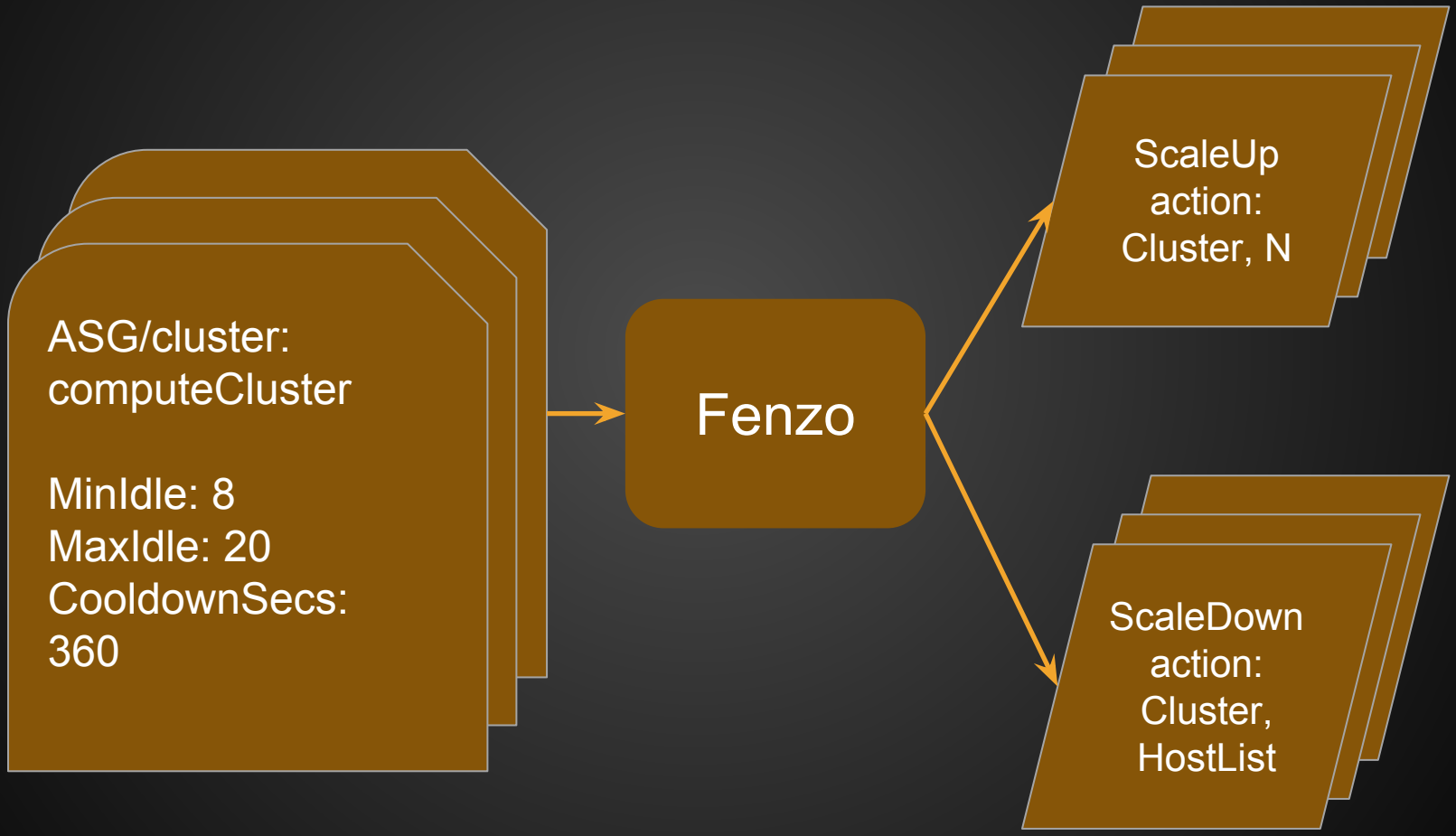
```
fitness = usedCPUs / totalCPUs
```



Composable fitness calculators

Fitness
    = (BinPackFitness * BinPackWeight +
        RuntimePackFitness * RuntimeWeight
) / 2.0

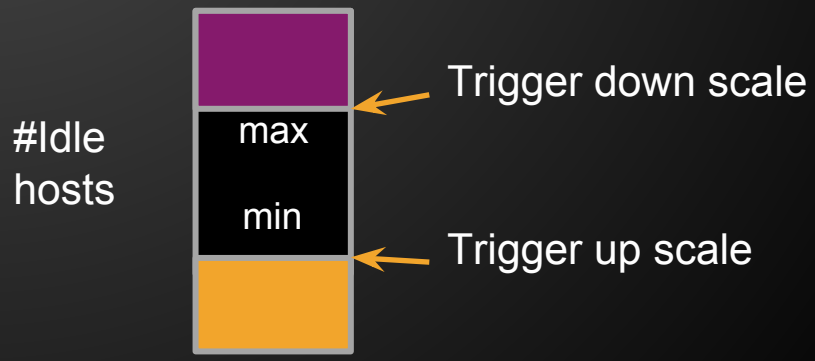
Cluster autoscaling in Fenzo



Rules based cluster autoscaling

- Set up rules per host attribute value
  - E.g., one autoscale rule per ASG/cluster, one cluster for network-intensive jobs, another for CPU/memory-intensive jobs
- Sample:

Cluster Name	Min Idle Count	Max Idle Count	Cooldown Secs
NetworkClstr	5	15	360
ComputeClstr	10	20	300

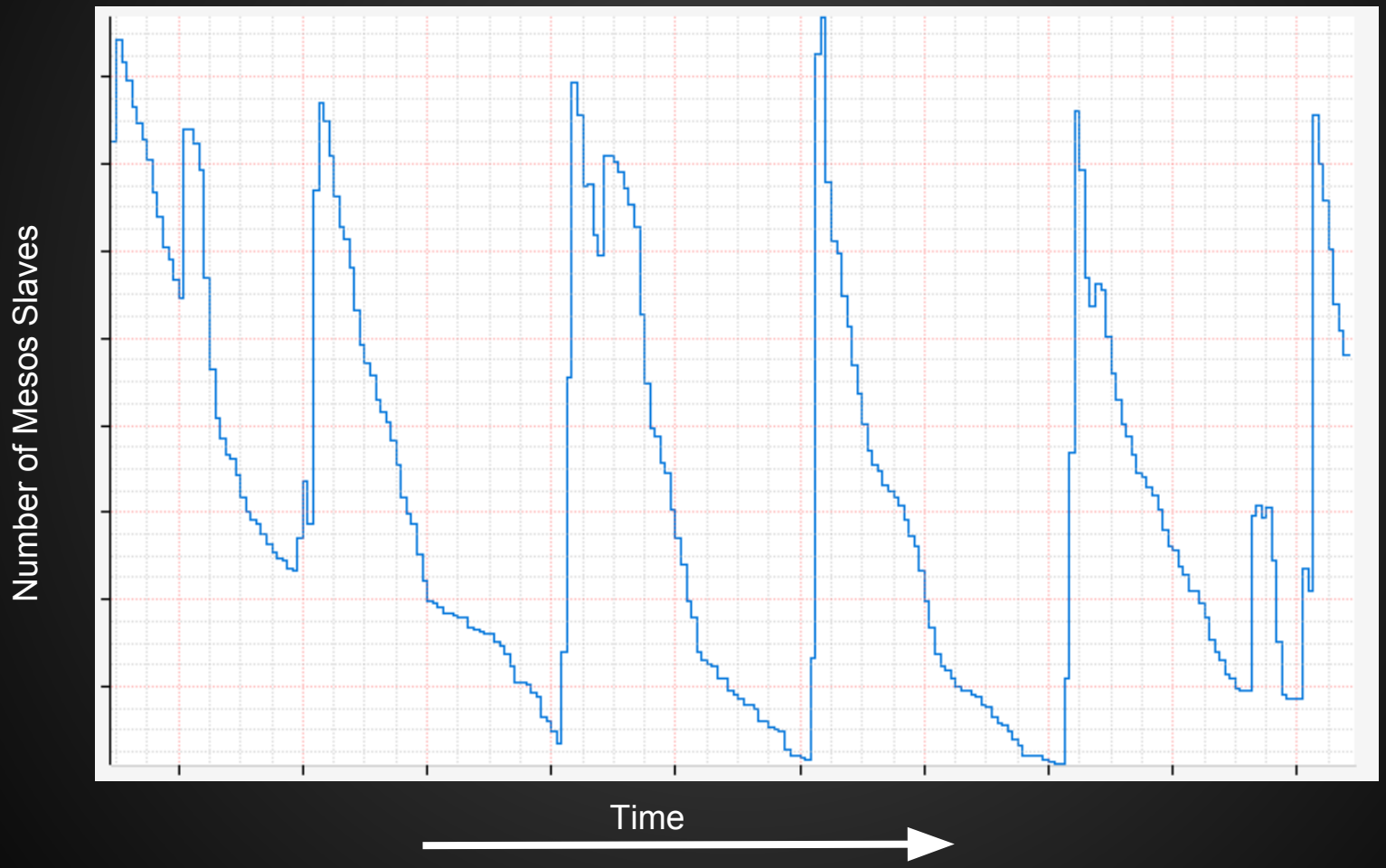


Shortfall analysis based autoscaling

- Rule-based scale up has a cool down period
  - What if there’s a surge of incoming requests?
- Pending requests trigger shortfall analysis
  - Scale up happens regardless of cool down period
  - Remembers which tasks have already been covered

Usage at Netflix

Cluster autoscaling



Scheduler run time (milliseconds)

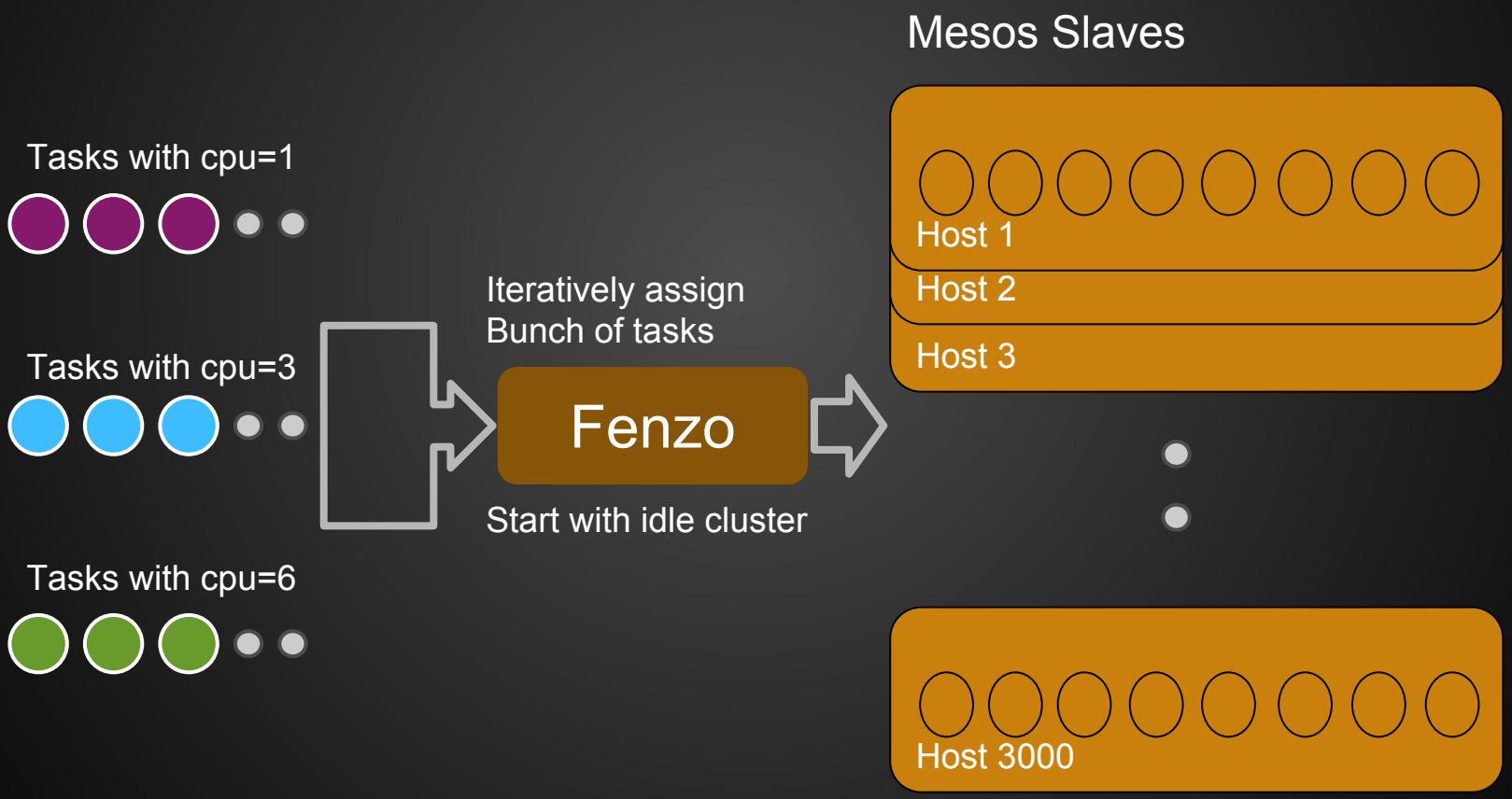
Scheduler run time in milliseconds (over a week)	
Average	2 mS
Maximum	38 mS

Note: times can vary depending on # of tasks, # and types of constraints, and # of hosts

Experimenting with Fenzo

Note: Experiments can be run without requiring a physical cluster

A bin packing experiment

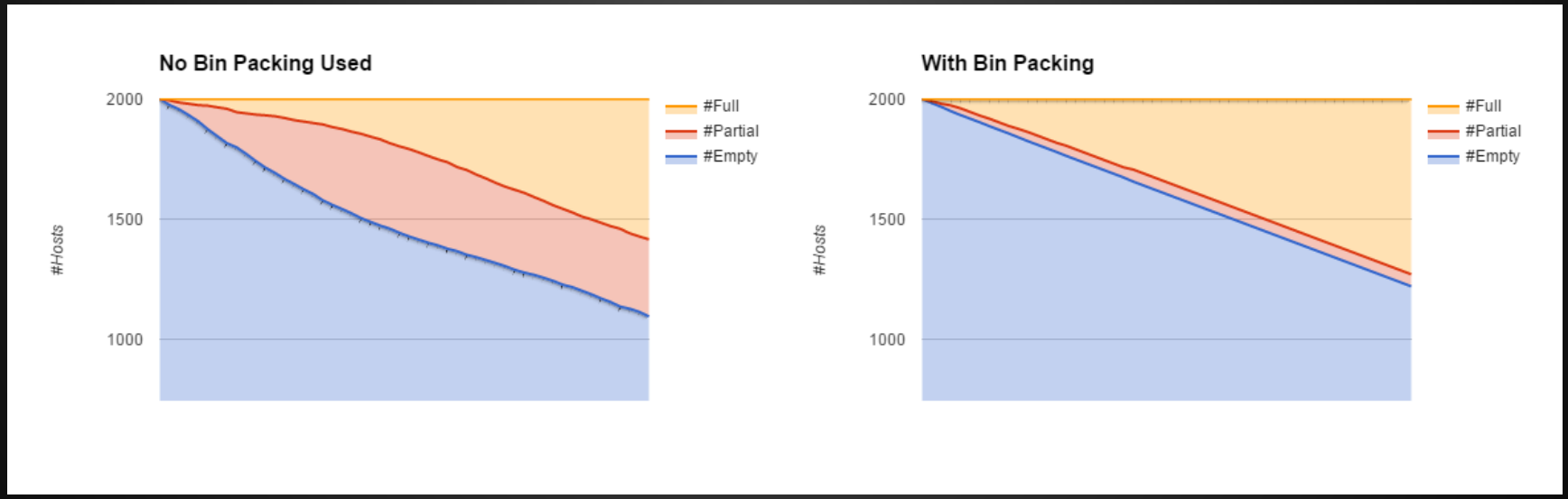


Bin packing sample results

Bin pack tasks using Fenzo's built-in CPU bin packer

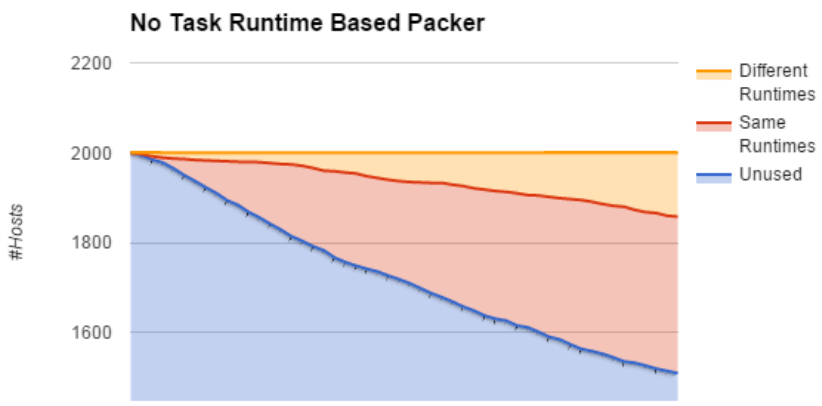
Bin packing sample results

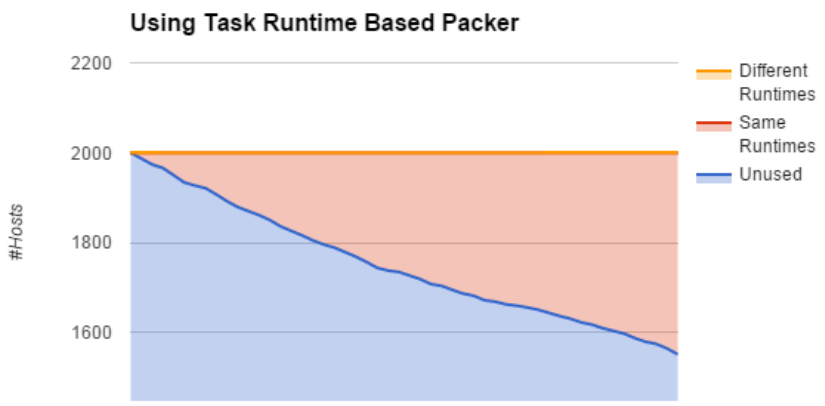
Bin pack tasks using Fenzo's built-in CPU bin packer



Task runtime bin packing sample

Bin pack tasks based on custom fitness calculator to pack short vs. long run time jobs separately





Scheduler speed experiment

Hosts: 8-CPUs each
Task mix: 20% running 1-CPU jobs, 40% running 4-CPU, and 40% running 6-CPU jobs
Goal: starting from an empty cluster, assign tasks to fill all hosts
Scheduling strategy: CPU bin packing

# of hosts	# of tasks to assign each time	Avg time	Avg time per task	Min time	Max time	Total time

Scheduler speed experiment

Hosts: 8-CPUs each
Task mix: 20% running 1-CPU jobs, 40% running 4-CPU, and 40% running 6-CPU jobs
Goal: starting from an empty cluster, assign tasks to fill all hosts
Scheduling strategy: CPU bin packing

# of hosts	# of tasks to assign each time	Avg time	Avg time per task	Min time	Max time	Total time
1,000	1	3 mS	3 mS	1 mS	188 mS	9 s
1,000	200	40 mS	0.2 mS	17 mS	100 mS	0.5 s

Scheduler speed experiment

Hosts: 8-CPUs each
Task mix: 20% running 1-CPU jobs, 40% running 4-CPU, and 40% running 6-CPU jobs
Goal: starting from an empty cluster, assign tasks to fill all hosts
Scheduling strategy: CPU bin packing

# of hosts	# of tasks to assign each time	Avg time	Avg time per task	Min time	Max time	Total time
1,000	1	3 mS	3 mS	1 mS	188 mS	9 s
1,000	200	40 mS	0.2 mS	17 mS	100 mS	0.5 s
10,000	1	29 mS	29 mS	10 mS	240 mS	870 s
10,000	200	132 mS	0.66 mS	22 mS	434 mS	19 s

Accessing Fenzo

Code at
https://github.com/Netflix/Fenzo

Wiki at
https://github.com/Netflix/Fenzo/wiki

Future directions

- Task management SLAs
- Support for newer Mesos features
- Collaboration

To summarize...

Fenzo: scheduling library for frameworks

Heterogeneous task requests

Heterogeneous resources

Plugins for Constraints, Fitness

Autoscaling of cluster

Visibility of scheduler actions

High speed

Fenzo is now available in Netflix OSS suite at
https://github.com/Netflix/Fenzo

Questions?

Heterogeneous Resource Scheduling Using Apache Mesos for Cloud Native Frameworks

Sharma Podila
spodila @ netflix . com

 @podila