ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

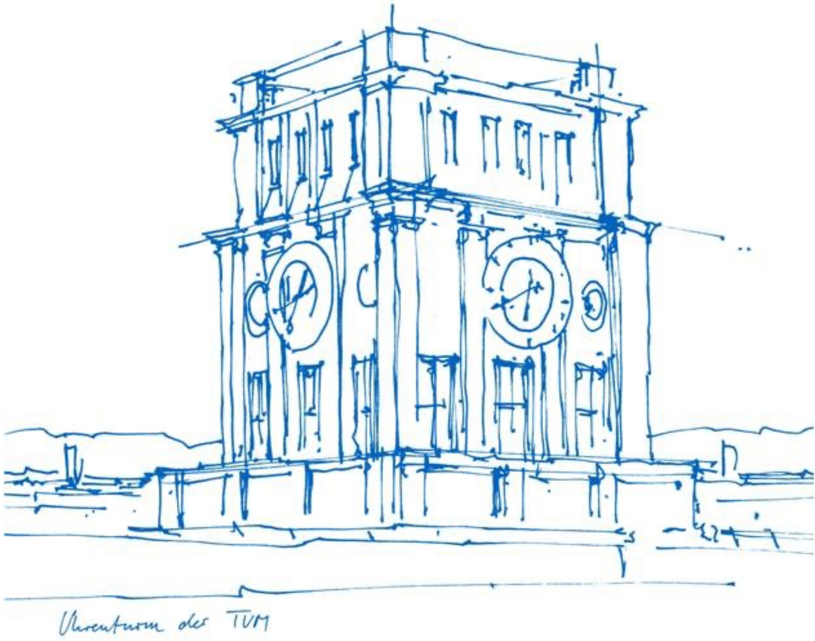
Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing

Giovanni Bartolomeo Mehdi Yosofie Simon Bäurle Oliver Haluszczynski Nitinder Mohan Jörg Ott

Chair of Connected Mobility

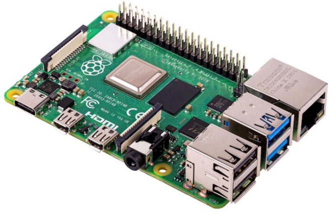
Technical University of Munich, Germany

oakestra.io



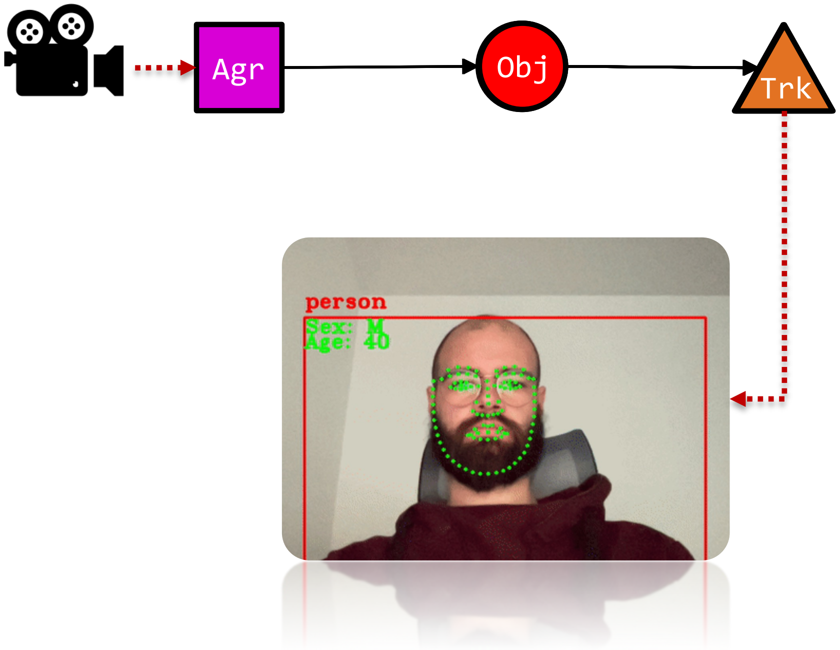
Edge Computing

- Constrained small footprint hardware
- Heterogeneous Infrastructures
    - CPU/GPU Architecture
    - Networking
    - Connectivity
    - Ownership
- User proximity
- Supports latency-critical applications



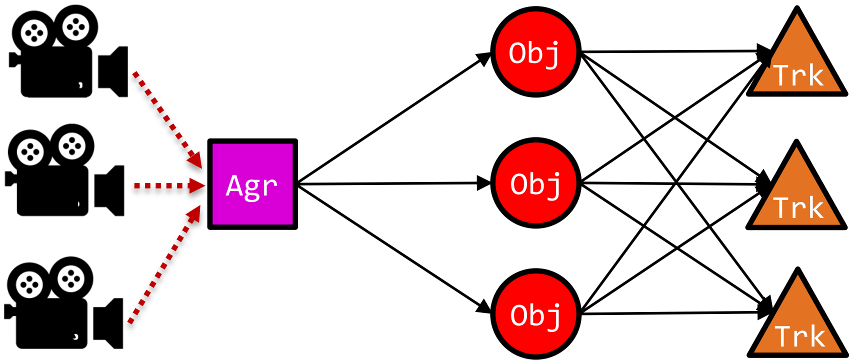
Service Orchestration

- Management and coordination of services across the available resources



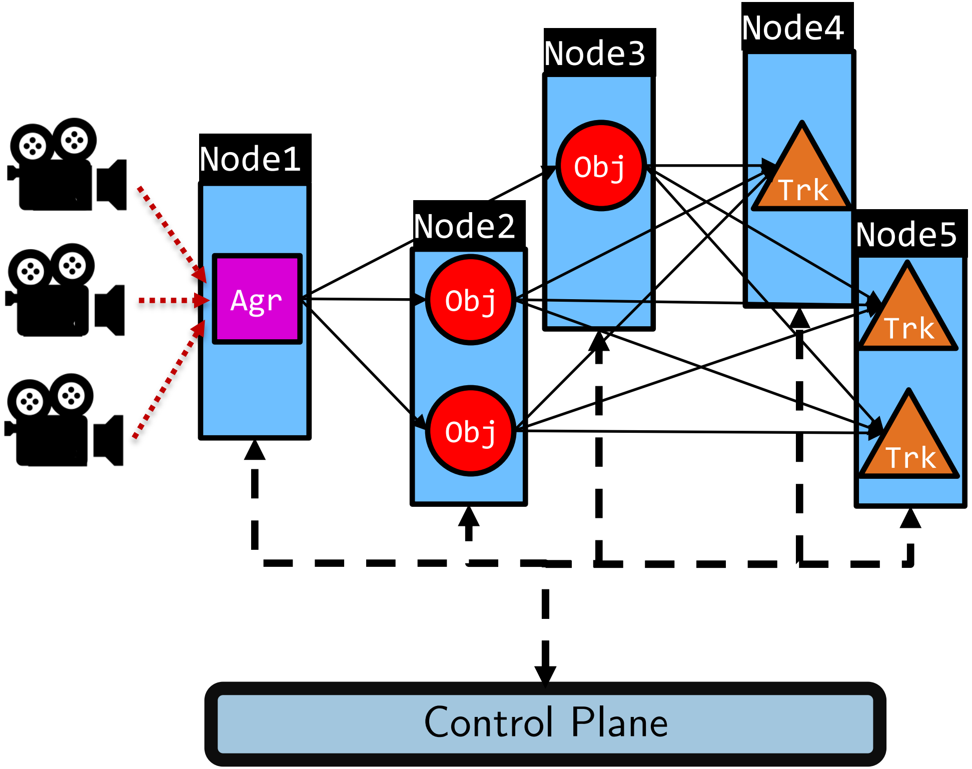
Service Orchestration

- Management and coordination of services across the available resources
  - Resources and services monitoring
  - Replicas scale-up/down
  - Workload migration
  - Services networking



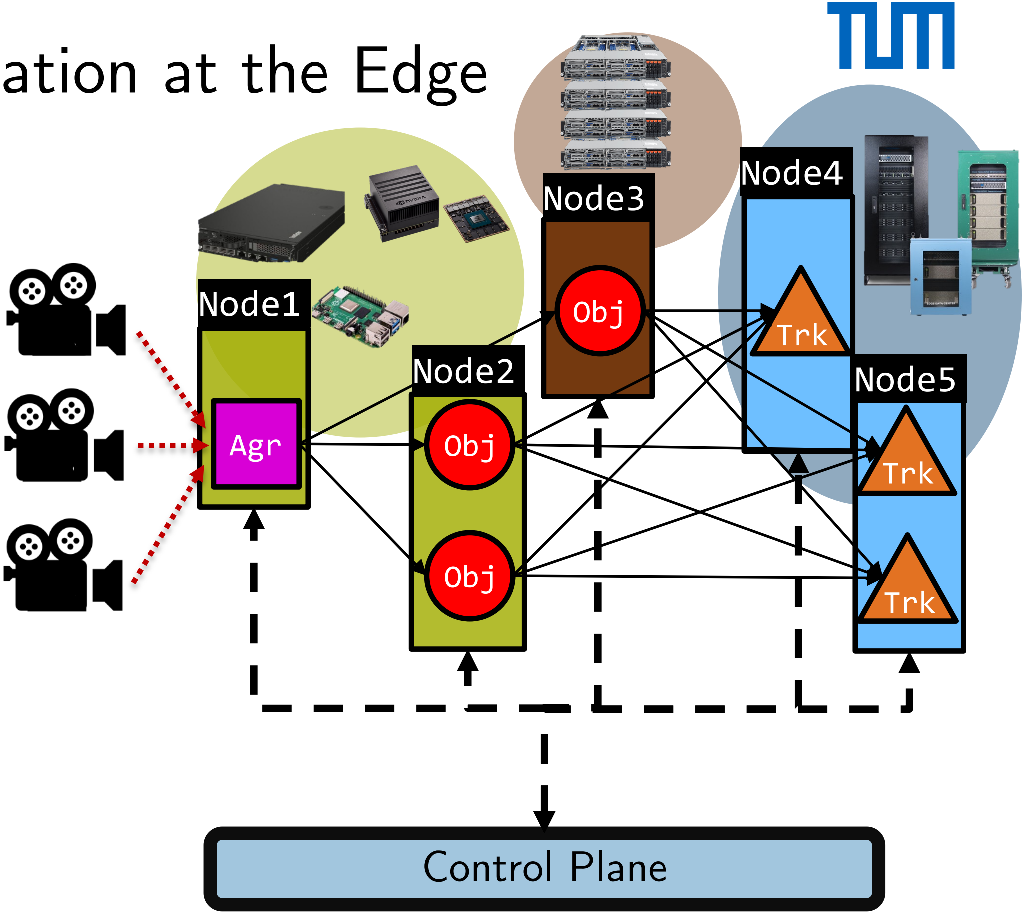
Service Orchestration

- Management and coordination of services across the available resources
  - Resources and services monitoring
  - Replicas scale-up/down
  - Workload migration
  - Services networking
  - … and more
- Control plane + Nodes
- Kubernetes (K8s) family



Challenges of Service Orchestration at the Edge

- Multiple infrastructure providers
- Solutions designed for datacenter environments
- Strong consistency of cluster status and resources limits performance at the Edge [3]
- Lightweight distributions like K3s, MicroK8s inherit the same design assumptions of K8s.
- Global state transfer requirement for networking



[1] Andrew Jeffery, Heidi Howard, and Richard Mortier. 2021. Rearchitecting Kubernetes for the Edge. 4th ACM EdgeSys (2021)

Three-tier hierarchical orchestration

Consolidation of multiple edge providers

Lightweight Implementation

Semantic overlay networking

Resource aggregation

Oakestra

Site-to-Site tunneling

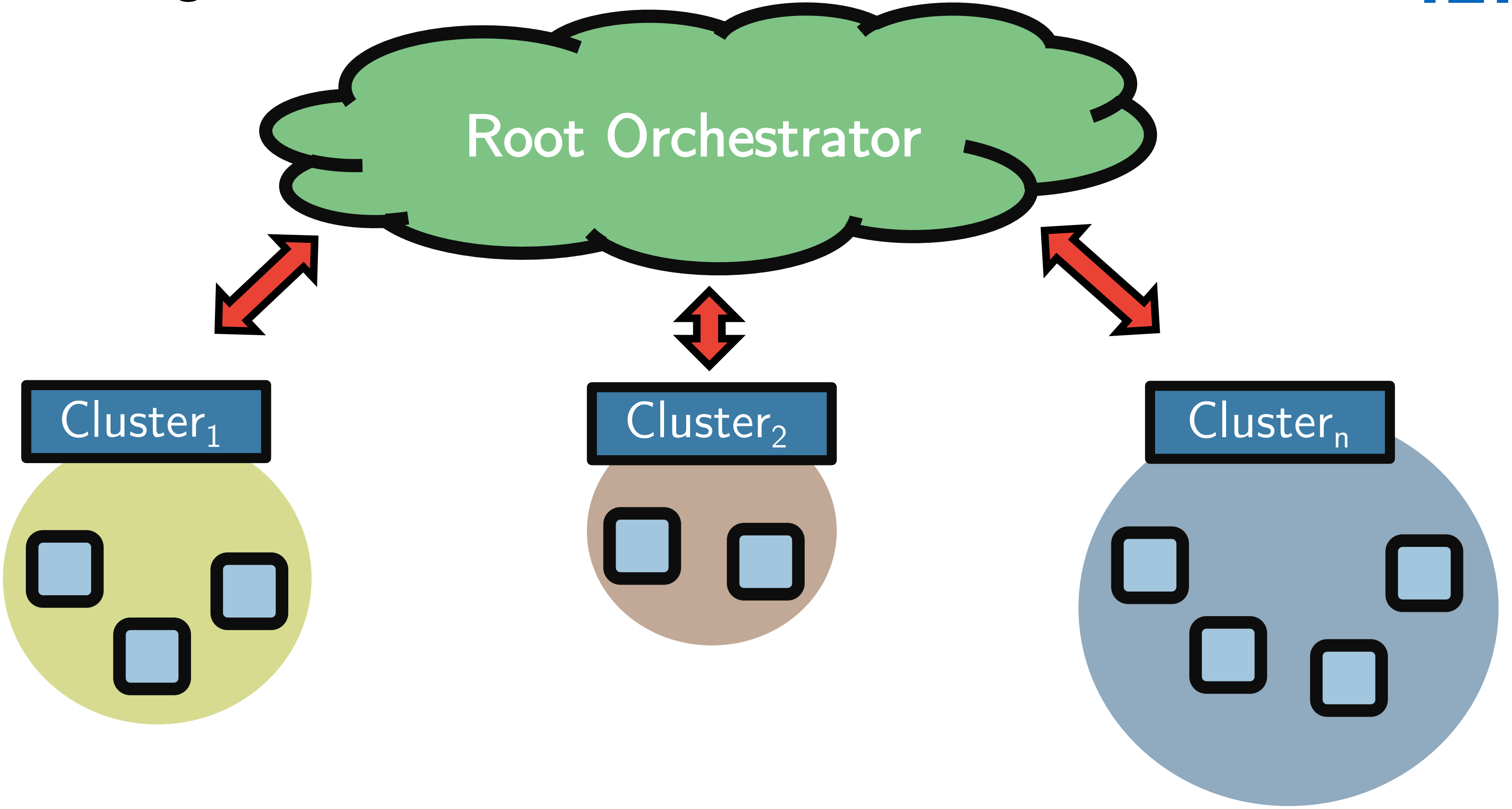
Multi-virtualization support

Deployment across geography

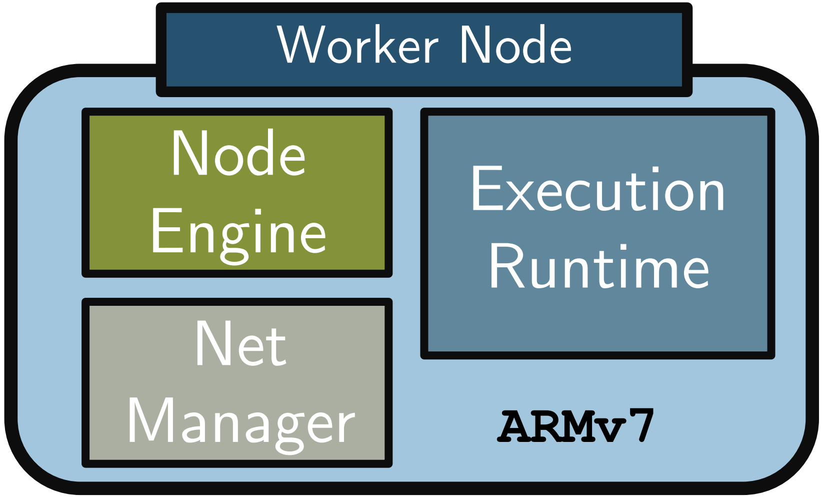
Delegated service scheduling

Fine-grained extensible SLA primitives

System Design

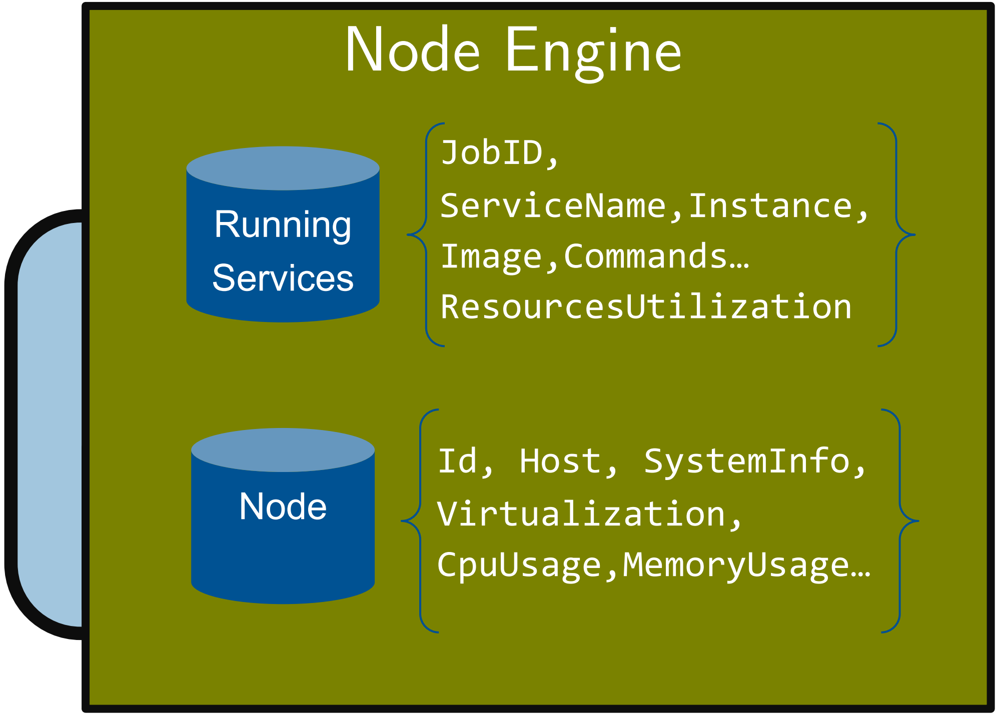


Worker Node



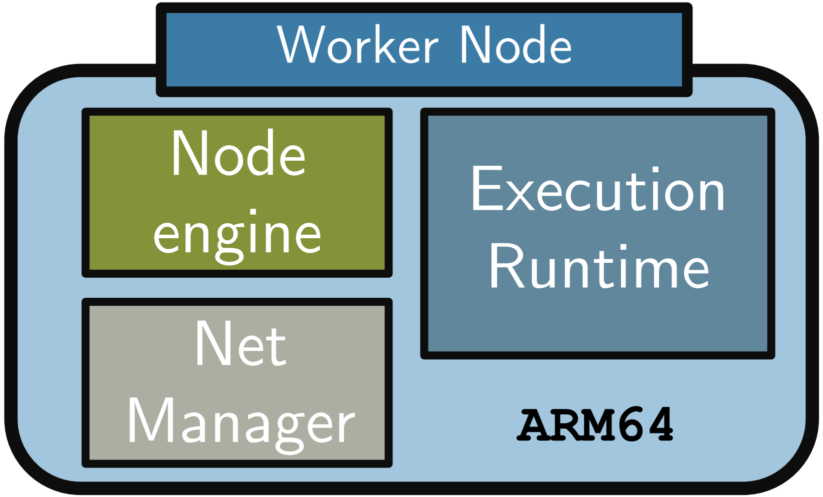
- Multiple architectures
- Multiple execution runtimes
  - Default: containerd
- Distributed networking management
- Resource/service monitoring

Worker Node

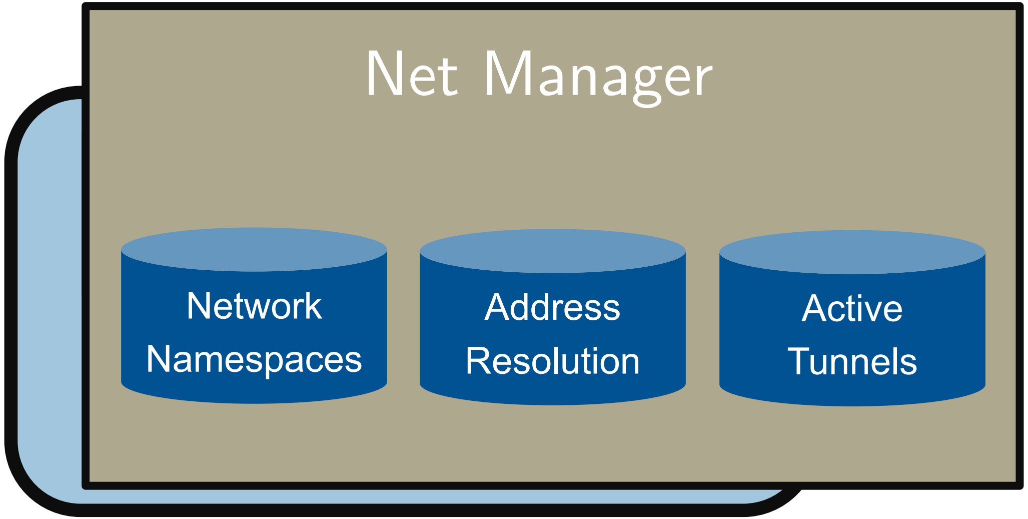


- Deployed service instances
- Service‘s resources utilization
- Node‘s system and real time info
- Periodical cluster updates

Worker Node

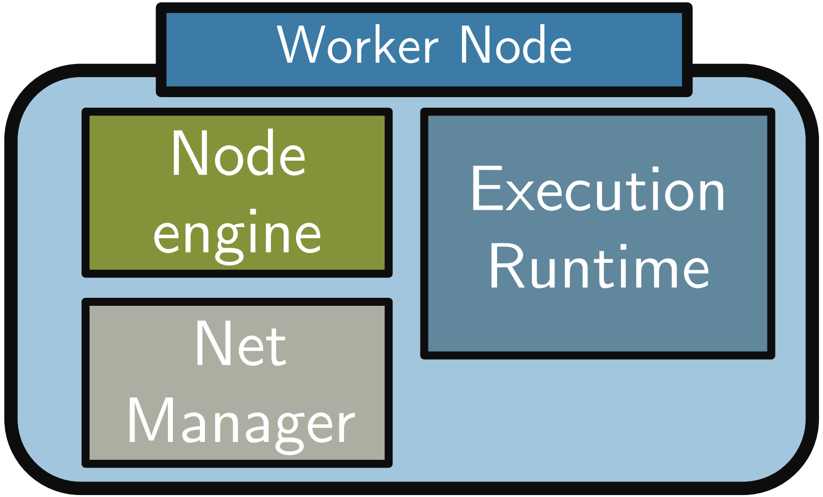

Worker Node
Node engine
Execution Runtime
Net Manager
ARM64

Worker Node

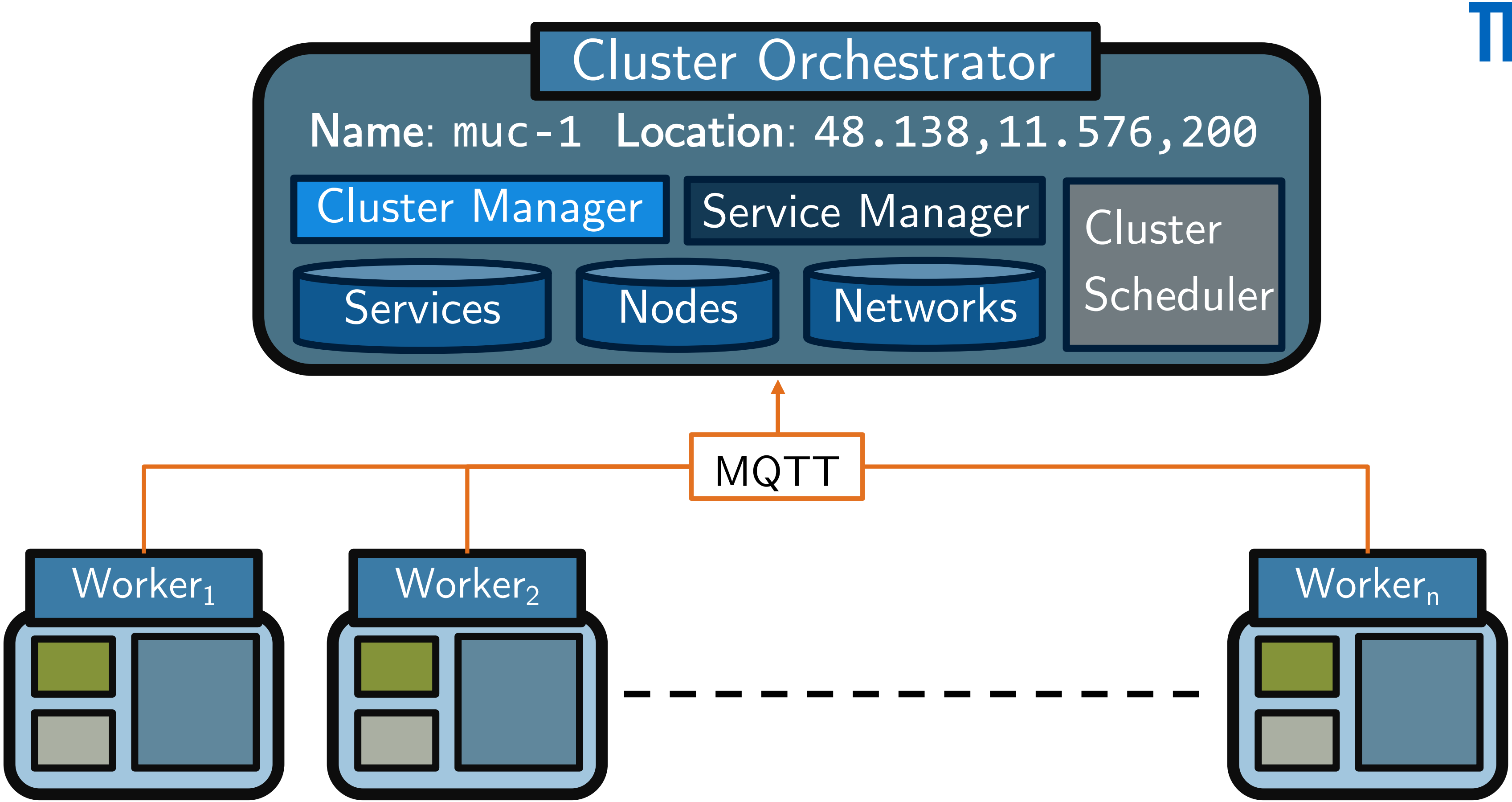


- Autonomously manages service addressing and traffic tunneling
- Creates the network namespaces for the services
- More details later...

Worker Node



A cluster can be composed of multiple nodes



Cluster Orchestrator

Name: `muc-1` Location: `48.138,11.576,200`

Cluster Manager

Service Manager

Cluster Scheduler

Services

Nodes

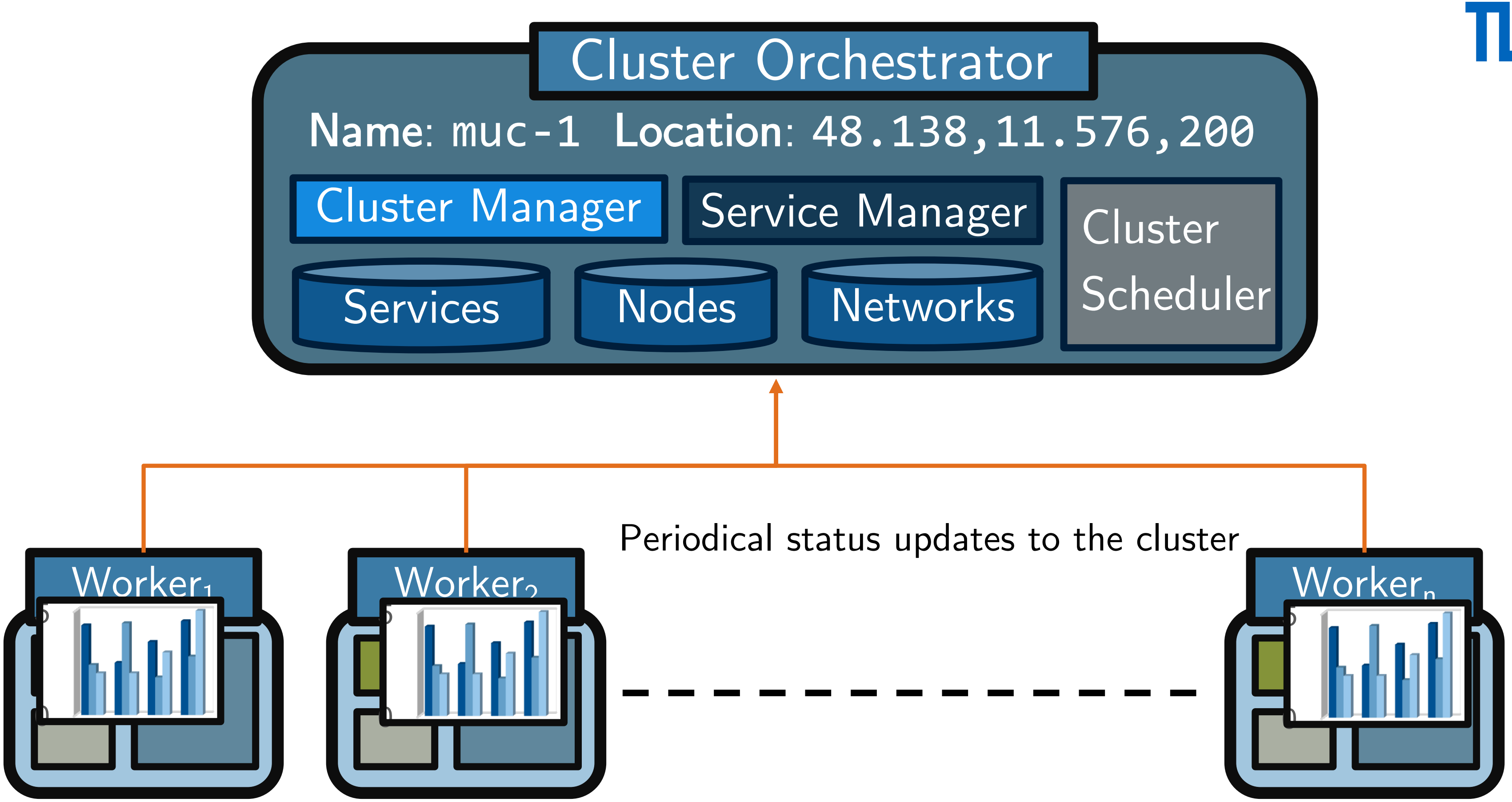
Networks

MQTT

Worker$_1$

Worker$_2$

Worker$_n$



Cluster Orchestrator

Name: `muc-1` Location: `48.138,11.576,200`

Cluster Manager

Service Manager

Cluster Scheduler

Services

Nodes

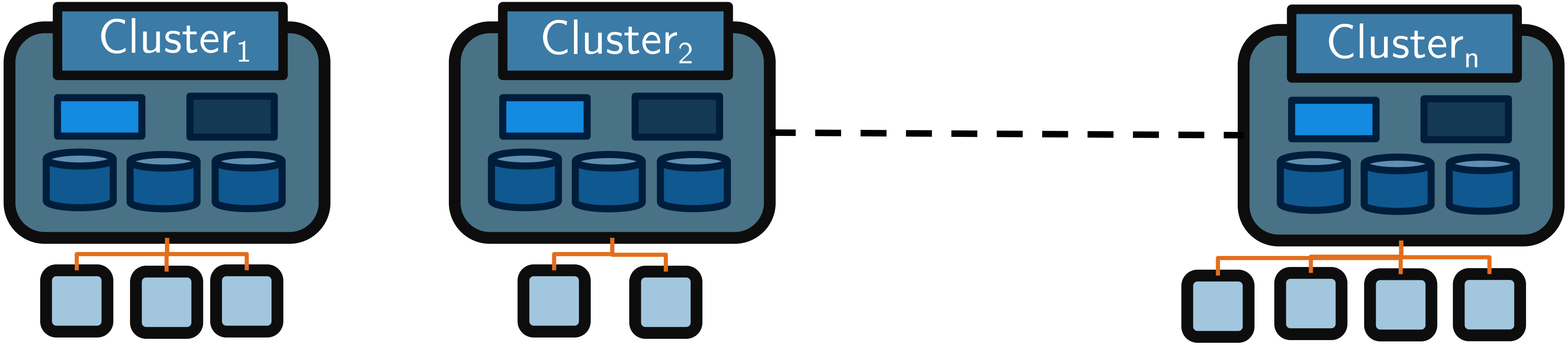
Networks

Periodical status updates to the cluster

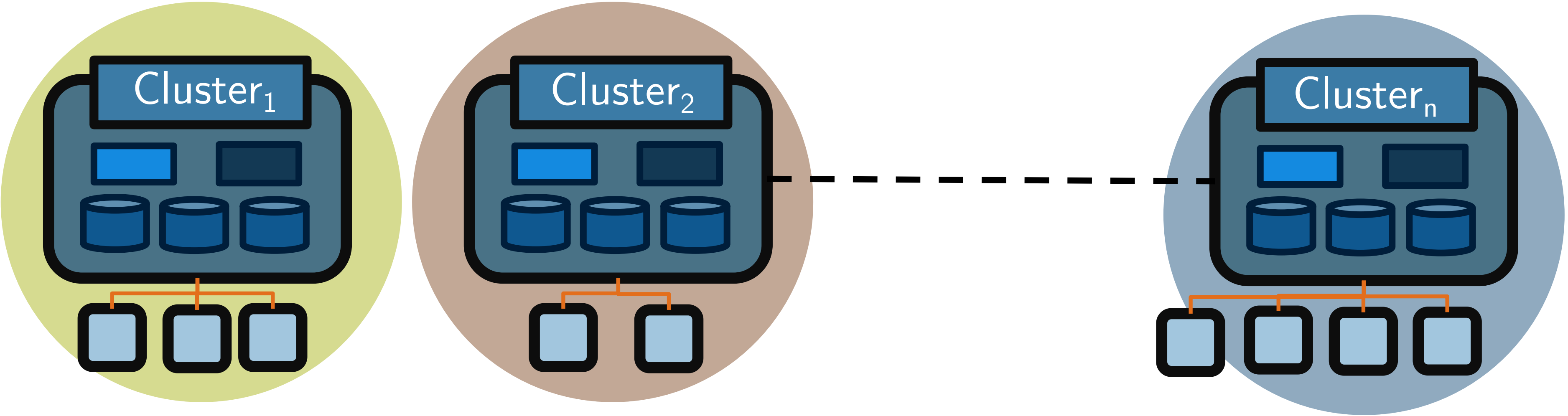
Worker$_1$

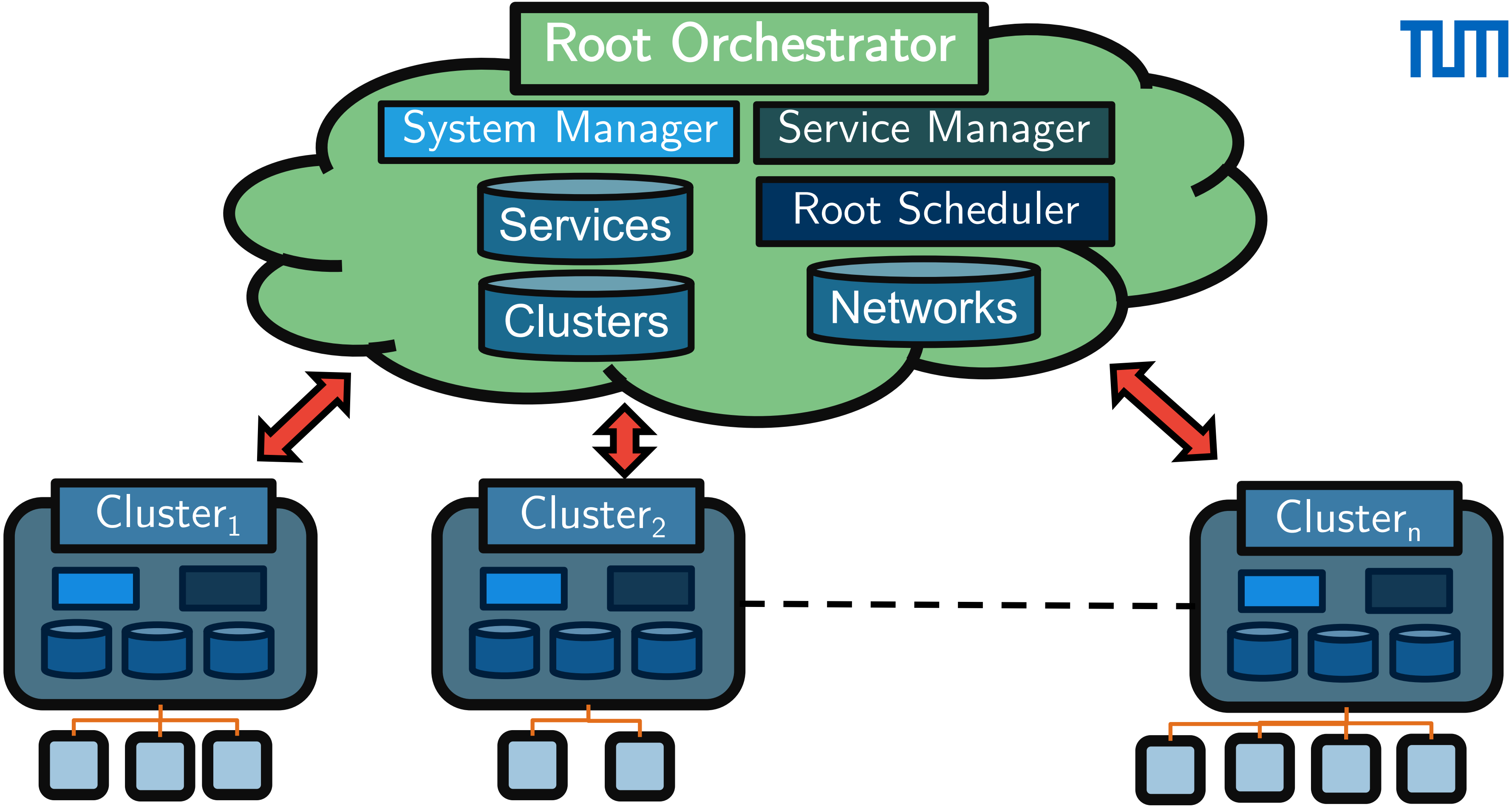
Worker$_2$

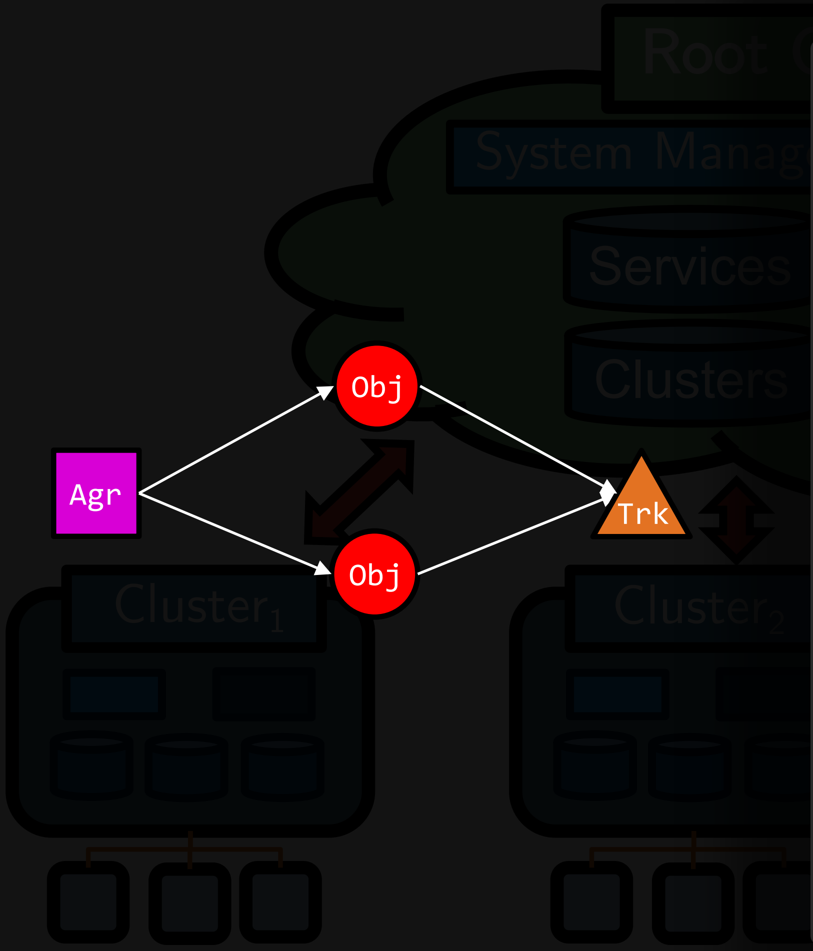
Worker$_n$

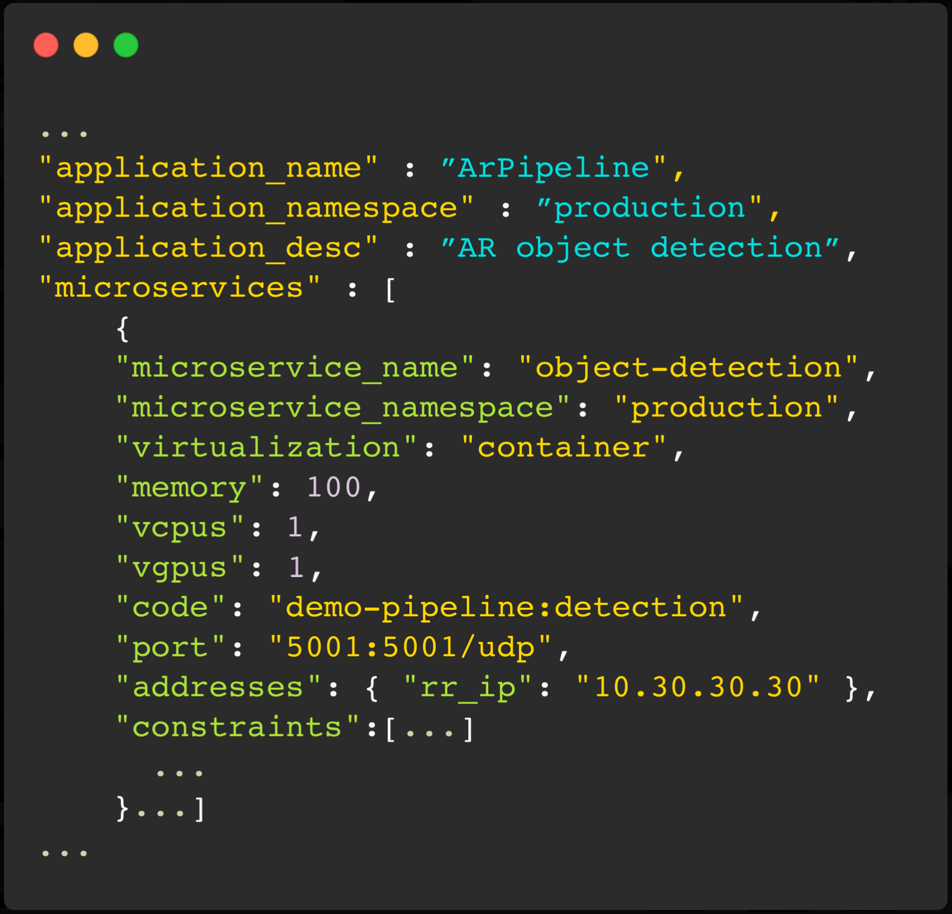

Cluster_1
Cluster_2
Cluster_n

- Different clusters can be administrated by different providers
- Resource aggregation to preserve minute details about internal infrastructure

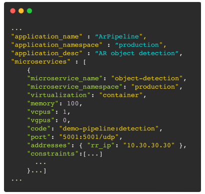



Root Orchestrator
System Manager
Service Manager
Services
Root Scheduler
Clusters
Networks
Cluster$_1$
Cluster$_2$
Cluster$_n$

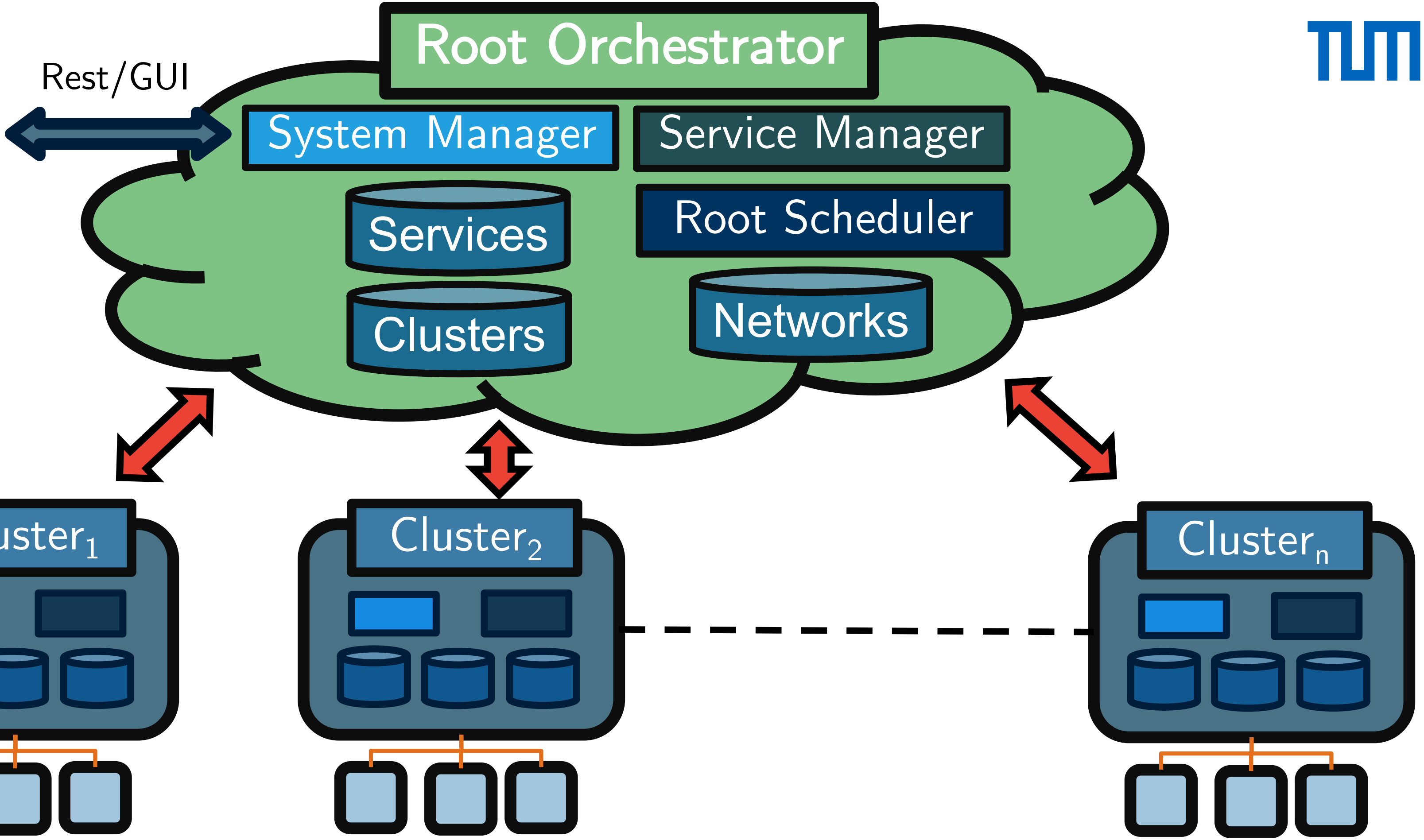


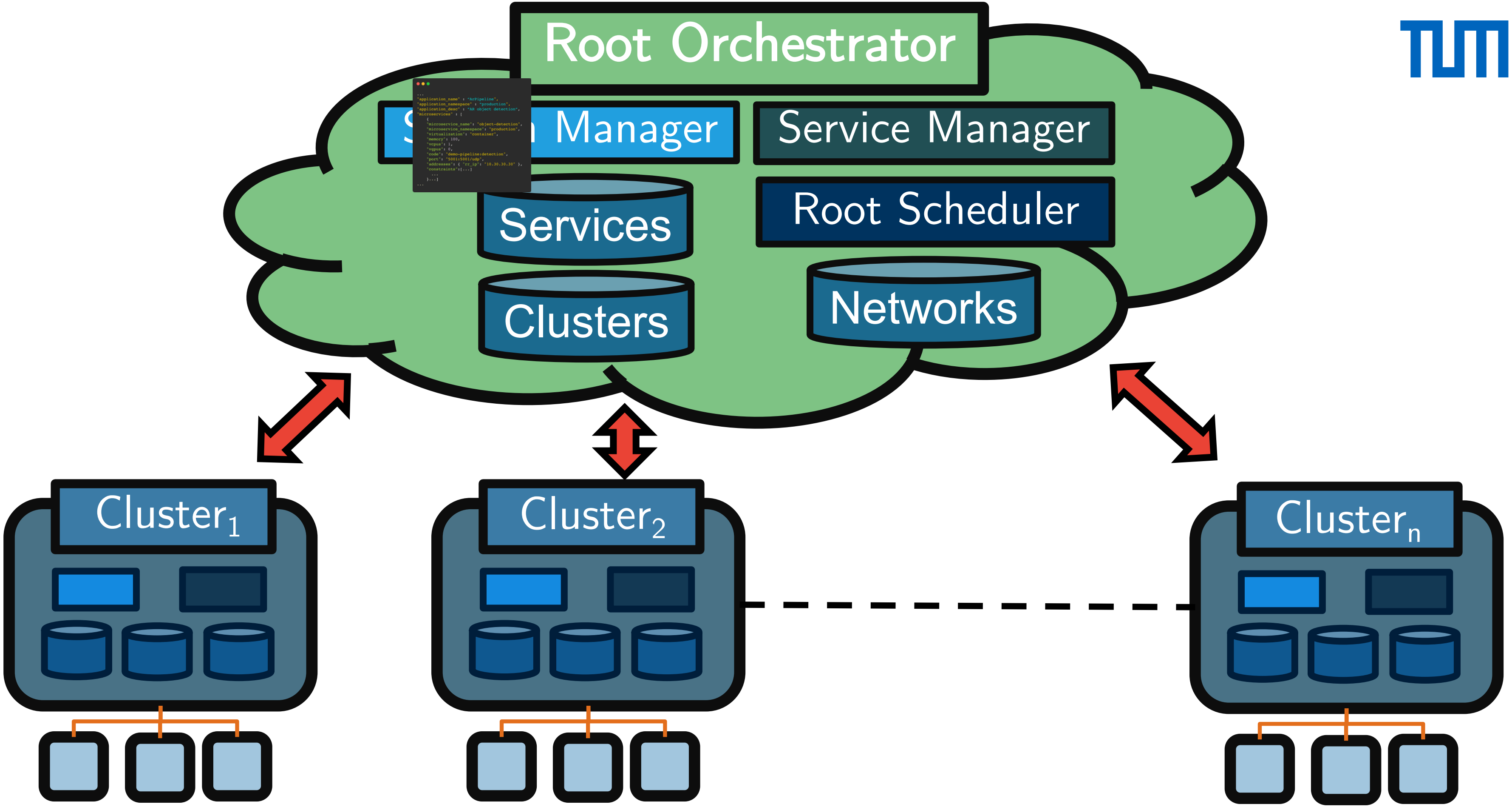


```
...
"application_name" : ”ArPipeline”,
"application_namespace" : ”production",
"application_desc" : ”AR object detection”,
"microservices" : [
    {
    "microservice_name": "object-detection",
    "microservice_namespace": "production",
    "virtualization": "container",
    "memory": 100,
    "vcpus": 1,
    "vgpus": 1,
    "code": "demo-pipeline:detection",
    "port": "5001:5001/udp",
    "addresses": { "rr_ip": "10.30.30.30" },
    "constraints":[...]
      ...
    }...]
...
```

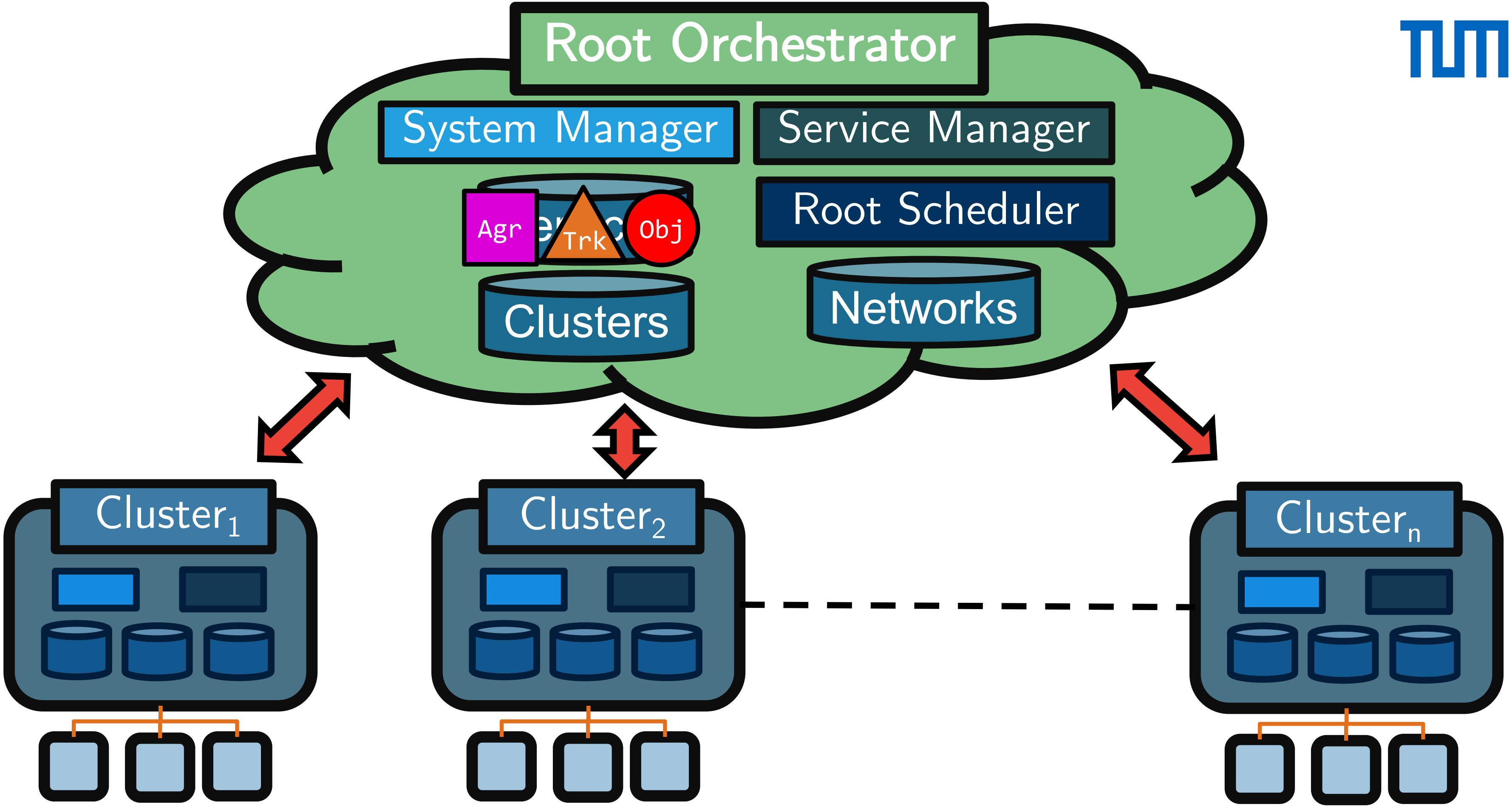


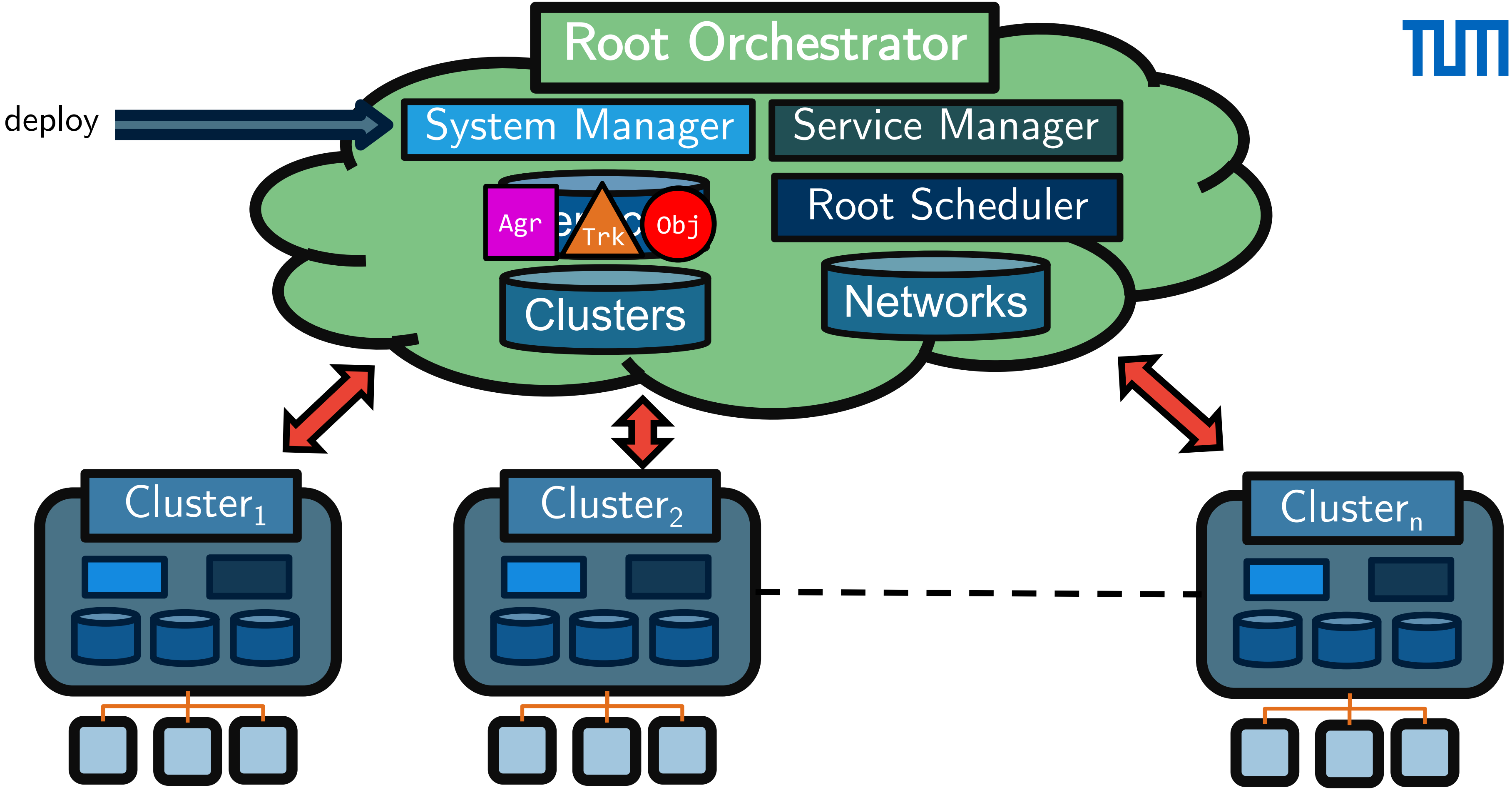
```
...
"application_name" : ”ArPipeline",
"application_namespace" : ”production",
"application_desc" : ”AR object detection”,
"microservices" : [
    {
    "microservice_name": "object-detection",
    "microservice_namespace": "production",
    "virtualization": "container",
    "memory": 100,
    "vcpus": 1,
    "vgpus": 0,
    "code": "demo-pipeline:detection",
    "port": "5001:5001/udp",
    "addresses": { "rr_ip": "10.30.30.30" },
    "constraints":[...]
      ...
    }...]
...
```





```
...
"application_name" : "ArPipeline",
"application_namespace" : "production",
"application_desc" : "AR object detection",
"microservices" : [
    {
    "microservice_name": "object-detection",
    "microservice_namespace": "production",
    "virtualization": "container",
    "memory": 100,
    "vcpus": 1,
    "vgpus": 0,
    "code": "demo-pipeline:detection",
    "port": "5001:5001/udp",
    "addresses": { "rr_ip": "10.30.30.30" },
    "constraints":[...]
      ...
    }...]
...
```


Root Orchestrator
System Manager
Service Manager
Agr
Trk
Obj
Root Scheduler
Clusters
Networks
Cluster$_1$
Cluster$_2$
Cluster$_n$



deploy

Root Orchestrator

System Manager

Service Manager

Root Scheduler

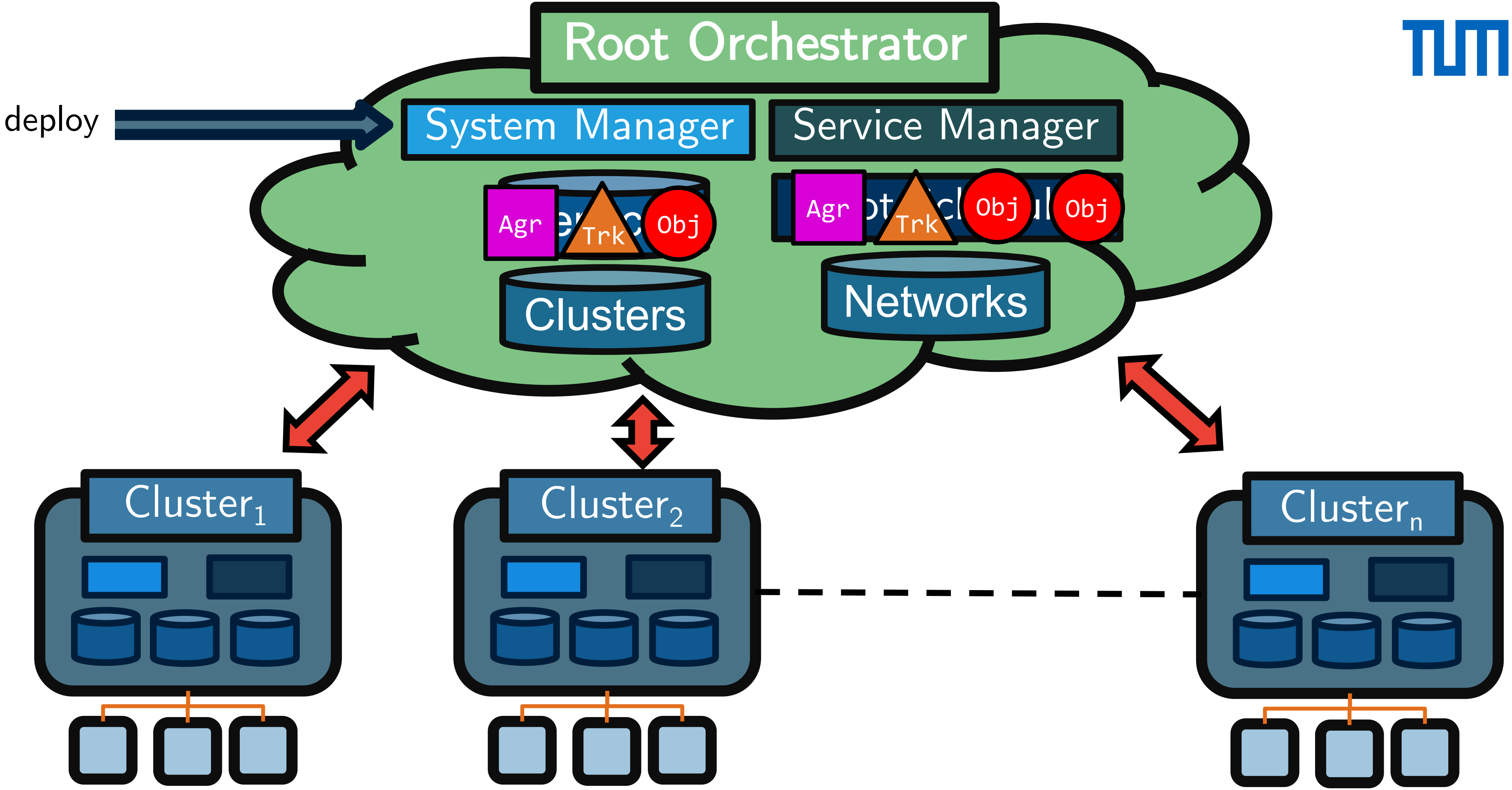
Clusters

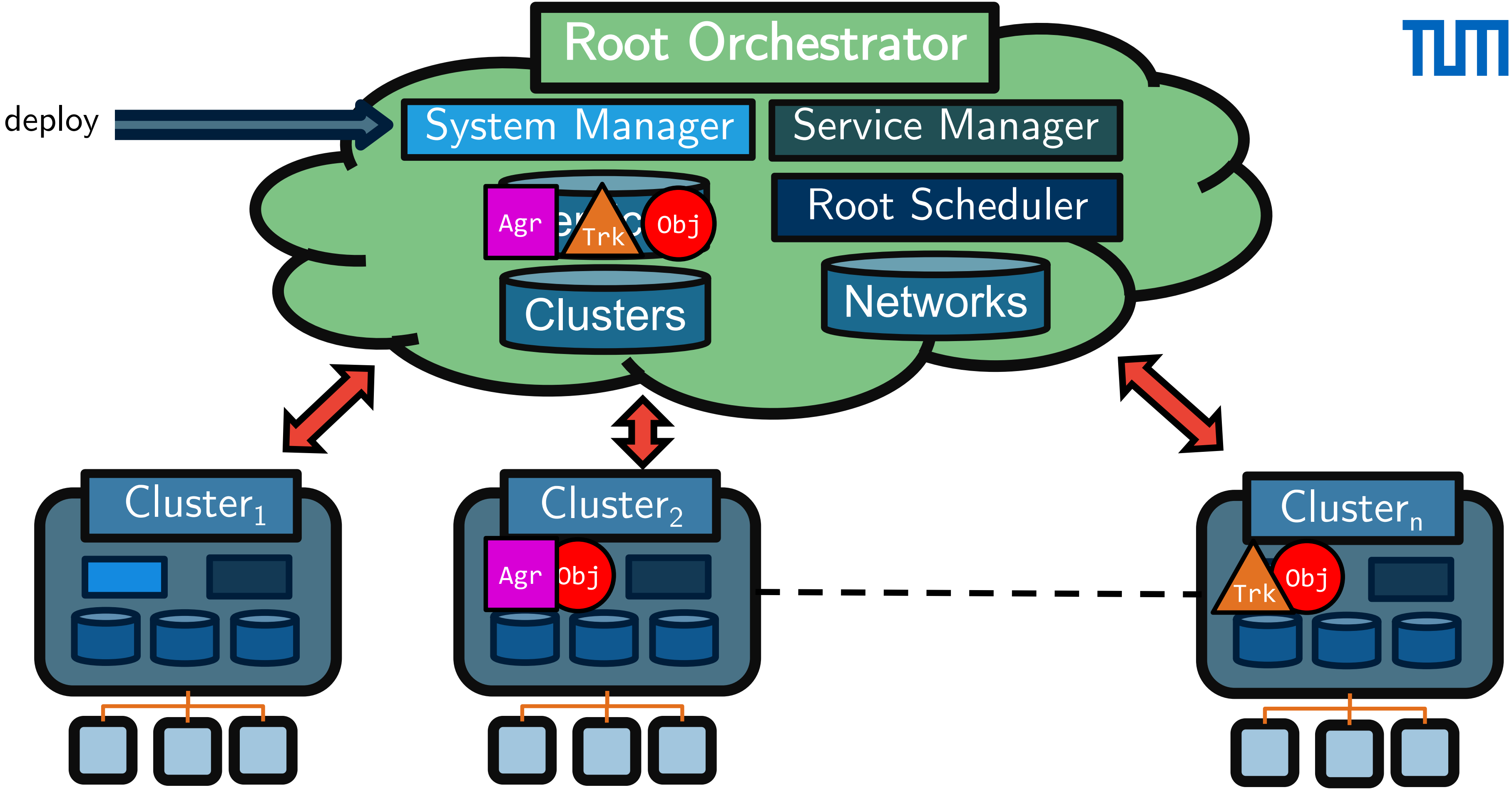
Networks

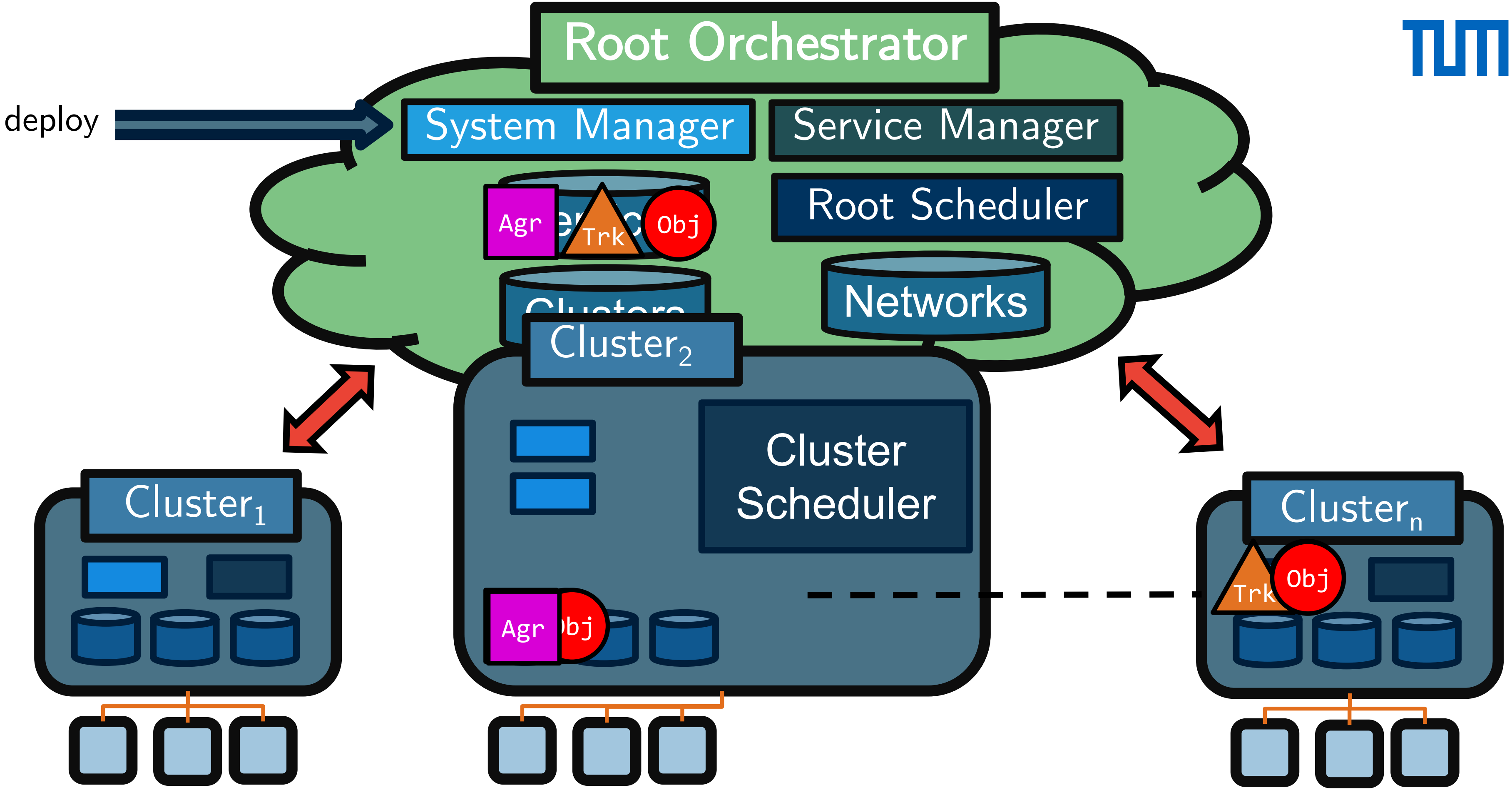
Cluster$_1$

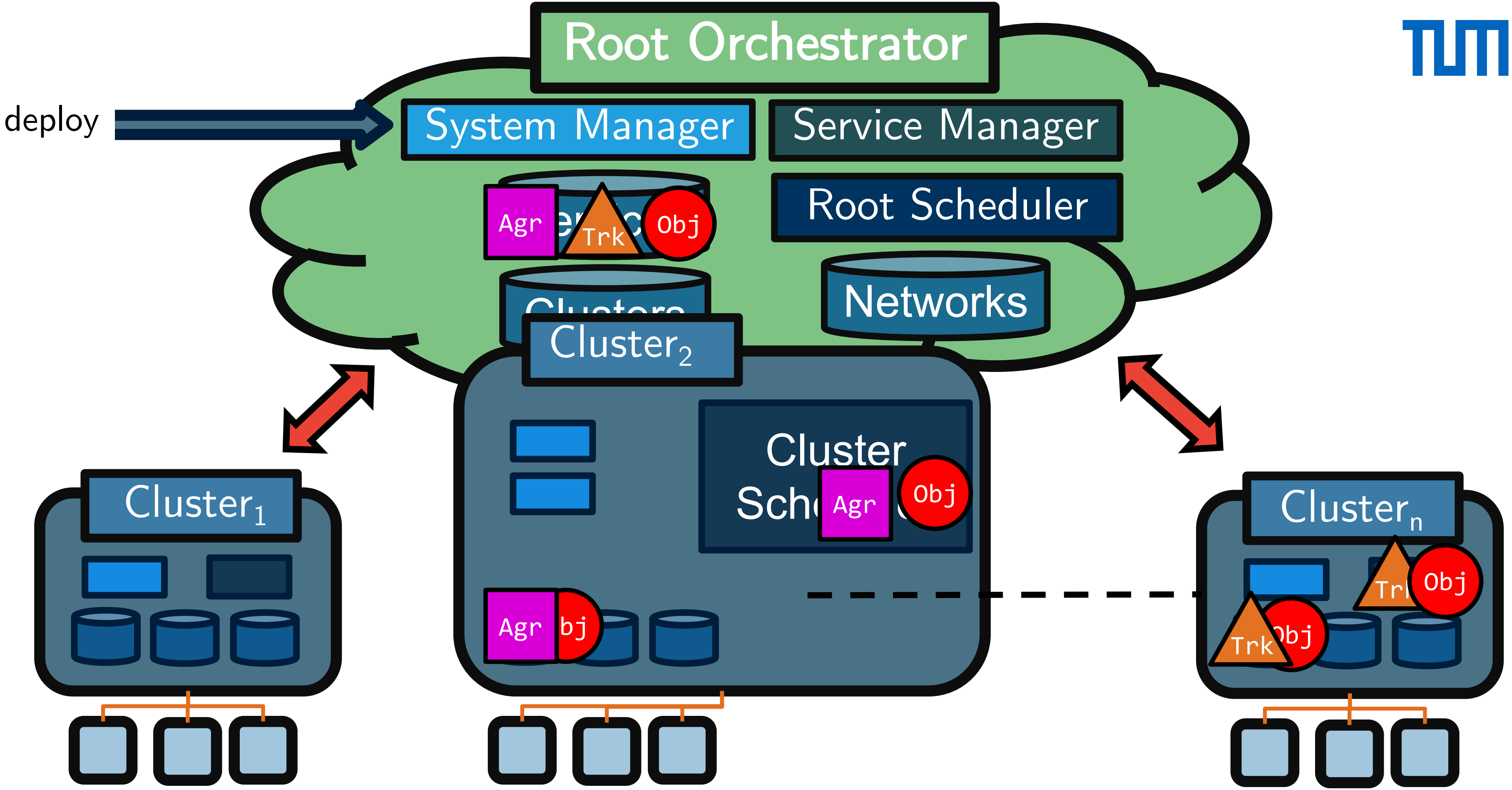
Cluster$_2$

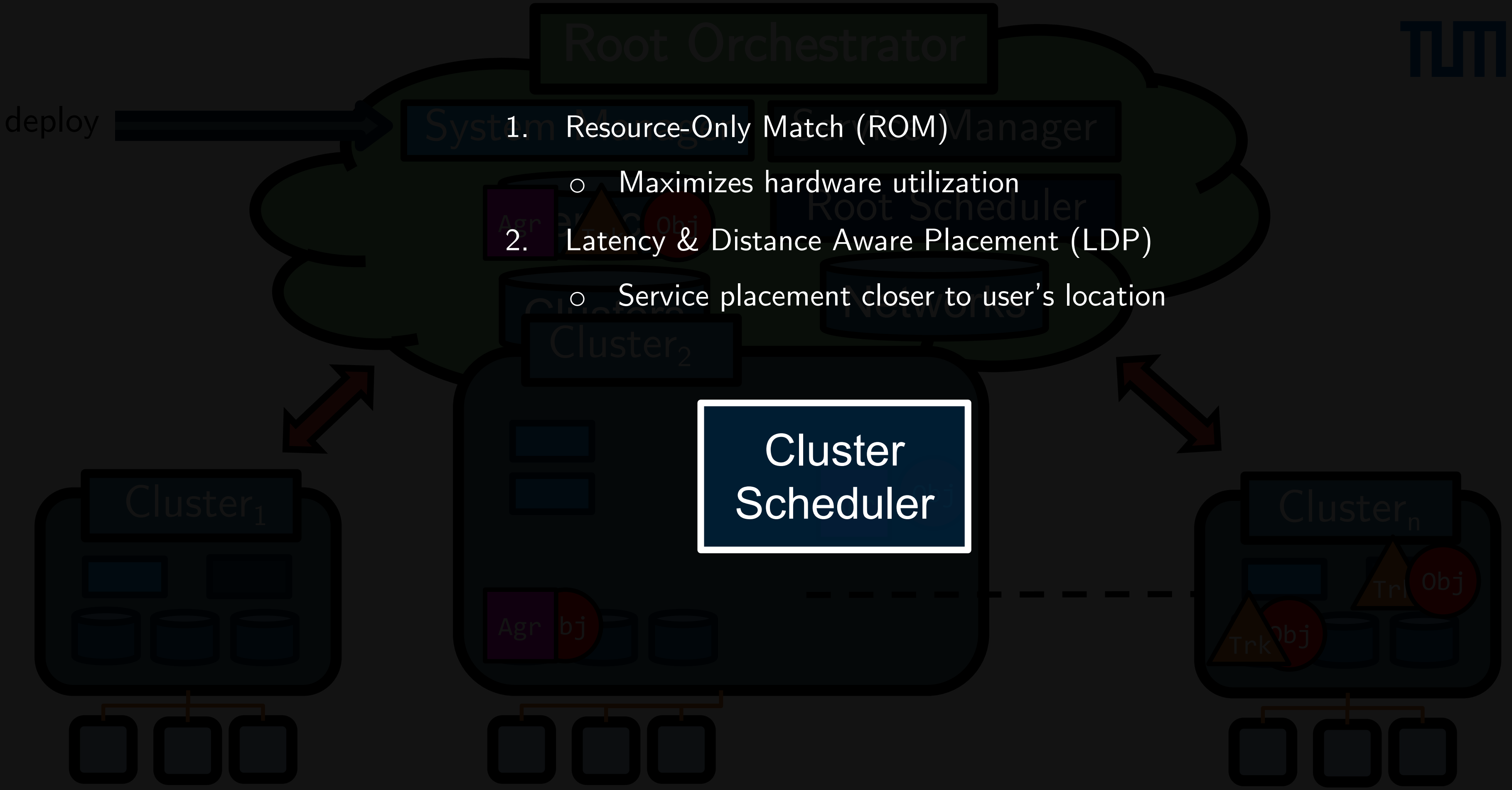
Cluster$_n$


Root Orchestrator
deploy
System Manager
Service Manager
Agr
Trk
Obj
Agr
Trk
Obj
Obj
Clusters
Networks
Cluster$_1$
Cluster$_2$
Cluster$_n$

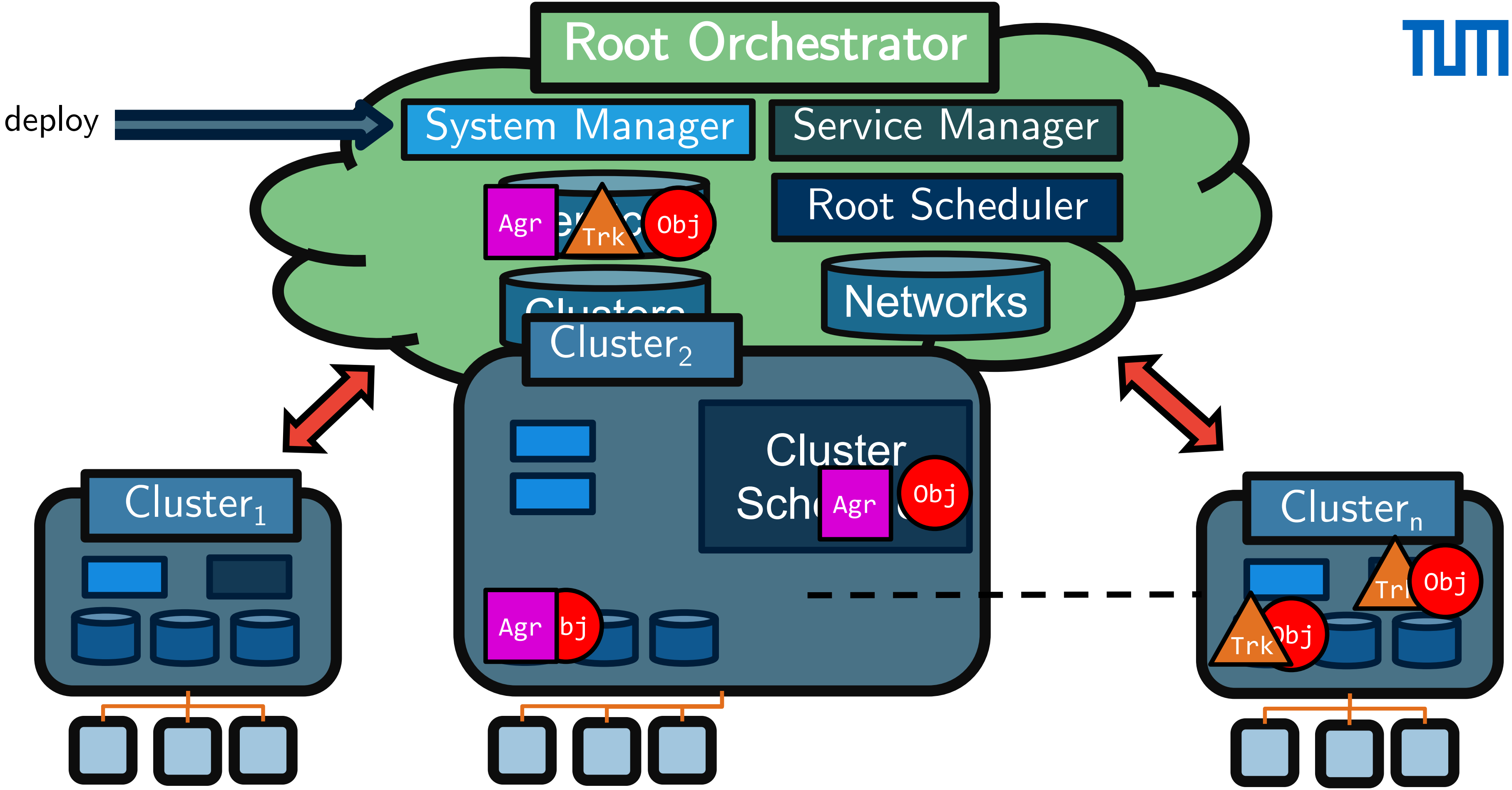

Root Orchestrator
deploy
System Manager
Service Manager
Agr
Trk
Obj
Root Scheduler
Clusters
Networks
Cluster$_1$
Cluster$_2$
Agr
Obj
Cluster$_n$
Trk
Obj

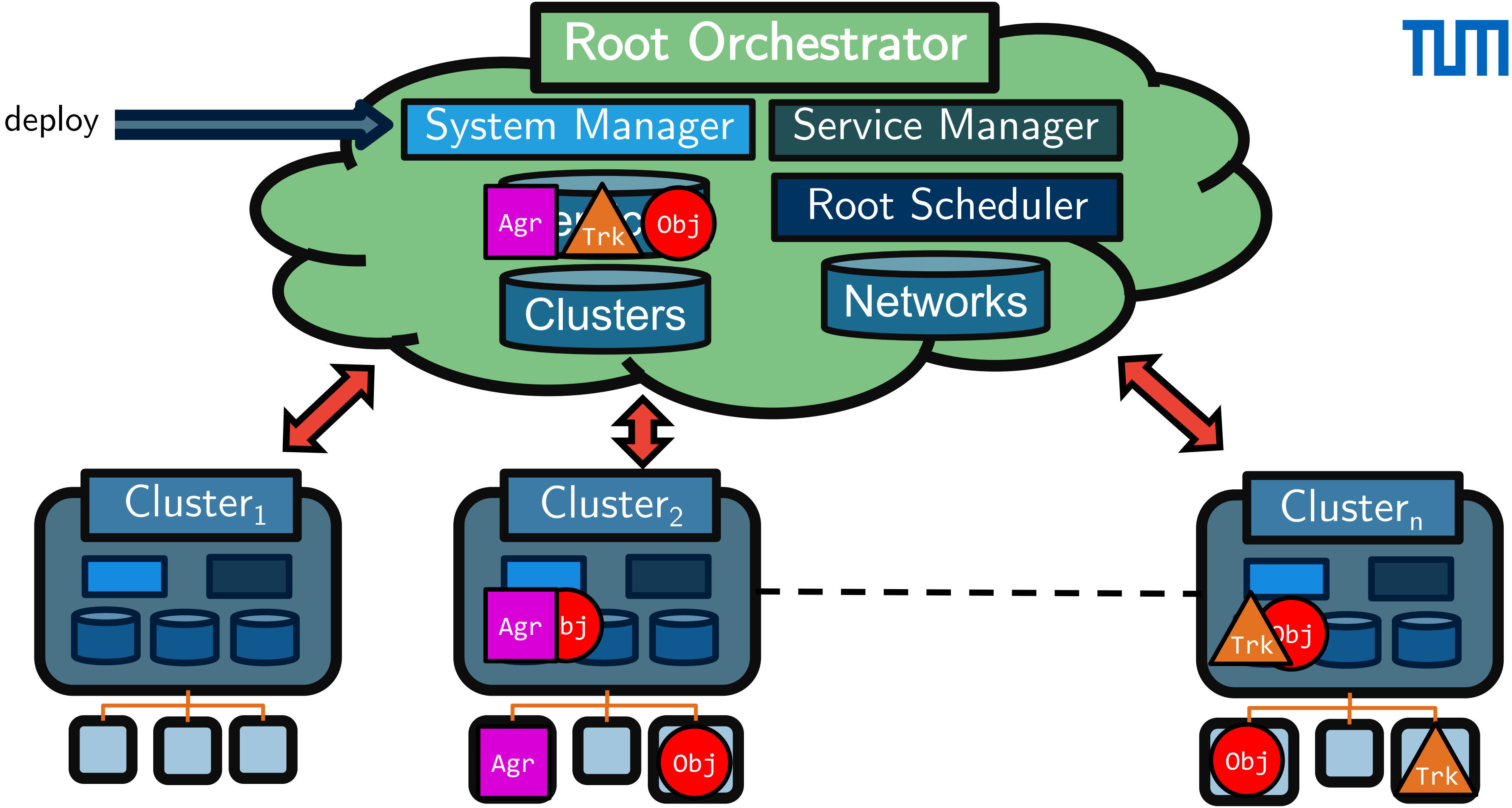

Root Orchestrator
deploy
System Manager
Service Manager
Root Scheduler
Agr
Trk
Obj
Clusters
Networks
Cluster$_2$
Cluster Scheduler
Agr
Obj
Cluster$_1$
Cluster$_n$
Trk
Obj


Root Orchestrator
deploy
System Manager
Service Manager
Root Scheduler
Agr
Trk
Obj
Clusters
Networks
Cluster$_2$
Cluster Sch
Agr
Obj
Agr
bj
Cluster$_1$
Cluster$_n$
Trk
Obj
Trk
Obj



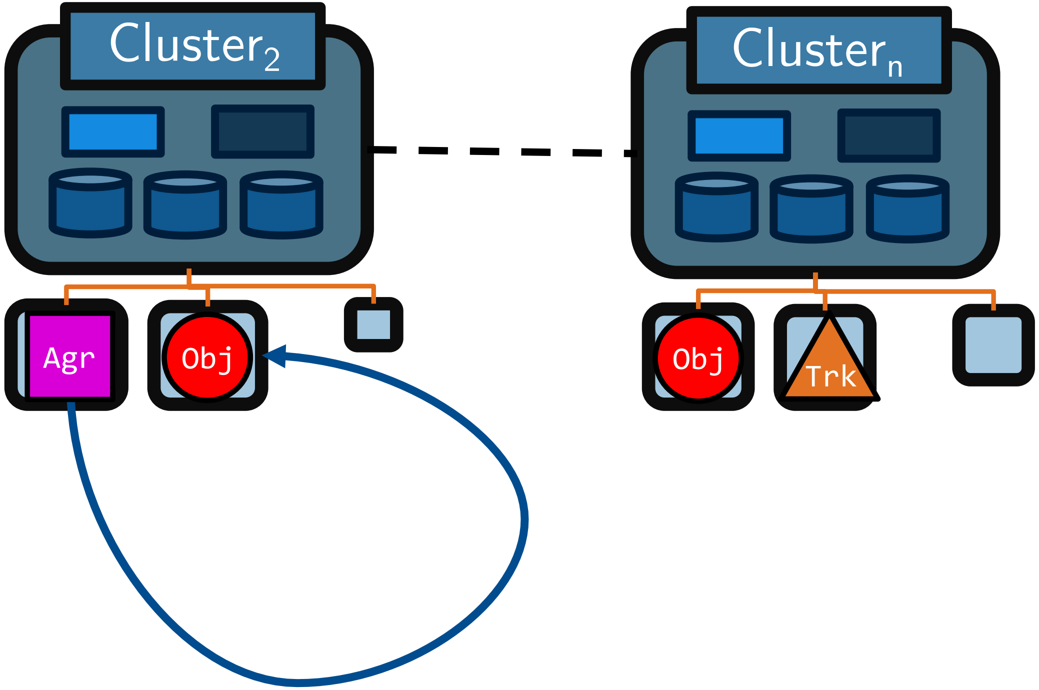
1. Resource-Only Match (ROM)
   - Maximizes hardware utilization
2. Latency & Distance Aware Placement (LDP)
   - Service placement closer to user’s location


Root Orchestrator
deploy
System Manager
Service Manager
Root Scheduler
Agr
Trk
Obj
Clusters
Networks
Cluster$_2$
Cluster Sch
Agr
Obj
Agr
bj
Cluster$_1$
Cluster$_n$
Trk
Obj
Trk
Obj


deploy
Root Orchestrator
System Manager
Service Manager
Agr
Trk
Obj
Root Scheduler
Clusters
Networks
Cluster$_1$
Cluster$_2$
Agr
Obj
Agr
Obj
Cluster$_n$
Trk
Obj
Obj
Trk

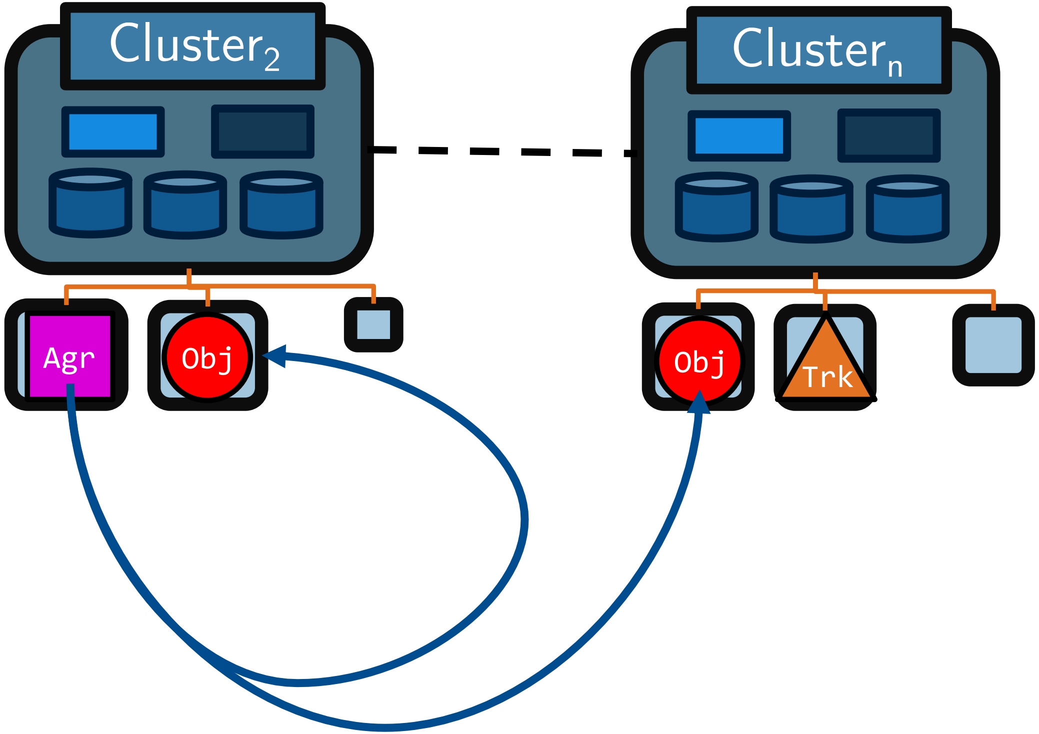
Networking

- Aggregation (Agr) needs to make a request to Object Detection (Obj)



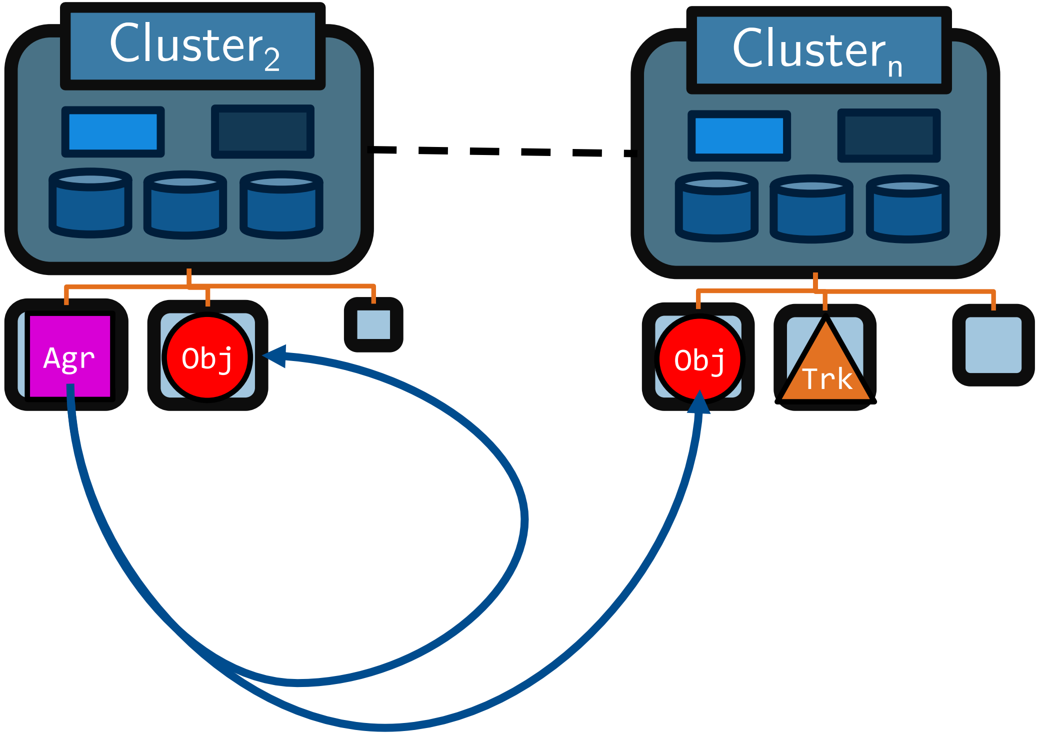
Networking

- Aggregation (Agr) needs to make a request to Object Detection (Obj)
- Obj has two instances in separate clusters



Networking

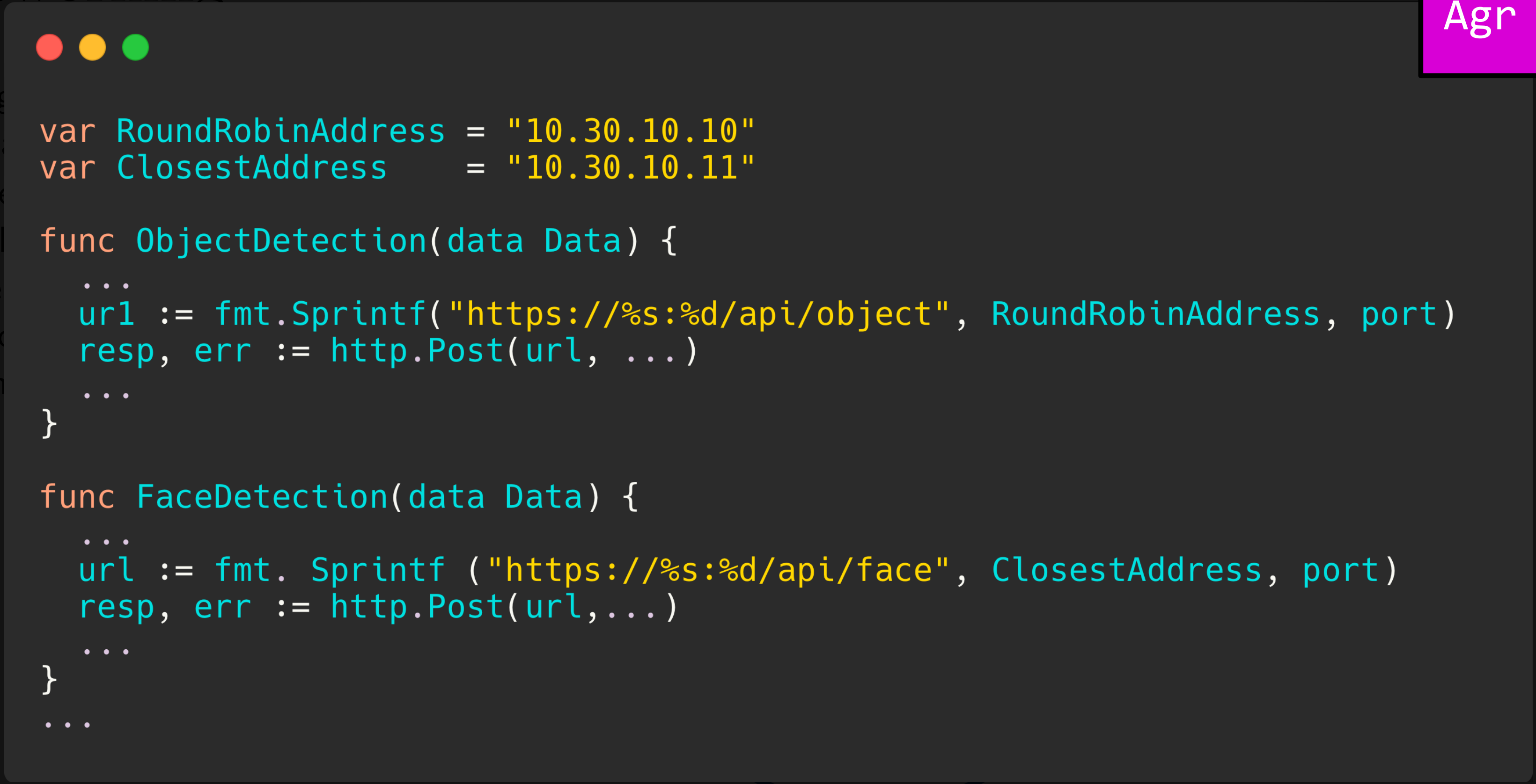
- Aggregation (Agr) needs to make a request to Object Detection (Obj)
- Obj has two instances in separate clusters
- How to exploit service locality and balancing dynamically?



Networking

Agr

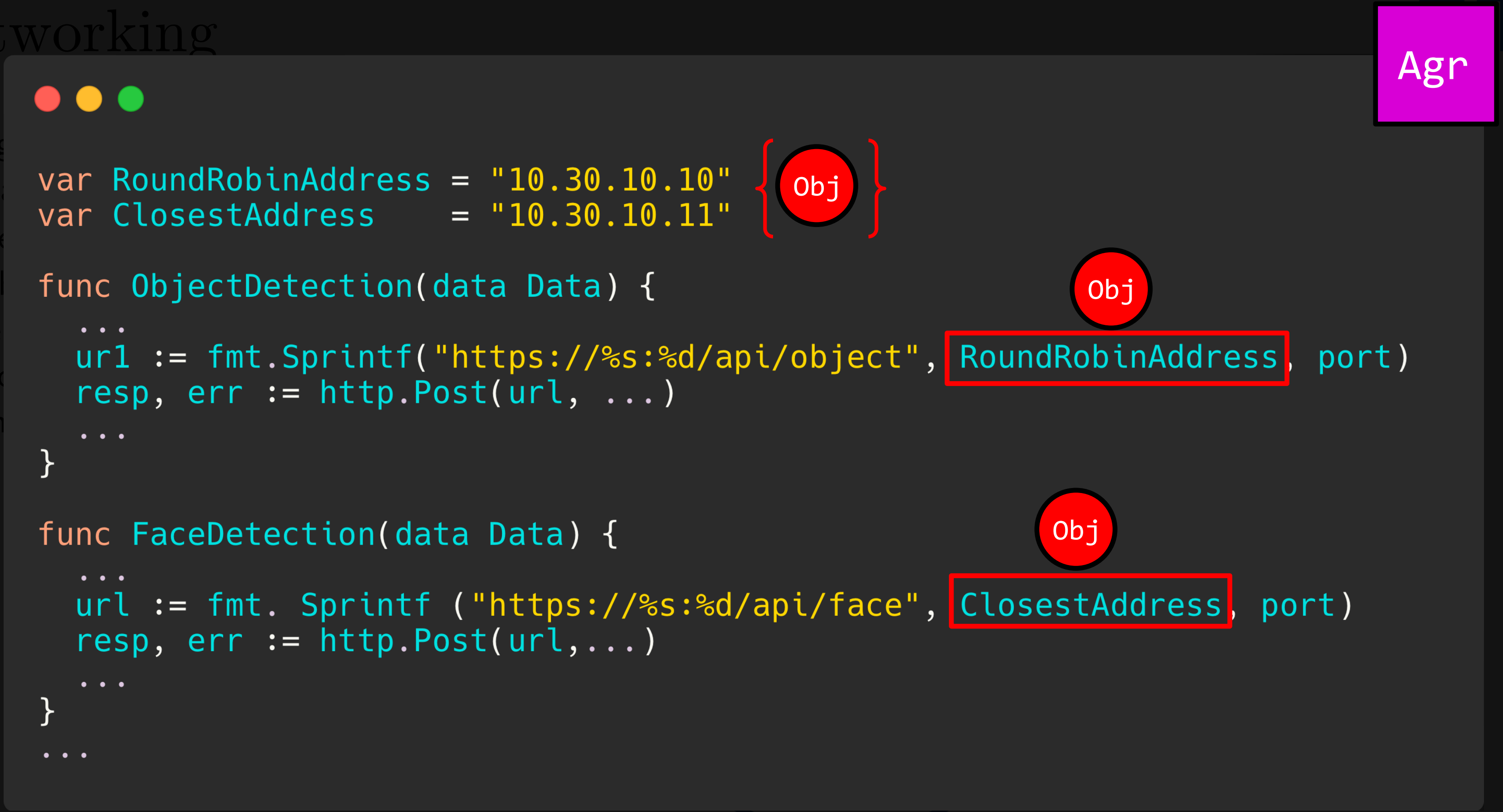
- Ag
  m
  De
- O
  se
- H
  an



```
var RoundRobinAddress = "10.30.10.10"
var ClosestAddress    = "10.30.10.11"

func ObjectDetection(data Data) {
  ...
  url := fmt.Sprintf("https://%s:%d/api/object", RoundRobinAddress, port)
  resp, err := http.Post(url, ...)
  ...
}

func FaceDetection(data Data) {
  ...
  url := fmt. Sprintf ("https://%s:%d/api/face", ClosestAddress, port)
  resp, err := http.Post(url,...)
  ...
}
...
```

Networking

- Ag
  m
  De
- Ol
  se
- Ho
  an

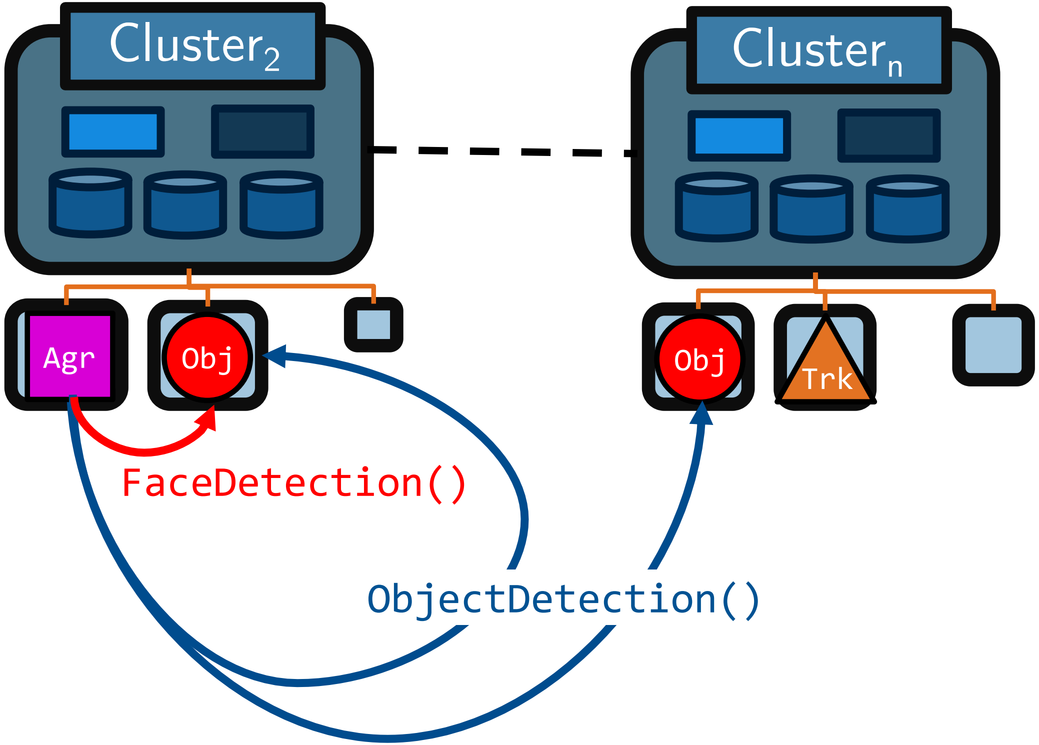


```
var RoundRobinAddress = "10.30.10.10"
var ClosestAddress    = "10.30.10.11"

func ObjectDetection(data Data) {
  ...
  url := fmt.Sprintf("https://%s:%d/api/object", RoundRobinAddress, port)
  resp, err := http.Post(url, ...)
  ...
}

func FaceDetection(data Data) {
  ...
  url := fmt. Sprintf ("https://%s:%d/api/face", ClosestAddress, port)
  resp, err := http.Post(url,...)
  ...
}
...
```

Networking

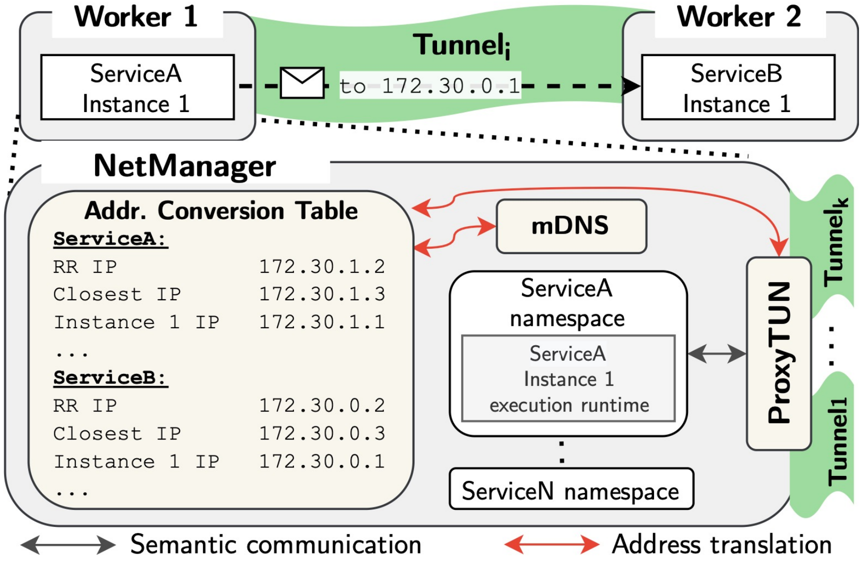
- Semantic Overlay
- Different IP addresses for different balancing policies
- DNS can be configured to resolve to this set of IP addresses

E.g., `http://obj.closest/api`



Networking

- Layer 4 implementation
- Packet tunneling across private subnets
- Async interest propagation
- Networking entirely handled at worker level



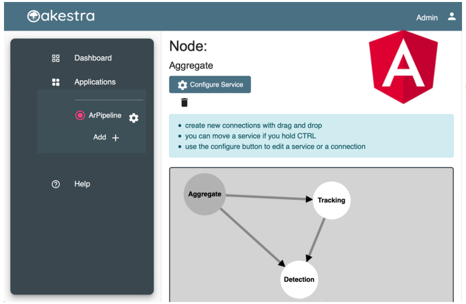
Addr. Conversion Table

ServiceA:	
RR IP	172.30.1.2
Closest IP	172.30.1.3
Instance 1 IP	172.30.1.1
...	
ServiceB:	
RR IP	172.30.0.2
Closest IP	172.30.0.3
Instance 1 IP	172.30.0.1
...	

See paper for details

Implementation

- Lightweight design & Implementation
- 18000 LOC
- Open Source
- Modular and Extensible
- Ready to host future research endeavors



oakestra Admin

Dashboard

Applications

ArPipeline

Add +

Help

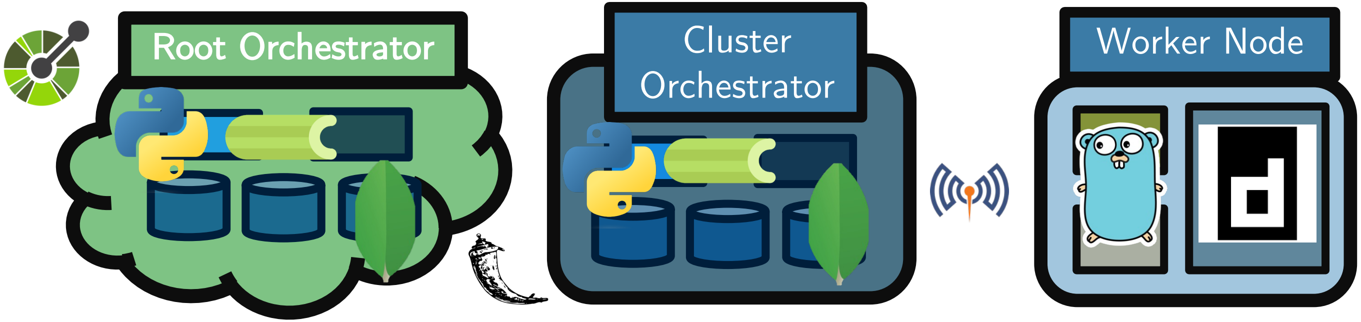
Node:

Aggregate

Configure Service

- create new connections with drag and drop
- you can move a service if you hold CTRL
- use the configure button to edit a service or a connection

Aggregate → Tracking → Detection



Main tech stack:

`python3, celery, mongo, flask, angular, openAPI, mosquitto, golang, containerd`

Evaluation

- Evaluated on:
  - High-Performance Computing (HPC) cluster
  - Heterogeneous Cluster
- Compared frameworks:
  - Kubernetes (K8s)
  - K3s
  - MicroK8s

Oakestra

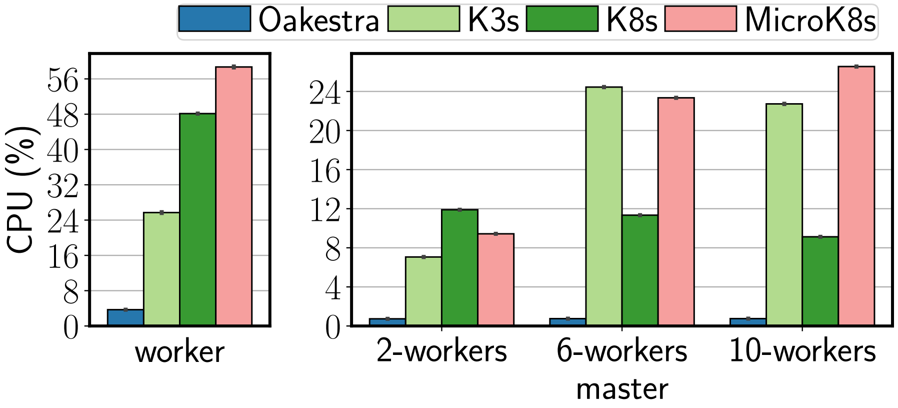
kubernetes

K3S

MicroK8s

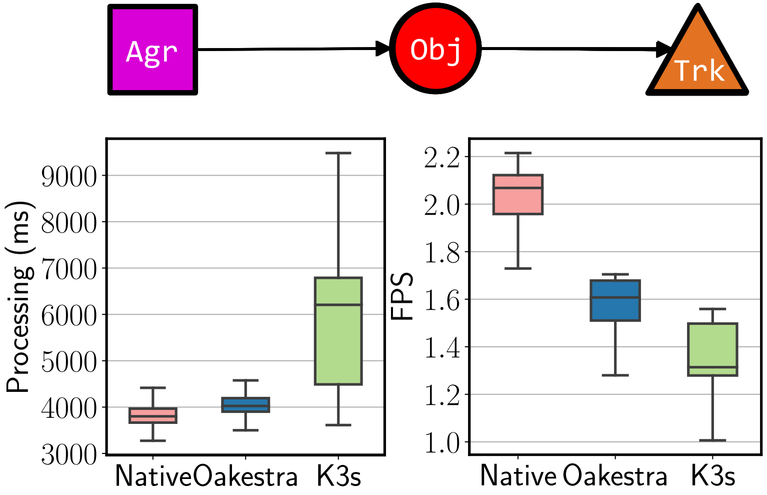
Resource Consumption on Constrained Hardware

- 6x CPU% reduction at worker level compared to K3s
- 10x CPU% reduction at master level compared to K3s



AR Application Performance

- Close to native bare-metal performance
- 10% application perfromance improvement compared to K3s
- Up to 3s faster object detection processing time



Summary:

- Up to 10x lower resource consumption
- 10% Application performance improvement

See paper for more results

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing

Giovanni Bartolomeo* Mehdi Yosofie Simon Bäurle Oliver Haluszczynski Nitinder Mohan Jörg Ott

Chair of Connected Mobility
Technical University of Munich, Germany

* giovanni.bartolomeo@tum.de

Summary:

- Hierarchical orchestration framework
- Delegated service scheduling
- Semantic overlay network
- 10% Application improvement
- 10x Reduction in resource usage
- Available on GitHub

oakestra.io

@oakestra

Oakestra