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
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

Three-tier hierarchical orchestration

Lightweight Implementation

Resource aggregation

Multi-virtualization support

Consolidation of multiple edge providers

Semantic overlay networking

Site-to-Site tunneling

Deployment across geography

Delegated service scheduling

Fine-grained extensible SLA primitives
System Design

Root Orchestrator

Cluster_1

Cluster_2

Cluster_n
Worker Node

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

Node Engine

- Running Services: JobID, ServiceName, Instance, Image, Commands, ResourcesUtilization
- Node: Id, Host, SystemInfo, Virtualization, CpuUsage, MemoryUsage

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

Node engine

Net Manager

Execution Runtime

ARM64
Worker Node

Net Manager

- 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
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
• Different clusters can be administrated by different providers
• Resource aggregation to preserve minute details about internal infrastructure
"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": [...] ...
    }
]...
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing

Giovanni Bartolomeo (TUM)
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
1. Resource-Only Match (ROM)
   - Maximizes hardware utilization

2. Latency & Distance Aware Placement (LDP)
   - Service placement closer to user’s location
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Giovanni Bartolomeo (TUM) | Oakestra: A Lightweight Hierarchical Orchestration Framework for Edge Computing
Networking

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

- Aggregation ($\text{Agr}$) needs to make a request to object detection ($\text{Obj}$)
- Obj has two instances in separate clusters
- How to exploit service locality and balancing dynamically?

```go
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

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

```go
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

 dağıtıcı (Obj) nesnesi bir ağ (Agr) üzerinden bir yüz tespiti (FaceDetection()) yapabilir. Ayrıca, bir kafa izlemesi (Trk) bulunabilir.
Networking

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

See paper for details
Implementation

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

Main tech stack:
python3, celery, mongo, flask, angular, openAPI, mosquitto, golang, containerd
Evaluation

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