Workshop on High-Performance and Reliable Big Data (co-located with SRDS 2021)

Geolocate: A geolocation-aware scheduling system for Edge Computing

João Vilaça, João Paulo, Ricardo Vilaça

HASLab - High-Assurance Software Lab, INESC TEC & U. Minho, Portugal

September 20, 2021

Motivation

- Computational and Storage Capabilities of IoT devices
  - Recent proliferation of **cheaper** but **more advanced edge computing solutions**
- 5G Networking
  - Seamlessly **connect a massive number of IoT devices** with extremely **lean and low-cost connectivity**
- Virtualization and Orchestration Technologies
  - Easy **management and orchestration of distributed services** running at **heterogeneous nodes**

Problem

- Globally distributed Edge nodes
  - Global data exchanges are significantly affected by **high latencies**
- Data protection and privacy laws/regulations
  - Limits to the transfer and processing of **personal data outside of geographic/economic regions**
- Business Objectives and Service Demand
  - Schedule workloads accordingly to the **origin and type of the data**

Related Work

- Zenith [1] and Dyme [2]
  - resource allocation and scheduling systems
  - no concept of geographic location

- Hydra [3]
  - complete self-contained container-orchestration system
  - concept of geographic location

[1] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, “Zenith: Utility-aware resource allocation for edge computing,” in 2017 IEEE International Conference on Edge Computing (EDGE), 2017, pp. 47–54.
[2] A. Samanta and J. Tang, “Dyme: Dynamic microservice scheduling in edge computing enabled iot,” IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6164–6174, 2020.
[3] L. L. Jimenez and O. Schelen, “Hydra: Decentralized location-aware orchestration of containerized applications,” IEEE Transactions on Cloud Computing, pp. 1–1, 2020.

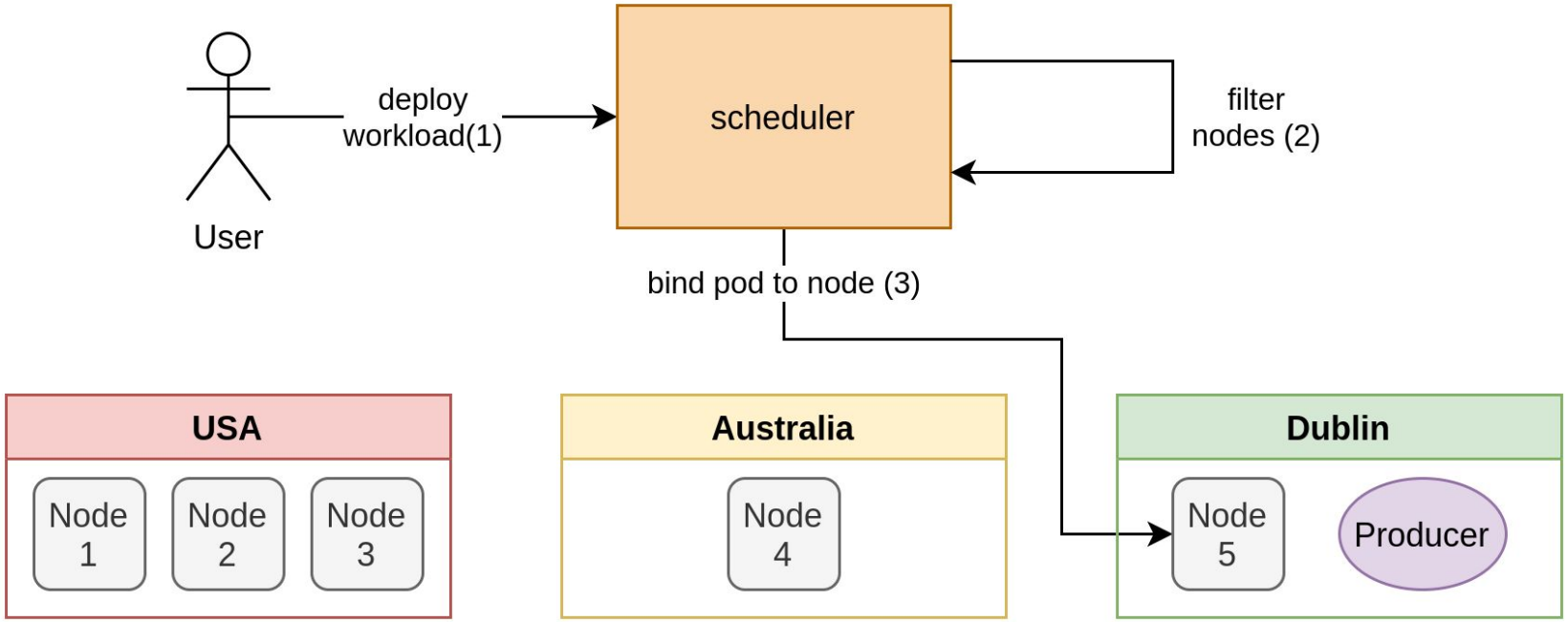
KubeEdge

- Container-orchestration system for Edge
- Kubernetes extension for the Edge
- No edge related alternatives to default scheduler

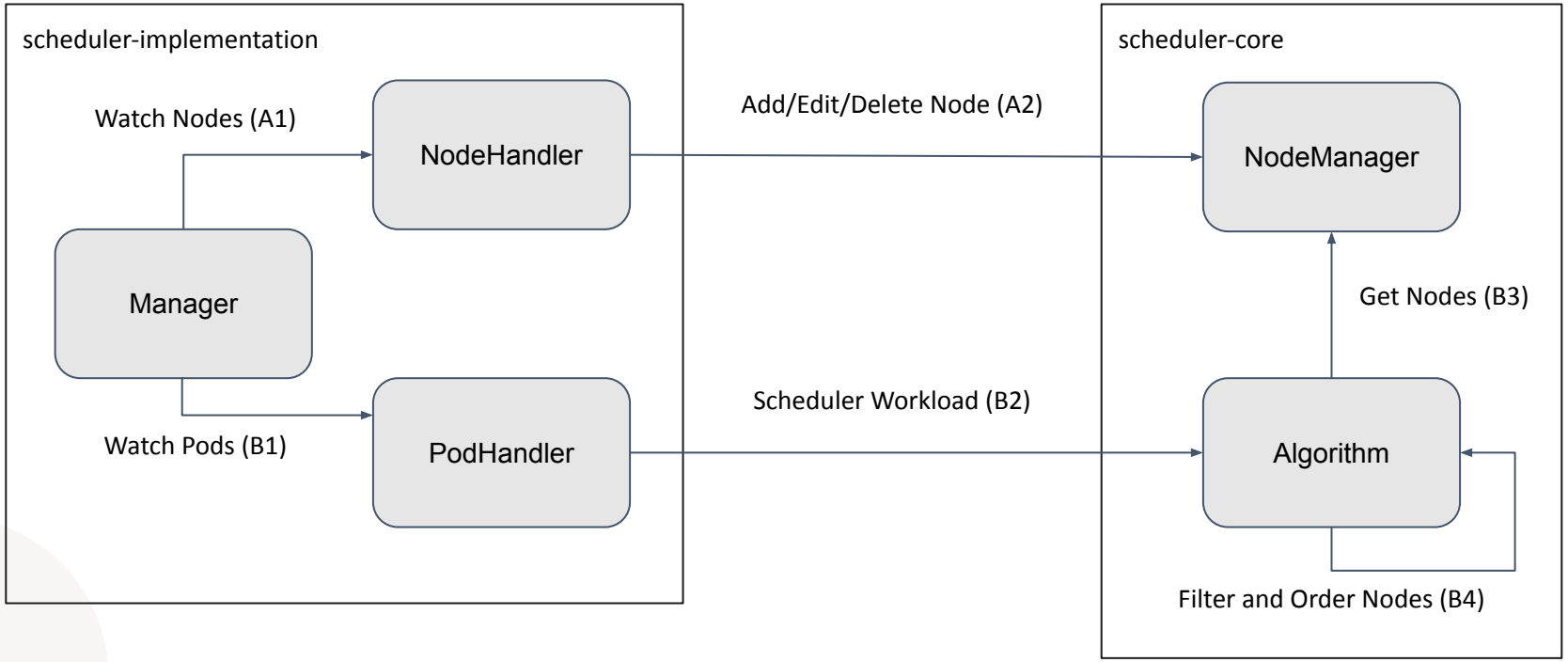
Contributions

- Scheduling framework for Cloud/Edge Computing
- Fully functional prototype, integrating the scheduling system with KubeEdge
- Placement algorithm based on nodes' geographic location and resource availability
- A preliminary experimental evaluation of our prototype in a real deployment and comparing different scheduling approaches

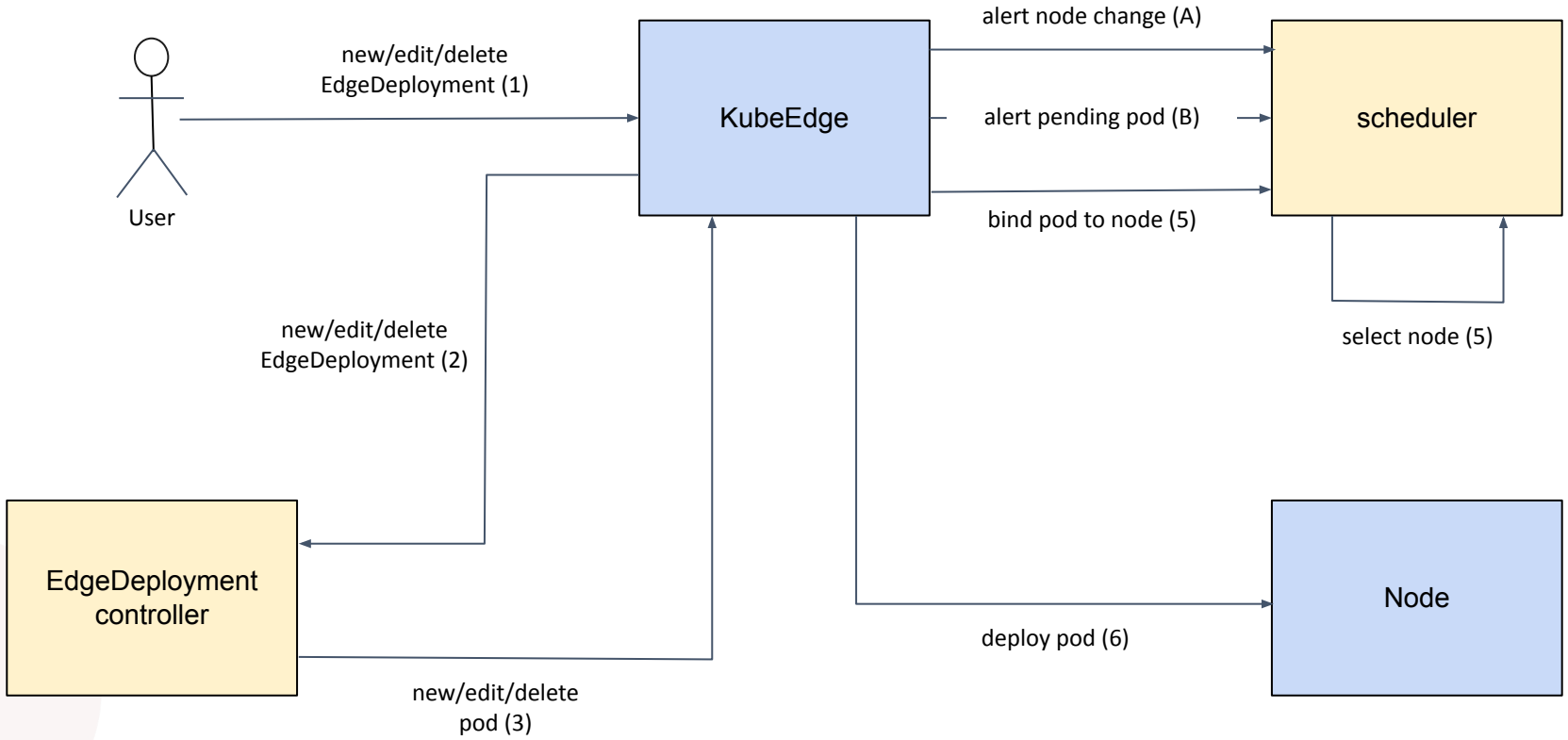
Overview


User
deploy
workload(1)
scheduler
filter
nodes (2)
bind pod to node (3)
USA
Node 1
Node 2
Node 3
Australia
Node 4
Dublin
Node 5
Producer

Geolocate


scheduler-implementation
Watch Nodes (A1)
NodeHandler
Manager
Watch Pods (B1)
PodHandler
Add/Edit/Delete Node (A2)
Scheduler Workload (B2)
scheduler-core
NodeManager
Get Nodes (B3)
Algorithm
Filter and Order Nodes (B4)

Prototype


User
new/edit/delete
EdgeDeployment (1)
KubeEdge
alert node change (A)
alert pending pod (B)
bind pod to node (5)
scheduler
select node (5)
new/edit/delete
EdgeDeployment (2)
EdgeDeployment
controller
new/edit/delete
pod (3)
deploy pod (6)
Node

Evaluation

- Verify the execution performance of the different scheduling algorithms
- Gains in terms of latency between services by scheduling method
- “Data Analytics with Apache Beam”
  - analytics service to get data from the MQTT broker in-stream format
  - apply rules on incoming data in real-time

Cluster setup

- KubeEdge cluster with 6 nodes
  - 2 Cloud Nodes (1 master - 1 worker)
  - 4 Edge Nodes

- Latency across regions simulation with Chaos Mesh (ms)

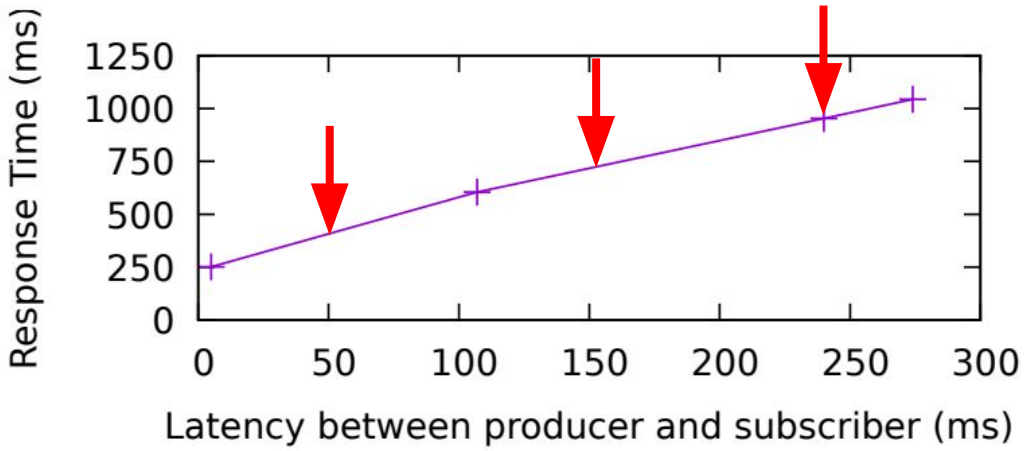
City	Austin	Dublin	Sydney	Tokyo
Austin	-	-	-	-
Dublin	107	-	-	-
Sydney	191	274	-	-
Tokyo	137	240	160	-

Scheduler overhead

- KubeEdge scheduling algorithm
  - μ = 312 μs
  - σ = 104 μs

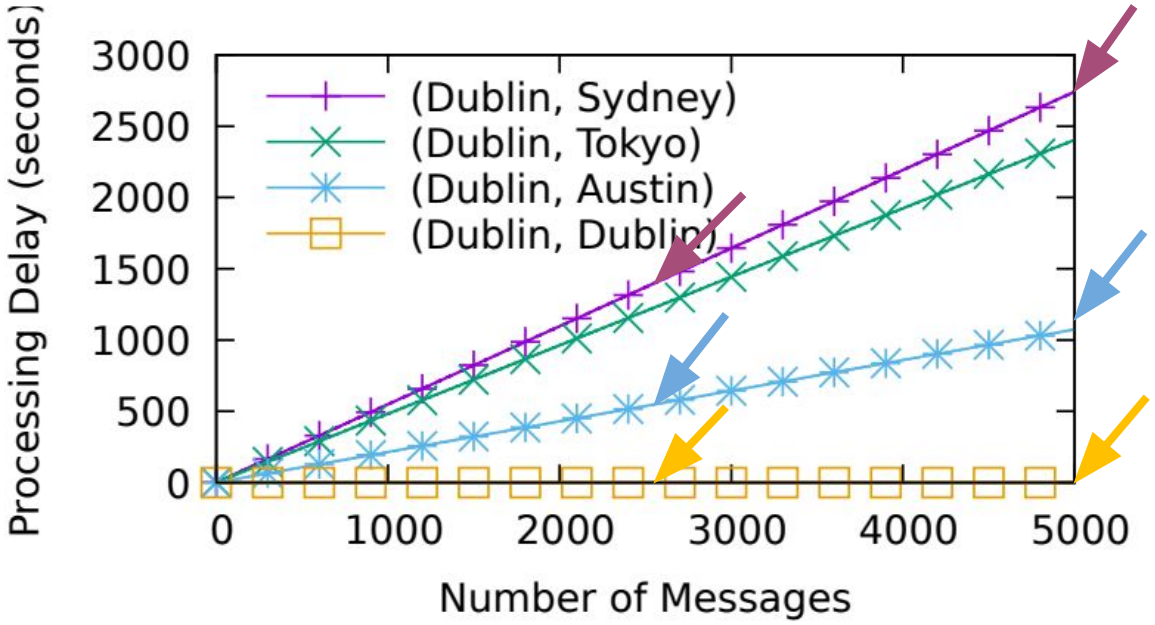
- Geolocate scheduling algorithm
  - μ = 286 μs
  - σ = 174 μs

Location-aware scheduling gains



- 50 ms latency => 400 ms response time
- 250 ms latency => 1s response time
- On average: 157 ms latency => 700 ms response time
- Location-aware: 62% gain in service response times

Geolocate with continuous producer



- (Dublin, Austin)
  - M2500 => 8 min delay
  - M5000 => 16 min delay
- (Dublin, Dublin)
  - M2500 => 253 ms delay
  - M5000 => 253 ms delay
- (Dublin, Sydney)
  - M2500 => 18 min delay
  - M5000 => 45 min delay

Conclusions

- Existing solutions don’t handle geographic location well or have limited performance

- New scheduling solution for globally distributed data-centric workloads
  - Easy integration with state of the art orchestration systems
  - Modularity and algorithm extensibility
  - Good algorithm execution performance

Future Work

- Additional evaluation experiments
- Implementation with other orchestration solutions
- Automatic and adaptable scheduling
- Scalability and fault-tolerance

Workshop on High-Performance and Reliable Big Data (co-located with SRDS 2021)

Geolocate: A geolocation-aware scheduling system for Edge Computing

João Vilaça, João Paulo, Ricardo Vilaça

HASLab - High-Assurance Software Lab, INESC TEC & U. Minho, Portugal

September 20, 2021