Twine: A Unified Cluster Management System for Shared Infrastructure

Ashvin Goel

Electrical and Computer Engineering
University of Toronto

ECE1724

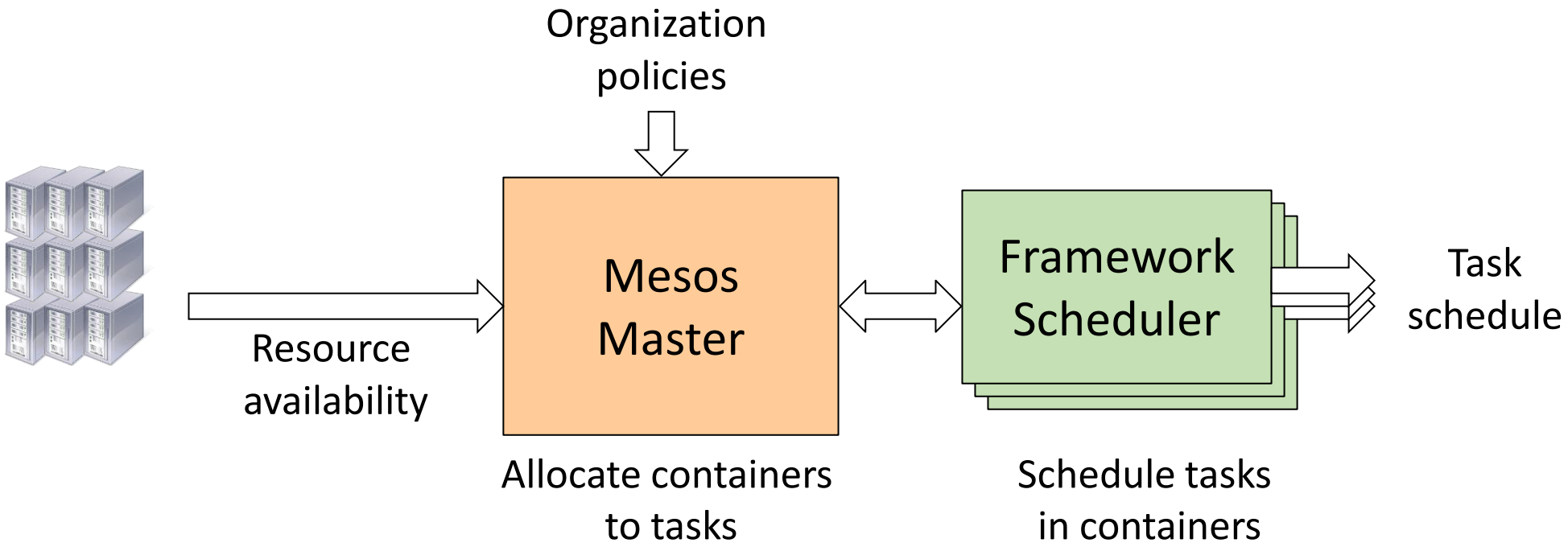
Authors: Facebook Infrastructure

Chunqiang Tang, Kenny Yu, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long Cheng, Ben Christensen, Alex Gartrell, Maxim Khutornenko, Sachin Kulkarni, Marcin Pawlowski, Tuomas Pelkonen, Andre Rodrigues, Rounak Tibrewal, Vaishnavi Venkatesan, and Peter Zhang

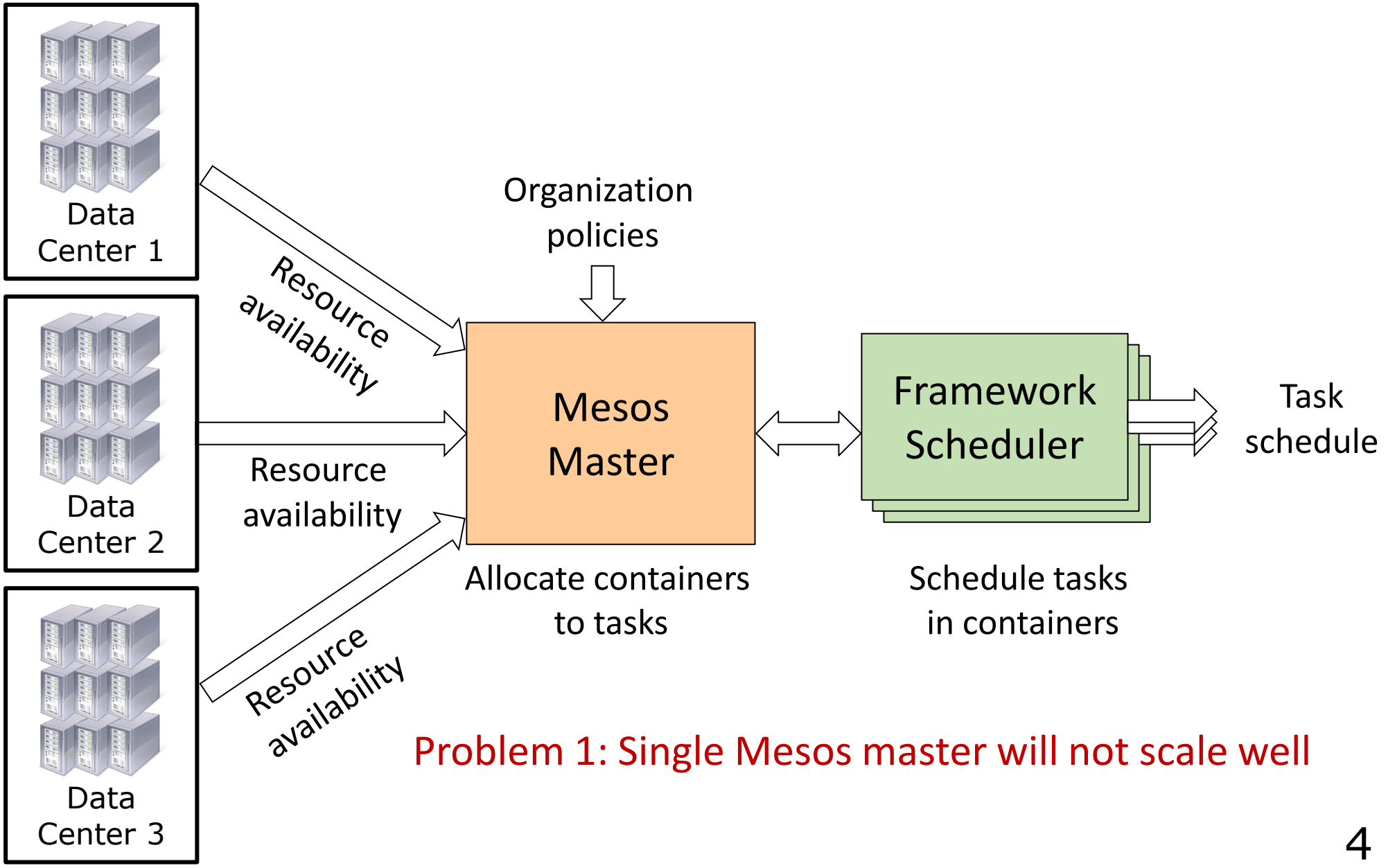
Motivation

- A region consists of a few closely-located data centers
  - Contains ~1 million machines!
- Previously, siloed pools of customized machines dedicated to individual workloads
- Need a scalable, fault tolerant shared resource allocator
  - Manage all machines in a region (across data centers)
  - Allow running workloads across data centers
  - Handle node failures, software upgrades

We Looked at Mesos

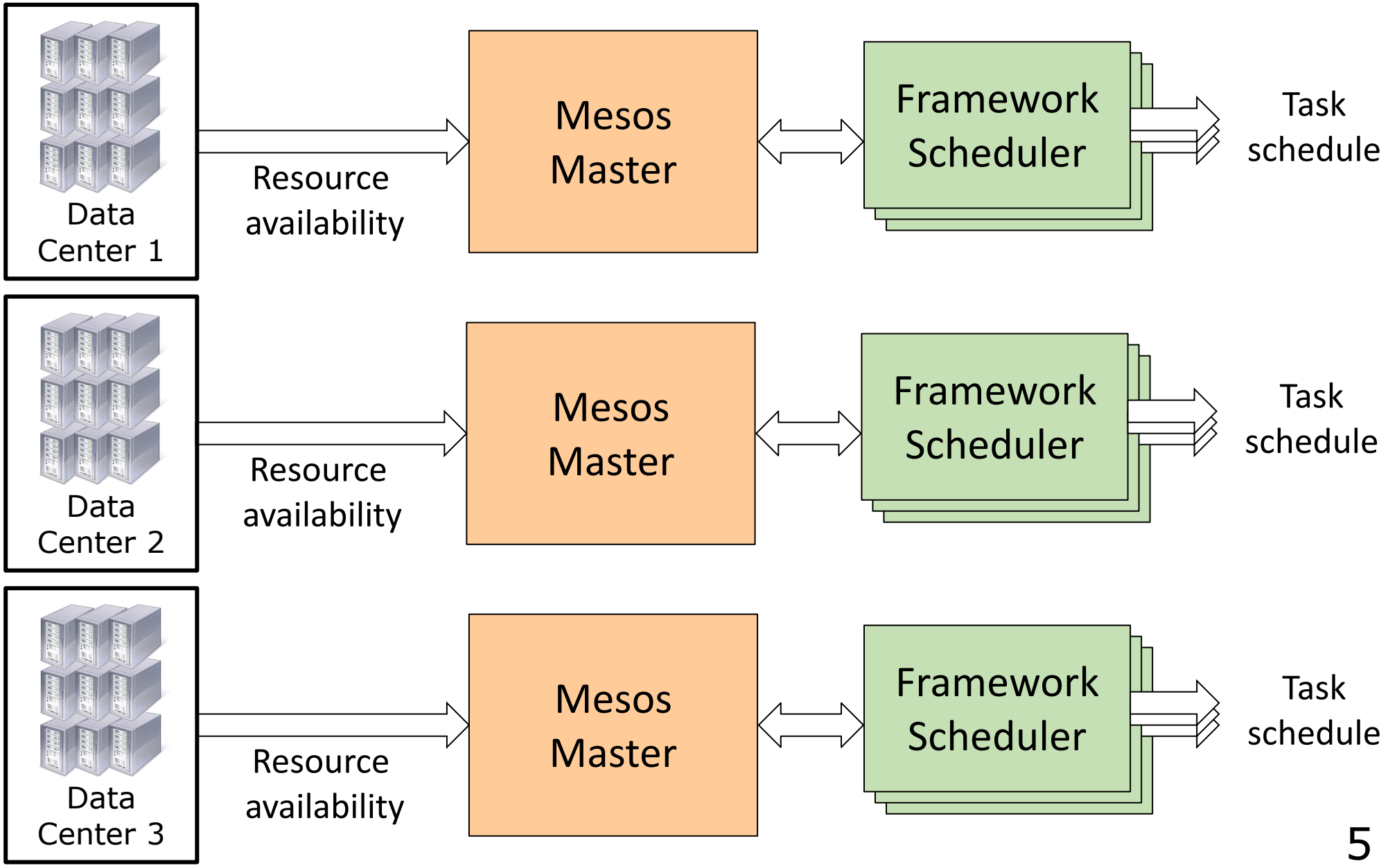


Why Would Mesos Not Work Well?

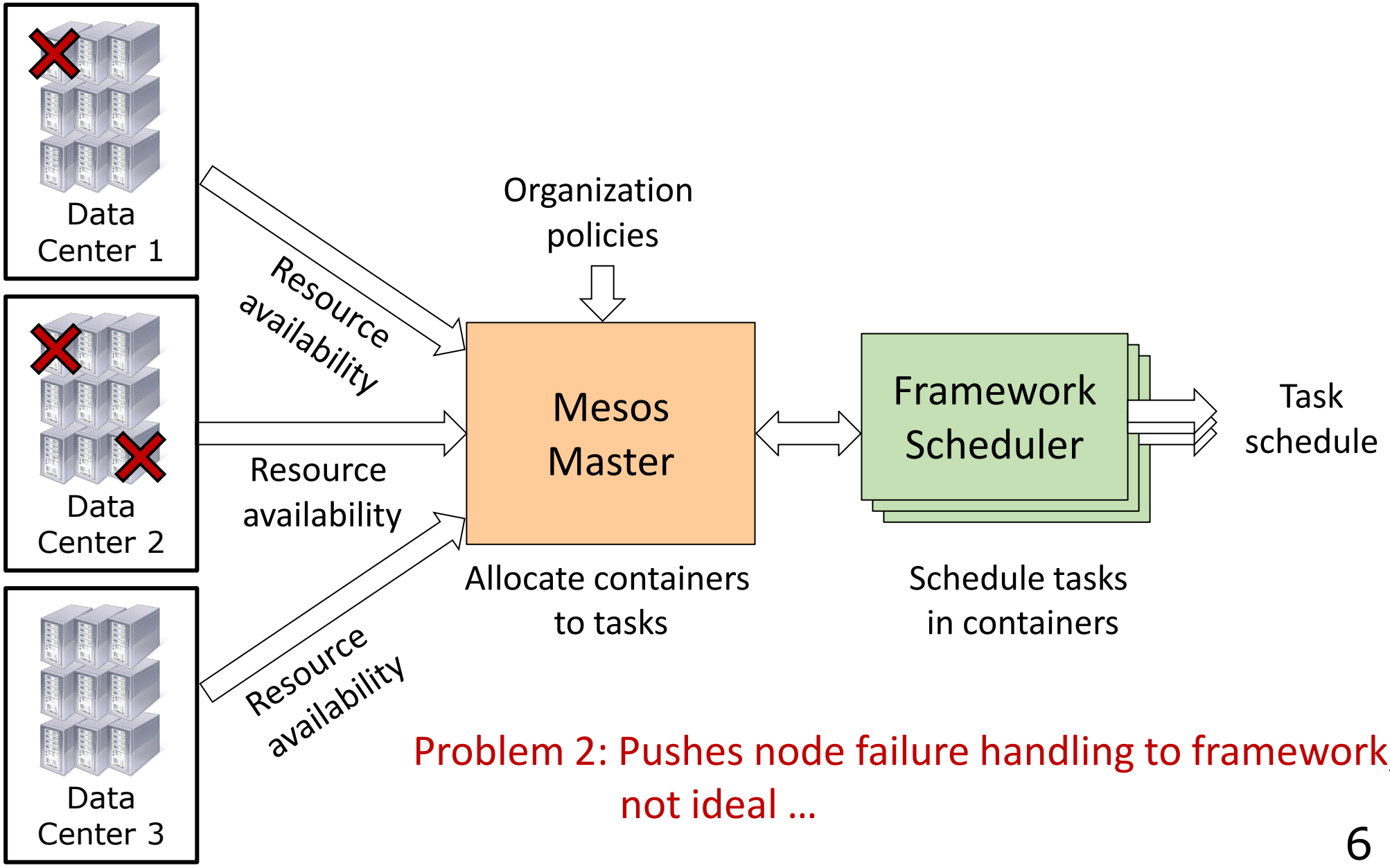


Problem 1: Single Mesos master will not scale well

What about This Approach?



How Does Mesos Handle Node Failures?

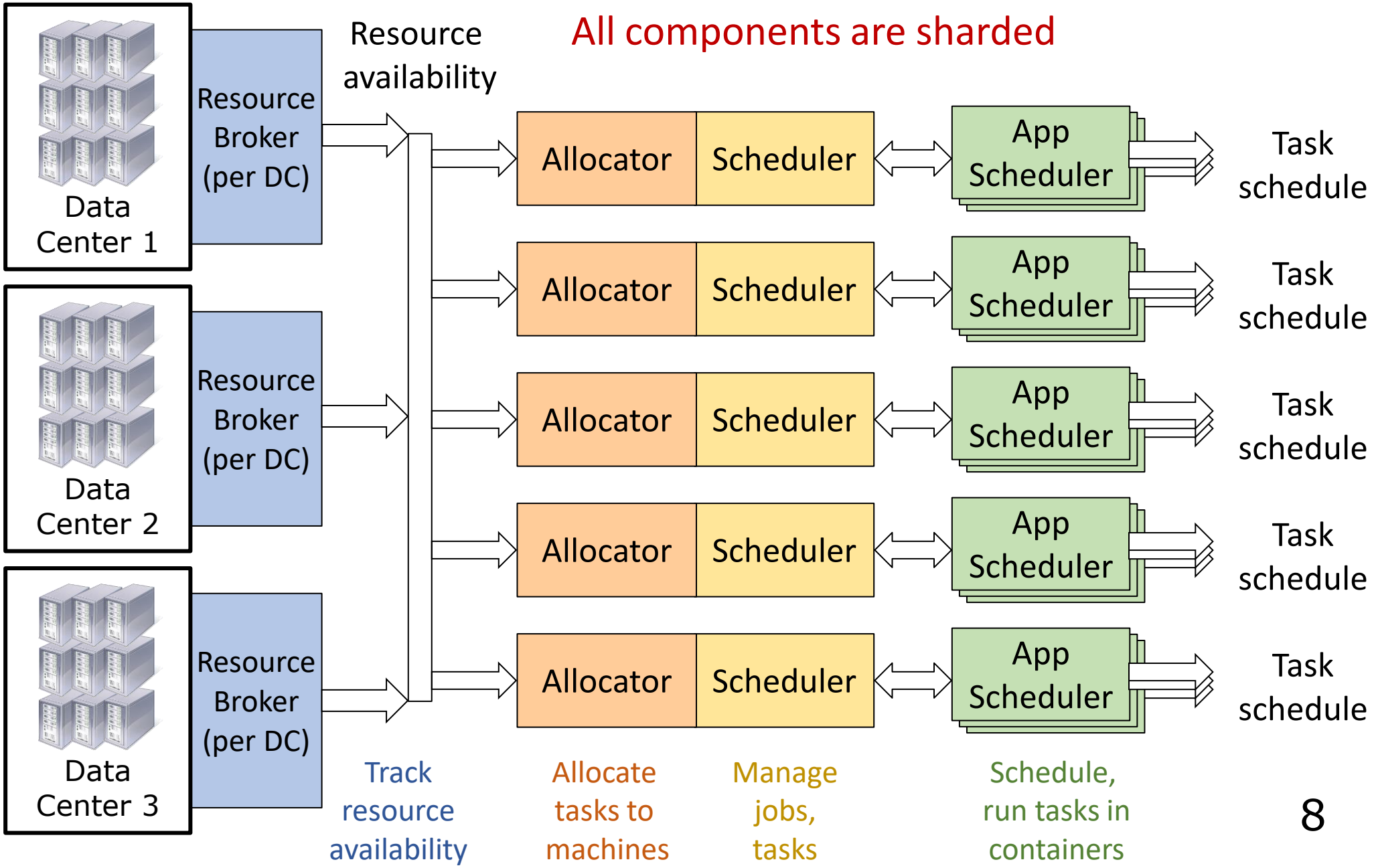


Problem 2: Pushes node failure handling to framework, not ideal …

Million Node Resource Management

- Scaling resource management is a clearly a challenge
- But there are many other issues
  - Hardware failures
    - Nodes may fail; racks may fail due to power failures
  - Maintenance operations
    - Node hardware needs upgrades
    - Node software, e.g., kernel, libraries, etc., need updates
  - Software management
    - Application software needs bug fixes, new software releases
    - Custom kernel installations
  - Job resizing, relocation
    - Job tasks need to be added/removed, moved for data locality

Twine Architecture In a Nutshell

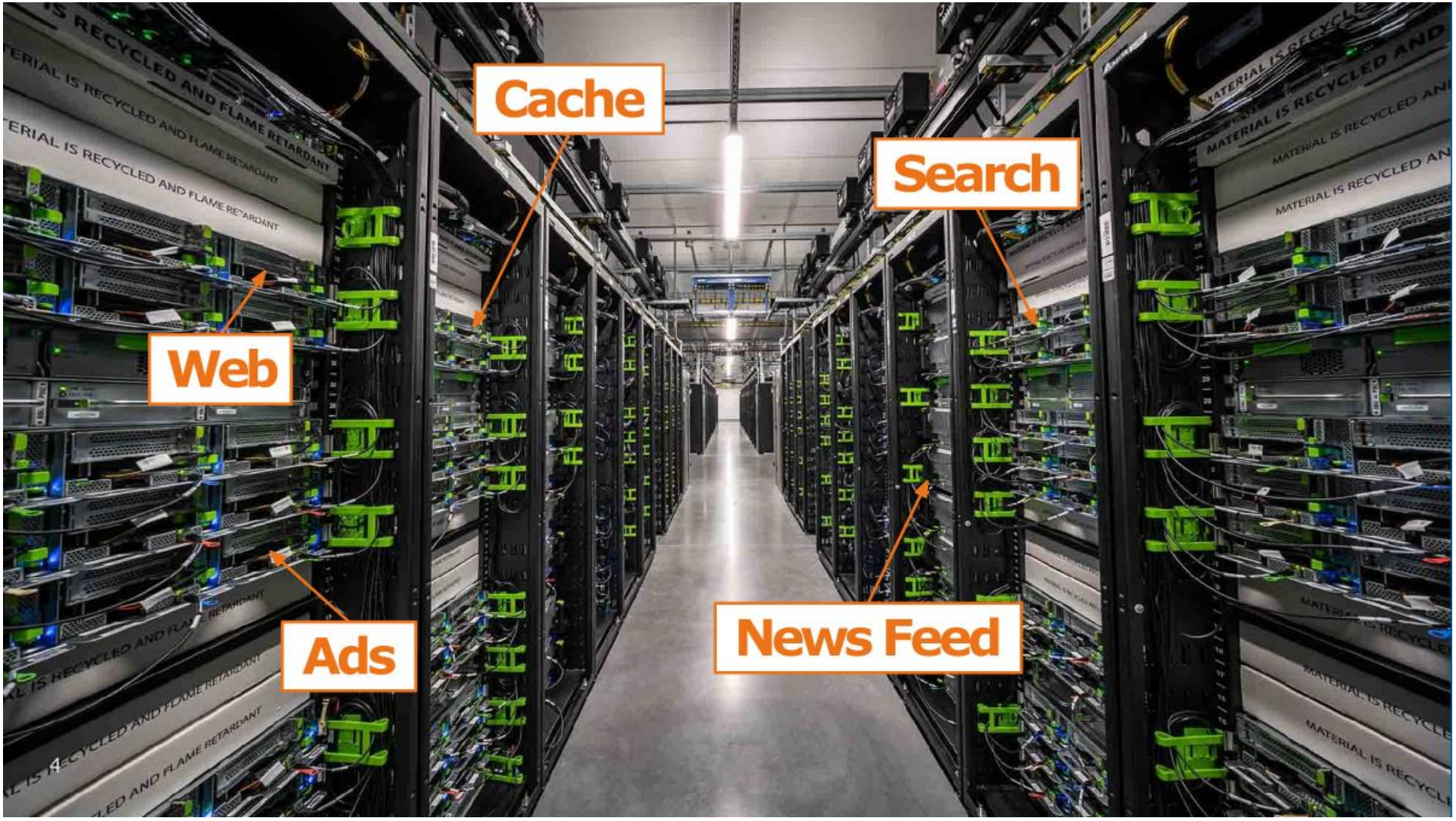


Twine Design Principles

- All stateful components such as the scheduler, allocator and resource brokers are sharded for scalability
  - E.g., with 20 schedulers, 1M machines, each scheduler manages ~50K machines
- All components are replicated for fault tolerance
  - Replicas located in different data centers, elect leader that serves requests
- Tasks keep running
  - Even if all components fail, existing tasks continue to run
  - But new jobs cannot be created, and existing tasks cannot be updated until Twine recovers

The next set of slides are a subset from the original Twine talk


Data center geographic regions
Papillion, NE
Prineville, OR
Eagle Mountain, UT
Los Lunas, NM
Fort Worth TX
Altoona, IA
DeKalb, IL
New Albany, OH
Gallatin, TN
Henrico, VA
Forest City, NC
Newton County, GA
Huntsville, AL
Luleå, Sweden
Odense, Denmark
Clonee, Ireland
Singapore
New construction
Serving traffic
3


Cache
Search
Web
Ads
News Feed


Cluster management systems
help manage all of our
services and machines.
5

What design decisions did Twine make differently?

Decision 1	Decision 2	Decision 3
Dynamic machine partitioning	Customization in shared infrastructure	Small machines
over	over	over
static clusters	private pools	big machines

What design decisions did Twine make differently?

Decision 1

Dynamic machine partitioning

over

static clusters

What if we used Kubernetes?

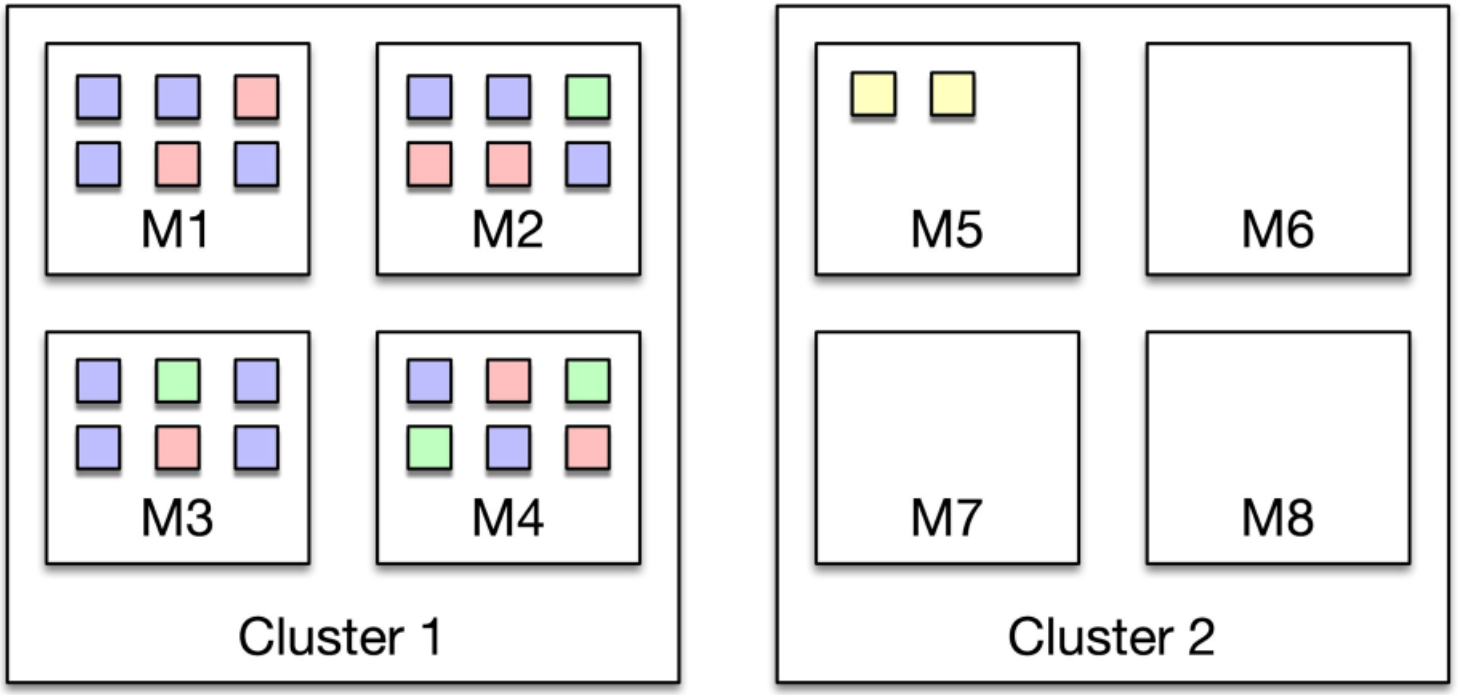
Kubernetes Documentation / Getting started / Best practices / Building large clusters

Building large clusters

Support

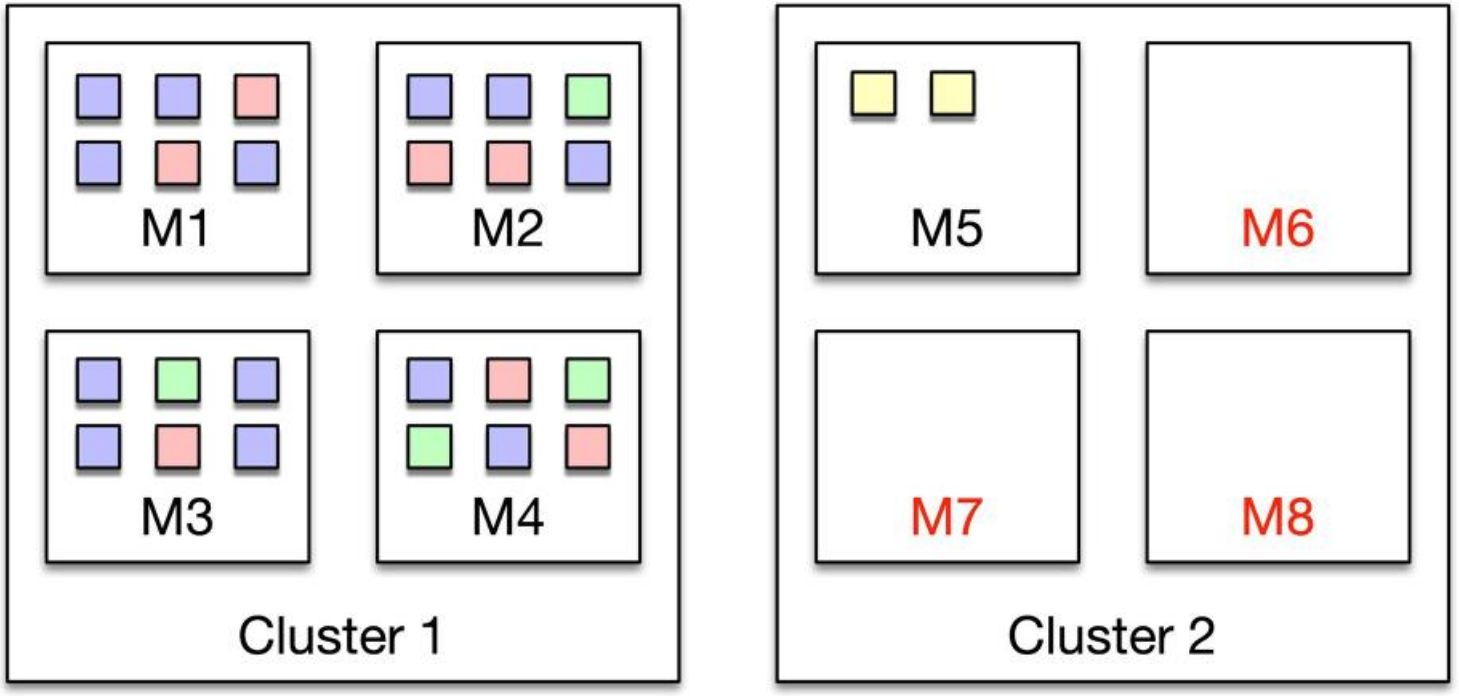
At v1.19, Kubernetes supports clusters with up to 5000 nodes. More specifically, we support configurations that meet *all* of the following criteria:

What if we used Kubernetes?



What if we used Kubernetes?

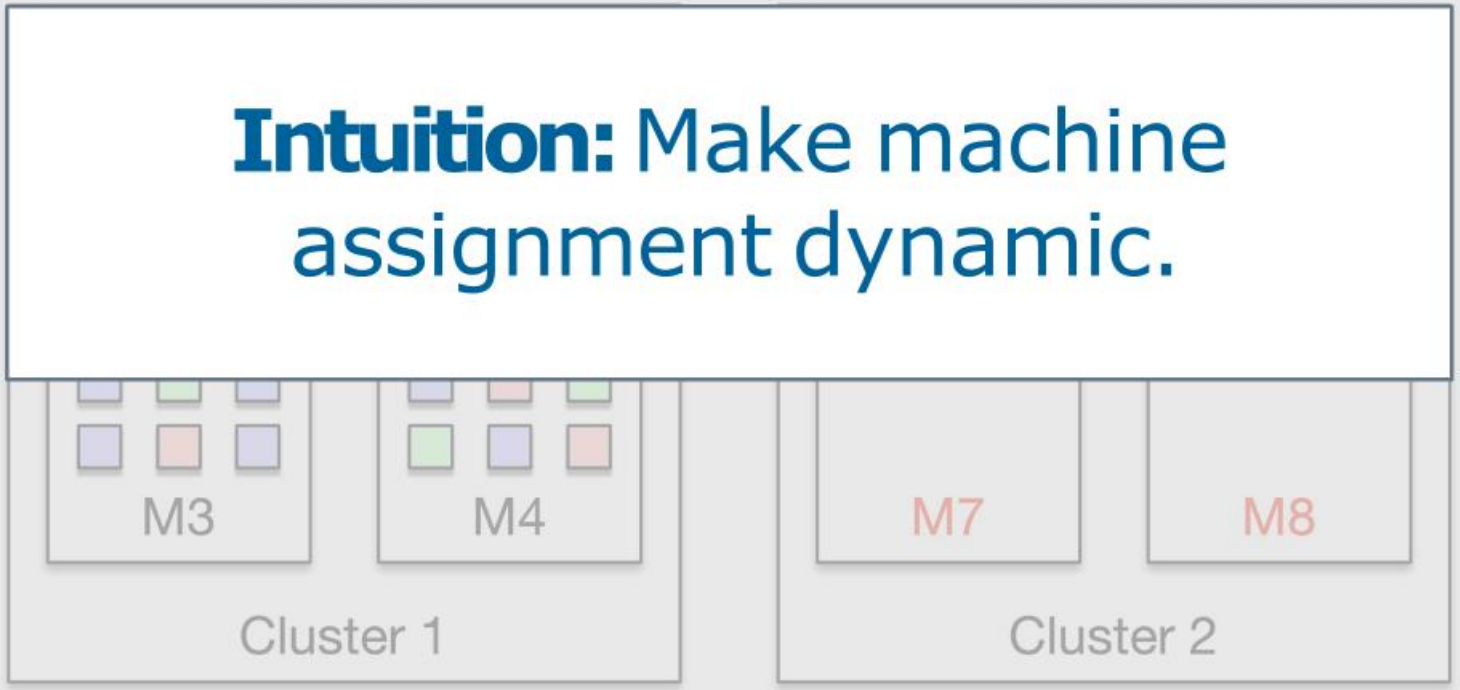
Stranded capacity: M6, M7, M8 are available, but jobs in Cluster 1 cannot use them



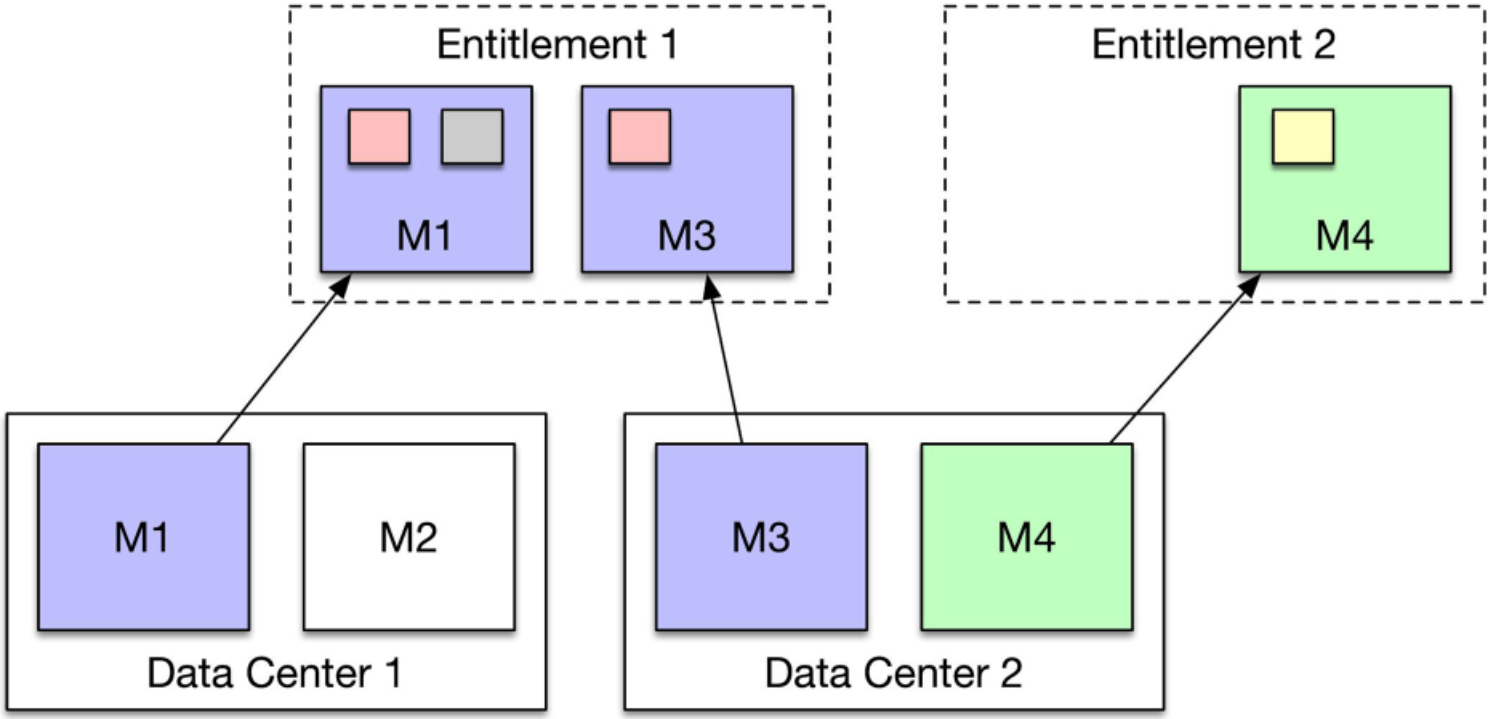
What if we used Kubernetes?

Stranded capacity: M6, M7, M8 are available, but jobs in Cluster 1 cannot use them

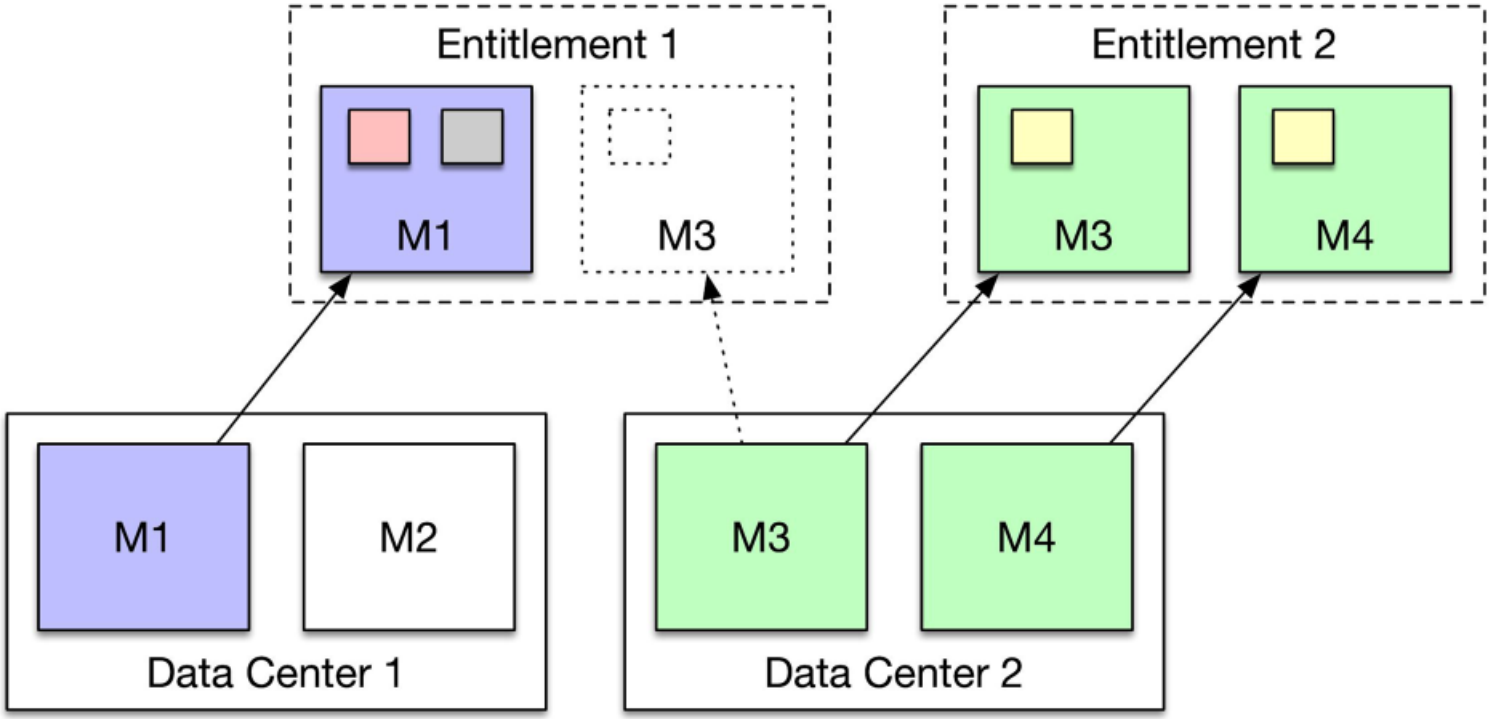
Intuition: Make machine assignment dynamic.



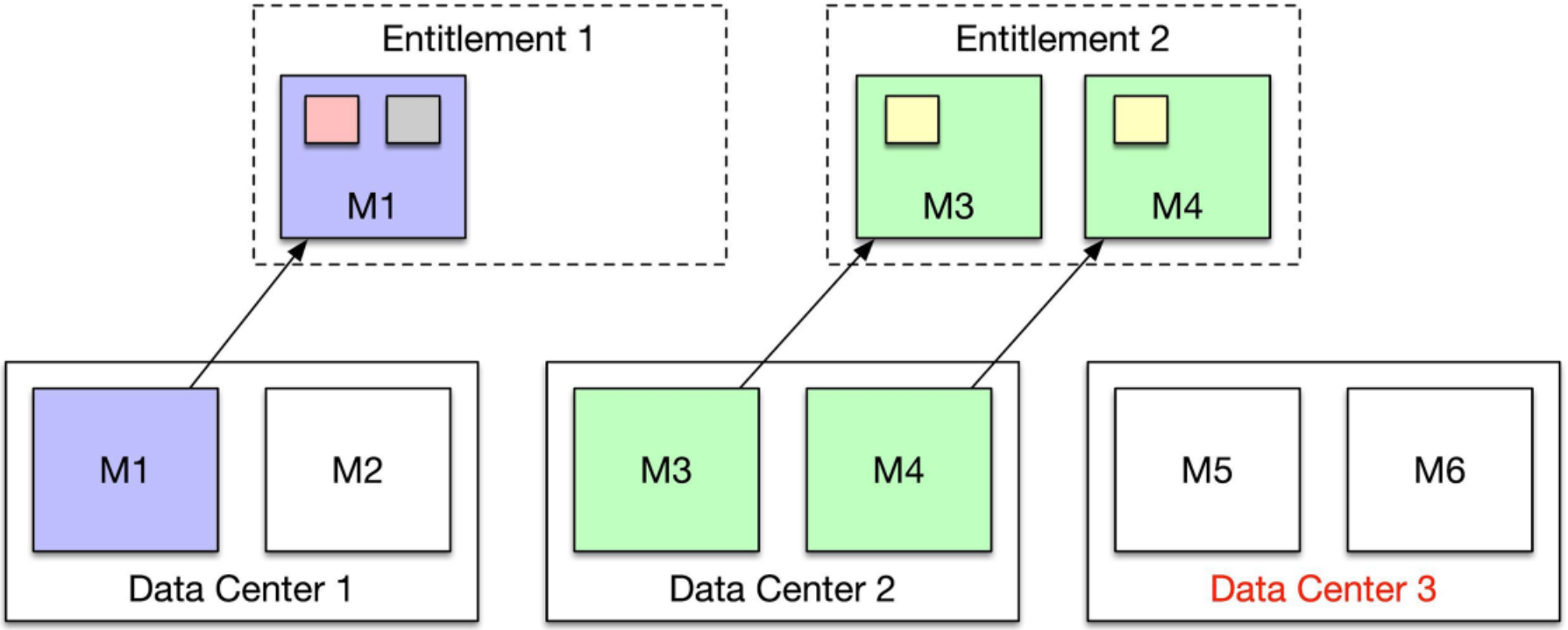
How does Twine avoid stranded capacity?



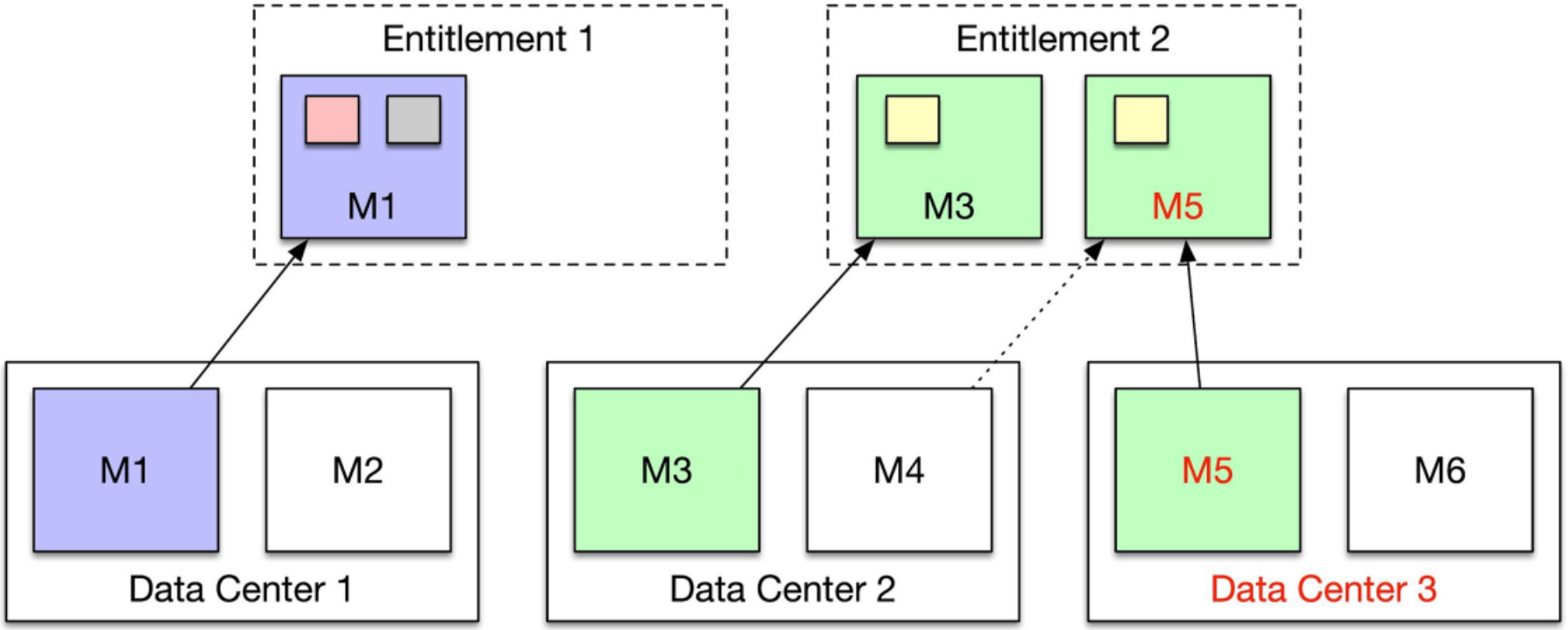
How does Twine avoid stranded capacity?



How does Twine perform fleet-wide optimization?

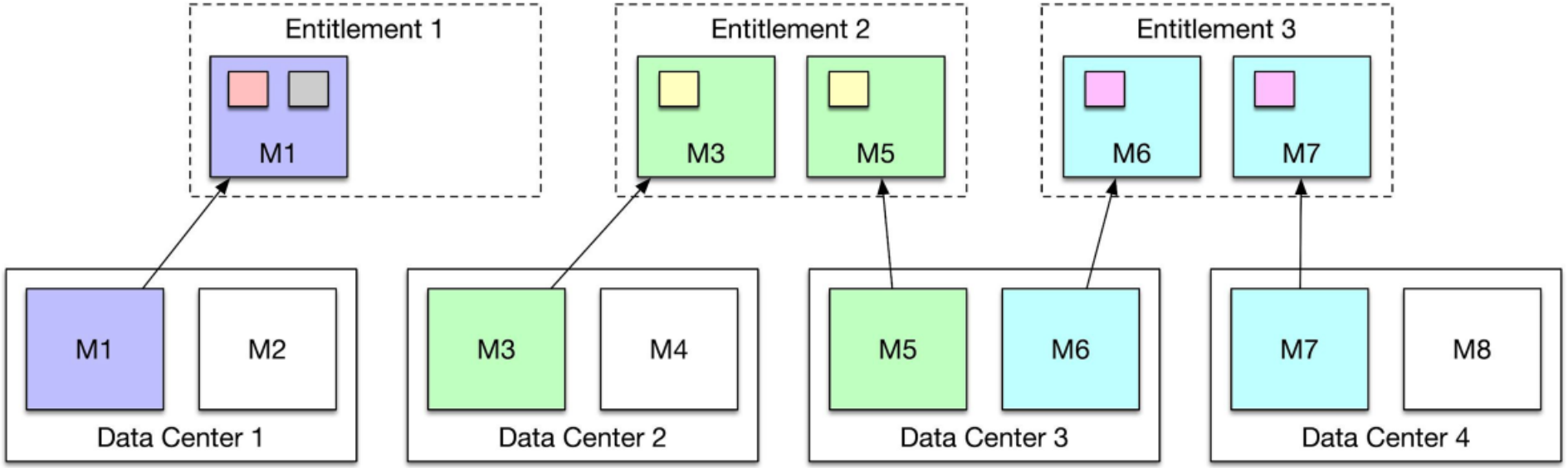


How does Twine perform fleet-wide optimization?

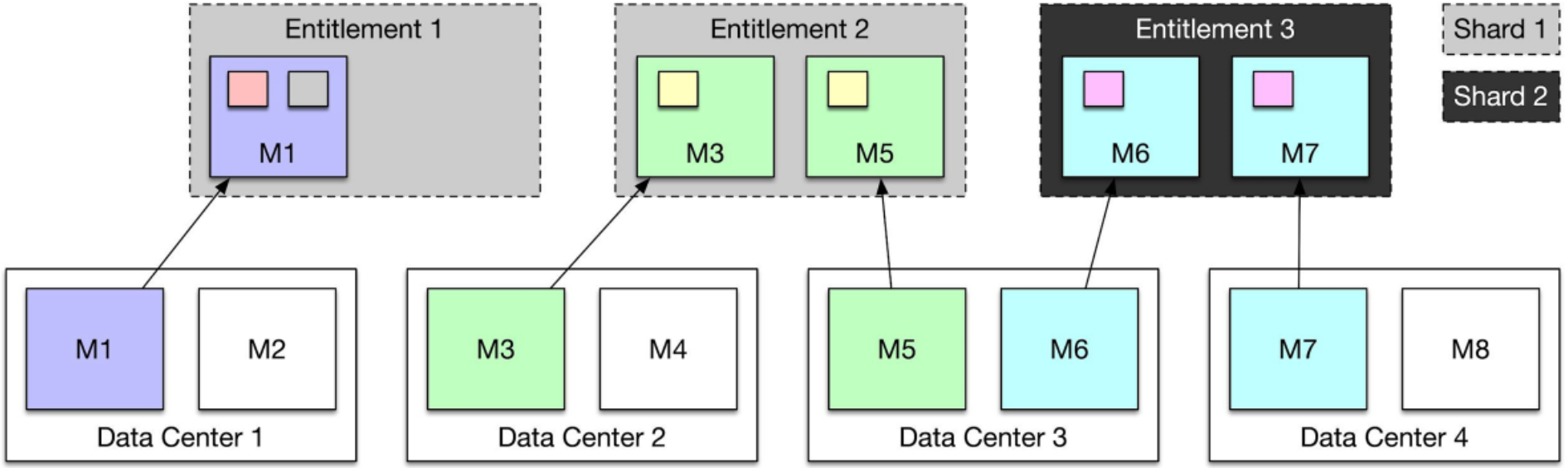


Use M5 in newly-constructed Data Center 3
to improve spread for fault tolerance

How does Twine perform fleet-wide optimization for an entire geographic region?

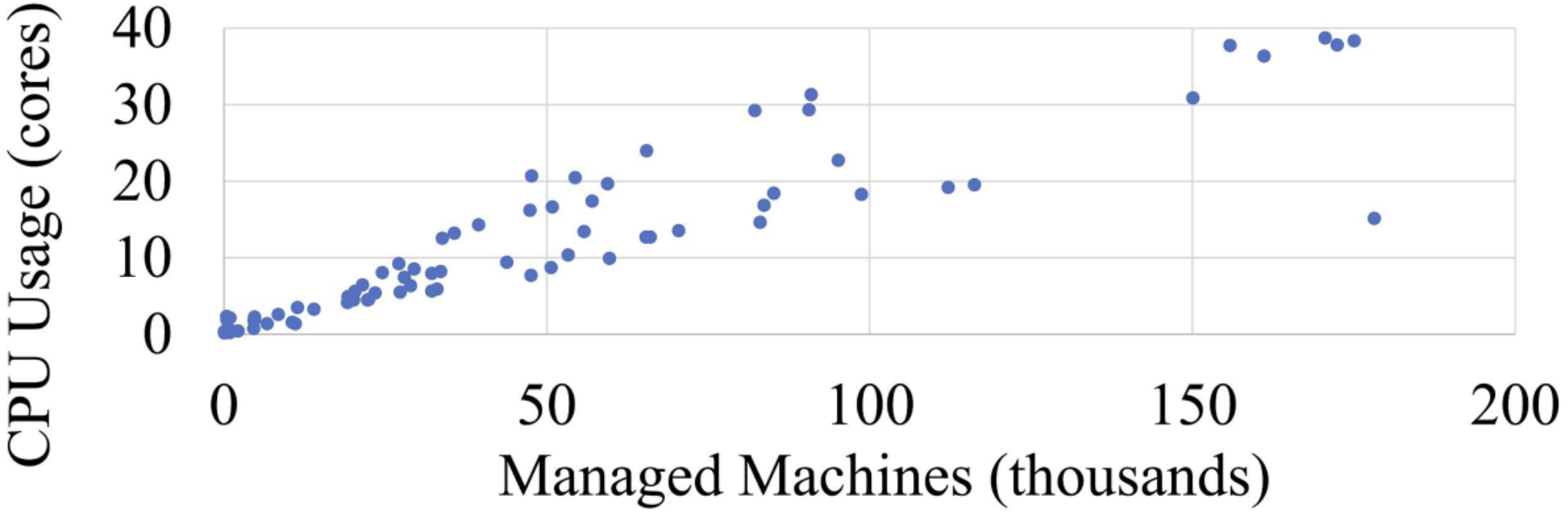


How does Twine perform fleet-wide optimization for an entire geographic region?



Shard Twine Scheduler by entitlements

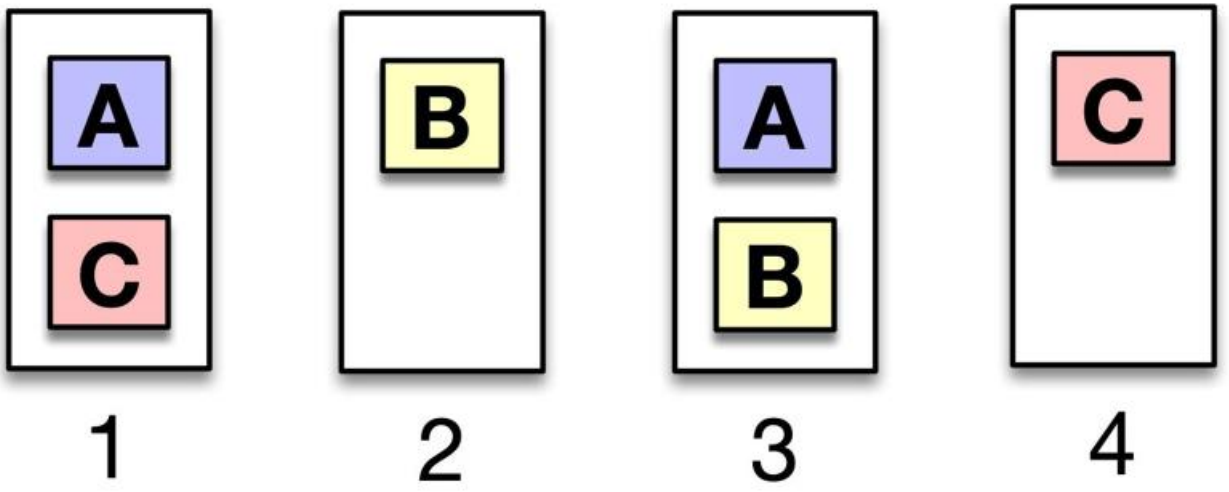
How well does the Twine scheduler scale?



What design decisions did Twine make differently?

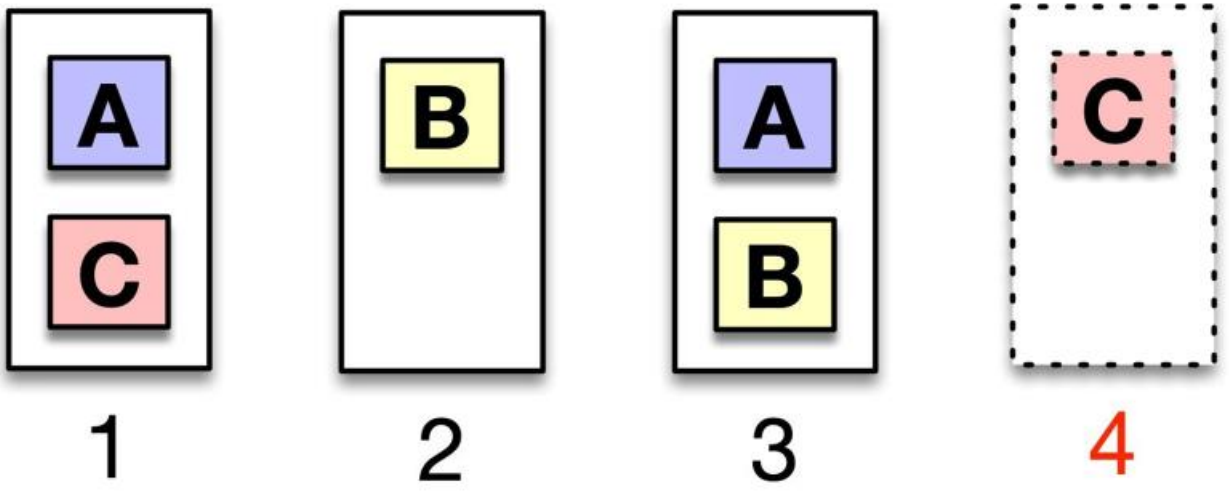
Decision 1	**Decision 2**	Decision 3
Dynamic machine partitioning	Customization in shared infrastructure	Small machines
over	over	over
static clusters	private pools	big machines

Challenge: Tasks are not homogenous



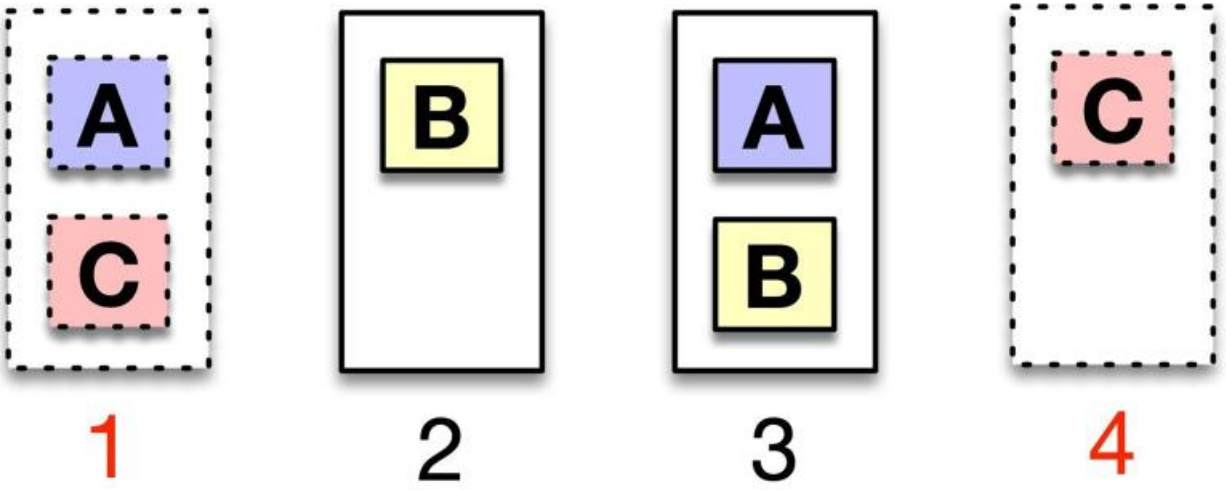
Need at least 1 replica
of each shard up
at all times

Challenge: Tasks are not homogenous



Task 4 becomes unavailable.

Challenge: Tasks are not homogenous



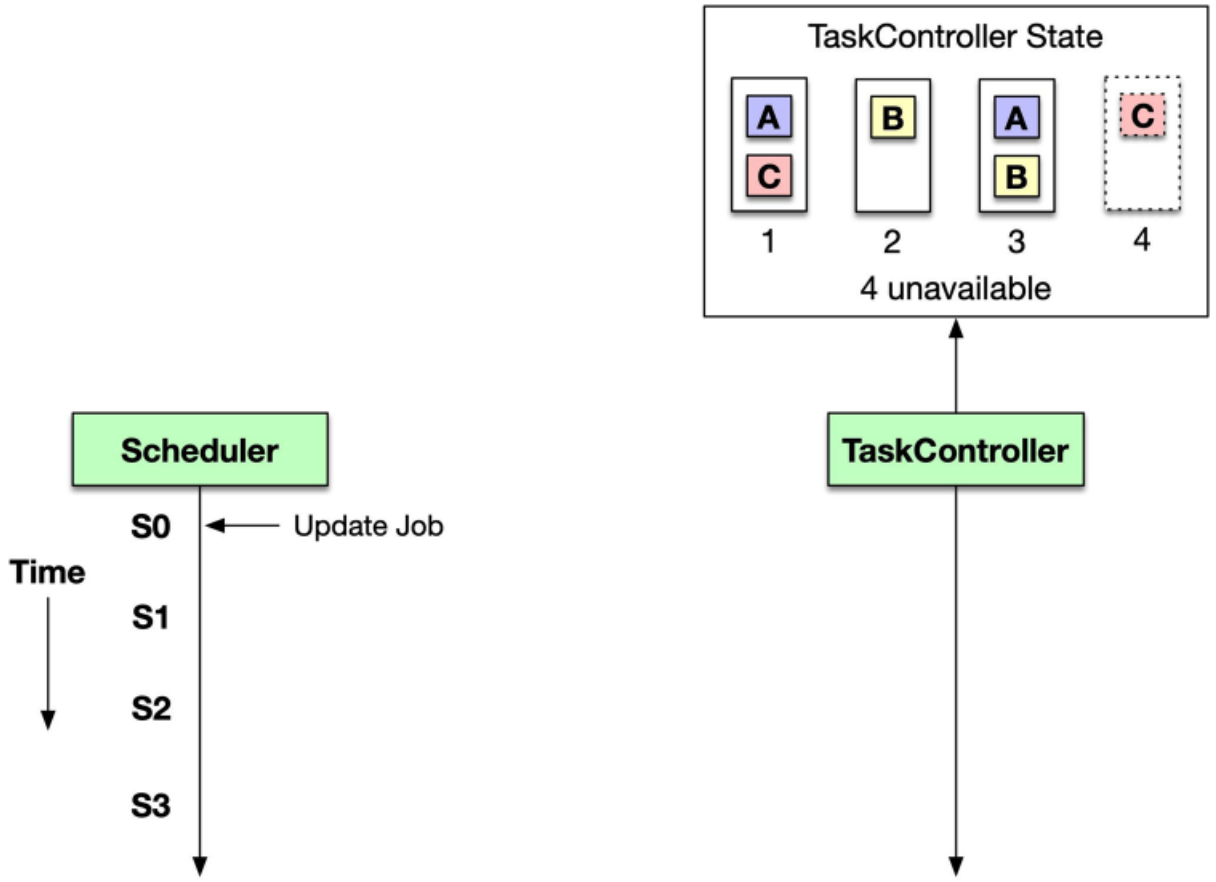
Task 1 restarted for
software release.
Shard C unavailable!

Challenge: Tasks are not homogenous

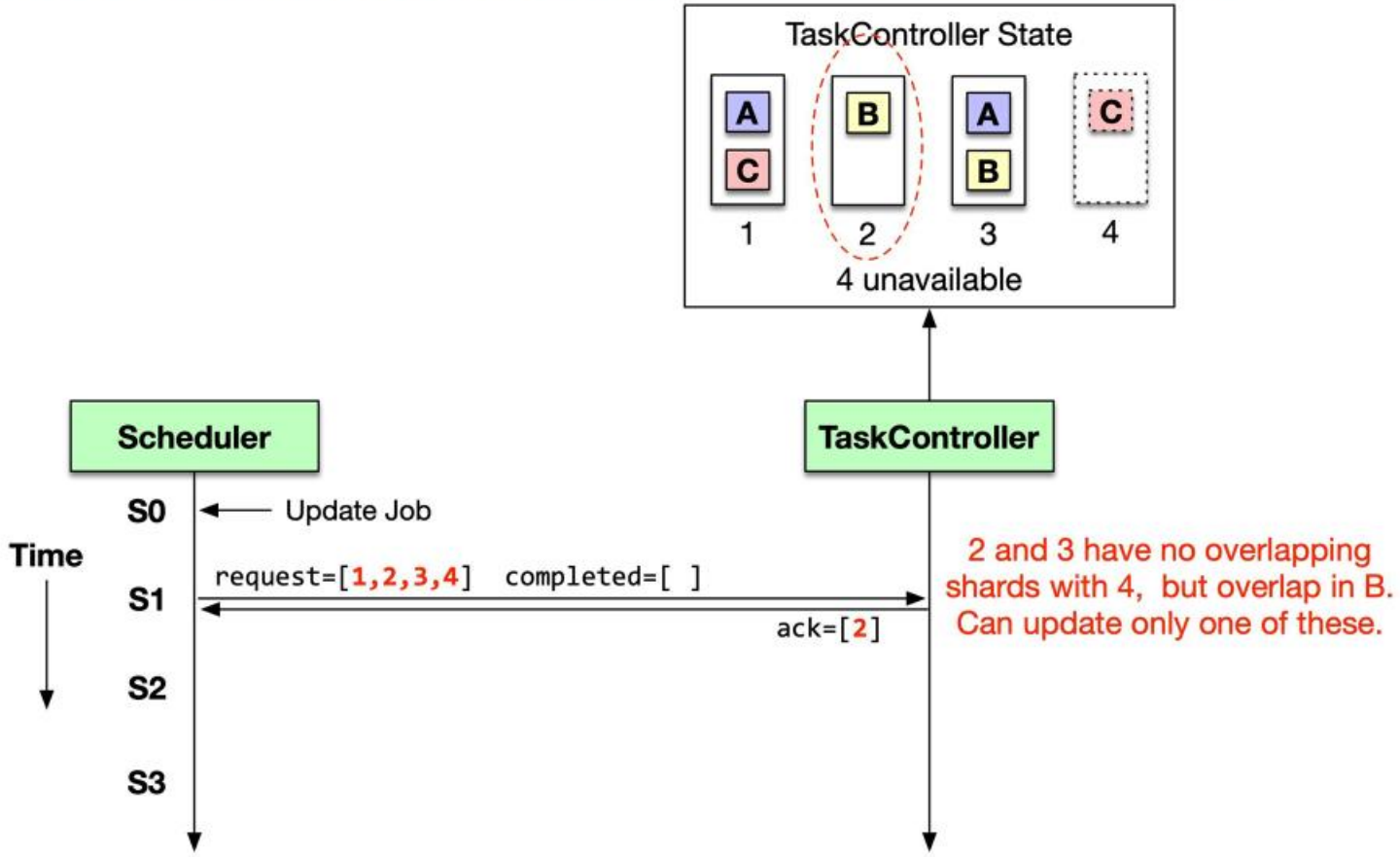
Intuition: Collaborate with applications to handle lifecycle events.

software release.
Shard C unavailable!

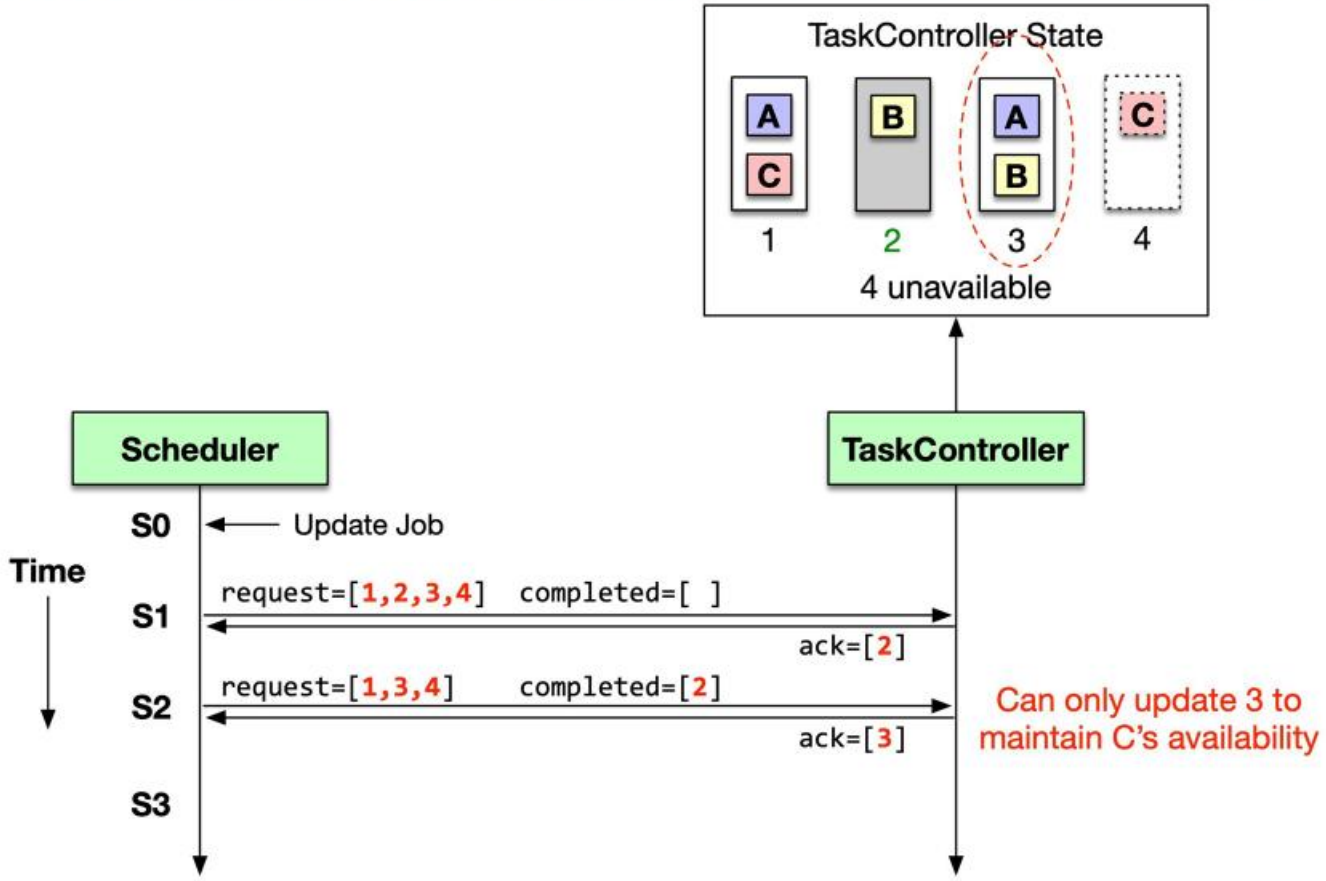
How does Twine collaborate with applications?



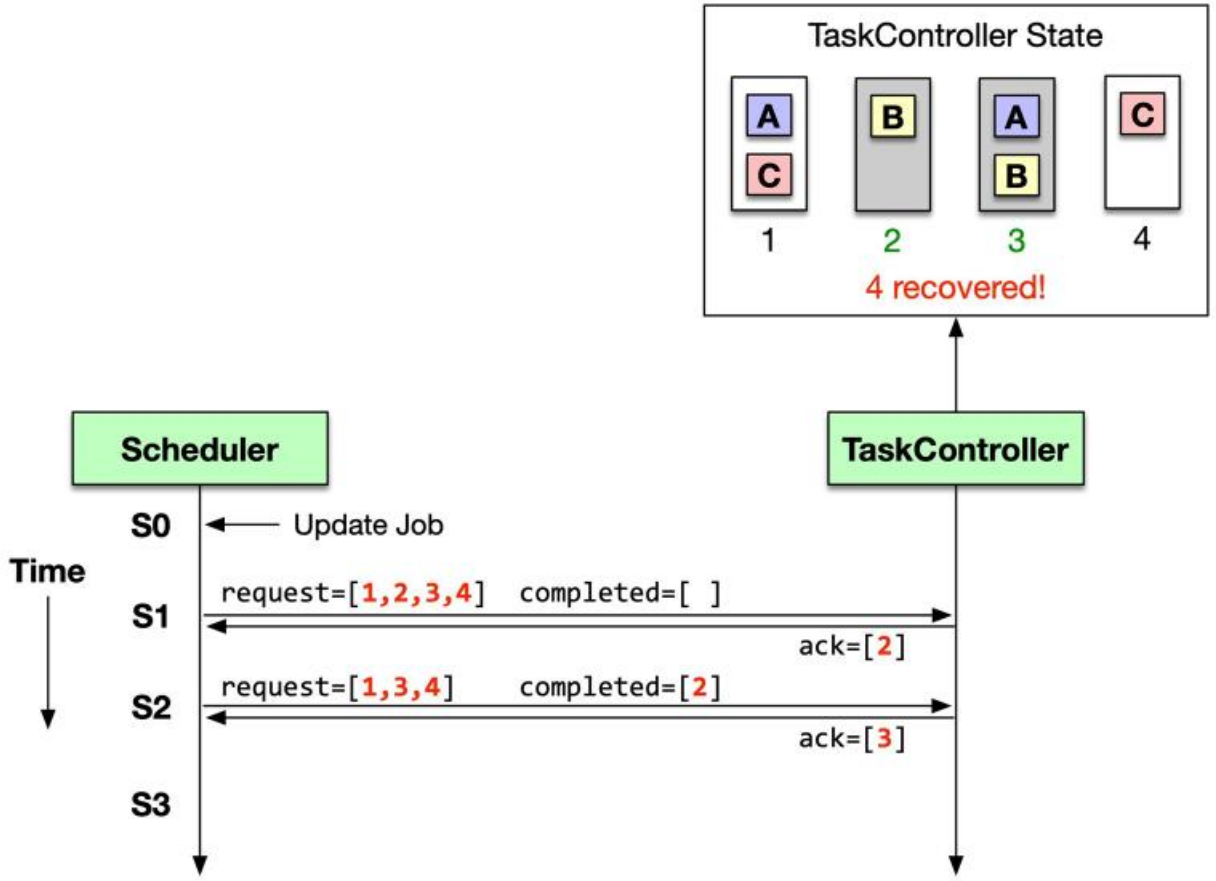
How does Twine collaborate with applications?



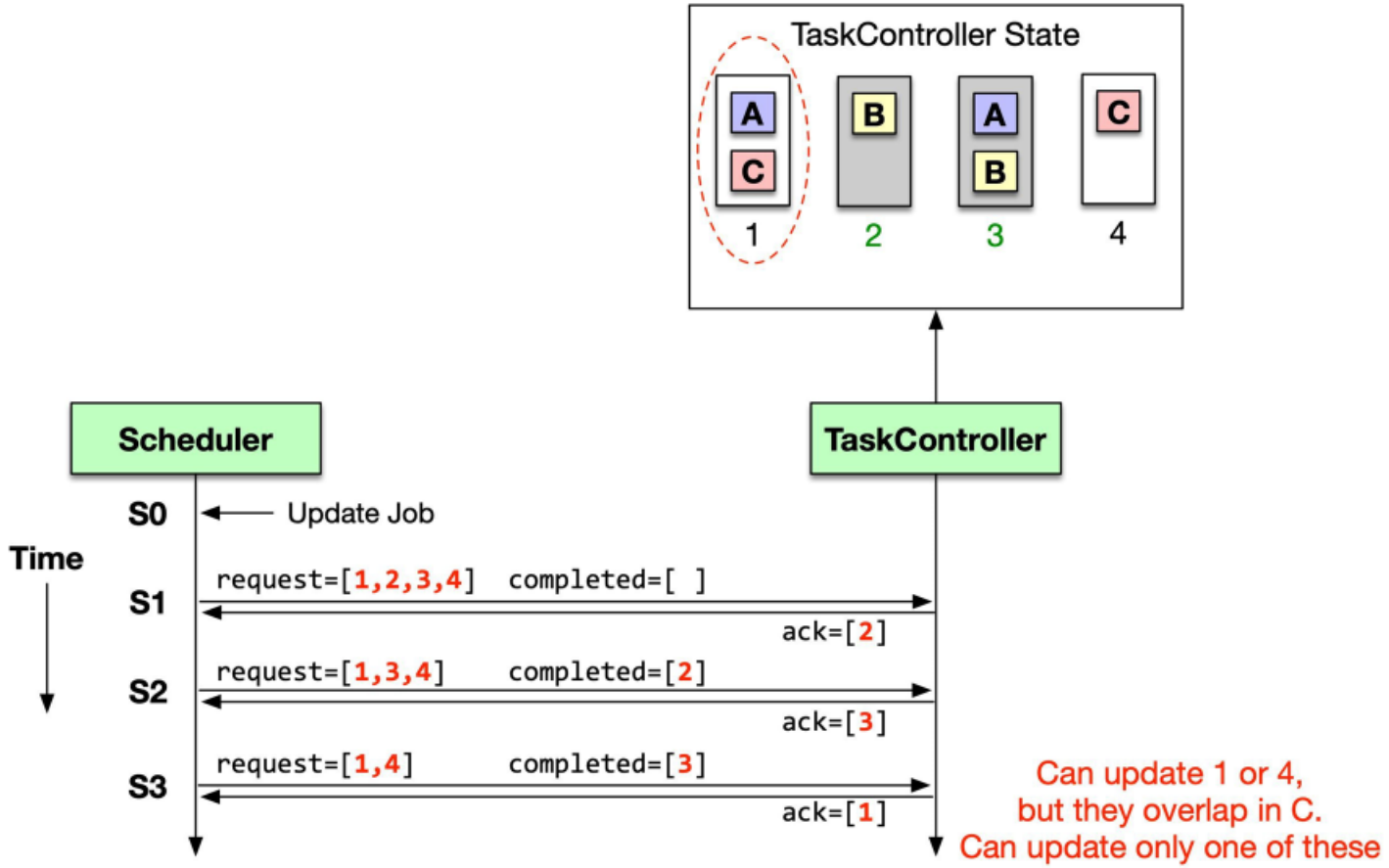
How does Twine collaborate with applications?



How does Twine collaborate with applications?

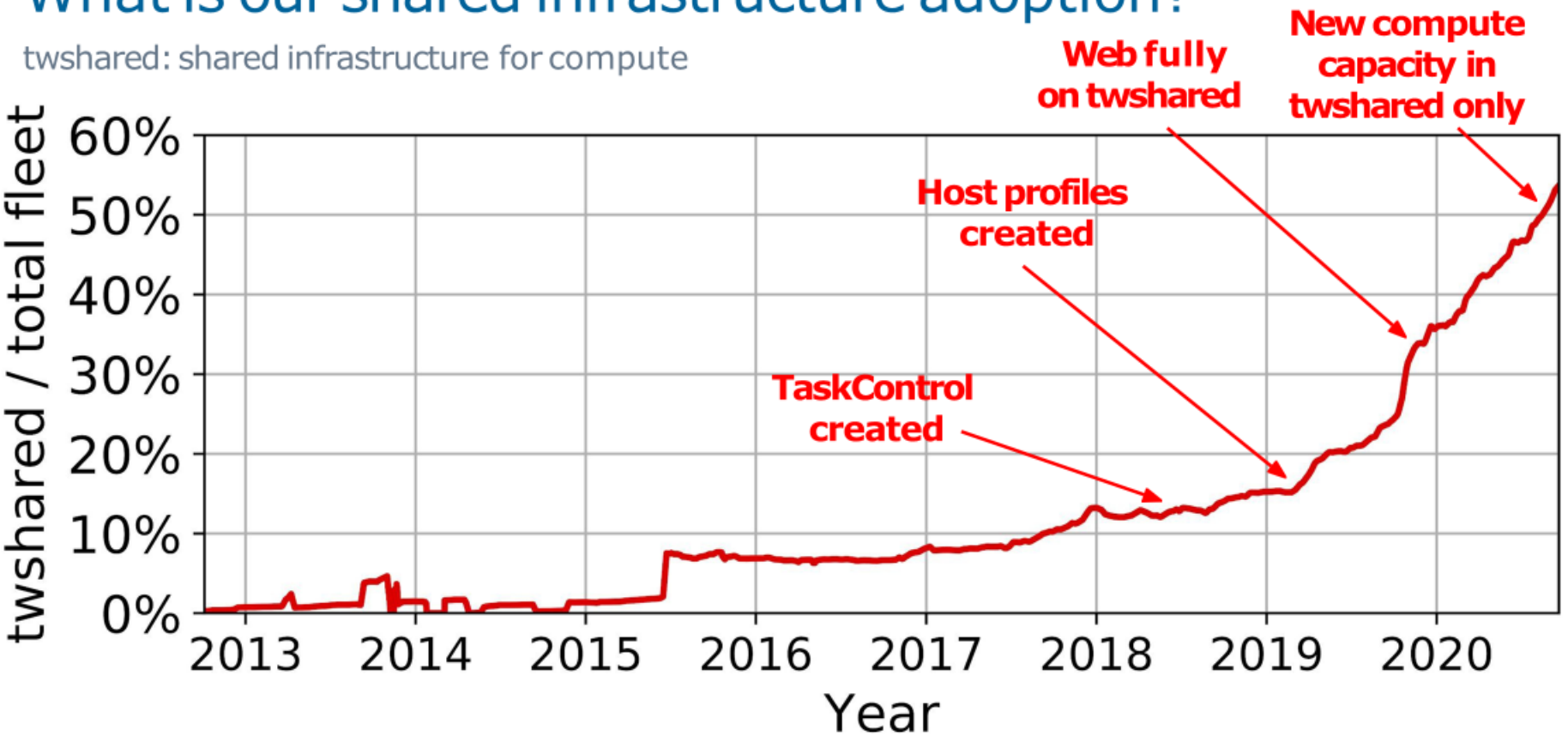


How does Twine collaborate with applications?



What is our shared infrastructure adoption?

twshared: shared infrastructure for compute



Conclusion

Evolving Twine over the past 10 years

Dynamic machine partitioning

Avoids stranded capacity in isolated clusters and enables fleet-wide optimizations

Customization in shared infrastructure

Support ubiquitous shared infrastructure to improve efficiency without sacrificing workload performance or capability

Small machines

Achieve higher power efficiency globally

Discussion

Q1

- Mesos uses a resource offer model to allocate containers for tasks. Twine requires app schedulers to specify the number of tasks and machines. Why the difference?

Q2

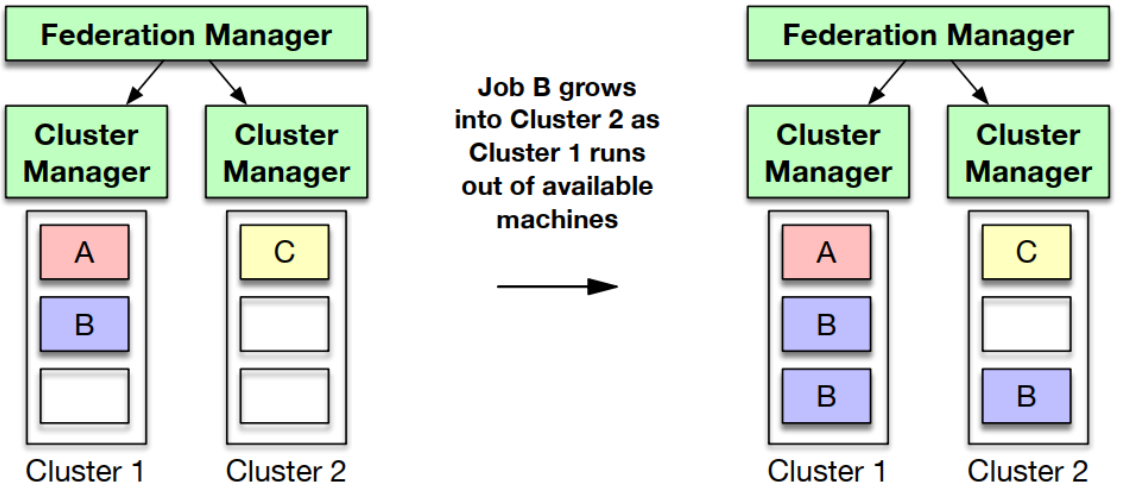
- Mesos is a two-level scheduler (Master, Frameworks). Why does Twine call itself a three-level scheduler?

Q3

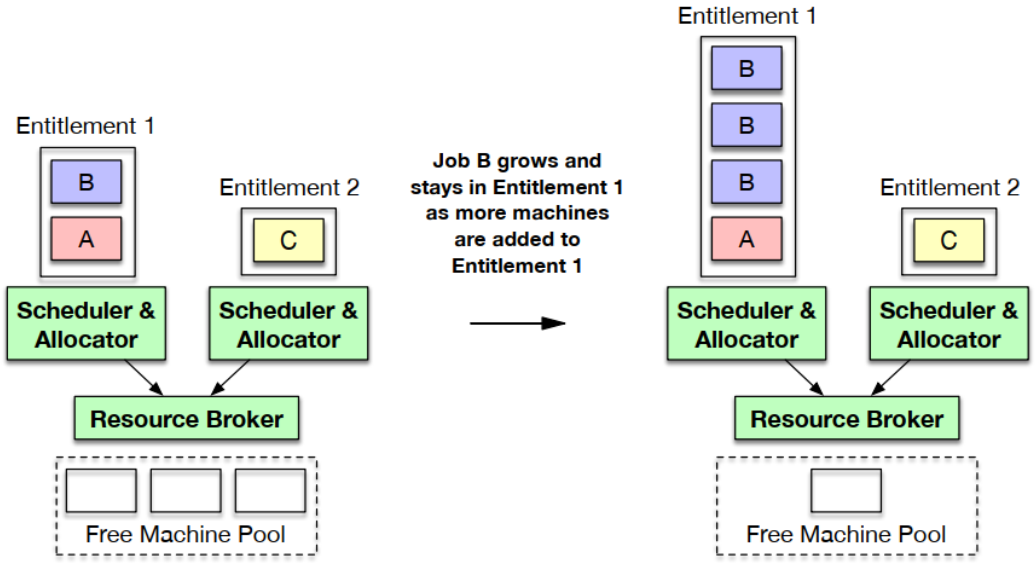
- What could be a drawback of assigning tasks to a logical cluster (entitlement) instead of a physical cluster? How does Twine address this drawback?

Q4

- Twine uses sharding instead of federation to scale resource allocation across data centers. What are the tradeoffs?



(a) Federation approach. This approach uses a Cluster Manager per cluster and introduces an additional Federation Manager layer. Each cluster has a set of statically configured machines. As job `B` in `Cluster 1` keeps growing, it overflows into `Cluster 2`.



(b) Twine's sharding approach. As job `B` grows, Twine adds more machines to `Entitlement 1`, and job `B` stays with the same entitlement and scheduler shard.